Cisco Systems

Cisco Press

CCIE

TCP/IP 路由技术（第2卷）

CCIE™ Professional Development Routing TCP/IP Volume II

A detailed examination of exterior routing protocols and advanced IP routing issues

〔美〕Jeff Doyle　Jennifer DeHaven Carroll　著
毕立波　魏亮　刘述　译

人民邮电出版社
POSTS & TELECOMMUNICATIONS PRESS

TCP/IP 路由技术

(第 2 卷)

[美] Jeff Doyle　Jennifer DeHaven Carroll　著

毕立波　魏　亮　刘　述　译

人 民 邮 电 出 版 社

图书在版编目(CIP)数据

TCP/IP 路由技术.第 2 卷 /（美）多伊尔（Doyle，J.）著；毕立波，魏亮，刘述译.
—北京：人民邮电出版社，2002.8
ISBN 7-115-10096-9

I. T… II. ①多…②毕…③魏…④刘… III. 计算机网络—通信协议—路由选择 IV. TN915.04
中国版本图书馆 CIP 数据核字(2002)第 029269 号

内 容 提 要

本书深入系统地阐述了 TCP/IP 路由技术，内容包括几种重要的网络协议，如外部网关协议(EGP)、边界网关协议(BGP4)，以及相应的高级 IP 路由技术与应用——网络地址翻译、IP 多播路由、IPv6 技术、路由器管理等。

本书内容全面，可读性强，含有协议配置、网络实施、故障排除等方面的大量实例，非常适合 CCIE 开发人员、网络与通信系统工程技术人员阅读。

TCP/IP 路由技术（第 2 卷）

◆ 著 [美] Jeff Doyle Jennifer DeHaven Carroll
◆ 译 毕立波 魏 亮 刘 述
责任编辑 陈万寿

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
读者热线 010-67180876
北京汉魂图文设计有限公司制作
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：787×1092 1/16
印张：43.5
字数：1061 千字 2002 年 8 月第 1 版
印数：1-4 000 册 2002 年 8 月北京第 1 次印刷
ISBN 7-115-10096-9/TN・1841

定价：72.00 元

版 权 声 明

惯用的命令语法

这本书中所用的命令语法与 IOS 命令参考中所用的相同。命令参考中描述的惯例如下：

- 竖线(|)分开可选的、互相排斥的成分。
- 中括号指示可选的成分。
- 大括号指示必选项。
- 中括号中的大括号指示可选项中的必选项。
- 黑体字表示合法输入的命令和关键字。在实际的配置和输出(不是通用的命令语法)中，黑体字表示由用户手工输入的命令(例如 **show** 命令)。
- 斜体字表示需要提供实际值的参量。

前　言

自从《TCP/IP 路由技术(第 1 卷)》发行以来，在 Cisco 出版社的 CCIE 开发系列中又出版了许多新书。CCIE 编程本身也扩展到包括专业方面的许多不同的领域。但是 IP 路由协议还保留着基本功能，CCIE 的报考者必须在这些基本知识的基础上掌握自己的专业技能。这就如同地基不牢，房子早晚会坍塌是同一个道理。

在第 1 卷的简介中我曾声明："……随着互联网在规模以及复杂性方面的增长，路由问题会立即变得巨大而且错综复杂。" 由于我们从内部网关协议转移到既检查自治系统之间的路由又要解决更多特殊的路由问题，例如多播和 IPv6，因此，扩展性以及对这种增长的管理将还是本书论述的中心问题。

通过 CCIE 实验室考试后，你的名字后面会增加一个经过考评的，有价值的 CCIE 号，但是这本书的主要目的是不仅要帮助你从这种 CCIE 实验室考试中走出来，而且要增加你的知识和技能从而使你能够无愧于 CCIE 资格。正如我在第 1 卷中所讲的，我的目的是使你成为一个真正的 CCIE，而不是简单地通过 CCIE 考试。在这本书中，你会发现更多的信息而不只是那些通过实验室考试所需要的知识，但是所有的材料对于一个被认知的互联网专家来讲都是非常重要的。

当我学习 CCIE 的时候，实验室大部分都是由 AGS+路由器组成的。当然，实验室和考试的本质相对于以前来讲都有了重大的改变。如果还有什么不同的话，就是现在实验室考试更难了。CCIE 过程增加的另外一个部分就是重新认证要求。甚至在我参加重新认证之前，就有人告诉我，第 1 卷为他们准备考试提供了很大的帮助——尤其是在准备 IS-IS 方面，因为除了业务供应商之外很少有人使用该协议。因此，我在写第 2 卷的时候，不仅考虑了 CCIE 的投考者，还考虑到已经是 CCIE 的人在进行重新认证时复习的需要。关于多播和 IPv6 的章节就直接起到这种指导作用。

我努力使本书与第 1 卷的结构相同。在第 1 卷中，通过普通的术语介绍一种协议，然后讲述通过 Cisco IOS 软件来进行协议配置的例子，最后讲述的是故障排除用到的 Cisco IOS 软件工具。在讲解 BGP 和 IP 多播的时候，这种结构对于一个单一的章节来讲太长了，因此我把它们分布到了多个章节之中。

我希望你们在读这本书的时候，能像我写这本书时所获得的知识一样多。

作者简介

Jeff Doyle，CCIE #1919，是位于美国丹佛的 Juniper Networks 公司的专业业务顾问。他精通 IP 路由协议以及 MPLS 业务量工程。在北美洲、欧洲和亚洲，Jeff 帮助设计并实现了许多大型的 Internet 业务供应商的网络。Jeff 还针对更高级的网络技术在业务供应商论坛例如 NANOG 和 APRICOT 做了讲座。在加入 Juniper Networks 之前，Jeff 是国际网络业务方面的资深网络顾问。Jeff 的联系方式是 jeff@juniper.net。

Jennifer DeHaven Carroll，是朗讯科技的主要顾问，也是 Cisco 认证的互联网方面的专家(CCIE #1402)。在过去的 13 年中，她规划、设计并组建了许多大型的网络，同时还开发和讲授全 IP 路由协议的理论以及 Cisco 的应用。Jenny 的联系方式是 jennifer.carroll@ieee.org。

审稿人简介

Henry Benjamin，CCIE #4695，CCNA，CCDA，工学学士，是 Cisco 认证的互联网专家并且是 Cisco 公司的 IT 网络设计工程师。他有 8 年多在 Cisco 的工作经验，包括规划、设计、组建大型的运行 IGRP、EIGRP 和 OSPF 协议的 IP 网络。现在 Henry 在澳大利亚悉尼的 Cisco 内部的 IP 设计组工作。Henry 获得了悉尼大学的 L学学士学位。

Peter J.Moyer，CCIE#3286，是 Juniper Networks 的专业业务顾问，他曾经在该公司设计并组建了大型的 ISP 网络。除了他的顾问工作，Peter 还为 Juniper 的客户以及合作者建立了高级 IP 培训课程并创办了 IP 网络设计研讨会。在网络讨论会上他负责讲述 MPLS 等先进的技术。在加入 Juniper 之前，Peter 是国际网络服务公司(INS)的资深网络顾问，他在这里设计并组建了大型的企业网。Peter 获得了马里兰大学计算机与信息科学的工学学士学。

声　明

Jeff Doyle：一本技术书籍的作者就像是一个由一群有才气以及有献身精神的人组成的小部队的前锋，这本书也不例外。听起来就好像我在进行接受学术奖的演讲，但是我还是要感谢许多人。

首先并且最重要的，我要感谢 Jenny Carroll，作为一名技术编辑，她在第 1 卷中的努力令人惊异。作为技术编辑，Jenny 不仅继续在本书中贡献她的专业知识，而且当我感觉到无法预期完成该书的时候，她作为一名合著者加入进来，在我的请求下，她写了最后两章。如果没有她无价的建议和对细节的注意，这本书将无法成功。

同样我要感谢 Pete Moyer，他是我的朋友和助手，他作为一名技术编辑加入到第 2 卷的工作当中。Pete 对该项目之外的我的生活有着重要的影响，我将永远感激他。

我还要感谢 Laurie McGuire 以及 Chris Cleveland，感谢他们作为开发编辑而给予我的专业指导。他们使该书更加完美同时也使我成了一名更好的作者。

感谢 Brett Bartow 以及 Cisco Press 的所有的朋友们，他们在我努力完成本书的过程中给了我极大的关心，从而使我能够在最终期限之前完成本书。我敢肯定，他们对我的第一本书不是很满意，但是在整个项目进行的过程中，他们还是非常善意地对待我。

最后，我要感谢你们，亲爱的读者，是你们让我的第一本书如此成功，并且如此耐心地等待第 2 卷的完成。我希望这本书值得你们的等待。

Jennifer Dehaven Carroll：我要感谢 Jeff Doyle，是他给了我机会，让我为他的书做了一些贡献，这是一个有趣且富有挑战性的工作。

目　　录

第一部分　外部网关协议(EGP)

第二部分 高级 IP 路由问题

第三部分 附 录

第一部分

外部网关协议(EGP)

第 1 章　外部网关协议

本章包括如下重点内容：

- **EGP 的起源**——本节讨论 EGP 发展史，见 RFC827(1982)。
- **EGP 的操作**——本节探讨 EGP 基本的机制，重点在于 EGP 拓扑问题、EGP 功能以及 EGP 信息格式。
- **EGP 的缺点**——本节探讨为什么 EGP 不再是一种可行的外部网关协议解决方案。
- **EGP 的配置**——本节给出了 4 个不同的案例研究，包括 EGP 末梢网关、EGP 核心网关、间接邻居以及缺省路由——用来说明不同类型的 EGP 配置。
- **EGP 的故障排除**——本节考察如何解释 EGP 邻居表，并且对于 EGP 网络较慢的会聚速度给出了一个案例研究，从而表明为什么 EGP 不再是一种流行的选择。

知识渊博的读者会(应该会)问的第一个问题是："为什么放弃其他的一些知识而用整个一章来讲解一个已经过时的协议——外部网关协议(EGP)？"毕竟，EGP 大部分都已经被 BGP 协议给取代了。关于这个问题有两个答案。

首先，虽然 EGP 现在已经很少用了，但偶尔还会遇到。例如，就像本书所写的一样，在少数几个美国军事网络中，还会看到 EGP。作为一个 CCIE，因为可能的少数的几种情况，还是需要理解 EGP。

其次，本章主要是讲述一些历史。考察开发 EGP 的动机以及最初 EGP 的缺点，从而为下面的两章提供一个序言。熟悉 BGP 演进的根源，对你来讲会更有意义。

1.1　EGP 的起源

在(20 世纪)80 年代初期，由路由器(网关)组成的 ARPANET(现代 Internet 的始祖)使用一种距离向量路由协议，它是网关到网关协议(GGP)。每一个网关都要知道到每一个可到达网络的一条路由，在这里，距离以跳(Hop)来计算。随着 ARPARENT 的发展，它的建设者预见的问题与现在许多正在发展的网络管理者遇到的问题相同，那就是：他们所采用路由协议的扩展性不好。

Eric Rosen 在 RFC 827[1] 中，记载了扩展性的问题：

- 由于所有的网关需要知道所有的路由，"路由算法的开销变得相当的庞大"。只要拓扑发生改变，可能的就是增加网络的规模，所有的路由器需要交换路由信息并且重新计算它们的路由表。即使网络处于稳定的状态下，路由表的规模以及路由的更新都成了一种与日俱增的负担。
- 随着 GGP 软件数量的增加以及它们运行平台变得越来越多样化，"将 Internet 再看成一个完整的通信系统是不可能的"。尤其是，维护和故障排除变得"几乎不可能"。
- 随着网关数量的增加，网关管理人员相应地也需要增加。结果是，软件升级的阻力

也变大了，“出现的任何改变必然会牵扯到不同地方的许多不同的人”。

在 RFC827 中提出的解决方案，是将 ARPANET 从一个单一的互联网络转移到一个互相连接的、自主控制的网络系统。在每一个互联网络也就是我们所说的自治系统(AS)内，管理机构可以按照他们自己的选择来管理这个 AS。这样做的效果是，自治系统的概念扩大了互联的范围并为这种分层机构加入了新的一层。曾经是一个单一的的互联网络——一个由网络组成的网络——现在是由自治系统组成的网络，这些自治系统本身都是一个独立的互联网络。由于网络是由 IP 地址来标识的，那么自治系统就由自治系统号来标识。一个 AS 号是一个 16bit 的数，由分配 IP 地址的权力机构进行分配。

> 注：　和 IP 地址一样，一些 AS 号留做专用。这些号码的范围是 64512~65535。具体内容请参见 RFC 1930(www.isi.edu/in-notes/rfc1930.txt)。

每个 AS 的管理机构可以自由选择的主要方面是在它们的网关上运行的路由协议。因为网关在 AS 的内部，它们的路由协议是 IGP。因为 GGP 是 ARPANET 的路由协议，它就是缺省的第一个 IGP。但是有意思的是，对于更现代的(并且更加简单的)路由信息协议(RIP)的兴趣是在 1982 年建立起来的，而且人们希望该协议以及其他还未计划的协议将来会用在众多自治系统中。目前，GGP 已经完全被 RIP、RIP-2、IGRP、EIGRP、OSPF 以及 IS-IS 协议所取代。

每个 AS 通过一个或者多个外部网关与其他的自治系统相连。RFC 827 提出，外部网关之间通过 EGP 协议来共享路由信息。与流行的观念相反，虽然 EGP 是一种距离向量协议，但它并不是路由协议。它没有在网络之间选择较优路由的算法；更确切一点地讲，它是外部网关用来与其他外部网关交换可到达性信息的一种通用语言。可到达性信息是主要网络地址(没有子网)以及到达它们需要经过的网关的一个简单列表。

1.2　EGP 的操作

RFC 827 提出了 EGP 版本 1。RFC 888[2] 提出了 EGP 版本 2，该版本对版本 1 做了轻微的改动。RFC 904[3] 提出了 EGPv2 正式的规范。

1.2.1　EGP 拓扑问题

EGP 信息在 EGP 邻居或者对等实体之间进行交换。如果邻居是在同一个 AS 域中，那么它们是内部邻居。如果它们在不同的 AS 域内，那么它们属于外部邻居。EGP 没有自动发现邻居的功能；邻居的地址是通过人工配置的，并且它们之间交换的信息对于人工配置的地址来讲是单向的。

因为一个 EGP 信息只能到达一个邻居而不应该传得更远，因此 RFC 888 建议把 EGP 信息的生存时间(TTL)设成一个较低的值。但是没有 EGP 功能要求 EGP 邻居共用同一条数据链路。例如，图 1-1 显示，两个 EGP 邻居被一个只支持 RIP 的路由器分开。因为对于邻居来讲，EGP 信息是单播的，它们能够通过路由器的边界。因此，Cisco 路由器将 EGP 的 TTL 值置为 255。

EGP 网关既可以是核心网关也可以是末梢网关。两种网关形式都可以接受其他自治系统

中网络的信息，但是末梢网关只能发送它自己域内的有关网络的信息。只有核心网关才可以既发送自己域内网络信息，也可以发送它学习到的其他自治系统中网络的信息。

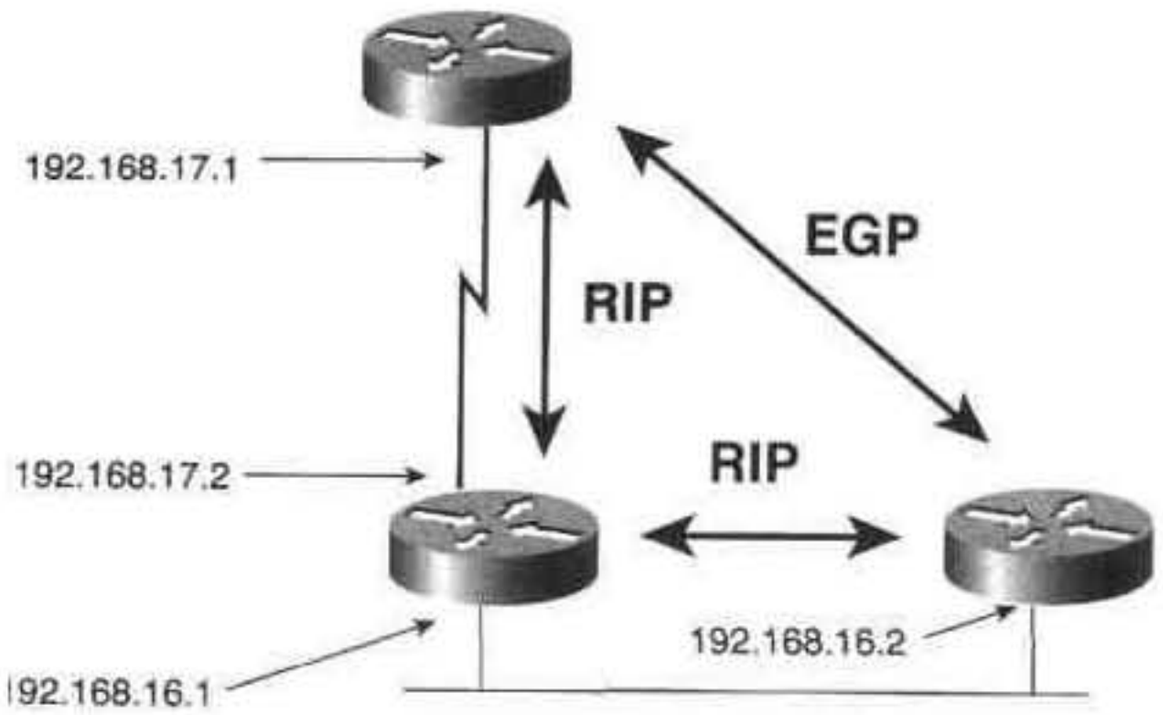


图 1-1　EGP 邻居没有必要与同一个网络相连

要想理解为什么 EGP 要定义核心网关和末梢网关，了解 EGP 结构的局限性是非常必要的。正如前面所讲的，EGP 不是一个路由协议，它只是更新可到达网络的名单，而不包括决定最短路径或者阻止路由环路的足够信息。因此，EGP 拓扑必须没有环路。

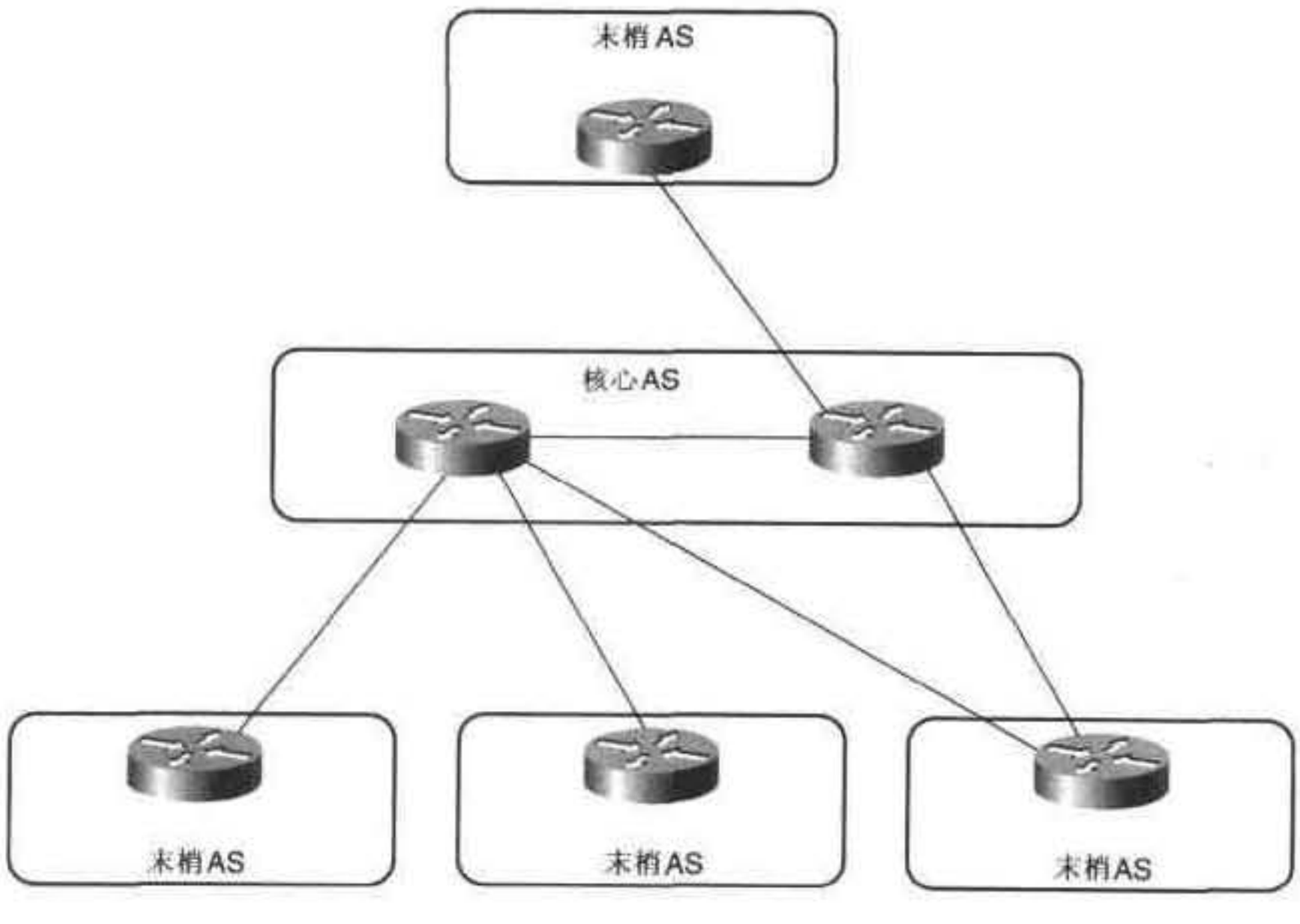


图 1-2　为了防止路由环路的出现而采用的一种 EGP 拓扑

图 1-2 给出了一个 EGP 拓扑，该图主要讲述的是为了阻止路由环路的出现，只有核心网关可以将从一个 AS 学习到的信息发送给另外一个 AS。这里只有一个核心 AS，其他的自治系统(末梢自治系统)都必须与它相连。这种两层的树型拓扑与 OSPF 的两层拓扑要求十分相似，并且它们的目的相同。在《TCP/IP 路由技术(第 1 卷)》域间 OSPF 路由就是基本的距离

向量而且对路由环路很敏感。要求两个非骨干 OSPF 域内的全部业务量都要通过骨干域，从而通过一个强制性的无环路域间拓扑来减少潜在路由环路出现的可能性。同样地，要求末梢自治系统之间所有 EGP 可到达信息都要通过核心 AS 从而在 EGP 拓扑中减少潜在路由环路的出现。

1.2.2　EGP 的功能

EGP 包含以下 3 种机制：

- 邻居获得协议
- 邻居可到达协议
- 网络可到达协议

这 3 种机制采用 10 种消息类型来建立邻居关系、维护邻居关系、与邻居交换网络可到达信息，并且通知邻居关于程序上或者格式上的差错。表 1-1 列出了所有 EGP 消息的类型以及使用每一种消息类型的机制。

表 1-1　　EGP 消息类型

消 息 类 型	机　　制
Neighbor Acquisition Request	邻居获得协议
Neighbor Acquisition Confirm	邻居获得协议
Neighbor Acquisition Refuse	邻居获得协议
Neighbor Cease	邻居获得协议
Neighbor Cease Acknowledgement	邻居获得协议
Hello	邻居可到达协议
I-Heard-You	邻居可到达协议
Poll	邻居可到达协议
Update	邻居可到达协议
Error	所有的功能

下面将会讨论每一种 EGP 机制的细节问题。本章中“EGP 消息格式”一节会涉及到信息的特殊细节。

1．邻居获得协议

在 EGP 邻居之间交换可到达信息之前，必须确认它们是一致的。这个过程由一个简单的双向握手来完成，其中一个邻居发送一个 Neighbor Acquisition Request 消息，它的邻居应答一个 Neighbor Acquisition Confirm 消息。

没有 RFC 规范明确两个 EGP 邻居最初是如何发现对方的。实际上，EGP 网关通过手工配置的邻居 IP 地址来了解到它的邻居。网关于是单播一个 Acquisition Request 消息给配置的邻居。该消息规定了一个 *Hello 间隔*，也就是网关可接受的从邻居发来的两个 Hello 消息之间

的间隔，该消息还规定了一个 *Poll 间隔*，它是出于路由更新的原因，网关可接受邻居轮询的最小间隔。邻居应答的 Acquisition Confirm 消息会包含关于这两个间隔它自己的值。如果邻居在该值上达成一致，它们就准备交换网络可到达信息了。

当一个网关第一次了解到一个邻居，它会认为该邻居处于 *空闲(Idle)*状态。在发送 Acquisition Request 消息之前，网关把邻居的状态改为 *Acquire* 状态；当网关收到了 Acquisition Confirm 消息时，它把邻居改为 *Down* 状态。

注：　有关 EGP 有限状态机的完整解释，请参见 RFC 904。

一个网关可以通过应答一个 Neighbor Acquisition Refuse 消息而不是 Acquisition Confirm 消息来拒绝接受一个邻居。Refuse 消息可以包括拒绝的理由，例如缺少空间；它也可以不标明任何理由而拒绝一个邻居。

一个网关同样可以通过发送一个 Neighbor Cease 消息来中断一个已经建立起来的邻居关系。对于这个 Refuse 消息，发起网关可以选择包括一个中断的理由或者不标明理由。邻居收到这个 Neighbor Cease 消息以后，应答一个 Neighbor Cease Acknowledgement 消息。

Neighbor Acquisition 过程的最后一种情况，是网关发送一个 Acquisition Request 消息但是邻居没有应答。RFC888 建议以“一个合理的速率，大概每 30s 左右”来重传 Acquisition 消息。Cisco 的 EGP 的执行并不是在一个固定的时间段内重发未答复的消息。确切一点地讲，它是在初始传输 Acquisition 消息以后 30s 后再重传未答复的 Acquisition 消息。在下一次传输以后，它再等待 60s，如果在第三次传输后 30s 内该网关没有收到应答，网关把邻居的状态从 Aquire 状态转到 Idle 状态(见例 1-1)。网关保持 300s(5 分钟)Idle 状态，然后再把状态转到 Acquire 状态并开始全部的过程。

例 1-1　debug ip egp transactions 命令的输出显示了 EGP 状态的转变

```
Shemp#debug ip egp transactions
EGP debugging is on
Shemp#
EGP: 192.168.16.2 going from IDLE to ACQUIRE
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: 192.168.16.2 going from ACQUIRE to IDLE
EGP: 192.168.16.2 going from IDLE to ACQUIRE
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=0
     Type=ACQUIRE, Code=REQUEST, Status=0 (UNSPECIFIED), Hello=60, Poll=180
EGP: 192.168.16.2 going from ACQUIRE to IDLE
```

注意在例 1-1 中每一个 EGP 消息都有一个顺序号。顺序号可用来标明 EGP 消息对(例如 Neighbor Acquisition Request/Confirm、Request/Refusal 以及 Cease/Cease-Ack 对)。在下一节“网络可到达协议”中将会详细讲述顺序号的使用方法。

当两个 EGP 网关成为邻居以后，一个是*主动*邻居，另一个是*被动*邻居。主动网关通过发送 Neighbor Acquisition Requests 来发起邻居关系。被动网关不发送 Acquisition Requests，它只是应答这些消息。对于将在接下来的一节中讲述的 Hello/I-Heard-You 消息对也是一样的：主动邻居发送 Hello 消息，被动邻居用 I-Heard-You(I-H-U)消息来应答该消息。一个被动网关可以发起一个 Neighbor Cease 消息，但是主动网关必须应答一个 Cease Acknowledgement 消息。

核心网关，可能是几个其他自治系统中路由器的邻居，也可能是一个相连邻居的主动网关或者是一个相连邻居的被动网关。Cisco 的 EGP 执行用 AS 号作为决定因素：AS 号较低的邻居是主动邻居。

2. 邻居可到达协议

一个网关获得一个邻居以后，它通过周期性地发送 Hello 消息来维持邻居关系。邻居给 Hello 消息应答一个 I-H-U 消息。RFC 904 没有规定 Hello 消息之间的标准间隔；Cisco 采用的缺省值为 60s，可以通过 **timers egp** 命令改变这个值。

当交换了 3 个 Hello/I-H-U 消息对以后，邻居状态从 Down 转到 Up(参见例 1-2，例中显示了双向握手的成功以及 EGP 状态的转变)。然后邻居之间就可以交换可到达信息，这部分将在下一节进行描述。

如果一个主动邻居连续发送了 3 个消息但是却没有收到应答，该邻居的状态将转到 Down。网关以正常的 Hello 间隔发送超过 3 个 Hello 消息，如果没有应答，邻居状态将会转到 Cease。网关以 60s 的间隔发送 3 个 Neighbor Cease 消息，如果邻居以 Cease Acknowledgement 消息应答任何一个 Neighbor Cease 消息，或者根本没有应答，网关会把邻居状态转成 Idle，等待 5 分钟以后，将邻居状态转成 Acquire，并且试图去重新申请该邻居。例 1-3 说明了这个时间的顺序，例中未答复的 EGP 消息之间的间隔为 60s。

例 1-2　debug ip egp transactions 命令的输出

```
EGP: 192.168.16.2 going from IDLE to ACQUIRE
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=ACQUIRE, Code=REQUEST, Status=1 (ACTIVE-MODE), Hello=60, Poll=180
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=ACQUIRE, Code=CONFIRM, Status=2 (PASSIVE-MODE), Hello=60, Poll=180
EGP: 192.168.16.2 going from ACQUIRE to DOWN
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=REACH, Code=I-HEARD-YOU, Status=2 (DOWN)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=REACH, Code=I-HEARD-YOU, Status=2 (DOWN)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=REACH, Code=I-HEARD-YOU, Status=2 (DOWN)
EGP: 192.168.16.2 going from DOWN to UP
```

例 1-3　192.168.16.2 处的邻居已经停止了应答

```
Shemp#
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=REACH, Code=I-HEARD-YOU, Status=1 (UP)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=2
     Type=POLL, Code=0, Status=1 (UP), Net=192.168.16.0
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: 192.168.16.2 going from UP to DOWN
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=REACH, Code=HELLO, Status=2 (DOWN)
EGP: 192.168.16.2 going from DOWN to CEASE
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=ACQUIRE, Code=CEASE, Status=5 (HALTING)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=ACQUIRE, Code=CEASE, Status=1 (ACTIVE-MODE)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=ACQUIRE, Code=CEASE, Status=1 (ACTIVE-MODE)
EGP: 192.168.16.2 going from CEASE to IDLE
```

例 1-4　邻居 192.168.16.1 已经停止了应答

```
Moe#
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=1
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=1
     Type=REACH, Code=I-HEARD-YOU, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=1
     Type=POLL, Code=0, Status=1 (UP), Net=192.168.16.0
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=2
     Type=POLL, Code=0, Status=1 (UP), Net=192.168.16.0
EGP: 192.168.16.1 going from UP to DOWN
EGP: 192.168.16.1 going from DOWN to CEASE
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=3
     Type=ACQUIRE, Code=CEASE, Status=5 (HALTING)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=3
     Type=ACQUIRE, Code=CEASE, Status=2 (PASSIVE-MODE)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=3
     Type=ACQUIRE, Code=CEASE, Status=2 (PASSIVE-MODE)
EGP: 192.168.16.1 going from CEASE to IDLE
```

例 1-4 给出了一个停机的邻居的例子，例中从 192.168.16.2 处得到了 debug 消息，该网关处于被动模式。不过这次一个处于被动模式的网关(192.168.16.2)发现一个死亡的邻居(192.168.16.1)。

在 60s 的 Hello 间隔内如果一个网关没有收到 Hello 消息，它会试图“唤醒”该邻居。因为一个处于被动模式的网关不能发送 Hello，它只能发送 Poll 消息。网关等待一个 Poll 间隔

的时间。(Cisco 缺省的 Poll 间隔是 180s 或者说 3 分钟。)如果没有收到应答，它会再发送一个 Poll 消息并且等待另一个 Poll 间隔。如果还没有应答，该网关就会把邻居的状态转成 Down 然后立即转成 Cease。正如例 1-3 所讲的，发送 3 个 Cease 消息以后，邻居的状态被转成 Idle。

3．网络可到达性协议

当邻居的状态是 UP 时，EGP 邻居可以开始交换可到达信息。每一个网关周期性地发送包含一些顺序号的 Poll 消息给它的邻居。邻居以包含相同顺序号和一个可到达网络的列表的更新消息来应答。例 1-5 表示了 Cisco 的 IOS 软件是如何利用这些顺序号的。

例 1-5　为了获得网络可到达更新消息，EGP 邻居周期性地轮询其他的邻居

```
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=120
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=120
     Type=REACH, Code=I-HEARD-YOU, Status=1 (UP)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=120
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=120
     Type=REACH, Code=I-HEARD-YOU, Status=1 (UP)
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=120
     Type=POLL, Code=0, Status=1 (UP), Net=192.168.16.0
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=120
     Type=UPDATE, Code=0, Status=1 (UP), IntGW=2, ExtGW=1, Net=192.168.16.0
     Network 172.17.0.0 via 192.168.16.2 in 0 hops
     Network 192.168.17.0 via 192.168.16.2 in 0 hops
     Network 10.0.0.0 via 192.168.16.2 in 3 hops
     Network 172.20.0.0 via 192.168.16.4 in 0 hops
     Network 192.168.18.0 via 192.168.16.3(e) in 3 hops
     Network 172.16.0.0 via 192.168.16.3(e) in 3 hops
     Network 172.18.0.0 via 192.168.16.3(e) in 3 hops
EGP: 192.168.16.2 updated 7 routes
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=3
     Type=POLL, Code=0, Status=1 (UP), Net=192.168.16.0
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=3
     Type=UPDATE, Code=0, Status=1 (UP), IntGW=1, ExtGW=0, Net=192.168.16.0
     Network 172.19.0.0 via 192.168.16.1 in 0 hops
EGP: from 192.168.16.1 to 192.168.16.2, version=2, asystem=1, sequence=121
     Type=REACH, Code=HELLO, Status=1 (UP)
EGP: from 192.168.16.2 to 192.168.16.1, version=2, asystem=2, sequence=121
     Type=REACH, Code=I-HEARD-YOU, Status=1 (UP)
```

在 Poll 消息被发送之前，每一对在邻居之间交换的 Hello/I-H-U 消息都包含相同的顺序号。Poll/Update 也使用同样的顺序号。接收到 Update 消息之后，主动邻居增加顺序号。在例 1-5 中，在 Poll/Update 中顺序号是 120，然后，被增加到了 121。注意，两个邻居都要发送一个 Poll，在这个例子中，从被动邻居(192.168.16.2)来的 Poll 有一个完全不同的顺序号(3)。邻居通常都是用与 Poll 有相同顺序号的 Update 来应答。

Cisco 的 IOS 软件所用的缺省的轮询间隔是 180s，可以通过 **timers egp** 命令来改变它的值。通常，只有当一个网关被轮询的时候，它才发送更新信息，但是这就意味着，一个拓扑的改变可能 3 分钟以后还没有得到公布。EGP 为这种可能性提供了一种解决办法，那就是，允许网关在每一个轮询间隔发送一个未经请求的更新信息——也就是说，这个更新消息并不

是应答一个轮询消息。但是，Cisco 并不支持未经请求的更新。

> 注：　**timers egp** 命令还可以用来更改 Hello 间隔。该命令的格式是 **timers egp** *hello polltime*。

轮询和更新消息中都包括源网络地址。例如，在例 1-5 中，Poll 和 Update 消息表明源网络地址是 192.168.16.0。所有可到达网络的信息都是在源网络处计算的——也就是说，所有被申请或者被公布的网络都可以通过连接到源网络的一个路由器到达。虽然对于这个网络来讲，两个邻居都与它相连，但是更确切一点地讲，它是这样的一个网络，Poll 申请信息，而 Update 是提供信息。EGP 是一个纯粹的有类别协议，并且源地址——和在更新消息中列出的网络地址一样——通常都是主要类别网络地址，不会是子网地址。

在源网络地址后面是一个或者多个路由器以及通过这些路由器可到达网络的列表。这个列表上路由器的共性是它们都连接在源网络上。如果列表上的某个路由器不是产生更新消息的 EGP 网关，该路由器就是一个间接或者第三方邻居。

图 1-3 说明了间接 EGP 邻居的概念。路由器 Moe 是一个核心网关，它是其他三个网关的对端。

例 1-5 中的 Debug 消息来自 AS 1 中的路由器 Shemp。注意在由 Moe(192.168.16.2)发起的更新消息中列出了三个通过 Moe 可到达的网络，同时，还列出了四个通过 Larry(192.168.16.4)和 Curly(192.168.16.3)可到达的网络。这两个路由器是通过 Moe 可到达的 Shemp 的间接邻居。AS 3 中的 Joe 不是一个间接邻居，因为它没有与源网络相连。它的网络只是被公布成通过 Moe 可到达。

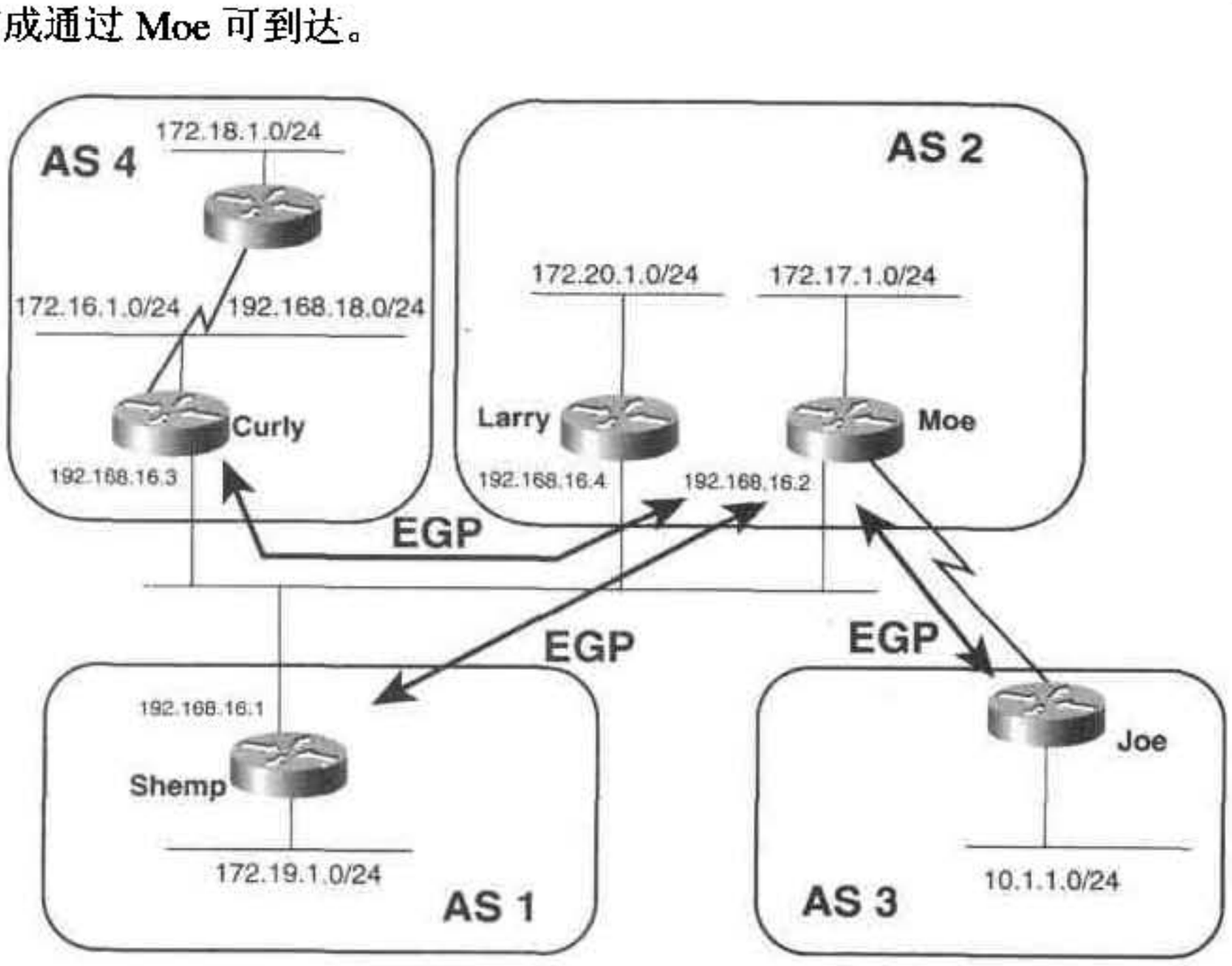


图 1-3　间接 EGP 邻居

间接邻居的公布可以节省一条共用链路上的带宽，但是更重要的是，间接邻居通过消除不必要的路由器跳数从而提高了效率。在图 1-3 中，Shemp 只和 Moe 建立对等关系，

而和任何其他的路由器都不对等。实际上，Larry 甚至不运行 EGP，它只是通过 RIP 把它的网络公布给 Moe。Moe 执行一种“优先重定向”，从而通知 Shemp 更好的下一跳路由器。

实际上，EGP 更新消息只包含间接邻居是可能的——也就是说，消息的发起者可能没把它自己作为到任何网络的下一跳。在这种情况下，消息的发起者是一个*路由服务器*。它已经从 IGP 或者静态路由学习到了可到达性信息，并且它自己没有执行任何包转发的功能，只是把这些信息公布给 EGP 的邻居。

通过对 EGP 网关的观察，发现一个邻居既可以是*内部网关*也可以是*外部网关*。如果它们在同一个 AS 内，该邻居是内部网关；如果它们在不同的 AS 内，那么该邻居属于外部网关。在图 1-3 中，所有的 EGP 网关都把它们的邻居看作外部网关。如果 Larry 运行 EGP 并且与 Moe 建立对等关系，这两个路由器就可以互相看作是内部网关。

一个 EGP 更新消息中包含两个用来描述在它列表中的路由器是内部还是外部网关的字段(参见下一节“EGP 消息格式”)。看一下例 1-5 中的第一个更新消息，你可以看到这些字段刚好在源网络 IntGW=2 和 ExtGW=1 的前面。这两个字段合起来给出了更新消息中列出的路由器数。确定的内部网关列在前面；因此，如果是 IntGW=2 和 ExtGW=1，那么在前面列出的两个路由器是内部网关，最后列出的路由器是外部网关。如果你将例 1-5 中来自 192.168.16.2 的更新消息与图 1-3 相比较，你会发现通过 Curly 可到达的 3 个网络在 Update 消息中被列在最后，并且被标记为外部——也就是说它们要通过 Moe 的一个外部网关才能到达。因为末梢网关不能公布它们自己 AS 域外的网络，因此只有来自核心网关的更新消息才能够包括外部网关。

EGP 更新消息中列出的每个网络都带有一个距离项。距离项是 8 比特，因此该距离的范围是 0~255。除了 255 用来指示不可到达的网络外，RFC 904 并没有规定如何解释这个距离。而且 RFC 也没有定义一种使用该距离来计算 AS 之间最短路径的算法。正如例 1-5 中所表示的，Cisco 将距离解释为跳。缺省的规则是非常基本的：

- 网关公布的所有本 AS 域内的网络，距离均为 0。
- 网关公布的在其他一个 AS 域内而不再本域内的网络，距离为 3。
- 如果一个网络的距离为 255，那么网关指示该网络不可到达。

例如，在例 1-5 和图 1-3 中可以看到虽然网络 172.20.0.0 与 Moe 之间有一个路由器跳，但是 Moe 公布该网络的距离为 0——和网络 172.17.0.0 具有相同的距离，而网络 172.17.0.0 与 Moe 是直接相连的。网络 10.0.0.0 也有一个路由器跳的距离，网络 172.18.0.0 是两跳的距离，当时它们都和 Moe 在不同的 AS 域中，因此它们被公布的距离都为 3。该例子可以说明 EGP 所用的距离在决定到一个网络的最佳路径方面其实是没有用的。

例 1-6 给出了 Shemp 的路由表以及由例 1-5 的更新消息得到的路由条目。

在该路由表中有两个很有意思的地方。第一，注意在 EGP 条目中有一个 140 的管理距离。这比任何一个 IGP(外部 EIGRP 是一个特例)的管理距离都大，因此在 EGP 公布同一个网络的基础上，路由器通常都是选择 IGP 路由。

第二，注意到达每一个 EGP 公布的网络的距离都比例 1-5 更新消息中的距离大 1。和 RIP 路由算法一样，Cisco 的 EGP 过程会把距离增加 1。

例 1-6　Shemp 的路由表

```
Shemp#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

E    10.0.0.0 [140/4] via 192.168.16.2, 00:00:52, Ethernet0
C    192.168.16.0 is directly connected, Ethernet0
E    192.168.17.0 [140/1] via 192.168.16.2, 00:00:52, Ethernet0
E    192.168.18.0 [140/4] via 192.168.16.3, 00:00:52, Ethernet0
E    172.20.0.0 [140/1] via 192.168.16.4, 00:00:52, Ethernet0
E    172.16.0.0 [140/4] via 192.168.16.3, 00:00:52, Ethernet0
E    172.17.0.0 [140/1] via 192.168.16.2, 00:00:52, Ethernet0
E    172.18.0.0 [140/4] via 192.168.16.3, 00:00:52, Ethernet0
     172.19.0.0 255.255.255.0 is subnetted, 1 subnets
C       172.19.1.0 is directly connected, Loopback0
Shemp#
```

1.2.3　EGP 消息格式

EGP 用 5 种不同消息格式对表 1-1 中的 10 种消息类型进行编码。所有的消息都有一个通用的信头，如图 1-4 所示。

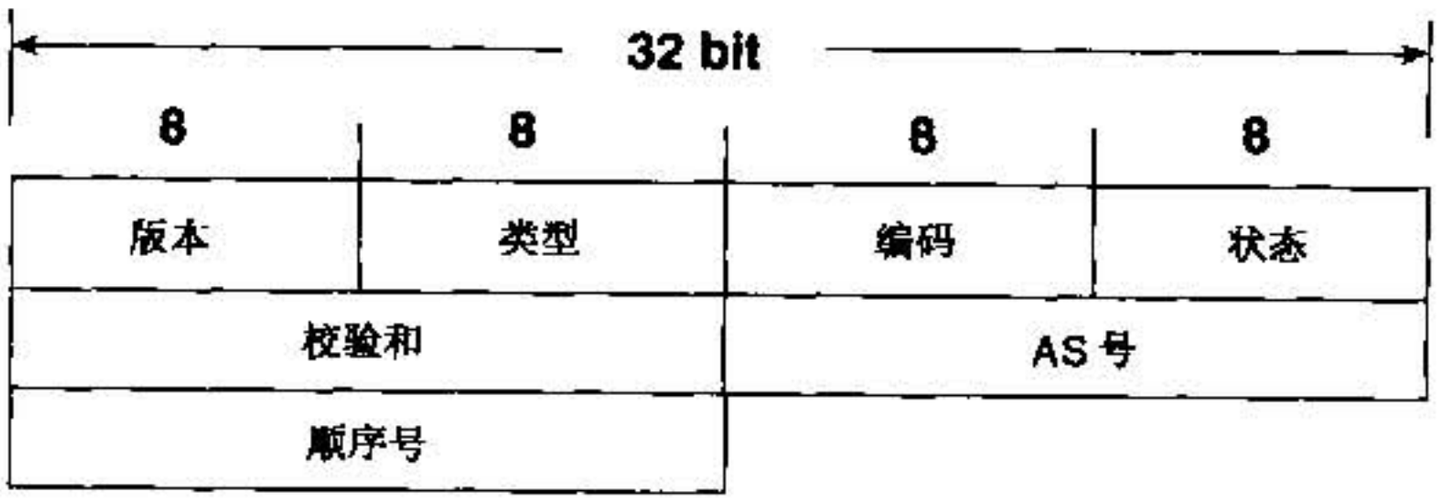


图 1-4　EGP 消息信头

EGP 消息信头中的字段定义如下：

- **版本**——指出当前 EGP 的版本号。如果接收到的消息中的版本号与接收端的版本号不符，那么该消息会被拒绝。当前所有 EGP 执行的版本号都是 2。
- **类型**——指出信头后面是 5 种消息格式中的哪一种。表 1-2 给出了 10 种 EGP 消息类型以及每种消息使用的类型号。
- **编码**——规定了子类型。例如，如果 type=5，该编码会指出这个消息是 Hello 还是 I-Heard-You。
- **状态**——该值会随着消息类型的不同而变化(与 Code 字段类似)。例如，一个 Neighbor Acquisition 消息可以用该状态值来说明这个邻居是主动还是被动，反之，Neighbor Reachability 消息可以用这个状态字段来指示 UP 或者 DOWN 状态。

• **校验和**——EGP 消息补码和的补码。这和 IP、TCP 以及 UDP 所采用的差错检查算法是相同的。

• **自治系统号**——确定消息发起者的 AS。

• **顺序号**——同步消息对(与本章前面所描述的一样)。例如，一个更新消息应该和它所应答的轮询消息具有相同的顺序号。

表 1-2　　EGP 消息类型

类　型	消　息
3	Neighbor Acquisition Request
3	Neighbor Acquisition Confirm
3	Neighbor Acquisition Refuse
3	Neighbor Cease
3	Neighbor Cease Acknowledgement
5	Hello
5	I-Heard-You
2	Poll
1	Update
8	Error

1．邻居获得消息(EGP 消息类型 3)

邻居获得消息是 EGP 消息类型 3。表 1-3 给出了指示 EGP 消息的编码。来自 RFC904 的表 1-4 给出了状态字段可能的值以及使用该特殊状态的原因。

图 1-5 给出了邻居获得消息的格式。Hello 间隔以及 Poll 间隔字段只出现在邻居获得要求(编码 0)消息以及邻居获得证实消息(编码 1)中。所有其他的邻居获得消息都有相同的信头，不再有其他的字段。

表 1-3　　消息 3 使用的编码

编　码	消　息
0	Neighbor Acquisition Request
1	Neighbor Acquisition Confirm
2	Neighbor Acquisition Refuse
3	Neighbor Cease
4	Neighbor Cease Acknowledgement

表 1-4　　消息类型 3 使用的状态号

状　态	描　述	使　用
0	未定义	不适合任何其他的状态
1	主动模式	只用于请求/证实
2	被动模式	只用于请求/证实
3	资源不足	1. 超出了表的空间 2. 超出了系统资源
4	管理禁止	1. 未知的 AS 2. 使用另外一个网关
5	转到终止状态	1. 操作者启动终止进程 2. 超时退出
6	周界问题	1. 无意义的轮询参数 2. 不能使用兼容模式
7	协议冲突	在这个状态接收的无效的命令或响应

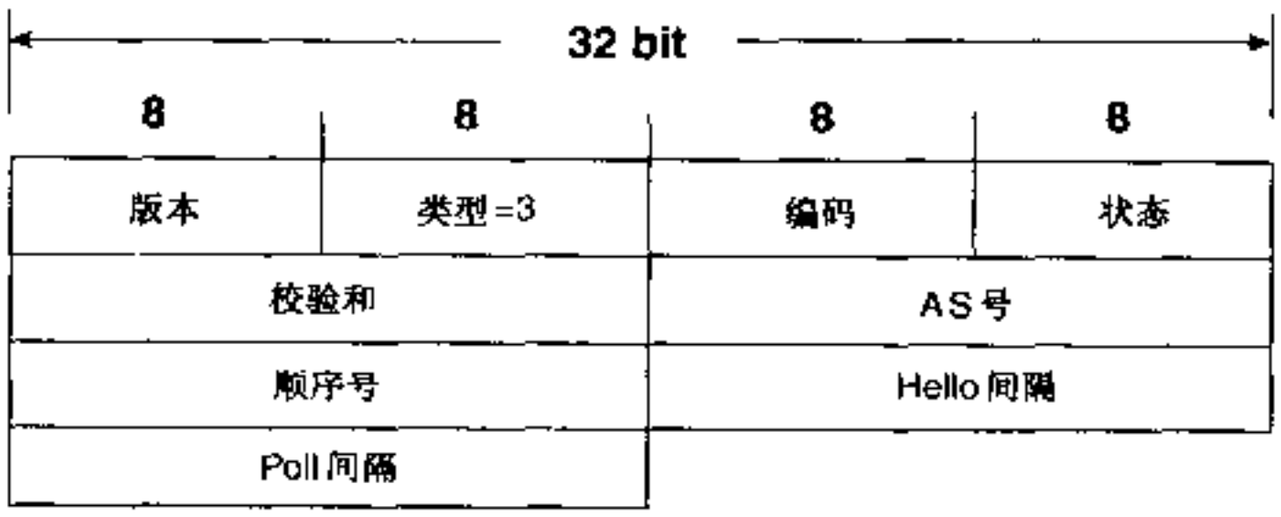


图 1-5　邻居获得消息

• **Hello 时间间隔**——以秒计算的、发起者能够接受的 Hello 之间的最小时间间隔。Cisco 的缺省 Hello 间隔是 60s，可以用 **timers egp** 指令改变该值。

• **Poll 时间间隔**——以秒计算的、发起者能够接受的轮询(Poll)之间的最小时间间隔。Cisco 的缺省 Poll 间隔是 180s，可以用 **timers egp** 指令改变该值。

2. 邻居可到达消息(EGP 消息类型 5)

在 EGP 邻居可到达消息(见图 1-6)的信头中，Type=5。不再有其他别的字段，因为所有必需的信息都携带在编码(见表 1-5)和状态(见表 1-6)字段中。

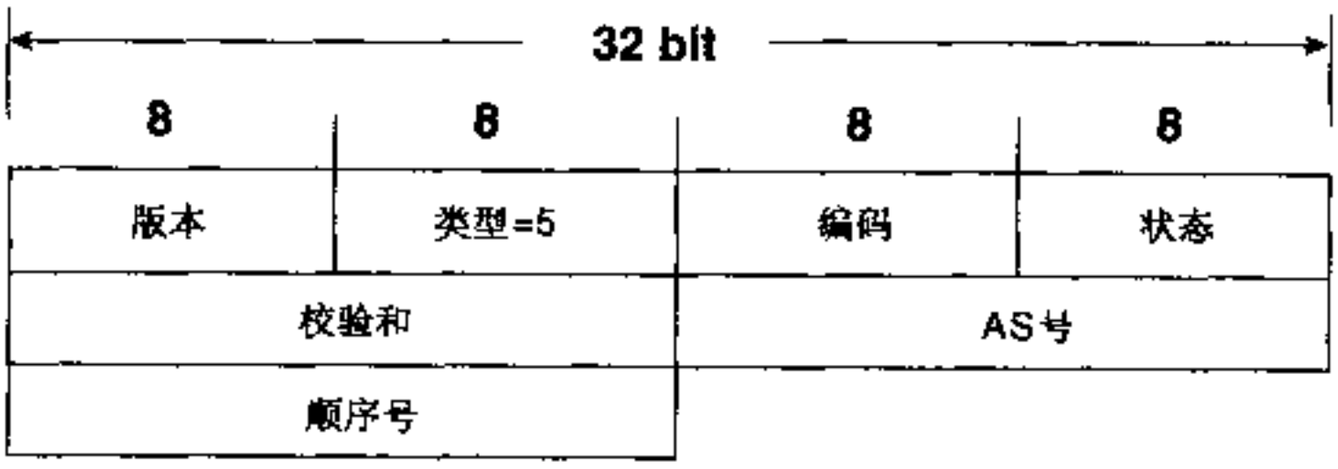


图 1-6　邻居可到达消息

表 1-5　　消息类型 5 使用的编码

编　码	消　息
0	Hello
1	I-Heard-You

表 1-6　　消息类型 5 和 2 使用的状态号

状　态	描　述
0	不确定状态
1	Up 状态
2	Down 状态

3．Poll 消息(EGP 消息类型 2)

在 EGP 消息信头中用来产生 Poll 消息(见图 1-7)的字段是 IP Source Network，它是可到达信息需要的网络。在这个字段中的 IP 地址通常是主要的 A 类、B 类或者 C 类网络地址。编码字段通常是 0，此处使用的状态号与表 1-6 中描述的相同(RFC888 给出了在 Poll 和差错消息中还未使用的状态字段)。

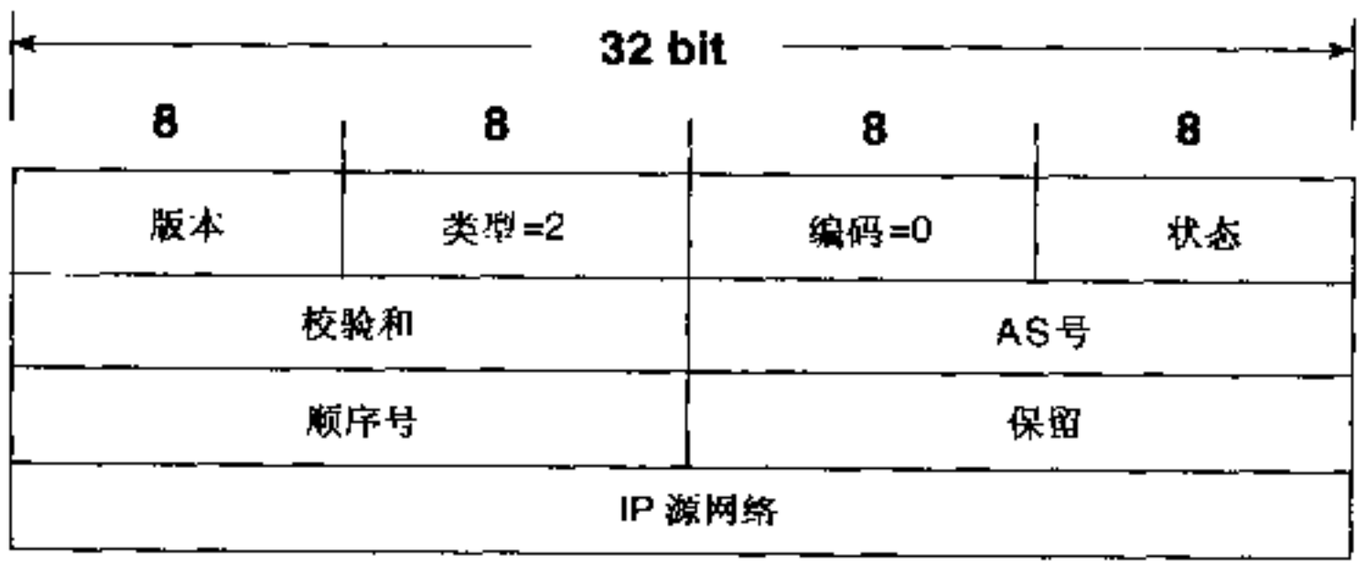


图 1-7　Poll 消息

4．Update 消息(EGP 消息类型 1)

和 Poll 消息一样，Update 消息的编码字段通常是 0。表 1-7 给出了状态字段可能的值，除了无需请求值以外，其他都与表 1-6 给出的值相同。

状态字段的最高有效比特为无需请求比特：如果该比特置位(使该字段的值为 128)，那么这个更新消息就是无需请求消息。无需请求比特可以和其他任何状态值结合使用。

表 1-7　　消息类型 1 所用到的状态号

状　态	描　述
0	不确定的
1	Up 状态
2	Down 状态
128	无需请求

更新消息包括一个四级分层列表。图 1-8 给出了更新消息的格式以及如何组织该分层列表。

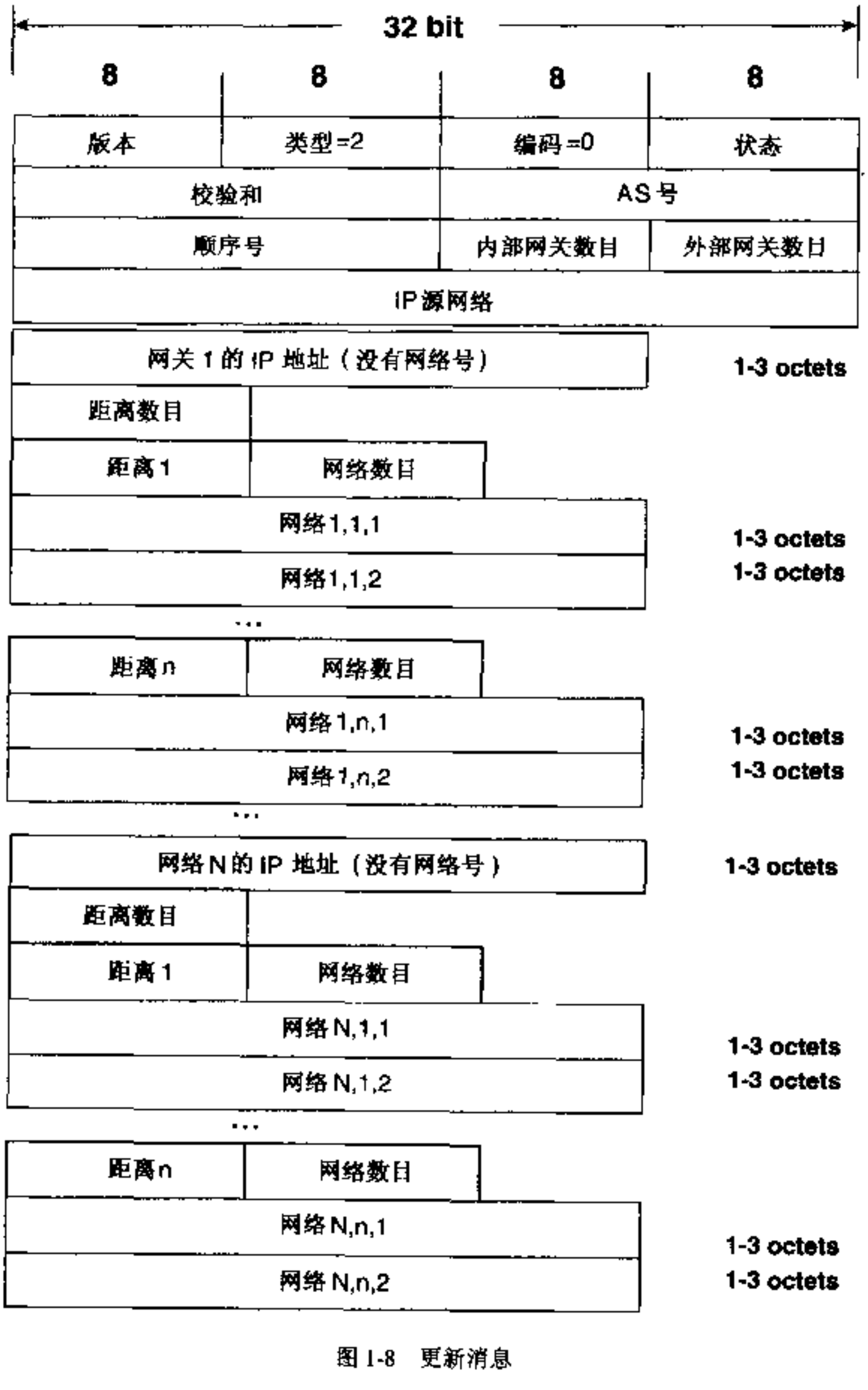


图 1-8　更新消息

在这个分层结构的最高一层是一个与源网络直接相连的所有路由器的列表。列表上网关的数量由内部网关数量字段以及外部网关数量字段的和来确定。

在下一层，将内部网关与外部网关区别开来。所有内部网关，包括消息发起者，被列在前面。如果有外部网关，那么它们列在内部网关的后面。

在这个分层结构的第三层，对于每一个列出的网关都有一个距离列表。针对内部和外部网关，分别有一个字段用来确定列表上距离的数量。

最后，对应每一个列出的距离，都有一个经过该距离以及通过该网关可到达网络的列表。有一个字段用来确定该列表上网络的数量。

对更新消息格式中字段的完整描述如下：

- **内部网关数目**——给出列表中内部网关的数目。
- **外部网关数目**——给出在内部网关列表后面的外部网关的数目。这个字段和内部网关字段的和，如图 1-8 中的 *N*，是更新消息中列出的所有网关总的数量。
- **IP 源网络**——给出可到达信息涉及到的网络。也就是说，更新消息中所有列出的网络都可以通过与该网络相连的一个网关到达。这个字段中的 IP 地址通常是主要的 A 类、B 类或者 C 类网络地址。
- **网关 IP 地址**——给出与源网络相连的网关的地址。只是列出了 A、B、C 类地址的主机部分；结果是，这个字段的长度变化是从 C 类地址的一个字节到 A 类地址的三个字节。从 IP 源网络字段可以知道该地址的网络部分。
- **距离数目**——给出列表中网关公布的距离总数。
- **距离**——给出列表中网关公布的特殊的距离。
- **网络数目**——给出在列表中网关所列出的距离下的网络总数。
- **网络**——给出了已公布网络的 IP 地址。在图 1-8 中可以看出，在网络列表中，每一个网络都属于一个特殊的网关、一个特殊的距离、网络列表中的一个特殊的顺序。与网关 IP 地址字段相同的是，网络字段也是可以变化的。与网关 IP 地址字段不同的是，网络字段列出的是 A、B 或者 C 类地址的网络部分而不是主机部分。

5. 差错消息(EGP 消息类型 8)

网关可以在任何时候发送一个差错消息(见图 1-9)给一个坏的 EGP 消息或者一个无效字段值的发送者。差错消息的编码字段通常是 0，状态字段是表 1-7 中所描述值的任何一个。

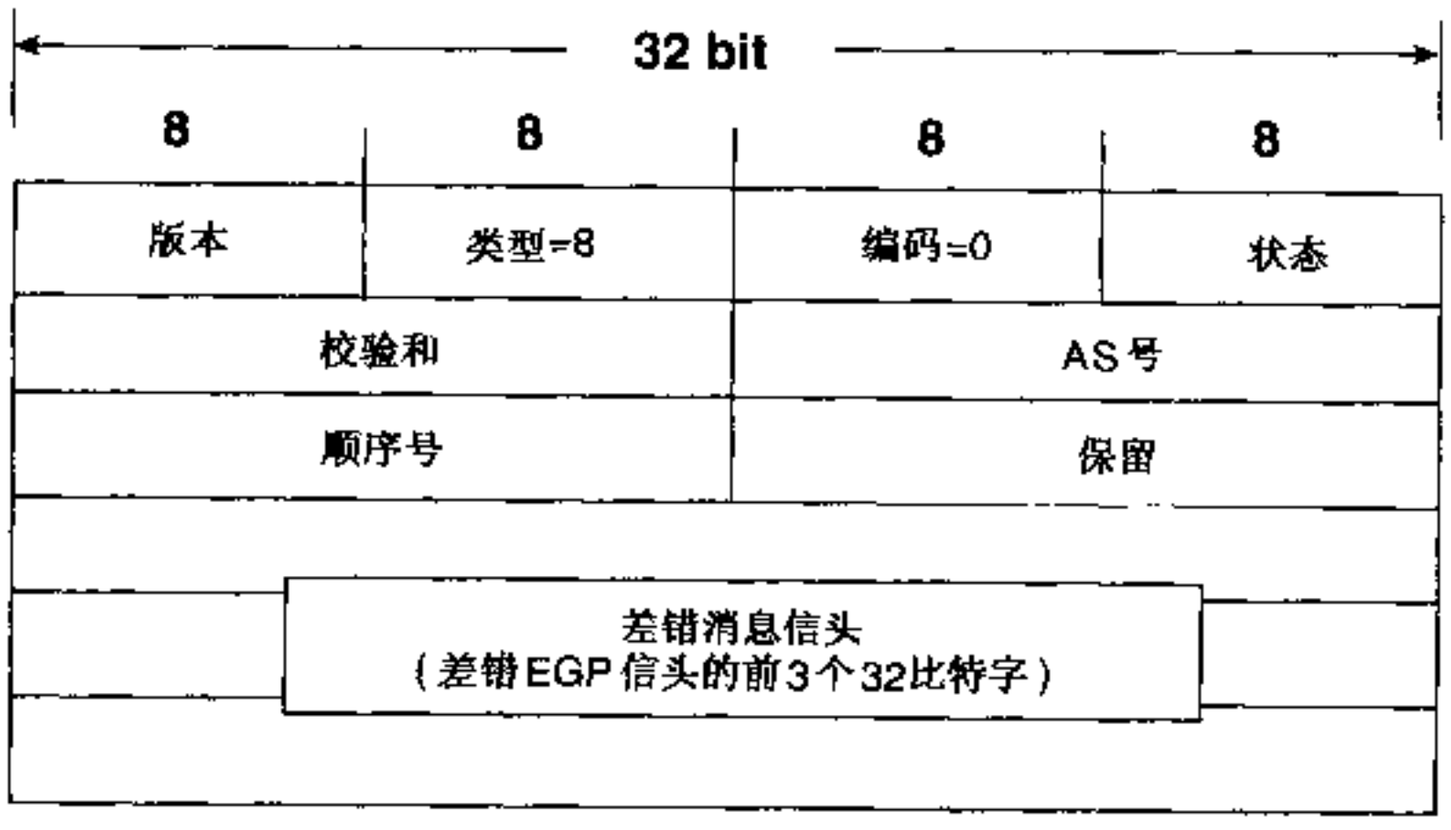


图 1-9　差错消息

注：RFC888 中显示差错消息(像在轮询消息)中的状态字段没有用。RFC904 定义了使用情况，如表 1-7 所示。

差错消息的发起者可以使用一个强制值作为顺序号。来自 RFC904 的表 1-8 描述了原因(Reason)字段可能的值。差错消息信头是 EGP 消息的前 12 个字节，用来提示这是一个差错消息。

表 1-8　　差错消息中原因(Reason)字段的值

原因字段值	描　　述	使　　用
0	没有定义	与其他的都不符
1	坏的 EGP 信头格式	1．坏的消息长度 2．无效的类型、编码或者状态字段
2	坏的 EGP 数据字段格式	1．混乱的轮询速率(请求/证实) 2．无效的更新消息格式 3.应答的 IP Network Address 字段与命令不匹配(更新消息)
3	可到达信息不可用	没有关于在 IP Network Address 中给出的网络的可到达信息(轮询)
4	多余的轮询速率	1．在一个 Hello 间隔内收到了两个或者多个 Hello 消息 2．在一个 Poll 间隔内收到了两个或者多个 Poll 消息 3．在一些(合理短)间隔内收到了两个或者多个 Request 消息
5	没有应答	在一些(合理短)间隔内没有收到针对 Poll 的更新消息

1.3　EGP 的不足

EGP 基本的问题就是它没有发现路由环路的能力。因为 EGP 所使用的距离有一个上限(255)，你可能会说，记数到无穷大至少也是一种检测到路由环路的机制。是的，这是一种办法，但是这种最高的限制与典型的 Poll 间隔结合起来使得这种记数是无用的。如果缺省的 Poll 间隔是 180s，那么 EGP 的对等记数到无穷大需要 13 个小时。

因此，EGP 必须运行在一个设计上无环路的拓扑上。虽然在 1983 年，这种情况不成问题，因为当时 EGP 只是连到与 ARPANET 骨干相连的末梢网关上，但是 EGP 的创始人已经预见到了这种有局限性的拓扑很快就会不适用。由自治系统组成的 Internet 应该演进成一种较少结构的网状，其中许多自治系统可以作为其他自治系统的转接系统。

随着 NSFnet 的出现，EGP 的局限性变得更加明显了。不仅是因为现在出现了多个骨干网，而且还因为现在使用的策略，这些策略主要考虑的是什么样的业务能够穿过什么样的骨干网。因为 EGP 不支持复杂的基于策略的路由，因此必须设计过渡的解决方案[4]。

EGP 另外一个主要的问题是它不能充分地与 IGP 互相合作，从而决定到另一个 AS 域内的

一个网络的最短路由。例如，EGP 的距离不能可靠地翻译成 RIP 的跳(HOP)数。如果 EGP 的距离导致跳数超过 15，RIP 会宣布该网络不可到达。EGP 其他的缺点包括当在很多的网络上传输信息时，它对故障的敏感性，以及它对有意或者无意产生的不充分的网络信息有相当的脆弱性。

最后一个，但肯定不是最小的缺陷，EGP 在公布一个网络变化方面相当地慢。在“解决 EGP 问题”一节中给出了一个例子，一个连接 AS 的 EGP 变成不可达网络。这个例子说明，在一个只有 4 跳距离远的网关确定网络出现故障的时候，几乎已经用去了一个小时的时间。

曾经有几次人们试图想创造一个 EGPv3，但是都没有成功。最终，因为全新 AS 域间协议——BGP 的出现，人们放弃了 EGP。结果是，现在外部网关协议不仅是一个协议的名字，而且也是一类协议的名字，使 ERP 这个概念得到了提升。然而，在现在的自治系统形式以及域间路由中，我们还沿用着 EGP 一些传统的方式。

1.4　配置 EGP

你可以在一个路由器上通过四个步骤来配置 EGP：

第一步　用 **autonomous-system** 命令确定路由器的 AS；

第二步　开始 EGP 过程并且用 **router egp** 命令确定邻居的 AS；

第三步　用 **neighbor** 命令确定 EGP 的邻居；

第四步　确定哪些网络需要由 EGP 公布。

前三步在最开始的案例研究中演示，而随后演示了实现第四步的各种方法。

1.4.1　案例研究：一个 EGP 末梢网关

图 1-10 给出了一个在 AS 65502 内的 EGP 末梢网关，与 AS 65501 内的一个核心网关相连。末梢 AS 的 IGP 是 RIP。

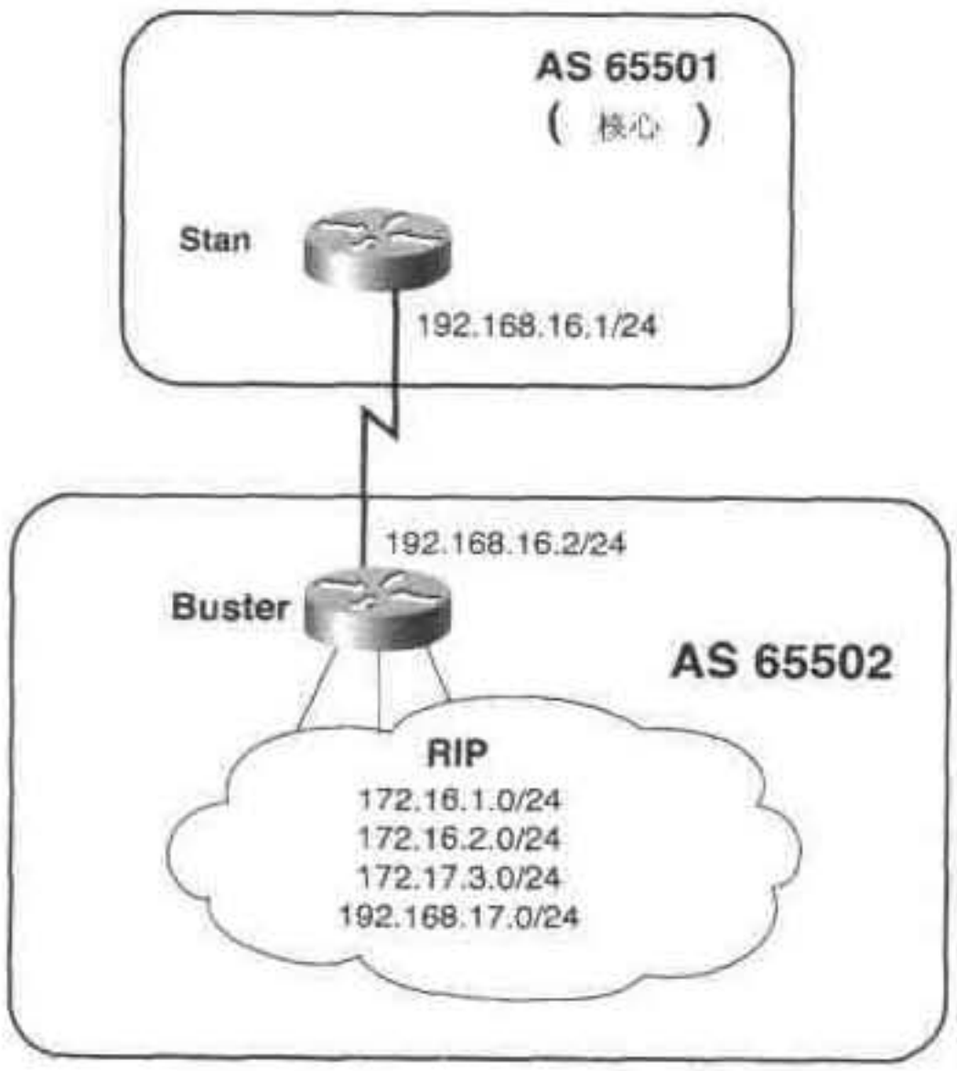


图 1-10　EGP 末梢网关将 AS 65502 的内部网络公布给核心网关

例 1-7 给出了末梢网关的初始配置。

例 1-7　图 1-10 中末梢网关的配置

```
autonomous-system 65502
!
router rip
 redistribute connected
 redistribute egp 65501 metric 5
 network 172.16.0.0
!
router egp 65501
 neighbor 192.168.16.1
```

注意由 **autonomous-system** 命令确认本地 AS(LAS)，远端 AS(FAS)是由 **router egp** 命令确认的。只有 LAS 配置好以后才可以配置 EGP。通过 **neighbor** 命令，EGP 被告知到哪里去寻找它的对端。Buster 的路由表(参见例 1-8)包含 EGP 从核心网关学习的路由条目以及从内部邻居学习到的 RIP。

EGP 学习到的路由被再分发到 RIP 中，这些路由的度量为 5(参见例 1-9)。

注意直连网络也同样被再分发给 RIP。这种配置对于把网络 192.168.16.0 公布到 LAS 是必需的。水平分割可以防止 Stan 通过 EGP 公布网络给 Buster。另外一种可选的的配置是在 RIP 的配置中加入网络 192.168.16.0，同时加入一个 **passive-interface**，从而使 RIP 不向 AS 之间的链路广播。

当 Buster 的 EGP 配置持续了较长时间以后，只能从核心获得网络信息，而没有内部网络信息公布给核心(见例 1-10)。

配置 EGP 用来公布内部网络的一种选择就是加入一个再分发命令。但是，对于互相再分发这里存在着危险。当存在着拓扑环路或者多个再分发点时，这种危险就会更加显著，但是即使是像例 1-10 中的简单的设计，对路由环路来讲也是十分脆弱的。为了安全起见，对于双向再分发的配置，通常要使用路由过滤器，从而保证不会从外部网关接收到内部网络地址，并且没有外部地址公布到外部网关。与双向再分发有关的问题在《TCP/IP 路由技术　第 1 卷》中有介绍，在本书的第 2 章以及第 3 章。中会有更进一步的讨论。

例 1-8　Buster 的路由表显示出从 EGP 邻居处以及从内部 RIP 邻居处学习到的路由条目

```
Buster#show ip route
Codes: C - connected, S   static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

E    10.0.0.0 [140/4] via 192.168.16.1, 00:02:12, Serial3
C    192.168.16.0 is directly connected, Serial3
R    192.168.17.0 [120/1] via 172.16.1.2, 00:00:05, Ethernet0
E    192.168.19.0 [140/4] via 192.168.16.1, 00:02:13, Serial3
E    192.168.20.0 [140/4] via 192.168.16.1, 00:02:13, Serial3
E    192.168.21.0 [140/4] via 192.168.16.1, 00:02:13, Serial3
E    192.168.22.0 [140/4] via 192.168.16.1, 00:02:13, Serial3
     172.16.0.0 255.255.255.0 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, Ethernet0
R       172.16.2.0 [120/1] via 172.16.1.2, 00:00:05, Ethernet0
R    172.17.0.0 [120/1] via 172.16.1.2, 00:00:05, Ethernet0
Buster#
```

例 1-9　从 AS 65502 内部的一个路由器的路由表中可以看出再分发的 EGP 路由

```
Charlie#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

R    10.0.0.0 [120/5] via 172.16.1.1, 00:00:13, Ethernet0
R    192.168.16.0 [120/1] via 172.16.1.1, 00:00:13, Ethernet0
C    192.168.17.0 is directly connected, Ethernet3
R    192.168.19.0 [120/5] via 172.16.1.1, 00:00:13, Ethernet0
R    192.168.20.0 [120/5] via 172.16.1.1, 00:00:13, Ethernet0
R    192.168.21.0 [120/5] via 172.16.1.1, 00:00:13, Ethernet0
R    192.168.22.0 [120/5] via 172.16.1.1, 00:00:13, Ethernet0
     172.16.0.0 255.255.255.0 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, Ethernet0
C       172.16.2.0 is directly connected, Ethernet1
     172.17.0.0 255.255.255.0 is subnetted, 1 subnets
C       172.17.3.0 is directly connected, Ethernet2
Charlie#
```

例 1-10　Stan 的路由表显示出，通过 Buster 没有学习到 AS 65502 的内部网络

```
Stan#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

E    10.0.0.0 [140/4] via 192.168.18.2, 00:01:56, Serial1
C    192.168.16.0 is directly connected, Serial0
C    192.168.18.0 is directly connected, Serial1
E    192.168.19.0 [140/1] via 192.168.18.2, 00:01:57, Serial1
E    192.168.20.0 [140/4] via 192.168.18.2, 00:01:57, Serial1
E    192.168.21.0 [140/4] via 192.168.18.2, 00:01:57, Serial1
E    192.168.22.0 [140/1] via 192.168.18.2, 00:01:57, Serial1
Stan#
```

配置 EGP 用来公布内部网络一个较好的办法是使用 **network** 命令。和 EGP 以及 BGP 一起使用时，与和 IGP 配置一起使用，**network** 命令有着不同的功能。例如，Buster 的 RIP 配置下面的 **network 172.16.0.0** 命令指示该路由器启用在主要网络 172.26.0.0 上具有一个 IP 地址的任何接口上的 RIP。当和 AS 域间协议结合起来使用的时候，**network** 命令告诉协议要公布什么网络地址。例 I-11 给出了 Buster 公布 AS 65502 中所有网络时的配置情况。

例 1-11　Buster 公布 AS 65502 中所有网络时的配置

```
autonomous-system 65502
!
router rip
 redistribute connected
 redistribute egp 65501 metric 5
 network 172.16.0.0
!
router egp 65501
 network 172.16.0.0
 network 172.17.0.0
 network 192.168.17.0
 neighbor 192.168.16.1
```

例 1-12 给出了在 Buster 的 EGP 配置中加入 **network** 命令以后，Stan 的路由表。

例 1-12　现在 Buster 向 Stan 公布 AS 65502 的内部网络

```
Stan#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

E    10.0.0.0 [140/4] via 192.168.18.2, 00:00:27, Serial1
C    192.168.16.0 is directly connected, Serial0
E    192.168.17.0 [140/1] via 192.168.16.2, 00:01:38, Serial0
C    192.168.18.0 is directly connected, Serial1
E    192.168.19.0 [140/1] via 192.168.18.2, 00:00:27, Serial1
E    192.168.20.0 [140/4] via 192.168.18.2, 00:00:27, Serial1
E    192.168.21.0 [140/4] via 192.168.18.2, 00:00:27, Serial1
E    192.168.22.0 [140/1] via 192.168.18.2, 00:00:27, Serial1
E    172.16.0.0 [140/1] via 192.168.16.2, 00:01:39, Serial0
E    172.17.0.0 [140/1] via 192.168.16.2, 00:01:39, Serial0
Stan#
```

在 EGP 下使用 **network** 命令而不使用再分发的好处，与使用静止路由而不是使用动态路由协议的优点类似：这两者都能够准确控制网络的可到达性。在 EGP 情况下，这种精确性受到了 EGP 无类别的限制。虽然你可以在 **network** 命令中不标明一个主要网络从而保持它的“保密性”，但是同样的情况对于一个子网来讲是不成立的。让我们再看一下例 1-8，在这个例子中，Buster 的路由表包含子网 172.16.1.0/24 和 172.16.2.0/24。重新查看一下例 1-8 中 EGP 更新消息格式，更新消息只携带 IP 网络主要类别部分：A 类网络的第一个字节，B 类网络的前两个字节，C 类网络的前三个字节。所以，EGP 下的 **network** 命令只能指定主要网络。

1.4.2　案例研究：一个 EGP 核心网关

根据定义，一个 EGP 核心网关可以与多个远端自治系统中的多个邻居建立对等关系，并

且可以将网络信息从一个 FAS 传到另一个 FAS。因为这一点，对核心网关的配置略有不同。图 1-11 说明了核心路由器 Stan 与一个 FAS 内的一个路由器(Buster)和 LAS 内的一个路由器(Ollie)有对等关系。

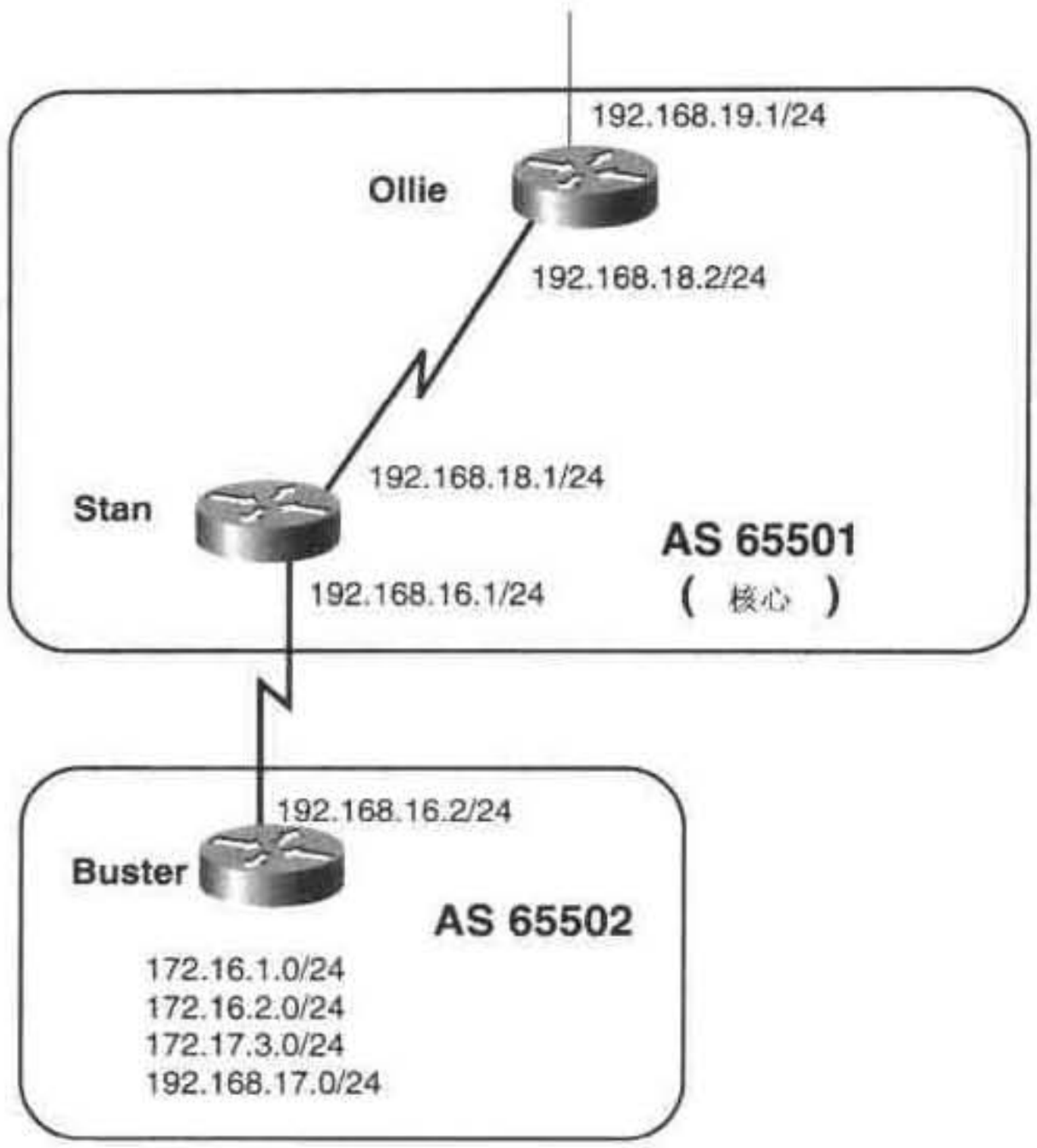


图 1-11　核心路由器 Stan 必须同时为远程邻居 Buster 以及本地邻居 Ollie 的对端

例 1-13 说明了图 1-11 中 Stan 的 EGP 配置。

例 1-13　图 1-11 中的网络拓扑配置核心网关

```
autonomous-system 65501
!
router egp 0
 network 192.168.16.0
 neighbor any
```

仍然由 **autonomous-system** 命令指定 LAS，但是 FAS 却不再由 **router egp** 命令来指定。相反，AS 号 0 用来指定任何一个 AS。同样地，邻居由 **neighbor any** 命令来指定，用来应答发送 Acquisition 消息的邻居。**neighbor any** 命令隐含地配置了邻居，而 **neighbor** 命令却显式地配置了邻居。核心网关可以有显式配置的邻居，但是当一个核心网关有大量的邻居时，正如核心网关所期待的，隐含的 **neighbor any** 可以使配置工作变得简单。

当然，至少有一个邻居必须是显式邻居配置；如果两个邻居都有 **neighbor any** 命令，那么它们不能互相发现对方。例 1-14 给出了图 1-11 中 Ollie 的邻居配置。

例 1-14　图 1-11 的网络拓扑中 Ollie 的邻居配置

```
autonomous-system 65501
!
router egp 0
 network 192.168.19.0
 neighbor 192.168.18.1
 neighbor any
```

虽然 Ollie 仍然用 **neighbor any** 命令获得它的外部邻居，但是 Stan 的地址是显式配置的。如果不是这样，Stan 和 Ollie 将不能互相知道双方的存在。

在例 1-14 配置中，核心网关要把外部网络的可到达信息传给自己的 AS 以及外部所有的 AS。但是，核心网关不会传递它自己 AS 域内的网络信息。例如，在例 1-8 中，Buster 的路由表中没有有关网络 192.168.18.0 的条目。如果必须公布内部网络，Stan 必须使用 **network** 命令从而使每一个网络都被公布。在这里，唯一的一条 **network** 是针对 192.168.16.0 的，它使得 Ollie 可以收到有关该网络的信息。再重新看一下 Buster 的路由表，注意这里有一个关于网络 192.168.19.0 的条目，这个条目是因为在例 1-14 中，在 Ollie 的配置里有 **network 192.168.19.0** 的结果。

如果核心网关不应该和每一个运行 EGP 的邻居对等时，会出现什么情况呢？在图 1-12 中，AS 65506 内的三个路由器都运行 EGP，但是 Stan 应该为 Spanky 和 Buckwheat 的对端，Alfalfa 应该为 Ollie 对端。当然，核心网关的管理者应该相信 AS 65506 的管理者已经通过 **neighbor** 命令正确配置了对等关系，但是在 AS 域间路由中，只有信任是不够的。

在这个例子中，在 AS 65506 的三个网关中都有用于 Stan 和 Ollie 的 **neighbor** 命令。为了管理这种对等关系，和 **neighbor any** 命令一起使用了一个访问列表，如例 1-15 中 Stan 的配置所示。

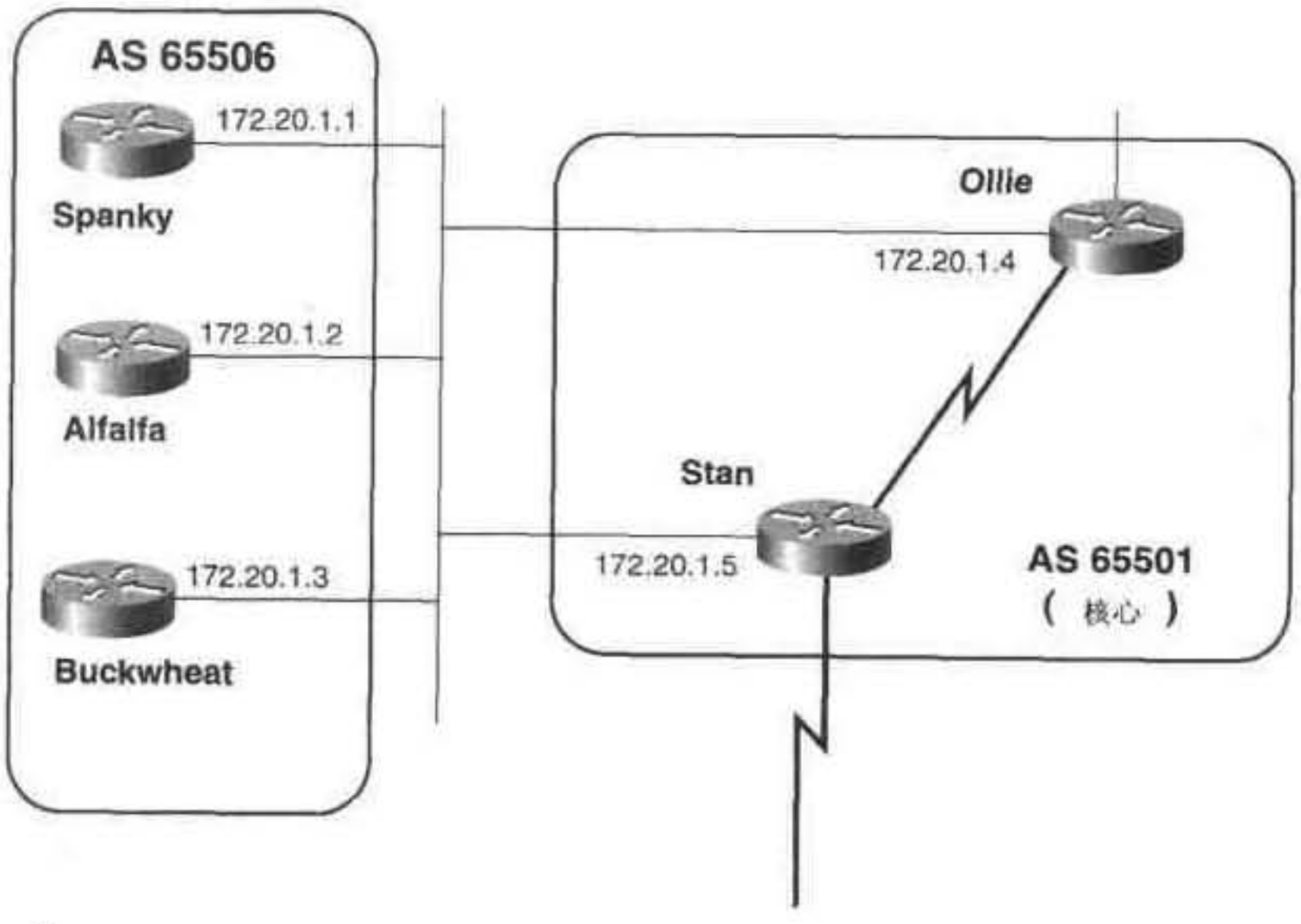


图 1-12　Spanky 和 Buckwheat 与 Stan 有对等关系，Alfalfa 必须只和 Ollie 有对等关系

例 1-15　通过 neighbor any 命令用访问列表来管理对等关系

```
autonomous-system 65501
!
router egp 0
 network 192.168.16.0
 neighbor any 10
!
access-list 10 deny 172.20.1.2
access-list 10 permit any
```

在例 1-15 中，**neighbor any** 命令提到了访问列表 10，它拒绝 Alfalfa(172.20.1.2)但是允许其他所有的邻居。在 Ollie 处的相似配置目的是拒绝 **Spanky** 和 **Buckwheat** 但是允许其他所有的邻居。例 1-16 给出了这种配置相应的结果。

例 1-16　show ip egp 命令显示了 EGP 邻居的信息

```
Stan#show ip egp
Local autonomous system is 65501

 EGP Neighbor     FAS/LAS  State      SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.18.2    65501/65501 UP    10      3      4    60   180   4 Temp, Act
*192.168.16.2    65502/65501 UP  3:20      39     39    60   180   4 Temp, Act
*172.20.1.1      65506/65501 UP    4       2      2    60   180   4 Temp, Act
*172.20.1.3      65506/65501 UP    10      4      4    60   180   4 Temp, Act
Stan#

Ollie#show ip egp
Local autonomous system is 65501

 EGP Neighbor     FAS/LAS  State      SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.18.1    65501/65501 UP    9       4      3    60   180   4 Perm, Pass
*172.20.1.2      65506/65501 UP    13      5      5    60   180   4 Temp, Act
Ollie#
```

对 Stan 和 Ollie 使用了 **show ip egp** 命令后显示出 Ollie 与 Alfalfa 有对等关系，Stan 与 Spanky 和 Buckwheat 有对等关系。

注：**show ip egp** 命令所显示字段的详细情况将在“EGP 的故障排除”一节中给予讨论。现在我们关心的问题是邻居的地址。

1.4.3　案例研究：间接邻居

在图 1-13 中，三个末梢网关(Groucho、Harpo 和 Chico)都连到核心网关 Ollic 上。Groucho 和 Harpo 在不同的自治系统内共享一个以太网，因此它们被配置成间接的或者第三方邻居。

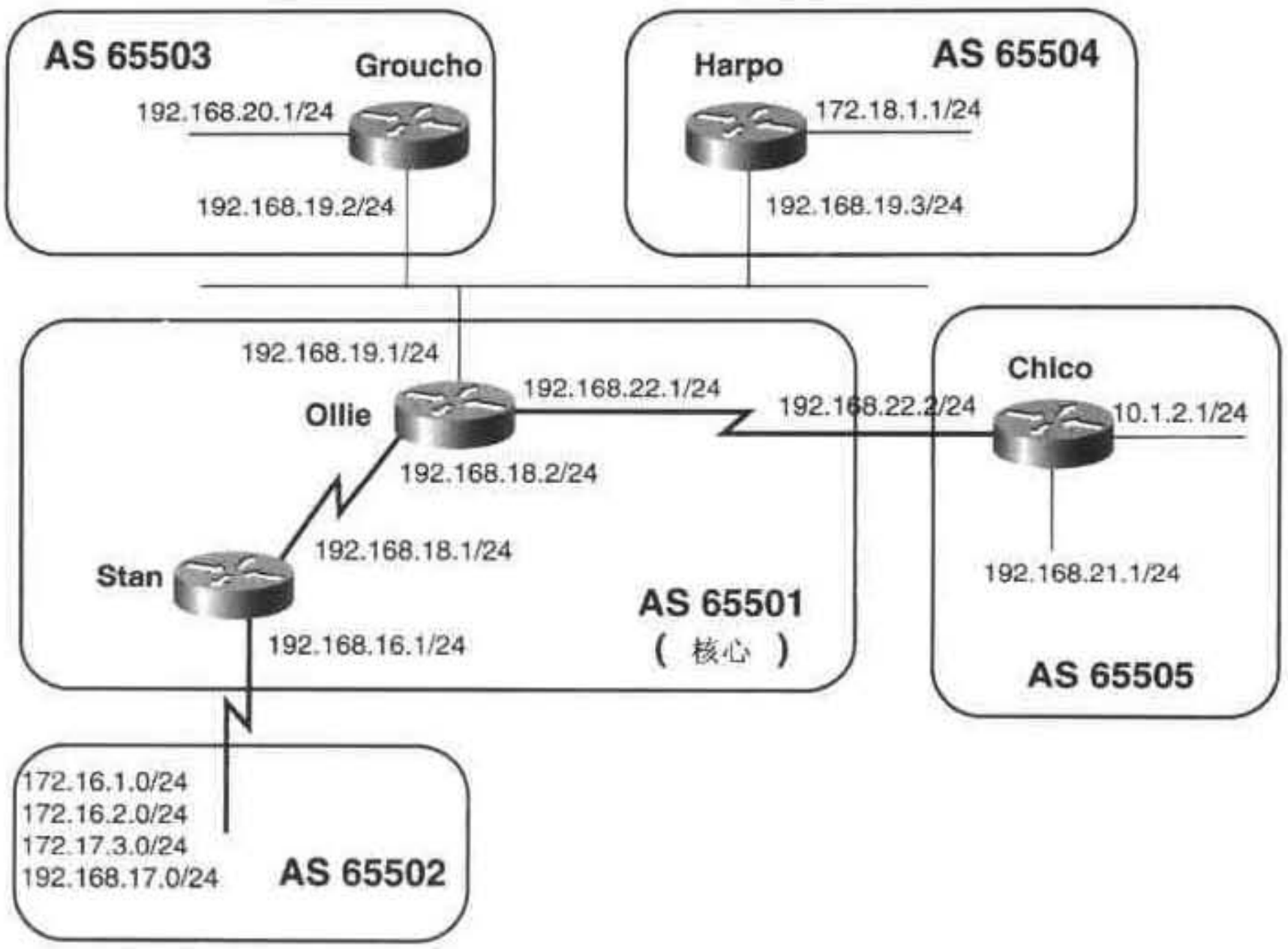


图 1-13　EGP 间接邻居

Groucho 和 Harpo 不能直接交换 EGP 信息，但是如果 Ollie 将它们公布为间接邻居，它们就能够互相直接传递数据包。例 1-17 给出了 Ollie 的配置。

在例 1-17 中的配置中，Groucho 和 Harpo 被显式地配置为邻居，在对两个路由器使用了 **neighbor** 命令之后是 **neighbor third-party** 命令。这些条目确认了有疑问的邻居，并且指明了在共享以太网上该网关的间接邻居。注意 Chico 不在共享以太网上，因此它在 **neighbor any** 命令下。例 1-18 中，核心网关的间接邻居记录为第三方(Third Party)。

例 1-17　彼此公布为间接 EGP 邻居从而在间接 EGP 邻居之间可以为数据包进行选路

```
autonomous-system 65501
!
router egp 0
 network 192.168.19.0
 network 192.168.22.0
 network 192.168.18.0
 neighbor 192.168.19.3
 neighbor 192.168.19.3 third-party 192.168.19.2
 neighbor 192.168.19.2

 neighbor 192.168.19.2 third-party 192.168.19.3
 neighbor 192.168.18.1
 neighbor any
```

例 1-18　显示核心网关的间接邻居

```
Ollie#show ip egp
Local autonomous system is 65501

 EGP Neighbor     FAS/LAS     State     SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.19.3    65504/65501 UP 5TE        8    249    60    180   4 Perm, Act
*192.168.19.2    65503/65501 UP 5TE        8   3177    60    180   4 Perm, Act
*192.168.18.1    65501/65501 UP 5TE        9   3192    60    180   4 Perm, Pass
*192.168.22.2    65505/65501 UP 5TE        5   3170    60    180   4 Temp, Act
 EGP Neighbor     Third Party
*192.168.19.3     192.168.19.2
*192.168.19.2     192.168.19.3
Ollie#
```

Ollie 的 EGP 邻居表指出，将 Groucho 和 Harpo(分别是 192.168.19.2 和 192.168.19.3)配置成彼此互为间接邻居。

Harpo 的路由表(见例 1-19)给出了间接邻居配置的结果。没有将核心网关指定为到 AS 65503 内网络 192.168.20.0 的下一跳，下一跳直接指向 Groucho(192.168.19.2)。

例 1-19 中 Harpo 的路由表显示出通过下一跳 192.168.19.2 可以直接到达网络 192.168.20.0。由于没有间接邻居配置，Harpo 会把 192.168.19.1 当作下一跳。

例 1-19　路由表显示到间接邻居的下一跳路由

```
Harpo#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

E    10.0.0.0 [140/4] via 192.168.19.1, 00:02:21, Ethernet0
E    192.168.16.0 [140/4] via 192.168.19.1, 00:02:21, Ethernet0
E    192.168.17.0 [140/4] via 192.168.19.1, 00:02:21, Ethernet0
E    192.168.18.0 [140/1] via 192.168.19.1, 00:02:21, Ethernet0
C    192.168.19.0 is directly connected, Ethernet0
E    192.168.20.0 [140/4] via 192.168.19.2, 00:02:21, Ethernet0
E    192.168.21.0 [140/4] via 192.168.19.1, 00:02:22, Ethernet0
E    192.168.22.0 [140/1] via 192.168.19.1, 00:02:22, Ethernet0
E    172.16.0.0 [140/4] via 192.168.19.1, 00:02:22, Ethernet0
E    172.17.0.0 [140/4] via 192.168.19.1, 00:02:22, Ethernet0
     172.18.0.0 255.255.255.0 is subnetted, 1 subnets
C       172.18.1.0 is directly connected, Loopback0
Harpo#
```

1.4.4　案例研究：缺省路由

EGP 可以配置为除了公布较为具体的路由以外还可以公布缺省路由。如果一个 AS 只有一个单一的外部网关，一条缺省路由通常要比所有的外部路由效率更高。路由器上不仅节省内存和处理时间，而且还可以节省链路上的带宽。

正如前面在图 1-13 中所示出的，为了向 AS 65502 公布一条缺省路由，必须按照例 1-20

所讲来配置 Stan。

default-information originate 命令用来产生缺省路由。不像其他的协议，当 EGP 使用这个命令时，没有可选的命令项。同时，还应该注意，这里加入了一个路由过滤器，该过滤器只允许缺省路由从 Stan 的 S0 端口公布到 AS 65502 中去。没有这个过滤器，缺省的以及所有更加具体的网络都应该被公布。例 1-21 给出了这种配置的结果。

例 1-20　公布一条缺省路由

```
router egp 0
 network 192.168.16.0
 neighbor any
 default-information originate
 distribute-list 20 out Serial0
!
access-list 20 permit 0.0.0.0
```

例 1-21　缺省路由的一个结果就是 192.168.20.1 是可到达的

```
Buster#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is 192.168.16.1 to network 0.0.0.0

C    192.168.16.0 is directly connected, Serial3
R    192.168.17.0 [120/1] via 172.16.1.2, 00:00:20, Ethernet0
     172.16.0.0 255.255.255.0 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, Ethernet0
R       172.16.2.0 [120/1] via 172.16.1.2, 00:00:21, Ethernet0
R    172.17.0.0 [120/1] via 172.16.1.2, 00:00:21, Ethernet0
E*   0.0.0.0 0.0.0.0 [140/4] via 192.168.16.1, 00:00:46, Serial3

Buster#ping 192.168.20.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.20.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 64/66/76 ms
Buster#
```

AS 65502 外部网关的路由表说明核心网关只公布缺省路由，通过这条缺省路由，图 1-13 中的所有外部网络都可到达。

1.5　EGP 的故障排除

在 1.3 节中针对为什么 EGP 不能用在复杂的 AS 域间拓扑中，讨论了几个原因。通过强制使用一种简单的拓扑，一个意外的好处是，EGP 比较容易进行故障排除。

和其他的路由协议一样，在 EGP 中排除故障的第一步是检查路由表。如果丢失了一条必需的路由或者出现了一条不需要的路由，路由表就会指导你找到问题的根源。因为 EGP 度量几乎没有什么意义，与其他路由协议相比，用路由表来解决问题是相当简单的。

当检查 EGP 的配置时，记住网关必须有一些 neighbor 之类的命令，对于每一个邻居来讲，可以是显式的也可以是 neighbor any。理解 **network** 命令的使用，同时理解与 IGP 中使用的 **network** 命令的不同，同样是十分重要。

debug ip egp transactions 命令，是一个十分重要的排除故障的工具，曾经在“EGP 的操作”一节中使用过几次。这个命令的输出可以显示出在邻居之间交换的所有 EGP 消息中的全部重要信息。

1.5.1　解释邻居表

使用 **show ip egp** 命令对 EGP 邻居表进行检查，从而可以知道网关邻居的状态以及配置情况。例 1-18 给出了这个命令的输出，例 1-22 给出了一些在使用 **show ip egp** 命令检查 Stan 的邻居表时的附加输出。

例 1-22　show ip egp 命令的输出显示了解决 EGP 对等问题上的有用信息

```
Stan#show ip egp
Local autonomous system is 65501

 EGP Neighbor     FAS/LAS    State    SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.18.2    65501/65501 UP  2:08   3227     43     60   180   4 Temp, Act
*192.168.16.2    65502/65501 UP  6d17   3233   3233     60   180   4 Temp, Act
Stan#
```

可以看到在 Stan 的邻居表中，邻居 192.168.18.2 是一个内部邻居，因为在这里 FAS 和 LAS 是一样的(65501)，给出了邻居的状态以及它们的正常的运行时间。192.168.18.2 已经正常运行了两个多小时，192.168.16.2 已经正常运行了 6 天又 17 个小时。给出了由网关使用的每个邻居的顺序号以及由邻居使用的顺序号。

经过了 Hello 和 Poll 间隔以后，在过去 4 个 Hello 间隔内接收到邻居可到达消息的数量被记录下来。在 *j*、*k* 门限值的基础上，这个数量可以用来决定是否应该将一个邻居宣布为 UP 或者 DOWN。*j* 门限是指在将一个 DOWN 邻居宣布为 UP 之前，在 4 个 Hello 间隔内必须收到的邻居可到达性消息的数量。*k* 门限规定了在 4 个 Hello 间隔内必须收到的邻居可到达性消息的最少数量，从而防止一个处于工作状态的邻居被宣布为停机状态。表 1-9 列出了这些门限值，对于主动和被动邻居来讲，这些值是不同的。

表 1-9　EGP 中 *j* 和 *k* 的门限

门　限	主　动	被　动	描　述
J	3	1	邻居 Up 的门限
K	1	4	邻居 Down 的门限

例 1-22 中下一个字段(标志)表明了该邻居是永久的还是暂时的。永久邻居是用 **neighbor** 命令来显式配置的邻居，而暂时邻居是在 **neighbor any** 命令下隐含对等的邻居。在例 1-22 中，Stan 的两个邻居都是暂时的；这与前面讨论的 Stan 的配置相符，在那个配置中只有一个

neighbor any 命令。将例 1-22 与例 1-18 对比一下，可以看出一个很有意思的地方，那就是虽然 Stan 把 Ollie(192.168.18.2)看作暂时邻居，Ollie 却把 Stan(192.168.18.1)看作永久邻居。检查一下例 1-23 中 Ollie 的配置就可以找到原因。

例 1-23　路由器 Ollie 的邻居配置

```
autonomous-system 65501
!
router egp 0
 network 192.168.19.0
 network 192.168.22.0
 network 192.168.18.0
 neighbor 192.168.19.3
 neighbor 192.168.19.3 third-party 192.168.19.2
 neighbor 192.168.19.2
 neighbor 192.168.19.2 third-party 192.168.19.3
 neighbor 192.168.18.1
 neighbor any
```

显式的 **neighbor 192.168.18.1** 命令使 Ollie 将 Stan 看作永久邻居。

最后一个字段表明本地路由器是主动的还是被动的邻居。例 1-22 表示 Stan 在两个对等关系中都是主动邻居，于是你可能就会希望 Ollie 是被动邻居。例 1-18 证明了这种假设并且指出 Ollie 在所有它的对等关系中是主动邻居。这同样也是我们所希望的，因为 AS 65501 的 AS 号要比其他的 AS 的号小。

1.5.2　案例研究：聚合到 Syrup 的速度

EGP 一个最显著的特点就是任何动作都很慢。邻居获得的过程慢，公布网络变化几乎就是凝固不动。结果就是，有时候你可能会错误地认为某个地方出现了问题，实际上并没有任何问题(除非是有 EGP 自身的缺陷造成的)。例如，假设图 1-13 中 AS 65503 中的用户抱怨他们无法到达 AS 65502 中的网络 172.17.0.0。检查 Groucho 的路由表，这里有一条到 172.17.0.0 的路由(参见例 1-24)，但是 ping 在该网络上的一个地址却失败了。你可能就会认为网络的业务量被错误地选路了，或称黑洞效应。

在 Ollie 的路由表中给出了该问题的一个线索(参见例 1-25)。注意一个关于网络 172.17.0.0 的更新消息在 16 分钟后还没有收到，但是到该网络的路由条目还是有效的并且还一直公布给 Ollie 的邻居。

例 1-24　图 1-13 中的 Groucho 有一条到 172.17.0.0 的路由，但是该网络是不可到达的

```
Groucho#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is 192.168.19.1 to network 0.0.0.0

E.    10.0.0.0 [140/4] via 192.168.19.1, 00:01:23, Ethernet0
E     192.168.16.0 [140/4] via 192.168.19.1, 00:01:23, Ethernet0
E     192.168.17.0 [140/4] via 192.168.19.1, 00:01:23, Ethernet0
C     192.168.19.0 is directly connected, Ethernet0
```

```
C    192.168.20.0 is directly connected, Loopback0
E    192.168.21.0 [140/4] via 192.168.19.1, 00:01:24, Ethernet0
E    192.168.22.0 [140/1] via 192.168.19.1, 00:01:24, Ethernet0
E    172.16.0.0 [140/4] via 192.168.19.1, 00:01:24, Ethernet0
E    172.17.0.0 [140/4] via 192.168.19.1, 00:01:24, Ethernet0
E    172.18.0.0 [140/4] via 192.168.19.1, 00:01:24, Ethernet0
E*   0.0.0.0 0.0.0.0 [140/4] via 192.168.19.1, 00:01:24, Ethernet0

Groucho#ping 172.17.3.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.3.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Groucho#
```

例 1-25　没有公布新的网络更新消息

```
Ollie#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

E    10.0.0.0/8 [140/1] via 192.168.22.2, 00:01:20, Serial1
E    192.168.16.0/24 [140/1] via 192.168.18.1, 00:01:13, Serial0
E    192.168.17.0/24 [140/4] via 192.168.18.1, 00:16:14, Serial0
C    192.168.18.0/24 is directly connected, Serial0
C    192.168.19.0/24 is directly connected, Ethernet0
E    192.168.20.0/24 [140/1] via 192.168.19.2, 00:02:06, Ethernet0
E    192.168.21.0/24 [140/1] via 192.168.22.2, 00:01:21, Serial1
C    192.168.22.0/24 is directly connected, Serial1
E    172.16.0.0/16 [140/4] via 192.168.18.1, 00:01:13, Serial0
E    172.17.0.0/16 [140/4] via 192.168.18.1, 00:16:14, Serial0
E    172.18.0.0/16 [140/1] via 192.168.19.3, 00:01:59, Ethernet0
Ollie#
```

在过去传给 Ollie 的 5 个更新消息中，Stan 没有包括网络 172.17.0.0，因此这里不存在黑洞效应；因为在 AS 65502 内一个路由器上的一个断开的以太网端口导致网络 172.17.0.0 不可到达。EGP 只有在它没有收到有关这条路由的 6 个连续的更新消息时，它才会宣布这条路由断掉了。将这种情况与 180s 的更新消息间隔结合起来，你可以看到 EGP 要花 18 分钟的时间来宣布一条路由断掉了。也只有在这时它才会停止在它的更新消息中包含该网络。在图 1-13 中的网络中，AS 65502 的外部网关宣布网络 172.17.0.0 断掉与 Groucho 宣布该网络断掉了之间会相差 54 分钟。

1.6　尾注

[1]Eric Rosen，“RFC 827：EXTERIOR GATEWAY PROTOCOL(EGP)”(正在研究过程中)

[2]Linda J. Seamonson 和 Eric C.Rosen，“RFC 888：‘STUB’ EXTERIOR GATEWAY PROTOCOL”(正在研究过程中)

[3]D.L. Mills，“RFC 904：Exterior Gateway Protocol Formal Specification”(正在研究过程中)

[4]J. Rekhter，“RFC 1092：EGP and Policy Based Routing in the New NSFNET Backbone”(正在研究过程中)

1.7　展望

本章探讨了发明 AS 域间路由协议的动机以及证明 EGP 已经不充分的理由。第 2 章要介绍取代 EGP 协议的边界网关协议，并且讨论它的操作。表 1-10 是对第 1 章命令的复习。

表 1-10　　第 1 章命令复习

命　令	作　用
autonomous-system *local-as*	明确 EGP 路由器所在的本地自治系统
debug ip egp transactions	显示 EGP 消息交换以及状态变化的信息
default-information originate	让 EGP 公布一个缺省的路由
neighbor *ip-address*	定义一个 EGP 邻居的 IP 地址
neighbor any [*acess-list-number* | *name*]	让 EGP 试着与任何一个发起邻居获得协议的路由器建立对等
neighbor any third-party *ip-address* [**internal** | **external**]	配置一个间接 EGP 邻居
Neighbor *ip-address* **third-party** *third-part-ip-address* [**internal** | **external**]	配置 EGP，让它根据间接邻居来发送更新消息
network *network-number*	明确应该向 EGP 对等公布的 IGP 路由表中的网络
router egp *remote-as*	配置一个 EGP 路由进程
router egp 0	配置一个 EGP 核心网关过程
Sbow ip egp	显示 EGP 连接以及邻居的信息
timers egp *hello polltime*	将 EGP Hello 和 Poll 间隔设定为与缺省值不同的一个值

1.8　复习问题

在附录 D“复习问题的答案”中可以找到复习问题的答案。

1．当前的 EGP 版本是多少？

2．什么是 EGP 内部邻居？什么是 EGP 外部邻居？

3．EGP 末梢网关和 EGP 核心网关之间主要的区别是什么？

4．为什么 EGP 采用核心或者骨干 AS 的概念？

5．主动 EGP 邻居和被动 EGP 邻居之间的区别是什么？

6．EGP 轮询(Poll)消息的目的是什么？

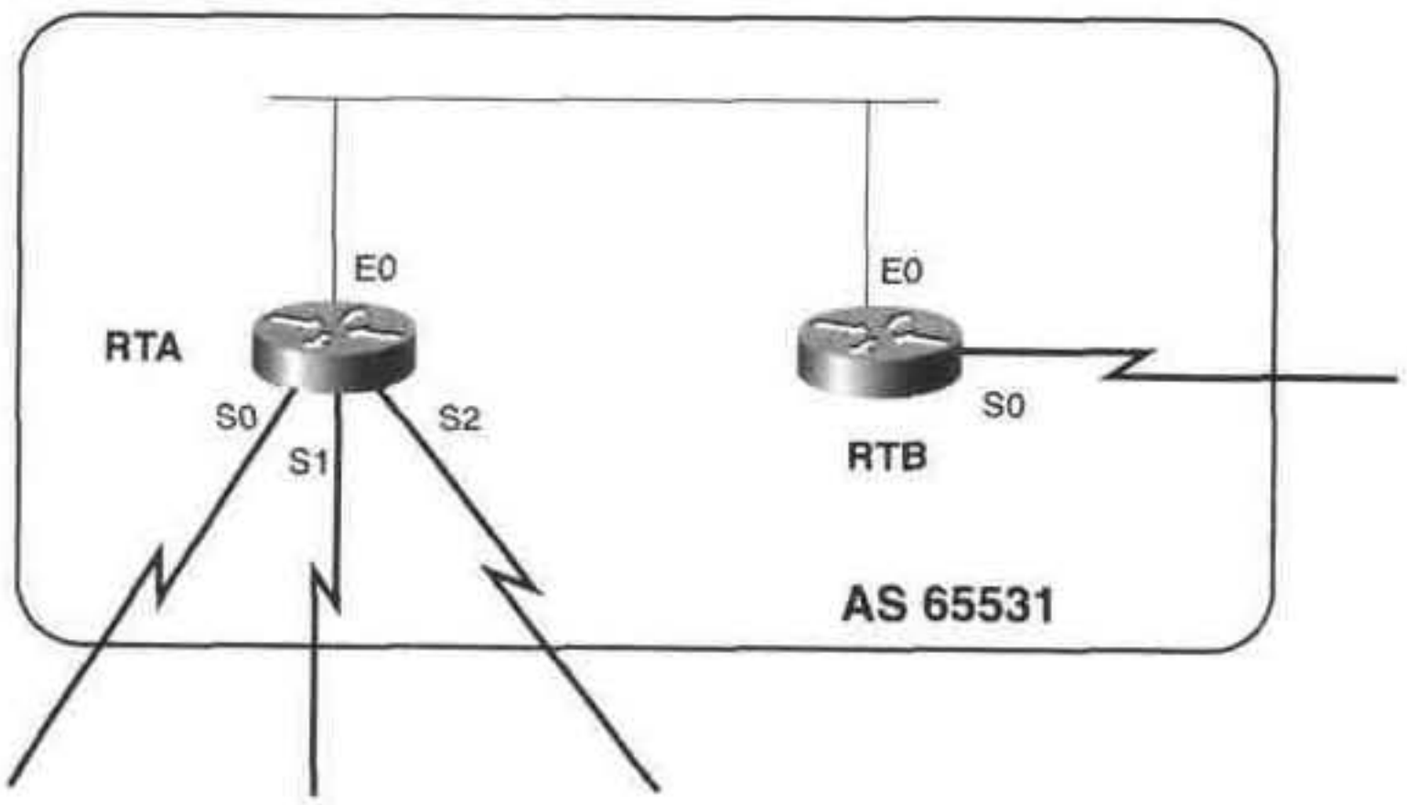
7．什么是间接的，或者说是第三方邻居？

8．EGP 如何使用它的度量来计算到一个目的地的最佳路径？

1.9　配置练习

在附录 E“配置练习的答案”中可以找到配置练习的答案。

1．图 1-14 中 AS 65531 是一个核心 AS。



RTA 接口	地址
E0	192.168.1.1/24
S0	192.168.2.1/24
S1	192.168.3.1/24
S2	192.168.4.1/24

RTB 接口	地址
E0	192.168.1.2/24
S0	192.168.5.1/24

图 1-14　配置练习 1 的网络

根据下面的条件，在 RTA 和 RTB 上配置 EGP：

(1) AS 的内部链路不能公布给任何一个外部邻居。

(2) RTA 将与它的 S1 接口相连的网络公布给 RTB；除此以外，在 RTA 和 RTB 之间没有公布其他 AS 之间的链路。

(3) 除了从其他 AS 网络学习以外，RTA 和 RTB 向它们的外部邻居公布了一条缺省路由。没有网关向内部的邻居公布缺省路由。

2. 例 1-26 给出了图 1-15 中 RTC 的路由表。

例 1-26　图 1-15 中 RTC 的路由表

```
RTC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set

I    192.168.105.0 [100/8976] via 192.168.6.2, 00:01:00, Serial1
I    192.168.110.0 [100/8976] via 192.168.6.2, 00:01:00, Serial1
I    192.168.100.0 [100/8976] via 192.168.10.2, 00:01:00, Serial2
I    192.168.120.0 [100/8976] via 192.168.10.2, 00:01:01, Serial2
C    192.168.2.0 is directly connected, Serial0
C    192.168.6.0 is directly connected, Serial1
C    192.168.10.0 is directly connected, Serial2
RTC#
```

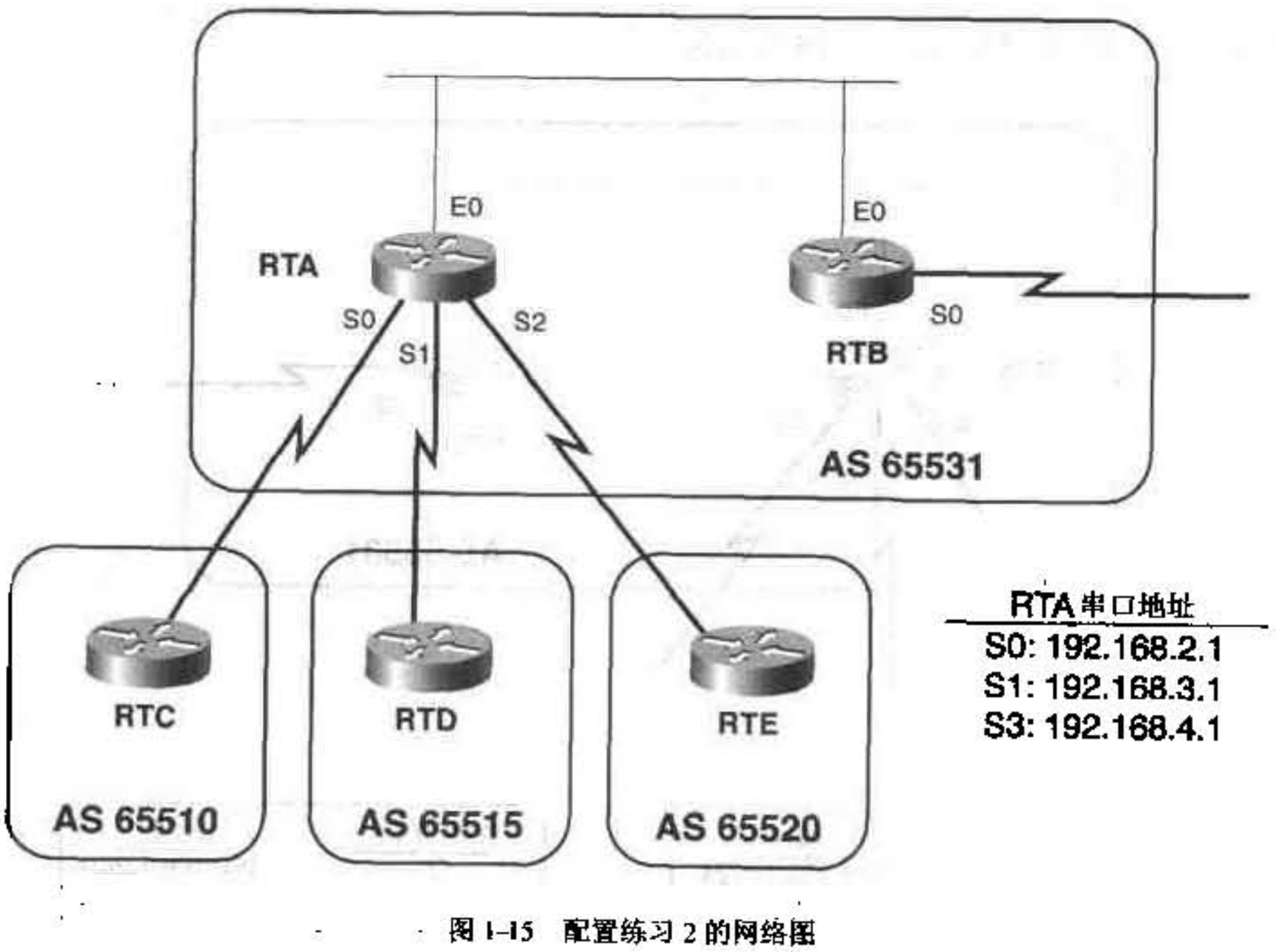


图 1-15　配置练习 2 的网络图

通过再分发，配置 RTC，让它将所有学习到的 EGP 网络公布给 AS 65510，将除了 192.168.105.0 之外的所有内部网络公布给核心 AS。保证 AS 65510 内部的网络不会通过 EGP

公布回来从而防止路由环路。这个配置的进程 ID 与本地 AS 号相同。

3. 例 1-27 给出了图 1-15 中 RTD 的路由表。

例 1-27　图 1-15 中 RTD 的路由表

```
RTD#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

C    192.168.3.0 is directly connected, Serial0
C    192.168.7.0 is directly connected, Serial1
R    192.168.230.0 [120/1] via 192.168.7.2, 00:00:14, Serial1
R    192.168.200.0 [120/2] via 192.168.7.2, 00:00:15, Serial1
R    192.168.220.0 [120/1] via 192.168.7.2, 00:00:15, Serial1
R    192.168.210.0 [120/2] via 192.168.7.2, 00:00:15, Serial1
RTD#
```

用下面的参数来配置 RTD：

只将 192.168.220.0 和 192.168.230.0 公布给 AS 65531。

没有路由协议再分发给 EGP。

将 EGP 再分发给 AS 65515 的 IGP。

将 192.168.3.0 公布给 AS 65515，它的度量是 1。

将来自 RTC 的 192.168.100.0 公布给 AS 65515，度量是 1。

将来自 RTC 的 192.168.120.0 公布给 AS 65515，度量是 3。

所有其他的路由公布给 AS 65515，度量是 5。

4. 例 1-28 给出了图 1-15 中 RTE 的路由表。

例 1-28　图 1-15 中 RTE 的路由表

```
RTE#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR


Gateway of last resort is not set

O    192.168.125.0/28 [110/74] via 192.168.130.6, 00:01:03, Serial1
C    192.168.4.0/24 is directly connected, Serial0
     192.168.225.0/28 is subnetted, 1 subnets
O E2    192.168.225.160 [110/50] via 192.168.130.18, 00:01:04, Ethernet0
     192.168.215.0/24 is variably subnetted, 3 subnets, 3 masks
O       192.168.215.161/32 [110/65] via 192.168.130.6, 00:01:04, Serial1
O E2    192.168.215.192/26 [110/50] via 192.168.130.18, 00:01:04, Ethernet0
O E1    192.168.215.96/28 [110/164] via 192.168.130.6, 00:01:04, Serial1
     192.168.130.0/24 is variably subnetted, 7 subnets, 4 masks
```

```
D       192.168.131.192/27 [90/2195456] via 192.168.130.6, 00:16:49, Serial1
D       192.168.131.96/27 [90/409600] via 192.168.130.18, 00:16:49, Ethernet0
O       192.168.131.97/32 [110/11] via 192.168.130.18, 00:01:05, Ethernet0
D       192.168.131.64/27 [90/409600] via 192.168.130.18, 00:15:01, Ethernet0
D       192.168.131.8/30 [90/2195456] via 192.168.130.6, 00:16:49, Serial1
C       192.168.131.4/30 is directly connected, Serial1
C       192.168.131.16/28 is directly connected, Ethernet0
RTE#
```

用下面的参数配置 RTE:

没有 IGP 再分发给 EGP。

EGP 不会再分发给任何的 IGP。

将 AS 65520 内部所有的网络都公布给 AS 65531。

AS 65520 内部所有的路由器都能够向由 RTA 公布的网络转发数据包。

所有进程 ID 都与 AS 号相同。

所有 OSPF 的接口都在区域 0 内。

5. 在图 1-16 中，在前一个练习的网络中加入了 AS 65525。RTF 以太网接口的 IP 地址为 192.168.1.3/24。

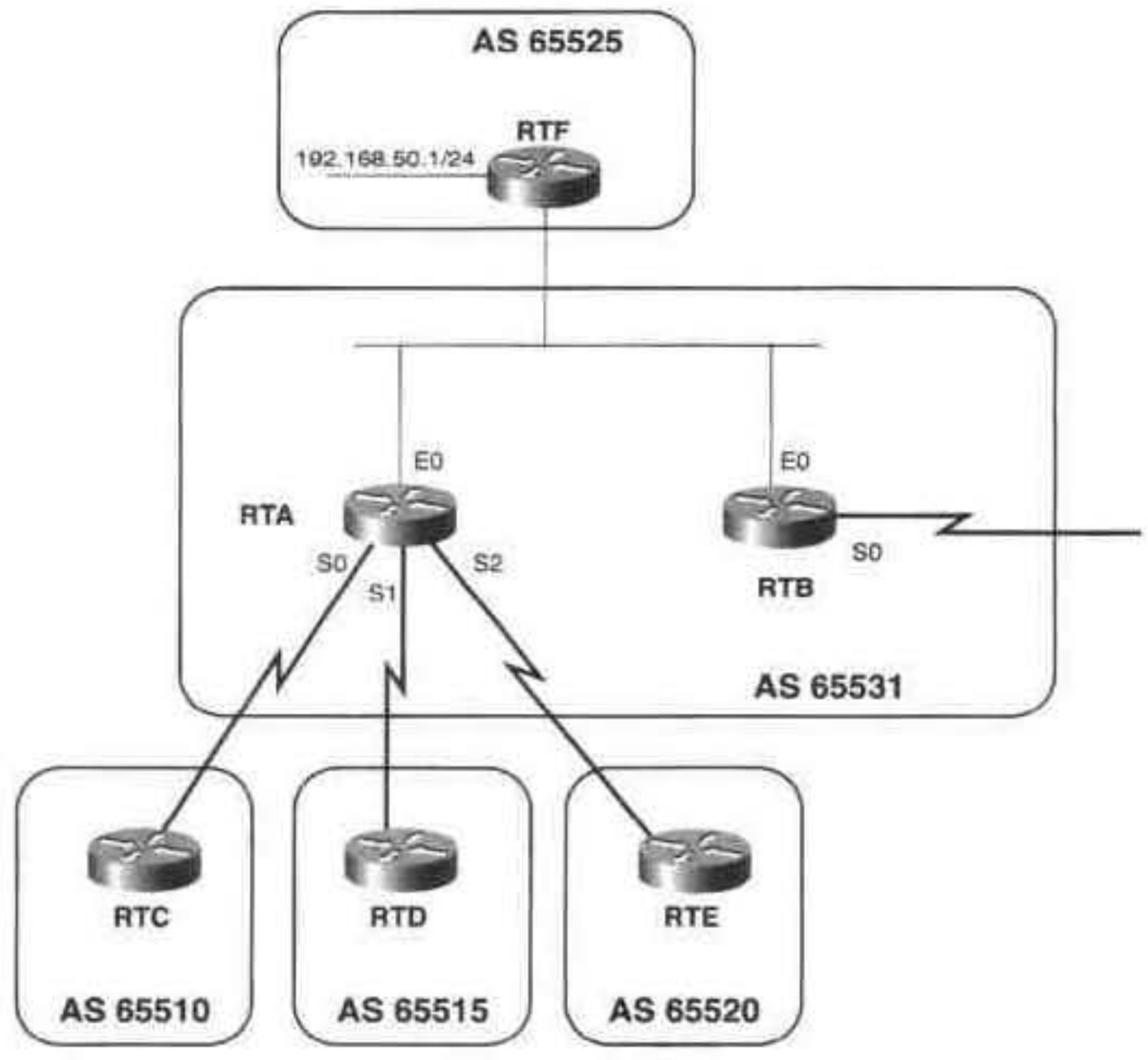


图 1-16　配置练习 5 的网络

配置新加入的路由器，让它只是 RTB 的对端，同时对配置进行必要的修改从而使它支持第三方邻居。

1.10　故障排除练习

在附录 F“故障排除练习的答案”中可以找到该故障排除练习的答案。

如图 1-17 所示，在网络中加入了路由器 RTG。

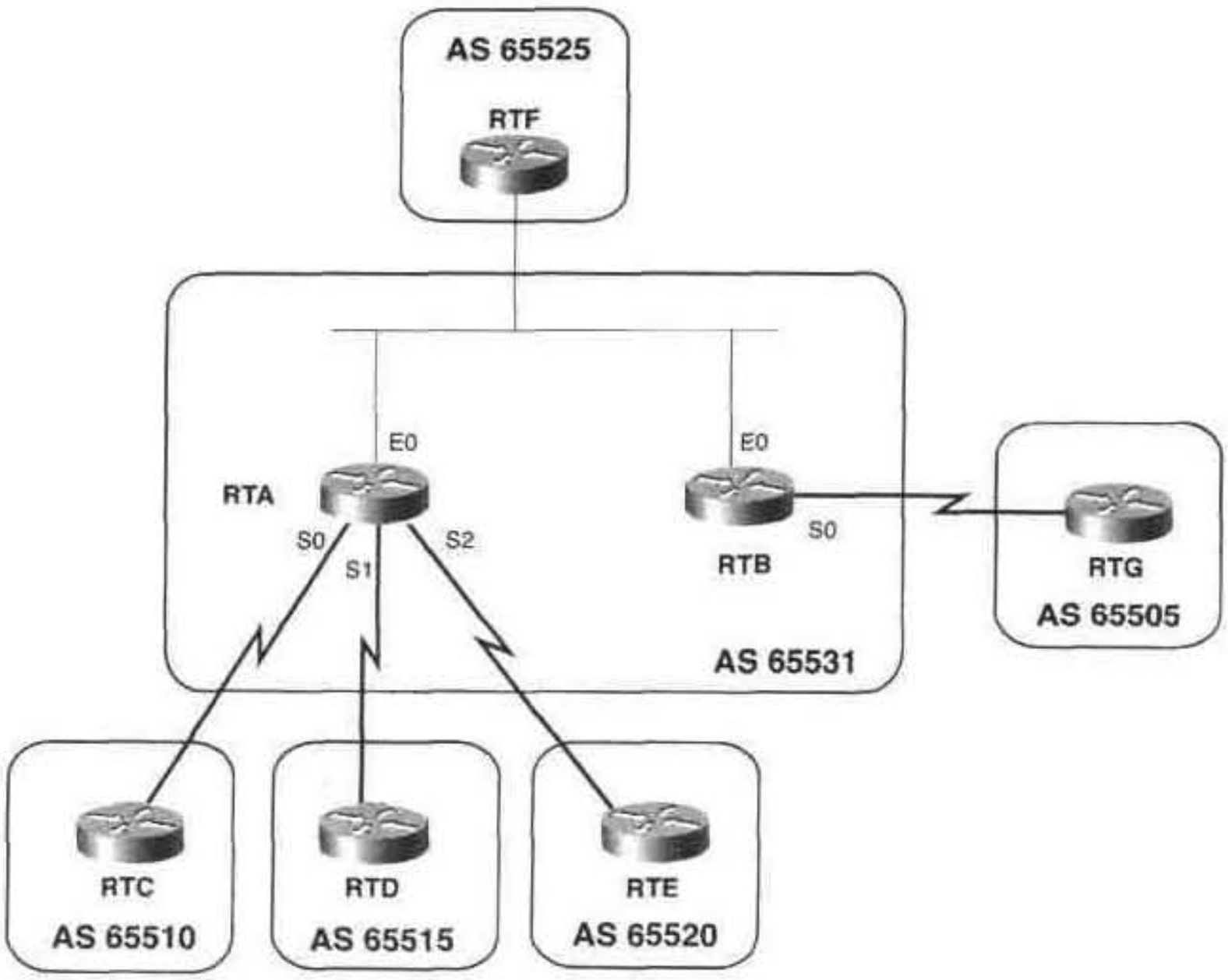


图 1-17　故障排除练习 I 的网络图

虽然它和 RTB 对等并且相互交换可到达信息，但是这里存在着一个配置错误。根据例 1-29 给出来的信息，指出错误是什么。

例 1-29　图 1-17 中 RTB 和 RTG 的 EGP 表

```
RTB#show ip egp
Local autonomous system is 65531

 EGP Neighbor       FAS/LAS  State     SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.1.1      65531/65531 UP      4      2      6     60    180   2 Perm, Pass
*192.168.1.3      65525/65531 UP      4      2    492     60    180   2 Perm, Pass
*192.168.5.2      65505/65531 UP      3      2     33     60    180   3 Temp, Pass
 EGP Neighbor      Third Party
*192.168.1.1       192.168.1.3(e)
*192.168.1.3       192.168.1.1
RTB#
```

```
RTG#show ip egp
Local autonomous system is 65505

 EGP Neighbor       FAS/LAS  State     SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.5.1      65505/65505 UP      9      36      3     60    180   4 Perm, Act
RTG#
```

第 2 章　BGP4 简介

本章所涉及的关键问题如下：

- **无类域间路由**——本节介绍 CIDR 并讨论它的优缺点。
- **谁需要 BGP**——本节考察几种 AS 域间的情况，着眼于 BGP 在哪些地方是必需的以及在哪些地方是不需要的。
- **BGP 的基本知识**——本节讨论 BGP 的基本知识，包括消息类型以及通路属性。
- **IBGP 和 IGP 同步**——本节提出了在一个 AS 内 IBGP 与 IGP 之间的同步问题，说明为什么在缺省状态要求同步以及如何避免同步问题。
- **管理大型 BGP 对等关系**——本节给出了控制大型 BGP 实施的四种工具。
- **BGP 消息格式**——本节讨论了不同 BGP 消息的细节问题。

对于任何一个 CCIE 来讲，BGP 都是一个非常重要的题目。在 CCIE 实验室，你会发现这对你的知识是一种彻底的挑战。

你已经学习了第 1 章“外部网关协议”。从 20 世纪 80 年代早期开始，ARPANET 的构建者们便开始认识到自治系统以及 AS 域间可到达协议对于维持快速增长的 Internet 的可管理性是十分必要的。最初的解决方案(EGP)对于骨干 ARPANET 来讲已经足够了。但是，从一开始，这些构建者们就认识到了转移到网状连接的 AS 域间拓扑的必要性。他们更进一步地认识到因为 EGP 不能检测到环路，因此在这种环境下，它不能高效地选路，它的聚合时间很长并且缺乏支持路由策略的工具。

人们曾经试着改进 EGP，但是最后提出了一个全新的 AS 域间协议，一个真正的路由协议。它不仅是可到达性的协议(例如 EGP)。AS 域间路由协议最初是在 1989 年的 RFC 1105[1] 中提出来的，它就是 BGP。BGP 的第一版在一年后的 RFC1163[2] 中得到了更新。在 1991 年的 EFC1267[3] 中 BGP 再一次被升级。人们习惯性地把这 3 个版本分别称为 BGP-1、BGP-2 以及 BGP-3。

BGP 现在的版本——BGP-4 出现在 1995 年的 RFC1771[4] 中。BGP-4 与以前的版本有很大的差别。最重要的区别就是 BGP-4 是无类的，而早期的版本都是有类的。做这种根本上的改变的动机涉及到外部网关协议存在的根本原因：为了保证 Internet 上路由的可管理性以及可靠性。无类域间路由(CIDR)(最早在 1993 年的 RFC 15175 中提出，作为标准的建议在同年的 RFC 15196 中完成，并在 RFC 1520[7] 中再次得到修改)就是因为以上的目的而产生的，而 BGP-4 的出现是为了支持 CIDR。

2.1　无类域间路由

自治系统以及外部网关协议的发明解决了 20 世纪 80 年代 Internet 的扩展问题。但是，在 20 世纪 90 年代早期，Internet 又出现了一种不同的扩展问题，包括以下几个方面：

• Internet 路由表爆炸性增长。成指数增长的路由表对于当时的路由器以及管理它们的人们来讲正在日益变得无法管理。仅路由表的规模就已经成了 Internet 资源的巨大负担，但是拓扑逐日的变化以及不稳定性更加重了这种负担。

• B 类地址的耗尽。在 1993 年 1 月，16382 个可用的 B 类地址已经分配出去了 7133 个。按 1993 年的增长速率，整个 B 类地址空间在 2 年的时间内就要完全用尽(摘自 RFC1519)。

• 整个 32 比特 IP 地址空间的最后枯竭。

无类域间路由为前面的两个问题提供了一个短期的解决方案，另一个短期的解决方案是网络地址翻译(NAT)，该方案将在第 4 章“网络地址翻译”中进行讨论。这些解决方案的目的就是为 Internet 的设计者赢得足够的时间去创造一个有足够地址空间的新的 IP 版本以供将来使用。开始称作下一代 IP(IPng)，最终导致有 128 位比特地址结构的 IPv6 的产生。将在第 8 章“IPv6”中进行讨论的 IPv6 是对第 3 个问题的一个长期的解决方案。有意思的是，CIDR 和 NAT 的成功使得现在在转移到 IPv6 这个事情上，几乎没有人再有当初的那份急切的心情了。

CIDR 只是一种政策批准的地址简化方案，它发挥了 Internet 分层机构的优势。因此在进一步讨论 CIDR 之前，最好先回顾一下地址简化和无类路由的知识，并看一下现代 Internet。

2.1.1　归纳摘要

归纳或者路由聚合(在《TCP/IP 路由技术(第 1 卷)》中有深入的讨论)是一种实践，也就是用一个单一的、较少细节的地址来公布一组相近的地址。基本上，归纳/路由聚合是通过减少子网掩码的长度来进行的，此时的掩码长度只是被归纳的所有地址的公共部分。例如在图 2-1 中，

图 2-1　路由聚合

将 4 个子网(172.16.100.192/28、172.16.100.208/28、172.16.100.224/28、172.16.100.240/28)归纳成一个单一的聚合地址 172.16.100.192/26。

当那些认为聚合是一个困难课题的网络工作者知道他们每天都在使用聚合的时候会非常惊讶。毕竟，什么是子网地址？它们不同与一组连续主机地址的归纳地址吗？例如，子网地址 192.168.5.255/27 是主机地址 192.168.5.224/32 到主机地址 192.168.5.255/32 的聚合(当然"主机地址" 192.168.5.224/32 是它自己数据链路的地址)。归纳地址的关键特性就是它的掩码比它归纳的地址的掩码短。最终的归纳地址是缺省的地址 0.0.0.0/0，通常写作 0/0。/0 表示掩码已经缩小到不存在网络比特——那么该地址就是所有 IP 地址的集合。

归纳也可以跨越类别界限。例如，可以将 4 个 C 类网络(192.168.0.0、192.168.1.0、192.168.2.0 以及 192.168.3.0)归纳成聚合地址 192.168.0.0/22。注意，该聚合地址有 32 比特的掩码，已经不再是合法的 C 类地址了。因此，为了支持主要类别网络地址的聚合，路由环境必须是无类别的。

2.1.2　无类路由

无类路由有两个特点：

- 无类别是路由协议的一个特性。
- 无类别是路由器的一个特性。

无类路由协议携带的是对每一个公布地址的网络部分的描述，并将之作为路由信息的一部分。网络地址的网络部分通常被称做*地址前缀*。一个地址前缀可以描述成包括一个地址掩码、一个描述地址中前缀比特数的长度字段或者只包括前缀比特的更新消息(见图 2-2)。无类 IP 路由协议有 RIP-2、EIGRP、OSPF、 IS-IS 以及 BGP-4。

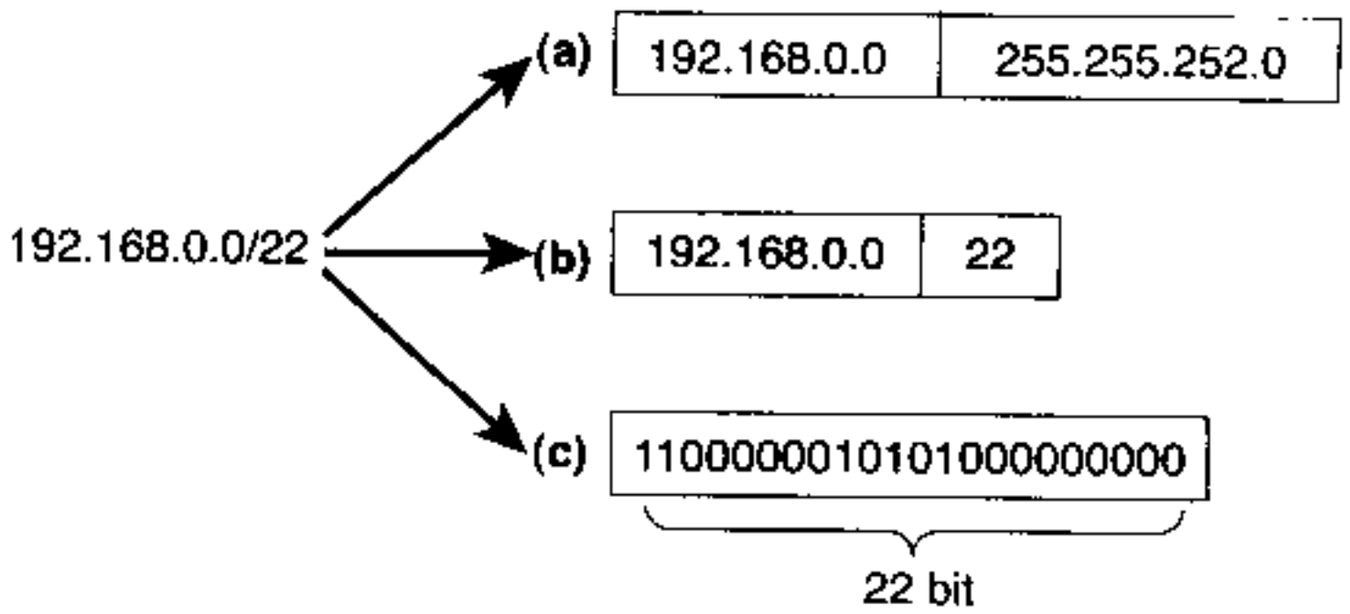


图 2-2　无类别路由协议中要宣布地址前缀长度

一个有类别的路由器以主要类别网络以及这些网络子网的方式来记录目的地址。当它执行一个路由查找时，它会先查找主要类别网络地址，然后再在该主要地址下的子网列表中寻找相匹配的地址。一个无类别路由器会忽略地址类别，而只是试图寻找"最长匹配"的地址。也就是说，对于任何给出的目的地址，它会选择与该地址最多比特相匹配的路由。例如，在例 2-1 的路由表中，给出了几个各种长度子网掩码的 IP 网络。如果路由器是无类别的，它就会为每一个目的地址寻找最长匹配的路由。

如果路由器收到了一个目的地址为 192.168.1.75 的数据包，在路由表中有如下几个条目与该地址匹配：192.168.0.0/16，192.168.1.0/24，192.168.1.0/25 以及 192.168.1.64/26。此时就会选择 192.168.1.64/26(见例 2-2)，因为它有 26 比特与目的地址相匹配——这就是最长匹配。

例 2-1　路由表包含几个有各种长度子网掩码的 IP 网络

```
Cleveland#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is 192.168.2.130 to network 0.0.0.0

O E2 192.168.125.0 [110/20] via 192.168.2.2, 00:11:19, Ethernet0
O    192.168.75.0 [110/74] via 192.168.2.130, 00:11:19, Serial0
O E2 192.168.8.0 [110/40] via 192.168.2.18, 00:11:19, Ethernet1
     192.168.1.0 is variably subnetted, 3 subnets, 3 masks
O E1    192.168.1.64 255.255.255.192
           [110/139] via 192.168.2.134, 00:11:20, Serial1
O E1    192.168.1.0 255.255.255.128
           [110/139] via 192.168.2.134, 00:00:34, Serial1
O E2    192.168.1.0 255.255.255.0
           [110/20] via 192.168.2.2, 00:11:20, Ethernet0
     192.168.2.0 is variably subnetted, 4 subnets, 2 masks
C       192.168.2.0 255.255.255.240 is directly connected, Ethernet0
C       192.168.2.16 255.255.255.240 is directly connected, Ethernet1
C       192.168.2.128 255.255.255.252 is directly connected, Serial0
C       192.168.2.132 255.255.255.252 is directly connected, Serial1
O E2 192.168.225.0 [110/20] via 192.168.2.2, 00:11:20, Ethernet0
O E2 192.168.230.0 [110/20] via 192.168.2.2, 00:11:21, Ethernet0
O E2 192.168.198.0 [110/20] via 192.168.2.2, 00:11:21, Ethernet0
O E2 192.168.215.0 [110/20] via 192.168.2.2, 00:11:21, Ethernet0
O E2 192.168.129.0 [110/20] via 192.168.2.2, 00:11:21, Ethernet0
O E2 192.168.131.0 [110/20] via 192.168.2.2, 00:11:21, Ethernet0
O E2 192.168.135.0 [110/20] via 192.168.2.2, 00:11:21, Ethernet0
O*E2 0.0.0.0 0.0.0.0 [110/1] via 192.168.2.130, 00:11:21, Serial0
O E2 192.168.0.0 255.255.0.0 [110/40] via 192.168.2.18, 00:11:22, Ethernet1
Cleveland#
```

例 2-2　目的地址是 192.168.1.75 的数据包被转发到端口 S1

```
Cleveland#show ip route 192.168.1.75
Routing entry for 192.168.1.64 255.255.255.192
  Known via "ospf 1", distance 110, metric 139, type extern 1
  Redistributing via ospf 1
  Last update from 192.168.2.134 on Serial1, 06:46:52 ago
  Routing Descriptor Blocks:
  * 192.168.2.134, from 192.168.7.1, 06:46:52 ago, via Serial1
      Route metric is 139, traffic share count is 1
```

而目的地址为 192.168.1.217 的数据包既不与 192.168.1.64/26 匹配，也不与 192168.1.0/25 匹配。这个地址的最长匹配是 192168.1.0/24，见例 2-3。

例 2-3　路由器将 192.168.1.217 与网络地址 192.168.1.0/24 相匹配

```
Cleveland#show ip route 192.168.1.217
Routing entry for 192.168.1.0 255.255.255.0
  Known via "ospf 1", distance 110, metric 20, type extern 2, forward metric 10
  Redistributing via ospf 1
  Last update from 192.168.2.2 on Ethernet0, 06:48:18 ago
  Routing Descriptor Blocks:
  * 192.168.2.2, from 10.2.1.1, 06:48:18 ago, via Ethernet0
      Route metric is 20, traffic share count is 1
```

如例 2-4 所述，目的地是 192.168.5.3 的数据包和一个更具体子网或者网络不匹配，它的最长匹配是聚合地址 192.168.0.0/16。

例 2-4　目的地是 192.168.5.3 的数据包和超网 192.168.0.0/16 相匹配

```
Cleveland#show ip route 192.168.5.3
Routing entry for 192.168.0.0 255.255.0.0, supernet
  Known via "ospf 1", distance 110, metric 139, type extern 1
  Redistributing via ospf 1
  Last update from 192.168.2.18 on Ethernet1, 06:49:26 ago
  Routing Descriptor Blocks:
  * 192.168.2.18, from 192.168.7.1, 06:49:26 ago, via Ethernet1
      Route metric is 139, traffic share count is 1
```

在例 2-5 中，目的地址 192.169.1.1 与路由表中的任何网络条目都不匹配。但是因为在例 2-1 中包含缺省路由，因此带有该目的地址的数据包不会被丢弃。这些数据包被送到下一跳路由器 192.168.2.130。

例 2-5　在路由表中找不到 192.169.1.1 的匹配条目。将目的地是 192.169.1.1 的数据包转发给缺省地址，出端口 S0

```
Cleveland#show ip route 192.169.1.1
% Network not in table
```

从 IOS 11.3 开始，Cisco 的路由器缺省都是无类别的。在这个版本之前，IOS 缺省都是有类别的。可以通过 **ip classless** 命令来更改缺省设置。

例 2-1 的路由表以及相关的例子说明了最长匹配路由的另一个特性。换句话说，一条到一个聚合地址的路由没有必要指向聚合地址中的每一个成员。图 2-3 给出了例 2-2 到例 2-5 中路由的向量。

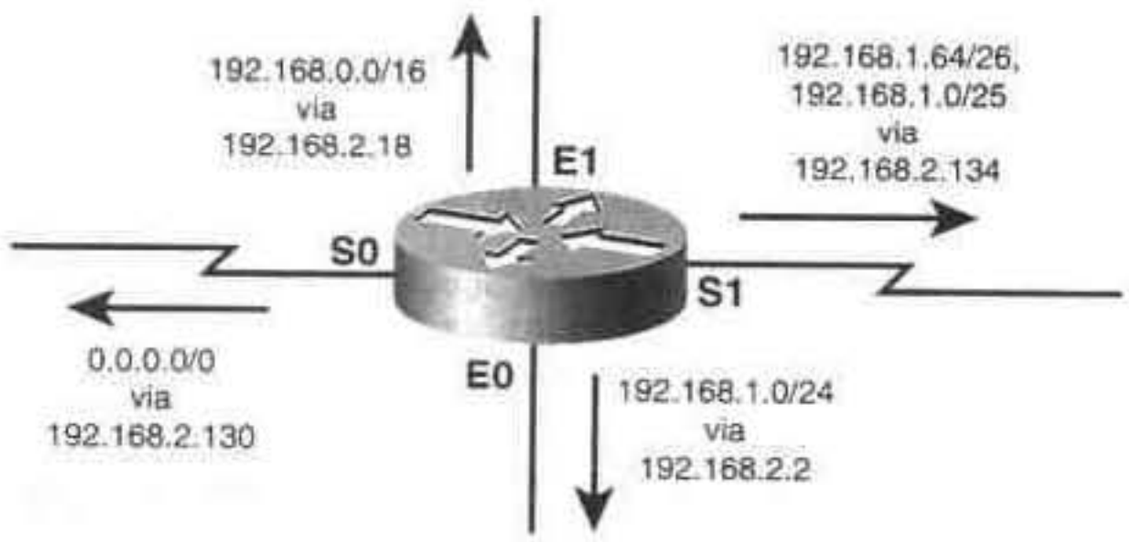


图 2-3　例 2-1 路由表中路由的向量

可以把网络 192.168.1.0/24 看作它所有子网的聚合；图 2-3 显示了到这个网络地址的路由直接把数据包发送到了接口 E0。但是，到它的两个子网 192.168.1.0/25 和 192.168.1.64/26 的路由，指向一个不同的接口——S1。

> 注：　实际上，192.168.1.64/26 本身是 192.168.1.0/25 的一个成员。这两个地址有两条截然不同的路由都指向端口 S1 的事实，就暗示这两个网络是由上游某处的不同的路由器公布的。

同样，192.168.1.0/24 是聚合地址 192.168.0.0/16 的一个成员，但是到这个较短掩码长度的路由指向 E1。掩码长度最短的地址 0.0.0.0/0 是所有其他地址的聚合，到它的路由是指向 S0。因为是最长匹配路由，因此到子网 192.168.1.64/26 和 192.168.1.0/25 的数据包被发送到端口 S1，同时到 192.168.1.0/24 其他子网的数据包被发送到端口 E0。目的地址是以 192.168 开始的数据包，除了 192.168.1 以外，都被发送到端口 E1，而目的地址不是以 192.168 开始的数据包被送到了端口 S0。

2.1.3　路由总结：优势、劣势以及不对称性

从存储路由表所需要内存的数量到传输与处理路由信息所必需的网络带宽和路由器传送与处理路由信息能力方面来讲，路由总结在节省网络资源方面是一个非常重要的工具。路由总结还通过“隐藏”网络的不稳定性而保护了网络资源。

例如，图 2-4 中的网络有一条不稳定的路由——由于物理连接或路由器接口较差，使得传输中断、恢复然后又中断。

如果没有路由总结，子网 192.168.1.176/28 的每一次中断或者恢复，都必须告知企业网内的每一个路由器。而每一个路由器必须处理这些变化消息并相应地调整它们的路由表。如果路由器 Nashville 用聚合地址 192.168.1.128/25 公布所有的上行路由，对任何一个更具体子网，它们的变化信息并没有通过该路由器进行公布。Nashville 是聚合点；即使它的一些成员不稳定但是该聚合点保持着稳定。

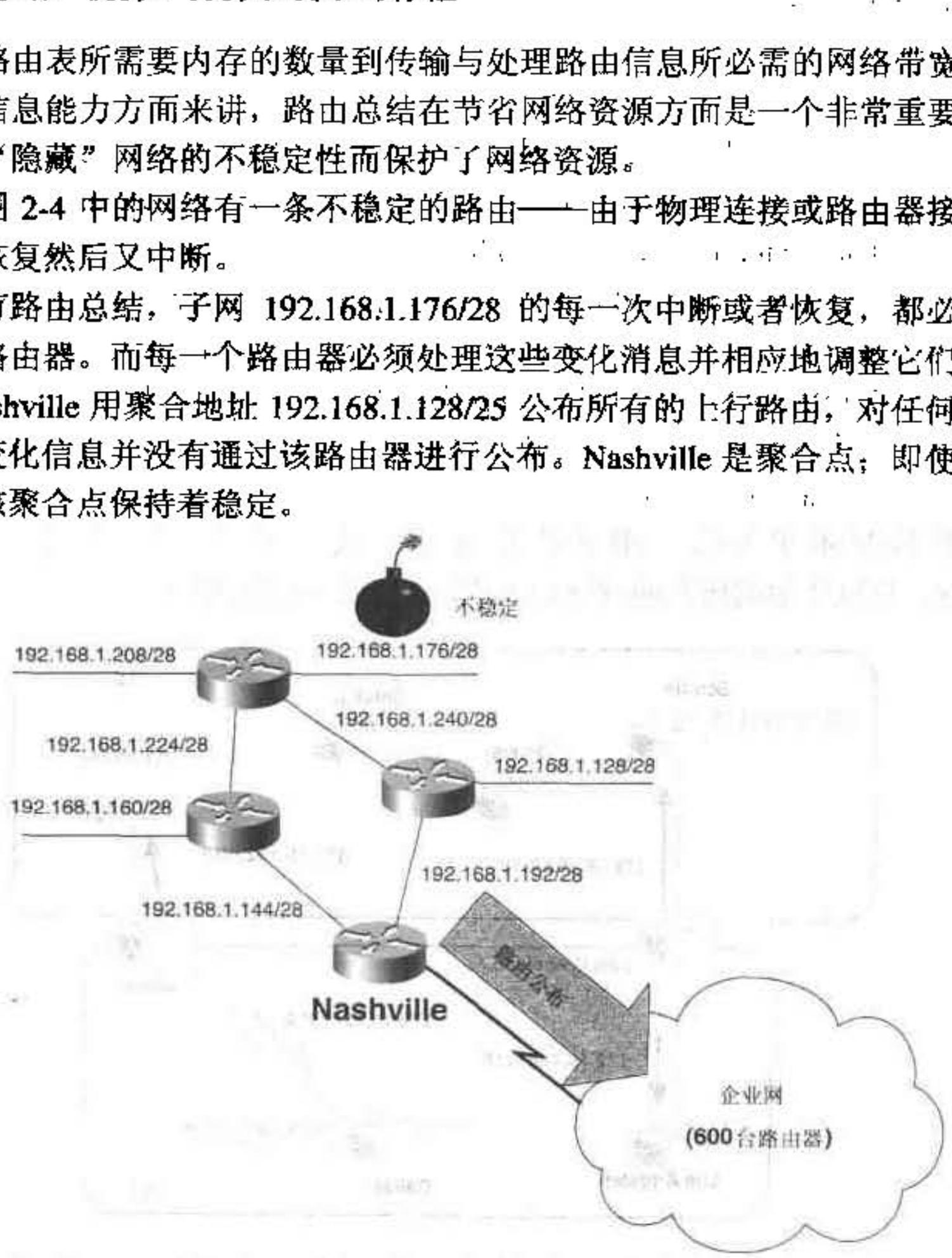


图 2-4　一条不稳定的路由可能会使整个网络不稳定

路由总结的代价是降低了路由的精确性。在例 2-6 中，图 2-3 中路由器的接口 S1 出现故障，导致从该接口的邻居上学习到的路由是无效的。通常转发到接口 S1 的数据包并没有被丢弃，目的地址为 192.168.1.75 的数据包现在与下一条最佳路由 192.168.1.0/24 相匹配，并且这些数据包被送到接口 E0(将该情况与例 2-2 相比较)。

例 2-6　一条故障路由可能会导致错误的数据包转发

```
Cleveland#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1, changed state to down
%LINK-3-UPDOWN: Interface Serial1, changed state to down
Cleveland#show ip route 192.168.1.75
Routing entry for 192.168.1.0 255.255.255.0
  Known via "ospf 1", distance 110, metric 20, type extern 2, forward metric 10
  Redistributing via ospf 1
  Last update from 192.168.2.2 on Ethernet0, 00:00:20 ago
  Routing Descriptor Blocks:
  * 192.168.2.2, from 10.2.1.1, 00:00:20 ago, via Ethernet0
      Route metric is 20, traffic share count is 1

Cleveland#
```

这种不精确性可能是也可能不是一个问题，这要看该网络其他部分的情况。让我们继续上面的例子，假设由于网络还没有聚合或者路由是静态输入的，下一跳路由器 192.168.2.2 还有一条通过路由器 Cleveland 到 192.168.1.64/26 的路由条目。在这种情况下，会出现路由环路。另一方面，一个通过 Cleveland 的 E0 接口可到达的路由器可能会有到子网 192.168.1.64/26 的“后门”路由，该路由只有在通过 Cleveland 的端口 S1 的首选路由失效时才使用。在第二个实例中，将到 192.168.1.0/24 的路由指定为备用路由，并且例 2-6 中的行为是有意安排的。

图 2-5 给出了一个网络，该网络因为缺少路由精确性而导致了另一种问题，如该图中所示，路由域 1 通过在 San Francisco 和 Atlanta 的路由器与路由域 2 相连。对于该例子来讲，如何定义这些域是不重要的。重要的是可以将域 1 中的所有的网络归纳成地址 172.16.192.0/18，将域 2 中的所有的网络归纳成地址 172.16.128.0/18。

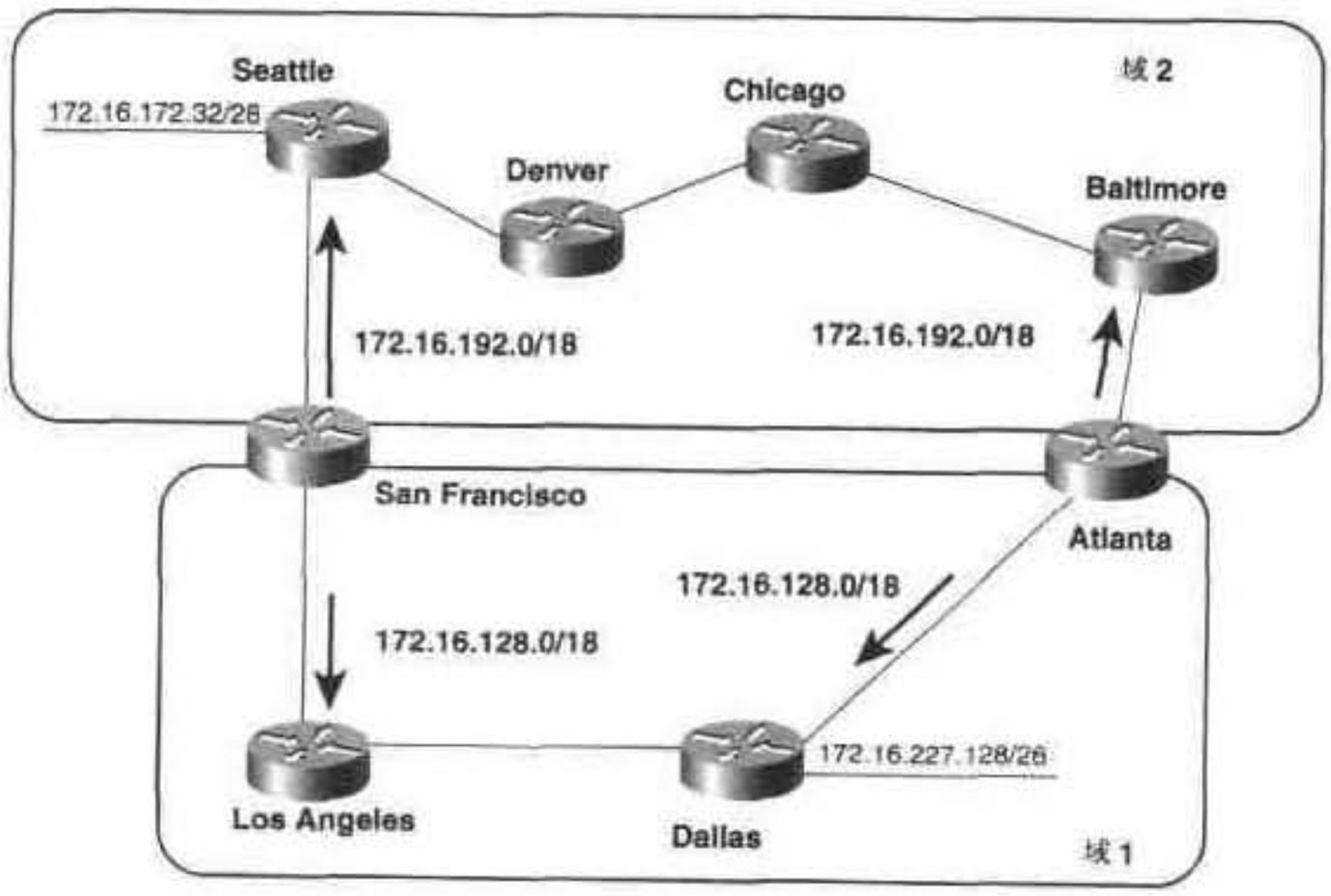


图 2-5　当多个路由器都公布同一个聚合地址时，缺少路由精确性将引发一定的问题

除了公布单个子网，Atlanta 和 San Francisco 还向这两个域公布归纳地址。如果在 Dallas 子网 172.16.227.128/26 上的一台主机发送一个数据包到 Seattle 子网 172.16.172.32/28 上的一台主机，数据包最有可能的就是选路到 Atlanta，因为 Atlanta 是公布域 2 归纳路由的最近的路由器。Atlanta 将数据包转发到域 2，并到达 Seattle。当子网 172.16.172.32/28 上的主机发送了一个应答消息时，Seattle 将该数据包转发到 San Francisco——公布归纳路由 172.16.192.0/18 的最近的路由器。

这里存在的问题是两个子网之间的业务量是不对称的：从 172.16.227.128/26 到 172.16.172.32/28 的数据包使用一条通路，从 172.16.172.32/28 到 172.16.227.128/26 的数据包使用另外一条不同的通路。不对称性的出现是因为 Dallas 和 Seattle 路由器没有到对方的完整的路由。它们只有到达宣告归纳地址路由器的路由，因此它们必须在这些路由的基础上转发数据包。换句话来讲，在 San Francisco 和 Atlanta 的归纳地址中隐藏了这些路由器后面网络的细节情况。

不对称的业务量是我们所不希望的，原因有几个。第一，网络业务量类型变得不可预测，使得估算基线、容量计划以及故障排除变得更加困难。第二，链路的利用率变得不平衡。一些链路利用率达到饱和，而另外一些链路却没有充分利用。第三，在流出的业务量和流入的业务量的时延方面会出现根本的不同。这种时延变化对一些对时延敏感的业务(例如话音以及实时视频)是有害的。

2.1.4　Internet：经过多年后还保持着分层结构

虽然 Internet 已经不再是第 1 章所描述的 ARPANET 的单一骨干结构，它仍然保持着一定的分层结构。最底层，Internet 用户与 Internet 业务供应商(ISP)相连。在许多情况中，ISP 是本地地理区域内许多小的业务供应商(称做*本地* ISP)中的一个。例如，在 Colorado 的区号为 303 的区域内现在大概有 200 个 ISP。反过来，这些本地 ISP 又是覆盖整个地理区域(例如一个州或者一些相邻州)的较大 ISP 的用户。这些较大的 ISP 被称做*区域业务供应商*。例如在 Colorado 有 CSD Internet 以及 Colorado Supernet。区域业务供应商通过覆盖全国或者全球区域的高速(DS-3 或者 OS-3 或者更高)骨干连接到大型 ISP 上。这些大的 ISP 就是*网络业务供应商*，包括 MCI/WorldCom(UUNET)、SprintNet、Cable&Wireless、Concentric Network 以及 PSINet 等公司。更为普遍地，这些不同的供应商分别被称为 Tier III、Tier II 以及 Tier I 供应商。

图 2-6 显示了这些不同类型的 ISP 是如何互连的。在每一种情况下，一个用户——终端用户或者低层的业务供应商——在ISP的呈现点(POP)与较高层的业务供应商相连。一个POP就是用户附近的一个路由器，通过拨号或者专用本地环路与该路由器相连。在最高层，网络业务供应商通过*网络接入点*(NAP)互连。一个 NAP 是一个 LAN 或者一台交换机(一般是以太网、FDDI 或者 ATM)通过它们，不同的业务供应商可以交换路由和数据业务量。

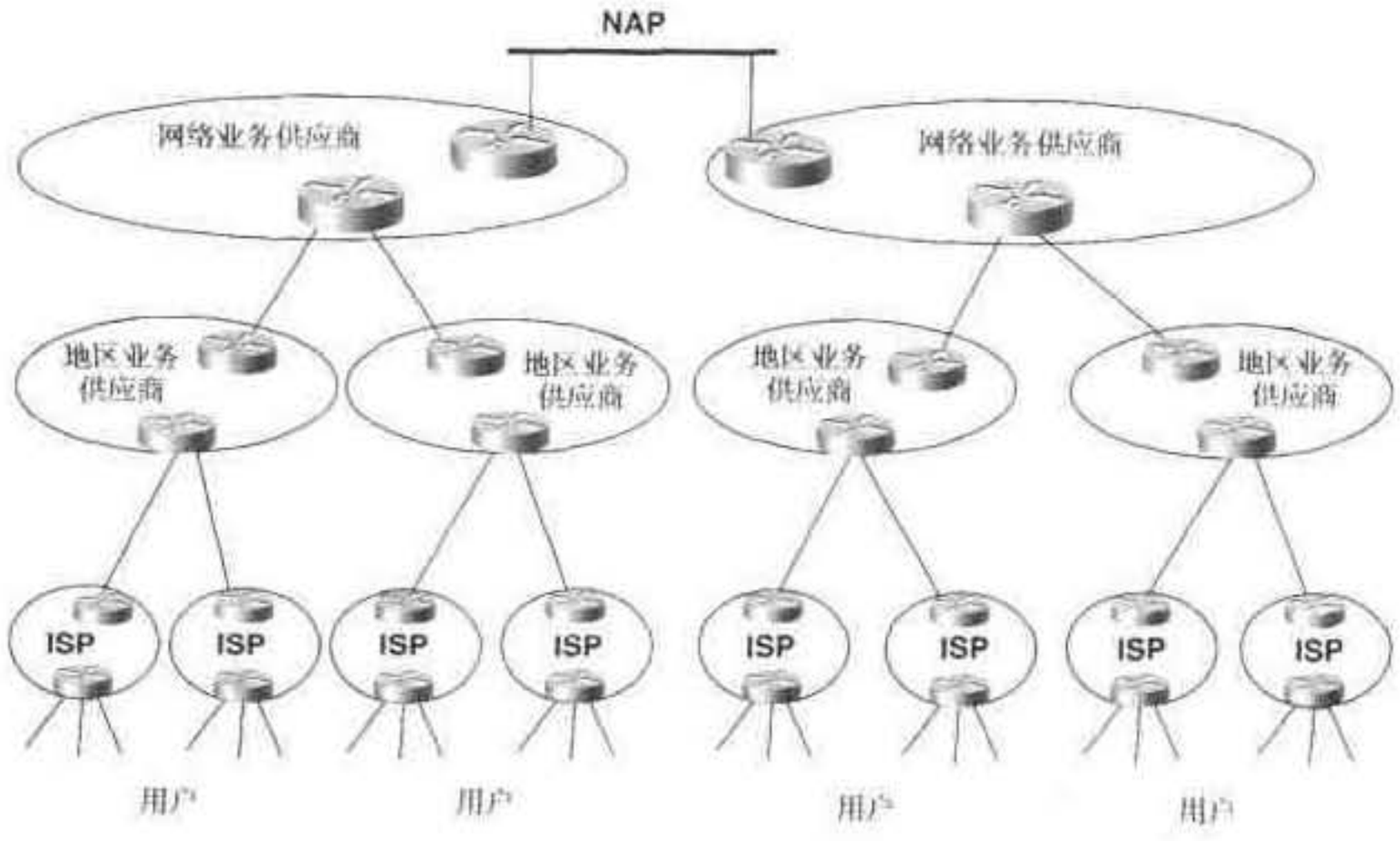


图 2-6　ISP/NAP 分层结构

如表 2-1 所示，一些 NAP 的名字是 Commercial Internet Exchange(CIX)、Federal Internet Exchange(FIX)以及 Metropolitan Area Exchange(MAE 最初被称为城域以太网，来自 Metropolitan Fiber Systerms,公司的创意)。CIX、FIX 和 MAE-East 最早具备了连接骨干网的经验；在这些连接点获得的运营经验的基础上，美国国家科学基金会在 1994 年部署了最初的 4 个 NAP，把它们作为取代 NSFnet 的一部分。

表 2-1　　美国著名的网络接入点

网络接入点	地　　点	负责维护的公司
New York NAP*	Pennasuken, New Jersy	Sprint
Chicago NAP*	Chicago. Illinois	Ameritech 和 Bellcore
San Francisco NAP*	San Francisco, California	Pacific Bell
Big East NAP	Bohemia, New York	ICS Network System
MAE-West	San Jose, California	MCI/WorldCom
MAE-East*	Washington, DC	MCI/WorldCom
MAE-LA	Los Angeles. California	MCI/WorldCom
MAE Houston	Houston, Texas	MCI/WorldCom
MAE-Dallas	Dallas. Texas	MCI/WorldCom
MAE-New York	New York City, New York	MCI/WorldCom
MAE-Chicago	Chicago, Illinois	MCI/WorldCom
FIX-East	College Park, Maryland	University of Maryland
FIX-West	Moffett Field, California	NASA Ames Research Center
CIX	Santa Clara, California	Wiltel
Digital PAIX	Palo Alto. California	Digital Equipment Corporation

*最初 4 个 NSF NAP 中的一个。

除了表 2-1 列出的主要的 NAP，在 NSP 集中的地方，会有许多小的 NAP。它们通常与较小的区域业务供应商相连。区域性 NAP 的例子有 Seattle Internet eXchange(SIX)和 New Maxico 网络接入点。

与 NAP 的构成相结合，NSF 资助了 Routing Arbiter(RA)项目。RA 的一个任务就是提高 Internet 的稳定性和可管理性。为了达到这个目的，RA 向业务供应商提出了建立路由(拓扑)和策略(首选的路径)数据库(RADB)的建议。数据库放在 NAP 处的一个*路由服务器*上，该服务器可以是一台 UNIX 工作站，也可以是一台运行 BGP 的工作站。此时每一个业务供应商的路由器在 NAP 处不再与每一台其他的路由器对等，而只需与路由服务器对等。路由和策略被送到服务器处，该服务器使用一种复杂的称为 RIPE-181 的数据库语言来处理并维护这些信

息。将合适的路由传给其他的路由器。

虽然路由服务器运行 BGP 并处理路由,但是它不执行分组转发。它的更新消息会通知路由器，最好的下一跳路由器可以通过 NAP 直接到达。根据第 1 章对 EGP 第三方邻居的讨论，你已经熟悉了这个概念。通过一对多的对等关系而不是多对多的对等关系，路由服务器提高了网络的稳定性、可管理性以及通过 NAP 的业务量流量。

NAP 和 RA 项目证明互相竞争的网络业务供应商能够互相合作，为 Internet 提供可管理的连通性和稳定性。结果是，NSF 在 1997 年 1 月 1 日停止了对路由服务器和 NAP 的资助，并把操作转到了商业利益上。虽然公众资助的 Internet 研究继续着一些项目，例如 Internet2、GigaPOP 以及甚高速骨干网络业务(vBNS)，但是现在的 Internet 已经可以看作是一种商业运作了。

商业控制 Internet 的一个结果就是现在 Internet 的拓扑已经远不及以前的拓扑整齐了。最大的业务供应商，在财政、竞争以及政策利益的驱使下，通常会选择直接建立对等关系而不是通过路由服务器建立对等关系。现在，对等关系会出现在许多的层面，而不仅仅出现在图 2-6 所示的最高层面。

当两个或者多个业务供应商同意通过一个 NAP 来共享路由时,无论是直接对等还是通过路由服务器对等，他们都要达成一个*对等协定*。一个对等协定可能会在两个业务供应商之间直接建立(一个*双边*对等协议)，也可能在一组规模相近的业务供应商之间建立(一个*多边*对等协议，或者 MLPA)。在决定协议财政特性方面，业务量类型起着主要的作用。如果对等双方在两个方向上业务量基本平衡，他们通常无须金钱往来。对等关系对于双方来讲是公平的。但是，如果经过对等点的业务量一个方向高于另外一个方向，例如当一个小型业务供应商与一个大型的业务供应商有对等关系，小型业务供应商通常要为他的对等特权付费，其原因就是小型业务供应商在对等中会比大型业务供应商获得更多的好处。

另一个使 Internet 拓扑混乱的因素是对等点的位置。集中多个业务供应商的 NAP(如表 2-1 所示)是*公共*对等点。除了这些公共点，业务供应商在他们与其他业务供应商共处的地方建立了数百个小的 NAP。在这些地点的对等协议通常就是两个或者几个业务供应商之间的一种*专用*协定。现在鼓励业务供应商使用专用协定，因为这样可以减轻国家 NAP 的拥塞程度，而且由于增加了路由的多样性，可以降低一些业务量的时延。

另一个隐含的事实告诉我们实际生活中的网络并不像图 2-6 所给出的那么整齐，为了与本地 ISP 进行直接的竞争，许多国家级和区域级的业务供应商也转让本地 Internet 入口。例如，例 2-7 中的路由跟踪的“起始点”是一个拨号 POP，它属于 Concentric Network——一个骨干业务供应商。在骨干 NAP 处也经常会出现区域业务供应商。它们可能通过 NAP 与一个或者多个网络业务供应商相连,或者避开任何网络业务供应商而通过 NAP 与其他的区域业务供应商相连。

例 2-7　在 Denver 的 Concentric Network POP 处进行的路由跟踪

```
--- traceroute to www.uswest.com (205.215.207.54),
    30 hops max, 18 byte packets

  1  (  207.155.168.5)  ts003e01.den-co.concentric.net  174 ms
  2  (  207.155.168.1)  rt001e0102.den-co.concentric.net.168.155.207.IN-ADDR.ARPA
162 ms
  3  (   207.88.24.29)  us-dc-wash-core1-a1-0d12.rtr.concentric.net  385 ms
  4  (   192.41.177.2)  maeeast2.bbnplanet.net  225 ms
```

```
 5  (      4.0.1.93)  p2-2.vienna1-nbr2.bbnplanet.net  232 ms
 6  (      4.0.3.130)  p3-1.nyc4-nbr2.bbnplanet.net  222 ms
 7  (      4.0.5.26)  p1-0.nyc4-nbr3.bbnplanet.net  223 ms
 8  (      4.0.3.121)  p2-1.chicago1-nbr1.bbnplanet.net  235 ms
 9  (      4.0.5.89)  p10-0-0.chicago1-br1.bbnplanet.net  239 ms
10  (      4.0.2.18)  h1-0.minneapol1-cr1.bbnplanet.net  258 ms
11  (   4.0.246.254)  h1-0.uswest-mn.bbnplanet.net  260 ms
12  (207.225.159.221)  207.225.159.221  249 ms
13  ( 205.215.207.54)  www.uswc.uswest.net  258 ms

--- traceroute to www.rmi.net (166.93.8.30),
    30 hops max, 18 byte packets

 1  (  207.155.168.5)  ts003e01.den-co.concentric.net  152 ms
 2  (  207.155.168.1)  rt001e0102.den-co.concentric.net.168.155.207.IN-ADDR.ARPA
161 ms
 3  (   207.88.24.21)  207.88.24.21  190 ms
 4  (   207.88.0.253)  us-ca-scl-core1-f9-0.rtr.concentric.net  189 ms
 5  (   207.88.0.178)  207.88.0.178  206 ms
 6  ( 144.228.207.73)  sl-gw18-chi-5-1-0-T3.sprintlink.net  210 ms
 7  (  144.232.0.217)  sl-bb11-chi-3-3.sprintlink.net  216 ms
 8  (  144.232.0.174)  sl-bb5-chi-4-0-0.sprintlink.net  211 ms
 9  (   144.232.8.85)  sl-bb7-pen-5-1-0.sprintlink.net  225 ms
10  (   144.232.5.53)  sl-bb10-pen-1-3.sprintlink.net  236 ms
11  (   144.232.5.62)  sl-nap1-pen-4-0-0.sprintlink.net  228 ms
12  (  192.157.69.13)  p219.t3.ans.net  263 ms
13  ( 140.223.60.209)  f1-1.t60-6.Reston.t3.ans.net  264 ms
14  (  140.223.65.17)  h12-1.t64-0.Houston.t3.ans.net  286 ms
15  (  140.223.25.14)  h13-1.t80-1.St-Louis.t3.ans.net  283 ms
16  (  140.223.25.29)  h14-1.t24-0.Chicago.t3.ans.net  292 ms
17  (   140.223.9.18)  h14-1.t96-0.Denver.t3.ans.net  309 ms
18  ( 140.222.96.122)  f1-0.c96-10.Denver.t3.ans.net  313 ms
19  (  207.25.224.14)  h1-0.enss3191.t3.ans.net  306 ms
20  (  166.93.46.246)  166.93.46.246  305 ms
21  (    166.93.8.30)  www.rmi.net  285 ms
```

例 2-7 中的路由跟踪给出了一点 Internet 的骨干结构。两个跟踪都是从 Denver 的 Concentric Network POP 处发起的。在第一个跟踪中，数据包穿过 Concentric Network 的骨干到达 MAE-East，然后它们从这里到达 BBN Planet 骨干(第 3 行和第 4 行)。数据包通过 BBN Planet 的骨干到达一个在 Minneapolis 由 BBN 和 US West 共享的 Tier II NAP(10 和 11 行)，然后到达在 US West 的目的地。

在第二次跟踪中的数据包在美国内部周游了相当长的一段时间才到达了目的地，其实目的地与源地点之间才有几公里的路程。首先，它们沿着 Concentric 的骨干通过一个在 Caifornia(第 4 行)的路由器，然后到达 Chicago 的 NAP，从这里它们再到 Sprint 的骨干(第 6 行)。数据包在 Pennsauken，New Jersey 被路由到 New York 的 NAP,然后被送到 ANS 骨干(11 和 12 行)。接着它们又经过了在 Reston、Houston、St.Louis 以及 Chicago(再一次)的路由器，最后才到达了在 Denver 的目的地。

和上一次跟踪中的数据包一样，我们经过了相当冗长并迂回的过程才又回到了我们需要讨论的题目——CIDR。

2.1.5　CIDR：减轻了路由表的爆炸性增长

给出了 Internet 的一些分层结构的知识，你就能够理解这种结构是怎样引出一个地址归

纳方案的。在高层，IANA(Internet Assigned Numbers Authority)将连续 C 类地址的大型地址块分配给全球不同的地址分配机构，就是我们所知的*区域 IP 注册处*。现在，有三个区域注册处。南北美、加勒比以及部分撒哈拉地区的区域注册处是 Internet 号码美国注册处(ARIN)。ARIN 同样也负责为全球业务供应商分配地址。欧洲、中东、北非以及亚洲的一部分地区(前苏联)的区域注册处是 RIPE。亚洲的其他地区以及太平洋地区的区域注册处是亚太网络信息中心(APNIC)。

注：1997 年，ARIN 从 InterNIC(由 Internet Solutions,Inc.经营)脱离出来，从而将 IP 地址和域名的管理分开。

表 2-2 给出了这些注册处为所服务的区域分配 C 类地址的最初方案。但是其中的一些分配方式已经过时了。如例 2-8 中所示，在例 2-7 中，当对地址 207.155.128.5 执行了 WHOIS，该地址就会以分配给 Concentric Network 的/17 CIDR 块的一部分而出现，标明“Other”的那部分地址块现在已经开始分配了。然后，这些区域注册处再把这些地址的一部分分配给大型业务供应商或者本地 IP 地址注册处。通常，在这个层面分配的地址块不会少于 32 个连续的 C 类地址(通常会大于 32 个)。例如，分配给 Concentric Network 的地址块是 207.155.128.0/17，它包括 128 个连续的 C 类地址(见例 2-8)。

业务供应商收到这些地址块以后再把它们分成更小的块，然后分配给他们的用户。如果那些用户也是 ISP，他们将把自己的地址块分成更小的块。分配这些被称做 *CIDR* 块的 C 类地址块的很明显的好处体现在把这些地址块归纳回分层结构的时候。如果需要了解在 Internet 范围内地址是如何分配的，可以参见 RFC 2050(www.isi.edu/in-notes/rfc2050.txt)。

为了说明这个问题，可以假设 Concentric Network 将它的 207.155.128.0/17 地址块的一部分分配给它的一个用户，该部分包括 207.155.144.0/20。如果那个用户是一个 ISP，它会把那个地址块的一部分，例如 207.155.148.0/22，分配它自己的一个用户。该用户会将它的/22(读做“斜杠 22”)地址块公布给它的 ISP。该 ISP 会用一个聚合的地址 207.155.144.0/20 将它所有的用户归纳给 Concentric Network，Concentric Network 再用一个单一的聚合的地址 207.155.128.0/17 来将它的用户归纳给与之相连的 NAP。

表 2-2　　按地理位置进行的 CIDR 地址分配

地　区	地址范围
多地区	192.0.0.0 - 193.255.255.255
欧洲	194.0.0.0 - 195.255.255.255
其他	196.0.0.0 - 197.255.255.255
北美	198.0.0.0 - 199.255.255.255
中/南美	200.0.0.0 - 201.255.255.255
Pacific Rim	202.0.0.0 - 203.255.255.255
其他	204.0.0.0 - 205.255.255.255
其他	206.0.0.0 - 207.255.255.255

例 2-8　在例 2-7 中对地址 207.155.128.5 执行了 WHOIS

```
--- looking up 207.155.128.5
--- performing WHOIS on "207.155.128.5", please wait...
--- contacting host whois.arin.net
--- smart query on "207.155.128"

Concentric Research Corp. (NETBLK-CONCENTRIC-CIDR)
   10590 N. Tantau Ave.
   Cupertino, CA  95014

   Netname: CONCENTRIC-CIDR
   Netblock: 207.155.128.0 - 207.155.255.255
   Maintainer: CRC

   Coordinator:
      DNS and IP ADMIN  (DIA-ORG-ARIN)  hostmaster@CONCENTRIC.NET
      (408) 342-2800
Fax- (408) 342-2810

   Domain System inverse mapping provided by:

   NAMESERVER3.CONCENTRIC.NET    206.173.119.72
   NAMESERVER2.CONCENTRIC.NET    207.155.184.72
   NAMESERVER1.CONCENTRIC.NET    207.155.183.73
   NAMESERVER.CONCENTRIC.NET    207.155.183.72

   Record last updated on 13-Feb-97.
   Database last updated on 29-Jan-99 16:12:40 EDT.
```

向高层域公布一个单一的聚合地址很显然会优于公布可能存在的上百个单独的地址。但是一个同样重要的好处就是，这样一个方案带给 Internet 的稳定性。如果一个低层网络的状态发生变化，那么，只有第一个聚合点会感受到这种变化，而不会产生更进一步的影响。

表 2-3 给出了 CIDR 块的不同规模，它们在 C 类网络中等价的规模以及每个块所能容纳的主机数目。

表 2-3　　**CIDR 块的大小**

CIDR 块前缀的大小	等价的 C 类地址的数目	容纳主机的数目
/24	1	254
/23	2	510
/22	4	1022
/21	8	2046
/20	16	4094
/19	32	8190

续表

CIDR 块前缀的大小	等价的 C 类地址的数目	容纳主机的数目
/18	64	16382
/17	128	32766
/16	256	65534
/15	512	131070
/14	1024	262142
/13	2048	524286

2.1.6　CIDR：降低了 B 类地址空间的消耗

B 类地址空间的枯竭主要源于设计 IP 地址类别时的内在缺陷。一个 C 类地址可提供 254 个主机地址，然而一个 B 类地址可以提供 65534 个主机地址。这是一个很大的差别。在 CIDR 出现之前，如果你的公司需要 500 个主机地址，那么一个 C 类地址将不能满足你的要求。你可能会申请一个 B 类地址，这样你就会浪费掉 65000 个主机地址。有了 CIDR，用一个/23 的地址块就能够满足你的要求。这样可能被浪费掉的主机地址就节省了下来。

2.1.7　CIDR 遇到的问题

虽然在放慢 Internet 路由表的增长速度以及降低 B 类地址耗尽的速度方面，CIDR 被证明是成功的，但是对于 CIDR 块的用户来讲，它还存在一定的问题。

第一个问题就是可移植性。如果分配给你一个 CIDR 地址块，很有可能该地址块是分配给你的 ISP 的一个更大地址块的一部分。假设，你的 ISP 没有满足你的期望或者遵循你们之间的协议，或者你可以从另外一个 ISP 获得更吸引人的条件。但是，更换 ISP 意味着你要被重新分配地址。没有哪一个 ISP 会允许它的用户在转移到新的供应商时还使用该 ISP 分配给它的地址。除了 ISP 不愿意放弃自己的一部分地址空间，区域注册处也强烈地鼓励更换 ISP 的用户归还以前的地址空间。

对于终端用户来讲，重新分配地址会给他们带来不同程度的困难。对于那些在自己的路由域使用专用地址空间以及在域边缘使用网络地址翻译(参见第 4 章)的用户来讲，该过程可能是最容易的。在这种情况下，只有“公共部分”的地址需要改变，对内部用户的影响最小。另一个极端就是终端用户静态地分配公开地址给所有内部网络设备。这种情况下，为了重新分配地址，终端用户别无选择，只能是访问内部每一个设备。

如果终端用户在整个域内使用 CIDR 地址块，通过使用 DHCP(或者 BOOTP)可以在某种程度上降低重新分配地址的痛苦。在这种情况下，必须要改动 DHCP 的范围，并且用户必须重新启动，但是对于那些属于静态分配地址的网络设备，例如服务器和路由器，必须单独地重新分配地址。

如果你不是一个终端用户而是一个 ISP，而且你想变换上行业务供应商，此时，不仅你自己的网络要重新编号，而且那些你已经给分配了 CIDR 地址块的用户也要进行重

新编号。

CIDR 也给那些想要与多个业务供应商相连的用户带来了一个问题。多宿主(将在本章的最后进行深入的探讨)用于冗余备份，这样，一个终端用户或者一个 ISP 对于单一上行业务供应商的故障不再脆弱。但问题在于如果你从一个 ISP 块中撤销了你的地址，你必须向第二个供应商公布这些地址。

图 2-7 显示了可能发生的事情。这里，用户有一个/23 CIDR 地址块，该地址块是 ISP 的/20 地址块的一部分。当用户与 ISP2 相连时，他想要保证来自 Internet 的业务量既可以通过 ISP1，也可以通过 ISP2 到达。为了保证这一点，他必须通过 ISP2 公布他的/23 地址块。当 ISP2 将/23 地址块公布给世界上其他部分的时候，问题出现了。现在所有“不在这里”的路由器都有一个 ISP1 公布的到 205.113.48.0/20 的路由以及 ISP2 公布的到 205.113.50.0/23 的路由。任何到达用户的数据包都被转发到更具体的路由，因此结果是，几乎所有的来自 Internet、去往该用户的数据包都经过 ISP2 转发——包括那些如果经过 ISP1 转发，在地理位置上与用户更接近的信源的业务量。

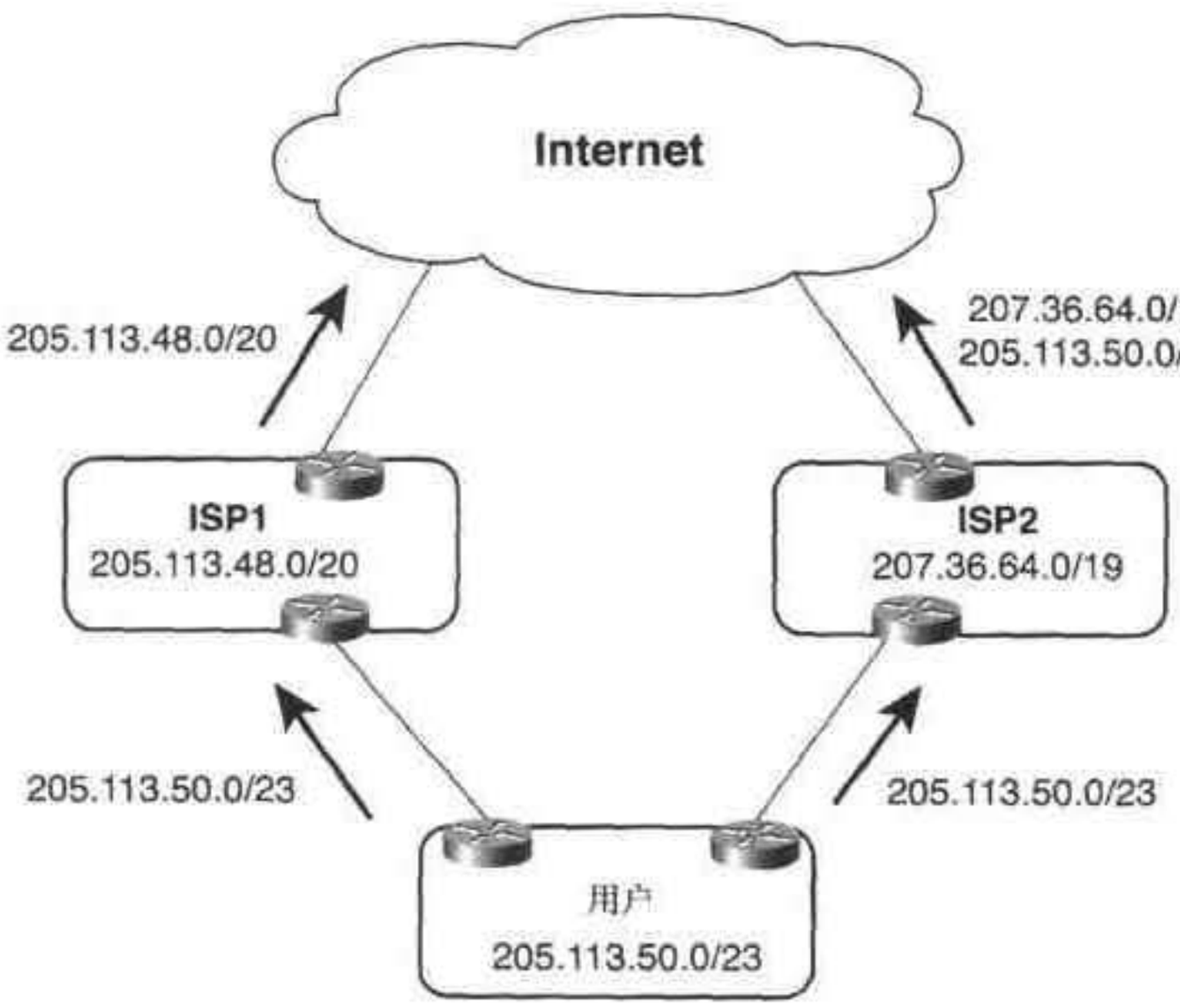


图 2-7　进入到 Internet 的业务量与最具体路由相匹配

图 2-7 中，有可能将来自 Internet 上 205.113.50.0/23 的路由公布给 ISP1。但是这种情况不会发生，因为为了防止自己的路由重新进入到自己的域内，大部分的 ISP 都设置了路由过滤器。但是，这并不能保证 ISP1 能正确地过滤。如果从 Internet 漏入了更具体的路由，来自 ISP1 其他用户的业务量会通过 Internet 以及 ISP2 到达 205.113.50.0/23，而不是通过更加直接的路径。

对于那些多宿主的用户，ISP1 除了公布它自己的 CIDR 地址块(见图 2-8)以外，还必须公布更具体的路由。大部分的业务供应商不同意这种安排，因为这样做意味着在他们自己的 CIDR 地址块里“打了一个洞”(有时被称做地址泄露)。除了降低了 CIDR 的整体效果，公布自己的 CIDR 块的一个具体的路由也为 ISP 带来了管理上的负担。

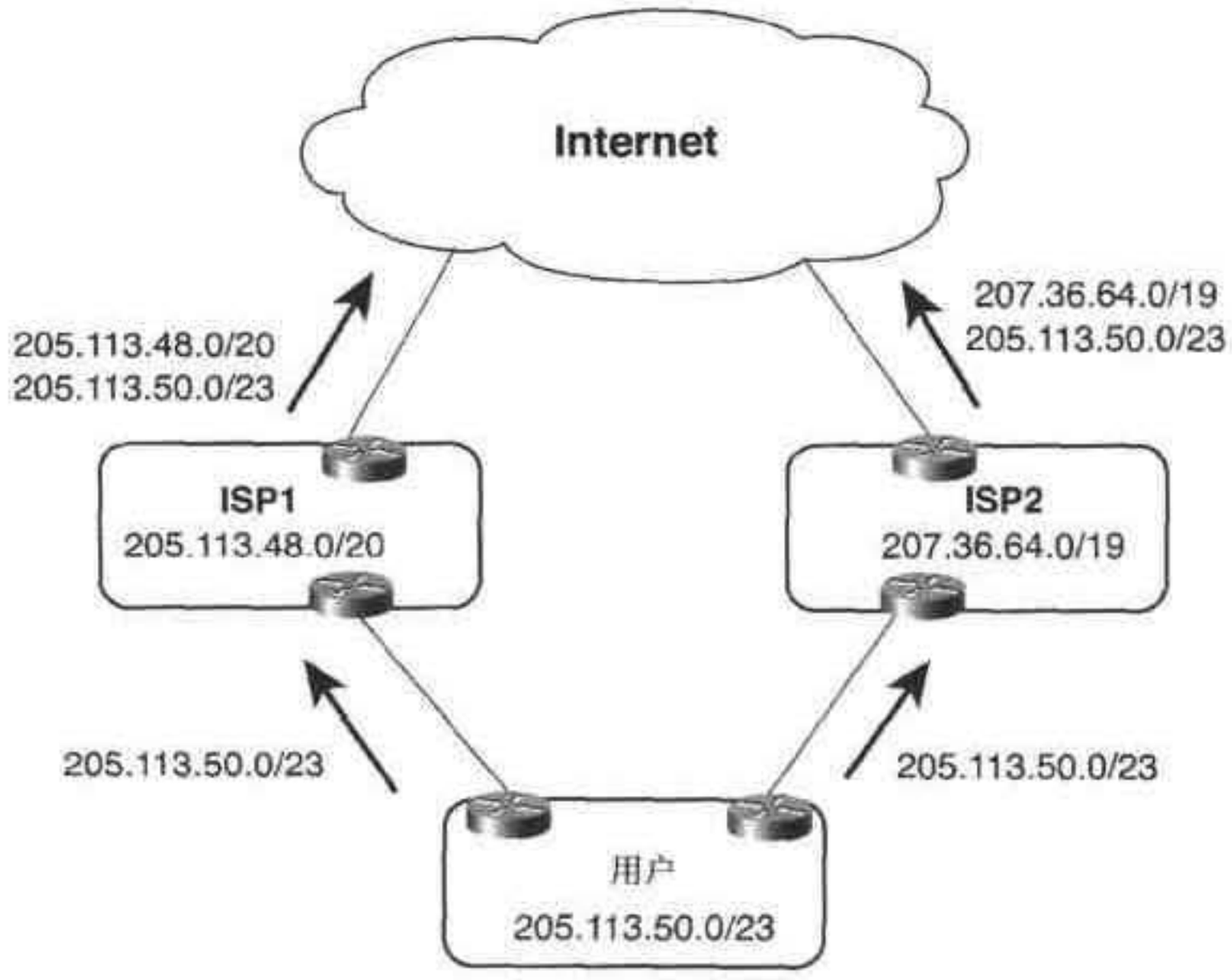


图 2-8　ISP1 在它的 CIDR 块上“打了一个洞”

虽然图 2-7 和 2-8 都显示 ISP1 到 Internet 只有一条连接，但是在大部分情况下，一个 ISP 到高层供应商以及到 NAP 都有多条连接。对于每一条连接，供应商都必须重新配置它的路由器，使得该路由器能够公布除了 CIDR 块之外的更具体的路由，同时可能必须要修改它的入路由过滤器。因为 ISP1 和 ISP2 不得不紧密合作，从而保证用户的/23 块能够被正确地公布，使得管理也变得复杂了。但是因为 ISP1 和 ISP2 是竞争者，他们中的一个或者两个可能都会反对如此紧密的合作。

即使图 2-8 中的用户能够让 ISP1 和 ISP2 同意公布它的/23 块，仍然存在着另外一个障碍。一个 Tier 1 的业务供应商为了控制骨干级路由表，只接受/19 或者更小的前缀。如果 ISP1 或 ISP2 或者两者都是从这样的网络业务提供者获得的 Internet 连接，那么它们就不能公布用户的/23 块。用大于/19 的前缀来过滤 CIDR 地址已经是众所周知的，因此/19 前缀通常被成为*全球可路由地址*。这就意味着如果你公布了一个较长的 CIDR 前缀，例如/21 或者/22，你的前缀可能就不能够公布到 Internet 的所有部分。那么任何一个不知道如何到达你的 Internet 部分，对于你来讲基本上也是不可到达的。

注：　因为有越来越多的用户抱怨上述的问题，许多 Tier I 供应商已经放宽了对/19 的规定。

对图 2-8 中的多宿主用户一个可能的解决方案就是获得一个独立于供应商的地址空间(也称作是可移植地址空间)。也就是说，用户可以申请一个既不属于 ISP1 也不属于 ISP2 CIDR 地址块的地址块；这样两个 ISP 都可以在不干扰它们自己地址空间的情况下，公布用户的地址块。从 ARIN 的形式来看，获得一个独立于供应商的地址块在某种程度上比在 InterNIC 情况下容易。虽然 ARIN 强烈建议你首先应当从你的供应商处获得地址空间，然后是从你的供应商的供应商处获得地址空间，最后才是从 ARIN 获得独立于供应商的地址空间，但是，你还是会面对一些困难。

首先，如果你想多宿主，那么很有可能你现在的地址空间来自于你最初的 ISP。转移到独立于供应商的地址空间意味着要重新编号，也就是说你可能会遇到上面讨论过的所有问题(当然，如果你是在 CIDR 出现之前获得的 IP 地址空间，你已经独立于供应商了，那么上面的问题就不存在了)。

其次，注册处分配地址空间是根据正当需要，而不是根据长期预测的需要。该政策说明可能只分配给你“刚好够用”的地址空间来满足你当前的需要以及预测到的未来三个月的需要。你要证明你充分使用了以前的地址空间，从而为进一步地申请空间给出一个正当的理由。例如，这里有两种办法来证明你的地址利用率：通过 Shared WHOIS Project(SWIP)或者通过 Referral WHOIS Server(RWHOIS)。ARIN 要求你必须使用其中的一种。最长使用的 SWIP 是在 SWIP 模板中加入 WHOIS 信息，然后把它 E-mail 给 ARIN。如果想使用 RWHOIS，你需要在前端设备上建立一个 RWHOIS 服务器，以使 ARIN 能够访问 WHOIS 信息。在这两种情况下，WHOIS 信息证明你已经充分地使用了并且将要用尽你现在的地址空间。

当然，如果你不能正当地获得全球可路由(/19)的地址空间，你还是会遇到一定的问题。最低是 CIDR 分配规则会使得多宿主对于小型用户和 ISP 来讲是一个困难的问题。在下一节会详细地讨论多宿主问题，同时讲一些可供选择的拓扑。

2.2　谁需要 BGP

实际上并不像你想象的那样有很多的网络需要 BGP。一个常见的错误的概念就是当一个网络必须分解成多个路由域时，一定要在不同的域之间运行 BGP。当然 BGP 是一种选择，但是为什么要在混合网络中加入一个不必要的路由协议而使事情变得复杂呢？

例如，一个跨国公司具有 3000 个路由器以及大约 150000 个用户。图 2-9 显示了如何构造一个大型的网络。整个网络用 OSPF 进行路由，并且为了更好地管理，该网络在地理上为分成八个 OSPF 路由域。该图只是说明了每一个 OSPF 域的骨干部分，但是每个 OSPF 域(domain)又被分成了多个 OSPF 区(area)，分别对应不同的地理亚区(subregions)。

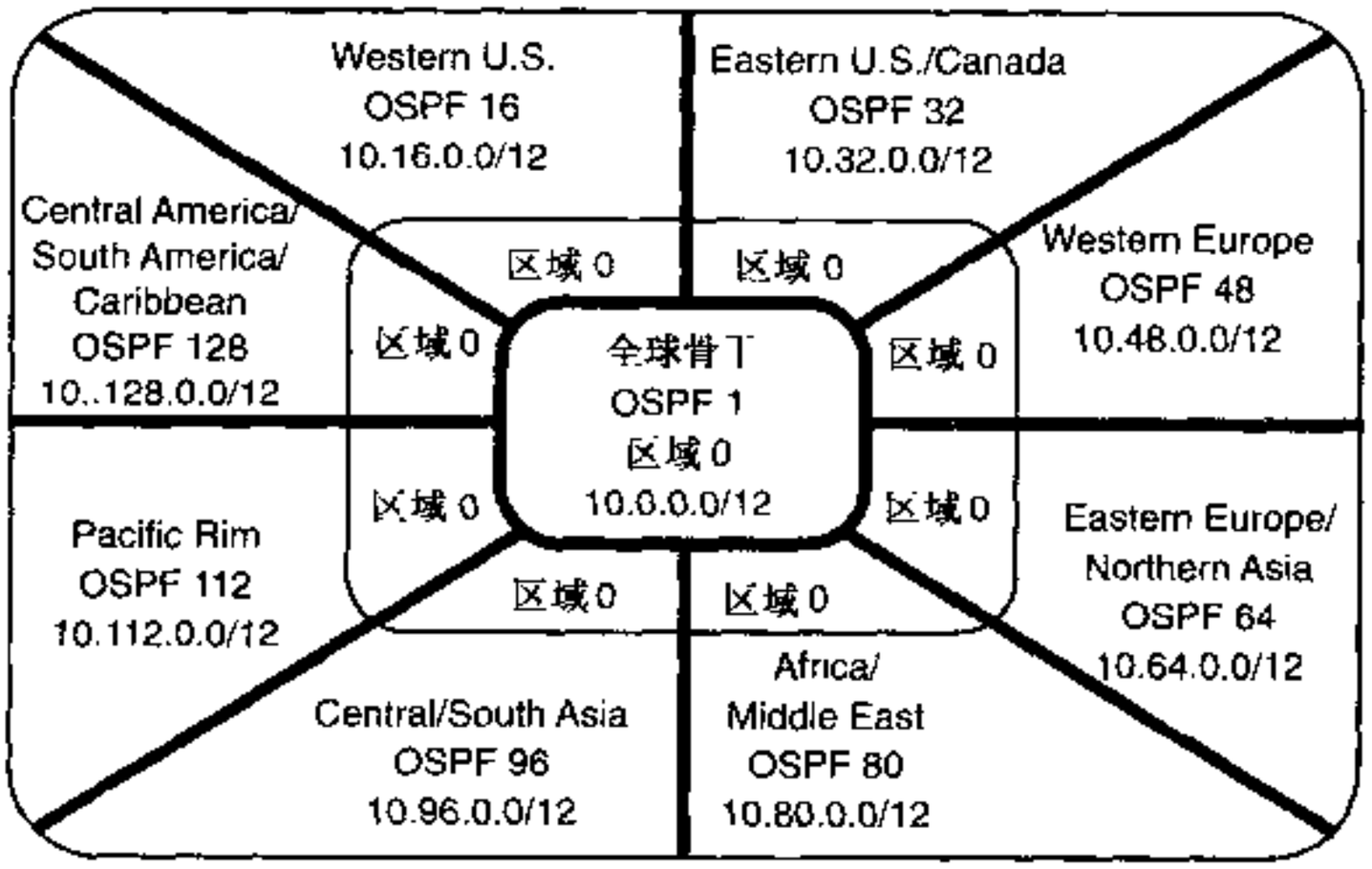


图 2-9　可以通过多个 IGP 域来构造一个非常大型的网络

可以在多个 OSPF 域之间使用 BGP 来提供连接，但这不是必须的。相反，8 个 OSPF 骨干部分的每一个都被再分发到了一个单一的全球的骨干当中。这个全球骨干是另外一个 OSPF 域，它只包括一个单一的 OSPF 区。虽然在这个核心有一个高端路由器来处理分组交换负载，但是来自路由器上的路由表和 OSPF 处理过程的负担其实很小。这跟整个网络的寻址方式有关，8 个 OSPF 域都只向全球骨干公布一个聚合路由。实际上，聚合的方式是该设计工作成功的一个基础。可以想象，在处理这样一个没有聚合 OSPF 的网络中如此大量的子网的过程中可能会造成“阻塞”。这样做的结果就是导致很差的性能以及路由器故障。

在图 2-9 中，物理拓扑的分层结构以及地址空间是简化互联网络三个因素中的两个。第三个因素是整个网络的通用管理实体。单一的管理意味着路由策略被公平地、一致地强加给整个网络。路由策略规定了在每个 OSPF 区内使用的地址范围，并且所有的 OSPF 过程只通过 OSPF 1 互连。

> 注：　路由策略就是通过控制路由以及它们的特性，在一个网络内控制业务量类型的设计和配置过程。在 Cisco IOS 软件中，最常用的执行路由策略的工具有再分发、路由过滤以及路由图。

当然，在实际生活中，几乎没有如图 2-9 所描述规模的公司如此“奢侈”地“从一开始”就将它们的网络建设得这么协调和有逻辑性。许多大型的互联网都是从合并的部门以及公司的小型互联网演变而来的。这样做的结果就是不同的网络管理者为互联网的不同部分做出了不同的设计选择；当该部分被合并以后，要考虑的第一个问题就是互通性。

第二个问题可能就是要加强路由策略。例如，可能会要求一些从互联网络的某些域到其他域的业务量尽量使用一定的链路或者路由，或者也可能是只在域之间公布一定的路由。在大部分情况下，必要的策略还是可以执行的，如在 IGP 之间执行再分发以及路由过滤器和路由图。只有是在工程技术原因的逼迫下，例如 IGP 不能提供必须的工具来执行路由策略，或者当归纳无法再控制路由表的规模时，才应该执行 BGP。当在域里使用许多不同的 IGP 时，BGP 被证明是十分有用的。此时，比起试图在所有的 IGP 内再分发，BGP 执行起来可能是比较简单的。

当考虑在一个互联网设计中 BGP 是否必须时，要记住为什么会首先发明 EGP。EGP 用于在自治系统之间进行路由——也就是说，在属于不同的管理机构的互连网域之间进行路由。在一个单一的企业网内，即使是一个具有属于不同的本地管理机构的不同域的大型网络，通常由一个中心机构通过 IGP 可用的工具来执行路由策略也足够了。但是，如果不同的自治系统必须互连，就可能要用 BGP 了。

使用 BGP 的大部分情况都涉及到 Internet 连接——在用户与 ISP 之间或者在 ISP 之间(大部分情况)。但是，即使是自治系统之间的互连，BGP 可能也不是必须的。本节的剩余部分我们会讨论典型的 AS 域间拓扑并说明哪些地方需要 BGP，哪些地方不需要 BGP。

2.2.1　一个单宿主自治系统

由图 2-10 可以看出，用户和 ISP 之间只有一个连接。在这种拓扑下，BGP 或者其他任何路由协议都是不必要的。当这条链路出现了故障，不需要做什么路由决定，因为没有其他的路由可以选择。一个路由协议做不了任何事情。在这种拓扑结构中，用户在边界路由器中加入了一条静态缺省路由并把该路由再分发到它的 AS 域内。

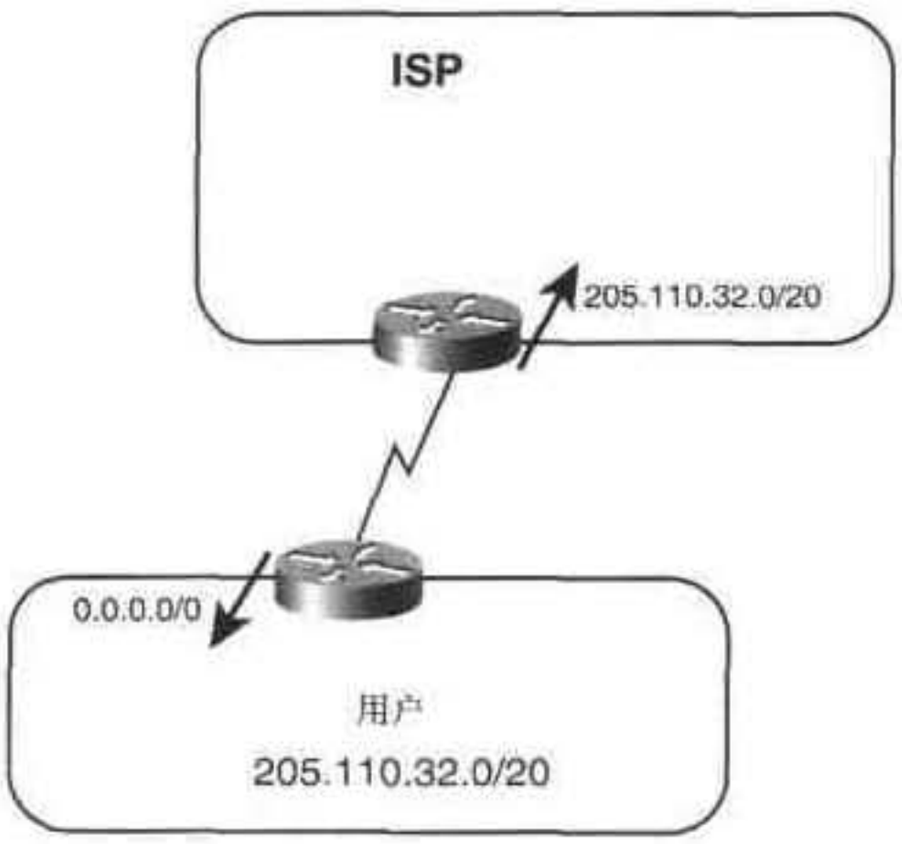


图 2-10　在单宿主拓扑结构中，只需要静态路由

同样地，ISP 在它的路由表中加入了一条指向用户地址范围的静态路由，并在它的 AS 域内公布该路由。当然，如果用户的地址空间就是 ISP 的地址空间的一部分，那么由 ISP 的路由器公布的路由只公布在 ISP 自己的 AS 域内而不会更远。“世界上其他的地方”通过路由到 ISP 公布的地址空间来找到该用户，而且到达该用户的更具体的路由只有在 ISP 的 AS 域内才能得到。

处理 AS 域间的业务量需要记住的另一个重要原则是，每一条物理链路实际上代表两条逻辑链路：一条用于入业务量，一条用于出业务量(如图 2-11 所示)。

在每一个方向上公布的路由会对业务量产生不同的影响。已经写了许多讨论 ISP 问题的优秀文章的 Avi Freedman，将路由公布称做一种承诺，即携带数据包到达路由中所描述的地址空间。在图 2-10 中，用户向它的本地 AS 公布了一条缺省路由——因为没有更具体的路由，这就是一个将数据包发送到任何目的地的承诺。ISP 的路由器公布了一条到 205.110.32.0/20 的路由，承诺将业务量发送到用户的 AS。来自用户 AS 的出业务量是缺省路由的结果，而到用户 AS 的入业务量是 ISP 的路由器公布路由的结果。这个概念看起来是微不足道并且是很显而易见的，但是当你遇到更复杂的拓扑时，记住这个概念非常重要。

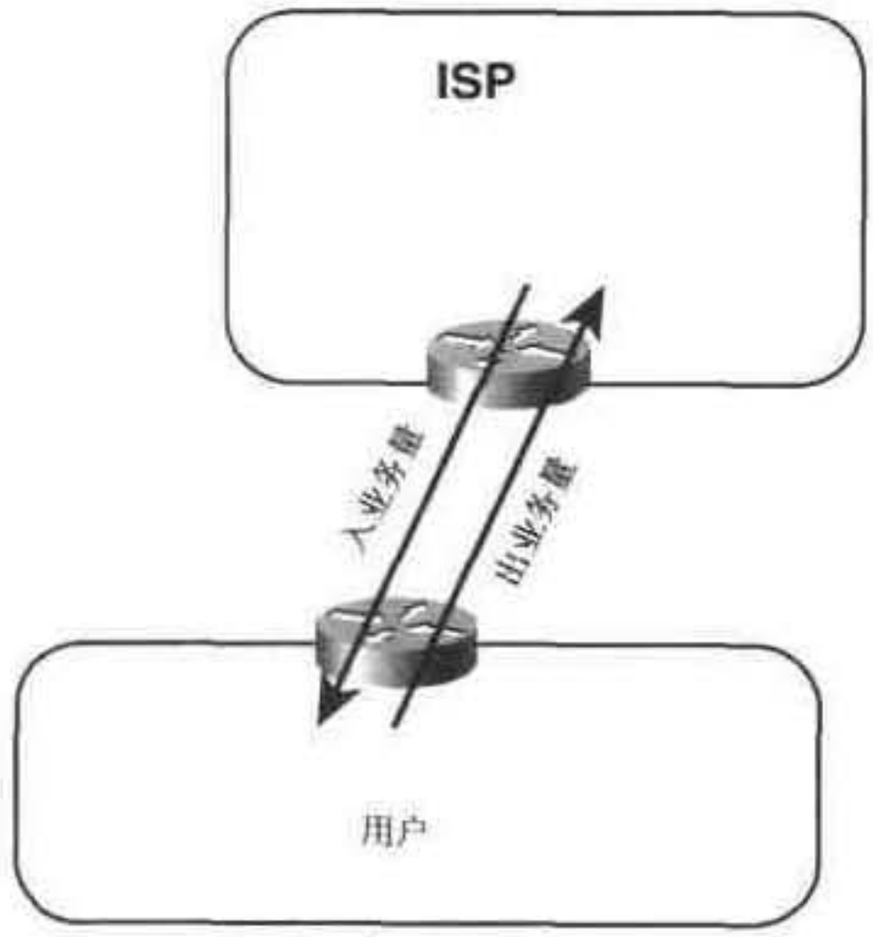


图 2-11　自治系统之间的每条物理链路都代表两条逻辑链路，分别携带入和出的数据包

图 2-10 中拓扑明显的脆弱性使整个连接形成了一个单点故障。如果出现下列情况：该单一数据链路出现故障，路由器或者它的一个端口出现故障，路由器的一个配置出现故障，路由器里的一个处理过程出现故障，路由器管理者犯了一个错误，那么该用户的整个 Internet 连接就会消失。因此该拓扑中缺少的就是冗余备份。

2.2.2　多宿主到一个单一的 AS

图 2-12 给出了一个改进的拓扑，到同一个供应商有了冗余的链路。根据链路的使用情况对通过这些链路的入业务量和出业务量进行处理。例如，多宿主到一个单一的供应商的典型设置就是将一条链路作为主用，共享 Internet 接入链路——例如，一个 T1——而其他的链路只作为备用。在这种方案下，备用链路通常都是那些低速连接。

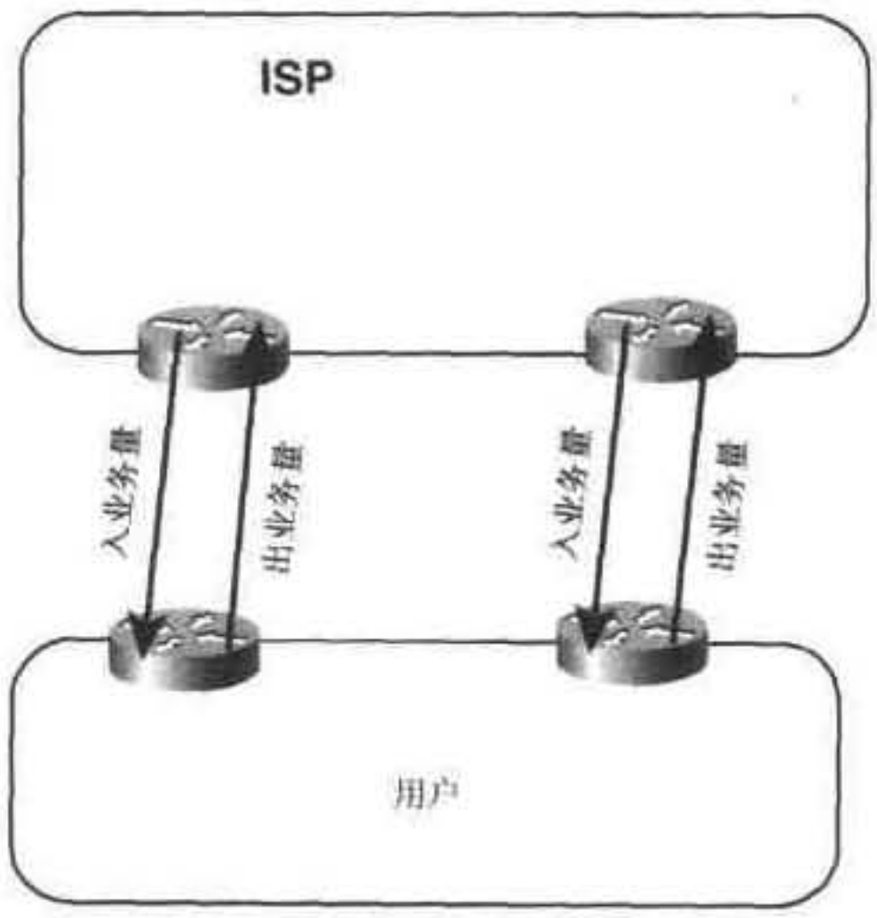


图 2-12　多宿主到一个单一的 AS

当冗余的链路只管理用作备份，就不再需要 BGP。路由可以同单宿方案一样进行公布，只是与备份链路有关的路由的距离设置得较高，因此只有当主用链路出现故障的时候，才使用备用链路。

例 2-9 给出了路由器携带主用和备用链路时的配置情况。

例 2-9　多宿主到一个单一的 AS 时主用与备用链路的配置

```
Primary Router
router ospf 100
 network 205.110.32.0 0.0.15.255 area 0
 default-information originate metric 10
!
ip route 0.0.0.0 0.0.0.0 205.110.168.108
```

```
Backup Router
router ospf 100
 network 205.110.32.0 0.0.15.255 area 0
 default-information originate metric 100
!
ip route 0.0.0.0 0.0.0.0 205.110.168.113 150
```

在这个配置中，备用路由器有一个缺省路由，它的管理距离被设置成了 150，因此只有当来自主用路由器的缺省路由不可用时，备用路由器中的缺省路由才会出现在路由表中。同样，在宣告路由时，备用缺省路由的度量高于主用缺省路由，从而保证 OSPF 域中的其他路由器能够优选主用缺省路由。两种路由的 OSPF 度量类型都是 E2，因此在整个 OSPF 域中公布的度量是一样的。这种一致性保证了在每个路由器中，主用缺省路由的度量低于备用缺省路由的度量，而不去考虑到达每一个边界路由器的内部开销。例 2-10 显示了从一个路由器内部到 OSPF 域的缺省路由。

例 2-10　首先显示的是主用外部路由；然后显示的是在主用路由出现故障以后使用的备用路由

```
Phoenix#show ip route 0.0.0.0
Routing entry for 0.0.0.0 0.0.0.0, supernet
  Known via "ospf 1", distance 110, metric 10, candidate default path
  Tag 1, type extern 2, forward metric 64
  Redistributing via ospf 1
  Last update from 205.110.36.1 on Serial0, 00:01:24 ago
  Routing Descriptor Blocks:
  * 205.110.36.1, from 205.110.36.1, 00:01:24 ago, via Serial0
      Route metric is 10, traffic share count is 1

Phoenix#show ip route 0.0.0.0
Routing entry for 0.0.0.0 0.0.0.0, supernet
  Known via "ospf 1", distance 110, metric 100, candidate default path
  Tag 1, type extern 2, forward metric 64
  Redistributing via ospf 1
  Last update from 205.110.38.1 on Serial1, 00:00:15 ago
  Routing Descriptor Blocks:
  * 205.110.38.1, from 205.110.38.1, 00:00:15 ago, via Serial1
      Route metric is 100, traffic share count is 1
```

虽然这种主/被用的设计满足了冗余的需要，但它并没有有效地利用可用的带宽。一种更好的设计就是在一条链路或者一个路由器出现故障时，它们能互为备用。这种情况下，两个路由器的配置如例 2-11 所示。

例 2-11　多宿主到同一个 AS 时，负载均衡的配置

```
router ospf 100
 network 205.110.32.0 0.0.15.255 area 0
 default-information originate metric 10 metric-type 1
!
ip route 0.0.0.0 0.0.0.0 205.110.168.108
```

两个路由器中的静态路由具有同样的管理距离并且用相同的度量(10)公布缺省路由。注意现在公布缺省路由的 OSPF 度量类型是 E1，如图 2-13 所示。由于使用该种度量类型，每个路由器不仅要考虑缺省路由本身的开销，还要考虑到边界路由器路由的内部开销。结果是，当选择了一条缺省路由后，每个路由器都选择最近的出口点。

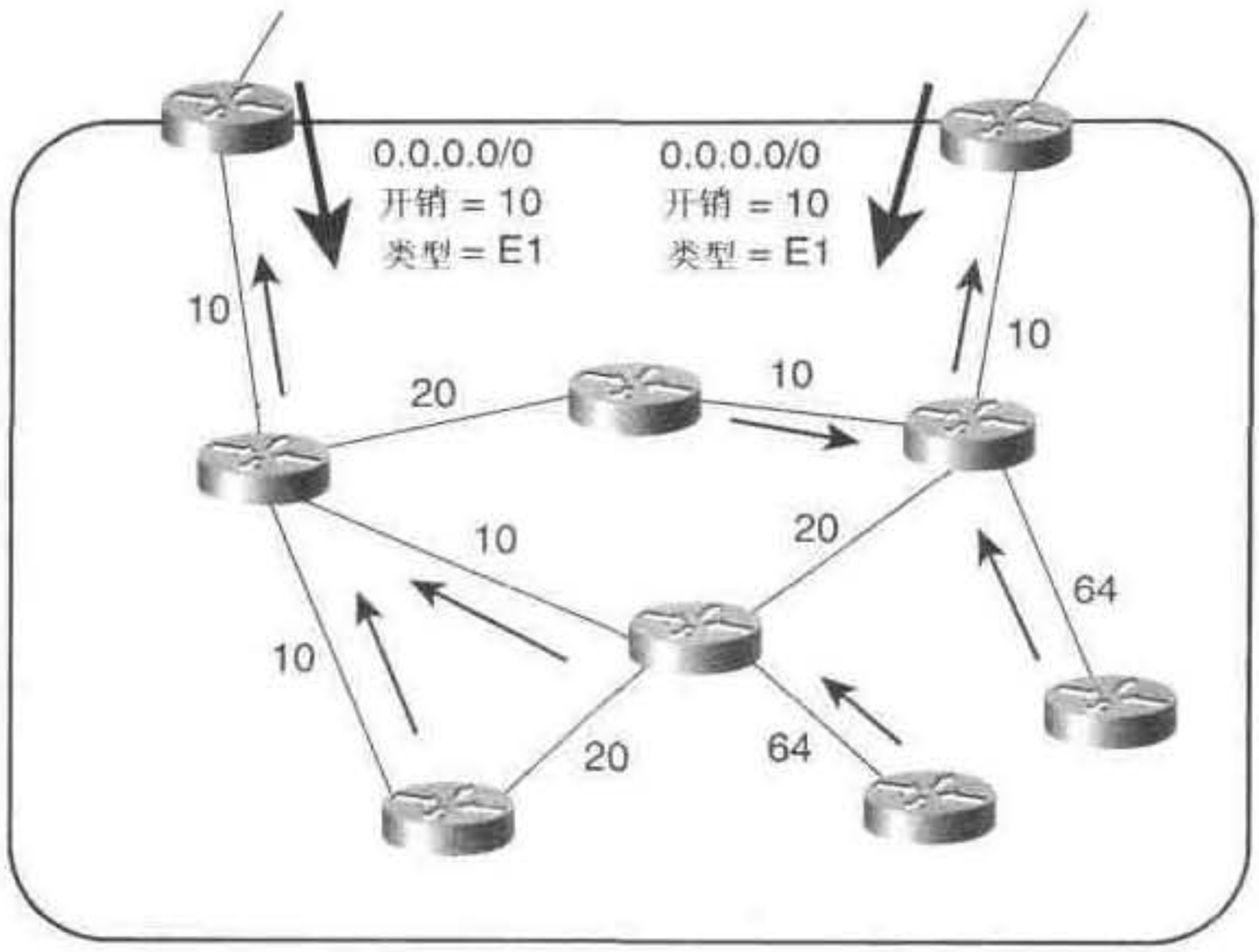


图 2-13　边界路由器以度量 10 和 OSPF 度量类型 E1 公布一条缺省路由

在大部分情况下，都是在多个出口公布缺省路由给 AS，归纳的地址空间在同一个出口离开 AS，这样可以获得足够好的互联网性能。现在值得考虑的是不对称的业务量会不会成为一个问题。如果两个(或者多个)出口点在地理上的距离很大，从而导致时延变化也很大，你可能就需要考虑一种更好的控制路由的方式了。这时就可以考虑 BGP。

例如，假设图 2-12 中两个出口路由器分别在 Los Angeles 和 London。你可能希望所有到东半球的业务量都使用 London 的路由器，所有到西半球的业务量都使用 Los Angeles 的路由器。记住入路由的宣告会影响你的出业务量。如果供应商通过 BGP 将路由宣告到你的 AS，你内部的路由器就会有关于外部目的地的更精确的信息。BGP 还提供了为外部目的地设置路由策略的工具。

同样地，出去路由的宣告会影响到你的入业务量。如果内部的路由是通过 BGP 公布给供应商的，你就有了干预哪条路由从哪个出口宣告的影响力，同样还有了影响供应商如何选择发送业务量到你的 AS 的工具(在一定的程度上)。

当考虑是否使用 BGP 时，一定要仔细衡量得到的好处以及增加了路由复杂性后的开销。只有当你意识到你会在业务量管理上获得一定的优势时，才应该使用 BGP。要将入业务量和出业务量分开考虑。如果只有控制入业务量是重要的，那么你可以使用 BGP 向你的供应商公布路由，同时只需要公布一条缺省路由到你的 AS。

另一方面，如果对你来讲，只有控制出业务量是重要的，那么你就可以使用 BGP 从你的供应商处接受路由。但是你要仔细考虑从供应商处接受路由所带来的后果。“接受所有的 BGP 路由”意味着你的供应商会将整个 Internet 路由表公布给你。如果将它写下来，如例 2-12 所示，大概会有 88000 个路由条目。存储和处理这种规模的路由表需要相当强大的路由器以及至少 64MB 的内存(推荐使用 128MB)。利用一个低端路由器和一个中等数量的内存，你就可以相当容易地执行一个简单的缺省路由方案。

例 2-12　这个 Internet 路由表概括地显示了 57624 条 BGP 条目

```
route-server>show ip route summary
Route Source    Networks    Subnets     Overhead    Memory (bytes)
connected       0           1           56          144
static          2           1           168         432
bgp 65000       76302       11967       4943064     12847416
  External: 88269 Internal: 0 Local: 0
internal        779                                 906756
Total           77083       11969       4943288     13754748
route-server>
```

注：　例 2-12 中的路由表摘要是从 router-sever.ip.att.net 上的一个公开可访问的路由服务器得来的。另一个你可以 Telnet 的服务器是 route-sever.cerf.net。每一个 BGP 条目的数量可能会有一些变化，但它们的规模都是相似的。

“接受部分 BGP 路由”是对接受全部路由和根本不接受路由的一种折衷的办法。从名字我们就可以知道，部分路由是整个 Internet 路由表的一些子集。例如，一个供应商可能只公布到他的其他用户的路由，在加上到 Internet 其他地方的一条缺省路由。在接下来的章节里面给出的情况证明了接受部分路由是有用的。

另外一个值得考虑的问题就是什么时候需要 BGP。一个用户的路由域必须由一个自治系统号来标识。和 IP 地址一样，自治系统号也是有限的，并且只能由区域地址注册处根据合理的需要来分配。同样与 IP 地址类似，一定范围的自治系统号(64512～65535)留做私用。几乎没有例外，用户使用该范围以外的自治系统号与一个单一业务供应商(单宿主或者多宿主)相连。业务供应商将专用 AS 号从公布的 BGP 路由中过滤出去。

虽然图 2-12 中的拓扑是对图 2-10 中拓扑的一种改进，加入了冗余的路由器和数据链路，但是它还是存在一个单一故障点：ISP 本身。如果 ISP 失去了与 Internet 其他地方的连接，那么用户也会失去这些连接。如果 ISP 面临一个严重的内部问题，那么单宿主的用户也会面临同样的问题。

2.2.3　多宿主到多个自治系统

图 2-14 给出了一个拓扑，在该拓扑中一个用户宿主到多个业务供应商。除了已经描述的多种好处，多宿主还会保护用户不会因为发生单一 ISP 故障而引起 Internet 连接的丢失。

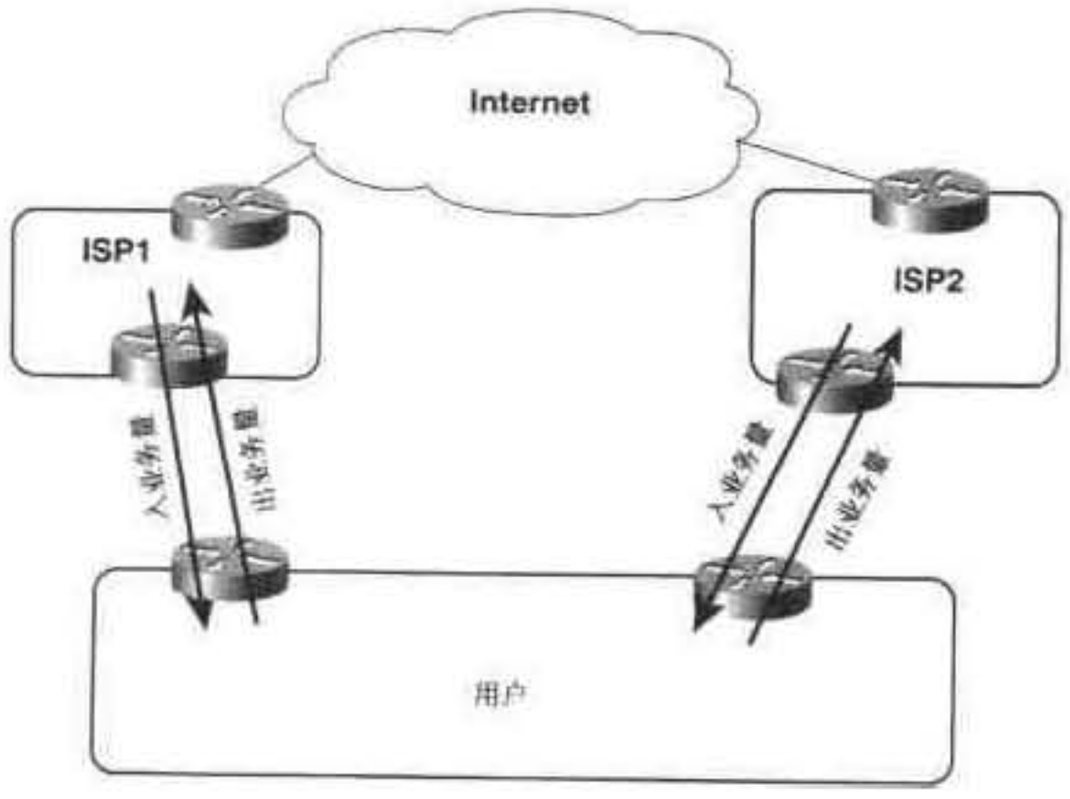


图 2-14　多宿主到多个自治系统

对于小公司或者小型 ISP，多宿主到多个业务供应商存在着许多问题。如果用户的地址空间是一个业务供应商大型地址空间的一部分时，可能会带来以下问题：

- 必须劝说最初的供应商在他的 CIDR 块中“打一个洞”。
- 必须劝说第二个供应商公布属于不同供应商的地址空间。
- 两个供应商在公布用户地址空间方面必须愿意紧密合作。
- 如果用户的地址空间小于/19(小型用户的地址空间可能会是这样)，一些骨干供应商可能不会接受他的路由。

多宿主到多个供应商最佳的候选人就是足够大的公司和 ISP，他们有资格享有独立于供应商的地址空间(或者已经有独立的地址空间)以及公用的自治系统号。

图 2-14 中的用户也可以放弃 BGP。一个选择就是用一个 ISP 作为主用 Internet 连接，其他的只用做备用；另一个选择就是把到两个供应商的路由都作为缺省路由，用赌博式选出可能的路由。但是如果用户已经为多宿主花了钱并且和多个供应商签了合同，那么这两种解决方案可能都是不可取的。在这种情况下，BGP 是一种较好的选择。

再说一遍，入和出的业务量应该分开考虑。对于入业务量，所有内部路由都公布给两个供应商是最可靠的。这种配置保证了通过任何一个 ISP 都可以到达用户 AS 内所有的目的地。即使两个供应商公布了同样的路由，但是入业务量会优先选择其中一条路由。BGP 为这些性能之间的沟通提供了工具。

对于出业务量，从供应商处接受路由应当仔细考虑一下。如果从两个供应商处接受所有的路由，就能为到任何一个 Internet 目的地选择一条最好的路径。在某些情况下，可能希望接受一个供应商全部的 Internet 连接，而对于另外一个供应商，可能只需要到达某些目的地的连接。在这种情况下，就可以接受一个供应商的全部路由，而只接受其他供应商的部分路由。例如，你只想通过第 2 个供应商到达其他的用户并为你的主用 Internet 供应商做备份(如图 2-15 所示，对于大部分的 Internet 连接来讲，ISP1 是首选的供应商；而 ISP2 只用于连接他的其他用户的网络以及用作 Internet 连接的备份)。第 2 个供应商发送他的用户路由，用户配置到第 2 个 ISP 的默认路由，以便到主用 ISP 的路由出现故障时使用。

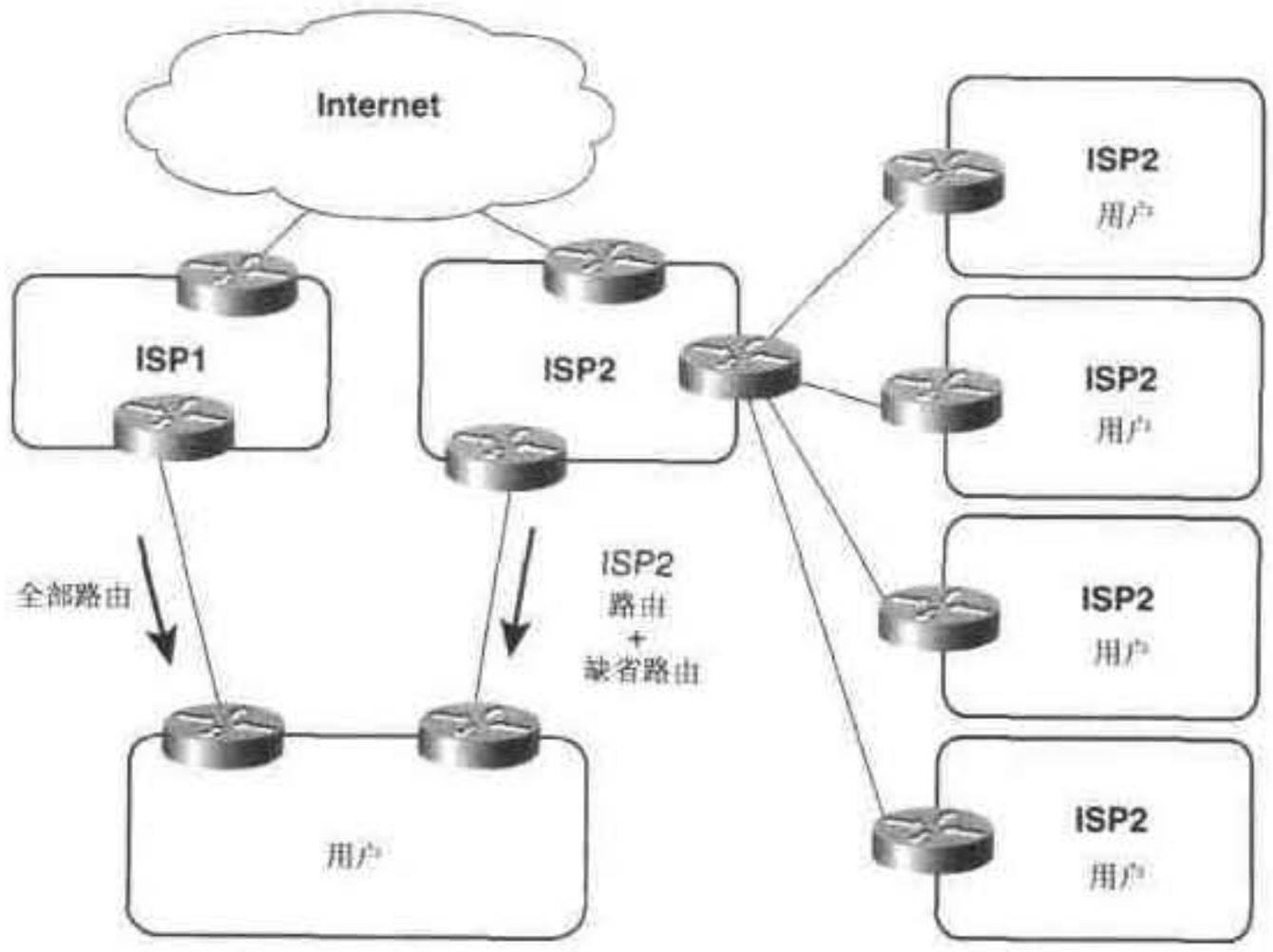


图 2-15　ISP1 是首选的供应商，而 ISP2 只用于连接他的其他用户的网络以及用作连接的备份

注意，由 ISP1 公布的全部的路由可能会包括 ISP2 的用户路由。因为在 ISP2 处接收到了同样的路由。但是通常用户的路由器会优先选择通过 ISP2 的较短的路由。如果到 ISP2 的链路出现了故障，用户会使用通过 ISP1 的较长的路由以及 Internet 的其他部分来到达 ISP2 的用户。

同样地，用户通常通过 ISP1 到达除了 ISP2 用户以外的所有的其他目的地。如果一些或者所有这些来自 ISP1 的更具体的路由出现了故障，用户会使用通过 ISP2 的缺省路由。

如果由于路由器 CPU 和内存的限制禁止它接受所有的路由，那么可以选择两个供应商的部分路由。每一个供应商都可能公布它自己的用户路由以及到两个供应商的用户点的缺省路由。在这种情况下，牺牲一些路由的精确性可以节省了路由器的硬件。

在另一种部分路由方案中，每一个 ISP 可能会公布他的客户路由以及他上级供应商的客户路由。例如，在图 2-16 中，ISP1 与 Sprint 相连，ISP2 与 MCI 相连。由 ISP1 发送给用户的部分路由包括 ISP1 所有客户的路由以及 Sprint 所有客户的路由。由 ISP2 发送给用户的部分路由包括 ISP2 所有客户的路由以及 MCI 所有客户的路由。用户用缺省路由指向两个供应商。由于两个骨干业务供应商的规模，用户有足够的路由做出到众多目的地的有效的路由决定。同时，部分路由还是比 Internet 全部的路由表要小。

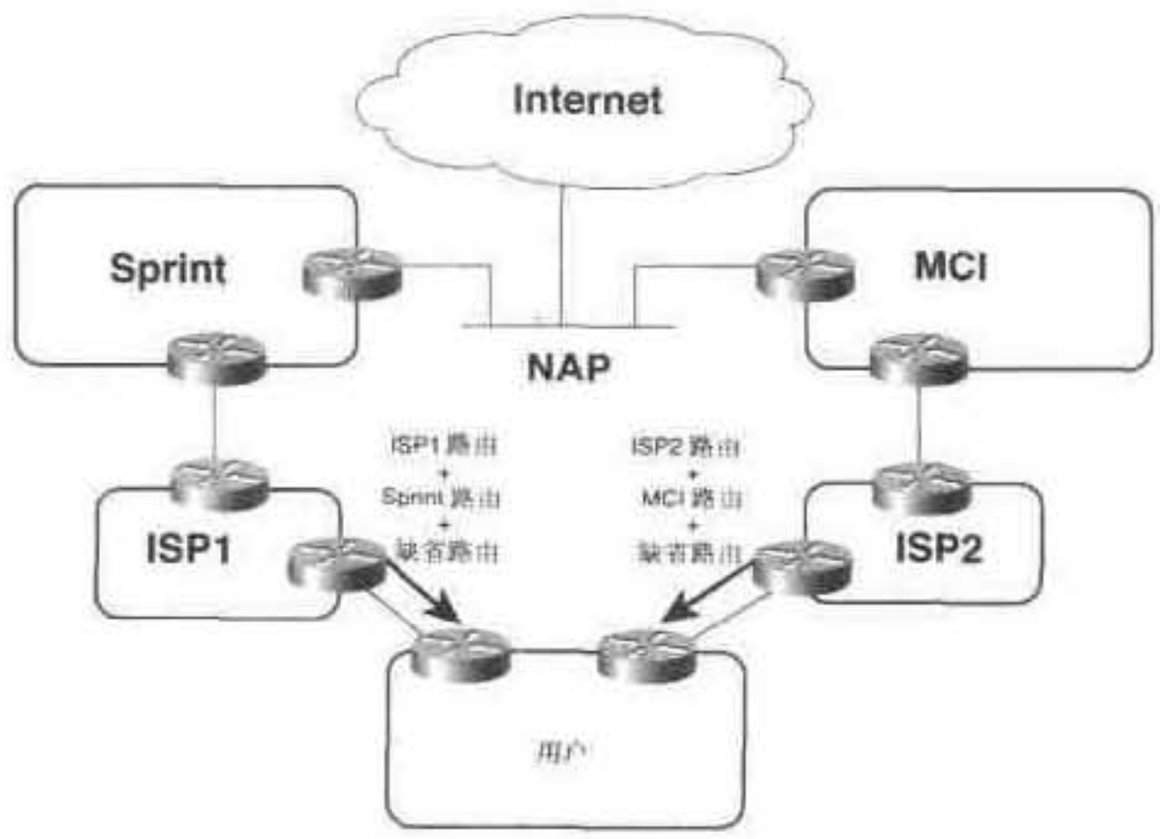


图 2-16　用户从两个 ISP 处得的路由

本章剩余的部分(两个短的提示部分)将介绍一下 BGP 的操作以及它为入和出业务量提供的设置优选项和策略的工具。

2.2.4　“负载均衡”中应当注意的一个问题

多宿主基本的好处就是冗余并且在一定程度上增加了带宽。提高的带宽并不意味着以相同的效率使用两条链路。你不应该期望两条链路上业务量负载均衡到 50/50；一个 ISP 对于另一个 ISP 通常是“较好的连接”。比起另外的 ISP，该 ISP 本身或者他的上级供应商可能有较好的路由器、较好的物理链路或者更多的 NAP 连接，或者一个 ISP 可能刚好在拓扑上与你的大部分用户通常要与之相连的目的地更接近。

这并不是说，你不能通过消耗一定的时间和精力来处理路由的优选项问题，从而在这两条链路上公平、均匀地平衡你的业务量。问题是通过强迫某些业务量选择非最优的路由去追

求所谓的负载均衡实际上可能降低了 Internet 性能。实际上，大部分情况下，你需要完成的就是在两条 ISP 链路利用率数量上的一个均衡。其实你不用太担心有 75%的业务量使用一条链路，而只有 25%的业务量使用另外一条链路。多宿主只是用于冗余备份和提高路由效率，并不来做负载均衡。

2.2.5　BGP 的危险

建立一个 BGP 对等关系会涉及到一个信任与不信任的问题。BGP 的对端是另外一个 AS，你必须相信另一端的网络管理者，而且要知道他或者她正在做什么。同时，如果你聪明的话，你会采取任何一种可用的方法来保护你自己不会受到对端错误的影响。当你执行一个 BGP 对等连接时，最好对任何一个细节都进行周密的考虑。

回忆一下把路由宣告作为一个承诺的早期描述，也就是承诺将数据包传送到宣告的地址。你公布的路由直接影响到你接收到的数据包，你接收到的路由直接影响到你发送的数据包。在一个好的 BGP 对等协议中，双方应该完全了解在每个方向上公布了什么路由。再一次提醒你，入和出业务量应该分开考虑。每一个对等应该保证他正在公布正确的路由，并且应该使用路由过滤器或者其他策略工具，例如 AS_PATH 过滤器(将在第 3 章描述)，从而保证他正在接收正确的路由。

如果你在 BGP 配置中犯了错误，你的 ISP 可能会对你缺乏耐心，但是最严重的问题可能是双方对等关系处理时的错误。例如，假设通过一些错误的配置，你公布 207.46.0.0/16 到你的 ISP。在接收端，ISP 不会过滤掉错误的路由，而是允许它被公布到 Internet 的其他部分。这个特殊的 CIDR 块属于 Microsoft，而且你刚好声明你有一条到那个目的地的路由。Internet 团体的相当一部分可能会决定到 Microsoft 的最好的路由是通过你的域。通过 Internet 连接，你将收到大量的多余的数据包，而且，更重要的是，你可能会对应该到 Microsoft 去的业务量造成了黑洞。这既不令人愉快，也不能为人所接受。

图 2-17 给出了 BGP 路由错误的另一个例子，用户将从 ISP2 学习到的路由公布给 ISP1，引起目的地是 ISP2 以及它的用户的数据包能够通过它的域。该网络和图 2-15 中的网络相同，但是此时用户从 ISP2 学习到的用户路由是 ISP1 在非故意的情况下公布的。

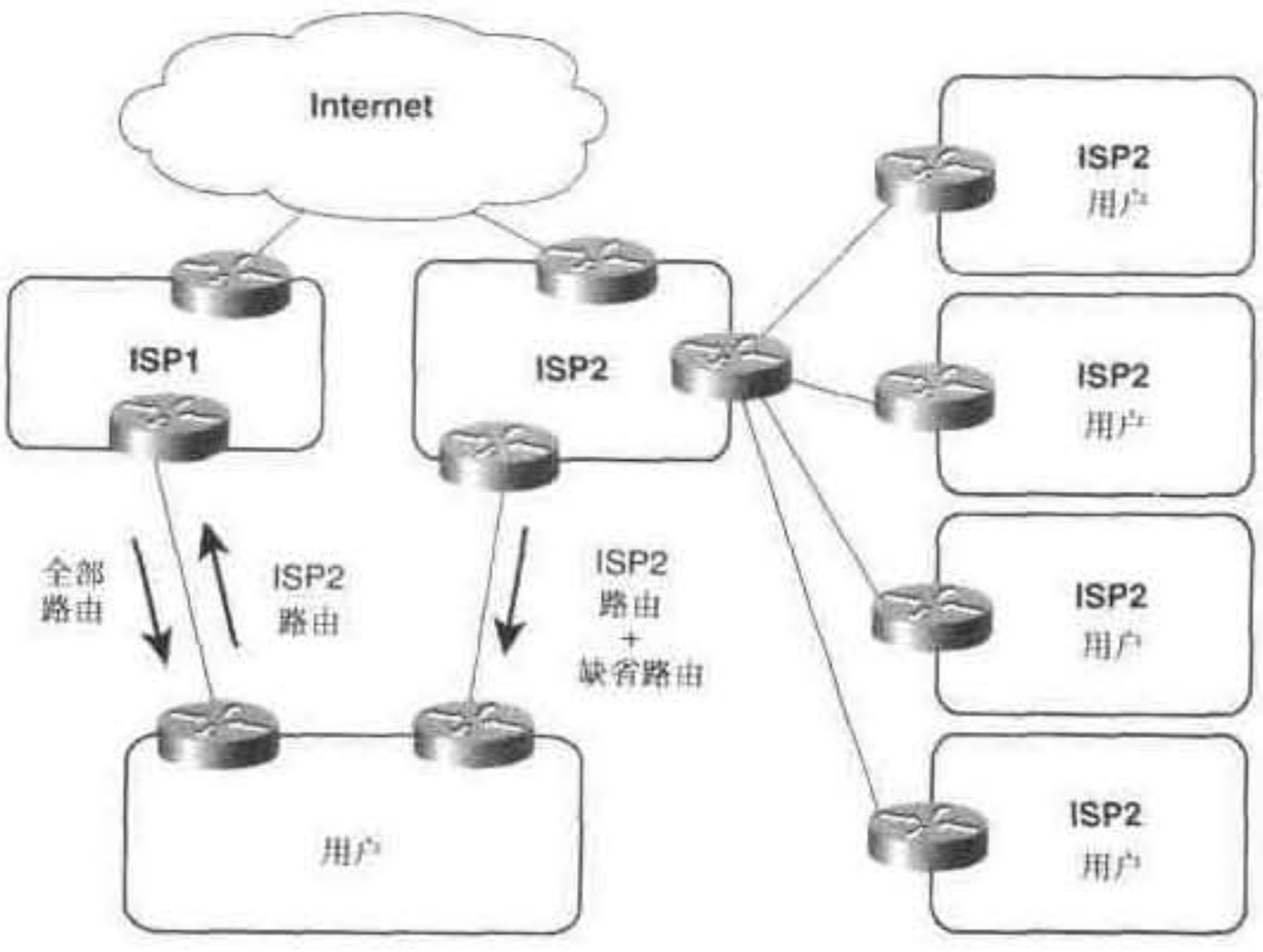


图 2-17　BGP 路由错误

对于所有的可能性，ISP1 和他的客户将把用户域看作是到 ISP2 和他的客户的最好的路径。在这种情况下，因为用户实际有到 ISP2 的路由，因此业务量没有形成黑洞。用户域成了数据包从 ISP1 到 ISP2 的转接域，这将造成对他自己业务量的一种损坏。因为从 ISP2 到 ISP1 的路由还是通过 Internet，这样用户为 ISP2 造成了一种非对称路由。

本节的重点是 BGP，因为它本身的特点，它被设计成在自治系统之间进行通信。一个成功和可靠的 BGP 对等协议不仅要求彻底地了解每个方向上公布的路由，同时还要彻底地了解每一个涉及方的路由策略。

2.3　BGP 基础知识

像 EGP 一样，BGP 到它每一个运行 BGP 的对端都形成了一个的独特的、基于单播的连接。为了提高对端连接的可靠性，BGP 使用 TCP(端口 179)作为它底层的传输机制。BGP 的更新机制也因为让 TCP 层来处理像确认、重传以及排序等任务而在一定程度上得到了简化。因为 BGP 运行于 TCP 之上，必须在每个对端之间建立单独的点到点连接。

BGP 是距离向量协议，因此每一个 BGP 节点要依靠下游邻居来将它路由表中的路由传送下去；节点在公布路由的基础上进行路由计算并将结果传给上行的邻居。但是，其他的距离向量将距离量化成单纯的数字来代表跳数，或者在 IGRP 和 EIGRP 中代表整个接口时延和最低带宽之和。相比之下，BGP 使用一个 AS 号列表，数据包必须通过这些 AS 才能到达目的地(见图 2-18)。因为这个列表完整地描述了数据包需要经过的路径，因此把 BGP 称为路径向量路由协议，从而和传统的距离向量协议形成一个对比。与一个 BGP 路由有关的 AS 号列表称做 *AS-PATH*，它是与路由有关的几个*路径属性*之一。在后续的章节中将详细地描述路径属性。

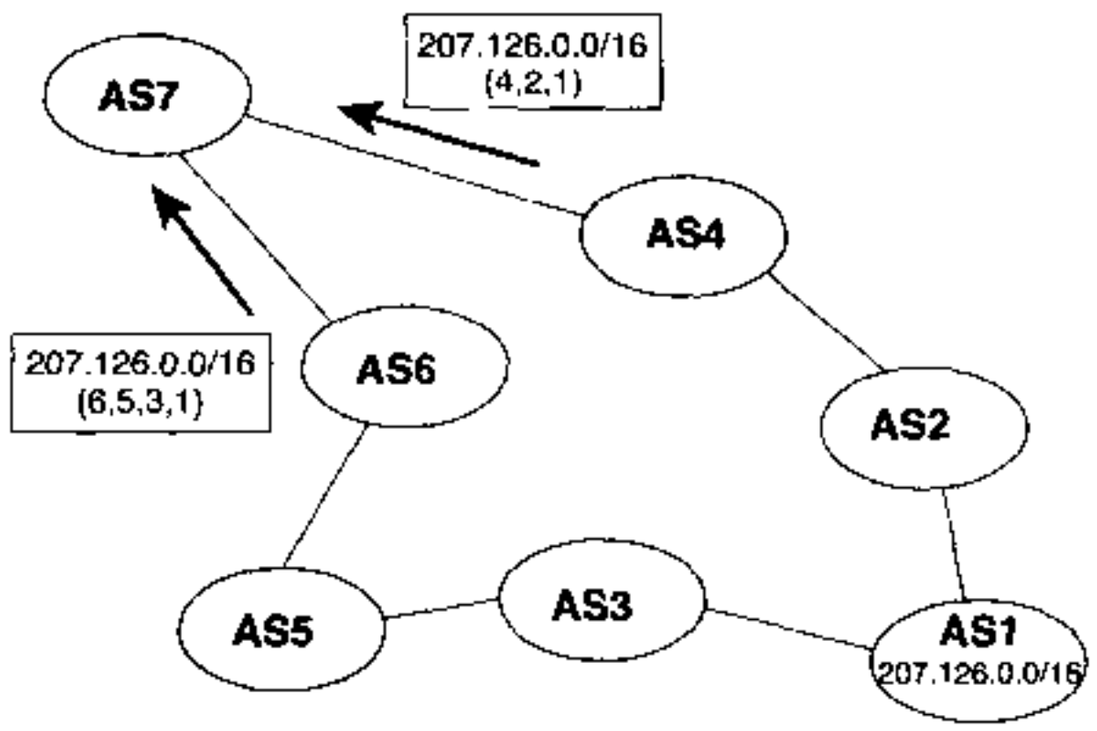


图 2-18　BGP 通过 AS_PATH 属性来决定最短的无环路 AS 之间的路径

回忆一下在第 1 章中我们曾经讲过 EGP 不是真正的路由协议，因为它不具备完整开发的计算最短路径的算法，而且它不能检测到路由环路。相比之下，就上面的两个方面来讲，AS-PATH 属性使得 BGP 有资格做一个路由协议。首先，根据最少 AS 数能够非常简单地决定最短的 AS 域间路径。在图 2-18 中，AS 7 收到了两条到 207.126.0.0/16 的路由，其中一条有 4 个 AS 跳，而另外一条有 3 个 AS 跳，AS 7 会选择最短的路径(4,2,1)。

通过 AS-PATH 属性，路由环路也很容易检测到。如果在路由器接收到的一条更新消息

的 AS-PATH 中包含本地的 AS 号，这就说明检测到了一条路由环路。在图 2-19 中，AS 7 公布了一条路由给 AS 8。AS 8 将该路由公布给 AS 9，AS 9 又将它公布给 AS 7。AS 7 在 AS-PATH 中看到了自己的 AS 号并拒绝接受该更新消息，这样就避免了潜在的路由环路。

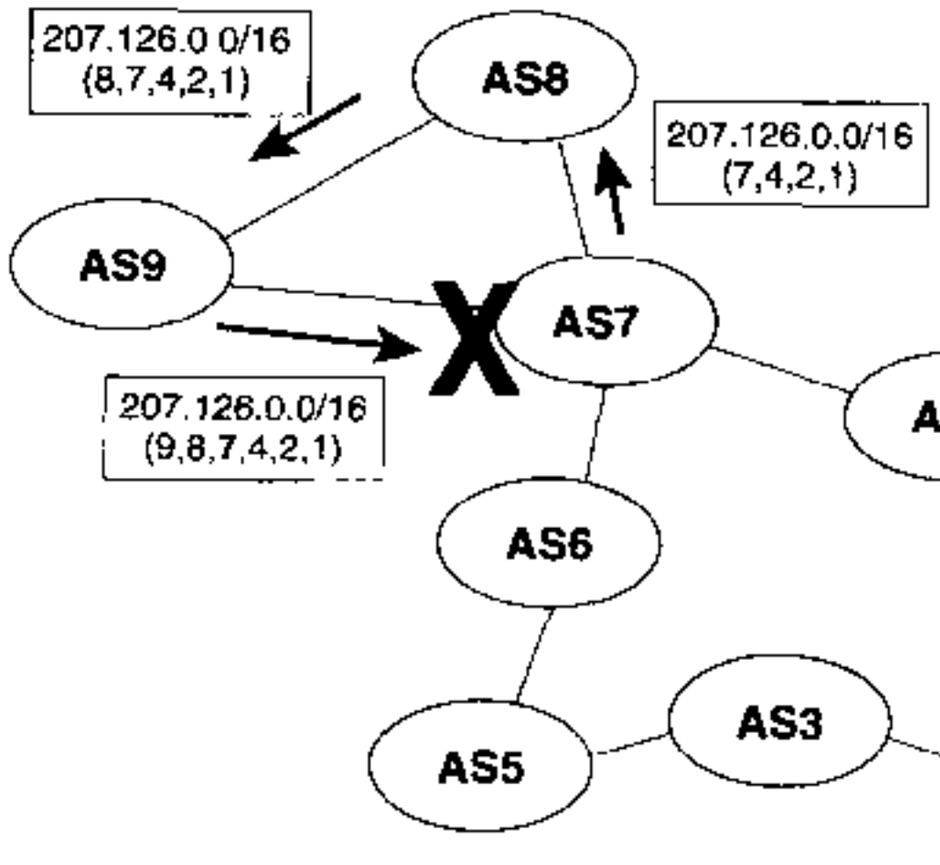


图 2-19　如果一个 BGP 路由器看到了自己的 AS 号，它会拒绝这个更新消息

BGP 没有给出每一个 AS 域内拓扑的细节。因为 BGP 只看到自治系统树，因此可以说 BGP 比 IGP 看到的 Internet 层面要高，IGP 只看到 AS 域内的拓扑。因为 BGP 在高层所看到的与 IGP 所看到的内容并不真正兼容，Cisco 路由器用一个单独的路由表来存放 BGP 路由。例 2-13 说明了一个用 **show ip bgp** 命令所看到的典型的 BGP 路由表。

例 2-13　show ip bgp 命令显示出了 BGP 路由表

```
route-server>show ip bgp
BGP table version is 4639209, local router ID is 12.0.1.28
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  3.0.0.0          192.205.31.225                         0 7018 701 80 i
*                   192.205.31.161                         0 7018 701 80 i
*>                  192.205.31.33                          0 7018 701 80 i
*                   192.205.31.97                          0 7018 701 80 i
*  4.0.0.0          192.205.31.225                         0 7018 1 i
*                   192.205.31.161                         0 7018 1 i
*>                  192.205.31.33                          0 7018 1 i
*                   192.205.31.97                          0 7018 1 i
*  6.0.0.0          192.205.31.226                         0 7018 568 721 1455 i
*                   192.205.31.225                         0 7018 568 721 1455 i
*                  192.205.31.161                 0 7018 701 6113 568 721 1455 i
*>                  192.205.31.34                          0 7018 568 721 1455 i
*                   192.205.31.33                          0 7018 568 721 1455 i
*                  192.205.31.97                  0 7018 1239 568 721 1455 i
*  9.2.0.0/16       192.205.31.225                         0 7018 1 1673 1675 i
*                   192.205.31.161                         0 7018 701 1673 1675 i
 --More--
```

虽然例 2-13 中的 BGP 路由表与用 **show ip route** 命令所显示的 AS 域内部的路由表有一定的不同，但它们还是有相同的部分。该路由表显示了目的地网络、下一跳路由器、选择最

短路径所使用的方法。**Metric**、**LocPrf** 和 **Weight** 列将在本节的后面进行讨论，现在最引人注意的是 **Path** 列。该列列出了每个网络的 AS-PATH 属性。注意每个 AS-PATH 都以 i 结尾。根据 **Origin Codes** 图表，这个 i 表示这个路径终止于一个 IGP。

同样我们也会注意到，对于每一个目的地网络，路由表中列出了多个下一跳路由器。不像 AS 内部路由表只列出当前正在使用的路由，BGP 路由表列出了所有可知的路径。在最左边一列，*号后面跟着一个>号，指示该路径是路由器正在使用的路径。具有最短 AS-PATH 的路径是最好的路径。正如例 2-13 所示，当多个路由有相同的路径时，路由器必须有一定的标准来决定选择哪条路径。本节的后面将会涉及到这个决定过程。

如例 2-13 所示，当到一个特殊的目的地有并列的、等开销的路径时，Cisco 缺省地执行 EBGP 从而只选择一条路径——这与其他的 IP 路由协议不同，其他路由协议在缺省的状况下是在多达 4 条路径上均衡负载。与其他路由协议相同，**maximum-paths** 命令用来改变并行路径缺省的最大数目，范围是从 1 到 6。注意负载均衡只用在 EBGP 中，IBGP 只使用其中一条链路。

一个运行 BGP 的路由器对端的邻居既可以在不同的 AS 域内，也可以在同一个 AS 域内。如果邻居的 AS 与自己的 AS 不同，那么就称该邻居为外部对端，BGP 被称做外部 BGP(EBGP)。如果邻居与自己在同一个 AS 域内，那么该邻居就是一个内部对端，BGP 被称做内部 BGP(IBGP)。当配置 IBGP 时必须要面对一系列独特的问题，这些问题将在“IBGP 和 IGP 的同步”一节中进行讨论。

当两个邻居第一次建立一个 BGP 对等连接时，它们要交换整个 BGP 路由表。交换完路由表以后，它们会交换增加的、部分的更新消息——也就是说，当它们的路由内容发生变化时，它们要交换路由信息而且只交换那些有变动的信息。因为 BGP 不使用周期性路由更新，对端必须交换 Keepalive 消息，从而保证连接得以维持。Cisco 缺省的 Keepalive 间隔为 60 秒(RFC1771 没有规定一个标准的 Keepalive 时间)。如果 3 个时间间隔(180 秒)以后，对端没有收到一个 Keepalive 消息，对端就会宣布它的邻居处于 DOWN 状态。可以用 **timers bgp** 命令来改变这些间隔。

2.3.1　BGP 消息类型

在建立一个 BGP 对等连接之前，两个邻居必须执行标准的 TCP 三次握手，并且打开一个到端口 179 的 TCP 连接。TCP 提供一个可靠连接所需要的分段、重传、确认以及排序功能，从而把 BGP 从这些任务中解脱出来。所有的 BGP 消息都通过 TCP 连接单播给一个邻居。

BGP 具有四种消息类型：

- Open
- Keepalive
- Update
- Notification

本节将讨论如何使用这些消息。关于消息格式以及每个消息字段变量的完整描述，参见“BGP 消息格式”一节。

1、Open 消息

TCP 会话建立起来以后，两个邻居都要发送一个 Open 消息。每个邻居都用该消息来标识自己，并且规定自己的 BGP 运行参数。Open 消息包括以下信息：

- **BGP 版本号**——它明确了发起者正在运行的 BGP 版本(2、3 或者 4)。除非通过 **neighbor**

version 命令使一个路由器运行较早的版本，否则缺省的版本是 BGP-4。如果一个邻居运行的是较早的 BGP 版本，它会拒绝版本 4 的 Open 消息；于是路由器将版本 4 改为版本 3 并且再发送一个确定了该版本的 Open 消息。这个协商过程一直持续到两个邻居对版本达成了一致。

- **AS 号**——这是发起会话路由器的 AS 号。它用来决定该 BGP 会话是 EBGP(如果两邻居的 AS 号不同)，还是 IEGP(两邻居的 AS 号相同)。
- **Hold time**——路由器必须收到一个 Keepalive 或是更新消息之前所允许经过的最大秒数。保持时间必须是 0 秒(在这种情况下，必须是没有发送 KEEPALIVE)或者至少 3 秒；Cisco 缺省的保持时间是 180 秒。如果两邻居之间的保持时间不同，那么这两个时间中较短的时间为两者所接受的保持时间。
- **BGP 标识符**——用来标识邻居的 IP 地址。Cisco IOS 决定 BGP 标识符的方式与它决定 OSPF 路由器 ID 的方式完全相同：使用数值最大的 loopback 地址；如果没有配置 IP 地址的 Loopback 接口，选用物理接口上数值最大的 IP 地址。
- **可选参数**——这个字段用于公布对一些可选功能的支持，如鉴别、多协议支持以及路由刷新。

2．Keepalive 消息

如果路由器接受了它的邻居在Open消息中的参数，它就会应答一个Keepalive消息。Cisco缺省的情况是，每隔 60 秒发送一个 Keepalive 消息，或者是以达成一致的保持时间的 1/3 为时间段发送 Keepalive 消息。

3．Update 消息

Update 消息用来公布可用的路由、撤消的路由或者两者兼顾。更新消息中包含以下信息：

- **网络层可到达信息(NLRI)**——这是一个或者多个(长度、前缀)用来公布 IP 地址前缀和前缀长度的字节组。例如，如果公布了地址 206.193.160.0/19，长度部分就是/19，前缀部分就是 206.193.160.
- **路径属性**——将在后续章节描述的路径属性是已公布的 NLRI 的特点。该属性为 BGP 提供了选择最短路径、检查到路由环路以及决定路由策略的信息。
- **撤消路由**——用来描述已经变成无法到达并且正从业务中撤消的目的地的字节组(长度、前缀)。

注意，虽然在 NLRI 字段中可能会包含多个前缀，但是每个更新消息只描述一条 BGP 路由(因为路径属性只描述一条路径，但是该路径可能会到达多个目的地)。这点又再一次说明 BGP 所看到 Internet 的层面要高于 IGP，因为 IGP 路由通常只有一个目的地 IP 地址。

4．Notification 消息

当检测到差错的时候就会发送 Notification 消息，通常这会导致 BGP 连接的终止。在“BGP 消息格式”一节中有一个可能会导致发送 Notification 消息的差错的列表。

使用 Notification 消息的一个例子就是邻居之间关于 BGP 版本的协商。如果已经建立一个 TCP 连接，运行 BGP-3 的邻居收到了一个版本 4 的 Open 消息，该路由器就会应答一个 Notification 消息，声明它不支持版本 4。该连接被终止，邻居会试图建立一个 BGP-3 的连接。

2.3.2　BGP 有限状态机

BGP 连接的建立和维持可以用有限状态机来描述。图 2-20 和表 2-4 给出了完整的 BGP

状态机以及引起状态改变的输入事件。

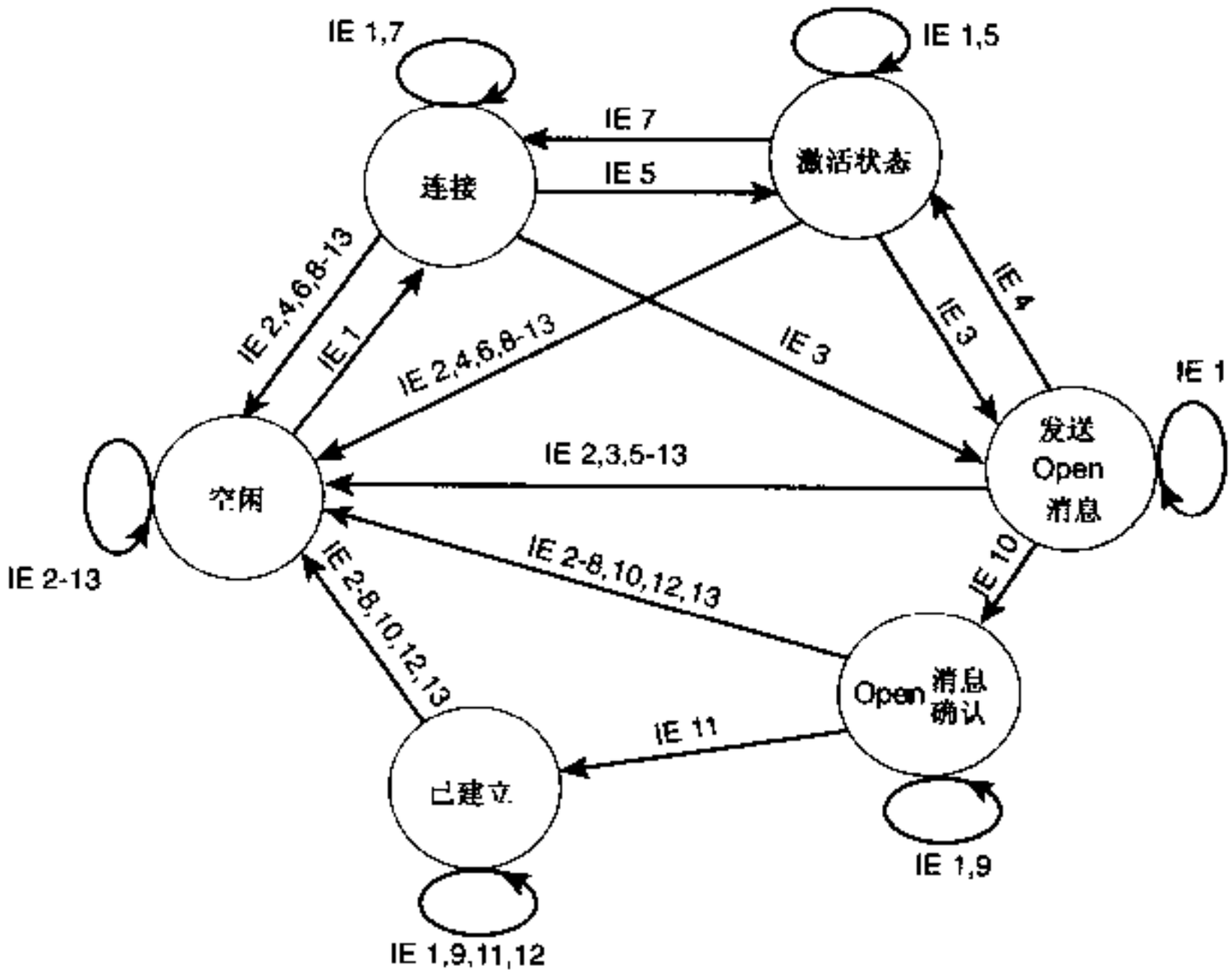


图 2-20　BGP 有限状态机

表 2-4　　图 2-20 的输入事件(IE)

IE	描　　述
1	BGP 开始
2	BGP 结束
3	BGP 传输连接打开
4	BGP 传输连接终止
5	BGP 传输连接打开失败
6	BGP 传输致命差错
7	重试连接计时器超时
8	持续时间终止
9	Keepalive 计时器终止
10	收到打开消息
11	收到 Keepalive 消息
12	收到更新消息
13	收到通知消息

接下来的章节对图 2-20 中的 6 个状态给出了详细的描述。

1．空闲状态

BGP 通常以空闲状态开始。在该状态下，它拒绝接收所有入连接。当一个开始事件(IE1)出现的时候，BGP 过程初始化所有的 BGP 资源，打开重试连接计时器、初始化到邻居的 TCP 连接、接听来自邻居的 TCP 初始化消息并将它的状态转到连接状态。开始事件是由一个操作者配置一个 BGP 过程，或者重置一个已经存在的过程，或者由路由器软件重置 BGP 过程引起的。

一个差错的出现会将 BGP 过程的状态转为空闲状态。路由器可能会自动试图发起另外一个开始事件。但是，对于路由器如何完成这个过程应当加以限制——在具有持续差错的条件下，坚持不懈地试图重新开始会导致摆动。因此，在第一次转回到空闲状态以后，路由器会启动重试连接(ConnectRetry)计时器，当计时器终止以后，路由器就会放弃重新开始 BGP。Cisco 最初的 ConnectRetry 时间是 60 秒。下一次 ConnectRetry 时间是前一次的两倍，依次类推。这也就是说，顺序等待的时间将成指数增加。

2．连接状态

在这种状态下，BGP 过程会等到 TCP 连接完成以后再决定后续的动作。如果 TCP 连接建立成功，BGP 连接将 ConnectRetry 清零，完成初始化过程，给邻居发送一个 Open 消息并转移到发送 Open 消息状态。如果 TCP 连接建立失败。BGP 过程会继续监听由邻居发起的连接、重置 ConnectRetry 计时器并转移到激活状态。

如果在连接状态下，ConnectRetry 计时器超时了，计时器将重新开始计时，并再一次试图与邻居建立一个 TCP 连接，BGP 状态继续保持在连接状态。任何一个其他输入事件的出现都会导致 BGP 状态转移到空闲状态。

3．激活状态

在这个状态下，BGP 过程试图与邻居建立一个 TCP 连接。如果 TCP 连接建立成功，BGP 过程将 ConnectRetry 计时器清零，完成初始化工作，给邻居发送一个 Open 消息并转移到发送 Open 消息状态。Hold 计时器被置成 4 分钟。

如果 BGP 在激活状态时，ConnectRetry 计时器超时，该过程回到连接状态并且重置 ConnectRetry 计时器。它同样也发起一个到对等的 TCP 连接并且继续监听来自对等的连接。如果邻居试图与一个未知的 IP 地址建立 TCP 会话，那么 ConnectRetry 计时器会被重置，连接被拒绝并且本地过程保持在激活状态。任何一个输入事件(除了开始事件，在激活状态下，该事件会被忽略)都会导致状态转移到空闲。

4．发送 Open 消息状态

在这种状态下，已经发送了 Open 消息，BGP 正在等待从邻居发来的 Open 消息。当收到一个 Open 消息以后，检查该消息所有的字段。如果发现了差错，会给它的邻居发送一个 Notification 消息并且将状态转移到空闲。

如果在接收到的 Open 消息中没有发现差错，BGP 给邻居发送一个 Keepalive 消息并且将 Keepalive 计时器置位。邻居之间协商一个 Hold 时间，它们会选用较小的值。如果协商的 Hold 时间是 0，则没有启动 Hold 和 Keepalive 计时器。根据对等的 AS 号，决定该连接是内部的还是外部的，并且将状态转移到 Open 消息确认。

如果收到了一个 TCP 断开消息，本地过程断开 BGP 连接，重置 ConnectRetry 计时器，

开始监听将要由邻居发起的新的连接并将状态转移到激活。任何输入事件(除了开始事件，在激活状态下，该事件会被忽略了)都会导致状态转移到空闲。

5．Open 消息确认状态

在这种状态下，BGP 过程会等待一个 Keepalive 或者 Notification 消息。如果收到 Keepalive 消息，转移到已建立状态。如果收到 Notification 消息或者 TCP 断开消息，状态转移到空闲。

如果 Hold 计时器超时，检测到一个差错或者出现了一个 Stop 事件，BGP 过程会给邻居发送一个 Notification 消息并且断开 BGP 连接，将状态转到空闲。

6．已建立状态

在这种状态之下，BGP 对等之间的连接完全建立起来了，对等之间可以交换 Update、Keepalive 和 Notification 消息。如果收到 Updae 或者 Keepalive 消息，重新启动 Hold 计时器(如果协商的 Hold 时间是非零)。如果收到 Notification 消息，状态会转移到空闲。任何其他的事件都会导致 Notification 消息的发送并且将状态转移到空闲。

2.3.3　路径属性

路径属性是已公布的 BGP 路由的一个特点。一些路径属性我们很熟悉，例如目的地 IP 地址和下一跳路由器，因为这是所有路由的共同特点。其他的，例如 ATOMIC_AGGREGAT，是 BGP 特有的，你可能不熟悉。除了提供基本路由功能的必须的信息，路径属性允许 BGP 设置和互通路由策略。

每一个路径属性可能是下面四种中的一种：

- 公认必选
- 公认自选
- 任选可透明传送
- 任选非可透明传送

从这四种类别的名字可以看出这存在着两个子集，每个子集都有它们自己的子集。首先，一个属性可以为*公认*的，这就意味着所有的 BGP 执行都必须识别它，或者它可以是*任选*的，意味着不要求 BGP 一定要支持该属性。

公认属性可以是强制性的，意味着所有的 BGP Update 消息都要包括该属性，或者该属性可以是任选的，也就是说在规定的更新消息中既可以包括它也可以不包括它。

如果一个任选属性是可传递的，即使 BGP 过程不支持该属性，它也应当接受包含该属性的路由并且把这个路径传送给它的对端。

如果一个任选属性是不可传递的，不识别该属性的 BGP 过程可以忽略包含这个属性的 Update 消息并且不向它的对端公布这条路径。

表 2-5 列出了路径属性，下面的章节会描述每种属性的使用情况。在第 3 章“BGP 的配置和故障排除”中会说明对路径属性的配置、过滤和操作。

1．ORIGIN 属性

ORIGIN 是一个公认必选的属性，它明确了路由更新消息的来源。当 BGP 有多条路由时，它会将 ORIGIN 当作一个决定较优路由的因素。它规定了下面几种源：

- **IGP**——从发起者 AS 的一个内部协议可以学习到网络层可到达信息(NLRI)。一个 IGP 源会得到 ORIGIN 值中最佳的选项。如果 BGP 路由是从 IGP 路由表通过 **network** 命令学

习到的，那么如第 3 章所描述的，这些 BGP 路由会传给一个 IGP 源。

- **EGP**——NLRI 是从 EGP 学习到的。相对于 IGP，EGP 是第二选择。
- **Incomplete**——NLRI 是通过其他手段学习到的。Incomplete 是 ORIGIN 值的最低的选择。Incomplete 并不代表路由在任何情况下都有故障，只代表决定路由来源的信息不完整。BGP 通过再分发学习到的路由会携带不完整的源属性，因为在这种情况下无法决定路由的初始源。

表 2-5　　路径属性*

属　性	类　别
ORIGIN	公认必选
AS_PATH	公认必选
NEXT_HOP	公认必选
LOCAL_PREF	公认自选
ATOMIC_AFFREGATE	公认自选
AGGREGATOR	任选可透明传送
COMMUNITY	任选可透明传送
MULTI_EXIT_DISC(MED)	任选非可透明传送
ORIGINATOR_ID	任选非可透明传送
CLUSTER_LIST	任选非可透明传送

*实际上，除了表 2-5 列出的属性之外还有一些属性，但是因为不仅 RFC 1771 没有规定而且 Cisco 也不支持这些属性，因此它们不在本书的范围之内。

2．AS_PATH 属性

AS_PATH 是一个公认必选(Well-Known Mandatory)属性，它用 AS 号的顺序来描述 AS 间的路径或者到 NLRI 所明确的目的地的路由。当一个运行 BGP 的路由器发起一条路由——当它在自己的 AS 域内公布一个有关目的地的 NLRI——它将自己的 AS 号加到 AS_PATH 中。当后续的运行 BGP 的路由器向外部的对端公布路由，它们将自己的 AS 号附加到 AS_PATH 中(参见图 2-21)。结果是 AS_PATH 可以描述所有它经过的自治系统，以最近的 AS 开始，以发起者的 AS 结束。

注意，只有将 Update 消息发送给在另外一个 AS 域内的邻居时，BGP 路由器才将它的 AS 号加到 AS_PATH 中。也就是说，只有在两个 EBGP 对等实体之间公布路由时，AS 号才被附加到 AS_PATH 中。如果路由是在 IBGP 对等实体之间公布——对等实体在相同的自治系统内——不加入任何 AS 号。

通常，在列表中有同一个 AS 号的多个实例是没有意义的，并且可能会破坏 AS_PATH 属性的目的。但是，在一种情况下，将一个特殊 AS 号的多个实例加入到 AS_PATH 属性中被证

明是有用的。你是否还记得在前面我们曾说过公布的出路由会直接影响到入业务量。在图 2-21 中，通常从 NAP 到 AS 100 的路由是经过 AS 300，因为该路由的 AS_PATH 短于其他路由但是如果到 AS 200 的链路是 AS 100 入业务量首选的路径呢？沿着(500,200,100)路径的链路可能都是 DS3，但是沿着(300,100)路径的链路只是 DS1。再或者 AS 200 是主用供应商，而 AS 300 只是备用的供应商。出业务量发送到 AS 200，当然希望入业务量也使用同样的路径。

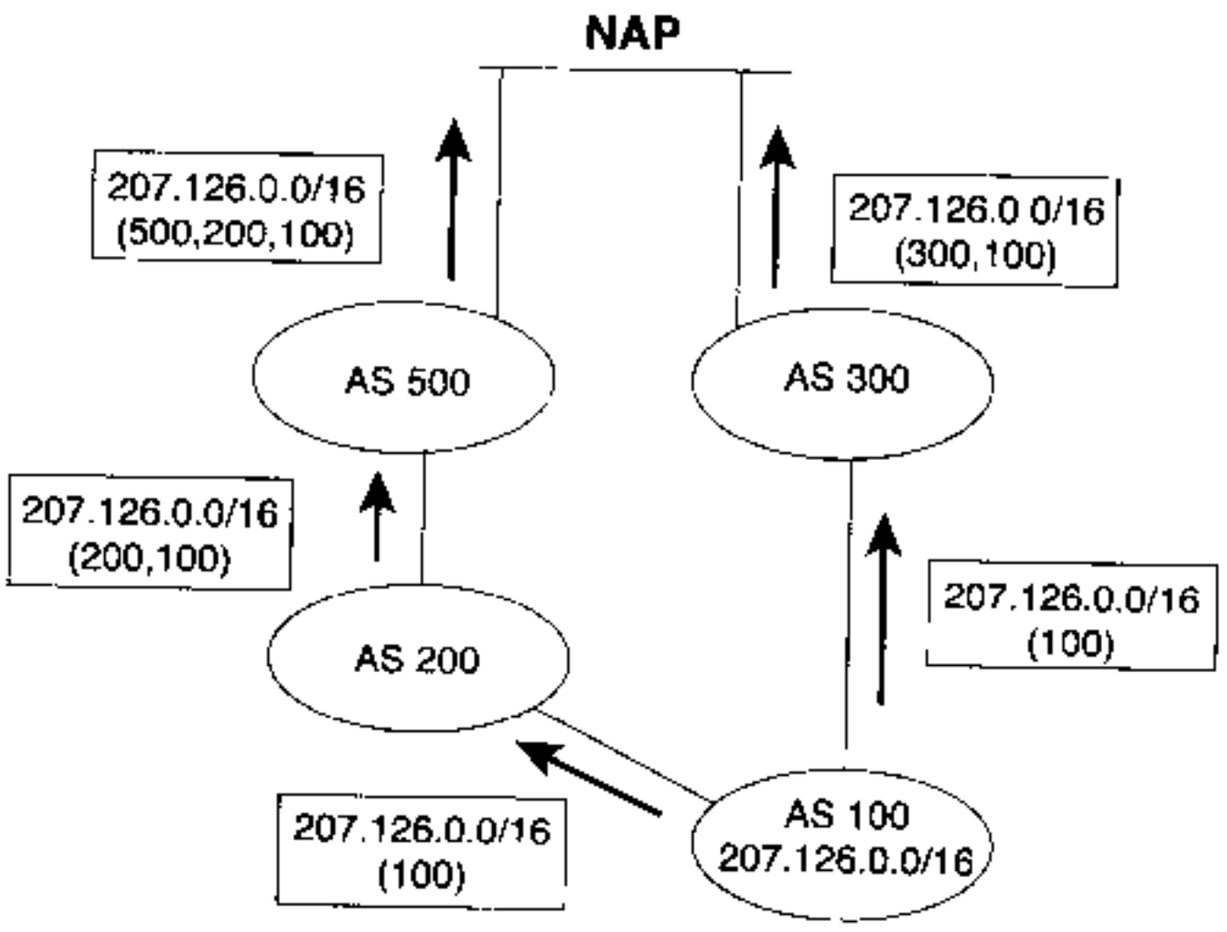


图 2-21　将 AS 号附加(加在该属性的前面)在 AS_PATH 属性中

AS 100 可以通过改变它公布路由的 AS_PATH 来影响它的入业务量(见图 2-22)。AS 100 通过在发送到 AS 300 的列表中加入自己 AS 号的多个实例，使得 NAP 处的路由器认为(500,200,100)路径是较短的路径。在 AS_PATH 中加入额外的 AS 号的过程被称做 AS 通路附加。

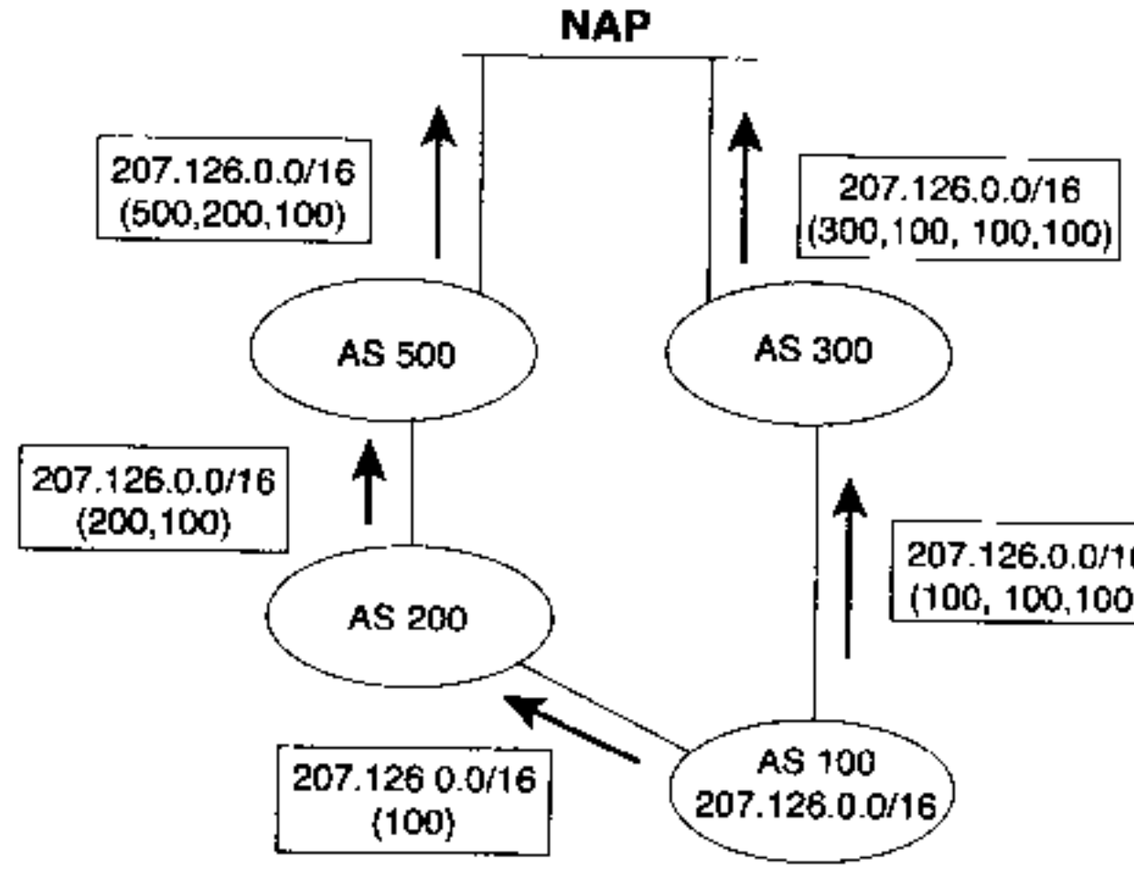


图 2-22　AS 100 用它自己 AS 号的多个实例开始公布给 AS 300 的 AS_PATH 属性

AS_PATH 属性的另一个功能就是避免路由环路，这一点我们已经在本章的前面讲过了。这个机制十分简单：如果 BGP 路由器从它的外部对端处收到了一条路由，而该路由的

AS_PATH 包含这个 BGP 路由器自己的 AS 号，于是该路由器就知道是条环路路由，从而将这样的路由丢弃掉。

3. NEXT_HOP 属性

正如名字所暗示的，该公认必选属性描述了到公布目的地的路径下一跳路由器的 IP 地址。由 BGP NEXT_HOP 属性所描述的 IP 地址不经常是邻居路由器的 IP 地址。要遵循下面的规则：

- 如果正在进行路由宣告的路由器和接收的路由器在不同的自治系统中(外部对等)，NEXT_HOP 是正在宣告路由器接口的 IP 地址。
- 如果正在进行路由宣告的路由器和接收的路由器在同一个 AS(内部对等)内，并且更新消息的 NLRI 指明目的地也在同一个 AS 内，那么 NEXT_HOP 就是已宣告路由的邻居的 IP 地址。
- 如果正在宣告的路由器和接收的路由器是内部对等实体，并且更新消息的 NLRI 指明目的地在不同的 AS，则 NEXT_HOP 就是学习到路由的外部对等实体的 IP 地址。

图 2-23 说明了第一条规则。这里，正在公布的路由器和接收的路由器在不同的自治系统内。NEXT_HOP 是外部对等的接口地址。迄今为止，这种行为和任何其他的路由协议都是相同的。

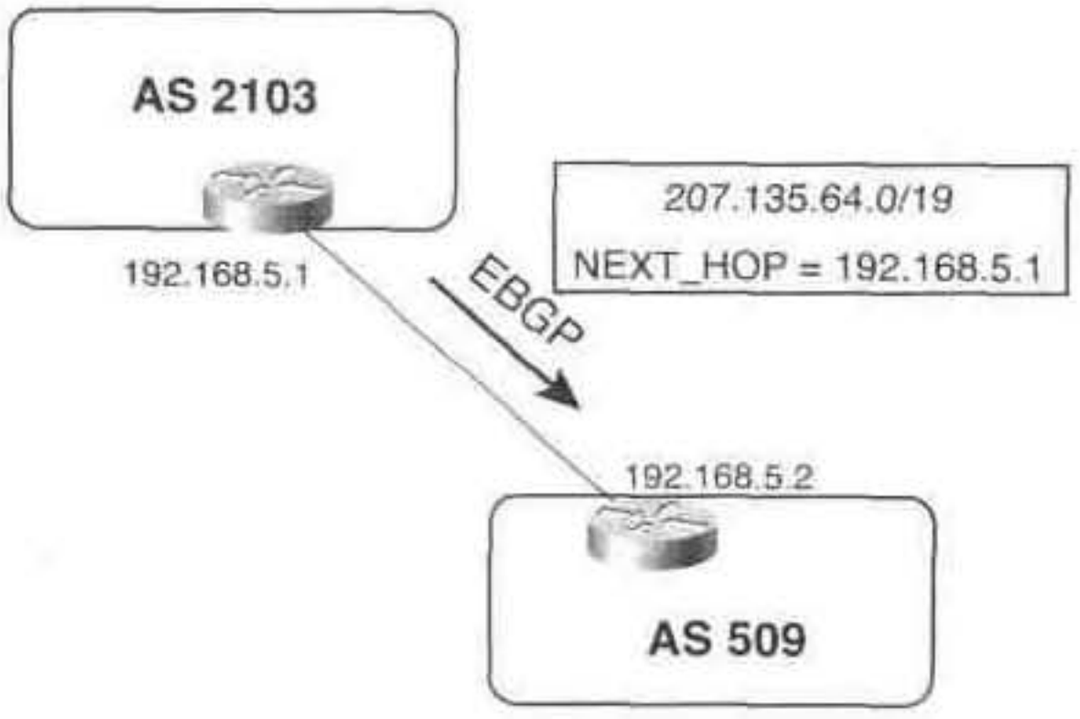


图 2-23　如果通过 EBGP 公布 BGP Update，NEXT_HOP 是外部对等的 IP 地址

图 2-24 说明了第二条规则，如图所示，如果通过 IBGP 宣告 BGP Update 消息而且公布的目的地也在同一个 AS 内，NEXT_HOP 的属性就是发起路由器的 IP 地址。正在进行路由宣告的路由器和正在接收的路由器在相同的 AS 内，正在被宣告的目的地也在同一个 AS 内。与 NLRI 有关联的 NEXT_HOP 是发起路由器的 IP 地址。

注意宣告和接收路由器并没有共享一条通用数据链路，但是 IBGP 的 TCP 连接却通过一个运行 IGP 的路由器。我们将在“IBGP”一节中仔细讨论这个问题。现在：重要的一点就是：为了发送数据包到公布的目的地，接收路由器必须执行一个循环路由查找过程(循环查找在《TCP/IP 路由技术 第 1 卷》中讨论过)。首先，它要查找目的地 172.16.5.30；该路由指示下一跳为 172.16.83.2。因为该 IP 地址不属于与子网相连的路由器中的任何一个，路由器必须查找到 172.16.83.2 的路由。这条通过 IGP 学习到的路由，指示下一跳为 172.16.101.1。现在数据包就被转发出去。这个例子对于理解 IBGP 对 IGP 的依赖非常重要。

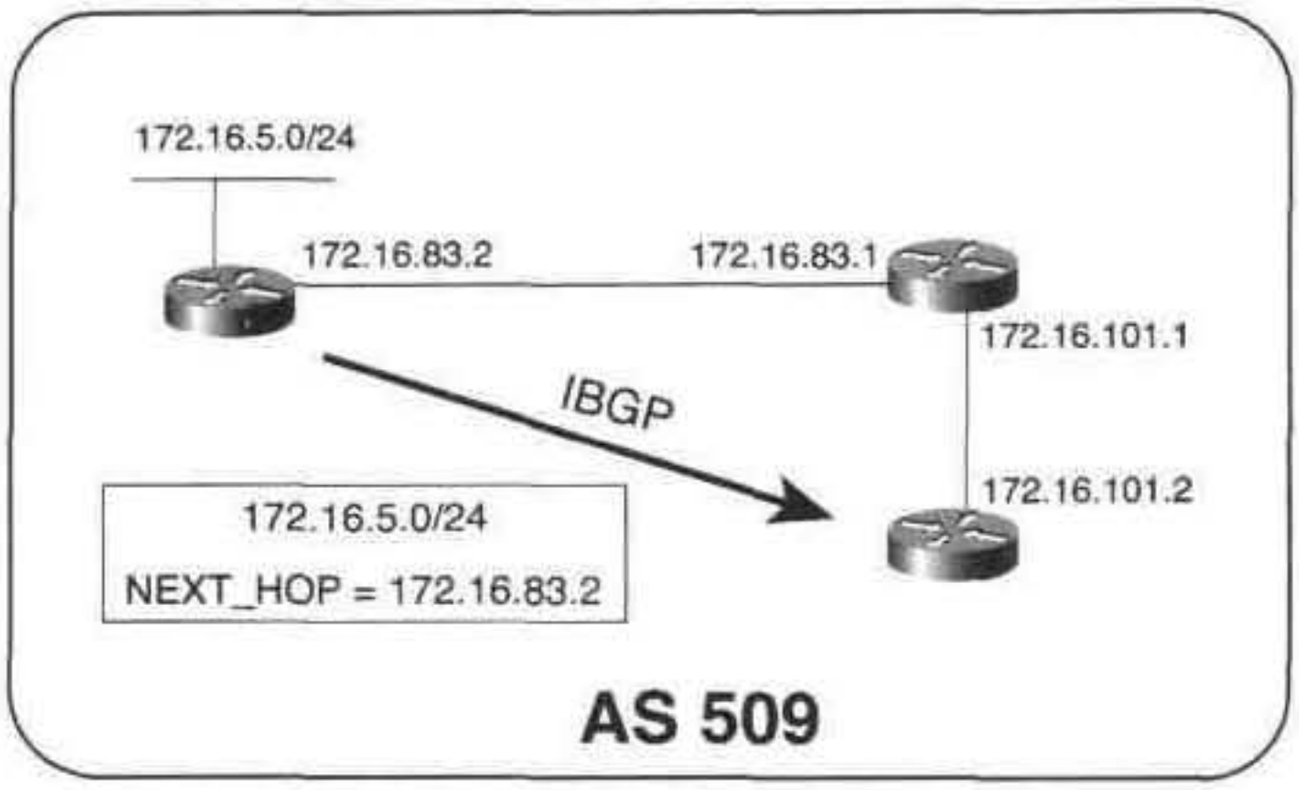


图 2-24　NEXT_HOP 的属性就是发起路由器的 IP 地址

图 2-25 说明了第 3 条规则，如果一条路由 BGP Update 是通过 IBGP 公布并且目的地在不同的 AS 域内，NEXT_HOP 属性就是学习到该路由的外部对等的 IP 地址。在这里，路由通过 EBGP 学习到并且将它传递给内部对等。因为目的地与公布、接收路由器在不同的 AS 域内，通过 IBGP 连接传递的路由的 NEXT_HOP 是学习到路由的外部路由器的接口地址。

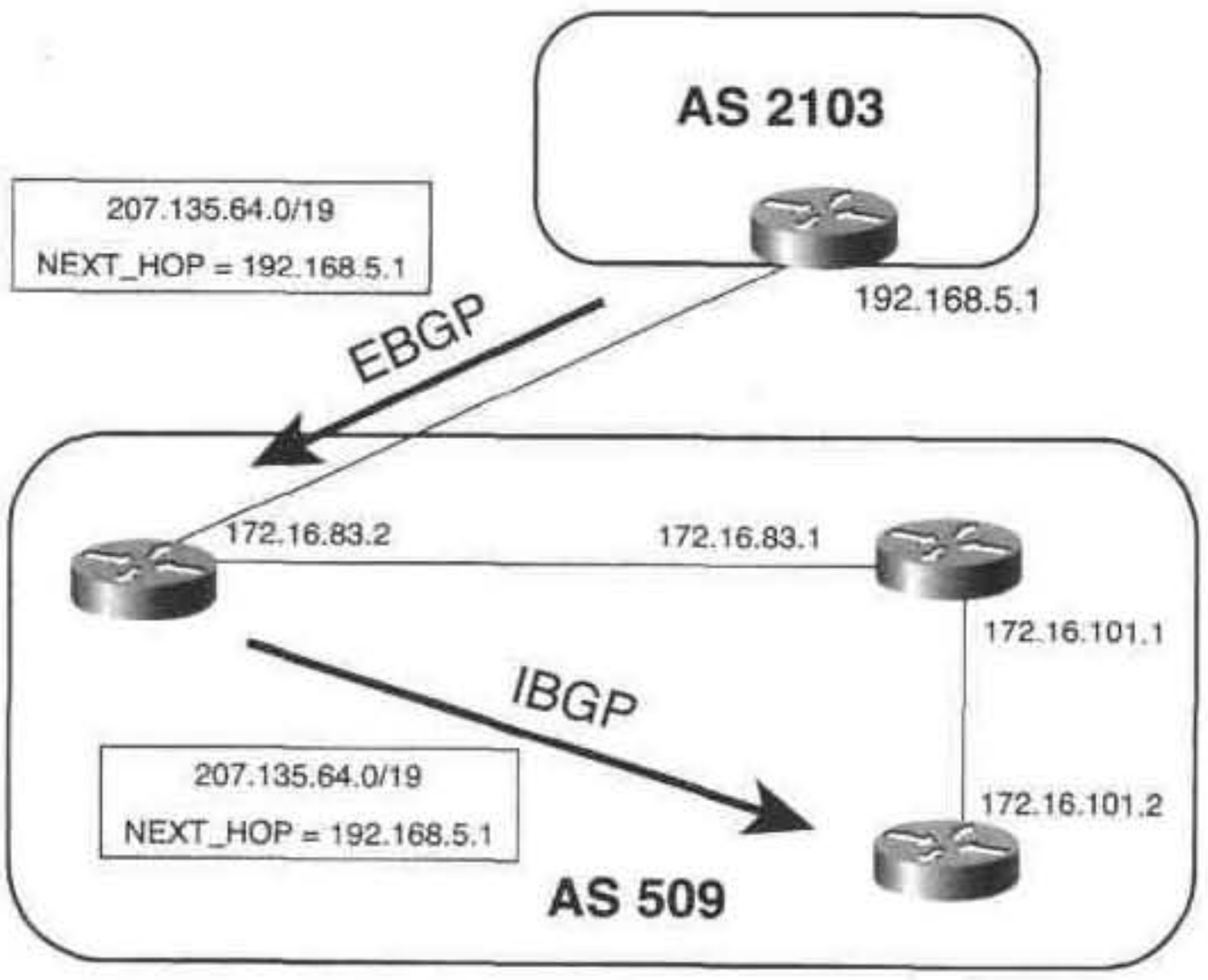


图 2-25　NEXT_HOP 属性就是在该处学习到路由的外部对等的 IP 地址

在图 2-25 中，为了将数据包转发到 207.135.64.0/19，IBGP 对等必须执行一个循环路由查找。但是，这里存在一个潜在的问题。下一跳地址所属的网络 192.168.5.0 不是 AS 509 的一部分。除非 AS 边界路由器将该网络公布给 AS 509，否则 IGP——此时也就是内部对等实体——将不会知道该网络。而且如果该网络不在路由表内，207.135.64.0/19 的下一跳地址就是不可到达的，因此到那个目的地的数据包就会被丢弃。实际上，虽然在内部对端的 BGP 路由表中建立了到 207.135.64.0/19 的路由，但是因为对于该路由器来讲，下一跳的地址是无效的，因此 IGP 路由表中并没有建立该路由。

对于这个问题第一种解决方案当然就是保证内部路由器知道与两个自治系统相连的外部网络。你可以使用静态路由的办法，但是实际的做法是在外部端口上以被动模式运行 IGP。但在某些情况下，该方法并不理想。第二种解决办法就是采用配置选项使得 AS 509 中的 AS 边界路由器将它自己的 IP 地址放置到 NEXT_HOP 属性中，从而取代外部对等的 IP 地址。于是内部对等实体就有了一个下一跳路由器地址 172.16.83.2，IGP 知道该地址。这种配置选项被称做 **next-hop-self**，将在第 3 章中进行描述。

4. LOCAL_PREF 属性

LOCAL_PREF 是本地首选项的简写。这个公认自选属性只用在内部网关对端之间的更新消息中；它不会传递给其他的自治系统。该属性用于对一条已公布路由的 BGP 路由器的首选项等级进行交流。如果一个内部运行 BGP 的路由器收到了到一个目的地的多条路由，它将这些路由的 LOCAL_PREF 属性进行比较。选择具有最高的 LOCAL_PREF 的路由。

图 2-26 显示了如何使用 LOCAL PREF 属性。AS 2101 从两个 ISP 获得路由，但是 ISP 1 为首选服务提供商。连接到 ISP 1 路由器宣告路由的 LOCAL_PREF 为 200，连接到 ISP 2 的路由器宣告的路由的 LOCAL_PREF 为 100(默认值)。所有的内部对等实体，包括连接到 ISP2 的路由器，对于同一目的，更优先选用从 ISP 1 学到的路由。

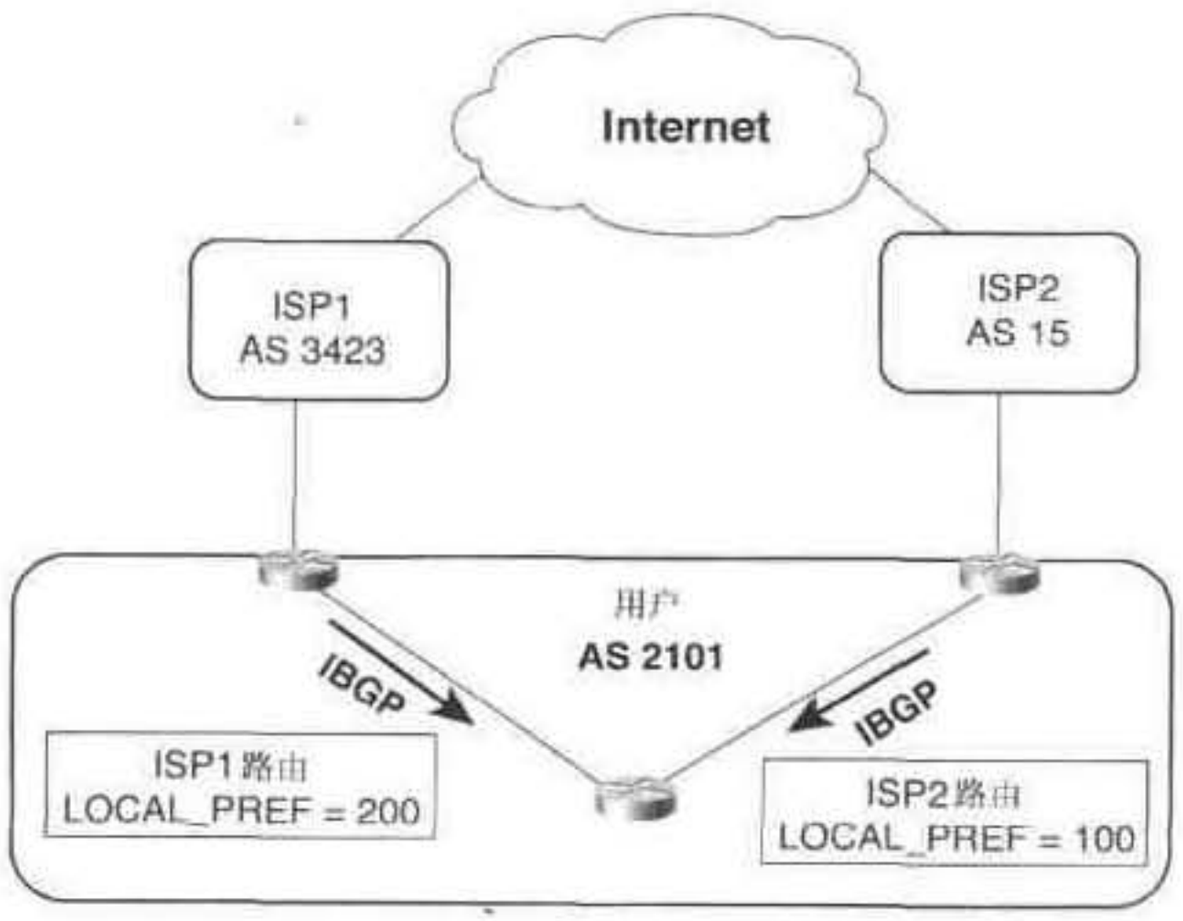


图 2-26　具有较高 LOCAL_PREF 值的路由为首选

5. MULTI_EXIT_DISC 属性

LOCAL_PREF 属性只影响离开 AS 域的业务量。如果想影响入业务量，应该使用 MULTI_EXIT_DISC 属性，它的简写是 MED。这个任选非传递属性于 EBGP 的 Update 消息中携带，它允许一个 AS 将它首选的入口点通知另外一个 AS。如果其他的参数都相同，收到到同一个目的地的多条路由的 AS 将这些路由的 MED 进行比较。与选用最高 LOCAL_PREF 值的路由不同，具有最低 MED 值的路由是首选。这是因为将 MED 看作是一个度量，并且最低的度量——最短的距离——是首选。

注：　在 BGP-2 和 BGP-3 中 MULTI_EXIT_DISC 属性被称做 AS 域间度量。

图 2-27 说明了如何使用 MED。在这里，一个用户到一个 ISP 是双宿主(dual-homed)结构。

AS 525 偏向于入业务量使用 DS-3 链路，而 DS-1 链路只用于备用。通过 DS-3 传送的更新消息中，MED 被置为 0(缺省值)，而通过 DS-1 传送的更新消息中，MED 被置为 100。如果这两条路由没有其他不同的选项，ISP 就会优先选择具有较低 MED 值的 DS-3 链路。

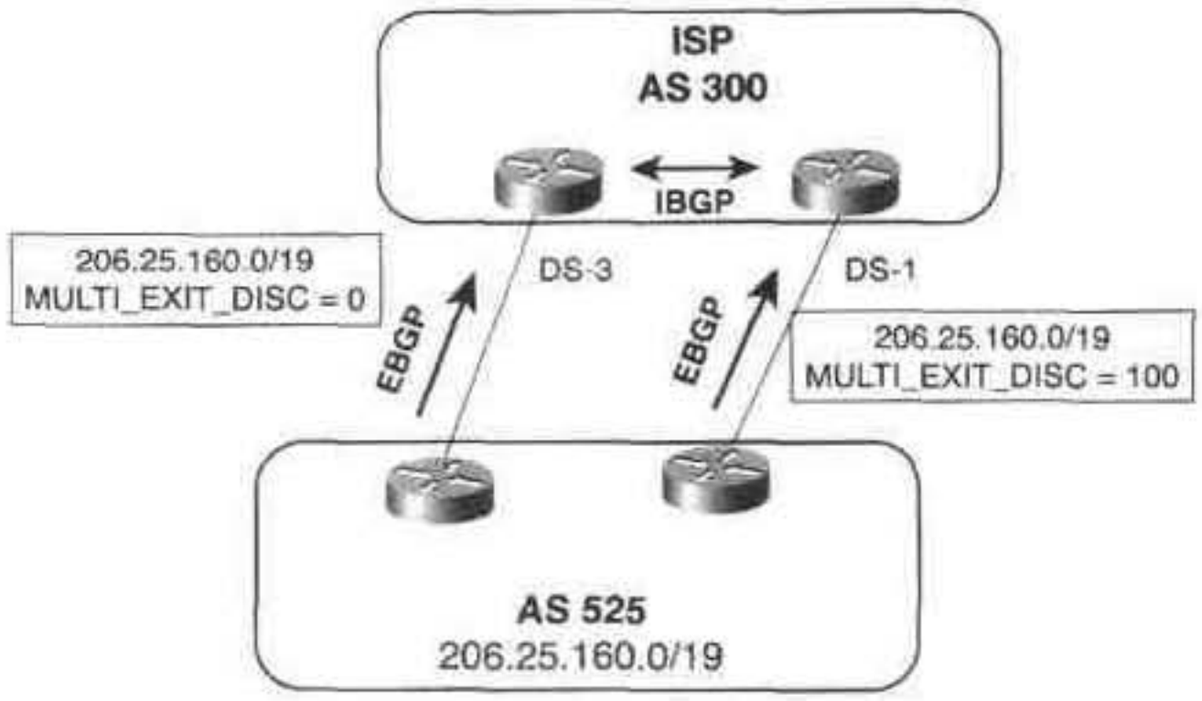


图 2-27　与在 DS-3 链路上传递的路由有关的较低的 MED 值使得 ISP 优选这条链路

注意，在 ISP 内，路由器之间使用 IBGP。在内部的对等实体之间传递来自 AS 525 的 MED，从而使它们都知道优选的路由。但是，不会在接收 AS 范围之外的地方传递 MED。例如，如果 ISP 向另外一个 AS 公布 206.25.160.0/19，它不会将发起 AS 所置的 MED 传递给那个 AS。这也就是说，MED 只是在直接相连的自治系统间影响业务量。为了影响相邻 AS 以外的路由选项，就像本节前面所描述的，必须对 AS_PATH 属性进行处理。

如果到同一个目的地的两条路由来自两个不同的自治系统，也不进行 MED 值的比较。例如，如果图 2-27 中的 ISP 不仅从 AS 525 收到关于 206.25.160.0/19 的路由宣告，而且还从另外一个 AS 收到关于它的路由宣告，那么来自不同的自治系统的 MED 不会进行比较。MED 只对单一的 AS 有意义，当该 AS 有多个入口点时，它用来说明首选项的等级。

6. ATOMIC_AGGREGATE 和 AGGREGATOR 属性

一个运行 BGP 的路由器能够向另外一个运行 BGP 的路由器传送重叠的路由。重叠路由是一些指向同一个目的地的不完全相同的路由。例如，在图 2-28 中，路由 206.25.192.0/19 和 206.25.128.0/17 是重叠路由。虽然第二条路由除了指向 206.25.192.0/19 以外，还指向其他的路由，但是第一条路由包含在第二条路由中。

在做最好路径的决定时，路由器通常选择更具体的路径。但是，在公布路由时，运行 BGP 的路由器有几种处理重叠路由的选项：

- 同时公布具体和不太具体的路由
- 只公布具体的路由
- 只公布路由中没有重叠的部分
- 聚合这两条路由并且公布聚合后的路由
- 两者都不公布

在这之前，我们曾经强调过，在执行归纳(路由聚合)时，会丢失一些路由信息，而且路由会变得不太准确。当在一个运行 BGP 的路由器中执行聚合时，所丢失的信息是路径的细节。图 2-28 说明了路径细节的丢失。

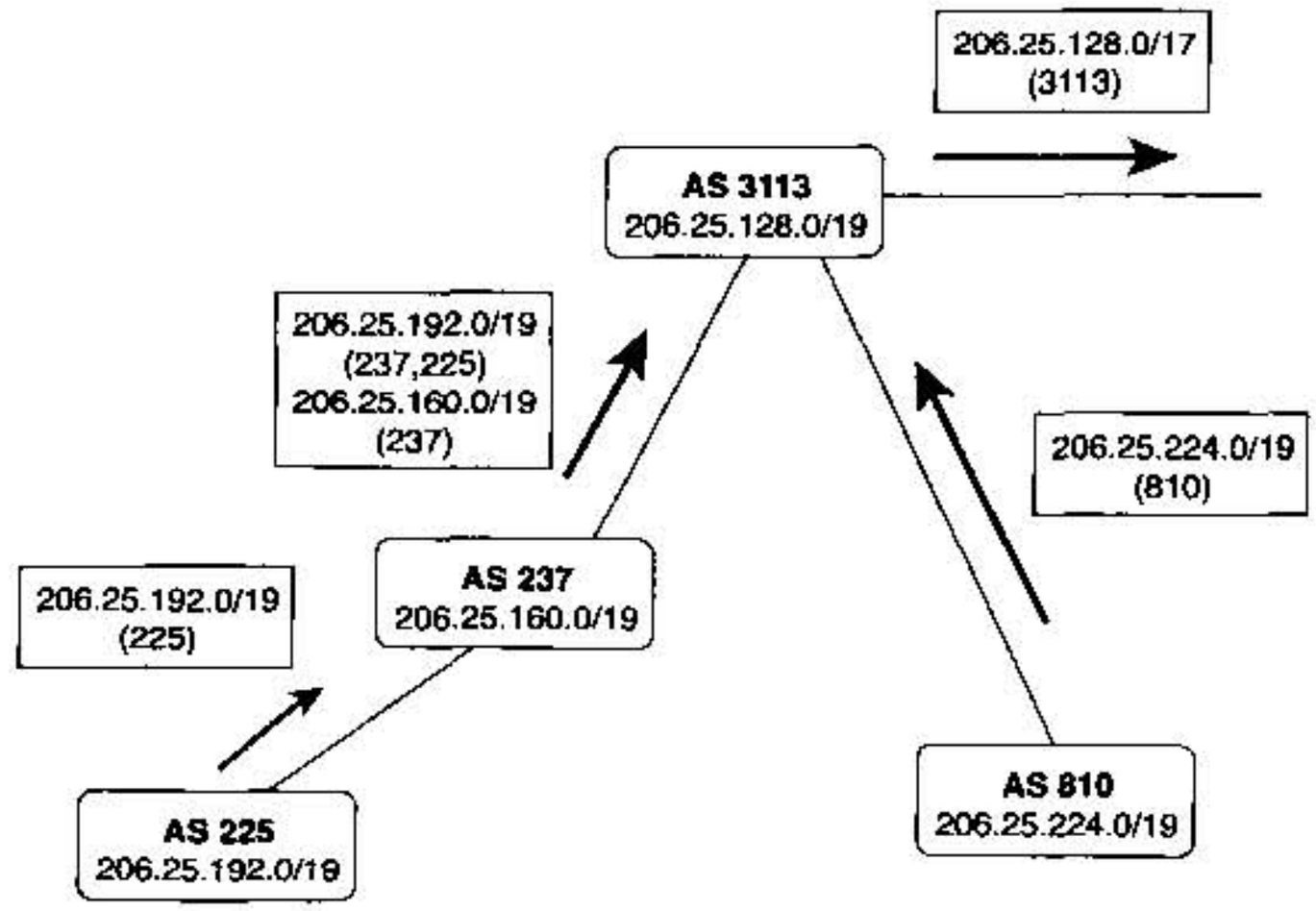


图 2-28　聚合 BGP 路由导致路径信息的丢失

AS 3113 正在公布一个代表几个自治系统内地址的聚合地址。因为该 AS 是聚合的发起者，因此在 AS_PATH 中只包含它自己的 AS 号。由该聚合代表的一些更具体的前缀的路径信息就丢失了。

ATOMIC_AGGREGATE 是一个公认必选的属性，它用来警告下游路由器出现了路径信息丢失。任何时候，当一个运行 BGP 的路由器将更具体的路由归纳为有较少细节的聚合路由(前面所述列表的第 5 选项)，而且已经出现了路径信息的丢失时，运行 BGP 的路由器必须将 ATOMIC_AGGREGATE 属性附加到聚合路由中。任何一台运行 BGP 的下游路由器收到有 ATOMIC_AGGREGATE 属性的路由，不能使这条路中的 NLRI 信息更详细。并且当把该路由公布给其他的对端时，ATOMIC_AGGREGATE 属性必须继续附加在该路由中。

当设置了 ATOMIC_AGGREGATE 属性，运行 BGP 的路由器可以选择附加 AGGREGATOR 属性。这个任选可透明传送属性包括发起聚合路由的路由器的 AS 号以及 IP 地址，从而提供了执行聚合的地点的信息(如图 2-29 所示，ATOMIC_AGGREGATE 属性指示出现了路径信息的丢失，AGGREGATOR 属性指示聚合出现在哪里)。Cisco BGP 的执行是在该属性中插入代替 BGP 路由器 IP 地址的路由器 ID 号。

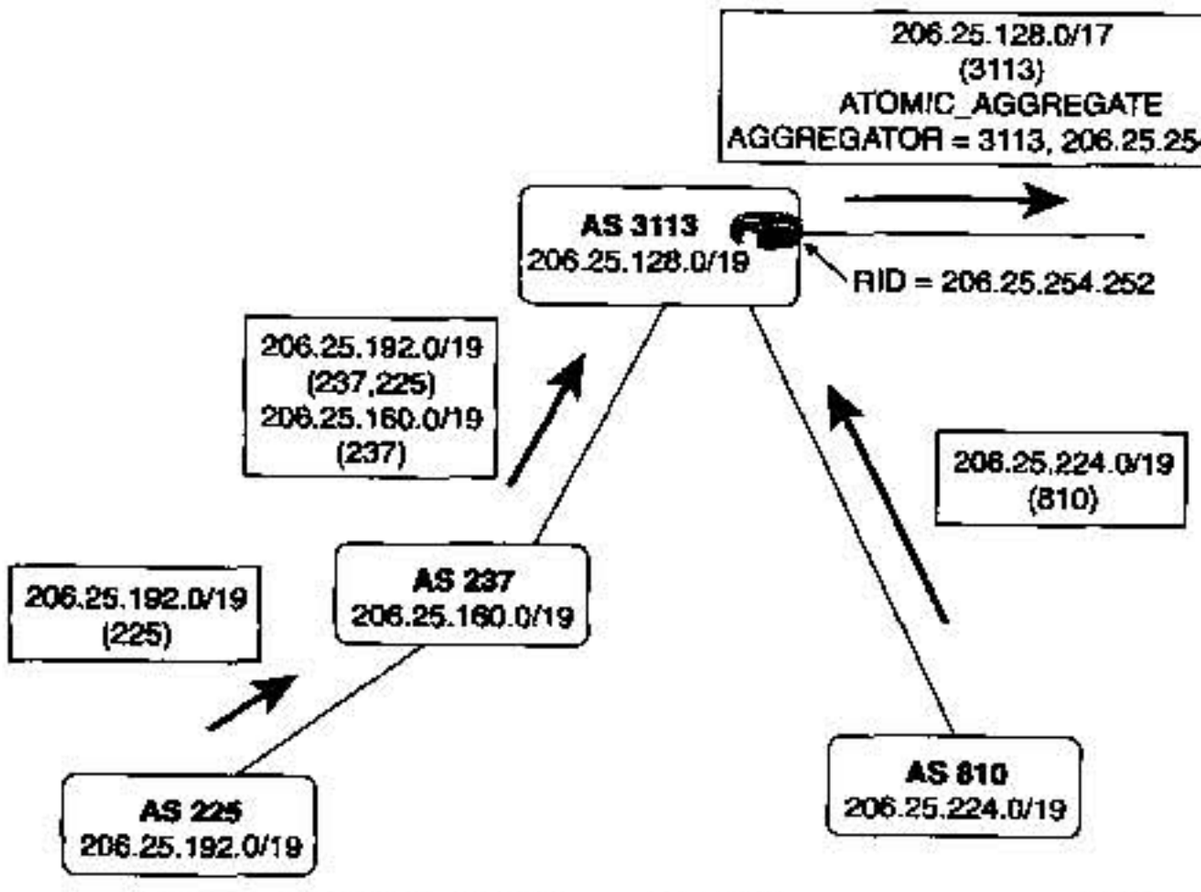


图 2-29　ATOMIC_AGGREGATE 属性与 AGGREGATOR 属性

7. COMMUNITY 属性

COMMUNITY 是一个任选可透明传送属性，它可以简化策略的执行。最初，它是 Cisco 特有的一个属性，现在在 RFC 1997[8] 中已被标准化。COMMUNITY 属性标明一个目的地作为一些目的地团体中的一个成员，这些目的地共享一个或者多个共同的特性。例如，一个 ISP 可能会为它所有用户的路由分配一个特殊的 COMMUNITY 属性。于是 ISP 就可以在 COMMUNITY 值的基础上设置它的 LOCAL-PREF 和 MED，而不是为每一条路由单独设置。

COMMUNITY 属性有四个字节。RFC 1997 规定前两个字节代表自治系统，后两个字节是管理上定义的表示符，格式是 AA：NN。Cisco 缺省的格式是 NN：AA。你可以通过 **ip bgpcommunity new-format** 命令将 Cisco 缺省的格式改为 RFC 1997 的格式。

例如，假设来自 AS 625 的一条路由有一个 COMMUNITY 标识符 70。AA：NN 格式下的 COMMUNITY 属性是 625：70，以 16 进制表示为两个相连的数：0x02710046。在这里 625=0x0271，70=0x0046。RFC 用 16 进制表示该属性，而 Cisco 路由器使用十进制来表示 COMMUNITY 属性。例如，625：70 用十进制表示是 40960070(16 进制的 0x02710046 用十进制表示就是 40960070)。

在团体的值中从 0(0x00000000)～65535(0x0000FFFF)和从 4294901760(0Xffff0000)～429467295(0xFFFFFFFF)是预留的。在这个预留范围外，定义了几个众所周知的团体：

- **INTERNET**——Internet 团体没有一个确定的值；所有属于这个团体的路由都有一个缺省值，可以自由地公布属于这个团体的路由。
- **NO_EXPORT(4294967041 或者 0xFFFFFF01)**——接收到的携带该值的路由不能公布给 EBGP 对等实体，或者如果配置了一个联盟，该路由不能在联盟范围以外公布(联盟在后面的章节“管理大型 BGP 对等关系”中有定义)。
- **NO_ADVERTISE(4294967042 或者 0xFFFFFF02)**——接收到携带该值的路由不能公布给 EBGP 或者 IBGP 的对等实体。
- **LOCAL_AS(4294967043 或者 0xFFFFFF03)**——RFC1997 称这个属性为 NO_EXPORT_SUBCONFED。不能将接收到的携带该值的路由公布给 EBGP 对等实体，以及在联盟内的其他自治系统内的对等。

第 3 章给出了通过团体帮助执行路由策略的例子。

8. ORIGINATOR_ID 和 CLUSTER_LIST 属性

ORIGINATOR_ID 和 CLUSTER_LIST 属性是可选非传递属性，由路由反射器(route reflector)使用。路由反射器将在“管理大型 BGP 对等关系”一节中进行描述。用这两个属性来防止路由环路。ORIGINATOR_ID 是由路由发起者产生的一个 32 比特的值。该值是本地 AS 里路由发起者的路由器 ID。如果路由发起者在接收到路由的 ORIGINATOR_ID 中发现了自己的 RID，就知道产生了路由环路，于是该路由就被忽略掉。

CLUSTER_LIST 是路由经过的路由反射器簇 ID 的一个序号。如果路由反射器在接收到路由的 CLUSTER_LIST 中发现了自己的本地簇 ID，就知道产生了路由环路，于是该路由就被忽略掉。

2.3.4　管理权值

管理权值是 Cisco 特有的 BGP 参数，只适用于一台路由器中的路由。该参数不会传递给

其他的路由器。这个权值是0~65535之间的一个数。它可以分配给不同的路由。权值越高，该路由的优先选择权就越高。当选择一个最优路径的时候，BGP决定过程会在除了指定的特性之外的所有路由特性之上考虑路由权值。缺省的情况下，从对等学习到的所有路由的权值都是0，由本地路由器产生的所有路由的权值都是32768。

可以为独立的路由或者从一个特定的邻居学习到的路由设置管理权值。例如，对等A和对等B可能会向一个运行BGP的路由器公布同样的路由。通过给从对等A接收到的路由分配较高的权值，可以使运行BGP的路由器优先选择通过该对等的路由。对于单个路由器来讲，这个优选项完全是本地性质的，权值既不包括在BGP更新消息中，也不以任何形式传递给运行BGP的路由器的对等。

2.3.5　AS_SET

AS_PATH属性用一个有规则的AS号序列就可以表示，这个AS序列描述到达一个特定的目的地所经过的路径。实际上，AS_PATH有两种类型：

- **AS_SEQUENCE**——正如前面所描述的，这是AS号的一个有规则的列表。
- **AS_SET**——这是到目的地的路径上AS号的一个无规则列表。

在AS_PATH属性中，这两种类型是通过一个类型编码来区分的，这个区别在“BGP消息格式”中有所描述。

> 注：　实际上，AS_PATH有四种类型。要了解其他两种类型：AS_CONFED_SEQUENCE和AS_CONFED_SET的详细情况，参见“联盟”一节。

我们曾经讲过，AS_PATH一个主要的好处就是防止路由环路。如果一个运行BGP的路由器在接收到的来自外部对等实体的一条路由中发现了它自己的AS号，就知道出现了路由环路并将该路由忽略。但是如图2-28所示，当执行路由聚合时，会丢失AS_PATH的一些细节。结果是，产生路由环路的潜在因素就增加了。

例如，假设图2-28中的AS 810到另外一个AS有可选连接(见图2-30)。来自AS 3113的路由聚合公布给AS 6571，然后从AS 6571又回到了AS 810。

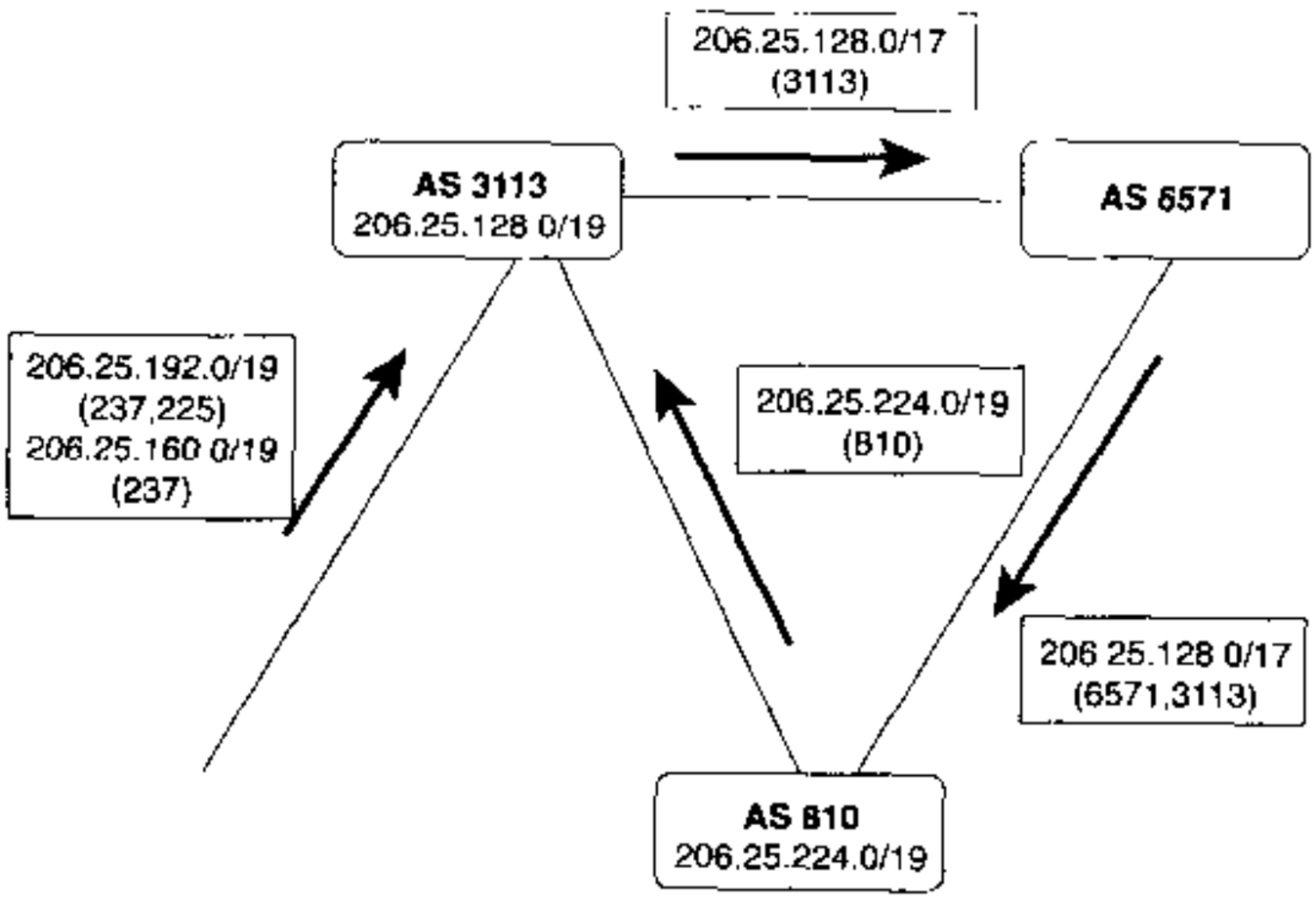


图2-30　一种路由视头的情况

因为在聚合点“后面”的 AS 号没有包括在 AS_PATH 中，AS 810 不会检测到潜在的路由环路。下一步，假设 AS 810 内的一个网络(例如 206.25.225.0/24)出现了故障，在这个 AS 内的路由器会与来自 AS 6571 的聚合路由相匹配，这样就会出现路由环路。

如果你想到了这一点，也就是 AS_PATH 的防止路由环路的功能没有要求包括的 AS 号有任何特殊的顺序，最重要的就是接收路由器应该能够识别它自己的 AS 号是否是 AS_PATH 的一部分。此处就涉及到了 AS_SET。

当一个运行 BGP 的路由器根据从其他自治系统学习到的 NLRI 生成了一个聚合路由，它包括 AS_PATH 中所有的 AS 号并将它们作为一个 AS_SET。例如，在图 2-31 中给出了图 2-28 中的网络，该网络在聚合路由中加入了 AS_SET。

聚合路由器开始了一个 AS_SEQUENCE，因此接收路由器可以沿着路径跟踪到聚合体，但是要加入 AS_SET 以防止路由环路。在这个例子中，你还可以看到为什么 AS_SET 是一个无序的列表。在 AS 3113 中聚合体的后面是到聚合路由所在的自治系统的分支的路径。因此也就没有办法用有序的列表来描述这些单独的路径。

当一个 AS_PATH 中包含了 AS_SET，在聚合路由中没有必要再加入 ATOMIC_AGGREGATE。AS_SET 用于通知下行的路由器产生了聚合，并且它要比 ATOMIC_AGGREGATE 带有更多的信息。

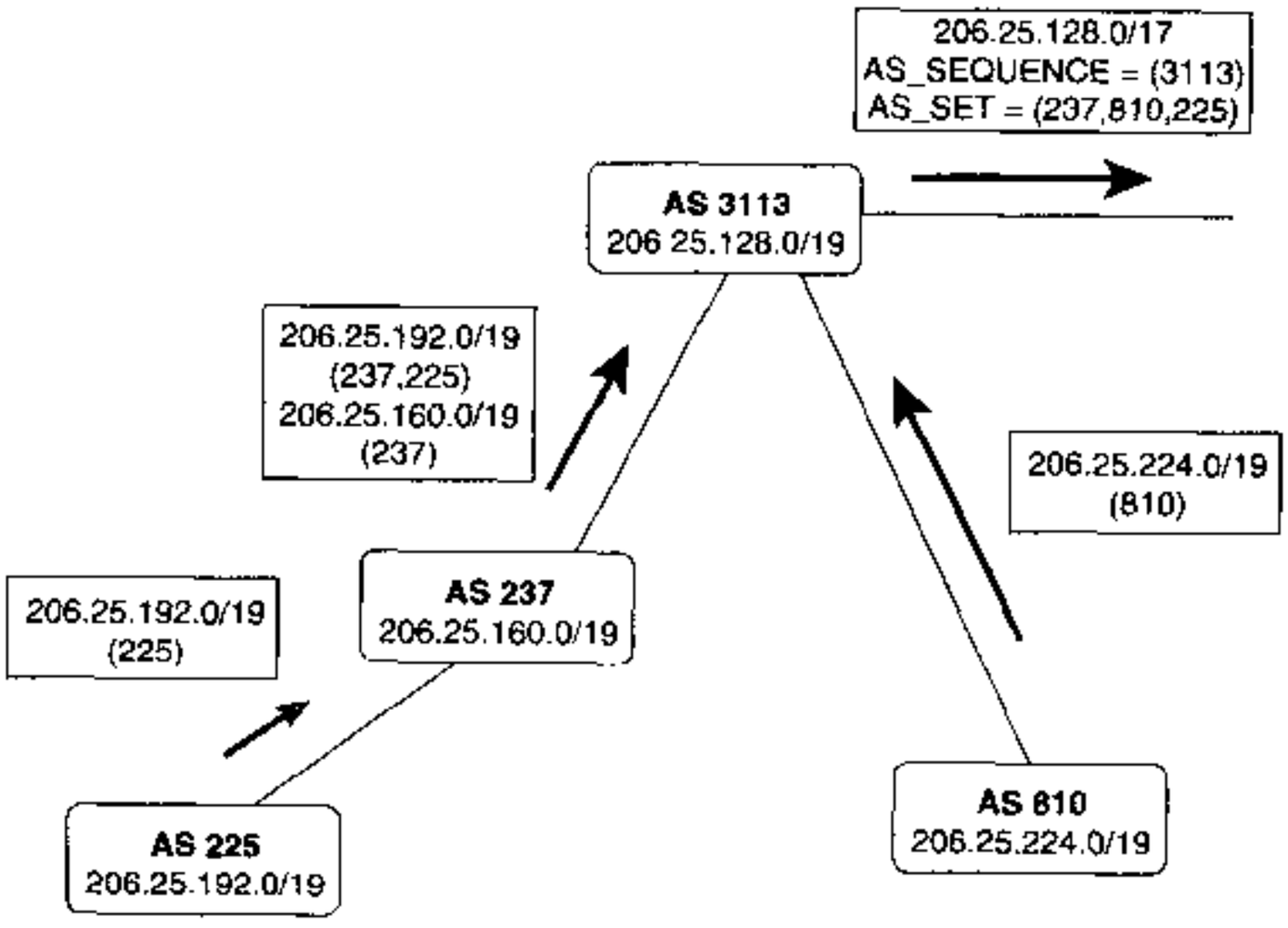


图 2-31　AS_SET 可以恢复在聚合过程中丢失的避免路由环路的信息

就像生活中的许多选择一样，使用 AS_SET 也是有代价的。你已经了解到路由归纳的一个好处就是路由的稳定。如果一个属于聚合的网络出现了故障，故障不会在聚合点以外的范围内公布。如果在 AS_PATH 中加入了 AS_SET，这种稳定性就会降低。如果图 2-31 中到 AS 225 的链路出现故障，例如 AS_SET 发生了变化，那么会在聚合点以外的范围内公布该变化。

2.3.6　BGP 决策过程

BGP 路由信息数据库(RIB)包括 3 个部分：

• **Adj-RIBs-In**——存储那些未经处理的路由信息，这些信息是来自从对等接收到的更新消息。Adj-RIBs-In 所包含的路由是可用路由。

• **Loc-RIB**——包含的路由是运行 BGP 的路由器通过对 Adj-RIBs-In 中的路由使用它的本地路由策略而选择的路由。

• **Adj-RIBs-Out**——包含运行 BGP 的路由器向它的对等公布的路由。

RIB 的这 3 个部分可能是 3 个不同的数据库，或者 RIB 是一个单一的数据库，但是用指针分别指向 3 个不同的部分。

BGP 决定过程是通过对 Adj-RIBs-In 中的路由使用本地路由策略，同时将选定的或者修改过的路由放到 Loc-RIB 和 Adj-RIBs-Out 中而选择路由。这个决定过程需要 3 个步骤：

• 第 1 步，为每一条可用路由计算首选等级。只要是路由器从相邻 AS 处的对端接收到的 BGP Update 消息中包含新的路由、发生变化的路由或者撤消的路由，就要调用该步骤。对每条路由都要分开考虑，得到一个非负整数指示该路由的首选级别。

• 第 2 步，从到特定目的地的所有可用路由中选取最好的路由，并把该路由放到 Loc-RIB 中。只有第一步完成以后才调用该步骤。

• 第 3 步，将合适的路由放到 Adj-RIBs-Out 中，用于向对端宣告。只有在 Loc-RIB 发生变化后，而且只有在第 2 步完成以后才调用该步骤。如果要执行路由集中的话，就在这步执行。

除非一个路由策略有特殊说明，否则第 2 步通常会在到达一个特定目的地的所有可用的路由中选取最具体的路由。如果由路由的 NEXT_HOP 属性所明确的地址是不可到达的，不要选取该路由，记住这一点是非常重要的。对于内部 BGP 来讲，这个情况有不同的分支，我们将在“IBGP 和 IGP 的同步”一节中讨论这个问题。

到现在为止，你应该对这些属性有一个正确的评价，这些属性能够分配给一个 BGP 路由从而在单个路由器内、在内部对等实体、在相邻的自治系统之间以及其他范围内执行路由策略。在使用这些属性的时候，应该考虑顺序和规则，尤其是一台路由器需要在到同一个目的地的多条路由之间进行选择的时候。以下是选路的准则：

1. 首选具有最高管理权值的路由。因为 BGP 管理权值是 Cisco 定义的参数，因此该功能是 Cisco 特有的。

2. 如果权值相同，首选具有最高 LOCAL_PREF 值的路由。

3. 如果 LOCAL_PREF 值相同，首选逻辑上在该路由器上发起的路由。也就是说，首选从同一个路由器上的 IGP 学习到的路由。

4. 如果 LOCAL_PREF 值相同，而且没有逻辑上发起的路由，首选具有最短 AS_PATH 的路由。

5. 如果 AS_PATH 长度相同，首选具有最低原码(ORIGIN CODE)的路由。IGP 低于 EGP，EGP 低于 Incomplete。

6. 如果源代码相同，首选具有最低 MULTI_EXIT_DISC 值的路由。只有当要考虑的所有路由的 AS 号都相同时，才会进行此项比较。

7. 如果 MED 相同，在 EBGP 路由和联盟 EBGP 路由中，首选 EBGP 路由，在联盟 EBGP 路由和 IBGP 路由中首选联盟 EBGP 路由。

8. 如果路由相同，首选到 BGP 下一跳最短的路径。这是一条到下一跳路由器的具有最

低 IGP 度量的路由。

9. 如果路由相同，它们来自相同的相邻 AS 并且通过 **maximum-paths** 命令使 BGP 多条路径可用，那么将所有开销相同的路由安装到 Loc-RIB 中。

10. 如果多条路径不可用，首选具有最低 BGP 路由器 ID 的路由。

2.3.7　路由抑制

路由摆动(route flap)是 Internet 不稳定的主要因素，而且对于其他网络也是一样。当一条有效，路由被宣布为无效，然后又被宣布为有效时就会出现摆动。出现这种情况是很显然的：因为每当路由状态发生改变时，这种变化必须在整个网络内部进行公布，并且每台路由器必须适当地重新计算路由。这样，就要消耗一定的带宽和 CPU 资源。

> 注：　你可能偶尔会听到路由振动(route oscillation)，它和路由摆动互换着使用。但是它们之间有一定的区别。振动是周期性的，而摆动不是。

大多数的人很快想到不稳定的物理链路或者出现故障的路由器接口是导致路由摆动的主要原因，确实，他们是对的。但是另外一个导致路由摆动的很普遍的原因，也许是所有原因中最普遍的，就是人本身。技术人员在 TELCO 中心办公室或者在你的配线室进行一些修复工作，他很有可能会引起断电从而引起路由摆动。但是，你不要忘了，一些没有经验的网络管理者对他的路由器所做的一些无意的动作，例如配置或者故障排除也可能会导致路由摆动。他可能重复地加入、删除一条路由，改变一个接口的状态或者重置一个 BGP 会话。如果将这种变化结果传递给他的 ISP，那么他粗心的工作会影响到整个网络。

不稳定的影响究竟有多大呢？我们可以只考虑单一的过载或者处理能力不足的 BGP 路由器。一条上行的连接变得不稳定，导致许多路由同时摆动。这个路由器不能处理这个变化，于是它失效了。现在，下游路由器不得不既处理最初摆动的路由，还要处理因为这个失效的路由器引起的所有现在不可到达的路由。这种影响迅速增长，蔓延到整个网络，还可能引起更多的路由器失效。这种情况是很难处理的。

你已经了解到路由聚合是如何帮助隐藏不稳定性的。如果聚合中的一条路由失效，聚合本身不会有变化。以这条失效路由为目的地的数据包还是会继续转发到这个聚合地址，聚合的发起者已经知道了这条无效路由，就会将这些数据包丢弃。

但是聚合也不是时时都起作用。例如，一个 ISP 用户可能会有独立于供应商的 IP 地址。因为该地址在供应商地址块的范围之外，那么用户的这个地址必须独立于供应商的聚合地址进行公布。而且在讨论多宿主的时候，我们也提过，当一个用户多宿主到多个供应商时，不能使用聚合。

但是即使一个 ISP 通过聚合用户路由，可以为 Internet 的其他部分提供稳定的路由，在 ISP 本身的 AS 内，这种聚合并没有为路由的稳定性起到什么作用。路由摆动仍然会影响到聚合点以后的所有路由器。

路由抑制的产生是为了阻止不稳定的路由在整个网络内扩散。它虽然不能阻止一个路由器接收不稳定的路由，但是它能够阻止公布不稳定路由。虽然路由抑制已经出现一段时间了，但是只是最近才在 RFC 2439 中将它定型(www.isi.edu/in-notes/tr.rfc2439.txt)。

使用路由抑制的路由器为每条路由分配一个动态的度量数字，用来反映路由稳定性的程

度。当一条路由出现摆动，就给它分配一个*惩罚值*。它摆动得越多，惩罚值累积得越多。这里还有一个时间段被称做*半衰期*。惩罚值以一定的速率降低，在每一个半衰期结束的时候，惩罚值应该降到原来值的一半。如果惩罚值超出了预先设定的门限值——抑制界限，该路由就会被抑制——也就是说，不再公布该路由。直到半衰期时间以后，惩罚值降低到低于另外一个门限值——重新使用界限时，才解除对该路由的抑制。那个时候，再次公布这条路由。但是可以选择手动重置路由的惩罚值。这种手动重置在某些情况下还是很有用的，例如路由的不稳定性经过修复并且要求立即重新使用该路由的时候。

除非抑制时间被设置得非常低，否则一个摆动不会引起路由的抑制。半衰期最终会使惩罚值降低到0。但是，如果一条路由摆动严重到惩罚值的增加速度比在半衰期它降低的速度快，惩罚值将会超出抑制时间。虽然在路由抑制过程中惩罚值还可以继续累积，但是路由抑制不能超过一个时间段，那就是最大抑制界限。这点就保证了一条可能在一个较短的时间内摆动很多次的路由不会累积一个很高的惩罚值，也就不会导致该路由被无限期地抑制。

对于路由抑制中不同的变量，Cisco的缺省值如下：

- **惩罚值**——每次摆动1000
- **抑制界限**——2000
- **重新使用界限**——750
- **半衰期**——15分钟
- **最大抑制时间**——60分钟，或者半衰期的4倍

在第3章的案例研究“路由抑制”中给出了在Cisco路由器上配置和使用路由抑制的例子。

2.4　IBGP和IGP的同步

几乎无一例外，在同一个AS内的对端实体，它们之间的内部BGP-BGP——只用于多宿主的情况。IBGP允许边缘路由器共享NLRI和相关的属性，从而执行系统范围内的路由策略。一个处于转接AS内的边缘路由器还使用IBGP将从一个外部对端学习到的路由传给其他的边缘路由器，从而向外部对等公布这些路由。

你可能会想在某些情况下将IBGP当作IGP使用。例如，一个ISP的AS通常都是通过EBGP与其他的自治系统相连，而且大部分携带的都是转接业务量。为什么不在AS内只使用IBGP，从而使AS内有一致的路由协议呢？问题在于对于网状连接来讲，每个IBGP路由器必须与其他每一个IBGP路由器对等——也就是说，IBGP网络互连必须是全网状的。本节会解释为什么IGP对于支持IBGP来讲是十分必要的，以及为什么IGP与BIGP之间的同步是十分重要的。采用全网状IBGP连接有以下两个原因：

- 在AS内防止BGP路由环路
- 保证BGP路由上的所有路由器都知道如何将数据包转发到目的地

当通过IBGP公布路由的时候，根据定义，它们是在同一个AS内进行公布。结果是，AS_PATH不会发生改变。实际上，在路由公布给EBGP对端之前，本地AS号不会附加到AS_PATH上。因此，IBGP路由不具备EBGP路由所具有的防止环路功能。为了防止环路，BGP不会将从一个IBGP对端学习到的路由宣告给另外一个IBGP对端。

图 2-32 说明了当 IBGP 对等关系不是完全网状连接时会出现的情况。在这里，Seattle 和 Tacoma 以及 Tacoma 和 Spokane 之间已经配置了 IBGP 对等会话。你可以看到 Seattle 和 Tacoma 之间正在交换有关它们本地网络的 NLRI，同时 Tacoma 和 Spokane 也在交换。但是 Seattle 和 Spokane 却无法学习到彼此的 NLRI。

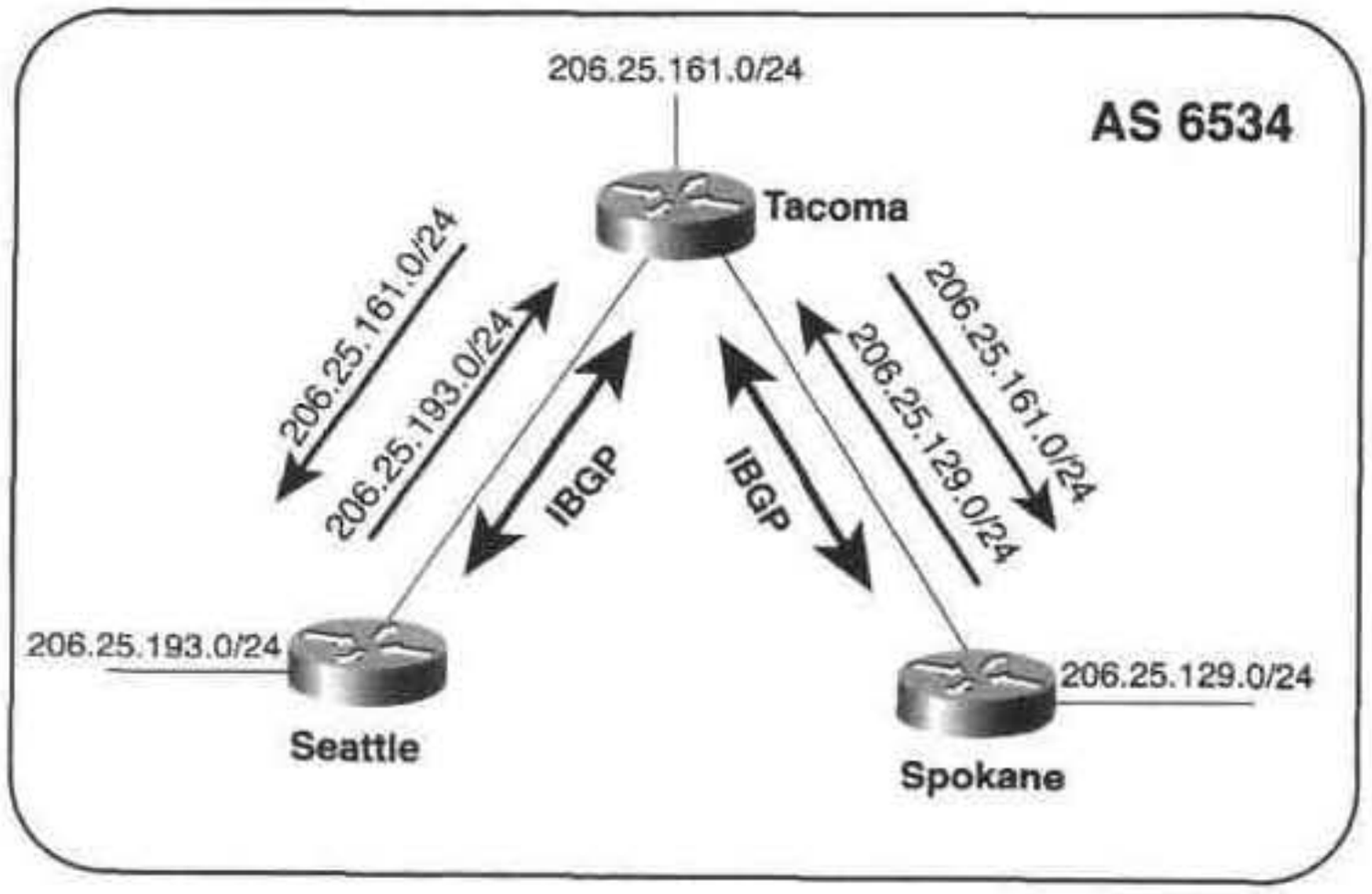


图 2-32　在部分网状 IBGP 环境下，所有的 NLRI 都没有公布

图 2-33 说明了通过全网状 IBGP 对等连接而获得的完全可到达性。注意，此时在 Seattle 和 Spokane 之间虽然没有直接的数据链路，但是它们是对等的。在 Seattle 和 Spokane 之间，BGP 使用的 TCP 连接要通过 Tacoma，但是逻辑上它们之间的连接是一个点到点的会话连接。这是非常重要的一点，因为要建立 TCP 会话，Seattle 和 Spokane 必须要知道将它们连接在一起的数据链路的地址。

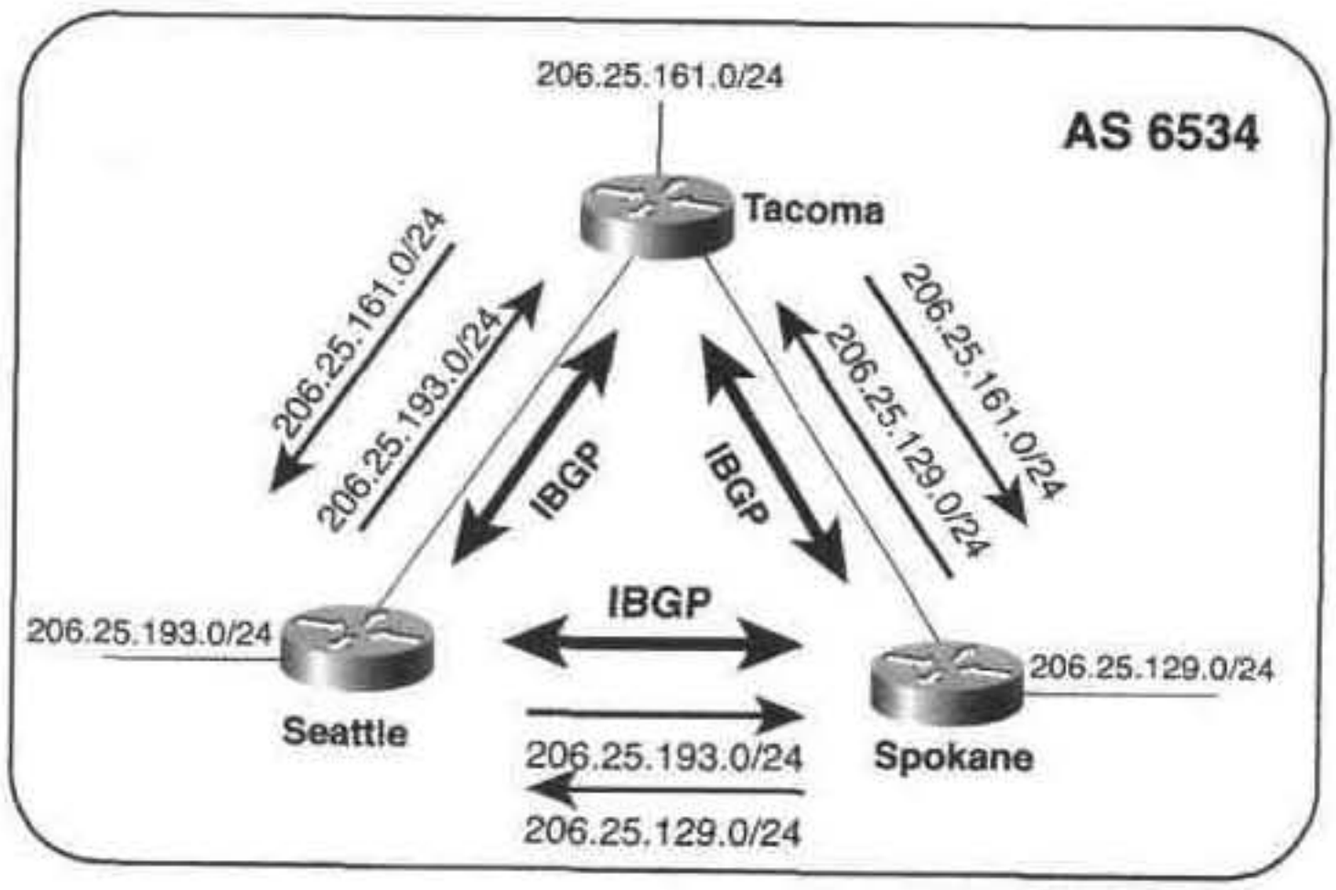


图 2-33　在全网状 IBGP 环境下，所有的 NLRI 都可以被交换

开始的时候，保证知道数据链路的地址看起来十分简单——因为每个路由器的地址必须包括在 **BGP network** 命令中(在第 3 章进行讨论)。但是，事情并不总是如此简单。

例 2-14 为 Seattle 的 BGP 路由表及其 IGP 路由表。对于前向分组的路由器，目的地必须包含在 IGP 路由表之中。

你可在例 2-14 的输出中看到，BGP 路由表中包含了一些路由，包括 Seattle-Tacoma 和 Spokane-Tacoma 数据链路的(192.168.1.0/24 和 192.168.2.0/24)地址。但是在 IGP 路由表中只有与 Seattle 直接相连的链路。同样在 BGP 路由表中没有 Spokane 的网络 206.25.129.0/24，这表明 Seattle 和 Spokane 没有正确地建立对等关系。

注：　注意在BGP路由表中直接相连链路的权值与从Tacoma学习到的路由的权值的比较。

例 2-14 说明了同步的问题。同步的规则如下：

一条从 IBGP 邻居学习到的路由在进入 IGP 路由表或者公布给一个 BGP 对端之前，通过 IGP 必须知道该路由。

在图 2-33 的互联网络中，BGP 路由不能进入 IGP 路由表，因为路由器没有运行 IGP，而且同步的规则也要求在路由进入 IGP 路由表之前，通过 IGP 应该能够知道这些路由。

要理解同步规则存在的原因，参考图 2-34。在这个例子中，没有将 IBGP 用作内部网关协议。相反，使用了一个正统的 IGP(OSPF)。Salt Lake 和 Provo 分别连接到两个独立的自治系统，并且它们通过 IBGP 来互相公布通过 EBGP 学习到的路由。这个 IBGP 的 TCP 会话要通过 Orem 和 Ogden。

例 2-14　虽然在 BGP 路由表中有几条路由，但是它们并不能自动进入到路由器的 IGP 路由表中

```
Seattle#show ip bgp
BGP table version is 7, local router ID is 206.25.193.1
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.0      0.0.0.0                  0         32768 i
* i                 192.168.1.1              0    100      0 i
*>i192.168.2.0      192.168.1.1              0    100      0 i
*>i206.25.161.0     192.168.1.1              0    100      0 i
*> 206.25.193.0     0.0.0.0                  0         32768 i

Seattle#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

C    206.25.193.0 is directly connected, Loopback0
C    192.168.1.0 is directly connected, Serial0
Seattle#
```

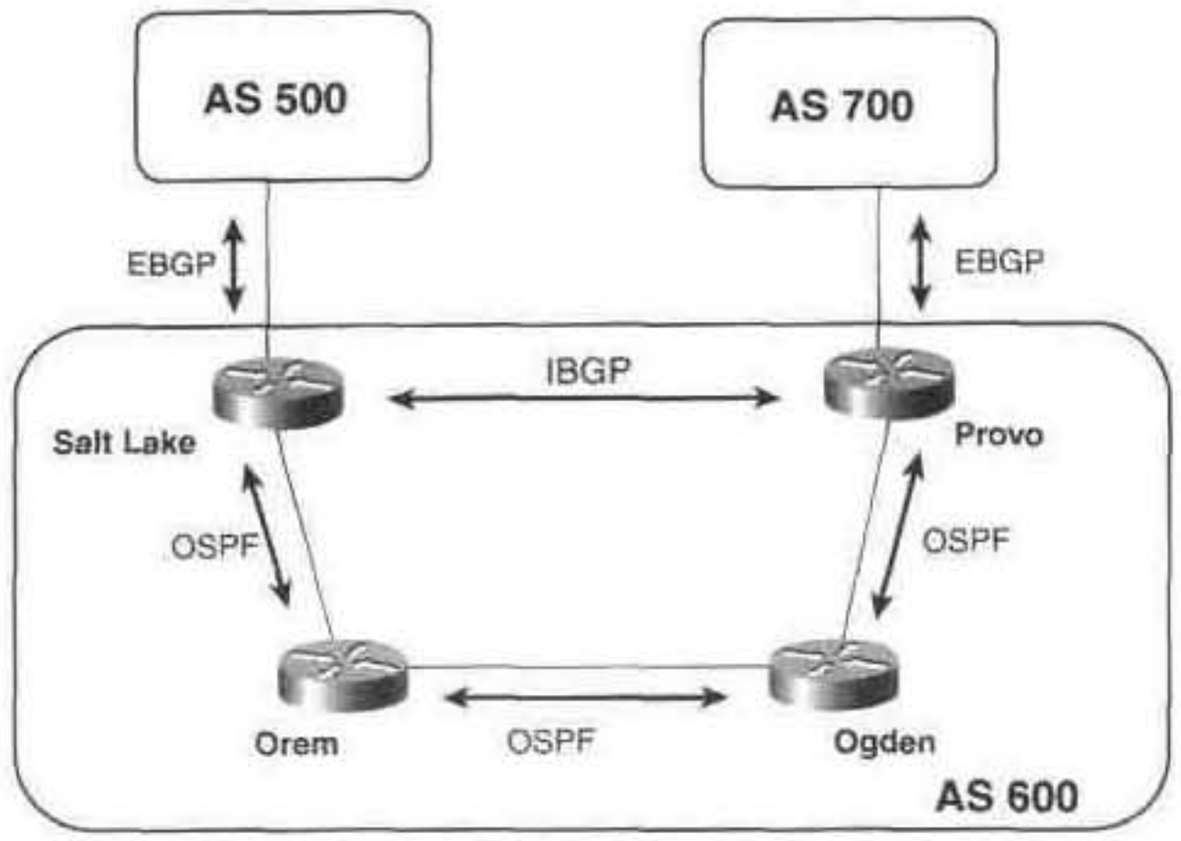


图 2-34　这个网络互连在 Salt Lake 和 Provo 之间运行部分网状 IBGP，并且将 OSPF 用作 IGP

下一步，假设 Salt Lake 从 AS 500 学习到了一条到 196.223.18.0/24 的路由，并且通过 IBGP 连接把它公布给 Provo，Provo 通过 **next-hop-self** 策略将 NEXT_HOP 属性改成自己的路由器 ID，然后将该路由公布给 AS 700。于是，AS 700 中的路由器开始转发目的地是 196.223.18.0/24 的数据包给 Provo(注意路由公布是传送数据包的一个承诺)。从这开始就出现了问题。Provo 执行了一个到 196.223.18.0/24 的路由查找，并且了解到通过 Salt Lake 该网络可以到达。于是它又查找 Salt Lake 的 IP 地址，通过查找它了解到通过下一跳路由器 Ogden 可到达 Salt Lake。但是外部路由是通过 IBGP 由 Salt Lake 和 Provo 共享的。OSPF 无法了解到外部路由。因此，当数据包被转发给 Ogden 以后，路由器进行路由查找但是没有找到到 196.223.18.0/24 的路由条目。于是该路由器就会丢弃数据包以及后续所有去往该地址的数据包，到网络 196.223.18.0/24 的业务量就成形成了黑洞。

当然，如果图 2-34 中的 OSPF 路由器能够知道外部路由，上面所描述的情况就不会出现了。Ogden 会知道通过 Salt Lake 可以到达 196.223.18.0/24 并且正确地转发数据包。同步可以防止数据包在一个 AS 内被一个没有足够信息的 IGP 造成黑洞现象。

当 Provo 收到来自 Salt Lake 关于 196.223.18.0/24 的路由公布，它将该路由添加到它的 BGP 路由表中。然后它会在它的 IGP 路由表中检查，看是否有关于这条路由的条目存在。如果没有，Provo 知道 IGP 不知道该路由，因此它不能公布该路由。如果，而且只有当 IGP 在路由表中加入了一条到达 196.223.18.0/24 的路由条目(也就是说，当 IGP 知道了该路由)，Provo 的路由会与 IGP 的路由进行同步，并且路由器可以开始自由地向它的 BGP 对端公布路由。

让我们回到图 2-33 和例 2-14 去看一下，为什么同步会导致全网状 IBGP 工作不正常。Tacoma 被限制在“第 22 条军规”中。它从 Seattle 和 Spokane 接收路由，但是它不能把这些路由加到它的 IGP 路由表中或者宣告它们，因为这些路由不在 IGP 路由表中，也没有 IGP 将它们加入到路由表中。

在某种程度上来讲，同步是 BGP 一个陈旧的特性，它承担将路由再分发到 IGP 的任务。但是，正如这个例子所描述的，对于全网状 IBGP，所有的路由器通过单独的 BGP 就可以知道所有必要的 BGP 路由。在这种情况下，同步只用于在 BGP 内保持 BGP 路由并且 IGP 只是用于建立 IBGP 连接。

幸运的是，Cisco 路由器有使同步关闭这样一个选项。例 2-15 给出了关闭了同步后，Seattle 的 BGP 和 IGP 路由表。Tacoma 转发来自 Spokane 的路由，而且数据包也能正确转发了。

这个例子就是要告诉我们，如果想让 IBGP 正确地工作，必须执行下面两个配置选项中的一个：

- 将外部路由再分发到 IGP 中从而保证 IGP 能够与 BGP 同步。这个方法的缺陷是，如果你从 BGP 得到了大量的路由，例如整个 Internet 的路由表，你会给 IGP 路由器带来巨大的处理和内存上的负担。在大部分情况下，路由器不能承受这样的负担，从而导致路由器瘫痪。实际上，几个大型的中断期的出现就是因为所有 BGP 路由被无意分发到 OSPF 或者 IS-IS 中而造成的。在一次事故中，一个主要供应商曾瘫痪了 19 个小时。
- IBGP 路由器必须是全网状连接，而且必须关闭同步功能。所有的路由器通过 BGP 都可以知道外部路由，而且关闭同步功能可以不用提前通知 IGP 就可以将路由加入到路由表中。这种方法的缺陷是，如果一个 AS 中有多个 IBGP 路由器，一个路由器要与其他每个路由器建立对等关系，这对管理工作来讲是一个挑战。虽然如此，在处理 Internet 路由时，通常采用这种方法。路由反射器和联盟的两个工具经常用来控制 IBGP 全网状连接的要求，相关内容将在下一节进行讨论。

例 2-15　图 2-33 中关闭了同步功能以后，在 Seattle 的 BGP 和 IGP 路由表中有全部的 NLRI

```
Seattle#show ip bgp
BGP table version is 11, local router ID is 206.25.193.1
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.0      0.0.0.0                  0         32768 i
* i                 192.168.1.1              0    100      0 i
*>i192.168.2.0      192.168.1.1              0    100      0 i
* i                 192.168.2.1              0    100      0 i
*>i206.25.129.0     192.168.2.1              0    100      0 i
*>i206.25.161.0     192.168.1.1              0    100      0 i
*> 206.25.193.0     0.0.0.0                  0         32768 i

Seattle#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

C    206.25.193.0 is directly connected, Loopback0
B    206.25.129.0 [200/0] via 192.168.2.1, 00:07:34
C    192.168.1.0 is directly connected, Serial0
B    192.168.2.0 [200/0] via 192.168.1.1, 00:07:42
B    206.25.161.0 [200/0] via 192.168.1.1, 00:07:43

Seattle#ping 206.25.129.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 206.25.129.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/5/8 ms
Seattle#
```

第 3 章给出了几个关于 IBGP 配置的例子。它也给出了这两个配置选项的缺陷，并且给出了几个较好的解决方案。

2.5　管理大型 BGP 对等关系

在前一节曾指出当一个 AS 规模变大时，试图生成全网状 IBGP 对等是令人畏缩的一项工作。这只是你工作于大型 BGP 时要遇到的众多问题中的一个。BGP 的四个有特色的工具可以简化对大量 BGP 对等关系的管理：

- 对等组
- 团体
- 路由反射器
- 联盟

头两个工具帮助简化对多个对等关系之间的路由策略的管理，不管是内部的还是外部的。后两个工具帮助在大量的对等中对 IBGP 进行管理。

2.5.1　对等组

通常在大型的互联网络中，一个路由器上的策略会应用到多个对等关系。例如，同样的属性会设置在更新消息中传给几个对端，或者对于来自几个对等的路由可能使用同一个过滤器。在这些情况下，你可以通过将共享公共策略的对等加入到一个对等组中来简化配置和管理。

在 Cisco 路由器中，用一个名字和一系列的路由策略来定义一个对等组。然后将对等关系加入到对等组中。任何对策略所做的改动会对整个组生效而不是针对每一个独立的对等。对等组在提高路由器的性能方面也十分有用。此时，路由器不需要为发给每个对端的每条 Update 消息重复地访问策略数据库。它只需要访问一次策略数据库，生成一个 Update 消息，然后将该消息的副本发给对等组中的每一个对等。

有时候，可能要对对等组中的一个或者多个成员使用附加的策略。在这种情况下，你可以对适当的邻居使用除了组中的公共策略以外的附加策略。

2.5.2　团体

对等组是对一组路由器使用策略，而团体是对一组路由使用策略。一个路由器通过将它的团体设置为某一个特定的值来表明它是这个团体的一个成员，从而可以在这个预先配置的团体中加入一条路由。在团体属性值的基础上，邻居路由器可以将它们的策略，例如过滤或者再分发策略应用于这条路由。可以设置成一个公认的值或者由网络管理人员定义特定值的团体属性，已经在本章的前面“团体属性”一节中有了更加全面的论述。

你可以为一条路由设置多个团体属性。接收到带有多个团体属性路由的路由器可以在所有这些属性的基础上或者一些属性子集的基础上选择设置策略。当对包括团体属性的路由进行聚合时，聚合路由继承了所有路由的全部团体属性。

2.5.3　路由反射器

当一个 AS 包含多个 IBGP 对等时，路由反射器十分有用。(想要了解更多的内容，参见

RFC 1966，网址是 www.isuedu/in-notes/rfc1771.txt。)除非 EBGP 路由被再分发到自治系统的 IGP 中，否则所有的 IBGP 必须是全网状连接。对于每 *n* 个路由器，在 AS 中都要有 *n*(*n*-2)/2 个 IBGP 连接。例如，图 2-35 给出了 6 个全网状连接的 IBGP 路由器，路由器不算很多，即使这样，还需要 15 条 IBGP 连接。

路由反射器为全网状 IBGP 对等提供了一个可供选择的办法。将一个路由器配置成路由反射器(RR)，其他的 IBGP 路由器当作*客户*，这些客户只与 RR 建立对等关系，而不与另外的每一个路由器对等(如图 2-36 所示，在一个路由反射器簇中，IBGP 客户只与路由反射器对等，从而降低了必要的 IBGP 连接的数量)。结果，对等会话的数目就从 *n*(*n*-2)/2 降到了 *n*-1 条。路由反射器和它的客户合称一个*簇*。

路由反射器是在放松了规则的情况下工作的，因为 IBGP 对端不能公布从其他 IBGP 对端学习到的路由。例如，在图 2-36 的互联网络中，路由反射器从它的每一个客户学习路由，不像其他的 IBGP 路由器，RR 能够将这些路由公布给它的其他客户以及非客户的对端。换句话说，来自一个 IBGP 客户的路由由 RR 反射给其他的客户。为了避免可能出现的路由环路或者其他的路由差错，路由反射器不能改动它从客户那接收到的路由的属性。

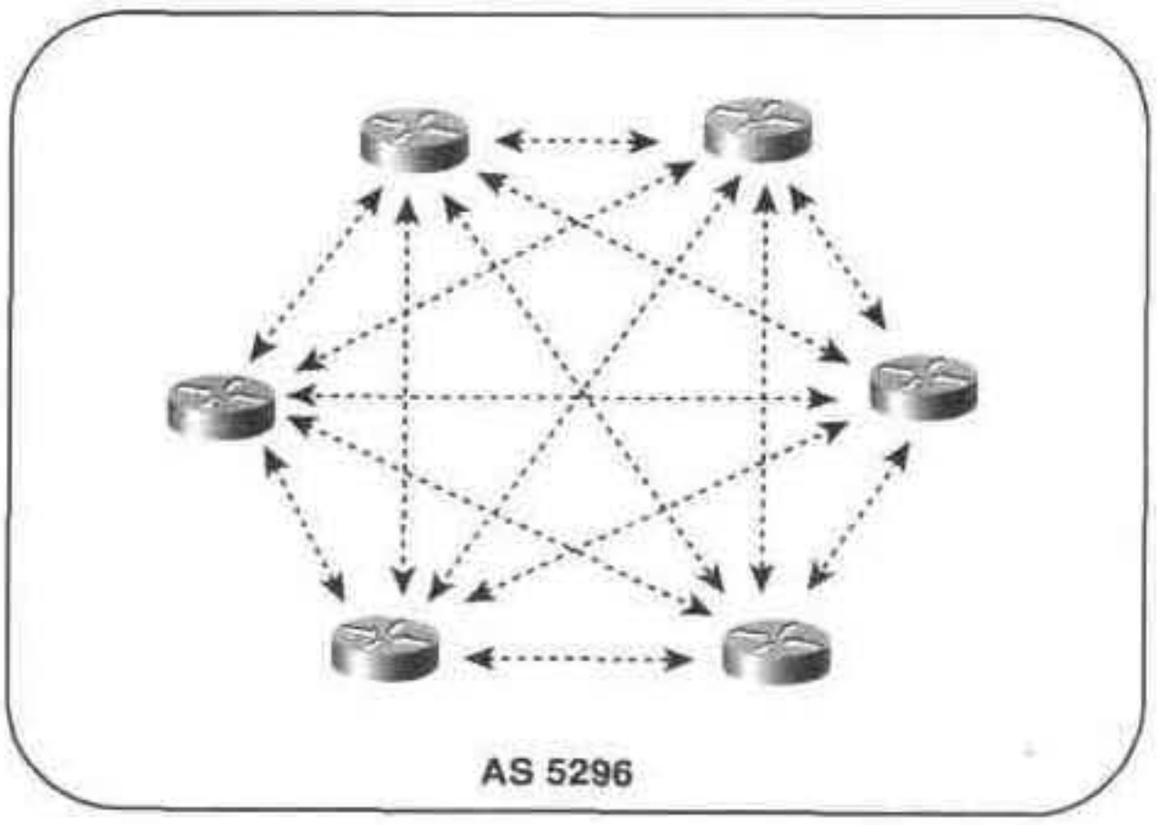


图 2-35　全网状连接的 IBGP 对等

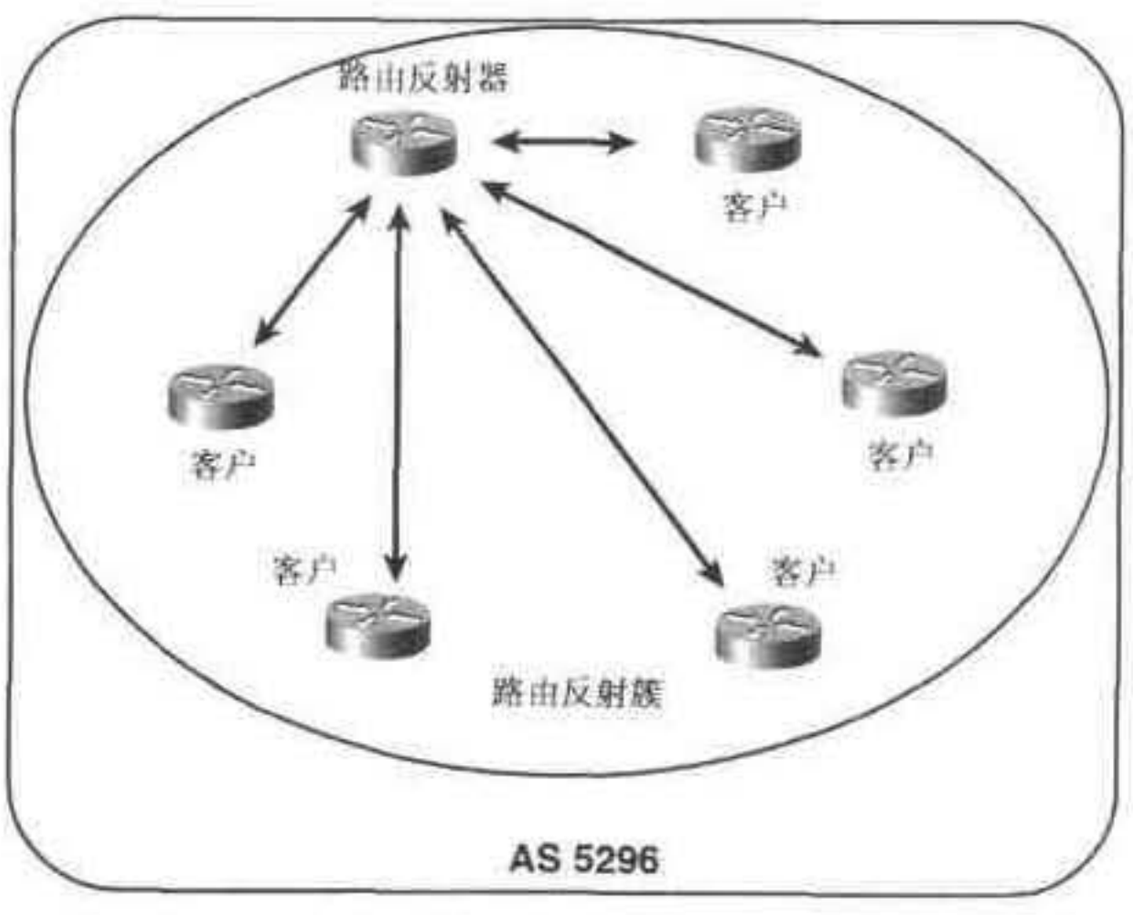


图 2-36　在一个路由反射器簇中

一个路由反射簇中的客户路由器能够与外部的邻居建立对等关系，但是它唯一可以对等的内部邻居就是它所在簇内的路由反射器或者该簇当中的其他客户。但是，RR 本身既可以与簇内也可以与簇外的邻居对等，并将它们的路由反射给它的客户(见图 2-37)。

如果一个 RR 接收到到同一个目的地的多条路由，它会使用通用的 BGP 选择过程来选择最优的路由。RFC 1966 定义了三个规则，根据学习到路由的途径，RR 通过这三条规则来决定将路由公布给谁：

- 如果路由是从非客户的 IBGP 对等学习到的，只将它反射给客户。
- 如果路由是从客户处学习到的，将它反射给除了发起该路由的客户外的所有非客户以及客户。
- 如果路由是从 EBGP 对等处学习到的，将它反射给所有的客户和非客户。

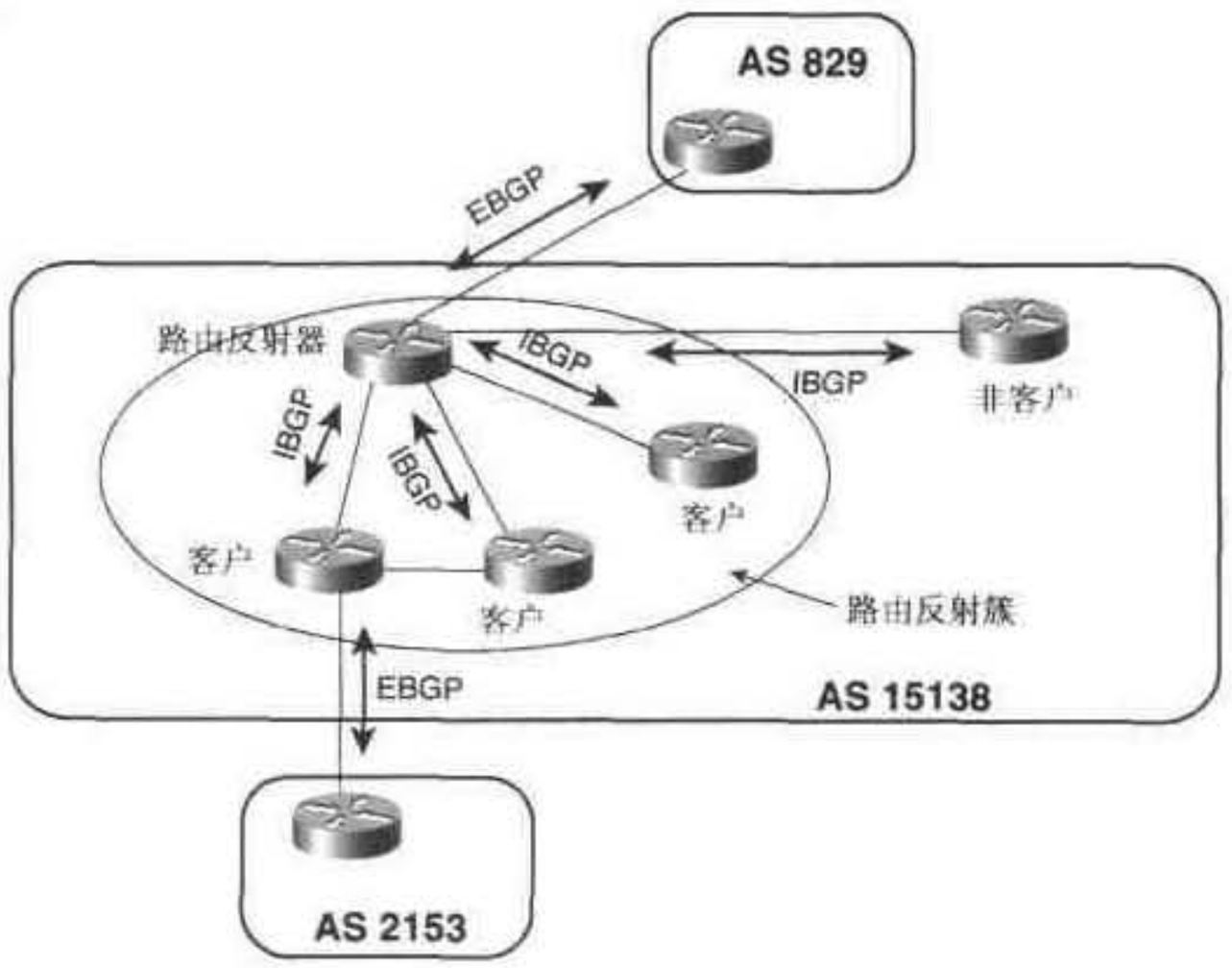


图 2-37　路由反射簇的对等关系

只有路由反射器本身支持路由反射器功能。从客户的观点来看，它们只与内部邻居为对端。这是路由反射器一个很有吸引力的特点，因为只具有相对基本 BGP 执行功能的路由器也可以是一个路由反射器簇的客户。

路由反射器的概念与在本章前面所讨论的路由服务器的概念相似。这两个设备的主要目的都是通过为多个邻居提供一个对等点来减少要求的对等会话的数量。于是这些邻居就依靠一个设备来学习路由。路由反射器和路由服务器的区别在于：路由反射器是路由器，而路由服务器不是路由器。

一个 RR 正如同一个路由服务器，为系统引进了一个单一的故障点。如果 RR 出现故障，客户将会丢失掉它们唯一的 NLRI 来源。因此，为了冗余，一个簇可以有多个 RR(见图 2-38)。客户有到每一个路由反射器的物理连接，并且它们和每一个路由反射器对等。如果一个 RR 出现故障，客户还有到其他 RR 的连接，因此不会丢失可到达消息。

注：　虽然一个客户可以只有到一个路由反射器的物理链路同时与多个 RR 对等，但是这种设置却破坏了冗余备份的目的。客户对与之有物理连接的 RR 的故障还是很敏感。

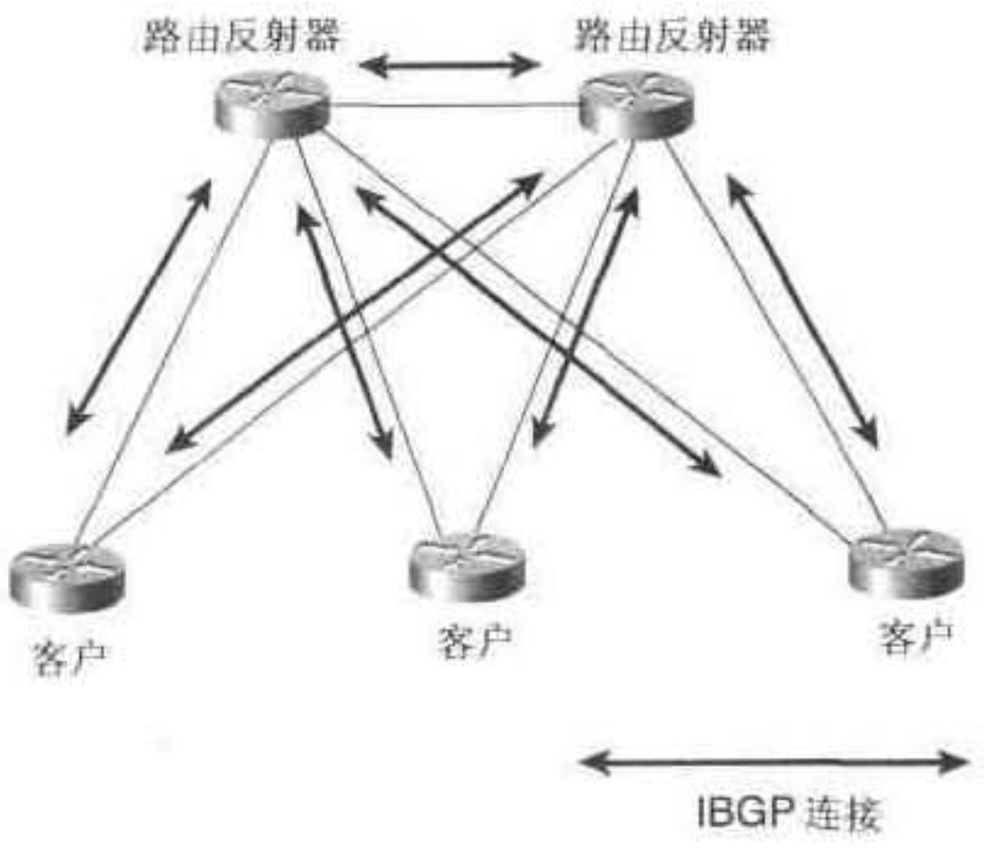


图 2-38　一个簇可以有多个路由反射器用于冗余备份

一个 AS 也可以有多个簇。图 2-39 给出了一个有两个簇的 AS。每个簇都有备份的路由反射器，并且两个簇之间也互相连接，相互备份。

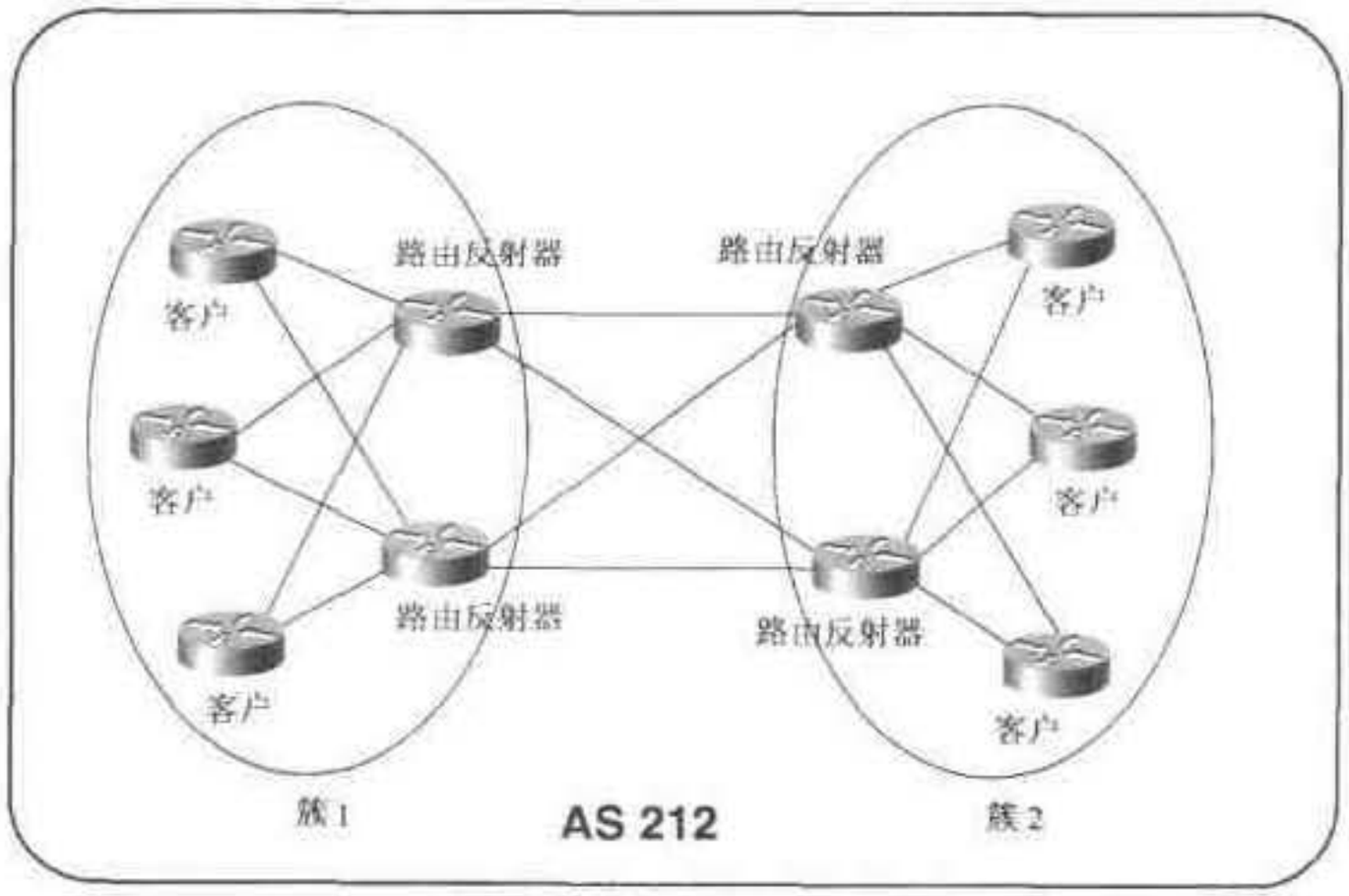


图 2-39　在一个 AS 内可以有多个路由反射器簇

因为客户并不知道它们是客户，因此一个路由反射器可以是另一个路由反射器的客户。结果是，你可以建立一个“嵌套” 的路由反射器簇(见图 2-40)。

虽然客户不能与它自己簇外的路由器对等，但是它们自己之间可以互相对等。结果是，一个路由反射器簇可以是全网状连接(见图 2-41)。当客户之间是全网状连接的时候，路由器被配置成不用反射从一个客户到另一个客户的路由。相反，它只反射从客户到非客户对端的路由，以及从非客户对端到客户的路由。

回忆一下，在“IBGP 和 IGP 同步”一节中，我们曾讨论过 BGP 不能转发从一个内部对端学习的路由到另外一个内部对端，因为在 AS 内部 AS_PATH 属性不能改变，这样就有可能出现路由环路。但是，请注意，路由反射器是一个 BGP 路由器，因此，这些规则已经得到了放松。为了防止出现路由环路，路由反射器使用两个 BGP 路径属性：ORIGINATOR_ID 和 CLUSTER_LIST。

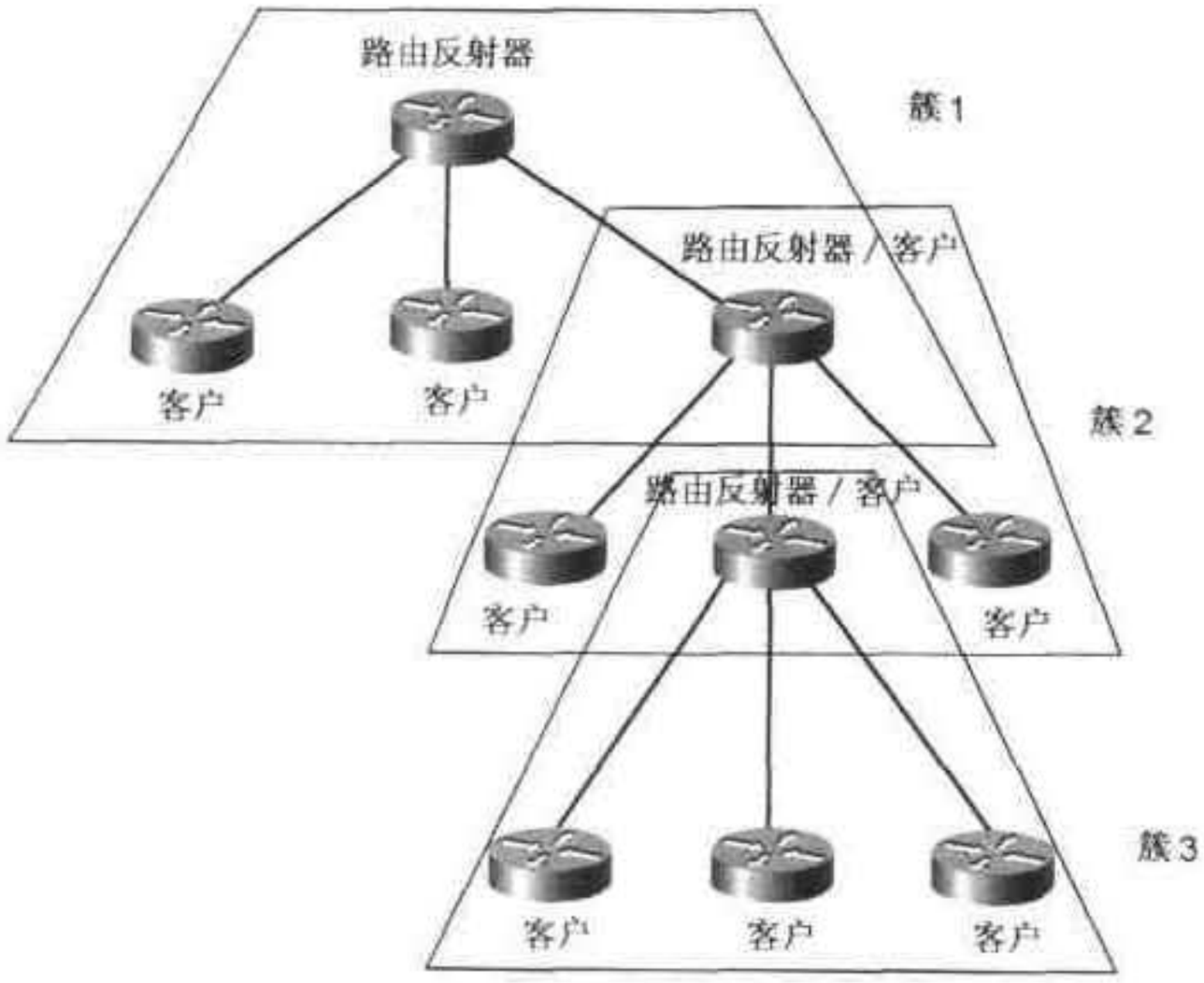


图 2-40　一个路由反射器可以是另一个路由反射器的客户

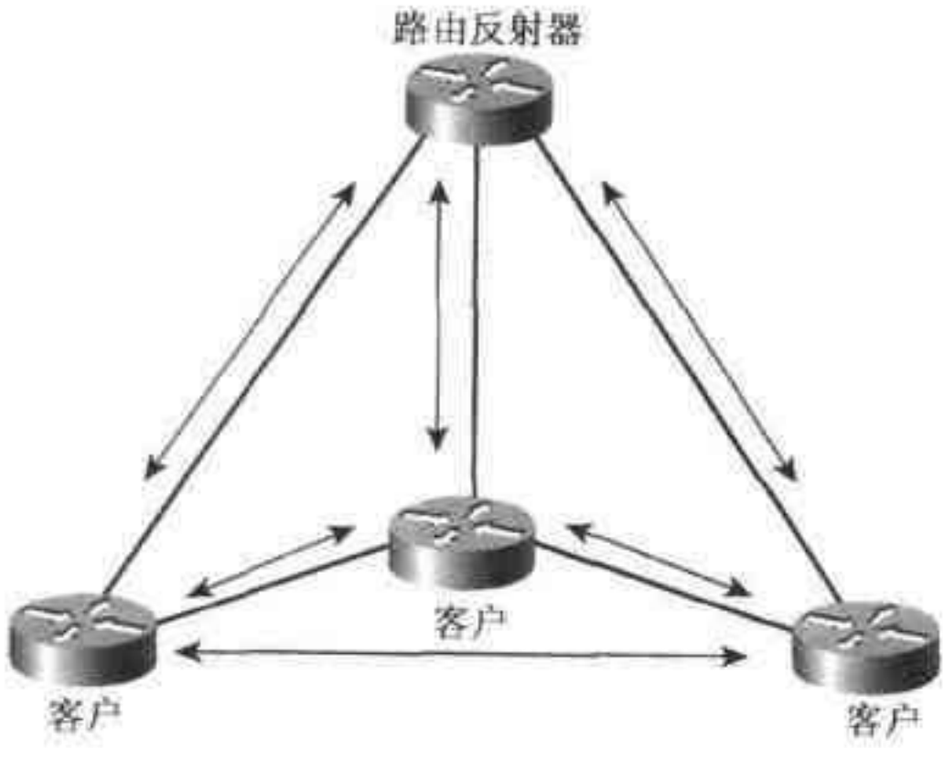


图 2-41　一个路由反射器簇可以是全网状连接

ORIGINATOR_ID 是一个任选非传递属性，由路由反射器生成。ORIGINATOR_ID 是本地 AS 内路由发起者的路由器 ID。路由反射器不会将一条路由公布给路由的发起者。尽管如此，如果发起者收到了一个带有它自己的 RID 的更新消息，它会不理睬该消息。

一个 AS 内的每一个簇必须用一个唯一的 4 字节*簇 ID* 来标识。如果簇内只有一个路由反射器，那么簇 ID 就是路由反射器的路由器 ID。如果簇内包含多个路由反射器，必须人工给每个 RR 配置一个簇 ID。

CLUSTER_LIST 是一个任选非传递属性，它跟踪簇 ID 的方式和 AS_PATH 属性跟踪 AS 号的方式相同。当一个 RR 将一条路由从一个客户反射到一个非客户，它将它的簇 ID 加到 CLUSTER_LIST 中。如果 CLUSTER_LIST 是空的，RR 就生成一个。当 RR 收到一个更新消息的时候，它检查 CLUSTER_LIST，如果在该列表中发现了自己的簇 ID，就知道出现了路由环路，它会不理睬这个更新消息。

2.5.4　联盟

联盟是控制大型 IBGP 对等的另一条途径。一个联盟是一组分成子自治系统组的 AS，被称做*会员自治系统*(见图 2-42)。在联盟内的运行 BGP 的路由器和在同一个会员 AS 内的对端之间使用 IBGP，和其他会员自治系统内的对端之间使用 EBGP。给每一个联盟分配一个联盟 ID。对外部联盟内的对端来讲，这个联盟 ID 代表整个联盟的 AS 号。外部对端看不到联盟的内部结构。它们只看到一个 AS。在图 2-42 中，AS 9184 是联盟 ID。

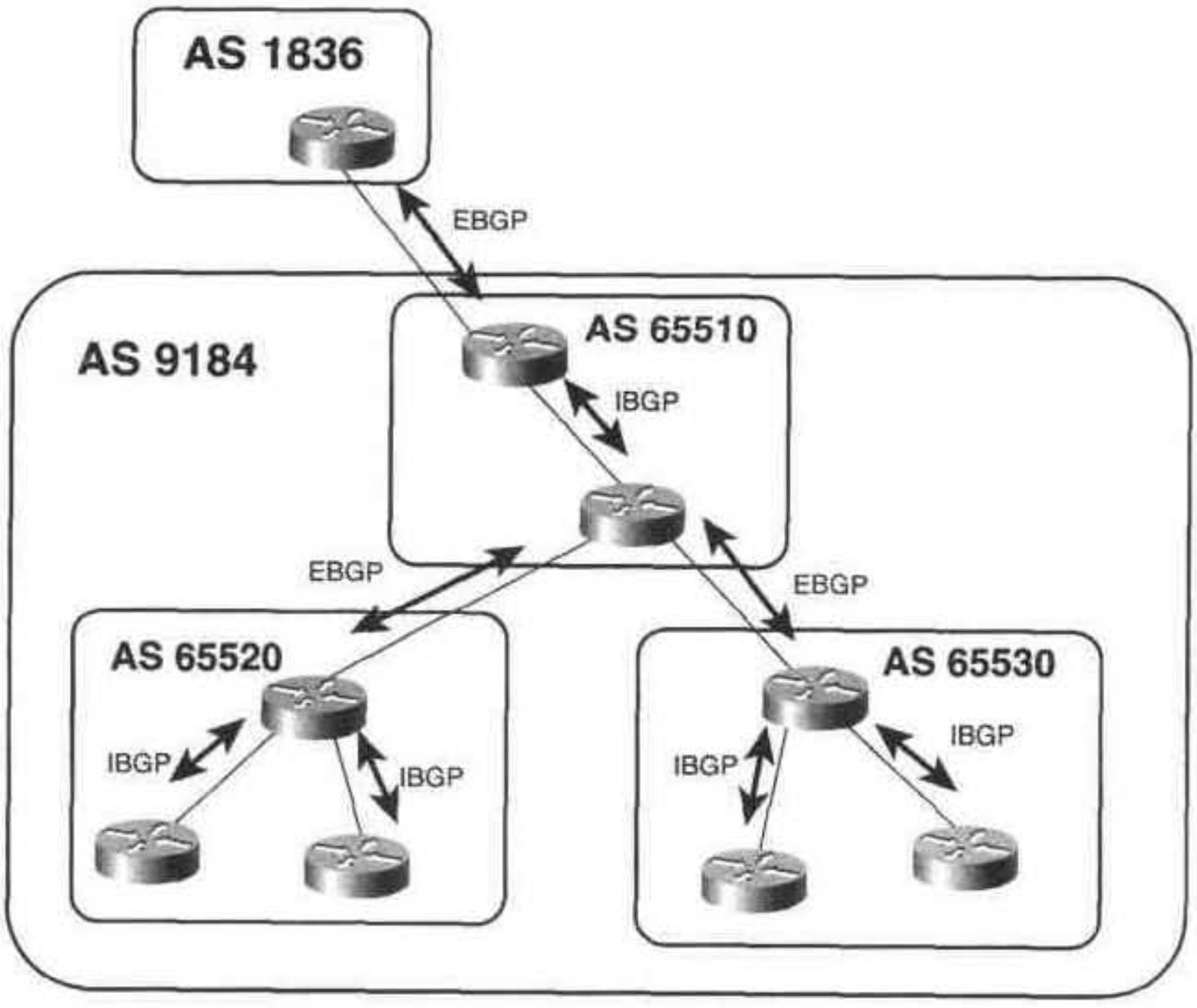


图 2-42　典型的联盟

你应该熟悉再分实体的概念，它能够进行更好的管理。IP 子网是 IP 网的再分(subdivision)，VLSM 又将子网再细分。同样地，自治系统是大型网络(例如 Internet)的细分。联盟是自治系统的再分。

在“AS_SET”一节中描述了 AS_PATH 的两类属性：AS_SEQUENCE 和 AS_SET。联盟为 AS_PATH 增加了两个属性类型：

- **AS_CONFED_SEQUENCE**——这是到目的地的一条路径上 AS 号的有序列表。它的用法和 AS_SEQUENCE 的用法是完全一样的，只是该列表中的 AS 号属于本地联盟中的自治系统。

• **AS_CONFED_SET**——这是到目的地的一条路径上 AS 号的一个无序列表。它的用法和 AS_SET 的用法一样，只是该列表中的 AS 号属于本地联盟中的自治系统。

因为 AS_PATH 属性用在自治系统成员之间的更新消息中，可以避免路由环路。从 AS 成员中 BGP 路由器的观点来看，所有其他自治系统成员中的对端都是外部邻居。

当你向该联盟外部的一个对端发送 Update 消息时，将 AS_CONFED_SEQUENCE 和 AS_CONFED_SET 信息从 AS_PATH 属性中去掉，将联盟 ID 加到 AS_PATH 中。因为这一点，外部对等就把联盟看作是一个 AS 而不是一个自治系统的集合。如图 2-42 所示，用预留范围 64512～65535 中的 AS 号来为联盟中的自治系统成员编号是一种通用的做法。

在选择路由的时候，BGP 决定过程还是一样的，只是增加了一条：到联盟的外部 EBGP 路由优于到自治系统成员的 EBGP 路由，而到自治系统成员的 EBGP 路由优于 IBGP 路由。联盟和标准的自治系统之间的另外一个不同就在于处理一些属性的方式上。一些属性，例如 NEXT_HOP 和 MED，可以不加修改地公布给联盟中的其他 AS 成员中的 EBGP 对端，而且也可以发送 LOCAL_PREF 属性。

与路由反射器环境不同，在路由反射器环境下，只有路由反射器本身一定要支持路由反射，而在联盟环境下，所有的路由器都要支持联盟功能。这个支持是必需的，因为所有的路由器都需要能够识别 AS_PATH 属性中的 AS_CONFED_SEQUENCE 和 AS_CONFED_SET 类别。因为向联盟外面公布路由时，要把这些 AS_PATH 类别去掉，因此，其他自治系统中的路由器不一定要支持联盟。

在超大型自治系统中，可以同时使用联盟和路由反射器。为了更好控制 IBGP 对等关系，你可以在一个或者多个自治系统成员内配置一个或者多个 RR 簇。

2.6　BGP 消息格式

在 TCP 段内携带 BGP 消息，使用端口 179。最大的消息尺寸是 4096 字节，最小的尺寸是 19 字节。所有的 BGP 消息都有一个通用的信头(见图 2-43)。根据消息的类型，在信头后面可以有数据字段，也可以没有数据字段。

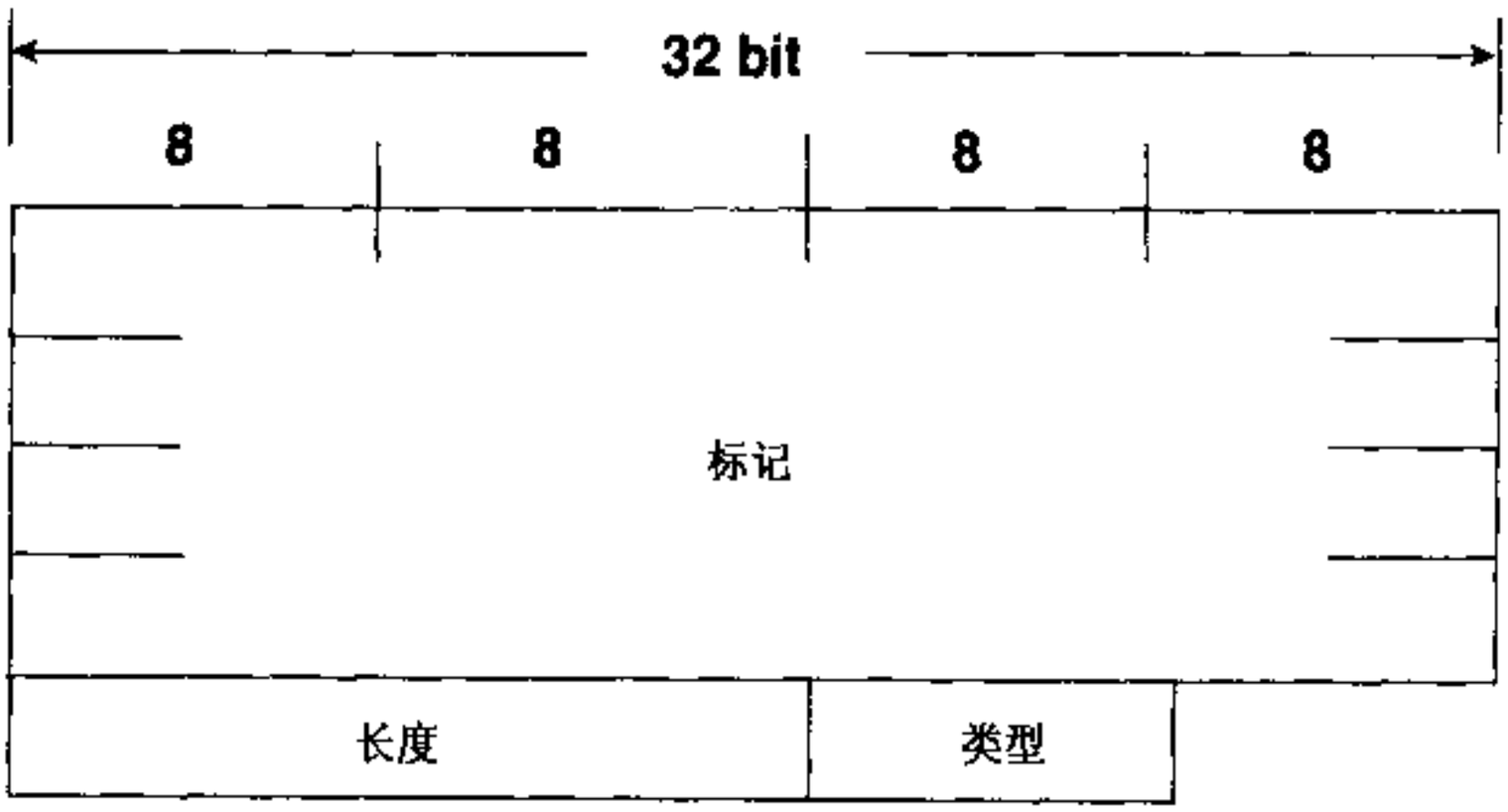


图 2-43　BGP 消息信头

*标记*是 16 字节的字段，用来检测 BGP 对等之间同步的丢失以及当支持鉴别功能时用来鉴别消息。如果消息类型是 Open 或者 Open 消息没有包含鉴别信息，标记字段全部置为 1。否则，标记的值可以通过一定的计算来预先得到，该过程也是鉴别过程的 一部分。

*长度*是一个 2 字节字段，用来指示消息的长度，其中包括信头的长度。

*类型*是 1 字节的字段，规定了消息的类型。表 2-6 给出了可能的类型码。

表 2-6　　BGP 类型码

编　码	类　型
1	Open
2	Update
3	Notification
4	Keepalive

2.6.1　Open 消息

Open 消息的格式如图 2-44 所示。在 TCP 连接建立起来以后，发送的第一个消息就是 Open 消息。如果收到的 Open 消息是可接受的，就会发送一个 Keepalive 消息来证实 Open 消息。Open 消息得到证实后，BGP 连接就处于连接状态，并且可以发送 Open、Keepalive 和 Notification 消息。

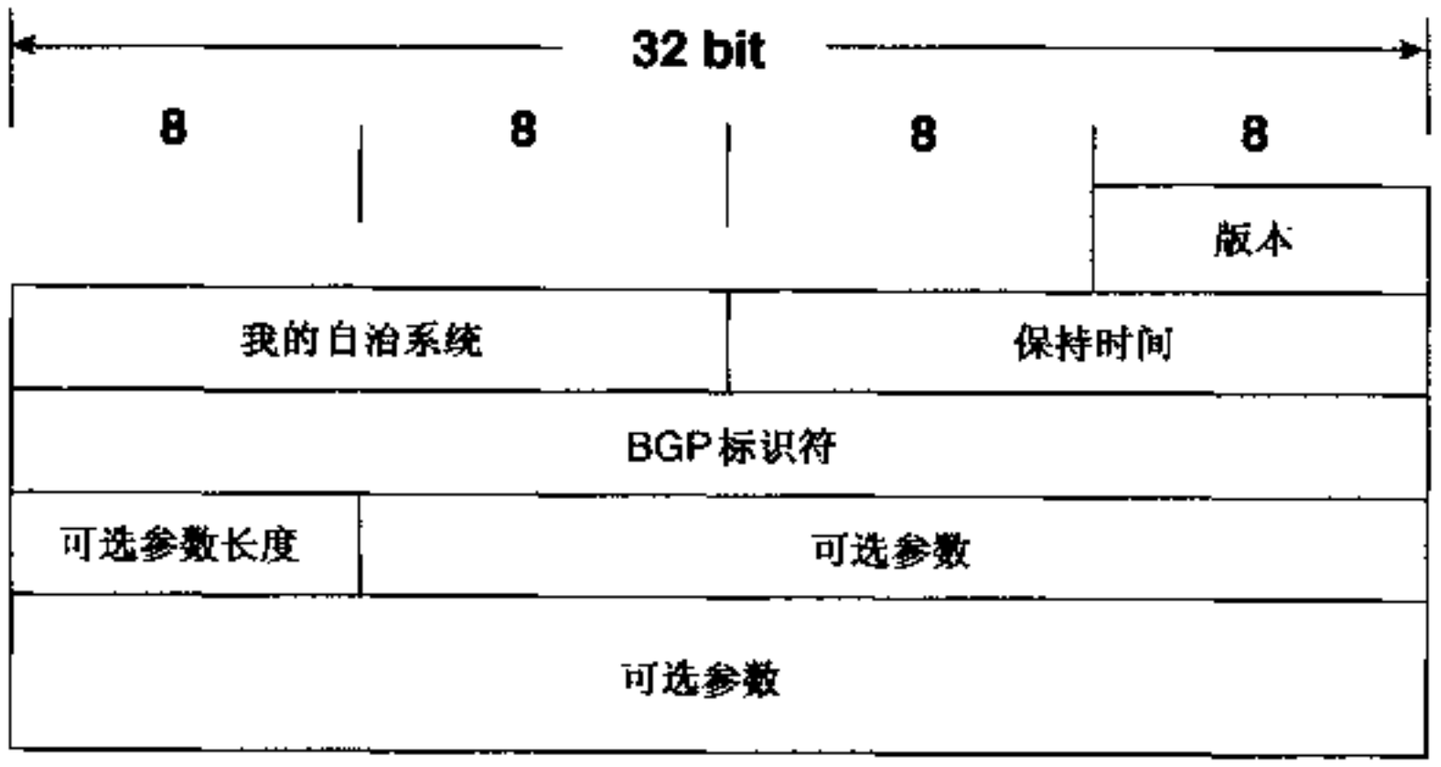


图 2-44　BGP Open 消息格式

BGP Open 消息包含以下的字段：

- **版本**——1 字节字段，规定发起者运行的 BGP 版本。
- **我的自治系统**——2 字节字段，规定发起者的 AS 号。
- **保持时间**——2 字节的字段，指示发送方为保持时间(Hold Time)建议的秒数。接收方将接收到的 Open 消息中的保持时间与自己所配置的保持时间相比较，接受较小的值或者拒绝该连接。保持时间可以是 0 或者至少 3 秒。
- **BGP 标识符**——发起者的路由器 ID。Cisco 路由器可以将它的路由器 ID 设置成任何

它的 loopback 接口中的最高 IP 地址，或者，如果没有配置 Loopback 接口，就设置成它的任何物理接口中最高的 IP 地址。

- **可选参数长度**——一个 1 字节的字段，指示接下来可选参数字段的整体长度，用字节来表示。如果这个字段是 0，那么在该消息中没有包括可选参数字段。
- **可选参数**——一个可变长度字段，包括一个可选参数列表。每个参数由 1 字节类型段、1 字节长度段和一个包含参数值的可变长度字段来明确。

2.6.2　Update 消息

图 2-45 给出了 Update 消息的格式，该消息用来向对端公布一个可用的路由，撤销多个不可用路由，或者完成这两项工作。

图 2-45　BGP Update 消息格式

BGP Update 消息包含以下字段：

- **不可用路由长度**——2 字节的字段，以字节的方式表示接下来的撤销路由(withdrawn routes)字段的整体长度。0 代表没有路由被撤销，并且在该消息中不包含撤销路由字段。
- **撤销的路由**——一个可变长度字段，是从服务中撤销的路由的一个列表。每条路由都由一个字节组(长度、前缀)来描述。在该字节组中，长度是指前缀的长度，前缀是被撤销路由的 IP 地址的前缀。如果该字节组的长度部分是 0，前缀就与所有的路由匹配。
- **全部路径属性长度**——2 字节字段，以字节指示接下来的路径属性字段的整体长度。该字段的值如果为 0，则说明在这个消息中不包含属性和 NLRI。
- **路径属性**——可变长度字段，列出了与下面字段中 NLRI 有关的属性。每个路径属性都是由可变长度的 3 字节组(属性类型、属性长度、属性值)组成的。3 字节组中属性类型部分是 2 字节的字段，包括 4 个标志比特、4 个未用比特和 1 个属性类型比特(见图 2-46)。

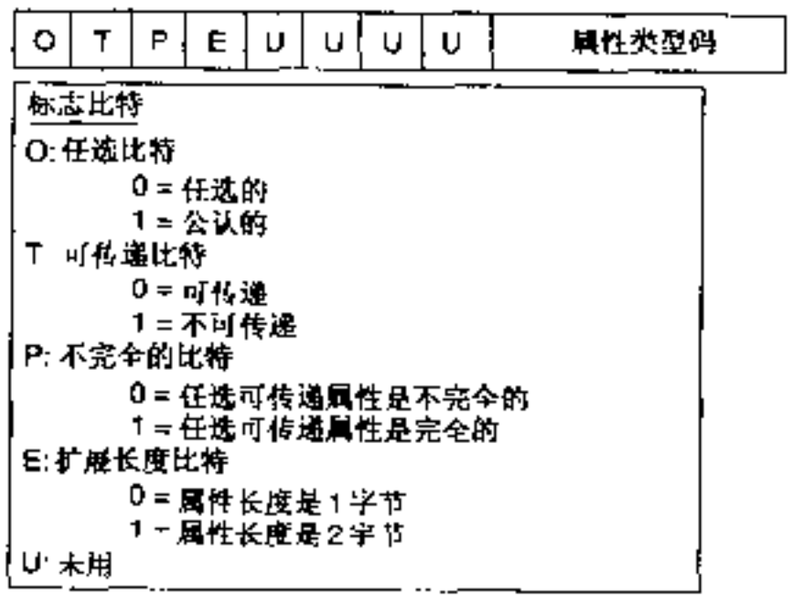


图 2-46　路径属性字段的属性类型部分

- **网络层可到达信息**——可变长度字段，包括一个字节组(长度、前缀)列表。长度部分用比特来表示下面的前缀的长度，前缀是 NLRI 的 IP 地址前缀。如果长度部分为 0，则表示该前缀与所有的 IP 地址匹配。

表 2-7 给出了最常见的属性类型码以及每个属性类型可能的属性值。

表 2-7　　属性类型和相关的属性值*

属性类型码	属性类型	属性值码	属性值
1	ORIGIN	0	IGP
		1	EGP
		2	Incomplete
2	AS_PATH	1	AS_SET
		2	AS_SEQUENCE
		3	AS_CONFED_SET
		4	AS_CONFED_SEQUENCE
3	NEXT_HOP	0	下一跳 IP 地址
4	MULTI_EXIT_DISC	0	4 字节 MED
5	LOCAL_PREF	0	4 字节 LOCAL_PREF
6	ATOMIC_AGGREGATE	0	没有
7	AGGREGATOR	0	聚合体的 AS 号和 IP 地址
8	团体	0	4 字节团体标识符
9	ORIGINATOR_ID	0	4 字节发起者的路由器 ID
10	CLUSTER_LIST	0	簇 ID 的可变长度列表

*还存在其他属性类型，但是它们属于非 Cisco 的厂家所有，因此不在本书所涉及的范围之内。

2.6.3　Keepalive 消息

Keepalive 消息以保持时间的 1/3 为间隔进行交换，但是该时间段不能低于 1 秒。如果协商的保持时间为 0，则不发送 Keepalive 消息。

2.6.4　Notification 消息

图 2-47 给出了 Notification 消息的格式，当检测到差错情况的时候会发送该消息。发送了该消息以后，BGP 连接会被立即中断。

BGP Notification 消息包含以下字段：

- **差错编码**——1 字节的字段，指示差错的类型。
- **差错子码**——1 字节的字段，提供有关差错的更详细的信息。
- **数据**——可变长度字段，用来诊断差错原因。Data 字段的内容取决于差错编码和差错子码。

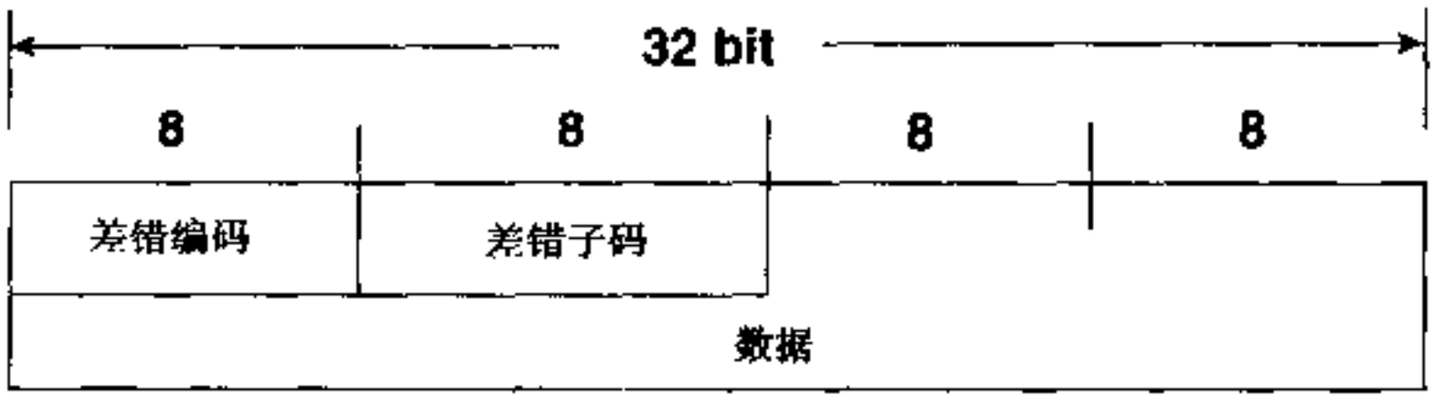


图 2-47　BGP Notification 消息格式

表 2-8　　BGP Notification 消息差错编码和差错子码

差错编码	差　　错	差错子码	子码的详细内容
1	消息信头差错	1	连接不同步
		2	坏的消息长度
		3	坏的消息类型
2	Open 消息差错	1	不支持的版本号
		2	坏的对等 AS
		3	坏的 BGP 标识符
		4	不支持的可选参数
		5	鉴别故障
		6	不可接受的保持时间
3	Update 消息差错	1	畸形的属性列表
		2	不可识别的 WELL-KNOWN 属性
		3	丢失 WELL-KNOWN 属性
		4	属性标志差错
		5	属性长度差错
		6	无效的 ORIGIN 属性
		7	AS 路由环路
		8	无效的 NEXT_HOP 属性
		9	可选属性差错
		10	无效网络段
		11	畸形的 AS_PATH
4	保持计时器到时	0	—
5	有限状态机差错	0	—
6	终止	0	—

2.7　尾注

[1]K.Lougheed 和 Y.Rekhter，“RFC 1105：A Border Gateway Protocol(BGP)”(正在进行中)

[2]K.Lougheed 和 Y.Rekhter，“RFC 1163：A Border Gateway Protocol(BGP)”(正在进行中)

[3]K.Lougheed 和 Y.Rekhter，“RFC 1267：Border Gateway Protocol 3(BGP-3)”(正在进行中)

[4]Y.Rekhter 和 T.Li，“RFC 1771：Border Gateway Protocol 4(BGP-4)”(正在进行中)

[5]Internet Engineering Steering Group，R.Hinden，编辑，“RFC 1517：Applicability Statement for the Implementation of Classless Inter-Domain Routing (CIDR)”(正在进行中)

[6]V.Fuller et al.，“RFC 1519：CIDR：An Address Assignment and Aggregation”(正在进行中)

[7]Y.Rekhter 和 C.Topolcic，“RFC 1520：Exchanging Routing Information Across Provider Boundaries in the CIDR Environment”(正在进行中)

[8]R.Chandra 和 P.Traina，“RFC 1997：BGP Communities Attribute”(正在进行中)

2.8　展望

现在你已经对 BGP 的基础知识以及相关的概念有了一定程度的了解，第 3 章将讲解在 Cisco 路由器上如何配置 BGP 以及为 BGP 进行故障排除。除了配置 BGP，你还会学到如何设置路由策略以及如何再分发 BGP 和 IGP。

2.9　推荐的读物

Halabi,B.和 D.McPherson，Internet Routing Architectures,第二版.Indianapolis, Indiana: Cisco Press;2000

该书被很多人认为是 BGP-4 方面的权威。

Stewart J.W.III.BGP4: *Inter-Domain Routing in the Internet*.Massachusetts：Addison Wesley Longman；1999

本书不是针对 Cisco 设备的，给出了一个关于 BGP 的简洁的总的观点。

2.10　复习题

1．BGP-4 和早期版本的 BGP 之间最重要的区别是什么？

2．开发 CIDR 是为了减轻哪两方面的问题？

3．有类别和无类别 IP 路由器之间的区别是什么？

4．有类别和无类别 IP 路由协议之间的区别是什么？

5．给出 4 个地址 172.17.208.0/23、172.17.210.0/23、172.17.212.0/23 和 172.17.214.0/23，用一个聚合地址来归纳以上地址，使用可能的最长的地址掩码。

6．什么是地址前缀？

7．例 2-16 中的路由表来自一个无类别路由器。路由器会将带有下列地址的数据包转发到哪一个下一跳地址？

172.20.3.5
172.20.1.67
172.21.255.254
172.16.50.50
172.16.0.224
172.16.51.50
172.17.40.1
172.17.41.1
172.30.1.1

例 2-16　复习题 7 的路由表

```
Stratford#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

     172.20.0.0 is variably subnetted, 6 subnets, 2 masks
D       172.20.0.0 255.255.0.0 [90/409600] via 172.20.5.2, 00:01:50, Ethernet0
D       172.20.2.0 255.255.255.0
           [90/409600] via 172.20.6.2, 00:01:50, Ethernet1
D       172.20.3.0 255.255.255.0
```

```
            [90/5401600] via 172.20.6.2, 00:01:50, Ethernet1
C       172.20.5.0 255.255.255.0 is directly connected, Ethernet0
C       172.20.6.0 255.255.255.0 is directly connected, Ethernet1
C       172.20.7.0 255.255.255.0 is directly connected, Ethernet2
     172.16.0.0 is variably subnetted, 3 subnets, 2 masks
D       172.16.50.0 255.255.255.0
            [90/409600] via 172.20.6.2, 00:01:50, Ethernet1
D       172.16.0.0 255.255.255.0
            [90/460800] via 172.20.6.2, 00:01:51, Ethernet1
D       172.16.0.0 255.255.0.0 [90/409600] via 172.20.7.2, 00:01:51, Ethernet2
     172.17.0.0 is subnetted (mask is 255.255.255.0), 1 subnets
D       172.17.40.0 [90/2841600] via 172.20.7.2, 00:01:52, Ethernet2
D   172.16.0.0 (mask is 255.240.0.0) [90/409600] via 172.20.5.2, 00:01:52, Ethernet0
Stratford#
```

8. 解释为什么路由总结可以帮助隐藏网络的不稳定性。

9. 解释为什么路由总结会引起不对称的业务量类型。

10. 不对称的业务量是不希望出现的吗？

11. 什么是 NAP？

12. 什么是路由服务器？

13. 独立于供应商的地址空间是什么，为什么有一个独立的地址空间会带来一定的好处？

14. 有一个独立于供应商的/21 的地址空间，会带来什么样的问题？

15. 什么是路由策略?

16. 提供到邻居可靠连接的 BGP 下层协议是什么?

17. BGP 的四个消息类型是什么，它们是如何使用的?

18. 在什么状态下，BGP 对等实体之间可以交换 Update 消息?

19. 什么是 NLRI?

20. 什么是路径属性?

21. BGP 路径属性的四个种类是什么?

22. AS_PATH 属性的目的是什么?

23. AS_PATH 的不同类型是什么?

24. NEXT_HOP 属性的作用是什么?

25. LOCAL_PREF 属性的作用是什么？

26. MULTI_EXIT_DISC 属性的作用是什么？

27. 如果运行 BGP 的路由器发起一个聚合路由，什么属性是有用的？

28. BGP 管理权值是什么？

29. 到同一个目的地，有一条 EBGP 路由和一条 IBGP 路由，BGP 路由器会优先选择哪一个？

30. 一个路由器有两条到同一个目的地的路由。路径 A 在 AS_PATH 中有一个值为 300 的 LOCAL_PREF 和 3 个 AS 号。路径 B 在 AS_PATH 中有一个值为 200 的 LOCAL_PREF 和 2 个 AS 号。假设没有其他区别，该路由器会选择哪条路径？

31. 什么是路由抑制？

32. 定义应用于路由抑制的惩罚、抑制界限、重新使用界限和半衰期。

33. 什么是 IGP 同步？为什么它很重要？

34．在什么情况下，你可以安全地关闭 IGP 同步功能？

35．什么是 BGP 对等组？

36．什么是 BGP 团体？

37．什么是路由反射器？什么是路由反射客户？什么是路由反射簇？

38．ORIGINATOR_ID 和 CLUSTER_LIST 路径属性的作用是什么？

39．什么是 BGP 联盟？

40．在联盟内可以使用路由反射器吗？

41．**next_hop_self** 功能的目的是什么？有什么合理的替代该功能的办法吗？

第 3 章　BGP4 的配置以及故障排除

本章涉及的主要内容如下：

- **基本的 BGP 配置**——本节为 BGP 配置提供了一系列的案例研究，包括对等 BGP 路由器、向 BGP 中注入 IBGP 路由、向 IGP 中注入 BGP 路由、没有 IGP 的 IBGP、IGP 上的 IBGP、EBGP 多跳以及聚合路由。
- **管理 BGP 连接**——本节给出各种不同的命令和工具，从管理以及维护的角度来讲，这些命令和工具使得 BGP 连接更加容易管理。
- **路由策略**——本节讨论了重新设置 BGP 连接并且给出了一系列的案例研究，包括用 NLRI、AS_PATH 以及路由地图(MAP)来过滤路由；管理权值；管理距离和后门路由；使用 LOCAL_PREF 和 MULTI_EXIT_DISC 属性；附加 AS_PATH；路由标签以及路由抑制。
- **大型 BGP**——本节为大型 BGP 设计提供了一系列的案例研究，包括 BGP 对等组、BGP 团体、专用 AS 号、BGP 联盟以及路由反射器。

许多刚刚接触 BGP 的人都会感到紧张。出现这种情况实际是因为 BGP 的使用要比 IGP 的使用少得多。除了 ISP，大部分的网络管理者处理 BGP 的机会都会远远低于处理 IGP，也可能根本没有机会遇到。即使使用 BGP 的时候，在小型 ISP 处和非 ISP 的用户处对它的配置也相当的简便。由于大部分的连网人员缺乏在这个协议方面的深入经验，因此该协议经常被看作是神秘或者令人恐惧的。

在第 2 章中，我们了解到 BGP 本身是一个相对简单的协议。确实 BGP 没有 EIGRP、OSPF 或者 IS-IS 复杂。BGP 的复杂性不在协议本身，而在于它所使用的情况以及与之相关的强大的工具。如果 AS 不是多宿主，或者只有基本的路由策略，通常 BGP 不是必需的。

本章从基本的 BGP 配置开始，然后给出了一些利用 BGP 来设置路由策略的例子，这些策略就是发送和接收路由公布的规则。在本章的最后会涉及到在大型 AS 内配置 BGP。

BGP 可用的配置选项有很多，只用几个实例并不能将故障排除讲解得很充分。因此本章在给出故障排除问题的同时都给出了许多配置选项和实例。

3.1　基本的 BGP 配置

本节给出了配置一个BGP过程的基本步骤以及控制BGP最常用的技术。对于大部分BGP的实施来讲，本节所给出的信息都是你所需要的。

3.1.1　案例研究：建立 BGP 路由器之间的对等

两个路由器之间的 BGP 对等通过以下两步来配置：

第一步　用 **router bgp** 命令建立 BGP 过程并且明确本地 AS 号。

第二步　用 **neighbor remote-as** 命令确定邻居以及邻居的 AS 号。

图 3-1 给出了两个在不同 AS 内的路由器。对这些路由器的 BGP 配置与 EGP 配置不同。在第十一章“外部网关协议”中，**router egp** 命令确定远端 AS，**autonomous-system** 命令确定本地 AS。相反，在这里 **router bgp** 确定的是本地的 AS。每个邻居的 AS 号由 **neighbor remote-as** 命令来确定。这个不同非常重要。因为只有核心路由器可以和多个 AS(使用 **router egp 0** 命令)对等，而任何一个 BGP 过程都可以和任何数目的远端 AS 对等。EGP 对通过一个核心 AS 相连的末梢自治系统的要求在这里被取消了：在 BGP 情况下自治系统可以全网状连接。

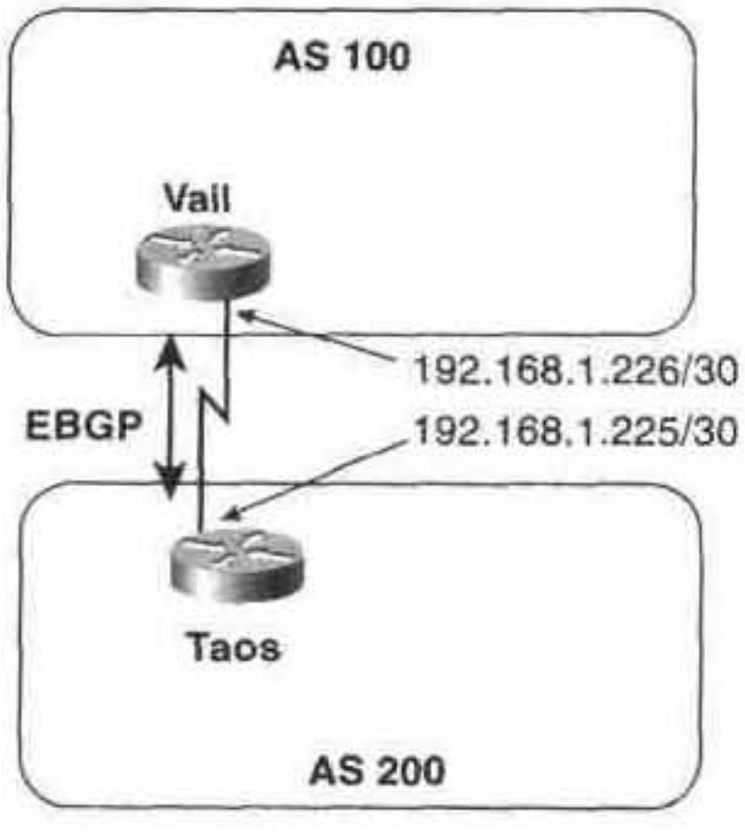


图 3-1　在 Taos 和 Vail 之间建立了一个 EBGP 会话

例 3-1 在图 3-1 中对 Taos 和 Vail 路由器的 EBGP 配置。

例 3-1　图 3-1 中路由器的 EBGP 配置

```
Taos
router bgp 200
 neighbor 192.168.1.226 remote-as 100
```

```
Vail
router bgp 100
 neighbor 192.168.1.225 remote-as 200
```

例 3-2 给出了 Vail 记录的有关 Taos 的信息。该屏幕显示的很多信息对故障排除都十分有用。在附录 A 中给出了由 **show ip bgp neighbors** 命令可得到的所有显示字段的一个完整的描述。

在例 3-2 中，第一行给出 Taos 的地址(192.168.1.225)、它的 AS 号以及到路由器(外部)的 BGP 连接的类型。第三行显示了在 Vail 和 Taos 之间使用的 BGP 的版本以及 Taos 的路由器 ID。第四行显示了 BGP 有限状态机的状态。当 BGP 路由表发生变化时，表的版本号会增加；在例 3-2 中，自从到 Taos 的连接建立以后，还没有发生变化，因此表版本号还是 1。正常运行时间给出了从对等连接建立到现在的时间。在例 3-2 中，Taos 持续对等关系的时间是 19 小时 32 分钟 2 秒。

例 3-2　show ip bgp neighbor 命令的输出包括与一个邻居的对等连接的详细信息

```
Vail#show ip bgp neighbors
BGP neighbor is 192.168.1.225,  remote AS 200, external link
 Index 1, Offset 0, Mask 0x2
  BGP version 4, remote router ID 192.168.1.225
  BGP state = Established, table version = 1, up for 19:32:02
  Last read 00:00:03, hold time is 180, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 30 seconds
  Received 1175 messages, 0 notifications, 0 in queue
  Sent 1175 messages, 0 notifications, 0 in queue
  Prefix advertised 0, suppressed 0, withdrawn 0
  Connections established 1; dropped 0
  Last reset never
  0 accepted prefixes consume 0 bytes
  0 history paths consume 0 bytes
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 192.168.1.226, Local port: 11025
Foreign host: 192.168.1.225, Foreign port: 179

Enqueued packets for retransmit: 0, input: 0  mis-ordered: 0 (0 bytes)

Event Timers (current time is 0x45FDF2C):
Timer          Starts    Wakeups            Next
Retrans          1176          0             0x0
TimeWait            0          0             0x0
AckHold          1175        885             0x0
SendWnd             0          0             0x0
KeepAlive           0          0             0x0
GiveUp              0          0             0x0
PmtuAger            0          0             0x0
DeadWait            0          0             0x0

iss: 4072889888  snduna: 4072912224  sndnxt: 4072912224     sndwnd:  16004
irs: 4121607729  rcvnxt: 4121630065  rcvwnd:      16004  delrcvwnd:    380

SRTT: 300 ms, RTTO: 607 ms, RTV: 3 ms, KRTT: 0 ms
minRTT: 4 ms, maxRTT: 340 ms, ACK hold: 200 ms
Flags: higher precedence, nagle

Datagrams (max data segment is 1460 bytes):
Rcvd: 2220 (out of order: 0), with data: 1175, total data bytes: 22335
Sent: 2077 (retransmit: 0), with data: 1175, total data bytes: 22335
Vail#
```

我们还对下面的TCP连接的详细信息感兴趣。例3-2将有关TCP连接的这些行给加亮了。这些行显示出 TCP 连接的状态是 Established，Vail 通过 TCP 端口 11025 发起 BGP 消息，在 Taos 上的目的地端口是 179。当你在一条承载多个 BGP 会话的链路上抓取数据包时，源端口显得十分重要。

在图 3-2 中，将另外一个路由器加到 AS 100 中。因为 Vail 和 Aspen 在同一个 AS 内，因此它们是内部邻居。例 3-3 显示了图 3-2 中 Vail 路由的配置。

例 3-4 显示了经过配置的 Aspen，如该例图所示，当 Aspen 和 Vail 建立对等关系的时候，**debug ip bgp events** 命令显示了 BGP 有限状态机的状态。BGP 调试手段用于发现生成的 BGP 对等会话。从 BGP 配置生成(18:24:13)到 BGP 对等之间开始协商(18:24:33)，经过的时间是

20 秒；在这个时间间隔内，建立起了 TCP 连接。于是 BGP 从 Idle 状态转到 Active 状态，整个协商过程大约持续 10 秒。

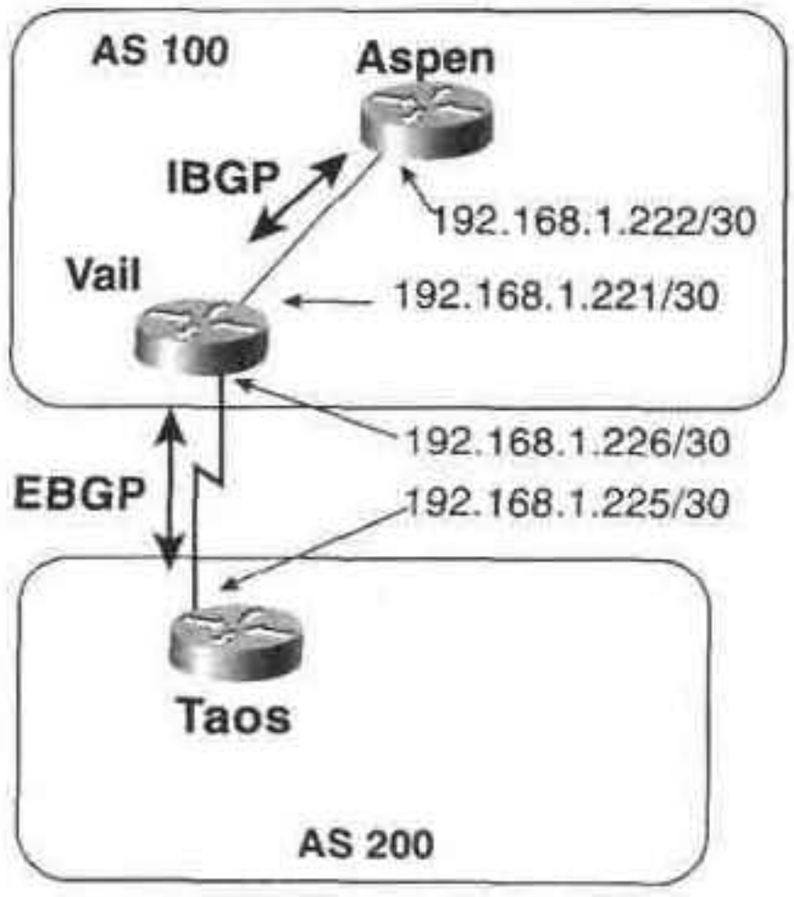


图 3-2　在 Vail 和 Aspen 之间使用 IBGP

例 3-3　图 3-2 中 Vail 路由器的配置

```
router bgp 100
 neighbor 192.168.1.222 remote-as 100
 neighbor 192.168.1.225 remote-as 200
```

例 3-4　Aspen 上 debug ip bgp events 的显示

```
Aspen#debug ip bgp events
BGP events debugging is on
Aspen#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Aspen(config)#router bgp 100
Aspen(config-router)#neighbor 192.168.1.221 remote-as 100
Aspen(config-router)#^Z
Aspen#
18:24:13: %SYS-5-CONFIG_I: Configured from console by console
Aspen#
18:24:33: BGP: 192.168.1.221 went from Idle to Active
18:24:41: BGP: 192.168.1.221 went from Active to OpenSent
18:24:42: BGP: 192.168.1.221 went from OpenSent to OpenConfirm
18:24:42: BGP: 192.168.1.221 went from OpenConfirm to Established
18:24:43: BGP: 192.168.1.221 computing updates, neighbor version 0, table version
n 1, starting at 0.0.0.0
18:24:43: BGP: 192.168.1.221 update run completed, ran for 0ms, neighbor version
 0, start version 1, throttled to 1, check point net 0.0.0.0
Aspen#
```

例 3-5 显示了 Aspen 的邻居信息的一部分。

例 3-5　Aspen 的邻居信息显示了 Vail 的路由器 ID 来自它的一个物理端口

```
Aspen#show ip bgp neighbors
BGP neighbor is 192.168.1.221,  remote AS 100, internal link
 Index 1, Offset 0, Mask 0x2
  BGP version 4, remote router ID 192.168.1.226
  BGP state = Established, table version = 1, up for 00:03:46
  Last read 00:00:46, hold time is 180, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 5 seconds
  Received 6 messages, 0 notifications, 0 in queue
  Sent 6 messages, 0 notifications, 0 in queue
  Prefix advertised 0, suppressed 0, withdrawn 0
  Connections established 1; dropped 0
  Last reset never
  0 accepted prefixes consume 0 bytes
  0 history paths consume 0 bytes
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 192.168.1.222, Local port: 179
Foreign host: 192.168.1.221, Foreign port: 11000
```

注意 Vail 的路由器 ID 是 192.168.1.226，它是 Vail 与 Taos 相连的端口的地址。选择一个 BGP 路由器 ID 的规则与选择一个 OSPF 路由器 ID 的规则相同：

- 路由器在它的任何 Loopback 接口上选择数字最高的 IP 地址。
- 如果没有配置 IP 地址的 Loopback 接口，路由器会在它所有的物理接口选择数字最高的 IP 地址。得到路由器 ID 的接口不要求运行 BGP。

因为 Vail 没有配置 Loopback 端口，路由器在它的物理接口中选择数字最高的 IP 地址。使用与 Loopback 接口有关的地址有两点好处：

- Loopback 接口比任何物理接口都稳定。当路由器启动时，它就处于激活状态，只有整个路由器完全瘫痪的时候，它才失效。
- 将可预知的或者可识别的地址用作路由器 ID，网络管理者可以有更多的退路。

Cisco 的 BGP 会继续使用从一个物理接口学习到的路由器 ID，即使该接口随后会出现故障或者被删除掉。因此，Loopback 接口的稳定性只是一个较小的优势。它主要的优势是控制路由器 ID 的能力，该地址可以很容易地与其他的 IP 地址区分开。

例 3-6 显示如何用一个唯一的路由器 ID 配置 Vail。

例 3-6　用一个唯一的路由器 ID 配置 Vail

```
interface loopback 0
 ip address 192.168.255.254 255.255.255.255
!
router bgp 100
 neighbor 192.168.1.222 remote-as 100
 neighbor 192.168.1.225 remote-as 200
```

但是，在正在工作的 BGP 路由器上配置一个 Loopback 接口不会改变路由器 ID。必须在 Vail 上使用 **clear ip bgp** 命令(在“配置路由策略”一节中将有详细的讨论)来清掉它所有的 BGP 会话。再看一下例 3-7 中 Aspen 的邻居信息，该信息显示出 Vail 的路由器 ID 现在是 Loopback0

的地址。

例 3-7 中我们感兴趣的另一点是，与例 3-5 比起来，它们的表版本号不同。Vail 的会话被重置以后，表版本号增加到 2。这个变化也可以在 **Connections established; drop ped** 信息组内反映出来。这些信息组不应该经常发生变化；如果它们经常发生变化，可能说明 Vail 有一个不稳定的邻居。

例 3-7　重置的 BGP 会话以后，Vail 的路由器 ID 是它的环回地址

```
Aspen#show ip bgp neighbors
BGP neighbor is 192.168.1.221,  remote AS 100, internal link
 Index 1, Offset 0, Mask 0x2
   BGP version 4, remote router ID 192.168.255.254
   BGP state = Established, table version = 2, up for 00:00:42
   Last read 00:00:42, hold time is 180, keepalive interval is 60 seconds
   Minimum time between advertisement runs is 5 seconds
   Received 37 messages, 0 notifications, 0 in queue
   Sent 37 messages, 0 notifications, 0 in queue
   Prefix advertised 0, suppressed 0, withdrawn 0
   Connections established 2; dropped 1
   Last reset 00:00:51, due to Peer closed the session
   0 accepted prefixes consume 0 bytes
   0 history paths consume 0 bytes
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 192.168.1.222, Local port: 179
Foreign host: 192.168.1.221, Foreign port: 11003
```

你也可以为运行 BGP 的路由器人工设置一个路由器 ID，而不用物理的以及 Loopback 接口的地址。完成这个工作使用 **bgp router-id**。例如，在例 3-8 中将 Vail 的 BGP 路由器 ID 设置为 1.1.3.2。

例 3-8　人工设置 BGP 路由器 ID

```
interface loopback 0
 ip address 192.168.255.254 255.255.255.255
!
router bgp 100
 bgp router-id 1.1.3.2
 neighbor 192.168.1.222 remote-as 100
 neighbor 192.168.1.225 remote-as 200
```

当因为其他的原因，例如 OSPF 路由器 ID 或者 SNMP 功能需要 Loopback 接口，但是该接口上的 IP 地址与你所希望的 BGP 路由器 ID 不同的时候，**bgp router-id** 命令是很有用的。

3.1.2　案例研究：向 BGP 中注入 IGP 路由

在第 2 章曾强调过在 AS 边界，出去的路由公布会影响到入业务量，进来的路由公布会影响到出业务量。因此，应该将出去的和进来的路由公布分开考虑。本节通过考察向 BGP 注入路由的基本方法开始讨论 BGP 的路由公布。

图 3-3 显示 AS 200 把 EIGRP 当作 IGP。Taos 必须向它的 EBGP 对端公布 3 个地址：通过 EIGRP 学习到的 192.168.200.0/24、直接连接到 Taos 的 192.168.100.0/24 以及连接 Taos 和

AngelFire 的 192.168.1.216/30。前两个地址都是 C 类地址，最后一个是个子网地址。192.168.1.0 的其他子网在 AS 200 的外面，因此并不需要公布主要网络地址，只需要公布子网即可。

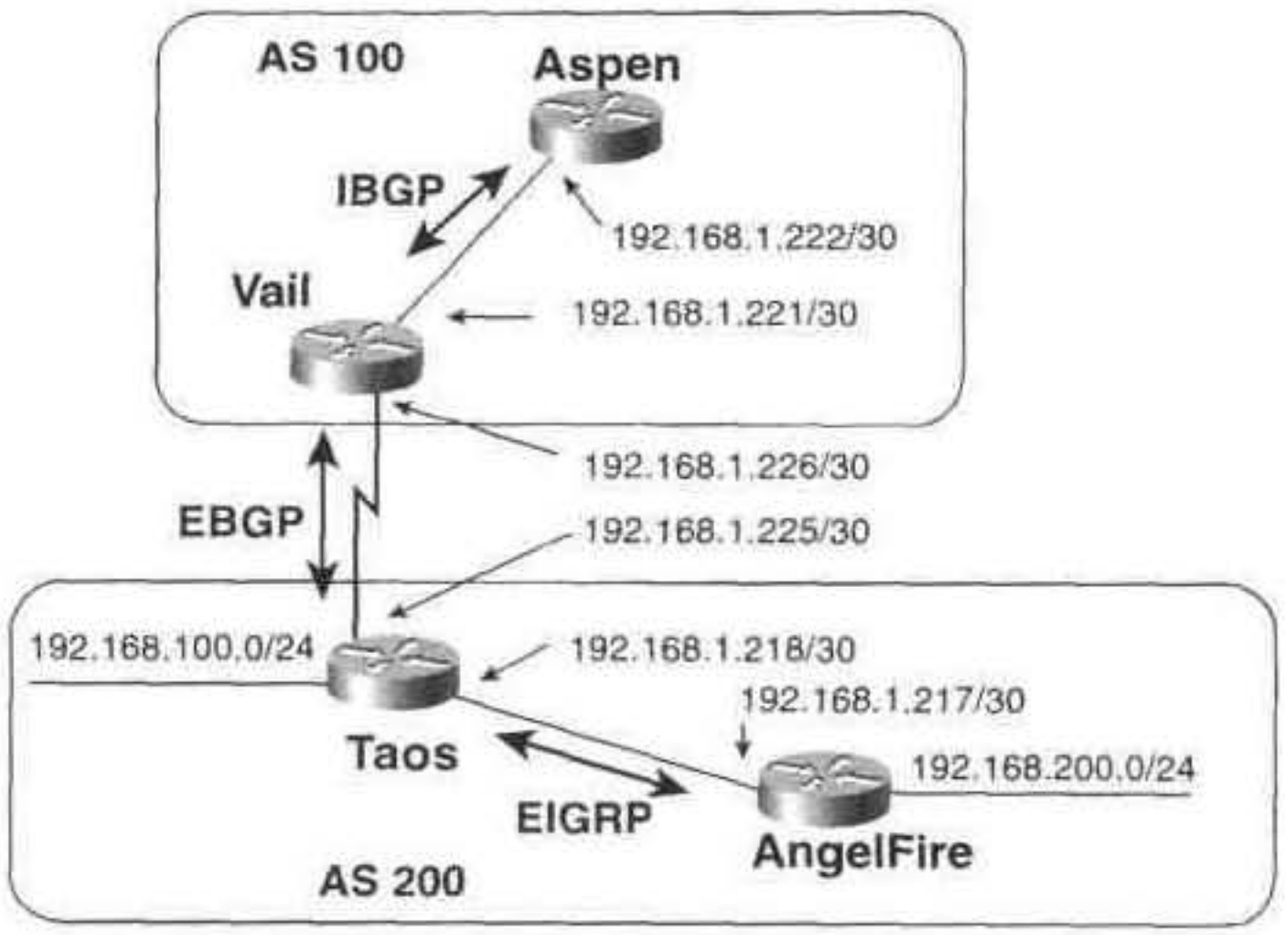


图 3-3　AS 200 将 EIGRP 当作 IGP

例 3-9 给出了 Taos“首先通过(first-pass)”的配置。

例 3-9　Taos 的基本的 EIGRP 和 BGP 配置

```
router eigrp 200
 passive-interface Serial0
 network 192.168.1.0
 network 192.168.100.0
!
router bgp 200
 redistribute eigrp 200
 neighbor 192.168.1.226 remote-as 100
```

例 3-10 给出了 Vail's BGP 路由表的结果，例中显示 Taos 正确地宣告了 192.168.100.0/24 和 192.168.200.0/24，但是将子网 192.168.1.216/30 归纳到了主要网络中。所有的 EIGRP 网络

例 3-10　Vail 的 BGP 路由表

```
Vail#show ip bgp
BGP table version is 15, local router ID is 192.168.255.254
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.0      192.168.1.225       281600             0 200 ?
*> 192.168.100.0    192.168.1.225            0             0 200 ?
*> 192.168.200.0    192.168.1.225       409600             0 200 ?
Vail#
```

都通过 EBGP 链路进行宣告。注意在这个配置中没有用 **redistribute** 命令来确定度量。结果是，所有路由的度量缺省都为 EIGRP 的度量，如例 3-11 中 Taos 的路由表所示。与它直接相连的网络的度量为 0，网络 192.168.200.0/24 的度量为 409600。你可以通过 **default-metric** 命令来改变这种缺省选择度量的方式。

> 注：　BGP 的度量是 MULTI_EXIT_DISC。关于该属性的使用与管理在“案例研究一节中的：使用 MULTI_EXIT_DISC 属性”中有说明。

例 3-11　Taos 的路由表显示 EIGRP 度量和 Vail 的 BGP 路由表中的度量相同

```
Taos#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
       T - traffic engineered route

Gateway of last resort is not set

D    192.168.200.0/24 [90/409600] via 192.168.1.217, 00:52:09, Ethernet0
     192.168.1.0/24 is variably subnetted, 3 subnets, 2 masks
D       192.168.1.0/24 is a summary, 00:52:11, Null0
C       192.168.1.224/30 is directly connected, Serial0
C       192.168.1.216/30 is directly connected, Ethernet0
C    192.168.100.0/24 is directly connected, Ethernet1
Taos#
```

AS 200 内的两个主要网络都被正确地公布了，但是在例 3-9 中可以看到子网 192.168.1.216/30 被归纳到了主要网络当中。出现这种情况的原因是虽然 BGP-4 是无类别的，但是缺省的情况下，它会在边界进行归纳。在图 3-3 的网络互连中，这种归纳没有问题。Vail 与 192.168.1.0 的其他两个子网直接相连，因此它知道两条更具体的路由。但是，当网络的规模增大并且在其他路由器上用到了该网络的其他子网时，归纳会导致错误的路由。为了关闭 BGP 的自动归纳功能，按照例 3-12 来配置 Vail。

例 3-12　配置 Vail 来关闭 BGP 的自动归纳功能

```
router eigrp 200
 passive-interface Serial0
 network 192.168.1.0
 network 192.168.100.0
!
router bgp 200
 redistribute eigrp 200
 neighbor 192.168.1.226 remote-as 100
 no auto-summary
```

例 3-13 给出了关闭自动归纳功能以后 Vail 的 BGP 表。现在公布了子网 192.168.1.0，而且除了子网以外，仍然公布主要网络 192.168.1.0。在例 3-12 里 Taos 的路由表中给出了出现

这种结果的原因。EIGRP 也执行自动路由归纳功能，同时在路由表中它已经向 Null0 加入了一条归纳路由。BGP 得到了除了子网之外的这条归纳路由并且将它公布给 Vail。

例 3-13　Taos 关闭了 BGP 自动归纳功能以后，Vail 的 BGP 表

```
Vail#show ip bgp
BGP table version is 17, local router ID is 192.168.255.254
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.0      192.168.1.225       281600             0 200 ?
*> 192.168.1.216/30 192.168.1.225            0             0 200 ?
*> 192.168.1.224/30 192.168.1.225            0             0 200 ?
*> 192.168.100.0    192.168.1.225            0             0 200 ?
*> 192.168.200.0    192.168.1.225       409600             0 200 ?
Vail#
```

用同样的命令 **no auto-summary**，可以关闭 Vail 路由器的 EIGRP 自动归纳功能，参见例 3-14 的说明。

例 3-14　为了关闭 EIGRP 的自动归纳功能，Vail 的配置

```
router eigrp 200
 passive-interface Serial0
 network 192.168.1.0
 network 192.168.100.0
 no auto-summary
!
router bgp 200
 redistribute eigrp 200
 neighbor 192.168.1.226 remote-as 100
 no auto-summary
```

例 3-15 给出了 Taos 关闭 EIGRP 自动归纳功能以后，Vail 处得到的 BGP 表。

例 3-15　Taos 关闭自动归纳功能以后，Vail 的 BGP 表

```
Vail#show ip bgp
BGP table version is 20, local router ID is 192.168.255.254
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.216/30 192.168.1.225            0             0 200 ?
*> 192.168.1.224/30 192.168.1.225            0             0 200 ?
*> 192.168.100.0    192.168.1.225            0             0 200 ?
*> 192.168.200.0    192.168.1.225       409600             0 200 ?
Vail#
```

利用再分发向 BGP 注入路由的优点是内部变化的宣告不会改变 BGP 配置或者改变很少。如果在 AS 200 的 EIGRP 域内加入或者去掉一个网络，该变化会自动公布给 Vail。但是，公布每一条 IGP 路由也是 IGP 向 BGP 再分发的一个主要的缺点。例如，例 3-15 所讲，自治系

统 100 和 200 的管理人员可能不希望从 Taos 向 Vail 公布子网 192.168.1.224/30。如果不应该公布一个子网，就必须使用路由过滤器。在本章的最后“路由策略”一节中会通过几个案例研究来说明配置路由过滤器的几种选项。

当把一条 IGP 路由再分发给 BGP 时，路由过滤器通常是必须的。缺省的情况下，每条被 IGP 识别的路由都要进行再分发。AS 的管理者可能只想公布 IGP 路由的一个子网，这样就必须过滤掉其他的，或者，可能一个多宿主的 AS 不想为它每一个邻居自治系统做转接网络，此时必须使用路由过滤器，从而阻止将从一个 AS 学习到的外部路由公布给其他的自治系统。这样就存在一个路由反馈(route feedback)的问题，从 EBGP 接收到的外部路由被公布给 IGP，然后通过再分发，这些路由又从 IGP 回到 EBGP。最好的办法是要求使用路由过滤器来保证只公布正确的路由。实际上，很少采用将 IGP 前缀再分发给 BGP 的这种方法，因为这种方法缺少准确性。

可以替代再分发 IGP 路由给 BGP 的一个的方法是使用 **network** 命令。在第一章我们曾经讨论过，在 EGP 和 BGP 下使用该命令与在 IGP 下使用该命令有所不同。在 IGP 下使用该命令时，它确定的是一个或一组将要使用路由协议的接口的地址。在 EGP 和 BGP 下使用该命令时，**network** 确定的是要公布的 IP 前缀。对于使用通过命令而确定的前缀，BGP 在路由表中进行查找，如果在路由表中有与 **network** 前缀完全匹配的条目，就将该前缀放到 BGP 表中并且将之公布。

例 3-16 给出了使用 **network** 命令而不是使用再分发的 Taos 的配置。

例 3-16　用 **network** 命令配置 Taos

```
router eigrp 200
 passive-interface Serial0
 network 192.168.1.0
 network 192.168.100.0
!
router bgp 200
 network 192.168.1.216 mask 255.255.255.252
 network 192.168.100.0
 network 192.168.200.0
 neighbor 192.168.1.226 remote-as 100
```

主要网络 192.168.100.0 和 192.168.200.0 是单独规定的。对于子网 192.168.1.216，也规定了一个掩码。子网和掩码可以只在 BGP-4 下可以定义；在 EGP 或早期版本的 BGP 下，它们都是有类别的，只能规定主要网络。

注意在这种配置下，在 EIGRP 和 BGP 中不用使用 **no auto-summary** 命令，因为不会出现再分发，因此没有必要关闭自动归纳功能。例 3-17 给出了该配置的结果。

与例 3-15 不同，在这里，没有公布子网 192.168.1.224/30，因为它不是通过 **network** 命令来定义的。比起再分发，管理者对这种情况有更多的控制权，因此没必要使用过滤器。比较例 3-15 和例 3-17 可以看出，它们的 ORIGIN 编码不同。例 3-15 中的再分发路由中用一个?来指示“Incomplete”的 ORIGIN，而在例 3-17 中，路由后都标记了一个 i，指示 IGP 的 ORIGIN。因为在第二章所讨论的 BGP 决定过程中，给 ORIGIN 编码 1(IGP) 一个高于编码 3(Incomplete)的首选权，因此这种标记在特定的环境下会产生不同的效果。

例 3-17　使用 BGP network 命令将 Taos 重新配置以后，Vail 的 BGP 表

```
Vail#show ip bgp
BGP table version is 36, local router ID is 192.168.255.254
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.216/30 192.168.1.225            0             0 200 i
*> 192.168.100.0    192.168.1.225            0             0 200 i
*> 192.168.200.0    192.168.1.225       409600             0 200 i
Vail#
```

使用 **network** 命令最多只能确定 200 个地址，如果你要通过 BGP 连接公布更多的地址，你必须使用再分发。

3.1.3　案例研究：向 IGP 注入 BGP 路由

从一个 EBGP 邻居学习到的前缀被自动加入到路由表中。例如，在图 3-4 中，AS 300 宣告了两条路由：192.168.250.0/24 和 192.168.1.21/30，在这个例子中，AS 300 的 IGP 以及路由器 Tahoe 的配置是不重要的。重要的一点是，Tahoe 公布给它外部对端的前缀以可到达的状态出现在 Taos 的路由表中,而且 Ping　AS　300 中的一个目的地成功了(参见例 3-18)。因为没有公布 Taos 串行端口的子网 192.168.1.224/30,在此使用了扩展 Ping。路由表中学习到的 BGP 路由被标记了一个 **B**。

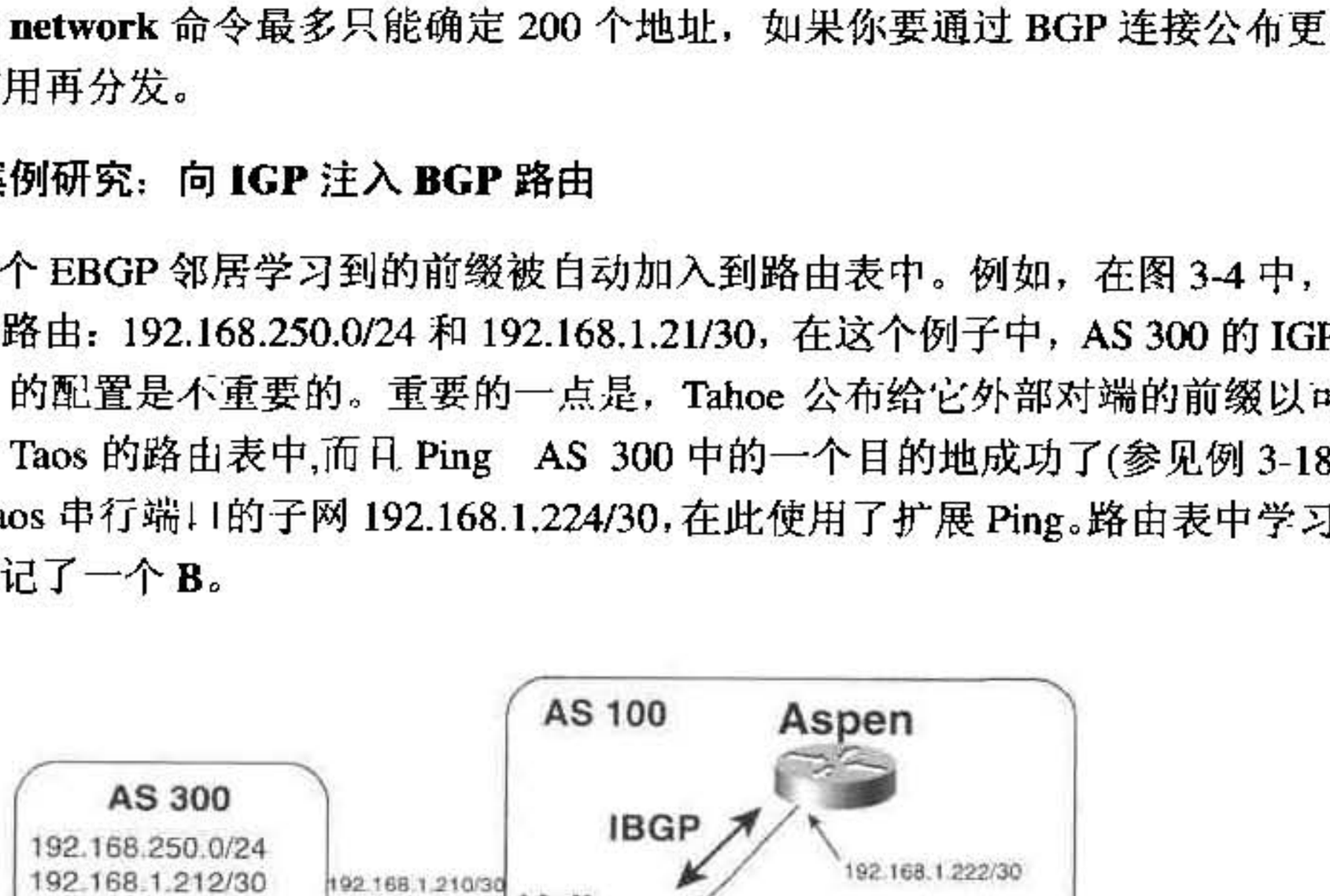


图 3-4　将 AS 300 加入到图 3-3 的拓扑中

虽然对于 Taos 来讲，AS 300 的网络是可以到达的，但是在这些网络对于 AS 200 内部的路由器来讲是可到达的之前，必须将 BGP 路由公布给 EIGRP。完成这项工作的一个办法就是在 Taos 处执行再分发，如例 3-19 中的配置所示。

例 3-18　成功地 Ping 通了 AS 300 中的一个地址

```
Taos#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
       T - traffic engineered route
Gateway of last resort is not set
D    192.168.200.0/24 [90/409600] via 192.168.1.217, 00:25:37, Ethernet0
B    192.168.250.0/24 [20/0] via 192.168.1.226, 16:18:12
     192.168.1.0/24 is variably subnetted, 4 subnets, 2 masks
D       192.168.1.0/24 is a summary, 00:25:43, Null0
C       192.168.1.224/30 is directly connected, Serial0
C       192.168.1.216/30 is directly connected, Ethernet0
B       192.168.1.212/30 [20/0] via 192.168.1.226, 16:18:12
C    192.168.100.0/24 is directly connected, Ethernet1

Taos#ping
Protocol [ip]:
Target IP address: 192.168.250.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 192.168.100.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.250.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/64/112 ms
Taos#
```

例 3-19　将 BGP 路由公布给 EIGRP

```
router eigrp 200
 redistribute bgp 200 metric 10000 100 255 1 1500
 passive-interface Serial0
 network 192.168.1.0
 network 192.168.100.0
!
router bgp 200
 network 192.168.1.216 mask 255.255.255.252
 network 192.168.100.0
 network 192.168.200.0
 neighbor 192.168.1.226 remote-as 100
```

例 3-20 显示了将 AS 300 的前缀公布给了 AngelFire 并且那些目的地是可到达的。但是许多再分发涉及到的问题会出现在入路由和出路由上。再分发会接受每一条 BGP 路由，但是管理者可能只想再分发 BGP 路由的一个子网。在这种情况下，再次需要使用路由过滤器来抑制不必要的路由。

警告 不将 BGP 路由再分发到 IGP 存在着另外一个极其重要的原因。整个 Internet 路由表包含着多于 80000 个的前缀，试图处理如此多的路由会导致 IGP 处理过程“阻塞”。再分发整个 Internet 路由表或者路由表的大部分，不可避免地会导致网络崩溃。本章给出了有关再分发的例子，它对于具有有限前缀的企业网来讲是有用的，但是你永远不要在一个面向 Internet 的路由器上使用 BGP 到 IGP 的再分发。

例 3-20　Taos 将它的 BGP 学习到的路由再分发给 EIGRP

```
AngelFire#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

D    192.168.100.0/24 [90/409600] via 192.168.1.218, 01:14:22, Ethernet0/0
     192.168.1.0/24 is variably subnetted, 4 subnets, 2 masks
D       192.168.1.224/30 [90/2195456] via 192.168.1.218, 01:16:44, Ethernet0/0
C       192.168.1.216/30 is directly connected, Ethernet0/0
D EX    192.168.1.212/30 [170/307200] via 192.168.1.218, 00:03:55, Ethernet0/0
D EX 192.168.250.0/24 [170/307200] via 192.168.1.218, 00:03:55, Ethernet0/0
C    192.168.200.0/24 is directly connected, Ethernet0/1

AngelFire#ping 192.168.250.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.250.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/8/12 ms
AngelFire#
```

为了更好地控制公布给 AS 200 的路由，你可以像例 3-21 所示的那样，使用静态路由。

例 3-21　通过静态路由来控制宣告到 AS 200 的路由

```
router eigrp 200
 redistribute static metric 10000 100 255 1 1500
 passive-interface Serial0
 network 192.168.1.0
 network 192.168.100.0
!
router bgp 200
 network 192.168.1.216 mask 255.255.255.252
 network 192.168.100.0
 network 192.168.200.0
 neighbor 192.168.1.226 remote-as 100
!
ip route 192.168.250.0 255.255.255.0 Serial0
```

在这个配置中，只将 192.168.250.0/24 公布给了 AS 200。如例 3-22 所示，AngelFire 不知道子网 192.168.1.212/30。在这个配置中使用静态路由能够更好地保护 AS 200，从而使它免

受不稳定性的影响。如果 192.168.250.0/24 在 AS 300 中摆动，这个变化只会公布给 AS 200 中的 Taos，而不会公布得更远。

例 3-22　没有将子网 192.168.1.212/30 公布给 AngelFire

```
AngelFire#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR

Gateway of last resort is not set

D    192.168.100.0/24 [90/409600] via 192.168.1.218, 00:14:33, Ethernet0/0
     192.168.1.0/24 is variably subnetted, 3 subnets, 2 masks
D       192.168.1.224/30 [90/2195456] via 192.168.1.218, 00:14:33, Ethernet0/0
C       192.168.1.216/30 is directly connected, Ethernet0/0
D EX 192.168.250.0/24 [170/307200] via 192.168.1.218, 00:11:17, Ethernet0/0
C    192.168.200.0/24 is directly connected, Ethernet0/1

AngelFire#ping 192.168.250.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.250.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/7/8 ms
AngelFire#
```

当然，对于单宿主的 AS，例如图 3-4 中的 AS 200，几乎没有理由要把任何外部路由公布到 AS 中。除非需要公布一条特殊路由给 AS，否则如例 3-23 所示的，缺省的路由已经足够了。

例 3-23　在单一宿主的 AS 内配置一条缺省路由

```
router eigrp 200
 redistribute static metric 10000 100 255 1 1500
 passive-interface Serial0
 network 192.168.1.0
 network 192.168.100.0
!
router bgp 200
 network 192.168.1.216 mask 255.255.255.252
 network 192.168.100.0
 network 192.168.200.0
 neighbor 192.168.1.226 remote-as 100
!
ip classless
ip route 0.0.0.0 0.0.0.0 Serial0
```

在例 3-22 的配置中，Taos 生成了一条缺省路由并且将它公布给所有的运行 EIGRP 的路由器；但是，也可以配置 BGP 从而生成一条缺省路由。从 Vail 向它的 BGP 邻居公布缺省路由，使用例 3-24 中的配置。

到 Null0 接口的缺省路由是静态产生的，并且路由是通过 **network** 命令来公布的。例 3-24

的配置是假设 Vail 具有所有的路由信息。所有的数据包都被转发到 Vail；丢弃任何不与更具体路由匹配而是与静态路由相匹配的目的地地址。

在一些设计实例里，通常将一条缺省路由发送给某些邻居而不再发给其他的邻居。为了从 Vail 发送一条缺省路由给 Taos，但是不发给 Vail 其他的任何邻居，使用例 3-25 中的配置。

例 3-24　配置一条到 BGP 邻居的缺省路由

```
router bgp 100
 network 0.0.0.0
 neighbor 192.168.1.218 remote-as 300
 neighbor 192.168.1.222 remote-as 100
 neighbor 192.168.1.225 remote-as 200
!
ip route 0.0.0.0 0.0.0.0 Null0
```

例 3-25　配置一条到特定 BGP 邻居的缺省路由

```
router bgp 100
 neighbor 192.168.1.218 remote-as 300
 neighbor 192.168.1.222 remote-as 100
 neighbor 192.168.1.225 remote-as 200
 neighbor 192.168.1.225 default-originate
```

BGP 的 **neighbor default-originate** 命令与 OSPF 的 **default-information-originate always** 命令相似，在这种命令情况下，不管路由器实际上有没有缺省路由，它都要公布一条缺省路由。注意在这个配置中，前一种配置中的静态路由已经不存在了；但是，到 0.0.0.0/0 的路由要继续公布给 Taos，如例 3-26 所示。例 3-26 还给出了 Tahoe 的路由表。从这个路由表中可以看出，与 Taos 不同，Tahoe 没有到 0.0.0.0/0 的路由条目。

例 3-26　公布一条缺省路由给 Taos，但是没公布给 Tahoe

```
Taos#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
       T - traffic engineered route

Gateway of last resort is 192.168.1.226 to network 0.0.0.0

D    192.168.200.0/24 [90/409600] via 192.168.1.217, 02:06:34, Ethernet0
B    192.168.250.0/24 [20/0] via 192.168.1.226, 00:46:03
     192.168.1.0/24 is variably subnetted, 4 subnets, 2 masks
D       192.168.1.0/24 is a summary, 02:06:34, Null0
C       192.168.1.224/30 is directly connected, Serial0
C       192.168.1.216/30 is directly connected, Ethernet0
B       192.168.1.212/30 [20/0] via 192.168.1.226, 00:46:04
C    192.168.100.0/24 is directly connected, Ethernet1
B*   0.0.0.0/0 [20/0] via 192.168.1.226, 00:47:03
Taos#

Tahoe#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
```

```
Gateway of last resort is not set

B    192.168.100.0 [20/0] via 192.168.1.209, 00:48:26
     192.168.1.0 255.255.255.252 is subnetted, 3 subnets
B       192.168.1.216 [20/0] via 192.168.1.209, 00:48:26
C       192.168.1.208 is directly connected, Serial0
C       192.168.1.212 is directly connected, Serial1
C    192.168.250.0 is directly connected, Ethernet0
B    192.168.200.0 [20/0] via 192.168.1.209, 00:48:27
Tahoe#
```

向一个 BGP 邻居公布一条缺省路由并不会抑制更具体的路由。在例 3-26 中可以看到来自 AS 300 的路由仍然出现在 Taos 的路由表中，在某些情况下，这是我们所希望的。例如，一个 ISP 可能会向他的一个客户发送到其他所有客户的路由(部分 Internet 路由表)，同时也向 Internet 的其他部分发送一条缺省路由。该情况在客户多宿主到一个 ISP 的时候很有用。于是客户网络可以选择到 ISP 客户最好的路径，同时通过缺省路由到其他外部的目的地。

如果只想发送缺省路由，那么就要通过路由过滤器来抑制所有更具体的路由。在例 3-27 的配置中，使用 **neighbor distribute-list** 命令，这是过滤 BGP 路由的一种方法。在“路由策略”一节中说明了其他方法。

例 3-27　用 **neighbor distribute-list** 命令过滤 BGP 路由

```
router bgp 100
 neighbor 192.168.1.210 remote-as 300
 neighbor 192.168.1.222 remote-as 100
 neighbor 192.168.1.225 remote-as 200
 neighbor 192.168.1.225 default-originate
 neighbor 192.168.1.225 distribute-list 1 out
!
access-list 1 permit 0.0.0.0
access-list 1 deny any
```

3.1.4　案例研究：没有 IGP 的 IBGP

在图 3-5 中，在 AS 100 中加入了另外一个路由器；它通过 EBGP 与另外一个 AS 相连。现在 AS 100 是一个转接 AS，它携带的业务量既不是由它发起的也不在它这儿终止。

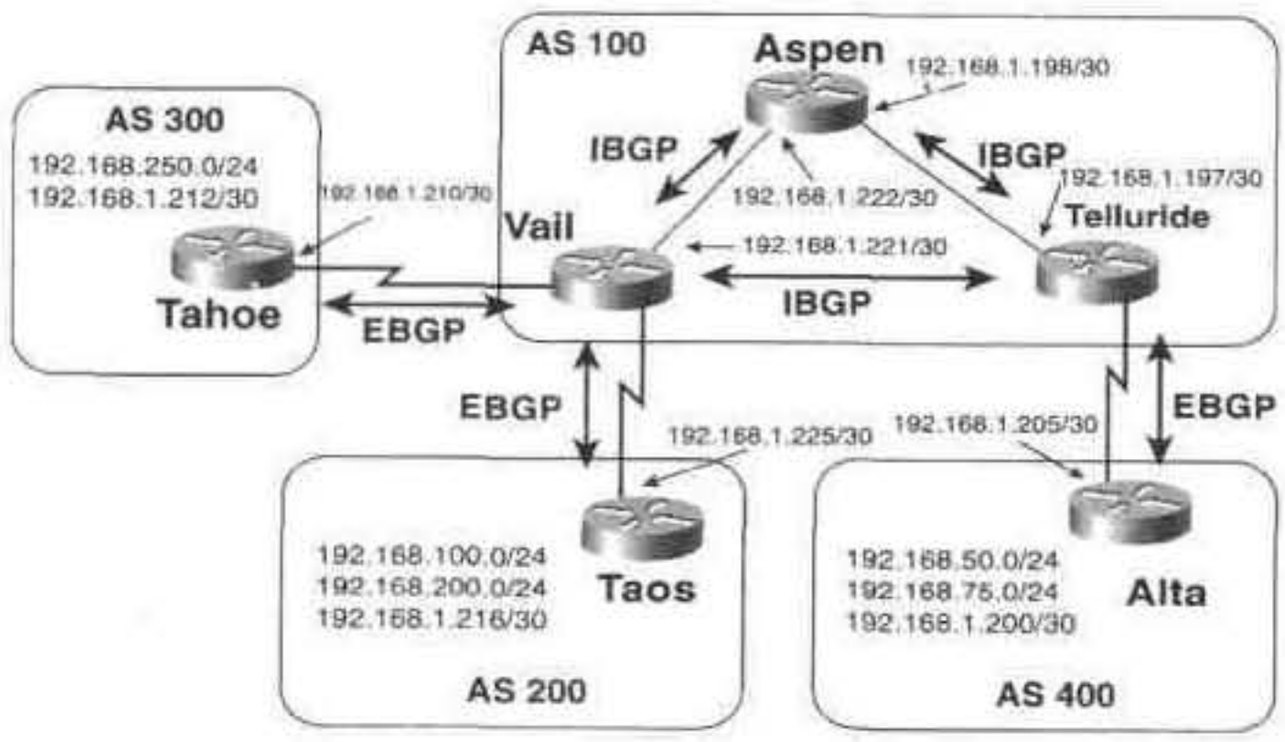


图 3-5　AS 100 运行 IBGP，它携带 AS 400 和其他两个自治系统之间转接的业务量

为了携带转接的业务量，AS 100 内部的路由器通过 IBGP 呈全网状连接，参见例 3-28 中的配置。

例 3-28　将 AS 100 内部的路由器配置为通过 IBGP 而形成的全网状连接

Vail

```
router bgp 100
 no synchronization
 network 192.168.1.208 mask 255.255.255.252
 network 192.168.1.224 mask 255.255.255.252
 neighbor 192.168.1.197 remote-as 100
 neighbor 192.168.1.210 remote-as 300
 neighbor 192.168.1.222 remote-as 100
 neighbor 192.168.1.225 remote-as 200
```

Aspen

```
router bgp 100
 no synchronization
 network 192.168.1.196 mask 255.255.255.252
 network 192.168.1.220 mask 255.255.255.252
 neighbor 192.168.1.197 remote-as 100
 neighbor 192.168.1.221 remote-as 100
```

Telluride

```
router bgp 100
 no synchronization
 network 192.168.1.204 mask 255.255.255.252
 neighbor 192.168.1.198 remote-as 100
 neighbor 192.168.1.205 remote-as 400
 neighbor 192.168.1.221 remote-as 100
```

例 3-29 给出了 Alta 的路由表；执行的几个 Ping 命令说明 AS 200 和 AS 300 内的目的地是可到达的。

当如图 3-5 所示来配置 IBGP 时，记住下面的几个要点：

必须关闭同步功能。

每个 IBGP 路由器必须与其他每一个 IBGP 路由器建立对等关系。

必须能识别与 IBGP 路由器相连的所有网络和子网。

在例 3-28 的配置中可以看到通过 **no synchronization** 命令关闭了同步功能。在第三章同步的规则我们曾讲到，一个路由器不能向一个 EBGP 对端公布 IBGP 路由，除非 IGP 能够识别该路由。换句话说，也就是 BGP 必须与 IGP 同步。无论是再分发还是 **network** 命令都不能公布一条不在路由表内的路由。

如果将学习到的 IBGP 路由加入到路由表中，同步的规则将被撤销。虽然 IGP 有可能不知道一条 IBGP 路由，但是再分发或者 **network** 命令可以与路由表中的该路由匹配并且公布它。因此当同步功能是打开的时候，IBGP 路由不会加入到路由表中。

例 3-30 显示了当同步功能是打开的时候，在 Aspen 处出现的问题。BGP 表显示路由器已经学习到了由它的 IBGP 对等公布的所有路由，但是路由表中却显示没有路由加入进来。尽管 Aspen 没有 EBGP 对等，但是转发却受到了影响。例如，如果 Telluride 转发一个数据包到 192.168.250.1，在 Aspen 的路由表中没有到该目的地的路由条目，这个数据包就会被丢弃。

例 3-29　来自 AS 200 和 AS 300 的路由要通过 AS 100 和 AS 400 内的 IBGP 连接

```
Alta#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
B    192.168.100.0 [20/0] via 192.168.1.206, 02:02:59
C    192.168.75.0 is directly connected, Ethernet1
C    192.168.50.0 is directly connected, Ethernet0
     192.168.1.0 255.255.255.252 is subnetted, 8 subnets
B       192.168.1.224 [20/0] via 192.168.1.206, 02:02:59
C       192.168.1.200 is directly connected, Ethernet2
C       192.168.1.204 is directly connected, Serial0
B       192.168.1.196 [20/0] via 192.168.1.206, 02:03:30
B       192.168.1.216 [20/0] via 192.168.1.206, 02:02:59
B       192.168.1.220 [20/0] via 192.168.1.206, 02:03:30
B       192.168.1.208 [20/0] via 192.168.1.206, 02:02:59
B       192.168.1.212 [20/0] via 192.168.1.206, 02:02:59
B    192.168.250.0 [20/0] via 192.168.1.206, 02:02:59
B    192.168.200.0 [20/0] via 192.168.1.206, 02:03:00

Alta#ping 192.168.250.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.250.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/5/8 ms

Alta#ping 192.168.200.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.200.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/9/12 ms
Alta#
```

例 3-30　当打开同步功能的时候，学习到的 IBGP 路由无法加入到路由表中去

```
Aspen#show ip bgp
BGP table version is 3, local router ID is 192.168.1.222
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.196/30 0.0.0.0                  0         32768 i
* i192.168.1.200/30 192.168.1.205            0    100      0 400 i
* i192.168.1.204/30 192.168.1.197            0    100      0 i
* i192.168.1.208/30 192.168.1.221      0  100    0 i
* i192.168.1.212/30 192.168.1.210            0    100      0 300 i
* i192.168.1.216/30 192.168.1.225            0    100      0 200 i
*> 192.168.1.220/30 0.0.0.0                  0         32768 i
* i192.168.1.224/30 192.168.1.221            0    100      0 i
* i192.168.50.0     192.168.1.205            0    100      0 400 i
* i192.168.75.0     192.168.1.205            0    100      0 400 i
* i192.168.100.0    192.168.1.225            0    100      0 200 i
* i192.168.200.0    192.168.1.225       409600    100      0 200 i
* i192.168.250.0    192.168.1.210            0    100      0 300 i
```

```
Aspen#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
       T - traffic engineered route
Gateway of last resort is not set

     192.168.1.0/30 is subnetted, 2 subnets
C       192.168.1.196 is directly connected, Ethernet1
C       192.168.1.220 is directly connected, Ethernet0
Aspen#
```

在例 3-31 中，关闭了 Aspen 处的同步功能，于是 IBGP 路由被加入到路由表中。

> 注：　如果你关闭一个工作当中的 BGP 过程的同步功能，在这种变化起作用之前，你必须用 **clear ip bgp** 命令重置该 BGP 连接。在“重置 BGP 连接”一节中有关于这个命令使用情况的完整解释。

在图 3-5 和 AS 100 中的路由器的配置中可以看出，三个路由器中的每一个都和另外两个建立了对等，这是因为一个路由器不会把从一个 IBGP 对等学习到的路由传给另外的 IBGP 对端。例如，Vail 从它和 Telluride 的 IBGP 会话中学习到了 AS 400 的地址。如果这个会话不存在，Vail 就不能从 Aspen 处学习到路由。通过相应的到对等的 IBGP 连接，Aspen 也从 Vail 和 Telluride 学习到了路由。如果 Aspen 不能学习到路由，那么它将不能在 Telluride 和 Vail 之间转发数据包。

当向一个 IBGP 对端公布一条学习到的 EBGP 路由，该路由的下一跳地址没有变化。观察一下例 3-30 中 Aspen 的 BGP 表，其中到其他自治系统中目的地的所有路由的下一跳地址都是发起该 EBGP 路由的路由器的接口地址。例如，到 192.168.200.0/24 路由的下一跳地址是 Taos 的接口地址 192.168.1.225，将这些下一跳地址加到路由表中。结果是，所有的 IBGP 路由器都必须知道如何到达下一跳地址。在图 3-5 的配置中，Vail 和 Telluride 都有用于到达 EBGP 对端的链路的子网地址的 **network** 命令。这些命令单独存在，因此 IBGP 对端知道如何到达这些链路的下一跳地址。

Aspen 也有用于它的两条数据链路的 **network** 命令。有这些命令，Telluride 就会知道如何到达 Vail 处的下一跳地址 192.168.1.221，同时 Vail 也知道如何到达 Telluride 处的下一跳地址 192.168.1.197。这些地址对于在 Vail 和 Telluride 之间建立 IBGP 对等会话也是非常重要的。虽然在这两个路由器之间存在着逻辑连接，如图 3-5 所示，但是，它们之间的 IBGP 会话使用的 TCP 连接要通过 Aspen。如果 Vail 和 Telluride 不知道如何找到对方，那么 TCP 连接就无法建立起来。

network 命令的位置也十分重要。例如，如果 **network 192.168.1.220 mask 255.255.255.252** 在 Vail 上，而不再 Aspen 上，那么子网就不会通过 Aspen 进行公布，因此 Telluride 会不知道如何到达下一跳地址 192.168.1.221。

例 3-31　关闭了同步功能以后，Aspen 的 IBGP 路由被加入到路由表中

```
Aspen#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
       T - traffic engineered route

Gateway of last resort is not set

B    192.168.75.0/24 [200/0] via 192.168.1.205, 00:01:00
B    192.168.200.0/24 [200/409600] via 192.168.1.225, 00:01:00
B    192.168.250.0/24 [200/0] via 192.168.1.210, 00:01:00
B    192.168.50.0/24 [200/0] via 192.168.1.205, 00:01:00
     192.168.1.0/30 is subnetted, 8 subnets
B       192.168.1.224 [200/0] via 192.168.1.221, 00:01:50
B       192.168.1.200 [200/0] via 192.168.1.205, 00:01:00
B       192.168.1.204 [200/0] via 192.168.1.197, 00:01:52
C       192.168.1.196 is directly connected, Ethernet1
B       192.168.1.216 [200/0] via 192.168.1.225, 00:01:01
C       192.168.1.220 is directly connected, Ethernet0
B       192.168.1.208 [200/0] via 192.168.1.221, 00:01:50
B       192.168.1.212 [200/0] via 192.168.1.210, 00:01:01
B    192.168.100.0/24 [200/0] via 192.168.1.225, 00:01:02
Aspen#
```

当将 EBGP 路由公布给 IBGP 对端时，EBGP 路由的下一跳地址不会改变，但是在相反的方向上，该原则不适用。如果路由器公布一条学习到的 IBGP 路由给 EBGP 对等，下一跳地址是公布该路由的路由器的接口地址，即使该路由最初是一条学习到的 EBGP 路由，这条原则也是成立的。比较一下例 3-31 中 Aspen 的 BGP 表中路由的下一跳地址和例 3-32 中 Alta 的 BGP 表中路由的下一跳地址，可以看出 Aspen 显示的到 192.168.250.0/24 的下一跳地址是 Tahoe 处的 192.168.1.210，但是 Alta 显示的到同一个目的地的下一跳地址是 Telluride 处的 192.168.1.206。实际上，Alta 上每一条学习到的 EBGP 路由都有相同的下一跳地址。

例 3-32　一条学习到的 EBGP 路由的下一跳地址通常是公布该路由的 EBGP 对等的地址

```
Alta#show ip bgp
BGP table version is 102, local router ID is 192.168.75.1
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.196/30 192.168.1.206                        0 100 i
*> 192.168.1.200/30 0.0.0.0                  0         32768 i
*> 192.168.1.204/30 192.168.1.206            0             0 100 i
*> 192.168.1.208/30 192.168.1.206                          0 100 i
*> 192.168.1.212/30 192.168.1.206                          0 100 300 i
*> 192.168.1.216/30 192.168.1.206                          0 100 200 i
*> 192.168.1.220/30 192.168.1.206                          0 100 i
*> 192.168.1.224/30 192.168.1.206                          0 100 i
*> 192.168.50.0     0.0.0.0                  0         32768 i
*> 192.168.75.0     0.0.0.0                  0         32768 i
*> 192.168.100.0    192.168.1.206                         0 100 200 i
*> 192.168.200.0    192.168.1.206                         0 100 200 i
*> 192.168.250.0    192.168.1.206                         0 100 300 i
Alta#
```

你可以使用 **neighbor next-hop-self** 命令而不用考虑当向一个 IBGP 对端公布一条 EBGP 路由时，该路由的下一跳地址不能改变的原则。例 3-33 在 AS 100 中 Vail 和 Telluride 的配置中说明了 **neighbor next-hop-self** 的使用情况。

例 3-33　当向一个 IBGP 对等公布 EBGP 路由时，强制改变该路由的下一跳地址

Vail

```
router bgp 100
 no synchronization
 neighbor 192.168.1.197 remote-as 100
 neighbor 192.168.1.197 next-hop-self
 neighbor 192.168.1.210 remote-as 300
 neighbor 192.168.1.222 remote-as 100
 neighbor 192.168.1.222 next-hop-self
 neighbor 192.168.1.225 remote-as 200
```

Telluride

```
router bgp 100
 no synchronization
 neighbor 192.168.1.198 remote-as 100
 neighbor 192.168.1.198 next-hop-self
 neighbor 192.168.1.205 remote-as 400
 neighbor 192.168.1.221 remote-as 100
 neighbor 192.168.1.221 next-hop-self
```

注意在例 3-33 中，在两个路由器里，去掉了前面配置中存在的 **network** 命令，因为现在这两个路由器在公布学习到的 EBGP 路由的时候，都把它们自己的地址作为下一跳地址，因此不再需要 **network** 命令了。例 3-34 给出了重新配置后 Aspen 的 BGP 表，如例中所显示现在 Vail 和 Telluride 把它们自己作为学习到的 EBGP 路由的下一跳，并将该路由公布给 Aspen。

例 3-34　重新配置后 Aspen 的 BGP 表

```
Aspen#show ip bgp
BGP table version is 35, local router ID is 192.168.1.222
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.196/30 0.0.0.0                  0         32768 i
*>i192.168.1.200/30 192.168.1.197            0    100      0 400 i
*>i192.168.1.212/30 192.168.1.221            0    100      0 300 i
*>i192.168.1.216/30 192.168.1.221            0    100      0 200 i
*> 192.168.1.220/30 0.0.0.0                  0         32768 i
*>i192.168.50.0     192.168.1.197            0    100      0 400 i
*>i192.168.75.0     192.168.1.197            0    100      0 400 i
*>i192.168.100.0    192.168.1.221            0    100      0 200 i
*>i192.168.200.0    192.168.1.221       409600    100      0 200 i
*>i192.168.250.0    192.168.1.221            0    100      0 300 i
Aspen#
```

本节主要目的是说明几个有关 IBGP 行为的基本概念。但是，说明这些概念的方法并不是标准的。虽然你在路由新闻组中能够找到许多有关 AS 中不带 IGP 的 IBGP 讨论，但是，在实际中，你很少能够找到这种执行情况。例如，本节给出了一种配置，在该配置中使用了 **network** 命令从而使内部路由器知道如何到达外部的下一跳地址，而在“实际生活”中，IBGP 执行在外部接口上使用 **next-hop-self** 功能或者运行被动模式的 IGP。第三种选择偶尔会遇到，就是在 AS 边界路由器上把连接的接口再分发到 IGP 中，但是这是一种比较笨的办法，通常人们不会选择该办法。

更重要的是，IGP 形成了 IBGP 所在的 TCP 会话，从而使 IBGP 本身更强壮。接下来的章节将要介绍更现实的 BGP 配置。

3.1.5　案例研究：IGP 上的 IBGP

在图 3-6 中，AS 100 内的路由器被重新配置了。在这个拓扑图中，OSPF 作为自治系统的 IGP，IBGP 只在 Vail 和 Telluride 之间运行。

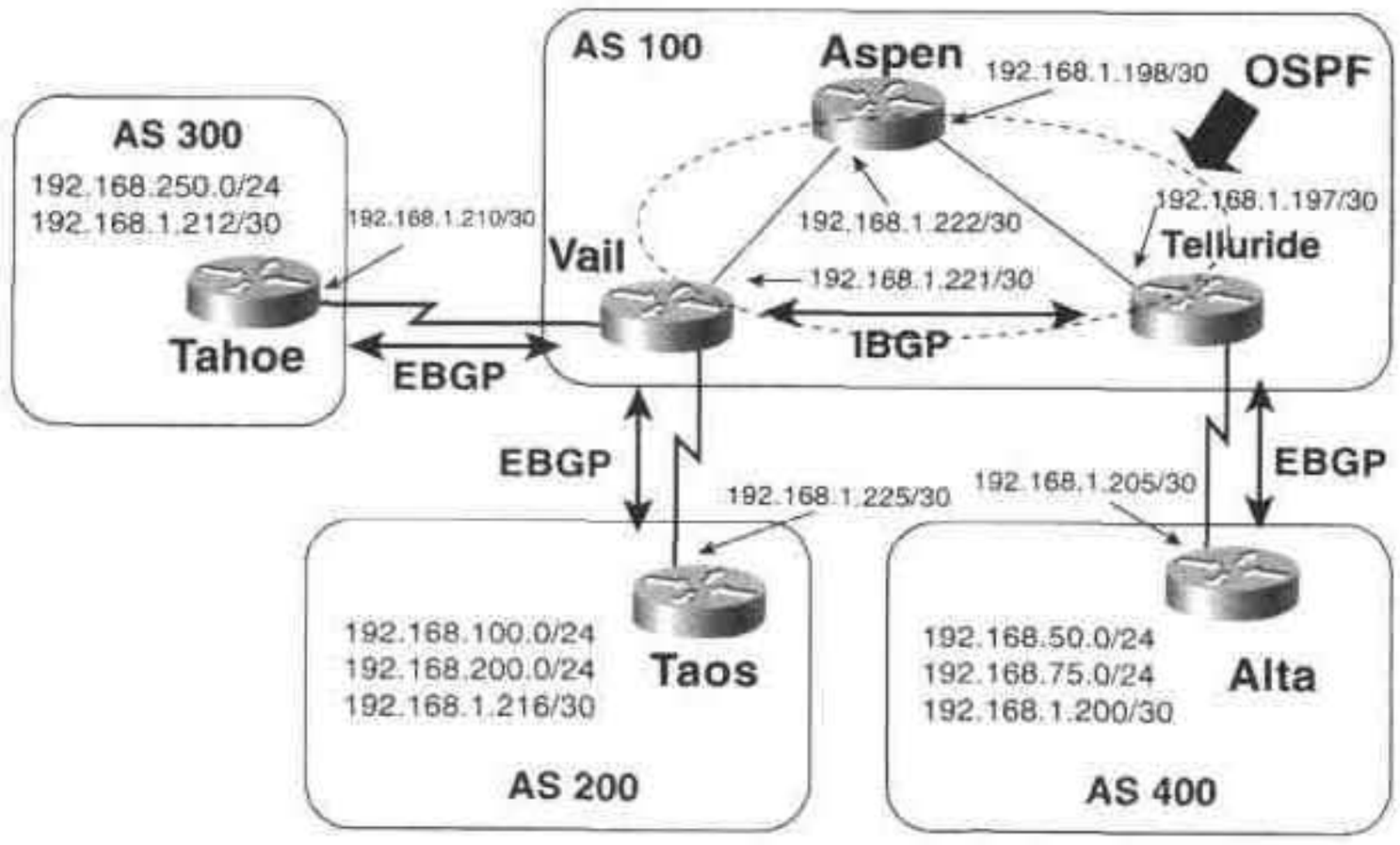


图 3-6　AS 100 中的路由器增加了 OSPF 协议

例 3-35 给出了 AS 100 中三个路由器的配置。

在 BGP 配置中打开了同步功能而且 EBGP 路由被再分发给 OSPF(打开同步功能是缺省的设置，因此在配置中没有出现任何命令)。以上两个配置步骤是 IBGP 链路正确操作所必需的。再分发的作用与前面的案例研究中 IBGP 链路对 Aspen 的作用是一样的。如果 Aspen 收到了一个起始于 AS 400 而目的地是 AS 200 的数据包，但是它不知道路由的话，它会丢弃该数据包。

同步保证再分发能够正确工作。例如，如果到 192.168.100.0/24 的路由没有从 Vail 的 OSPF 中再分发，那么在 Telluride 的路由表中不会出现该路由。Telluride 从 IBGP 连接知道该路由，但是因为它没有出现在它的路由表当中，路由器就不会向 Alta 公布该路由。到 192.168.100.0/24 的业务量不会从 AS 400 转发给 AS 100。如果存在一条可替代的路由是从 AS 400 到 AS 200(在图 3-6 中没有表示出来)，那么就可以使用这条路由。

例 3-35　AS 100 中 Vail、Aspen 和 Telluride 的配置

```
Vail
router ospf 100
 redistribute bgp 100 subnets
 network 192.168.1.221 0.0.0.0 area 0
!
router bgp 100
 neighbor 192.168.1.197 remote-as 100
 neighbor 192.168.1.197 next-hop-self
 neighbor 192.168.1.210 remote-as 300
 neighbor 192.168.1.225 remote-as 200
```

```
Aspen
router ospf 100
 network 192.168.1.0 0.0.0.255 area 0
```

```
Telluride
router ospf 100
 redistribute bgp 100 subnets
 network 192.168.1.197 0.0.0.0 area 0
!
router bgp 100
 neighbor 192.168.1.205 remote-as 400
 neighbor 192.168.1.221 remote-as 100
 neighbor 192.168.1.221 next-hop-self
```

例3-36给出了Telluride的BGP表和路由表，例3-37给出了Alta的路由表。注意在Telluride的配置中没有路由从 OSPF 再分发到 BGP，而且没有使用 **network** 命令。在 Telluride 的路由表中已经具有了所有必需的路由，而且这些路由也已经公布给了 Alta。Telluride 路由表中的路由只是为了满足同步的要求。

例 3-36　图 3-6 中 Telluride 的 BGP 表和路由表

```
Telluride#show ip bgp
BGP table version is 9, local router ID is 192.168.1.206
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.200/30 192.168.1.205            0             0 400 i
*>i192.168.1.212/30 192.168.1.221            0    100      0 300 i
*>i192.168.1.216/30 192.168.1.221            0    100      0 200 i
*> 192.168.50.0     192.168.1.205            0             0 400 i
*> 192.168.75.0     192.168.1.205            0             0 400 i
*>i192.168.100.0    192.168.1.221            0    100      0 200 i
*>i192.168.200.0    192.168.1.221       409600    100      0 200 i
*>i192.168.250.0    192.168.1.221            0    100      0 300 i

Telluride#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
       T - traffic engineered route

Gateway of last resort is not set

B    192.168.75.0/24 [20/0] via 192.168.1.205, 15:16:37
O E2 192.168.200.0/24 [110/1] via 192.168.1.198, 15:15:38, Ethernet0
O E2 192.168.250.0/24 [110/1] via 192.168.1.198, 15:15:38, Ethernet0
B    192.168.50.0/24 [20/0] via 192.168.1.205, 15:16:38
     192.168.1.0/30 is subnetted, 6 subnets
B       192.168.1.200 [20/0] via 192.168.1.205, 15:16:38
```

```
C       192.168.1.204 is directly connected, Serial0
C       192.168.1.196 is directly connected, Ethernet0
O E2    192.168.1.216 [110/1] via 192.168.1.198, 15:15:38, Ethernet0
O       192.168.1.220 [110/20] via 192.168.1.198, 15:18:22, Ethernet0
O E2    192.168.1.212 [110/1] via 192.168.1.198, 15:15:38, Ethernet0
O E2 192.168.100.0/24 [110/1] via 192.168.1.198, 15:15:39, Ethernet0
Telluride#
```

例 3-37　图 3-6 中 Alta 的路由表

```
Alta#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set
B    192.168.100.0 [20/0] via 192.168.1.206, 15:34:05
C    192.168.75.0 is directly connected, Ethernet1
C    192.168.50.0 is directly connected, Loopback0
     192.168.1.0 255.255.255.252 is subnetted, 4 subnets
C       192.168.1.200 is directly connected, Ethernet2
C       192.168.1.204 is directly connected, Serial0
B       192.168.1.216 [20/0] via 192.168.1.206, 15:33:37
B       192.168.1.212 [20/0] via 192.168.1.206, 15:33:37
B    192.168.250.0 [20/0] via 192.168.1.206, 15:34:05
B    192.168.200.0 [20/0] via 192.168.1.206, 15:34:05
```

图 3-6 中的拓扑有一个非常严重的弱点，如果 Aspen 或者它的一条链路出现故障，AS 400 就会和互联网络的其他部分隔离开来。在图 3-7 中，在 Vail 和 Telluride 之间加入了一条冗余链路，并且在该链路上建立了第 2 个 IBGP 会话。

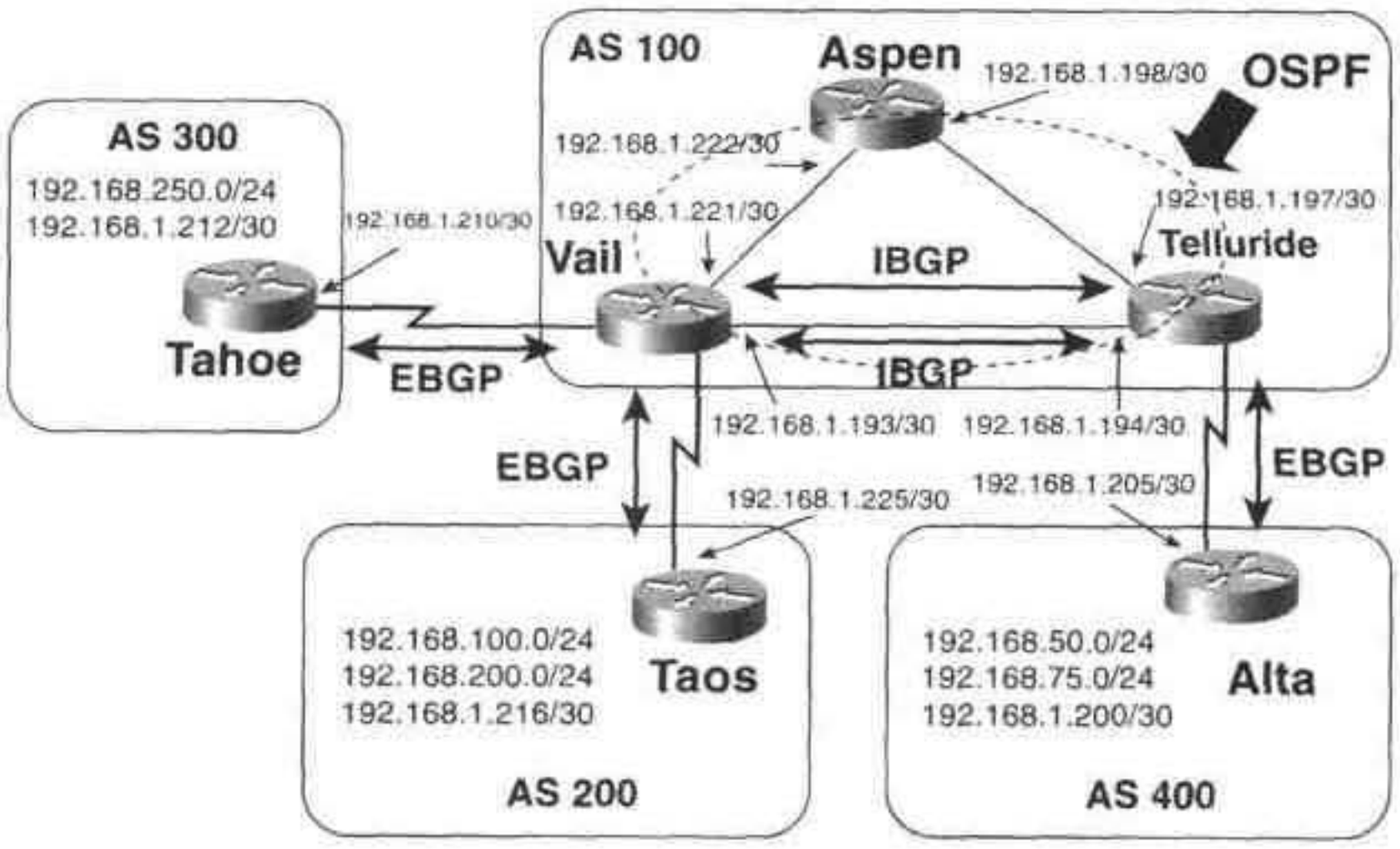


图 3-7　在 Vail 和 Telluride 之间加入了一条新的链路并建立了第二个 IBGP 会话，起备份作用

例 3-38 给出了 Vail 和 Telluride 的配置。

例 3-39 给出了 Telluride 随之产生的 BGP 表。所有从 Vail 学习到的路由都给出了两个下

一跳地址，分别代表两个 IBGP 连接。>表示正在使用的路径。如果该链路出现故障，则使用另外一条。

例 3-38　AS 100 中 Vail 和 Telluride 的配置

Vail

```
router ospf 100
 redistribute bgp 100 subnets
 network 192.168.1.193 0.0.0.0 area 0
 network 192.168.1.221 0.0.0.0 area 0
!
router bgp 100
 neighbor 192.168.1.194 remote-as 100
 neighbor 192.168.1.194 next-hop-self
 neighbor 192.168.1.197 remote-as 100
 neighbor 192.168.1.197 next-hop-self
 neighbor 192.168.1.210 remote-as 300
 neighbor 192.168.1.225 remote-as 200
```

Telluride

```
router ospf 100
 redistribute bgp 100 subnets
 network 192.168.1.194 0.0.0.0 area 0
 network 192.168.1.197 0.0.0.0 area 0
!
router bgp 100
 neighbor 192.168.1.193 remote-as 100
 neighbor 192.168.1.193 next-hop-self
 neighbor 192.168.1.205 remote-as 400
 neighbor 192.168.1.221 remote-as 100
 neighbor 192.168.1.221 next-hop-self
```

例 3-39　Telluride 的路由表给出了可替代来自 Vail 路由的路径

```
Telluride#show ip bgp
BGP table version is 17, local router ID is 192.168.255.253
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.200/30 192.168.1.205            0             0 400 i
*>i192.168.1.212/30 192.168.1.193            0    100      0 300 i
* i                  192.168.1.221            0    100      0 300 i
*>i192.168.1.216/30 192.168.1.193            0    100      0 200 i
* i                  192.168.1.221            0    100      0 200 i
*> 192.168.50.0     192.168.1.205            0             0 400 i
*> 192.168.75.0     192.168.1.205            0             0 400 i
*>i192.168.100.0    192.168.1.193            0    100      0 200 i
* i                  192.168.1.221            0    100      0 200 i
*>i192.168.200.0    192.168.1.193       409600    100      0 200 i
* i                  192.168.1.221       409600    100      0 200 i
*>i192.168.250.0    192.168.1.193            0    100      0 300 i
* i                  192.168.1.221            0    100      0 300 i
Telluride#
```

虽然图 3-7 提供了备份链路，但是故障倒换(FAILOVER)可能是一个较慢的过程。缺省的情况下，如例 3-40 所示，BGP Keepalive 间隔是 60 秒而保持时间是 180 秒。可能，BGP 检

测到一个出现故障的 IBGP 连接并且将它倒换到另外一条链路上时间很可能超过 180 秒。你可以用 **timer bgp** 命令来重新设置 BGP 的 Keepalive 和保持时间从而改进倒换时间。例如，**timer bgp 3 9** 将 Keepalive 间隔设为 3 秒而保持时间为 9 秒。

例 3-40　BGP 缺省的 Keepalive 时间是 60 秒，保持时间是 180 秒

```
Telluride#show ip bgp neighbor 192.168.1.193
BGP neighbor is 192.168.1.193,  remote AS 100, internal link
 Index 2, Offset 0, Mask 0x4
  NEXT_HOP is always this router
  BGP version 4, remote router ID 192.168.255.254
  BGP state = Established, table version = 14, up for 00:01:30
  Last read 00:00:31, hold time is 180, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 5 seconds
  Received 6 messages, 0 notifications, 0 in queue
  Sent 5 messages, 0 notifications, 0 in queue
  Last reset 00:02:51, due to User reset
  3 accepted prefixes consume 96 bytes
  0 history paths consume 0 bytes
--More--
```

图 3-8 给出了一个增加冗余链路的较好的办法。不是在可互相替换的路径上生成两个 IBGP 会话，而是在路由器的 Loopback 接口上生成单一的 IBGP 会话。OSPF 负责为 IBGP 会话找到最佳路径并且如果一条链路出现故障，它会以较快的速度为该会话重新路由。

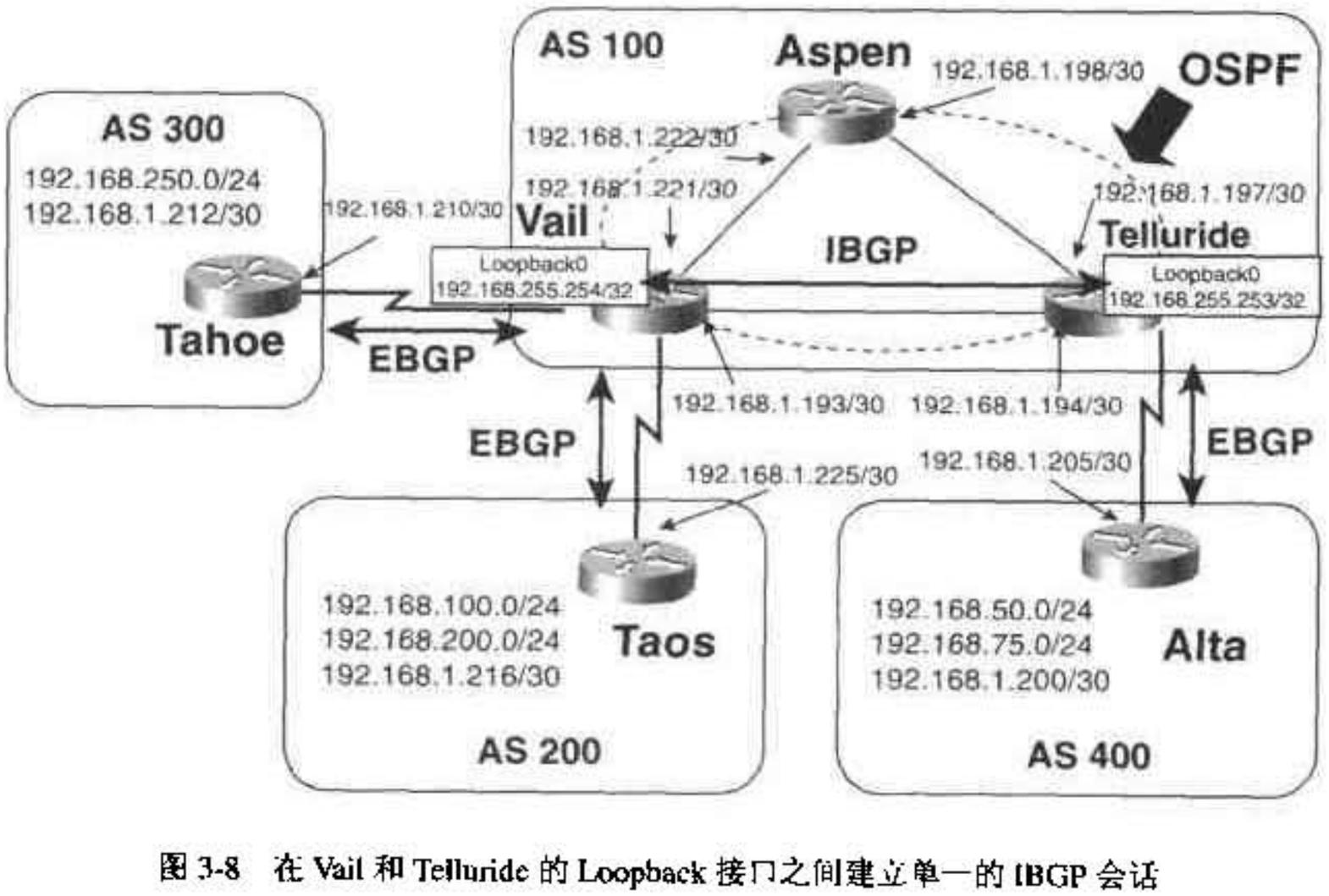


图 3-8　在 Vail 和 Telluride 的 Loopback 接口之间建立单一的 IBGP 会话

例 3-41 给出了 Vail 和 Telluride 为了图 3-8 中的设置所需的配置。

这些配置中最大的区别，除了显而易见的 Loopback 接口以外，就是 neighbor update-source 命令。这个命令使得 BGP 消息起源于 Loopback 接口的 IP 地址而不是起源于消息正在发送的物理接口。没有它，TCP 会话的 TCP 源将会是出站点接口的地址，而 TCP 会话的终点可能会不匹配而且还可能因此而无法找到。另一个重要的方面在是 OSPF 下附加的 network 命令，

是它公布了 Loopback 地址。没有它，该地址是不可到达的，因此 IBGP 会话就无法建立。例 3-42 给出重新配置以后,Telluride 的 BGP 表。

例 3-41　在 Vail 和 Telluride 的 Loopback 接口之间配置单一的 IBGP 会话

Vail

```
interface Loopback0
 ip address 192.168.255.254 255.255.255.255
!
router ospf 100
 redistribute bgp 100 subnets
 network 192.168.1.193 0.0.0.0 area 0
 network 192.168.1.221 0.0.0.0 area 0
 network 192.168.255.254 0.0.0.0 area 0
!
router bgp 100
 neighbor 192.168.1.210 remote-as 300
 neighbor 192.168.1.225 remote-as 200
 neighbor 192.168.255.253 remote-as 100
 neighbor 192.168.255.253 update-source Loopback0
 neighbor 192.168.255.253 next-hop-self
```

Telluride

```
interface Loopback0
 ip address 192.168.255.253 255.255.255.255
!
router ospf 100
 redistribute bgp 100 subnets
 network 192.168.1.194 0.0.0.0 area 0
 network 192.168.1.197 0.0.0.0 area 0
 network 192.168.255.253 0.0.0.0 area 0
!
router bgp 100
 neighbor 192.168.1.205 remote-as 400
 neighbor 192.168.255.254 remote-as 100
 neighbor 192.168.255.254 update-source Loopback0
 neighbor 192.168.255.254 next-hop-self
```

例 3-42　来自 Vail 路由的下一跳地址是 Vail 的 Loopback 地址

```
Telluride#show ip bgp
BGP table version is 7, local router ID is 192.168.255.253
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.200/30 192.168.1.205            0             0 400 i
*>i192.168.1.212/30 192.168.255.254          0    100      0 300 i
*>i192.168.1.216/30 192.168.255.254          0    100      0 200 i
*> 192.168.50.0     192.168.1.205            0             0 400 i
*> 192.168.75.0     192.168.1.205            0             0 400 i
*>i192.168.100.0    192.168.255.254          0    100      0 200 i
*>i192.168.200.0    192.168.255.254     409600    100      0 200 i
*>i192.168.250.0    192.168.255.254          0    100      0 300 i
Telluride#
```

注意：本节中的例子使用 BGP 到 IGP 的再分发从而更好地说明了基本的 IBGP 行为。但是，如果你从一个外部 BGP 对端接收到了大量的路由，再分发这些路由到 IGP 是十分危险的，因此这个时候该行为就没有用了。像例 3-8 中的拓扑，最安全的办法就是配置全网状的 BGP 连接——在 AS 100 中所有三个路由器的 Loopback 接口上都建立 IBGP 会话。于是 Aspen 就会学习到来自 BGP 直接转发数据包的必需的信息，此时不再需要再分发。

3.1.6　案例研究：EBGP 多跳

在前面的案例研究中曾经讲过，可以在 Loopback 接口之间建立 IBGP 会话，同样在 Loopback 接口之间也可以建立 EBGP 会话。图 3-9 就给出这样一个会话的例子。此时，在 Telluride 和 Alta 之间的 EBGP 会话的端点就是这两个路由器上的 Loopback 接口。

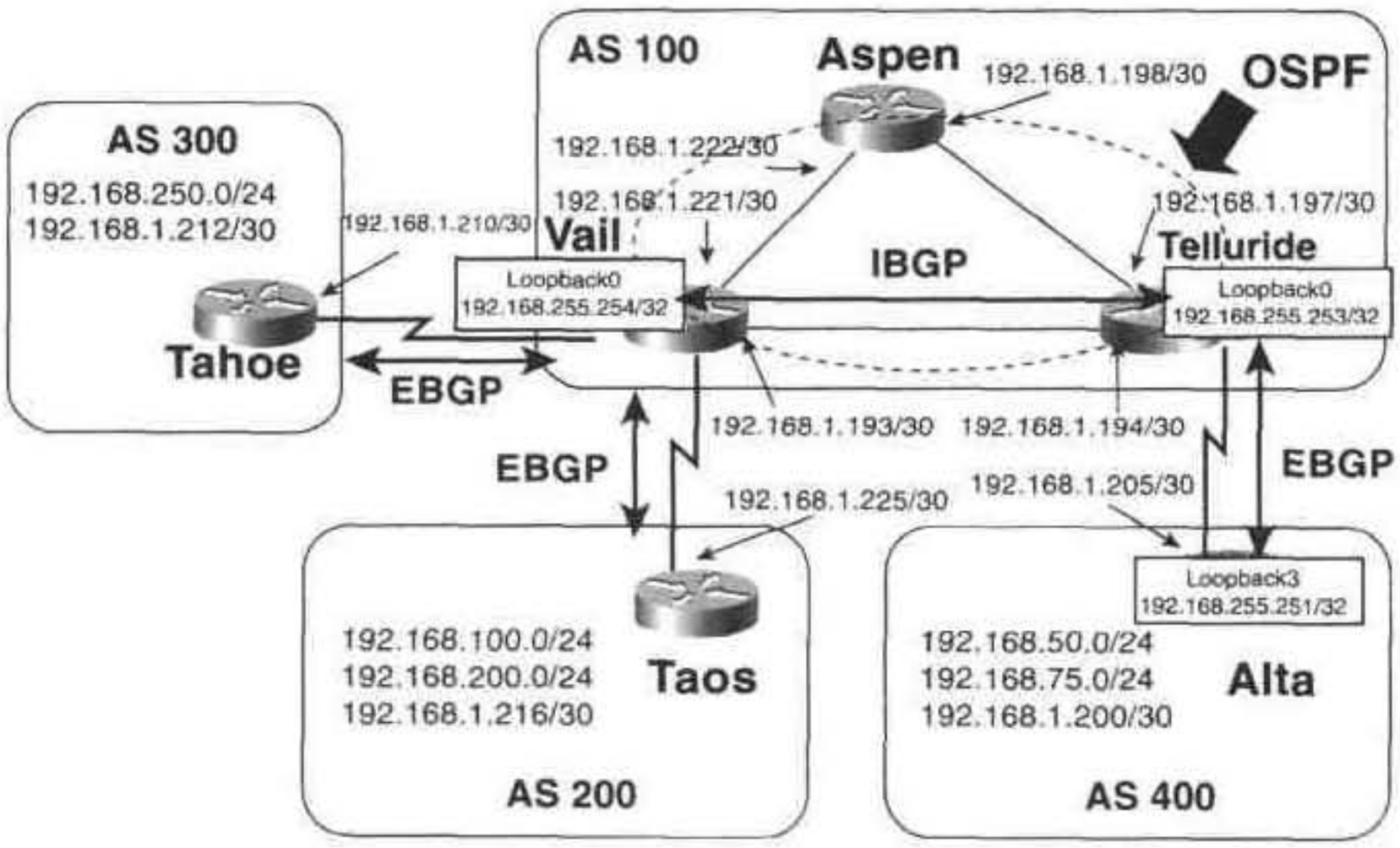


图 3-9　在 Telluride 和 Alta 的 Loopback 接口之间建立一个 EBGP 会话

例 3-43 给出了图 3-9 中两个路由器的初步配置。

注意在每个路由器上加入了静态路由。这些路由是必需的，通过它们，每个路由器都知道如何找到它邻居的 Loopback 接口的地址从而开始 TCP 会话。在前面的案例研究中，在 OSPF 下加入 **network** 命令执行的是同样的功能。在这种情况下，路由器之间没有运行 IGP，因此必须使用静态路由。排除 IBGP 故障时，记住在建立一个 IBGP 会话并交换 BGP 路由信息时，IBGP 必须知道如何找到它的对等。如果两个 IBGP 邻居不能对等，你首先应该检查的就是路由器是否知道如何到达对方。

不幸的是，用例 3-43 给出的配置并不能使邻居对等。例 3-44 给出了出现这个问题的线索。加亮线标出的那一行显示出邻居并不是直接相连的。你已经了解了这一点，也就是说，实际上，环路接口的地址并没有直接相连，这就是要求使用静态路由的原因。但是 BGP 指出的这个事实非常重要。

例 3-43　在 Telluride 与 Alta 的 Loopback 接口间配置 EBGP 会话

Telluride

```
router bgp 100
 network 192.168.1.204 mask 255.255.255.252
 neighbor 192.168.255.251 remote-as 400
 neighbor 192.168.255.251 update-source Loopback0
 neighbor 192.168.255.254 remote-as 100
 neighbor 192.168.255.254 update-source Loopback0
 neighbor 192.168.255.254 next-hop-self
!
ip route 192.168.255.251 255.255.255.255 192.168.1.205
```

Alta

```
router bgp 400
 network 192.168.50.0
 network 192.168.75.0
 network 192.168.1.200 mask 255.255.255.252
 neighbor 192.168.255.253 remote-as 100
 neighbor 192.168.255.253 update-source Loopback3
 no auto-summary
!
ip route 192.168.255.253 255.255.255.255 192.168.1.206
```

例 3-44　**show ip bgp neighbors** 命令的输出显示到 Alta 的 EBGP 连接并没有建立起来

```
Telluride#show ip bgp neighbor 192.168.255.251
BGP neighbor is 192.168.255.251,  remote AS 400, external link
 Index 1, Offset 0, Mask 0x2
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Idle, table version = 0
  Last read 00:00:11, hold time is 180, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 30 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Prefix advertised 0, suppressed 0, withdrawn 0
  Connections established 0; dropped 0
  Last reset never
  0 accepted prefixes consume 0 bytes
  0 history paths consume 0 bytes
  External BGP neighbor not directly connected.
  No active TCP connection
Telluride
```

该重要性在于虽然你可以通过多跳路由器来建立一个 IBGP，但是在缺省的情况下，EBGP 邻居必须直接相连。在图 3-9 中，源于 Alta Loopback 接口的数据包必须选路到 Alta 的串行端口。在 Telluride，必须将数据包从 Telluride 的串行端口路由到它的 Loopback 接口。换句话说，TCP 数据包在 loopback 接口之间要经过两个路由器跳。

neighbor ebgp-multihop 命令允许改变数据包的 TTL 缺省值 1，通过这种改变可以使你越过缺省一跳的 EBGP 限制。例 3-45 给出了邻居的配置，该配置通过 **neighbor ebgp-multihop**

命令将 EBGP 数据包的 TTL 缺省值改成了 2。

例 3-45　使用 **neighbor ebgp-multihop** 命令使得 Alta 和 Telluride 越过了缺省一跳的 EBGP 限制

Telluride

```
router bgp 100
 network 192.168.1.204 mask 255.255.255.252
 neighbor 192.168.255.251 remote-as 400
 neighbor 192.168.255.251 ebgp-multihop 2
 neighbor 192.168.255.251 update-source Loopback0
 neighbor 192.168.255.254 remote-as 100
 neighbor 192.168.255.254 update-source Loopback0
 neighbor 192.168.255.254 next-hop-self
!
ip route 192.168.255.251 255.255.255.255 192.168.1.205
```

Alta

```
router bgp 400
 network 192.168.50.0
 network 192.168.75.0
 network 192.168.1.200 mask 255.255.255.252
 neighbor 192.168.255.253 remote-as 100
 neighbor 192.168.255.253 ebgp-multihop 2
 neighbor 192.168.255.253 update-source Loopback3
 no auto-summary
!
ip route 192.168.255.253 255.255.255.255 192.168.1.206
```

例 3-46 给出了配置改变后的结果。EBGP 会话建立起来了，而且输出结果显示了新的跳限制。

例 3-46　show ip bgp neighbors 的输出显示出到 Alta 的 EBGP 连接建立起来了

```
Telluride#show ip bgp neighbor 192.168.255.251
BGP neighbor is 192.168.255.251,  remote AS 400, external link
 Index 1, Offset 0, Mask 0x2
  BGP version 4, remote router ID 192.168.255.251
  BGP state = Established, table version = 9, up for 00:04:44
  Last read 00:00:14, hold time is 180, keepalive interval is 60 seconds
  Minimum time between advertisement runs is 30 seconds
  Received 9 messages, 0 notifications, 0 in queue
  Sent 11 messages, 0 notifications, 0 in queue
  Prefix advertised 4, suppressed 0, withdrawn 0
  Connections established 1; dropped 0
  Last reset 00:25:59, due to User reset
  3 accepted prefixes consume 96 bytes
  0 history paths consume 0 bytes
  External BGP neighbor may be up to 2 hops away
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 192.168.255.253, Local port: 11001
Foreign host: 192.168.255.251, Foreign port: 179
```

通常是在环回端口之间配置 IBGP，与之不同的是，大部分的 EBGP 是在直接相连的端

口之间配置，因此，并不是经常需要 **ebgp-multihop** 命令。在 Loopback 接口之间建立 EBGP 的一个例子就是两个外部邻居通过多条用于备份的链路直接相连(例如，多条 ATM 或者 FR 的虚电路)，但是只需要一个 EBGP 会话。当 EBGP 会话所在的链路出现故障，该会话就会被选路到另外一条备份链路上去。

3.1.7　案例研究：聚合路由

图3-10中AS 100里包含了8个C类网络地址，可以将它们归纳为一个聚合地址192.168.192.0/21。Stowe 通过 EIGRP 学习内部网络，通过 EBGP 公布聚合地址给 Sugarbush。

在 BGP 下有两种生成聚合地址的方法。第一种方法是在路由表中为聚合路由建立一条静态路由条目然后用 **network** 命令将它公布出去。第二种方法是使用 **aggregate-address** 命令生成聚合地址。

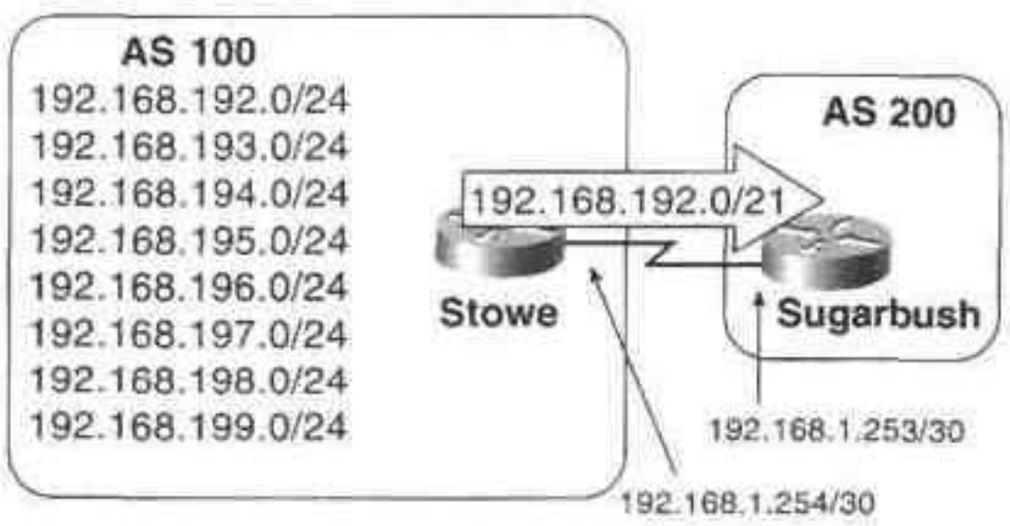


图 3-10　AS 100 中所有的内部网络都可以聚合成一个单一的地址 192.168.192.0/21

1．使用静态路由的聚合

例 3-47 说明了 Stowe 的一个配置，它使用静态路由条目聚合地址，该地址通过 **network** 命令进行公布。

例 3-47　通过由 **network** 命令公布的静态条目在 BGP 下生成一个聚合地址

```
router eigrp 100
 network 192.168.199.0
!
router bgp 100
 network 192.168.192.0 mask 255.255.248.0
 neighbor 192.168.1.253 remote-as 200
!
ip classless
ip route 192.168.192.0 255.255.248.0 Null0
```

静态路由指向 Null 接口，因为聚合地址本身不是一个合法的目的地。在 Stowe 的路由表中它只代表更具体的地址。目的地址属于 AS 100 中一个 C 类地址的数据包与 AS 100 外部一个路由器上的聚合地址相匹配，并且被转发到 Stowe。在 Stowe 处，数据包与更具体的地址相匹配并被转发到正确的下一跳路由器。如果由于某种原因，更具体的 C 类地址不再 Stowe 的路由表中，将这些数据包转发到 Null 接口并丢弃。

例 3-48 给出了 Stowe 和 Sugarbush 的 BGP 表。在 Stowe 的 BGP 表中只有聚合地址；路由器的 BGP 配置没有加入任何其他的地址。

例 3-48　Stowe 和 Sugarbush 的 BGP 表都只包含聚合路由

```
Stowe#show ip bgp
BGP table version is 2, local router ID is 192.168.199.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0/21 0.0.0.0                  0         32768 i
Stowe#

Sugarbush#show ip bgp
BGP table version is 18, local router ID is 172.17.3.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0/21 192.168.1.254            0             0 100 i
Sugarbush#
```

2．抑制更具体的路由

对于图 3-10 的简单拓扑来讲，通常使用第一种生成聚合地址的方法已经足够了。但是，随着拓扑以及路由策略的日益复杂，**aggregate-address** 命令中的选项使得该方法会更有用。该案例研究的剩余部分会讨论 **aggregate-address** 命令以及它的选项。

要宣告由 **aggregate-address** 命令确定的聚合地址，至少要将属于聚合的更具体地址中的一个地址加入到 BGP 表中，可以通过再分发也可以通过 **network** 命令来完成该工作。例 3-49 说明了 Stowe 使用 **aggregate-address** 命令以及再分发的一个配置。

例 3-49　使用 **aggregate-address** 命令在 BGP 下生成一个聚合地址

```
router eigrp 100
 network 192.168.199.0
!
router bgp 100
 aggregate-address 192.168.192.0 255.255.248.0 summary-only
 redistribute eigrp 100
 neighbor 192.168.1.253 remote-as 200
```

例 3-50 给出了 Stowe 和 Sugarbush 相应的 BGP 表。Stowe 的 BGP 表看起来与例 3-48 中它的 BGP 表完全不同——在例 3-50 中囊括了所有的更具体的地址。但是，Sugarbush 的 BGP 表看起来完全相同，同样只是公布了聚合地址。

例 3-50　Stowe 的 BGP 表囊括了所有的更具体地址；而 Sugarbush 只公布了聚合地址。

```
Stowe#show ip bgp
BGP table version is 23, local router ID is 192.168.199.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
s> 192.168.192.0    192.168.199.1      2297856         32768 ?
*> 192.168.192.0/21 0.0.0.0                            32768 i
```

```
s> 192.168.193.0    192.168.199.1          2297856         32768 ?
s> 192.168.194.0    192.168.199.1          2297856         32768 ?
s> 192.168.195.0    192.168.199.1          2297856         32768 ?
s> 192.168.196.0    192.168.199.1          2297856         32768 ?
s> 192.168.197.0    192.168.199.1          2297856         32768 ?
s> 192.168.198.0    192.168.199.1          2297856         32768 ?
s> 192.168.199.0    0.0.0.0                      0         32768 ?
Stowe#
```

```
Sugarbush#show ip bgp
BGP table version is 2, local router ID is 172.17.3.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0/21 192.168.1.254                            0 100 i
Sugarbush#
```

Stowe 的 BGP 表中更具体的地址左边的关键字指示这些路由已经被抑制了。这些抑制来自 **aggregate-address** 命令中的 **summary-only** 选项。如果没有该选项，聚合地址以及更具体地址都将被公布。

3．公布聚合地址以及更具体地址

对于图 3-10 中的简单拓扑，同时公布聚合地址以及更具体地址是没有意义的。但是图 3-11 给出了一个环境，在该种环境下，两种地址同时公布的方案是理想的。在这里，AS 100 多宿主到 AS 200，AS 200 需要来自 AS 100 的所有路由来设置路由策略，但是它必须只将聚合地址公布给 AS 300。

虽然 AS 100 的更具体路由都向 AS 200 公布，但是它们都携带了一个 NO_EXPORT 的团体属性。正如第 2 章所讨论的，携带该属性的路由不能公布给 EBGP 对端，结果是，AS 200 知道这些路由但是不把它们公布给 AS 300。只有没携带团体 NO_EXPORT 属性的聚合路由被公布给了 AS 300。例 3-51 给出了 Stowe 的配置，Mammoth 的配置与它相似，将在本节的后面给出。

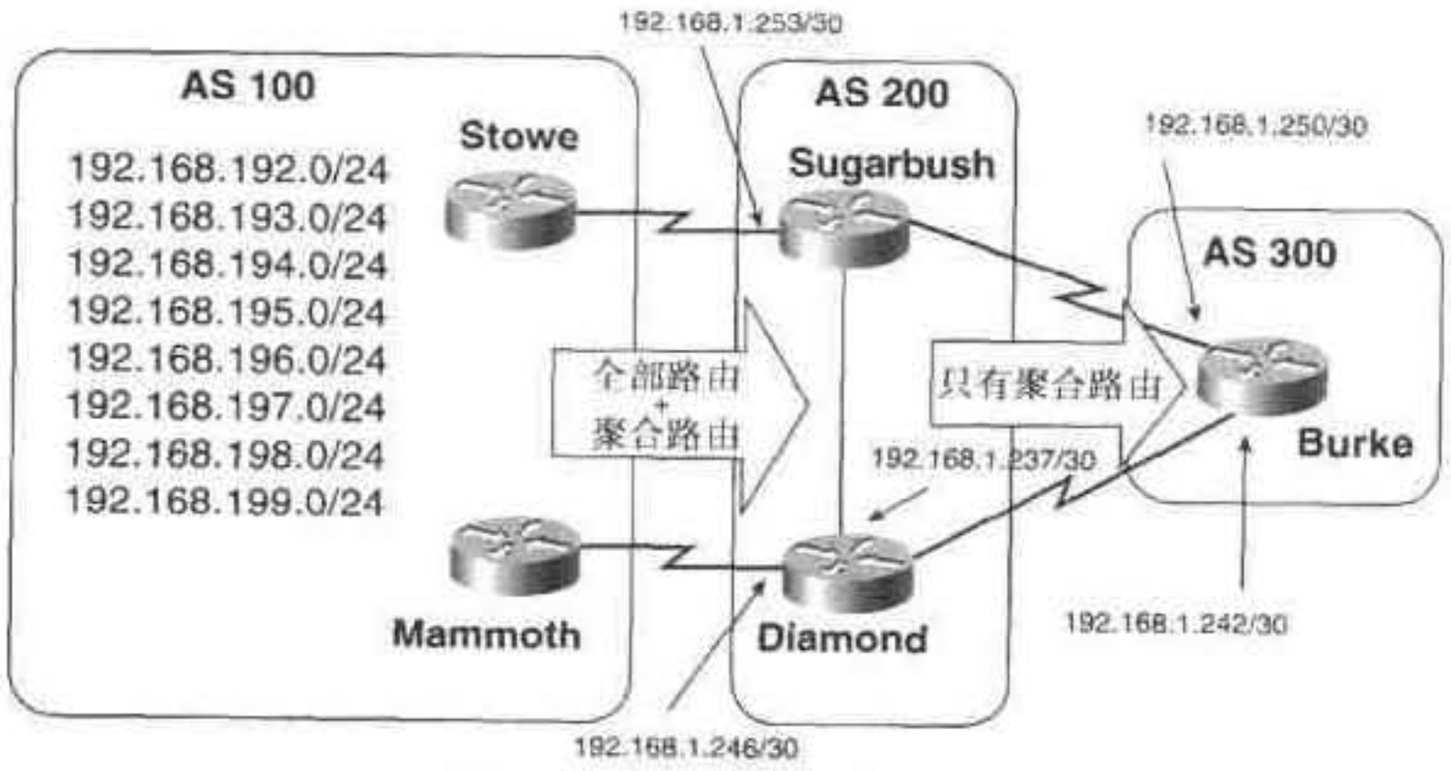


图 3-11　AS 100 多宿主到 AS 200

在 **aggregate-address** 命令中已经去掉了 **summary-only** 关键字，于是聚合地址和更具体地址都公布给了 AS 200。**neighbor 192.168.1.253 send-community** 命令明确了将

COMMUNITY 属性发送给 Sugarbush。**neighbor 192.168.1.253 route-map COMMUNITY out** 命令通过名为 COMMUNITY 的路由映射而过滤掉出去的 BGP 路由。如果路由图与访问列表 101 匹配，则不设置 COMMUNITY 属性。如果路由与访问列表 101 不匹配，那么就给该路由分配一个 NO_EXPORT 的 COMMUNITY 属性。

例 3-51　将 Stowe 配置成既公布聚合地址又公布更具体地址

```
router eigrp 100
 network 192.168.199.0
!
router bgp 100
 aggregate-address 192.168.192.0 255.255.248.0
 redistribute eigrp 100
 neighbor 192.168.1.253 remote-as 200
 neighbor 192.168.1.253 send-community
 neighbor 192.168.1.253 route-map COMMUNITY out
!
ip classless
!
access-list 101 permit ip host 192.168.192.0 host 255.255.248.0
!
route-map COMMUNITY permit 10
 match ip address 101
 set community none
!
route-map COMMUNITY permit 20
 set community no-export
```

访问列表 101 的使用对你来讲可能是新鲜的。通常，一个扩展的 IP 访问列表所明确的第一个地址是源地址，第二个地址是目的地址。但是，在这个应用中，第一个地址是路由的前缀，第二个地址是前缀的掩码。必须使用这种特殊访问列表的原因是必须确定精确的前缀。如果使用 **access-list 1 permit 192.168.192.0 0.0.7.255**，它将同时与聚合地址 192.168.192.0/21 和更具体地址 192.168.192.0/24 匹配。

例 3-52 给出了 Sugarbush 的 BGP 表，可以看出它包含了聚合地址和更具体地址。而且，还使用了 **show ip bgp community no_export** 命令来显示携带了 NO_EXPORT 团体属性的路由。所有来自 Stowe 的路由除了聚合路由都被列了出来。

例 3-52　Sugarbush 的 BGP 表

```
Sugarbush#show ip bgp
BGP table version is 30, local router ID is 172.17.3.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i192.168.192.0    192.168.1.237    2297856    100      0 100 ?
*>                  192.168.1.254    2297856             0 100 ?
* i192.168.192.0/21 192.168.1.237               100      0 100 i
*>                  192.168.1.254                         0 100 i
* i192.168.193.0    192.168.1.237    2297856    100      0 100 ?
*>                  192.168.1.254    2297856             0 100 ?
* i192.168.194.0    192.168.1.237    2297856    100      0 100 ?
*>                  192.168.1.254    2297856             0 100 ?
* i192.168.195.0    192.168.1.237    2297856    100      0 100 ?
*>                  192.168.1.254    2297856             0 100 ?
```

```
* i192.168.196.0    192.168.1.237    2297856    100      0 100 ?
*>                  192.168.1.254    2297856             0 100 ?
* i192.168.197.0    192.168.1.237    2297856    100      0 100 ?
*>                  192.168.1.254    2297856             0 100 ?
*>i192.168.198.0    192.168.1.237          0    100      0 100 ?
*                   192.168.1.254    2681856             0 100 ?
* i192.168.199.0    192.168.1.237    2681856    100      0 100 ?
*>                  192.168.1.254          0             0 100 ?

Sugarbush#show ip bgp community no-export
BGP table version is 10, local router ID is 172.17.3.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0    192.168.1.254    2297856             0 100 ?
*> 192.168.193.0    192.168.1.254    2297856             0 100 ?
*> 192.168.194.0    192.168.1.254    2297856             0 100 ?
*> 192.168.195.0    192.168.1.254    2297856             0 100 ?
*> 192.168.196.0    192.168.1.254    2297856             0 100 ?
*> 192.168.197.0    192.168.1.254    2297856             0 100 ?
*  192.168.198.0    192.168.1.254    2681856             0 100 ?
*> 192.168.199.0    192.168.1.254          0             0 100 ?
Sugarbush#
```

例 3-53 给出了 Burke 的 BGP 表。除了聚合路由以外没有公布任何其他的路由。

例 3-53　Burke 的 BGP 表中只包含聚合路由

```
Burke#show ip bgp
BGP table version is 15, local router ID is 172.21.1.1
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0/21 192.168.1.249                         0 200 100 i
*                   192.168.1.241                         0 200 100 i
Burke#
```

Mammoth 的公布情况和 Stowe 的相同。为了说明的目的，Mammoth 采用了不同的配置方式(例 3-54)来到达了同样的结果。在 Mammoth 使用了 IP 前缀列表而没有采用访问列表。

例 3-54　图 3-11 中 Mammoth 的配置

```
router eigrp 100
 network 192.168.198.0
!
router bgp 100
 aggregate-address 192.168.192.0 255.255.248.0
 redistribute eigrp 100
 neighbor 192.168.1.246 remote-as 200
 neighbor 192.168.1.246 send-community
 neighbor 192.168.1.246 route-map COMMUNITY out
!
ip classless
ip route 192.168.255.251 255.255.255.255 192.168.1.205
!
```

```
!
ip prefix-list AGGREGATE seq 5 permit 192.168.192.0/21
!
route-map COMMUNITY permit 10
 match ip address prefix-list AGGREGATE
 set community none
!
route-map COMMUNITY permit 20
 set community no-export
```

与路由图相同，前缀表也是用一个名字来表示而不用数字。在例 3-54 中，前缀表的名字是 AGGREGATE。列表的行用一个顺序号(**seq**)区分开来，它表明在一个多行的列表当中每一行的位置,并且使编辑该表更加容易。如果你加入一行时没有键入顺序号，Cisco IOS 软件会按照你加入行的顺序自动为它加入一个顺序号。在 **permitldeny** 关键字后面，确定了前缀和前缀的长度。

例 3-54 中 Mammoth 配置里的前缀列表与 192.168.192.0/21 完全匹配。但是，你还可以加入一个选项从而与一定范围内的前缀匹配。例如，**ip prefix-list AGGREGATE seq 5 permit 192.168.192.0/21 ge 24** 命令与所有那些前 21 个比特与 192.168.192.0 相匹配而且长度大于或者等于 24 的前缀相匹配，这一行与 AS 100 中所有更具体的地址相匹配。另一方面，关键字 **le** 用于与那些长度短于或者等于规定比特数的前缀相匹配。

Diamond 的 BGP 表看起来和 Sugarbush 的 BGP 表很相像：它们都加入了更具体的路由和聚合路由。除了来自 Sugarbush 的聚合地址，例 3-53 中 Burke 的 BGP 表显示 Diamond 也公布了该聚合地址。例 3-55 更详细地显示了来自 DiamondBGP 表的聚合路由和一条更具体的路由。可以看出聚合路由没有任何团体属性(虽然它没有团体属性但是作为一个聚合路由，它具有 **ATOMIC_AGGREGATE** 的 **AGGREGATOR** 属性集)，而且更具体路由也没有团体属性。

例 3-55　更进一步地观察来自 Diamond BGP 表中的两条路由的各自属性

```
Diamond# show ip bgp 192.168.192.0 255.255.248.0
BGP routing table entry for 192.168.192.0/21, version 59
Paths: (2 available, best #1)
  Advertised to non peer-group peers:
    192.168.1.238 192.168.1.242
  100, (aggregated by 100 192.168.198.2)
    192.168.1.245 from 192.168.1.245 (192.168.198.2)
      Origin IGP, localpref 100, valid, external, atomic-aggregate, best, ref 2
  100, (aggregated by 100 192.168.199.2)
    192.168.1.238 from 192.168.1.238 (192.168.1.253)
      Origin IGP, localpref 100, valid, internal, not synchronized, atomic-aggregate,
ref 2
Diamond#

Diamond#show ip bgp 192.168.199.0
BGP routing table entry for 192.168.199.0/24, version 58
Paths: (2 available, best #1, not advertised to EBGP peer)
  Advertised to non peer-group peers:
    192.168.1.238
  100
```

```
    192.168.1.245 from 192.168.1.245 (192.168.198.2)
     Origin incomplete, metric 2681856, localpref 100, valid, external, best, ref 2
     Community: no-export
  100
    192.168.1.238 from 192.168.1.238 (192.168.1.253)
     Origin incomplete, metric 0, localpref 100, valid, internal, not synchronized,
ref 2
Diamond#
```

4. 宣告聚合路由及挑选出来的更具体的路由

在前一种方案中，将 AS 100 的更具体的路由发送给了 AS 200，因此 AS 200 能够执行路由策略。也就是说，AS 200 通过这些路由来设置路由选项，从而向 AS 100 发送业务量。AS 100 也可以通过操纵输出的路由宣告来影响它的入站业务量。例如，在 Stowe/Sugarbush 链路上公布 192.168.193.0/24 而不是在 Mammoth/Diamond 链路上公布会导致入业务量使用 Stowe/Sugarbush 链路，如果 AS 有地域上的不同，管理者可能会希望能够执行以上的策略。例如，Stowe 可能在 Vermont，而 Mammoth 在 California，管理者可能会希望入业务量使用与目的地接近的入站点，从而使内部路由最小化。

为了说明上述的情况，图 3-11 中的 AS 100 应该执行以下的路由策略：

在 Stowe/Sugarbush 链路上公布 192.168.192.0/24，192.168.193.0/24 和 192.168.194.0/24。

在 Mammoth/Diamond 链路上公布 192.168.196.0/24，192.168.197.0/24 和 192.168.198.0/24。

不公布 192.168.195.0/24 和 192.168.199.0/24。

在两条链路上同时公布聚合路由，用于备份。当一条链路出现故障时，所有的入业务量都会路由到另外那条链路上去。

要抑制聚合路由的一个子网，可以和 **aggregate-address** 命令一起使用 **suppress-map** 选项。Stowe 和 Mammoth 通过例 3-56 中的配置来执行列出的策略。团体路由图和 EIGRP 配置与前面章节的配置相同，因此就不在此列出了。

例 3-56　用 **aggregate-address** 命令中的 **suppress-map** 选项来抑制选中的前缀

Stowe

```
router bgp 100
 aggregate-address 192.168.192.0 255.255.248.0 suppress-map VERMONT
 redistribute eigrp 100
 neighbor 192.168.1.253 remote-as 200
 neighbor 192.168.1.253 send-community
 neighbor 192.168.1.253 route-map COMMUNITY out
!
access-list 1 permit 192.168.195.0 0.0.0.255
access-list 1 permit 192.168.196.0 0.0.3.255
!
route-map VERMONT permit 10
 match ip address 1
```

Mammoth

```
router bgp 100
 aggregate-address 192.168.192.0 255.255.248.0 suppress-map CALIFORNIA
 redistribute eigrp 100
 neighbor 192.168.1.246 remote-as 200
 neighbor 192.168.1.246 send-community
 neighbor 192.168.1.246 route-map COMMUNITY out
```

```
!
ip prefix-list SUPPRESSEDROUTES seq 5 permit 192.168.192.0/22 le 24
ip prefix-list SUPPRESSEDROUTES seq 10 permit 192.168.199.0/24
!
route-map CALIFORNIA permit 10
 match ip address prefix-list SUPPRESSEDROUTES
```

在 Stowe 的配置中，它使用了一个名为 VERMONT 的路由图，来决定需要抑制的路由，路由图使用访问列表 1 来确定合适的路由。访问列表允许抑制 192.168.195.0/24 以及所有前 22 个比特与 192.168.196.0/22 相匹配的前缀。所有其他的前缀都与列表最后隐含的“拒绝任何一个”相匹配，因此都不会被抑制。

> 注：　路由图的逻辑有时候会造成一定的误解。在这个例子中，被访问列表拒绝的路由是“拒绝被抑制”，也就是说，允许公布该路由。另一方面，被访问列表允许的路由，是“允许被抑制”，因此抑制该路由。

Mammoth 的配置是使用一个名为 CALIFORNIA 的路由图来决定将要抑制的路由。Mammoth 再一次使用了前缀列表而没有使用访问列表来决定合适的抑制路由。5 号前缀列表的说明允许所有前 22 个比特与 192.168.192.0/22 相匹配而且长度短于或者等于 24 的前缀。前缀列表的序列号 10 允许前缀 192.168.199.0/24。带有被允许前缀的路由的公布被抑制了；所有其他路由隐含地被前缀列表拒绝，因此没有抑制这些路由。

例 3-57 给出了 Sugarbush 和 Diamond 相应的 BGP 表，如例中所示 BGP 表显示 Stowe 和 Mammoth 公布了 AS 100 中更具体路由的不同子网，但它们还同时公布了聚合路由。例如，Sugarbush 将目的地是 192.168.193.0/24 的数据包转发给 Stowe(192.168.1.254)，而将目的地是 192.168.196.0/24 的数据包转发给 Diamond(192.168.1.237)。而 Diamond，将目的地是 192.168.196.0/24 的数据包转发给 Mammoth(192.168.1.245)。Stowe 和 Mammoth 都公布了聚合路由；如果到这两个路由器的链路中的一条出现了故障，通常会通过与聚合路由匹配的剩余链路来转发数据包。

例 3-57　Sugarbush 和 Diamond 相应的 BGP 表

```
Sugarbush#show ip bgp
BGP table version is 79, local router ID is 192.168.1.253
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0    192.168.1.254    2297856             0 100 ?
* i192.168.192.0/21 192.168.1.237               100      0 100 i
*>                  192.168.1.254                         0 100 i
*> 192.168.193.0    192.168.1.254    2297856             0 100 ?
*> 192.168.194.0    192.168.1.254    2297856             0 100 ?
*>i192.168.196.0    192.168.1.237    2297856    100      0 100 ?
*>i192.168.197.0    192.168.1.237    2297856    100      0 100 ?
*>i192.168.198.0    192.168.1.237          0    100      0 100 ?
Sugarbush#
```

```
Diamond#show ip bgp
BGP table version is 137, local router ID is 172.18.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
   Network          Next Hop            Metric LocPrf Weight Path
* i192.168.192.0    192.168.1.238      2297856    100      0 100 ?
*> 192.168.192.0/21 192.168.1.245                           0 100 i
* i                 192.168.1.238              100      0 100 i
* i192.168.193.0    192.168.1.238      2297856    100      0 100 ?
* i192.168.194.0    192.168.1.238      2297856    100      0 100 ?
*> 192.168.196.0    192.168.1.245      2297856             0 100 ?
*> 192.168.197.0    192.168.1.245      2297856             0 100 ?
*> 192.168.198.0    192.168.1.245            0             0 100 ?
Diamond#
```

5. 改变聚合的属性

你可以和 **aggregate-address** 命令一起使用 **attribute-map** 选项。该选项使你可以改变聚合路由的属性。例如，在例 3-57 中，因为路由是从 EIGRP 再分发给 BGP，因此所有的更具体路由都有一个 Incomplete 的 ORIGIN 属性。但是，聚合路由有一个 IGP 的 ORIGIN 属性，因为它们是在 Stowe 和 Mammoth 的 BGP 进程中发起的。假设管理者希望 AS 200 将 Mammoth/Diamond 链路上的聚合路由用于所有的业务量，而只把 Stowe/Sugarbush 链路用作备份。BGP 决定过程，正如第三章所讨论的，在考虑两条到同一目的地的路由时，它会优先选择 ORIGIN 属性为 IGP 的路由而不选择 ORIGIN 属性为 Incomplete 的路由。如果 Stowe 将它的聚合路由的 ORIGIN 属性改为 Incomplete，AS 200 中的路由器将会优先考虑 Mammoth/Diamond 链路。例 3-58 给出了 Stowe 的配置。

例 3-58　Stowe 将它的聚合路由的 ORIGIN 改为 Incomplete

```
router eigrp 100
 network 192.168.199.0
!
router bgp 100
 aggregate-address 192.168.192.0 255.255.248.0 attribute-map ORIGIN suppress-map
VERMONT
 redistribute eigrp 100
 neighbor 192.168.1.253 remote-as 200
 neighbor 192.168.1.253 send-community
 neighbor 192.168.1.253 route-map COMMUNITY out
!
access-list 1 permit 192.168.195.0 0.0.0.255
!
access-list 101 permit ip host 192.168.192.0 host 255.255.248.0
!
route-map ORIGIN permit 10
 set origin incomplete
!
route-map COMMUNITY permit 10
 match ip address 101
 set community none
!
route-map COMMUNITY permit 20
 set community no-export
!
route-map VERMONT permit 10
 match ip address 1
```

例 3-59 给出了 Sugarbush 相应的 BGP 表，如例中所示现在由 Stowe(192.168.1.254)宣告的聚合路由有一个 Incomplete 的 ORIGIN 属性；因此由 Diamond 公布的、Mammoth 发起的聚合路由是优选。在重新配置之前，Sugarbush 优选通过 EBGP 学习到的聚合路由而不选通过 IBGP

学习到的聚合路由(见例 3-57)。但是，ORIGIN 属性在决定过程中有高于 IBGP/EBGP 的优先级，因此来自 Diamond 的 IBGP 路由，因为带有一个 IGP 的 ORIGIN 属性，因此现在是优选。

例 3-59　Sugarbush 相应的 BGP 表

```
Sugarbush#show ip bgp
BGP table version is 17, local router ID is 192.168.1.253
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0    192.168.1.254    2297856             0 100 ?
*>i192.168.192.0/21 192.168.1.237            100        0 100 i
*                   192.168.1.254                       0 100 ?
*> 192.168.193.0    192.168.1.254    2297856             0 100 ?
*> 192.168.194.0    192.168.1.254    2297856             0 100 ?
*>i192.168.196.0    192.168.1.237    2297856    100      0 100 ?
*>i192.168.197.0    192.168.1.237    2297856    100      0 100 ?
*>i192.168.198.0    192.168.1.237          0    100      0 100 ?
Sugarbush#
```

有意思地是，重新配置也影响了 AS 300 中的路由策略，如例 3-60 所示。Sugarbush 的优选路由是 IBGP 路由；打开了同步功能，因此 IBGP 路由不能公布给 EBGP 对端 Burke。结果是，Burke 只能从 Diamond 学习到聚合路由。

例 3-60　只有 Diamond 向 Burke 公布聚合路由

```
Burke#show ip bgp
BGP table version is 3, local router ID is 172.21.1.1
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0/21 192.168.1.241                        0 200 100 i
Burke#
```

6. 和聚合一起使用 AS_SET

图 3-12 是图 3-11 中的互联网络修改后的版本，包括改变了聚合地址的源。现在，AS 100 和 AS 200 将 AS 100 全部的路由公布给 AS 300 和 AS 400，但是不包括聚合路由。

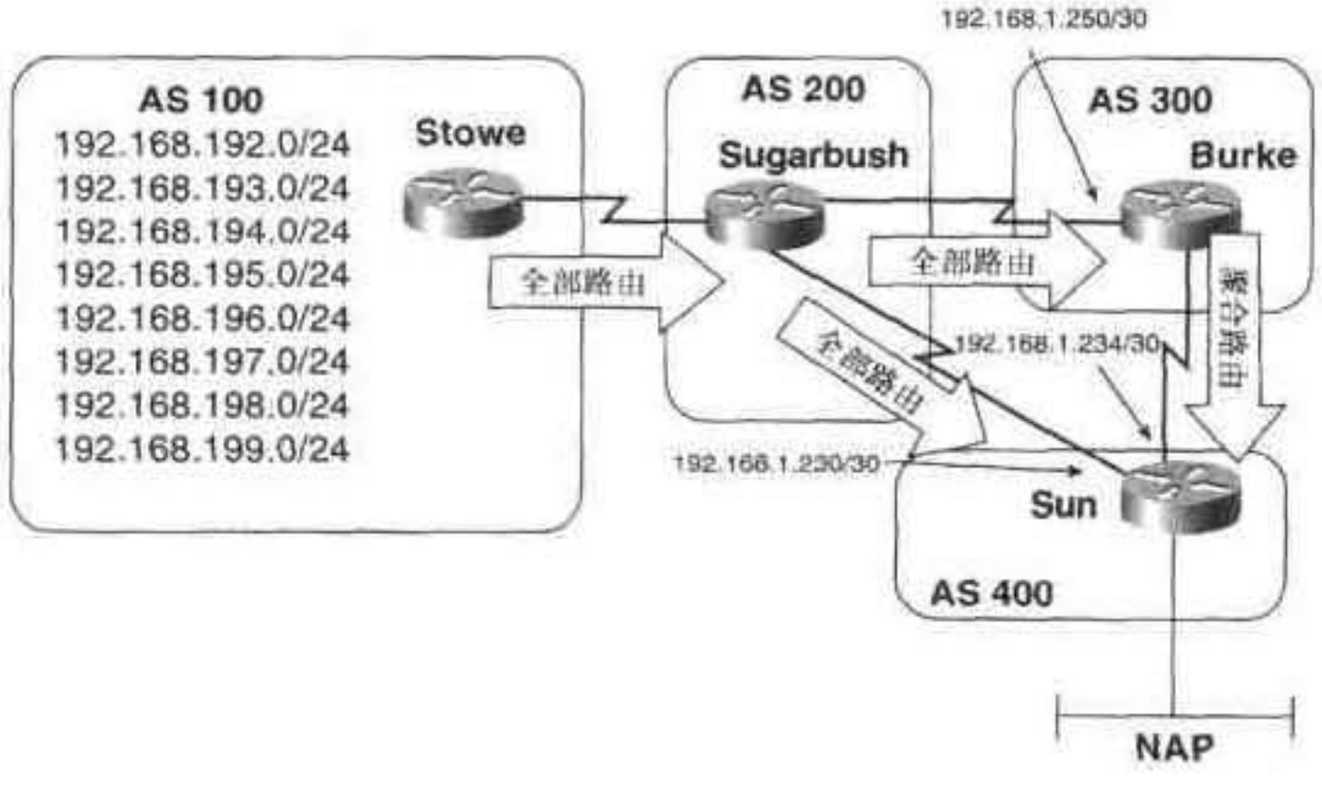


图 3-12　Burke 生成一个聚合路由并把它公布给 Sun

AS 300 中的路由器 Burke，抑制了 AS 100 中的更具体地址同时发送一个聚合路由给 AS 400 中的 Sun。例 3-61 中 Burke 的配置与这个案例研究中你所遇到的配置相同。

例 3-61　Burke 抑制了 AS 100 中的具体地址同时发送一个聚合路由给 AS 400 中的 Sun

```
router bgp 300
 aggregate-address 192.168.192.0 255.255.248.0 summary-only
 neighbor 192.168.1.234 remote-as 400
 neighbor 192.168.1.234 next-hop-self
 neighbor 192.168.1.249 remote-as 200
 neighbor 192.168.1.249 distribute-list 1 out
!
access-list 1 deny   192.168.192.0
access-list 1 permit any
```

例 3-61 中 Burke 的配置与前面配置之间的一个不同之处就是 **neighbor distribute-list** 命令的使用。这个命令执行了路由过滤功能而且与《TCP/IP 路由技术　第 1 卷》中讨论的 **distribute-list** 命令具有同样的操作方式。在本例中，过滤器防止了把聚合的路由再宣告回 Sugarbush。例 3-62 显示了 Sun 的 BGP 路由表。如所预想的一样，表中包括了来自 Sugarbush 更具体的路由信息与来自 Burke 的聚合的路由信息。在这个案例研究中一个有意思的地方就是与聚合路由有关的 AS_PATH 属性。一个聚合路由的 AS_PATH 属性中的 AS_SEQUENCE 是从发起聚合的 AS 开始的。Burke 发起了该聚合，于是 AS_SEQUENCE 只包括 AS 300。实际上，该聚合路由指向 AS 100 中的目的地；正如任何归纳都会出现的情况一样，聚合导致了路由信息的丢失。

例 3-62　来自 Burke 聚合的 AS_PATH 只包括发起该聚合路由的 AS 300

```
Sun#show ip bgp
BGP table version is 20, local router ID is 192.168.1.234
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0    192.168.1.229                          0 200 100 ?
*> 192.168.192.0/21 192.168.1.233                          0 300 i
*> 192.168.193.0    192.168.1.229                          0 200 100 ?
*> 192.168.194.0    192.168.1.229                          0 200 100 ?
*> 192.168.195.0    192.168.1.229                          0 200 100 ?
*> 192.168.196.0    192.168.1.229                          0 200 100 ?
*> 192.168.197.0    192.168.1.229                          0 200 100 ?
*> 192.168.198.0    192.168.1.229                          0 200 100 ?
*> 192.168.199.0    192.168.1.229                          0 200 100 ?
Sun#
```

例 3-63 中，在聚合路由里 Burke 设置了 ATOMIC_AGGREGATE 和 AGGREGATOR 属性(由 AS 300：　192.168.1.250 发起)，用于暗示出现了路由信息的丢失。

对于图 3-12 中的拓扑，路径信息的丢失会导致一定的问题。与 Burke 不同，在适当的位置上 Sun 没有路由过滤器，用来阻止把聚合路由公布给 Sugarbush。在来自 Sun 聚合路由的

AS_PATH 属性中，Sugarbush 看不到它自己的 AS 号，它就会将该聚合路由放到它自己的 BGP 表中，如例 3-64 所示。

例 3-63　来自 Burke 的聚合路由设置了 ATOMIC_AGGREGATE 和 AGGREGATOR 属性

```
Sun#show ip bgp 192.168.192.0 255.255.248.0
BGP routing table entry for 192.168.192.0/21, version 23
Paths: (1 available, best #1)
  Advertised to non peer-group peers:
    192.168.1.229
  300, (aggregated by 300 192.168.1.250)
    192.168.1.233 from 192.168.1.233 (192.168.1.250)
      Origin IGP, localpref 100, valid, external, atomic-aggregate, best, ref 2
Sun#
```

例 3-64　Sugarbush 接受了来自 Sun 的聚合路由

```
Sugarbush#show ip bgp
BGP table version is 19, local router ID is 172.20.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0    192.168.1.254      2297856             0 100 ?
*> 192.168.192.0/21 192.168.1.230                          0 400 300 i
*> 192.168.193.0    192.168.1.254      2297856             0 100 ?
*> 192.168.194.0    192.168.1.254      2297856             0 100 ?
*> 192.168.195.0    192.168.1.254      2297856             0 100 ?
*> 192.168.196.0    192.168.1.254      2297856             0 100 ?
*> 192.168.197.0    192.168.1.254      2297856             0 100 ?
*> 192.168.198.0    192.168.1.254      2681856             0 100 ?
*> 192.168.199.0    192.168.1.254            0             0 100 ?
Sugarbush#
```

如果来自 AS 100 的一条更具体路由失效了，Sugarbush 会丢弃任何目的地是该网络的数据包。但是如果存在聚合路由，这些数据包会与聚合路由匹配。例如，假设到 AS 100 中网络 192.168.197.0/24 的接口出现了故障，Stowe 公布了这个事实，于是所有的 BGP 表都会去掉到该目的地的路由。接下来，Sugarbush 收到了一个目的地为 192.168.197.5 的数据包，它没有找到更具体的路由，路由器会将该数据包与聚合路由匹配并且将该数据包转发给 Sun。Sun 同样没有找到更具体的路由，也将该数据包与聚合路由匹配，并将数据包转发给 Burke。Burke 是聚合路由的发起者，它也没有更具体的路由，因此它会将数据包丢弃。到一个无效目的地的数据包在被丢弃之前，没有必要地经过了两个路由器地转发。如果 Sugarbush 公布一个聚合路由给 Burke，这个问题会变得更加严重，在这种情况下，直到数据包的 TTL 超时，它们才会被丢弃掉。

为了改进这个问题，除了公布 AS_PATH 属性中的 AS_SEQUENCE 以外，Burke 还可以通过在 **aggregate-address** 命令中加入 **as-set** 关键字从而公布 AS_PATH 属性中的 AS_SET 来解决这个问题。正如第二章所讨论的，AS_SET 是到形成聚合路由的更具体地址的路径上 AS 号的一个无序列表。与 AS_SEQUENCE 不同，AS_SET 并不用来决定最短路径；它唯一的目

的是恢复被聚合路由丢失掉的环路检测功能。

例 3-65 给出了 Burke 公布 AS_SET 的配置。

例 3-65　配置 Burke，使它公布 AS_SET

```
router bgp 300
 aggregate-address 192.168.192.0 255.255.248.0 as-set summary-only
 neighbor 192.168.1.234 remote-as 400
 neighbor 192.168.1.234 next-hop-self
 neighbor 192.168.1.249 remote-as 200
 neighbor 192.168.1.249 distribute-list 1 out
!
access-list 1 deny   192.168.192.0
access-list 1 permit any
```

例 3-66 给出了 Sun 相应的 BGP 表，如例中所示，当将 Burke 配置成在它的 AS_PATH 属性中包含 AS_SET 时，聚合路由的 AS_PATH 属性中包括了聚合路由的路径上的所有 AS 号。到更具体路由的路径上的所有 AS 号都包括在聚合路由的 AS_PATH 属性中。当将聚合路由公布给 Sugarbush 时，路由器能够在 AS_PATH 属性中识别它自己 AS 号 200 从而不接受该路由。

例 3-66　Sun 相应的 BGP 表

```
Sun#show ip bgp
BGP table version is 10, local router ID is 172.21.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0    192.168.1.229                          0 200 100 ?
*> 192.168.192.0/21 192.168.1.233                          0 300 200 100 ?
*> 192.168.193.0    192.168.1.229                          0 200 100 ?
*> 192.168.194.0    192.168.1.229                          0 200 100 ?
*> 192.168.195.0    192.168.1.229                          0 200 100 ?
*> 192.168.196.0    192.168.1.229                          0 200 100 ?
*> 192.168.197.0    192.168.1.229                          0 200 100 ?
*> 192.168.198.0    192.168.1.229                          0 200 100 ?
*> 192.168.199.0    192.168.1.229                          0 200 100 ?
Sun#
```

需要注意的一点是当公布了 AS_SET 以后，聚合路由就继承了被聚合路由所有的属性。在图 3-12 的例子中，所有更具体路由的 AS_PATH 都是(300,200,100)，结果是，出现在 Sun BGP 表中的 AS_SET 是一个有序的序列，与 AS_SEQUENCE 没有什么区别。

图 3-13 给出了一个不同的拓扑。在该拓扑中加入了一个新的 AS，同时将网络 192.168.197.0/24 从 AS 100 转移到了 AS 500 中。Burke 也会收到同样的路由，但是并不是所有的 AS_PATH 属性都与它们匹配，因此，现在的 AS_SET 是以一个无序的序列来公布的，如例 3-67 所示出现在 Sun BGP 表中的 AS_SET 是一个无序的序列，与有序的 AS_SEQUENCE 相区别。

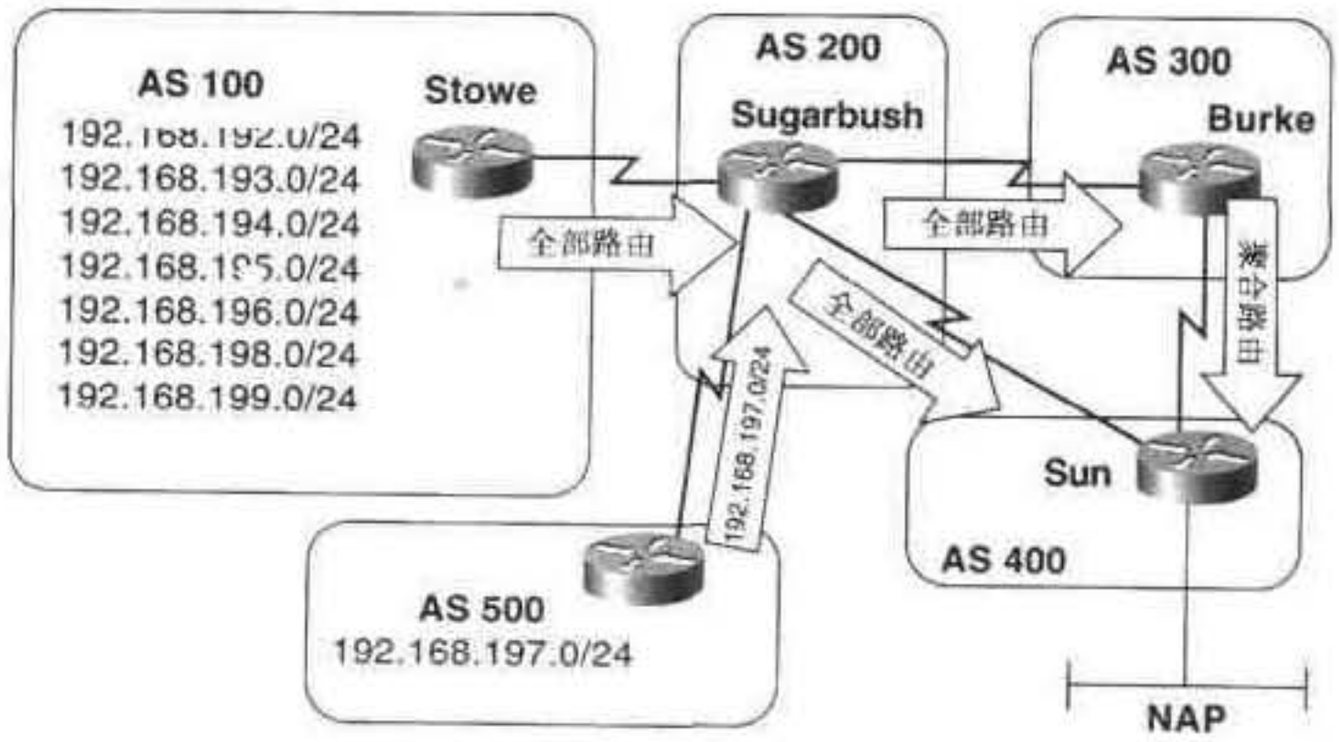


图 3-13　现在 Burke 不能再将 AS_SET 描述成一个有序的序列

例 3-67　Sun 的 BGP 表中 AS_SET 是个无序序列

```
Sun#show ip bgp
BGP table version is 35, local router ID is 172.21.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0    192.168.1.229                          0 200 100 ?
*> 192.168.192.0/21 192.168.1.233                          0 300 {200,100,500} ?
*> 192.168.193.0    192.168.1.229                          0 200 100 ?
*> 192.168.194.0    192.168.1.229                          0 200 100 ?
*> 192.168.195.0    192.168.1.229                          0 200 100 ?
*> 192.168.196.0    192.168.1.229                          0 200 100 ?
*> 192.168.197.0    192.168.1.229                          0 200 500 i
*> 192.168.198.0    192.168.1.229                          0 200 100 ?
*> 192.168.199.0    192.168.1.229                          0 200 100 ?
Sun#
```

7. 基于选中的更具体路由的聚合路由

在某些情况下，你可能希望公布一个带有 AS_SET 的聚合路由但是你不希望该聚合路由继承所有被聚合路由的所有的属性。在图 3-14 中，如图所示，Sugarbush 收到了来自 AS 100 和 AS 500 的所有路由同时它公布一条聚合路由给 Burke，Sugarbush 公布了一个带有 AS_SET 的聚合路由，但是该聚合路由必须不能继承来自 192.168.197.0/24 的 NO_EXPORT COMMUNITY 属性。

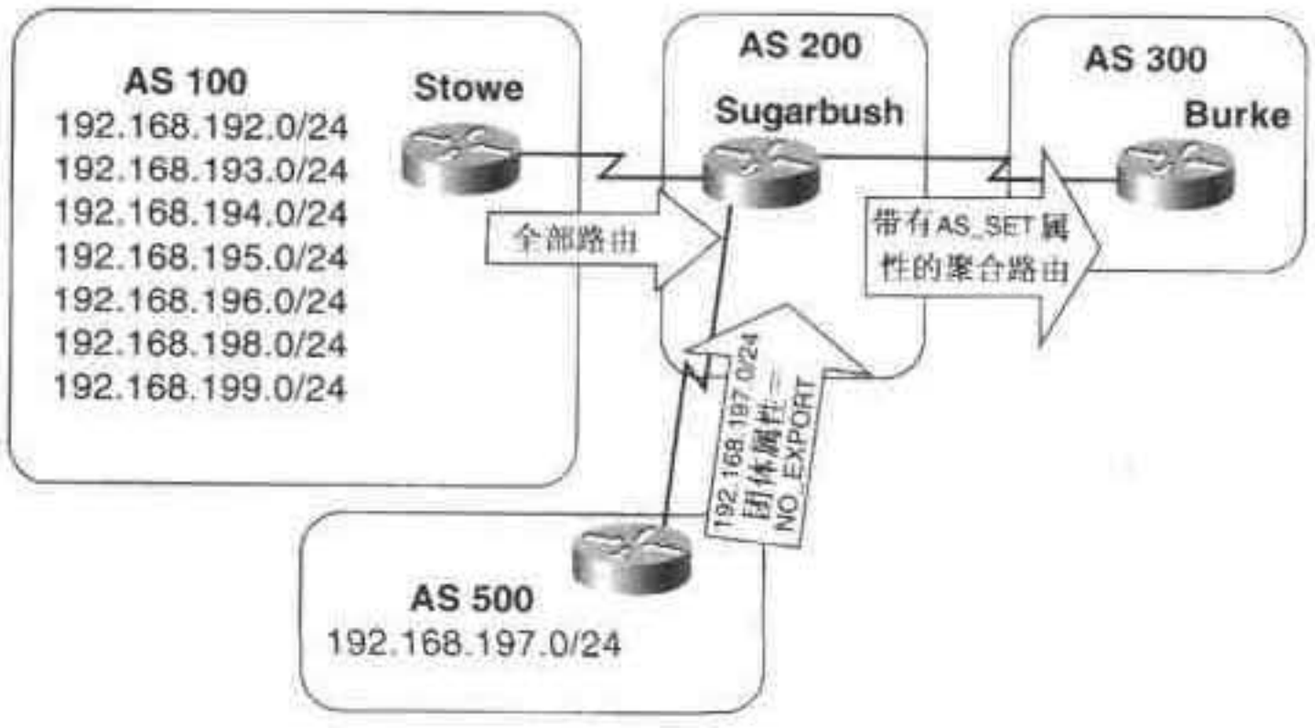


图 3-14　Sugarbush 公布了一个带有 AS_SET 的聚合路由

图 3-14 设置的问题在于 AS 500 公布了带有 NO_EXPORT 团体属性的 192.168.197.0/24。当 Sugarbush 使用 AS_SET 选项时，聚合路由继承了 NO_EXPORT 属性，如例 3-68 所示(在该例中，通过 **show ip bgp community no-export** 命令可以显示出所有的带有 NO_EXPORT 团体属性的路由；在这种情况下，聚合路由继承了被聚合路由 192.168.197.0/24 的属性)。注意 NO_EXPORT 属性只分配给本地的聚合路由，而不会加入到宣告的聚合路由当中。结果是，Sugarbush 对该属性起作用，它不会公布该聚合路由。

例 3-68　show ip bgp community no-export 命令的显示结果

```
Sugarbush#show ip bgp community no-export
BGP table version is 19, local router ID is 172.20.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.192.0/21 0.0.0.0                              32768 {100,500} ?
s> 192.168.197.0    192.168.1.1              0             0 500 i
Sugarbush#
```

将要讨论的 **aggregate-address** 命令的最后一个选项是 **advertise-map**，在图 3-14 的互联网络情况下，如果 Sugarbush 在形成聚合路由时不考虑 192.168.197.0/24，聚合路由就不会继承该路由的属性。例 3-69 给出了使用 **aggregate-address** 命令 **advertise-map** 选项的 Sugarbush 的配置。

例 3-69　配置 Sugarbush，使它选择构成聚合路由的路由

```
router bgp 200
 aggregate-address 192.168.192.0 255.255.248.0 as-set summary-only advertise-map
ALLOW_ROUTE
 neighbor 192.168.1.1 remote-as 500
 neighbor 192.168.1.250 remote-as 300
 neighbor 192.168.1.254 remote-as 100
!
access-list 1 deny   192.168.197.0
access-list 1 permit any
!
route-map ALLOW_ROUTE permit 10
 match ip address 1
```

例 3-69 中的配置里 **advertise-map** 选项指向一个名为 ALLOW_ROUTE 的路由图，该路由图标识构成聚合路由基础的更具体路由。路由图指向访问列表 1，该访问列表拒绝 192.168.197.0 但是接受其他路由。现在，因为 Sugarbush 在形成聚合路由时忽略了 192.168.197.0/24，因此该聚合路由不会继承 NO_EXPORT 属性，如例 3-70 所示。

例 3-70　Sugarbush 用 advertise-map 选项重新配置以后，聚合路由不再具有 NO_EXPORT 属性

```
Sugarbush#show ip bgp community no-export
BGP table version is 18, local router ID is 172.20.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
s> 192.168.197.0    192.168.1.1              0             0 500 i
Sugarbush#
```

限制构成聚合路由的更具体路由的前缀可能会带来一定的脆弱性。在图 3-14 的网络以及相关的配置中，假设 Stowe 和 Sugarbush 之间的链路出现故障，而聚合路由只是由来自 AS 100 的前缀形成的，于是不再公布该聚合路由。结果是，经过 AS 300 以及再往上的 AS 将无法到达 AS 500 内的目的地。

3.1.8　管理 BGP 连接

Cisco IOS 提供了几个帮助管理 BGP 对等之间连接的特性。第一个就是 **neighbor description** 命令。正如 **description** 命令可以在一个接口配置下输入一样，**neighbor description** 命令对路由器没有功能上的影响，相反，它为配置加入一个消息文本串。你已经遇到了许多应用于 BGP 邻居的配置选项，在本章剩余的章节中你还会遇到几个。当配置变得详细的时候，**neighbor description** 命令帮助提供一个提示，说明每个邻居是谁以及所在的位置。例 3-71 说明了 **neighbor description** 命令的使用情况。

例 3-71　neighbor description 命令帮助减少对详细 BGP 配置的迷惑

```
router bgp 200
 aggregate-address 192.168.192.0 255.255.248.0 as-set summary-only
 neighbor 192.168.1.1 remote-as 500
 neighbor 192.168.1.1 description ******T1 to Sun, Ckt. ID 54.HCGS.123456
 neighbor 192.168.1.237 remote-as 200
 neighbor 192.168.1.237 description ****Ethernet to Diamond, Interface E0
 neighbor 192.168.1.250 remote-as 300
 neighbor 192.168.1.250 description ****T1 to Burke, Ckt. ID 54.HCGS.654321
 neighbor 192.168.1.254 remote-as 100
 neighbor 192.168.1.254 description ****56K to Stowe, Ckt. ID 54.DWDA.987654
```

description 可以保留任何你想加入的有用的信息，它可以保留高达 80 个字符的信息。这个例子包括了链路类型、邻居路由器的名字以及线路 ID。

两个 BGP 邻居还可以使用 **neighbor password** 命令，通过一个口令来互相验证。在例 3-72 的配置中，为邻居 192.168.1.253 输入了一个口令 noTeU2n0。

例 3-72　BGP 邻居认证

```
service password-encryption
!
router bgp 100
 redistribute eigrp 100
 neighbor 192.168.1.253 remote-as 200
 neighbor 192.168.1.253 description ****56K to Sugarbush, Ckt. ID 54.DWDA.987654
 neighbor 192.168.1.253 password 7 14191D3F5831782574
```

注意在路由器的全局配置中还使用了 **service password-encryption** 选项，当显示配置的时候将口令加密，增加了保密性。

不同邻居连接之间的口令可以不同，也可以全部相同。你可以用一个对等组的名字取代一个 IP 地址，从而对一个对等组使用一个通用的口令而不是为每个邻居单独分配一个口令。但是，对于每个 BGP 连接，两个邻居必须有相同的口令。

当一个 BGP 连接配置了口令以后，IOS 采用 MD5 验证方式。MD5 是由 RSA Data Security,Inc.提出的单向消息摘要*(message digest)*或*安全 hash* 功能，偶尔它还被称做*密码校验和*，因为在某种程度上它的工作方式与算术校验和相同。MD5 根据任意长度的纯文本消息(在这个例子中，是 BGP 消息)和口令计算出一个 128 比特的 hash 值。这个“指纹”会随着消息一起传输。接收端与发送端有着相同的口令，它会计算它自己的 hash 值。如果消息没有发生任何变化，接收端的 hash 值应该与发送端的随消息一起传过来的 hash 值相匹配。如果不知道口令，不可能将 hash 值解密(没有如此巨大的计算能力)，因此一个没有通过认证的路由器不管是恶意的还是由于意外的事件，都不可能与一个运行邻居认证的路由器建立对等关系。

另外一个用于邻居间连接的可选项是 **neighbor advertisement-interval** 和 **neighbor version** 命令。第一个命令将 BGP 缺省的 Update 间隔改为 0~600 秒之间的一个值。如果在一条链路上有大量的路由信息需要交换，通过该命令增加宣告之间的间隔可以减少对链路可用带宽的影响。但是，你不应该改变宣告间隔，除非你完全了解了这样做的后果，例如可能会缩短再收敛的时间。

如果邻居不能支持缺省的 BGP-4，那么 **neighbor version** 命令会很有用。不用让支持 BGP-4 的路由器协商到邻居所支持的版本，可以配置路由器根据特定的邻居来运行一个特定的 BGP 版本。结果是，建立一个 BGP 连接需要的时间降低了。如果两个邻居都运行 BGP-4，就没必要使用该命令。

Cisco IOS 执行的 BGP 决定进程在多个到同一个目的地的路由之间进行选择时，它会考虑到 AS_PATH 属性的长度。但是，RFC 1771 没有包括这个步骤，因此，Cisco 运行 BGP 的路由器偶尔会与另外一个厂商的不考虑 AS_PATH 长度的路由器对等。如果这种情况会产生潜在的路由决定的不一致，可以通过 **bgp bestpatb as-path ignore** 命令通知 Cisco 的路由器在路由决定进程中忽略 AS_PATH 的长度。

在某些策略环境下，你可能希望限制允许一个路由器从一个邻居接受前缀的数量。例如，一个特定的 AS 可能只能公布一定数量的前缀，任何超出这个数量的前缀可能就意味着 AS 管理者的一个配置错误。或者，可能是因为存在着一个服务协议，也就是说，你同意只接受有限数量的前缀；如果邻居要公布大量的前缀，该 AS 的管理者首先要得到你的同意。为了执行这样一个策略，你可以使用 **neigbbor maximum-prefix** 命令。看一下例 3-73 中的配置。

例 3-73　限制一个路由器可以从一个邻居接受的前缀的数量

```
router bgp 100
 redistribute eigrp 100
 neighbor 192.168.1.253 remote-as 200
 neighbor 192.168.1.253 maximum-prefix 300
```

例 3-73 中，配置一个路由器使得它可以从邻居 192.168.1.253 接受的最大前缀数量为 300 个。如果前缀数量超过了该限制，路由器就会关闭与邻居之间的 BGP 会话，只有你使用了 **clear ip bgp 192.168.1.253**，该会话才能重新建立起来。

因为超过前缀的最大数量而中断一个对等会话可能会导致非常严重的后果，但是，当发生这种情况的时候，你还是希望被通知。例 3-74 中的配置没有中断会话，而是由路由器生成

了日志消息。

例 3-74　配置路由器，当邻居发送的前缀超过了限制，路由器就要生成一个日志消息

```
router bgp 100
 redistribute eigrp 100
 neighbor 192.168.1.253 remote-as 200
 neighbor 192.168.1.253 maximum-prefix 300 warning-only
```

当邻居公布的前缀超过了最大值的 75%，路由器就要生成一个日志消息——在这个例子里，就是 225 个前缀。你可以改变这个缺省设置。例 3-75 中，当邻居公布的前缀超过了最大值的 90%时，路由器就生成一个日志消息。

例 3-75　配置路由器，当邻居发送的前缀超过了最大值的 90%时，路由器就要生成一个日志消息

```
router bgp 100
 redistribute eigrp 100
 neighbor 192.168.1.253 remote-as 200
 neighbor 192.168.1.253 maximum-prefix 300 90 warning-only
```

偶尔你可能暂时需要中断一个对端的连接但是并不是删除它的邻居配置。例 3-76 的配置中使用了 **neighbor shutdown** 命令来中断到 192.168.1.237 的连接。

例 3-76　暂时中断到一个邻居的连接

```
aggregate-address 192.168.192.0 255.255.248.0 as-set summary-only
 neighbor 192.168.1.1 remote-as 500
 neighbor 192.168.1.1 description ******T1 to Sun, Ckt. ID 54.HCGS.123456
 neighbor 192.168.1.237 remote-as 200
 neighbor 192.168.1.237 description ****Ethernet to Diamond, Interface E0
 neighbor 192.168.1.237 shutdown
 neighbor 192.168.1.250 remote-as 300
 neighbor 192.168.1.250 description ****T1 to Burke, Ckt. ID 54.HCGS.654321
 neighbor 192.168.1.254 remote-as 100
 neighbor 192.168.1.254 description ****56K to Stowe, Ckt. ID 54.DWDA.987654
```

例 3-76 中的 **neighbor shutdown** 命令中断了 TCP 端口 179 到一个特定邻居的连接，该命令与 **shutdown** 命令关闭一个端口的方式相似。在使用 **show ip bgp neighbor** 命令后，邻居的状态为管理上的中断。

最后，你可以通过 **timers bpg** 命令来改变缺省的 60 秒的 BGP Keepalive 时间和 180 秒的保持时间。有一个论点是降低这些时间间隔可以加快检测到不可靠邻居的过程，但是对于不可靠邻居问题实际的解决办法是去除使邻居不可靠的因素。使用该命令以后缺省时间间隔的改变会应用于每一个邻居，而不只是只应用于一个单一的邻居。即使一个邻居有不同的 Keepalive 和保持时间间隔，对等进程也会把自动协商使用的时间间隔作为该进程的一部分，因此，在通常情况下，很少会用到这个命令。

3.2　路由策略

Webster 字典中 *Policy* 的定义是“a high-level overall plan embracing the general goals and acceptable procedure”。路由策略就是定义路由器如何接受路由以及公布路由的一个计划。该策略的目标是正确地转发或抑制 IP 数据包。*可接受的过程*意味着执行的路由策略对路由器的 CPU 和内存资源、链路的带宽资源以及邻居路由器的策略造成最少的负面的影响。

> 注　不要把用于本节以及全书的术语*路由策略*与 Cisco 的特色策略路由相混淆。这些特殊版本的静态路由是通过 **ip policy route-map** 命令来执行的，并在 TCP/IP 路由技术，第 1 卷，第 14 章的“路由图(Route Map)”中讨论过。

路由策略十分重要，尤其是在 BGP 环境下更是如此。根据它的特点，BGP 将自治系统互连起来，而邻居的自治系统不在你的管理控制之下，这时你要仔细地计划你的路由策略。你必须要全面地了解哪些数据包要转发给哪些邻居，而你又要从这些邻居接收哪些数据包，以及在什么情况下，才能转发和接收这些数据包。当规划出一个完整的路由策略以后，你就要准备设计配置来使这些策略可用了，这个过程要求你完全了解可用的 BGP 配置选项。实验室在这个过程就显得十分有用了，你可以在将该设计用于已经在运行的网络之前，在实验室测试你的设计并验证你的假设。一个配置只有在经过了完全的设计、理解和验证以后，才可以执行它。

在上面的每一步中出现错误都会给你的互联网络造成严重的后果，并且可能会影响用户、客户、业务供应商以及管理人员。因为可能会破坏通过 Internet 的业务量，许多业务供应商都会中断与那些经常错误配置策略的客户的对等连接。由于 BGP 对等被拒绝而造成的经济影响——尤其是当客户本身就是一个业务供应商的情况下——是很严重的。没有其他的 IP 路由策略能够像 BGP 一样提供如此强大的策略特性，但是也没有其他的 IP 路由策略能够向 BGP 一样具有如此大的将你陷于困境的潜能。

本节将说明在 BGP 下配置路由策略可用的选项。你应该已经了解了一些最基本的配置路由策略的工具。如果已经阅读过了第一卷而且本卷也是从头开始阅读的，那么你应该知道如何配置任何一个 IP 路由协议来公布选中的路由以及如何将路由从一个协议再分发给另外一个协议。你也应该知道如何使用路由过滤器和路由图以及如何使用不同 IP 路由策略的管理距离和度量。你应该了解有多于一条的路径进入或者离开一个区、一个路由域或者一个 AS 可能会造成的危害而且你也应该知道一些避免这些危害的策略。在本章中，你已经见过了几个有关一些 BGP 特殊工具的详细的例子，例如使用 ORIGIN 和团体属性以及过滤来自一个单一邻居的 NLRI。

最后，你应该了解出站路由的公布会影响入业务量而且入站路由的公布会影响出业务量。在设计一个路由策略时，将公布路由与接收路由分开考虑以及同时设计入站和出站路由策略是极其重要的。

3.2.1　重置 BGP 连接

当运行 BGP 的路由器的设置发生变化时，为了使该变化起作用，首先重置到有关邻居

的 BGP 连接通常是很有必要的。Cisco IOS 软件命令概要中列出了必须重置一个 BGP 连接的几种情况：

- 补充或者改变与 BGP 有关的访问列表
- 改变与 BGP 有关的权值
- 改变与 BGP 有关的分配列表
- 改变与 BGP 有关的计时器的相关规定
- 改变 BGP 的管理距离
- 改变与 BGP 有关的路由地图

上面列表中的所有项都会以某种方式影响一条路由的 BGP 路由策略，这是为什么会要求重置 BGP 连接的一个原因。如果你正在改变路由策略，你当然不会希望该策略会不起作用。而且，你肯定希望充分地配置一个新的策略然后再执行该策略。当输入路由策略命令的时候，如果允许每条命令都在你输入的时候就起作用，可能会引起路由环路、黑洞、安全缺口或者其他等同意义的令人厌烦的结果。

可以通过 **clear ip bgp** 命令来重置连接，在 IOS Exec 模式中提出了该命令。可以重置到特定目的地、一个对等组或者所有路由器邻居的连接。重置到一个特定邻居的连接，必须要明确邻居的 IP 地址。例如，要重置到邻居 **192.168.1.253** 的连接，使用的命令是 **clear ip bgp 192.168.1.253**。要重置到一个名为 **subscribers** 的对等组中所有成员的连接，使用的命令是 **clear ip bgp subscribers**。要重置路由器所有的 BGP 连接，使用的命令是 **clear ip bgp ***。

实际上，你应该只重置那些会受到你所做变动影响的连接。当一个连接被重置以后，运行 BGP 的路由器会向邻居发送一个 Cease Notification 消息，于是 BGP 会话被关闭，TCP 会话也被关闭而且所有的高速缓存都处于无效状态，然后一个新的 BGP 会话开始了。当这个过程发生的时候，来自和去往该连接的业务都会中断。如果只有一、两条连接会受到新配置的影响，但是你重置了所有的连接，在生产环境下将会导致非常严重的后果。

Cisco 为全部重置提供了一个可以替代的办法，就是所谓的软重配置。软重新配置并不是完全中断所有的连接，然后再重新建立 TCP 和 BGP 连接，它只是通过触发更新来使新的路由策略生效。软重新配置可以只用于出站连接、入站连接或者同时作用于入站和出站的连接。当影响出业务量的策略发生变化，采用出重新配置；当影响入业务量的策略发生变化则采用入重新配置。和“硬”重置类似，你可以重置到一个特定邻居、一个对等组或者所有 BGP 连接。

例如，假设你改变一个 BGP 路由器上的策略，而该策略会影响到邻居 192.168.1.253 的出业务量。在新策略下触发到该邻居的更新信息，命令是 **clear ip bgp 192.168.1.253 soft out**。

当你改变的策略影响到入业务量时，使用入站软重新配置。从 Cisco IOS 软件版本 12.1 开始，入路由支持动态软重配置。但是，在这之前的版本，在使用入站软重配置之前，必须在 BGP 配置中加入一个 **neighbor soft-reconfiguration inbound** 命令。然后对业务量受到新策略影响的每一个邻居使用 **clear ip bgp soft in** 命令。假设你改变图 3-14 中 Stowe 路由器上的入路由策略，并且该策略会影响到从邻居 192.168.1.235 接收到的业务量，例 3-77 给出了 Stowe 的 BGP 配置情况。

例 3-77　将邻居配置成入站软重配置

```
router bgp 100
 redistribute eigrp 100
 neighbor 192.168.1.253 remote-as 200
 neighbor 192.168.1.253 soft-reconfiguration inbound
```

当加入了 **neighbor soft-reconfiguration inbound** 命令以后，路由器开始存储指定邻居的更新信息。这些更新信息没有被任何入站策略修改，因此当触发了软重配置以后，路由器能够正确使用新的策略。在所示的例子中，当为 Stowe 配置了新的入站策略，在 Exec 模式下输入 **clear ip bgp 192.168.1.235 soft in** 命令，于是，路由器就使用存储的、未经改动的更新信息来执行新的入站策略。

也可以同时触发入站和出站策略的软重新配置。例如，没有 **in** 或者 **out** 关键词的 **clear ip bgp 192.168.1.235 soft** 命令在向邻居 192.168.1.235 发送更新信息的同时也将入站策略应用于来自该邻居的存储的更新信息。

使用入站软重新配置一个很明显的缺点就是要求大量的内存来存储更新信息。如果邻居公布了大量的路由，或者有来自多个邻居的更新信息需要存储，那么对本地路由器的内存的影响是非常大的。但是这种内存上的负担可以避免。当一个路由器上的入站 BGP 路由策略发生变化时，它的邻居可以发送一个出站软重新配置。本地路由器接收到了来自它的邻居的更新信息，它就会使用新的入站策略。只有当你不能发送或者已经准备发送时，你才应该使用入站软重新配置，你可能会发现如果策略的改变影响到来自多数邻居的业务量，以及当你必须同时对所有的邻居执行策略变化时，入站软重新配置是必需的。在任何情况下，你都必须仔细考虑对本地内存的影响。

3.2.2　案例研究：通过 NLRI 过滤路由

路由过滤器几乎是任何一个路由策略的核心部分。如果你同时具有入站和出站路由策略，你最想定义的就是路由器要接受哪条路由以及公布哪条路由。

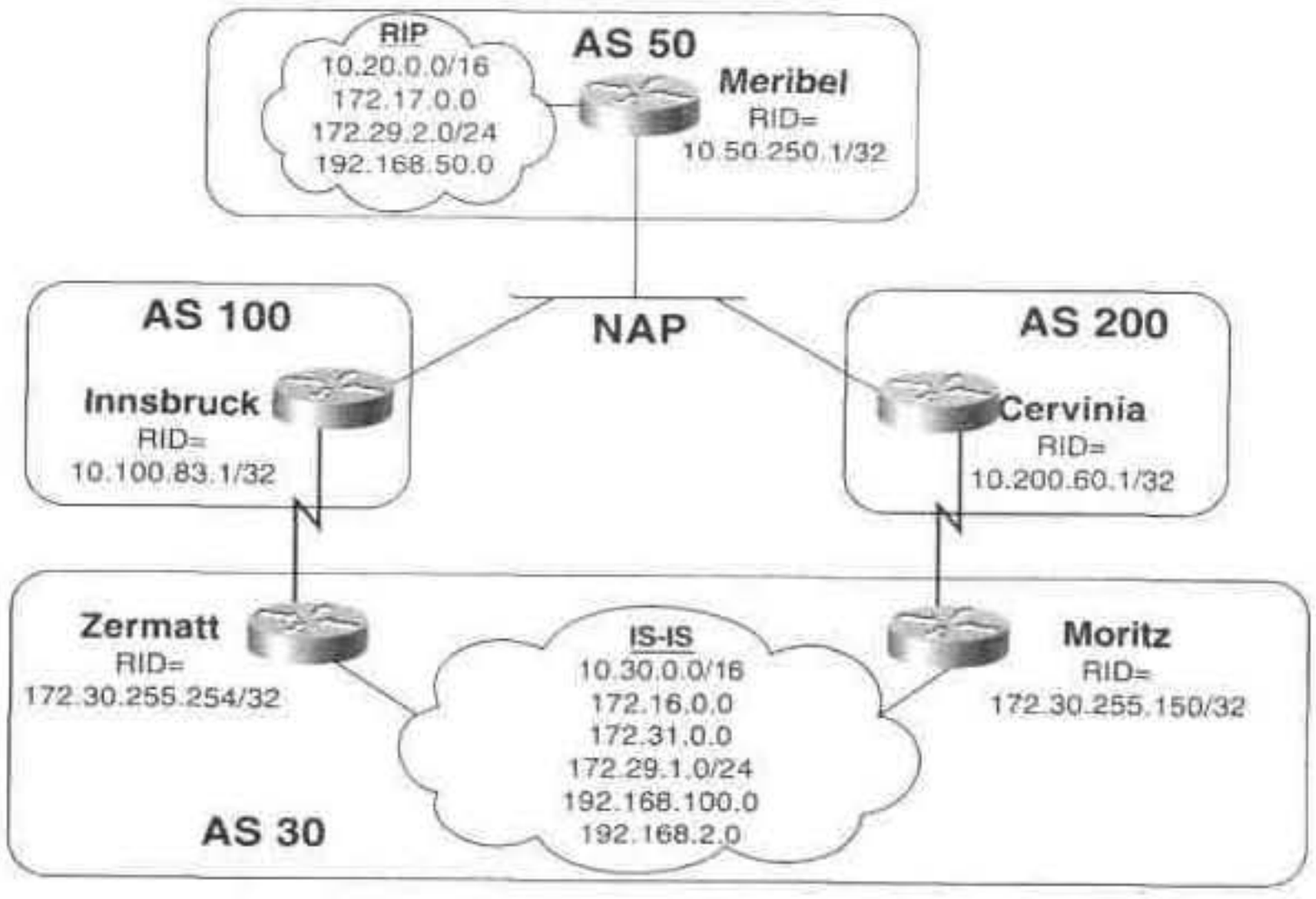


图 3-15　AS 30 多宿主到不同的转接 AS

最初 BGP 可用的最简单的路由过滤器是由 **distribute-list** 命令定义的。这个路由过滤器可以是为一个邻居或是为一个对等组定义的并且该路由过滤器指向一个它将要对之起作用的定义前缀的访问列表或者是 NLRI。

图 3-15 中的网络互联是用来对这个以及后面两种情况的研究。

AS 50 的 IGP 是 RIP，AS 30 的 IGP 是 IS-IS。AS 100 和 AS 200 是转接 AS。例 3-78 给出了图 3-15 中这 5 个路由器初步的配置。

例 3-78　通过 NLRI 过滤路由：初始路由器配置

Zermatt

```
interface Loopback0
 ip address 172.30.255.254 255.255.255.255
 ip router isis
!
router isis
 net 30.5678.1234.defa.00
 redistribute bgp 30 metric 0 metric-type internal level-2
!
router bgp 30
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 no auto-summary
!
ip classless
ip route 10.100.83.1 255.255.255.255 Serial1.906
```

Moritz

```
interface Loopback0
 ip address 172.30.255.150 255.255.255.255
 ip router isis
!
router isis
 net 30.1234.5678.abcd.00
 redistribute bgp 30 metric 0 metric-type internal level-2
!
router bgp 30
 redistribute isis level-2
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 no auto-summary
!
ip route 10.200.60.1 255.255.255.255 Serial1.803
```

Innsbruck

```
interface Loopback0
 ip address 10.100.83.1 255.255.255.255
!
router bgp 100
 neighbor 10.50.250.1 remote-as 50
 neighbor 10.50.250.1 ebgp-multihop 2
 neighbor 10.50.250.1 update-source Loopback0
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 neighbor 172.30.255.254 remote-as 30
```

```
 neighbor 172.30.255.254 ebgp-multihop 2
 neighbor 172.30.255.254 update-source Loopback0
 no auto-summary
!
ip classless
ip route 10.50.250.1 255.255.255.255 Ethernet0
ip route 10.200.60.1 255.255.255.255 Ethernet0
ip route 172.30.255.254 255.255.255.255 Serial1.609
```

Cervinia

```
interface Loopback0
 ip address 10.200.60.1 255.255.255.255
!
router bgp 200
 neighbor 10.50.250.1 remote-as 50
 neighbor 10.50.250.1 ebgp-multihop 2
 neighbor 10.50.250.1 update-source Loopback0
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 172.30.255.150 remote-as 30
 neighbor 172.30.255.150 ebgp-multihop 2
 neighbor 172.30.255.150 update-source Loopback0
 no auto-summary
!
ip classless
ip route 10.50.250.1 255.255.255.255 Ethernet0/0
ip route 10.100.83.1 255.255.255.255 192.168.4.2
ip route 172.30.255.150 255.255.255.255 Serial0/1.308
```

Meribel

```
interface Loopback0
 ip address 10.50.250.1 255.255.255.255
!
router rip
 redistribute bgp 50 metric 1
 network 10.0.0.0
!
router bgp 50
 redistribute rip
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 no auto-summary
!
ip classless
ip route 10.100.83.1 255.255.255.255 Ethernet0
ip route 10.200.60.1 255.255.255.255 Ethernet0
```

注意在图 3-15 中没有给出任何数据链路的 IP 地址。所有的 BGP 会话都是在路由器 ID 之间进行配置，这些路由器 ID 是由路由器的 Loopback 接口定义的，因此数据链路的地址与本例无关。在这些配置中还有一个重要的方面就是静态路由，这些静态路由告诉路由器如何找到邻居路由器的 ID，没有这些静态路由，不能建立 BGP 会话。

注意：　这个案例研究和接下来的案例研究都使用 IGP 和 BGP 之间的双向再分发，以便较为容易地解释路由策略的应用。但是，再一次，请记住在实际情况中，双向再分发通常不是一个好的做法。更重要的是，如果涉及到许多前缀，通常，从 BGP 到任何一个 IGP 的再分发是一个很不好的做法。

图 3-15 中的 AS 30 因为备份的原因是多宿主的，但是它不能作为转接 AS。也就是说，AS 100 和 AS 200 之间的业务量不能通过 AS 30 进行转发。例 3-79 中，Innsbruck 的 BGP 表显示到 AS 50 内目的地的一个下一跳路由器是 Zermatt(172.30.255.254)。出现这种情况的原因是 Meribel 同时把这些地址公布给了 Cervinia 和 Innsbruck。Cervinia 将这些地址公布给了 Moritz，而 Moritz 又把它们再分发到了 IS-IS 中，于是 Zermatt 在 AS 30 内从它的 IS-IS 邻居处学习到了这些地址，将它们再分发给 BGP，并公布给 Innsbruck。

例 3-79　Innsbruck 将 Zermatt 看作是到 AS 50 内目的地的一个可行的下一跳

```
Innsbruck#show ip bgp
BGP table version is 21, local router ID is 10.100.83.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*  10.20.0.0/16     172.30.255.254          20             0 30 ?
*                   10.200.60.1                            0 200 50 ?
*>                  10.50.250.1              0             0 50 ?
*  10.30.0.0/16     10.50.250.1                            0 50 200 30 ?
*>                  172.30.255.254          20             0 30 ?
*                   10.200.60.1                            0 200 30 ?
*  10.50.250.1/32   172.30.255.254          20             0 30 ?
*                   10.200.60.1                            0 200 50 ?
*>                  10.50.250.1              0             0 50 ?
*> 10.200.60.1/32   10.50.250.1              0             0 50 ?
*  172.16.0.0       10.50.250.1                            0 50 200 30 ?
*>                  172.30.255.254          20             0 30 ?
*                   10.200.60.1                            0 200 30 ?
*  172.17.0.0       172.30.255.254          20             0 30 ?
*                   10.200.60.1                            0 200 50 ?
*>                  10.50.250.1              1             0 50 ?
*  172.29.0.0       172.30.255.254          20             0 30 ?
*                   10.200.60.1                            0 200 50 ?
*>                  10.50.250.1              1             0 50 ?
*  172.29.1.0/24    10.50.250.1                            0 50 200 30 ?
*>                  172.30.255.254          20             0 30 ?
*                   10.200.60.1                            0 200 30 ?
*> 172.30.255.150/32 172.30.255.254          30             0 30 ?
*  172.30.255.254/32 10.50.250.1                           0 50 200 30 ?
*>                  10.200.60.1                            0 200 30 ?
*  172.31.0.0       10.50.250.1                            0 50 200 30 ?
*>                  172.30.255.254          20             0 30 ?
*                   10.200.60.1                            0 200 30 ?
*  192.168.2.0/30   10.50.250.1                            0 50 200 30 ?
*>                  10.200.60.1                            0 200 30 ?
*> 192.168.2.4/30   172.30.255.254          20             0 30 ?
*  192.168.50.0     172.30.255.254          20             0 30 ?
*                   10.200.60.1                            0 200 50 ?
*>                  10.50.250.1              1             0 50 ?
*  192.168.100.0    10.50.250.1                            0 50 200 30 ?
*>                  172.30.255.254          20             0 30 ?
*                   10.200.60.1                            0 200 30 ?
Innsbruck#
```

如果 Innsbruck 丢失了与 NAP 的连接，它会把目的地是 AS 50 的数据包转发到 Zermatt，将 AS 30 作为一个转接 AS。为了防止出现这种情况，在 Zermatt 和 Moritz 处执行了一个出站策略，也就是说，它们只能公布 AS 30 内部的地址。例 3-80 给出了 Zermatt 的 BGP 配置。

例 3-80　在 Zermatt 处执行的出站策略使得它只公布 AS 30 内部的地址

```
router bgp 30
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 distribute-list 1 out
 no auto-summary
!
access-list 1 permit 192.168.100.0
access-list 1 permit 10.30.0.0
access-list 1 permit 192.168.2.0
access-list 1 permit 172.29.1.0
access-list 1 permit 172.31.0.0
access-list 1 permit 172.16.0.0
```

在该例子中只显示了与本例有关的 Zermatt 的部分配置。

Moritz 的 distribute-list 配置与 Zermatt 的相同，只是它们引用的参考邻居有所不同。在两个路由器中，访问列表 1 表示只接受内部的路由而拒绝所有的外部路由。

例 3-79 中另外一个问题就是 Innsbruck 不仅把 Meribel(10.50.250.1)作为到 AS 50 内目的地的下一跳路由器，还把 Cervinia(10.200.60.1)也作为一个到 AS 50 内目的地的下一跳路由器。同样地，这样两个条目也出现在 Cervinia 的 BGP 表中，Meribel 和 Innsbruck 都是 Cervinia 到 AS 50 中目的地的下一跳路由。出现这个问题是因为不仅 Innsbruck 和 Cervinia 都是 Meribel 的对端，而且它们还互相有对等关系。

这个问题不会导致任何路由功能障碍——如果 Meribel 撤消了一条路由，该撤消动作会同时公布给 Cervinia 和 Innsbruck。现在的问题已经不在是美观与清晰的问题了，BGP 表中不应该存在无效路由。

为解决这个问题，例 3-81 给出了对 Innsbruck 的 BGP 配置。

例 3-81　去掉 Innsbruck BGP 配置中的冗余下一跳路由器

```
router bgp 100
 neighbor 10.50.250.1 remote-as 50
 neighbor 10.50.250.1 ebgp-multihop 2
 neighbor 10.50.250.1 update-source Loopback0
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 neighbor 10.200.50.1 distribute-list 1 out
 neighbor 172.30.255.254 remote-as 30
 neighbor 172.30.255.254 ebgp-multihop 2
 neighbor 172.30.255.254 update-source Loopback0
 no auto-summary
!
access-list 1 deny   10.20.0.0
access-list 1 deny   192.168.50.0
access-list 1 deny   172.29.0.0
access-list 1 deny   172.17.0.0
access-list 1 permit any
```

Cervinia 有一个和 Innsbruck 一样的路由过滤器。该过滤器阻止 AS 50 的地址在 Innsbruck 和 Cervinia 之间的 BGP 连接上进行宣告。该配置中比较有意思的地方就是访问列表的第三行，图 3-15 指示子网 172.29.2.0/24 位于 AS 50 内，但是访问列表过滤的是网络 172.29.0.0，这是因为有类协议 RIP 没有将该子网再分发给 Meribel 处的 BGP，而且将它归纳成了主网络地址。子网 172.29.1.0/24 在 AS 30 内；IS-IS 是无类的，因此它会将子网再分发给 Zermatt 和 Moritz 的

BGP。Innsbruck 的访问列表对这个子网的公布不起作用，因为第三行要与主网络地址完全匹配。

最后，例 3-79 给出了 AS 30 的一些地址，例如 192.168.100.0，通过 Meribel 将它公布给 Innsbruck。此时这个问题与前面出现的一样——Cervinia 将该路由公布给 Meribel，Meribel 再将它公布给 Innsbruck。例 3-82 给出了 Innsbruck 阻止除了 AS 50 内部路由之外的所有来自 Meribel 的入站路由的配置。

例 3-82　配置 Innsbruck 阻止除了 AS 50 的内部路由之外的来自 Meribel 的入站路由

```
router bgp 100
 neighbor 10.50.250.1 remote-as 50
 neighbor 10.50.250.1 ebgp-multihop 2
 neighbor 10.50.250.1 update-source Loopback0
 neighbor 10.50.250.1 distribute-list 2 in
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 neighbor 10.200.60.1 distribute-list 1 out
 neighbor 172.30.255.254 remote-as 30
 neighbor 172.30.255.254 ebgp-multihop 2
 neighbor 172.30.255.254 update-source Loopback0
 no auto-summary
!
access-list 1 deny   10.20.0.0
access-list 1 deny   192.168.50.0
access-list 1 deny   172.29.0.0
access-list 1 deny   172.17.0.0
access-list 1 permit any
access-list 2 permit 10.20.0.0
access-list 2 permit 192.168.50.0
access-list 2 permit 172.29.0.0
access-list 2 permit 172.17.0.0
```

例 3-83 给出了 Innsbruck 相应的 BGP 表。与例 3-79 中的 BGP 表相比较，很容易就可以看出例 3-83 中的 BGP 表较小而且更容易理解。

例 3-83　加入了路由过滤器以后 Innsbruck 的 BGP 表

```
Innsbruck#show ip bgp
BGP table version is 12, local router ID is 10.100.83.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.20.0.0/16     10.50.250.1              0             0 50 ?
*  10.30.0.0/16     10.200.60.1                            0 200 30 ?
*>                  172.30.255.254          20             0 30 ?
*> 10.50.250.1/32   10.50.250.1              0             0 50 ?
*  172.16.0.0       10.200.60.1                            0 200 30 ?
*>                  172.30.255.254          20             0 30 ?
*> 172.17.0.0       10.50.250.1              1             0 50 ?
*> 172.29.0.0       10.50.250.1              1             0 50 ?
*  172.29.1.0/24    10.200.60.1                            0 200 30 ?
*>                  172.30.255.254          20             0 30 ?
*  172.31.0.0       10.200.60.1                            0 200 30 ?
*>                  172.30.255.254          20             0 30 ?
*> 192.168.2.0/30   10.200.60.1                            0 200 30 ?
*> 192.168.50.0     10.50.250.1              1             0 50 ?
*  192.168.100.0    10.200.60.1                            0 200 30 ?
*>                  172.30.255.254          20             0 30 ?
Innsbruck#
```

3.2.3　案例研究：通过 AS_PATH 过滤路由

在面对大量的宣告的地址的时候，用 NLRI 来过滤路由很快就变得很不轻松或者说是完全不切实际。在图 3-15 中只有几个地址需要公布，但是前一节中给出的访问列表已经显得有些冗长了。

前一节的例子中一个共同的特点就是对于每一个例子，访问列表都是标识一个 AS 内的所有地址。这种情况下，在一个访问列表中，用 AS 号来过滤路由而不是采用列举每一个内部地址的方法会容易些。**Ip as-path access-list** 命令定义了访问列表的一个变量，该变量用来定义 AS 号。正如 **neighbor distribute-list** 可以引出了定义 NLRI 的访问列表，**neighbor filter-list** 命令可以引出 AS_PATH 访问列表。

AS_PATH 访问列表使用了一个名为*正则表达式(regular expressions)*的功能强大的文本分析工具，或者简写是 regex。正则表达式通常用于一些程序语言当中，例如在搜索引擎里的 Perl、Expect、awk 和 Tcl 以及 UNIX 的工具中例如 egrep。正则表达式使用字符串，它们可能是*特殊字符(metacharacters)*也可能是*文字(literals)*，来与文本的内容相匹配。在 AS_PATH 访问列表中，它们与 BGP 更新消息中的 AS_PATH 属性相匹配。

文字是用来描述匹配内容的文字，而特殊字符用来描述如何匹配。例如，regex ^[4-7]与任何以 4~7 之间的数字开始的文本字符串相匹配。在这个表达式中，4 和 7 是文字。在该表达式中^、[]、-是特殊字符。加字符号^指示一行的开始："字符串将出现在后面。"方括号指示一组字符，也就是我们所熟知的*字符类(character class)*："任何字符在方括号中。"连字符指示范围："从第一个字符到最后一个字符之间序列的任何一个。"表 3-1 归纳了最常见的特殊字符，在附录 B，"正则表达式指南"中有一个对如何使用正则表达式的一个简短的指导。如果你没有用过正则表达式，在继续本节之前请先阅读附录 B。

表 3-1　　与 AS_PATH 访问列表有关的正则表达式特殊字符

特 殊 字 符	匹配的内容
.	任何一个单一字符，包括白空格
[]	在方括号之间列出的任何一个字符
[^]	任何一个没有在方括号中列出的字符。(加字符号放在文字序列的前面)
-	(连字号)被连字符号分开的两个文字之间的任何一个字符
?	零或者一个字符或者类型的实例
*	零或者多个字符或者类型的实例
+	字符或者类型的一个或者多个实例
^	开始一行
$	结束一行
|	被该特殊字符分开两个文字中的任何一个
_	(下划线)一个逗号、一行的开始、一行的结束或者一个空格

其他的特殊字符以及表中给出的一些字符，可以与更多的内容相匹配。根据简单性的要求，在这个列表中只列出了与 AS_PATH 访问列表有关的特殊字符以及它们所匹配的内容。

在前一节中，为了阻止AS 100或者AS 200试图将AS 30当作转接AS，配置路由器Zermatt和 Moritz，使它们只公布到 AS 30 内部地址的路由，而将其他的路由过滤掉。为了执行过滤功能，AS 30 的所有地址都单独地列在访问列表中。例 3-84 给出了 Zermatt 的配置，它通过 AS_PATH 访问列表获得了与前一个案例研究相同的结果。

例 3-84　使用 AS_PATH 访问列表配置 Zermatt，从而使它只公布到 AS 30 内部的地址的路由

```
router bgp 30
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 filter-list 1 out
 no auto-summary
!
ip as-path access-list 1 permit ^$
```

Moritz 的配置中有一个相同的 AS_PATH 访问列表。此时正则表达式使用两个特殊字符——第一个与行的开始相匹配，第二个与行的结尾相匹配，这里没有包括文字。与 AS_PATH 相匹配的 regex 没有包括 AS 号。在例 3-85 中，Zermatt 的 BGP 表里只有 AS_PATH 为空的路由是 AS 30 内部。它们与 AS_PATH 列表的声明相匹配并且被允许。和其他的访问列表相似，AS_PATH 列表在最后有一个隐含的“拒绝任何一个”；例 3-85 中的其他所有的路由都与这个隐含拒绝相匹配并且不被公布。

同样在前一个案例研究中，配置路由器 Innsbruck 和 Cervinia，，只接受来自 Meribel 的内部的路由。除此以外，这两个路由器不能互相宣告从 Meribel 学习到的路由。例 3-86 给出了 Innsbruck 的配置，它通过访问列表实现了相同目的。

例 3-85　Zermatt 的 BGP 表中 AS_PATH 为空的路由是那些到 AS 30 内部地址的路由

```
Zermatt#show ip bgp
BGP table version is 109, local router ID is 172.30.255.254
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.20.0.0/16     10.100.83.1                          0 100 50 ?
*> 10.30.0.0/16     192.168.2.1             20         32768 ?
*> 10.50.250.1/32   10.100.83.1                          0 100 50 ?
*> 172.16.0.0       192.168.2.1             20         32768 ?
*> 172.17.0.0       10.100.83.1                          0 100 50 ?
*> 172.29.0.0       10.100.83.1                          0 100 50 ?
*> 172.29.1.0/24    192.168.2.1             20         32768 ?
*> 172.30.255.150/32 192.168.2.1            30         32768 ?
*> 172.31.0.0       192.168.2.1             20         32768 ?
*> 192.168.2.4/30   192.168.2.1             20         32768 ?
*> 192.168.50.0     10.100.83.1                          0 100 50 ?
*> 192.168.100.0    192.168.2.1             20         32768 ?
Zermatt#
```

例 3-86　配置 Innsbruck，使用 AS_PATH 访问列表接受来自 Meribel 的属于 AS 50 内部路由的路由

```
router bgp 100
 neighbor 10.50.250.1 remote-as 50
 neighbor 10.50.250.1 ebgp-multihop 2
 neighbor 10.50.250.1 update-source Loopback0
 neighbor 10.50.250.1 filter-list 2 in
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 neighbor 10.200.60.1 filter-list 1 out
 neighbor 172.30.255.254 remote-as 30
 neighbor 172.30.255.254 ebgp-multihop 2
 neighbor 172.30.255.254 update-source Loopback0
 no auto-summary
!
ip as-path access-list 1 deny _50_
ip as-path access-list 1 permit .*
ip as-path access-list 2 permit ^50$
```

列表 1 应用于到发往 Cervinia 的更新消息。该列表的第一个声明是拒绝任何在 AS_PATH 中包括 AS 50 的更新消息。50 前和后的特殊字符保证只有 50 这个数字与该声明相匹配。如果省略了特殊字符，该声明不仅与 50 相匹配，同时还和 500、5000、350 等相匹配。正则表达式的第二个声明说，“与任何字符相匹配并且与 0 或者该字符的多个具体值相匹配。”换句话说，与任何字符相匹配。这是 AS_PATH 访问列表的“接受任何一个”的版本。这两行命令的结果是 Innsbruck 没有向 Cervinia 公布任何从 Meribel 学习到的路由，但是它向 Cervinia 公布了所有其他的路由。

列表 2 应用于来自 Meribel 的入站更新消息。其中一行的 regex 说明，“开始一行，然后是 50，最后是一行的结束。”换句话说，与包括 AS 50 的 AS_PATH 相匹配而不与其他的相匹配。这些路由是被允许的。Meribel 宣告的从任何其他 AS 学习到的路由在其 AS_PATH 中都有除了 AS 50 之外的更多的 AS 号。因为在该列表的最后有隐含的“拒绝”因此这些路由将被拒绝。

对于图 3-15 中的拓扑，Cervinia 的 AS_PATH 访问列表与 Innsbruck 的访问列表相同。但是，在图 3-16 中，这种拓扑已经被修改了。此时，在该拓扑中加入了 AS 125 并且作为 AS 200 和 AS 50 的转接 AS。如果 Meribel 没有了到 NAP 的链路，流入以及流出 AS 50 的业务量要穿越 AS 125 和 AS 200。根据前一个例子中给出的 AS_PATH 访问列表，列表 1 会拒绝任何一条在 AS_PATH 中包含 AS 50 的路由。这些路由包括从路由器 Oberstdorf 学习到的到 AS 50 的路由，这些路由具有一个(125,50)的 AS_PATH。结果是，Innsbruck 不能从 Cervinia 学习到这些路由并且利用到 AS 50 备用路由的优势。

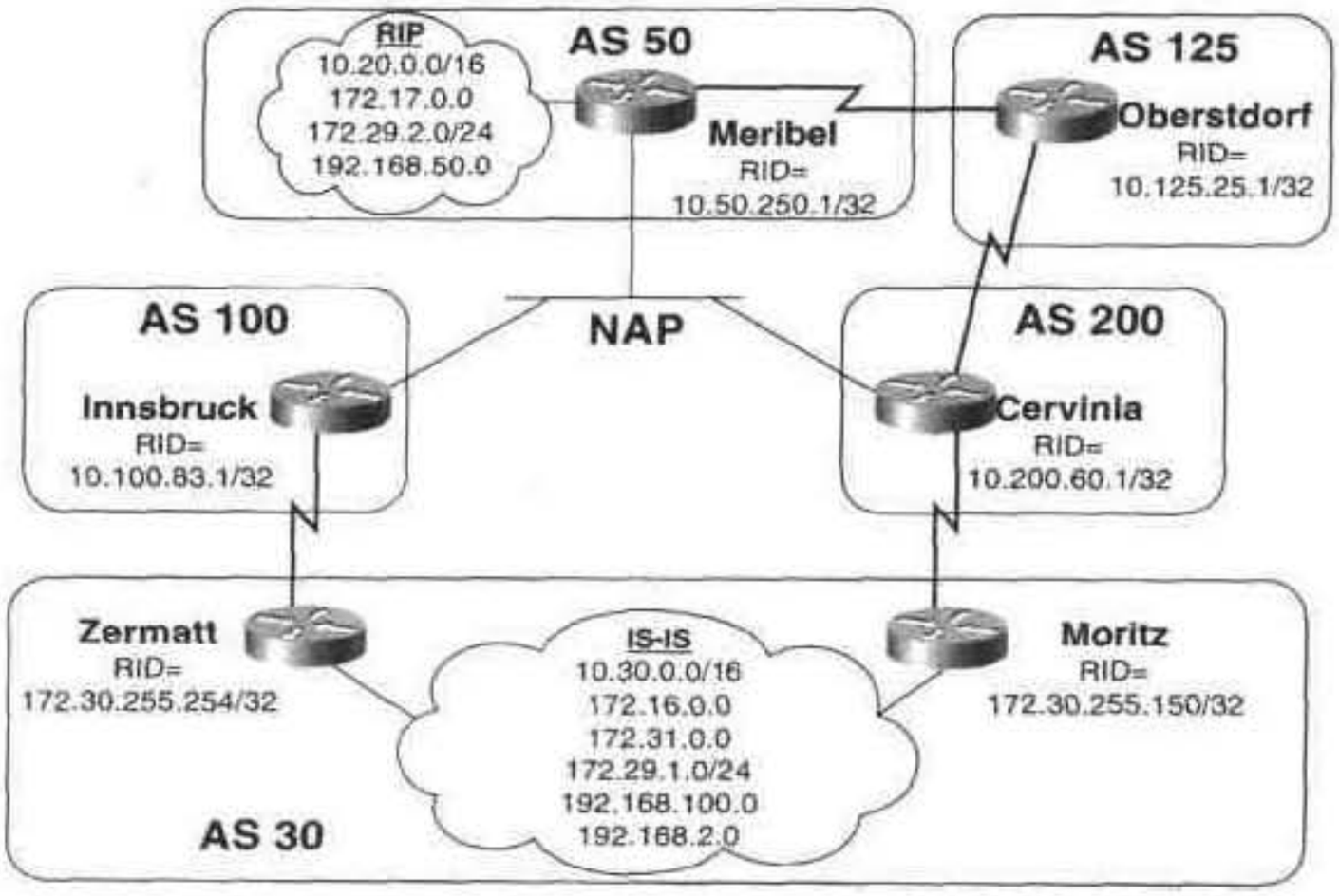


图 3-16　AS 125 提供了一个到 AS 50 的可替换路由

为了弥补这种情况的不足，例 3-87 给出了 Cervinia 的配置。

例 3-87　配置 Cervinia，使 AS 125 作为到 AS 50 的一个转接 AS

```
router bgp 200
 neighbor 10.50.250.1 remote-as 50
 neighbor 10.50.250.1 ebgp-multihop 2
 neighbor 10.50.250.1 update-source Loopback0
 neighbor 10.50.250.1 filter-list 2 in
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 filter-list 1 out
 neighbor 10.125.25.1 remote-as 125
 neighbor 10.125.25.1 ebgp-multihop 2
 neighbor 10.125.25.1 update-source Loopback0
 neighbor 172.30.255.150 remote-as 30
 neighbor 172.30.255.150 ebgp-multihop 2
 neighbor 172.30.255.150 update-source Loopback0
 no auto-summary
!
ip as-path access-list 1 deny ^50$
ip as-path access-list 1 permit .*
ip as-path access-list 2 permit ^50$
```

例 3-87 和前一个例子相似，列表 1 的第一个声明拒绝只包括 AS 50 的 AS_PATH，而不是包括 AS 50 的所有的 AS_PATH。Cervinia 从 Meribel 直接学习到的路由被拒绝，同时从 Oberstdorf 学习到的到 AS 50 的路由是被允许的。

3.2.4　案例研究：通过路由图过滤路由

还可以通过路由图来执行路由过滤。路由图可以使用访问列表，通过 NLRI 来过滤路由，也可以使用 AS_PATH 访问列表，通过 AS_PATH 属性来过滤路由。

例 3-88 显示了图 3-15 中 Zermatt 的可能配置。

例 3-88　配置 Zermatt，让它通过路由图来过滤路由

```
router bgp 30
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 route-map Innsbruck out
 no auto-summary
!
access-list 1 permit 192.168.100.0
access-list 1 permit 10.30.0.0
access-list 1 permit 192.168.2.0
access-list 1 permit 172.29.1.0
access-list 1 permit 172.31.0.0
access-list 1 permit 172.16.0.0
!
route-map Innsbruck permit 10
 match ip address 1
```

此处的访问列表 1 和案例研究“通过 NLRI 过滤路由”中的访问列表相同，在案例研究“通过 NLRI 过滤路由”中是通过 **neighbor distribute-list** 命令来引出访问列表；而在这个例子中，**neighbor route-map** 命令指示的是名为 Innsbruck 的路由图的出站路由，而它使用 **match ip address** 命令引出访问列表。路由图允许任何被访问列表允许的路由而拒绝其他的路由。路由图也可以称做 AS_PATH 访问列表，如例 3-89 中 Zermatt 的配置。

在例 3-89 中 Zermatt 的配置与例 3-88 中的 Zermatt 的配置相同，但是在例 3-89 中使用的是 **match as-path** 命令而例 3-88 中使用的是 **match ip address** 命令。该例中的 AS_PATH 访问列表和案例研究“通过 AS-PATH 过滤路由”中的访问列表相同。AS_PATH 属性为空的路由——由 AS 30 发起的路由——是被允许的，其他的路由都将被拒绝。

例 3-89　用路由图来配置 Zermatt 从而引出一个 AS_PATH 访问列表

```
router bgp 30
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 route-map Innsbruck out
 no auto-summary
!
ip as-path access-list 1 permit ^$
!
route-map Innsbruck permit 10
 match as-path 1
```

路由图可以过滤入站以及出站的 BGP 更新消息。在前一个案例研究中，路由器 Innsbruck 对来自 Merible 的路由进行过滤并接受 AS_PATH 属性为 50，而且不包含其他 AS 号的路由。该路由器还对到 Cernivia 的出站路由进行过滤并且允许 AS_PATH 属性为 30 的路由出去。例 3-90 给出了 Innsbruck 的配置，因为相同的原因而使用路由图。

例 3-90　配置 Innsbruck，从而通过路由图来过滤路由

```
router bgp 100
 neighbor 10.50.250.1 remote-as 50
 neighbor 10.50.250.1 ebgp-multihop 2
 neighbor 10.50.250.1 update-source Loopback0
 neighbor 10.50.250.1 route-map Meribel in
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 neighbor 10.200.60.1 route-map Cervinia-to-Meribel out
 neighbor 172.30.255.254 remote-as 30
 neighbor 172.30.255.254 ebgp-multihop 2
 neighbor 172.30.255.254 update-source Loopback0
 no auto-summary
!
ip as-path access-list 1 deny _50_
ip as-path access-list 1 permit .*
ip as-path access-list 2 permit ^50$
!
route-map Meribel permit 10
 match as-path 2
!
route-map Cervinia-to-Meribel permit 10
 match as-path 1
```

当必须要在一个路由器上配置多个路由过滤器时，用路由图比使用分发列表或者过滤列表更有用。因为路由图使用名字而不是使用数字，一个直观的名字使得这样的一个配置比较容易解释。在例 3-88 和例 3-89 中，将路由图的命名为 Innsbruck 可以十分清楚地指明与该路由图有关系的邻居。在例 3-90 中，名字明确了通过邻居的路由。

但是，使用路由图的主要原因，是因为它们不仅能够通过 **match** 命令来明确特殊的路由，而且还可以通过 **set** 命令来更改它们的属性。下面的五个例子说明了使用路由图来执行更复杂路由策略的情况。这五个例子分别说明了影响路由优选项的方式：

- 在一个路由器内(多个 BGP 路由到同一个目的地)
- 在一个路由器内(来自不同路由协议，到同一个目的地的路由)
- 在本地 AS 内
- 在相邻 AS 内
- 在超出相邻 AS 以外的 AS 内

3.2.5　案例研究：管理权值

经常会出现这种情况，也就是一个路由器到同一个目的地有多条路由。虽然 BGP 由缺省的方式来选择路由，但是偶尔还是需要忽略这些缺省设置，通过执行路由策略来选择路由。虽然 RFC 没有提供在一个路由器上改变路由优选项的方法，但是 Cisco IOS 软件提供了一些方法。

Cisco 所提供的特有工具的第一个就是*管理权值*。每条路由都被分配给了一个权值，权值的范围是 0~65535。如果到一个目的地有多条路由，路由器会优先选择具有最高权值的路由。在缺省的条件下，分配给由路由器发起的 BGP 路由的权值是 32768，分配给从邻居学习到的路由的权值是 0。

如果有多条路由到达同一个目的地，在 BGP 决定进程中，管理权值会忽略所有其他决定路由的因素。但是管理权值对于路由器来讲还是具有本地性。也就是说，不会将它公布给任何邻居的运行 BGP 的路由器。因此，一个路由器上权值的分配不会影响其他路由器上路由的优选项。

在图 3-17 中，AS 30 的连接性有了改进。Zermatt 和 Moritz 都是多宿主到 AS 100 以及 AS 200 从而增加了冗余性。每个路由器都从 Innsbruck 和 Cervinia 接收 AS 50 的路由。回忆一下在第 2 章讨论的 BGP 决定进程，在到同一个目的地的多条路由中选取一条优选路由时，如果其他所有的属性都相同，BGP 会选择来自路由器 ID 为最低的邻居的路由。这意味着图 3-17 中的 Zermatt 和 Moritz 都会通过 Innsbruck 到达 AS 50 内的目的地，因为它的路由器 ID 比 Cervinia 低，如例 3-91 中的输出所示。Zermatt 和 Moritz 显示出 AS 50 内的目的地通过 Innsbruck(10.100.83.1)和 Cervinia(10.200.60.1) 可以到达，因为 Innsbruck 的路由器 ID 值较低，因此两个路由器都将来自 Innsbruck 的路由标记为最佳路由。

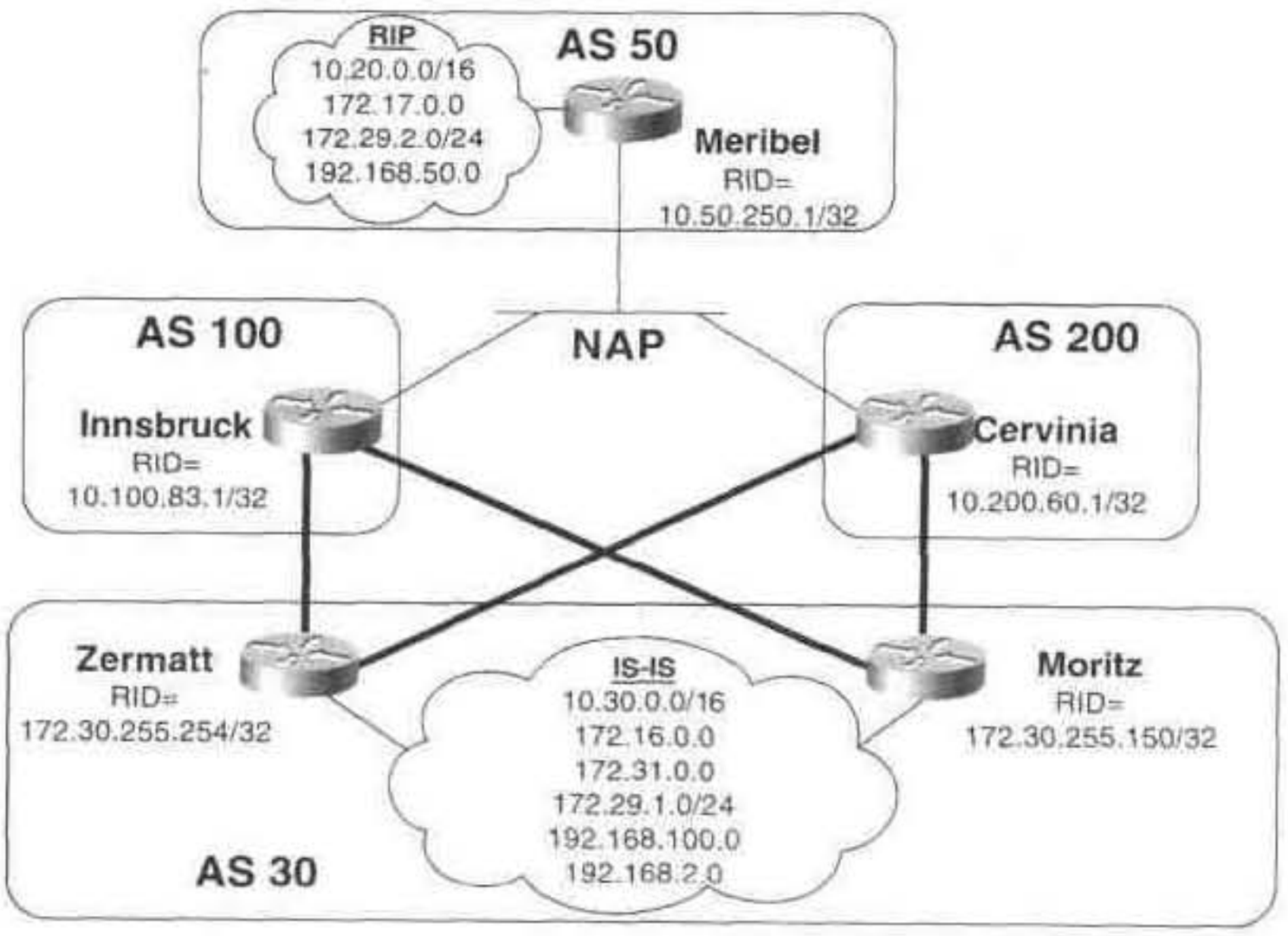


图 3-17　为了备份，Zermatt 的 Moritz 都是多宿主到 AS 100 和 AS 200

为了更公平地分配流量，Zermatt 应使用到 Innsbruck 的链路到达 AS 50，用到 Cervinia 的链路作为备份。Moritz 应使用到 Cervinia 的链路而用到 Innsbruck 的链路备份。两个路由器通过例 3-92 中所示的配置来控制路由权值，实现这个策略。

例 3-92 的配置是通过 **neighbor weight** 命令来为不同邻居公布的路由分配权值。Zermatt 为从 Innsbruck(10.100.83.1)学习到的路由分配的权值为 50000，为从 Cervinia(10.200.60.1)学习到的路由分配的权值为 20000。Moritz 为路由分配的权值与 Zermatt 为路由分配的权值正好相反。结果是，两个路由器都优选具有较高权值的路由，当优选路由失效了，则使用可替换路由，如例 3-93 的输出所示 Zermatt 和 Moritz 将具有最高权值的路由指定为“最佳”路由：本地发起路由的权值还是缺省值 32768。

例 3-91　图 3-17 中 Zermatt 和 Moritz 的 BGP 表

```
Zermatt# show ip bgp
BGP table version is 34, local router ID is 172.30.255.254
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  10.20.0.0/16     10.200.60.1                            0 200 50 ?
*>                  10.100.83.1                            0 100 50 ?
*> 10.30.0.0/16     192.168.2.1             20         32768 ?
*  10.50.250.1/32   10.200.60.1                            0 200 50 ?
*>                  10.100.83.1                            0 100 50 ?
*> 10.100.83.1/32   10.200.60.1                            0 200 50 ?
*> 10.200.60.1/32   10.100.83.1                            0 100 50 ?
*> 172.16.0.0       192.168.2.1             20         32768 ?
*  172.17.0.0       10.200.60.1                            0 200 50 ?
*>                  10.100.83.1                            0 100 50 ?
*  172.29.0.0       10.200.60.1                            0 200 50 ?
*>                  10.100.83.1                            0 100 50 ?
*> 172.29.1.0/24    192.168.2.1             20         32768 ?
*> 172.30.255.150/32 192.168.2.1            30         32768 ?
*> 172.31.0.0       192.168.2.1             20         32768 ?
*> 192.168.2.4/30   192.168.2.1             20         32768 ?
*  192.168.50.0     10.200.60.1                            0 200 50 ?
*>                  10.100.83.1                            0 100 50 ?
*> 192.168.100.0    192.168.2.1             20         32768 ?
Zermatt#
```

```
Moritz#show ip bgp
BGP table version is 33, local router ID is 172.30.255.150
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.20.0.0/16     10.100.83.1                            0 100 50 ?
*                   10.200.60.1                            0 200 50 ?
*> 10.30.0.0/16     192.168.2.5             20         32768 ?
*> 10.50.250.1/32   10.100.83.1                            0 100 50 ?
*                   10.200.60.1                            0 200 50 ?
*> 10.100.83.1/32   10.200.60.1                            0 200 50 ?
*> 10.200.60.1/32   10.100.83.1                            0 100 50 ?
*> 172.16.0.0       192.168.2.5             20         32768 ?
*> 172.17.0.0       10.100.83.1                            0 100 50 ?
*                   10.200.60.1                            0 200 50 ?
*> 172.29.0.0       10.100.83.1                            0 100 50 ?
*                   10.200.60.1                            0 200 50 ?
*> 172.29.1.0/24    192.168.2.5             20         32768 ?
*> 172.30.255.254/32 192.168.2.5            30         32768 ?
*> 172.31.0.0       192.168.2.5             20         32768 ?
*> 192.168.2.0/30   192.168.2.5             20         32768 ?
*> 192.168.50.0     10.100.83.1                            0 100 50 ?
*                   10.200.60.1                            0 200 50 ?
   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.100.0    192.168.2.5             20         32768 ?
Moritz#
```

例 3-92　配置 Zermatt 和 Moritz，通过管理路由权值来遵循路由策略

Zermatt

```
router bgp 30
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 filter-list 1 out
 neighbor 10.100.83.1 weight 50000
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 neighbor 10.200.60.1 filter-list 1 out
 neighbor 10.200.60.1 weight 20000
 no auto-summary
```

Moritz

```
router bgp 30
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 filter-list 1 out
 neighbor 10.100.83.1 weight 20000
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 neighbor 10.200.60.1 filter-list 1 out
 neighbor 10.200.60.1 weight 50000
 no auto-summary
```

如果从一个特殊邻居接收到的所有路由的权值是相同的，那么 **neighbor weight** 命令会很有用的。但是，有些时候，必须给来自同一个邻居的路由分配不同的权值。要求执行这个策略的一个办法就是通过 **neighbor filter-list weight** 命令。与在路由过滤中使用的 **neighbor filter-list** 命令相似，这个命令通过一个访问列表，根据路由 AS_PATH 属性的详细情况来识别他们。但是，虽然一个特定的邻居至多有一个 **neighbor filter-list in** 命令和一个 **neighbor filter-list out** 命令，但是可能会有 **neighbor filter-list weight** 命令的多个实例。你可以为同一个邻居配置 **neighbor filter-list** 和 **neighbor filter-list weight** 命令；虽然这两个命令看起来很相似，但是它们的目的和效果完全不同。

图 3-18 描述了与图 3-17 相同的拓扑，但是另一个 AS 连接到 NAP 上。Innsbruck 与 Cervinia 把 AS 50 与 AS 75 中的路由同时宣告到 Zermatt 与 Moritz。有一条新的路由策略，Moritz 用 Cervinia 到达 AS 75 中的目的，用 Innsbruck 到达 AS 50 中的目的地。

例 3-94 给出了 Moritz 使用 **neighbor filter-list weight** 命令后的配置。

列表 2 中的正则表达式_75$指明 AS_PATH 以 75 结束并且与路径(100,75)和(200,75)相匹配。同样地，列表 3 与以 50 结束的路径相匹配。在 Innsbruck(10.100.83.1)的邻居的配置中，分配给到 AS 75 的路由的权值为 40000，分配到 AS 50 的路由的权值为 60000。在 Cervinia(10.200.60.1)的邻居的配置中，对路由权值的配置刚好相反，到 AS 75 的路由的权值为 60000，到 AS 50 的路由的权值为 40000。例 3-95 给出了相应的 Moritz 的 BGP 表。

例 3-93　配置后 Zermatt 与 Moritz 的 BGP 路由表

```
Zermatt#show ip bgp
BGP table version is 104, local router ID is 172.30.255.254
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.20.0.0/16     10.100.83.1                           50000 100 50 ?
*                   10.200.60.1                           20000 200 50 ?
*> 10.30.0.0/16     192.168.2.1              20           32768 ?
*> 10.50.250.1/32   10.100.83.1                           50000 100 50 ?
*                   10.200.60.1                           20000 200 50 ?
*> 10.100.83.1/32   10.200.60.1                           20000 200 50 ?
*> 10.200.60.1/32   10.100.83.1                           50000 100 50 ?
*> 172.16.0.0       192.168.2.1              20           32768 ?
*> 172.17.0.0       10.100.83.1                           50000 100 50 ?
*                   10.200.60.1                           20000 200 50 ?
*> 172.29.0.0       10.100.83.1                           50000 100 50 ?
*                   10.200.60.1                           20000 200 50 ?
*> 172.29.1.0/24    192.168.2.1              20           32768 ?
*> 172.30.255.150/32 192.168.2.1             30           32768 ?
*> 172.31.0.0       192.168.2.1              20           32768 ?
*> 192.168.2.4/30   192.168.2.1              20           32768 ?
*> 192.168.50.0     10.100.83.1                           50000 100 50 ?
*                   10.200.60.1                           20000 200 50 ?
*> 192.168.100.0    192.168.2.1              20           32768 ?
Zermatt#
```

```
Moritz#show ip bgp
BGP table version is 20, local router ID is 172.30.255.150
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  10.20.0.0/16     10.100.83.1                           20000 100 50 ?
*>                  10.200.60.1                           50000 200 50 ?
*> 10.30.0.0/16     192.168.2.5              20           32768 ?
*  10.50.250.1/32   10.100.83.1                           20000 100 50 ?
*>                  10.200.60.1                           50000 200 50 ?
*> 10.100.83.1/32   10.200.60.1                           50000 200 50 ?
*> 10.200.60.1/32   10.100.83.1                           20000 100 50 ?
*> 172.16.0.0       192.168.2.5              20           32768 ?
*  172.17.0.0       10.100.83.1                           20000 100 50 ?
*>                  10.200.60.1                           50000 200 50 ?
*  172.29.0.0       10.100.83.1                           20000 100 50 ?
*>                  10.200.60.1                           50000 200 50 ?
*> 172.29.1.0/24    192.168.2.5              20           32768 ?
*> 172.30.255.254/32 192.168.2.5             30           32768 ?
*> 172.31.0.0       192.168.2.5              20           32768 ?
*> 192.168.2.0/30   192.168.2.5              20           32768 ?
*  192.168.50.0     10.100.83.1                           20000 100 50 ?
*>                  10.200.60.1                           50000 200 50 ?
*> 192.168.100.0    192.168.2.5              20           32768 ?
Moritz#
```

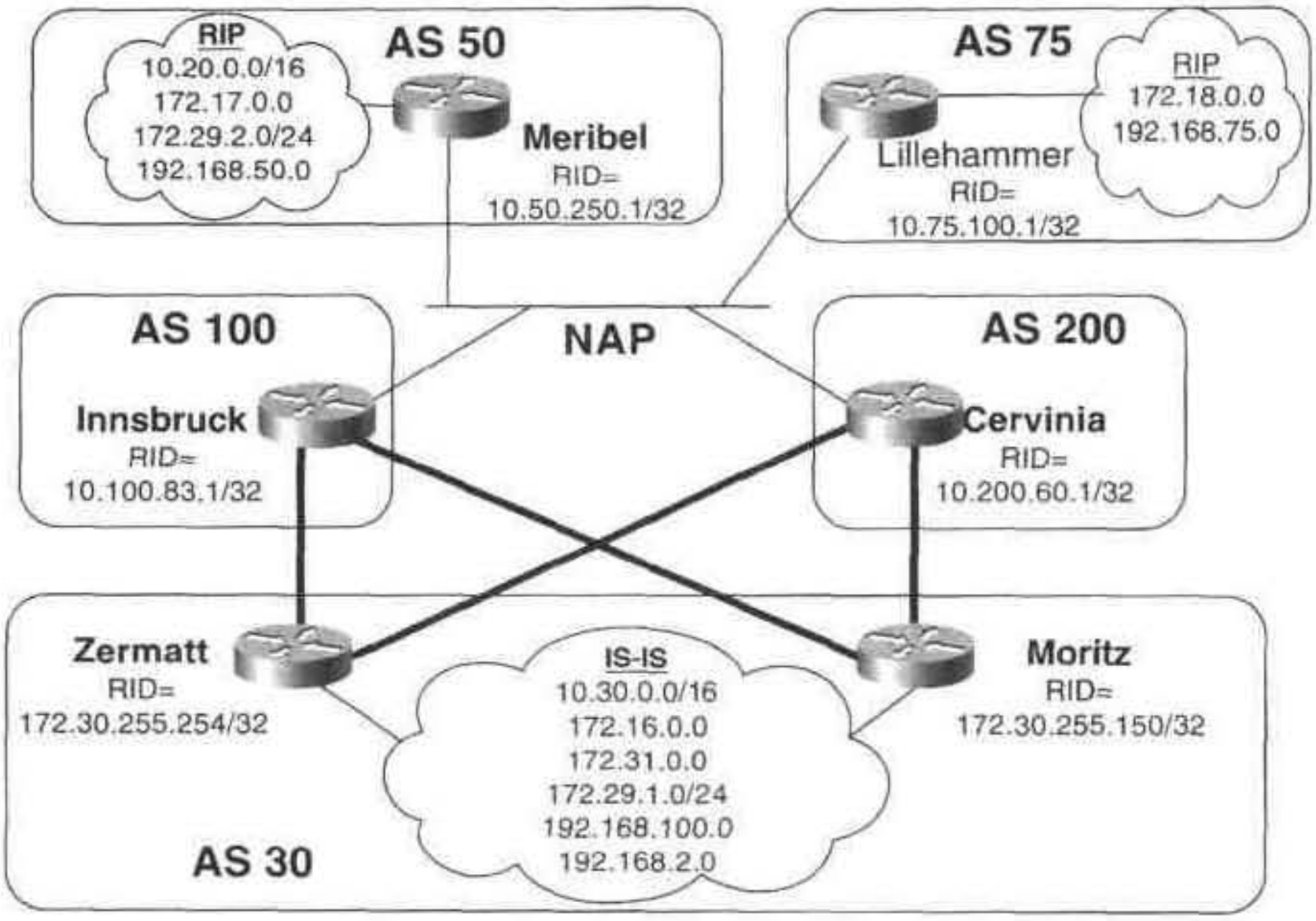


图 3-18　AS 75 与 NAP 相连

例 3-94　配置 Moritz，使它优选 Cervinia 作为到 AS 75 的下一跳路由器

```
router bgp 30
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 filter-list 1 out
 neighbor 10.100.83.1 filter-list 2 weight 40000
 neighbor 10.100.83.1 filter-list 3 weight 60000
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 neighbor 10.200.60.1 filter-list 1 out
 neighbor 10.200.60.1 filter-list 2 weight 60000
 neighbor 10.200.60.1 filter-list 3 weight 40000
 no auto-summary
!
ip as-path access-list 1 permit ^$
ip as-path access-list 2 permit _75$
ip as-path access-list 3 permit _50$
```

如果在一个邻居的配置中，同时使用了 **neighbor weight** 命令和 **neighbor filter-list weight** 命令，那么 **neighbor filter-list weight** 命令占有优先地位。任何来自对等的权值不是通过 **neighbor filter-list weight** 命令来设置的路由，会通过 **neighbor weight** 命令来设置他们的权值。

还可以通过 **neighbor route-map** 命令来管理权值。例 3-96 给出了 Moritz 使用路由图的

配置，该配置得到了与例 3-94 中配置一样的结果。

例 3-95　Cervinia 是到 AS 75 的优选下一跳路由器，而 Innsbruck 是到 AS 50 的优选下一跳路由器

```
Moritz#show ip bgp
BGP table version is 19, local router ID is 172.30.255.150
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.20.0.0/16     10.100.83.1                           60000 100 50 ?
*                   10.200.60.1                           40000 200 50 ?
*> 10.30.0.0/16     192.168.2.5              20           32768 ?
*> 10.50.250.1/32   10.100.83.1                           60000 100 50 ?
*                   10.200.60.1                           40000 200 50 ?
*> 10.100.83.1/32   10.200.60.1                           40000 200 50 ?
*> 10.200.60.1/32   10.100.83.1                           60000 100 50 ?
*> 172.16.0.0       192.168.2.5              20           32768 ?
*> 172.17.0.0       10.100.83.1                           60000 100 50 ?
*                   10.200.60.1                           40000 200 50 ?
*   172.18.0.0      10.100.83.1                           40000 100 75 1
*>                  10.200.60.1                           60000 200 75 1
*> 172.29.0.0       10.100.83.1                           60000 100 50 ?
*                   10.200.60.1                           40000 200 50 ?
*> 172.29.1.0/24    192.168.2.5              20           32768 ?
*> 172.30.255.254/32 192.168.2.5             30           32768 ?
*> 172.31.0.0       192.168.2.5              20           32768 ?
*> 192.168.2.0/30   192.168.2.5              20           32768 ?
*> 192.168.50.0     10.100.83.1                           60000 100 50 ?
*                   10.200.60.1                           40000 200 50 ?
*   192.168.75.0    10.100.83.1                           40000 100 75 1
*>                  10.200.60.1                           60000 200 75 1
*> 192.168.100.0    192.168.2.5              20           32768 ?
Moritz#
```

例 3-96　配置 Moritz 使用路由图，从而使它优选 Cervinia 作为到 AS 75 的下一跳路由器

```
router bgp 30
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 route-map Innsbruck in
 neighbor 10.100.83.1 filter-list 1 out
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 neighbor 10.200.60.1 route-map Cervinia in
 neighbor 10.200.60.1 filter-list 1 out
 no auto-summary
!
ip as-path access-list 1 permit ^$
ip as-path access-list 2 permit _75$
ip as-path access-list 3 permit _50$
!
route-map Innsbruck permit 10
 match as-path 2
 set weight 40000
!
route-map Innsbruck permit 20
 match as-path 3
 set weight 60000
!
route-map Cervinia permit 10
 match as-path 2
 set weight 60000
!
route-map Cervinia permit 20
 match as-path 3
 set weight 40000
```

当通过路由图来设置权值时，只有 AS_PATH 会与之匹配；独立的 IP 地址不会与 **match ip address** 命令相匹配。还可以把设置权值的路由图和 **neighbor filiter-list weight** 命令和 **neighbor weight** 命令一起用于同一个邻居配置中，但是设置权值的路由图会优于这两个命令中的任何一个。

3.2.6　案例研究：管理距离以及后门路由

另一个 Cisco 特有的用于在路由器上管理优选项的工具是*管理距离(administrative distance)*。管理权值会影响到从不同 BGP 对端学习到的到同一个目的地的多条路由的优选项，而管理距离会影响到从不同的路由协议学习到的到同一个目的地的多条路由的优选项。这也就是说，在 BGP 表中会看到管理权值的影响，而在 IP 路由表中能够看到管理距离的影响。

通常，是根据路由协议或者学到这条路由的源为一条路由分配管理距离，优先选择距离短的路由。表 3-2 给出了不同协议的缺省管理距离。可以看到在一个 AS 内部，如果路由器从 RIP 或者 OSPF 处学习到了路由，那么 OSPF 路由是优选，因为它的距离是(110)，低于 RIP 路由的距离(120)。

表 3-2　　Cisco 的缺省管理距离

路　由　源	管　理　距　离
连接的接口	0
静态路由	1
EIGRP 归纳路由	5
外部 BGP	20
EIGRP	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EGP	140
外部 EIGRP	170
内部 BGP	200
本地 BGP	200
未知	255

当一条静态路由指示的是一个接口而不是下一跳地址，目的地将被看作是一个直连网络。

EBGP 有一个缺省的距离 20，它低于任何一个 IGP 的距离值。起先，对于如图 3-18 所示的互联网络，这种情况看起来好像是个问题。当 Zermatt 公布一个 AS 30 内部的地址给

Innsbruck，这个地址又传给了 Cervinia，而 Cervinia 又将该地址传回给了 Moritz。Moritz，通过 EBGP 获得该路由，因为 IS-IS 路由的距离为 115，因此在到达同一个目的地，通过 EBGP 学习到的路由以及 IS-IS 路由之间，Moritz 优先选择通过 EBGP 学习到的路由。实际上，因为存在着基本的 BGP 环路避免机制，这种情况是不会发生的。Moritz 在来自 Cervinia 的路由的 AS_PATH 中发现 AS 30 从而将该路由丢弃。

另一方面，IBGP 没有在 AS_PATH 中加入 AS 号，因此从 IGP 学习到的路由传递给 AS 内的 IBGP 对等时，可能会导致路由环路或者黑洞。因为这个原因，IBGP 路由的距离为 200，高于其他任何的 IGP 路由。如果到一个目的地存在着学习到的 IGP 路由以及 IBGP 路由，则 IGP 路由优于 IBGP 路由。

本地 BGP 路由是指使用 BGP **network** 命令以后，由本地路由器发起的路由。和 IBGP 路由一样，它们的缺省距离为 200，因此它们不会优于 IGP 路由。

在第 1 卷的第 13 章，“路由过滤”中包含了一个案例研究，该案例研究说明了如何管理 IGP 路由的缺省距离。通过 **distance bgp** 命令，可以改变 BGP 路由的缺省距离。该命令分别为 EBGP、IBGP 和本地 BGP 路由设置了距离。在例 3-97 中，将 IBGP 的管理距离改为了 95，使得到同一个目的地的所有路由中，IBGP 路由优于除了 EIGRP 路由以外其他所有的 IGP 路由。

例 3-97　将 IBGP 路由的距离改为 95

```
router bgp 30
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 distance bgp 20 95 200
```

与 IGP 不同，在这里，我们很少能够为改变所有路由的缺省 BGP 距离找到一个好的理由。但是，在有一种情况下，应该改变 BGP 路由的缺省距离。在图 3-19 中，在路由器 Meribel 和 Lillehammmer 之间加入了一条专用链路而且在该链路上，路由器使用的 RIP 协议，这条链

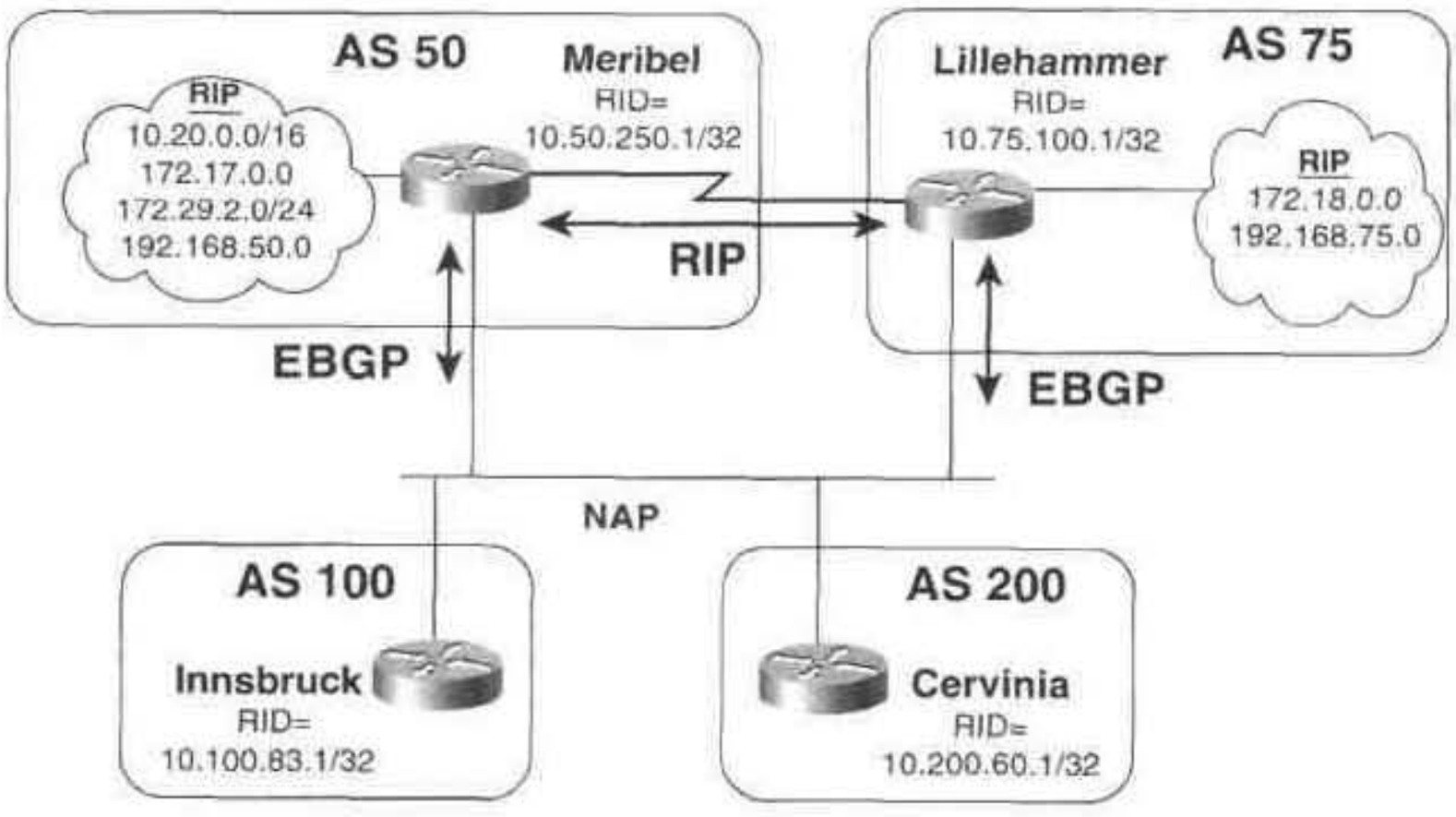


图 3-19　在 AS 50 和 AS 75 之间加入了一条专用的后门链路

路就用作后门路由。也就是说，AS 50 和 AS 75 之间的一些业务量应该在这条专用的后门链路上发送而不是通过公共的 NAP 来传送。也许因为 AS 50 和 AS 75 是生意伙伴关系，并且他们希望他们之间的一些通信能够在他们之间的专用链路上传送而不是通过公共 Internet。

在这个例子中，AS 50 中的 172.17.0.0 和 AS 75 中的 172.18.0.0 之间的业务量应该通过后门链路来传送，只有该专用后门链路出现故障时，才使用 NAP 路由。问题是管理距离，例如，Lillehammer 通过后门链路上的 RIP 以及通过 NAP 链路上的 EBGP 从 Meribel 学习到了到 172.17.0.0 的路由。EBGP 的距离为 20，RIP 的距离为 120，因此 EBGP 路由是优选路由，如例 3-98 中的输出所示。

例 3-98　Lillehammer 通过 EBGP 学习到的路由是优选路由

```
Lillehammer#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
       T - traffic engineered route

Gateway of last resort is not set

C    192.168.75.0/24 is directly connected, Ethernet2
B    172.17.0.0/16 [20/1] via 10.50.250.1, 00:01:24
B    172.16.0.0/16 [20/0] via 10.100.83.1, 00:01:22
C    172.18.0.0/16 is directly connected, Ethernet1
     172.29.0.0/16 is variably subnetted, 2 subnets, 2 masks
B       172.29.1.0/24 [20/0] via 10.100.83.1, 00:01:22
B       172.29.0.0/16 [20/1] via 10.50.250.1, 00:01:24
B    172.31.0.0/16 [20/0] via 10.100.83.1, 00:01:22
     192.168.4.0/29 is subnetted, 1 subnets
C       192.168.4.0 is directly connected, Ethernet0
     10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
B       10.30.0.0/16 [20/0] via 10.100.83.1, 00:01:22
B       10.20.0.0/16 [20/0] via 10.50.250.1, 00:01:24
C       10.21.0.0/16 is directly connected, Serial1.507
C       10.75.100.1/32 is directly connected, Loopback0
S       10.100.83.1/32 is directly connected, Ethernet0
S       10.50.250.1/32 is directly connected, Ethernet0
S       10.200.60.1/32 is directly connected, Ethernet0
B    192.168.50.0/24 [20/1] via 10.50.250.1, 00:01:27
B    192.168.100.0/24 [20/0] via 10.100.83.1, 00:01:25
Lillehammer#
```

一个解决方案就是使用 BGP **network** 命令，如例 3-99 所示。

在例 3-99 的配置中，**network** 命令将通过 EBGP 学习到的路由看作是本地 BGP 路由。例如，通过 EBGP 将网络 172.17.0.0 公布给 Lillehammer，而 Lillehammer 将它加入到路由表中。虽然 172.17.0.0 不是真正的本地路由，但是在 Lillehammer 的 BGP 配置中还是加入了 **network 172.17.0.0** 命令。因为该地址存在于路由表中，**network** 命令就会与之匹配从而使它成为一条本地路由。

例 3-99　使用 network 命令将通过 EBGP 学习到的路由看作是本地 BGP 路由

Lillehammer

```
router rip
 redistribute bgp 75
 network 10.0.0.0
 network 172.18.0.0
 network 192.168.75.0
!
router bgp 75
 network 172.17.0.0
 network 172.18.0.0
 network 192.168.75.0
 neighbor 10.50.250.1 remote-as 50
 neighbor 10.50.250.1 ebgp-multihop 2
 neighbor 10.50.250.1 update-source Loopback0
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
```

Meribel

```
router rip
 redistribute bgp 50 metric 1
 network 10.0.0.0
!
router bgp 50
 network 172.18.0.0
 redistribute rip
 neighbor 10.75.100.1 remote-as 75
 neighbor 10.75.100.1 ebgp-multihop 2
 neighbor 10.75.100.1 update-source Loopback0
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 no auto-summary
```

这个逻辑听起来很奇怪，但是它很起作用。作为一条 EBGP 路由，使用 **network** 命令将 172.17.0.0 改变成本地 BGP 路由。因为现在 Lillehammer 把 172.17.0.0 看作是本地路由，就分配给它一个 200 的管理距离。到 172.17.0.0 的 RIP 路由现在有了一个较低的距离因此它就成了优选路由，如例 3-100 所示，Lillehammer 把到 172.17.0.0 的 EBGP 路由看作是本地 BGP 路由，本地路由的管理权值为 200，因此到该网络的 RIP 路由就成为了优选路由。

虽然正确地利用了管理权值，但是这种配置还是存在着一定的问题。利用 **network** 命令将 EBGP 路由转换成本地路由，现在，本地 BGP 路由器在它的 EBGP 更新消息中公布了该路由。例如，现在，通过 NAP 路由，Lillehammer 在它的 EBGP 更新消息中将 172.17.0.0 公布给它的对端。因为 Meribel 的 BGP 进程通过再分发学习到了到 172.17.0.0 的路由，因此它在公布该路由时携带了 IGP 的 ORIGIN 属性。但是，因为有 **network** 命令，Lillehammer 公布的路由具有 IGP 的 ORIGIN 属性，因此如例 3-101 所示，Cervinia 和 Innsbruck 都选

Lillehammer 作为到 172.17.0.0 的最佳下一跳。到 172.17.0.0 去的外部业务量被转发到 Lillehammer，Lillehammer 再在后门链路上转发这些业务量。只有 172.17.0.0 和 172.18.0.0 之间的业务量才被认为会使用后门链路；所有其他的业务量应该使用 NAP 路由。

例 3-100　RIP 为优选路由

```
Lillehammer#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
       T - traffic engineered route

Gateway of last resort is not set

C    192.168.75.0/24 is directly connected, Ethernet2
R    172.17.0.0/16 [120/2] via 10.21.1.1, 00:00:06, Serial1.507
B    172.16.0.0/16 [20/0] via 10.200.60.1, 00:00:36
C    172.18.0.0/16 is directly connected, Ethernet1
     172.29.0.0/16 is variably subnetted, 2 subnets, 2 masks
B       172.29.1.0/24 [20/0] via 10.200.60.1, 00:00:36
B       172.29.0.0/16 [20/1] via 10.50.250.1, 00:00:24
B    172.31.0.0/16 [20/0] via 10.200.60.1, 00:00:36
     192.168.4.0/29 is subnetted, 1 subnets
C       192.168.4.0 is directly connected, Ethernet0
     10.0.0.0/8 is variably subnetted, 7 subnets, 2 masks
B       10.30.0.0/16 [20/0] via 10.200.60.1, 00:00:36
B       10.20.0.0/16 [20/0] via 10.50.250.1, 00:00:24
C       10.21.0.0/16 is directly connected, Serial1.507
C       10.75.100.1/32 is directly connected, Loopback0
S       10.100.83.1/32 is directly connected, Ethernet0
S       10.50.250.1/32 is directly connected, Ethernet0
S       10.200.60.1/32 is directly connected, Ethernet0
B    192.168.50.0/24 [20/1] via 10.50.250.1, 00:00:25
B    192.168.100.0/24 [20/0] via 10.200.60.1, 00:00:37
Lillehammer#
```

例 3-101　Lillehammer 和 Meribel 之间的后门链路成了到 172.17.0.0 的所有外部业务量的转接网络

```
Cervinia#show ip bgp 172.17.0.0
BGP routing table entry for 172.17.0.0/16, version 474
Paths: (3 available, best #2, advertised over EBGP)
  100 75
    10.100.83.1 from 10.100.83.1
      Origin IGP, localpref 100, valid, external
  75
    10.75.100.1 from 10.75.100.1 (192.168.75.1)
      Origin IGP, metric 2, localpref 100, valid, external, best
  50
    10.50.250.1 from 10.50.250.1
      Origin incomplete, metric 1, localpref 100, valid, external
Cervinia#
```

例 3-102 给出了通过 **network backdoor** 命令解决这个问题的示例，这是另一个 Cisco 特有的工具。

例 3-102　限制 Lillehammer 和 Meribel 之间后门链路上的外部业务量

Lillehammer

```
router rip
 redistribute bgp 75
 network 10.0.0.0
 network 172.18.0.0
 network 192.168.75.0
!
router bgp 75
 network 172.17.0.0 backdoor
 network 172.18.0.0
 network 192.168.75.0
 neighbor 10.50.250.1 remote-as 50
 neighbor 10.50.250.1 ebgp-multihop 2
 neighbor 10.50.250.1 update-source Loopback0
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
```

Meribel

```
router rip
 redistribute bgp 50 metric 1
 network 10.0.0.0
!
router bgp 50
 network 172.18.0.0 backdoor
 redistribute rip
 neighbor 10.75.100.1 remote-as 75
 neighbor 10.75.100.1 ebgp-multihop 2
 neighbor 10.75.100.1 update-source Loopback0
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 no auto-summary
```

network backdoor 命令和 **network** 命令的效果是一样的：将 EBGP 路由看作是本地 BGP 路由并且将它的管理距离改为 200。这两个命令的区别是 **network backdoor** 命令确定的地址不会公布给 EBGP 对端。在网络 172.17.0.0 中，如例 3-100 所示，通过新的配置在 Lillehammer 处得到了相同的路由表。但是 Cervinia 的 BGP 不再包含通过 Lillehammer 到达该网络的路由。

3.2.7　案例研究：使用 LOCAL_PREF 属性

LOCAL_PREF 属性用于在多条到同一个目的地的路由之间设置优选项。与管理权值不同

的是，LOCAL_PREF 属性并不只限于单一的路由器。而且，它是和 IBGP 对端进行通信，该属性不与 EBGP 对端通信——因此它的名字是本地优选项。

一个路由的 LOCAL_PREF 属性可以是 0 和 4294967295 之间的任何一个数；数越大，该路由优选的级别越高。缺省的情况下，所有公布给 IBGP 对端的路由的 LOCAL_PREF 都是 100。可以通过 **ip default local-preference** 命令来改变这个缺省值。可以通过路由图以及 **set local-preference** 命令来改变独立路由的 LOCAL_PREF 属性。

在图 3-20 中，AS 30 多宿主到一个 AS。为了备份的原因，在 Zermatt 和 Moritz 之间以及 Innsbruck 和 Saalbach 之间均加入了链路；在这些链路上运行的是 IBGP。

AS 30 中的路由策略要求所有到 AS 75 的业务量必须使用 Moritz-Saalbach 链路，所有到 AS 50 的业务量必须使用 Zermatt-Innsbruck 链路。在每种情况下，只有当优选路由不可用时，才使用其他的链路。例 3-103 给出了 Zermatt 和 Moritz 的配置。

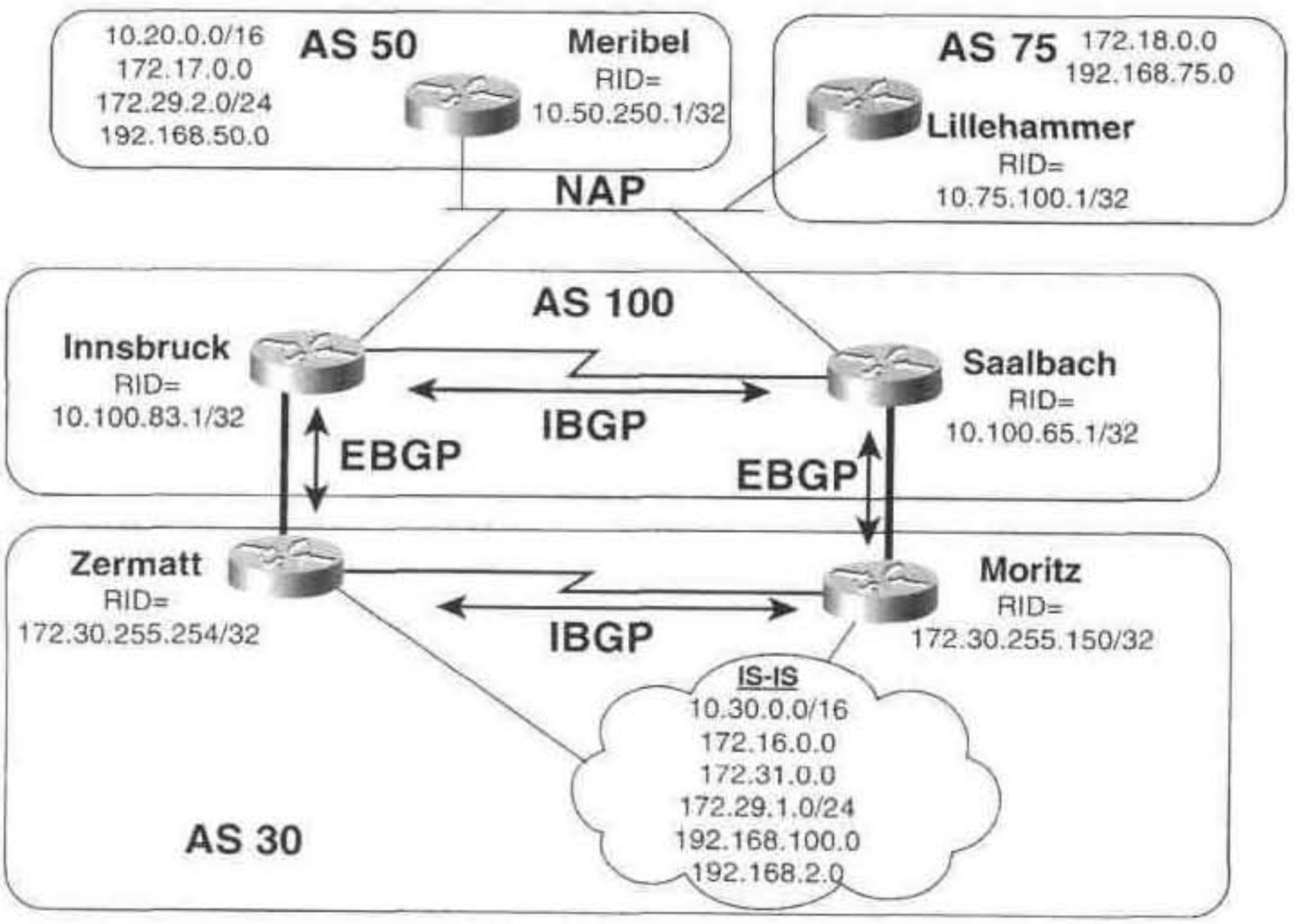


图 3-20　AS 30 多宿主到一个单一的 AS

每个路由器都将来自 EBGP 对等的入站路由链接到一个名为 PREF 的路由图，这个路由图的序列 10 识别入站路由的 AS_PATH。给 Zermatt 处的 AS_PATH 以 50 结束的入站路由分配一个值为 200 的 LOCAL_PREF。给 Moritz 处的 AS_PATH 以 75 结束的入站路由分配一个值为 300 的 LOCAL_PREF。任何与序列 10 不匹配的路由都被序列 20 允许而且分配给它们一个缺省的 LOCAL_PREF，值为 100。

> 注：　实际上，可能会给两个路由器分配相同的 LOCAL_PREF 值，在该例中为两个路由器分配了不同的值只是为较为容易地观察路由图的影响。

例 3-103　通过 LOCAL_PREF 属性来影响路由的优选项

Zermatt

```
router isis
 net 30.5678.1234.defa.00
 default-information originate
!
router bgp 30
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 route-map PREF in
 neighbor 10.100.83.1 filter-list 1 out
 neighbor 172.30.255.150 remote-as 30
 neighbor 172.30.255.150 ebgp-multihop 2
 neighbor 172.30.255.150 update-source Loopback0
 neighbor 172.30.255.150 next-hop-self
 no auto-summary
!
ip route 10.100.83.1 255.255.255.255 Serial1.906
ip route 172.30.255.150 255.255.255.255 Serial1.908
!
ip as-path access-list 1 permit ^$
ip as-path access-list 2 permit _50$
!
route-map PREF permit 10
 match as-path 2
 set local-preference 200
!
route-map PREF permit 20
```

Moritz

```
router isis
 net 30.1234.5678.abcd.00
 default-information originate
!
router bgp 30
 redistribute isis level-2
 neighbor 10.100.65.1 remote-as 100
 neighbor 10.100.65.1 ebgp-multihop 2
 neighbor 10.100.65.1 update-source Loopback0
 neighbor 10.100.65.1 route-map PREF in
 neighbor 10.100.65.1 filter-list 1 out
 neighbor 172.30.255.254 remote-as 30
 neighbor 172.30.255.254 ebgp-multihop 2
 neighbor 172.30.255.254 update-source Loopback0
 neighbor 172.30.255.254 next-hop-self
 no auto-summary
!
ip route 10.100.65.1 255.255.255.255 Serial1.803
ip route 172.30.255.254 255.255.255.255 Serial1.809
!
ip as-path access-list 1 permit ^$
ip as-path access-list 2 permit _75$
!
route-map PREF permit 10
 match as-path 2
 set local-preference 300
!
route-map PREF permit 20
```

例 3-104 给出了两个路由器相应的 BGP 表。

例 3-104　Zermatt 给到 AS 50 内目的地的路由分配了一个值为 200 的 LOCAL_PREF，Moritz 给到 AS 75 内目的地的路由分配了一个值为 300 的 LOCAL_PREF

```
Zermatt#show ip bgp
BGP table version is 20, local router ID is 172.30.255.254
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.20.0.0/16     10.100.83.1                   200      0 100 50 ?
*> 10.30.0.0/16     192.168.2.1             20         32768 ?
* i                 172.30.255.150          20    100      0 ?
*> 10.50.250.1/32   10.100.83.1                   200      0 100 50 ?
*> 10.75.100.1/32   10.100.83.1                   100      0 100 75 ?
*> 10.100.65.1/32   10.100.83.1                   200      0 100 50 ?
*>i10.100.83.1/32   172.30.255.150                100      0 100 50 ?
*> 172.16.0.0/16    192.168.2.1             20         32768 ?
* i                 172.30.255.150          20    100      0 ?
*> 172.17.0.0       10.100.83.1                   200      0 100 50 ?
*  172.18.0.0       10.100.83.1                            0 100 75 i
*>i                 172.30.255.150                300      0 100 75 i
*> 172.29.0.0       10.100.83.1                   200      0 100 50 ?
*> 172.29.1.0/24    192.168.2.1             20         32768 ?
* i                 172.30.255.150          20    100      0 ?
*> 172.31.0.0       192.168.2.1             20         32768 ?
* i                 172.30.255.150          20    100      0 ?
*>i192.168.2.0/30   172.30.255.150          20    100      0 ?
*> 192.168.2.4/30   192.168.2.1             20         32768 ?
*> 192.168.50.0     10.100.83.1                   200      0 100 50 ?
*  192.168.75.0     10.100.83.1                            0 100 75 i
*>i                 172.30.255.150                300      0 100 75 i
*> 192.168.100.0    192.168.2.1             20         32768 ?
* i                 172.30.255.150          20    100      0 ?
Zermatt#
```

```
Moritz#show ip bgp
BGP table version is 25, local router ID is 172.30.255.150
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*  10.20.0.0/16     10.100.65.1                            0 100 50 ?
*>i                 172.30.255.254                200      0 100 50 ?
*> 10.30.0.0/16     192.168.2.5             20         32768 ?
* i                 172.30.255.254          20    100      0 ?
*  10.50.250.1/32   10.100.65.1                            0 100 50 ?
*>i                 172.30.255.254                200      0 100 50 ?
*  10.75.100.1/32   10.100.65.1                            0 100 75 ?
*>i                 172.30.255.254                100      0 100 75 ?
*>i10.100.65.1/32   172.30.255.254                200      0 100 50 ?
*> 10.100.83.1/32   10.100.65.1                            0 100 50 ?
*> 172.16.0.0/16    192.168.2.5             20         32768 ?
* i                 172.30.255.254          20    100      0 ?
*  172.17.0.0       10.100.65.1                            0 100 50 ?
*>i                 172.30.255.254                200      0 100 50 ?
*> 172.18.0.0       10.100.65.1                   300      0 100 75 i
*  172.29.0.0       10.100.65.1                            0 100 50 ?
*>i                 172.30.255.254                200      0 100 50 ?
*> 172.29.1.0/24    192.168.2.5             20         32768 ?
* i                 172.30.255.254          20    100      0 ?
*> 172.31.0.0       192.168.2.5             20         32768 ?
* i                 172.30.255.254          20    100      0 ?
*> 192.168.2.0/30   192.168.2.5             20         32768 ?
*>i192.168.2.4/30   172.30.255.254          20    100      0 ?
*  192.168.50.0     10.100.65.1                            0 100 50 ?
*>i                 172.30.255.254                200      0 100 50 ?
*> 192.168.75.0     10.100.65.1                   300      0 100 75 i
*> 192.168.100.0    192.168.2.5             20         32768 ?
* i                 172.30.255.254          20    100      0 ?
Moritz#
```

注意例 3-103 中的配置与前几个例子中 IS-IS 配置的根本不同之处。在这里，没有将 BGP 路由再分发到 IS-IS 域里，相反的是，每个路由器公布了一个缺省路由。当将业务量向一个外部目的地发送的时候，内部路由器将数据包转发到最近的缺省地址——Zermatt 或者 Innsbruck。根据外部路由的 LOCAL_PREF 属性，这些路由器将数据包转发到它们相应的 EBGP 对等或者通过冗余链路转发到 IBGP 对端。

取消了将 BGP 再分发到 IS-IS 并不是一时的冲动而是这种拓扑以及这种路由策略的需要。例如，如果 Moritz 将到 172.18.0.0 的 EBGP 路由再分发到 IS-IS，该路由会经由 IS-IS 域公布给 Zermatt。Zermatt 接受该路由并将它再分发回 BGP，将它放入自己的 BGP 表。因为 Zermatt 将该路由放入自己的 BGP 表，这条路由就被认为是由本地发起的并且分配给它一个权值 32768。管理权值会无视 LOCAL_PREF 的存在，于是，Zermatt 会把通过 IS-IS 域的路由看作是最佳路由而不是通过到 Moritz 的直连链路。

3.2.8　案例研究：使用 MULTI_EXIT_DISC 属性

MULTI_EXIT_DISC 属性，也称为 MED，用于影响相邻 AS 内的路由决定。MED 有时也被称做*外部度量*，并且实际上，在路由表中它被标为“度量”。和 LOCAL_PREF 一样，MED 是一个 4 字节的数，因此它可以是 0~4294967295 中的任何一个值。

当一个运行 BGP 的路由器从一个对端学习到一条路由，它会将该路由的 MED 传给任何一个 IBGP 对端，但是不传给 EBGP 对端。结果是，MED 只适用于相邻的 AS 之间。如果图 3-20 中的 Zermatt 将具有一定 MED 的 172.16.0.0 公布给 Innsbruck，Innsbruck 能够将 MED 公布给 Saalbach。但是，当 Innsbruck 和 Saalbach 将该路由宣告给它们在 AS 50 和 AS 75 内的 EBGP 对等时，它们不能在路由中包括 MED。

MED 是一个相对较弱的属性，在 BGP 决定过程中，到同一个目的地的多条路由的权值、LOCAL_PREF、AS_PATH 长度以及 ORIGIN 都先于 MED 考虑。但是，如果所有这些变量值都相同，将选择具有最低 MED 的路由。

> 提示：LOCAL_PREF 最高的路由是优选路由，但是 MED 最低的路由是优选路由，这可以让你感觉到有些困惑。MED 的一个另一个术语是*度量(metric)*，而度量的另一个术语是*距离(distance)*。因此只需记住“最高的优选，最短的距离”。

在路由图下，可以通过 **set metric** 命令来管理 MED。在图 3-20 中，AS 30 希望 AS 100 将到网络 172.16.0.0 的入业务量通过 Saalbach-Moritz 链路来传送，而到网络 172.31.0.0 的业务量则通过 Innsbruck-Zermatt 链路进行传送。如果发送其他的业务量，AS 100 可以选择任意一条链路。例 3-105 给出了 Zermatt 和 Moritz 的配置。

每一个路由器将到它 EBGP 对端的输出路由链接到一个名为 MED 的路由图。这个路由图的序列 10 根据访问列表 1 为匹配的路由分配了一个 MED 值。Zermatt 为到 172.31.0.0 的路由分配了一个 MED，值为 100，Moritz 为到 172.16.0.0 的路由分配了一个 MED，值为 100。任何与序列 10 所指示的访问列表 1 不匹配的路由被序列 20 允许并且为它们分配一个值为 200 的 MED。例 3-106 给出了在 Innsbruck 相应的 BGP 表，如例中所示根据在路由中优选度量值较低的路由的原则，Innsbruck 将目的地为 172.16.0.0 的数据包转发给 Saalbach(10.100.65.1)；目的地为 172.31.0.0 的数据包被转发到 Zermatt(172.30.255.254)。

例 3-105　配置 Zermatt 和 Moritz 来管理 MED 属性

Zermatt

```
router bgp 30
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 route-map PREF in
 neighbor 10.100.83.1 route-map MED out
 neighbor 10.100.83.1 filter-list 1 out
 neighbor 172.30.255.150 remote-as 30
 neighbor 172.30.255.150 ebgp-multihop 2
 neighbor 172.30.255.150 update-source Loopback0
 neighbor 172.30.255.150 next-hop-self
!
access-list 1 permit 172.31.0.0
access-list 2 permit any
!
route-map MED permit 10
 match ip address 1
 set metric 100
!
route-map MED permit
 match ip address 2
 set metric 200
```

Moritz

```
router bgp 30
 redistribute isis level-2
 neighbor 10.100.65.1 remote-as 100
 neighbor 10.100.65.1 ebgp-multihop 2
 neighbor 10.100.65.1 update-source Loopback0
 neighbor 10.100.65.1 route-map PREF in
 neighbor 10.100.65.1 route-map MED out
 neighbor 10.100.65.1 filter-list 1 out
 neighbor 172.30.255.254 remote-as 30
 neighbor 172.30.255.254 ebgp-multihop 2
 neighbor 172.30.255.254 update-source Loopback0
 neighbor 172.30.255.254 next-hop-self
 no auto-summary
!
access-list 1 permit 172.16.0.0
access-list 2 permit any
!
route-map MED permit 10
 match ip address 1
 set metric 100
!
route-map MED permit 20
 match ip address 2
 set metric 200
```

例 3-106　Inssbruck 的 BGP 表

```
Innsbruck#show ip bgp 172.16.0.0
BGP routing table entry for 172.16.0.0/16, version 10
Paths: (2 available, best #2)
  Advertised to non peer-group peers:
    10.50.250.1 10.75.100.1
  30
    172.30.255.254 from 172.30.255.254 (172.30.255.254)
      Origin incomplete, metric 200, localpref 100, valid, external, ref 2
  30
    10.100.65.1 from 10.100.65.1 (10.100.65.1)
      Origin incomplete, metric 100, localpref 100, valid, internal, best, ref 2

Innsbruck#show ip bgp 172.31.0.0
BGP routing table entry for 172.31.0.0/16, version 26
Paths: (2 available, best #1)
  Advertised to non peer-group peers:
    10.50.250.1 10.75.100.1
  30
    172.30.255.254 from 172.30.255.254 (172.30.255.254)
      Origin incomplete, metric 100, localpref 100, valid, external, best, ref 2
  30
    10.100.65.1 from 10.100.65.1 (10.100.65.1)
      Origin incomplete, metric 200, localpref 100, valid, internal, ref 2
Innsbruck#
```

通常，只有到同一个目的地的多条路由是由同一个 AS 发起时，才会比较这些路由的 MED，毕竟，当在一个 AS 内存在着多条到相邻 AS 的链路时， MED 的目的是允许该 AS 为入业务量而交流它的优先权。通常，比较两个不同 AS 之间的优选项是没有什么意义的，但是，偶尔会有异常的应用。

图 3-21 再次给出了 AS 50 和 AS 75 之间的一条后门链路。和案例研究“管理距离以及后门路由”中，网络 172.17.0.0 和 172.17.0.0 是通过 RIP 在后门链路上进行宣告的而且它们用于在两个 AS 之间进行秘密通信。

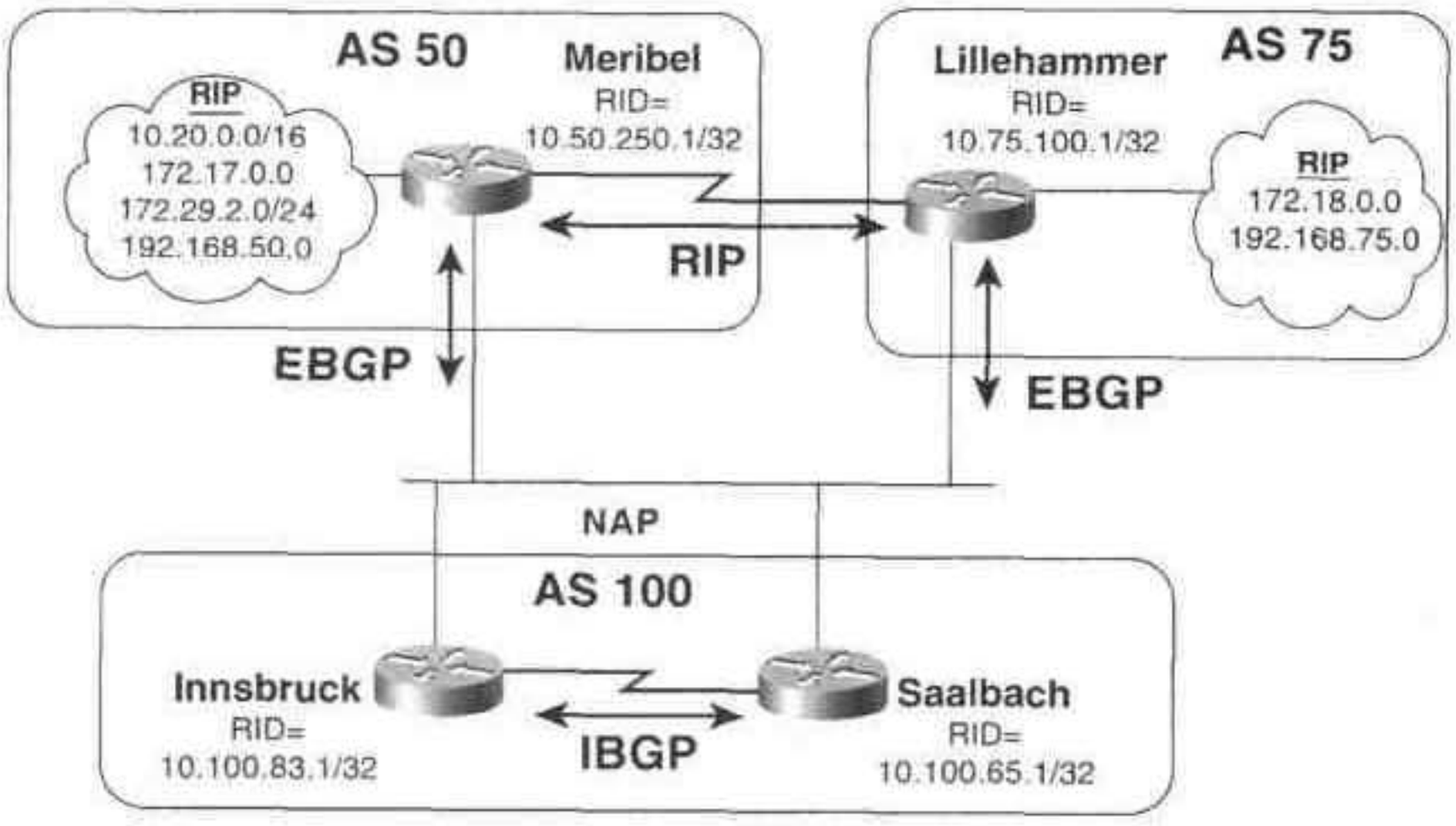


图 3-21　AS 50 和 AS 75 间的后门路由也应用做备份路由

以前，通过 **network backdoor** 命令使 AS 75 不公布 172.17.0.0，同时 AS 50 不公布 172.18.0.0。但是，在这个例子中，AS 50 和 AS 75 希望 AS 100 中的路由器能够把该后门链路作为一条备用路由。例如，如果 Meribel 到 NAP 的接口出现了故障，Innsbruck 和 Saalbach 会把目的地是 172.17.0.0 的数据包转发给 Lillehammer，然后再通过秘密路由将它们转发到 AS 50 内。这就要求 Meribel 和 Lillehammer 公布不在它们自己 AS 内的网络并且清楚地明确这些路由为备用路由。在这种情况下，因为到同一个目的地的路由是由在不同 AS 内的路由器发起的，因此比较不同 AS 的 MED 就变得有意义了。

有两个命令与图 3-21 中的情况相似。第一个是 **set metric-type internal**，这个命令，作为路由图的一部分，将一个 BGP 路由的 MED 设置成与到同一个目的地的 IGP 路由相同的度量。例如，Meribel 到 172.17.0.0 的 RIP 路由是一跳，而到同一个目的地，Lillehammer 通过后门链路学习到的 RIP 路由是两跳。**set metric-type internal** 命令使得这些路由器公布网络的路由继承了那些度量。结果是，AS 100 内的路由器优选 MED 为 1 的 Meribel 的路由，而没有优选到同一个目的地的 Lillehammer 的路由，因为它的路由的 MED 为 2。

第二个与这个环境有关的命令是用在接收一侧——如图 3-21 所示，是用在 AS 100 内的路由器上，该命令是 **bgp always-compare-med**。该命令告诉路由器去比较到同一个目的地的多条路由的 MED，而这些路由有可能是由不同的 AS 发起的。通过使用这两个命令，例 3-107 给出了 Meribel、Lillehammer 和 Saalbach 的配置，并允许 Saalbach 比较来自不同 AS 的 MED 值。

例 3-107　图 3-21 中 Meribel、Lillehammer 和 Saalbach 的配置

```
Meribel
router bgp 50
 network 172.17.0.0
 network 172.18.0.0
 redistribute rip
 neighbor 10.75.100.1 remote-as 75
 neighbor 10.75.100.1 ebgp-multihop 2
 neighbor 10.75.100.1 update-source Loopback0
 neighbor 10.75.100.1 distribute-list 2 in
 neighbor 10.100.65.1 remote-as 100
 neighbor 10.100.65.1 ebgp-multihop 2
 neighbor 10.100.65.1 update-source Loopback0
 neighbor 10.100.65.1 route-map MED out
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 route-map MED out
no auto-summary
!
ip as-path access-list 1 permit ^$
!
access-list 2 permit 192.168.75.0
access-list 2 permit 172.18.0.0
!
route-map MED permit 10
 match as-path 1
 set metric-type internal
```

Lillehammer

```
router bgp 75
 network 172.17.0.0
 network 172.18.0.0
 network 192.168.75.0
 neighbor 10.50.250.1 remote-as 50
 neighbor 10.50.250.1 ebgp-multihop 2
 neighbor 10.50.250.1 update-source Loopback0
 neighbor 10.50.250.1 distribute-list 2 in
 neighbor 10.100.65.1 remote-as 100
 neighbor 10.100.65.1 ebgp-multihop 2
 neighbor 10.100.65.1 update-source Loopback0
 neighbor 10.100.65.1 route-map MED out
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 route-map MED out
!
ip as-path access-list 1 permit ^$
!
access-list 2 permit 10.20.0.0
access-list 2 permit 172.17.0.0
access-list 2 permit 172.29.0.0
access-list 2 permit 192.168.50.0
!
route-map MED permit 10
 match as-path 1
 set metric-type internal
```

Saalbach

```
router bgp 100
 no synchronization
 bgp always-compare-med
 neighbor 10.50.250.1 remote-as 50
 neighbor 10.50.250.1 ebgp-multihop 2
 neighbor 10.50.250.1 update-source Loopback0
 neighbor 10.50.250.1 filter-list 2 in
 neighbor 10.75.100.1 remote-as 75
 neighbor 10.75.100.1 ebgp-multihop 2
 neighbor 10.75.100.1 update-source Loopback0
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 next-hop-self
 neighbor 10.100.83.1 filter-list 1 in
 neighbor 172.30.255.150 remote-as 30
 neighbor 172.30.255.150 ebgp-multihop 2
 neighbor 172.30.255.150 update-source Loopback0
 no auto-summary
```

注意在 Meribel 的配置中，虽然 172.17.0.0 是再分发给 BGP 的，还是有一条 **network** 命令用于该网络。因为有一条 **network** 命令用于 Lillehammer 处的这条路由，因此这个命令在将该路由的 ORIGIN 改为 IGP 时是必需的。如果没有该命令，来自 Meribel 路由的 ORIGIN 将是 Incomplete。在 BGP 的决定过程中，ORIGIN 的优先级高于 MED，也就是说 AS 100 会优选 Lillehammer 处的路由，虽然它的 MED 值较高。

在 Meribel 的配置中另外一个重要的细节是，分配列表会对 Lillehammer 的 NLRI 进行过滤。该过滤器允许到 192.168.75.0 和 172.18.0.0 的路由，拒绝所有其他的路由。尤其重要的

一点是由 Lillehammer 公布的到 172.17.0.0 的路由被该过滤器拒绝了。否则，管理距离为 20 的 EBGP 路由，将会优于 Meribel 上的 RIP 路由，从而导致出现路由环路。

例 3-108 给出了 Saalbach 相应的 BGP 表。从 Meribel 到 172.17.0.0 的路由，MED 为 1，因此它优于 MED 为 2 的从 Lillehammer 到 172.17.0.0 的路由。172.18.0.0 直接与 Lillehammer 相连，因此本地度量以及相应的来自该路由器的 MED 为 0，该路由通过 RIP 公布给 Meribel 以后，在 Meribel 处的本地度量为 1，而且该结果也可以在 Meribel 路由的 MED 中反映出来。如果优选路由的任何一个出现故障，会使用替换路由，到那个目的地的业务量会使用后门链路。

例 3-108　到 172.17.0.0 和 172.18.0.0 路由的 MED 与 AS 50 和 AS 75 的内部 RIP 度量相匹配

```
Saalbach#show ip bgp
BGP table version is 54, local router ID is 10.100.65.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.20.0.0/16     10.50.250.1              0             0 50 ?
*> 10.21.0.0/16     10.50.250.1              0             0 50 ?
* i10.30.0.0/16     10.100.83.1            200    100      0 30 ?
*>                  172.30.255.150         200             0 30 ?
*> 10.50.250.1/32   10.50.250.1              0             0 50 ?
*> 10.75.100.1/32   10.50.250.1              0             0 50 ?
*> 10.100.83.1/32   10.50.250.1              0             0 50 ?
*> 172.16.0.0       172.30.255.150         100             0 30 ?
*> 172.17.0.0       10.50.250.1              1             0 50 i
*                   10.75.100.1              2             0 75 i
*  172.18.0.0       10.50.250.1              1             0 50 i
*>                  10.75.100.1              0             0 75 i
*> 172.29.0.0       10.50.250.1              1             0 50 ?
* i172.29.1.0/24    10.100.83.1            200    100      0 30 ?
*>                  172.30.255.150         200             0 30 ?
*>i172.31.0.0       10.100.83.1            100    100      0 30 ?
*                   172.30.255.150         200             0 30 ?
*> 192.168.50.0     10.50.250.1              1             0 50 ?
*> 192.168.75.0     10.75.100.1              0             0 75 i
* i192.168.100.0    10.100.83.1            200    100      0 30 ?
*>                  172.30.255.150         200             0 30 ?
Saalbach#
```

注：在本例中没有给出 Innsbruck 的配置和 BGP 表，但是它们是相似的。

3.2.9　案例分析：附加 AS_PATH

MULTI_EXIT_DISC 属性会影响来自邻居的入业务量，但是它不能影响更远程 AS 的路由决定。

图 3-22 重复了前面出现过的案例分析的拓扑图。看一下例 3-109 中 Meribel 的 BGP 表，该例主要讲述的是 Meribel 的 BGP 表给出了到 AS 30 内目的地的两条路径；因为 Innsbruck 的路由器 ID 值(10.100.83.1)低于 Cervinia(10.200.60.1)的路由器 ID 值，因此 Innsbruck 被选作到所有目的地的最佳路径，可以看到该路由器到 AS 30 内目的地有两条一样的、开销相同的路径，因为所有其他的值都相同，Meribel 的 BGP 决定过程会为所有到 AS 30 的业务量选取

Innsbruck 作为下一跳路由器，因为它有较低的路由器 ID。结果是，从 AS 50 到 AS 30 的所有业务量都没有使用 Cervinia-Moritz 链路；因此可用带宽的利用率很低。

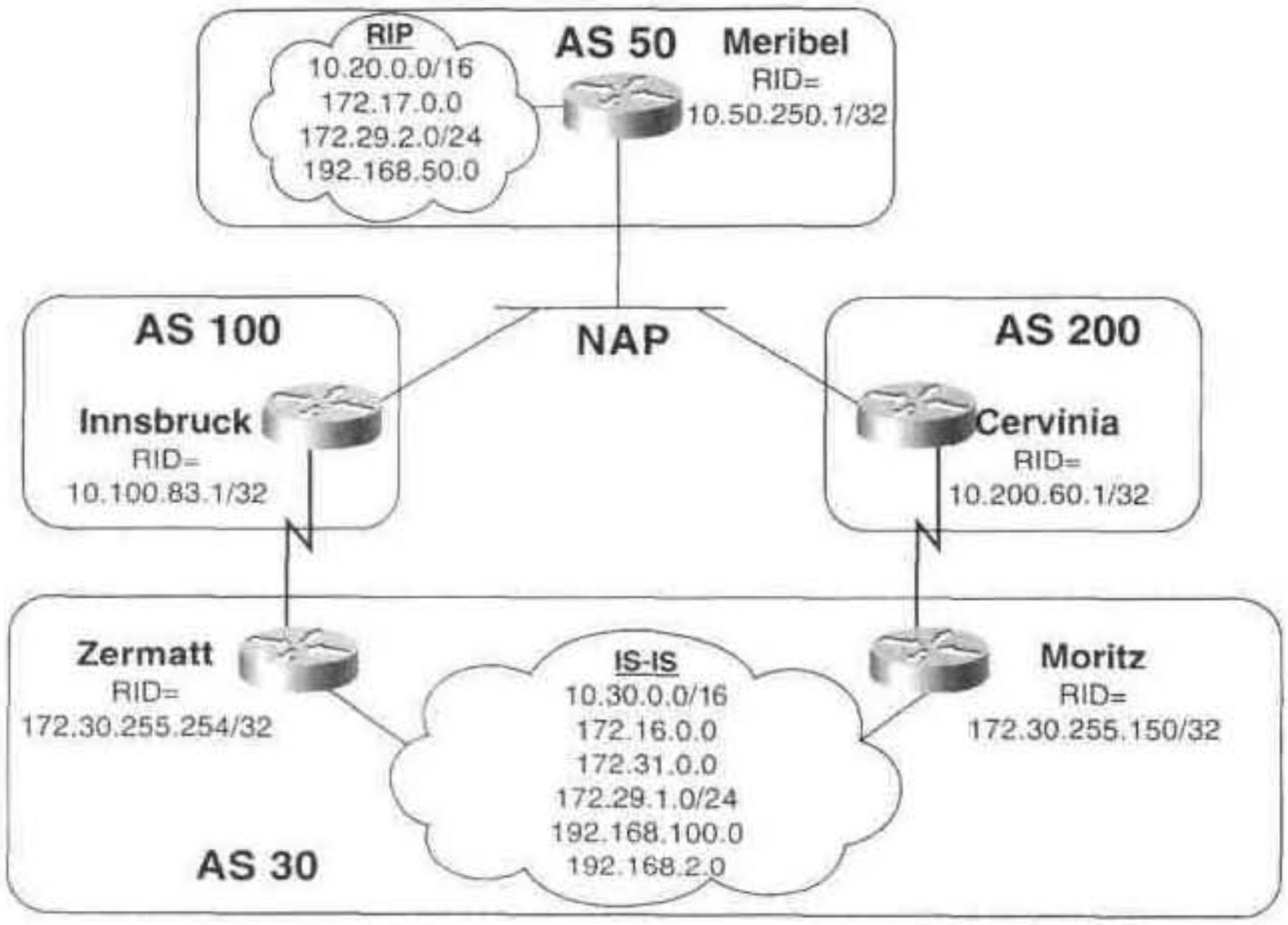


图 3-22　在 AS 50 和 AS 30 之间存在两条开销相同的路径

例 3-109　Meribel 选中 Innsbruck 作为到所有目的地的最佳路径

```
Meribel#show ip bgp
BGP table version is 18, local router ID is 10.50.250.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.20.0.0/16     0.0.0.0                  0         32768 ?
*  10.30.0.0/16     10.200.60.1                            0 200 30 ?
*>                  10.100.83.1                            0 100 30 ?
*> 10.50.250.1/32   0.0.0.0                  0         32768 ?
*> 10.100.65.1/32   0.0.0.0                  0         32768 ?
*> 10.100.83.1/32   0.0.0.0                  0         32768 ?
*> 10.200.60.1/32   0.0.0.0                  0         32768 ?
*  172.16.0.0       10.200.60.1                            0 200 30 ?
*>                  10.100.83.1                            0 100 30 ?
*> 172.17.0.0       10.20.1.1                1         32768 i
*> 172.29.0.0       10.20.1.1                1         32768 ?
*  172.29.1.0/24    10.200.60.1                            0 200 30 ?
*>                  10.100.83.1                            0 100 30 ?
*> 172.30.255.150/32 10.100.83.1                           0 100 30 ?
*> 172.30.255.254/32 10.200.60.1                           0 200 30 ?
*  172.31.0.0       10.200.60.1                            0 200 30 ?
*>                  10.100.83.1                            0 100 30 ?
*> 192.168.2.0/30   10.200.60.1                            0 200 30 ?
*> 192.168.2.4/30   10.100.83.1                            0 100 30 ?
*> 192.168.50.0     10.20.1.1                1         32768 ?
*  192.168.100.0    10.200.60.1                            0 200 30 ?
*>                  10.100.83.1                            0 100 30 ?
Meribel#
```

AS 30 通过 MED 无法影响 AS 50 的路由决定，因为这两个 AS 不是直接邻居。但是 AS 30 可以使用 **set as-path prepend** 命令修改它公布路由的 AS_PATH 从而影响 AS 50 的路由决定。假设 AS 30 希望 AS 50 能够将到 172.16.0.0 和 172.31.0.0 的业务量转发给 Cervinia 而将到 10.30.0.0、172.29.1.0/24 和 192.168.100.0 的业务量转发给 Innsbruck。例 3-110 给出了 Zermatt 和 Moritz 的配置。

例 3-110　配置 Zermatt 和 Moritz 的 AS_PATH 附加功能

Zermatt

```
router bgp 30
 no synchronization
 redistribute isis level-2
 neighbor 10.100.83.1 remote-as 100
 neighbor 10.100.83.1 ebgp-multihop 2
 neighbor 10.100.83.1 update-source Loopback0
 neighbor 10.100.83.1 route-map PATH out
 neighbor 10.100.83.1 filter-list 1 out
 no auto-summary
!
ip as-path access-list 1 permit ^$
!
access-list 3 permit 172.31.0.0
access-list 3 permit 172.16.0.0
!
route-map PATH permit 10
 match ip address 3
 set as-path prepend 30
!
route-map PATH permit 20
```

Moritz

```
router bgp 30
 no synchronization
 redistribute isis level-2
 neighbor 10.200.60.1 remote-as 200
 neighbor 10.200.60.1 ebgp-multihop 2
 neighbor 10.200.60.1 update-source Loopback0
 neighbor 10.200.60.1 route-map PATH out
 neighbor 10.200.60.1 filter-list 1 out
 no auto-summary
!
ip as-path access-list 1 permit ^$
!
access-list 3 permit 192.168.100.0
access-list 3 permit 10.30.0.0
access-list 3 permit 172.29.1.0
!
route-map PATH permit 10
 match ip address 3
 set as-path prepend 30
!
route-map PATH permit 20
```

每个路由器通过一个名为 PATH 的路由图过滤流出的数据包。路由图的序列 10 使用访问列表 3，通过它们的 NLRI 来识别特定的路由；同时与在 AS-PATH 中加入了 AS 30 的路由相

匹配。注意这个 AS 30 是在 AS-PATH 中正常地加入了 AS 30 以后再次加入的。路由图的序列 20 允许不与访问列表 3 匹配的路由。

例 3-111 给出了 Meribel 相应的 BGP 表。现在附加了 AS 的路由比优选路径的路由长，因此路由器就会选择有较短 AS_PATH 的路由。

例 3-111　路由器选择 AS_PATH 较短的路由

```
Meribel#show ip bgp
BGP table version is 70, local router ID is 10.50.250.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.20.0.0/16     0.0.0.0                  0         32768 ?
*  10.30.0.0/16     10.200.60.1                            0 200 30 30 ?
*>                  10.100.83.1                            0 100 30 ?
*> 10.50.250.1/32   0.0.0.0                  0         32768 ?
*> 10.100.65.1/32   0.0.0.0                  0         32768 ?
*> 10.100.83.1/32   0.0.0.0                  0         32768 ?
*> 10.200.60.1/32   0.0.0.0                  0         32768 ?
*> 172.16.0.0       10.200.60.1                            0 200 30 ?
*                   10.100.83.1                            0 100 30 30 ?
*> 172.17.0.0       10.20.1.1                1         32768 i
*> 172.29.0.0       10.20.1.1                1         32768 ?
*  172.29.1.0/24    10.200.60.1                            0 200 30 30 ?
*>                  10.100.83.1                            0 100 30 ?
*> 172.30.255.150/32 10.100.83.1                           0 100 30 ?
*> 172.30.255.254/32 10.200.60.1                           0 200 30 ?
*> 172.31.0.0       10.200.60.1                            0 200 30 ?
*                   10.100.83.1                            0 100 30 30 ?
*> 192.168.2.0/30   10.200.60.1                            0 200 30 ?
*> 192.168.2.4/30   10.100.83.1                            0 100 30 ?
*> 192.168.50.0     10.20.1.1                1         32768 ?
*  192.168.100.0    10.200.60.1                            0 200 30 30 ?
*>                  10.100.83.1                            0 100 30 ?
Meribel#
```

你应该非常小心地使用 AS_PATH 附加功能。如果你没有完全理解采用该配置后的结果，可能会出现意外的路由或者破坏路由。假设，在 Moritz 的配置中使用了 **set as-path prepend 30 30** 命令，该命令在 AS_PATH 中加入了两次 AS 30 而不是一次。查看一下对到 10.30.0.0 路由的影响，图 3-22 中的 Cervinia 从 Moritz 收到路由的 AS_PATH 是(30,30,30)，从 Meribel 收到的到同一个目的地路由的 AS_PATH 是(50,100,30)。因为这些路由有相同的 AS-PATH 长度，Cervinia 会选择具有最低下一跳地址的路由，也就是来自 Meribel 的路由。该配置的本意只是想影响 AS 50 内的路由，但是，此时这种配置也使得 AS 200 选择了一条较长的到目的地的路径。

通常，使用附加的时候，附加路由器所在 AS 的 AS 号也是非常重要的。如果使用了另外的 AS 号，而且当被宣告路由遇到了正在使用该号码的 AS，那个 AS 不会接受该路由。

3.2.10　案例分析：路由标记

可以将一个路由标记段看作是路由更新消息中的一种“口袋”，用于在一个路由域

内传输信息。由标记所代表的信息与路由协议本身没有关系，而且路由协议没有以任何方式作用于标记。当将一条路由从协议 A 再分发到协议 B 时而且在其他的地点再分发回协议 A 时，标记是有用的。在转接路由协议的更新消息中，标记段允许协议 A 发送消息给在转接域中另外一边的对端。通常，这些信息对于转接路由协议来讲是不一致或者无意义的。

RIP-2、EIGRP、综合 IS-IS、OSPF 以及 BGP 都支持路由标记，RIP-1 和 IGRP 不支持路由标记。第一册的第 14 章介绍了路由标记并且给出了一些有关它的用法的例子。很显然，这个案例分析重点是放在 BGP 环境下标记的使用。

图 3-23 描述了一个环境，在这个环境下路由标记是有用的。AS 1300 负责传递几个 AS 之间的业务量。三个边缘 AS 的每条路由都通过 EBGP 公布给 AS 1300 中 3 个边界路由器中的一个。路由被再分发给 OSPF，然后又再分发给另外两个边界路由器的 BGP，最后公布给它们的 EBGP 对端。

> 注：　现在所描述的这种配置：将 BGP 再分发给 IGP，在一些大型的企业网中会有此应用。但是，本章已经多次声明，在业务供应商的 AS 或者在一个接收到了大量 BGP 前缀的 AS 里，永远也不应该把 BGP 路由再分发到 IGP 中。

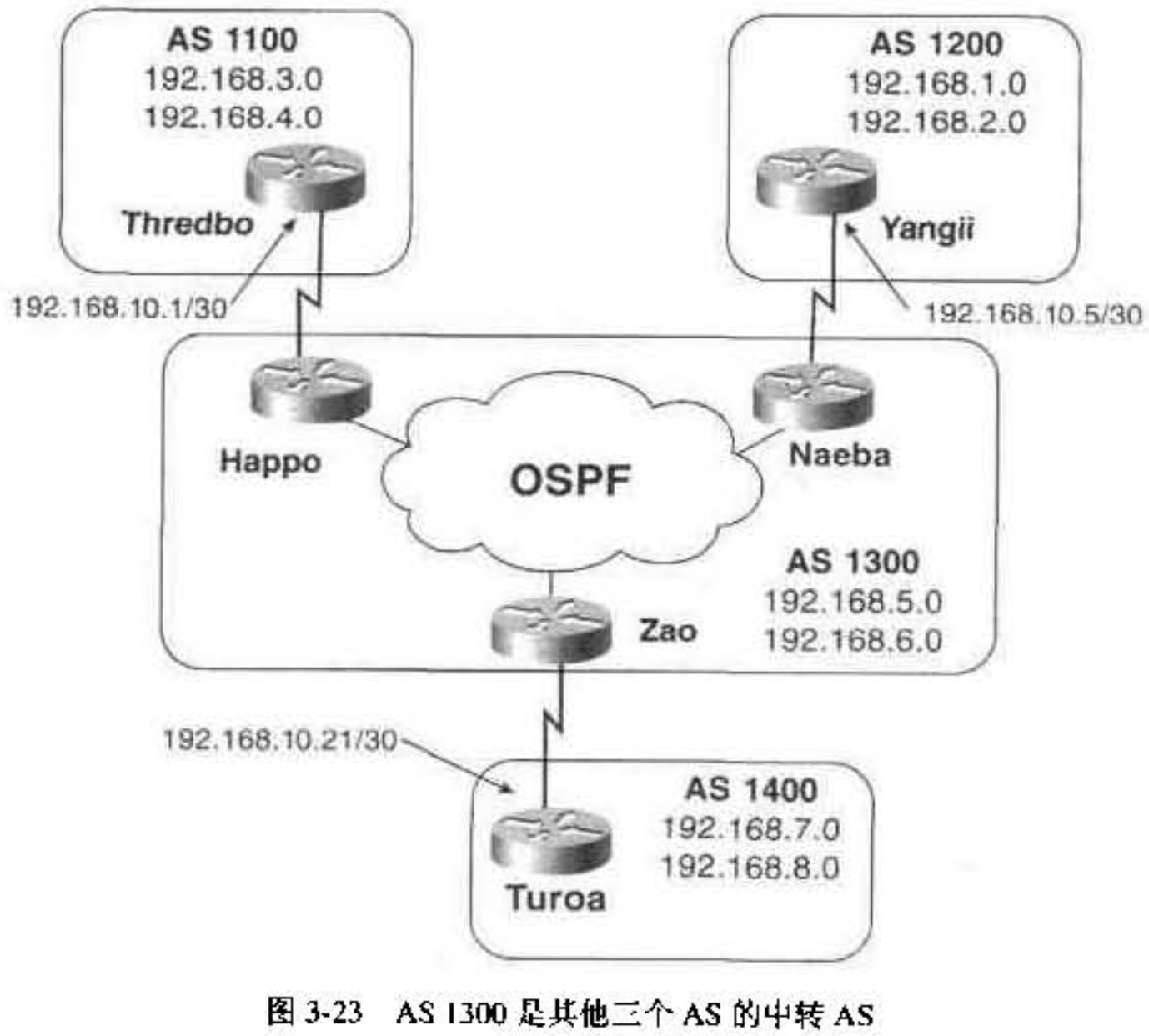


图 3-23　AS 1300 是其他三个 AS 的中转 AS

图 3-23 中拓扑的问题在于，三个边远 AS 的 BGP 进程必须通过 OSPF 来共享它们的路由信息，但是 OSPF 无法理解 BGP 路径的属性，结果是，路径消息丢失了。例 3-112 给出了 Turoa 的 BGP 表，该表讲述的是来自 AS 1100 和 AS 1200 的路由的 AS_PATH 属性在通过 AS 1300 的 OSPF 域时没有保存下来；结果是，从 AS 1300 学习到的路由好像都是由该 AS 发起的。如果 AS 1400 存在一条到 AS 1100 或者 AS 1200 的可替换路径，因为上面所讲的路径消息的丢失，AS 1400 不能做出精确的路由决定。

例 3-112　从 AS 1300 学习到的路由好像都是由该 AS 发起的

```
Turoa#show ip bgp
BGP table version is 44, local router ID is 192.168.8.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.0      192.168.10.22            1             0 1300 ?
*> 192.168.2.0      192.168.10.22            1             0 1300 ?
*> 192.168.3.0      192.168.10.22            1             0 1300 ?
*> 192.168.4.0      192.168.10.22            1             0 1300 ?
*> 192.168.5.0      192.168.10.22           20             0 1300 ?
*> 192.168.6.0      192.168.10.22           20             0 1300 ?
*> 192.168.7.0      0.0.0.0                  0         32768 i
*> 192.168.8.0      0.0.0.0                  0         32768 i
*> 192.168.10.0     192.168.10.22          192             0 1300 ?
Turoa#
```

BGP 可以在 OSPF 数据包中使用路由标记段从而在 OSPF 域中传输 AS_PATH 信息。实际上，Cisco 的 BGP 执行进程会自动执行这个动作。例 3-113 给出了 Zao 到 AS 1200 中 192.168.1.0 的路由的详细情况，该路由通过了 OSPF 域，如例中所显示，当路由器 Naeba 将它到 192.168.1.0 的 EBGP 路由再分发给 OSPF 时，它将该路由的 AS_PATH 写入 OSPF 的外部 AS LSA 的外部路由标记段中；在 OSPF 域的另一端，在 Zao 的路由条目中可以看到该标记。注意标记段用 1200 来标记。

例 3-113　Naeba 将 EBGP 路由 192.168.1.0 再分发时，将其做了标记

```
Zao#show ip route 192.168.1.0
Routing entry for 192.168.1.0/24
  Known via "ospf 1300", distance 110, metric 1
  Tag 1200, type extern 2, forward metric 128
  Redistributing via ospf 1300, bgp 1300
  Advertised by bgp 1300 match internal external 2
  Last update from 192.168.10.18 on Serial1.503, 00:13:33 ago
  Routing Descriptor Blocks:
  * 192.168.10.18, from 192.168.10.13, 00:13:33 ago, via Serial1.503
      Route metric is 1, traffic share count is 1
Zao#
```

想要了解有关 OSPF 外部 AS LSA 在格式以及使用方面的详细情况，参见第 1 卷，第 9 章的“开放式最短路径优先(OSPF)”。

但是，当将 IGP 路由再分发到 BGP 时，BGP 进程不会自动认为在 IGP 的标记段中包含了 AS_PATH 信息，你必须对该过程进行配置从而让它恢复 AS_PATH 信息。在再分发路由的标记中恢复 AS_PATH 信息的一种方法是通过 **set as-path tag** 命令。例 3-114 给出了 Zao 使用 **set as-path tag** 命令的配置情况。

例 3-114　配置 Zao，使它在再分发路由的标记中恢复 AS_PATH 信息

```
router ospf 1300
 redistribute bgp 1300
 network 192.168.10.0 0.0.0.255 area 0
!
router bgp 1300
 redistribute ospf 1300 match internal external 2 route-map GET_TAG
 neighbor 192.168.10.21 remote-as 1400
!
route-map GET_TAG permit 10
 set as-path tag
```

在 BGP 配置下的 **redistribute ospf** 命令涉及到了一个名为 GET_TAG 的路由图，该路由图将再分发路由的 AS_PATH 属性设置为在 OSPF 标记段内的值。例 3-115 显示出，现在 Turoa 的 BGP 表中包含了到 AS 1100 和 AS 1200 的路由的精确的 AS_PATH 信息。

例 3-115　现在 Turoa 的 BGP 表包含了精确的 AS_PATH 信息

```
Turoa#show ip bgp
BGP table version is 148, local router ID is 192.168.8.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.0      192.168.10.22            1             0 1300 1200 ?
*> 192.168.2.0      192.168.10.22            1             0 1300 1200 ?
*> 192.168.3.0      192.168.10.22            1             0 1300 1100 ?
*> 192.168.4.0      192.168.10.22            1             0 1300 1100 ?
*> 192.168.5.0      192.168.10.22           20             0 1300 ?
*> 192.168.6.0      192.168.10.22           20             0 1300 ?
*> 192.168.7.0      0.0.0.0                  0         32768 i
*> 192.168.8.0      0.0.0.0                  0         32768 i
*> 192.168.10.0     192.168.10.22          192             0 1300 ?
Turoa#
```

为了 Zao 能够利用 **set as-path tag** 命令在 OSPF 标记段中恢复 AS_PATH 信息，在 Happo 或者 Naeba 处不需要特殊配置。但是，自动加入到标记段的信息只有 AS_PATH。注意在例 3-115 中，来自 AS 1100 和 AS 1200 的路由的 ORIGIN 为 Incomplete。路由器 Thredbo 和 Yangii 通过 BGP **network** 命令以 IGP 的 ORIGIN 公布它们的内部路由，但是 Happo 和 Naeba 没有将这些信息加入到 OSPF 的标记中。因为在 Zao 处将路由从 OSPF 再分发到 BGP，因此 Turoa 所看到的 ORIGIN 是 Incomplete。

根据 Turoa 是否有到 AS 1100 以及 AS 1200 的可替换路由，以及路由的 ORIGIN 是否会影响到 BGP 的决定过程，以上的情况可能是问题也有可能不是问题。为此 Cisco 提供了一个可选的配置，称为*自动标记*，这样，在标记段不仅加入 AS_PATH 信息，还加入了 ORIGIN 码。通过 **set automatic-tag** 命令设置自动标记，而且包含该命令的路由图是通过 **table-map** 命令在 BGP 处理中调用的。在路由器上使用 **set as-path tag** 命令，目的是将 IGP 路由再分发到 BGP 中，而与 **set as-path tag** 命令不同的是，**set automatic-tag** 命令是将 BGP 路由再分发给 IGP。

图 3-23 里，为了设置自动标记，AS 1300 中的三个路由器的配置完全相同。例 3-116 给出了 Naeba 的配置。

例 3-116　配置 Naeba，加入 AS_PATH 信息和 ORIGIN 码

```
router ospf 1300
 redistribute bgp 1300
 network 192.168.0.0 0.0.255.255 area 0
!
router bgp 1300
 table-map SET_TAG
 redistribute ospf 1300 match internal external 2
 neighbor 192.168.10.5 remote-as 1200
!
ip as-path access-list 1 permit .*
!
route-map SET_TAG permit 10
 match as-path 1
 set automatic-tag
```

例 3-117 给出了 Turoa 相应的 BGP 表。可以看出它与例 3-115 中的 BGP 表几乎相同，但是来自 AS 1100 和 AS 1200 的路由现在可以正确地反映 IGP 的 ORIGIN 属性了。

例 3-117　现在 Turoa 的 BGP 表不仅包含了正确的 AS_PATH 信息，还包含了正确的 ORIGIN

```
Turoa#show ip bgp
BGP table version is 228, local router ID is 192.168.8.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 192.168.1.0      192.168.10.22            1             0 1300 1200 i
*> 192.168.2.0      192.168.10.22            1             0 1300 1200 i
*> 192.168.3.0      192.168.10.22            1             0 1300 1100 i
*> 192.168.4.0      192.168.10.22            1             0 1300 1100 i
*> 192.168.5.0      192.168.10.22           20             0 1300 ?
*> 192.168.6.0      192.168.10.22           20             0 1300 ?
*> 192.168.7.0      0.0.0.0                  0         32768 i
*> 192.168.8.0      0.0.0.0                  0         32768 i
*> 192.168.10.0     192.168.10.22          192             0 1300 ?
Turoa#
```

路由标记的另外一个用处是出于过滤的原因而明确一定的路由组。在图 3-23 中，配置路由器 Naeba 和 Happo，使它们在将 EBGP 路由再分发给 OSPF 之前，标记它们 EBGP 路由的一些子网。在 OSPF 域内得到这些路由的 Zao，可以通过它们共同的标记来识别它们而无须通过 NLRI 来过滤它们。回忆一下，在第 2 章曾讲过，设计 BGP 团体属性的目的是为了在一个公共的组中识别路由。但是，在图 3-23 的拓扑中，团体属性不能在 OSPF 域内进行通信。如果需要在这个目的下如何使用的标记的例子，可以参见第 1 卷，第 14 章的“案例分析：路由标记”。

最后，只有将 IGP 再分发给 BGP 时，可以使用标记。在使用 **network** 命令的时候，认为 BGP 路由是由本地发起的，因此它不会继承 IGP 路由的任何属性包括标记。

3.2.11　案例分析：路由抑制

路由抑制，正如第二章所描述的，它是一个能够给摆动路由分配惩罚的进程。如果路由积累了一定的惩罚，该路由就会被抑制——也就是说，它不再被公布——在一定的时间

段内。缺省的情况下，每次摆动分配给路由的惩罚值是 1000。如果路由累积惩罚的值超过了 2000，路由就会被抑制直到惩罚值降到低于 750。这些高、低的门限分别被称为*抑制限制*和*再使用限制*。累积惩罚每 5 秒减少一次，减少的速率是按照每 15 分钟惩罚减少一半的速率。可以看出这个名为*半衰期*的速率，是指数形式的。如果惩罚是 3000，15 分钟后它会减少为 1500；如果惩罚是 300，15 分钟后它是 150。而且还有一个路由抑制的最长时间，称为*最大抑制限制*。在缺省的情况下，该限制是半衰期的 4 倍，或者可以说是 60 分钟。

在 BGP 进程配置中，通过 **bgp dampening** 命令使路由抑制生效。如果想改变缺省值，语法是 **bgp dampening** *half-life reuse supress max-supress*。

图 3-24 给出了一个拓扑图，在该图中，路由器 Colorado 多宿主到 5 个其他的 AS。如果由任何一个远程 AS 公布的路由出现了摆动，Colorado 必须向其他所有的 EBGP 对端公布这个变化。对于一个简单的拓扑来讲，这可能不算很重的负担，但是，想象一下，如果 Colorado 有 150 个 EBGP 对等而不是 5 个，那后果该什么样呢？一个有规律摆动的路由会给核心路由器带来非常沉重的处理负担。

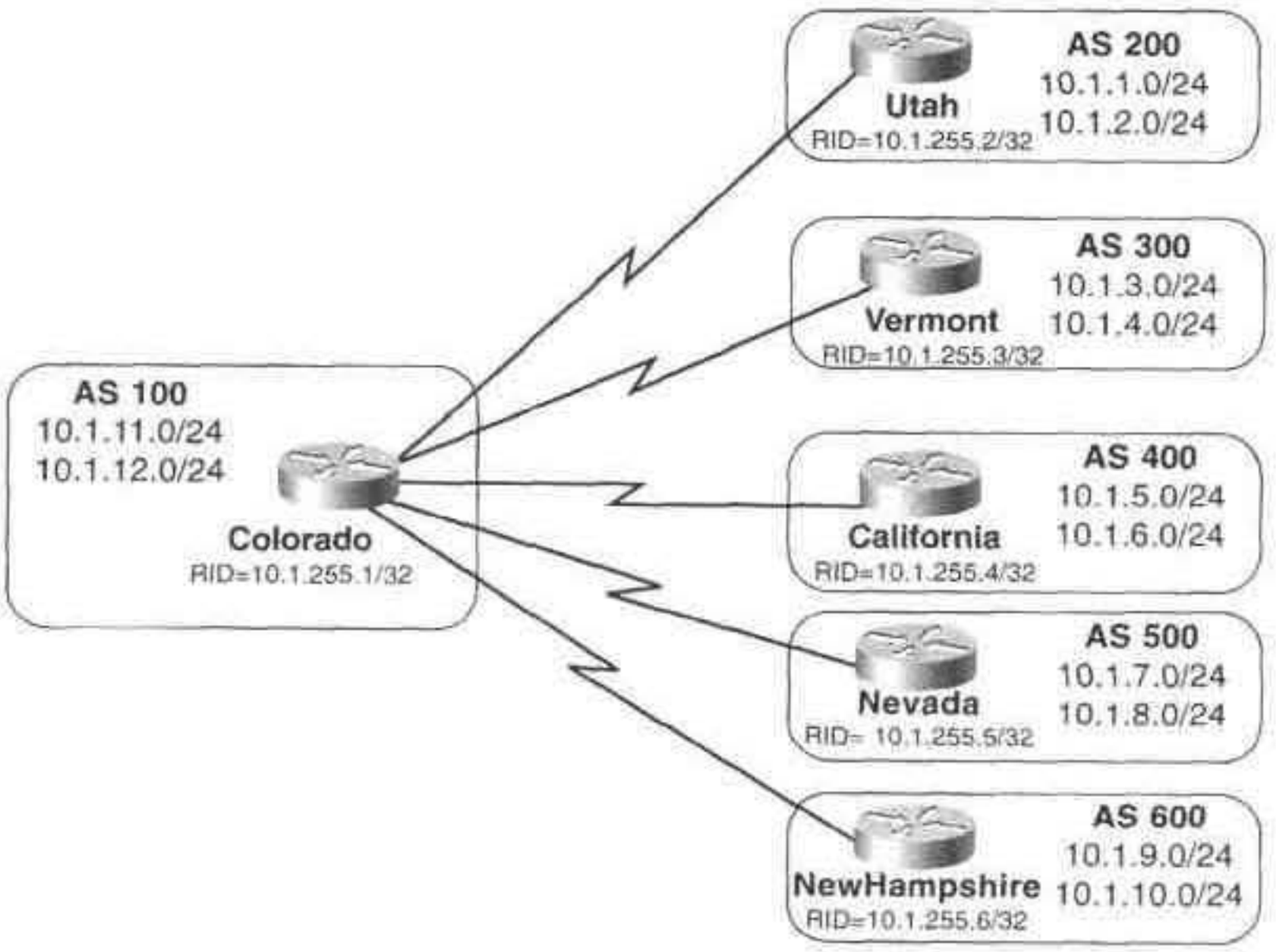


图 3-24　Colorado 必须向它所有的 EBGP 对端公布其他 AS 上的路由变化

例 3-118 给出了 Colorado 的 BGP 配置，如例中所示。当出现路由摆动时，它可以向它的 EBGP 对等发送一个更新消息，从而向它们公布这个变化。

例 3-118　Colorado 的配置

```
router bgp 100
 bgp dampening
 network 10.1.11.0 mask 255.255.255.0
 network 10.1.12.0 mask 255.255.255.0
 neighbor 10.1.255.2 remote-as 200
 neighbor 10.1.255.2 ebgp-multihop 2
 neighbor 10.1.255.2 update-source Loopback2
 neighbor 10.1.255.3 remote-as 300
```

```
neighbor 10.1.255.3 ebgp-multihop 2
neighbor 10.1.255.3 update-source Loopback2
neighbor 10.1.255.4 remote-as 400
neighbor 10.1.255.4 ebgp-multihop 2
neighbor 10.1.255.4 update-source Loopback2
neighbor 10.1.255.5 remote-as 500
neighbor 10.1.255.5 ebgp-multihop 2
neighbor 10.1.255.5 update-source Loopback2
neighbor 10.1.255.6 remote-as 600
neighbor 10.1.255.6 ebgp-multihop 2
neighbor 10.1.255.6 update-source Loopback2
no auto-summary
```

例 3-119 给出了 Colorado 的 BGP 表。注意 10.1.4.0/24 前面有一个标记 d，指示该路由已经被被抑制了。10.1.7.0/24 前面有一个标记 h，意味着该路由曾经发生过摆动；也就是说，虽然该路由没有积累到足够大的惩罚从而使它被抑制，但是它确实存在着惩罚。

例 3-119　有两条路由 10.1.4.0/24 和 10.1.7.0/24 都已经积累了惩罚

```
Colorado#show ip bgp
BGP table version is 756, local router ID is 10.1.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      10.1.255.2               0             0 200 i
*> 10.1.2.0/24      10.1.255.2               0             0 200 i
*> 10.1.3.0/24      10.1.255.3               0             0 300 i
*d 10.1.4.0/24      10.1.255.3               0             0 300 i
*> 10.1.5.0/24      10.1.255.4               0             0 400 i
*> 10.1.6.0/24      10.1.255.4               0             0 400 i
 h 10.1.7.0/24      10.1.255.5               0             0 500 i
*> 10.1.8.0/24      10.1.255.5               0             0 500 i
*> 10.1.9.0/24      10.1.255.6               0             0 600 i

*> 10.1.10.0/24     10.1.255.6               0             0 600 i
*> 10.1.11.0/24     0.0.0.0                  0         32768 i
*> 10.1.12.0/24     0.0.0.0                  0         32768 i
```

在例 3-119 的 BGP 表中因为没有太多的路由条目，因此不稳定路由是相当明显的。但是，当一个 BGP 表中有上千条路由条目时，通过查找 d 和 h 来发现不稳定路由显然是不实际的。有两个命令使得这个查找工作变得简单了：**show ip bgp flap-statistics** 以及 **show ip bgp dampened-paths**。正如它们的名字所暗示的，第一个命令可以显示所有出现摆动的路由以及该路由出现摆动的次数。第二条命令只显示被抑制的路由。例 3-120 给出了这些命令在 Colorado 的使用情况；注意对于被抑制的路由来讲，这两个命令的输出都显示了希望再次公布这些路由的时间。这个时间对于不再分发更多惩罚的路由来讲是偶发的。注意只有配置了 BGP 路由抑制，才会记录摆动统计。当路由器没有运行路由抑制进程的时候，不能使用 **show ip bgp flap-statistics** 命令在一个路由器上查找不稳定路由。

例 3-120　在 BGP 表中只能显示已经摆动的路由，或者已经被抑制的路由

```
Colorado#show ip bgp flap-statistics
BGP table version is 756, local router ID is 10.1.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
   Network          From            Flaps Duration Reuse    Path
*d 10.1.4.0/24      10.1.255.3      3     00:15:52 00:19:40 300
 h 10.1.7.0/24      10.1.255.5      2     00:20:49          500

Colorado#show ip bgp dampened-paths
BGP table version is 757, local router ID is 10.1.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          From            Reuse   Path
*d 10.1.4.0/24      10.1.255.3      00:19:2 300 i
```

一个路由的详细情况不仅给出了在例 3-120 中所显示的统计数据，而且还给出了惩罚值。在例 3-121 中，可以看到到 10.1.4.0/24 路由的惩罚值为 1815；半衰期进程将惩罚降低到抑制门限 2000 之下，但是还没有到达再使用门限 750。第二个门限值可能在 19 分 10 秒内到达。

例 3-121　如果用 **shop ip bgp** 命令来指定一条不稳定路由，会显示该路由的惩罚值

```
Colorado#show ip bgp 10.1.4.0
BGP routing table entry for 10.1.4.0/24, version 755
Paths: (1 available, no best path, advertised over EBGP)
  300, (suppressed due to dampening)
    10.1.255.3 from 10.1.255.3
      Origin IGP, metric 0, localpref 100, valid, external
      Dampinfo: penalty 1815, flapped 3 times in 00:16:28, reuse in 00:19:10
Colorado#
```

在某些情况下，在再使用限制到达之前，你可能希望重新使用一条抑制路由。例如，AS 300 的管理者可能会保证已经确定并且已经清除了引起子网 10.1.4.0/24 摆动的原因，现在他希望能够恢复这条路由上的业务。在这种情况下，有两个命令可用：**clear ip bgp flap-statistics** 和 **clear ip bgp dampening**。在清除一条路由或者全部路由(要看是否通过命令指定了一条路由)惩罚方面，这两个命令有着相同的效果，但是第二个命令只能清除已经被抑制路由的惩罚。**clear ip bgp flap-statistics** 命令还使你能够通过 AS 路径，或者通过指定一个过滤列表或者使用正则表达式来识别一组路由。例如，**clear ip bgp flap-statistics regexp _30_**命令会清除所有在 AS_PATH 属性中有 AS 号 30 的路由的摆动统计。当 AS 30 是一个转接 AS，并且一条坏的链路已经导致通过该 AS 能够到达的所有目的地积累了惩罚的时候，该命令是很有用的。

3.3　大型 BGP

*大型*BGP 有些像是一个主观术语。当你的 BGP 拓扑增长到一定程度的时候，你可以自己决定是否需要调整使用的工具，这些工具在本节均有讨论。但是，通常在中型到较大型网络互连中会使用对等组和团体，但是通常只有在最大型的 BGP 拓扑中才可以看到联盟，例如大型 ISP 的 BGP 拓扑。下面的案例分析就会分别讨论这些工具。

3.3.1　案例分析：BGP 对等组

在前面的案例分析中，图 3-24 给出了一个 BGP 拓扑，在这个拓扑中，一个 AS 多宿主到其他的几个 AS。但是，假设路由器 Colorado 有 150 个 EBGP 对等而不是 5 个。除了标准的配置以外，每个与邻居之间的连接都有一个出站以及入站路由过滤器。因此，对于每个邻居来讲，有 5 个 BGP 配置命令：

- **neighbor remote-as** 命令
- **neighbor ebgp-multihop** 命令，因为连接是在 Loopback 地址上。
- **neighbor update-source** 命令，原因同上。
- **neighbor filter-list out** 命令
- **neighbor filter-in** 命令

如果是 150 个对等，就会有 150 个配置命令。

当同样的路由策略应用于多个 BGP 对端的时候，将对端指定为一个对等组的成员将会极大地简化一个路由器的 BGP 配置。曾经需要为每个邻居定义的大部分配置选项以及路由策略现在只要定义一次，也就是只需为对等组定义一次。一个对等组只和定义该对等组的路由器有关并且该路由器不会把这个对等组公布给它的对端。可以按照下面的三个步骤来生成一个对等组：

步骤 1　为对等组指定一个名字。

步骤 2　为对等组成员指定共同的路由策略和配置选项。

步骤 3　指定属于这个对等组的邻居。

例 3-122 的配置在图 3-24 的路由器 Colorado 上生成了一个名为 CLIENTS 的对等组。

例 3-122　在路由器 Colorado 上生成了一个名为 CLIENTS 的对等组

```
router bgp 100
 network 10.1.11.0 mask 255.255.255.0
 network 10.1.12.0 mask 255.255.255.0
 neighbor CLIENTS peer-group
 neighbor CLIENTS ebgp-multihop 2
 neighbor CLIENTS update-source Loopback2
 neighbor CLIENTS filter-list 2 in
 neighbor CLIENTS filter-list 1 out
 neighbor 10.1.255.2 remote-as 200
 neighbor 10.1.255.2 peer-group CLIENTS
 neighbor 10.1.255.3 remote-as 300
 neighbor 10.1.255.3 peer-group CLIENTS
 neighbor 10.1.255.4 remote-as 400
 neighbor 10.1.255.4 peer-group CLIENTS
 neighbor 10.1.255.5 remote-as 500
 neighbor 10.1.255.5 peer-group CLIENTS
 neighbor 10.1.255.6 remote-as 600
 neighbor 10.1.255.6 peer-group CLIENTS
 no auto-summary
!
ip as-path access-list 1 permit ^$
ip as-path access-list 2 permit ^[2-6]00$
```

neighbor CLIENTS peer-group 命令生成了一个对等组，下面的 4 个命令定义了对等组的所有成员共同的选项和策略。然后通过 **neighbor remote-as** 命令指定 EBPG 邻居，并且在

这个命令的后面又加入了一个命令指定这些邻居是对等组 CLIENTS 的成员。

通过合并共用的选项和策略，对等组可以在很大程度上缩短 BGP 配置。让我们再回到 Colorado 有 150 个 EBGP 对等的情况下，如果所有的对等都是对等组 CLIENTS 的成员，配置就会从 750 条命令减少到 305 条。这种配置也变得很容易解释。所有的选项都在一个地方配置，并且必须需要了解的就是哪个邻居是哪个对等组的成员。

当一个对等组的所有成员都属于同一个 AS 时，在对等组配置下指定共同的 AS 可以更进一步地缩短配置。在同一个远程 AS 中，所有的成员有可能都是 EBGP 对等，但是在大部分情况下，同一个 AS 内大量的对等可能都是 IBGP 对等。在图 3-25 中，加入的 NewMexico 和 Idaho 是 Colorado 的 IBGP 对等。

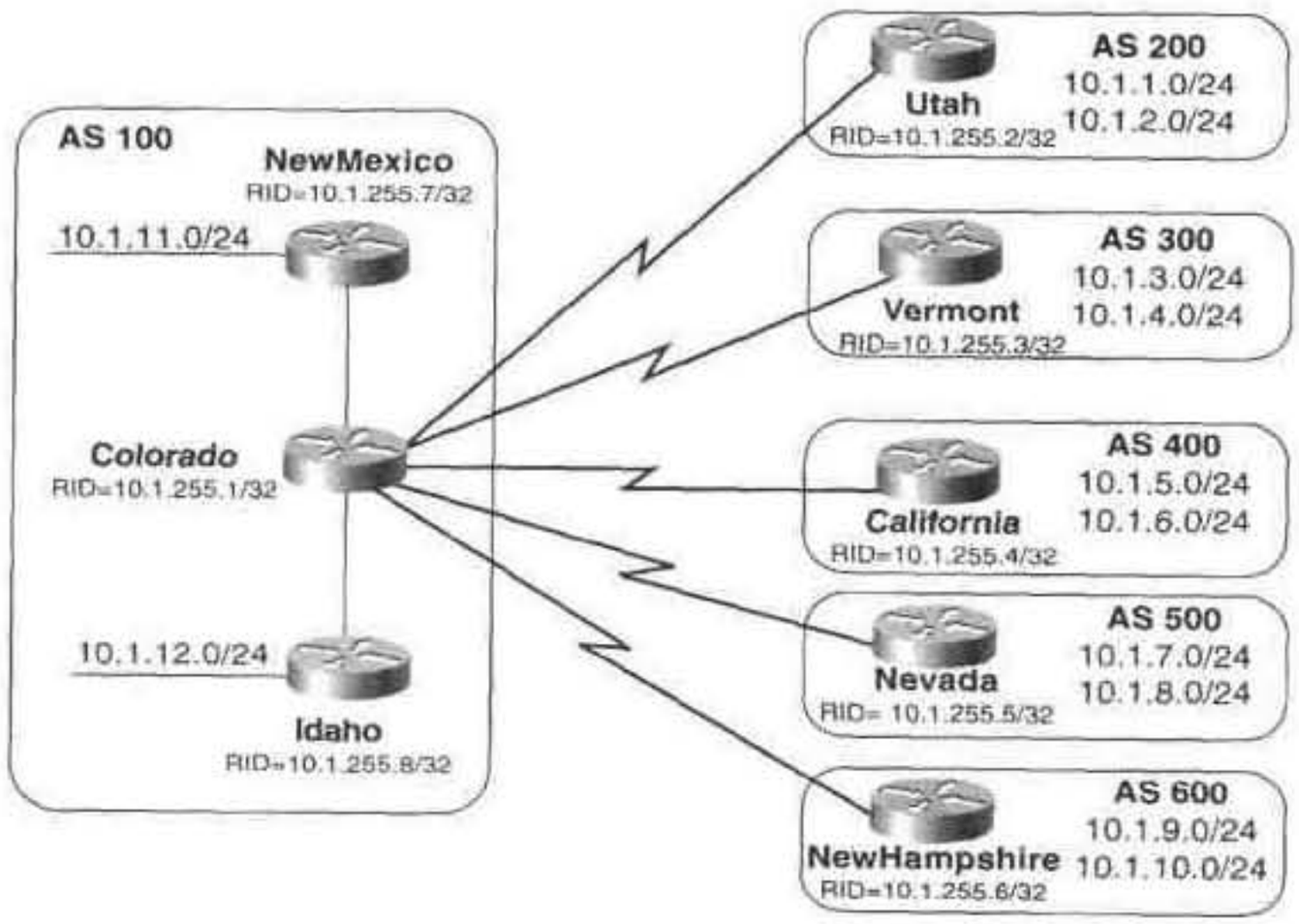


图 3-25　在 AS 100 中加入了两个 IBGP 对等

例 3-123 给出了 Colorado 的配置。

例 3-123　为 Colorado 的内部对等端置了一个对等组

```
router bgp 100
 no synchronization
 network 10.1.11.0 mask 255.255.255.0
 network 10.1.12.0 mask 255.255.255.0
 neighbor CLIENTS peer-group
 neighbor CLIENTS ebgp-multihop 2
 neighbor CLIENTS update-source Loopback2
 neighbor CLIENTS filter-list 2 in
 neighbor CLIENTS filter-list 1 out
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 100
 neighbor LOCAL next-hop-self
 neighbor LOCAL filter-list 3 out
 neighbor 10.1.255.2 remote-as 200
 neighbor 10.1.255.2 peer-group CLIENTS
 neighbor 10.1.255.3 remote-as 300
 neighbor 10.1.255.3 peer-group CLIENTS
 neighbor 10.1.255.4 remote-as 400
 neighbor 10.1.255.4 peer-group CLIENTS
```

```
 neighbor 10.1.255.5 remote-as 500
 neighbor 10.1.255.5 peer-group CLIENTS
 neighbor 10.1.255.6 remote-as 600
 neighbor 10.1.255.6 peer-group CLIENTS
 neighbor 10.1.255.7 peer-group LOCAL
 neighbor 10.1.255.8 peer-group LOCAL
 no auto-summary
!
ip as-path access-list 1 permit ^$
ip as-path access-list 2 permit ^[2-6]00$
ip as-path access-list 3 permit ^[246]00$
```

将 New Mexico 和 Idaho 加入到对等组 LOCAL 中，在对等组的配置中明确了它们共有的 AS 号，同时也明确了它们共同的出站路由策略。

> 注：　AS_PATH 列表 3 允许从来自 AS 200、400 或者 600 的任何路由。来自 AS 300 和 AS 500 的路由以由 Colorado 发起的本地路由被隐含拒绝。

为一个对等组成员定义的入站路由策略优先于为对等组定义的入站路由策略。例如，假设，Colorado 应该只接受来自 EBGP 对等 California 的子网 10.1.5.0/24，但是使用了所有其他对等组的策略和选项。例 3-124 给出了 Colorado 新的配置。

例 3-124　向对等组内的一个邻居应用路由策略

```
router bgp 100
 no synchronization
 network 10.1.11.0 mask 255.255.255.0
 network 10.1.12.0 mask 255.255.255.0
 neighbor CLIENTS peer-group
 neighbor CLIENTS ebgp-multihop 2
 neighbor CLIENTS update-source Loopback2
 neighbor CLIENTS filter-list 2 in
 neighbor CLIENTS filter-list 1 out
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 100
 neighbor LOCAL next-hop-self
 neighbor LOCAL filter-list 3 out
 neighbor 10.1.255.2 remote-as 200
 neighbor 10.1.255.2 peer-group CLIENTS
 neighbor 10.1.255.3 remote-as 300
 neighbor 10.1.255.3 peer-group CLIENTS
 neighbor 10.1.255.4 remote-as 400
 neighbor 10.1.255.4 peer-group CLIENTS
 neighbor 10.1.255.4 distribute-list 10 in
 neighbor 10.1.255.5 remote-as 500
 neighbor 10.1.255.5 peer-group CLIENTS
 neighbor 10.1.255.6 remote-as 600
 neighbor 10.1.255.6 peer-group CLIENTS
 neighbor 10.1.255.7 peer-group LOCAL
 neighbor 10.1.255.8 peer-group LOCAL
 no auto-summary
!
ip as-path access-list 1 permit ^$
ip as-path access-list 2 permit ^[2-6]00$
ip as-path access-list 3 permit ^[246]00$
access-list 10 permit 10.1.5.0
```

在邻居 California(10.1.255.4)中加入了分配列表 10。虽然 Colorado 的配置将 California 定义为 CLIENTS 对等组的一个成员，但是分配列表的存在可以不用考虑用于对等组的入站过滤列表 2。

可以通过 **show ip bgp peer-group** 命令显示一个路由器上定义的对等组的详细情况，如例 3-125 所示。还可以在该命令的最后指定一个对等组的名字，从而去观察一个单一对等组的详细情况。

例 3-125　show ip bgp peer-groups 命令显示了一个路由器上对等组的详细情况

```
Colorado#show ip bgp peer-group
BGP neighbor is CLIENTS, peer-group leader
 Index 1, Offset 0, Mask 0x2
  BGP version 4
  Minimum time between advertisement runs is 5 seconds
  Incoming update AS path filter list is 2
  Outgoing update AS path filter list is 1

BGP neighbor is LOCAL, peer-group leader,  remote AS 100
 Index 0, Offset 0, Mask 0x0
  NEXT_HOP is always this router
  BGP version 4
  Minimum time between advertisement runs is 5 seconds
  Outgoing update AS path filter list is 3
Colorado#
```

3.3.2　案例分析：BGP 团体

对等组可以使你对一组邻居使用共同的策略，而团体使你可以向一组路由使用相同的策略。团体是一个路由属性而且可以从一个运行 BGP 的路由器传给另外一个运行 BGP 的路由器。

配置一个团体属性，可以遵循下面的三个步骤：

步骤 1　通过路由图来识别已经设置了该属性的路由。

步骤 2　通过 **set community** 命令来设置属性。

步骤 3　使用 **neighbor send-community** 命令来明确该属性将要发送给哪个路由器。

在图 3-26 中，AS 100 通过 NAP 与 AS 2000 相连。AS 2000 内的一个路由策略声明应该将子网 10.2.2.0/24 公布给 AS 100，但是不能公布给任何与 Colorado 相连的 EBGP 对端。为了执行这个策略，就用到了 NO_EXPORT 团体属性。该属性允许在一个邻居 AS 内公布一条路由，但是不允许该 AS 将这条路由公布给其他的 AS。例 3-126 给出路由器 Austria 的配置。

例 3-127 给出了例 3-126 配置的结果，如例中所示，到 10.2.2.0/24 的路由设置了 NO_EXPORT 属性，因此 Colorado 没有向它的 EBGP 对端，例如 Nevada，公布这条路由。Colorado 的 BGP 表包含了到 10.2.2.0/24 的路由，暗示 Idaho 已经向它的 IBGP 对端公布了该路由，但是 Nevada 的 BGP 表中没有包含该路由，说明 Colorado 已经接受了 NO_EXPORT 属性并且向它的 EBGP 对端抑制了这条路由。

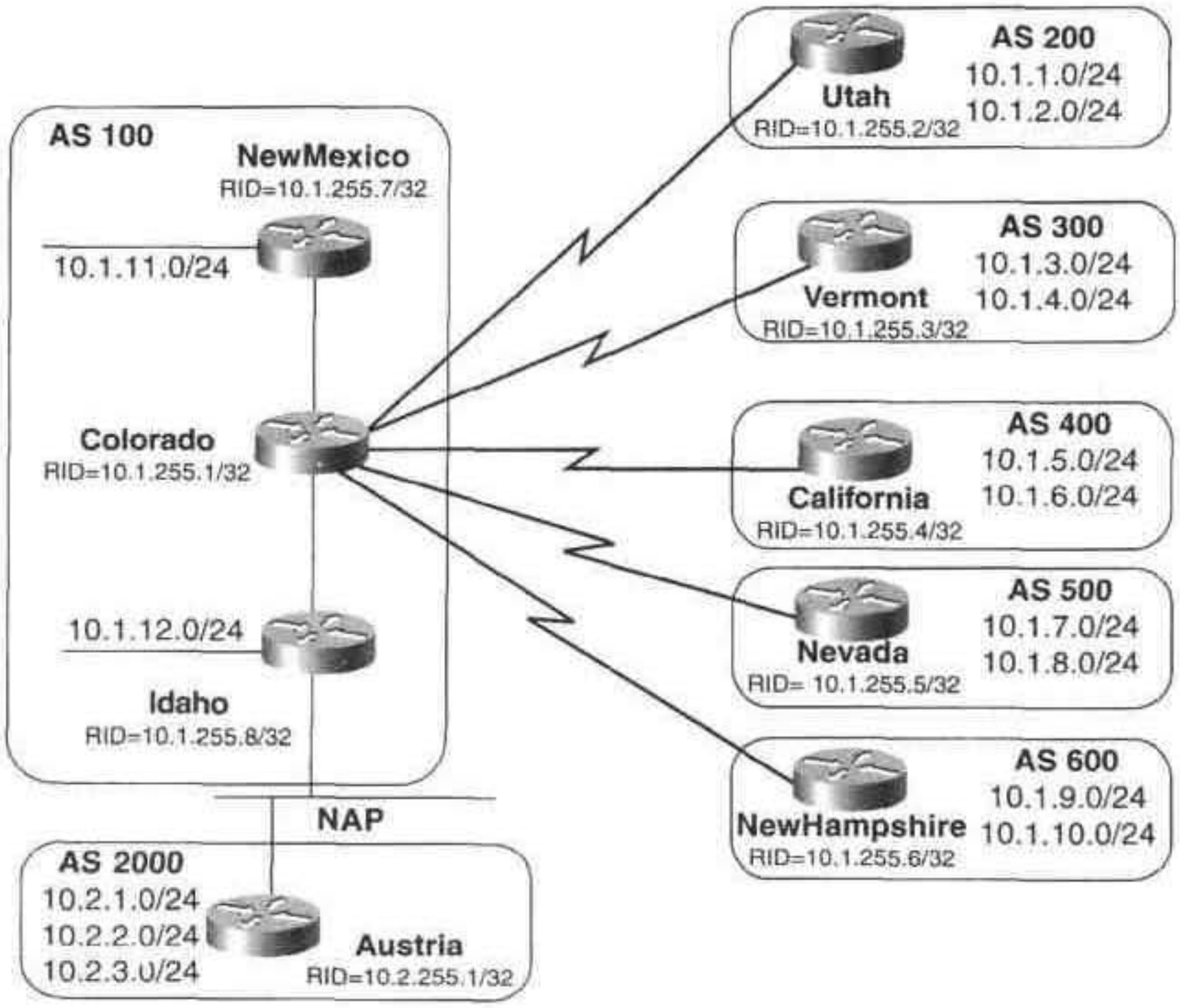


图 3-26　BGP 团体案例分析的网络拓扑

例 3-126　限制 AS 2000 内的一个子网在 AS 100 范围以外的地方公布

```
router bgp 2000
 network 10.2.1.0 mask 255.255.255.0
 network 10.2.2.0 mask 255.255.255.0
 network 10.2.3.0 mask 255.255.255.0
 neighbor 10.1.255.8 remote-as 100
 neighbor 10.1.255.8 ebgp-multihop 2
 neighbor 10.1.255.8 update-source Loopback0
 neighbor 10.1.255.8 send-community
 neighbor 10.1.255.8 route-map AUSTRIA out
 no auto-summary
!
access-list 1 permit 10.2.2.0
!
route-map AUSTRIA permit 10
 match ip address 1
 set community no-export
!
route-map AUSTRIA permit 20
```

当然，如果其他 AS 能够告诉一个 AS 应该做什么，那么该 AS 并不是真正的自治。假设 AS 100 想忽略由 Austria 设置的 NO_EXPORT 属性并且向 Colorado 的 EBGP 对等公布 10.2.2.0/24。例 3-128 给出了为了执行这样一个策略，Idaho 应该具有的配置。

例 3-127　10.2.2.0 路由有 NO_EXPORT 团体属性

```
Colorado#show ip bgp 10.2.2.0
BGP routing table entry for 10.2.2.0/24, version 42
Paths: (1 available, best #1, not advertised to EBGP peer)
  2000
    10.1.255.8 from 10.1.255.8
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Community: no-export
Colorado#
```

```
Nevada#show ip bgp 10.2.2.0
% Network not in table
Nevada#
```

例 3-128　配置 Idaho，删除由 AS 2000 设置的团体属性

```
router bgp 100
 no synchronization
 network 10.1.12.0 mask 255.255.255.0
 neighbor 10.1.255.1 remote-as 100
 neighbor 10.1.255.1 update-source Loopback0
 neighbor 10.1.255.1 next-hop-self
 neighbor 10.1.255.1 route-map IDAHO out
 neighbor 10.1.255.7 remote-as 100
 neighbor 10.1.255.7 update-source Loopback0
 neighbor 10.2.255.1 remote-as 2000
 neighbor 10.2.255.1 ebgp-multihop 2
 neighbor 10.2.255.1 update-source Loopback0
 no auto-summary
!
access-list 1 permit 10.2.2.0
!
route-map IDAHO permit 10
 match ip address 1
 set community none
!
route-map IDAHO permit 20
```

Idaho 配置中的 **set community none** 命令并不是设置一个团体属性而是删除现存的团体属性。这也是在这个配置中没有出现 **neighbor send-community** 命令的原因。例 3-129 给出了 Colorado 和 Nevada 处相应的结果，如例中所示，Colorado 没有看到到 10.2.2.0/24 的路由中有 NO_EXPORT 属性，因此它向它的 EBGP 对端公布了该路由。

例 3-129　Colorado 没有看到到 10.2.2.0/24 的路由中有 NO_EXPORT 属性，宣告到 EBGP 对端

```
Colorado#show ip bgp 10.2.2.0
BGP routing table entry for 10.2.2.0/24, version 90
Paths: (1 available, best #1, advertised over EBGP)
  2000
    10.1.255.8 from 10.1.255.8
      Origin IGP, metric 0, localpref 100, valid, internal, best
Colorado#
```

```
Nevada#show ip bgp 10.2.2.0
BGP routing table entry for 10.2.2.0 255.255.255.0, version 325
Paths: (1 available, best #1)
  100 2000
    10.1.255.1 from 10.1.255.1
      Origin IGP, valid, external, best
Nevada#
```

NO_ADVERTISE 团体属性与 NO_EXPORT 命令发送的消息相同——它告诉路由器不要向它的任何对端公布设置了该属性的路由。不同之处在于 NO_ADVERTISE 是发送给 IBGP 对端而不是 EBGP 对等。假设图 3-26 中的 Idaho 希望将子网 10.2.1.0/24 和 10.2.3.0/24 公布给 Colorado，但是不希望 Colorado 把这两条路由公布给它的任何 IBGP 或者 EBGP 对端，例 3-130 给出了为了达到这个目的，Idaho 的配置。

例 3-130　在 Idaho 处为选中的前缀设置 NO_ADVERTISE 属性

```
router bgp 100
 no synchronization
 network 10.1.12.0 mask 255.255.255.0
 neighbor 10.1.255.1 remote-as 100
 neighbor 10.1.255.1 update-source Loopback0
 neighbor 10.1.255.1 next-hop-self
 neighbor 10.1.255.1 send-community
 neighbor 10.1.255.1 route-map IDAHO out
 neighbor 10.1.255.7 remote-as 100
 neighbor 10.1.255.7 update-source Loopback0
 neighbor 10.2.255.1 remote-as 2000
 neighbor 10.2.255.1 ebgp-multihop 2
 neighbor 10.2.255.1 update-source Loopback0
 no auto-summary
!
ip as-path access-list 2 permit ^2000$
!
access-list 1 permit 10.2.2.0
!
route-map IDAHO permit 10
 match ip address 1
 set community none
!
route-map IDAHO permit 20
 match as-path 2
 set community no-advertise
!
route-map IDAHO permit 30
```

回想一下，曾经配置 Austria，让它以 NO_EXPORT 团体属性来公布 10.2.2.0/24，而 AS 2000 的另外两个子网没有携带团体属性。Idaho 现在把这个策略完全颠倒了过来。子网 10.2.2.0/24 没有团体属性，而 10.2.1.0/24 和 10.2.3.0/24 带有 NO_ADVERTISE 属性，从而使它们不会在 AS 100 以外的范围内被公布。例 3-131 给出了这种配置相应的结果。

注：　例 3-131 还给出了 **show ip bgp** 命令的另外一种用法。这里，通过一个正则表达式来显示所有在 AS_PATH 中包含 2000 的路由。

一个名为 LOCAL_AS 的团体属性在某种程度上讲是 NO_EXPORT 和 NO_ADVERTISE 的混合。这个属性需要与 BGP 联盟结合起来使用，在联盟中，是用同一个 AS 来配置子自治系统。可以将一个具有 LOCAL_AS 属性的路由公布给联盟内的其他子自治系统但是不能在构成联盟的 AS 以外进行公布。

个运行 BGP 的路由器会自动遵循前面已经说明的众所周知的团体属性。但是，你也可以配置只具有你所定义的含义的团体属性。可以通过两种方式来指定团体：

- 十进制格式，使用 1~4294967200 之间的数字。
- AA:NN 格式，这个格式中，AA 是 1~65535 之间的 16 比特 AS 号，NN 是 1~65440 之间的任意一个 16 比特数。

例 3-131　Colorado 知道来自 AS 2000 的三条路由，但是它向它的对端公布了 10.2.2.0/24

```
Colorado#show ip bgp regexp 2000
BGP table version is 138, local router ID is 10.1.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i10.2.1.0/24      10.1.255.8               0    100      0 2000 i
*>i10.2.2.0/24      10.1.255.8               0    100      0 2000 i
*>i10.2.3.0/24      10.1.255.8               0    100      0 2000 i
Colorado#
```

```
Nevada#show ip bgp regexp 2000
BGP table version is 355, local router ID is 10.1.255.5
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.2.2.0/24      10.1.255.1                             0 100 2000 i
Nevada#
```

在图 3-26 中，AS 100 的每个“客户”AS 都有两个子网。假设 AS 100 给每个客户 AS 两个子网中的一个实施了一个特定的策略，而给另外一个子网实施了不同的策略。可以在 Colorado 处使用一个冗长的访问列表，通过每条路由的 NLRI(记住这不是只有 5 个客户 AS 的情况，而是有 150 个客户 AS)来识别它们从而执行这些策略。另外一个办法就是让每个客户 AS 将两个子网分别分配到两个预先设定的团体中。例如，每个客户可以将一个子网分配给团体 5，而将另一个子网分配给团体 10。例 3-132 给出了 Utah 应有的配置。

例 3-132　在 Utah 将子网分配给不同团体

```
router bgp 200
 network 10.1.1.0 mask 255.255.255.0
 network 10.1.2.0 mask 255.255.255.0
 neighbor 10.1.255.1 remote-as 100
 neighbor 10.1.255.1 ebgp-multihop 2
 neighbor 10.1.255.1 update-source Loopback0
 neighbor 10.1.255.1 send-community
 neighbor 10.1.255.1 route-map UTAH out
 no auto-summary
!
access-list 1 permit 10.1.1.0
access-list 2 permit 10.1.2.0
!
route-map UTAH permit 10
 match ip address 1
 set community 5
!
route-map UTAH permit 20
 match ip address 2
 set community 10
```

Colorado 其他所有的 EBGP 对端都有相似的配置，如例 3-133 所示。除了使 Colorado 能够更容易识别采用不同策略的路由，这种方法还给客户 AS 的管理者留有一定的余地，使他们可以决定对不同的路由所采用的策略。

例 3-133　由 Colorado 的 EBGP 对端公布的每条路由都是团体 5 或者团体 10 的一个成员

```
Colorado#show ip bgp community 5
BGP table version is 60, local router ID is 10.1.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      10.1.255.2               0             0 200 i
*> 10.1.3.0/24      10.1.255.3               0             0 300 i
*> 10.1.5.0/24      10.1.255.4               0             0 400 i
*> 10.1.7.0/24      10.1.255.5               0             0 500 i
*> 10.1.9.0/24      10.1.255.6               0             0 600 i
Colorado#show ip bgp community 10
BGP table version is 60, local router ID is 10.1.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.2.0/24      10.1.255.2               0             0 200 i
*> 10.1.4.0/24      10.1.255.3               0             0 300 i
*> 10.1.6.0/24      10.1.255.4               0             0 400 i
*> 10.1.8.0/24      10.1.255.5               0             0 500 i
*> 10.1.10.0/24     10.1.255.6               0             0 600 i
Colorado#
```

一个团体列表用于通过路由的团体属性来识别路由。这个列表是访问列表的一个特殊用法：这个列表可能会有多行内容，每一行都有一个“允许”或者“拒绝”标识。列表由 1~99 之间的一个数字来确定，在每个列表的最后都有一个隐含地“拒绝任何一个”。在例 3-134 的配置中，Colorado 根据路由的团体值，通过团体列表来为这些路由分配 LOCAL_PREF 属性。

> 注：　如果客户 AS 是多宿主到 AS 100，那么该策略是很有用的。为了简化内容，在本例中没有给出这样的拓扑。

例 3-134　根据路由的团体值，通过团体列表为路由分配 LOCAL_PREF 属性

```
router bgp 100
 no synchronization
 network 10.1.11.0 mask 255.255.255.0
 network 10.1.12.0 mask 255.255.255.0
 neighbor CLIENTS peer-group
 neighbor CLIENTS ebgp-multihop 2
 neighbor CLIENTS update-source Loopback2
 neighbor CLIENTS next-hop-self
 neighbor CLIENTS send-community
 neighbor CLIENTS route-map COMM_PREF in
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 100
 neighbor LOCAL next-hop-self
 neighbor 10.1.255.2 remote-as 200

 neighbor 10.1.255.2 peer-group CLIENTS
 neighbor 10.1.255.3 remote-as 300
```

```
 neighbor 10.1.255.3 peer-group CLIENTS
 neighbor 10.1.255.4 remote-as 400
 neighbor 10.1.255.4 peer-group CLIENTS
 neighbor 10.1.255.5 remote-as 500
 neighbor 10.1.255.5 peer-group CLIENTS
 neighbor 10.1.255.6 remote-as 600
 neighbor 10.1.255.6 peer-group CLIENTS
 neighbor 10.1.255.7 peer-group LOCAL
 neighbor 10.1.255.8 peer-group LOCAL
 no auto-summary
!
ip community-list 1 permit 5
ip community-list 2 permit 10
!
route-map COMM_PREF permit 10
 match community 1
 set local-preference 150
!
route-map COMM_PREF permit 20
 match community 2
 set local-preference 200
```

将来自对等组 CLIENTS 成员的入站路由发送给一个名为 COMM_PREF 的路由图。路由图的序列 10 用团体列表 1 识别团体值为 5 的路由并且给它们分配一个值 150 的 LOCAL_PREF。序列 20 用团体 2 识别团体值为 10 的路由并且为它们分配一个值为 200 的 LOCAL_PREF。例 3-135 在 Colorado 的 BGP 表中给出了相应的结果。

例 3-135　团体 5 的路由 LOCAL_PREF 为 150，团体 10 的路由的 LOCAL_PREF 为 200

```
Colorado#show ip bgp
BGP table version is 16, local router ID is 10.1.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      10.1.255.2               0    150      0 200 i
*> 10.1.2.0/24      10.1.255.2               0    200      0 200 i
*> 10.1.3.0/24      10.1.255.3               0    150      0 300 i
*> 10.1.4.0/24      10.1.255.3               0    200      0 300 i
*> 10.1.5.0/24      10.1.255.4               0    150      0 400 i
*> 10.1.6.0/24      10.1.255.4               0    200      0 400 i
*> 10.1.7.0/24      10.1.255.5               0    150      0 500 i
*> 10.1.8.0/24      10.1.255.5               0    200      0 500 i
*> 10.1.9.0/24      10.1.255.6               0    150      0 600 i
*> 10.1.10.0/24     10.1.255.6               0    200      0 600 i
*>i10.1.11.0/24     10.1.255.7               0    100      0 i
*>i10.1.12.0/24     10.1.255.8               0    100      0 i
*>i10.2.1.0/24      10.1.255.8               0    100      0 2000 i
*>i10.2.2.0/24      10.1.255.8               0    100      0 2000 i
*>i10.2.3.0/24      10.1.255.8               0    100      0 2000 i
Colorado#
```

在描述 BGP 团体的 RFC 1997 和 RFC 1998 中，说明了 AA:NN 格式的使用情况。在缺省的情况下，Cisco IOS 使用旧的、十进制的格式。为了使用新的格式，需要在所有路由器的配置中加入 **ip bgp-community new-format** 命令。在这种格式下加入团体时，可以直接在 AA:NN 格式中键入团体值，可以是十六进制也可以是十进制。例如，下面的任何一个条目都表示团体 400:50(AS 400，团体号为 50)：

- **set community　400:50**
- **set community　0x1900032**
- **set community　26214450**

所有这些命令都描述了一个 32 比特的数，前 16 比特是以十进制表示的 400，后 16 比特是以十进制表示的 50。不管使用了这 3 个命令中的哪一个，在路由器配置文件以及 BGP 表中显示的团体都是 400:50。

例 3-136 显示了以 AA：NN 格式来配置 Utah

例 3-136　配置 Utah，使它以 AA:NN 格式显示团体

```
router bgp 200
 network 10.1.1.0 mask 255.255.255.0
 network 10.1.2.0 mask 255.255.255.0
 neighbor 10.1.255.1 remote-as 100
 neighbor 10.1.255.1 ebgp-multihop 2
 neighbor 10.1.255.1 update-source Loopback0
 neighbor 10.1.255.1 send-community
 neighbor 10.1.255.1 route-map UTAH out
 no auto-summary
!
ip bgp-community new-format
!
access-list 1 permit 10.1.1.0
access-list 2 permit 10.1.2.0
!
route-map UTAH permit 10
 match ip address 1
 set community 200:5
!
route-map UTAH permit 20
 match ip address 2
 set community 200:10
```

正如访问列表有标准以及扩展两种方式一样，团体列表也有标准和扩展两种形式。和 IP 访问列表相似，标准的团体列表是从 1~99 进行编号，扩展的团体列表从 100~199 进行编号。两种团体之间的区别就在于扩展列表允许你用正则表达式来指定团体(在使用 AA:NN 格式时，这种用法十分有用)。为了在 Colorado 执行前面描述过的同样的 LOCAL_PREF 策略，但是使用 AA:NN 格式，按照例 3-137 所示的来配置 Colorado。

例 3-137　使用扩展的团体列表

```
router bgp 100
 no synchronization
 network 10.1.11.0 mask 255.255.255.0
 network 10.1.12.0 mask 255.255.255.0
 neighbor CLIENTS peer-group
 neighbor CLIENTS ebgp-multihop 2
 neighbor CLIENTS update-source Loopback2
 neighbor CLIENTS next-hop-self
 neighbor CLIENTS send-community
 neighbor CLIENTS route-map COMM_PREF in
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 100
 neighbor LOCAL next-hop-self
 neighbor 10.1.255.2 remote-as 200
```

```
 neighbor 10.1.255.2 peer-group CLIENTS
 neighbor 10.1.255.3 remote-as 300
 neighbor 10.1.255.3 peer-group CLIENTS
 neighbor 10.1.255.4 remote-as 400
 neighbor 10.1.255.4 peer-group CLIENTS
 neighbor 10.1.255.5 remote-as 500
 neighbor 10.1.255.5 peer-group CLIENTS
 neighbor 10.1.255.6 remote-as 600
 neighbor 10.1.255.6 peer-group CLIENTS
 neighbor 10.1.255.7 peer-group LOCAL
 neighbor 10.1.255.8 peer-group LOCAL
 no auto-summary
!
ip community-list 101 permit .*:5
ip community-list 102 permit .*:10
!
route-map COMM_PREF permit 10
 match community 101
 set local-preference 150
!
route-map COMM_PREF permit 20
 match community 102
 set local-preference 200
```

除了在例 3-137 的配置中使用了扩展团体列表以外，本例中的配置与前面例 3-134 中给出的配置相同。如果使用标准的团体列表，需要单独的一行从而与每个 AS 的每个团体相匹配，如例 3-138 中的描述。

例 3-138　使用标准的团体列表

```
ip community-list 1 permit 200:5
ip community-list 1 permit 300:5
ip community-list 1 permit 400:5
ip community-list 1 permit 500:5
ip community-list 1 permit 600:5
```

如果使用扩展的团体列表，需用一行来指定一个匹配。正则表达式.*与任何一个 AS 匹配，5 与团体号中的通用部分相匹配。

例 3-139 显示了在 Colorado 处与团体 102 和 102 相匹配的路由。在可能有几万条 BGP 路由条目的大型 BGP 的执行过程中，团体列表与 **show ip bgp community-list** 命令的结合变得非常重要。利用独特的团体属性来寻找路由成了一种非常简单的事情，只要输入一个团体列表就可以显示与该列表相匹配的路由。

例 3-139　**show ip bgp community-list** 命令显示了与指定团体列表相匹配的 BGP 路由

```
Colorado#show ip bgp community-list 101
BGP table version is 19, local router ID is 10.1.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      10.1.255.2               0    150      0 200 i
*> 10.1.3.0/24      10.1.255.3               0    150      0 300 i
*> 10.1.5.0/24      10.1.255.4               0    150      0 400 i
*> 10.1.7.0/24      10.1.255.5               0    150      0 500 i
*> 10.1.9.0/24      10.1.255.6               0    150      0 600 I

Colorado#show ip bgp community-list 102
BGP table version is 19, local router ID is 10.1.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
```

```
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.2.0/24      10.1.255.2               0    200      0 200 i
*> 10.1.4.0/24      10.1.255.3               0    200      0 300 i
*> 10.1.6.0/24      10.1.255.4               0    200      0 400 i
*> 10.1.8.0/24      10.1.255.5               0    200      0 500 i
*> 10.1.10.0/24     10.1.255.6               0    200      0 600 i
Colorado#
```

一条路由可以有多个团体属性。假设图 3-26 中的路由器 Austria 用团体 2000:100 将它所有的子网公布给 Idaho。在 Idaho，来自 Austria 的路由又成了团体 100:2000 的成员。例 3-140 给出了 Idaho 的配置。

例 3-140　配置 Idaho，使它为来自 Austria 的路由加入另外的团体值

```
router bgp 100
 no synchronization
 network 10.1.12.0 mask 255.255.255.0
 neighbor 10.1.255.1 remote-as 100
 neighbor 10.1.255.1 update-source Loopback0
 neighbor 10.1.255.1 next-hop-self
 neighbor 10.1.255.1 send-community
 neighbor 10.1.255.1 route-map IDAHO out
 neighbor 10.1.255.7 remote-as 100
 neighbor 10.1.255.7 update-source Loopback0
 neighbor 10.1.255.7 next-hop-self
 neighbor 10.2.255.1 remote-as 2000
 neighbor 10.2.255.1 ebgp-multihop 2
 neighbor 10.2.255.1 update-source Loopback0
 no auto-summary
!
ip as-path access-list 2 permit ^2000$
!
route-map IDAHO permit 10
 match as-path 2
 set community 6555600 additive
!
route-map IDAHO permit 20
```

这个配置有两点很有意思的地方。第一，路由图 IDAHO 的序列 10 设置的团体是 6555600。IOS 12.0 以及以后的版本会支持 **ip bgp-community new-format** 命令；路由器 Idaho 运行的是 IOS 11.0 因此它不能理解 AA:NN 的格式。但是它会进行一个快速的计算并显示十进制数 6555600 等同于 32 比特数 100:2000(或者 0x6407d0)。而 Colorado 运行 IOS 12.0，因此它能够正确地理解 AA:NN 格式中的 32 比特数。在这儿重要的一点就是虽然可以用 AA:NN、十进制或者十六进制来表示团体属性，但是实际上它还是一个 32 比特的数。

另外一个有趣的地方是与 **set community** 命令一起使用的关键词 **additive**，如果单独使用 **set community 6555600** 命令而没有 **additive** 关键词，Idaho 会用团体 100:2000 取代任何匹配路由现存的团体属性。这个例子的目的是再另加一个团体属性而不是取代 Austria 发送过来的团体。例 3-141 给出 Colorado 处相应的结果。

当一条路由具有多个团体的时候，团体列表和匹配命令的使用略微有些不同。**ip community-list 1 permit 2000:100** 命令与把 2000:100 作为一个团体的任何路由相匹配。另一

方面，**ip community-list 1 permit 2000:100 100:2000** 与含有这两个团体中的任何一个或者两个全都都含有的路由相匹配。如果你只想与是 2000:100 和 100:2000 这两个团体成员的路由相匹配，那么，如例 3-142 所示，可以在匹配声明中使用关键词 **exact-match**。通过这个关键词，只含有 2000:100 或者 100:2000 的路由，以及含有这两个团体但是同时还含有其他团体的路由，将不再符合匹配要求。

例 3-141　团体属性 2000:100 是 Austria 加入的；而团体属性 100:2000 是 Idaho 加入的

```
Colorado#show ip bgp 10.2.1.0
BGP routing table entry for 10.2.1.0/24, version 49
Paths: (1 available, best #1, advertised over EBGP)
  2000
    10.1.255.8 from 10.1.255.8
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Community: 100:2000 2000:100
Colorado#
```

例 3-142　关键词 **exact-match** 给出了与指定的团体列表完全匹配的路由

```
Colorado#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Colorado(config)#ip community-list 10 permit 2000:100 100:2000
Colorado(config)#^Z
Colorado#
%SYS-5-CONFIG_I: Configured from console by console
Colorado#show ip bgp community-list 10 exact-match
BGP table version is 52, local router ID is 10.1.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i10.2.1.0/24      10.1.255.8               0    100      0 2000 i
*>i10.2.2.0/24      10.1.255.8               0    100      0 2000 i
*>i10.2.3.0/24      10.1.255.8               0    100      0 2000 i
Colorado#
```

如果可以为一条路由分配多个团体属性，那么就应该有办法删除一些团体属性而不是像 **set community none** 命令那么删除所有的属性。通过 **set community-list delete** 命令可以完成这个工作。

在前面的匹配中，到图 3-26 中 AS 2000 内子网的路由具有 2000:100 和 100:2000 的团体属性。在 AS 400 中，应该保留团体 100:2000 而不是 2000:100。例 3-143 给出了 California 的配置。

路由图 DROP_COMM 涉及到了团体列表 1 并且删除了由该列表指定的团体。例 3-144 给出了结果,如例中所示,在 California 的 BGP 列表中没有找到同时含有 2000:100 和 100:2000 团体属性的路由，但是，来自 AS 2000 的路由都含有单个的团体属性 100:2000。在前面的配置中，一个团体列表行可以指定多个团体属性。但是，当 **set community-list delete** 命令使用该列表时，列表的每一行只能指定一个团体。因此，如果想从路由中删除团体 2000:100、

NO_EXPORT 和 300:5，你必须通过三个独立的行来配置这个团体列表。

例 3-143　配置 California，让它有选择地删除一些团体值

```
router bgp 400
 network 10.1.5.0 mask 255.255.255.0
 network 10.1.6.0 mask 255.255.255.0
 neighbor 10.1.255.1 remote-as 100
 neighbor 10.1.255.1 ebgp-multihop 2
 neighbor 10.1.255.1 update-source Loopback0
 neighbor 10.1.255.1 send-community
 neighbor 10.1.255.1 route-map DROP_COMM in
!
ip bgp-community new-format
!
ip community-list 1 permit 2000:100
!
route-map DROP_COMM permit 10
 set comm-list 1 delete
```

例 3-144　California 的 BGP 表中没有 2000:100 的团体属性，却有 100:2000

```
California#show ip bgp community 2000:100 100:2000 exact-match
California#show ip bgp community 100:2000 exact-match
BGP table version is 16, local router ID is 10.1.255.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.2.1.0/24      10.1.255.1                             0 100 2000 i
*> 10.2.2.0/24      10.1.255.1                             0 100 2000 i
*> 10.2.3.0/24      10.1.255.1                             0 100 2000 i
California#
```

3.3.3　案例分析：专用 AS 号

你一定会熟悉在 RFC 1918 中规定的专用 IP 地址。这些地址的范围是 10.0.0.0～10.255.255.255、172.16.0.0～172.31.255.255 以及 192.168.0.0～192.168.255.255，设计这些地址的目的是为了缓和 IP 地址的耗尽的速度。当一个互联网络需要一个 IP 地址，但是没有必要公开这个地址的时候(也就是说，不需要通过 Internet 来访问它们)，就可以使用专用地址而不使用公开地址。因为任何人都可以使用任何一个专用 IP 地址，因此这些地址并不是唯一的而且永远也不能将这些地址公布给 Internet。

同样也存在着专用的 AS 号，与专用 IP 地址一样，设计专用 AS 号的目的也是为了缓和公用 AS 号的耗尽。从 64512 到 65535 的 AS 号预留作为专用。如果一个运行 BGP 的用户只与一个 ISP 相连，那么该用户可以而且应该被鼓励使用专用 AS 号。

例如，在前面的案例分析中，如图 3-26 所示，把与路由器 Colorado 相连的 AS 看作是 AS 100 的“客户”AS。通过一个 NAP 与 AS 100 相连的 AS 2000 代表 Internet。AS 100 可能是一个 ISP 而与它相连的 AS 就是它的用户，或者 AS 100 可能是一个大型企业网与公网互连的部分，而其他的 AS 是它的专用部分。但是不管是哪种情况，图 3-26 中的五个“客户”AS

都只能通过 NAP 以及 AS 100 才能互相到达。它们有独立的 AS 号是因为可以通过 EBGP 将它们与 AS 100 相连；而 AS 可以将它们的路由公布给 Internet 而不必包括它们的 AS 号。图 3-27 给出了与图 3-26 一样的网络，但是，此时，“客户”AS 使用了专用 AS 池以内的 AS 号。

记住，专用 AS 号和专用 IP 地址一样都不能公布给 Internet，因为它们不是唯一的。例 3-145 给出了在没有更多配置的情况下，AS 100 的客户 AS 号通过 NAP 公布给路由器 Austria，在 Austria 的 BGP 表中可以看到，来自 AS 100 的一些路由的 AS_PATH 中包含了专用 AS 号；如果 Austria 是 Internet 的一部分，它通过 NAP 与 AS 100 相连，那么在这些路由中不应该包括专用 AS 号。

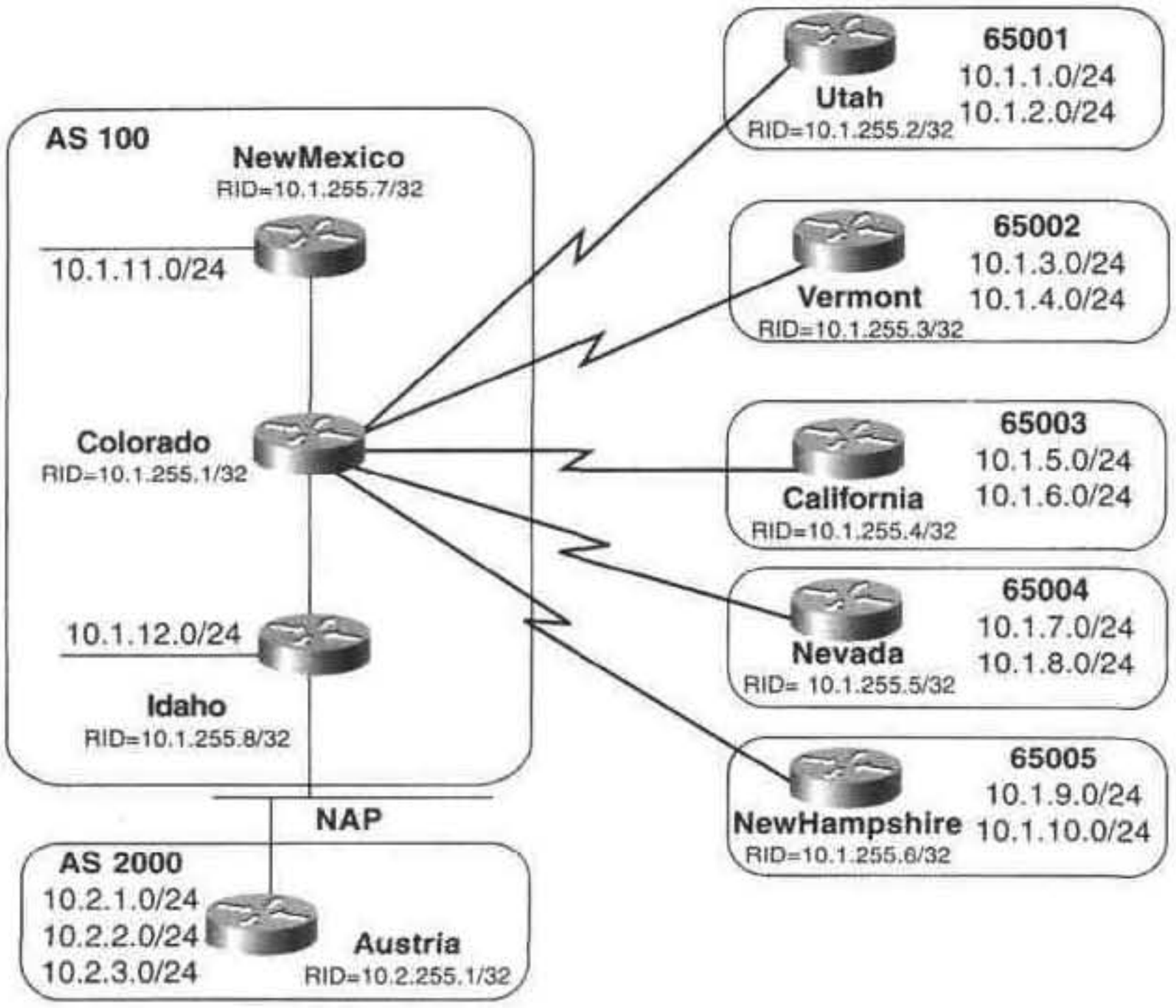


图 3-27　AS 通过专用 AS 号与 Colorado 相连

例 3-145　AS 100 的客户 AS 号通过 NAP 公布给路由器 Austria

```
Austria#show ip bgp
BGP table version is 189, local router ID is 10.2.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      10.1.255.8                             0 100 65001 i
*> 10.1.2.0/24      10.1.255.8                             0 100 65001 i
*> 10.1.3.0/24      10.1.255.8                             0 100 65002 i
*> 10.1.4.0/24      10.1.255.8                             0 100 65002 i
*> 10.1.5.0/24      10.1.255.8                             0 100 65003 i
*> 10.1.6.0/24      10.1.255.8                             0 100 65003 i
*> 10.1.7.0/24      10.1.255.8                             0 100 65004 i
*> 10.1.8.0/24      10.1.255.8                             0 100 65004 i
```

```
*> 10.1.9.0/24       10.1.255.8                             0 100 65005 i
*> 10.1.10.0/24      10.1.255.8                             0 100 65005 i
*> 10.1.11.0/24      10.1.255.8                             0 100 i
*> 10.1.12.0/24      10.1.255.8               0             0 100 i
*> 10.2.1.0/24       0.0.0.0                  0         32768 i
*> 10.2.2.0/24       0.0.0.0                  0         32768 i
*> 10.2.3.0/24       0.0.0.0                  0         32768 i
Austria#
```

在例 3-146 中，对 Idaho 进行配置，以阻止通过 NAP 公布专用 AS 号。

例 3-146　过滤掉专用 AS 号

```
router bgp 100
 no synchronization
 network 10.1.12.0 mask 255.255.255.0
 neighbor 10.1.255.1 remote-as 100
 neighbor 10.1.255.1 update-source Loopback0
 neighbor 10.1.255.1 next-hop-self
 neighbor 10.1.255.1 send-community
 neighbor 10.1.255.7 remote-as 100
 neighbor 10.1.255.7 update-source Loopback0
 neighbor 10.1.255.7 next-hop-self
 neighbor 10.2.255.1 remote-as 2000
 neighbor 10.2.255.1 ebgp-multihop 2
 neighbor 10.2.255.1 update-source Loopback0
 neighbor 10.2.255.1 remove-private-AS
 no auto-summary
```

neighbor remove-private-AS 命令很明显是在将路由公布给指定的邻居之前，从它们的 AS_PATH 中过滤掉专用 AS 号。在例 3-147 中，可以看到所有从 Idaho 公布给 Austria 的路由，现在它们的 AS_PATH 中只含有 AS 100。而 AS 100 内部的路由器还是含有 AS 100 客户 AS 的全部的路径信息并且能够将数据包转发给正确的目的地 AS。把客户子网作为 AS 100 的一部分进行公布是 AS 层的一种归纳形式。

例 3-147　在 Idaho 向 Austria 隐藏了 AS 100 客户自治系统的专用 AS 号

```
Austria#show ip bgp
BGP table version is 214, local router ID is 10.2.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      10.1.255.8                             0 100 i
*> 10.1.2.0/24      10.1.255.8                             0 100 i
*> 10.1.3.0/24      10.1.255.8                             0 100 i
*> 10.1.4.0/24      10.1.255.8                             0 100 i
*> 10.1.5.0/24      10.1.255.8                             0 100 i
*> 10.1.6.0/24      10.1.255.8                             0 100 i
*> 10.1.7.0/24      10.1.255.8                             0 100 i
*> 10.1.8.0/24      10.1.255.8                             0 100 i
*> 10.1.9.0/24      10.1.255.8                             0 100 i
*> 10.1.10.0/24     10.1.255.8                             0 100 i
*> 10.1.11.0/24     10.1.255.8                             0 100 i
*> 10.1.12.0/24     10.1.255.8               0             0 100 i
*> 10.2.1.0/24      0.0.0.0                  0         32768 i
*> 10.2.2.0/24      0.0.0.0                  0         32768 i
*> 10.2.3.0/24      0.0.0.0                  0         32768 i
Austria#
```

3.3.4　案例分析：BGP 联盟

BGP 联盟使 AS 管理者可以将大型转接 AS 分割成子自治系统，从而使大型 AS 更容易管理。被分割的 AS 本身是联盟，而那些子系统就是该 AS 的成员。在联盟以外的 AS 将整个联盟看作是一个 AS，它们看不到联盟内部的成员 AS。因为这些成员 AS 是对外隐藏的，因此它们既可以使用公共 AS 号也可以使用专用 AS 号，但是，实际中建议它们使用专用 AS 号。

联盟的优势是它们急剧地降低了 IBGP 会话的数量。通常在成员 AS 之间使用 IBGP，但是在联盟的成员 AS 之间运行着一个特殊版本的 EBGP，名为*联盟 EBGP*。在联盟内，一个 AS 的一个 BGP SPEAKER 与另外一个 AS 的一个 BGP SPEAKER 之间不会配置 IBGP 会话。

图 3-28 给出了一个联盟的例子。AS 1200 被分成了三个联盟子自治系统：AS 65533、AS 65534 和 AS 65535。在外部 AS，例如 AS 1000 和 AS 1500 看来，这个联盟是一个自治系统：AS 1200。外部 AS 无法看到联盟的成员 AS。

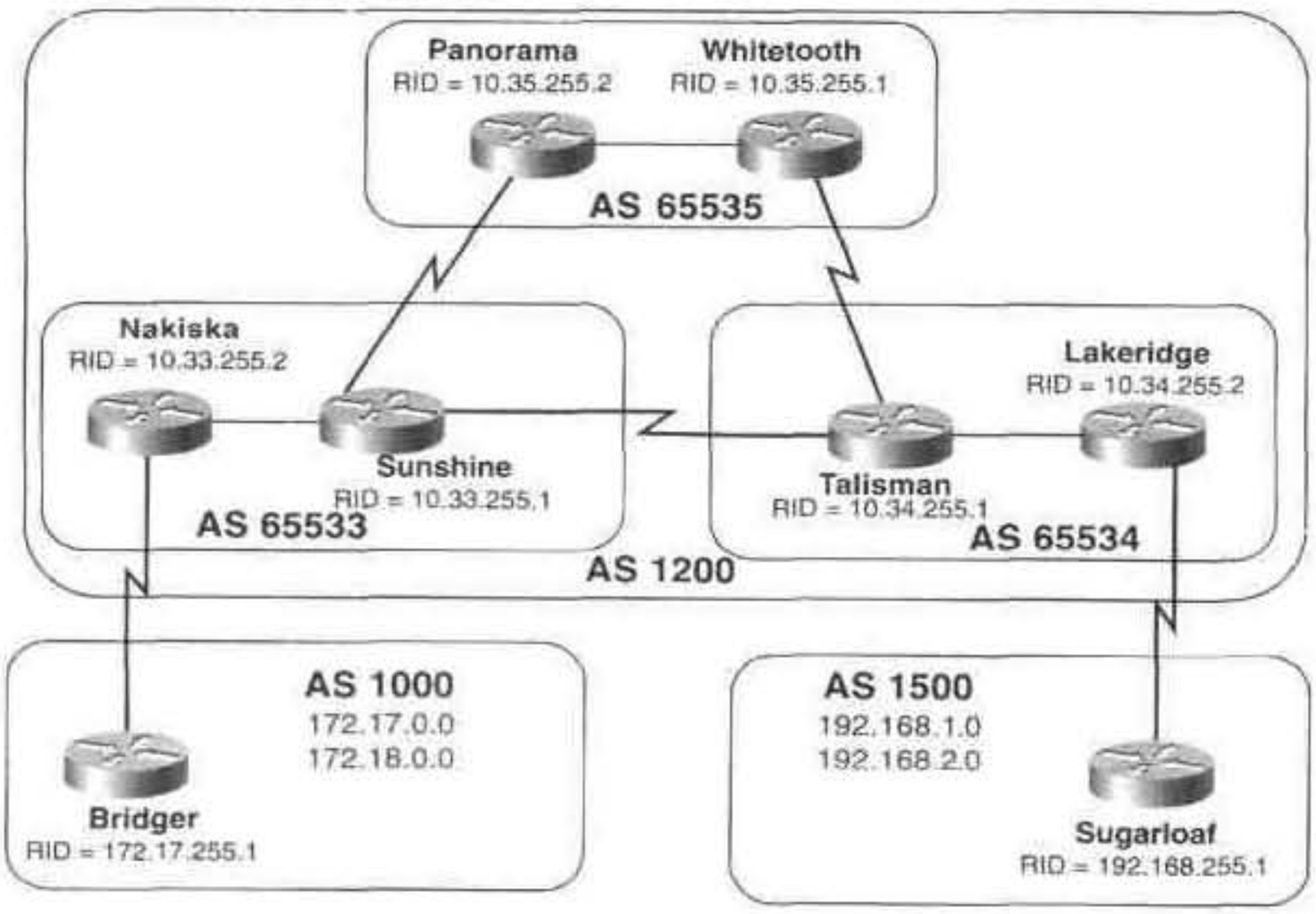


图 3-28　AS 1200 是一个 BGP 联盟：虽然它包含了几个子自治系统

在 Panorama 和 Sunshine 之间、Sunshine 和 Talisman 之间、Talisman 和 Whitetooth 之间运行的是联盟 EBGP。例 3-148 给出了 Talisman 的配置。

例 3-148　将 Talisman 配置为一个联盟路由器

```
router ospf 65534
 network 10.34.0.0 0.0.255.255 area 65534
 network 10.255.0.0 0.0.255.255 area 0
!
router bgp 65534
 no synchronization
 bgp confederation identifier 1200
 bgp confederation peers 65533 65535
 neighbor Confed peer-group
 neighbor Confed ebgp-multihop 2
 neighbor Confed update-source Loopback
```

```
neighbor Confed next-hop-self
neighbor MyGroup peer-group
neighbor MyGroup remote-as 65534
neighbor MyGroup update-source Loopback0
neighbor 10.33.255.1 remote-as 65533
neighbor 10.33.255.1 peer-group Confed
neighbor 10.34.255.2 peer-group MyGroup
neighbor 10.35.255.1 remote-as 65535
neighbor 10.35.255.1 peer-group Confed
```

Talisman 被配置为联盟路由器因此它的本地 AS 号为 65534。它与 Whitetooth 以及 Sunshine 之间连接的设置与其他的 EBGP 一样，它与 Lakeridge 之间的连接是 IBGP。**bgp confederation identifier** 命令告诉路由器它是联盟的一个成员以及它的联盟 id。**bpg confederation peers** 命令列出了与 Talisman 相连的联盟成员 AS，这个命令告诉 BGP 进程，此时的 EBGP 是联盟 EBGP 而不是通常的 EBGP。

一个联盟可以在整个联盟范围内只运行 BGP，一个通用的 IGP，或者在每个成员 AS 之间使用不同的 IGP。在图 3-28 中，AS 1200 内的所有的路由器都运行 OSPF。OSPF 允许在联盟内进行本地通信并且告诉 BGP 进程如何找到它们的邻居。在例 3-148 的配置中，在任何路由器处没有路由在 OSPF 和 BGP 之间进行再分发。在接下来的配置示例中没有给出 OSPF 配置。

例 3-149 显示了 Lakeridge 与 Sugarloaf 的配置。

例 3-149　在联盟路由器 Lakeridge 和外部路由器 Sugarloaf 之间配置 EBGP

Lakeridge

```
router bgp 65534
 no synchronization
 bgp confederation identifier 1200
 neighbor 10.34.255.1 remote-as 65534
 neighbor 10.34.255.1 update-source Loopback0
 neighbor 10.34.255.1 next-hop-self
 neighbor 192.168.255.1 remote-as 1500
 neighbor 192.168.255.1 ebgp-multihop 2
 neighbor 192.168.255.1 update-source Loopback0
```

Sugarloaf

```
router bgp 1500
 network 192.168.1.0
 network 192.168.2.0
 neighbor 10.34.255.2 remote-as 1200
 neighbor 10.34.255.2 ebgp-multihop 2
 neighbor 10.34.255.2 update-source Loopback0
```

因为 Lakeridge 没有运行联盟 EBGP，因此它没有使用 **bpg confederation peers** 命令。但是，它确实具有到 Sugarloaf 的普通 EBPG 连接。注意从 Sugarloaf 的角度来看，Lakeridge 属于 AS 1200，而不是属于 AS 65534。因为 Sugarloaf 在联盟的外面，因此它无法看到联盟的成员 AS。

联盟 EBGP 在某种程度上来讲，可以说是 BGP 和 IBGP 的一种混合。尤其是在联盟内部具有下面的一些特性：

- 在联盟内部保留联盟外部路由的 NEXT_HOP 属性。
- 公布给联盟的路由的 MULIT_EXIT_DISC 属性在整个联盟范围内予以保留。

- 路由的 LOCAL_PREF 属性在整个联盟范围内予以保留，而不只是在分配它们的成员 AS 内保留。
- 在联盟范围内，将成员自治系统的 AS 号加入到 AS_PATH 中，但是并不将它们公布到联盟的范围以外。缺省的情况下，将成员 AS 号作为 AS_PATH 属性类型 4 ——AS_CONFED_SEQUENCE 的形式在 AS_PATH 中列出来。如果在联盟内使用了 **aggregate-address** 命令，**as-set** 关键词使得在聚合点以后的成员 AS 号以 AS_PATH 属性类型 3 ——AS_CONFED_SET 的形式在 AS_PATH 中列出来。
- AS_PATH 中的联盟 AS 号用于避免出现路由环路，但是当在联盟内选择一个最短的 AS_PATH 时，可以不用考虑联盟 AS 号。

这些特点的大部分都是因为在联盟的外部看，这个联盟就是一个单独的自治系统。接下来对每个特点都分别给出了实例。

在图 3-28 中，AS 1000 中的路由从 Bridger 公布给 Nakiska，它的 NEXT_HOP 属性是 172.17.255.1。当通过 IBGP 从 Nakiska 向 Sunshine 公布这条路由时，保留了它的 NEXT_HOP 属性。如果 Sunshine 是通过普通的 EBGP 连接与 Talisman 相连，在将这条路由公布给 Talisman 之前，它会将 NEXT_HOP 属性改为 10.33.255.1，但是，因为它们之间的连接是联盟 EBGP，因此，该路由的初始 NEXT_HOP 属性被保留了下来。结果是，Lakeridge 到 172.17.0.0 和 172.18.0.0 的路由条目的下一跳地址是 172.17.255.1。Lakeridge 到 Sugarloaf 的连接是普通的 EBGP，因此公布给 Sugarloaf 的这条路由的 NEXT_HOP 属性是 10.34.255.2。

在图 3-28 的整个联盟内使用了 **neighbor next-hop-self** 命令，因此通过 IGP 可以了解到所有的下一跳地址。在 Talisman 和 Lakeridge 的配置中，可以看到这些命令。

配置 Bridger，使它公布路由的 MED 为 50，而 Nakiska 将同一条路由的 LOCAL_PREF 设置为 200。在例 3-150 中可以看到这些配置的结果。在普通的 EBGP 会话中，Sunshine 不会公布从 AS 1000 发起的 MED，或者只在 AS 65533 中适用的 LOCAL_PREF，但是，因为从联盟的外部来看，联盟是一个单独的 AS，因此在整个联盟范围内，这些值都应该是一致的。

例 3-150　联盟 EBGP 连接，保留了 AS 1000 的路由 MED 为 50，LOCAL_PREF 为 200

```
Lakeridge#show ip bgp
BGP table version is 28, local router ID is 10.34.255.2
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i172.17.0.0       10.33.255.1              50    200      0 (65533) 1000 i
*>i172.18.0.0       10.33.255.1              50    200      0 (65533) 1000 i
*> 192.168.1.0      192.168.255.1             0             0 1500 i
*> 192.168.2.0      192.168.255.1             0             0 1500 i
Lakeridge#
```

在例 3-150 中，还可以看到，到 AS 1000 中网络的路由的 AS_PATH 中包含了 AS 65533。AS_CONFED_SEQUENCE 出现在圆括号中，原因有两个。第一，如例 3-151 所示，在图 3-28 中，Sugarloaf 将联盟看作是一个 AS 并且它看不到成员 AS；在例 3-150 中出现在圆括号中的 AS_CONFED_SEQUENCE，在本例中被联盟 ID 1200 取代了，因此，不会将它公布到联盟的

外面。第二，在联盟内部它只用于避免路由环路，而不是用于路径选择。

例 3-151　联盟 ID 在对外部 AS 宣告路由时取代了 AS_CONFED_SEQUENCE

```
Sugarloaf#show ip bgp
BGP table version is 32, local router ID is 192.168.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 172.17.0.0       10.34.255.2                          0 1200 1000 i
*> 172.18.0.0       10.34.255.2                          0 1200 1000 i
*> 192.168.1.0      0.0.0.0                  0         32768 i
*> 192.168.2.0      0.0.0.0                  0         32768 i
Sugarloaf#
```

在例 3-152，Whitetooth 和 Panorama 的 BGP 表中，可以看到一个事实，就是成员 AS 号并没有影响到路径选择过程。对于 AS 1000 和 AS 1500 内的目的地，这两个路由器都有两条到这些目的地的路由——一条是通过它们的 IBGP 邻居，另外一条是通过它们的联盟 EBGP 邻居。例如，Whitetooth，有两条到网络 172.17.0.0 的路由，其中一条的 AS_PATH 是(65534，65533，1000)，另外一条是(65533，1000)。很显然，后面一条路由较短，但是成员 AS 号是忽略不计的，因此，这两条路由被看作是等同的：(1000)。如果其他所有的条件都相同，在存在普通 EBGP 路由和联盟 EBGP 路由的情况下，BGP 决定进程会优先选择普通 EBGP 路由，如果是联盟 EBGP 路由和 IBGP 路由同时存在，BGP 决定进程会优先选择联盟 EBGP 路由。例 3-152，是在联盟 EBGP 路由和 IBGP 路由之间作出选择，如例中所示，在一个 AS 联盟内部选择最短路径时，出现在 Whitetooth 和 Panorama 的 BGP 表内圆括号中的 AS_CONFED_SEQUENCE 不予考虑。

例 3-152　AS 联盟内部选择最短路径

```
Whitetooth#show ip bgp
BGP table version is 9, local router ID is 10.35.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 172.17.0.0       10.34.255.1              50     200      0 (65534 65533) 1000 i
* i                 10.33.255.1              50     200      0 (65533) 1000 i
*> 172.18.0.0       10.34.255.1              50     200      0 (65534 65533) 1000 i
* i                 10.33.255.1              50     200      0 (65533) 1000 i
*> 192.168.1.0      10.34.255.1               0     100      0 (65534) 1500 i
* i                 10.33.255.1               0     100      0 (65533 65534) 1500 i
*> 192.168.2.0      10.34.255.1               0     100      0 (65534) 1500 i
* i                 10.33.255.1               0     100      0 (65533 65534) 1500 i
Whitetooth#
```

```
Panorama#show ip bgp
BGP table version is 5, local router ID is 10.35.255.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i172.17.0.0       10.34.255.1              50     200      0 (65534 65533) 1000 i
*>                  10.33.255.1              50     200      0 (65533) 1000 i
* i172.18.0.0       10.34.255.1              50     200      0 (65534 65533) 1000 i
*>                  10.33.255.1              50     200      0 (65533) 1000 i
* i192.168.1.0      10.34.255.1               0     100      0 (65534) 1500 i
*>                  10.33.255.1               0     100      0 (65533 65534) 1500 i
* i192.168.2.0      10.34.255.1               0     100      0 (65534) 1500 i
*>                  10.33.255.1               0     100      0 (65533 65534) 1500 i
Panorama#
```

注意在这两个路由器的 BGP 表中，每个实例都选择了联盟 EBGP 路径。

在图 3-28 的拓扑中，忽略成员 AS 号不会带来任何麻烦，但是看一下图 3-29 中的拓扑，除了将 AS 65534 和 AS 65535 中的 BGP 路由器 ID 交换了以外，这两个拓扑完全一样。这个变化看起来没什么，但是，它会对 Sunshine 的 BGP 决定进程造成一定的影响。Talisman 和 Panorama 都会公布到 AS 1500 内网络的路由，因为忽略了成员 AS 号，因此这些路由的 AS_PATH 长度相同而且它们的邻居都是联盟 EBGP 对端。Talisman 和 Panorama 都使用了 **neighbor next-hop-self** 命令，因此到这两条路由的下一跳地址的 IGP 路径都是相同的，因此现在的关键问题就在于哪个相邻路由器 ID 最低，在本例中 Panorama 的路由器 ID 更低。在这种情况下，Sunshine 就会选择通过 AS 65535 内 Panorama 的路径而不去选择通过 Talisman 的更直接的路径，如例 3-153 所示。

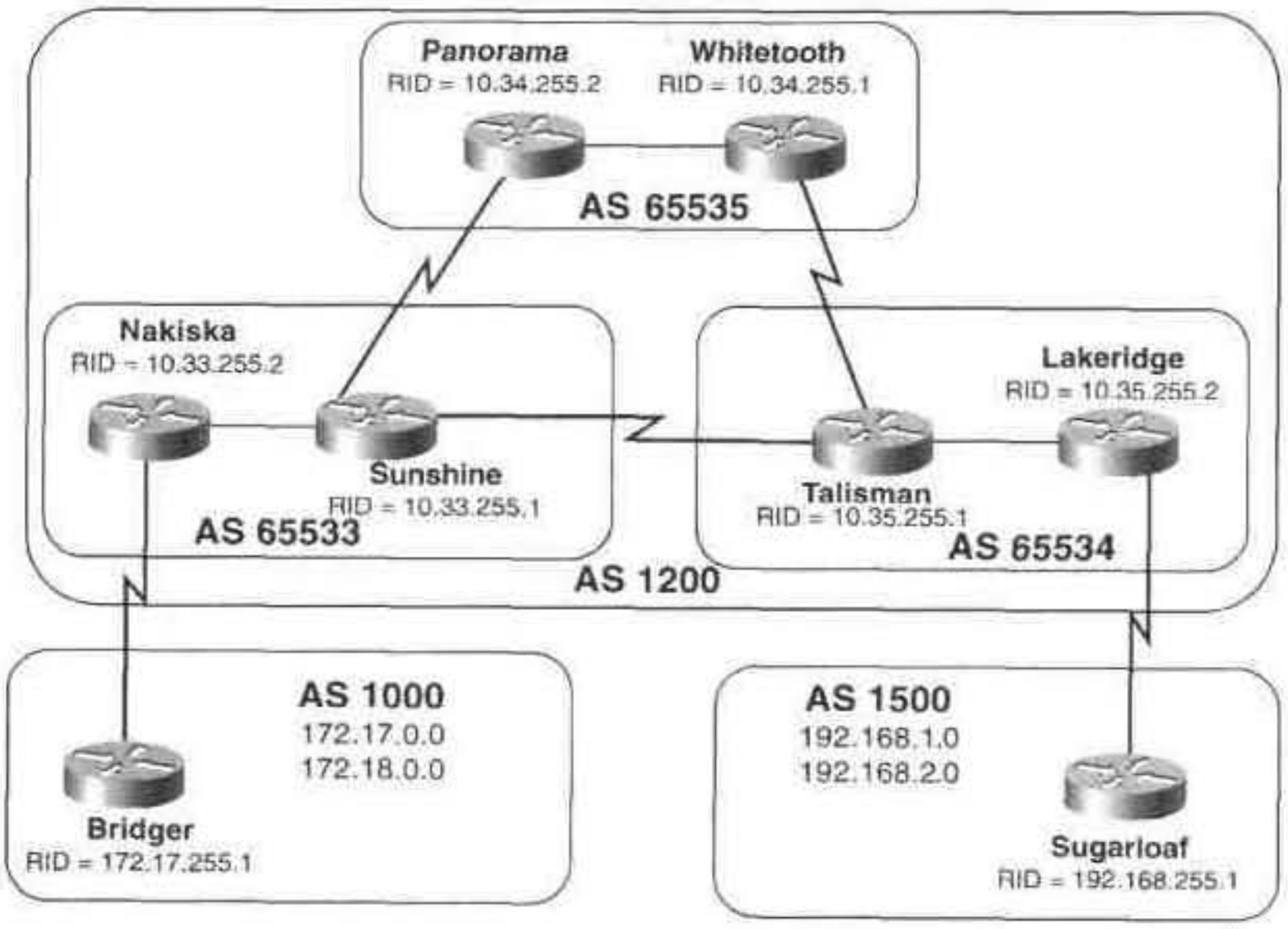


图 3-29　将 AS 65534 和 AS 65535 中路由器 ID 进行交换

例 3-153　根据 Panorama 较低的路由器 ID，Sunshine 选择了到 AS 1500 内网络的非最佳路由

```
Sunshine#show ip bgp
BGP table version is 17, local router ID is 10.33.255.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i172.17.0.0       10.33.255.2              50    200      0 1000 i
*>i172.18.0.0       10.33.255.2              50    200      0 1000 i
*> 192.168.1.0      10.34.255.2               0    100      0 (65535 65534) 1500 i
*                   10.35.255.1               0    100      0 (65534) 1500 i
*> 192.168.2.0      10.34.255.2               0    100      0 (65535 65534) 1500 i
*                   10.35.255.1               0    100      0 (65534) 1500 i
Sunshine#
```

几乎没有什么办法可以补救图 3-29 中拓扑的问题。试图去过滤路由或者改变管理权值只会使配置变得更加复杂，首先就使得生成联盟已经没有意义了。试图通过 LOCAL_PREF 或者 MED 属性来控制路由的选择是十分危险的，因为这些属性是在整个联盟范围内进行公布的；因为拓扑中存在的环路，这些属性会在路由选择时有意想不到的结果。

你必须重新设计联盟从而使图 3-29 拓扑中出现的那些问题不会再出现。一个通用的设计技巧来自 OSPF：在这个设计技巧中，必须让所有的区域通过一个骨干区域进行互连，从而消除了在区域间出现环路的可能性。

图 3-30 给出了与前面例子相同的路由器，只是对成员 AS 进行了重新设计。AS 65000 是一个骨干 AS，所有其他的 AS 必须通过它来进行互连。结果是，从任何非骨干 AS 到任何非骨干的其他 AS 的路径都具有相同的距离。AS 65000 和 AS 65535 之间的连接说明，还可以有备份连接，但不是在非骨干 AS 之间。BGP 的环路-避免机制防止了非最佳 AS 间路径出现的可能性。

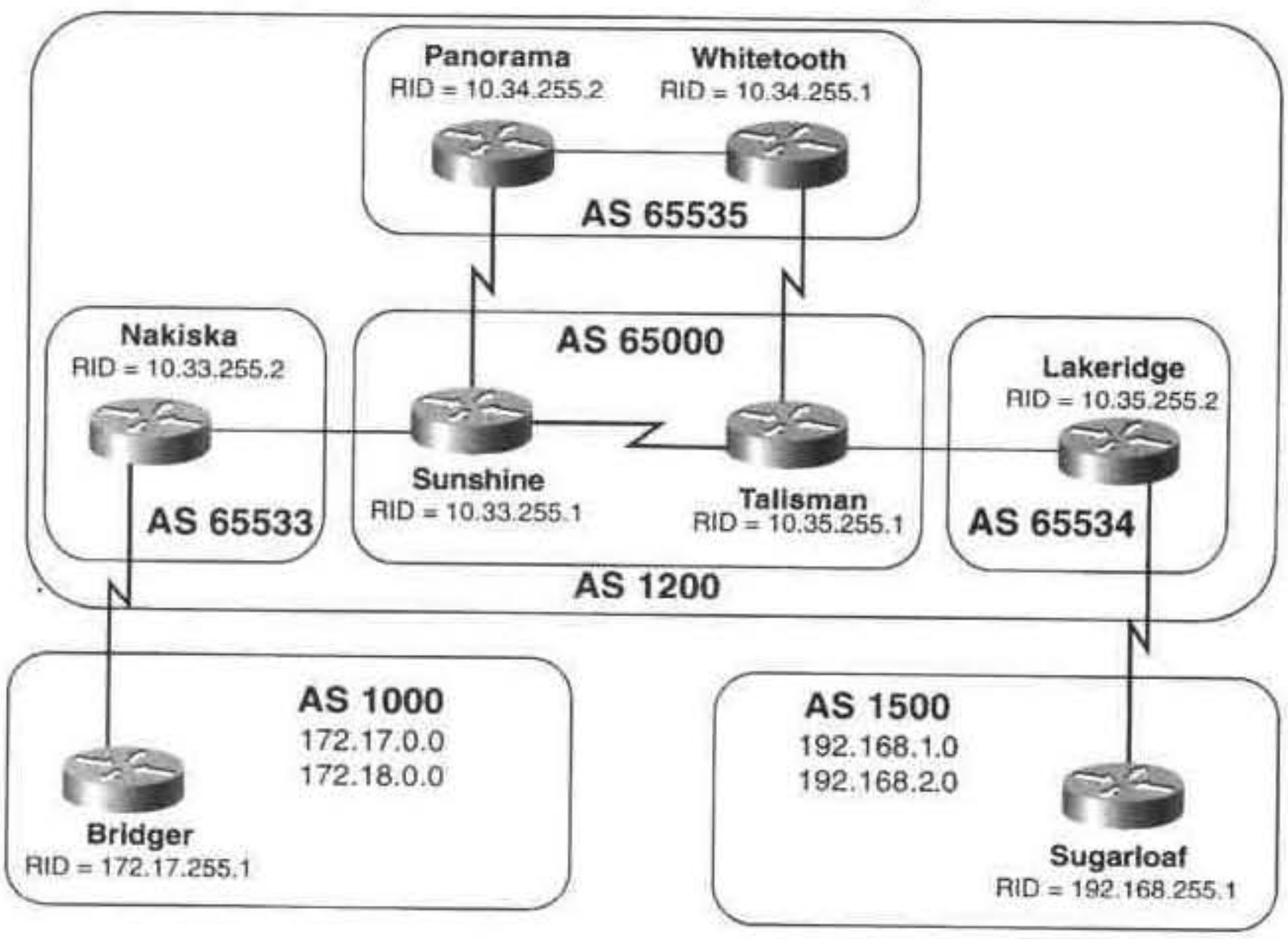


图 3-30　在联盟内部 AS 65000 是一个骨干 AS；其他所有的区域都通过它进行互连

如图 3-30 中无环路拓扑的另外一个好处是现在可以在成员 AS 之间使用 MED 属性。为了理解为什么在这个拓扑使用 MED 是安全的，让我们先来看一下图 3-31 中的拓扑。该拓扑与图 3-28 中的联盟相似，只是此时 AS 65534 到 AS 65535 有冗余连接。假设使用了 MED，于是，AS 65535 会为到 AS 1500 内目的地的业务量优选 Whitetooth/Lakeridge 链路而不选 Panorama/Talisman 链路。在这些两个 AS 之间可以作出正确选择，但是问题在于，这些 MED 会从 AS 65534 转发给 AS 65533。根据 AS 65533 处理 MED 的方式以及 Talisman 转发给它的 MED，AS 65533 有可能会再次选择非最佳路由。

在图 3-30 中，AS 65000 能够安全地向 AS 65535 发送 MED。从 AS 65535 到其他非骨干

AS 的路径必须通过骨干 AS。Sunshine 或者 Talisman 不会接受在 AS_PATH 中包含 65000 的路由，因此其他的成员 AS 不会看到从这些路由器发送给 AS 65535 的 MED。

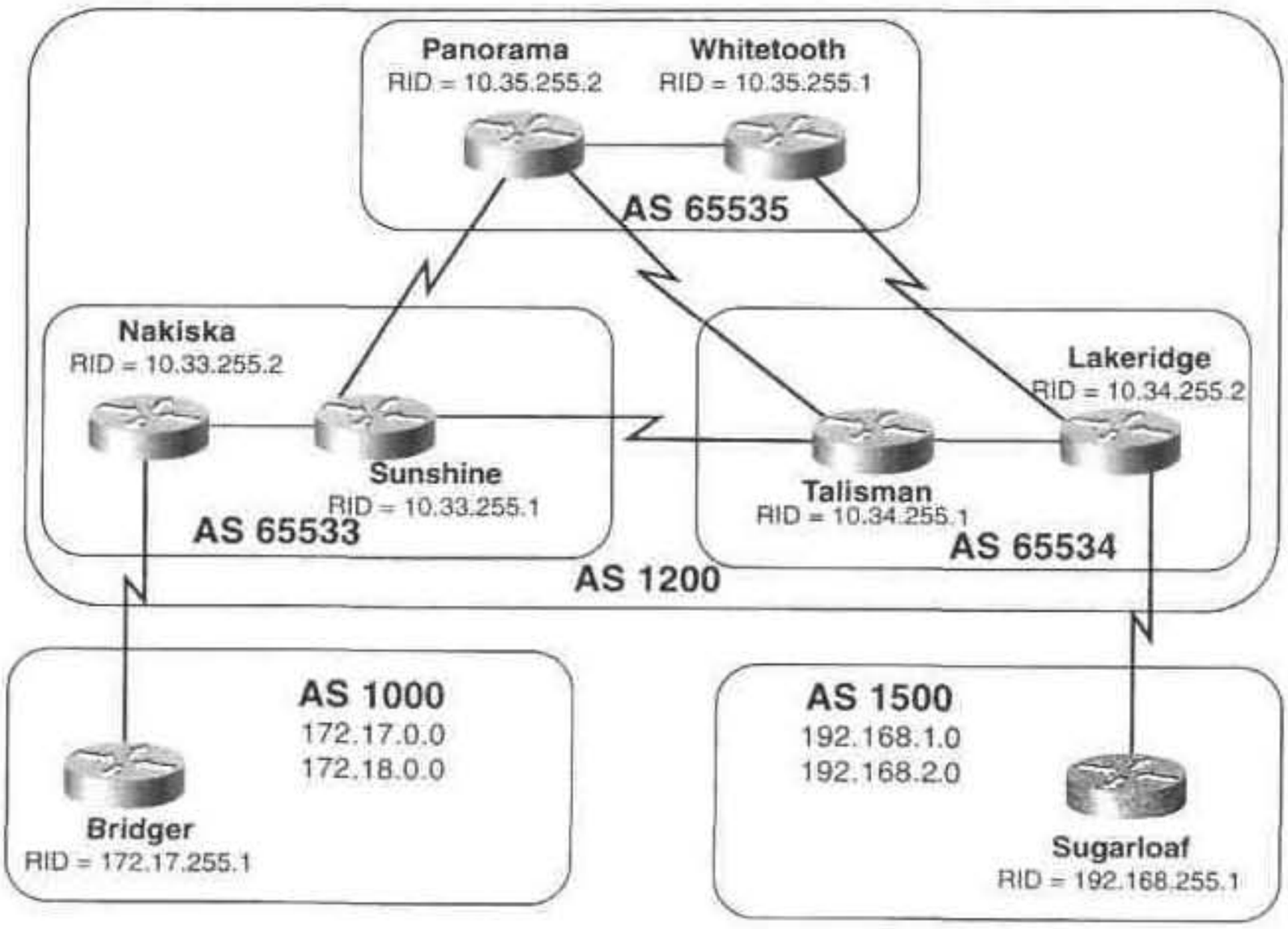


图 3-31　如果 AS 65534 通过 MED 来影响 AS 65535 的优选项，AS 65533 也会接收到 MED

缺省的情况下，图 3-30 中的 Panorama 和 Whitetooth 会优选联盟 EBGP 路由而不选 IBGP 路由，因此 Panorama 将所有到 AS 1000 以及 AS 1500 内目的地的业务量发送给 Sunshine；而 Whitetooth 会将到达同样目的地的业务量发送给 Talisman。此时可以使用 MED，因此 AS 65535 会把目的地是 AS 1000 内网络的业务量通过 Panorama/Sunshine 链路来发送，而目的地是 AS 1500 内网络的业务量会通过 Whitetooth/Talisman 链路来发送。例 3-154 给出了 Sunshine 和 Talisman 的配置。

Sunshine 将 AS_PATH 中包含 1000 的路由的 MED 设置为 100；将 AS_PATH 中包含 1500 的路由的 MED 设置为 200，Talisman 的设置刚好与之相反。例 3-155 给出了 MED 配置前后 Panorama 的 BGP 表的对比。在第一个 BGP 表中，到所有的目的地该路由器都优选联盟的 EBGP 连接；而在第二个 BGP 表中，因为改变了 MED，于是 Panorama 将到 AS 1500 内目的地的业务量通过 IBGP 链路发送给 Whitetooth，Whitetooth 通过它优选的联盟 EBGP 链路进行业务量的转发。

在图 3-32 中，在联盟内加入了两个本地子网：在 AS 65533 内加入了 10.33.5.0/24，在 AS 65535 内加入了 10.35.5.0/24。对 Sunshine 和 Talisman 进行了重新配置，使它们对这两个子网所采用的路由策略与它们对外部网络所采用的路由策略相同。也就是说，设置了 MED，从而使 AS 65535 通过 Panorama/Sunshine 链路来发送目的地是 10.33.5.0/24 的业务量，通过 Whitetooth/Talisman 链路发送目的地是 10.35.5.0/24 的业务量。

从例 3-156 中 Panorama 的 BGP 表可以看出，采用的策略并没有起到预期的效果。已经正确地配置了 MED，但是对于到 10.35.5.0/24 的业务量，该路由器还是优选到这两个子网的

联盟 EBGP 路径，而不是选取具有较低 MED 值的 IBGP 路径。出现这种情况的一个原因是，在缺省的情况下，在 BGP 决定进程中，不考虑联盟内部路由(在 AS_PATH 中没有任何外部 AS 号)的 MED 值。

例 3-154　配置 Sunshine 与 Talisman，使其向 AS 65535 发送 MED

Sunshine

```
router bgp 65000
 no synchronization
 bgp confederation identifier 1200
 bgp confederation peers 65533 65535
 neighbor 10.33.255.2 remote-as 65533
 neighbor 10.33.255.2 ebgp-multihop 2
 neighbor 10.33.255.2 update-source Loopback0
 neighbor 10.34.255.2 update-source Loopback0
 neighbor 10.34.255.2 ebgp-multihop 2
 neighbor 10.34.255.2 update-source Loopback0
 neighbor 10.34.255.2 next-hop-self
 neighbor 10.34.255.2 route-map SETMED out
 neighbor 10.35.255.1 remote-as 65000
 neighbor 10.35.255.1 update-source Loopback0
!
ip as-path access-list 1 permit _1000_
ip as-path access-list 2 permit _1500_
!
route-map SETMED permit 10
 match as-path 1
 set metric 100
!
route-map SETMED permit 20
 match as-path 2
 set metric 200
!
route-map SETMED permit 30
```

Talisman

```
router bgp 65000
 no synchronization
 bgp confederation identifier 1200
 bgp confederation peers 65534 65535
 neighbor 10.33.255.1 remote-as 65000
 neighbor 10.34.255.1 remote-as 65535
 neighbor 10.34.255.1 ebgp-multihop 2
 neighbor 10.34.255.1 update-source Loopback0
 neighbor 10.34.255.1 next-hop-self
 neighbor 10.34.255.1 route-map SETMED out
 neighbor 10.35.255.2 remote-as 65534
 neighbor 10.35.255.2 ebgp-multihop 2
 neighbor 10.35.255.2 update-source Loopback0
!
ip as-path access-list 1 permit _1500_
ip as-path access-list 2 permit _1000_
!
route-map SETMED permit 10
 match as-path 1
 set metric 100
!
```

```
route-map SETMED permit 20
 match as-path 2
 set metric 200
!
route-map SETMED permit 30
```

例 3-155　配置 AS 65000 内的路由器，在它们发送 MED 属性前后，Panorama 的 BGP 表对比

```
Panorama#show ip bgp
BGP table version is 34, local router ID is 10.35.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i172.17.0.0       10.35.255.1              0    100      0 (65000 65533) 1000 i
*>                  10.33.255.1              0    100      0 (65000 65533) 1000 i
* i172.18.0.0       10.35.255.1              0    100      0 (65000 65533) 1000 i
*>                  10.33.255.1              0    100      0 (65000 65533) 1000 i
* i192.168.1.0      10.35.255.1              0    100      0 (65000 65534) 1500 i
*>                  10.33.255.1              0    100      0 (65000 65534) 1500 i
* i192.168.2.0      10.35.255.1              0    100      0 (65000 65534) 1500 i
*>                  10.33.255.1              0    100      0 (65000 65534) 1500 i
Panorama#

Panorama#show ip bgp
BGP table version is 47, local router ID is 10.35.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i172.17.0.0       10.35.255.1             200    100      0 (65000 65533) 1000 i
*>                  10.33.255.1             200    100      0 (65000 65533) 1000 i
* i172.18.0.0       10.35.255.1             200    100      0 (65000 65533) 1000 i
*>                  10.33.255.1             200    100      0 (65000 65533) 1000 i
*>i192.168.1.0      10.35.255.1             100    100      0 (65000 65534) 1500 i
*                   10.33.255.1             200    100      0 (65000 65534) 1500 i
*>i192.168.2.0      10.35.255.1             100    100      0 (65000 65534) 1500 i
*                   10.33.255.1             200    100      0 (65000 65534) 1500 i
Panorama#
```

例 3-156　Panorama 在选择联盟内部路径时不考虑 MED 值

```
Panorama#show ip bgp
BGP table version is 127, local router ID is 10.35.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i10.33.5.0/24     10.35.255.1             200    100      0 (65000 65533) i
*>                  10.33.255.1             100    100      0 (65000 65533) i
*> 10.35.5.0/24     10.33.255.1             200    100      0 (65000 65534) i
* i                 10.35.255.1             100    100      0 (65000 65534) i
*> 172.17.0.0       10.33.255.1             100    100      0 (65000 65533) 1000 i
*> 172.18.0.0       10.33.255.1             100    100      0 (65000 65533) 1000 i
*>i192.168.1.0      10.35.255.1             100    100      0 (65000 65534) 1500 i
*                   10.33.255.1             200    100      0 (65000 65534) 1500 i
*>i192.168.2.0      10.35.255.1             100    100      0 (65000 65534) 1500 i
*                   10.33.255.1             200    100      0 (65000 65534) 1500 i
Panorama#
```

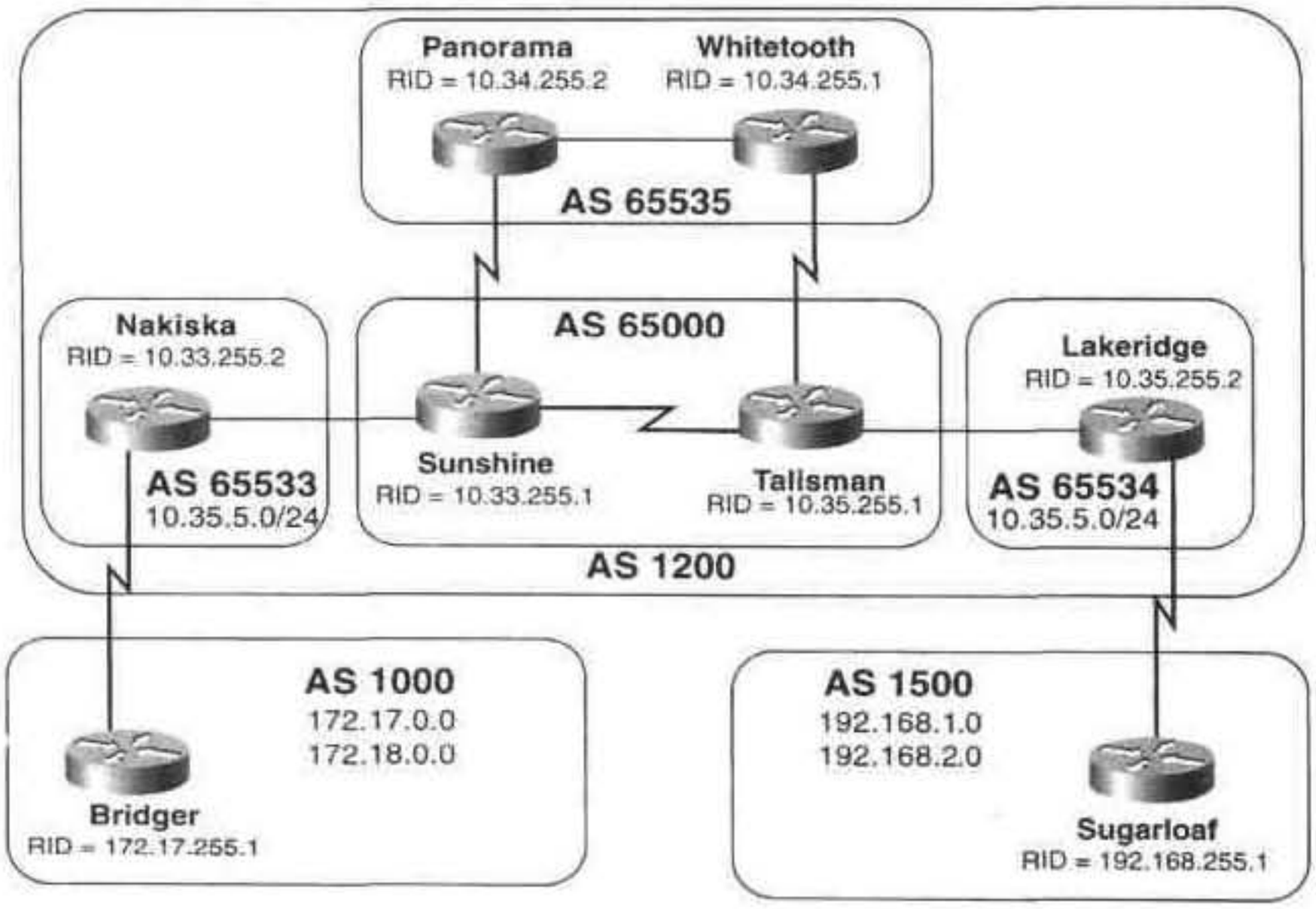


图 3-32　在 AS 65533 和 AS 65535 中加入了本地子网

bgp deterministic-med 命令告诉 BGP 进程在选择到联盟内部目的地的路径时，比较它们的 MED。例 3-157 给出了使用 **bgp deterministic-med** 命令的 Panorama 的配置。

例 3-157　配置 Panorama，当选择到联盟内部目的地的路径时，比较路径的 MED

```
router bgp 65535
 no synchronization
 bgp confederation identifier 1200
 bgp confederation peers 65000
 bgp deterministic-med
 neighbor 10.33.255.1 remote-as 65000
 neighbor 10.33.255.1 ebgp-multihop 2
 neighbor 10.33.255.1 update-source Loopback0
 neighbor 10.34.255.1 remote-as 65535
 neighbor 10.34.255.1 update-source Loopback0
```

例 3-158 给出了利用 **bgp deterministic-med** 命令配置 Panorama 以后的结果。现在 Panorama 使用具有最低 MED 值的路径而不管该路径对于成员 AS 来讲是内部的还是外部的。通过在前面的案例分析中讨论过的 **bgp always-compare-med** 命令，也可以得到同样的结果。这两个命令的区别在于，**bgp always-compare-med** 命令会比较到同一个目的地的路径的 MED，而不考虑这些 MED 是否是同一个 AS 公布的。在基于骨干的联盟中，例如图 3-32 所示的拓扑，这不是一个问题，因为没有 AS 有一条路径是到多个邻居 AS 的。

通过另外一个命令也可以达到同样的目的，这个命令就是：**bgp bestpath med confed**。这个命令与 **bgp deterministic-med** 命令具有相同的效果，但是它们之间有一点不同。在使用 **bgp bestpath med confed** 命令时，当一条路由的 AS_PATH 中有一个外部 AS 号，而其他的到同一个目的地路由的 AS_PATH 中只含有联盟 AS 号时，路由器会选择 MED 值最低的联盟内部路径而忽略含有外部 AS 号的路径。但是，这种情况非常少见。如果存在着两条到同一个目的地的路由，一条指示目的地在联盟内部，而另外一条指示目的地在联盟外部，这很有可

能是一个错误的配置或者是一个拙劣的设计。

例 3-158　当 Panorama 在联盟内部和联盟外部的路由之间做选择时，会考虑 MED

```
Panorama#show ip bgp
BGP table version is 10, local router ID is 10.35.2.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.33.5.0/24     10.33.255.1              100    100      0 (65000 65533) i
* i                 10.35.255.1              200    100      0 (65000 65533) i
*>i10.35.5.0/24     10.35.255.1              100    100      0 (65000 65534) i
*                   10.33.255.1              200    100      0 (65000 65534) i
*> 172.17.0.0       10.33.255.1              100    100      0 (65000 65533) 1000 i
*> 172.18.0.0       10.33.255.1              100    100      0 (65000 65533) 1000 i
*>i192.168.1.0      10.35.255.1              100    100      0 (65000 65534) 1500 i
*                   10.33.255.1              200    100      0 (65000 65534) 1500 i
*>i192.168.2.0      10.35.255.1              100    100      0 (65000 65534) 1500 i
*                   10.33.255.1              200    100      0 (65000 65534) 1500 i
Panorama#
```

3.3.5　案例分析：路由反射器

在大型 AS 中，路由反射器是降低 IBGP 对等连接数量的另外一种方法。相对于联盟来讲，使用路由反射器有两种优势：

- 联盟中的所有路由器都必须理解并支持联盟。但是只有路由反射器本身必须理解路由反射；客户路由器将它们到 RR 的连接只看作是另外一个 IBGP 连接。
- 路由反射执行起来比较容易，无论是在所需的命令方面还是在拓扑问题上。

另一方面，你可能希望使用在 EBGP 方面可用的控制工具来管理大型的 AS。在这种情况下，联盟又是一个较好的选择，这个案例分析给出了同时使用这两种方法的可能性。

图 3-33 对图 3-32 中的 AS 65535 进行了调整。Fortress 是路由反射器，Nakiska 和 Marmot 是客户。

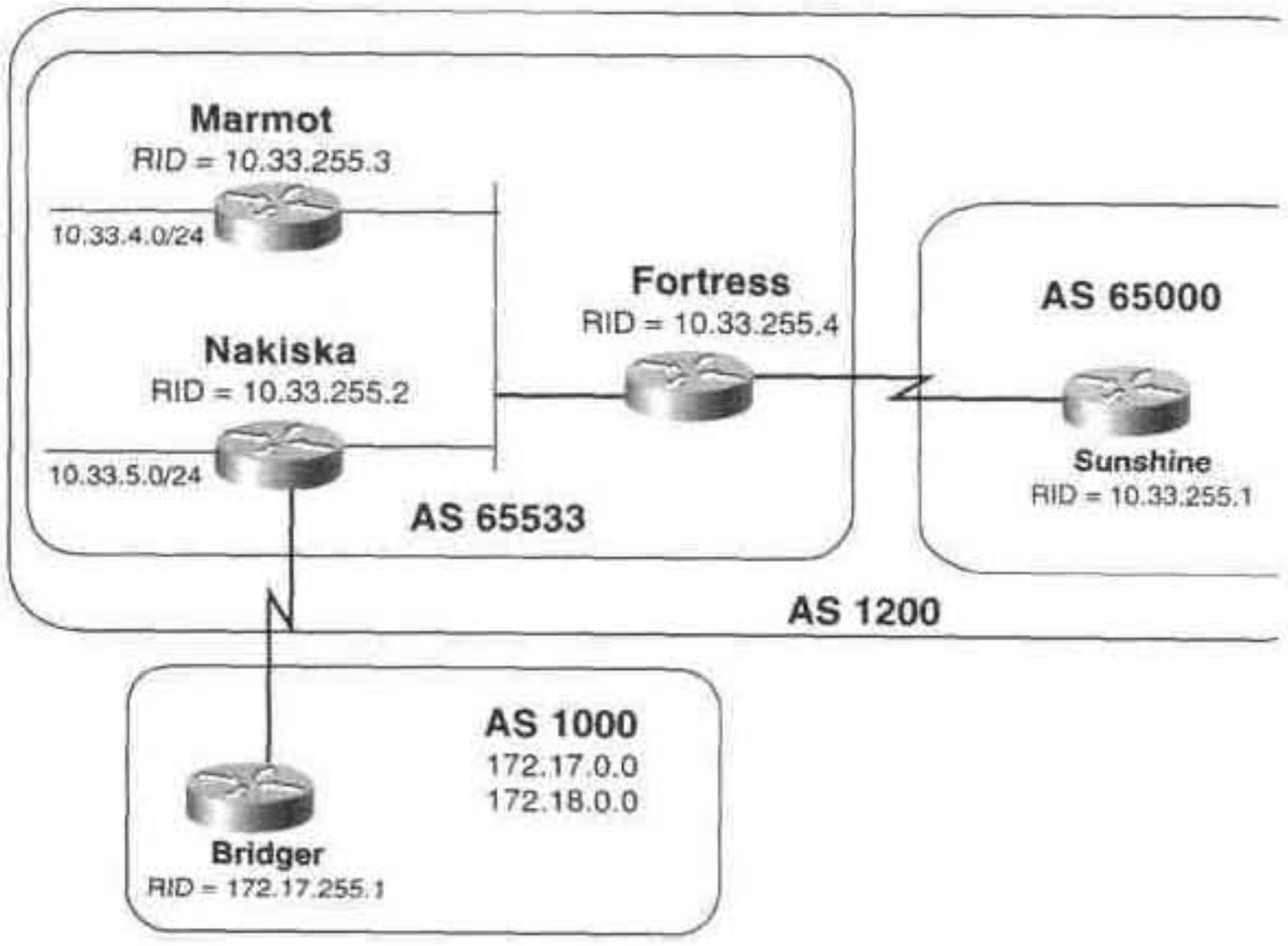


图 3-33　带有路由反射器的网络拓扑：Fortress 是路由反射器，Nakiska 以及 Marmot 是客户

例 3-159 给出了这三个路由器的配置。

例 3-159　将 Fortress 配置为路由反射器，Nakiska 和 Marmot 为客户

Fortress

```
router bgp 65533
 no synchronization
 bgp confederation identifier 1200
 bgp confederation peers 65000
 neighbor 10.33.255.1 remote-as 65000
 neighbor 10.33.255.1 ebgp-multihop 2
 neighbor 10.33.255.1 update-source Loopback0
 neighbor 10.33.255.2 remote-as 65533
 neighbor 10.33.255.2 update-source Loopback0
 neighbor 10.33.255.2 route-reflector-client
 neighbor 10.33.255.2 next-hop-self
 neighbor 10.33.255.3 remote-as 65533
 neighbor 10.33.255.3 update-source Loopback0
 neighbor 10.33.255.3 route-reflector-client
 neighbor 10.33.255.3 next-hop-self
```

Nakiska

```
router bgp 65533
 no synchronization
 bgp confederation identifier 1200
 network 10.33.5.0 mask 255.255.255.0
 neighbor 10.33.255.4 remote-as 65533
 neighbor 10.33.255.4 update-source Loopback0
 neighbor 10.33.255.4 next-hop-self
 neighbor 172.17.255.1 remote-as 1000
 neighbor 172.17.255.1 ebgp-multihop 2
 neighbor 172.17.255.1 update-source Loopback0
```

Marmot

```
router bgp 65533
 no synchronization
 bgp confederation identifier 1200
 network 10.33.4.0 mask 255.255.255.0
 neighbor 10.33.255.4 remote-as 65533
 neighbor 10.33.255.4 update-source Loopback0
 neighbor 10.33.255.4 next-hop-self
```

Nakiska 和 Marmot 具有普通的 IBGP 配置，它们只和 RR 有对等关系，它们之间没有。Nakiska 还和路由反射簇外部的一个路由器 Bridger 有对等关系。在 Fortress 配置中多使用的唯一一个命令就是用于它的每个客户的 **neighbor route-reflector-client**，该命令使它成为一个路由反射器。这个命令执行第 2 章讨论过的路由反射必需的但并不严格的 IBGP 规则：也就是说，将从一个客户学习到的 IBGP 路由公布给其他的客户以及这个反射簇外的 IBGP 对等，把从反射簇外的 IBGP 对等学习到的路由公布给客户。

例 3-160 中，在 Marmot 的 BGP 表中给出了到 10.33.5.0/24 的路由条目。在最后一行给出的 ORIGINATOR_ID 和 CLUSTER_LIST 属性是由 RR 加入的。ORIGINATOR_ID 由 RR 加入并且指出了公布这条路由的客户；到 10.33.5.0/24 的路由的发起者是 Nakiska(10.33.255.2)。这个属性保证在反射簇内不会出现路由环路。如果 Fortress 在更新消息中收到了这个 NLRI，它会在属性中识别 Nakiska 的路由器 ID 从而忽略该路由。这个属性是任选非传递的，因此，虽

然如果路由反射簇中的路由器不支持或者不理解该属性，有可能会失去一定的环路保护功能，但是并不要求一个路由反射簇中的路由器必须支持或者理解这个属性。

例 3-160　Marmot 有关 10.33.5.0/24 这一子网的 BGP 条目中显示了路由反射器把 ORIGINATOR_ID 与 CLUSTER_LIST 属性加到路由中

```
Marmot#show ip bgp 10.33.5.0
BGP routing table entry for 10.33.5.0 255.255.255.0, version 16
Paths: (1 available, best #1)
  Local
    10.33.255.2 (metric 11) from 10.33.255.4 (10.33.255.2)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Originator : 10.33.255.2, Cluster list: 10.33.255.4
Marmot#
```

和 ORIGINATOR_ID 一样，CLUSTER_LIST 也是防止出现路由环路的一个方法。一个 4 字节的簇 ID 用于识别反射簇，RR 将这个数字加入到 CLUSTER_LIST 中。如果 RR 接收到一个在 CLUSTER_LIST 中带有它本身簇 ID 的更新消息，它知道出现了环路从而放弃该路由。当路径需要通过多个路由反射簇时，该功能被证明是十分重要的。在图 3-33 中，路由的 CLUSTER_LIST 就是 Fortress 的路由器 ID，10.33.255.4。缺省的情况下，RR 将它自己的 BGP RID 加入到 CLUSTER_LIST 中。为了确定一个簇 ID 而不是 RR 的 RID，可以使用 **bgp cluster-id** 命令。可以将簇 ID 指定为 1~4294967295 之间的任何一个数或者是以点隔开的十进制格式。

如果在一个簇中有多于一个的路由反射器，必须使用 **bgp cluster-id** 命令来保证所有的 RR 都确认自己是这个簇的一个成员。在图 3-34 中，加入路由器 Norquay 并且将它配置成为第二个 RR，从而为该簇添加了冗余度。

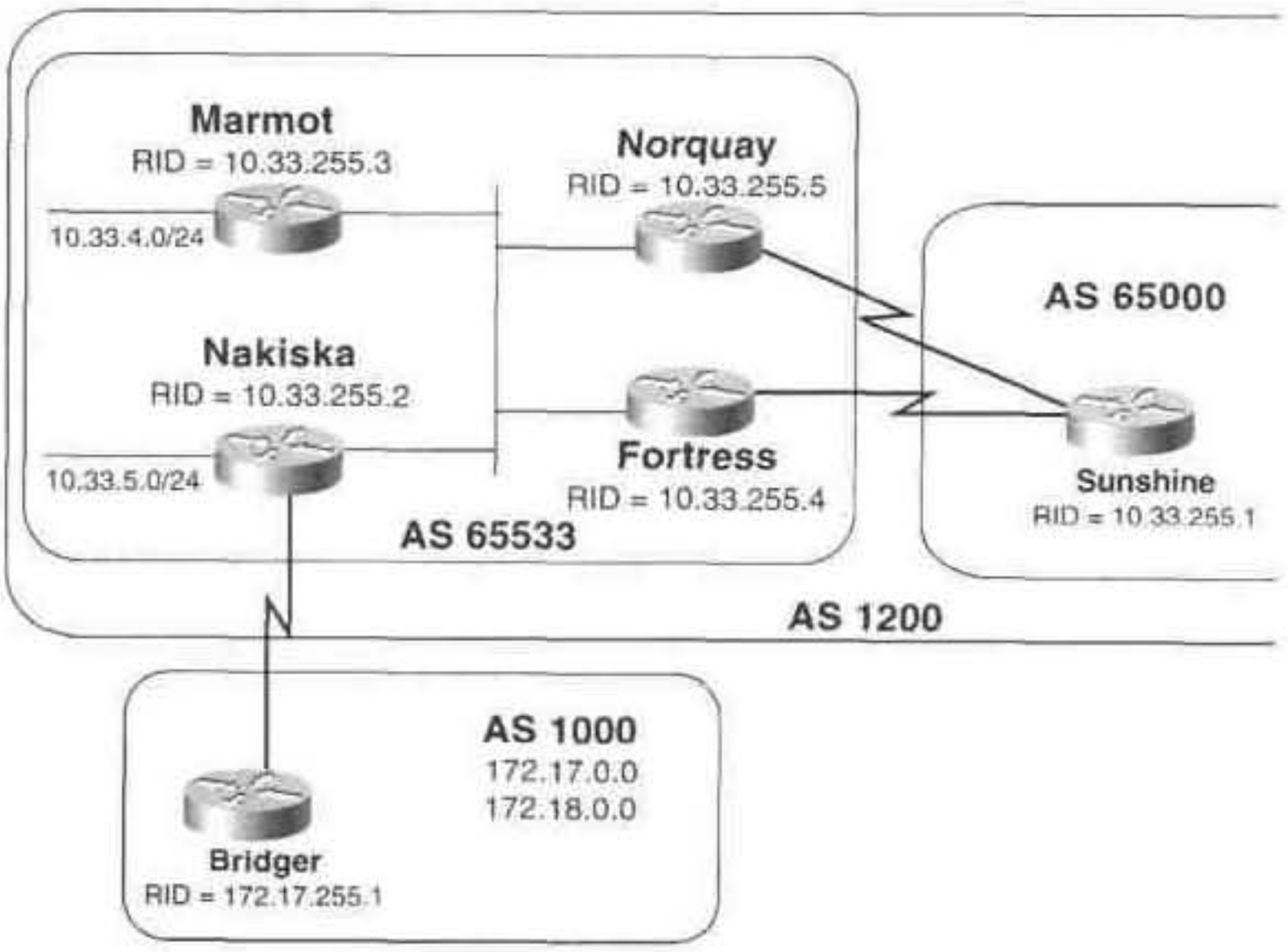


图 3-34　在路由反射簇中加入 Norquay 以增加冗余度

例 3-161 给出了 Fortress 和 Norquay 的配置。

例 3-161　将 Fortress 和 Norquay 配置为路由反射器

Fortress

```
router bgp 65533
 no synchronization
 bgp cluster-id 33
 bgp confederation identifier 1200
 bgp confederation peers 65000
 neighbor 10.33.255.1 remote-as 65000
 neighbor 10.33.255.1 ebgp-multihop 2
 neighbor 10.33.255.1 update-source Loopback0
 neighbor 10.33.255.2 remote-as 65533
 neighbor 10.33.255.2 update-source Loopback0
 neighbor 10.33.255.2 route-reflector-client
 neighbor 10.33.255.2 next-hop-self
 neighbor 10.33.255.3 remote-as 65533
 neighbor 10.33.255.3 update-source Loopback0
 neighbor 10.33.255.3 route-reflector-client
 neighbor 10.33.255.3 next-hop-self
 neighbor 10.33.255.5 remote-as 65533
 neighbor 10.33.255.5 update-source Loopback0
 neighbor 10.33.255.5 next-hop-self
```

Norquay

```
router bgp 65533
 no synchronization
 bgp cluster-id 33
 bgp confederation identifier 1200
 bgp confederation peers 65000
 neighbor 10.33.255.1 remote-as 65000
 neighbor 10.33.255.1 ebgp-multihop 2
 neighbor 10.33.255.1 update-source Loopback0
 neighbor 10.33.255.2 remote-as 65533
 neighbor 10.33.255.2 route-reflector-client
 neighbor 10.33.255.2 update-source Loopback0
 neighbor 10.33.255.2 next-hop-self
 neighbor 10.33.255.3 remote-as 65533
 neighbor 10.33.255.3 route-reflector-client
 neighbor 10.33.255.3 update-source Loopback0
 neighbor 10.33.255.3 next-hop-self
 neighbor 10.33.255.4 remote-as 65533
 neighbor 10.33.255.4 update-source Loopback0
 neighbor 10.33.255.4 next-hop-self
```

两个 RR 的簇 ID 都是 33。它们通过标准的 IBGP 互相对等而且通过 **neighbor route-reflector-client** 命令与路由反射客户建立对等关系。结果是，两个 RR 将路由反射给客户，但是 IBGP 的规则可以阻止它们将 IBGP 路由公布给对方。

对于客户配置来讲，唯一的变化就是为 Norquay 增加了一个 IBGP 配置，如例 3-162 所示。

例 3-163 给出了 Marmot 的 BGP 表中到子网 10.33.5.0/24 的相应的条目。到同样的目的地，在例 3-160 中只有一条路径，而在本例中有两条路径。这两条路径完全相同；因为没有使用 **maximum-paths** 命令配置路由器让它使用这两条路径，路由器会选择来自 10.33.255.4 的路

由，因为它具有最低的下一跳地址。

例 3-162　客户 Nakiska 和 Marmot 都与 Fortress 和 Norquay 这两个反射器建立对等关系

Nakiska

```
router bgp 65533
 no synchronization
 bgp confederation identifier 1200
 network 10.33.5.0 mask 255.255.255.0
 neighbor 10.33.255.4 remote-as 65533
 neighbor 10.33.255.4 update-source Loopback0
 neighbor 10.33.255.4 next-hop-self
 neighbor 10.33.255.5 remote-as 65533
 neighbor 10.33.255.5 update-source Loopback0
 neighbor 10.33.255.5 next-hop-self
 neighbor 172.17.255.1 remote-as 1000
 neighbor 172.17.255.1 ebgp-multihop 2
 neighbor 172.17.255.1 update-source Loopback0
```

Marmot

```
router bgp 65533
 no synchronization
 bgp confederation identifier 1200
 network 10.33.4.0 mask 255.255.255.0
 neighbor 10.33.255.4 remote-as 65533
 neighbor 10.33.255.4 update-source Loopback0
 neighbor 10.33.255.4 next-hop-self
 neighbor 10.33.255.5 remote-as 65533
 neighbor 10.33.255.5 update-source Loopback0
 neighbor 10.33.255.5 next-hop-self
```

例 3-163　Marmot 接收来自 RR Fortress 和 Norquay 的路由

```
Marmot#show ip bgp 10.33.5.0
BGP routing table entry for 10.33.5.0 255.255.255.0, version 2
Paths: (2 available, best #1)
  Local
    10.33.255.2 (metric 11) from 10.33.255.4 (10.33.255.2)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Originator : 10.33.255.2, Cluster list: 0.0.0.33
  Local
    10.33.255.2 (metric 11) from 10.33.255.5 (10.33.255.2)
      Origin IGP, metric 0, localpref 100, valid, internal
      Originator : 10.33.255.2, Cluster list: 0.0.0.33
Marmot#
```

例 3-161 中的路由反射器属于同一个簇，它们需要共享簇 ID。但是，路由反射器也可以属于不同的簇，而且客户的配置也不会改变，如例 3-162 所示。这个概念的关键就在于实际上客户并不知道它自己是客户。现在，许多的路由反射设计都采用这种对等的客户在多个簇而不是在同一个簇中有冗余的路由反射器的方法。

虽然路由反射客户可以有 EBGP 连接，如图 3-34 中 Nokiska，但是除了路由反射器，客户通常不应该有任何 IBGP 邻居。这也就意味着只能在路由反射器之间进行簇的连接，而不是在客户之间进行，因为客户不会检查接收到路由的 CLUSTER_LIST 属性。一个客户不会

检测到内部簇环。RR 间建立标准 IBGP 对等关系并且遵守所有的 IBGP 规则。在 RR 之间传递的唯一的附加信息就是用于防止产生环路的 CLUSTER_LIST 属性。如图 3-35 所示，AS 中所有路由反射器以及属于 AS 内部但是不属于任何一个簇的路由器之间要形成全网状连接。

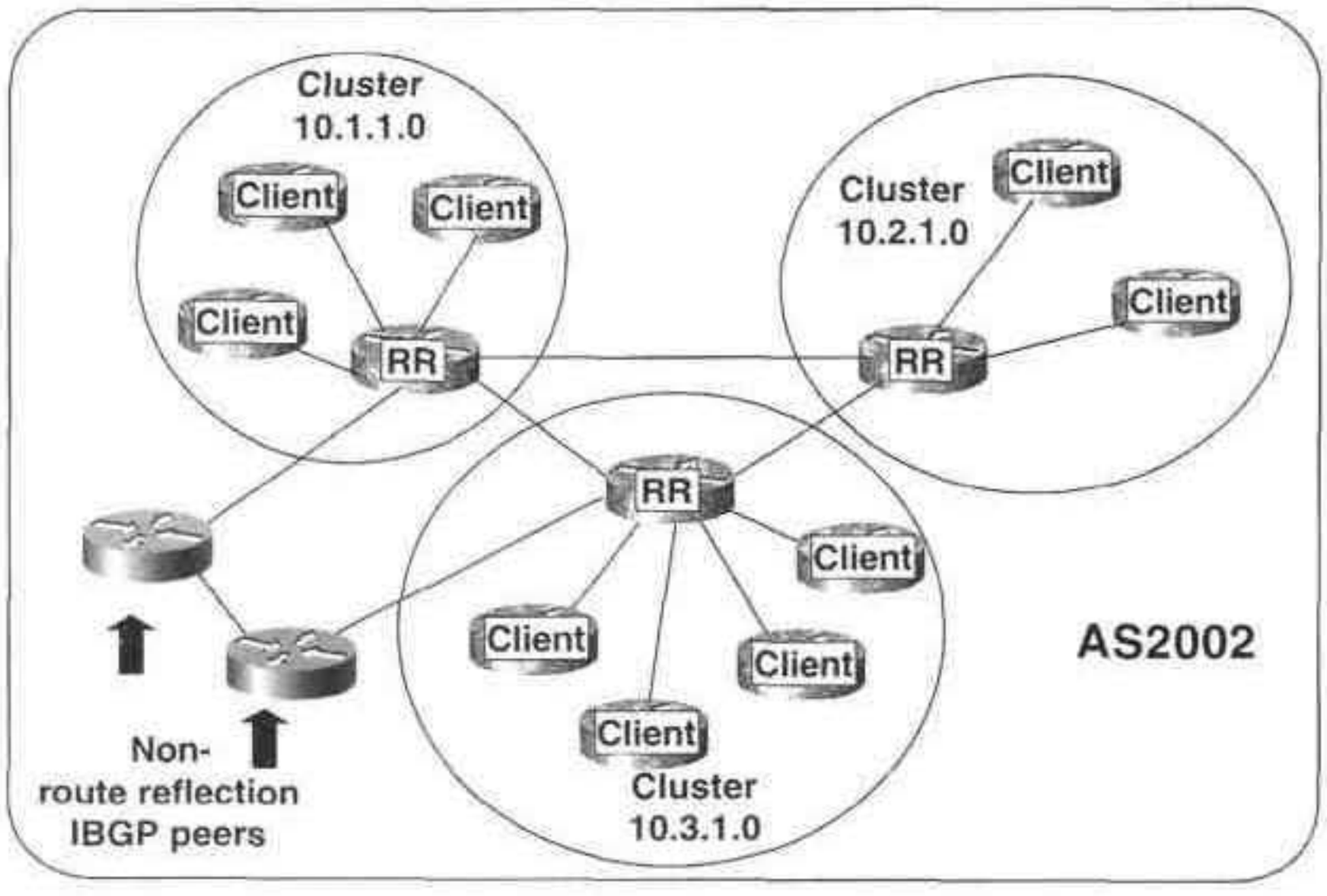


图 3-35　要在路由反射器之间而不是客户之间建立连接簇的链路

客户只能与它的 RR 建立对等关系，但是可以有两个特例。第一个，客户本身可以是另外一个簇的路由反射器。这样就允许路由反射器“嵌套”或者生成一个分层的簇，如第 2 章图 2-40 给出的示例。

第 2 个特例如第 2 章图 2-41 所示，当在客户之间存在着全 IBGP 网状连接时，全网状连接的客户为网络提供了更好的强壮性。当使用这样一个设计时，可以用 **no bgp client-to-client reflection** 命令来配置路由反射器，这样路由就按照普通 IBGP 规则在全网状连接的客户之间进行通信，而不需要 RR 再将路由从一个客户反射到另外一个客户。但是，还是需要 RR 将路由从客户反射给簇外的对端，以及将路由从簇外的对端反射给簇内客户。

3.4　展望

本章结束了这本书对 EGP 的讨论。接下来的章节将会讨论高级 IP 路由问题，包括网络地址翻译、多播路由以及 QoS。在以后的章节中涉及到 EGP 的案例分析时会使用 BGP。

3.5　推荐的读物

Halabi,S. Internet Routing Achitectures，　Second Edition. Indianapolis, Indiana: Cisco Press; 2000.

本书被认为是使用 Cisco IOS 软件配置 BGP-4 方面的权威版本。

3.6　命令归纳

表 3-3 给出了本章讨论的命令的一个列表以及对这些命令的描述。

表 3-3　　命令归纳

命　　令	描　　述
aggregate-address *address mask*[**as-set**] [**summary-only**][**suppress-map** *map-name*] {**advertise-map** *map-name*} |**attribute-map** *map-name*}	在 BGP 路由表中生成一个聚合条目
auto-summary	启动从子网到主要网络地址的自动归纳功能
bgp always-compare-med	比较由不同 AS 内的对等公布的到同一个目的地的路由的 MED 属性
bgp bestpath as-path ignore	在 BGP 决定进程中，告诉 BGP 进程忽略 AS_PATH 的长度
bgp bestpath med confed	比较由联盟 EBGP 对等公布的路由的 MED 属性
bgp client-to-client reflection	IBGP 对等之间的路由反射可用
bgp cluster-id *cluster-id*	为一个 BGP 路由反射簇设置簇 ID
bgp confederation identifier *autonomous-system*	指定联盟 ID(联盟外部对等能够看到的 AS 号)
bgp confederation peers *autonomous-system* [*autonomous-system*]	列出一个路由器的联盟 EBGP 对等的成员 AS
Bgp dampening [*half-life reuse suppress max-suppress-time*] [**route-map** *map*]	路由抑制可用并改变路由抑制的缺省值
bgp default local-preference *value*	改变缺省 LOCAL_PREF 的值(原为 100)
bgp deterministic-med	比较由同一个邻居成员 AS 内的联盟 EBGP 对等公布的路由的 MED 属性
bgp log-neighbor-changes	访问邻居重置日志可用
clear ip bgp {**address|peer-group-name*} [**soft** [**in|out**]]	重置一个或者多个 BGP 对等连接
clear ip bgp dampening [*address mask*]	清除 BGP 抑制信息并且不再抑制被抑制的路由
clear ip bgp flap-statistics [{**regexp** *regexp*}]| {**filter-list** *list*}|{*address mask*}	清除 BGP 摆动统计数据

续表

命　令	描　述
clear ip bgp peer-group *peer-group-name*	去掉一个对等组的所有成员
default-information originate	引起一个路由器向它的 BGP 对等公布缺省地址 0.0.0.0
default-metric *number*	设置缺省度量(MED)，一个 BGP 路由器将它加入到公布给 EBGP 对等的路由中
distance bgp *external-distance* *internal-distance local-distance*	改变 BGP 路由的缺省管理距离
ip as-path access-list *access-list-number* {**permit**|**deny**}*regexp*	定义一个检查 BGP 路由 AS_PATH 属性的访问列表
ip bgp-community new format	以 AA:NN 格式显示团体属性
ip community-list *community-list-number* {**permit** | **deny**} *community-number*	定义一个访问列表，通过路由的团体属性来识别 BGP 路由
ip prefix-list *prefix-list-name* [**seq** *sequencenumber*] {**permit** | **deny**} {*ip-prefix* | *length*} [**ge** | **le**] *min/max-length*	定义一个访问列表，检查在一个路由中公布的 NLRI 前缀和长度
match as-path *path-list-number*	生成一个到 AS_PATH 访问列表的呼叫
match community-list *community-list-number* [**exact**]	生成一个到团体访问列表的呼叫
neighbor {*ip-address* | *peer-group-name*} **advertisement-interval** *seconds*	改变缺省的向 BGP 对等发送路由更新消息的最短公布间隔
neighbor {*ip-address* | *peer-group-name*} **default-originate** [**route-map** *map-name*]	引起一个路由器向指定的邻居或者对等组公布缺省地址 0.0.0.0
neighbor {*ip-address* | *peer-group-name*} **description** *text*	把一个描述性的文本串与一个邻居的配置联系起来
neighbor {*ip-address* | *peer-group-name*} **distribute-list** {*access-list-number* | **prefix- list** *prefix-list-name* } {**in** | **out**}	通过 NLRI 过滤到或者来自一个邻居或对等组的路由
neighbor {*ip-address* | *peer-group-name*} **ebgp-multihop** *hops*	改变发送给邻居的携带 BGP 消息的数据包的缺省 TTL 值
neighbor {*ip-address* | *peer-group-name*} **filter-list** *access-list-number* {**in** | **out** | **weight** *weight*}	通过 AS_PATH 过滤入站或者出站 BGP 路由，或者设置入站路由的权值

续表

命　令	描　述
neighbor {*ip-address* | *peer-group-name*} **maximum-prefix** *maximum* [*threshold*] [**warning-only**]	为一个邻居或者对等组能够公布的 NLRI 前缀设置一个最大值
neighbor {*ip-address* | *peer-group-name*} **next-hop-self**	
neighbor {*ip-address* | *peer-group-name*} **password** *string*	启动对等之间的 MD5 认证
neighbor *ip-address* **peer-group** *peer-group-name*	为一个对等组分配一个邻居
neighbor *peer-group-name* **peer-group**	生成一个对等组
neighbor {*ip-address* | *peer-group-name*} **prefix-list** *prefix-list-name* {**in** | **out**}	根据前缀列表所明确的 NLRI 来过滤去到或者来自一个邻居的路由
neighbor {*ip-address* | *peer-group-name*} **remote-as** as-number	向 BGP 邻居表中加入一个邻居并且明确该邻居是 EBGP 还是 IBGP
neighbor {*ip-address* | *peer-group-name*} **route-map** *map-name* {**in** | **out**}	参考一个路由图为去往或者来自一个邻居或对等组的路由设置策略
neighbor *ip-address* **route-reflector-client**	将一个路由器设置为路由反射器并且确定邻居为路由反射客户
neighbor {*ip-address* | *peer-group-name*} **send-community**	确定团体属性将要发往的邻居
neighbor {*ip-address* | *peer-group-name*} **shutdown**	禁止一个邻居或者对等组
neighbor {*ip-address* | *peer-group-name*} **soft-reconfiguration inbound**	使一个路由器存储来自邻居的未经修改的更新消息用于入站软再配置
neighbor {*ip-address* | *peer-group-name*} **timers** *keepalive holdtime*	改变一个邻居缺省的 Keepalive 和保持间隔
neighbor {*ip-address* | *peer-group-name*} **update-source** *interface*	确定发起 IBGP 更新消息的接口 IP 地址
neighbor {*ip-address* | *peer-group-name*} **version** *version*	将 BPG 进程设置为同一个版本

续表

命　令	描　述
neighbor {*ip-address* | *peer-group-name*} **weight** *weight*	为从邻居接收来的路由设置一个权值
network *network-number* [**mask** *network-mask*]	指定将要由 BGP 进程公布的网络
network *network-number* **backdoor**	将指定的 EBGP 路由的管理距离设置为 200 并且禁止该路由的 EBGP 公布
network *address mask* **weight** *weight* [**route-map** *map-name*]	为到一个指定目的地的路由分配一个权值
router bgp *autonomous-system*	启动一个路由器上的 BGP 进程并且明确本地 AS 号
set as-path {**tag** | **prepend** *as-path-string*}	
set comm-list *community-list-number* **delete**	从一个指定的路由中去掉由被叫团体列表确定的团体属性
set community {community-number [**additive**]} | **none**	在一个确定的路由中设置团体属性
set dampening *half-life reuse suppress* *max-suppress-time*	改变一条指定路由的缺省抑制因数
set metric-type internal	设置一条指定路由的 MED 属性值，从而使它与下一跳的 IGP 度量相匹配
set origin {**igp** | **egp** *autonomous-system* | **incomplete**}	改变一条指定路由的 ORIGIN 属性
set weight *weight*	为一条指定路由设置权值
show ip bgp [*network*] [*network-mask*] [**longer-prefixes**]	显示一个 BGP 表中的所有条目
show ip bgp cidr-only	显示一个 BGP 表中具有非自然(有类别的)网络掩码的条目
show ip bgp community {*community-name* | *community-number*} [**exact**]	显示一个 BGP 表中具有指定团体属性的条目
show ip bgp community-list *community-list-number* [**exact**]	显示一个 BGP 表中被指定的团体列表允许的条目
show ip bgp dampened-paths	显示抑制路径
show ip bgp filter-list *access-list-number*	显示一个 BGP 表中与指定访问列表相匹配的条目
show ip bgp flap-statistics [{**regexp** *regexp*} |{**filter-list** *list-number* } | {*address mask* [**longer-prefix**]}]	显示 BGP 表中一个或者多个条目的扰动统计数据

续表

命　　令	描　　述
show ip bgp inconsistent-as	显示一个 BGP 表中，到同一个目的地的多条路径但是却具有不一致的 AS_PATH 属性的任何条目
show ip bgp neighbors [*address*][**received-routes** | **routes** | **advertised-routes** | {**paths** regex} | **dampened-routes**]	显示到邻居的 TCP 和 BGP 连接的信息
show ip bgp paths	显示一个数据库中所有的 BGP 路径
show ip bgp peer-group [*peer-group-name*][**summary**]	显示由一个对等组共享的特征的信息
show ip bgp regexp *regexp*	显示一个 BGP 表中 AS_PATH 属性与指定的正则表达式相匹配的条目
show ip bgp summary	显示所有 BGP 连接的归纳信息
synchronization	在 BGP 和本地 IGP 之间保持同步
table-map route-map-name	启动对路由图的调用，来修改输入到 IGP 路由表中一条路由的量度与标记的值
timers bgp keepalive holdtime	改变整个 BGP 进程缺省的 Keepalive 和保持间隔

3.7　配置练习

表 3-4 给出了用于配置练习 1~13 的路由器和地址。

表 3-4　　用于配置练习 1~13 的路由器/地址

自治系统	路由器	接　　口	IP 地址/掩码
1	R1	L0	10.255.255.1/32
		S0	192.168.100.1/30
		E0	192.168.100.5/30
		E1	192.168.100.13/30
	R2	L0	10.255.255.2/32
		S0	192.168.100.9/30
		S1	192.168.100.57/30
		E0	192.168.100.6/30
		E1	192.168.100.17/30
	R3	L0	10.255.255.3/32
		S0	192.168.100.25/30
		E0	192.168.100.18/30
		E1	192.168.100.21/30

续表

自治系统	路由器	接　口	IP 地址/掩码
	R4	L0	10.255.25.4/32
		S0	192.168.100.2/30
		S1	192 168.100.33/30
		E0	192.168.100.22/30
		E1	192.168.100.14/30
2	R5	S0	192.168.100.2/30
		E0	192.168.1.29/26
	R6	S0	192.168.100.10/30
		E0	192.168.1.130/26
3	R7	L0	10.255.255.7/32
		S0	192.168 100.26/30
		S1	192.168.100.41/30
		E0	192.168.100.37/30
		E1	172.16.1.1/24
4	R8	L0	10.255.255.8/32
		S0	192.168.100.30/30
		S1	192.168.100.45/30
		E0	192.168.100.38/30
		E1	172.16 2 1/24
5	R9	L0	10.255.255.9/32
		S0	192.168.100.42/30
		E0	192.168.9.1/24
		E1	192.168.150.1/24
	R10	L0	10 255.255.10/32
		S0	192.168.100.46/30
		E0	192.168.10.1/24
		E1	192.168.100.53/30
		E2	192.168.150 2/24

续表

自治系统	路由器	接　口	IP地址/掩码
	R11	L0	10.255.255.11/32
		S0	192.168.100.34/30
		E0	192.168.100.54/30
		E1	192.168.11.1/24
6	R12	L0	192.168.255.1/32
		S0	192.168.100.58/30
		E0	192.168.16.83/27

表3-4列出了配置练习1~13所需要的AS、路由器、接口以及地址，给出了路由器的所有接口。对于每个练习来讲，如果表中显示出路由器有Loopback接口，那么该接口一定是所有IBGP连接的源。除非在练习中有明确的规定，否则通常应该在物理接口地址间建立EBGP连接。提示：在做练习之前，先根据表中给出的子网情况，把网络图画出来。

1．表3-4中的AS 1是一个转接AS，它采用的IGP是OSPF。区域0跨越整个AS。没有把AS内部的网络公布到AS外。运行EBGP的子网不能公布给AS 1。写出AS 1内路由器的BGP配置，将所有的内部邻居放到一个名为LOCAL的对等组中。只有R3是在环回接口上执行EBGP对等。通过口令**ExeRcise1**来验证所有的IBGP连接。

2．表3-4中的AS 2是一个末梢AS(非转接)，它的IGP是EIGRP。对AS 2内的路由器进行配置，使它们对任何一个外部对等都运行EBGP而且它们会将任何一个EIGRP路由再分发给BGP。将学习到的BGP路由再分发给EIGRP。使用必要的过滤手段阻止再分发不正确的路由。

3．网络192.168.1.0、192.168.2.0、192.168.3.0、192.168.4.0在AS 2内，这个AS的管理者希望相邻AS在向192.168.1.0和192.168.3.0发送业务量时，优选R5；在向192.168.2.0和192.168.4.0发送业务量时，优选R6。在这种情况下，非优选链路作为优选链路的备份链路。192.168.5.0是一个专用网络，因此不能将它公布给任何EBGP对端。修改练习2中的配置让AS 2执行这个策略。

4．配置 R5 和 R6 的 EBGP 邻居，让它们公布一条缺省路由给 AS 2，不再公布其他路由。

5．AS 2 的相邻 AS 的管理者对练习 2 中的策略不是完全同意。他希望他所有的路由器在向 192.168.3.0 发送业务量时，都选择 R6，而把 R5 作为备份；所有发送给 192.168.4.0 的业务量都经过 R5，而把 R6 作为备份。练习 2 中的其他策略是可以接受的。写出执行这个策略的配置。

6．表 3-4 中的 AS 3 是一个末梢 AS，AS 4 是一个转接 AS。这两个 AS 的 IGP 都是 OSPF，R7 和 R8 的内部接口都在区域 0 内。写出 R7 和 R8 的 BGP 和 OSPF 配置，将表 3-4 中给出的内部地址公布给所有的 EBGP 对端并且保证在 OSPF 域内的路由器能够到达任何一个外部地址。在两个方向都不要进行路由再分发，同时，还要保证 R7 的 BGP 路由器 ID 为 192.168.3.254。

7．调整练习 6 的配置，从而使 R7 和 R8 在它们直连的链路上运行 OSPF；去掉链路上的 BGP。子网 172.16.3.0/24 和 172.16.4.0 之间的业务量应当优选这条直连链路，而把任何的 EBGP 链路作为备份。AS 3 内部和 AS 4 内部其他地址之间的业务量应该使用 EBGP 链路而把直连链路作为备份。而且，来自其他 AS 的业务量可以把直连链路作为备份路由。例如，如果到 AS 4 的 EBGP 链路出现了故障，相邻 AS 可以把目的地是 AS 4 的业务量发送给 AS 3，然后再由 AS 3 通过直连链路将这些业务量转发给 AS 4。

8．表 3-4 中的 AS 5 是一个转接 AS，它的 IGP 是 IS-IS。二层区域 47.0001 扩展到整个 AS。内部的网络是 192.168.9.0、192.168.11.0 以及 192.168.12.0。写出 R9 和 R10 的 IS-IS 和 BGP 配置。保证 IS-IS 域内的路由器能够了解所有外部的路由而且将内部所有的网络都公布给所有的 EBGP 对等。不要将 IS-IS 路由再分发给 BGP。

9．修改练习 8 中的配置，只让 AS 4 知道网络 192.168.12.0，其他所有的 AS 都不知道。

10. 调整练习 9 中的配置，使 AS 3 和 AS 4 优选通过 AS 1 的路径到达 192.168.11.0。

11. 表 3-4 中 AS 6 的内部网络是 192.168.16.0、192.168.17.0、192.168.18.0 和 192.168.19.0。写出 R12 的 BGP 配置，让它把这些网络公布给邻居 AS 并且公布一个网络的归纳路由。而邻居 AS 应该只将这个归纳路由公布给其他的 AS。

12. 调整 R12 的 EBGP 邻居最近的配置，使该邻居不接受不属于 R12 公布的聚合地址的前缀，不接受长于 24 比特的前缀，不接受多余 5 个的前缀。

13. 例 3-164 给出了表 3-4 中 R7 的 BGP 配置。表 3-4 中给出了内部前缀由 OSPF 进行公布。

例 3-164　路由器 R7 的 BPG 配置

```
router bgp 3
 redistribute ospf 1
 neighbor NEIGHBORS peer-group
 neighbor NEIGHBORS ebgp-multihop 2
 neighbor NEIGHBORS update-source Loopback0
 neighbor NEIGHBORS route-map EX13 out
 neighbor 10.255.255.8 remote-as 4
 neighbor 10.255.255.8 peer-group NEIGHBORS
 neighbor 10.255.255.9 remote-as 5
 neighbor 10.255.255.9 peer-group NEIGHBORS
 neighbor 10.255.255.3 remote-as 1
 neighbor 10.255.255.3 peer-group NEIGHBORS
 no auto-summary
!
ip classless
ip as-path access-list 1 permit ^1 2$
!
access-list 1 permit 172.16.1.0
access-list 2 permit 172.16.3.0
!
route-map EX13 permit 10
 match ip address 1
 set as-path prepend 2
!
route-map EX13 permit 20
```

```
 match ip address 2
 set as-path prepend 1
!
route-map EX13 permit 30
 match as-path 1
 set as-path prepend 4 5
!
route-map EX13 deny 40
```

解释路由图 EX13 的作用。

14. 图 3-36 中的路由器 R1 是路由器 R2、R3 和 R4 的路由反射器并且通过 FR PVC 与这些邻居互连。写出 R1 的 BGP 配置，为与这 4 个路由器相连的网络提供所有的连接性。簇 ID 是 6500。

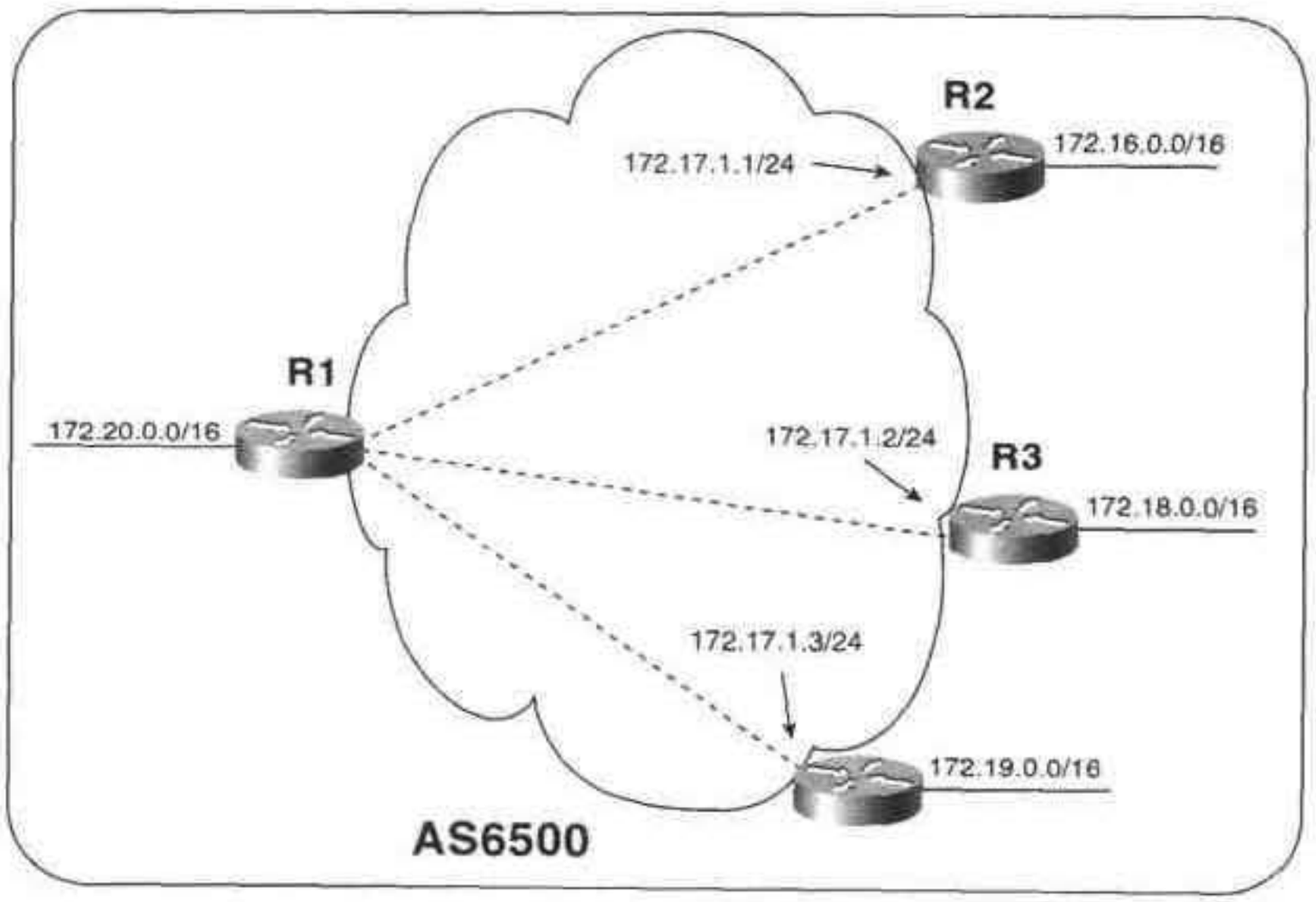


图 3-36　配置练习 14 的路由反射簇

3.8　故障排除练习

图 3-37 给出了故障排除练习 1~6 的网络互连图。

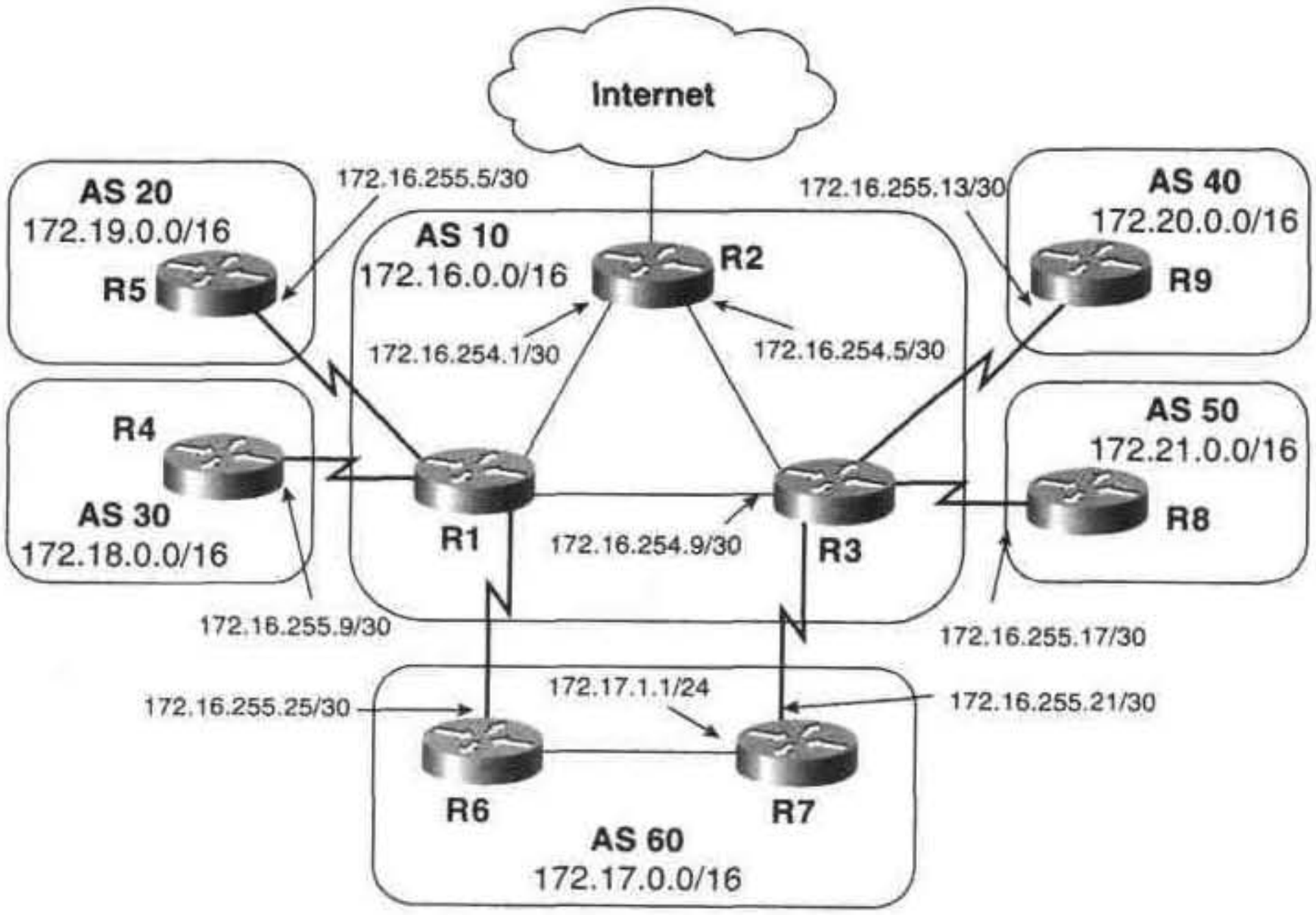


图 3-37　解决问题练习 1~6 的网络互连图

1. 例 3-165 给出了图 3-37 中路由器 R2 的 BGP 配置。

例 3-165　路由器 R2 的 BGP 配置

```
router bgp 10
 no synchronization
 network 0.0.0.0
 neighbor 172.16.254.2 remote-as 10
 neighbor 172.16.254.2 next-hop-self
 neighbor 172.16.254.6 remote-as 10
 neighbor 172.16.254.6 next-hop-self
 no auto-summary
!
ip classless
ip route 0.0.0.0 0.0.0.0 Ethernet10
```

例 3-166 给出了 R2 的 BGP 表和路由表。在图 3-37 中，虽然有路由到 AS 内的目的地，但是 Ping 这些目的地却失败了。为什么？

例 3-166　图 3-37 中 R2 的 BGP 表和路由表

```
R2#show ip bgp
BGP table version is 7, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 0.0.0.0          0.0.0.0                  0         32768 i
*>i172.17.0.0       172.16.255.21            0    100      0 60 i
*>i172.18.0.0       172.16.255.9             0    100      0 30 i
*>i172.19.0.0       172.16.255.5             0    100      0 20 i
*>i172.20.0.0       172.16.255.13            0    100      0 40 i
```

```
*>i172.21.0.0       172.16.255.17            0    100      0 50 i

R2#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is 0.0.0.0 to network 0.0.0.0

     10.0.0.0 255.255.255.0 is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Ethernet11
B    172.20.0.0 [200/0] via 172.16.255.13, 00:01:15
B    172.21.0.0 [200/0] via 172.16.255.17, 00:01:16
     172.16.0.0 255.255.255.252 is subnetted, 2 subnets
C       172.16.254.0 is directly connected, Ethernet12
C       172.16.254.4 is directly connected, Ethernet13
B    172.17.0.0 [200/0] via 172.16.255.21, 00:01:16
B    172.18.0.0 [200/0] via 172.16.255.9, 00:00:59
B    172.19.0.0 [200/0] via 172.16.255.5, 00:00:59
S*   0.0.0.0 0.0.0.0 is directly connected, Ethernet10
R2#ping 172.17.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
R2#
```

2. 例 3-167 给出了图 3-37 中路由器 R1 和 R5 的 **debug** 输出。这些信息暗示了什么问题？

例 3-167　图 3-37 中 R1 与 R5 的 debug 输出

```
R1#debug ip bgp
BGP debugging is on
R1#
BGP: 172.16.255.5 open active, local address 172.16.255.6
BGP: 172.16.255.5 sending OPEN, version 4
BGP: 172.16.255.5 received NOTIFICATION 2/2 (peer in wrong AS) 2 bytes 000A
BGP: 172.16.255.5 closing
```

```
R5#
6d08h: BGP: 172.16.255.6 open active, delay 28272ms
6d08h: BGP: 172.16.255.6 open active, local address 172.16.255.5
6d08h: BGP: 172.16.255.6 sending OPEN, version 4
6d08h: BGP: 172.16.255.6 OPEN rcvd, version 4
6d08h: BGP: 172.16.255.6 bad OPEN, remote AS is 10, expected 30
6d08h: BGP: 172.16.255.6 sending NOTIFICATION 2/2 (peer in wrong AS) 2 bytes 000A
6d08h: BGP: 172.16.255.6 remote close, state CLOSEWAIT
6d08h: BGP: 172.16.255.6 closing
```

3. 例 3-168 给出了图 3-37 中 R1 和 R3 的 BGP 表。第一个表指示通过 R6(172.16.255.25)或者 R3(172.16.254.9)都可以到达 172.17.0.0/24。R1 会选择哪条路径，为什么？

例 3-168 图 3-37 中 R1 和 R3 的 BGP 表

```
R1#show ip bgp
BGP table version is 8, local router ID is 172.20.7.1
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i0.0.0.0          172.16.254.1             0    100      0 i
* i172.17.0.0       172.16.254.9             0    100      0 60 i
*>                  172.16.255.25            0             0 60 i
*> 172.18.0.0       172.16.255.9             0             0 30 i
*> 172.19.0.0       172.16.255.5             0             0 20 i
*>i172.20.0.0       172.16.254.9             0    100      0 40 i
*>i172.21.0.0       172.16.254.9             0    100      0 50 i
R1#
```

```
R3#show ip bgp
BGP table version is 5, local router ID is 172.16.255.22
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i0.0.0.0          172.16.254.5             0    100      0 i
* i172.17.0.0       172.16.254.10            0    100      0 60 i
*>                  172.16.255.21            0             0 60 i
* i172.18.0.0       172.16.254.10            0    100      0 30 i
* i172.19.0.0       172.16.254.10            0    100      0 20 i
*> 172.20.0.0       172.16.255.13            0             0 40 i
*> 172.21.0.0       172.16.255.17            0             0 50 i
R3#
```

4. 例 3-169 给出了图 3-37 中 R1、R3、R6 和 R7 的 BGP 和 IGP 配置。

例 3-169 路由器 R1、R3、R6 和 R7 的 BGP 和 IGP 配置

R1

```
router bgp 10
 neighbor 172.16.254.1 remote-as 10
 neighbor 172.16.254.1 next-hop-self
 neighbor 172.16.254.9 remote-as 10
 neighbor 172.16.254.9 next-hop-self
 neighbor 172.16.255.5 remote-as 20
 neighbor 172.16.255.9 remote-as 30
 neighbor 172.16.255.25 remote-as 60
```

R3

```
router bgp 10
 neighbor 172.16.254.5 remote-as 10
 neighbor 172.16.254.5 next-hop-self
 neighbor 172.16.254.10 remote-as 10
 neighbor 172.16.254.10 next-hop-self
 neighbor 172.16.255.13 remote-as 40
 neighbor 172.16.255.17 remote-as 50
 neighbor 172.16.255.21 remote-as 60
 neighbor 172.16.255.21 next-hop-self
```

R6

```
router eigrp 60
```

```
 redistribute bgp 60 metric 1000 100 255 1 1500
 network 172.17.0.0
!
router bgp 60
 network 172.17.0.0
 neighbor 172.16.255.26 remote-as 10
```

R7

```
router eigrp 60
 redistribute bgp 60 metric 1000 100 255 1 1500
 network 172.17.0.0
!
router bgp 60
 network 172.17.0.0
 neighbor 172.16.255.22 remote-as 10
```

例 3-168 给出了 R1 和 R3 的 BGP 表。对于下面给出的每一个目的地，R6 会选用哪个下一跳地址？解释一下 R6 选用这些地址的原因。

目的地:

172.20.7.102

172.18.58.35

10.53.12.6

5．例 3-170 给出了图 3-37 中 R1 和 R3 的 BGP 配置。

例 3-170　路由器 R1 和 R3 的 BGP 配置

R1

```
router bgp 10
no synchronization
aggregate-address 172.16.0.0 255.255.248.0 summary-only
neighbor 172.16.254.1 remote-as 10
neighbor 172.16.254.1 next-hop-self
neighbor 172.16.254.9 remote-as 10
neighbor 172.16.254.9 next-hop-self
neighbor 172.16.255.5 remote-as 20
neighbor 172.16.255.9 remote-as 30
neighbor 172.16.255.25 remote-as 60
```

R3

```
router bgp 10
 no synchronization
 aggregate-address 172.16.0.0 255.255.248.0 summary-only
 neighbor 172.16.254.5 remote-as 10
 neighbor 172.16.254.5 next-hop-self
 neighbor 172.16.254.10 remote-as 10
 neighbor 172.16.254.10 next-hop-self
 neighbor 172.16.255.13 remote-as 40
 neighbor 172.16.255.17 remote-as 50
 neighbor 172.16.255.21 remote-as 60
 neighbor 172.16.255.21 next-hop-self
```

该配置的目的是抑制所有的更具体路由而只公布聚合路由。但是在例 3-171 R8 的 BGP 表中还是显示了更具体路由。问题出在哪里？

例 3-171　图 3-37 中 R8 的 BGP 表

```
R8#show ip bgp
BGP table version is 163, local router ID is 172.21.1.1
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 0.0.0.0          172.16.255.18                          0 10 i
*> 172.17.0.0       172.16.255.18                          0 10 60 i
*> 172.18.0.0       172.16.255.18                          0 10 30 i
*> 172.19.0.0       172.16.255.18                          0 10 20 i
*> 172.20.0.0       172.16.255.18                          0 10 40 i
*> 172.21.0.0       0.0.0.0                  0         32768 i
R8#
```

6. 在图 3-37 中，来自 AS 60，目的地是任何其他 AS 的数据包都应该通过 R6 和 R1 之间的链路进行转发，R7 和 R3 之间的链路应该只用做这些业务量的备份路由，但是目的地是 Internet 的业务量可以使用该链路。为了执行这个策略，R3 应该只公布缺省路由以及聚合路由 172.16.0.0/13 而 R1 应该公布更具体路由。例 3-172 给出了路由器 R1、R3、R6 和 R7 的配置。

例 3-172　路由器 R1、R3、R6 和 R7 的配置

R1

```
router bgp 10
no synchronization
neighbor 172.16.254.1 remote-as 10
neighbor 172.16.254.1 next-hop-self
neighbor 172.16.254.9 remote-as 10
neighbor 172.16.254.9 next-hop-self
neighbor 172.16.255.5 remote-as 20
neighbor 172.16.255.9 remote-as 30
neighbor 172.16.255.25 remote-as 60
```

R3

```
router bgp 10
 no synchronization
 aggregate-address 172.16.0.0 255.248.0.0 summary-only
 neighbor 172.16.254.5 remote-as 10
 neighbor 172.16.254.5 next-hop-self
 neighbor 172.16.254.10 remote-as 10
 neighbor 172.16.254.10 next-hop-self
 neighbor 172.16.255.13 remote-as 40
 neighbor 172.16.255.17 remote-as 50
 neighbor 172.16.255.21 remote-as 60
 neighbor 172.16.255.21 next-hop-self
```

R6

```
redistribute bgp 60 metric 1000 100 255 1 1500
 network 172.17.0.0
```

```
!
router bgp 60
 network 172.17.0.0
 neighbor 172.16.255.26 remote-as 10
```

R7

```
router eigrp 60
 redistribute bgp 60 metric 1000 100 255 1 1500
 network 172.17.0.0
!
router bgp 60
 network 172.17.0.0
 neighbor 172.16.255.22 remote-as 10
```

例 3-173 给出了 R7 的路由表。本例中提出的要求能够达到吗？如果不能，为什么？

例 3-173　故障排除练习 6 中 R7 的路由表

```
R7#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
       T - traffic engineered route

Gateway of last resort is 172.16.255.22 to network 0.0.0.0

     172.17.0.0/24 is subnetted, 3 subnets
C       172.17.1.0 is directly connected, Ethernet0
D       172.17.3.0 [90/409600] via 172.17.1.2, 09:18:50, Ethernet0
C       172.17.2.0 is directly connected, Ethernet1
     172.16.0.0/30 is subnetted, 1 subnets
C       172.16.255.20 is directly connected, Serial0
D EX 172.19.0.0/16 [170/2611200] via 172.17.1.2, 00:19:08, Ethernet0
D EX 172.18.0.0/16 [170/2611200] via 172.17.1.2, 00:19:08, Ethernet0
B*   0.0.0.0/0 [20/0] via 172.16.255.22, 00:18:37
B    172.16.0.0/13 [20/0] via 172.16.255.22, 00:18:09
R7#
```

7．让我们重新看一下图 3-19 和例 3-98 以及相关的讨论。Meribel 将它的本地路由 172.17.0.0 公布给它的 EBGP 对等，公布路由的 ORIGIN 为 Incomplete，然而 Lillehammer 以 ORIGIN 为 IGP 的形式将该路由再次公布给了 Meribel,这样会导致路由环路吗？

8．例 3-174 给出了图 3-24 中路由器 Colorado 的配置。图 3-24 中所有的路由器 ID 都是在环回接口上配置的，而且路由器上除了 BGP 以外没有运行其他的路由协议。假设图中所有的链路工作都正常，其他五个路由器都是 Colorado 的 EBGP 对等吗？如果不是，

为什么？

例 3-174　图 3-24 中路由器 Colorado 的配置

```
router bgp 100
 network 10.1.11.0 mask 255.255.255.0
 network 10.1.12.0 mask 255.255.255.0
 neighbor CLIENTS peer-group
 neighbor CLIENTS ebgp-multihop 2
 neighbor CLIENTS update-source Loopback2
 neighbor CLIENTS filter-list 2 in
 neighbor CLIENTS filter-list 1 out
 neighbor 10.1.255.2 remote-as 200
 neighbor 10.1.255.2 peer-group CLIENTS
 neighbor 10.1.255.3 remote-as 300
 neighbor 10.1.255.3 peer-group CLIENTS
 neighbor 10.1.255.4 remote-as 400
 neighbor 10.1.255.4 peer-group CLIENTS
 neighbor 10.1.255.5 remote-as 500
 neighbor 10.1.255.5 peer-group CLIENTS
 neighbor 10.1.255.6 remote-as 600
 neighbor 10.1.255.6 peer-group CLIENTS
 no auto-summary
!
ip classless
ip route 10.1.255.2 255.255.255.255 Serial0/1.305
ip route 10.1.255.3 255.255.255.255 Serial0/1.306
ip route 10.1.255.4 255.255.255.255 Serial0/1.307
ip route 10.1.255.5 255.255.255.255 Serial0/1.308
!
ip as-path access-list 1 permit ^$
ip as-path access-list 2 permit ^[2-6]00$
```

9. 参考故障排除练习 8 中对图 3-24 中路由器 Colorado 的配置。如果在配置中将 **no atuo-summary** 命令去掉，会产生什么结果？

10. 参考故障排除练习 8 中的配置，入站路由过滤器允许的路由有哪些？

11. 参考图 3-24 以及故障排除练习 8 中路由器 Colorado 的配置。除了它自己 AS 的本地子网或者 AS 间的链路以外，子网 10.1.3.0/24 上的一个主机可以 ping 通哪个子网？

第二部分

高级 IP 路由问题

第 4 章　网络地址翻译

第 5 章　IP 多播路由介绍

第 6 章　IP 多播路由的配置和故障排除

第 7 章　大范围 IP 多播路由

第 8 章　IPv6

第 9 章　路由器管理

第 4 章　网络地址翻译

本章涉及到下列关键主题：

- **NAT 的操作**——本节讨论网络地址翻译的基本理论，包括基本概念和术语，以及典型的 NAT 应用。
- **NAT 问题**——本节讨论在使用 NAT 过程中可能遇到的一些潜在问题。要解决这些问题，既可以通过 Cisco IOS 软件的功能，也可以通过设计技巧。针对这两个方面，本节都给出了一定的解决方案。
- **配置 NAT**——本节给出一些案例分析来说明应该如何配置 Cisco IOS 软件，以使它能够执行典型的 NAT 功能。
- **解决 NAT 中出现的问题**——本节讨论解决 Cisco NAT 中出现问题的不同的方法和工具。

网络地址翻译(NAT)是一项功能，数据包中的 IP 地址可以通过该功能被不同的 IP 地址所取代。通常由路由器或者防火墙来执行这项功能。当然，本章的重点主要是路由器上的 NAT。

注：　缩写 NAT 既可以代表网络地址翻译，也可以代表网络地址翻译器(运行 NAT 功能的软件)。

4.1　NAT 的操作

在 RFC1631 中对 NAT 进行了描述。[1]和无类别域间路由(CIDR)一样，NAT 最初的目的也是通过允许用较少的公用 IP 地址代表多数的专用 IP 地址来减缓可用 IP 地址空间枯竭的速度。从那时候起，用户就发现在网络升级整合、服务器负载均衡以及生成“虚拟服务器”方面，NAT 是一个很有用的工具。本节会分别讨论这些应用，但是首先先介绍一下 NAT 的基本功能和术语。

4.1.1　NAT 的基本概念

图 4-1 描述了一个简单的 NAT 功能。设备 A 具有一个 IP 地址，该地址在由 RFC 1918 指定的专用地址范围之内，相反，设备 B 具有一个公用 IP 地址。当设备 A 向设备 B 发送一个数据包时，数据包要通过一个运行 NAT 的路由器。NAT 将数据包中源地址字段内设备 A 的专用地址(192.168.2.23)换成一个可以在 Internet 上进行选路的公开地址(203.10.5.23)，并将数据包转发出去。设备 B 给设备 A 发送应答的时候，数据包中目的地地址是 203.10.5.23。数据包再次通过 NAT 路由器，目的地地址又换成了设备 A 的专用地址。

NAT 对于地址翻译中涉及到的终端系统来讲是透明的。在图 4-1 中，设备 A 只知道它的 IP 地址是 192.168.2.23；它并不知道 203.10.5.23 这个地址。另一方面，对于设备 B 来讲，它

认为设备 A 的地址是 203.10.5.23；它不知道 192.168.2.23 这个地址，这个地址对设备 B“隐藏”了。

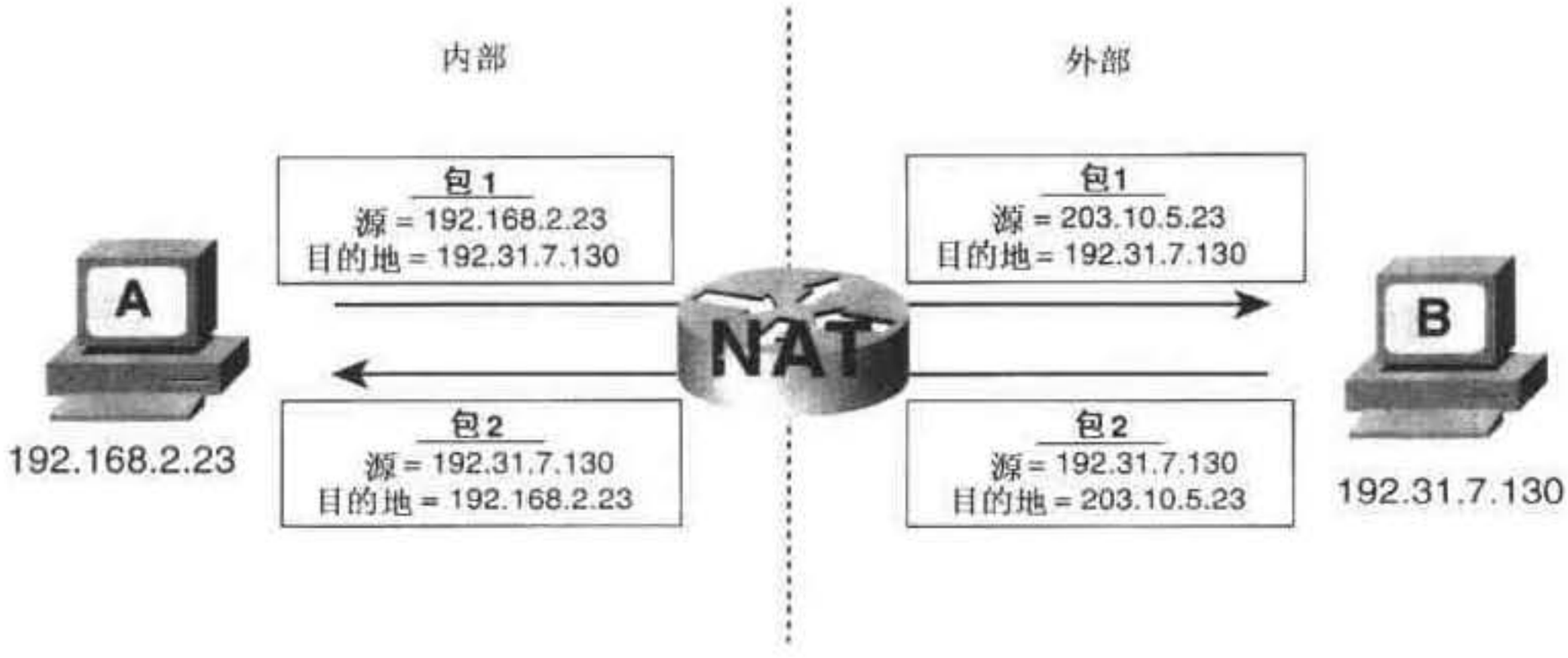


图 4-1　NAT 路由器用一个公开可寻路的地址(203.10.5.23)替换设备 A 的专用地址(192.168.2.23)

NAT 可以在两个方向上隐藏地址。在图 4-2 中，在设备 A 和设备 B 的地址上都执行了 NAT。设备 A 认为设备 B 的地址是 172.16.80.91，而实际上，设备 B 的真实地址是 192.31.7.130。可以看出，为了支持这个地址方案，NAT 在两个方向上都翻译了源地址和目的地地址。

Cisco NAT 设备分成了内和外两部分。典型的情况下，内部是私人企业网或者 ISP，外部是公共 Internet 或是面向 Internet 的业务供应商。除此以外，Cisco NAT 设备还将地址分成本地或者全局两部分。本地地址是指在内部能够被该设备本身看到的地址，而全局地址是指能够被该设备外部的设备看到的地址。下面给出了 4 个术语，一个地址可以是这 4 个类型地址中的一个：

- **内部本地(IL，Inside local)**——分配给内部设备的地址。这些地址不会对外公布。
- **内部全局(IG，Inside global)**——通过这个地址，外部可以知道内部设备。
- **外部全局(OG，Outside global)**——分配给外部设备的地址。这些地址不会向内部公布。
- **外部本地(OL，Outside local)**——通过这个地址，内部设备可以知道外部设备。

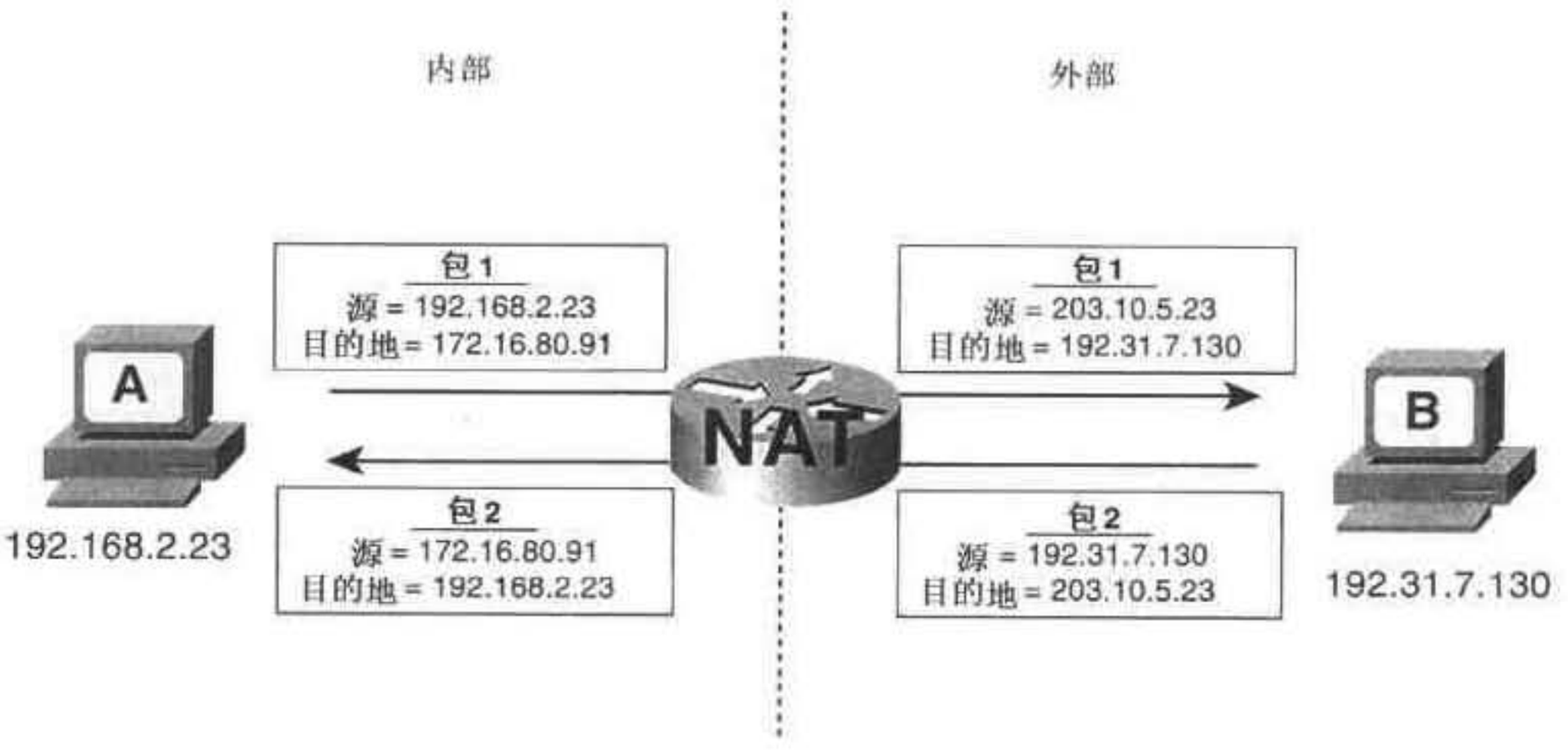


图 4-2　NAT 路由器在两个方向上翻译源地址和目的地地址

如图 4-2 所示，设备 A 在内部而设备 B 在外部。192.168.2.23 是内部本地地址，而 203.10.5.23 是内部全局地址。 172.16.80.91 是外部本地地址， 而 192.31.7.130 是外部全局地址。

IG 地址映射为 IL 地址，OL 地址映射为 OG 地址。NAT 设备通过一个地址翻译表跟踪这些映射。例 4-1 给出了图 4-2 中 NAT 路由器的地址翻译表。这个翻译表包含了三个条目。让我们从最底下一行来读一下这个翻译表， 第一行将 OL 地址 172.16.80.91 映射为 OG 地址 192.31.7.130。下一个条目将 IG 地址 203.10.5.23 映射为 IL 地址 192.168.2.23。这两个条目都是静态条目， 是在配置路由器来翻译特定地址时生成的。最后一个条目(最上面的一行)将内部地址映射为外部地址。这个条目是动态的，是设备 A 第一次向设备 B 发送数据包时生成的。

例 4-1　图 4-2 中 NAT 路由器的地址翻译表

```
NATrouter#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
--- 203.10.5.23        192.168.2.23       172.16.80.91       192.31.7.130
--- 203.10.5.23        192.168.2.23       ---                ---
--- ---                ---                172.16.80.91       192.31.7.130
NATrouter#
```

正如前面一段所描述的，一个 NAT 条目可以是静态的也可以是动态的。静态条目是本地地址和全局地址一对一的映射。也就是说，一个唯一的本地地址映射为一个唯一的全局地址。动态条目可以是一对多或多对一的映射。一个多对一的映射意味着多个地址可以映射成一个单一的地址， 而在一对多的映射中， 一个地址可以映射成多个可用地址中的一个。

下面的章节描述几个 NAT 的常见的应用，并且更加清晰地说明静态 NAT 和动态 NAT 不同的执行方式是如何运作的。

4.1.2　NAT 和 IP 地址的保存

NAT 最初出现的目的就是要缓解 IP 地址紧张的局面，这也是 RFC 1631 的重点。这个概念主要是假设在任何时候一个企业都只有一些主机需要与 Internet 相连， 而其他一些设备(例

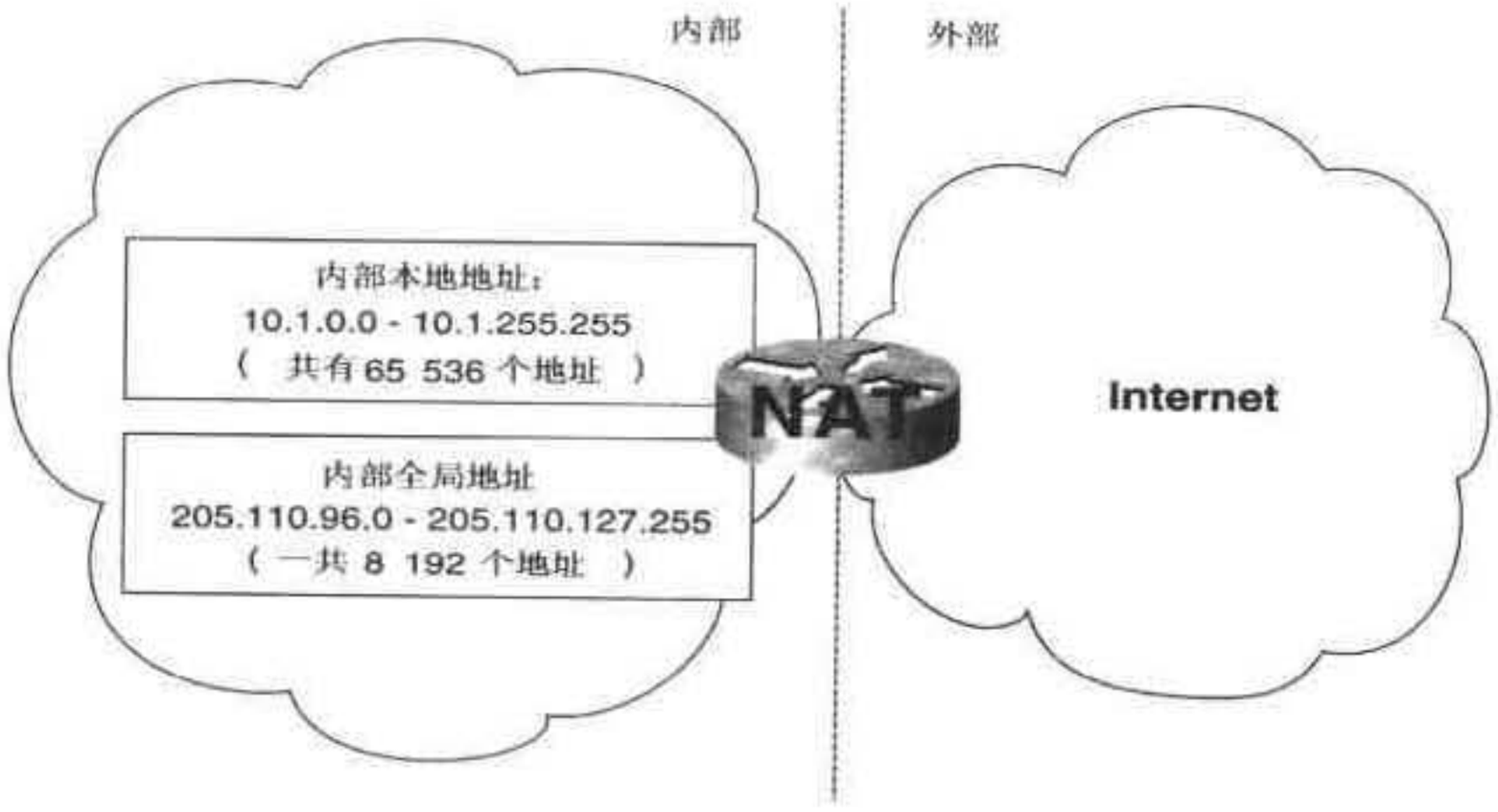


图 4-3　在这个 NAT 设计中，一个公用 IP 地址池为这些地址空间 8 倍的空间专用地址服务

如，打印服务器以及 DHCP 服务器)根本不需要企业外部的连接。因此企业可以在专用 RFC 1918 地址空间以外进行寻址，有数量相当少的一些唯一分配的公开地址放在了位于企业网边缘的 NAT 的一个地址池中，如图 4-3 所示。非唯一的专用地址是 IL 地址，而公开地址是 IG 地址。

当一个内部设备向 Internet 发送数据包的时候，NAT 在内部全局地址池中动态地选择一个公开地址并把它映射成设备的内部本地地址。这个映射被加入到 NAT 表中。例如，例 4-2 显示出图 4-3 中企业网中的三个内部设备——10.1.1.20、10.1.197.64 和 10.1.63.148——通过 NAT 向外发送数据包。来自 IG 组的三个地址——205.110.96.2、205.110.96.3 和 205.110.96.1 分别一一被映射为 IL 地址。

例 4-2　图 4-3 中来自内部本地地址空间的三个地址被自动映射为内部全局地址池内的三个地址

```
NATrouter#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
--- 205.110.96.2       10.1.1.20          ---                ---
--- 205.110.96.3       10.1.197.64        ---                ---
--- 205.110.96.1       10.1.63.148        ---                ---
NATrouter#
```

应答内部设备的外部设备的任何数据包的目的地地址都是 IG 地址。因此，初始映射必须在 NAT 表中保持一定的时间从而保证一个特定连接的所有数据包的翻译是一致的。当同一个设备定期地向同一个或者多个外部目的地发送数据包时，把 NAT 表中的一个条目保持一定的时间也减少了随后的查找工作。

把一个条目第一次放入 NAT 表时就启动一个定时器；定时器的时间段为翻译超时时间(*translation timeout*)。每次使用该条目翻译后续数据包的源地址或者目的地地址时，都重新启动定时器。如果定时器超时了，就从 NAT 表去掉该条目并且将动态分配的地址放回到地址池中。Cisco 缺省的翻译超时时间是 86400 秒(24 小时)；可以通过 **ip nat translation timeout** 指令来改变这个缺省值。

> 注：　根据协议，不同缺省的翻译超时时间也不同。在本章后面的表 4-3 中，给出了这些不同的值。

一个特殊的 NAT 应用是多对一的应用，因为对于地址池中的每个 IG 地址，都可能会有多个 IL 地址映射成它。在图 4-3 所示的实例中，存在着 8 对 1 的关系。这是一个很常见的概念——当 telcos 在设计只处理部分用户的交换机和中继的时候，就使用了这个概念，当航班超订的时候，航空公司也会使用这个概念。可以将这个概念看作是 IL 地址到 IG 地址的统计复用。对于 telcos 和航空公司来讲，使用这个概念的风险是他们可能低估了峰值使用时间以及容量的耗尽。

对于本地地址空间规模和地址池规模的比例没有限制。在图 4-3 中，IL 范围和/或 IG 范围可以更大或者更小从而满足特殊的要求。例如，IL 地址范围 10.0.0.0/8 包含了 16 000 000 个地址，可以映射到一个有 4 个地址的地址池或者更小的地址池，这 4 个地址是 205.110.96.1

——205.110.96.4。真正的限制不在于指定 IL 地址范围内的地址数量，而是在于使用这个范围内地址的真正设备数。如果只有 4 个设备使用 10.0.0.0/8 范围外的地址，那么在地址池中并不需要多于 4 个的地址。如果在内部有 500 000 个设备，那么就需要一个更大的地址池。

当动态地址池内的一个地址在 NAT 表中时，它就不能映射为任何其他的地址。如果地址池内的所有地址都被占用了，NAT 路由器就不能翻译接下来的试图通过它的内部数据包并将之丢弃。因此，保证 NAT 地址池足够大、翻译超时时间足够短是十分重要的，只有这样，动态地址池才不会用尽。

几乎所有的企业都有从外部可以访问的一些系统，例如邮件、Web 以及 FTP 服务器等。这些系统的地址必须保持相同；否则外部的主机就不知道如何再次访问同一个系统了。因此，对于这些系统，不能使用动态 NAT；它们的 IL 地址与 IG 地址之间必须是静态的映射。用于静态映射的 IG 地址不能包括在动态地址池中；虽然这些 IG 地址永久地存在于 NAT 表，但是在动态地址池中仍然可以选到同样的地址，这样可能会产生地址歧义。

本节所描述的 NAT 技术对于衡量一个正在发展的企业来讲是十分有用的。它无须不断地从地址分配机构或者 ISP 申请地址空间，可以将现存的公开地址转移到 NAT 地址池中并且在专用地址空间中为内部设备重新编号。根据公司的规模和现存地址分配结构，可以把重新编号作为一个独立的项目来进行。

4.1.3　NAT 和 ISP 的变更

正如在第 2 章“BGP-4 的介绍”中所讨论的，CIDR 的一个缺点就是它增加了改变 Internet 业务供应商的难度。如果你从 ISP1 处得到了一个地址块，但是你想变更到 ISP2，那么你必须归还 ISP1 的地址并且重新向 ISP2 申请一个新的地址范围。地址的归还意味着要在企业内进行一个令人头疼的、开销很大的地址改写项目。

> 提示：　不要过分地强调如果首先地址方案设计得好就可以急剧地减轻地址转移所带来的痛苦并降低开销。

假设你是 ISP1 的一个用户，该 ISP1 的 CIDR 块为 205.113.48.0/20 并且该 ISP 分配给你的地址空间为 205.113.50.0/23。然后你决定将 Internet 业务转到 ISP2，该 ISP2 的 CIDR 块为 207.36.76.0/19，而它分配给你的新的地址空间为 207.36.76.0/23。此时你可以使用 NAT 而不必为你的内部系统重新编号(如图 4-4 所示，这个企业有一个属于 ISP1 的内部本地地址空间，但是它却是 ISP2 的一个用户，它使用 NAT 将 IL 地址翻译成分配给 ISP2 CIDR 块之外的 IG 地址)。205.113.50.0/23 这个地址空间已经归还给 ISP1，但是你可以继续使用该地址空间，把它们用于 IL 地址。虽然这些地址来自公开地址空间，但你不能再用它们代表你与公共 Internet 相连，你可以把来自 ISP2 的地址 207.36.76.0/23 作为 IG 地址，并且把 IL 地址映射(静态或者动态地)为这些 IG 地址。

使用这个方案存在着这样一种危险的可能性，即任何一个内部本地地址都可能泄漏给公共 Internet。如果出现了这种情况，这个被泄露的地址就可能与具有该地址合法使用权的 ISP1 发生冲突。如果 ISP2 使用了适当的、但是超过正常范围的路由过滤器，这样的一个错误不会导致向 Internet 泄漏地址。但是，正如第 2 章所强调的，你永远也不要假设能够正确地过滤一个外部 AS 对等。因此，在这方面你要极为注意从而保证在数据包进入 ISP2 之前，已经翻

译了所有的 IL 地址。

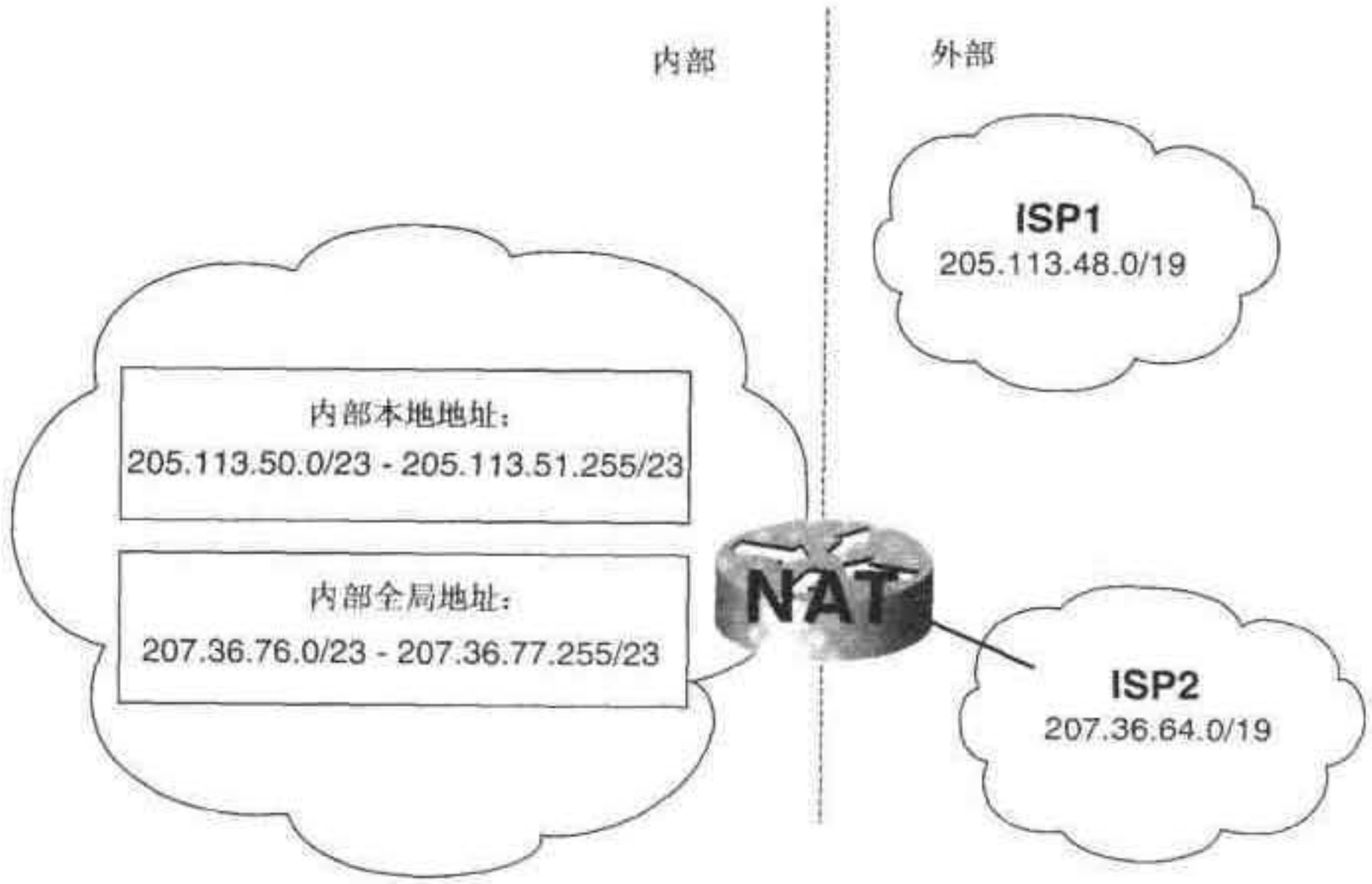


图 4-4　用 NAT 实现 ISP 的变更

这个方案还有可能引发另外一个问题，即 ISP1 可能将 205.113.50.0/23 范围再分配给另外一个用户。那个用户对于你来讲就是不可到达的。例如，假设你的网络上的一个主机想给 newbie@ISP1.com 发送一个数据包，DNS 将目的地地址翻译为 205.113.50.100，于是该主机就使用了这个地址。但是，不幸的是，解释出来的这个地址属于你的本地网，因此该地址是错误路由或由于不可到达而被丢弃。

解释上述这个例子的意义在于说明对于临时的情况来讲，本节所描述的转移方案对于立即转移这种情况的复杂性的降低是十分有用的。但是，从长远的角度讲，你还是应该用专用地址来为你的网络重新编排地址。

4.1.4　NAT 和多宿主 AS

CIDR 的另一个缺点是多宿主到不同的业务供应商变得十分困难。图 4-5 重新提出了第 2 章曾经讨论过的问题。一个用户多宿主到 ISP1 和 ISP2，它的 CIDR 块是 ISP1 块的一个子集。为了建立与 Internet 的正常通信，ISP1 和 ISP2 都必须公布用户具体的地址空间 205.113.50.0/23。如果 ISP2 没有公布这个地址，所有该用户的来话业务量都通过 ISP1 来转发。而且如果 ISP2 公布了 205.113.50.0/23，而 ISP1 只公布了它自己的 CIDR 块，所有该用户的来话业务量都会与更具体的路由相匹配，从而这些业务量只通过 ISP2 来转发。这就会出现几个问题：

- ISP1 必须在它自己的 CIDR 块上“打一个洞”，这可能意味着要调整许多路由器上的过滤器和相关的策略。
- ISP2 必须公布竞争者的部分地址空间，这种行为对于两个 ISP 来讲都可能是不可接受的。

- 公布用户更具体的地址空间可能会对 CIDR 控制 Internet 路由表规模的有效性方面有一定的影响。
- 一些国家业务供应商不会接受长于/19 的前缀，这就意味着 Internet 的有些部分可能无法知道用户通过 ISP2 的路由。

图 4-6 给出通过 NAT 来帮助解决在多宿主环境下 CIDR 问题的方法。翻译功能在与 ISP2 相连的路由器上设置，IG 地址池是由 ISP2 分配的一个 CIDR 块。ISP2 不再公布 ISP1 的地址空间，因此 ISP1 也就不再需要公布用户更具体的聚合地址。用户企业内的主机既可以通过选择最近的边缘路由器也可以通过已经建立的策略来访问 Internet。无论主机发出的数据包要经过哪个路由器，它们的 IL 地址都是相同的；但是，如果数据包被转发到 ISP2，那么这个 IL 地址就需要翻译。因此，从 Internet 的角度来看，根据转发数据包的 ISP 的不同，来自用户的数据包的源地址也会有所不同。

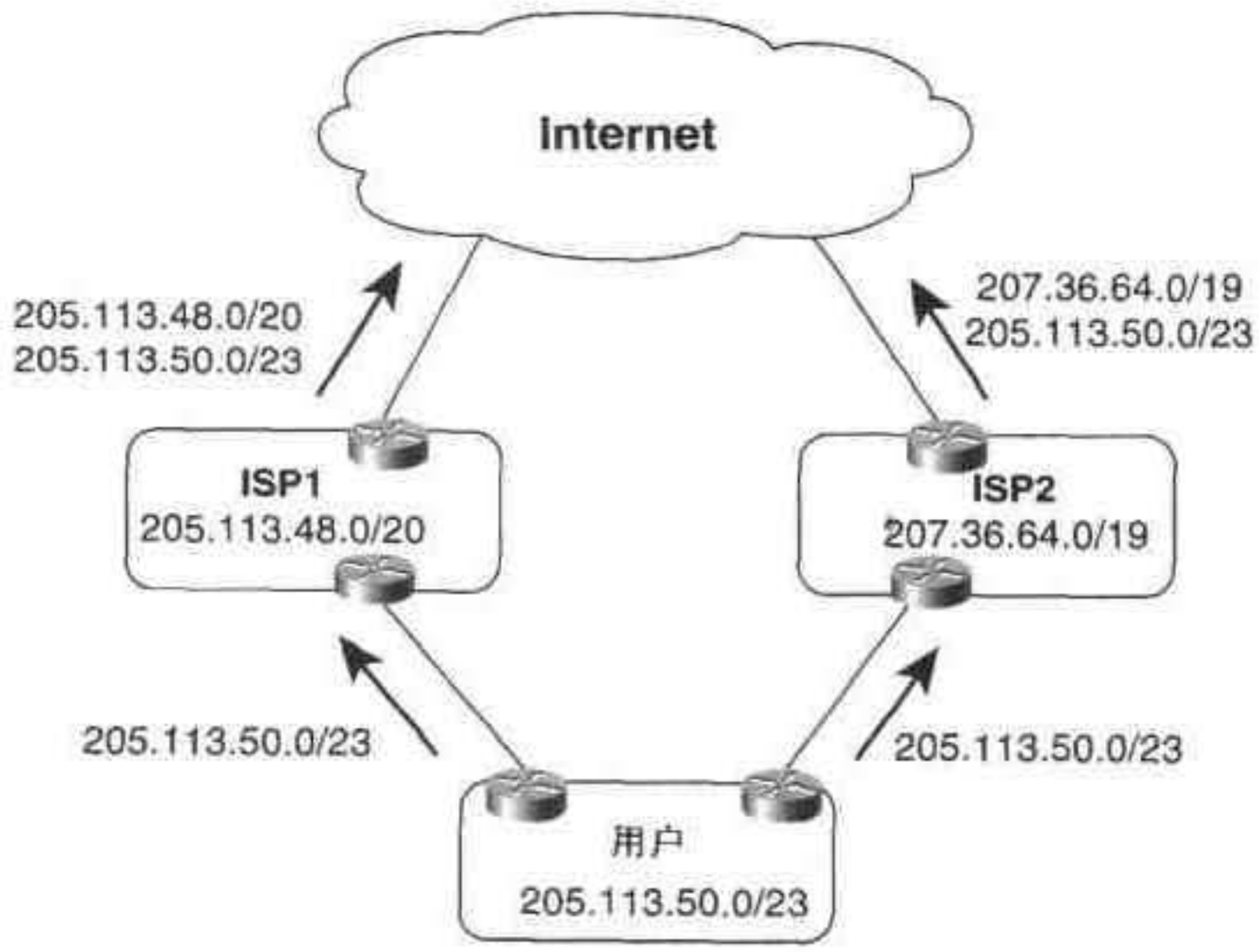


图 4-5　多宿主用户的 CIDR 块产生的问题

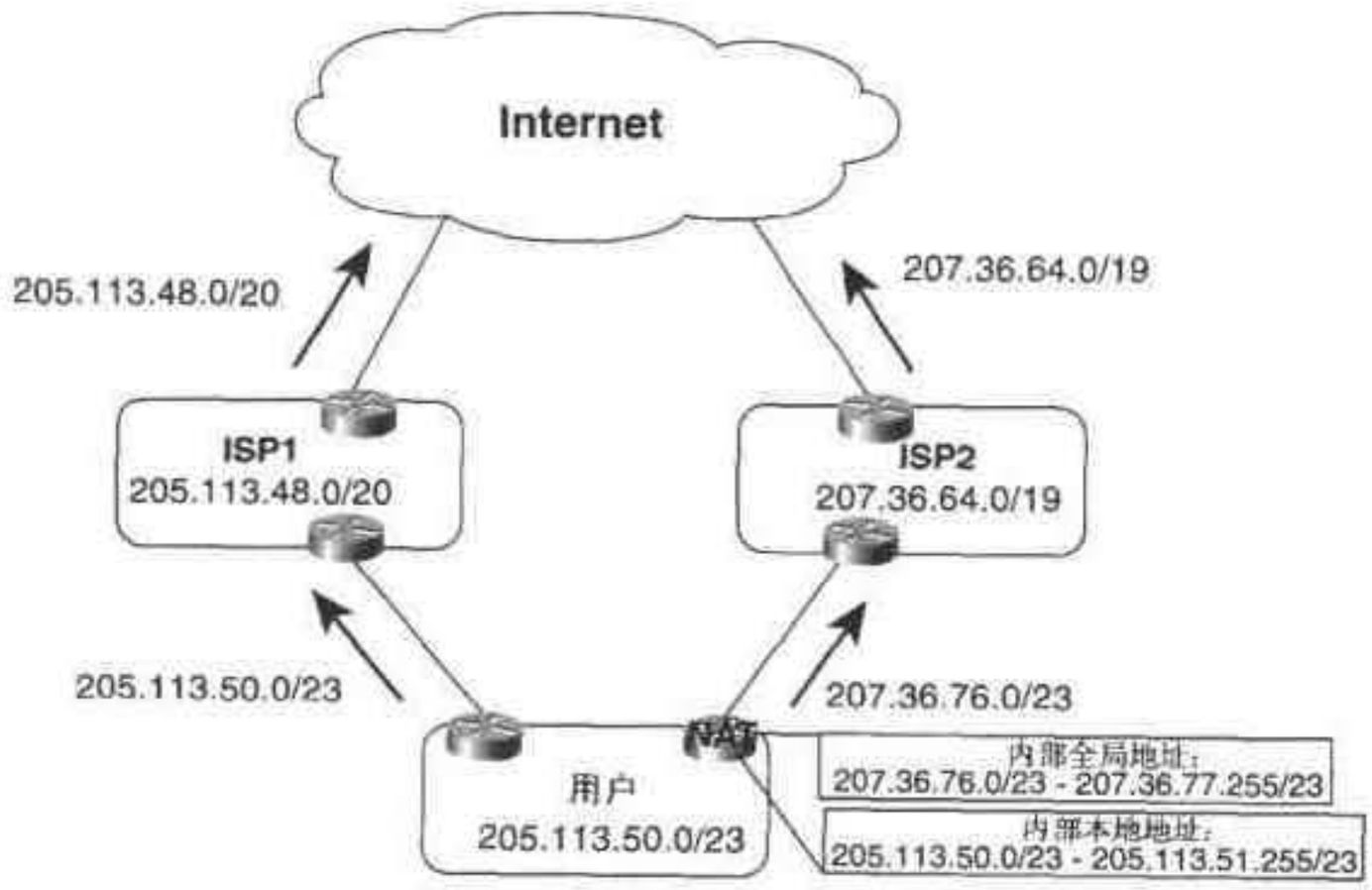


图 4-6　用 NAT 解决图 4-5 所描述的 CIDR 问题

图 4-7 给出了一个更有效的设计：在两个边缘路由器上都实施了 NAT，而且来自每个 ISP 的 CIDR 块分别变为相应 NAT 的 IG 地址池。IL 地址来自专用的 10.0.0.0 地址空间。该企业可以相对容易地改变 ISP，当 ISP 发生变化的时候，它只需要重新配置一下 IG 地址池。

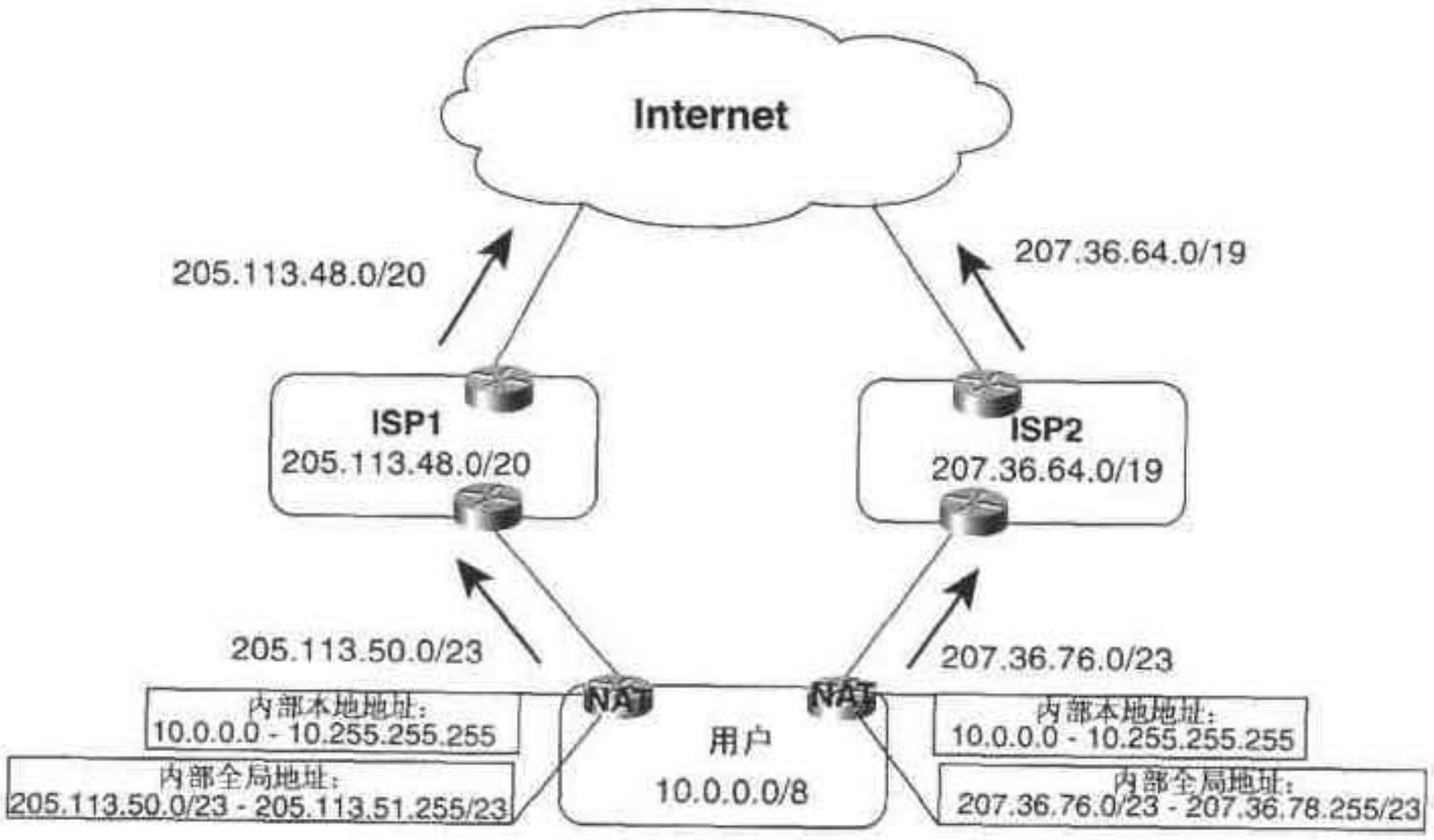


图 4-7　该企业的 IL 地址与任何 ISP 都没有关系：所有的 ISP CIDR 块都分配给 NAT 内部全局地址池

4.1.5　端口地址翻译

对 NAT 多对一的应用讨论到现在，涉及到一个将大范围的地址统计多路复用到一个小型地址池内的问题。但是，这是单个地址一对一的映射问题。例如当将一个来自内部全局地址池的地址映射为一个内部本地地址时，只有在该映射被清除以后，这个 IG 地址才能再映射为其他的地址。但是，NAT 有一个特殊的功能，它允许多个地址同时映射为一个地址。Cisco 把这种功能称为端口地址翻译(PAT)。同样的功能在有些文章中被称为网络地址和端口翻译(NAPT)或者 IP 伪装*(IP masquerading)*。有些时候还把它称为地址过载*(address overloading)*。

一个 TCP/IP 会话并不等同于两个 IP 地址之间数据包的交换，但它与两个 IP Socket 之间的交换相似。一个 socket 是一个(地址，端口)组。例如，一个 Telnet 会话可能包含 192.168.5.2,23 和 172.16.100.6,1026 之间的数据包交换。PAT 翻译了 IP 地址和端口地址。来自不同地址的数据包可能被翻译成一个通用地址，但是它们被翻译成该地址的不同端口地址，因此它们可以共享同一个地址。图 4-8 给出了 PAT 的工作模式。

四个带有内部本地地址的数据包到达 NAT。注意数据包 1 和 4 来自同一个地址的不同源端口，数据包 2 和 3 来自不同的地址但是具有相同的源端口。这四个数据包的源地址都被翻译成同一个内部全局地址，但是因为这些数据包被翻译成不同的源端口地址，因此它们仍然保持着唯一性。通过翻译端口，大约有 32 000 个不同的内部本地 socket 可以翻译为一个内部全局地址。结果是，PAT 对于小型办公室/家庭办公室(SOHO)装置来讲是一种非常有用的应用，因为可以有几个设备可以共享与 ISP 连接上指配的一个地址。

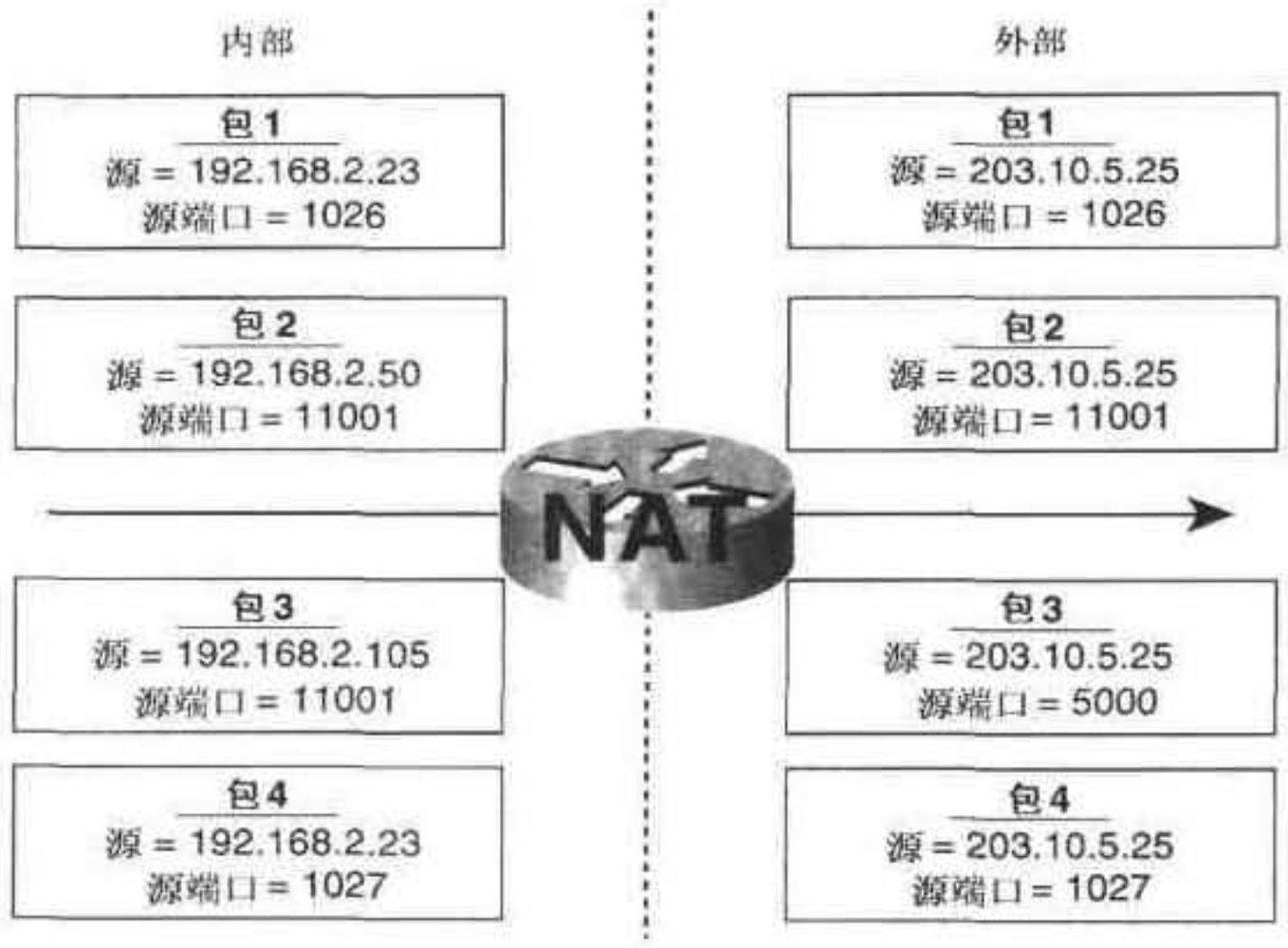


图 4-8　通过翻译 IP 地址和相关的端口地址，PAT 允许多个主机同时使用一个全局地址

4.1.6　NAT 和 TCP 负载分配

可以使用 NAT 来代表多个、同样的服务器，就如同它们具有同一个地址一样。在图 4-9 中，外部的一个设备到达一个地址为 206.35.91.10 的服务器，如图所示，TCP 数据包发送给一个服务器组，该服务器组由同一个地址 206.35.91.10 所代表，将这个地址通过循环方式翻译给四个相同服务器的实际地址。实际上，在内部有四个与之相对应的服务器，NAT 以循环的方式在它们之间分配会话。注意来自不同信源的数据包 1 到 4 的目的地地址被翻译到服务器 1 到 4。代表来自另外一个源的会话的数据包 5 被翻译到服务器 1。

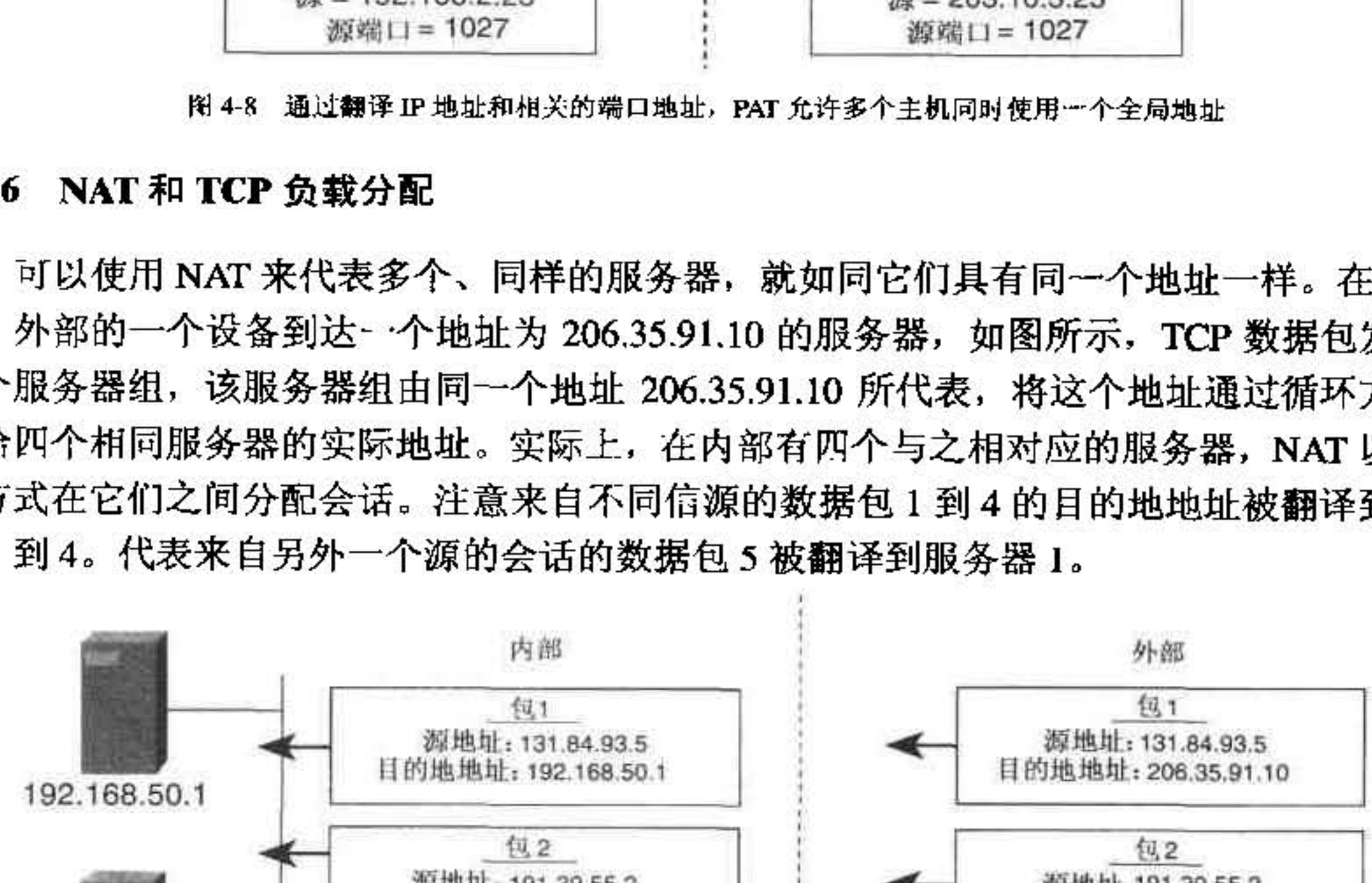


图 4-9　NAT 实现 TCP 负载分配

很显然，图4-9中四个服务器的可访问内容必须是一样的。一个访问服务器组(server farm)的主机可能这次遇到服务器2，下次就遇到服务器4，但是在这两种情况下，对于主机来讲，它遇到的是同一个服务器。

这个方案与基于DNS的负载均衡相似，在DNS的负载均衡方式中，一个名字以循环方式被解析为几个IP地址。基于DNS的负载均衡的缺点在于当一个主机接收到名字/地址解析时，它将这个解析缓存起来，未来的会话会被发送到同一个地址，这就降低了负载均衡的效率。而基于NAT的负载均衡仅当从外部建立了一个新TCP连接时才会执行地址翻译，从而可以均匀地分配会话。在NAT TCP负载均衡过程中，非TCP数据包会保持原样通过NAT，没有翻译过程。

基于NAT的负载均衡与基于DNS的负载均衡都是不健壮的，注意到这一点很重要。NAT没有办法知道什么时候一个服务器会停机，它会继续将数据包发送给那个地址，因此，一个出现故障或者离线的服务器都会导致发送给这个服务器组的一些业务出现了黑洞效应。

4.1.7　NAT和虚拟服务器

NAT还允许将业务分配给不同的地址，从而给人的感觉是通过一个地址可以访问所有的业务(参见图4-10)。

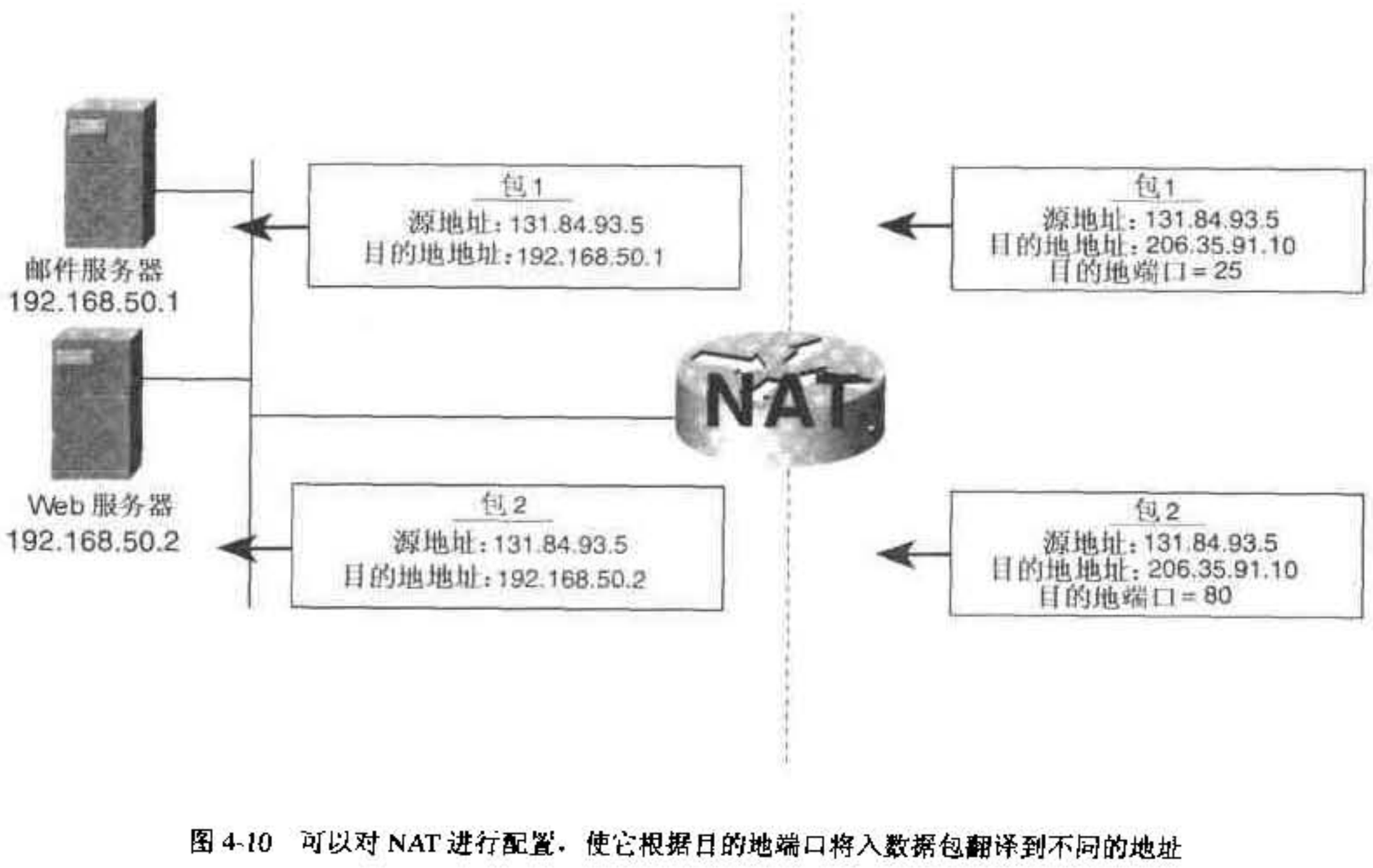


图4-10　可以对NAT进行配置，使它根据目的地端口将入数据包翻译到不同的地址

在图4-10中，企业有一个邮件服务器，本地地址是192.168.50.1；还有一个HTTP服务器，本地地址是192.168.50.2，两个服务器都有一个全局地址206.35.91.10。当外部一个主机向内部发送数据包时，NAT除了检查数据包的目的地地址，还要检查它们的目的地端口。如图4-10所示，一个主机向206.35.91.10发送了一个数据包，目的地端口是25，代表的是邮件，NAT就将这个数据包的目的地地址翻译到邮件服务器的地址192.168.50.1上。而来自同一个主机的第二个数据包的目的地端口是80，代表的是HTTP，NAT就将这个数据包的目的地地

址翻译到 Web 服务器的地址 192.168.50.2 上。

4.2　NAT 的问题

虽然到目前为止所描述的 NAT 的应用都是简单易懂的，但是，NAT 的基本功能就不这么明确了，其原因有两个：

- 对 IP 和 TCP 信头的综合处理
- 一些特殊协议和应用的特点

改变 IP 地址或者 TCP 端口的内容可能就会改变其他一些字段尤其是校验和的意义。而且许多协议和应用在它们的数据字段内携带 IP 地址或者基于 IP 地址的信息。改变信头中的 IP 地址可能就改变了封装的数据的意义，而且还有可能会中断应用。本节将讨论 NAT 操作方面的一些常见问题。

4.2.1　信头校验和

IP 数据包的校验和是根据整个信头计算出来的。因此，如果源地址或者目的地址发生变化，或者两个地址都发生变化，就必须重新计算校验和。对于 TCP 信头也是一样的，校验和是根据 TCP 信头和数据计算出来的，而且还要依据包括源地址和目的地地址的伪信头计算。因此，如果一个 IP 地址或者一个端口号发生变化，TCP 校验和就必须改变。Cisco 的 NAT 运行这些校验和的重新计算。

4.2.2　分段

在章节“NAT 和虚拟服务器”中曾讲过，可以使用 NAT 根据目的地端口号将数据包翻译给不同的本地地址。例如，将目的地端口为 25 的数据包翻译到特定的 IL 地址，同时，具有其他目的地端口号的数据包被翻译到其他的地址。但是，如果目的地端口为 25 的数据包在到达 NAT 之前，在网络上的某点被分段了，结果会如何呢？包含源和目的地地址端口号的 TCP 或者 UDP 信头，只在第一个分段中。如果只翻译并且转发了这个段，NAT 就无法知道是否必须要翻译后续的分段。

IP 不能保证数据包能够按顺序传送。因此很有可能后面的分段都已经到达 NAT，而第一个分段还没有到达。必须设计 NAT 让它能够处理这种可能的情况。

Cisco 的 NAT 保留了有关分段的有用信息。如果翻译了第一个分段，有用信息就被保留下来，从而能够按同样的方式翻译后续的分段。如果有分段在第一个分段之前到达了 NAT，NAT 就只能先缓存这个分段直到第一个分段到达 NAT 并接受了处理以后，NAT 才开始处理后续的分段。

4.2.3　加密

Cisco 的 NAT 能够改变许多应用中数据字段所携带的 IP 地址信息，你很快就会看到这部分内容。但是，如果数据字段被加密了，NAT 就无法阅读这个数据。因此，如果要执行 NAT 功能，IP 地址和任何从它们(例如 TCP 信头校验和)获得的信息都不能被加密。另外一个需要关注的就是虚拟专用网(VPN)的使用，例如它对 IPSec 的使用。对于一定的 IPSec 模式，如果

IPSec 数据包中的一个 IP 地址发生变化，IPSec 就变得没有意义，而且 VPN 也被破坏。当使用了任何形式的加密以后，必须把 NAT 放在安全一侧，而不能放在加密的路径上。

4.2.4　安全性

有些人还把 NAT 看作安全计划的一部分，因为它对外隐藏了内部网络的细节。一个翻译过的主机今天可能以一个地址出现在 Internet 上，明天它可能就使用了另外的地址。但是这种情况的安全性最差。虽然 NAT 可以通过强迫攻击者去玩一种通过 IP 地址的骗局，从而减缓该攻击者攻击一个特定主机的速度，但是它无法阻止任何态度坚决而且知识渊博的攻击者。而且，更严重的是，NAT 不会采取任何行动来阻止普通的攻击，例如拒绝服务或者窃取会话。

4.2.5　具体协议涉及到的问题

对于通过 NAT 发送数据包的终端系统来讲，NAT 应该是透明的。但是，许多应用——商业应用或者是属于 TCP/IP 协议集一部分的应用——使用 IP 地址。在数据字段中的信息可能和 IP 地址有关，或者是数据字段中带有 IP 地址。如果 NAT 翻译了 IP 信头中的地址，但是不知道对数据将要造成的影响，该应用就有可能被破坏。

表 4-1 列出了 Cisco NAT 设备所支持的应用。对于列出的在应用数据中携带 IP 地址信息的应用，NAT 知道这些应用并且会对这些应用的数据给予适当的修改。注意这个表所列出的应用是写本书时的版本。如果你需要使用 NAT，你应该到 Cisco 的网站或者 TAC 中去查找，看看最近是否加入了新的 NAT 能够支持的应用。

表 4-1　Cisco NAT 支持的 IP 业务类型/应用

支持的业务类型/应用
任何在应用数据流中不承载源和/或目的地 IP 地址的 TCP/UDP 业务
HTTP
TFTP
Telnet
archie
finger
NTP
NFS
rlogin,rsh,rcp
支持的在其数据流中具有 IP 地址的业务类型/应用
ICMP
FTP(包括 PORT 与 PASV)
TCP/IP 上的 NetBIOS(数据报、名称与会话服务)

续表

支持的在其数据流中具有 IP 地址的业务类型/应用
Progressive Network 的 RealAudio
White Pines 的 CuSeeMe
Xing Technologies 的 SteamWorks
DNS A 与 PTR 的查询与响应
H.323/NetMeeting [12.0(1)/12.0(1)T 与更新版本]
VDOLive [11.3(4)/11.3(4)T 与更新版本]
Vxtreme [11.3(4)/11.3(4)T 与更新版本]
IP 多播[12.0(1)T](只翻译源地址)
不支持的业务类型/应用
路由表更新
DNS 区域传送
BOOTP
talk、ntalk
SNMP
NetShow

表 4-1 可以在 Cisco 书面文件“Cisco IOS 网络地址翻译(NAT)包更新”中直接得到，网址为 www.cisco.com。

1．ICMP

一些 ICMP 消息含有导致产生该消息的数据包的 IP 信头。表 4-2 列出了这些消息类型。Cisco 的 NAT 检查列出的消息类型；如果消息中的 IP 信息与信头中翻译过来的一个 IP 地址相匹配，那么 NAT 也要翻译 IP 信息。除此以外，NAT 修改 ICMP 信头中校验和的方式和修改 TCP 和 UDP 中校验和的方式相同。

表 4-2　　在消息主体中携带 IP 信头信息的 ICMP 消息类型

消　息	类 型 号
Destination Unreachable	3
Source Quench	4
Redirect	5
Time Exceeded	11
Parameter Problem	12

2. DNS

任何 TCP/IP 网络尤其是 Internet 的一个核心功能就是域名系统(DNS)。如果一个系统通过 NAT 无法得到 DNS 询问和应答，DNS 就会变得复杂。图 4-11 给出一个例子，即在 NAT 的周围部署 DNS 服务器，但是该 DNS 无法翻译 DNS 数据包。

图 4-11 中的 NAT 在两个方向上翻译——对于内部来讲，外部的主机看起来是在 10.0.0.0 这个网络上，而对于外部来讲，内部主机看起来是在 204.13.55.0 这个网络上。在内部和外部都设有 DNS 服务器，而且每台服务器都包含将名字映射为对它那一侧的 NAT 来讲是适当地址的资源记录。

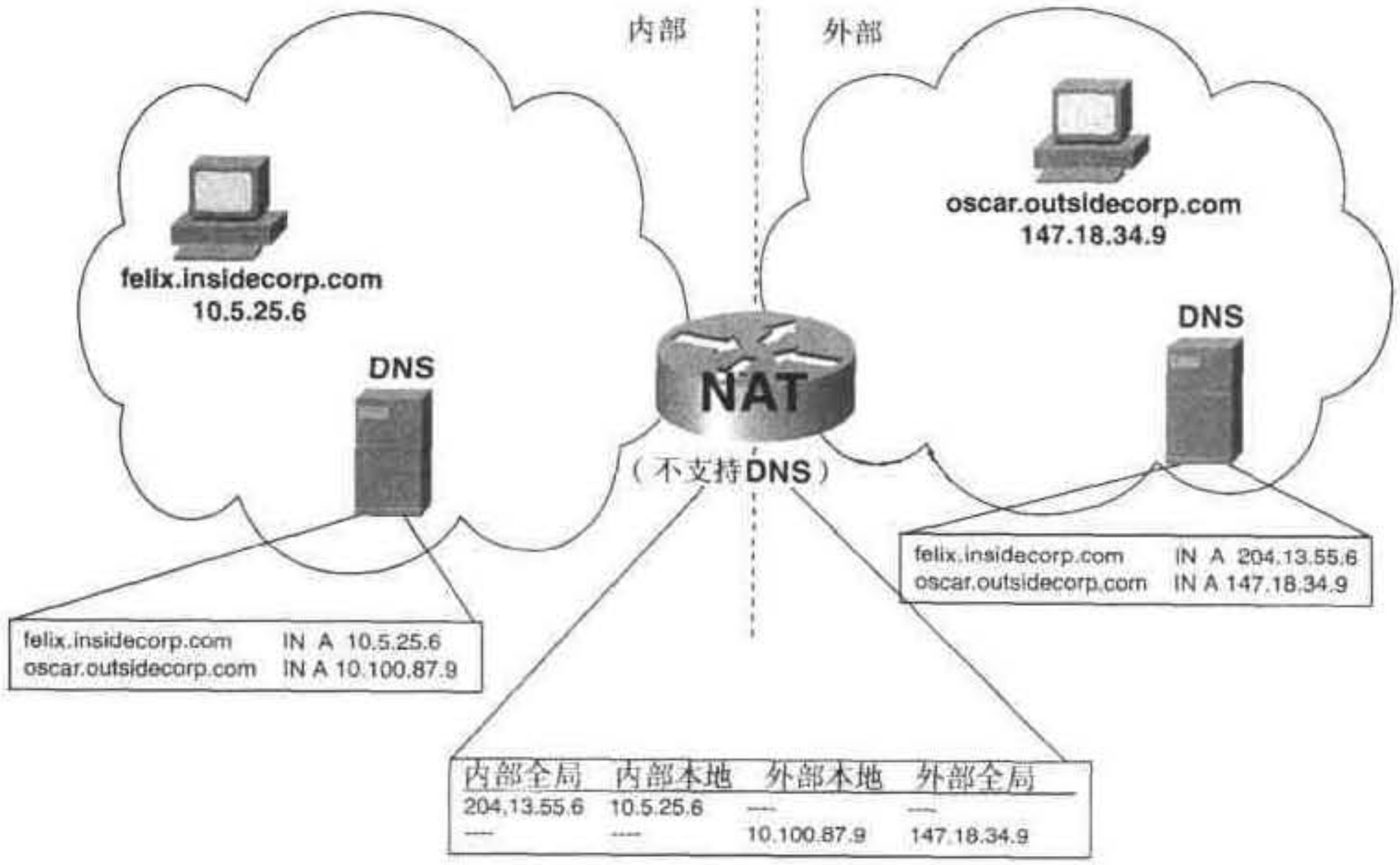


图 4-11　如果 NAT 不支持 DNS，那么域名服务器必须放在 NAT 点的两侧

这样的方法有一个问题，即在两个 DNS 服务器上维护不一致的记录时有一定的难度。一个更严重的问题是 NAT 映射必须是静态的，以匹配 DNS 资源记录的映射。不合理的 NAT 由于映射动态变化而不能工作。一个更好的方法，也是 Cisco 支持的实现 NAT 的方法，就是让 NAT 支持 DNS 查询翻译。

虽然对 DNS 操作的详细讨论已经超出了本书的范围，但对关键概念的简单学习会帮助你理解在哪里 DNS 可以与 NAT 共存，而在哪里不行。你对域名结构已经很熟悉了；比如域名 cisco.com 描述的是从顶级域名 com 之下二级域名 cisco。所有的 IP 命名空间为树型结构，从主机名顺次到较高级域名，直到根一级的域名。

> 注：　Paul Albitz 与 Cricket Liu 的“DNS 和 BIND”(O'Reilly and Associates，1992)是一个关于 DNS 的优秀文件。

域名服务器存有有关一部分域名空间的信息。在某一个特别的域名服务器上的信息可能是整个域的信息以及其他域的一部分信息，甚至是多个域的信息。服务器包含信息的那一部分域名空间是服务器的区域(zone)。

DNS 服务器分为主服务器与次服务器，一台主服务器从在其运行的主机的本地存放的文件中处获得区域的信息，这台服务器称为在这个区域内受权的服务器。次服务器从主服务器处获得区域内的信息。它通过在一个被称为区域传送的过程中下载主服务器区域文件来获取信息。

因为一个区域传送是文件传送，所以 NAT 不能解析这个文件中的地址信息。即使它可以做到这一点，区域文件通常也会很大，这会给 NAT 设备增加相当大的负担。因此，对于相同的区域，主服务器与次 DNS 服务器不能位于 NAT 两侧，因为区域文件中的信息在区域传送过程中不会被翻译。

区域文件中的信息由称为资源记录(RR)的条目构成。这些资源记录有多种类型，比如起始权威机构(Start-of-Authority SOA)记录，它指明这个域的受权服务器；规范域名(Canonical Name CNAME)记录，它指明别名；邮件交换(Mail Exchange MX)记录，它指明这个域的邮件服务器；视窗 Internet 命名服务器(WINS)记录，它在一些 Windows NT 服务器上使用。对于 NAT 来说，两个最重要的记录是将主机名映射到 IP 地址的地址(A)记录与将 IP 地址映射到主机名的指针(PTR)记录。当一台主机必须找到某一主机名的 IP 地址，它的 DNS 解析器查询 DNS 服务器中的 A 记录。如果主机要找到某一 IP 地址的主机名(反向查询)，它查询服务器的 PTR 记录。

图 4-12 显示一个 DNS 消息的格式，这个消息同时携载主机的查询与服务器的响应。这个头与其他头一样，是一组携带管理信息与消息处理的场。对于 NAT 来说很重要的头信息包括指明其为查询还是响应的一个比特和指明其余四部分中每一部分所含 RR 数目的场。

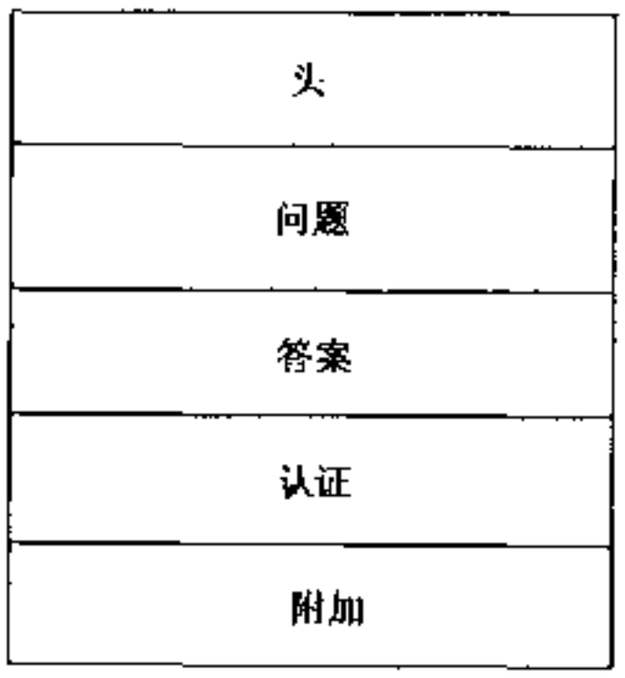


图 4-12　DNS 消息格式

问题部分如同其名称明确指示的那样，为一组向服务器询问的问题项。这里的问题可能包含一个主机名，服务器必须在从它的 A 记录中找到与之匹配的地址；这里的问题也可能包含一个地址，服务器必须在它的 PTR 记录中找到与之匹配的主机名。每一个 DNS 消息都包含一个问题，且只能包含一个问题。

答案部分包含 RR，当然，这就是对问题的回答。这个答案可能列出一个或多个 RR 或根本就没有 RR。权威机构部分与附加部分中包含对这个答案的补充信息，也可能为空。

当一个 DNS 包通过 Cisco NAT，问题、答案与附加部分均会受到检查。如果这个消息为对一个 IP 地址匹配名称的查询，不包括任何地址，那么答案与附加部分为空，不进行任何翻译。可是，如果是对一个查询的响应，答案部分或可能在附加部分中包括一个或多个 A RR。

NAT 在它的表中检索匹配这些记录的地址，如果有匹配的就对其进行翻译。如果没有就丢弃这个消息。

如果 DNS 消息为对匹配一个已知 IP 地址的名称的查询(反向查询)，NAT 检查它的表中是否有与问题部分地址相匹配的地址。如果有匹配的就翻译这个地址，否则这个消息被丢弃。对这个查询的响应中的答案部分或可能在附加部分中包含一个或多个 PTR RR，这些记录中的地址被翻译，或者这个消息被丢弃。

总的来说，当 DNS 与 NAT 同时使用时，要记住如下两点事实：

- DNS 的 A 与 PTR 查询可以通过 Cisco NAT，所以 NAT 一侧的主机可以查询 NAT 另一侧的 DNS；
- DNS 区域传送不能通过 Cisco NAT，所以同一区域主 DNS 服务器与次 DNS 服务器不能位于 NAT 的两侧。

3．FTP

文件传送协议(FTP)是种很不寻常的应用协议，它占用了两个连接(见图 4-13，图中显示一个 FTP 会话由两个独立的 TCP 连接组成，主机发起控制连接，服务器发起数据连接)。控制连接由主机发起，用于与主机交换 FTP 命令。数据连接由服务器发起，用于实际的文件传送。

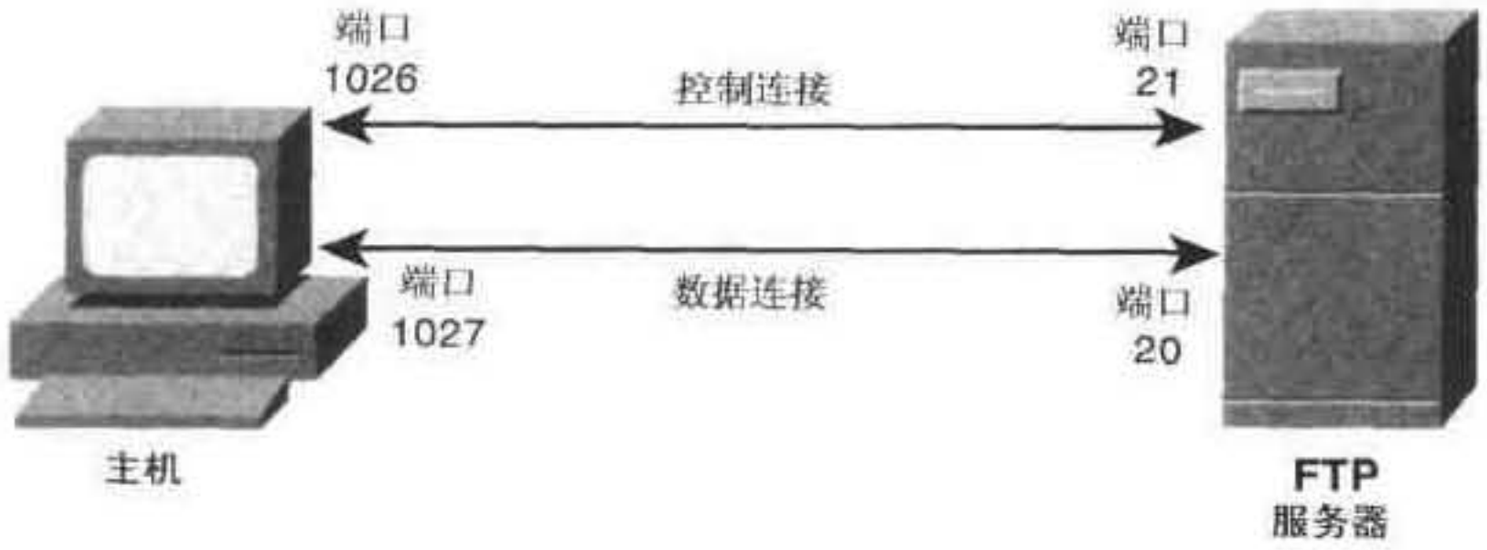


图 4-13　FTP 的连接

建立 FTP 会话和传送文件发生的事件顺序如下：

(1) FTP 服务器在 TCP 端口 21 有一个被动开路(也就是监听一个连接请求)，这个端口为控制端口；

(2) 主机选择一个暂时的(临时的)端口建立控制连接与数据连接。如图 4-13 所示，分别有两个端口 1026 与 1027；

(3) 主机在数据端口上进行被动开路；

(4) 主机进行控制连接的主动开路，在它的控制端口(在图 4-13 中为 1026)与服务器的端口 21 之间建立 TCP 连接；

(5) 为了传送文件，主机通过控制连接发送 **PORT** 命令，告诉服务器在主机的数据端口(在图 4-13 中为 1027)打开一个数据连接；

(6) 服务器进行数据连接的主动开路，在它的端口 20 与主机的数据端口之间建立 TCP 连接；

(7) 请求的文件通过数据连接传送。

这一系列事件在有些安全防护的网络中会产生问题。特别是配置了防火墙与访问表的网络不允许外部网络发起与随机的端口建立连接，这是一种常见的安全惯例。通过查询 TCP 包头中未设置的 ACK 与 RST 比特，表明这是一个连接请求，这样就可以完成这个检查。你可以看到，当 FTP 服务器要通过防火墙与主机暂时的端口建立连接时，连接会被拒绝。

注：关键词 **established** 可以告诉 Cisco 访问表检查没有设置 ACK 与 RST 比特的 TCP 包头。

为了解决这个问题，主机可以发出 **PASV** 命令代替 **PORT** 命令来建立数据连接。这个命令，要求服务器被动地打开一个数据端口并告诉主机这个端口号。然后主机主动连接服务器的端口，建立数据连接。因为这个连接请求是从防火墙内输出的，所以连接不会被拒绝。

对于 NAT 来说，最关心的是 **PORT** 与 **PASV** 命令中不仅携带端口号而且也携带 IP 地址。如果这个消息通过 NAT，那么这些地址必须进行翻译。更糟的是这 IP 地址用 ASCII 表示成带点的十进制形式。这意味着 FTP 消息中的 IP 地址不像用 32 比特二进制的表示法那样，而是不定长的。比如，地址 10.1.5.4 是 8 个 ASCII 字符(包括点)，而 204.192.14.237 为 14 个 ASCII 字符。所以当地址进行翻译时，消息的长度也改变了。

如果被翻译的 FTP 消息长度仍与原来一样，Cisco NAT 只对 TCP 的校验和进行重复计算(除了 IP 头的许多操作之外)。如果翻译的结果为更短的消息，NAT 用 ASCII 字符 0 来填充以使它与原来的消息长度一样。

如果翻译的消息长度比原来的长度更长，问题就更复杂了，因为 TCP 序号与确认号与 TCP 的分段直接相关。Cisco NAT 保留一张表以跟踪 SEQ 与 ACK 号。当 FTP 消息进行翻译时，就有一个条目加入表中，该条目包括源地址与目的地地址及端口号、起始序列号与序列号差值、时间标签。这些信息可以正确调整 FTP 消息中的 SEQ 与 ACK 号。它在 FTP 连接结束后删除。

4．SMTP

简单邮件传送协议(Simple Mail Transfer Protocol SMTP)消息一般包含域名，而不是 IP 地址。不过，它请求邮件传送时，可以用 IP 地址而不用域名。因此 Cisco NAT 检查 SMTP 消息中相应的参数，当发现 IP 地址时就进行翻译。

SMTP 协议用于上载邮件和在服务器间传送邮件，与之不同，邮局协议(Post Office Protocol POP)与 Internet 消息访问协议(Internet Message Access Protocol IMAP)仅用于从邮件服务器上下载邮件到客户端。这两个协议在消息实体中都只用主机名，而不用 IP 地址，因此，这些协议经过 NAT 时不需要进行特别的检查。

5．SNMP

简单网络管理协议(Simple Network Management Protocol SNMP)采用了丰富多样的消息信息库(MIB)来对各式各样的网络设备进行管理。除了许多 Internet 标准 MIB 组外，各个厂商的设备都创建了自己专用的 MIB。

从这些简单的描述中，你可以推测出许多 MIB 可以包含一个或多个 IP 地址。因为 SNMP 可能有多种消息、格式与变量，NAT 不能很容易地对 SNMP 消息内容中的 IP 地址进行检查。因此 NAT 不支持 SNMP 消息中 IP 地址的翻译。

6. 路由协议

IP 路由协议与 SNMP 有相同的问题。现在有许多种路由协议，每一种都有自己的包的格式与其特有的操作特性。因此，NAT 不能翻译 IP 路由协议包。一台 NAT 路由器可以在内部接口上运行一种路由协议，在外部接口上运行另一种路由协议。但不能有路由协议包穿过 NAT 边界，宣告的地址的改变都发生在这个边界内不论是通过一种路由协议还是经再分发而造成的。这个限制不会产生问题，因为 NAT 路由器会在路由域的边界上设置，因此可以使用默认路由或一组总结的路由地址。

7. Traceroute

路由跟踪实用程序可以有些许变化。有一些程序，如 Cisco 的 **trace** 命令，使用的是 ICMP 包。其他的程序，比如 Microsoft Windows 95 中的 tracert，使用的是 UDP 包。不过它们基本的功能是一样的：发向目的地的包中有不断增加的 TTL，发送 ICMP Time Exceeded 差错消息的中间系统的地址被记录下来。你在前面 ICMP 的章节中看到 Cisco NAT 可以传送 Time Exceeded 消息，所以可以通过 NAT 对路由进行跟踪。

图 4-14 中的 NAT 对双向都进行翻译。路由器 jerry.insidenet.com 有一个 IP 地址 10.1.16.50，它被翻译成 IG 地址 204.13.55.6。设备 berferd.outsidenet.com 有一个地址 147.18.34.9，它被翻译成 OL 地址 10.2.1.3。因此，jerry 认为 berferd 的地址就是这个 OL 地址。

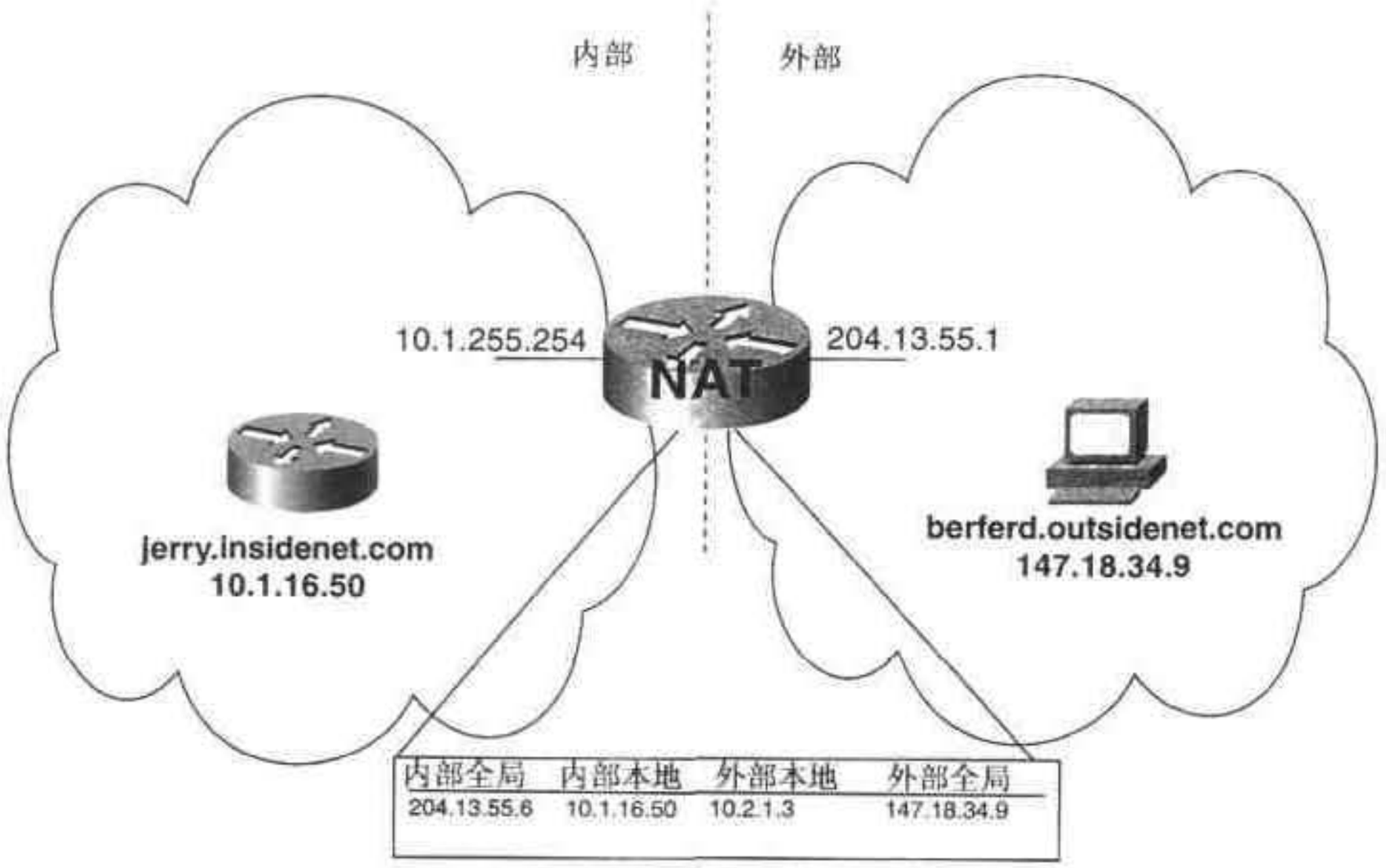


图 4-14　NAT 对双向都进行翻译

当 jerry 对 berferd 进行跟踪时，目标为 10.2.1.3。例 4-3 显示第一跳是 NAT 路由器。然后 NAT 把目的地地址翻译成 147.18.34.9，把源地址翻译成 204.13.55.6，并从其外部接口将这个包转发出去。当 berferd 收到这个包时，这个包是发往一个假的端口，它响应一个 ICMP Port Unreachable 错误包。这个包的目的地地址为 204.13.55.6，源地址为 147.18.34.9。NAT 将目的地地址翻译为 10.1.16.50，将源地址翻译为 10.2.1.3，然后转发给 jerry。因此，这个跟踪是成功的，不过，内部设备只看到了外部设备的本地地址。

例 4-3　如图 4-14 所示，jerry.insidenet.com 对 berferd.outsidenet.com 的跟踪是成功的，NAT 对内部“隐藏”了外部全局地址。

```
Jerry#trace berferd.outsidenet.com

Type escape sequence to abort.
Tracing the route to berferd.outsidenet.com (10.2.1.3)

  1 10.1.255.254 8 msec 8 msec 4 msec
  2 berferd.outsidenet.com (10.2.1.3) 12 msec *  8 msec
Jerry#
```

4.3　配置 NAT

配置 NAT 的第一步是指定内部和外部接口。除此之外，配置还取决于你是采用静态 NAT 还是动态 NAT。对于静态 NAT，你就在 NAT 表中建立相应的映射条目即可。对于动态 NAT，你必须建立一个地址池用于翻译并建立访问表以标识要进行翻译的地址。然后用一条命令把这个地址池与访问表联系起来。

这一部分演示了 NAT 最常见的配置技术。

4.3.1　案例研究：静态 NAT

在图 4-15 中，内部网络地址来自地址空间 10.0.0.0。有两个设备主机 A 与主机 C，它们必须能与外部网络通信。这两个设备翻译成公开地址 204.15.87.1/24 与 204.15.87.2/24。

例 4-4 中显示在 Mazatlan 上实现这个 NAT 配置。

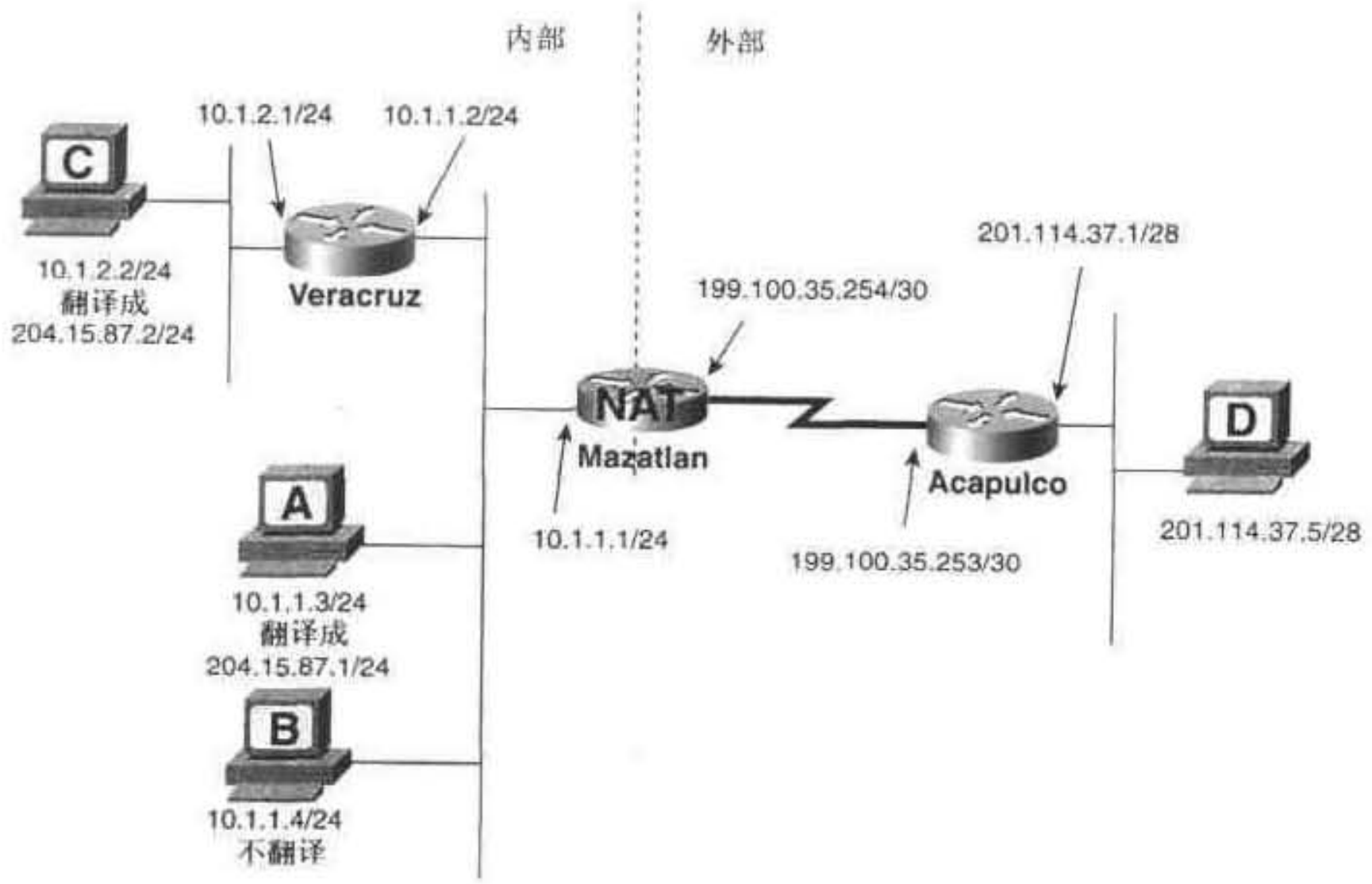


图 4-15　设备 A 与 C 的内部本地地址由路由器 Mazatlan 静态翻译成内部全局地址

例 4-4　路由器 Mazatlan 上实施图 4-15 中的静态 NAT

```
interface Ethernet0
 ip address 10.1.1.1 255.255.255.0
 ip nat inside
!
interface Serial1
 no ip address
 encapsulation frame-relay
!
interface Serial1.705 point-to-point
 ip address 199.100.35.254 255.255.255.252
 ip nat outside
 frame-relay interface-dlci 705
!
router ospf 100
 network 10.1.1.1 0.0.0.0 area 0
 default-information originate
!
ip nat inside source static 10.1.2.2 204.15.87.2
ip nat inside source static 10.1.1.3 204.15.87.1
!
ip route 0.0.0.0 0.0.0.0 199.100.35.253
!
```

路由器 E0 接口由 **ip nat inside** 命令指定为内部接口，帧中继子接口 S1.705 用 **ip nat outside** 命令指定为外部接口。

下一步，内部接口地址用 **ip nat inside source static** 命令映射到内部全局地址。这里有两个命令，一个针对主机 C，另一个针对主机 A。例 4-5 显示 NAT 表的结果。

例 4-5　主机 A 与 C 的 IL 地址静态翻译成 IG 地址

```
Mazatlan#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
--- 204.15.87.2        10.1.2.2           ---                ---
--- 204.15.87.1        10.1.1.3           ---                ---
Mazatlan#
```

当主机 A 与主机 C 向外部发送包时，Mazatlan 看到源地址在其 NAT 表中，于是进行相应的翻译。路由器 Acapulco 有一条到网络 204.15.87.0 的路由(本案例中为静态路由)，并且对网络 10.0.0.0 的情况毫不了解。因此，Acapulco 与主机 D 可以响应主机 A 与 C 的包。如果主机 B 或路由器 Veracruz 向主机 D 发送一个包，这个包不经翻译就进行转发；当主机 D 响应这些没有经过翻译的 IL 地址时，Acapulco 不知道路由，于是将这些包丢弃，如例 4-6 所示，例中图 4-15 中的主机 D 响应主机 B 没有经过翻译的 IL 地址，Acapulco 不知道到 10.0.0.0 的路由，把这个包丢弃。

外部全局地址也可以静态翻译成外部本地地址。比如假设图 4-15 中内部网络的管理员想让主机 D“看起来”像内部网络的一部分——譬如说，其地址为 10.1.3.1。例 4-7 显示 Mazatlan 的 NAT 配置。

例 4-6　Acapulco 丢弃到 10.0.0.0 的包

```
Acapulco#debug ip icmp
ICMP packet debugging is on
Acapulco#
1d00h: ICMP: dst (10.1.1.4) host unreachable sent to 201.114.37.5
1d00h: ICMP: dst (10.1.1.4) host unreachable sent to 201.114.37.5
1d00h: ICMP: dst (10.1.1.4) host unreachable sent to 201.114.37.5
1d00h: ICMP: dst (10.1.1.4) host unreachable sent to 201.114.37.5
1d00h: ICMP: dst (10.1.1.4) host unreachable sent to 201.114.37.5
```

例 4-7　把 Mazatlan 配置成将外部全局地址静态翻译成外部本地地址

```
ip nat inside source static 10.1.1.3 204.15.87.1
ip nat inside source static 10.1.2.2 204.15.87.2
ip nat outside source static 201.114.37.5 10.1.3.1
```

路由器 NAT 配置除了多出一条 **ip nat outside source static** 命令外保持原样，这条命令使 OG 地址 201.114.37.5 映射到 OL 地址 10.1.3.1。例 4-8 中显示其生成的 NAT 表。

例 4-8　Mazatlan 中用额外的命令在 NAT 表中加入 OG 到 OL 地址的映射

```
Mazatlan#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
--- 204.15.87.2        10.1.2.2           ---                ---
--- 204.15.87.1        10.1.1.3           ---                ---
--- ---                ---                10.1.3.1           201.114.37.5
Mazatlan#
```

虽然这个案例中只解决了静态映射的问题，但在有些流量流经主机 A 与主机 D 之间及主机 C 与主机 D 之间后，也会造成一些动态映射，如例 4-9 所示。在每一个案例中，内部映射自动映射到外部映射。

例 4-9　主机 A 与 C 的内部地址自动映射到主机 D 的外部地址

```
Mazatlan#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
--- 204.15.87.2        10.1.2.2           ---                ---
--- 204.15.87.1        10.1.1.3           ---                ---
--- ---                ---                10.1.3.1           201.114.37.5
--- 204.15.87.1        10.1.1.3           10.1.3.1           201.114.37.5
--- 204.15.87.2        10.1.2.2           10.1.3.1           201.114.37.5
Mazatlan#
```

这个配置对于防止内部网络的主机向主机 D 的 OG 地址发送包来说是毫无作用的，理解这一点很重要。如图 4-16 所示，主机 A 可以成功地 ping 通主机 D 的 OL 地址(10.1.3.1)或其 OG 地址(201.114.37.5)。

事实上，例 4-10 中主机 C 的查错输出揭示了有关这个网络行为更详细的内容。主机 C 在其 OG 地址上 ping 主机 D，但主机 D 响应的却是其自己的 OL 地址。ICMP Echo Request 消息发向目的地地址 201.114.37.5 的包可以不经改变通过 NAT，但 ICMP Echo Reply 包有源地址 201.114.37.5，由 NAT 翻译成 OL 地址。

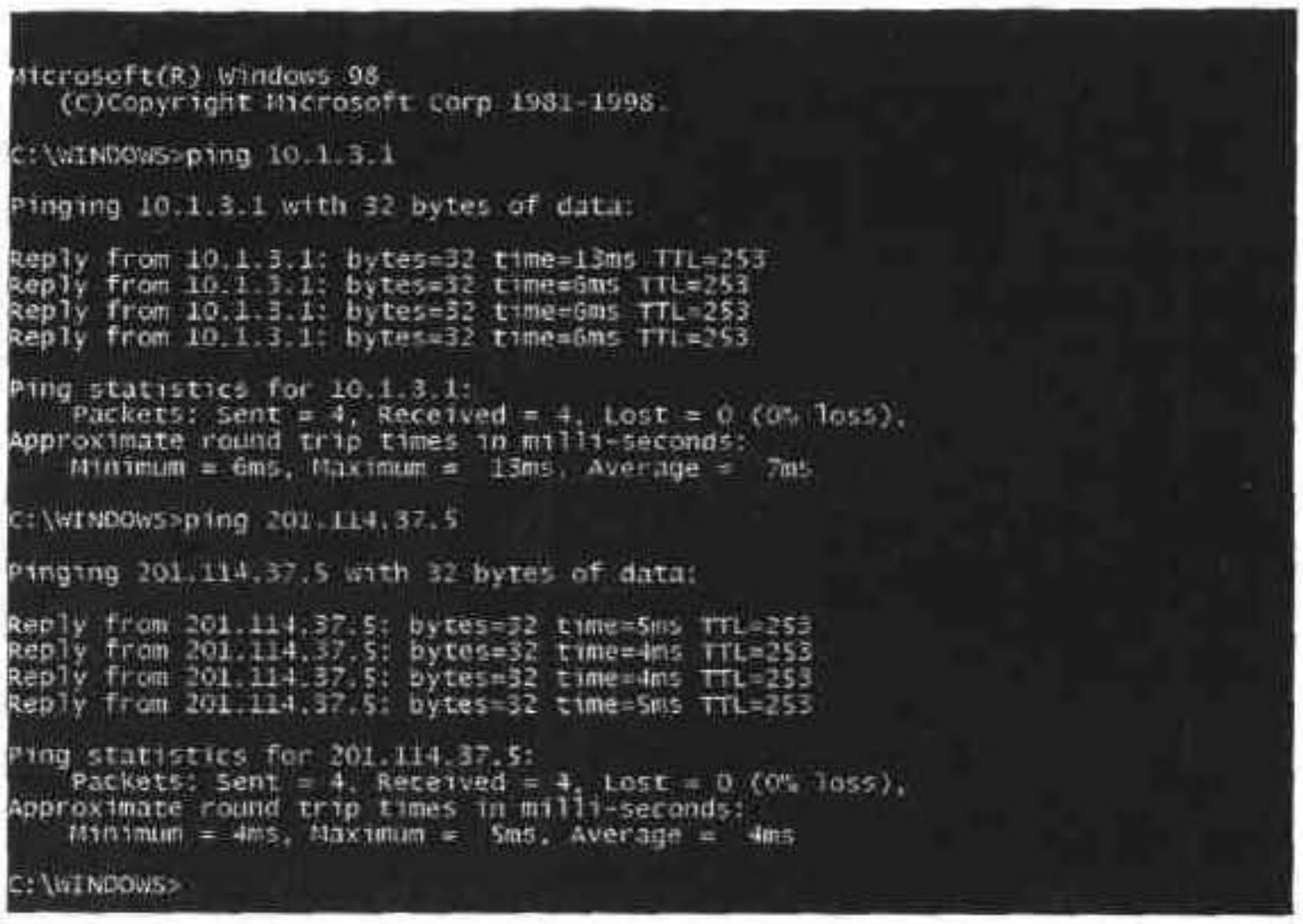


图 4-16　主机 A 可以向主机 D 的 OL 或 OG 地址发送包

例 4-10　虽然主机 C 能 ping 201.114.37.5，但 NAT 使得应答的包的源地址为 10.1.3.1

```
HostC#debug ip icmp
ICMP packet debugging is on
HostC#ping 201.114.37.5

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 201.114.37.5, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/12/20 ms
HostC#
ICMP: echo reply rcvd, src 10.1.3.1, dst 10.1.2.2
ICMP: echo reply rcvd, src 10.1.3.1, dst 10.1.2.2
ICMP: echo reply rcvd, src 10.1.3.1, dst 10.1.2.2
ICMP: echo reply rcvd, src 10.1.3.1, dst 10.1.2.2
ICMP: echo reply rcvd, src 10.1.3.1, dst 10.1.2.2
HostC#
```

提示：　如果你在试验室中重复这个例子，例 4-10 揭示了一个有用的小技巧。主机 C 实际上是一台不能进行 IP 选路(**no ip routing**)的 Cisco 路由器，用一条 **ip default-gateway** 命令指向本地连接在接口上的 Veracruz。如例 4-10 所示，这个设置使你能够利用 IOS 强大的查错工具来从主机角度观察网络的行为。

如果内部网络的管理员想防止发送到 OG 地址的流量，他必须采用过滤器，如例 4-11 所示。

例 4-11　用一个过滤器防止内部网络的流量被送到 OG 地址

```
interface Ethernet0
 ip address 10.1.1.1 255.255.255.0
 ip access-group 101 in
 ip nat inside
!
interface Serial1
 no ip address
 encapsulation frame-relay
!
interface Serial1.705 point-to-point
 ip address 199.100.35.254 255.255.255.252
 ip nat outside
 frame-relay interface-dlci 705
!
router ospf 100
 network 10.1.1.1 0.0.0.0 area 0
 default-information originate
!
ip nat inside source static 10.1.1.3 204.15.87.1
ip nat inside source static 10.1.2.2 204.15.87.2
ip nat outside source static 201.114.37.5 10.1.3.1
!
ip route 0.0.0.0 0.0.0.0 199.100.35.253
!
access-list 101 permit ip any host 10.1.3.1
!
```

注意，输入过滤器的接口为 E0。这个过滤必须在地址翻译之前进行；一个在 S1.705 上的输出过滤器无法区分已经翻译过的目的地地址。图 4-17 显示这个过滤的结果；主机 A 可以在其 OL 地址上访问主机 D，但到 OG 地址的包就被阻止。

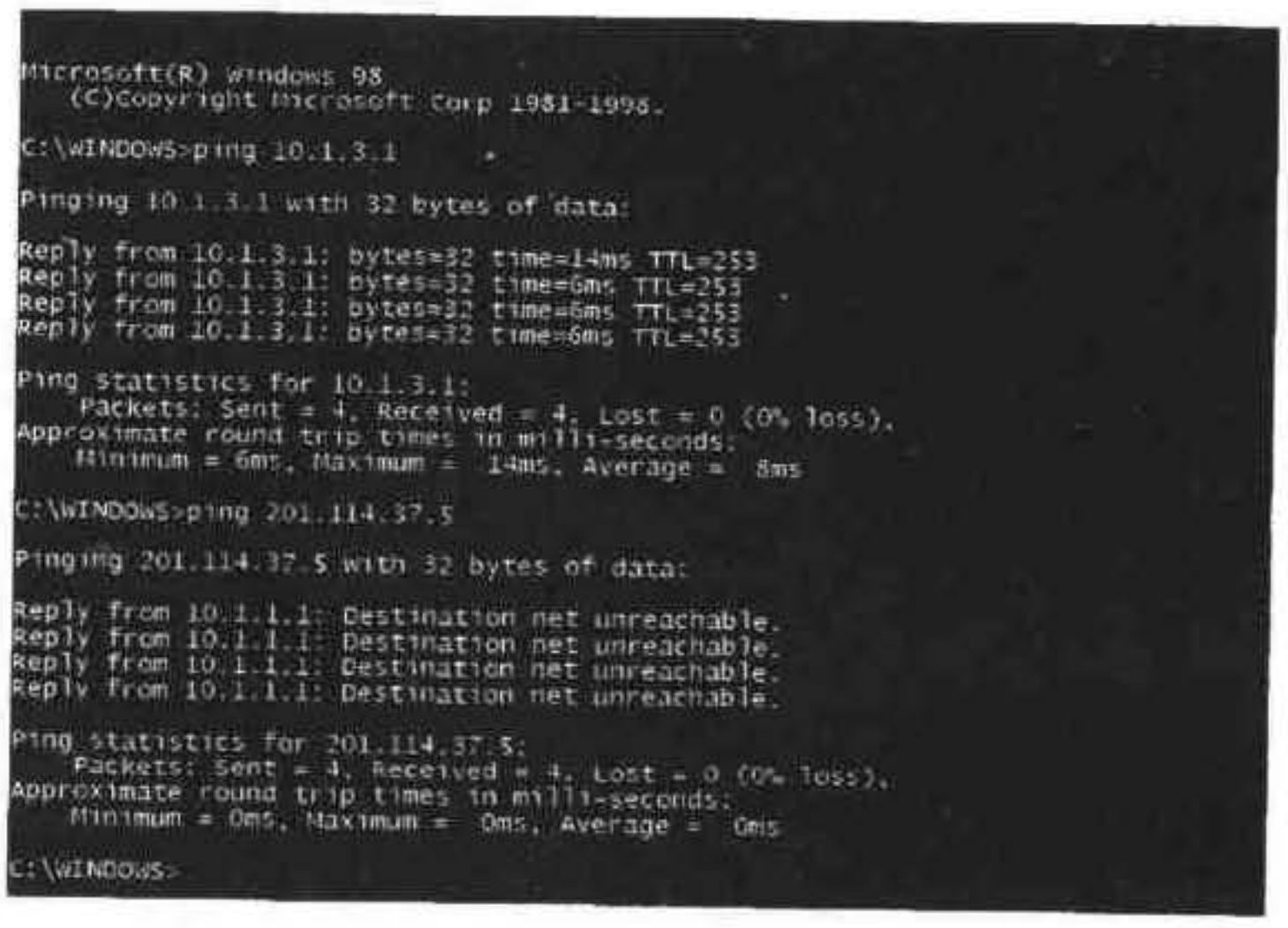


图 4-17　在 Mazatlan 上进行过滤后，内部主机只能经由其 OL 地址访问主机 D

在例 4-10 中与例 4-6 中的 **debug** 输出强调 NAT 自身不保证专用的或非法的地址不会泄

露到公网上。明智的管理员在连接 ISP 的接口上对专用的 A、B、C 类地址进行过滤。明智的 ISP 会在连接用户的接口上做同样的过滤。

直到现在，这个案例研究的各种配置的困难在于实际中很少设备要使用 IP 地址来访问别的设备。域名会经常使用到。因此 DNS 必须与其所在的 NAT 侧正确的 IP 地址相关。在图 4-18 中，DNS 服务器在内部网与外部网中都有。DNS1 可以有下面名称到地址的映射：

```
HostA.insidenet.com     IN   A    10.1.1.3
HostB.insidenet.com     IN   A    10.1.1.4
HostC.insidenet.com     IN   A    10.1.2.2
```

在此，所有主机都有本地地址(内部网内的地址)。DNS2 可能有下述名称到地址的映射：

```
HostD.outsidenet.com    IN   A    201.114.37.5
```

这些条目全都映射到全局地址。DNS1 是 inside.net 授权的，DNS2 是 outside.net 授权的。例 4-12 显示 Mazatlan 的 NAT 配置。

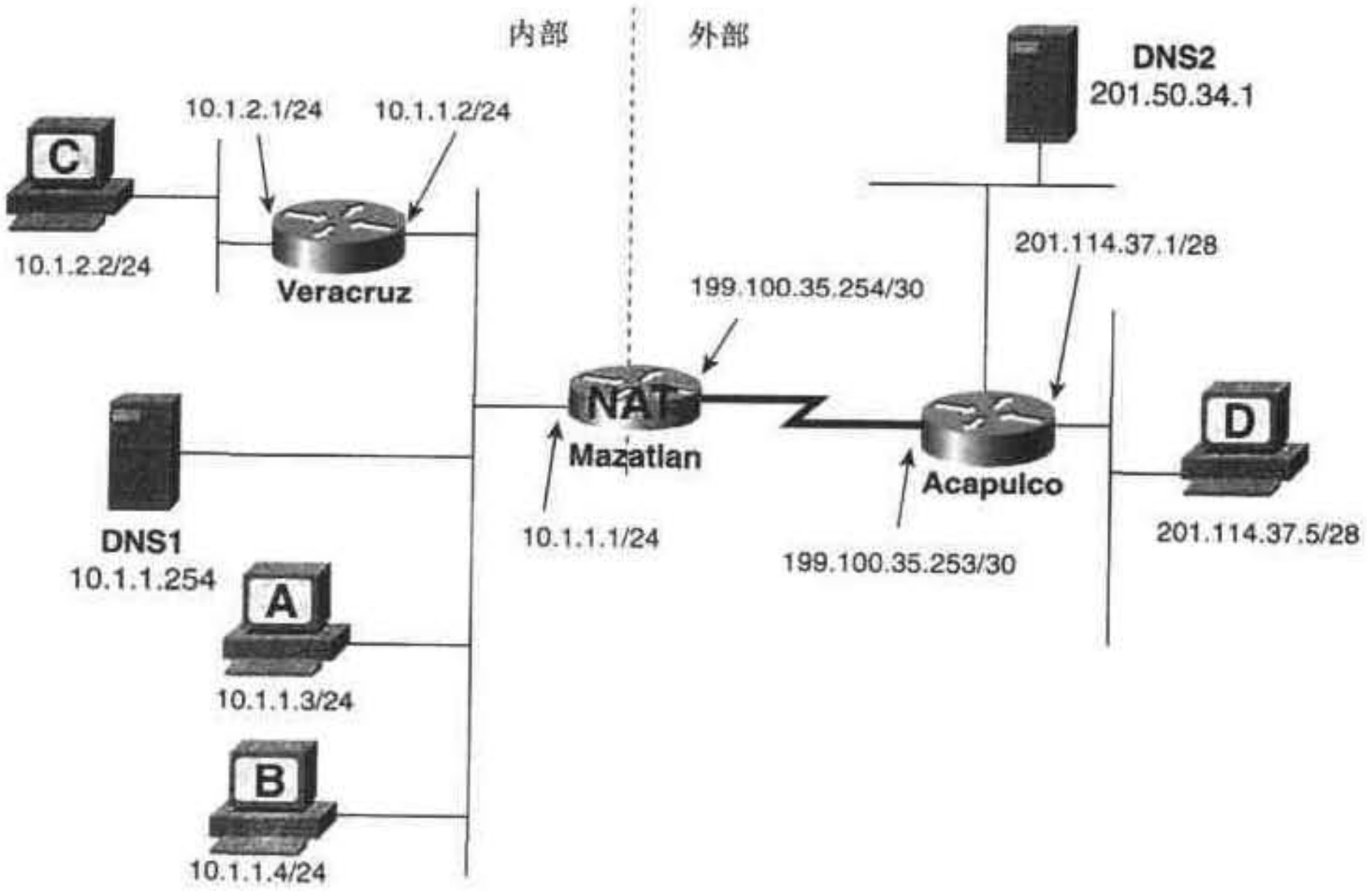


图 4-18　DNS1 是内部网络授权的，DNS2 是外部网络授权的

例 4-12　Mazatlan 的 NAT 配置，支持图 4-18 中 DNS1 与 DNS2

```
ip nat inside source static 10.1.1.3 204.15.87.1
ip nat inside source static 10.1.2.2 204.15.87.2
ip nat inside source static 10.1.1.4 204.15.87.3
ip nat inside source static 10.1.1.254 204.15.87.254
ip nat outside source static 201.114.37.5 10.1.3.1
ip nat outside source static 201.50.34.1 10.1.3.2
```

例 4-12 中除了三个内部主机和一个外部主机的配置，还有两个 DNS 服务器的条目。如

果主机 A 希望向主机 D 发送一个包，它就向 DNS1 发送一个 DNS 查询消息查询 HostD.outsidenet.com 的地址。然后 DNS1 向 DNS2 查询，DNS2 返回一个地址 201.114.37.5。当这个 DNS 消息通过 NAT 时，这个地址被翻译成 10.1.3.1，DNS1 向主机 A 转发这个消息。于是主机 A 向这个地址发送包，NAT 对这个包的源与目的地地址进行翻译。

如果主机 D 想与内部网络的一台主机通信，情况相反，主机 D 可能向 DNS2 查询 HostC.insidenet.com 的地址，DNS2 则向查询 DNS1。DNS1 响应并返回一个地址 10.1.2.2，这个地址由 NAT 翻译成 204.15.87.2，并由 DNS2 转发给主机 D。当一个包在主机 D 与主机 C 间交换时，NAT 负责翻译源与目的地地址。

4.3.2　案例研究：动态 NAT

前一个案例研究中的配置存在扩展性问题。如果图 4-18 中的内部设备不是 4 台，而是 60 或 6000 台，会有什么情况发生呢？

维护静态 NAT 映射，如同维护静态路由表一样，随着网络的发展，很快成为管理的负担。

如图 4-19 所示，内部网络使用 10.1.1.0 至 10.1.1.255 的地址作为其 IL 地址空间，并由 ISP 分配了公开地址 204.15.86.0/23。这个公开地址空间作为地址池，IG 地址从这个池中动态选择为 IL 地址进行映射。为了使网络有更加可管理性和可预测性，空间 10.1.1.0/24 映射到 204.15.86.0/24，空间 10.1.2.0/24 映射到 204.15.87.0/24。

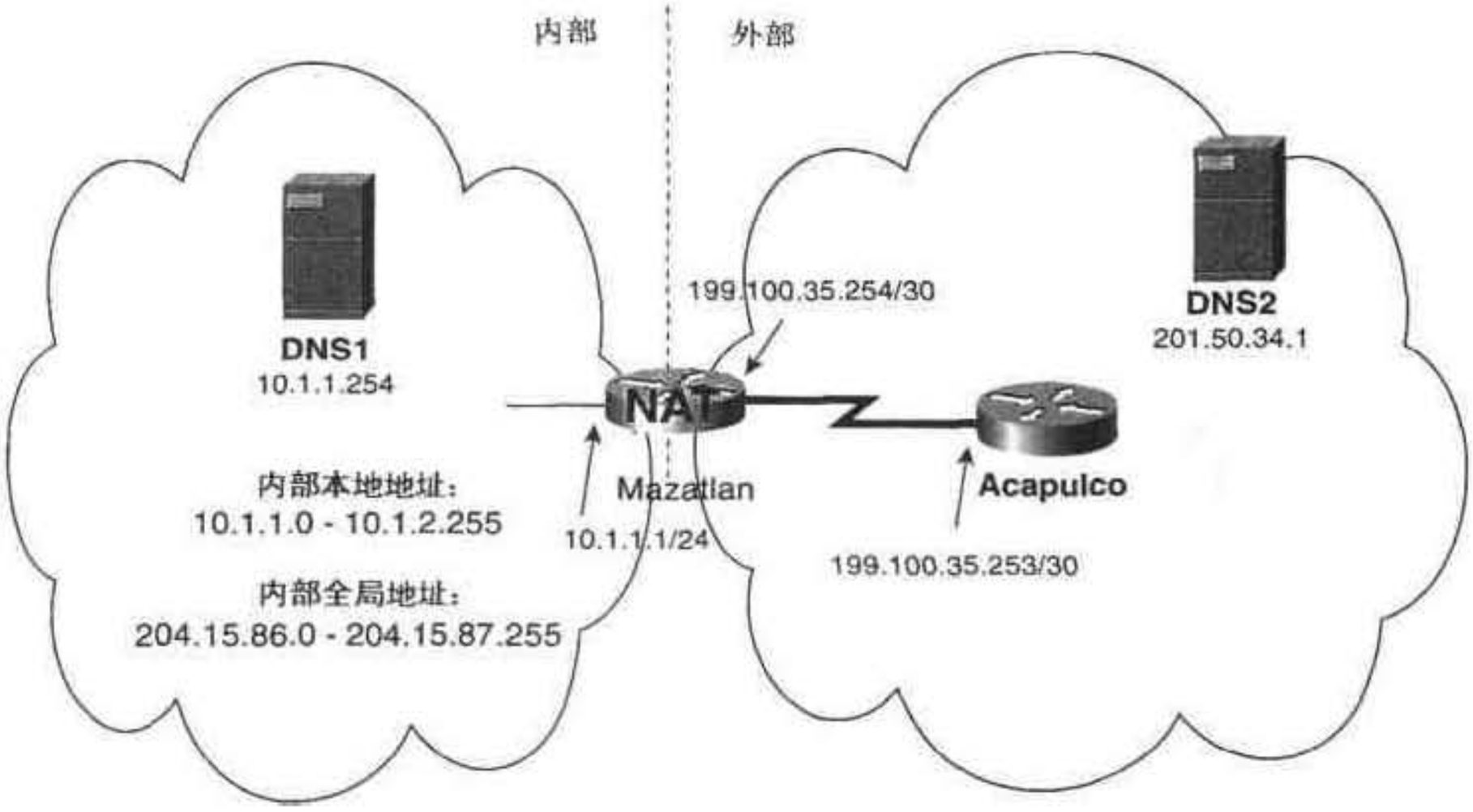


图 4-19　内部网络有大范围的 IL 与 IG 地址

命令 **ip nat pool** 建立一个地址池，并进行命名。然后，用 **ip nat inside source list** 命令指定这个地址池为 IG 地址池，并将该地址池与一个 IL 地址范围联系起来。例 4-13 显示 Mazatlan 的配置。

这个配置中建立了两个地址池，分别名为 PoolOne 与 PoolTwo。分配给 PoolOne 地址范围 204.15.86.1-204.15.86.254。分配给 PoolTwo 地址范围 204.15.87.1-204.15.87.253。注意，地址范围不包括网络地址与广播地址；命令中的 **netmask** 部分进行检查，保证如 204.15.87.255 这样的地址不会进行映射。可以用 **prefix-length** 代替关键词 **netmask**，比如：

例 4-13　Mazatlan 被配置成动态从地址池中分配地址

```
interface Ethernet0
 ip address 10.1.1.1 255.255.255.0
 ip nat inside
!
interface Serial1
 no ip address
 encapsulation frame-relay
!
interface Serial1.705 point-to-point
 ip address 199.100.35.254 255.255.255.252
 ip nat outside
 frame-relay interface-dlci 705
!
router ospf 100
 network 10.1.1.1 0.0.0.0 area 0
 default-information originate
!
ip nat pool PoolOne 204.15.86.1 204.15.86.254 netmask 255.255.255.0
ip nat pool PoolTwo 204.15.87.1 204.15.87.253 netmask 255.255.255.0
ip nat inside source list 1 pool PoolOne
ip nat inside source list 2 pool PoolTwo
ip nat inside source static 10.1.1.254 204.15.87.254
!
ip route 0.0.0.0 0.0.0.0 199.100.35.253
!
access-list 1 permit 10.1.1.0 0.0.0.255
access-list 2 permit 10.1.2.0 0.0.0.255
!
```

ip nat pool PoolTwo 204.15.87.1 204.15.87.253 prefix-length 24

与关键词 **netmask 255.255.255.0** 有相同的效果。因为有了这些命令，你可以对 204.15.86.0-204.15.86.255 这样的地址范围进行分配，地址“0”与“255”不会被映射。不过为了避免混淆，只配置实际的地址是很好的习惯。

注意，PoolTwo 不包括地址 204.15.87.254。这个地址静态分配给 DNS1，所以不在地址池中。像 DNS1 的这种情况，任何时候一个外部设备必须能与内部网络中的设备进行会话，必须对其分配一个静态地址。如果它的 IG 地址为动态的，外部设备不能知道要向哪个地址发送包。

下一步，用访问表标明要进行翻译的地址。在 Mazatlan 的配置中，访问表 1 标明 IL 地址范围为 10.1.1.0-10.1.1.255，访问表 2 标明 IL 地址范围为 10.1.2.0-10.1.2.255。

最后，IL 地址与正确的地址池相链接。比如，声明 **ip nat inside source list 1 pool PoolOne** 说明一个源于内部的 IP 地址(就是 IL 地址)，如果它匹配访问表 1 中的定义，则被翻译成 PoolOne 中的 IG 地址。

例 4-14 显示添加了动态 NAT 后的 Mazatlan 的 NAT 表。当 Mazatlan 第一次配置动态 NAT，NAT 表中除了一个静态条目，没有别的条目。你可以看到只有一条 DNS1 的静态映射。

例 4-14　Mazatlan 刚配置后的 NAT 表

```
Mazatlan#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
--- 204.15.87.254      10.1.1.254         ---                ---
Mazatlan#
```

例 4-15 显示几个内部设备产生到外部的流量后的 NAT 表。按顺序从每个地址池中分配 IG 地址，从最小的地址开始。

例 4-15　内部网络向外部网络发送几个包后，NAT 表中 IL 到 IG 的动态映射

```
Mazatlan#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
--- 204.15.86.4        10.1.1.3           ---                ---
--- 204.15.86.3        10.1.1.83          ---                ---
--- 204.15.86.2        10.1.1.239         ---                ---
--- 204.15.86.1        10.1.1.4           ---                ---
--- 204.15.87.3        10.1.2.164         ---                ---
--- 204.15.87.2        10.1.2.57          ---                ---
--- 204.15.87.1        10.1.2.2           ---                ---
--- 204.15.87.254      10.1.1.254         ---                ---
Mazatlan#
```

有时，网络管理员可能希望 IG 地址的主机部分与其映射的 IL 地址的主机部分相匹配。为了实现这一点，在定义地址池的声明最后加上关键词 **type match-host**，如例 4-16 所示。

例 4-16　配置 IG 地址的主机部分使之与其映射的 IL 地址的主机部分相匹配

```
ip nat pool PoolOne 204.15.86.1 204.15.86.254 netmask 255.255.255.0 type match-host
ip nat pool PoolTwo 204.15.87.1 204.15.87.253 netmask 255.255.255.0 type match-host
ip nat inside source list 1 pool PoolOne
ip nat inside source list 2 pool PoolTwo
ip nat inside source static 10.1.1.254 204.15.87.254
!
ip route 0.0.0.0 0.0.0.0 199.100.35.253
!
access-list 1 permit 10.1.1.0 0.0.0.255
access-list 2 permit 10.1.2.0 0.0.0.255
```

例 4-17 中显示相应的 NAT 表的结果。与例 4-15 中的表相比较，你可以看出所有相同的 IL 地址都进行了翻译。然而，与顺序地从各自地址池中选择 IG 地址不同，应选择适配主机部分的 IG 地址。

例 4-17　IG 地址的主机部分与其映射的 IL 地址的主机部分相匹配

```
Mazatlan#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
--- 204.15.86.4        10.1.1.4           ---                ---
--- 204.15.86.3        10.1.1.3           ---                ---
--- 204.15.86.83       10.1.1.83          ---                ---
--- 204.15.86.239      10.1.1.239         ---                ---
--- 204.15.87.2        10.1.2.2           ---                ---
--- 204.15.87.57       10.1.2.57          ---                ---
--- 204.15.87.164      10.1.2.164         ---                ---
--- 204.15.87.254      10.1.1.254         ---                ---
Mazatlan#
```

默认情况下，动态条目在 NAT 表中保持 86 400 秒(24 小时)。你可以用命令 **ip nat translation timeout** 来改变这个时间，范围从 0 到 2、147、483、647 秒(大约 68 年)。这个超时时间从第一次进行翻译开始，每一次通过映射进行包翻译时重置。当一个地址池中的地址在 NAT 表中映射到一个地址时，它就不能映射为其他的地址。如果过了超时时间，这个条目会从表中删除，这个地址池中的地址会返回到地址池中，可以再次使用。如果你在 **ip nat translation timeout** 命令中设置 0 秒或关键词 **never**，映射就永远不会从 NAT 中删除。

每个条目的翻译超时时间可以用 **show ip nat translations verbose** 命令来察看，如例 4-18 所示。这条命令显示了这个映射输入到 NAT 表中有多长时间、最近一次使用有多长时间、在超时时间末之前还有多少剩余时间。你可以用标志指示其他翻译类型不是动态的。比如在例 4-18 中，最后一个条目是静态翻译。

例 4-18　show ip nat translation verbose 命令显示有关每一个映射的翻译超时时间的信息

```
Mazatlan#show ip nat translations verbose
Pro Inside global      Inside local       Outside local      Outside global
--- 204.15.86.4        10.1.1.3           ---                ---
    create 00:31:55, use 00:31:55, left 23:28:04, flags: none
--- 204.15.86.3        10.1.1.83          ---                ---
    create 00:32:19, use 00:32:19, left 23:27:40, flags: none
--- 204.15.86.2        10.1.1.239         ---                ---
    create 00:33:38, use 00:33:38, left 23:26:21, flags: none
--- 204.15.86.1        10.1.1.4           ---                ---
    create 00:34:25, use 00:00:05, left 23:59:54, flags: none
--- 204.15.87.3        10.1.2.164         ---                ---
    create 00:31:02, use 00:31:02, left 23:28:57, flags: none
--- 204.15.87.2        10.1.2.57          ---                ---
    create 00:34:10, use 00:34:10, left 23:25:49, flags: none
--- 204.15.87.1        10.1.2.2           ---                ---
    create 00:35:04, use 00:35:04, left 23:24:55, flags: none
--- 204.15.87.254      10.1.1.254         ---                ---
    create 03:59:32, use 03:59:32, flags: static
Mazatlan#
```

翻译超时时间在 IL 地址池大于 IG 地址池时很重要。考虑例 4-19 中的配置。

例 4-19　1022 个 IL 地址共享 254 个 IG 地址

```
ip nat pool GlobalPool 204.15.86.1 204.15.86.254 prefix-length 24
ip nat inside source list 1 pool GlobalPool
!
access-list 1 permit 10.1.0.0 0.0.3.255
```

这里有 1022 个可能的 IL 地址——10.1.0.1 到 10.1.3.254——用 254 个 IG 地址的地址翻译。这意味着当 NAT 表中有 254 个映射条目时，不会有空闲的 IG 地址存在。没有被映射的 IL 地址发送的包会被丢弃。这个寻址方案的设计者冒险认为网络中的全部用户中只有一小部分要接入外部网络。然而如果将 NAT 表中每一个映射保持 24 小时，占用全部 IG 的概率大大增

加。通过减少翻译超时的时间，设计者可以降低这种可能性。

4.3.3　案例研究：网络合并

NAT 在防止网际网络间可能的地址冲突是很有用的。前两个案例研究说明了采用专用地址空间的网络与采用公开地址的网络的连接。这个公开寻址的网络可能是其他企业或是 Internet。使用专用的 RFC 1918 地址基本原则是这些地址必须进行翻译，因为它们不是唯一的。整个 Internet 上，许多企业在它们的网络中使用相同的地址，这些地址被 NAT“隐藏”了起来。

在内部网络要访问公开地址空间，但内部网络没有地址分配机构分配的地址的情况下，可以用前面案例研究的配置。比如，内部网络的地址空间可能是 171.68.0.0/16。要与 Internet 连接，必须要使用 NAT，因为这个地址空间分给了别的公司。将未翻译的包送到 Internet 上会造成路由的冲突。

另一种情况是两个原先独立的网络要合并时，有可能产生地址冲突。在图 4-20 中，Surf Corporation 与 Sand, Inc.进行公司的合并，组成 Surf n' Sand Enterprise。两个网络的合并也是公司合并的一部分。不幸的是，当两个网络最初建设时，设计者都选择使用 10.0.0.0 地址空间。因此，许多 Surf Corp.的网络设备与 Sand Inc.的网络设备有相同的地址。

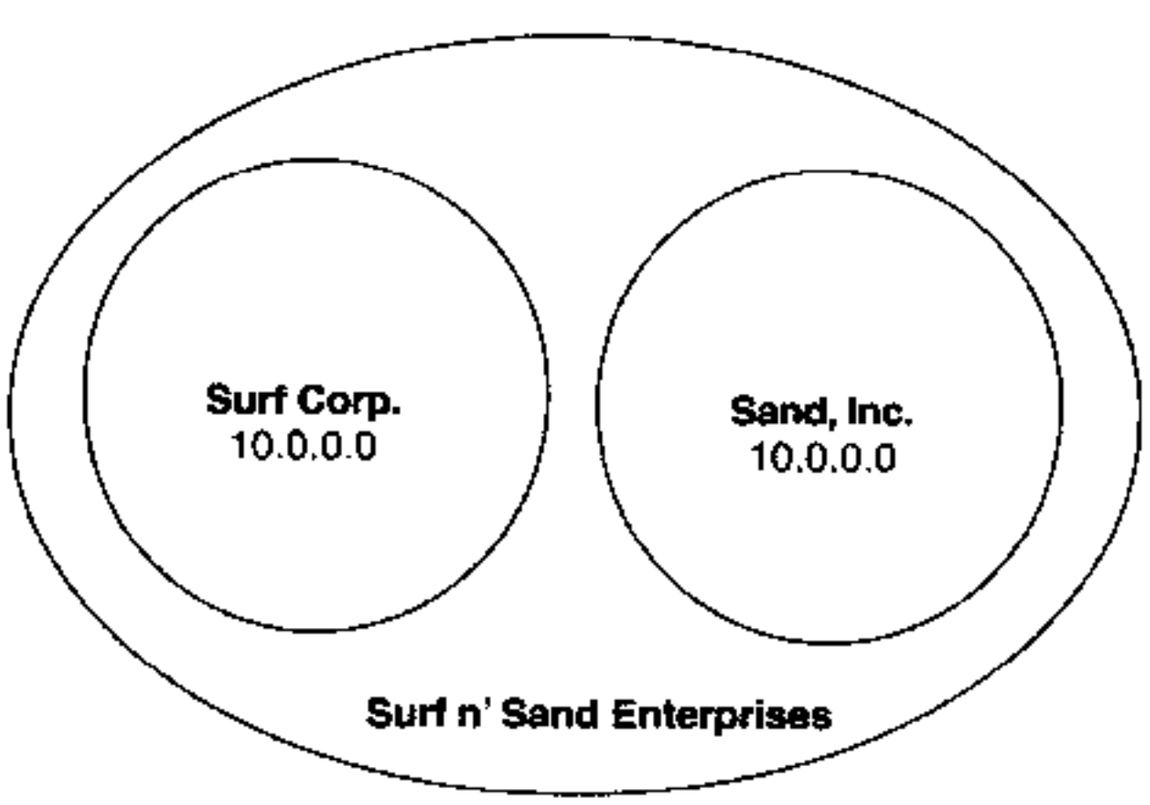


图 4-20　两个必须互联的网络有重叠的地址

最好的解决方法是对新的互联网络重新分配地址。不过地址方案经常设计得很不合理，这就使得重新分配地址成为一个重大的项目。比如，在 Surf n' Sand 网络中，所有的设备都有人工分配的 IP 地址，而不是由 DHCP 分配的 IP 地址。NAT 可以作为连接两个网络的过渡方案，直到完成重新分配计划。

> 注：　注意，在这种应用中，NAT 总是被当作过渡方案。在网络中允许地址冲突不明确地存在是很糟的选择。

Surf n' Sand 的管理员首先向其 ISP 或地址管理机构申请公开地址空间，并获得了 CIDR 地址块 206.100.160.0/19。这个块再划成两块，206.100.160.0/20 分配给原来的 Sand 网络，

206.100.176.0/20 分配给原来的 Surf 网络。这里作一个假设，虽然 10.0.0.0 网络有能力支持一百六十多万个主机地址，但实际中这两个网络没有一个会有为超过/20 地址空间范围服务的主机。

图 4-21 中的路由器 Cozumel 与 Guaymas 以例 4-20 中的配置连接两个网络。

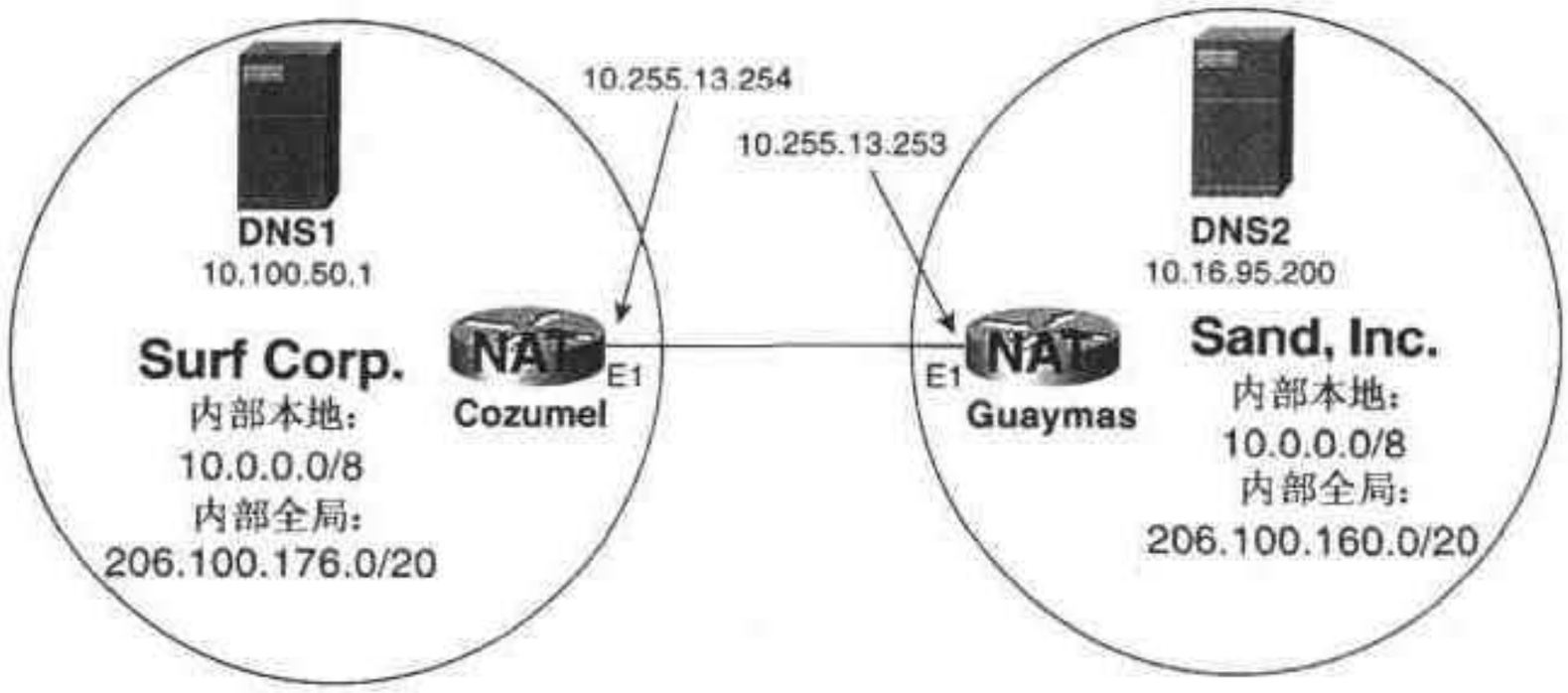


图 4-21　在两个互联网络的边界上采用 NAT 来纠正地址的冲突

DNS 服务器在这个设计中是非常重要的。在 NAT 配置中，每一个服务器有一个静态 IL 到 IG 的映射。假设在 Sand 网络中有一台设备 Beachball.sand.com，它想向 Surf 网络中的 Snorkel.surf.com 发送一个包。进一步假设这两台路由器均有 IP 地址 10.1.2.2，下面是发生的一系列事件：

(1) 主机 Beachball 向 DNS2 查询 Snorkel.surf.com 的地址；

(2) DNS2 询问负责 surf.com 域的 DNS1，这个查询的源地址为 10.16.95.200，目的地地址为 206.100.176.1。这个包转发到 Guaymas，这台路由器宣告到 EIGRP 的路由为 206.100.176.0/20；

(3) Guaymas 根据静态 NAT 条目把这个源地址由 10.16.95.200 翻译成 206.100.160.1，并把这个包转发给 Cozumel；

(4) Cozumel 根据静态 NAT 条日把日的地址由 206.100.176.1 翻译成 10.100.50.1，前转给 DNS1；

(5) DNS1 响应这个查询，指出 Snorkel.surf.com 的 IP 地址是 10.1.2.2。这个响应消息的源地址是 10.100.50.1，目的地地址是 206.100.160.1。这个响应转发给 Cozumel，它宣告到 OSPF 的路由为 206.100.160.0/20；

(6) Cozumel 将 DNS 响应的源地址翻译成 206.100.176.1。NAT 也发现在消息 Answer 项中的地址为 10.1.2.2；这个地址与访问表 1 匹配，于是这个地址被翻译成称为 Surf 的地址池中的一个地址。比如这个地址是 206.100.176.3。这个映射输入到 NAT 表中，这个响应继续转发给 Guaymas；

(7) Guaymas 将 DNS 响应的目的地地址翻译成 10.16.95.200，把这个消息转发给 DNS2；

(8) DNS2 通知 Beachball，Snorkel.surf.com 的 IP 地址是 206.100.176.3；

(9) Beachball 向 Snorkel 发送一个包，源地址为 10.1.2.2，目的地地址为 206.100.176.3，这个包再一次转发给 Guaymas；

例 4-20　图 4-21 中路由器 Cozumel 与 Guaymas 的 NAT 配置

Cozumel

```
interface Ethernet0
 ip address 10.100.85.1 255.255.255.0
 ip nat inside
!
interface Ethernet1
 ip address 10.255.13.254 255.255.255.248
 ip nat outside
!
router ospf 1
 redistribute static
 network 10.100.85.1 0.0.0.0 area 18
!
ip nat pool Surf 206.100.176.2 206.100.191.254 prefix-length 20
ip nat inside source list 1 pool Surf
ip nat inside source static 10.100.50.1 206.100.176.1
!
ip route 206.100.160.0 255.255.240.0 10.255.13.253
!
access-list 1 deny   10.255.13.254
access-list 1 permit any
```

Guaymas

```
interface Ethernet0
 ip address 10.16.95.1 255.255.255.0
 ip nat inside
!
interface Ethernet1
 ip address 10.255.13.253 255.255.255.248
 ip nat outside
!
interface Serial1
 no ip address
 encapsulation frame-relay
!
interface Serial1.508 point-to-point
 ip address 10.18.3.253 255.255.255.0
 ip nat inside
 frame-relay interface-dlci 508
!
router eigrp 100
 redistribute static metric 1000 100 255 1 1500
 passive-interface Ethernet1
 network 10.0.0.0
 no auto-summary
!
ip nat pool Sand 206.100.160.2 206.100.175.254 prefix-length 20
ip nat inside source list 1 pool Sand
ip nat inside source static 10.16.95.200 206.100.160.1
!
ip route 206.100.176.0 255.255.240.0 10.255.13.254
!
access-list 1 deny   10.255.13.253
access-list 1 permit 10.0.0.0
!
```

(10) 在 Guaymas，源地址满足访问表 1，一个地址从称为 Sand 的地址池里选出。比如这个地址是 206.100.160.2。源地址经过翻译，映射输入到 NAT 表中，这个包转发给 Cozumel；

(11) Cozumel 发现目的地地址为 206.100.176.3，在 NAT 表中映射成 10.1.2.2，并将这个地址翻译成 IL 地址。这个包转发给 Snorkel；

(12) Snorkel 发送一个包作为响应，其源地址为 10.1.2.2，目的地地址为 206.100.160.2，这个包转发给 Cozumel；

(13) Cozumel 把这个包的源地址翻译成 206.100.176.3 并转发给 Guaymas；

(14) Guaymas 把这个包的目的地地址翻译成 10.1.2.2 并转发给 Beachball。

在这个例子里，你可以看到两个有相同 IP 地址的设备，彼此不知道对方的真实地址。实现这一点的关键是 Cozumel 与 Guaymas 上的路由配置。双方都不向对方透露 10.0.0.0 网络的信息。双方的配置都不允许目的地地址为 10.0.0.0 的包发送给其他路由器上，除非包的目的地是直连的 10.255.13.248/29 子网。访问表 1 按照使之不翻译来自每个路由器 E1 接口的包的原则来进行配置。

注：　在排错练习 3 应要求进一步考虑对这个访问表的配置问题。

例 4-20 中另一重要的内容是 Guaymas 有多于一个的内部接口。多个内部接口是可接受的。

在配置中不很明显的一个重要的题目是NAT翻译超时时间与DNS缓存的生存时间(TTL)周期的协调，当 DNS 服务器收到另一个 DNS 服务器的资源记录。它把这些记录放在缓存里，以便以后查询时，可以直接应答。在这个案例研究中，DNS2 会将映射 Snorkel.surf.com 到 206.100.176.3 的 ARR 放到缓存中。DNS2 随后可以不必询问 DNS1 就对 Snorkel 的 IP 地址请求直接响应。这个缓存的 RR 有一个与其有关的 TTL 周期，当 TTL 超时，它会被清除。DNS 的 TTL 周期必须比 NAT 翻译时间要短。

例如，假设 10.1.2.2 到 206.100.176.3 的 NAT 翻译时间超时，IG 地址返还到池中，206.100.176.3 在 Surf 网络中会映射到另一个 IL 地址。不过 DNS2 仍有一个 RR 将 Snorkel.surf.com 映射到 206.100.176.3。如果 Sand 中的一个网络设备向 DNS2 查询 Snorkel 的地址，DNS2 用已经在 DNS1 作废的信息响应，那么包都发给错误的主机。

最后要注意的是这个设计中关于对 Internet 的访问。你可以在 Cozumel 与 Guaymas 间的子网中简单地加一台接入路由器(如图 4-22 所示，图中 Internet 接入路由器不支持 NAT；所有访问 Internet 的流量的翻译由 Cozumel 与 Guaymas 完成)。源自 Surf 与 Sand 网络的包的源地址已经翻译成合法的公网地址，所需的只是在 Cozumel 与 Guaymas 上把默认路由指向 Internet 接入路由器。

4.3.4　案例研究：用 NAT 实现 ISP 多宿

在较早的“NAT 和多宿主自治系统”一节中，显示用 NAT 可以克服具有不同 CIDR 块的不同 ISP 的地址多宿的困难。图 4-7 中的用户是多宿的，每一个 ISP 看到的包的源地址在它自己的地址空间内。任何一个 ISP 不会从用户收到源地址属于另一个 ISP 地址块空间的包。

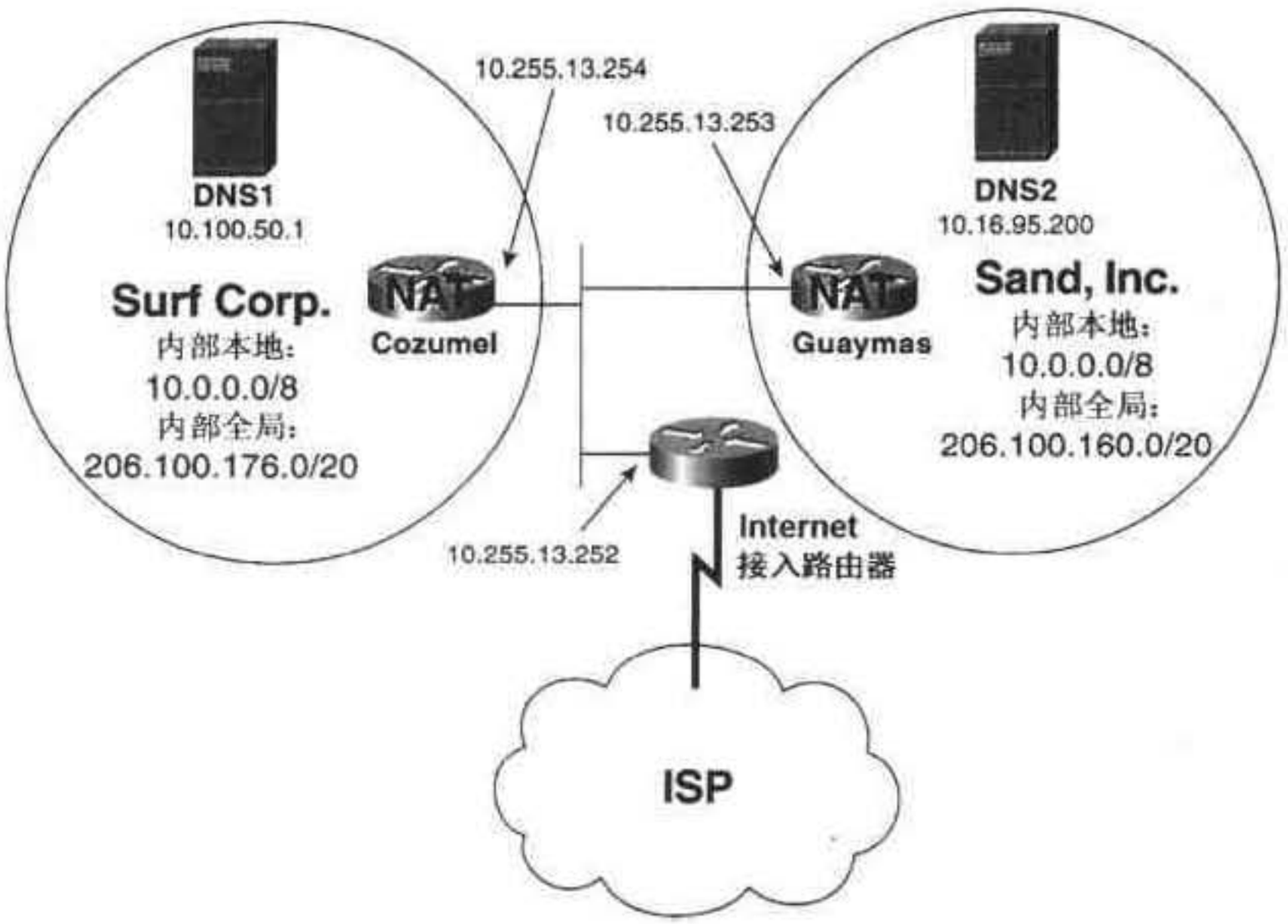


图 4-22　经 NAT 访问 Internet

基于你已经看到的案例研究，你可以很容易地写出图 4-7 中两台 NAT 路由器的配置。不过如果一台路由器多宿于两个 ISP，如图 4-23 所示，图中 ISP1 与 ISP2 均分给 JamaicaNet 一个 CIDR 地址块；当一个包前转给一个 ISP，它必须有属于这个 ISP 的正确地址，这会是什么情况呢？Montego 从两台路由器上收到完整的 BGP 路由，所以它可以选择到任何目的地的最好的服务商。当一个包转发给 ISP1，这个包的源地址必须为 ISP1 分配的 205.113.50.0/23 地址块；当一个包发送给 ISP2，它的源地址必须为 ISP2 分配的 207.36.76.0/23 地址块。

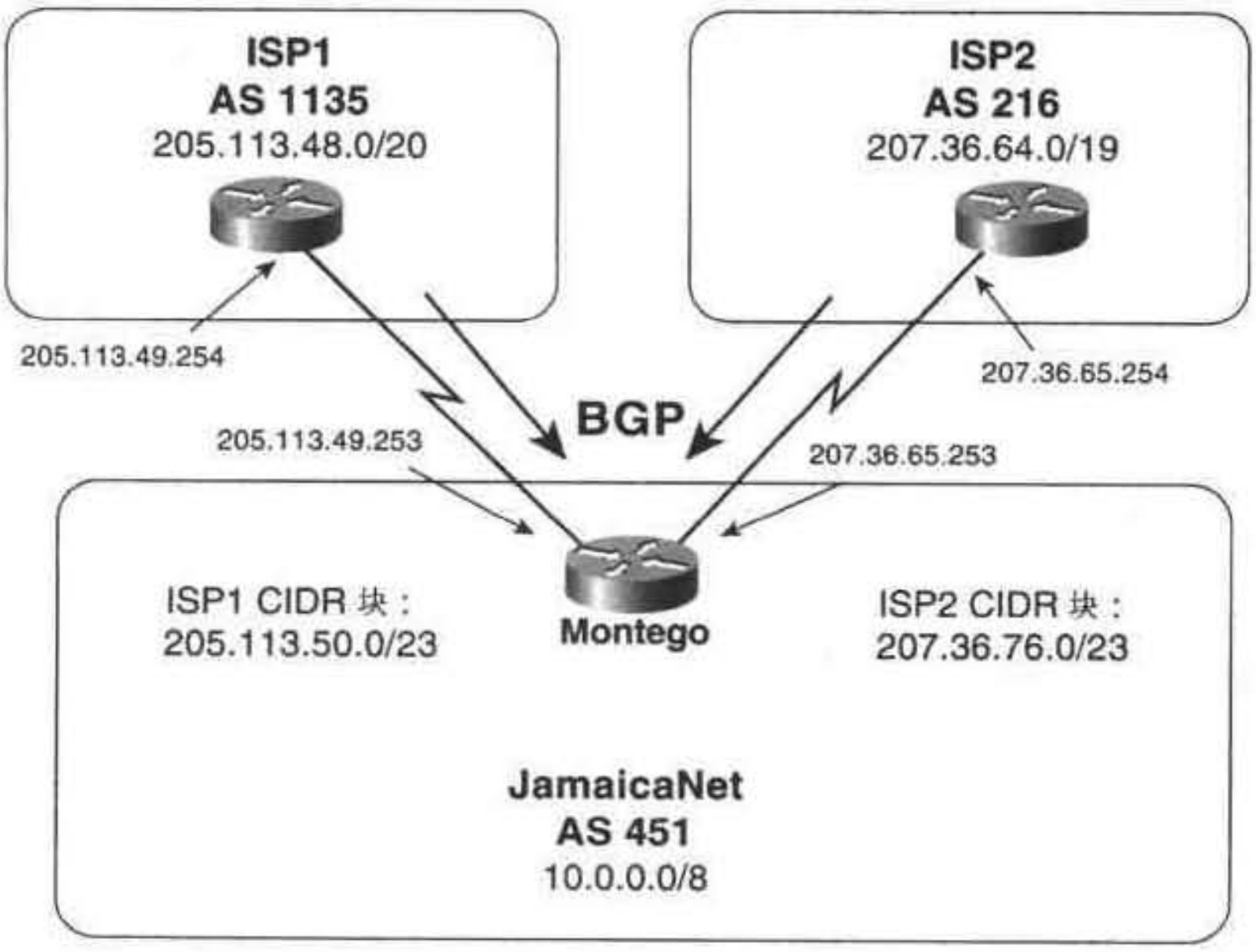


图 4-23　JamaicaNet 有 ISP1 和 ISP2 分配的 CIDR 地址块

例 4-21 显示 Montego 的配置，不同的接口上使用不同的地址池。

例 4-21　图 4-23 中 Montego 的配置

```
interface Ethernet0
 ip address 10.1.1.1 255.255.255.0
 ip nat inside
!
interface Ethernet1
 ip address 10.5.1.1 255.255.255.0
 ip nat inside
!
interface Serial1
 no ip address
 encapsulation frame-relay
!
interface Serial1.708 point-to-point
 description PVC to ISP1
 ip address 205.113.49.253 255.255.255.252
 ip nat outside
 frame-relay interface-dlci 708
!
interface Serial1.709 point-to-point
 description PVC to ISP2
 ip address 207.36.65.253 255.255.255.252
 ip nat outside
 frame-relay interface-dlci 709
!
router ospf 10
 network 10.0.0.0 0.255.255.255 area 10
 default-information originate always
!
router bgp 451
 neighbor 205.113.49.254 remote-as 1135
 neighbor 207.36.65.254 remote-as 216
!
ip nat pool ISP1 205.113.50.1 205.113.51.254 prefix-length 23
ip nat pool ISP2 207.36.76.1 207.36.77.254 prefix-length 23
ip nat inside source route-map ISP1_MAP pool ISP1
ip nat inside source route-map ISP2_MAP pool ISP2
!
access-list 1 permit 10.0.0.0 0.255.255.255
access-list 2 permit 207.36.65.254
!
route-map ISP1_MAP permit 10
 match ip address 1
 match interface Serial1.708
!
route-map ISP2_MAP permit 10
 match ip address 1
 match ip next-hop 2
!
```

由 ISP 分配的地址块在地址池 ISP1 与 ISP2 中指定。这个 NAT 配置中重要的特点是 **ip nat inside source** 声明调用的是路由映射(route map)而不是访问表。用路由映射，你不仅可以指定 IL 地址，还可以指定包转发的接口或下一跳地址。ISP1_MAP 指定源地址在 10.0.0.0 地址内的包(在访问表 1 中标明)会从接口 s1.708 转发给 ISP1。ISP2_MAP 也指定源地址为转发到下

一跳地址 206.36.65.254 的 10.0.0.0 的包发送到 ISP2。

> 注：　一般来说，不论使用 **match interface** 还是 **match ip next-hop** 命令，两个路由映射要一致。这里出于演示的目的，两个命令都用到了。

比如，一个地址为 10.1.2.2 的内部设备向目的地地址 137.19.1.1 发送一个包。这个包发送到 Montego，因为这台路由器通过 OSPF 向 JamaicaNet 宣告一条默认路由。Montego 查询了一下路由，发现到达目的地的最佳路由是由接口 S1.709 经过 ISP2，下一跳地址为 206.36.65.254。第一条 **ip nat inside source** 声明根据这个信息检查路由映射 ISP1_MAP。虽然源地址满足要求，可是输出接口不能匹配。第二条 **ip nat inside source** 根据这个信息检查路由映射 ISP2_MAP，源地址与下一跳地址均满足，所以源地址翻译成地址池 ISP2 中的一个地址。

例 4-22 显示了在一些流量传向 ISP 后，Montego 的 NAT 表。因为 IL 的地址可以从多个地址池中映射，所以这个地址映射为扩展地址映射，同时还显示了协议类型与端口号。扩展映射在案例研究“端口地址翻译”中有详细的描述。

例 4-22　Montego 的 NAT 表显示根据包被转发到哪一个 ISP 选择 IG 地址进行翻译

```
Montego#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
udp 207.36.76.2:4953   10.1.2.2:4953      137.19.1.1:69      137.19.1.1:69
udp 205.113.50.2:2716  10.1.1.2:2716      171.35.100.4:514   171.35.100.4:514
tcp 205.113.50.1:11009 10.5.1.2:11009     205.113.48.1:23    205.113.48.1:23
tcp 207.36.76.1:11002  10.1.1.2:11002     198.15.61.1:23     198.15.61.1:23
tcp 205.113.50.3:11007 10.1.2.2:11007     171.35.16.1:23     171.35.16.1:23
tcp 207.36.76.2:11008  10.1.2.2:11008     207.36.64.1:23     207.36.64.1:23
Montego#
```

例 4-22 的 NAT 表中重要的一点是 IL 地址 10.1.2.2 的三个条目。UDP 流量与一个 TCP 会话流经 ISP2 到达同一个目的地。这个 IL 地址映射成 IG 地址 207.36.76.2。另一个 TCP 会话经过 ISP1，所以映射到 IG 地址 205.113.50.3。这些条目显示 IG 地址选择的地址池根据包转发的目的地改变了，甚至对于同一个源地址也是如此。

图 4-24 显示了 3 个自治系统的 DNS 服务器。ISP1 与 ISP2 中的服务器必须访问 Ochee，这台服务器具有为 JamiacaNet 提供服务的权力。这意味着 Ochee 必须有静态 NAT 条目，以便能对两个 CIDR 块进行寻址。静态地将一个 IL 地址映射成多个 IG 地址一般是不允许的，因为这样的映射会产生混淆。在本案例中不会有混淆是因为采用同一个 NAT 来完成两组地址的映射。当 Montego 将到 Ochee 的 DNS 查询与其对 DNS1 与 DNS2 的响应进行寻路时，就对地址进行了相应的翻译。

为了允许静态 NAT 将一个 IL 地址映射到多个 IG 地址，在映射声明后加上关键词 **extendable**。例 4-23 显示 Montego 的 NAT 配置。

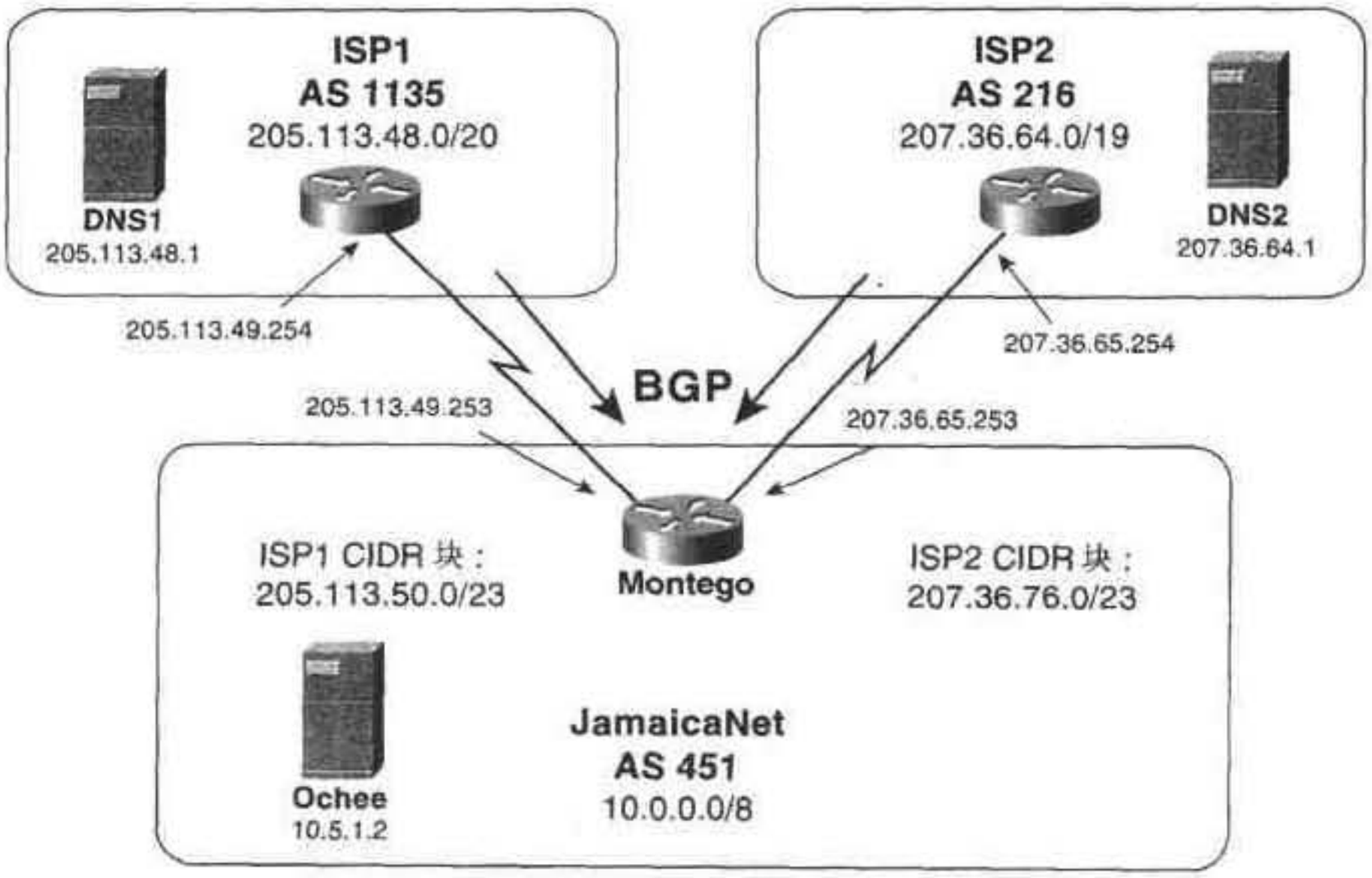


图 4-24　DNS 服务器 Ochee 必须有静态 IL 到 IG 的映射，这样它可以由 DNS1 与 DNS2 查询

例 4-23　Montego 中允许静态 NAT 将一个 IL 地址映射到多个 IG 地址的 NAT 配置

```
ip nat pool ISP1 205.113.50.2 205.113.51.254 prefix-length 23
ip nat pool ISP2 prefix-length 23
 address 207.36.76.1 207.36.76.99
 address 207.36.76.101 207.36.77.254
ip nat inside source route-map ISP1_MAP pool ISP1
ip nat inside source route-map ISP2_MAP pool ISP2
ip nat inside source static 10.5.1.2 207.36.76.100 extendable
ip nat inside source static 10.5.1.2 205.113.50.1 extendable
!
access-list 1 permit 10.0.0.0 0.255.255.255
access-list 2 permit 207.36.65.254
!
route-map ISP1_MAP permit 10
 match ip address 1
 match interface Serial1.708
!
route-map ISP2_MAP permit 10
 match ip address 1
 match ip next-hop 2
```

从 DNS1 的角度来看，Ochee 的地址是 205.113.50.1。注意，NAT 池 ISP1 经过修改不包括这个地址。从 DNS2 的角度来看，Ochee 的地址为 206.36.76.100。这个地址取自 207.36.70.0/23 地址块中间，而不是取自头或尾，这使得地址池 ISP2 不连续。地址池经修改配置而指定了两个范围的地址：在 Ochee 地址之前的地址和 Ochee 地址之后的地址。

你通过首先对地址池进行命名，再定义前缀长度或掩码来对不连续的地址范围进行配置。这个配置在提示符中显示出来，你可以输入一组地址范围。例 4-24 显示了 ISP2 地址池的配置步骤，包括提示符。

例 4-24　NAT 池中配置不连续的地址范围

```
Montego(config)#ip nat pool ISP2 prefix-length 23
Montego(config-ipnat-pool)#address 207.36.76.1 207.36.76.99
Montego(config-ipnat-pool)#address 207.36.76.101 207.36.77.254
```

4.3.5　端口地址翻译

与前一个案例研究中的多宿 NAT 路由器相反的另一个极端情况是 SOHO(小办公室、家庭办公室)路由器。这个环境中，路由器将多个设备连接到 Internet 上。不同于为每个设备映射一个公网地址，端口地址翻译(PAT)允许所有 SOHO 设备共享一个 IG 地址。

PAT 允许“过载”，或者说允许多个 IL 地址映射到同一个 IG 地址。为了完成这一任务，路由表里的 NAT 条目为扩展条目——不仅要看相关的 IP 地址，还要看协议和端口号。对包的 IP 地址与端口地址进行翻译，理论上最多可以把 65535 个 IL 地址映射到一个 IG 地址上(因为端口号为 16bit)。

> 注：　每一个 NAT 条目要占大约 160 字节的内存，所以 65535 条条目会占用多于 10MB 的内存和相当多的 CPU 资源。实际中，不会有这个数量的地址映射到 PAT 配置中。

Cisco 的 NAT 试图保留 BSD 的语法，在可能的情况下把 IL 的端口号映射到同样的 IG 端口号。不同的 IG 端口号只在与 IL 地址相关的端口号在另一个映射中被占用时使用。

图 4-25 显示连接到 ISP 的三个设备。

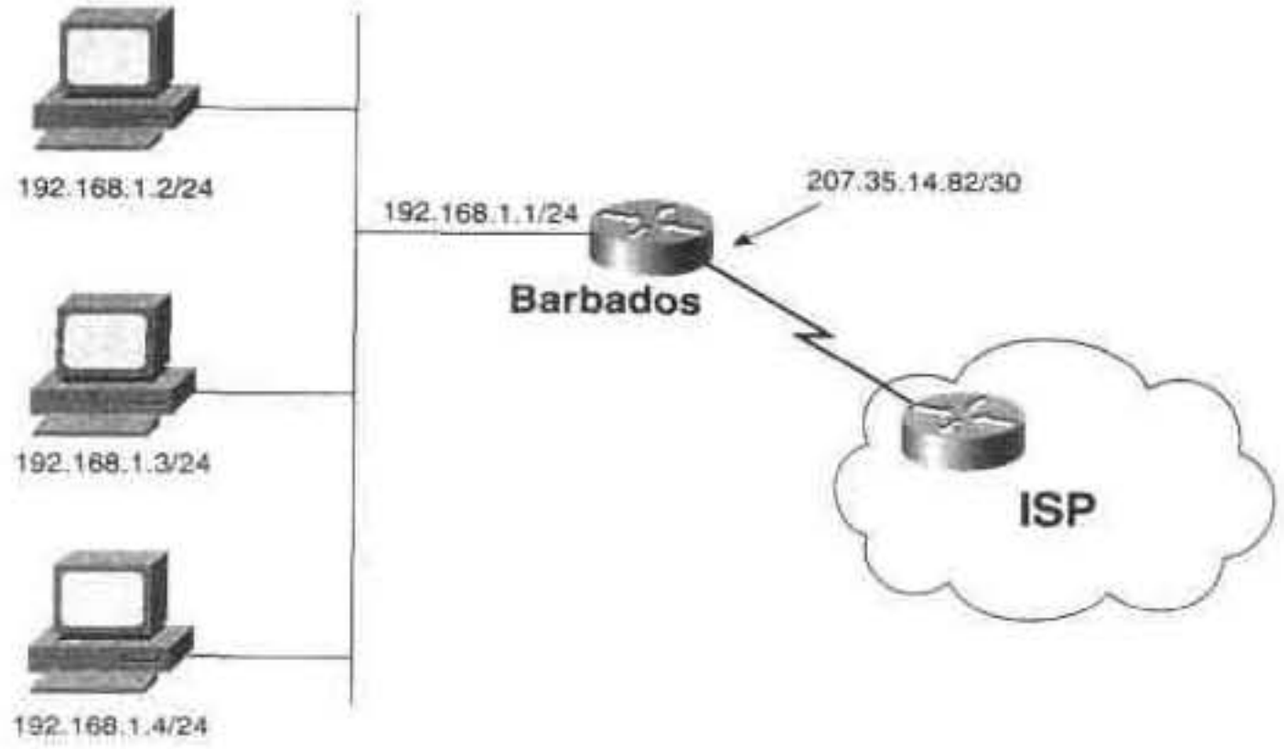


图 4-25　Barbados 用 PAT 将三个内部主机映射到一个串行接口的地址上

接入路由器的串行口上有一个由 ISP 分配的公网 IP 地址，例 4-25 显示了这一配置。

例 4-25　使图 4-25 中的路由器 Barbados 的 PAT 启动

```
interface Ethernet0
 ip address 192.168.1.1 255.255.255.0
 ip nat inside
!
interface Serial0
 ip address 207.35.14.82 255.255.255.252
 ip nat outside
!
ip nat inside source list 1 interface Serial0 overload
!
ip route 0.0.0.0 0.0.0.0 Serial0
```

```
!
access-list 1 permit 192.168.1.0 0.0.0.255
!
```

关键词 **overload** 启动了 PAT 的支持。虽然 **ip nat inside source** 命令涉及到一个地址池，但在这个案例中，它只涉及到配置 IG 地址的那个接口。和往常一样，用访问表来标明 IL 地址。

例 4-26 显示在传送了几个包后，接入路由器的 NAT 表。多数 IG 端口与 IL 端口是匹配的，但是注意到 IL 套接口的端口号已经被占用的两个实例(192.168.1.2:11000 与 192.168.1.2:11001)。因此，NAT 选择了未用的端口分配给这些套接口，它们与 IL 端口不匹配。

例 4-26　不同的 IL 地址映射到同一个 IG 地址的不同端口上

```
Barbados#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
tcp 207.35.14.82:11011 192.168.1.3:11011  191.115.37.2:23    191.115.37.2:23
tcp 207.35.14.82:5000  192.168.1.2:11000  191.115.37.2:23    191.115.37.2:23
udp 207.35.14.82:3749  192.168.1.2:3749   135.88.131.55:514  135.88.131.55:514
tcp 207.35.14.82:11000 192.168.1.4:11000  191.115.37.2:23    191.115.37.2:23
tcp 207.35.14.82:11002 192.168.1.2:11002  118.50.47.210:23   118.50.47.210:23
udp 207.35.14.82:9371  192.168.1.2:9371   135.88.131.55:514  135.88.131.55:514
icmp 207.35.14.82:7428 192.168.1.3:7428   135.88.131.55:7428 135.88.131.55:7428
tcp 207.35.14.82:5001  192.168.1.2:11001  135.88.131.55:23   135.88.131.55:23
tcp 207.35.14.82:11001 192.168.1.4:11001  135.88.131.55:23   135.88.131.55:23
Barbados#
```

4.3.6　案例研究：TCP 负载均衡

图 4-26 显示的拓扑与 PAT 案例中的很相似。可是这三个内部设备不是主机，而是有镜像内容的相同的服务器。这是为了建立一个地址为 199.198.5.1 的“虚拟服务器”；也就是说从外部看来好像 IG 地址有一台单独的服务器。实际上路由器 Barbados 配置成对这 3 个 IL 地址进行轮流翻译。

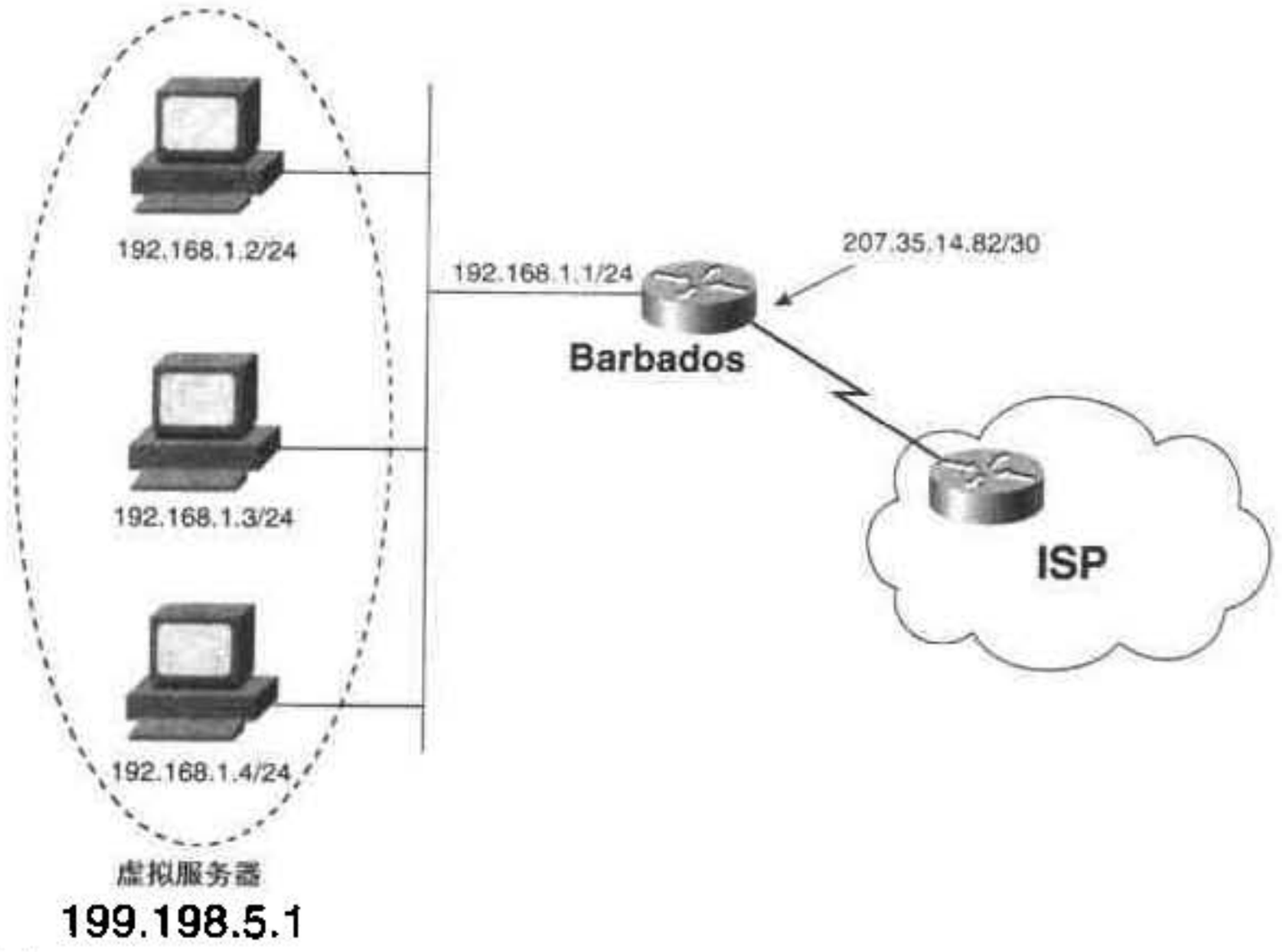


图 4-26　三个内部设备是有镜像内容的相同的服务器，从外部看来像一台单独的服务器

例 4-27 显示了 Barbados 的配置，Barbados 的 NAT 配置将 TCP 负载平均分配到 3 台相同的服务器上，外部设备只看到一个单独的内部全局地址。

例 4-27　Barbados 的配置

```
interface Ethernet0
 ip address 192.168.1.1 255.255.255.0
 ip nat inside
!
interface Serial0
 ip address 207.35.14.82 255.255.255.252
 ip nat outside
!
ip nat pool V-Server 192.168.1.2 192.168.1.4 prefix-length 24 type rotary
ip nat inside destination list 1 pool V-Server
!
ip route 0.0.0.0 0.0.0.0 Serial0
!
access-list 1 permit 199.198.5.1
!
```

与大多数前面案例中显示的对 IL 地址进行翻译不同，这个配置对 IG 地址进行翻译。地址池 V-Server 包含有可用的 IL 地址一览表，关键词 **type rotary** 使得对池内地址进行轮流分配。访问表与往常一样，标明要进行翻译的地址——在本案例中目的地地址为 199.198.5.1。

例 4-28 显示 4 个外部设备向虚拟服务器发送 TCP 流量后产生的 NAT 表。你可以观察到前 3 个连接(从底向上)是按从地址池中最低 IL 地址到最高 IL 地址顺序分配的。地址池中只有 3 个地址，所以第 4 个连接又重新映射到最低的 IL 地址。

例 4-28　到虚拟服务器地址 199.198.5.1 的 TCP 连接是在三台真实的服务器地址上分摊

```
Barbados#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
tcp 199.198.5.1:23     192.168.1.2:23     203.1.2.3:11003    203.1.2.3:11003
tcp 199.198.5.1:23     192.168.1.4:23     135.88.131.55:11002 135.88.131.55:11002
tcp 199.198.5.1:23     192.168.1.3:23     118.50.47.210:11001 118.50.47.210:11001
tcp 199.198.5.1:23     192.168.1.2:23     191.115.37.2:11000 191.115.37.2:11000
Barbados#
```

4.3.7　案例研究：服务分配

你也可以根据连接是由 TCP 连接还是由 UDP 服务进行分配，用 NAT 来建立虚拟服务器，。图 4-27 中的拓扑与图 4-26 中的很相似，除了服务器不一样以外。不同的服务器提供不同的服务，而从外部看来这三台服务器为一个地址为 199.198.5.1 的单独的服务器。

例 4-29 显示 Barbados 的 NAT 配置。

这里没有地址池或访问表；相反的，配置中只有一系列简单的 IL 到 IG 的映射表。这个声明与前面看到的静态声明的差别在于指明了 TCP 或 UDP 与源和目的地端口。这里使用了 **extendable** 关键词，因为同一个地址——这里是 IG 地址——在不只一个声明中出现。你不一定要输入这个关键词：Cisco IOS 软件已经自动加上了。按顺序，声明映射 SMTP(TCP 端口 25)、syslog(UDP 端口 514)、TFTP(UDP 端口 69)、FTP(TCP 端口 20 与 21)与 HTTP(TCP 端口 80)。

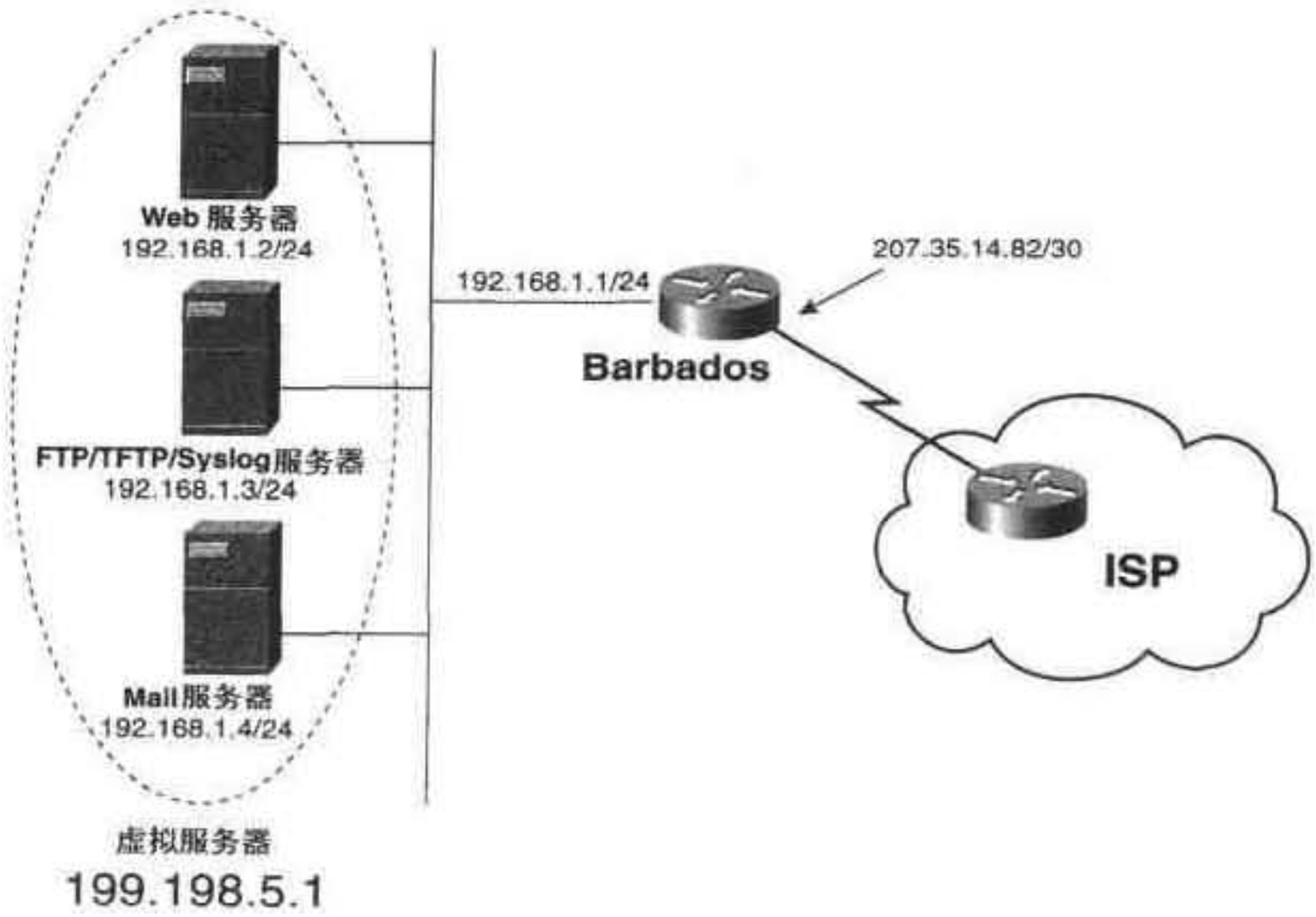


图 4-27　三台提供不同的服务的内部设备从外部看来是一台单独的服务器

例 4-29　Barbados 的 NAT 配置根据与地址相关的 TCP 或 UDP 的端口翻译虚拟 IG 地址

```
interface Ethernet0
 ip address 192.168.1.1 255.255.255.0
 ip nat inside
!
interface Serial0
 ip address 207.35.14.82 255.255.255.252
 ip nat outside
!
ip nat inside source static tcp 192.168.1.4 25 199.198.5.1 25 extendable
ip nat inside source static udp 192.168.1.3 514 199.198.5.1 514 extendable
ip nat inside source static udp 192.168.1.3 69 199.198.5.1 69 extendable
ip nat inside source static tcp 192.168.1.3 21 199.198.5.1 21 extendable
ip nat inside source static tcp 192.168.1.3 20 199.198.5.1 20 extendable
ip nat inside source static tcp 192.168.1.2 80 199.198.5.1 80 extendable
!
ip route 0.0.0.0 0.0.0.0 Serial0
!
```

例 4-30 显示 Barbados 配置后的 NAT 表，在任何动态翻译发生前，Barbados 的 NAT 表只包括 IL 套接口到 IG 套接口的静态映射；所有的条目都为静态条目。

例 4-30　没有地址翻译前，Barbados 的 NAT 表

```
Barbados#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
udp 199.198.5.1:514    192.168.1.3:514    ---                ---
udp 199.198.5.1:69     192.168.1.3:69     ---                ---
tcp 199.198.5.1:80     192.168.1.2:80     ---                ---
tcp 199.198.5.1:21     192.168.1.3:21     ---                ---
tcp 199.198.5.1:20     192.168.1.3:20     ---                ---
tcp 199.198.5.1:25     192.168.1.4:25     ---                ---
Barbados#
```

例 4-31 显示有一些流量通过 Barbados 后的 NAT 表，UDP 与 TCP 根据它们相关的端口号，被映射到不同的 IL 地址上。注意所有的动态映射中只有两 个 OG 地址出现。可是会话根据与 IG 地址相关的端口号，被映射到不同的 IL 地址。

例 4-31　有地址翻译后，Barbados 的 NAT 表

```
Barbados#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
udp 199.198.5.1:514    192.168.1.3:514    ---                ---
tcp 199.198.5.1:25     192.168.1.4:25     207.35.14.81:11003 207.35.14.81:11003
udp 199.198.5.1:69     192.168.1.3:69     ---                ---
tcp 199.198.5.1:80     192.168.1.2:80     ---                ---
tcp 199.198.5.1:21     192.168.1.3:21     ---                ---
tcp 199.198.5.1:20     192.168.1.3:20     ---                ---
tcp 199.198.5.1:25     192.168.1.4:25     ---                ---
tcp 199.198.5.1:20     192.168.1.3:20     191.115.37.2:1027  191.115.37.2:1027
tcp 199.198.5.1:21     192.168.1.3:21     191.115.37.2:1026  191.115.37.2:1026
tcp 199.198.5.1:80     192.168.1.2:80     191.115.37.2:1030  191.115.37.2:1030
udp 199.198.5.1:69     192.168.1.3:69     191.115.37.2:1028  191.115.37.2:1028
udp 199.198.5.1:514    192.168.1.3:514    207.35.14.81:1029  207.35.14.81:1029
Barbados#
```

4.4　NAT 故障排除

Cisco NAT 可以使你完成很多工作，配置也很直观。如果它不能正常工作，你可以通过问下列问题，来寻找 些常见的原因：

- 动态地址池中是否有正确范围的地址？
- 动态地址池间是否有重复的地址？
- 静态映射的地址与动态地址池中的地址之间是否有重复？
- 访问表中是否指明了要翻译的正确地址？是否漏掉了 些地址？是否包括了 些不该包括的地址？
- 是否指明了正确的内部与外部接口。

一个新的 NAT 配置最常见的问题不是 NAT 本身，而是选路。记住，你要在包中改变源和目的地地址；在这个翻译之后，路由器是否知道如何处理这个新的地址？

另 个问题是超时，如果被翻译的地址在某些系统中被缓存下来，可是 NAT 表中的动态条目超时，包就会被送到错误的地址或目标看起来好像消失了。除了已经讨论过的 **ip nat traslation timeout** 命令，你还可以改变几个默认的时间限制。表 4-3 中列出了所有 **ip nat translation** 命令可以用到的关键词和它们默认的超时时间。你可以在 0～2、147、483、647 秒间改变所有的这些默认值。

表 4-3　　**动态 NAT 表超时的值**

IP NAT 翻译	默认时长(秒)	描　　述
timeout	86400 (24 小时)	所有与端口无关的动态翻译的超时
dns-timeout	60	DNS 连接的超时

续表

IP NAT 翻译	默认时长(秒)	描　　述
finrst-timeout	60	在收到有 FIN 与 RST 标志的 TCP 包的超时 (结束 TCP 会话)
icmp-timeout	60	ICMP 翻译的超时
port-timeout tcp	60	TCP 端口翻译的超时
port-timeout udp	60	UDP 端口翻译的超时
syn-timeout	60	收到有 SYN 标志的 TCP 包，而没有其他会话包的超时
tcp-timeout	86 400 (24 小时)	TCP 翻译的超时(与端口无关)
udp-port	300 (5 分钟)	UDP 翻译的超时(与端口无关)

理论上讲，NAT 表中可映射的数目是没有限制的。实际中，内存与 CPU 或者可用地址范围或端口空间就对条目数目进行限制。每一个 NAT 映射用大约 160 字节的内存。很少情况下，为了性能或策略的原因，必须对条目数目进行限制。你可以用 **ip nat translation max-entries** 来定义。

另一个有用的排错命令是 **show ip nat statistics**，如例 4-32 所示。这个命令显示 NAT 配置的总结及激活的翻译类型的数目、现有映射选中的次数、未选中的次数(会导致新的映射的建立)及超时的翻译。对于动态地址池，池的类型、可用的地址数目、分配的地址数目、分配失败的次数和使用地址池(refcount)进行翻译的次数。

例 4-32　show ip nat statistics 显示了许多用于 NAT 配置的分析与排错的有用的详细信息

```
StCroix#show ip nat statistics
Total active translations: 3 (2 static, 1 dynamic; 3 extended)
Outside interfaces:
  Serial0, Serial1.708, Serial1.709
Inside interfaces:
  Ethernet0, Ethernet1
Hits: 980  Misses: 43
Expired translations: 54
Dynamic mappings:
-- Inside Source
access-list 1 interface Serial0 refcount 0
StCroix#
```

最后，你可以从 NAT 表中人工清除动态 NAT 条目。这个行为在你需要去除一些讨厌的条目、不想等待超时，或者你要清除整个 NAT 表来重新配置地址池时，是非常有用的。注意，Cisco IOS 软件不允许你在地址池中的地址映射在 NAT 表中时，改变或删除这个地址池。**clear ip nat translations** 命令可以清除条目；你可以用全局和局部地址或 TCP 和 UDP 翻译(包括端口号)指定一个条目，或者你可以用(*)来清除整个表。当然，只有动态条目会被清除；这个命令不会清除静态条目。

4.5　尾注

[1]Egevang, K.B.与 P. Francis. “RFC 1631: The IP Network Anndress Translator(NAT)” (工作正在进行中)。

4.6　展望

你已经看到 NAT 帮你更有效地使用网络地址。在下一章“IP 多播路由介绍”中，讨论在设备组必须共享同一信息时，如何用多播路由协议使网络资源更有效利用。

4.7　命令归纳

表 4-4 提供本章中讨论的命令的列表和与之相应的描述。

表 4-4　　命 令 归 纳

命　令	描　述
clear ip nat translation {* | [inside |tcp {inside [全局 IP [全局端口] 局部 IP [局部端口]]}]| **udp {inside** [全局 IP [全局端口] 局部 IP [局部端口]]}| **[inside** 全局 IP 局部 IP] | [outside 局部 IP 全局 IP]}	清除 NAT 表中的动态条目
ip nat {inside|outside}	指定内部与外部接口，由从一个接口产生的或到一个接口上的流量会经过 NAT 检查
ip nat inside destination list {访问表号|名称} **pool** 名称	启动内部目的地址的翻译
ip nat inside source {**list**{访问表号|名称} **pool** 名称 **[overload] |static** 全局 IP 局部 IP}	启动内部源地址的翻译
ip nat outside source {list {访问表号|名称} **pool** 名称| static 全局 IP 局部 IP}	启动外部源地址的翻译
ip nat pool name 开始 IP 结束 IP **{netmask** 掩码| **prefix-length** 前缀长度) **type {rotary | match-host}**	定义用于地址翻译的地址池
ip nat translation max-entries 条目数	设置 NAT 表中条目数量的限制
ip nat translation {timeout | udp-timeout | dns-timeout | tcp-timeout | finrst-timeout | icmp-timeout | syn-timeout | port-timeout {tcp | udp}} 秒	改变动态条目从 NAT 表中删除和地址返回到地址池中的默认时间
show ip nat statistics	显示 NAT 统计
show ip nat translation [verbose]	显示 NAT 表

4.8　配置练习

配置练习1～5参考图4-28。

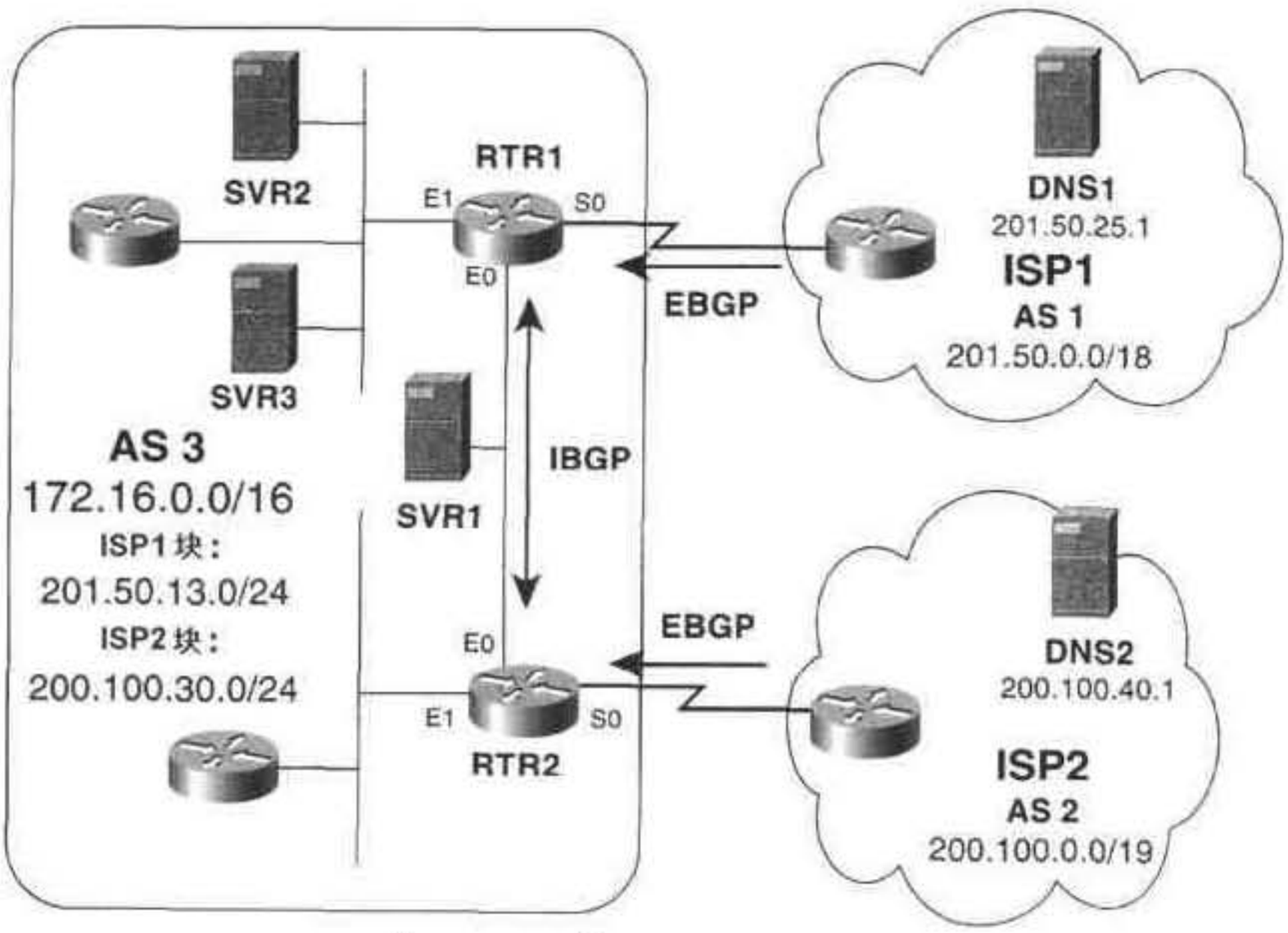


图4-28　配置练习1～5中的网络拓扑

1．图4-28中的ISP1把地址块201.50.13.0/24分配给AS 3。ISP2把地址块200.100.30.0/24也分配给AS 3。RTR1与RTR2从ISP的路由器上收到完整的BGP路由，但不能向ISP传送任何路由。在两台路由器之间运行IBGP，在所有的以太网接口上运行OSPF。在BGP与OSPF之间没有路由再分配。路由器接口的地址如下：

RTR1，E0：172.16.3.1/24

RTR1，E1：172.16.2.1/24

RTR1，S0：201.50.26.13/30

RTR2，E0：172.16.3.2/24

RTR2，E1：172.16.1.1/24

RTR2，S0：200.100.29.241/30

SVR1是AS 3中授权的DNS服务器；它的地址为172.16.3.3。DNS1访问SVR1用的地址为201.50.13.1，而DNS2访问同一台服务器用的地址是200.100.30.254。写出RTR1与RTR2中路由与NAT的配置，把内部地址正确地翻译成每一个ISP分配的地址块中的地址。任何一个内部设备必须能访问每一个ISP，不过，在这种情况下，不能让具有专用源地址的包离开AS 3。

2．图4-28中SVR2的地址为172.16.2.2，SVR3的地址为172.16.2.3。修改配置练习1中的配置，使ISP1的AS中的设备访问这些服务器时，访问地址201.50.13.3，而实际是对这

些服务器轮流进行访问。

3．从 ISP2 中发往 200.100.30.50 的 HTTP 包，实际发送到了图 4-28 中的 SVR2 处。从 ISP2 中发往 200.100.30.500 的 SMTP 包，实际发送到了 SVR3。修改前一个练习中的配置以实现这种翻译。

4．图 4-28 中，5 个外部设备 201.50.12.67-201.50.12.71，必须在 AS 3 看来分别具有地址 192.168.1.1-192.168.1.5。在前一个配置中加入相应的 NAT 配置。

5．图 4-28 的 AS 3 中的设备具有地址子网 172.16.100.0/24，当向 ISP2 发送包时，应具有 IG 地址 200.100.30.75。修改前一练习的配置以实现这一点。

6．图 4-29 中，RTR1 与 RTR2 上加了冗余链路，使每一台路由器都与两个 ISP 相连，每一台路由器都从两个 ISP 接收完整的 BGP 路由。RTR1 的 S1 地址是 200.100.29.137/30，RTR2 的 S1 地址为 201.50.26.93/30。写出这两台路由器的配置，保证以上练习中所加的特性仍能正确工作。

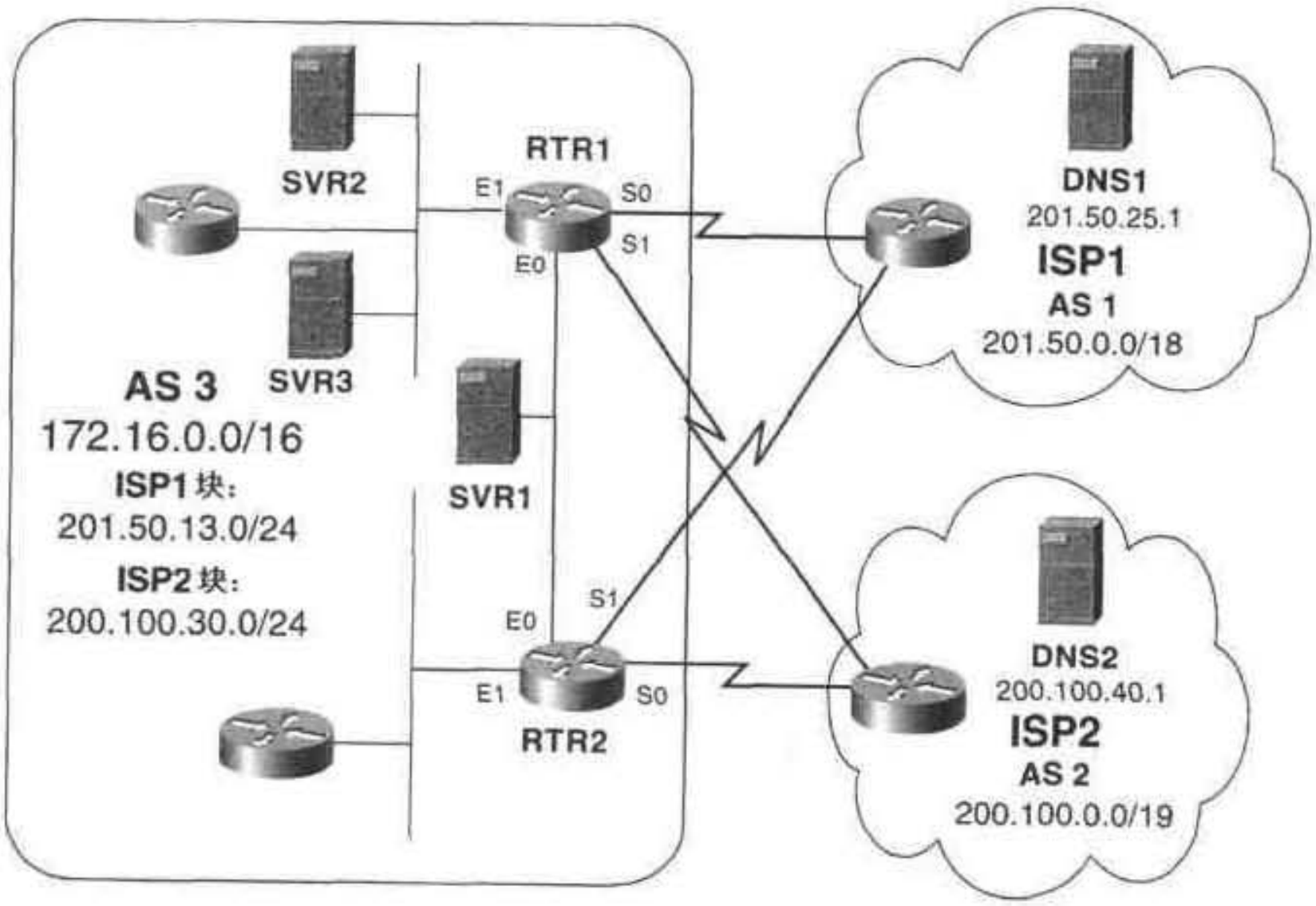


图 4-29　配置练习 6 的网络拓扑

4.9　故障排除练习

1. 指出例 4-33 配置中的错误。

例 4-33　排错练习 1 的配置

```
ip nat pool EX1 192.168.1.1 192.168.1.254 netmask 255.255.255.0 type match-host
ip nat pool EX1A netmask 255.255.255.240
 address 172.21.1.33 172.21.1.38
 address 172.21.1.40 172.21.1.46
ip nat inside source list 1 pool EX1
ip nat inside source static 10.18.53.210 192.168.1.1
ip nat outside source list 2 pool EX1A
!
access-list 1 permit 10.0.0.0 0.255.255.255
access-list 2 permit 192.168.2.0 0.0.0.255
```

2. 图 4-30 中的 RTR1 连接到有重叠地址的两个网络。路由器上的 NAT 配置如例 4-34 所示，但设备不能经路由器相互通信，有什么错误？

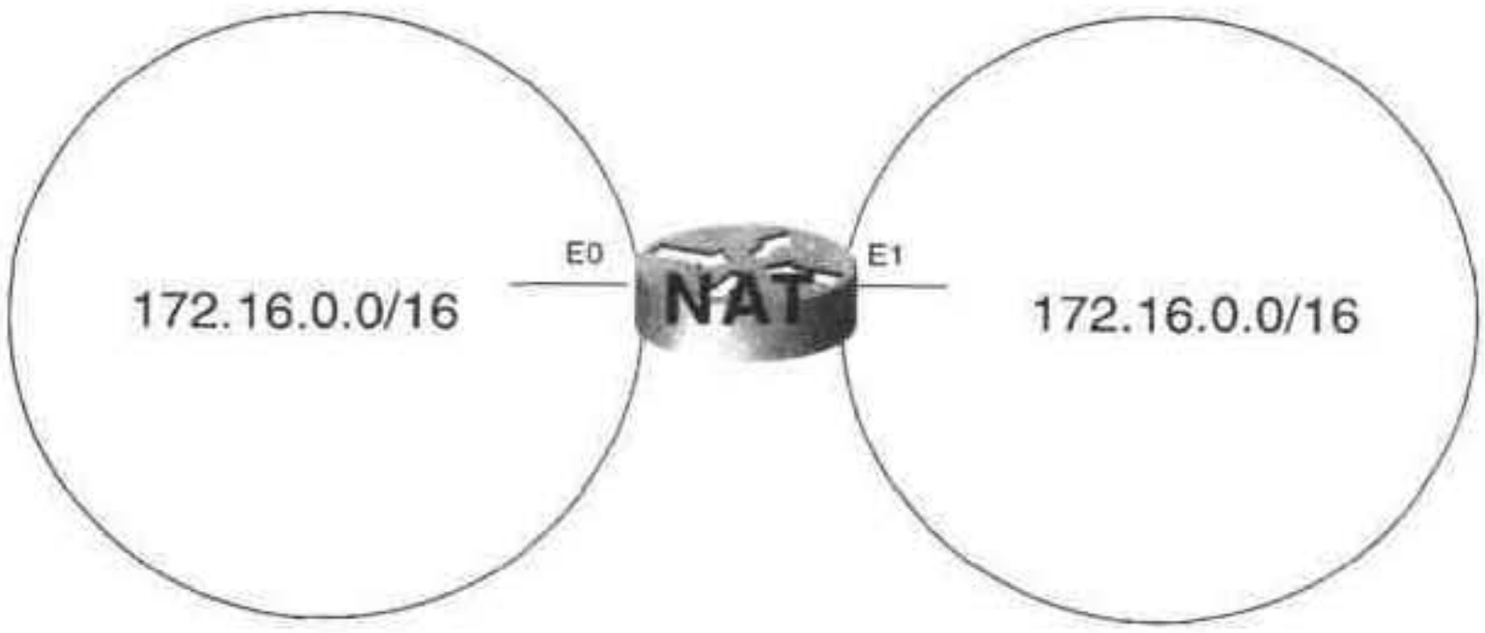


图 4-30　排错练习 2 的网络拓扑

例 4-34　排错练习 2 的配置

```
interface Ethernet0
 ip address 172.16.10.1 255.255.255.0
 ip nat inside
!
interface Ethernet1
 ip address 172.16.255.254 255.255.255.0
 ip nat outside
!
router ospf 1
 redistribute static metric 10 metric-type 1 subnets
 network 10.0.0.0 0.255.255.255 area 0
!
ip nat translation timeout 500
ip nat pool NET1 10.1.1.1 10.1.255.254 netmask 255.255.0.0
ip nat pool NET2 192.168.1.1 192.168.255.254 netmask 255.255.0.0
ip nat inside source list 1 pool NET1
ip nat outside source list 1 pool NET2
!
ip classless
!
ip route 10.1.0.0 255.255.0.0 Ethernet0
ip route 192.168.0.0 255.255.0.0 Ethernet1
!
access-list 1 permit 172.16.0.0 0.0.255.255
```

3. 参考图 4-21 中的 Cozumel 与 Guaymas 的配置。如果访问表的第一行去掉，会有什么结果？Guaymas 与 Cozumel 能不能 ping 通对方？

第 5 章　IP 多播路由介绍

- **IP 多播的需要**——本节解释了 IP 多播的基本概念，并讨论了关于进行有效多播所必须的功能，如寻址与信令。
- **多播寻路问题**——本节描述了有关所有 IP 多播路由协议的常见的问题。
- **DVMRP 的操作**——本节描述了 DVMRP 的操作。
- **多播 OSPF(MOSPF)的操作**——本节描述了 MOSPF 的操作。
- **CBT 的操作**——本节描述了 CBT 的操作。
- **对于与协议无关的多播(PIM)的介绍**——本节讨论了 PIM-DM 与 PIM-SM 共同的 PIM 基本功能。
- **PIM-DM 的操作**——本节描述了 PIM-DM 的操作。
- **PIM-SM 的操作**——本节描述了 PIM-SM 的操作。

多播是向一组接收者发送数据的方法。有人会说单播和广播是多播的子集：在单播情况下，组中只有一个接收者；在广播情况下，所有可能的接收者均是组中的成员。本章将阐明为什么这样的观点只是在概念层面上正确；至少在联网的环境中，多播、单播和广播之间存在着明显的区别。

电台广播和电视节目统称为“广播”，但事实上是多播。发送器用一定的频率发送数据，而需要接收数据的用户组就调频到这个频率上，这个频率就像是多播的地址。所有在发送范围内的接收者均可以收到这个信号，不过只有那些调到正确频率上的接收者才能最终收到该信号。

信号的覆盖范围引出了另一个重要的概念：电台和电视的传送有一定的范围——由于这个范围会受到发射器功率的限制，因此在发射范围之外的接收器将接收不到该信号。在本章中你会了解到 IP 的多播也是有范围的。

在第 1 卷的 RIP-2、EIGRP 和 OSPF 为了有效地进行路由信息的通信，采用了多播。应用采用多播也是同样的原因——提高网络效率和保存网络资源。图 5-1 描述了一组 IP 主机，其中一台是数据源(S)，数据必须被送到一组接收者(G)处。接收者多于一个，但组中并不包括所有可能的接收者。

一种方法是让源使用重复的单播。这就是说，源为组中的每个目的主机分别创建一个含有相同数据的包，接着每个包再被单播到特定的主机上，如图 5-2 所示。

如果只有几个目的主机，那么这种方法可以正常工作。事实上，当前正在使用的许多“多播”应用就是采用了重复的单播。但是，当接收者的数目增加到成百上千时，主机上创建和发送这么多相同数据的负担也同时增加了。更重要的是，主机的接口、直连的介质、直连的路由器和低速 WAN 链路都成了潜在的瓶颈。如果数据对时延敏感，并且不能在一个单独的数据包中传送，那么也会带来一些别的问题。如果所有二号数据包的备份都必须等所有一号数据包的备份排队和发送完毕之后才能发送，那么排队的时延会给数据流带来不可忍受的时间间隔。

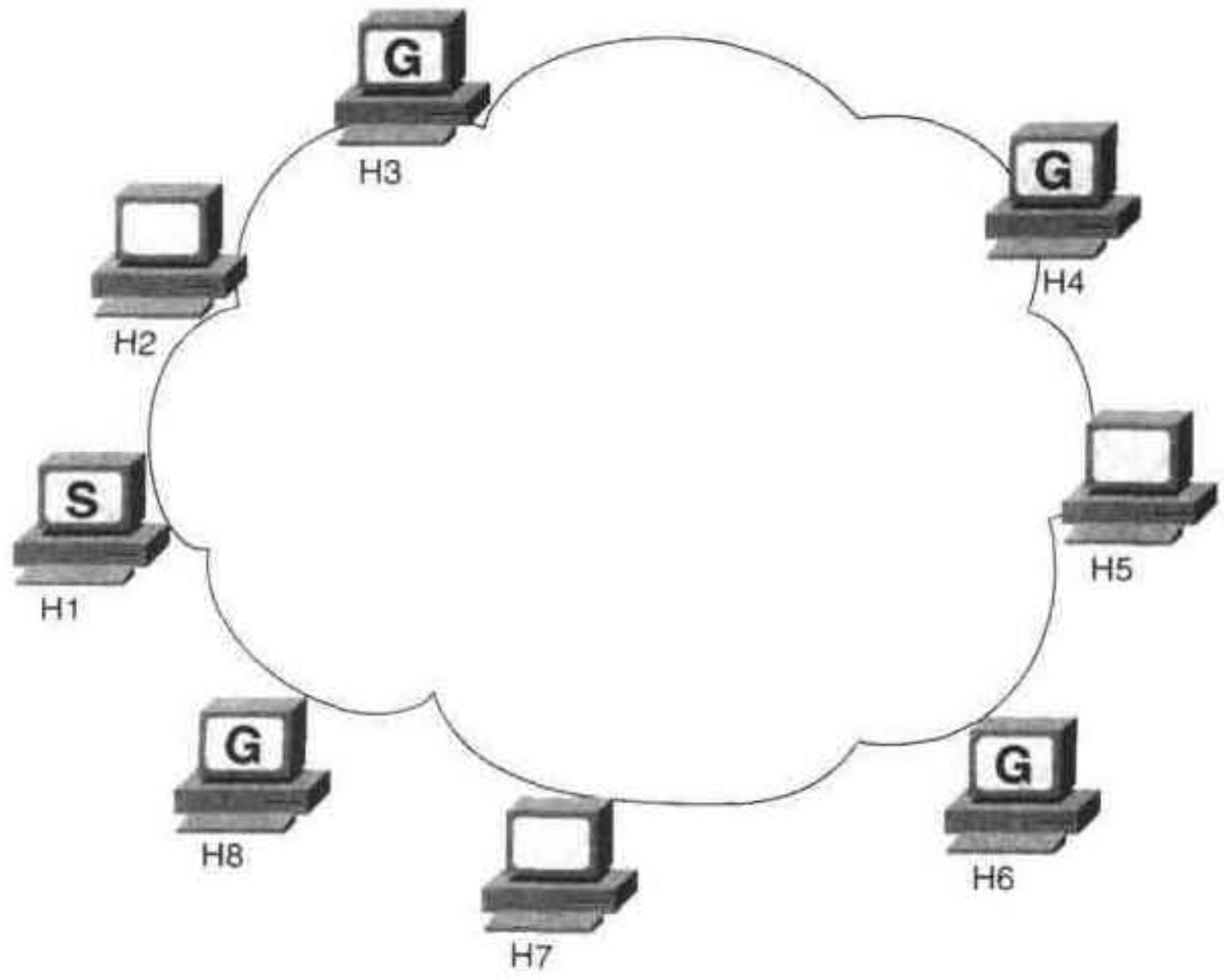


图 5-1　源必须向多个接收者发送相同的数据

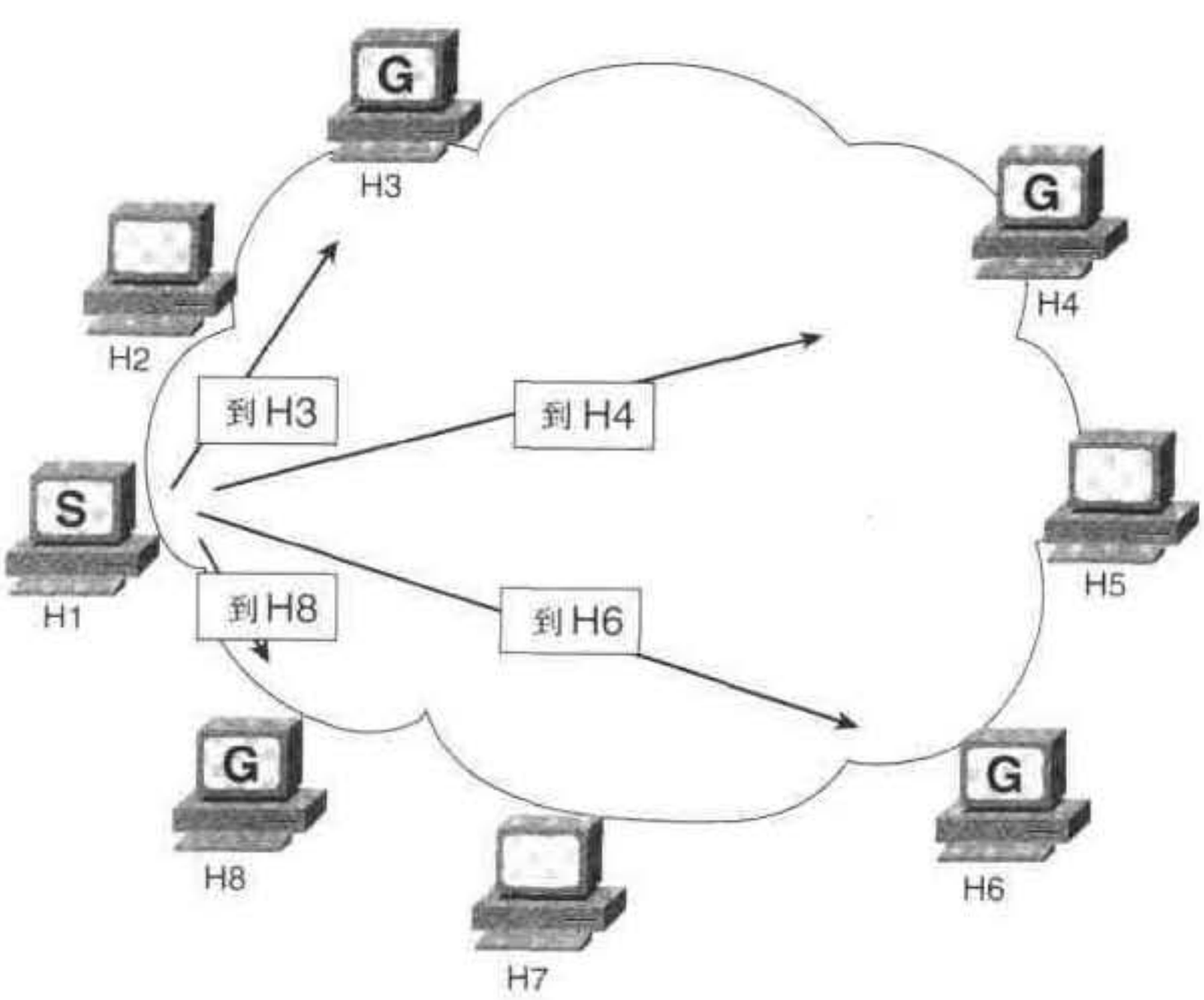


图 5-2　用单播的方法向多个接收者发送相同的数据会给源带来很大的负担

另一种实现多播的方法是用图 5-3 所示的方法广播数据，它把负担从源和其本地设备上转移开，源只需发送一个单一的数据包，不过如此一来，负担却转移到了网络中的其他主机上。每台主机都必须接收一份广播的数据包，并对它进行处理。只有在高层协议或应用程序中，无关的主机才能认出这是个需要丢弃的包。如果接收组中主机的数目相对于网络中的全

部主机来讲数目很少，那么这种处理负担也会不可忍受。

> 注：　当组员的数目相对于多播域中主机的总数要少一些时，这样的域是稀疏分布的。你将在本章后述的内容中再见到这个概念。

广播的另一个难点是 IP 路由器不能将数据包前转到广播的目的主机。如果图 5-3 中所示的云所示为一个网际的网络，而并非一个单独的广播介质，那么广播的包就不能到达远端主机。可以将广播直接转接，不过这可能会是一个最糟的方法。不仅是所有的主机收到数据包，而且源还要为复制数据包承受很重的负担。

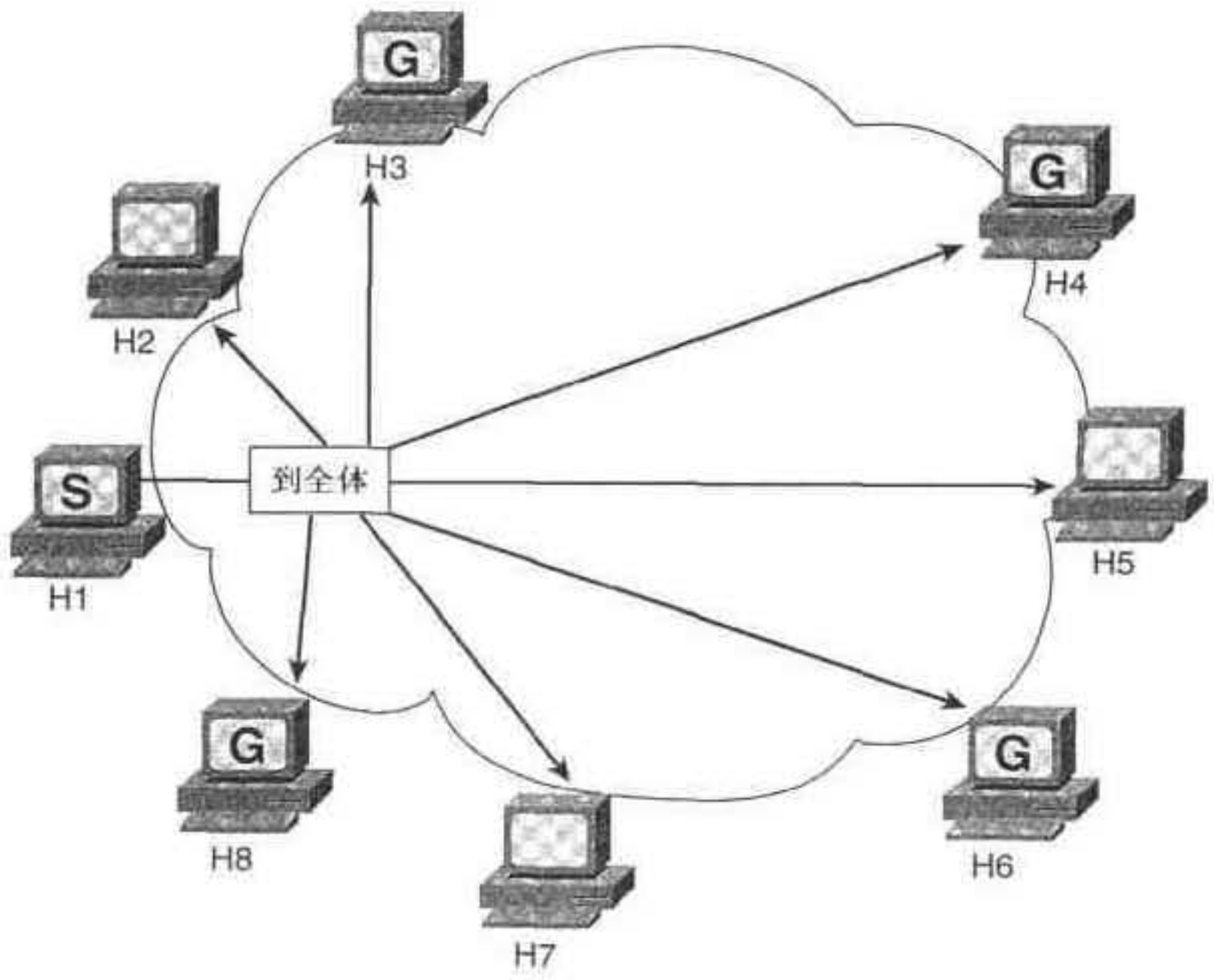


图 5-3　广播数据会给网络增加负担

多播允许源向一个单独的多播目的地址发送一个单独的包，这样避免了复制数据包的负担。任何监听这个多播地址的接收者都可以收到这个数据包，避免了无关的主机处理不需要的数据包。与广播不同的是，有多播能力的路由器可以把多播数据包前转。

IP 多播的很多特性本章并不涉及。本书只关心 IP 的选路，所以本章主要的重点是 IP 多播的选路。其他的内容只是在与选路相关时才提及。对于 IP 多播完整的处理可参阅本章结尾处的“推荐读物”。

5.1　对 IP 多播的要求

IP 多播并非一个新的概念，Steve Deering 于 1986 年撰写了第一个关于多播主机要求的 RFC 文件[1]。但仅过了几年之后，企业对点到多点和多点到多点通信的需求使多播引起了很多人广泛的兴趣。

点到多点应用的实例包括用于远程教学或公司新闻的视频和音频信号、软件发布、基于

网络的娱乐节目、新闻和股市信息，以及数据库或网站的复制。经典的多点到多点的应用是电话、电视会议和共享白板。多玩家的游戏是另一个多点到多点应用，尽管大多数公司并不喜欢把它列到计划中来。随着基于组的应用在实际使用中不断增加，用多播的方法来发送数据包在效率和性能方面比广播和重复单播具有更大的优势，因此引起了更多的注意。

在实现 IP 多播时，必须对各种协议进行选择，目测多播主要是应用在企业网中，因为在这些企业中有一个单独的管理部门有权决定设计的选择。但如果多播的用户越来越多，那么用户要求 ISP 支持整个 Internet 上的多播的压力也会越来越大。由于重复的单播在 Internet 上的流量越来越大，占用了越来越多的带宽，因此 ISP 对多播的兴趣也与日俱增。虽然各大公司对多播的兴趣已经有一段时间了，但最终使 IP 多播走向成熟的所谓“终极应用”将会是 Internet 上的娱乐项目。

多播作为多播骨干网(Multicast Backbone)或 Mbone 被引入 Internet 的一个子网已经有一段时间了。ISP 也开始为用户提供多播业务，比如 UUNET 的 Uucast。但是，在整个 Internet 上实现多播服务还需等待进一步的研究和诸如多协议 BGP(MBGP)与边界网关多播协议(BGMP)等 AS 间协议的发展。当前，还没有一种 IP 多播路由协议支持的选路策略能与 BGP 选路策略相譬美的。除非有充足的工具来加强选路策略，否则多播不太可能会在 Internet 上被广泛地接受。

在需要选路的网际网络中，要支持多播，需满足下述 3 个基本要求：

- 要有能被多播组识别的地址集；
- 要有主机加入和退出组的机制；
- 要有一个能使路由器有效传送多播流量到各个组成员，且不会过度消耗网络资源的路由协议。

本节将探讨每一要求的基本点，下面各节会详细讲述能满足各项要求的当前的各种协议。

1．多播 IP 地址

IANA 把 D 类 IP 地址留给了多播地址。根据第 1 卷第 2 章“TCP/IP 回顾”中描述的第一个 8 位组的规则，D 类地址的前 4 个比特总是 1110，如图 5-4 所示。根据这一限制，D 类地址的范围是 224.0.0.0～239.255.255.255。

规则	最小值与最大值	十进制的范围
Class A: 第一个比特总是0。	**0**0000000 = 0 **0**1111111 = 127	1 ~ 126* *0和127被保留。
Class B: 前二个比特为10。	**10**000000 = 128 **10**111111 = 191	128 ~ 191
Class C: 前三个比特为110。	**110**00000 = 192 **110**11111 = 223	192 ~ 223
Class D: 前四个比特为1110。	**1110**0000 = 224 **1110**1111 = 239	224 ~ 239

图 5-4　D 类地址的范围为 224.0.0.0~239.255.255.255

不同于 A、B、C 类地址，D 类地址是“扁平”的——也就是说，如图 5-5 所示，不使用子网。因此，28bit 就能有 2^{28} 个组(大于 268 000 000)在 D 类地址空间中寻址。

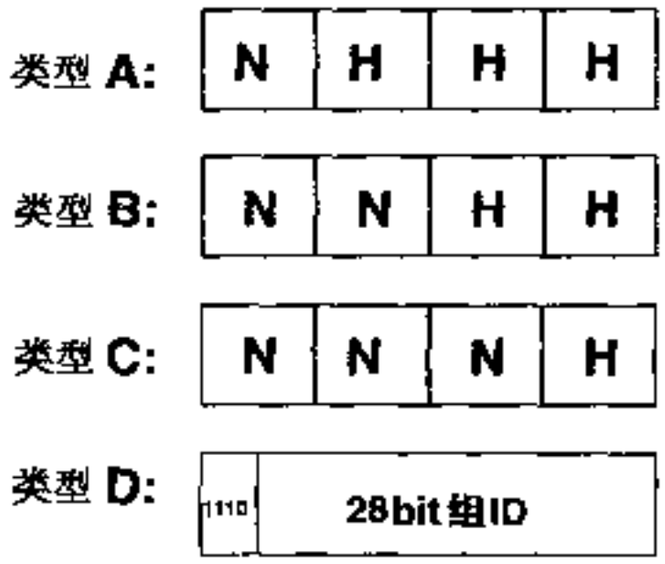


图 5-5　不同于 A、B、C 类地址，D 类地址没有网络部分与主机部分

一个多播组由其多播 IP 地址进行定义；组可以是永久的，也可以是暂时的。“永久”指的是一个组被永久地分配一个地址，而并非是成员被永久地分到一个组中。事实上，主机可以自由地加入或离开任何组。暂时的组，就像字面上的意思一样，并不是长久存在的，比如电视会议。这些组被分配到一些非保留的地址，当这些组不存在时，也同时将地址放弃。

表 5-1 中列出了 IANA 分配给一些永久组的众所周知的地址。你在以前学习路由协议时，曾接触过大部分地址。例如，你知道在一个多路访问的网络中，OSPF DRother 向地址为 224.0.0.6 的 OSPF DR 和 BDR 发送更新消息，DR 向地址 224.0.0.5 的 DRothers 发送数据包。

表 5-1　　一些众所周知的保留的多播地址

地　址	组
224.0.0.1	子网中的所有系统
224.0.0.2	于网中的所有路由器
224.0.0.4	DVMRP 路由器
224.0.0.5	所有 OSPF 路由器
224.0.0.6	OSPF 指定路由器
224.0.0.9	RIP-2 路由器
224.0.0.10	EIGRP 路由器
224.0.0.13	PIM 路由器
224.0.0.15	CBT 路由器
224.0.1.39	Cisco-RP-Announce
224.0.1.40	Cisco-RP-Discovery

IANA 把 224.0.0.0～224.255.255.255 范围内的地址全都保留给了路由协议和其他网络维护功能。多播路由器不会把目的地址在这一范围里的数据包前转，也有些该范围外的地址保

留给了开放或商业多播组；比如 224.0.1.1 保留给网络时间协议(NTP)，224.0.1.8 分配给 SUN NIS+，224.0.6.0～224.0.6.127 分配给 Cornell ISIS 计划。另外一个保留范围是 239.0.0.0～239.255.255.255。这最后一组地址的使用在本章稍后的“多播的范围”一节中讨论。D 类保留地址的完整的列表可以参见附录 C“保留的多播地址”或 RFC 1700。

组员的网络接口卡(NIC)也必须有多播功能。当一台主机加入组时，NIC 决定一个可预知的 MAC 地址。为了实现这一目的，所有具有多播功能的以太网、令牌环和 FDDI 的 NIC 都采用保留的 IEEE 802 地址 0100.5E00.0000 来决定唯一的多播 MAC 地址。这个地址的第 8 个比特为 1 时是有意义的，这一比特在 802 格式中为个人/组(I/G)比特。当这一位被设置时，则表明这个地址是个多播地址。

2. 在以太网和 FDDI 上进行多播

以太网和 FDDI 接口把组 IP 地址的最后 23bit 映射到保留的 MAC 地址的最后 23bit 组成一个多播 MAC 地址，如图 5-6 所示。图中，以太网与 FDDI 网的多播 MAC 地址由 IP 地址的最后 23bit 连接在 MAC 地址 0100.5E00.0000 的前 256bit 后构成。在这里，D 类 IP 地址 235.147.18.23 被用来创建 MAC 地址 0100.5E13.1217。

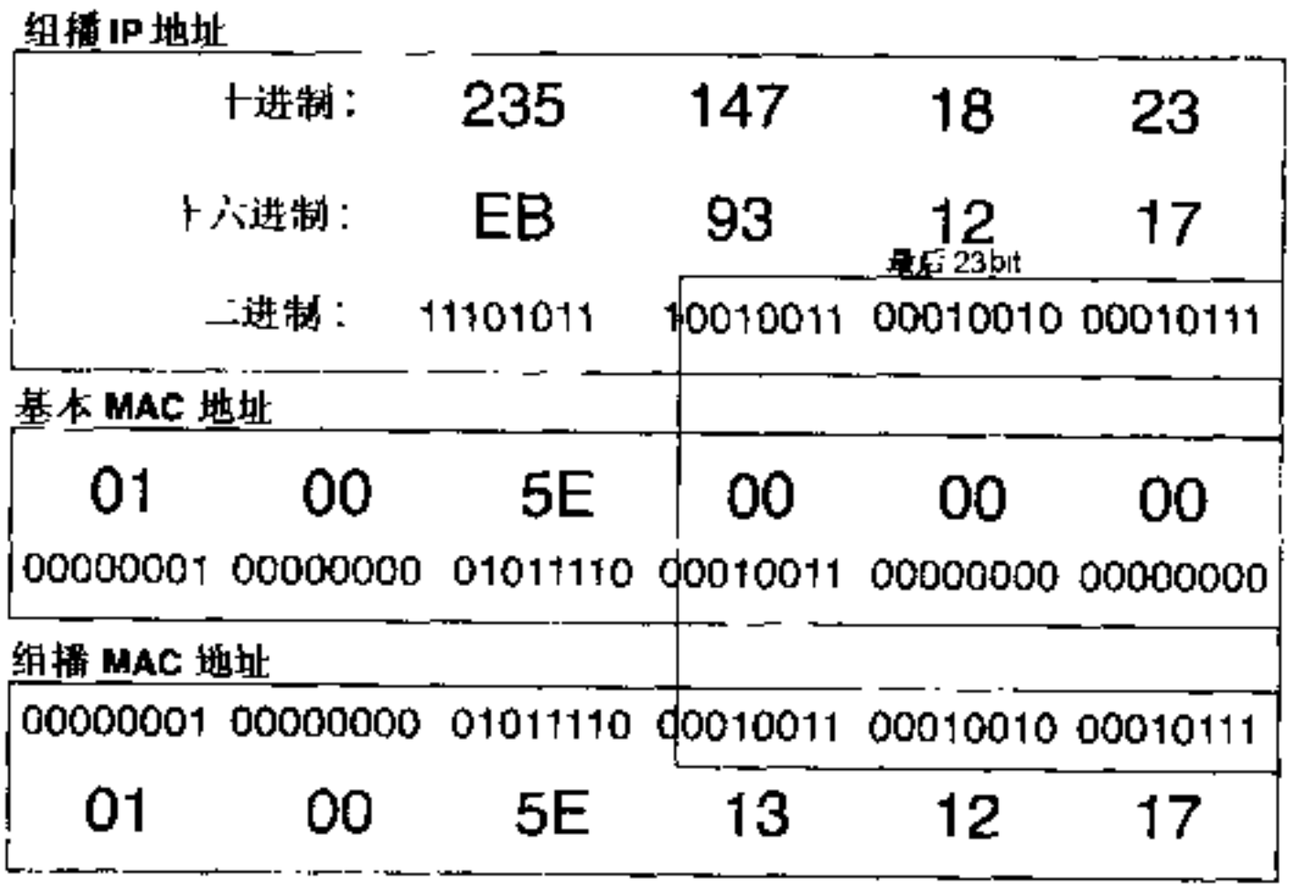


图 5-6　以太网与 FDDI 网的多播 MAC 地址构成

你已经多次遇到这些地址。回忆第 1 卷第 9 章“开放最短路径优先”中简单描述了所有 OSPF 路由器地址 224.0.0.5 使用 MAC 地址 0100.5E00.0005，所有 OSPF 指定路由器地址 224.0.0.6 使用 MAC 地址 0100.5E00.0006，现在你应该知道这是为什么了。

因为只有 IP 地址的最后 23bit 被映射到 MAC 地址中，导致多播 MAC 地址不是唯一的，比如 IP 地址 225.19.18.23 会同 235.147.18.23 一样，产生 MAC 地址 0100.5E13.1217。事实上，将 D 类地址的数量(2^{28})和所有具有保留前缀的可能的 MAC 地址数目(2^{23})相比进行计算，可以看到 32 个不同的 D 类 IP 地址能映射成一个 MAC 地址。

IETF 的观点是，这种在同一 LAN 中有两个或多个组地址产生相同 MAC 地址的不正常情况的可能性极小。在极少的情况下，这样的冲突会发生，LAN 中两个组内的成员会收到彼此的流量。多数情况下，每一个组的包会到达不同的端口号或有不同应用层的认证方式；每一个组的成员会在传输层或更高层丢弃别的组的数据包。

这种可预见的 MAC 地址的方法具有双重性：

• 局域网中的多播源或路由器为了让LAN中的所有成员都能收到数据包，只能向多播MAC地址发送单独的帧；

• 因为如果组的地址是已知的，那么MAC地址也已知，所以就不需要ARP的处理。

3．令牌环上的多播

在令牌环网上传送多播用不同的方式进行处理。令牌环定义了特定的亦称为与功能相关的MAC地址来寻找如Active Monitor、Ring Parameter Server和Ring Error Monitor等一般TR功能运行的工作站。TR MAC地址的第一个八位组的第一个比特是I/G地址，它指示这个地址是单播地址(I/G=0)还是广播/多播地址(I/G=1)。第二个比特是通用/本地(U/L)比特，它用于指明这个地址是厂商烧入的(U/L=0)还是本地管理的(U/L=1)。另外，第二个八位组的第一个比特是特定地址指示(FAI)，其作用是把特定地址(I/G=1，U/L=1，FAI=0)和本地管理的组地址(I/G=1，U/L=1，FAI=1)区分开。一个具体的特定地址就是把FAI后余下的31个比特位中的一个，且仅一个设为1。比如，Active Monitor的特定地址是C000.0000.0001，桥接为C000.0000.0100。因为31个比特中仅有一个可以设定，所以有31个特定地址。这个规定对IP多播很重要。

令牌环MAC地址采用小端格式，即每一个八位组从右向左读取；以太网采用大端格式，每个八位组从左向右读取。因此，以太网多播MAC地址0100.5E13.1217在令牌环中读为8000.7AC8.48E6。TR地址的FAI比特为0，可是其后31个比特中不只一个比特设为1，因此令牌环会认为这个地址是非法的特定地址。

注：FDDI也采用小端格式，不过它不像令牌环那样使用特定地址，因此能支持像以太网一样的地址映射方法。

因为IP地址不能像映射到以太网地址一样被映射到令牌环地址中，所以采用了另一种方法来解决这个问题。当前，有两种方法用于为TR帧承载IP多播包分配地址[2]：

• 对承载多播的帧都用广播地址FFFF.FFFF.FFFF；

• 用一个单独保留的特定地址C000.0004.0000。

Cisco路由器对第一种方式是默认的，而对于第二种方式，则可以用**ip multicast use-functional**命令来配置TR接口。

这两种方法都有缺陷。第一种方法的效率很低，把多播包传送到环上的每一个工作站，依赖于高层协议是接收还是拒绝数据包。第二种方法只能在环上的所有工作站上的TR NIC都能识别这个特定地址的情况下才能使用，但并不是所有的NIC都能做到。第二种方法的另一个问题是能识别这个特定地址的TR NIC向工作站的CPU发送中断。如果环上的IP多播流量并不很大，特别是当几个组的多播流量都映射到一个特定地址上时，主机的性能会受到很大的影响。因为这些限制，所以对于IP多播，令牌环不是个很好的选择。

5.2　组成员概念

在一个主机加入组之前，它(或它的用户)必须知道有哪些组可以加入，如何加入。可以有各种不同的机制用于对外告知多播组，就像在线的“电视预告”，或如图5-7所示的基于Web的时间表。图中显示了一种用基于Web的公告方式来确定多播组的方式，这样的MBone

会话的时间表可以在 www.cilea.it/MBone/browse.htm 上找到。

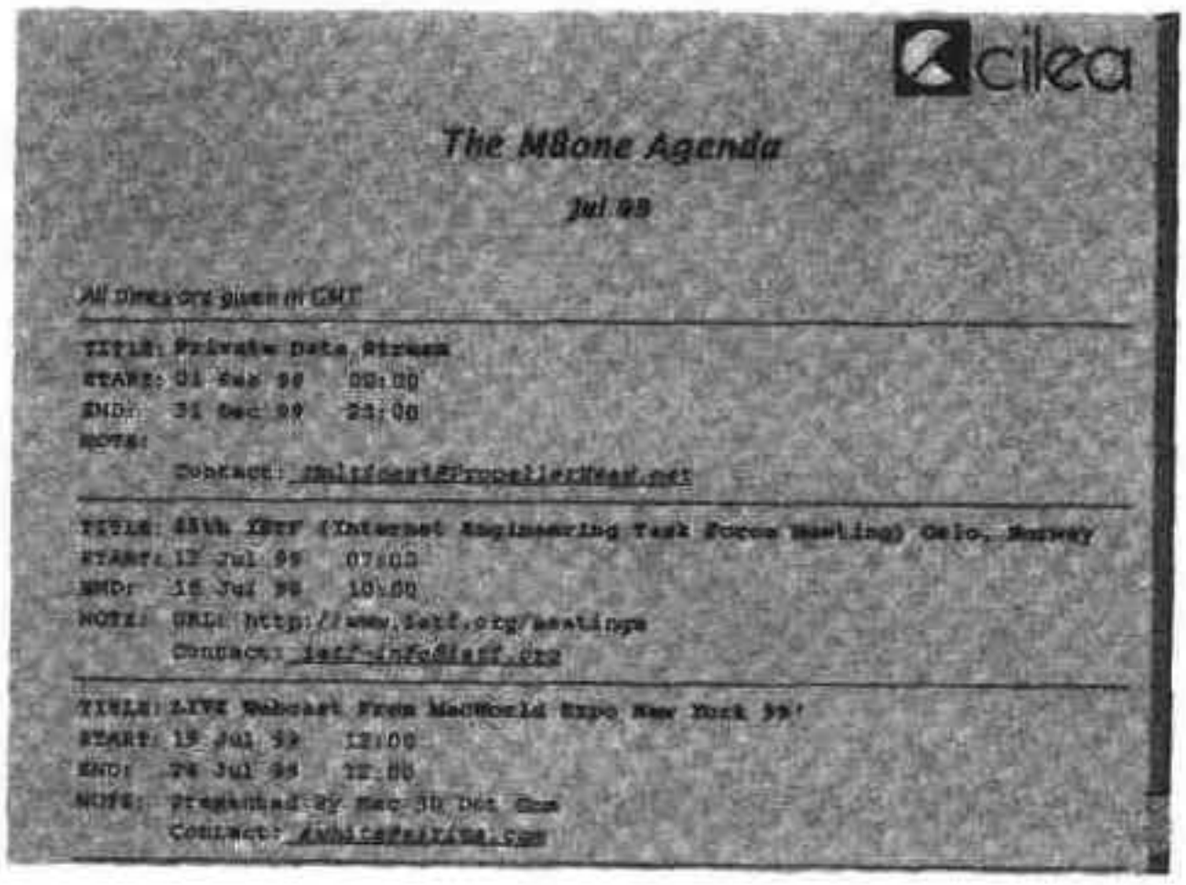


图 5-7　基于 web 的时间表

也有使用诸如用于描述多播事件和对外告知这些描述的会话描述协议(SDP)和会话宣告协议(SAP)这一类的工具的。图 5-8 显式了一个采用这些协议的应用的例子。一个用户可能经过邀请，比如通过 E-mail 来得知一个多播会话。

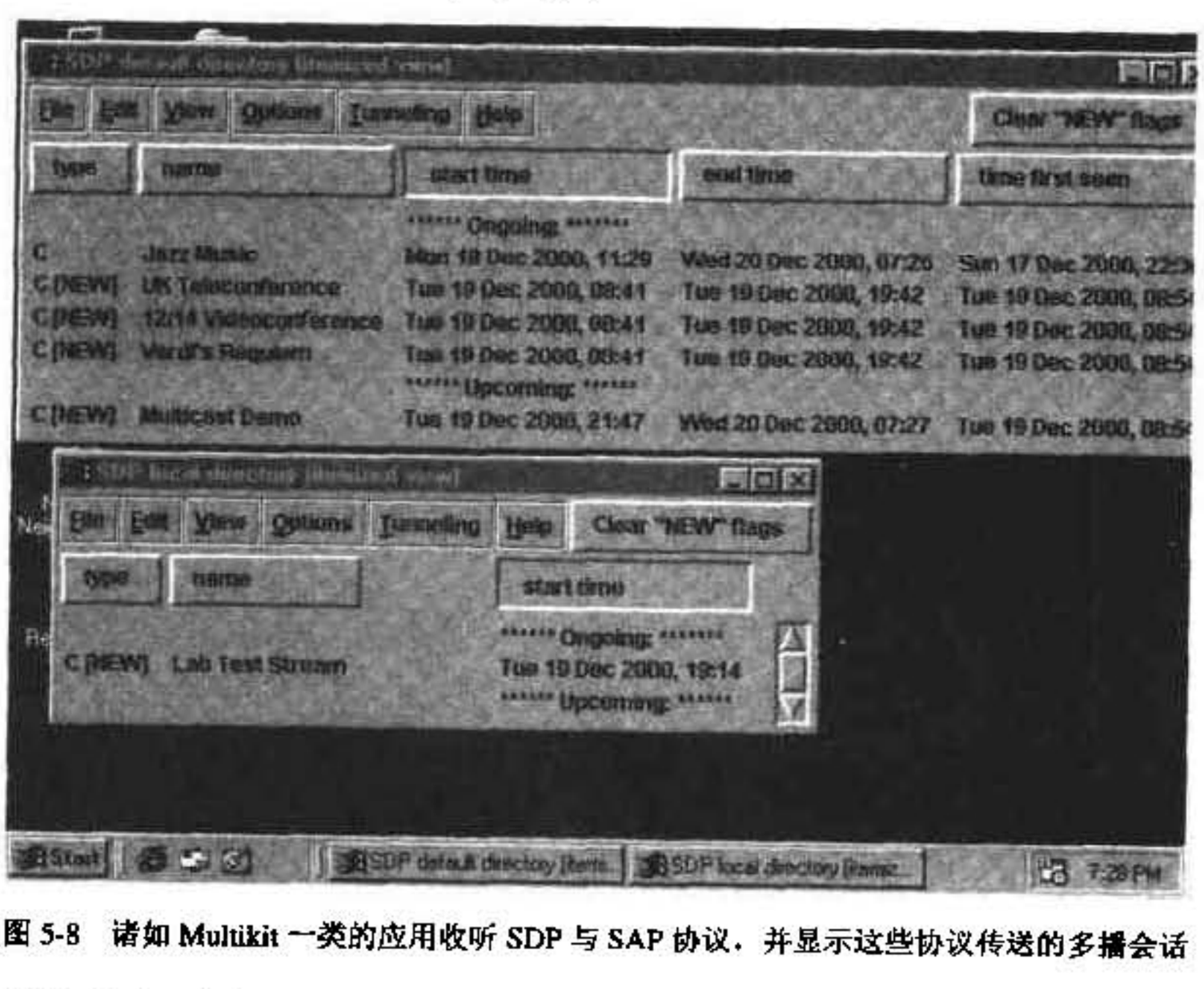


图 5-8　诸如 Multikit 一类的应用收听 SDP 与 SAP 协议，并显示这些协议传送的多播会话

对这些机制的详细讨论已经超出了本书的范围。本节假设主机已经知道多播组，所以主要讨论加入和离开组的问题。在讨论完这些问题后，你会看到在独立的子网上既成事实的 IP 多播组管理协议——Internet 组管理协议(IGMP)如何处理这些问题。

5.2.1　加入和退出组

一个多播会话的多播源并不一定要成为接收它发送的流量的组中的成员。事实上，在典

型情况下，源并不知道什么主机是组员。接收者任何时候都能自由加入和离开小组。这和早先电台和电视的类比相一致：观众可以随时调换频道，而电台不能直接了解到谁在收听。

如果源和所有的组员共享一个 LAN，那么就不需要其他的协议。源向多播 IP 地址(与多播 MAC 地址)发送数据包。不过发送的多播流量要通过需经选路的网际网络，问题就要复杂得多。每个路由器只能把所有的多播包前转到每个局域网上，以防局域网上有组员。不过这样绕开了多播的目的：保存网络资源。如果局域网内没有组员，那么不仅在子网内，而且所有通路上的数据链路和路由器上的带宽、处理过程会有很大的浪费。

因此，一台路由器必须通过某种方法获知连接的网络中是否有组员，如果有，是哪个组的组员。当一个路由器知晓有一个多播会话时，它会向其所属的子网查询是否有主机要加入这个接收组。这个查询会发向“子网中所有的系统”地址 224.0.0.1，或它可以向组内正在查询的某一特定地址发送。如果有一个或多个主机回应，那么这个路由器会将会话的数据包前转到正确的子网中，如图 5-9 所示。

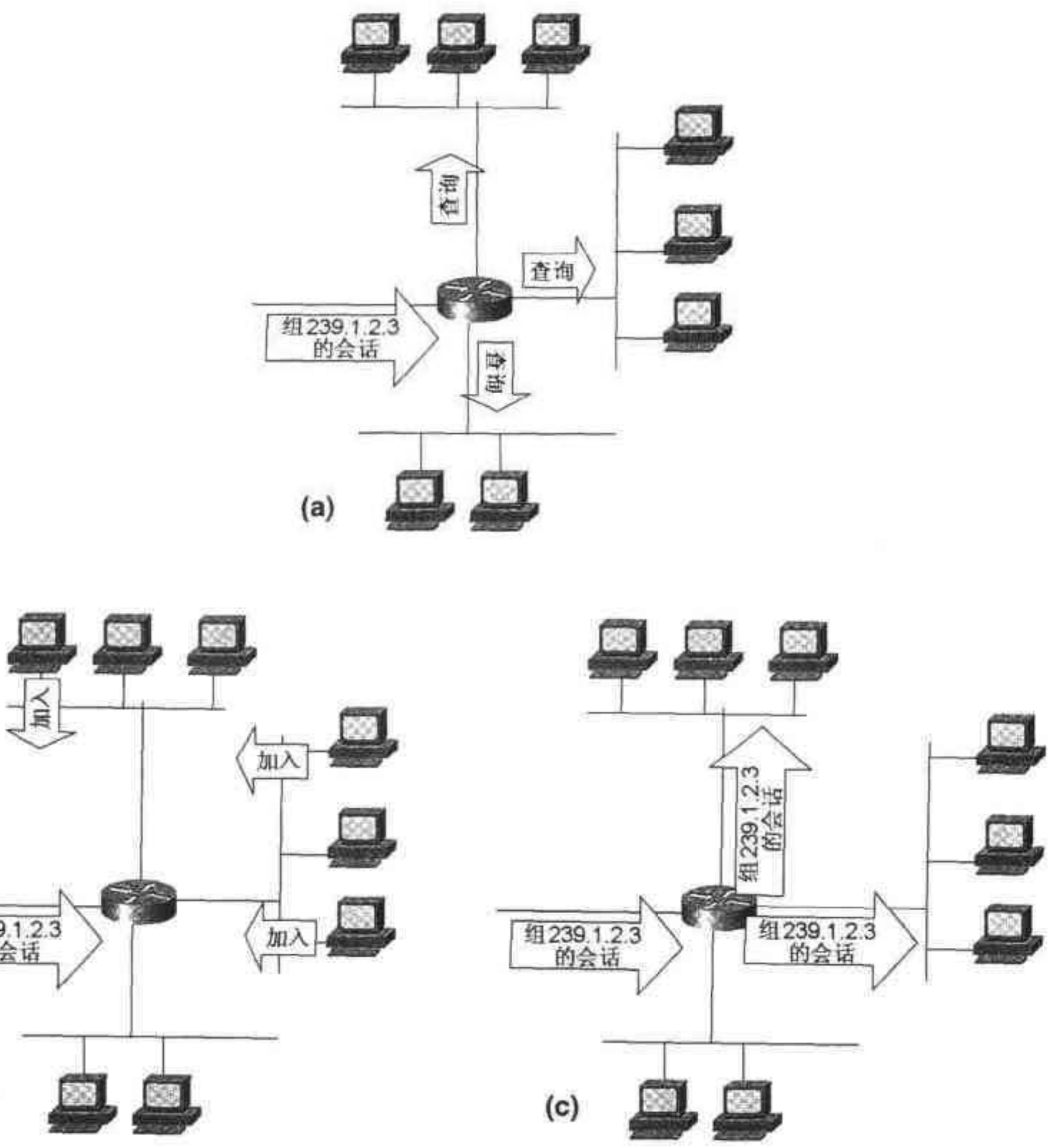


图 5-9　多播组成员的发现

路由器可以周期性地重复向子网发送查询。如果子网中仍有组员，那么它们会响应所有的查询，让路由器知道它们在组中仍是激活的。如果没有主机回应，那么路由器就认为子网中所有的主机都已经离开这个组，于是停止向这个子网前转数据包。

1. 加入时延

迄今为止，这种方法带来的问题是如果主机知道它要加入的组，那么要让主机等待路由器向组进行查询就不很实际了。为了减少这段等待时间，主机可以无需等待，直接向路由器发送一个请求加入的消息。收到这个加入请求后，路由器马上把多播流量前转到这个子网中。

这种处理不仅是对局部的子网有好处。在本章稍后的“多播路由概念”一节中，你会发现由主机发起加入组会使多播路由协议更加有效。如果一台路由器，其所有连接的子网内没有组员，并且这些子网也不在到别的路由器的多播流量通路上，那么这台路由器自己就可以请求上游的路由器不要向其前转多播流量。这样的结果就是，流量不会流入没有组员的网络的其他部分中。如果路由器收到与其连接的子网中的一个加入请求，它就会向上游请求收到相关数据流。

这种方法不利的一面是如果主机向本地的路由器发送一个加入请求，接着必须等待路由器向上游请求相应的流量，则加入时延会增加。加入时延是从主机发出加入请求到主机真正收到组内流量的时间间隔。当然，如果当主机想加入时，子网中已经有了别的组员，那么加入时延实际为 0。主机并不一定要向路由器发送加入请求，它可以监听前转给其子网中其他组员的包。

2. 退出时延

当一台主机退出一个组时，允许其明确通知本地的路由器也能提高效率。路由器不需等待子网中是否有主机响应它的查询来断定子网中没有组员，它可以主动判断子网中是否还有组员。收到主机的退出通知后，路由器马上向这个子网发送一个查询，询问是否还有剩余的组员。如果没有响应，路由器就认为已没有组员，于是停止向这个子网前转数据包。这样的结果可以降低退出时延，退出时延就是子网中最后一个组员退出组与路由器停止向子网前转组内流量的时间间隔。

由主机发起的退出也能提高路由协议的效率。如果路由器知道它连接的任何子网中没有任何组员，那么它可以把自己从多播树中剪除。路由器越早判断出没有组员，它就能越早把自己剪除。

减少加入和退出时延也能提高多播网络整体的质量。主机可能知道很多的组。较低的加入和退出时延意味着终端用户可以容易地用同一种方式在不同组间选择，就像用户随意切换电台和电视台频道一样。

3. 组的维护

主机向路由器发送的用于指示其希望参加一个组的消息称为报告。主机可以向多种可能的目的地址发送报告：

- 这个报告通过单播发向发出查询的路由器。这儿有一个问题，就是子网中可能有不只一个的路由器在搜寻这个组。所有相关的路由器都必须能收到这个报告。
- 这个报告可以发向“子网中的所有路由器”224.0.0.2 这个地址。不过你很快会发现，子网中的其他组员也能收到这个报告是很有用的。

- 为了保证子网中的其他组员也收到这个报告，可以把报告发向“子网中的所有系统”224.0.0.1 这个地址。不过这样的方法降低了多播的效率，从而会强制子网内所有支持多播的主机，而不仅仅是组员，都在第二层以上处理这个报告。
- 这个报告可以向这个组地址发送。这种方法可以保证子网中所有的组员和监听组员的所有路由器都能收到这个报告。不是组员的主机的 NIC 可根据第二层地址将这个报告拒绝。

如果子网中所有的组员都响应一个查询，那会无益地浪费带宽。毕竟，路由器只需知道子网中至少有一个组员，而不必了解具体有多少个组员，它们分别是谁。全体组员响应同一查询的另一个问题是当所有组员马上响应时，可能会产生碰撞。后退重传会消耗更多的网络与主机资源。如果子网中有多个组员，那么在每个组员都发送完的报告结束之前，有可能发生多次的碰撞。

向组地址发送报告能减少子网中多次的报告。当收到一个查询后，每一个组员启动一个随机时长的计时器，组员在计时器超时前不发送报告。由于这个计时器是随机的，因此非常有可能一个组员的计时器先于其他定时器超时。这个组员发出一个报告，因为这个报告是发向组地址的，所有的组员都能收到。其他的组员在收到这个报告后，就停止自己的计时器，并且不会发送自己的报告。结果，子网中只会有一个报告，一个子网中只一个报告正是路由器所需要的。

4．网络中有多个路由器

前一节提到子网中有多个路由器的可能性是不小的，每一个路由器都需要了解子网中是否有组员，图 5-10 中显示了一个例子。两台路由器连在同一子网中，它们经过不同的路由收到同一源的多播流。如果一台路由器或一条路由失效，组员则能继续从另一台路由器收到多播的会话。一般情况下，两台路由器同时向子网转发相同的数据流不是有效率的做法。

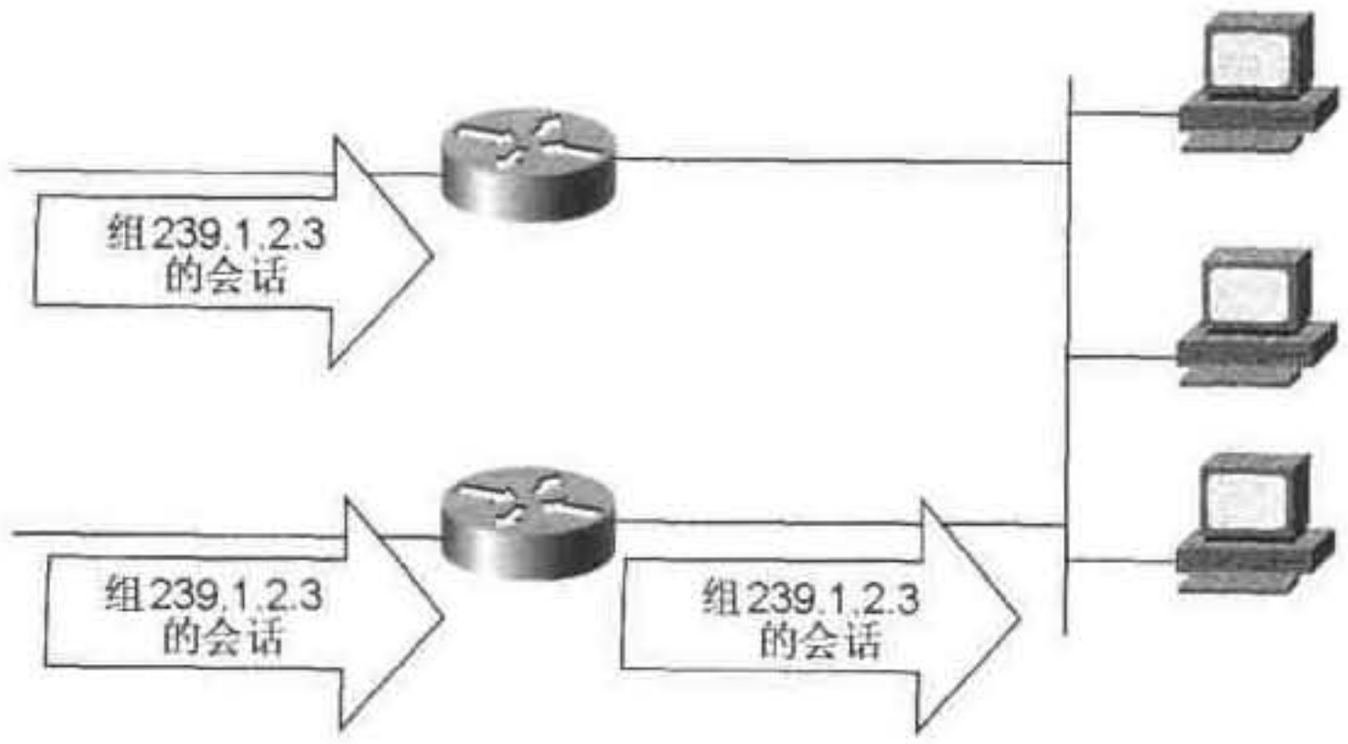


图 5-10　两个路由器收到相同的多播会话，但只有一个把数据前转到子网中

路由器通过路由协议能够彼此知道对方的存在。因此，保证只有一台路由器向同一子网前转数据的方法是在多播路由协议中加入指定路由器(或查询者)功能。查询者负责多播流的前转，其他的路由器只进行监听，只有当查询者失效时才开始前转数据流。

用路由协议来选出查询者的问题是现在有多种 IP 多播协议。如果图 5-10 中的两台路由器运行不兼容的路由协议，那么它们各自的查询者选举的过程不能检测到彼此的存在。每种路由协议都会决定自己的查询者，双方都会前转数据流。

不过，本地的组管理协议是独立于路由协议的。路由器必须运行这一通用协议来查询组员，所以给组管理协议赋予查询者的功能是很合理的。这能保证子网中的路由器使用通用的语言，可以对由哪一台路由器进行前转达成共识。

5.2.2　因特网组管理协议(IGMP)

不论多播网际网络中使用了多少种路由协议，IGMP 始终是主机和路由器间的“语言”。所有要加入多播组的主机和所有连接到有多播主机的子网中的路由器都必须使用 IGMP。这是一种和 ICMP 有很多相似之处的控制协议。同 ICMP 一样，它负责高层数据的交换。IGMP 消息被封装在如同 ICMP 一样的 IP 包头中(协议号为 2)，不过与 ICMP 不同的是，这消息限制在本地的数据链路上。IGMP 实现规则要求路由器不能把 IGMP 消息进行前转，IGMP 的 IP 包头中的 TTL 参数也总设为 1，这两点保证了 IGMP 的应用范围。

现在有两种版本的 IGMP：IGMPv1 在 RFC1112[3] 中描述，IGMPv2 在 RFC2236[4] 中描述。Cisco IOS Software Release 11.1 和随后的版本中默认支持 IGMPv2；不过有很多 TCP/IP 的应用仍只支持 v1(比如安装比 SP4 更早版本的 Windows NT 4.0)。因为这个原因，这个默认值可以通过 **ip igmp version** 命令进行修改。

下面的章节讨论 IGMPv2，并将其与 IGMPv1 进行比较。IGMPv3[5] 也已提出，尽管 IOS 现在还不支持。不过本章也简单讨论了 v3，预期 Cisco IOS Software 在近期会支持它。

1．IGMPv2 主机的功能

运行 IGMPv2 的主机使用 3 类消息：

- Membership Report 消息
- Version 1 Membership Report 消息
- Leave Group 消息

Membership Report 消息用于指示一台主机希望加入一个组，这个消息在一台主机第一次加入组时发送，有时也用来响应本地路由器发出的 Membership Query 消息。

当一台主机第一次知道一个组，并且希望加入时，它并不等待本地的路由器发送查询。正如在本章介绍到的各种路由协议，路由器可能不会，也许根本就不知道主机要参加哪个组，因此也就不必向组员询问了。如果主机必须等待查询，那它也许永远也不会有机会加入多播组。相反地，当主机首次加入一个组时，它会主动向组发送 Membership Report 消息。

路由器通过一对(源、组)地址识别多播会话，源地址是会话发起者的地址，组地址是 D 类组地址。如果本地的多播路由器事先并不知道主机要加入的多播会话的相关信息，那么它将向数据源发送一个上行的请求。收到数据流后，路由器开始把这些数据前转到子网中请求成为多播组成员的主机处。

Membership Report 消息 IP 包头里的目的地址是这个组的地址，这个消息本身也有该组的组地址。为了保证本地的路由器能够收到主动发出的 Membership Report 消息，主机在短时间内发出一个或两个复制的报告，在 RFC 2236 中建议这个时间间隔为 10s。

IGMPv2 主机为了后向的兼容性，支持 IGMPv1 Membership Report 消息。IGMPv2 用于检测和支持其子网中的 IGMPv1 主机和路由器的机制在“IGMPv1 同 IGMPv2 的比较”一节中讨论。

注：　所有支持多播的设备均是“子网中的所有系统”组的组员，组地址为 224.0.0.1。因为这一默认的设置，主机不向这个组发送 Membership Report 消息。

因为 Membership Report 消息的目的地址是组地址，所以除了路由器之外，子网中的其他组员也可能听到这个报告。如果主机在它的计时器超时前收到一个 Membership Report 消息，它将不再向这个组发送 Membership Report 消息。通过这种方法，路由器被告知子网中至少有一个组员，而不需所有组员的报告充斥整个子网。

当主机退出一个组时，它用 Leave Group 消息通知本地的路由器。这个消息包含有退出的组的地址，但与 Membership Report 消息不同的是，Leave Group 消息是发向“子网中的所有路由器”地址 224.0.0.2 的。这是因为只有子网中的多播路由器需要知道主机已经退出了，而其他组员不需要知道。

RFC2236 推荐只在退出的成员为最后一个发出 Membership Report 消息来响应查询消息的主机时才发出 Leave Group 消息。如下一节中所解释的，本地的路由器总是通过查询子网中是否还有组员来响应 Leave Group 消息的。如果是组员，而不是“最后的响应者”，悄悄离开这个组，那么路由器仍继续前转会话，而不进行查询，因此能够节省一点带宽，不过并不一定要求这样做。如果多播应用的设计者不想为记忆一台主机是否为最后一个响应查询消息而设计状态变量，那么应用可以总是在退出组时发送 Leave Group 消息。

2．IGMPv2 路由器功能

路由器发送的 IGMP 消息只有查询一种类型。在 IGMPv2 中，有两类查询：

- General Query
- Group-Specific Query

路由器通过 General Query 消息向与其连接的所有子网进行轮询来发现是否有组员存在，并在子网中没有组员时检测到这一情况。查询可按默认的每 60s 的时间间隔发送；这个值可通过 **ip igmp query-interveal** 命令在 0～65 535 范围内设置。

如前一节中描述，这个查询包含一个称为 Max Response Time 的值。这个值规定了主机用 Membership Report 消息响应这个查询的最长等待时间。这个值默认为 10s，但你可以用 **ip igmp query-max-response-time** 命令来改变它。该值在这个消息中占了 8bit，单位是 1/10s(尽管 **ip igmp query-max-response-time** 是以 s 为单位来设定这个值的)。比如，默认的 10s 在这个消息中表示为 100 个 1/10s。因此，这个值的范围可以设为 0～255 个 1/10s，或 0～25.5s。

General Query 消息被发向“子网中的所有系统”224.0.0.1 这个地址，而且不涉及任何一个具体组。这样的结果就是这一个消息会轮询子网中所有组中可能存在的组员。路由器跟踪已知的组和连接到有组员的子网里的网络接口，如例 5-1 中的输出显示。

如果一台 Cisco 多播路由器在 3 次查询的时间间隔里(默认为 3min)没有收到一个特定子网中的 Membership Report 消息，那么这个路由器将宣布这个子网中没有组员。这包括了唯一的组员切断连接或没有遵从 IGMPv2 退出组的规定等可能情况。

例 5-1　show ip igmp groups 命令显示了路由器所知晓的 IP 多播组

```
Gold#show ip igmp groups
IGMP Connected Group Membership
Group Address    Interface          Uptime    Expires   Last Reporter
224.0.1.40       Serial0/1.306      3d01h     never     0.0.0.0
228.0.5.3        Ethernet0/0        00:09:07  00:02:55  172.16.1.254
239.1.2.3        Ethernet0/0        1d08h     00:02:53  172.16.1.23

Gold#
```

注：　这与 RFC2236 中规定不同的是，这个文件中的规定为两次查询的时间间隔加上一个 Max Response Time 时间间隔。

主机在正常情况下退出组时要发送一个 Leave Group 消息。当路由器收到 Leave Group 消息时，必须判断子网中是否仍有组员存在。为了达到这个目的，路由器发送一个 Group-Specific Query 消息，与 Grenerat Query 消息不同的是，它包含组的地址，并且用组的地址作为目的地址。

如果 Group-Specific Query 消息被丢弃或受到了破坏，子网中其他的组员是不会发送报告的。这样的话，路由器会错误地认为子网中没有组员，而停止前转会话包。为了防止这种可能性，路由器会每隔 1s 分别发送两个 Group-Specific Query 消息。

当支持多播的路由器首先在子网中被激活，它认为自己是查询者——负责向子网发送所有 General Query 和 Group-Specific Query 消息的路由器时，会马上发送 General Query 消息。

注：　RFC 2236 推荐发送多条查询消息；可是，Cisco 的 IGMPv2 只发送一条。

这个行为能快速发现子网中的激活的组员，并通知给子网中可能存在的其他多播路由器。当子网中有多个路由器时，选举查询者的规则就十分简单：有较小 IP 地址的路由器成为查询者。所以子网中现有的路由器在收到新路由器的 General Qucry 消息后，就检查源地址。如果它的 IP 地址的值更小，则会继续发送查询。当新的路由器收到其中一个查询，并发现这台路由器有较小的 IP 地址时，它就变成非查询者。

如果非查询者在一段时间(称为 Other Query Present Interval)内没有收到查询者的查询，那么它认为查询者已经不存在了，并充当这个角色。Cisco IOS 软件中 Other Query Present Interval 的默认值为查询时间间隔的两倍，即 120s；可以通过 **ip igmp query-timeout** 命令来改变该值。

3．IGMPv1

IGMPv1 与 IGMPv2 的重要差别如下：

- IGMPv1 没有 Leave Group 消息，这就意味着从最后一个组员退出组到路由器停止转发组的流量之间的时间间隔会更长；
- IGMPv1 没有 Group-Specfic Query 消息，这与没有 Leave Group 消息相一致；
- IGMPv1 不在查询消息中规定最大响应时间。相反地，主机有一个固定的最大响应时间，为 10s；
- IGMPv1 没有查询者选举的过程，它是依靠 IP 多播路由协议来选出子网中的指定路由器。因为不同的协议采用了不同的选举机制，因此在 IGMPv1 中，有可能在子网中有多个

查询者；

在“IGMP 消息格式”一节中可以看到这些差异对 IGMPv1 和 IGMPv2 消息格式的影响。

在某些情况下，IGMPv1 和 IGMPv2 可能同时在同一子网中使用：

- 某些组员运行 IGMPv1，而另一些运行 IGMPv2。
- 某些组员运行 IGMPv2，而路由器运行 IGMPv1。
- 路由器运行 IGMPv2，而某些组员运行 IGMPv1。
- 一台路由器运行 IGMPv1，而子网中的另一路由器运行 IGMPv2。

RFC2236 中描述了一些机制，可以使 IGMPv2 能满足这些情况。如果同一子网中同时有版本 1 和版本 2 的成员，那么版本 2 的成员在决定是否取消自己的 Membership Report 消息时，将对版本 1 和版本 2 的 Membership Report 以同样的方式进行处理。也就是说，如果一个版本 2 的成员听到路由器的查询，随后又收到同一组中版本 1 的 Membership Report 消息，那么它就不再发送自己的 Membership Report 消息。另一方面，版本 1 的主机则忽略版本 2 的消息。但是，如果一个版本 2 的 Membership Report 消息已经先被发送到组了，则版本 1 的成员在计时器超时后仍将发送它的报告。这不会对版本 2 的主机造成影响。对于版本 2 的路由器来说，知道版本 1 组员的存在是很重要的。

如果一台主机运行版本 2，而本地路由器运行版本 1，那么 IGMPv1 路由器将忽略版本 2 的消息。所以版本 2 的主机收到版本 1 的查询后，将用版本 1 的 Membership Report 消息来进行响应。IGMPv1 的查询没有定义最大响应时间，所以 IGMPv2 的主机采用版本 1 中固定的 10s 的时长。主机可以，也可以不发送 Leave Group 消息，IGMPv1 路由器不能识别这个消息，将不理会它。

如果版本 2 的路由器收到版本 1 的 Membership Report 消息，它会将所有的组员按版本 1 对待。路由器会忽略 Leave Group 消息，也不发送 Group-Specific Query，因为版本 1 的成员不理会这个消息。相反地，它会设定一个称为旧主机存在计时器(Old Host Present Time)的时钟(如例 5-2)。这个计时器的时间与组成员资格时间间隔是一样的。每当收到一个新的版本 1 的 Membership Report 消息，这个计时器就会重新设置；如果计时器超时，那么路由器会认为子网中没有版本 1 的组员，而恢复版本 2 的消息和处理过程。

> 注：　如先前描述的一样，组成员资格时间间隔是路由器宣布子网中没有组员前等待的时间，Cisco 的默认值为查询间隔的 3 倍。

例 5-2　为版本 1 的组设置一个计时器

```
Gold#debug ip igmp
IGMP debugging is on
Gold#
IGMP: Send v2 Query on Ethernet0/0 to 224.0.0.1
IGMP: Received v2 Report from 172.16.1.23 (Ethernet0/0) for 239.1.2.3
IGMP: Received v1 Report from 172.16.1.254 (Ethernet0/0) for 228.0.5.3
IGMP: Starting old host present timer for 228.0.5.3 on Ethernet0/0
IGMP: Send v2 Query on Ethernet0/0 to 224.0.0.1
IGMP: Received v2 Report from 172.16.1.23 (Ethernet0/0) for 239.1.2.3
IGMP: Received v1 Report from 172.16.1.254 (Ethernet0/0) for 228.0.5.3
IGMP: Starting old host present timer for 228.0.5.3 on Ethernet0/0
```

注意到例 5-2 中的路由器不断发送版本 2 的 General Queries 消息，这和版本 1 的查询最大的不同之处在于最大响应时间非 0。这个例子中多播路由器收到一个组 239.1.2.3 的 IGMPv2 Membership Report 消息与组 228.0.5.3 的 IGMPv1 Membership Report 消息。这个版本 1 的消息会使路由器为这个组设置一个 Old Host Present Timer。在版本 1 中这一参数是不使用的，版本 1 的主机将忽略它。结果是，主机将版本 2 的查询理解为版本 1 的查询。

例 5-2 中另一有趣点是只为组 228.0.5.3 设定了 Old Host Present Timer。路由器只将这一个组当作 IGMPv1 组来处理，而同一接口上的组 239.1.2.3 仍按版本 2 处理。

如果版本 1 和版本 2 的路由器同时存在于同一子网中，那么版本 1 的路由器不会参与查询者的选举。因为这一点，因此为了同版本 1 的路由器保持一致，版本 2 的路由器应如版本 1 那样运行。但是，版本 2 的路由器并不会自动转换成版本 1 路由器，必须通过 **ip igmp version1**，命令人工对其进行配置。

4．IGMPv3

因为 IGMPv3 仍在开发阶段，尚未被支持，所以这一节没有像对版本 1 和版本 2 那样，对其进行详细讨论。本节会对这一版本真正应用时增加的主要特点进行总结。

IGMPv3 中主要增加了 Group-and-Source-Specific Query，它能允许一个组不仅用组地址进行识别，而且也能被源地址识别。为了能支持这一识别，Membership Report 和 Leave Group 消息也进行了修改。

当一个组有多个源(多点到多点的组)时，IGMPv3 路由器可以根据组员的请求对源进行过滤。比如，某一特定组员可能希望只从某一特定的源地址收到组的流量，或可能希望收到除了某些源地址之外的所有流量。这个组员可以在 Membership Report 消息的包含和不包含过滤请求中表达它的愿望。如果一个特定子网中没有一个组员想收到特定源的流量，那么路由器不会把这个源的流量前转到这个子网中。

5．IGMP 消息格式

IGMPv2 采用了一种消息格式，如图 5-11 所示。封装这个消息的 IP 包头的协议号为 2。因为 IGMP 消息不会离开产生它的那个子网，所以 TTL 总是设为 1。另外，IGMPv2 消息承载着 IP 路由器告警选项，以通知各路由器“更仔细地检查这个包”[6]。

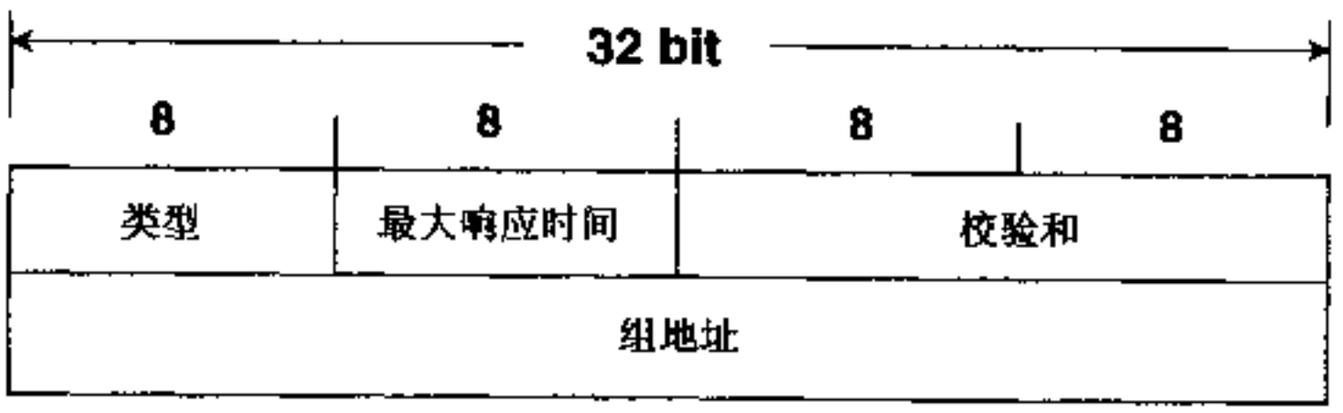


图 5-11　IGMPv2 消息格式

IGMPv2 消息的各部分的参数定义如下：

- 类型为下面 4 种消息之一：

— Membership Query(0x11)用于多播路由器发现子网中的组员。General Membership Query 消息将组地址设为 0.0.0.0，而 Group-Specific Query 将这个值设为要查询的组的地址。

— 版本 2 的 Membership Report(0x16)由组员发送，用于告知路由器子网中至少有一个

组员存在。

— 版本 1 的 Membership Report(0x12)用于 IGMPv2 的主机与 IGMPv1 的后向兼容。

— 若组员为最后一个发送 Membership Report 的主机，则当它退出组时，它会发送 Leave Group(0x17)，以通知路由器。

• 最大响应时间只在查询消息中设置。在其他所有消息类型中，这个值设为 0x00。这个值定义了一个以 1/10s 为单位的时间段，在这段时间里至少要有一个组员响应 Membership Report 消息。

• 校验和是一个 IGMP 消息的 16bit 的反码和的反码。这是 TCP/IP 中使用的标准校验算法。

• 组地址在 General Query 消息中设为 0.0.0.0，在 Group-Specific 消息中设为组的地址。Membership Report 消息中的这个值为报告的组地址；Leave Group 消息中的这个值为退出组的地址。

图 5-12 中显示了 IGMPv1 消息格式。

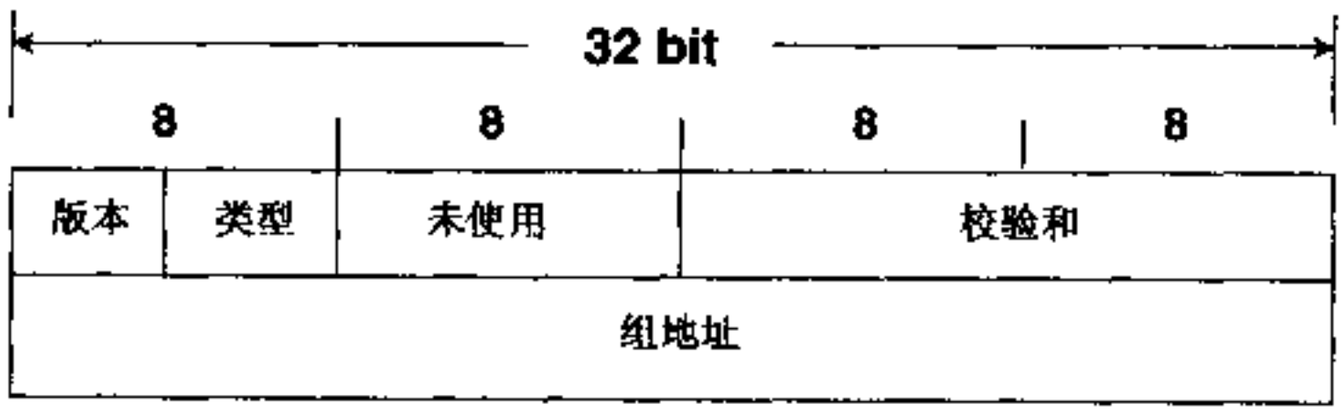


图 5-12　IGMPv1 消息格式

IGMPv1 与 IGMPv2 格式的不同只在于：

• 第一个 8 位组分成了 4bit 的版本域和 4bit 的类型域；

• 第二个 8 位组在版本 2 中为最大响应时间，而在这里却没有使用，这个值设为 0x00。

另一个不同之处为在 IGMPv1 的 IP 包头中不设置路由器告警选项。

IGMPv1 只定义了两种类型的消息：

• Host Membership Query(类型 1)

• Host Membership Report(类型 2)

版本值总是设为 1，因此在 Host Membership Query 消息中，版本和类型两个参数的组合为 0x11，这与 IGMPv2 中的 Membership Query 中的 8bit 类型参数是一样的。Host Membership Report 消息的版本和类型组成的值为 0x12，而 IGMPv2 的 Membership Report 消息的类型为 0x16。

5.2.3　Cisco 组员资格协议(CGMP)

IP 多播设计的基本原则就是流量只发给希望收到流量的目的地。你已经看到 D 类地址和相关的 MAC 地址在数据链路层是如何满足这一目标的，IGMP 使路由器如何决定是否向一个子网转发会话数据。你在随后的章节里将看到，IP 多播路由协议如何把这一原则扩展到网际网络环境中去，并将多播会话的数据只传送给连接在有组员的子网上的路由器。

可是图 5-13 所示的交换网络会怎么样呢？除非把交换式的校园网划分为多个 VLAN，否则它仍由一个广播域组成。这里，路由器的端口定义了一个第 3 层的子网，任何广播帧均从全部 384 个端口送出。大的办公建筑和校园中有很多这样的网络。以太网交换机具有高性能、高端口密度的透明的桥接等特性，它可以通过获知哪个 MAC 地址与哪个端口相关联来限制

单播的流量。它们可以根据这些信息过滤或前转帧。不过广播流量会前转给交换机上的每个端口。一个大型网络，如图 5-13 所示，一般会分为若干个虚 LAN(VLAN)，以控制广播流量的范围。但是很少有这么大的交换式网络呈扁平结构——一个单独的大子网或广播域。

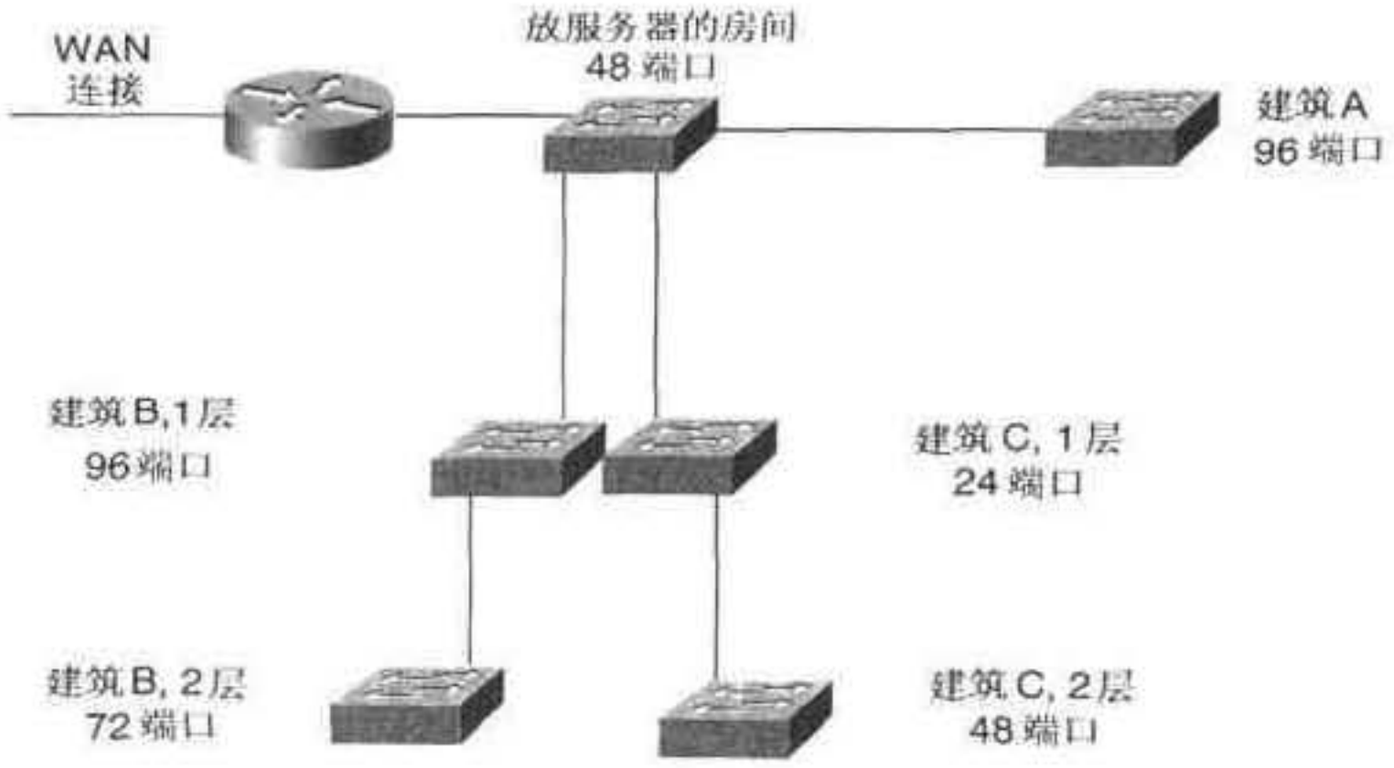


图 5-13　一个大规模的广播域

正像广播帧前转到广播域的各个端口一样，承载 IP 多播包的帧也是这样。毕竟广播域只不过是全体主机都是组员的一个多播组。图 5-14 表示出了这个问题，有 3 个组的成员连在一个 24 端口的交换机上。图中，3 个组员中的一个发送 IGMP Membership Report 消息，加入到多播组 A 中(a)。当路由器前转多播会话时，交换机把帧复制到除了源端口之外的所有的端口上(b)。一个 IGMP Membership Report 消息被发到这个路由器后，路由器开始将相应的多播会话数据前转到子网里。因为 IGMP 是第三层协议，所以以太网交换机没法简单地判断组员在哪些端口上，结果导致多播的流量向另外 23 个端口转发(除了源端口)。

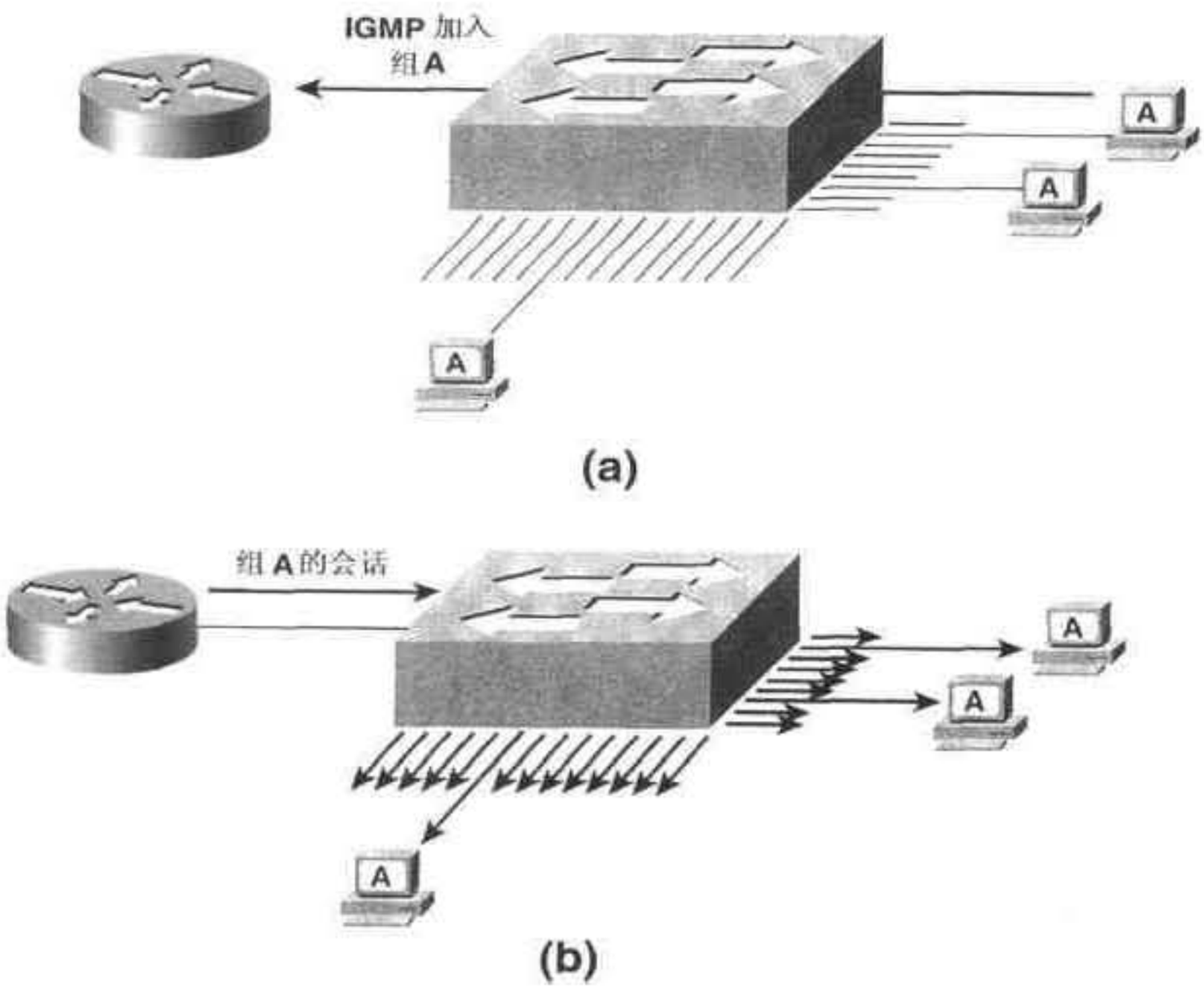


图 5-14　交换机不能分辨组员在哪个端口上

显而易见，交换机将多播数据只从组员连接的那个端口前转是更好的方式。如果能实现的话，不仅能更高效地进行交换，而且还能顺利实现承载多播会话的 LAN。比如，电视会议的多播流占用大约 1Mbit/s 的带宽，MPEG II 的视频流可能占用 4Mbit/s 的带宽。如果这些会话被限制在组员的端口上，那么网络和主机的资源都能节省。

Cisco 组员资格协议(CGMP)正是为把多播会话的数据转发到有组员的端口上而设计的。在分析 CGMP 协议前，下一节将先简单介绍一下其他调节交换多播流量的解决方案。

1．交换网络上多播控制的可选方法

除了 CGMP 外，在交换环境中限制多播流量的方法还有 3 种，Cisco 的 Catalyst 软件全部都支持。

- 手工配置的交换式多播树
- GMRP
- IGMP 监听

因为这3种方法都与选路没有直接的关系，本节只是一个概述。Cisco 的 CCO 上对 Catalyst Switch Software Documentation 有详细和完整的配置指导。

手工配置的交换式多播树就是你在交换机的桥接表里加一个静态的条目，Cisco 的 Catalyst 交换机称这个表为内容可寻址存储器(CAM)表。假设图 5-13 中的组员在交换机端口 2/3，2/4 和 2/19 上，而路由器在端口 1/1 上，组地址为 239.0.5.10，这个 IP 地址将造成多播的 MAC 地址为 0100.5E00.050A，把这个信息手工输入 Catalyst CAM 表中的命令为

set cam permanent 01-00-5e-00-05-01 2/3-4, 2/19

set multicast router 1/1

上述命令在 CAM 表中加入一个条目，并写入交换机 NVRAM 中；这个条目可以用 **clear cam** 和 **clear config** 命令清除。另外，可以用关键字 **static**，而不是 **permanent**。这样的话，这个条目将不会写入 NVRAM，在重启动时就删除了。

第 2 条命令是可选的，它告诉交换机多播路由器在哪个端口上，以进一步在交换机内限制多播流量的范围。

采用人工配置有几个局限，其中两个最明显的就是非动态，不能扩展。如果另一个组员在不同的端口上加入组、组员退出或交换机上加上不同的组，那么这些信息就必须手工配置。除了小规模、固定的组之外，手工配置并不实用。

另一个局限就是手工配置不能跨跃 VLAN 的边界。比如，如果组 239.0.5.10 在 VLAN1，而这个交换机上同时有 VLAN2，那么 VLAN2 上就不能有 239.0.5.10 组的成员，因为它们必须在同一 VLAN 中。

另一种技术就是使用 GARP 多播注册协议(GMRP)，一个 IEEE 802.1p 中规定的开放协议，它让 MAC 层的多播组地址动态地在交换机上注册和取消。GMRP 在交换机上用命令 **set gmrp enable** 启动；路由器上不用作任何配置。正如 IEEE 802.1p 标准所建议的，GMRP 是个严格的第 2 层协议。

第 3 种技术就是 IGMP 监听，在 Catalyst 交换机上用命令 **set igmp enable** 把这个功能激活。采用这种技术后，交换机软件通过检查 IGMP 消息，就能知道多播路由器和组员的位置。不同于 CGMP 的是，IGMP 监听有多厂家的支持。在有多厂家产品的交换网络里，

它是一个更好的选择；不过，检测 IGMP 消息意味着所有的 IP 包都要进行检查。当这方法用软件方式来实施时，会严重降低交换机的性能。你只有在多播环境中的所有交换机用硬件方式(特别是采用专用 ASIC 芯片，以线速检查 IP 包)完成这一功能的情况下，才能用 IGMP 监听。比如，有 NetFlow Feature Card II(NFFC II)的 Cisco Catalyst 交换机就支持这一功能。

2. CGMP 的操作

虽然 Cisco 的路由器和交换机都必须配置运行 CGMP，可是只有路由器才能产生 CGMP 包。交换机上的 CGMP 的处理只是读取这些包。有两种 CGMP 包：

- Join 包由路由器发出，告诉交换机向多播组中加入一个或多个组员。
- Leave 包由路由器发出，告诉交换机从多播组中删除一个或多个组员，或删除整个组。

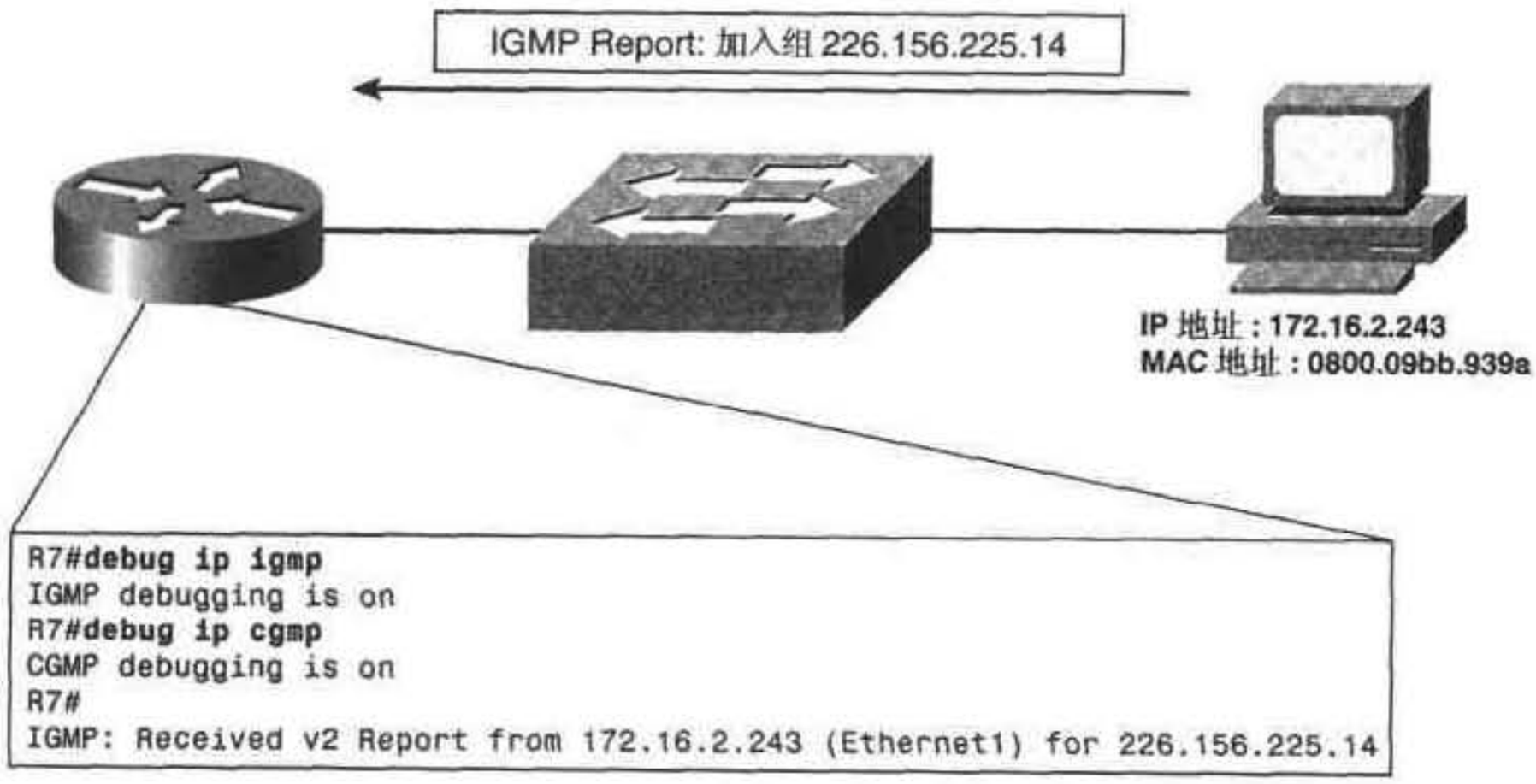
这两种类型的包有相同的格式，包的目的地址总是为保留的 MAC 地址 0100.0cdd.dddd。激活 CGMP 功能的交换机会侦听这个地址。

这两种包里的基本信息是一个或多个 MAC 地址对：

- 组目的地址(Group Destination Address GDA)
- 单播源地址(Unicast Source Address USA)

当 CGMP 的路由器启动后，它向交换机发送一个 CGMP 的 Join 包，这个包中 GDA 设为 0(0000.0000.0000)，USA 设为自己的 MAC 地址。于是 CGMP-speaking 交换机知道多播路由器连在它收到这个包的端口上。路由器每 60s 发送一个包来保持存活。

当一台主机要加入一个组时，它发送一个 IGMP Membership Report 消息，如图 5-15(a)所示。交换机收到 IGMP Membership Report 消息后，用 CGMP Join 消息告诉交换机主机 MAC 与组 MAC 地址信息。遵从 IEEE 802.1 标准的交换机将主机的 MAC 地址输入到它的 CAM 表中。



(a)

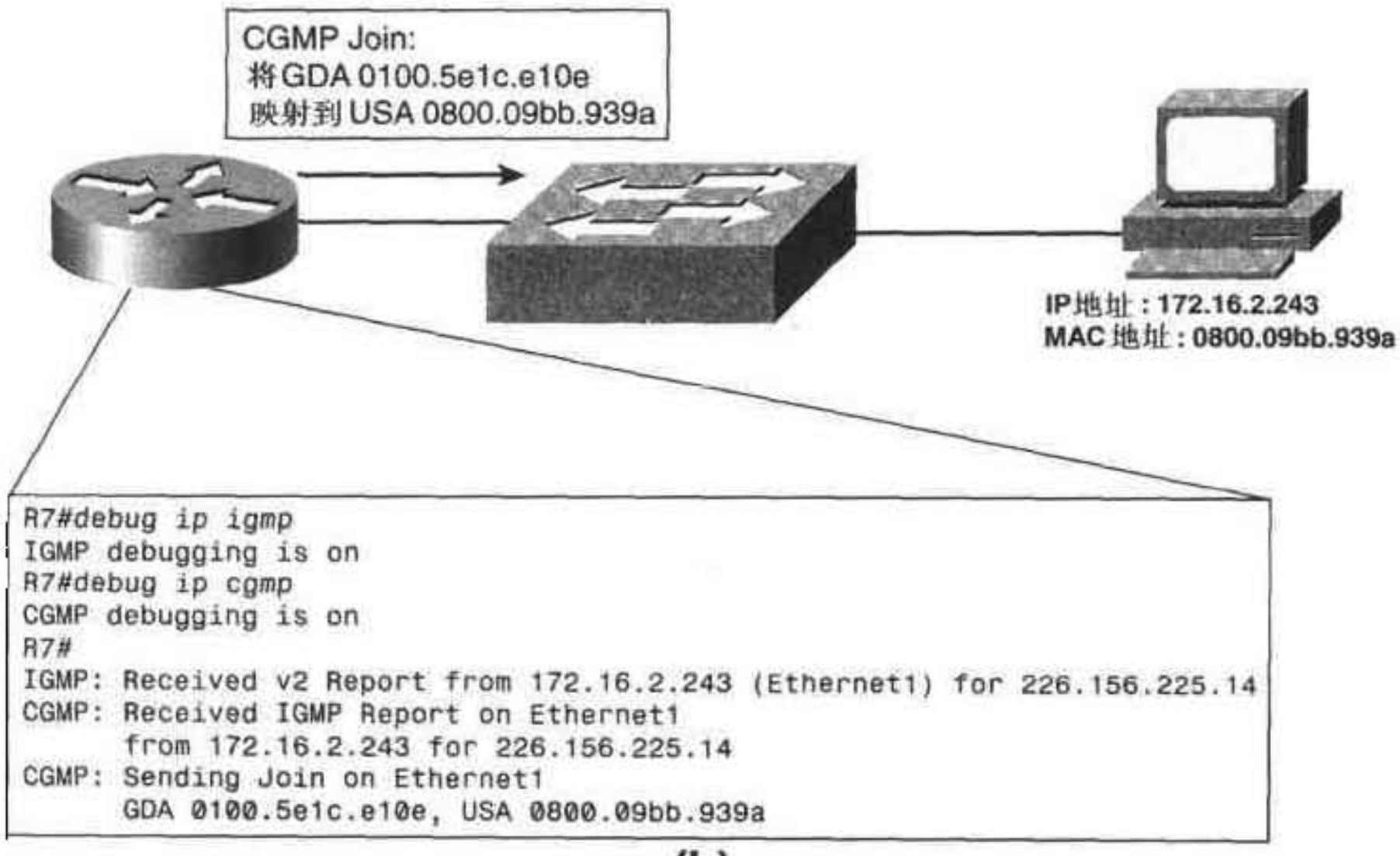


图 5-15　当 Cisco 路由器在 CGMP 接口上收到 IGMP Membership Report 消息(a)，它发送 CGMP Join 包通知交换机进行主机 MAC 地址与组 MAC 地址的映射(b)

注：　Catalyst 的 CAM 表是一个记录侦听到的 MAC 地址和侦听到这个地址的端口的桥接表。

当路由器收到 IGMP Membership Report 消息后，它向交换机发送一个 CGMP Join 包，这个包中 GDA 设为这个组的 MAC 地址，而 USA 设为主机的 MAC 的地址，如图 5-15(b)所示。交换机这时已经知道主机在哪个端口上，它把这个端口加入组中。当路由器向这个组的 MAC 地址发出帧，交换机将向这个组的所有相关端口(除了路由器端口)转发一个该帧的拷贝。

只要还有组员在这个交换式网络里，路由器就每隔 60s 发送一个 IGMP 查询消息，交换机再把这个查询前转到组员处。交换机把应答查询的 IGMP 报告转发给路由器。

当主机发送一个 IGMPv2 的 Leave Group 消息，这个消息前转给路由器，如图 5-16(a)所示。路由器发送两个 Group-Specific Query 消息，交换机把这个消息前转到所有组员的端口上。如果另一组员响应 Group-Specific Query 消息，那么路由器向交换机发送一个 CGMP Leave 包，这个包中的 GDA 设为这个组的 MAC 地址，USA 设为退出组员的 MAC 地址，如图 5-16(b)所示，这个包告诉交换机删除退出组员所在的那个端口。如果没有组员应答 Group-Specific Query 消息，那么路由器认为这一网段中没有组员，它会向交换机发送一个 CGMP Leave 包，这个包中的 GDA 设为这个组的 MAC 地址，USA 设为 0，如图 5-16(c)所示，这个包告诉交换机从 CAM 表中删除这个组。

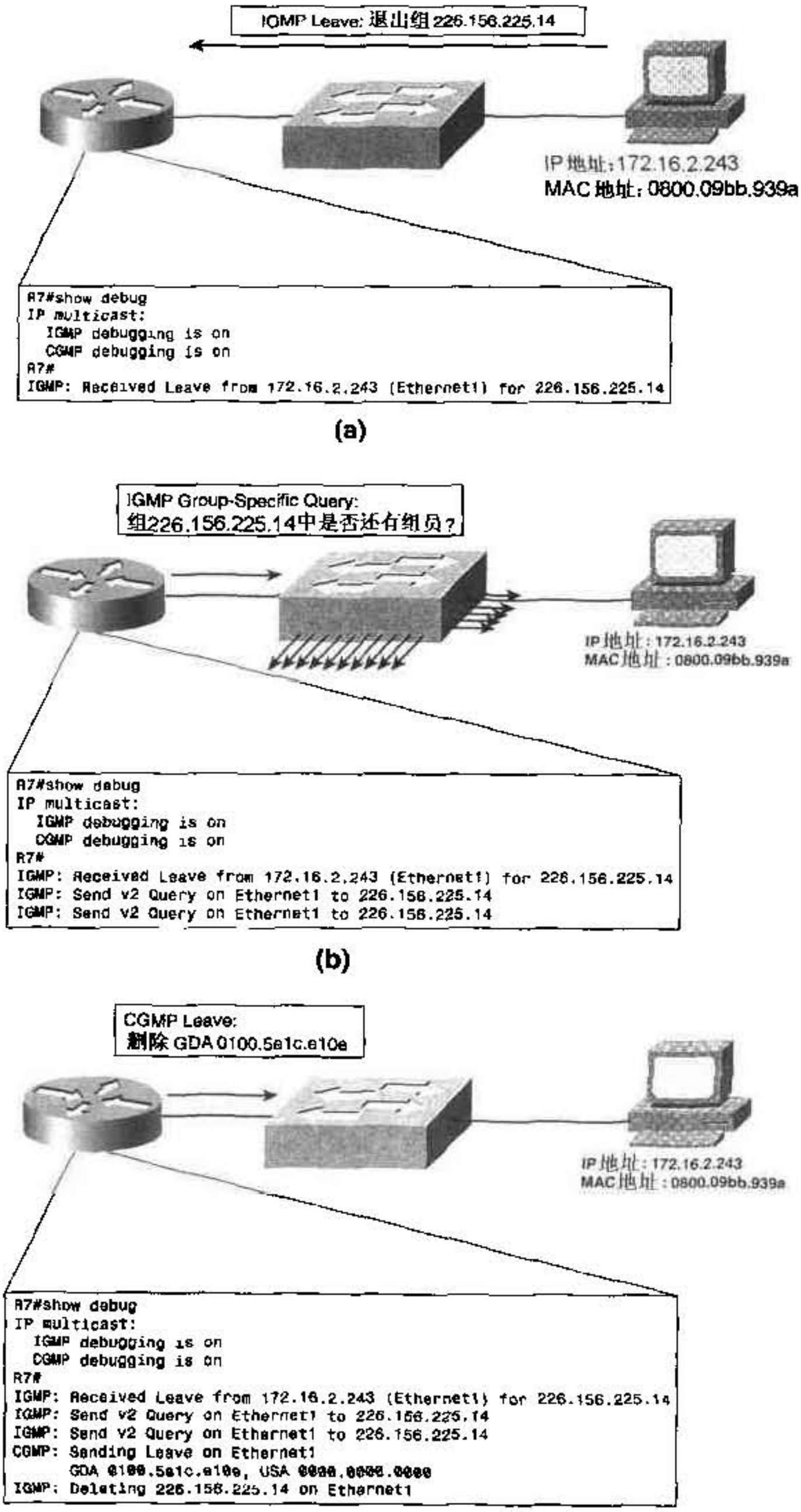


图 5-16　交换机收到 IGMP Leave 消息后，用 CGMP Leave 消息通知交换机删除一个组员或整个组

(a)IGMP Leave：离开组 226.156.225.14

(b)IGMP Group-Specific Query：组 226.156.225.14 中是否还有其他成员

(c)CGMP Leave：删除 GDA 0100.5e1c.e10e

表 5-2 总结了 GDA 和 USA 在 CMGP 包中的各种可能的值和其所代表的含义，只有最后两个 Leave 包没有讨论过。Leave 包中的 GDA 设为 0，USA 设为路由器的 MAC 地址，表示收到这个消息的交换机将删除 CAM 表中所有与路由器端口相关的组和端口。这个消息将在路由器上连接交换机的端口关闭 CGMP 功能时发出。Leave 包中的 GDA 和 USA 都设为 0，告诉收到包的交换机从 CAM 表中删除所有的组和相关的端口。这个消息将在路由器设定 **clear ip cgmp** 命令时发出。

3、CGMP 包的格式

承载 CGMP 包的帧的源 MAC 地址就是发起路由器的 MAC 地址，目的地址就是保留的 MAC 地址 0100.0cdd.dddd。只有路由器才能产生 CGMP 包。在帧中，这个包由 SNAP 头封装。SNAP 头的 OUI 部分设为 0x00000c，类型设为 0x2001。

表 5-2　　CGMP 包

类　型	GDA	USA	功　能
Join	0	路由器的 MAC	确定端口为一多播路由器的端口
Join	组的 MAC	组员的 MAC	确定多播组，并把成员的端口加入到组中
Leave	组的 MAC	组员的 MAC	从一个组中删除成员的端口
Leave	组的 MAC	0	从 CAM 中删除组
Leave	0	路由器的 MAC	从 CAM 中删除所有与路由器端口相关的组和端口
Leave	0	0	从所有交换机上删除所有的组

图 5-17 中显示了 CGMP 包的格式。

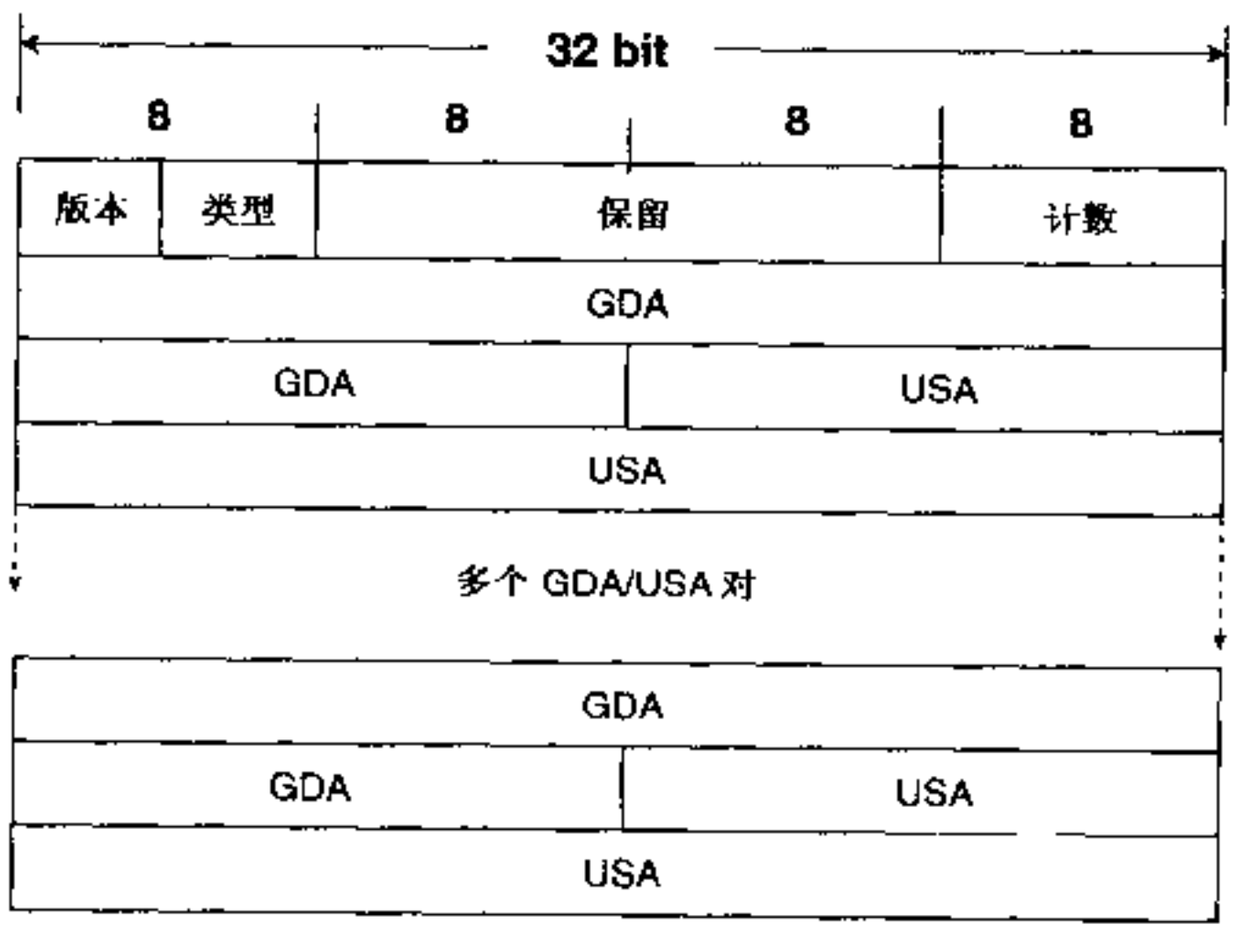


图 5-17　CGMP 包格式

CGMP 包中各个参数的定义如下：

- 版本总是设为 0x1，表示版本 1。
- 类型用于定义这个包为 Join(0x0)，还是 Leave(0x1)。
- 保留总是设为 0(0x0000)。
- 计数用于定义这个包中有多少 GDA/USA 对。

• GDA 为组目的地址。当这个参数为非 0 时，它定义了一个多播组 MAC 地址。当这个参数为 0(0000.0000.0000)时，它表示所有可能的组。

• USA 为单播源地址。当这个参数为非 0 时，它表示发起路由器的 MAC 地址或组员的 MAC 地址。当其为 0 时，它表示所有的组员和发起路由器。

5.3　多播路由的问题

当前，有 5 种不同发展阶段和应用范围的 IP 多播路由协议：

- 距离向量多播路由协议(Distance Vector Multicasting Routing Protocol DVMRP)
- 多播 OSPF(MOSPF)
- 基于核心的树(Core-Based Trees CBT)
- 与协议无关的多播，密集模式(PIM-DM)
- 与协议无关的多播，稀疏模式(PIM-SM)

每一种协议都会在下面各节中分别讲述，包括它们各自的优点和缺点。虽然 CISCO IOS 软件并不支持所有 5 种协议，但对每种协议的学习还是会帮助你了解其被支持与不被支持的道理的，Cisco IOS 支持这 5 种协议中的 PIM-DM 与 PIM-SM。PIM 对于 DVMRP 的支持使 PIM 网络可以同 DVMRP 网互通。这 5 种多播协议是多播的 IGP 协议。跨跃 AS 边界的多播将在第 7 章“大规模 IP 多播路由”中讨论。

这 5 种 IP 多播路由协议相互间有很大的差别，但同单播路由协议一样，它们也有很多共同的特点。这一节将对这些多播协议设计的一般性问题进行讨论。

5.3.1　多播的前转

同其他的路由协议一样，多播路由器的两个基本功能是路由的发现和包的前转。这一节主要讲述多播前转的特殊要求，下一节为多播路由发现的要求。

单播包前转就是把一个包前转给一个既定的目的地址。除非规定了某些策略，否则单播路由器不会关心这个包的源地址。收到这个包后，路由器检查这个包的目的 IP 地址，进行最大匹配路由的查询，然后这个包被前转到能够到达这个目的地的一个网络接口上。

多播不是将包前转到一个目的，而是将包由源前转。这个差别可能看上去不是很重要，不过这是对多播前转的一种根本的修正。多播包由一个源产生，不过是发向一组目的的。在一个特定的路由器上，一个包的多个备份可能从多个网络接口上发出。

如果有环路的存在，那么一个或多个包会返回到其输入的端口，而且这个包也会经复制发到其他输出端口上。这一结果可能导致多播风暴，这个包不断在路由器与交换机间环回并复制，直到 TTL 减为 0。由于这个现象是个复制的过程，因此会比单播环路严重得多。因此，所有的多播路由器必须知道多播包的源，也必须把包从源地址转发走。

两个有用的常用术语分别为上游和下游。多播包应从源到目的，向下游传送，而不能向上游，从目的到源。为了保证这个行为，每一个多播路由器都有一张多播前转表，这张表里记录着(源，组)，或简写为(S, G)地址对。来自一个特定的源地址，发向一个特定的组的包总是在上游的接口处收到，再转发到一个或多个下游的接口上。上游的接口应比任何下游接口

更接近源，如图 5-18 所示。如果一台路由器从不是在源的上游接口上收到一个多播包，那么它会自行把这个包丢弃。

当然，路由器需要有机制来判断一个(S,G)对的上游接口和下游接口。多播协议就是用来完成这个工作的。

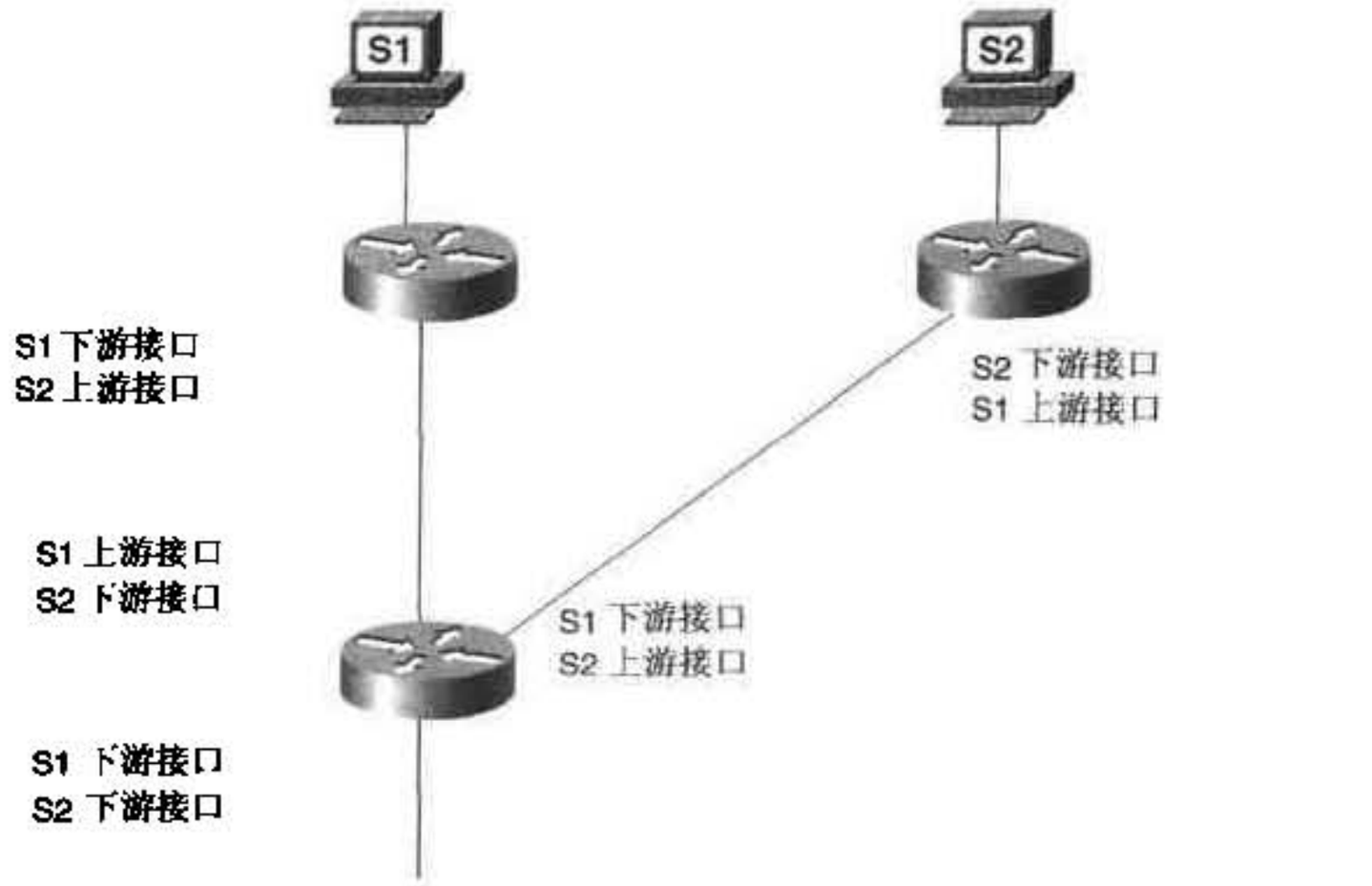


图 5-18　通过区分与每一多播源相关的上游接口与下游接口，路由器避免了多播路由环

5.3.2　多播路由

单播路由协议的功能是用来找出到一个特定目的的最短路径。通过邻接的路由器发出的宣告(距离向量)，或通过拓扑数据库算出最短路径的树(链路状态)，即可以完成这个判断。这两种情况都会在路由表中产生一个条目，指示能把包转发出去的接口和下一跳路由器。从单播路由器的角度看，这里所说的接口是下游接口，是在去目的的通路上——也就是离目的更近的端口。

与单播相反，多播路由协议的功能是判断上游接口——离源更近的接口。因为多播路由协议更关心到源的最短路径，这一前转的过程称为反向路径前转。

多播路由协议判断到源最短路径的最简单方法就是查询单播前转表。不过，正如前一节所指出的，多播包前转是基于独立的多播前转表的。因为路由器不仅要记录对一特定(S，G)对的源上游接口，而且还要记录与组相关的下游接口。

最简单的前转包的方法是向除了上游端口外的所有端口发数据，这种被称为“反向路径广播”(Reverse Path Broadcast RPB)的方法有显而易见的缺点。就像其名称中隐含的一样，包向网际网络中的所有子网发送，那么组员可能在子网的一个子集中——可能为很小的一个子集。将一个多播的包扩散到各个子网中不仅违背了多播的目标：把包只送给对其感兴趣的接收者，而且也违背了选路的原则。

一个较为改进的处理过程是“修剪的反向路径广播”(Trancated Reverse Path Broadcast TRPB)。当一台路由器通过 IGMP 发现与它相连的一个子网上有一个没有组员，而且子网中也没有下一跳的路由器时，路由器就停止向这个子网发送多播流量。不能通过的子网，用树形结构的术语，称为叶网络。虽然 TRPB 保护了叶网络中的资源，可实际上并没有比 RPB 有多大的改进。不论是否需要，带宽非常宝贵的路由器间的链路仍承载着多播的流量。

所以多播路由协议的第二项功能就是判断与(S,G)对相关联的实际下游网络接口。当所有

的路由器都能判断出它们对某一特定源和组的上/下游接口时，一个多播树就能够被建立起来(参见图 5-19)。这个树的根是与源直连的路由器，树枝伸向所有有组员的子网，而不会有树枝伸到“空”的子网里——即没有相关组的组员的子网。将包只在能到达组员的接口上进行前转称为反向路径多播(Reverse Path Multicast RPM)。

多播树只会在多播会话期间存活。因为组员可以在存活期内加入和退出组，树的结构是动态的，多播路由协议的第三个功能就是对树进行管理，组员加入时接上树枝，组员退出组后剪去树枝。下面三节都是围绕这 3 个功能进行讨论的。

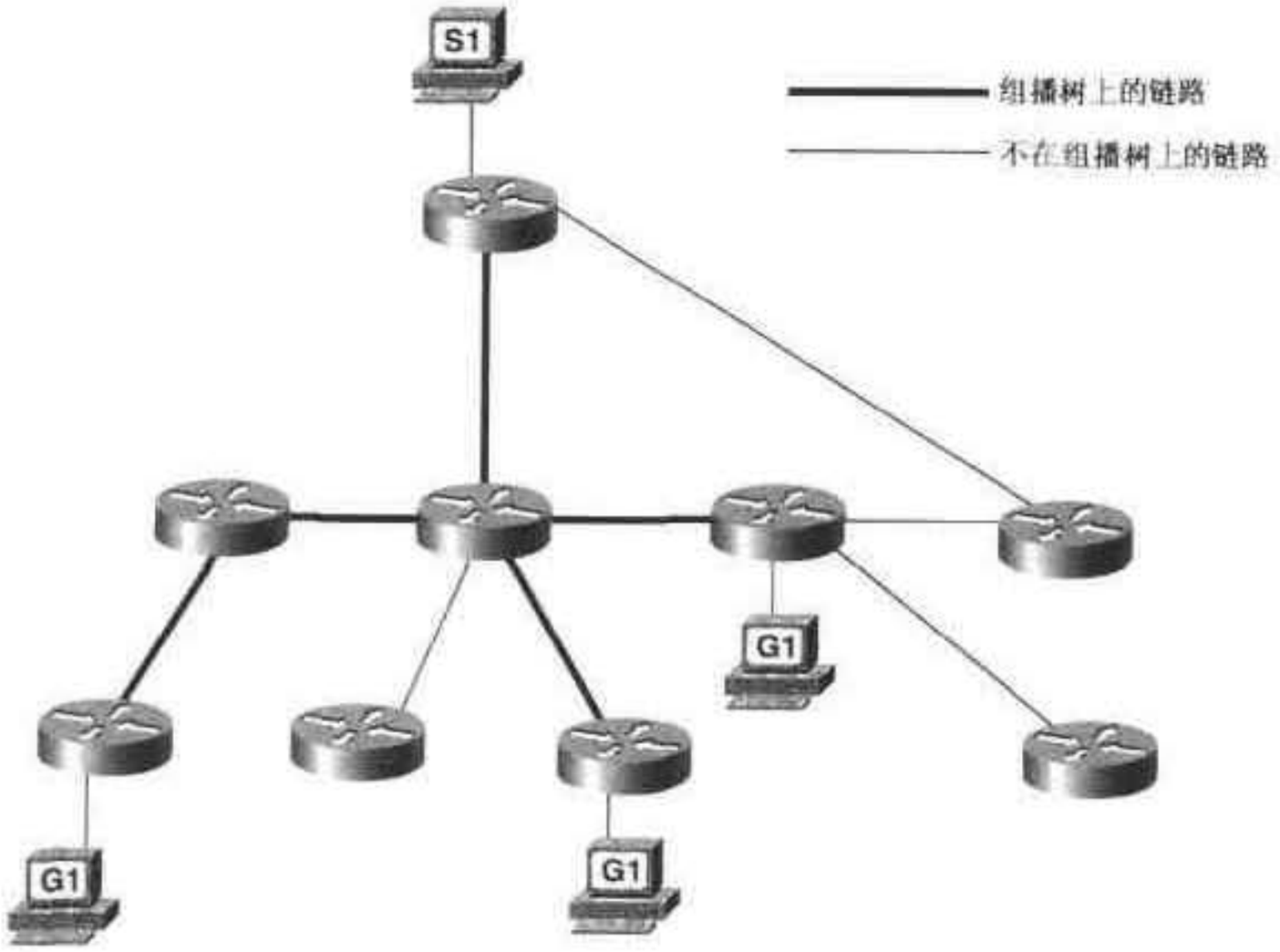


图 5-19　从多播源到所有组员所在子网的路径构成一个多播树

5.3.3　稀疏与密集拓扑的比较

密集的拓扑结构是一种在网际网络中多播组的成员相对主机总数来说比较多的情况，而稀疏的的拓扑结构则是多播组的成员相对主机总数来说比较少的情况。稀疏并不意味着有很少的主机，在稀疏拓扑结构中，可能存在在 100 000 个主机中有 2 000 个组员的组。

稀疏和密集之间没有具体的数字来划分，不过稳妥地说，密集模式一般用在交换的 LAN 和校园网环境中，而稀疏模式一般用在 WAN 中。重要的一点是，多播路由协议是为了在一种或多种网络结构中都能正常工作的，可分为密集模式和稀疏模式。表 5-3 中显示了 5 种多播路由协议分别属于哪一类。

表 5-3　　密集模式和稀疏模式多播路由协议

协　议	密集模式	稀疏模式
DVMRP	X	
MOSPF	X	
PIM-DM	X	
PIM-SM		X
CBT		X

5.3.4　隐式加入与显式加入的比较

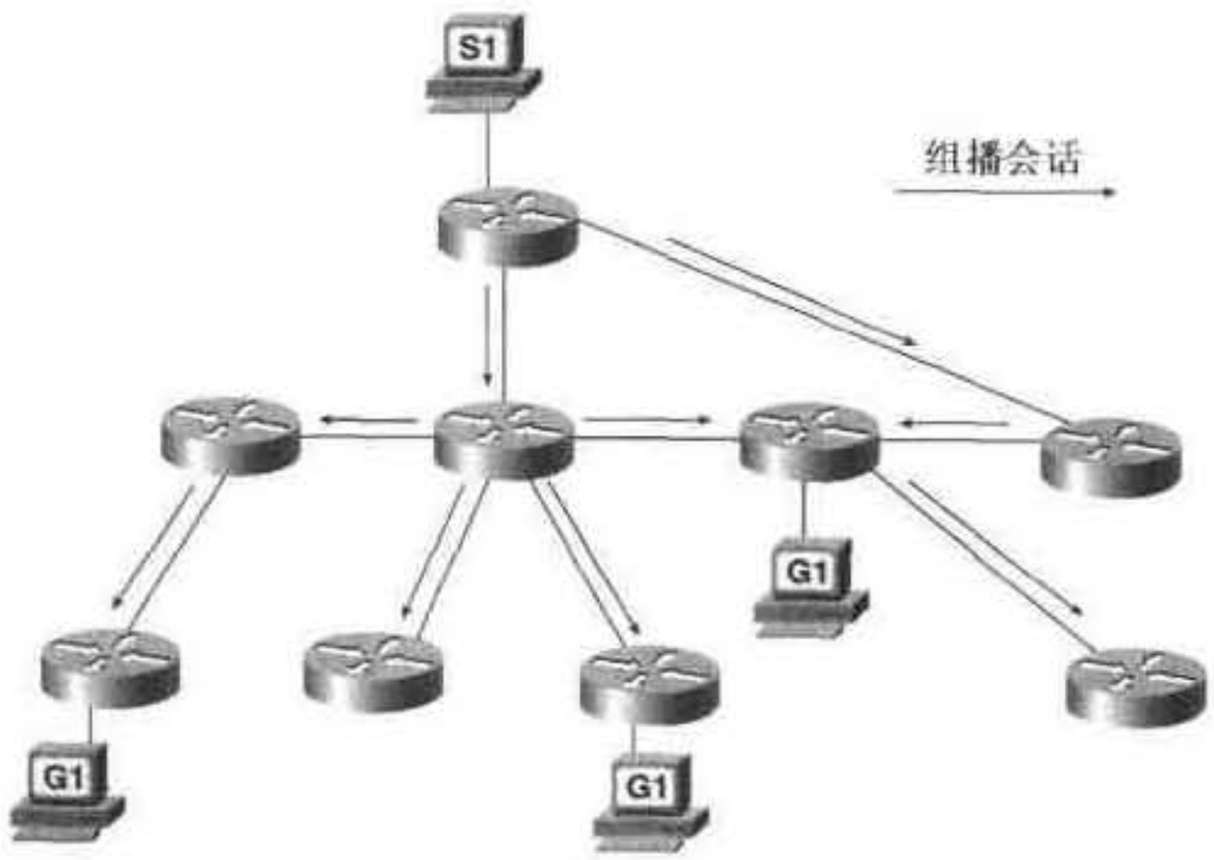
如前面所看到的，组员可以在多播会话存活的时段内随时加入和退出，所以多播树可以在组员加入时接入一个树枝，组员退出时剪除一个树枝，如此动态地进行改变。

多播路由协议可以通过隐式加入与显式加入完成这项任务，隐式加入为发送者发起的，而显式加入为接收者发起的。

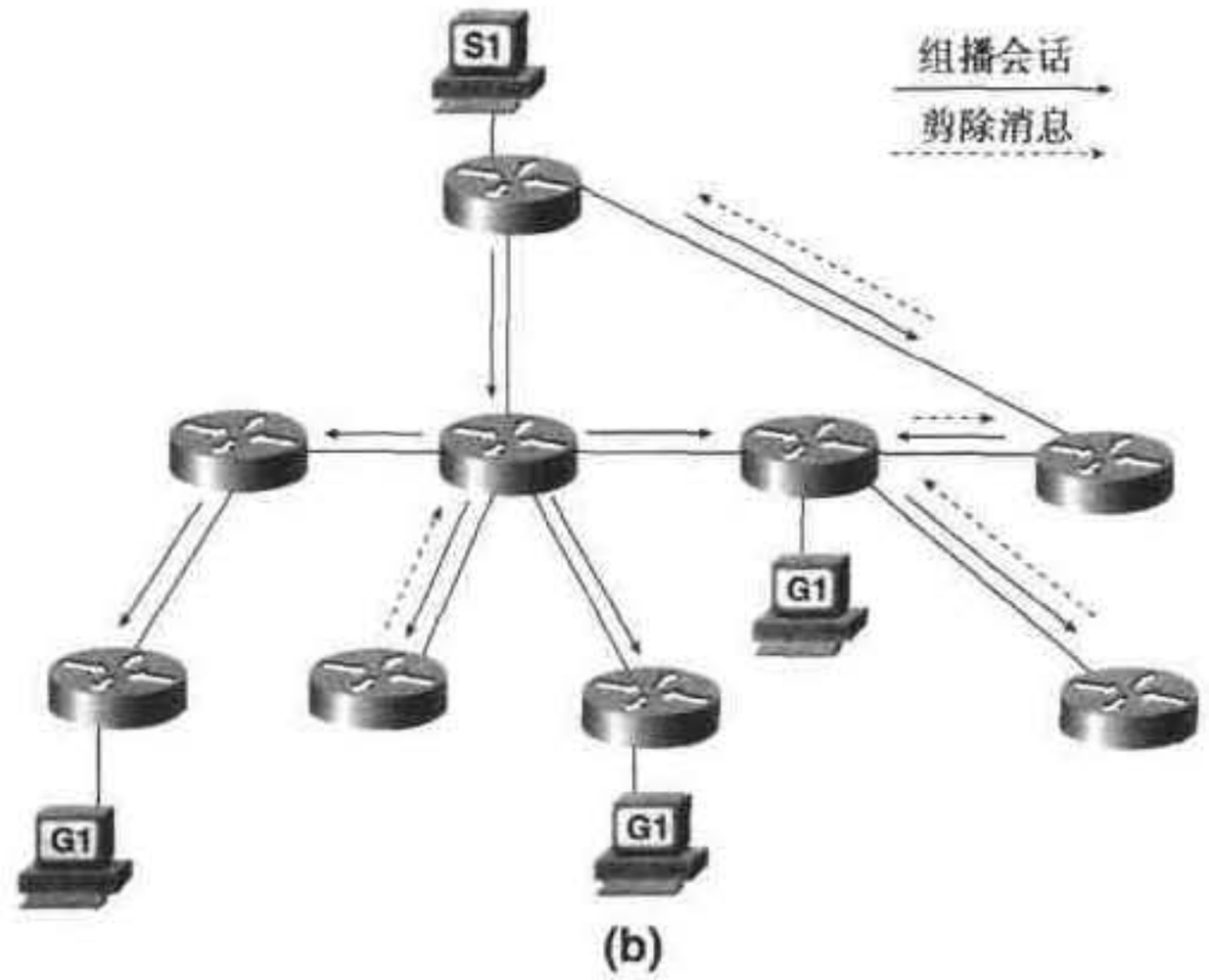
通过隐式加入来维护其多播树的多播路由协议一般称为“广播与剪除”或“扩散与剪除”协议。当发送者发起一个会话时，网际网络中的每一个路由器均用反向路径广播把这个包从除了上游端口外的其他端口上前转出去。所以，多播会话在开始时会传到网际网络中的每一台路由器上。当路由器收到一个多播流量时，它用 IGMP 来判断它直连的子网中是否有组员。如果没有，并且也没有必须接收这个包的下游路由器，那么路由器就向上游路由器发送一个反向毒化的消息，这个消息称为剪除消息。上游的路由器于是停止向这个剪除的路由器前转流量。如果上游的路由器的子网中也没有组员，那么这台路由器也向上游发送剪除消息，而且所有下游路由器都从树中剪除。这个结果最终导致多播树把所有没有组员的路由器连接的树枝剪除。图 5-20 中画出了“广播和剪除”技术，图中，“广播与剪除”协议首先用 RPB 来前转一个多播会话到所有网际网络的子网中(a)，将没有连接组员的路由器从树中剪除(b)，这样将导致多播树只达有组员的路由器(c)。

每一个在网际网络中的路由器，对于每一个(S,G)对，前转表都维护着它所有下游接口的状态。这个状态不是前转，就是剪除。剪除状态有一个与之相关的计时器，如果计时器超时，会话的流量会再发送到与这个端口相邻的路由器上。每一个相邻的路由器再一次检查它子网内的组员，并把这个会话流量继续转发到与它相邻的下游路由器上。如是发现了新的组员，则会继续接收流量，否则将向上游发送一个新的剪除消息。

广播和剪除技术对于密集拓扑结构更适合。当许多或大多数树枝被剪除时，起始的扩散过程、剪除状态超时后周期性的重扩散过程和维护剪除状态均会浪费网络资源。而且，维护剪除状态要求不加入多播树的路由器记住自己不是树中的树枝也极不合逻辑。



(a)



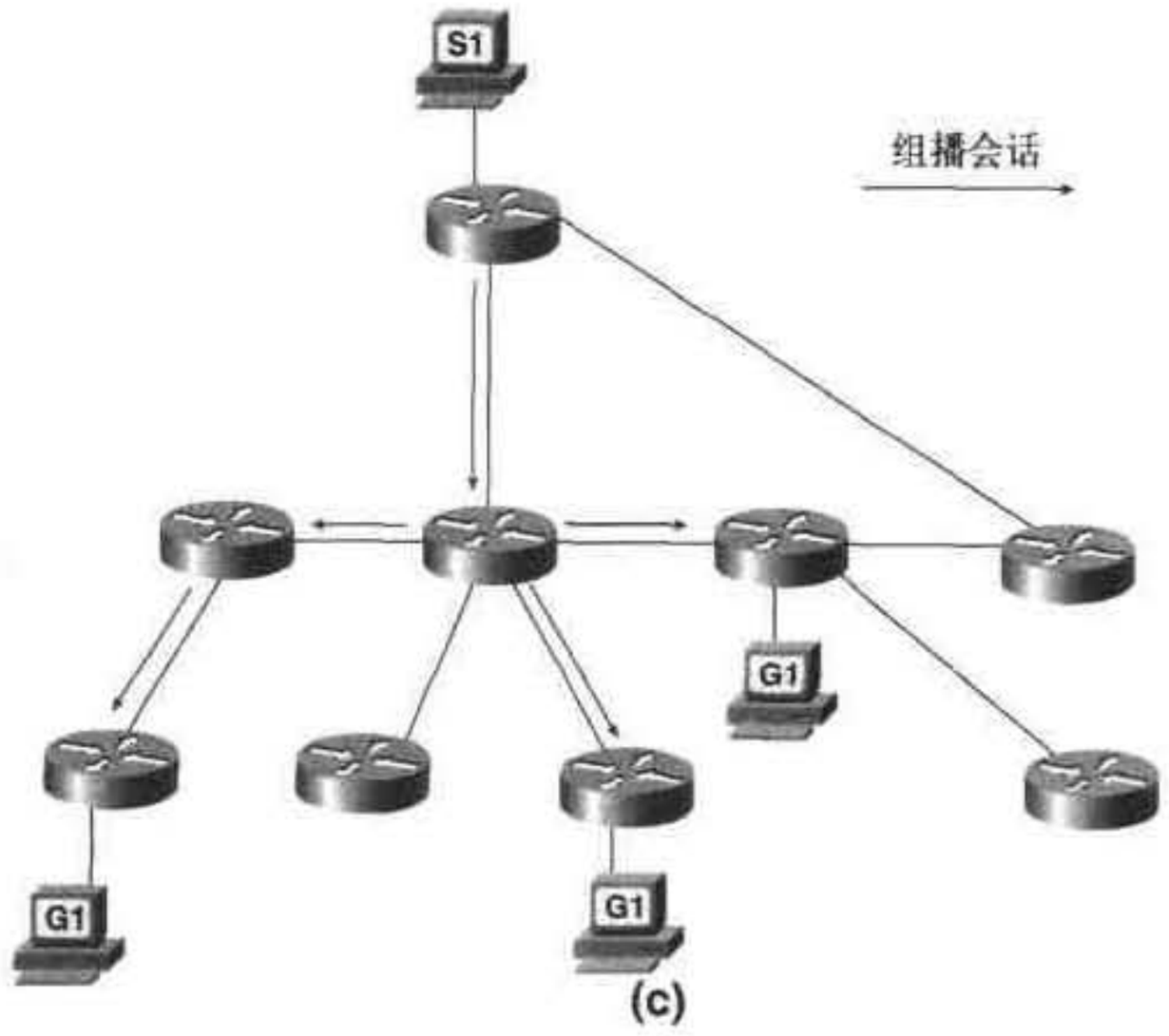


图 5-20　“广播与剪除”协议

对稀疏拓扑结构来说，更好的一种技术就是显式加入，让直连着组员的路由器发起加入过程。当一个组员通过 IGMP 告诉路由器它要加入一个组时，路由器向上游源的方向发送一个消息表示要加入。与剪除消息不同的是，可以认为这是个接入消息；发送这个消息的路由器则把自己接入到多播树中。如果路由器上所有的组员退出，且路由器没有这个组的相邻下游路由器，则路由器将把自己从树中剪除。

因为流量不会转到没有明确请求这个流量的路由器上，所以网络的资源会受到保护。由于不相关的路由器不保留剪除状态，因此也能节省总体的内存的使用。所以说，显示加入对稀疏拓扑更适合。当然也可以说，显式加入不论对稀疏还是密集都是更好的。表 5-4 中列出了 5 种多播路由协议中哪一个是隐式加入，哪一个是显式加入。

表 5-4　　隐式加入和显式加入协议

协　　议	隐 式 加 入	显 示 加 入
DVMRP	X	
MOSPF		X
PIM-DM	X	
PIM-SM		X
CBT		X

5.3.5　基于源的树与共享树的比较

一些多播路由协议为每一个源建立各自的多播树。这些树就是基于源的树，因为它们的根是源节点。在前面章节中列举的多播树都是基于源的树。

你已经知道在多播会话存活期间，当一个组员加入和退出组时，多播树都会发生变化，多播路由协议应负责动态修改多播树来适应这一变化。不过，这个树的某些部分可能不发生变化。图 5-21 中显示了两个多播树在一个相同的网际网络中的情况。注意到虽然这些树有不同的源和成员，但它们的通路上至少有一个相同的路由器。

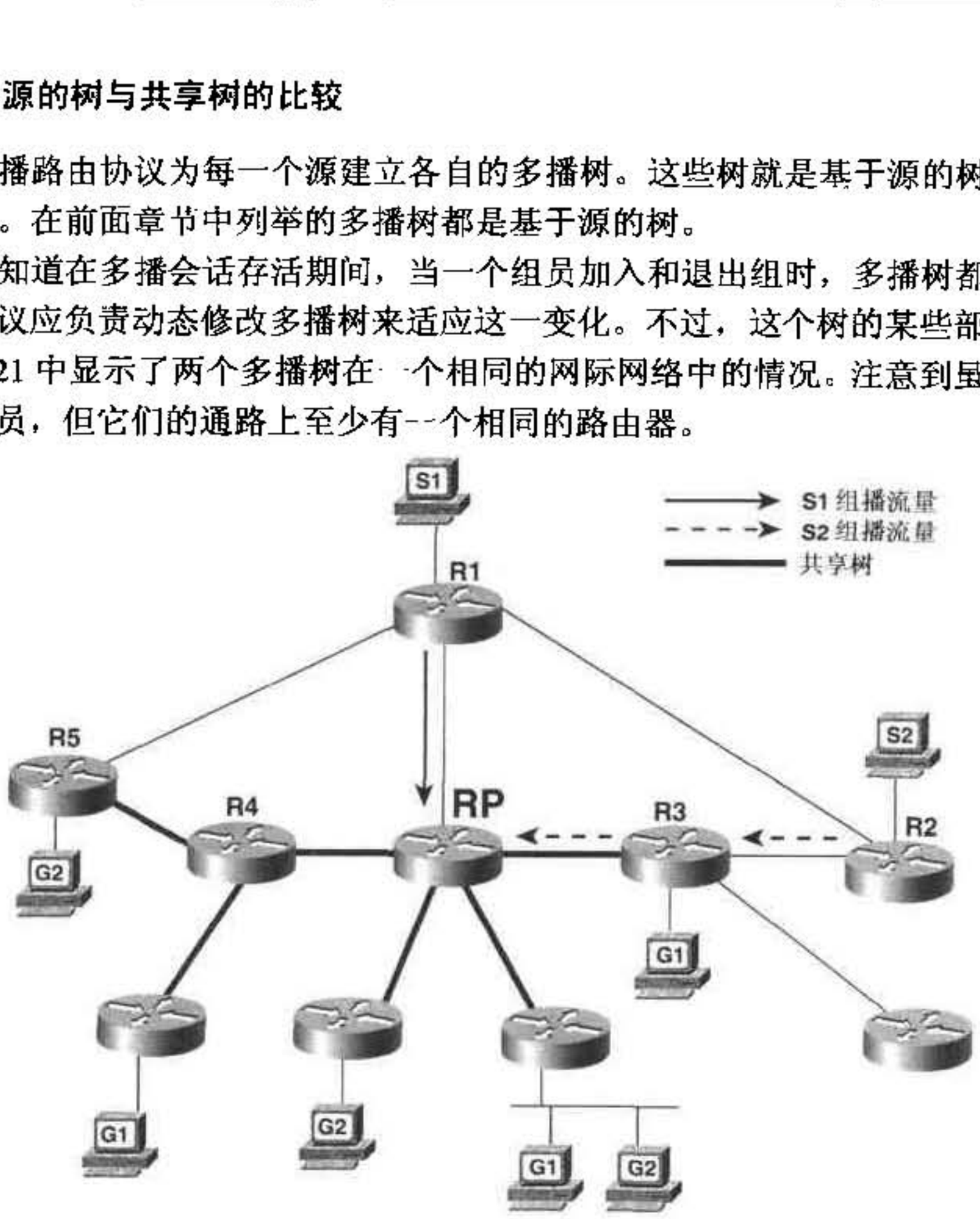


图 5-21　这两个多播树有不同的结构，不过它们均经过同一 RP 路由器

共享树是利用了这些多播树可以在网络中共享一个路由器的事实。与其让每个树根植在不同的源节点上，倒不如让这些树根植在一个指定点(Rendezvous Point RP)，或者称为核心(core)。RP 是事先有意图地设在网际网络中的。当一个源要启动一个多播会话时，它向 RP 进行注册。它可能靠和源直连的路由器来判断到 RP 的最短路由或者由 RP 来找出到每一个源

的最短路径。用显示加入的方式来建立从带有组员的路由器到 RP 的树。与基于源的树采用(S,G)对不同，共享树采用了(*,G)状态。这个状态反映了多播组的树的根是 RP，并且 RP 上游可能有多个源这一事实。更重要的一点是，在基于源的树中，对于不同的源，记录一个(S,G)对。共享树只为一个组记录一个(*,G)对。

(S,G)对的效果可以通过一些简单的计算来说明。假设在一个有一些基于源的树、扩散和剪除的多播域中有 200 个多播组，平均每个组有 30 个源。每个路由器必须为一个组记录要 30 个(S,G)对，或者说记录 30×200＝6 000 个记录。如果有 200 个组中的每个组都有 150 个源，那么记录条目的数目就是 150×200＝30 000 个。

注：　请记住在交互式的多播应用中，许多组员(接收者)同时也是一个源(发送者)。

与之相反，共享树的路由器为每个组只记录一个(*,G)条目。这样的话，如果在一个共享树的多播域中有 200 个组，则 RP 有 200 个(*,G)记录条目，最有意义的是，这个数目不会因为源数目的改变而改变。用另一种方法来阐述这一事实，基于树的多播规模为($S^G \times G^N$)，共享树的多播规模为(G^N)，这里的 G^N 表示多播域中组的数目，S^G 表示每个组中源的数目。在非 RP 的路由器上，各种影响也大大降低，因为它们不用保留不为其转发包的那些组的状态。这些路由器为每一个下游的组记录一个(*,G)条目。

这种可扩展性意味着，共享树更适于在稀疏拓扑中，不过通常这是一种折衷的方法。首先，从源到 RP 的通路可能不是到每个组的每个成员的最佳通路。再审视一下图 5-21，可以注意到组 2 连接在 R5 上，从 S2 到该组成员的最佳通路是 R2-R1-R5，可是源的流量必须先到达 RP，这样经过的通路为 R2-R3-RP-R4-R5。必须仔细选择 RP，以把次优化路由最小化。另一个缺点是在有多个高宽带多播会话的时候，RP 会成为瓶颈。因为次优化路由和 RP 的阻塞、时延在设计得不好的基于共享树的网际网络中会是一个很大的问题。RP 也会引入单点的故障，最终共享树可能很难进行排错的工作。

表 5-5 列出了多播路由协议使用的是基于源的树，还是共享树。与表 5-4 比较，会发现虽然 MOSPF 采用显式加入，可它还采用基于源的树。这并不是一个真正的矛盾，只有采用共享树的协议才必须使用显式加入，因为它没其他办法维护一个无环的树。

表 5-5　　基于源的树与共享树协议

协　　议	基于源的树	共　享　树
DVMRP	X	
MOSPF	X	
PIM-DM	X	
PIM-SM		X
CBT		X

5.3.6　多播的范围

你在前面关于多播路由的描述中已经知道虽然多播路由的确比其他的方法，比如复制的单播和简单的广播扩散占用更少的网络资源，但它仍然在一定的情况下造成很大的浪费。特

别是在稀疏拓扑结构中使用广播和剪除协议时，这种情况尤为突出。在某些情况下，多播的源和所有组员同整个网际网络相比距离很近，在这种情况下，采用一种可以把多播业务量限制在组员所在的网际网络范围内的机制能够节省资源。还有一些情况，出于安全和其他策略的考虑，多播流量的范围在某种程度上必须受限。

当多播流量限制在“岛内”时，流量会被规划在一个范围内。从另一方面来讲，限制多播就是给多播流量加一个边界。

1．TTL 限制

一种建立限制多播流量范围的边界的方法是在输出端口上设置一个特殊的过滤器，用于检查所有多播包的 TTL 值。只有那些经过减一操作的 TTL 值仍然在一个特定门限之上的包才被前转，而其他的多播包将被丢弃。

图 5-22 中显示了一个例子。在这个路由器上，一个 TTL 为 13 的多播包到达接口 E2，路由器将这个 TTL 值减 1 得 12，接口 E0 有一个多播 TTL=0 的门限，这个值为默认值。没有多播包会因为其 TTL 值而被阻塞，因此从 E0 上前转一个多播包。同样，一个包在接口 E1 上前转，因为这个接口上的 TTL 门限是 5，这个门限小于包的 TTL 值。不过，这个包不会从 E3 前转。这个接口的 TTL 门限为 30，意味着只有 TTL＞30 的包才能被前转。

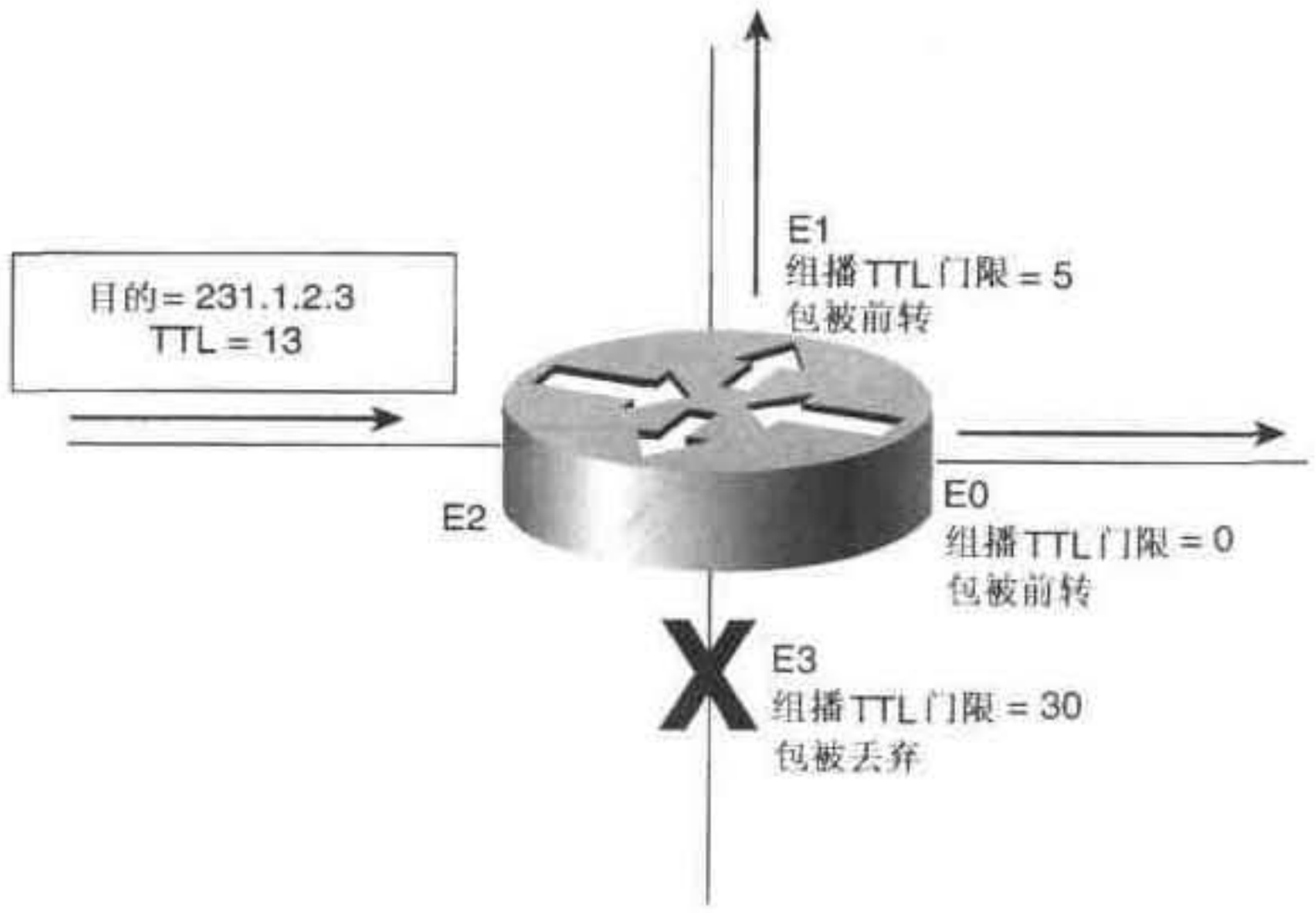


图 5-22　多播包只从 TTL 门限小于输出包的 TTL 值的下游接口上前转

TTL 的限制已经在 MBone 中使用了一段时间，Mbone 是由通过 IP over IP 隧道连接起来的 Internet 上的一些区域性多播网络构成的。表 5-6 列出了 MBone 中限制多播流量的典型 TTL 门限。如果你希望某些流量只在一个场地范围内——比如宽带实时电视——那么你要配置源应用发送的包的 TTL 值不超过 15。

TTL 限制有几个缺点。首先，它不灵活。一个接口的 TTL 门限对所有的多播包都起作用。如果你想让某些多播会话通过这个门限，而其他仍受限，那么必须处理不同会话源的应用。这就引起了另一个问题：必须相信用户能正确设置应用的 TTL 值。如果会话设置了过高的 TTL 值，那么它就可能超出你所设置的边界范围。

表 5-6　　MBone TTL 的门限

TTL 值	限　　制
0	限制在同一台主机上
1	限制在同一子网内
15	限制在同一场地
63	限制在同一区域
127	全球范围
191	全球范围，带宽受限
255	不受限

TTL 限制的另一个问题是它只能用在最简单的拓扑结构里，而很难用于复杂的拓扑。当你的网际网络不断扩展，变得复杂时，预测能限制和传送会话的正确门限是个巨大的挑战。

最后，TTL 限制可以导致“广播和剪除”协议的低效。图 5-23 显示了这个问题。网际网络是一个多播场所，边界路由器上连接其他网际网络的接口的 TTL 门限设为 8。多播源产生的会话中所有 TTL 值设为 8，以保持本地的策略，并把它的流量限制在多播场所内。左边的树枝上没有组员，所以路由器会反向把自己剪除，直到直连到源的路由器上为止。事实上，你可以看到一台路由器已经向上游发送剪除消息。

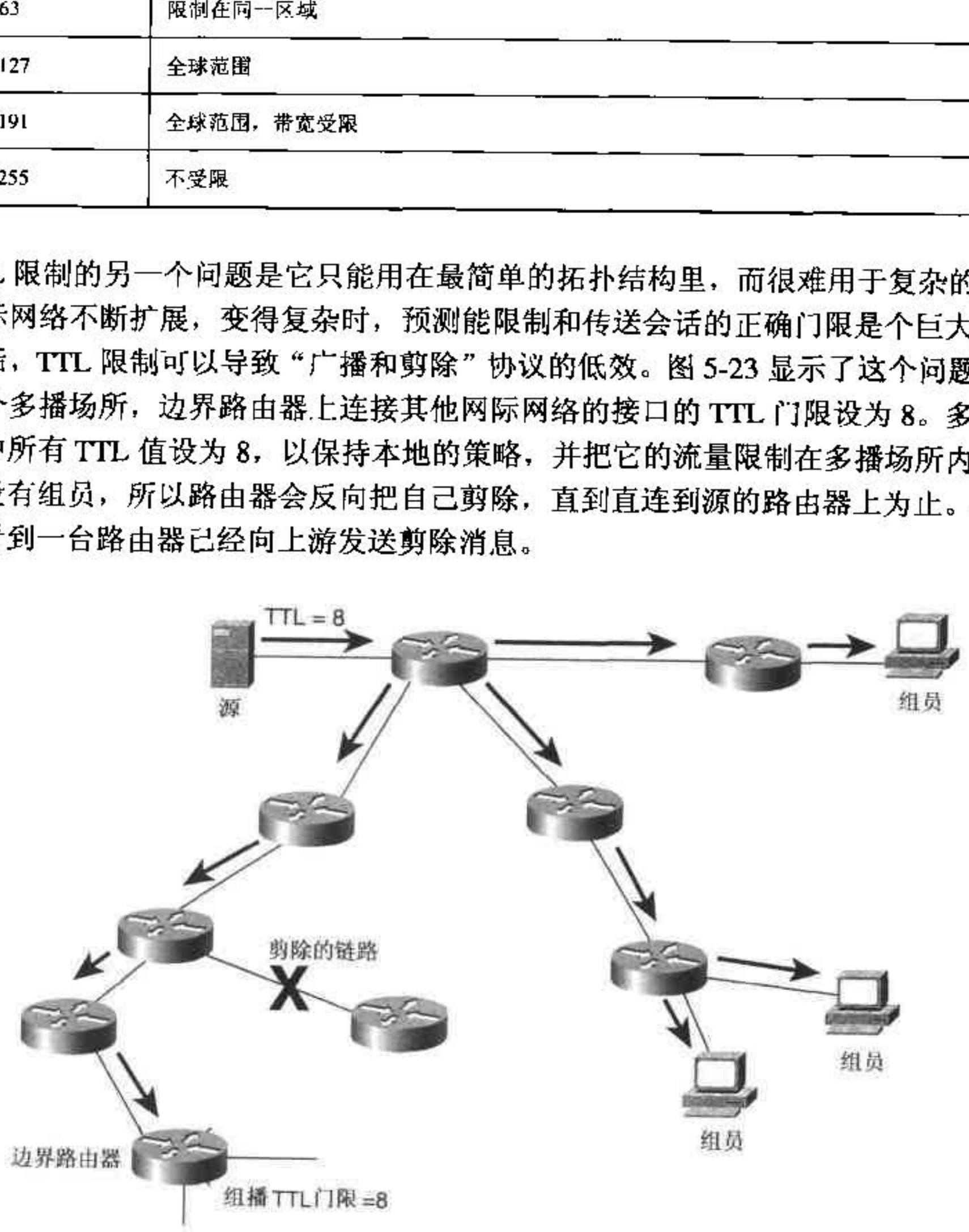


图 5-23　边界路由器上的 TTL 多播过滤器禁止向上游发送剪除消息

边界路由器与它设置的 TTL 过滤器存在着问题。当多播包到达这台路由器时，这个包在两个下游接口上被丢弃，因为它的 TTL 值小于 TTL 的门限。这正是所需要的。不过，包的

丢弃也意味着查找主机的 IGMP 的查询不会起作用。没有这些查询，路由器就不能向上游发送剪除消息。这样，多播流量仍会从源前转，尽管没有必要，经过所有路由器后到达边界路由器。

2. 管理限制

管理限制在 RFC 23657 中描述，它采用不同的方法来规定多播的范围。它用一组保留的 D 类地址用于限制，对这些地址进行过滤可以设立边界。这个保留的多播地址范围是 239.0.0.0～239.255.255.255。

用于管理限制的地址空间可进一步成为分层的结构。比如 RFC 2365 中建议 239.255.0.0/16 用于本地，239.192.0.0/14 用于整个组织内部。不过一个企业可以按照自己的需求自由使用这一地址空间。这样看来，保留的 D 类地址与 RFC 1918 中专用的保留地址很相似。和这些地址一样，管理限制的多播地址空间不是唯一的，所以过滤 239.0.0.0～239.255.255.255 这一地址范围内的地址很重要，目的是为了使这些地址不会泄露到公开的 Internet 上。

你已经在这一章和其他地方知道了 TTL 的限制和基于地址的限制。记住：IGMP 和 MOSPF 的 TTL 总是设成 1，以防止这个包被收到它的路由器继续转发，这样就可把这个包限制在本地的子网中。与之相似，路由器不把地址范围为 224.0.0.0～224.255.255.255 的包前转。这一范围内的地址包括了表 5-1 中的所有地址，它也应限制在本地的子网中。

5.4　距离向量多播路由协议(DVMRP)的操作

DVMRP 采用广播和剪除方法来为每个多播源建立一个基于源的树。它采用了 RIP 协议的变量来发现到源的最短路由，因此称为距离向量多播路由协议。每个多播树在组员退出和加入组时剪除和加入树枝来进行动态维护。

DVMRP 有 7 种包的类型：

- DVMRP Probe
- DVMRP Report
- DVMRP Prune
- DVMRP Graft
- DVMRP Graft Acknowledgement
- DVMRP Ask Neighbors2
- DVMRP Neighbors2

所有包的目的地址都是 224.0.0.4，这个地址是保留的“所有 DVMRP 路由器”地址(见表 5-1)。各种包的用途将在下面的章节中讲述，在“DVMRP 包格式”一节中会详细描述包的格式。

DVMRP 有多个版本。版本 1 在 RFC 1075[8] 中描述，最新的一个版本——版本 3 在一个 Internet 草案 9 中描述，这一章描述的是版本 3 的协议。你应该知道前面的版本不论是在功能，还是包结构上都有很大的变化。虽然本章对 DVMRPv3 和较早的版本间的差别做了比较，但把所有的不同之处都含概进去会使本章过长，且过于复杂。在这一章中，除非作特殊说明，“DVMRP”一词指的就是 DVMRPv3，如果你使用的是较早的版本，且对这些差异有兴趣，

可以阅读 RFC 1075、相关的 mgated 的文档，或支持较早版本的路由器协议的软件文档。

> 注：　MBone 中的多数路由器运行 DVMRP，而且多数运行某一版本的 mrouted 或 mgated。

Cisco IOS 软件不完全支持 DVMRP；不过，它却能支持与 MBone 这样的 DVMRP 网络的连接。

5.4.1　对邻居的发现和维护

DVMRP 路由器启动时第一个步骤是就是用 Probe 包发现邻居，每一个 Probe 包都有如下的信息：

- 一组描述发起路由器 DVMRP 能力的标志，作用是为了与这个协议较早版本的后向兼容；
- 一个生成 ID，用于检测邻居状态的变化；
- 一组发起路由器收到 Probe 包的邻居地址。

所有这些信息中，最基本的是这一组邻居的地址。当一个 DVMRP 路由器收到一个 Probe 包时，它记录下这个发起路由器的地址和收到这个包的网络接口。记住，接收到包的路由器不能把目的地址在 224.0.0.0/24 范围内的包前转。因为 Probe 包的目的地址是 224.0.0.4，还因为它的 TTL 值为 1，这个 DVMRP 路由器知道发起路由器是个直连的邻居。当这个路由器发出自己的 Probe 包时，将列出所有在其子网中获得的邻居地址。当一个路由器看到它自己的 IP 地址在邻居发出的 Probe 包中时，它就知道它与邻居建立了双向通信。

发现邻居后，路由器继续发送 Probe 包来进行保持。Probe 包每隔 10s 发送一个，如果在 35s 内没有收到 Probe 包，那么这个邻居就被宣布死亡。

早期版本的 DVMRP 不采用 Probe 包，它是通过邻居发出的路由宣告来发现邻居的。

在发现邻居的过程中，DVMRP 较早的版本当发现子网中有多于一个路由器上有组员时，便选出一个指定的路由器，这个指定的路由器是唯一能发送多播会话、响应子网中 IGMP 查询的路由器，它是子网中有最小 IP 值的那台路由器。DVMRPv3 通过 IGMPv2 的查询来决定指定路由器，而不是通过读出收到路由宣告消息的源 IP 地址。

正如你前面知道的一样，广播和剪除多播路由协议必须存储剪除状态。如果路由器重启，那么它就不能知道是否发出和收到了剪除消息。如果必须等待下一次常规的路由更新，则重新建立多播包的前转可能会很慢。生成 ID 的设计就是为了避免这个问题。生成 ID 是一个不会减小的 32bit 数值，它的值从一些变化来参考，比如从 time-of-day 时钟获得。当一 DVMRP 路由器重启时，它的生成 ID 也会改变。当邻居通过这个路由器的 Probe 包检测到这个变化时，它会刷新这个路由器发出的所有剪除状态，它们也会立即向这个邻居发送自己的路由表。多播数据因为清除了剪除信息，所以会流到这台重启的路由器上。这台路由器将决定是将自己剪除，还是作为树的一部分。

5.4.2　DVMRP 路由表

DVMRP 路由表的主要目的就是对每个多播源判断可到它的上游接口。在本章前面曾讲到，这一过程对避免环路的产生很重要；如果一个源的包在非上游接口(离源最近的接口)上

收到，那么这个包会被丢弃。

DVMRP 采用了 RIP 的许多变量来对外告知路由表和直连的子网。路由通过 DVMRP Report 消息从“所有 DVMRP 路由器”地址 224.0.0.4 对外告知，路由更新每 60s 发送一次，这个时间间隔称为路由报告间隔。这个规定也有例外，当发现了一台新的路由器时，路由表马上通过单播送到新的路由器处。快速的更新可以减少收敛的时间。

如果一条路由在 140s 内(路由超时时间)没有更新，那么这条路由还将继续保持两个报告间隔(120s)。在这段时间内，这条路由在对外宣告时设它的量度为无限；当保持时间也过期后，这条路由将从路由表中被删除。

量度与每一条路由相关，是跳数的总和。如果设为 32 跳，意味着跳数无限。不过，路由器可以把量度设成 1～63：1～31 表示可达源，33～63 表示依赖路由。

为了能正确剪除，DVMRP 路由器必须知道靠它前转多播包的下游邻居。对每一个源网络来说，下游的路由器向上游路由器发出一条逆向毒化的路由，这条路由是依赖于上游路由器的。逆向毒化的路由的量度设为其在宣告中的量度加上无限(32)。比如，假设路由器 A 向路由器 B 宣告 172.16.1.0/24 到路由器 B 有 3 跳，路由器 B 判断路由器 A 是这个子网的上游路由器，那么路由器 B 就必须向路由器 A 表示它要依靠路由器 A 从源前转多播流量到其子网。因此，路由器 B 告诉路由器 A 路由 172.16.1.0/24，并把其量度设为 35(32+3)。路由器认为这样宣告的路由就是依赖路由。

DVMRP 路由表的另一个功能就是选举指定前转器。如图 5-24 所示，当多个上游路由器连接到一个多路访问的网络中时，只有指定的前转器才能向下游转发包，这就避免了一个包的多个备份被前转到多路访问的网络中。当两个或多个路由器在一个多路访问的网络中交换路由信息时，它们可以告诉对方谁离多播源更近。那么，这台路由器就成为指定前转器。图 5-24 中，上游路由器 B 将成为指定前转器，因为它是唯一的可以一跳到达多播源的路由器；上游路由器 A 是两跳。如果这些路由器到达多播源的距离相等，则共享网络中有较低 IP 值的路由器会成为指定的前转器。

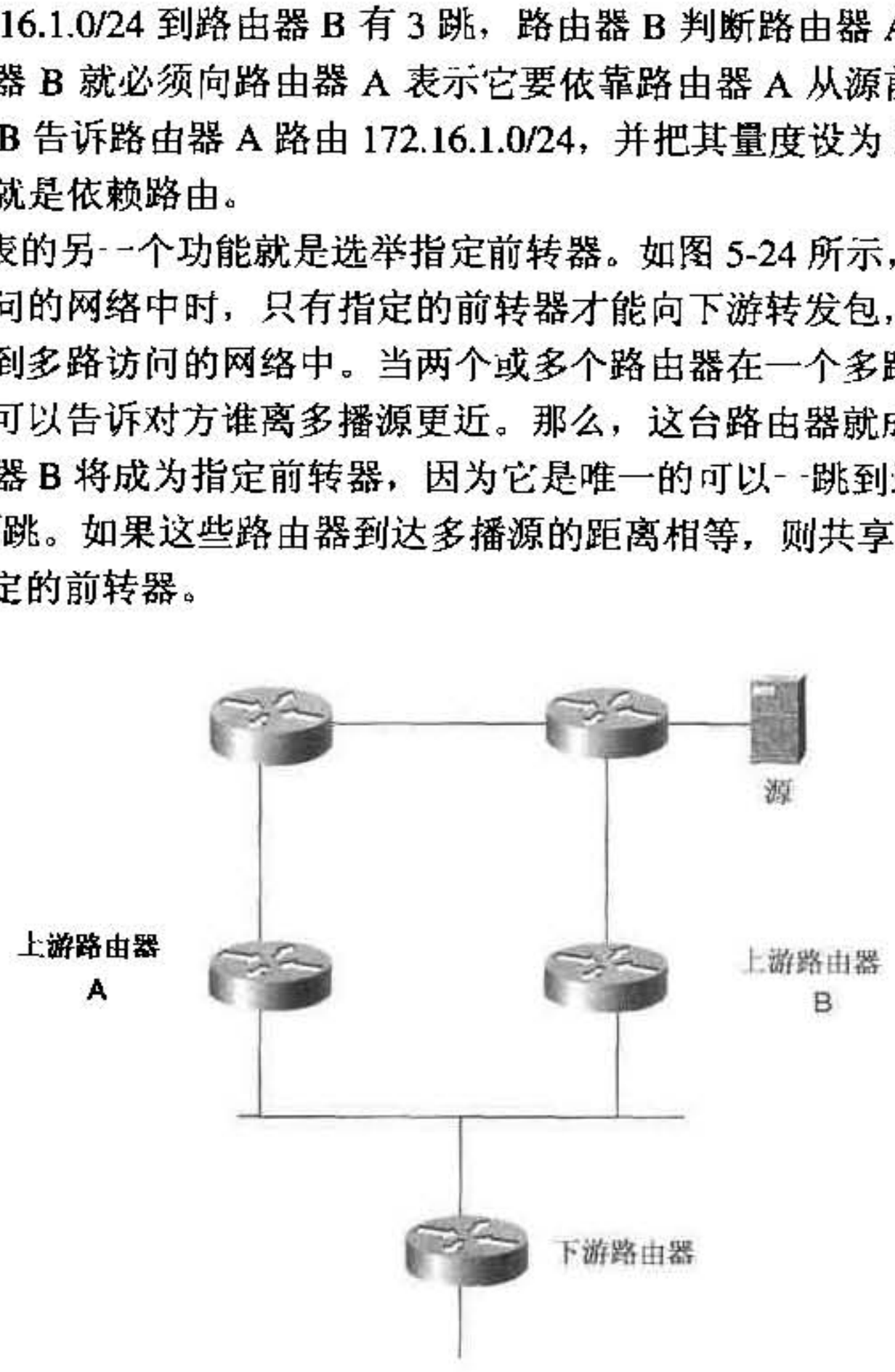


图 5-24　当多个上游路由器连接在同一数据链路上时，将选出一个指定路由器

5.4.3　DVMRP 包的前转

当路由器从一个多播源第一次收到一个多播包时，它会进行 RPF 检查，就是用路由表来核实这个包是在连接多播源的上游接口上收到的。如果这个包是在其他接口上收到的，则被丢弃；如果这个包的确是在上游接口上收到的，那么就在前转表中记录下一个(S,G)对，这个包会前转到下游各相关的邻居处。这个路由器也会用 IGMP 来查询它的每个叶网络(这些网络中没有邻居)中是否有组员。备份也会被前转到任何有组员的叶网络中。

如果没有下游邻居，而且叶网络中也没有组员，那么这个路由器将向上游路由器发送一个剪除消息。如果上游路由器也没有本地的组员，并且在收到所有下游路由器的剪除消息后，再向自己的上游邻居发送一个剪除消息。这样，多播树会动态减除，只保留可以到达有激活组员的树枝为止。

一个剪除消息里有剪除存活时间这一参数，它指示了上游路由器在需要恢复前转多播数据、查询被剪除的路由器之前，应保持多久剪除状态，默认值为 2 个小时。如果路由器收到了剪除消息后还要向上游发送剪除消息，那么它会把发出的剪除消息中的存活时间设为 2 个小时与在下游收到对同一(S,G)对的剪除消息里设置的存活时间两者中的最小值。

如前所述，一台主机可以随时向其本地路由器发送一个 IGMP Group Membership Report 消息，表示要加入一个多播组。如果这台路由器此前已经把自己从多播组的树中剪除，那么它必须把自己重新接入树中。路由器向上游路由器发送 DVMRP Graft 消息，这一消息向上游一跳一跳地传送，直到建立一条多播树的树枝为止。

如果路由器发出了一个接入消息，但却没有开始接收请求的流量，它就必须通过一种机制来获知多播源是否已经停止发送流量，或者接入消息被丢弃了。因此，在每一跳上，上游路由器都会应答一个 Graft Ack 消息，表示收到下游的邻居的 Graft 消息。产生 Graft 消息的路由器中设置了一个 Graft 重发计时器；如果 Graft Ack 消息在计时器超时前没有收到，则路由器会发出另一个 Graft 消息，这个计时器重置。Graft 重发计时器开始设为 5s，随后的值可以通过二进制指数后退算法算出。

5.4.4　DVMRP 消息的格式

DVMRP 消息的 IP 包头的协议号定为 2。这一协议号与 IGMP 相同，IGMP 是 DVMRP 开始阶段的遗物，原来是这个协议的一个子集。这一节描述 DVMRPv3 的消息格式；较早版本的描述可在 RFC1075 或其他文档中找到。

1. DVMRP 消息头

图 5-25 显示了 DVMRP 头的格式，每一个 DVMRP 消息都是以其开始的。

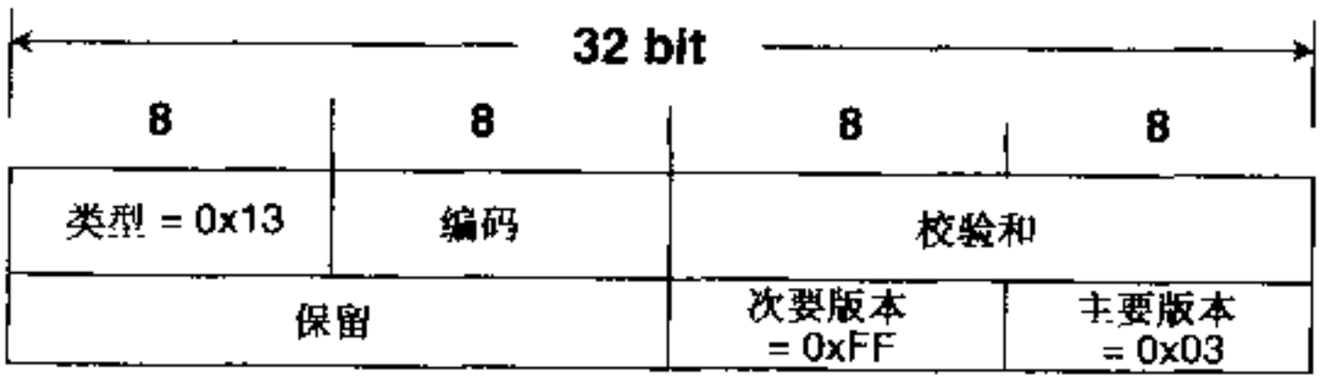


图 5-25　DVMRP 消息头

DVMRP 消息的各部分描述如下：

- 类型是 IGMP 的类型号，对于所有的 DVMRP 消息均设为 0x13。RFC1075 中，这部分定义为 4bit 版本和 4bit 类型，版本为 0x1，类型为 0x3。版本 1 中的这 8bit 值为 0x13，与版本 3 相同，使得版本 3 能后向兼容。实际的 DVMRPv3 的版本在“主要版本”参数中规定。
- 校验和是标准的 IP 风格的校验和，为一个 16bitDVMRP 消息补码和的补码。
- 次要版本和主要版本对于所有的 DVMRP 消息分别设为 0xFF 和 0x03。
- 编码定义了 DVMRPv3 消息的消息类型。表 5-7 中列出了编码可能的取值和相应消息的类型。

表 5-7　**DVMRP 消息类型**

编　码	DVMRP 消息类型
1	Probe
2	Report
3	Ask Neighbors
4	Neighbors
5	Ask Neighbors2
6	Neighbors2
7	Prune
8	Graft
9	Graft Ack

Ask Neighbors(编码 3)消息和 Neighbors(编码 4)消息由 Ask Neighbors2(编码 5)消息和 Neighbors2(编码 6)消息替代。这些消息还没有讨论过，它们由 **mrinfo** 和 **mstat** 诊断命令使用，将在第 6 章的“排错”一节中进行讨论。

2．DVMRP Probe 消息格式

DVMRP Probe 消息有 4 种功能：

- 它们让路由器通过发起路由器列出所有检测到的 DVMRP 路由器，互相知道彼此的位置；
- 它们可以让 DVMRP 路由器互相通告彼此的能力；
- 在有多条通路都可以到达下游组员时，它们可以选出指定前转器；
- 它们每 10s 发送一次，具有保持存活的功能，如果 35s 内没有听到邻居的 Probe 消息，这个邻居将宣布为死亡。

图 5-26 显示了 Probe 消息的格式。

DVMRP Probe 消息的各参数描述如下：

- 能力占用了包头中保留的 8bit。Probe 消息是 DVMRP 中唯一一个修改了包头中参数的消息。表 5-8 列出了能力标识值和它们代表的意思。如果一个标识设为 1，则发起路由器支持相关的能力。

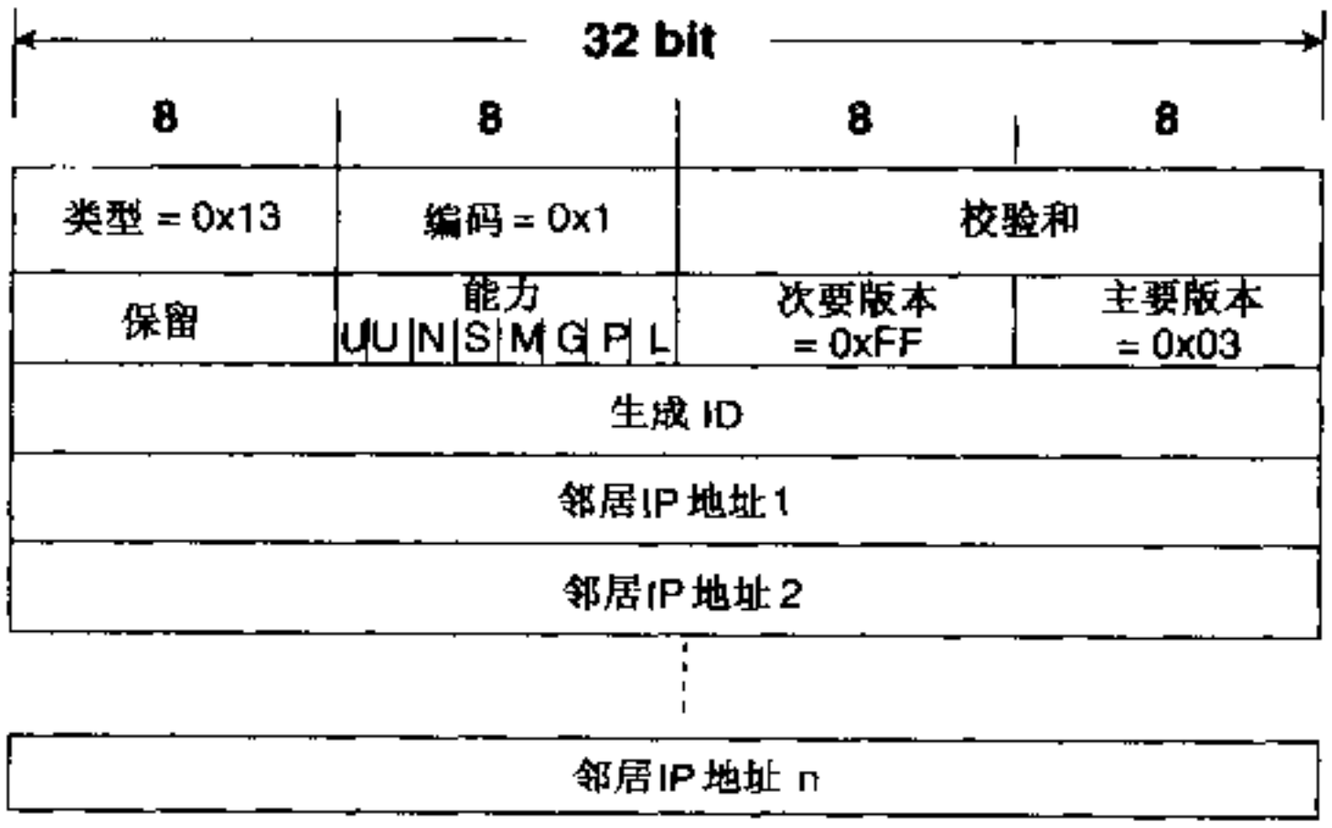


图 5-26　DVMRP Probe 消息格式

表 5-8　　DVMRP 能力标识

比　特	标　识	能　力
0	L	这个路由器是叶路由器
1	P	这个路由器理解剪除
2	G	这个路由器发送生成 ID
3	M	这个路由器能处理 Mtrace 请求
4	S	这个路由器支持 DVMRP MIB
5	N	这个路由器理解 Prune、Graft、Graft Ack 消息中附加的掩码
6，7	U	未使用

• 生成 ID 是一个不会减少的 32bit 数字，用于检测路由器的重启。当检测到一个生成 ID 有变化时，任何从这个发起路由器发出的剪除消息均为无效，并被清除。如果有一剪除消息曾被发向上游，则应再发出一个接入消息。这一过程的结果就是把重启的路由器按多播树里的一个新的路由器进行处理，重新开始一次广播与剪除过程。

• 邻居地址列出了发起路由器收到的发出 Probe 消息的邻居的地址。

3．DVMRP Route Report 消息格式

Route Report 消息每 60s 发送一个，其格式如图 5-27 所示。这条消息由一组或多组网络掩码构成，对于每一个掩码，有对应的一个或多个网络地址和相应的量度值。虽然图 5-27 所示的源网络的长度均为 3 个 8 位组，但实际中，这个长度如本节中将要讲述的一样，是可变的。

DVMRP Route Report 消息中的各参数定义如下：

• 掩码为网络掩码。网络掩码的第一个八位组总是设为 255，所以只有后 3 个八位组在掩码这个参数里。请注意，这个假设意味着 DVMRP 路由不能收敛于掩码小于 8 的地址。

• 源网络是源网络地址，它的前缀长度同前面的掩码相对应。源网络的长度根据前面的

掩码长度而变化。比如，如果网络掩码为 255.0.0，这个参数表示这个掩码为 255.255.0.0(第一个八位总是为 255)。这个参数后的源网络根据前缀长度的规定为 2 个八位组。一条默认路由被定义为网络掩码为 0.0.0，源网络的第一个八位组为 0。DVMRP 路由器将这条路由解释为 0.0.0.0/0，而不是 0.0.0.0/8。

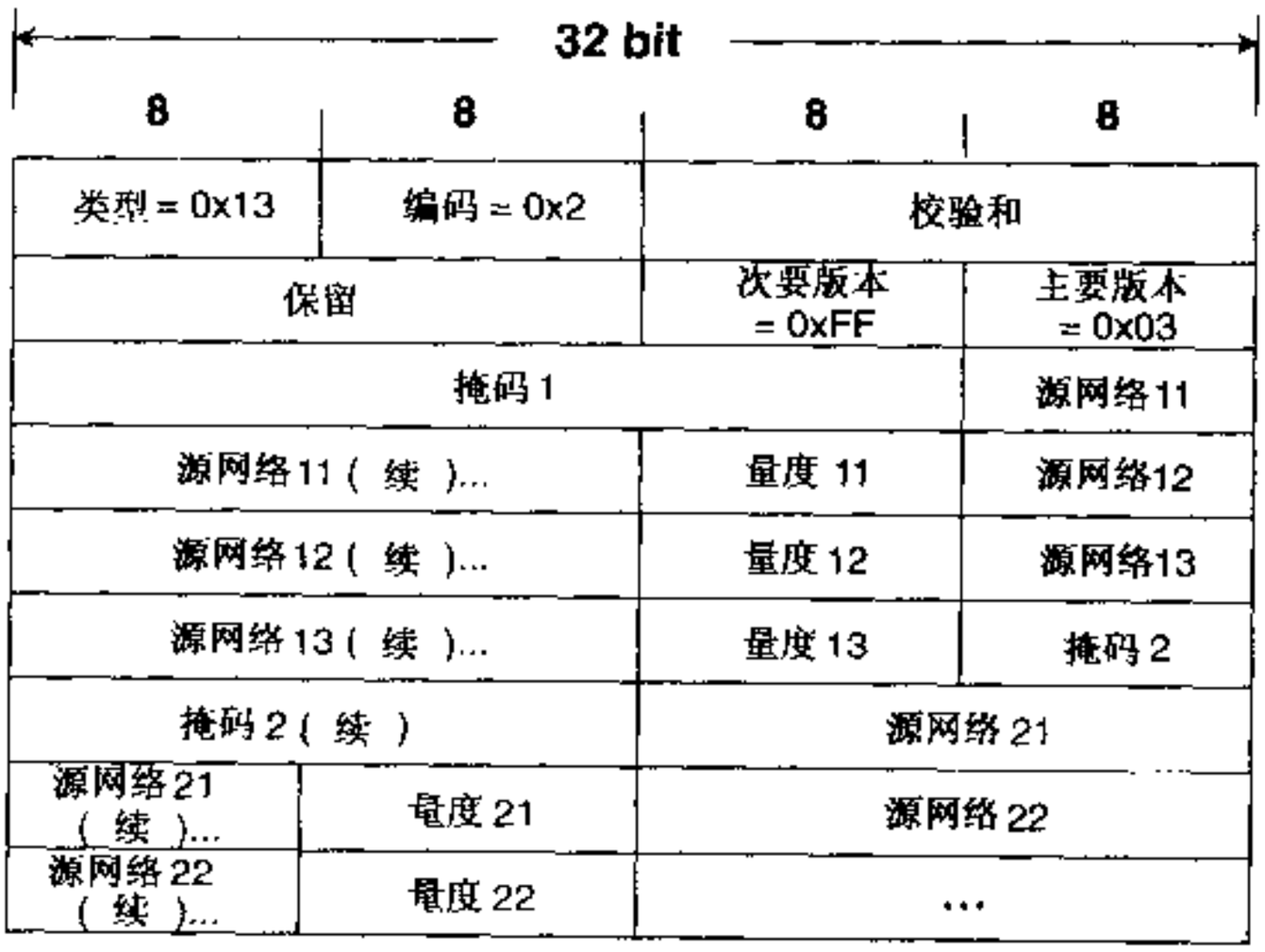


图 5-27　DVMRP Route Report 消息格式

- 量度是发起这个报告的路由器和源网络间所有接口的量度之和。这个量度是跳数之和，32 表示无限的跳数，但量度的取值范围为 1～63。正如“DVMRP 路由表”一节所述，路由器用逆向毒化向外告知一条依赖于上游路由器的路由，这条路由的量度为收到的量度加上无限(32)。因此，33～63 的量度表示下游依赖路由。

4．DVMRP Prune 消息格式

图 5-28 中显式了一个 Prune 消息格式。

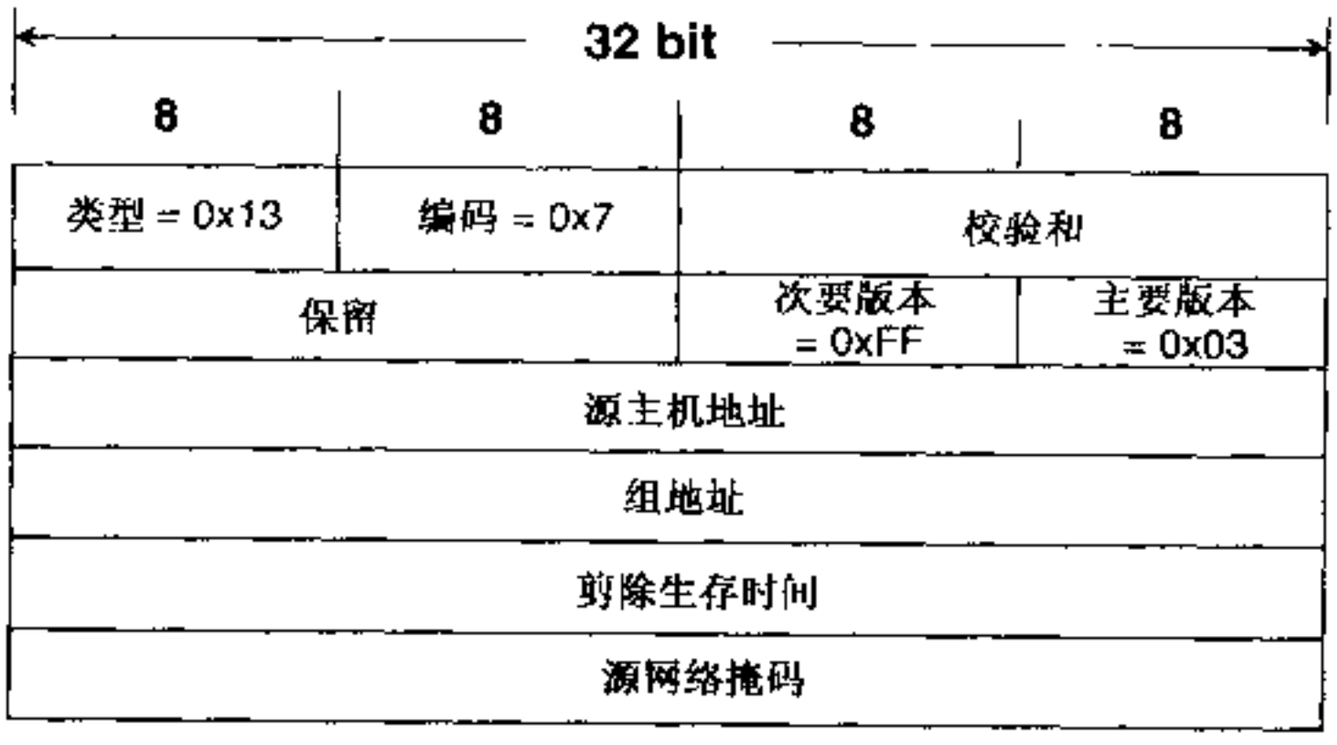


图 5-28　DVMRP Prune 消息格式

DVMRP Prune 消息的各参数定义如下：

- 源主机地址为发起主机的 IP 地址。

- 组地址为将被剪除的组的 IP 地址。
- 剪除生存时间是以 s 为单位的上游路由器保存剪除状态的时间。这个值或者是这个组中收到的下游剪除消息中的最小值，或者是在没有下游剪除消息的情况下使用的默认的存活时间 2 小时。
- 源网络掩码为将被剪除的多播组源网络的掩码。这个参数是可选的，仅当上游邻居在 Probe 消息中表明它支持网络掩码时才包括这个参数。

5. DVMRP Graft 消息格式

图 5-29 中显示了 Graft 消息的格式。

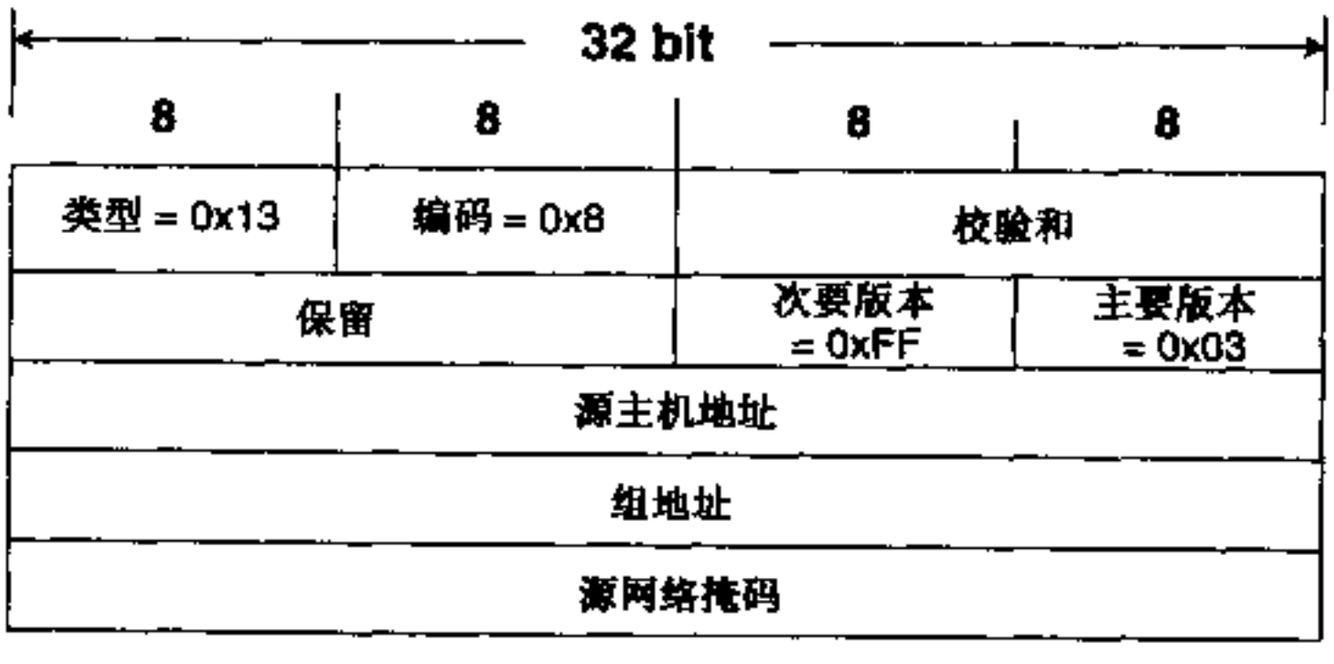


图 5-29　DVMRP Graft 消息格式

DVMRP Graft 消息的参数定义如下：

- 源主机地址是发起主机的 IP 地址。
- 组地址是要接入的组的 IP 地址。
- 源网络掩码为将被接入的多播组源网络的掩码。这个参数是可选的，仅当上游邻居在 Probe 消息中表明它支持网络掩码时才包括这个参数。

6. DVMRP Graft Acknowledgement 消息格式

图 5-30 显示了 Graft Acknowledgement 消息的格式。除了头中的编码参数不一致外，格式同 Graft 消息一样。

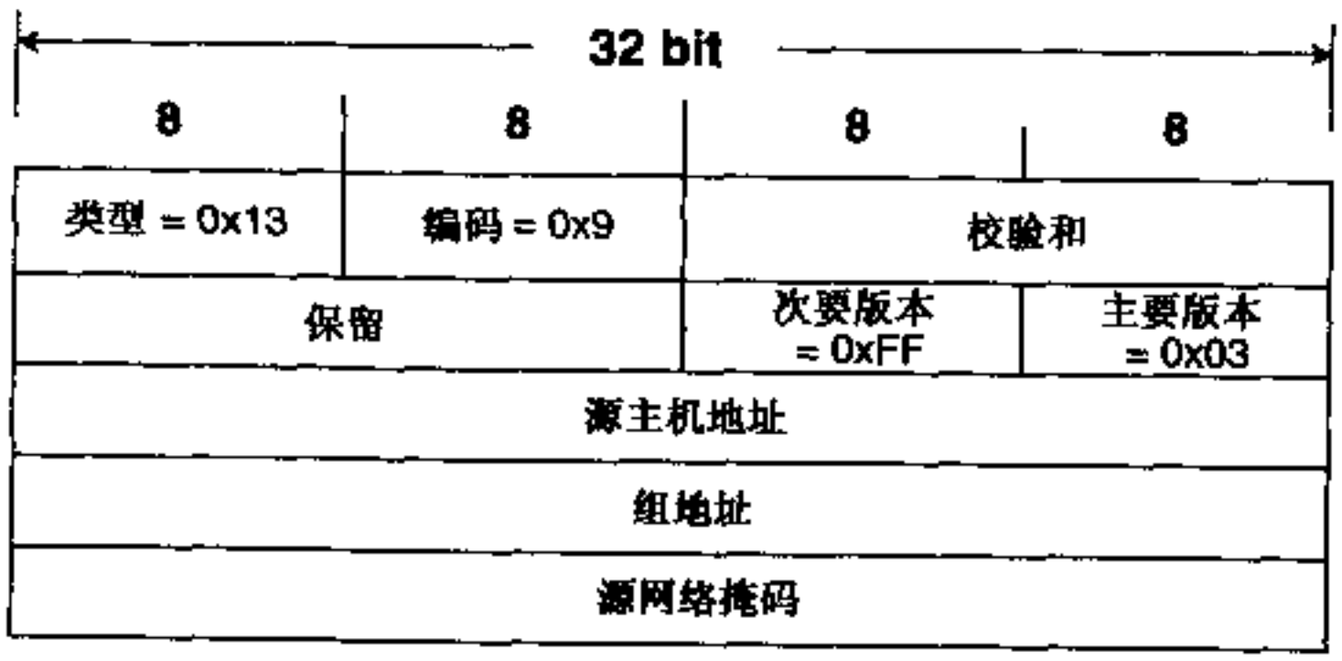


图 5-30　DVMRP Graft Acknowledgement 消息格式

7. DVMRP Ask Neighbors2 消息格式

DVMRP Ask Neighbors2 消息是两个用于排错的消息中的一个(另一个是 Neighbors2 消

息，将在本节之后讨论)。数字“2”用于与作废的 Ask Neighbors 消息区别开来。Ask Neighbors 2 消息(如图 5-31 中所示)是一个发向一特定目的地的单播。当路由器收到 Ask Neighbors2 消息时，它应用单播消息 Neighbors2 向消息发起者应答。如图 5-31 所示，这个消息仅是一个 DVMRP 消息头，其编码设为 0x5。

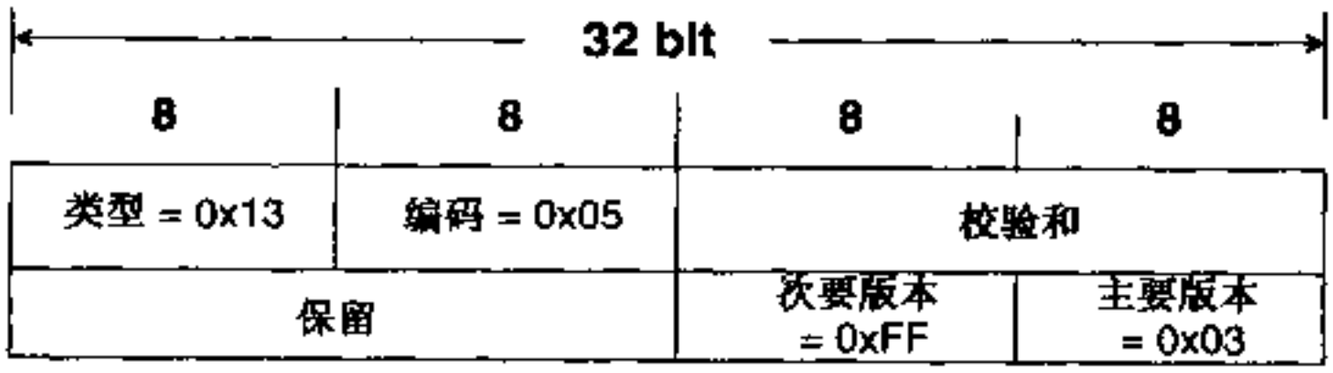


图 5-31　DVMRP Ask Neighbors 2 消息格式

8．DVMRP Neighbors2 消息格式

DVMRP 发送 Neighbors2 来响应 Ask Neighbors2 消息，图 5-32 显示了这个消息的格式。这条消息是向 Ask Neighbors2 消息的发送者发送的单播。这个消息列出了路由器的能力和发送者的各逻辑端口的地址。对于列出的每个端口，均规定了这个接口的 DVMRP 参数，而且所知道的这个接口上的 DVMRP 邻居也列出了。

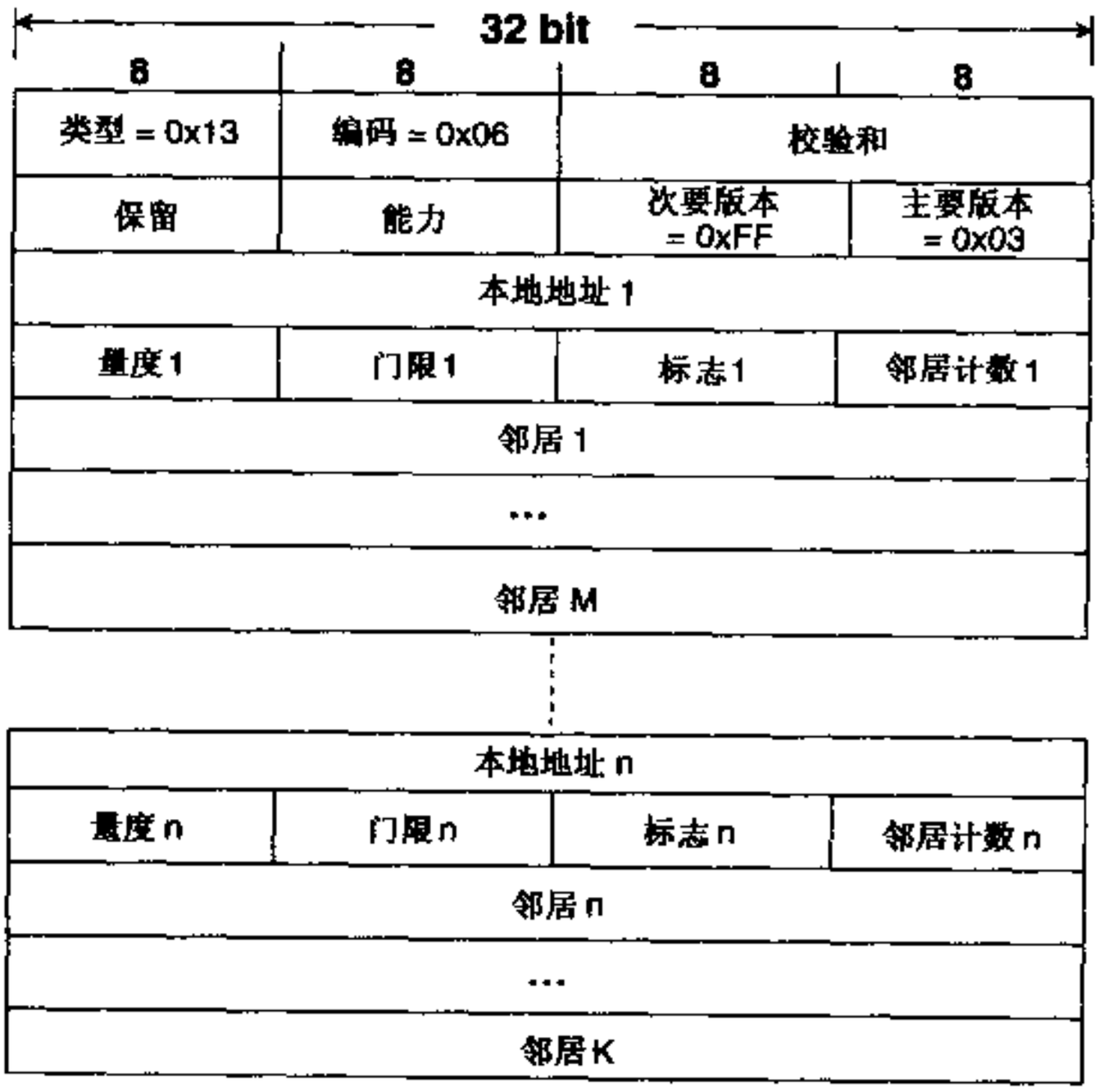


图 5-32　DVMRP Neighbors 2 消息格式

DVMRP Neighbors2 消息的各参数定义如下：

- 能力定义了发起路由器的 DVMRP 能力。这一参数与 Prune 消息中的能力参数相同，它各比特的含义见表 5-8。

- 本地地址是这个路由器的上一个接口的地址。如果这个接口被宕掉或禁用，那么它和一个单一的邻居地址相对应，这个邻居地址为 0.0.0.0。
- 量度定义了这个接口的 DVMRP 的量度。
- 门限定义了接口管理范围的门限。
- 邻居计数定义了这个接口上列出的邻居的数目。
- 邻居是知道这个接口上的 DVMRP 邻居路由器的 IP 地址。
- 标识是一系列比特，用于描述这个接口上的可选参数。表 5-9 列出了这个参数中每个比特代表的意思。

表 5-9　　Neighbor2 接口标识消息

比　特	标　识	描　述
0	隧道	通过隧道到达邻居
1	源路由	隧道使用 IP 源地址路由
2	保留	未使用
3	保留	未使用
4	关闭	操作的状态关闭
5	禁用	管理的状态关闭
6	查询	对接口的查询
7	叶	这个接口没有下游的邻居

5.5　MOSPF 的操作

多播 OSPF(MOSPF)比 DVMRP 在两个方面有了很大的提高。首先，它是一个链路状态的协议，DVMRP 是一个距离向量的协议。这个差别在于所有的链路状态的协议都强于距离向量协议：因为它有更好的收敛特性、更好的防止环路，以及更少的周期性控制流量。第二点就是 MOSPF 在密集的环境中有更好的扩展性，在某种程度上因为它是链路状态协议，但是也是因为 MOSPF 采用的是显性加入，而不是通过扩散与剪除的隐性加入。

MOSPF 并非是与 OSPF 截然不同的协议，它更像是 OSPF 的一个扩展，如同描述它的那个 RFC[10] 文件的名称一样。OSPF 中定义了 3 个扩展来支持多播，第一个定义了新的 LSA，称为组员资格 LSA，这个 LSA 的类型为 6。

可选项经扩展后包括了一个标识，称为 MC 比特，用于指示对多播的支持。这一个可选项在第 1 卷第 9 章中有描述，它在 OSPF Hello 消息和 Database Description 消息中承载，在所有的 LSA 中都存在。MC 比特所隐含的意思是 OSPF 与 MOSPF 可以共同在一个网际网络中使用，MOSPF 的路由器用 MC 比特来指示它们对多播的支持。不能设置 MC 比特的路由器也可以建立邻接关系。不过只有那些设置了 MC 比特的邻居才能在进行数据库同步时，以 Database Description 包交换组员资格 LSA。而且，只有设置了 MC 的 LSA 才能用于多播最短

路径的计算。

最后，路由器 LSA 的 rtype 项在扩展后包括了一个标识，称为 W 比特，这个标识用于指示发起路由器是一个通配多播接收者。通配多播接收者将在“区域间 MOSPF”一节中进行定义。

同 OSPF 采用基于 Dijkstra SPF 算法来计算到目的的最短路径一样，MOSPF 计算从多播源到多播目的的树。单播树和多播树都以同一个链路状态数据库计算。但不同的是，单播的 SPF 树根植于源路由器，而多播 SPF 则根植于源多播子网。

5.5.1　MOSPF 基础

开始描述 MOSPF 最好是从连接着组员的本地多路访问介质开始。同单播 OSPF 一样，MOSPF 选出一个指定路由器和一台备份指定路由器。所有连接的 MOSPF 路由器应通过在本地链路上运行 IGMP 来发现组员，但只有 DR 可以发送 IGMP 查询消息，并监听 IGMP Membership Report。

回忆表 5-4 中 MOSPF 为显示加入。当一个组员发送 IGMP 消息来指示它希望加入一个组时，MOSPF DR 在本地组数据库中建立一个条目。这个本地组数据库条目记录了组和有组员的那个连接的网络。比如，图 5-33 中的路由器有 3 个连接的子网，其中的 2 个子网中有 3 个组员。两个不同的子网中的组员属于同一个多播组。路由器必须知道多播组和有组员的子网，但不必知道每一个组员。

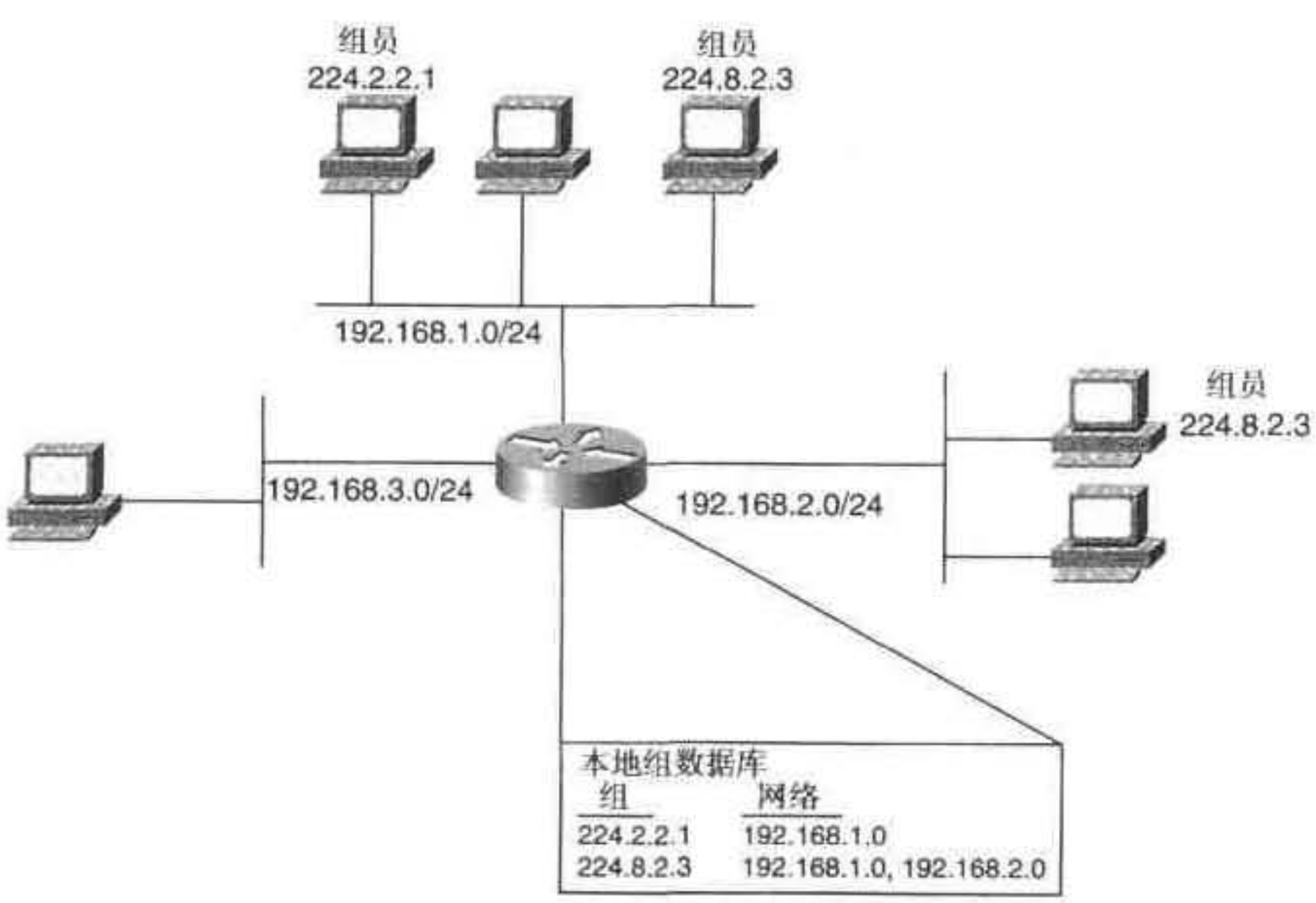


图 5-33　本地组数据库记录有组员的相邻组与子网

这时，DR 为每一个连接的组产生一个组员资格 LSA，这个 LSA 中定义了组地址和发起路由器的 ID 及所有有组员的连接的子网。在某些情况下，路由器本身运行一个多播应用，使其成为一个组员。LSA 中有一个类型项，可以用来指示路由器宣告自己是一个组员。

这个 LSA 将扩散到发起路由器的区域内。组员资格 LSA(类型 6)与网络 LSA 在两个方面很相像：

- 如同网络 LSA 一样，只有指定路由器才能发起组员资格 LSA；
- 如同网络 LSA 一样，组员资格 LSA 只在区域中有效，也就是说，这个 LSA 不会扩散到发起路由器的区域之外。

LSA 扩散的目的就是保证一个区域内的 MOSPF 路由器有一个组员资格 LSA 的备份。同单播 OSPF 一样，一个区域内所有的 MOSPF 都有一个一致的链路状态数据库。在一个区域中，OSPF 与 MOSPF 的链路状态数据库的不同在于类型 6 的 LSA。

有了同步的数据库后，每一个区域中的 MOSPF 路由器可以计算相同的最短路径树。这个树根植于源网络，并通过树枝伸向各个有组员的网络。不过这个树不是马上计算出来的，相反，当一个组的多播包到达时，路由器在请求下才进行计算。这有它自己的道理，因为虽然同步的路由器知道所有的目的在哪里，但它并不知道源在哪里。

SPF 计算通过组员资格 LSA 知道的所有连接着组员的路由器，并且根据这个组中第一个包的源地址和目的地址知道源的位置。一般设置了 MC 比特的单播路由器和网络 LSA 都可用于计算从源到每一个目的的最小开销路径。

基于组员资格 LSA 的显式加入和按需计算 SPF 的很大的优越性在于，路由器在计算前已经知道目的网络的位置，所以与“扩散和剪除”协议(比如 DVMRP)不同，包从来不会被前转到路由域的其他部分。可以说，MOSPF 是“预剪除”的。

在 SPF 计算结果的基础上，路由表的多播前转表中加入了很多条目。最短路径树是无环的，每一台路由器都知道哪个接口是上游接口，哪个接口是下游接口，因此不需要像 DVMRP 中的 RPF 检查一样。对于特定的(S,G)对，前转表条目指明了应该从其收到包的上游邻居和必须前转到的下游邻居。本地组数据库也用于建立到有组员的本地网络的前转表。

要记住几个关于 MOSPF 的注意事项：

首先，虽然单播 OSPF 支持等开销的多通路，但 MOSPF 不支持。MOSPF 的最短路径树是从源到有组员网络的一条单独的通路。

第二，如果 OSPF 与 MOSPF 路由器同时存在于一个多路访问的网络中，必须注意保证 MOSPF 的路由器选了 DR。如果一台 OSPF 路由器成为 DR，则不会为网络中的组员产生组员资格 LSA，于是就不会有多播组的包被前转到这个网络中。

最后，当一台 MOSPF 路由器在 MOSPF 域中的拓扑发生变化时，必须清除它的前转表，再重新计算其最短路径。因此，尽量保持整个域的稳定是很重要的。

5.5.2　区域间的 MOSPF

前一节中描述了当源和所有组员在一个区域内时的 MOSPF 的行为，重点强调了组员资格 LSA 不会扩散到其发起的区域之外。这样，当组员在一个或多个与源不同的区域时，会有什么情况？

已经知道第 1 卷第 9 章中，区域间的 OSPF 通信是通过区域边界路由器(Area Border Router，ABR)来管理的。ABR 是骨干区域的一员，同时也是一个或多个非骨干区域的成员。它们如同区域内的路由器一样，通过路由器 LSA 和网络 LSA 从连接的区域中获知所有的目的网络。然后，ABR 建立网络总结 LSA(类型 3)，它把 ABR 连接的一个区域内的网络通知到

ABR 连接的其他区域内。同类型 1 和类型 2 的 LSA 一样，类型 3 的 LSA 不会扩散到它产生的那个区域外。当一个 ABR 通过骨干区域收到另一个区域的一个网络总结 LSA 时，它建立自己的网络总结 LSA，并把这些信息告诉给连接的其他非骨干网区域。图 5-34 画出了 ABR 如何使用类型 1,2,3 的 LSA。

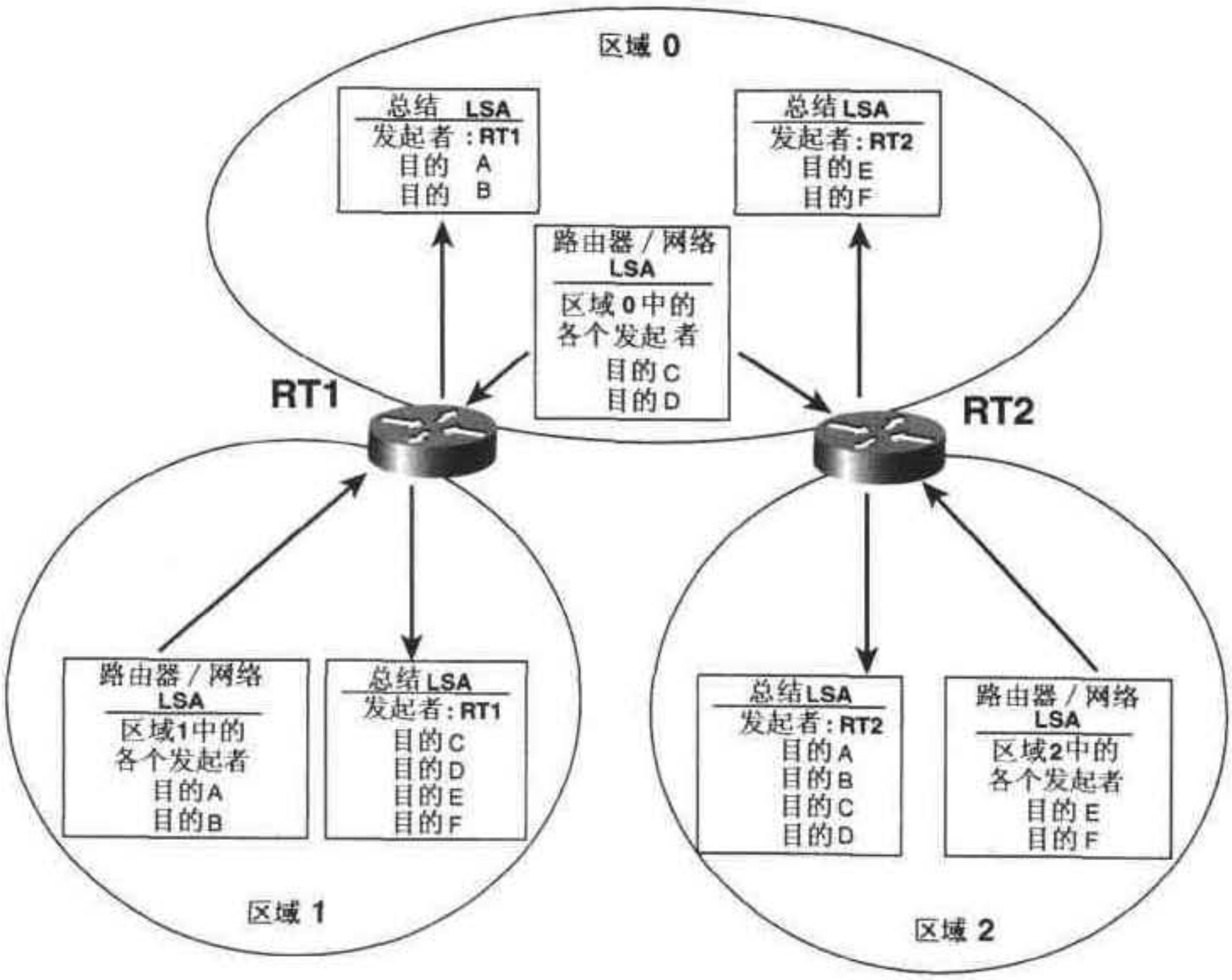


图 5-34　网络总结 LSA 把一个连接区域中获知的路由向另一个连接的区域宣告

MOSPF ABR 被误称为区域间多播前转器。它与单播 ABR 的操作既有相同点，也有不同点。一个区域间多播前转器根据与其连接的非骨干区域里收到的组员资格 LSA 来获悉每一个连接的区域中是否有组员。对于每个已知的组，前转器建立一个新的组员资格 LSA，并将这个 LSA 扩散到骨干网中，如图 5-35 所示。到现在为止，这些行为与 ABR 使用类型 3 的 LSA 来总结从类型 1 和类型 2 的 LSA 获得的信息，然后再发到骨干网中这一过程很相似。

到了这一步，就开始与单播 ABR 不同了。与类型 3 的 LSA 使用方式不同，区域内的多播前转不向非骨干区域发送类型 6 的 LSA 来宣告区域外有组的存在。比如图 5-35 中，RT1 收到一个 RT2 发出的类型 6 的 LSA，用于告知组 C，不过它不会建立一个类型 6 的 LSA 来向区域 1 告知组 C 的存在。这是因为为每一个组计算 SPF 树要在骨干区域中进行，树枝延伸到每个有组员区域的区域间多播前转器处。这些非骨干区域对任何他们区域外的组员一无所知。

如果图 5-35 中组 C 的源在区域 1，它那么的包怎么才能到达区域 2 和区域 3 中的组员处呢？答案是通配(wildcard)多播接收者。这些设备在对外宣告它们的路由器 LSA 的 rtype 项中

设置了 W 比特。在一个区域里，多播流量总是前转到所有通配接收者处的。在一个非骨干区域中，一个区域间多播前转器(多播 ABR)总是通配多播接收者。

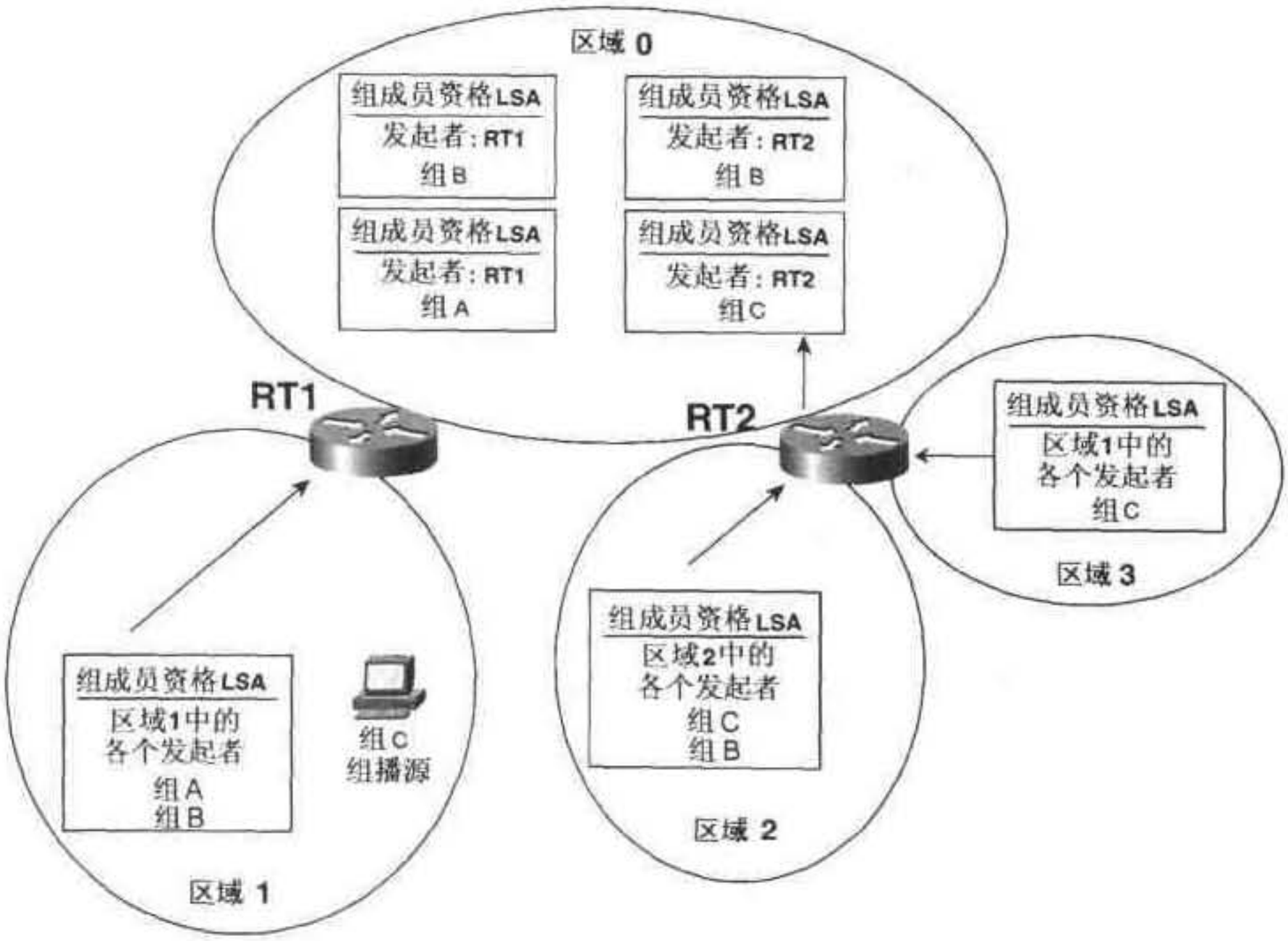


图 5-35　区域间组成员资格 LSA 向骨干区域宣告非骨干区域中现有组员的情况

当图中 5-35 中组 C 的源产生一个组 C 的包，这个包被前转给 RT1，RT1 它是区域 1 的通配多播接收者。RT1 也是骨干区域的一员，所以它还计算到那些连接区域中有组员的区域间多播前转器的最短路径树。知道了 RT2 对外宣告组 C 中的组员后，这个包通过骨干区域前转到该路由器。RT2 作为区域 2 和区域 3 的组员，为每个区域内的组 C 的成员分别计算 SPF 树，并把这个包前转到组 C 中的目的。

注：　如果组 C 的成员在区域 1，而这个包的备份被前转到 RT1 之外，那么当然还会通过本地的 SPF 树前转到这些组员处。

注意到，通配多播接收者不一定要在骨干区域处。对于 MOSPF 域中的每一个组，在区域 0 中计算一个 SPF 树。树的树枝延伸到这个区域中的组员中，或连在其他区域里的区域间多播前转器处。这样，如果一个源位于骨干区域，那么它的包可以沿正确的树前转。

5.5.3　AS 间的 MOSPF

RFC1584 提供了多播包在传入和从 MOSPF 域中发出的寻路方式。第一卷第 9 章中已经介绍到在 OSPF 域中对其他路由协议的路由的再分发是通过自治系统边界路由器(ASBR)来完成的。ASBR 用外部 AS LSA(类型 5)来宣告 OSPF 域外的目的地，用 ASBR 总结 LSA(类型 4)来宣告他们的位置。这些 LSA 会扩散到 OSPF 域的所有区域里，除了末梢区域之外。

把 MOSPF 域与某些其他多播域(有可能是 DVMRP，在未来可能是多播 EGP)连在一起的路由器称为 AS 间多播前转器。这些路由器与区域间多播前转器的行为非常相似。AS 间多播前转器把包前转到 MOSPF 域外，它将路由器 LSA 中的 W 比特进行设置，成为通配多播前转器。当路由器把包从外部源前转入 MOSPF 域中时，它就变成了“代理源”，其外部链路成为组 SPF 树的根。

同 ASBR 一样，AS 间多播前转器可以存在于任何区域里。不过要注意，通配多播前转的能力是由类型 1 的 LSA 中的 W 比特置为 1 来标识的，而类型 1 的 LSA 不会扩散到区域外。如果 AS 间前转器在区域 0，这不是个问题；区域间多播前转器已经把所有的多播流量送到骨干区域中。但如果 AS 间前转器在非骨干区域，则这个区域的区域间前转器也必须成为通配前转器，因此推荐把 AS 间多播前转器置于区域 0 中。

另外也建议把 AS 前转器安放在 MOSPF 域中时要仔细，因为所有的域内多播流量都前转到这些路由器上，使得到这些路由器的链路很容易阻塞。

5.5.4　MOSPF 扩展的格式

这一节只描述 OSPF 支持多播的扩展。对于 OSPF 包和 LSA 完整的描述，可以参阅第 1 卷第 9 章。

1．组员资格 LSA 格式

组员资格 LSA 中有标准的 LSA 头和类型标号 6，图 5-36 显示了组员资格 LSA 的格式。只有 MOSPF 指定路由器才会发送组员资格 LSA。注意，在这个格式里，LSA 没有相关的量度参数。

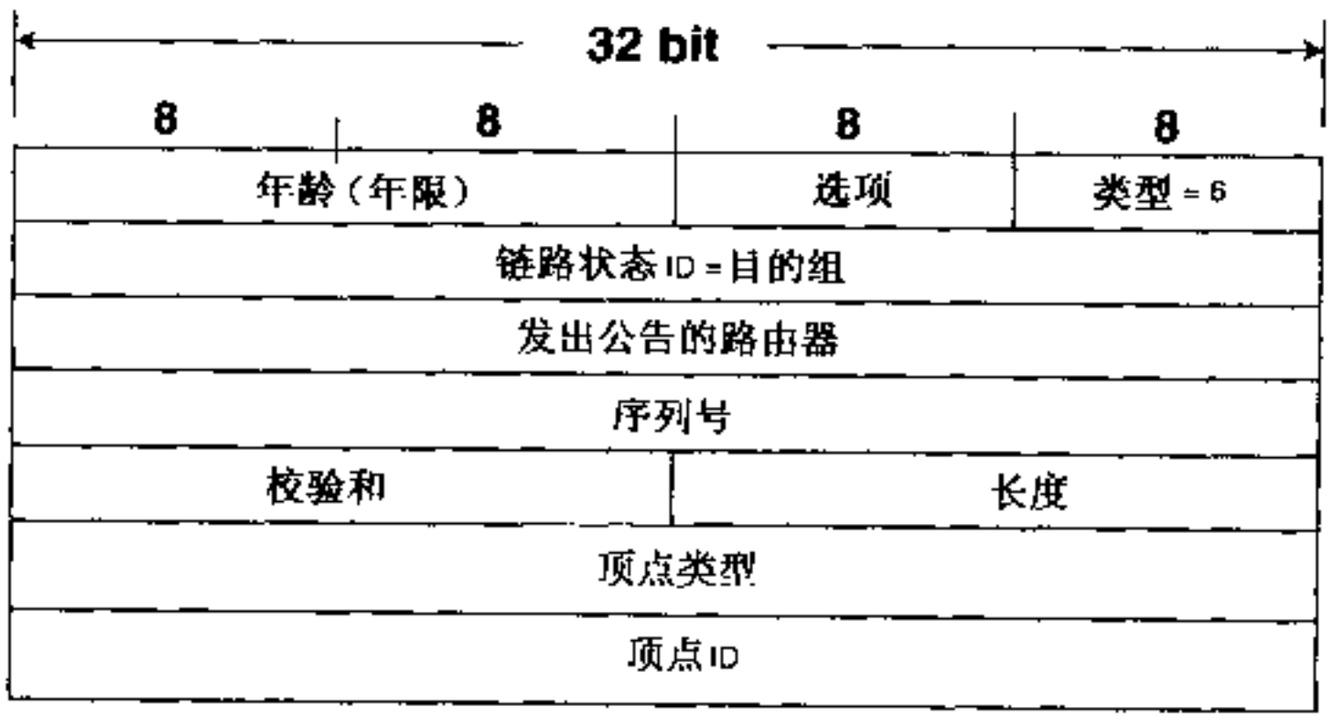


图 5-36　MOSPF 组员资格 LSA 的格式

组员资格 LSA 中的各项定义如下：

- 链路状态 ID 为进行宣告的多播组的地址。
- 宣告路由器总是多路访问网络中 MOSPF 指定路由器的路由器 ID，因为只有指定路由器才能产生类型 6 的 LSA。
- 顶点类型定义了目的地是一个路由器(类型 1)，还是一个转接的网络(类型 2)。如果发起路由器运行了某些需要它成为一个多播组的成员的应用时，类型 1 被定义。转接网络指的是发起路由器直连到一个网络上，通过这个网络，包才能到达别的组员处。

- 顶点 ID 是发起路由器的路由器 ID。

2. 扩展路由器 LSA 格式

图 5-37 显示了一个经扩展、支持 MOSPF 的路由器 LSA(类型 1)的格式。其格式与第 1 卷中的图 9-55 是一样的，只是 rtype 项中有一个 W 比特。这个 W 比特由区域间和 AS 间的多播前转器设置，用于告诉其他 MOSPF 路由器它们是通配多播前转器。

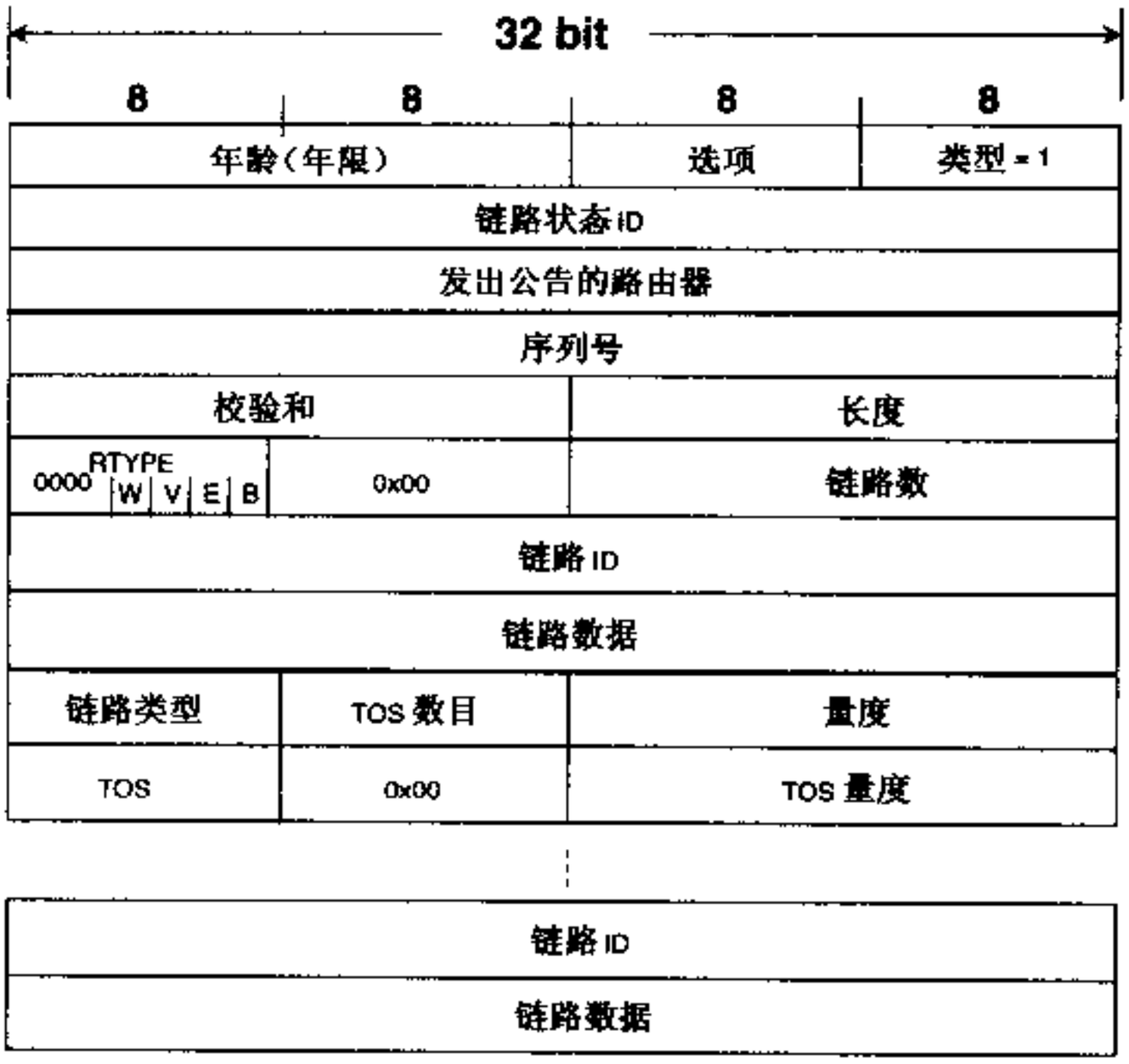


图 5-37　扩展路由器 LSA 格式，在 rtype 项中加了一个 W 比特来支持 MOSPF

3. 扩展的可选项的格式

可选项如图 5-38 所示，是 OSPF Hello 和 Database Description 包及所有 LSA 头中的一部分。这一项中其他标识的描述请参阅第 1 卷第 9 章。与本章有关的标识是 MC 比特。当这一比特被设置时，它表明发起路由器有多播的能力。

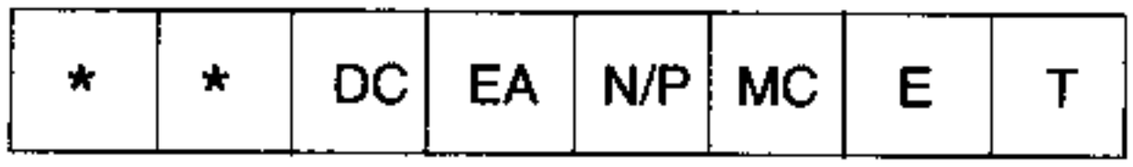


图 5-38　可选项的格式

Hello 包里的 MC 比特只是一个多播能力的信号。如果有两个路由器，一台设置了 MC 比特，而另一台没有，那么它们仍能保持邻接的关系。MC 比特的真正用处在于 Database Description 包和 LSA 中的使用。

在数据库同步时，一台 MOSPF 路由器的邻居发送的 DD 包中若设置了 MC 比特，那么这台路由器会向邻居发送类型 6 的 LSA。同样，只有设置了 MC 比特的 LSA 才能用于 MOSPF SPF 的计算。

5.6　基于核心的树(CBT)的操作

DVMRP 与 MOSPF 有两个共同的局限性。首先，它们都是密集模式的协议，不能在稀疏拓扑中有很好的扩展性。也就是说，当网络中组员的数目相对于总的主机数要少很多，且组员分布于整个网络中时，DVMRP 与 MOSPF 都会消耗掉大量的网络资源才能到达组员。很多资源消耗在用于计算的开销和为每一个以多播源为根的树保持状态。第二，两个协议均受限于以单播的路由协议来判断多播树——DVMRP 是基于 RIP 的协议，而 MOSPF 是基于 OSPF 的。基于核心的树(CBT)则与它们不同，它是一个与协议无关、稀疏模式的、共享树协议。

与协议无关意味着 CBT 可以用下层的任何路由协议来发现源和其他 CBT 路由器，并建立自己的树。除了提高了灵活性，采用现有的路由协议，而不是为了多播加入一个新协议，从而也减小了开销。CBT 树根植于一个核心 CBT 路由器，而非源网络。这个核心可被放置在网络中的任何位置，许多个组的树都可以根植在一个核心上，以使协议更适于稀疏多播拓扑。

当前有 3 个版本的 CBT。RFC 2189[11] 中描述了 CBTv2，使 CBTv1 作废。CBTv3 正在提出。这 3 个版本都是试验性的，没有一个被广泛使用。CBTv2 和 CBTv3 均不和前一版本后向兼容。这一章主要集中在 CBTv2；当使用术语“CBT”时，它专指这一版的协议。

5.6.1　CBT 基础

CBT 采用了 9 种消息类型：

- JOIN_REQUEST
- JOIN_ACK
- ECHO_REQUEST
- ECHO_REPLY
- QUIT_NOTIFICATION
- FLUSH_TREE
- Candidate Core Advertisement
- Bootstrap
- HELLO

做为“CBT 指定路由器”一节中唯一的特例，所有 CBT 消息均发向保留的多播地址 224.0.0.15(见表 5-1)。这些消息的 TTL 设为 1，意味着所有 CBT 消息不能通过一跳接一跳的方式在多播域中传送。每一消息类型的格式在“CBT 消息格式”一节中详细描述。

同 IP 多播的其他协议一样，CBT 通过 IGMP 组员资格消息被告知一台连接的主机要加入一个组。CBT 采用显式加入，所以当一台 CBT 路由器必须为一特定组前转包时，它必须把自己接入到组的多播树中。路由器首先检查它的单播路由表，确定对一特定组其核心的位置，然后向上游核心方向发送一个 JOIN_REQUEST 消息，这个消息中有 3 个重要的信息：

- 多播组的地址

- 核心的地址
- 消息发起者的地址

> 注：　路由器如何知道到哪里找到核心会在下一节中讲述，这一节理所应当称为“寻找核心”。

当下一跳的路由器收到 JOIN_REQUEST 消息时，它检查组地址和核心地址。基于这些信息，路由器决定它所应担当的角色：

- 作为核心的路由器
- 连接到组的多播树上
- 不是核心，也不是树上的路由器

如果路由器是核心，或是树上的路由器，那么它向 JOIN_REQUEST 的消息发起者送一个 JOIN_ACK 消息，表明消息发起者已经成功地加入到了树中。这台路由器把收到 JOIN_REQUEST 消息的接口加到前转表中，开始为该组在该接口上前转数据。

如果路由器不是核心，也不是树的组成部分，则它必须也加入到树中。路由器通过查询自己单播的路由表，找出核心的位置，并向其发送一个 JOIN_REQUEST 消息的备份。它启动一个瞬时加入状态，在这个状态时，路由器记录多播组和收到 JOIN_REQUEST 消息的接口，以及发送 JOIN_REQUEST 消息的接口。这时再启动一个计时器，如果在 7.5s 中没有收到 JOIN_ACK (瞬时加入的时长)，那么这个瞬时加入状态会被删除，加入被认为不成功。

在 CBT 的用法中，到核心的上游接口为父接口，到下游组员的接口为子接口。同样，上游的邻居为父路由器，而下游的邻居为子路由器。一旦收到 JOIN_ACK 消息，树就建立起来。子路由器每 60s 向其父路由器发送一个 ECHO_REQUEST 消息，ECHO_REQUEST 消息只包含发起子路由器的地址。父路由器用 ECHO_REPLY 消息回应，这个消息里列出了这个父路由器负责在这条链路上进行包前转的所有组。

如果在 70s 之内没有收到 ECHO_REPLY 消息，父路由器就被宣布为不可到达。同样，如果在过去的 90s 内所有的 ECHO_REPLY 消息中都没有列出一个特定的组，这个组就被宣布为无效。这时，子路由器向上游父路由器发送一条 QUIT_NOTIFICATION 消息，也向下游它的子路由器发送一条 FLUSH_TREE 消息。FLUSH_TREE 消息中列出了所有成为不合法组的地址，收到消息的子路由器刷新所有关于合法组的路由信息，这些子路由器这时再向它们的子路由器发送相应的 FLUSH_TREE 消息。这一过程将一直进行到树的所有树枝被删除为止。

QUIT_NOTIFICATION 消息也用于剪除。如果路由器通过 IGMP Leave Group 消息知道与它连接的子网中没有某个组的组员，它就向其父路由器发送一个 QUIT_NOTIFICATION 消息，列出要剪除的组的地址。如果父路由器既没有连接着的组员，也没有这个组的子接口，它也向上游发送一个 QUIT_NOTIFICATION 消息。这个树枝将一直剪到一个连在树上的路由器，或者到核心。

5.6.2　寻找核心

对于 CBT 路由器建立到核心的树的先决条件就是让路由器知道什么路由器是核心。一种可以满足这需求的方法是所有路由器事先配置好每个组的核心路由器的地址。这种方法对于

小型的多播网络来说很有效，它能提供良好的网络控制，不过这样的管理需求不能扩展到大型的网际网络中。

另一种方法就是用自举机制。采用这种方法，一组 CBT 域里的路由器都配置成候选者核心路由器。这些路由器交换核心候选者的消息，它们当中的一台根据其优先权，或在优先权相等时有最大 IP 值而被选为自举路由器(BSR)。其他的核心候选者路由器每 60s 向 BSR 单播一个候选核心消息。根据这些核心候选者的消息，BSR 组成一个核心候选者集(CC-set)并通过 Bootstrap 消息向域中所有的 CBT 路由器进行宣告。当一台路由器通过 IGMP 请求加入一个组时，它根据这个 CC-set 运行一个 hash 算法，再为这个组决定一个正确的核心路由器。

CBT 和 PIM-SM 都采用了相同的自举协议。因为这一章更着重于后者的讲述，所以自举机制在此只做概述，在“与协议无关多播，稀疏模式(PIM-SM)”一节中将有详细的描述。

5.6.3　CBT 指定路由器

CBT 用 HELLO 消息选出多播网络中的指定路由器。CBT DR 采用的基本原理与 DVMRP 指定前转器和 MOSPF DR 是一样的。如图 5-39 所示，因为 CBT 在前转包时不用 RPF 检查，所以 DR 在有多路径到核心时防止环路的出现是很重要的。

第一个 CBT 接口都配置一个取值为 0～255 的优先权，这个值也会出现在 HELLO 消息中。一个 1～254 的值表示这个路由器可以成为 DR，值越小表示优先权越大——就是说，优先权为 10 的路由器比优先权为 20 的路由器更符合条件。若优先权设为 0，这台路由器就是 DR。

当 CBT 路由器在一条多播链路上第一次启动时，它先后发送两个 HELLO 消息来宣告它的存在和它的优先权。这时，这台路由器监听 HELLO 消息，这个消息中会导致 3 个结果：

- 听到网络上有一个优先权值更低的 HELLO 消息；
- 所有听到的 HELLO 消息的优先权的值都比较高；
- 没有听到网络中有 HELLO 消息。

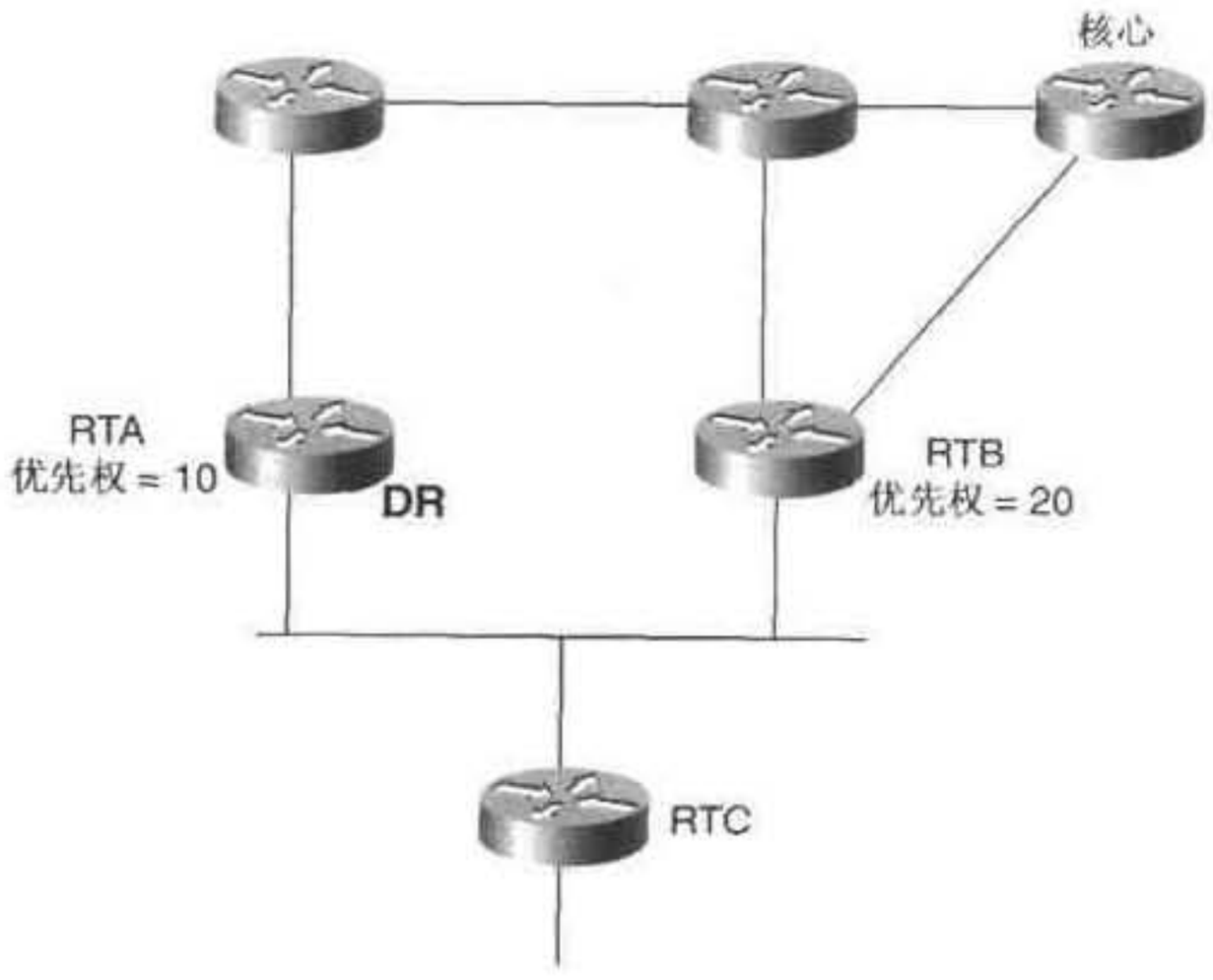


图 5-39　CBT 在多路访问的网络中选出一台指定路由器，以管理到核心的上游通路

CBT 最基本的特性在这一行为中体现，也就是 CBT 采用了双向树。换一句话讲，多播流量可以沿树向下游传，也可以沿树向上游从，从一个源到核心。这与其他共享树协议不同，PIM-SM 就是一个单向树。

当然，不是所有的源都是组员。因此，CBT 也必须有一种机制来适应非组员多播源。这一机制是一个简单的 IP-in-IP 隧道，图 5-41 显示了这一机制。在这里，同样的主机为组 1 产生多播流量，不过主机本身不是组的成员。当它的本地路由器收到流量时，它建立一个到核心的隧道(假设这台路由器运行 CBT，因此知道核心的地址)。多播流量用单播发给核心，核心再把流量送到多播树中。

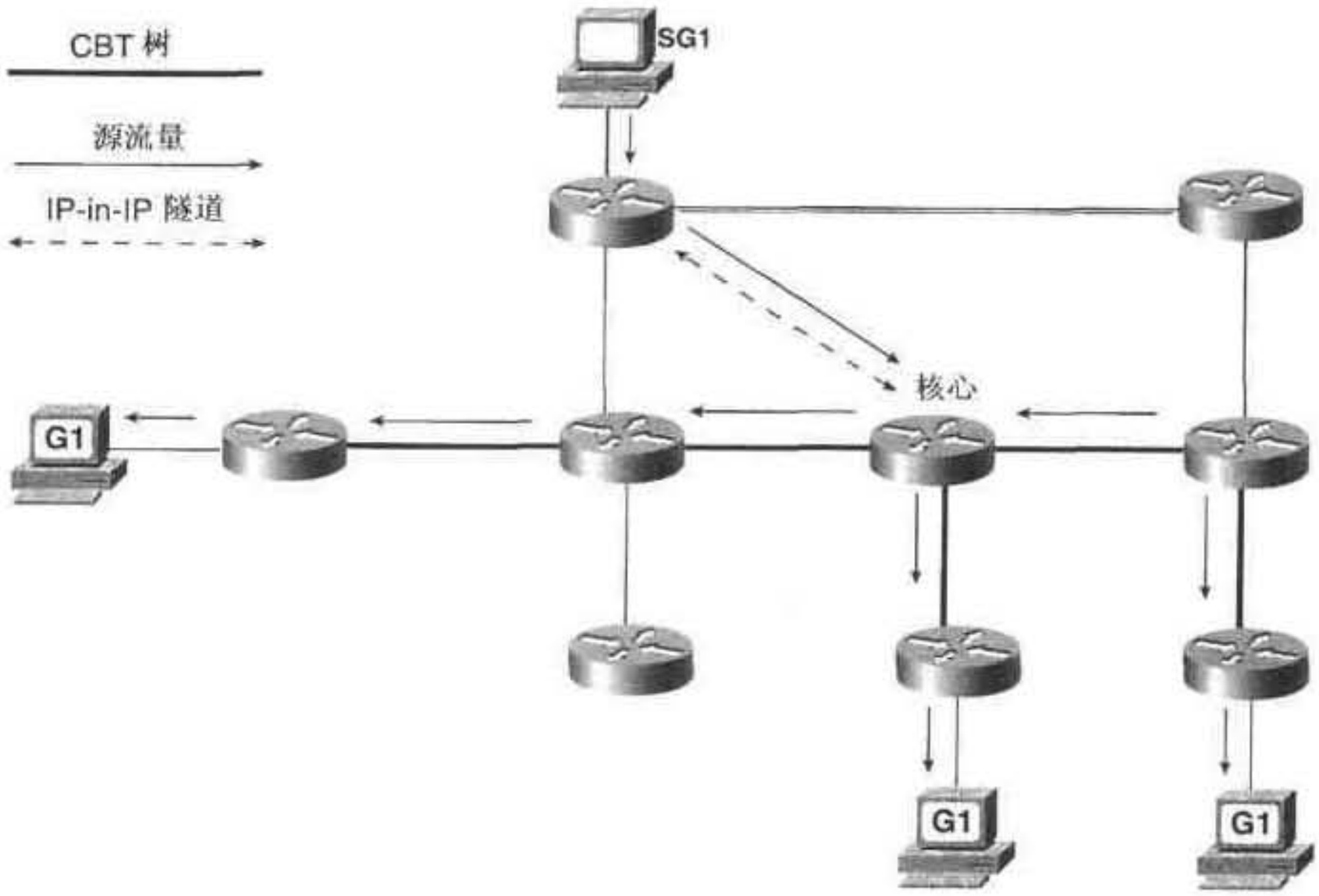


图 5-41　非组员多播源的机制

5.6.5　CBT 消息格式

CBT 消息以 IP 头封装，它的协议号是 7。除了前面说到的单播的特殊情况外，包传送的目的地址为 224.0.0.15，TTL 设为 1。图 5-42 显示了 CBT 消息的包头。

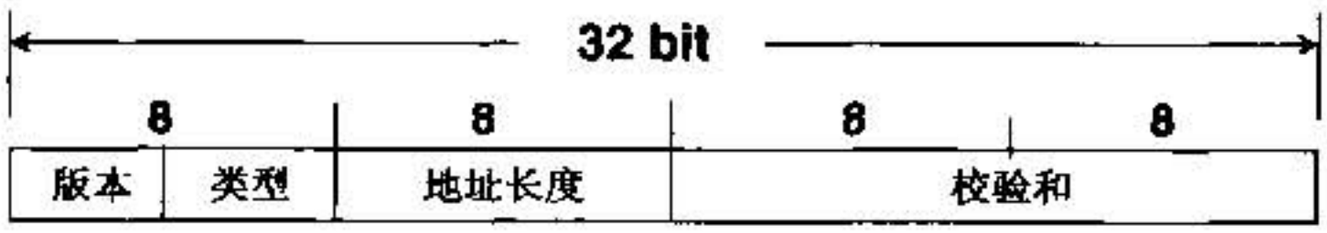


图 5-42　CBT 消息头的格式

CBT 消息包头中各项的定义如下：

- 版本定义了 CBT 的版本号。这一节专门讲述版本 2，除此之外还有一个已作废的版本 1 和一个正在提出的版本 3。
- 类型规定了消息的类型。表 5-10 列出了 CBT 消息使用的类型号。

表 5-10　　CBT 消息类型

类　型	消　息
0	HELLO
1	JOIN_REQUEST
2	JOIN_ACK
3	QUIT_NOTIFICATION
4	ECHO_REQUEST
5	ECHO_REPLY
6	FLUSH_TREE
7	Bootstrap
8	Candidate Core Advertisement

- 地址长度规定了相关消息中包含的单播或多播的地址长度，以字节计。
- 校验和是标准的整个 CBT 消息反码和的反码。

1. CBT HELLO 消息格式

HELLO 消息的格式在图 5-43 中画出，这个消息用于选举多播网络中的指定路由器。它们也由 DR 每隔 60s 发送一次，以保持存活。

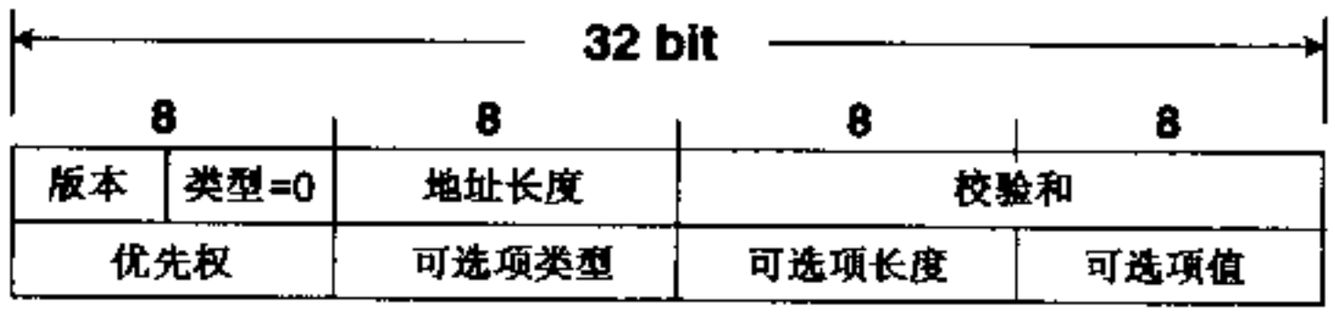


图 5-43　CBT HELLO 消息格式

CBT HELLO 各项的定义如下：

- 优先权取值为 0～255。值 1～254 表示由发起路由器变为 DR 的“合格程度”，优先权值越低就越合格。值为 0 表示这个 HELLO 消息由 DR 产生。当路由器首次启动时，它发送两个有它的优先权的 HELLO 消息，以触发一个 DR 选举(甚至在已有 DR 存在的情况下)，若任何路由器的优先权值都比它高(不如它合格)，则不进行响应。若一台路由器有更低的优先权值(更合格)，则用含有自己的优先权的 HELLO 消息应答。新的路由器若没有收到应答的 HELLO 消息，就成为 DR，或它知道有其他更低优先权值的路由器已成为 DR，于是停止发送 HELLO 消息。
- 可选项类型定义了可选项值中的可选项类型。CBTv2 只定义了一种选项——边界路由器(BR)，它在本章前没有定义过。一个 BR 是连接 CBT 域与其他多播域的路由器。BR 产生的 HELLO 消息的可选类型为 0。
- 可选项长度定义了以字节计的可选项值的长度。由 BR 产生的 HELLO 消息的可选长

度为 0。

• 可选项值是一个变长的项，用于承载可选项值。由 BR 产生的 HELLO 消息的可选项值为 0。

2. CBT JOIN_REQUEST 消息格式

路由器收到 IGMP Membership Report 消息后，希望能接入到 CBT 树中，加入一个特定的组，于是发出一个 JOIN_REQUEST 消息，其格式如图 5-44 所示。

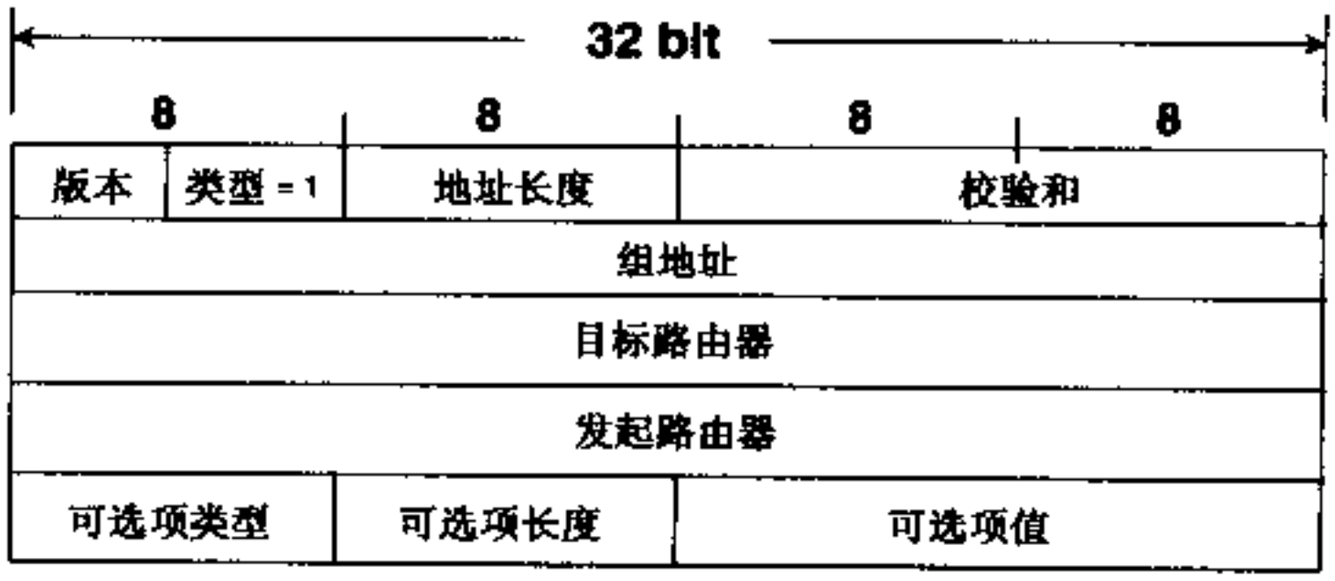


图 5-44　CBT JOIN_REQUEST 消息格式

CBT JOIN_REQUEST 消息中各项的定义如下：

- 组地址为要加入的组的多播地址。
- 目标路由器为这个组的核心路由器的地址。
- 发起路由器为产生这个消息的路由器的地址。
- 可选项类型、可选项长度、可选项值与 HELLO 消息中定义相同。

3. CBT JOIN_ACK 消息格式

核心路由器或接入多播树的路由器以 JOIN_ACK 消息来响应 JOIN_REQUEST 消息，其格式在图 5-45 中显示。这个消息发回给发起 JOIN_REQUEST 消息的路由器，表示它们已成功地加入到多播树中。

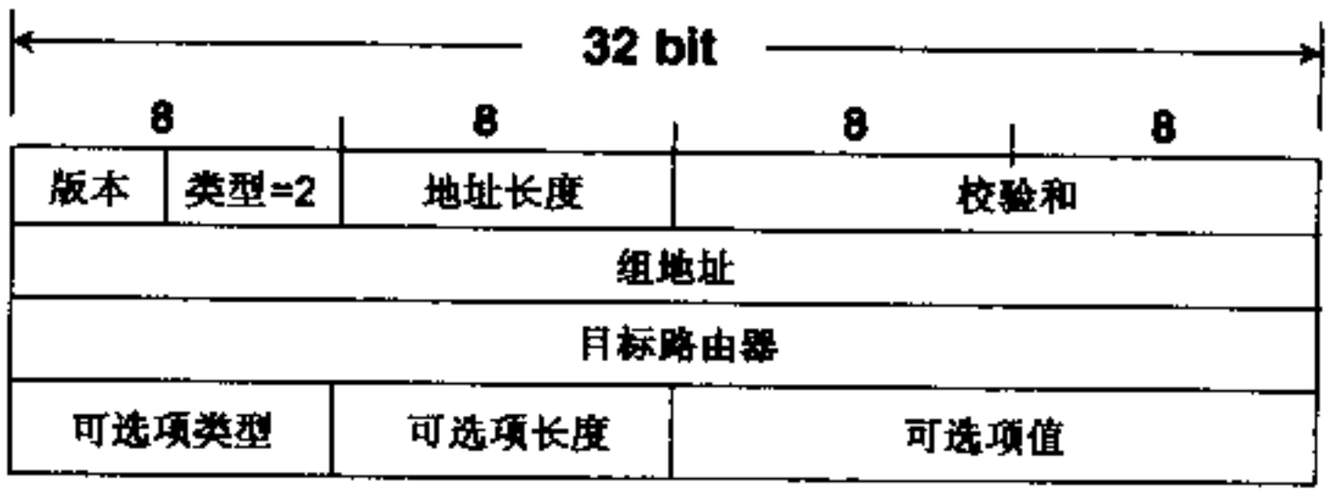


图 5-45　CBT JOIN_ACK 消息格式

CBT JOIN_ACK 消息的各项定义如下：

- 组地址为加入组的多播地址。
- 目标路由器为 JOIN_ACK 发向的那个路由器的地址。这个地址可以在 JOIN_REQUEST 消息中的发起路由器项中找到。
- 可选项类型、可选项长度、可选项值与 HELLO 消息中定义相同。

4. CBT QUIT_NOTIFICATION 消息格式

QUIT_NOTIFICATION 消息的格示如图 5-46 所示，这个消息是向父路由器发送的(直接

发到上游)，以请求从一个特定的多播树中剪除。当一台路由器收到 IGMP Leave Group 消息，查询超时或从它的子路由器收到 QUIT_NOTIFICATION 消息，确定没有这个组的其他下游接口，便产生一个 QUIT_NOTIFICATION 消息。

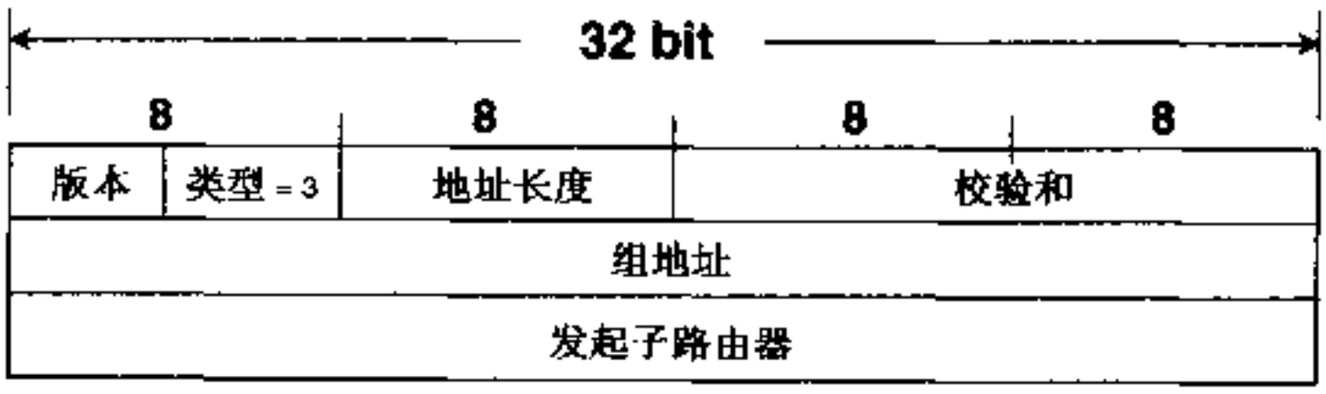


图 5-46　CBT QUIT_NOTIFICATION 消息的格式

CBT QUIT_NOTIFICATION 消息各项定义如下：

- 组地址为退出的组的多播地址。
- 发起子路由器为发起这个消息的路由器的地址。

5．CBT ECHO_REQUEST 消息格式

子路由器有义务维护到父路由器的链路状态。子路由器每 60s 发送 ECHO_REQUEST 消息来完成这个任务。如图 5-47 所示，ECHO_REQUEST 消息仅由 CBT 头与发起的子路由器的地址组成。

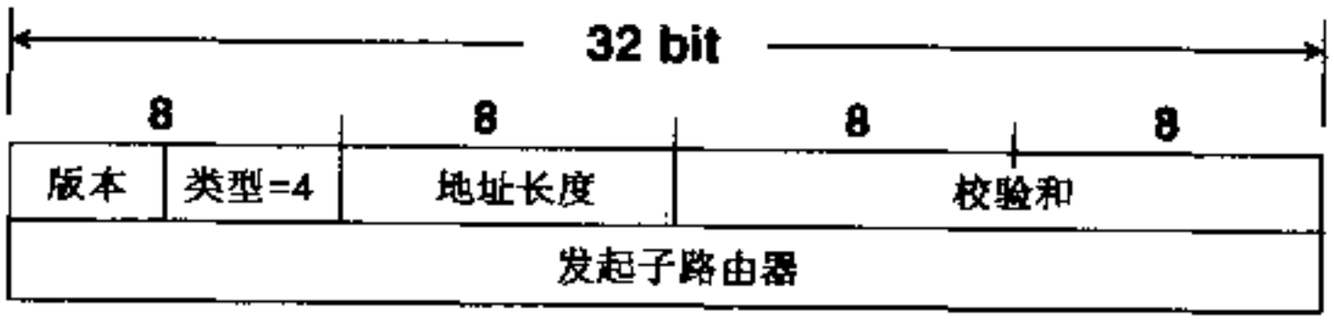


图 5-47　CBT ECHO_REQUEST 消息的格式

6．CBT ECHO_REPLY 消息格式

父路由器以 ECHO_REPLY 消息来响应了路由器的 ECHO_REQUST 消息，其格式如图 5-48 所示。这两个消息构成了父子路由器间的链路保持存活的机制。

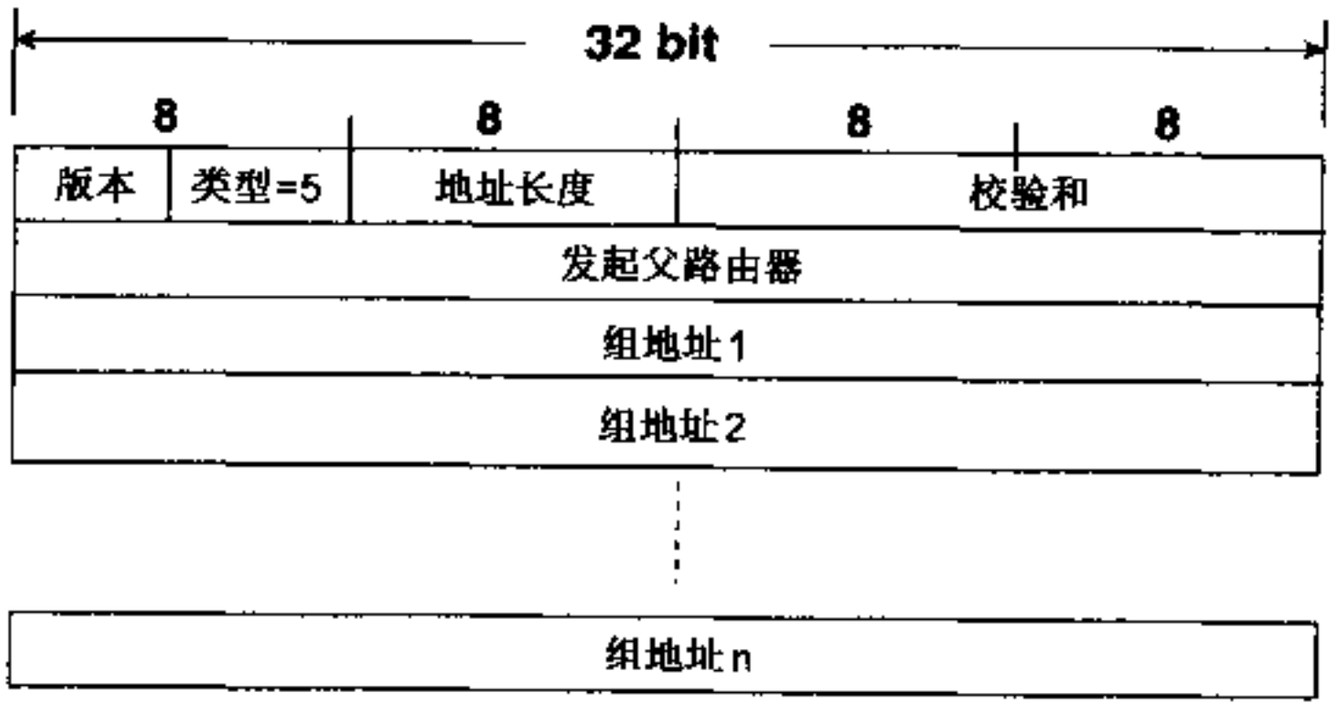


图 5-48　CBT ECHO_REPLY 消息的格式

CBT ECHO_REPLY 各项的定义如下：

- 发起父路由器为消息发起者的地址。
- 组地址为一个或多个列出组地址的项，父路由器为子路由器前转这些组的包。

7. CBT FLUSH_TREE 消息格式

FLUSH_TREE 消息的格式如图 5-49 所示。当 CBT 路由器失去与父路由器的连接，它向下游子路由器发送这个消息。子路由器收到 FLUSH_TREE 消息后，将清除所有列在这个消息中的多播组的相关信息。

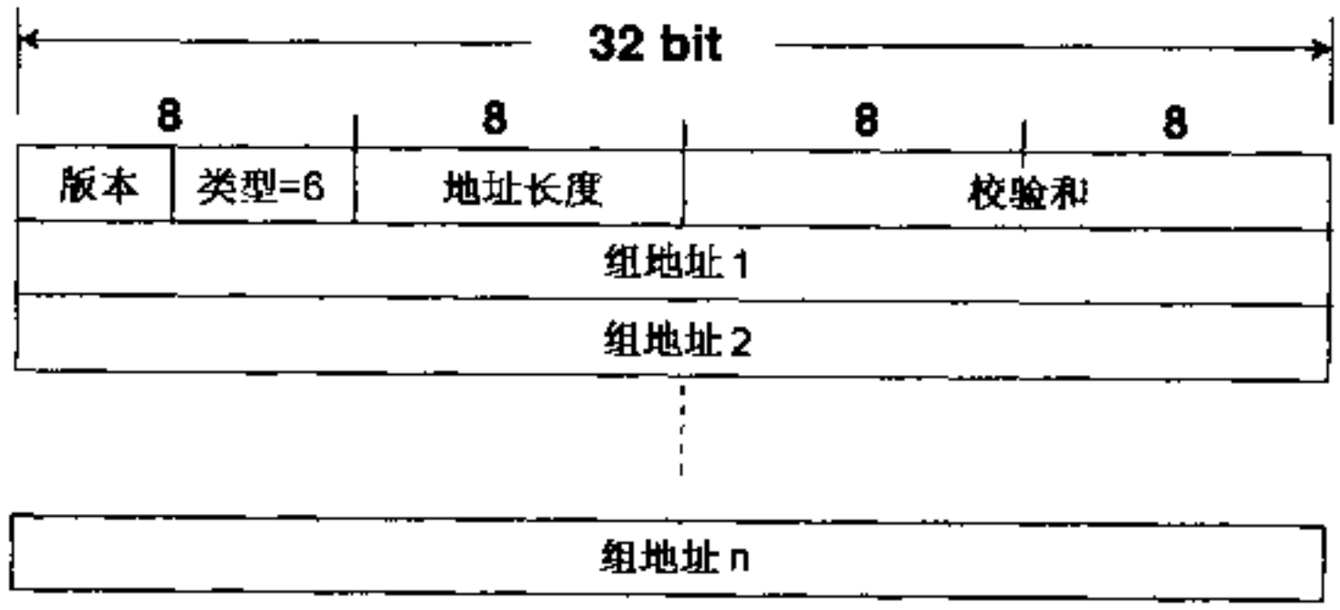


图 5-49　CBT FLUSH_TREE 消息的格式

这个消息中的组地址为一个或多个组地址项，发起路由器已经同这些地址中断联系，子路由器收到这些地址后，要清除前转状态。

5.7　与协议无关的多播(PIM)的介绍

如果你是一个 CCIE 的候选人，并且知道 Cisco 并不支持或者部分支持(DVMRP)前面章节中介绍的协议，这对于你可能是个打击。可是每一种协议都让你了解到哪些是多播选路所需要的，而哪些是不需要的。

DVMRP 同单播的距离向量协议有共同的特征，那就是很容易实现——所需的工作仅仅是把它启动。不过其简单性将带来很大的开销，除了在小型的、宽带的网络中有较密集的组员这种环境下，这个协议都会有扩展性的问题。

MOSPF 把链路状态的优点带入了路由表中，不过代价是设计复杂性的增加。它采用显式加入，避免了 DVMRP 混乱的规则：路由器不仅不为一个组前转包，却还要记住(保持状态)它不为这个组前转。这样可以减少对网络资源的冲击。不过，MOSPF 基于源的树使协议不适于稀疏的拓扑。由于这个局限，许多(不是大多数)网络设计者不愿在 MOSPF 较为复杂的拓扑需求上花费很多。

DVMRP 是一个“完备的”协议，它用自己内建的协议来定位建立和维护多播树所需的单播 IP 地址。它完全不依赖于任何下层的单播路由协议。不过独立的代价就是占用网络资源来收集单播路由表中的信息。

MOSPF 是一个单播协议的扩展，所以不需要单独的单播协议，它不能独立于 OSPF 运行。

CBT 与协议真正无关。它通过查询单播的路由表找到单播的目的地址，而不关心什么单

播协议来维护这个路由表。CBT 在稀疏拓扑中也可以扩展，虽然核心的设置必须仔细计划来最小化次优化路由的影响和流量瓶颈。这时 CBT 卡在“第 22 条军规”中：因为它还不成熟，所以实际应用对它的兴趣很有限；这个协议不成熟是因为它在实际中有限的应用。CBT 不太可能受到主流的接受，除非它的设计者能引入比当前受到人们喜爱的、更灵活的 PIM-SM 更多的优点。

PIM 是 Cisco IOS 完全支持的一种多播路由协议(DVMRP 在 PIM 可以连接到 DVMRP 网络时才被支持)。

同 CBT 一样，也正如它的名称所声称的一样，PIM 是与协议无关的。也就是说，它采用单播的路由表来定位单播地址，而不用关心路由表如何知道这些地址。

有一组 PIM 消息格式的标准列表，某些消息只在 PIM-DM 中使用，而某些只在 PIM-SM 中使用。所有消息格式，包括那些只在 PIM-DM 中使用的，将在最后一节“与协议无关多播，稀疏模式”中描述。

当前 PIM 的版本为 PIMv2，版本 1 的 IP 包头中协议号为 2(IGMP)，采用多播地址 224.0.0.2。PIMv2 在 Cisco IOS 11.3(2)T 中就开始支持，并使用它自己的协议号 103 和保留多播地址 224.0.0.13。当 PIMv2 路由器与 PIMv1 通信时，它自动设置这个接口为 PIMv1。

5.8　与协议无关多播，密集模式(PIM-DM)的操作

到本书写作时，还没有 RFC 来描述 PIM-DM。不过，它在 Internet 草案[12]中有描述。除了有共同的消息格式外，你可能会发现 DVMRP 比 PIM-SM 更接近于 PIM-DM。

5.8.1　PIM-DM 基础

PIM-DM 使用了 5 个 PIMv2 消息：

- Hello
- Join/Prune
- Graft
- Graft-Ack
- Assert

PIMv2 路由器用 Hello 消息来发现邻居。一旦 PIMv2 路由器(PIM-DM 或 PIM-SM)启动，它就周期性地在每个配置 PIM 的接口上发送 Hello 消息。PIMv1 路由器有相同的功能，但它们采用 Query 消息。Hello 消息(或 Query 消息)中有一个*保持时间*，这个时间参数定义了邻居等待下一个消息的最长时间，如果没有等到这个消息，邻居就宣布这个发起路由器已经死亡。PIMv2 的 Hello 消息和 PIMv1 的 Query 消息的间隔在 Cisco IOS 中默认为 30s。它们可以用 **ip pim query-interval** 命令加以改变。保持时间自动设为 Hello/Query 消息间隔的 3.5 倍。

例 5-3 表明了 **debug** 命令对 PIM 接收和发送消息的捕获过程。注意到路由器有 PIMv1 和 PIMv2 两个邻居，分别由关键词 **Hello** 和 **Router-Query** 来区分。另外也应该注意到路由器在接口 E0 上发送 Hello 消息，但这个接口既没有收到 Hello 消息也没有收到 Query 消息，这就表明这个子网上没有 PIM 邻居。

例 5-3　Steel 路由器在接口 E0、E1 与 S1.708 上发送查询消息，在接口 E1、S1.708 上接收对方发来的信号

```
Steel#debug ip pim
PIM debugging is on
Steel#
PIM: Received v2 Hello on Ethernet1 from 172.16.6.3
PIM: Received Router-Query on Serial1.708 from 172.16.2.242
PIM: Send v2 Hello on Ethernet1
PIM: Send v2 Hello on Ethernet0
PIM: Send Router-Query on Serial1.708  (dual PIMv1v2)
PIM: Received v2 Hello on Ethernet1 from 172.16.6.3
PIM: Received Router-Query on Serial1.708 from 172.16.2.242
PIM: Send v2 Hello on Ethernet1
PIM: Send v2 Hello on Ethernet0
PIM: Send Router-Query on Serial1.708  (dual PIMv1v2)
PIM: Received v2 Hello on Ethernet1 from 172.16.6.3
```

在例 5-4 中，使用 **debug ip packet detail** 命令可以进一步查看 PIM 消息。这里，你可以看到 PIMv2 的消息被送到 224.0.0.13，协议号为 103，而 PIMv1 的消息则被送到 224.0.0.2，协议号为 2。

例 5-4　通过这个 **debug** 命令可以获得 PIMv1 与 PIMv2 多播的目的地址与协议号

```
Steel#debug ip packet detail 101
IP packet debugging is on (detailed) for access list 101
Steel#
IP: s=172.16.6.3 (Ethernet1), d=224.0.0.13, len 38, rcvd 0, proto=103
IP: s=172.16.2.241 (local), d=224.0.0.2 (Serial1.708), len 35, sending
broad/multicast, proto=2
IP: s=172.16.2.242 (Serial1.708), d=224.0.0.2, len 32, rcvd 0, proto=2
IP: s=172.16.6.1 (local), d=224.0.0.13 (Ethernet1), len 30, sending broad/multicast,
proto=103
IP: s=172.16.5.1 (local), d=224.0.0.13 (Ethernet0), len 30, sending broad/multicast,
proto=103
IP: s=172.16.6.3 (Ethernet1), d=224.0.0.13, len 38, rcvd 0, proto=103
IP: s=172.16.2.241 (local), d=224.0.0.2 (Serial1.708), len 35, sending
broad/multicast, proto=2
IP: s=172.16.2.242 (Serial1.708), d=224.0.0.2, len 32, rcvd 0, proto=2
IP: s=172.16.6.1 (local), d=224.0.0.13 (Ethernet1), len 30, sending broad/multicast,
proto=103
IP: s=172.16.5.1 (local), d=224.0.0.13 (Ethernet0), len 30, sending broad/multicast,
proto=103
```

在例 5-5 中，使用 **show ip pim neighbor** 命令可以查看 PIM 的邻居表。

例 5-5　PIM 邻居表中记录着例 5-3 中探测到的邻居

```
Steel#show ip pim neighbor
PIM Neighbor Table
Neighbor Address  Interface          Uptime    Expires   Ver  Mode
172.16.6.3        Ethernet1          01:57:22  00:01:29  v2   Dense   (DR)
172.16.2.242      Serial1.708        04:55:56  00:01:05  v1   Dense
Steel#
```

当源开始发送多播包，PIM-DM 用扩散与剪除的方式建立多播树。由于每一个 PIM-DM 路由器收到多播包后，就在它的多播前转表里增加一个条目。最终，包扩散到所有的叶路由器上——也就是所有没有下游邻居的路由器。如果一个叶路由器收到一个多播包，但它没有连接的组员，路由器必须把自己从多播树中剪除。这个过程是通过向上游邻居发送 Prune 消息来实现的。Prune 消息的目的地址为 224.0.0.13，上游路由器的地址封装在这个消息中。如果上游邻居没有这个组的已连接组员，也没有下游邻居或收到任何下游邻居的 Prune 消息，它就向自己的上游邻居发送一个 Prune 消息。

参考后面的 PIMv2 的消息类型，你会看到没有 Prune 这个消息类型。但是有一个 Join/Prune 消息，这是一个单独的消息，消息中分别列出要加入的组和要剪除的组。本节中还用“剪除消息”和“加入消息”来加以区分，不过你要知道 Prune 消息就是 Join/Prune 消息中在剪除部分列出的一组地址，同样，Join 消息就是 Join/Prune 消息中有在加入部分列出的一组地址。

例 5-6 显示了一个多播组 239.70.49.238 的前转表。你可以看到(S,G)对，中源地址为 172.16.1.1。路由器查询自己的单播路由表，找出到达源的上游接口，这个接口是 S1.708，以及到达源的上游路由器的地址，这个地址为 172.16.2.242。这个信息输入到多播前转表中，用于 RPF 检查。同 DVMRP 一样，如果除 S1.708 外的其他接口收到源地址为 172.16.1.1，目的地址为 239.70.49.238 的包，RPF 检查将会失败，这个包被丢弃。

> 注：　例 5-6 中没有显示与这个组相关的所有前转表中的信息；某些信息为了表达清楚，给删除了。第 6 章中有一个更详细的前转表。

有两个计时器与(S,G)对相关联。第一个计时器表明这个条目在表中有多长时间了，第二个计时器表明这个条目将过期的时间。如果在 2 分 59 秒内没有一个包通过这个(S,G)对前转，这个条目就会被删除。

> 注：　Cisco IOS 软件的超时计时器为 2.5 分钟，Internet 草案中推荐的是 3.5 分钟。

例 5-6　show ip mroute 命令显示多播前转表

```
Steel#show ip mroute 239.70.49.238
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

 (172.16.1.1, 239.70.49.238), 01:56:27/00:02:59, flags: CT
  Incoming interface: Serial1.708, RPF nbr 172.16.2.242
  Outgoing interface list:
    Ethernet1, Prune/Dense, 01:40:23/00:00:39
    Ethernet0, Forward/Dense, 00:00:46/00:00:00

Steel#
```

例 5-6 中的条目有两个相关的标识，第一个标识 C 表示有一个组员直连在路由器的子网中。第二个标识 T 表示路由器是最短路径树(SPT)中的一员，用 CBT 的说法，就是“在树上”。

注：　PIM 称基于源的树为最短路径树，共享树为会聚点树(Rendezvous Point Tree RPT)。SPT 是一个说明性的名称，再随后的一章中，你会发现有时这些树可以跨跃比 RPT 更短的路径到达源。

在例 5-6 中输出接口列表中有两个接口，第一个接口 E1 在剪除的状态，是密集模式，因此可以知道这个接口的下游邻居发送了 Prune 消息。计时器显示，这个接口启动了 1 小时 40 分钟 23 秒，剪除状态在 39s 后结束。当收到一个 Prune 消息，一个 210s 的计时器开始启动。这个剪除状态一直维持着，直到这个计时器超时，这时状态变为前转状态并且开始将包向下游转发。此时，仍然要靠下游路由器向上游邻居发送 Prune 消息；这与 DVMRP 非常相似。

第二个接口，E0 处于前转状态。回忆例 5-3 中路由器在 E0 上发送 Hello 消息，可是没收到邻居的 Hello 消息。根据例 5-3 和例 5-6 中的信息，你知道路由器通过 E0 前转包是因为这个子网上有一个组员。例 5-7 证实这一结论，注意到例 5-6 中与接口相关的有一个启动时间，但没有超时的时间。这是因为对于相邻的状态不需要超时。而当 IGMP 通知子网中没有组员，或例 5-7 中的计时器计时到 0，路由器把这个接口从前转表中删除时，则需要有计时器。

例 5-7　show ip igmp group 命令显示 IGMP 组员表中连接的组员

```
Steel#show ip igmp group 239.70.49.238
IGMP Connected Group Membership
Group Address    Interface              Uptime    Expires   Last Reporter
239.70.49.238    Ethernet0              01:52:23  00:02:34  172.16.5.2
Steel#
```

例 5-8 显示了源方向上相邻的上游路由的前转表。通过接口 S1.803 和上游邻居 172.16.2.254，RPF 对(172.16.1.1,239.70.49.238)进行检查，这里只有一个下游接口。把这个条目的标识与例 5-6 中条目的标识进行比较，可以看到路由器在最短路径树上，不过它没有直连的组员。

例 5-8　这个条目的标帜表明这路由器在 SPT 上，但它没有直连的组员

```
Nickel#show ip mroute 239.70.49.238
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set
Timers: Uptime/Expires

 (172.16.1.1/32, 239.70.49.238), uptime 02:05:23, expires 0:02:58, flags: T
  Incoming interface: Serial1.803, RPF neighbor 172.16.2.254
  Outgoing interface list:
    Serial1.807, Forward state, Dense mode, uptime 02:05:24, expires 0:02:34

Nickel#
```

注：　例 5-8 的输出与前面前转表的输出有一些不同，这是因为 Cisco IOS 的软件版本不同。不过你可以看到的信息是一样的。

向上游继续一步，例 5-9 中显示了这个组的另一个前转表，标识显示这路由为“Connected”，不过在这个例子中连接的不是组员。注意到输入的接口为 E0/0，它的 RPF 邻居地址为 0.0.0.0，这表示它所连接的设备就是这个组的源。

例 5-9　路由器连接到源 172.16.1.1，RPF 邻居地址为 0.0.0.0

```
Bronze#show ip mroute 239.70.49.238
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop, State/Mode

 (172.16.1.1/32, 239.70.49.238), 02:10:43/00:02:59, flags: CT
  Incoming interface: Ethernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1.305, Prune/Dense, 02:10:43/00:01:28
    Serial0/1.308, Forward/Dense, 02:10:43/00:00:00

Bronze#
```

例 5-9 中也显示了(172.16.1.1,239.70.49.238)的两个输出接口。一个在前转状态，另一个在剪除状态。如同所有的扩散和剪除协议一样，PIM-DM 必须可以为所有接口维护一个剪除状态表。这是因为只有这样，才能使那些从多播树中剪除的路由器在必要时把自己接入树中。

比如，例 5-10 显示了(172.16.1.1,239.70.49.238)的一个路由器路由条目，这里面没有连接的组员，也没有下游的邻居，因此输出的接口是空的，标识 P 表示这台路由器向上游邻居 172.16.2.246 发送了一个 Prune 消息。这时如果连接的主机发送一个 IGMP 消息，请求加入这个组，路由器再向上游，也就是源的方向发送一个 PIM Graft 消息。但是，多播包在最初的扩散过程是路由器知道源的地址的唯一方式。这样，必须如例中所示维持一个剪除状态。

例 5-10　路由器输出接口列表对组(172.16.1.1, 239.70.49.238)为空，于是从树中剪除

```
Lead#show ip mroute 239.70.49.238
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set
Timers: Uptime/Expires
Interface state: Interface, Next-Hop, State/Mode

 (172.16.1.1/32, 239.70.49.238), 02:32:42/0:00:17, flags: PT
  Incoming interface: Serial1.605, RPF nbr 172.16.2.246
  Outgoing interface list: Null

Lead#
```

Graft 消息是用单播方式传送到上游邻居，当上游路由器收到 Graft 消息，就把收到这个消息的接口加到它的输出接口列表中。这个接口设为前转状态，并且马上用单播方式给新的

下游邻居发送一个 Graft Ack 消息，如果已经为其他下游邻居前转包，那么路由器就可以做任何事情了。不过，如果路由器已经把自己从多播树中剪除，那么它必须向自己的上游路由器发送一个 Graft 消息。当路由器发送了 Graft 消息，就会等待 Graft-Ack 消息，等待时间为 3s。如果在这段时间内没有收到确认信号，路由器就重新发送 Graft 消息。

PIM-DM 的扩散与剪除机制同 DVMRP 非常相似。不过有一个重要的不同。回忆在“DVMRP 路由表”一节中，DVMRP 用逆向毒化来表示与上游邻居路由的依赖关系，这种依赖告诉上游 DVMRP 路由器某一下游路由器靠它来前转一个特定组的包。所有这些在源开始发包前就发生了，这依赖于 DVMRP 内置的路由协议。所以在某些拓扑结构下，DVMRP 可以限制扩散的范围。PIM-DM 没有这种能力，因为它没有内置的路由协议。因此，PIM-DM 总是在整个 PIM 域内进行扩散。协议设计者在它的规范后说道：我们选择接收附加开销，是为了得到因不依赖于某些拓扑发现协议的特定类型而具有的简单性和灵活性。

5.8.2　Prune 消息的覆盖

DVMRP 下游依赖性机制的另一优点在剪除过程中更能显现。在图 5-50 中，一台单独的路由器有多个多播下游邻居，上游路由器 Mercury，把组内的多播包扩散到通过 LAN 连接的三台路由器上。Copper 的输出接口表为空，于是向 Mercury 发送一个剪除消息。但是 Silver 有一个连接的组员，于是希望收到多播业务。

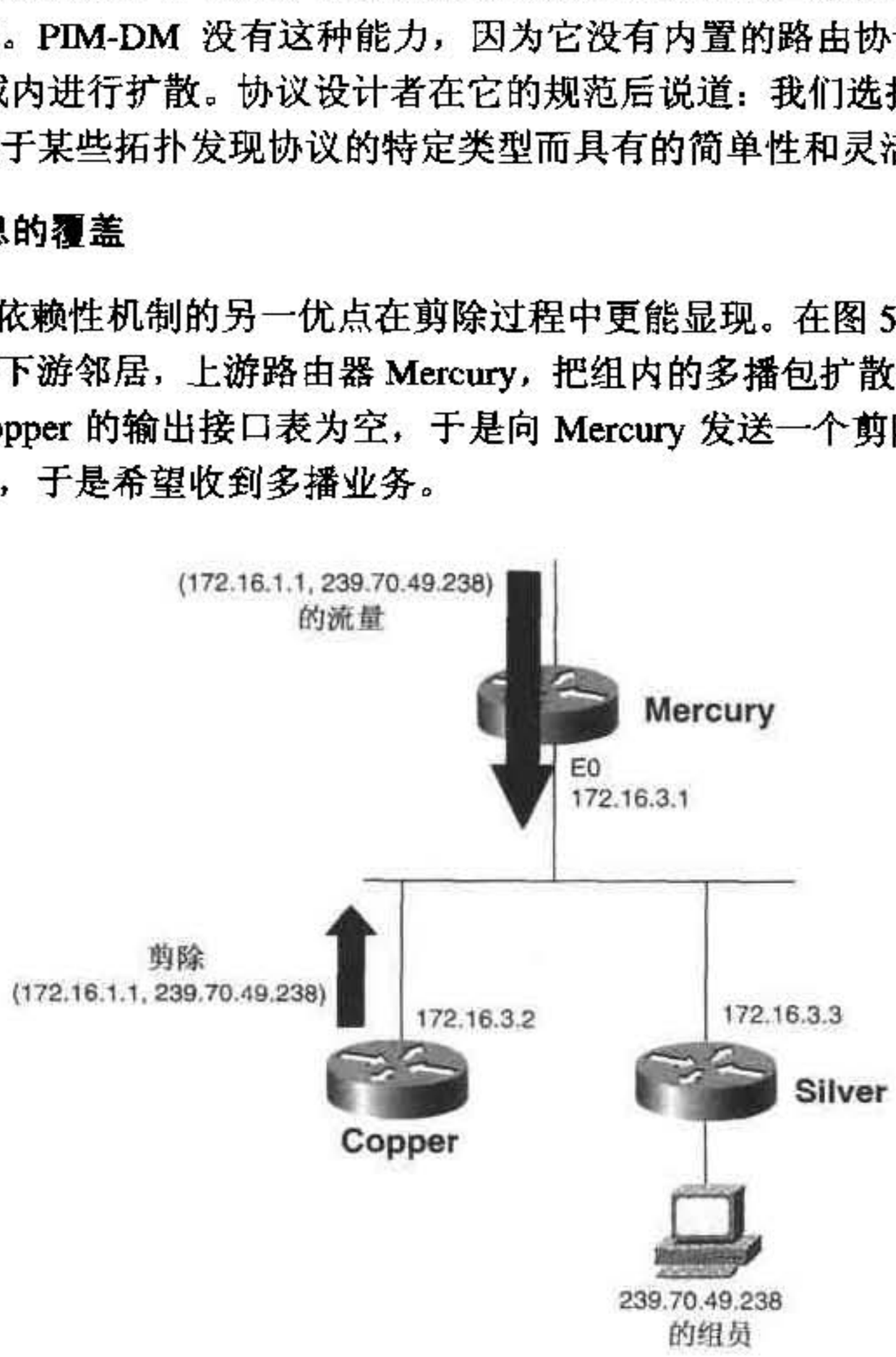


图 5-50　Copper 向(172.16.1.1.238.70.49.238)发送剪除消息，而 Silver 则希望继续接收业务信息

如果这 3 台路由器都运行 DVMRP，这没有问题。Mercury 知道它的下游地点对多播源的依赖，因为它只从 Copper 处收到了 Prune 消息，所以就继续向 Silver 转发流量。

假设图 5-50 中的路由器运行 PIM-DM，Mercury 根据 Hello 消息明确知道它有两个邻居，不过 Hello 消息中没有对依赖性有描述，所以当 Copper 发送一个 Prune 消息，Mercury 不知

道是否把该 LAN 接口剪除。

PIM-DM 通过一个称为剪除覆盖的处理过程来避免这个问题。Copper 向 Mercury 发送一个 Prune 消息，但 Mercury 的地址是封装在这个消息中的。承载这个消息的包的目的地址是所有 PIM 路由器 224.0.0.13。当 Mercury 收到这个消息，并不立即剪除这个接口。它设置一个 3s 的计时器。同时，因为 Prune 消息的目的地址是多播地址,Silver 也收到这个消息，它看到这个要剪除的组是它要继续接收的那个，而这个消息也已经发送到了它的上游邻居处。因此，Silver 向 Mercury 发送一个 Join 消息，如图 5-51 中所示，这样 Silver 发出的 Join 消息就覆盖了 Copper 发出的 Prune 消息。只要 Mercury 在 3s 定时结束前收到这个 Join 消息，流量就不会中断。

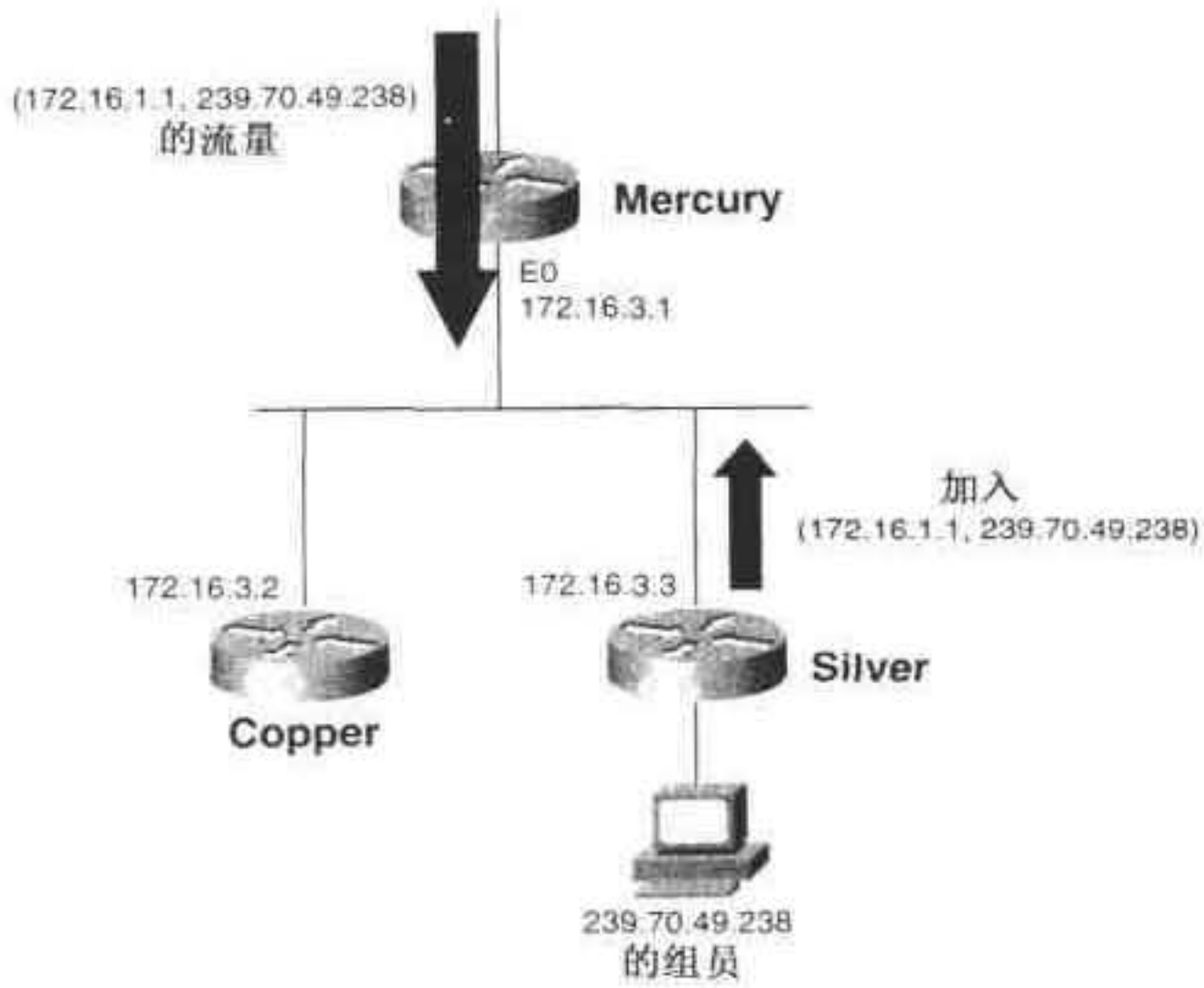


图 5-51　Silver 用 Join 消息把 Copper 的 Prune 消息覆盖

例 5-11 中显示了一个覆盖行为，用 **debug** 来捕获在图 5-50 和图 5-51 过程中 Mercury 上 PIM 的各种活动，第一条消息显示在接口 E0 上收到从 Copper(172.16.3.2)发来的一条针对(172.16.1.1,239.70.49.238)的 Prune 消息(一条把 239.70.49.238 列在 Join/Prune 消息的剪除项里)。注意到第一行的消息指示“to us”，这表示 Mercury 已认出它自己的地址被封装在消息中。

例 5-11　图 5-51 中，路由器 Mercury 先收到来自 Copper(172.16.3.2)的 Prune 消息，然后 Silver(172.16.3.3)发送一条 Join 消息，覆盖掉 Copper 的 Prune 消息。

```
Mercury#debug ip pim
PIM debugging is on
Mercury#
PIM: Received Join/Prune on Ethernet0 from 172.16.3.2, to us
PIM: Prune-list: (172.16.1.1/32, 239.70.49.238)
PIM: Schedule to prune Ethernet0 for (172.16.1.1/32, 239.70.49.238)
PIM: Received Join/Prune on Ethernet0 from 172.16.3.3, to us
PIM: Join-list: (172.16.1.1/32, 239.70.49.238)
PIM: Add Ethernet0/172.16.3.3 to (172.16.1.1/32, 239.70.49.238), Forward state
```

第二和第三行显示 Mercury 要把(S,G)条目从接口 E0 上删除，也就是说 3s 的计时器已经启动。第四行，Mercury 收到 Silver(172.16.3.3)发出的 Join 消息。在第五行和第六行，E0 口对于(S,G)对被设为前转状态。Copper 的 Prune 消息已经被覆盖了。

5.8.3　单播路由的改变

当拓扑结构改变，单播的路由表也要改变。如果单播路由的改变影响到到多播源的路由，PIM-DM 的路由表也要改变。一个显而易见的情况就是拓扑的变化导致了到源的上一跳路由器的改变。

当一个源的 RPF 路由器改变，PIM-DM 首先向原来的路由器发送一个剪除消息，再向新的 RPF 路由器发送一个 Graft 消息，建立新的树。

5.8.4　PIM-DM 指定路由器

PIM-DM 在多访问的网络中选举一个指定路由器。协议本身不需要一个 DR，但 IGMPv1 没有查询过程，要靠路由协议来选出 DR 以管理 IGMP 查询。这正是 PIM-DM(也包括 PIM-SM)指定路由器的原因。

DR 选举的过程非常简单。正如你所了解的一样，每一个 PIM-DM 路由器每 30s 发送一个 PIMv2 Hello 消息或 PIMv1 Query 消息来发现邻居。在多路访问的网络中，有最大 IP 地址的 PIM-DM 路由器即成为 DR，如例 5-12 中的输出内容所示。其他路由器监视 DR 的 Hello 消息；如果在 105s 内没有听到任何消息，则该 DR 被宣布死亡，该网络重新选出新的 DR。

例 5-12　在例 5-11 中，从 Mercury 的 PIM 邻居表中可以看出 Silver 具有最大的 IP 地址，因此被选为指定路由器

```
Mercury#show ip pim neighbor
PIM Neighbor Table
Neighbor Address  Interface          Uptime    Expires   Ver  Mode
172.16.3.3        Ethernet0          2d23h     00:01:17  v1v2 Dense     (DR)
172.16.3.2        Ethernet0          2d23h     00:01:21  v1   Dense
172.16.2.250      Serial1.503        09:15:11  00:01:17  v1   Dense
Mercury#
```

5.8.5　PIM 前转器的选举

在图 5-52 中 Mercury 与 Copper 均连在到源地址为 172.16.1.1 的路由器上，它们也都有一个通向多播组 239.70.49.23 组员的下游接口，这个组员连接在一个多路访问的网络上。Mercury 和 Copper 从源收到同样的多播包，但显然两个路由器都把包前转到同一网络中是不可取的。图中显示 Copper 与 Mercury 均收到多播源 172.16.1.1 发送的多播包，但只有一台路由器会将包前转到子网 172.16.3.0/24。

为了避免这样的情况，PIM 路由器在这个共享的网络中选出一个前转器。DVMRP 也有相似的功能，即指定前转器。DVMRP 中由于部分路由在多路访问网络中交换，故能选出指定前转器。PIM 没有自己的路由协议，但它采用 Assert 消息来选出前转器。

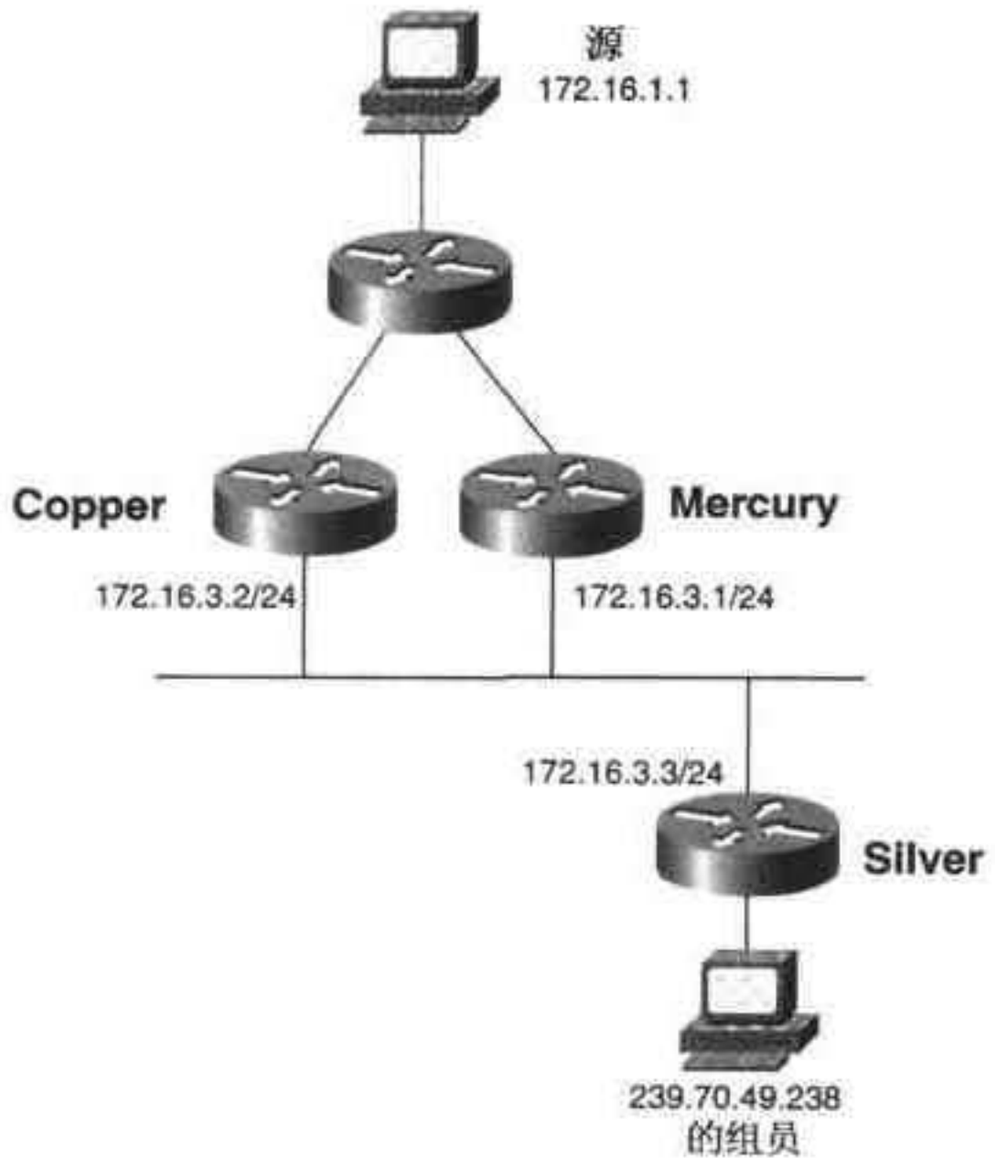


图 5-52　两台路由器收到同一个包，但只有一台会把包前转

当路由器在它的输出接口上收到多播包时，向网络发出 Assert 消息。Assert 消息包括源和组的地址，以及到源单播路由的量度与发现这条路由的路由协议的量度优先权(Cisco 的术语为管理的距离)。产生同样包的路由器将这些消息进行比较，根据下述标准选出前转器：

- 宣布有最低量度优先权的路由器成为前转器。若到源的路由是由不同的单播协议发现的，则路由器只宣告不同的量度优先权。
- 如果量度优先权相同，那么宣告有最低量度的路由器成为前转器。换句话说，如果路由器运行相同的单播路由协议，则从度量上讲，离源更近的路由器成为前转器。
- 如果量度优先权和量度是相同的，那么前转器为网络中有最高 IP 值的那个路由器。

前转器继续向多路访问网络发送多播流量，而其他的路由器停止发送，并把其下游接口从输出接口表中删除。

当图 5-52 中的多播源首次开始向组 239.70.49.238 发送包，比如 Copper 和 Mercury 都收到了包，那么两个路由器均向 172.16.3.0/24 前转这个包，如图 5-53(a)所示。当 Copper 在以太网接口上收到 Mercury 发往(172.16.1.1,239.70.49.238)的包，它看到这个接口对于这个(S,G)对是输出接口，所以向这个子网发出 Assert 消息。当 Mercury 收到 Copper 发出的多播包后，也采取同样的措施，如图 5-53(b)所示。

例 5-13 显示了 Silver 的单播路由表和它的多播前转表，其中单播路由表显示 Silver 到源 172.16.1.1 的子网需经两跳路由器；多播路由表显示有最高 IP 地址值的下一跳路由器被选为前转器。这个单播路由表表明到 172.16.1.1 的 OSPF 路由开销相等。因为这些路由是 OSPF 的，它们的管理距离均为 110；因为两条路由的 OSPF 开销均为 74，所以前转器为最高 IP 值的路由器。

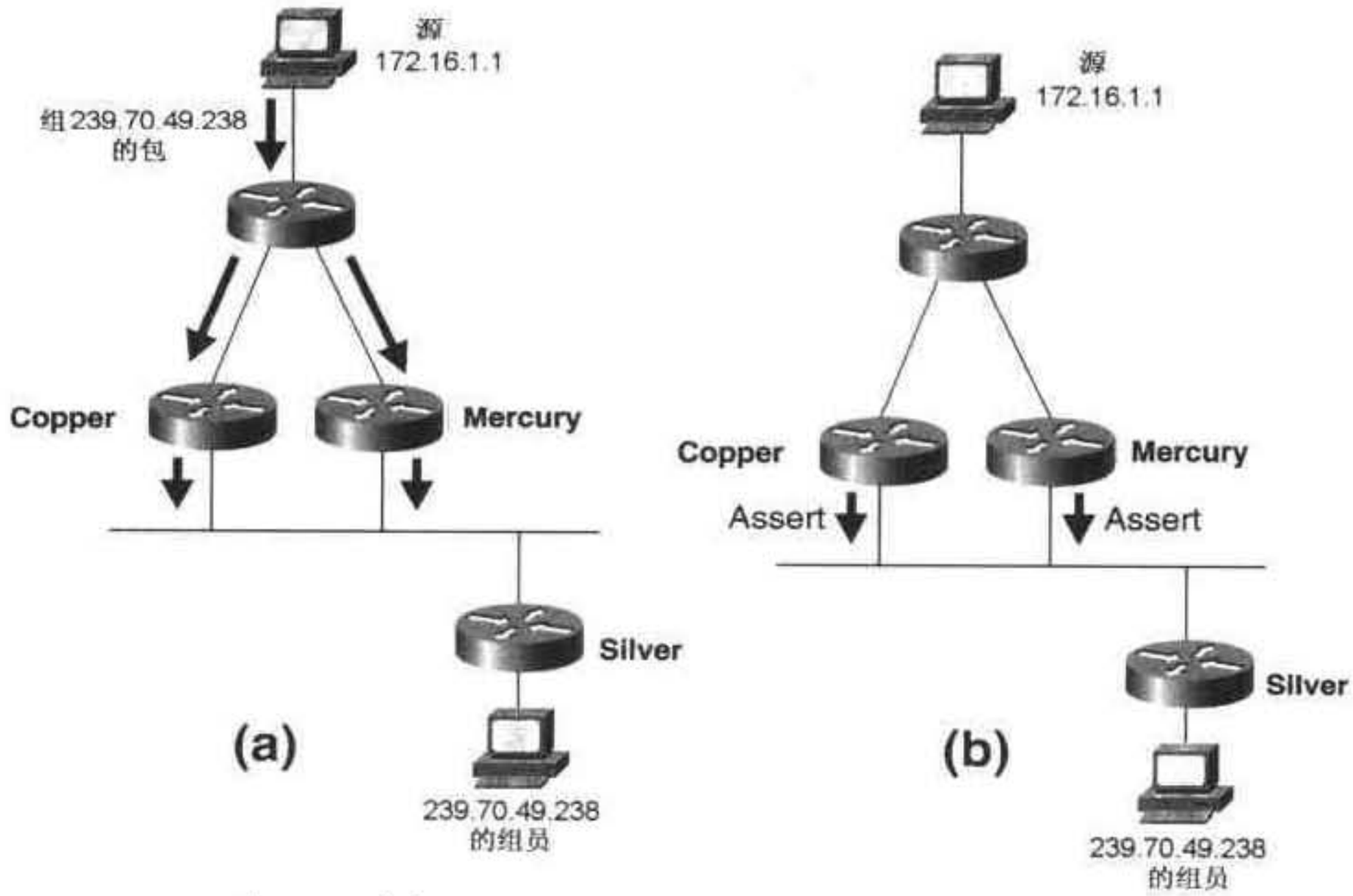


图 5-53　两台路由器在下游接口收到多播包，要判断哪一个成为前转器

例 5-13　Silver 的单播路由表和它的多播前转表

```
Silver#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX  EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
       T - traffic engineered route
Gateway of last resort is not set
     172.16.0.0/16 is variably subnetted, 8 subnets, 2 masks
O       172.16.2.252/30 [110/138] via 172.16.3.1, 00:02:16, Ethernet1
                        [110/138] via 172.16.3.2, 00:02:16, Ethernet1
O       172.16.2.248/30 [110/74] via 172.16.3.1, 00:02:16, Ethernet1
O       172.16.2.244/30 [110/74] via 172.16.3.2, 00:02:16, Ethernet1
                        [110/74] via 172.16.3.1, 00:02:16, Ethernet1
O       172.16.2.240/30 [110/138] via 172.16.3.1, 00:02:16, Ethernet1
O       172.16.2.236/30 [110/74] via 172.16.3.1, 00:02:16, Ethernet1
C       172.16.5.0/24 is directly connected, Ethernet0
O       172.16.1.0/24 [110/84] via 172.16.3.1, 00:02:16, Ethernet1
                      [110/84] via 172.16.3.2, 00:02:16, Ethernet1
C       172.16.3.0/24 is directly connected, Ethernet1
Silver#

Silver#show ip mroute 172.16.1.1 239.70.49.238
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(172.16.1.1, 239.70.49.238), 00:02:02/00:02:59, flags: CT
  Incoming interface: Ethernet1, RPF nbr 172.16.3.2
  Outgoing interface list:
    Ethernet0, Forward/Dense, 00:01:50/00:00:00

Silver#
```

5.9　与协议无关的多播，稀疏模式(PIM-SM)的操作

你已经在前面知道共享树在稀疏分布的多播网际网络中如何有更好的扩展性，它们甚至能在密集分布的网际网络中使用。这可能给你留下一个印象：共享式的多播树比基于源的多播树更好，但事实上不是这样的。

图 5-54 显示了一个基于源的树要比共享树更适合的情况。在这个拓扑里，源和目的间的距离要比它们到共享树所根植的核心要近，在源和目的间建立基于源的树，从减少开销的角度来讲是更适合的。

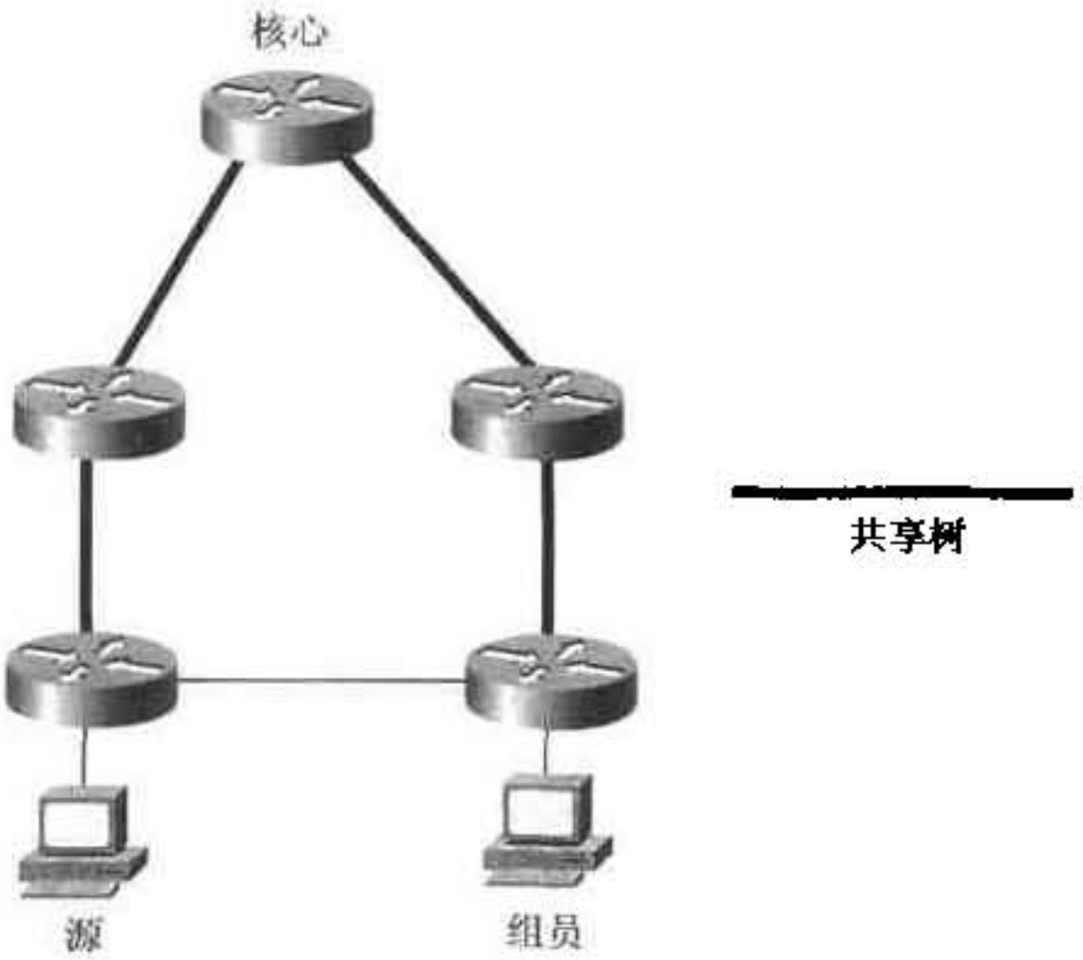


图 5-54　这个网络中的共享树更适于基于源的树

不同于 CBT，PIM-SM 同时支持共享式的和基于源式的树，这是当前多数网络选用 PIM-SM 作为多播路由协议的原因。

PIM-SM 的规范是 RFC2362[13]。

5.9.1　PIM-SM 基础

PIM-SM 采用了 7 种 PIMv2 的消息：

- Hello
- Bootstrap
- Candidate-RP-Advertisement
- Join/Prune
- Assert
- Register
- Register-Stop

注意到，有 3 个消息(Hello、Join/Prune、Assert)也在 PIM-DM 中使用，而另 4 个消息为

PIM-SM 独有，正像 PIM-DM 中有两个独有的消息(Graft 与 Graft-Ack)一样。

PIM-SM 与 PIM-DM 有几个共同的功能：

- 通过交换 Hello 消息发现邻居；
- 当单播路由发生变化时，重新计算 RPF 接口；
- 在多路访问的网络中选举指定路由器；
- 在多路访问的网络中使用剪除覆盖；
- 在多路访问网络中用 Assert 消息选举指定前转器。

这些功能在 PIM-DM 一节中已经进行了描述，在此就不再赘述。

不同于 PIM-DM 的是，PIM-SM 使用显式加入，这使得无论是共享式的树，还是基于源的树的建立都更加有效。

5.9.2　查找会聚点

正如你所知道的一样，共享树根植于多播网络中的一台路由器，而并非源。CBT 中称这台路由器为核心，PIM-SM 称之为会聚点(rendezvous point RP)。在共享树建立之前，加入的路由器必须知道如何找到 RP。路由器可以通过 3 种方法知道 RP 地址：

- RP 地址可以静态地配置于每一台路由器中；
- 用开放标准的自举协议来指定和宣告 RP；
- 以 Cisco 拥有知识产权的自动 RP 协议来指定和宣告 RP。

这 3 种方式的使用将在第 6 章中讲述。

如同静态路由一样，在所有路由器上静态配置 RP 地址在对互联的网络提供明确的管理方面的确有优势，但是管理的开销会很大。静态 RP 配置只可用于小型的多播网络中。

1．自举协议

Cisco IOS Software Release 11.3T 最早支持自举协议，这基本上是同 CBT 宣告核心路由器使用同一个协议，只是在消息名称与格式上有一些改动。为了运行自举协议，候选自举路由器(Candidate bootstrap router C-BSR)和候选会聚点(Candidate Rendezvous Point C-RP)在网络中用管理方式指定。一般情况下，一组路由器同时配置成 C-BSR 和 C-RP。C-BSR 同 C-RP 通过 IP 地址来识别他们自己，这个 IP 地址一般配置为 loopback 接口的地址。

第一步是从 C-BSR 中选出 BSR。每一个 C-BSR 都分配一个 0～255 的优先级(默认为 0)和一个 BSR IP 地址。当路由器配置成候选 BSR 时，它设自举计时器为 130s，并开始监听 Bootstrap 消息。

Bootstrap 消息宣告了消息发起者的优先级和它的 BSR IP 地址。当一个 C-BSR 收到一个 Bootstrap 消息，它把发起路由器的优先级与自己的进行比较：如果消息发起者有更高的优先级，接收者将重设它的自举计时器，继续监听；如果接收者的优先级较高，它宣布自己为 BSR，并开始每 60s 发送一个 Bootstrap 消息。如果优先级一样，则要靠 BSR IP 地址来做评判。

如果 C-BSR 的 130s 的自举计时器超时，路由器则认为网中没有 BSR，它宣布自己为 BSR，开始每隔 60s 发送一个 Bootstrap 消息。

Bootstrap 消息采用了“所有 PIM 路由器”这个多播地址 224.0.0.13，TTL 设为 1。当一台 PIM 路由器收到 Bootstrap 消息，它在除收到这个消息外的接口上把这个消息发出。这一过程不仅可以保证 Bootstrap 消息扩散到多播域中，也能保证每一个 PIM 路由器都收到这个

包，从而知道 BSR 是哪个路由器。

一台 C-RP 配置一个 RP IP 地址和在 0～255 间取值的优先级。路由器可以配置成只能为某些特定的的多播组做 C-RP，或者也可以配置成所有多播组的 C-RP。当收到 Bootstrap 消息，知道 BSR 的位置后，C-RP 通过单播向 BSR 传送 Candidate-RP-Advertisement 消息。这些消息中有消息发起者的 RP 地址，担当 C-RP 那个组的地址以及它的优先级。

BSR 对所有 C-RP 加以整理，列出它们的优先级和它们相应的组，编成“RP 集”。它通过 Bootstrap 消息向整个多播域宣告 RP 集，Bootstrap 消息中包括一个 8bit 的 hash 掩码。所有的 PIM 路由器都将收到 Bootstrap 消息，因为它的目的地址是 224.0.0.13。

当一台路由器收到 IGMP 消息或 PIM Join 消息，必须加入一个共享树时，它检查通过 Bootstrap 消息从 BSR 处获知的 RP 集。

- 如果组中只有一个 C-RP，这台路由器被选 RP；
- 如果组中有多个 C-RP，每一个有不同的优先权，则选有最低优先权的路由器；
- 如果组中有多个 C-RP，它们有相同的优先权，那么也运行一个 hash 函数。这个函数输入的是组的前缀、hash 掩码和 C-RP 的地址，输出为一组数字。有最大结果的 C-RP 成为 RP。
- 如果有不只一个 C-RP 的 hash 函数返回相同的值，则有最大 IP 地址的 C-RP 成为 RP。

注：　如果你一定要知道这个 hash 函数，这个算法如下：

Value(G,M,C)=(1103515245*((1103515245*(G&M)+12345)XOR C)+12345)mod 2^{31}

G=组地址前缀

M=hash 掩码

C=C-CP 地址

这一组处理过程保证了域内所有的路由器为一个组选择了同一个 RP。这个 hash 函数很必要的唯一原因就是配合 hash 掩码，hash 掩码可以允许连续组的地址号映射到同一 RP 上。关于 hash 掩码的使用在第 6 章中讲述。

2. 自动 RP 协议

自动 RP 最早是在 Cisco IOS Software Release 11.1(6)中支持的。它是 Cisco 在定义 PIM-SM 的自举协议前，为了自动发现 RP 而开发的。同自举协议一样，候选 RP(C-RP)在 PIM-SM 域中指定，并用分配的 IP 地址进行识别，通常为 loopback 接口的地址。再设置一个或多个 RP 映射代理，它们是与 BSR 作用相似的路由器。与自举协议的不同点最主要有 4 个方面：

- 自动 RP 是有 Cisco 自主产权的，不能在多厂商的环境中使用，不过有些厂商支持自动 RP；
- RP 映射代理是指定的，而不是像 BSR 从一组候选者中选举出来的；
- RP 映射代理将组映射到 RP，而不是宣告 RP 集，并在全网分布式地选择 RP；
- 自举协议使用 224.0.0.13 多播地址，这个地址能被所有的 PIM 路由器所理解。而自动 RP 采用保留的多播地址 224.0.1.39 和 224.0.1.40。

当 Cisco PIM-SM 路由器配置成一个组或多个组的候选 RP，它在 RP-Announce 消息里对自己和多播组宣告它是 C-RP。这个消息每 60s 向保留的 Cisco RP 通告地址 224.0.1.39 发送一次，这个域中配置好的映射代理在这个地址上监听。从收到的 RP-Announce 消息中，映射代理从组中的 C-RP 中选出有最大 IP 地址值的一台作为 RP。

这个 RP 映射代理在 RP 发现消息中宣告完整的组到 RP 的映射。这个消息每 60s 发送一次，其目的地址为保留的 Cisco RP 发现地址 224.0.1.40。所有 Cisco PIM-SM 路由器都监听这个地址，以获得每个已知组的正确 RP。

5.9.3　PIM-SM 和共享树

共享树路由条目与基于源的路由条目(也称 SPT 路由条目)的主要差别是共享树路由是与源无关的——事实上就是能有许多源共享的同一树。因此，路由条目是(*,G)对，这里的星号表示向组 G 发送流量的任何或全部源地址的通配符。

当一个 PIM-SM DR 收到一个组员的 IGMP Membership Report 消息请求加入一个多播组时，它先检查多播表里是否有这个组的条目。如果有这个组的条目，那么收到 IGMP 的那个接口将作为输出接口加入到这个条目中，而不需要其他的行为。

如果没有相应的条目存在，就要为这个组建立一个(*,G)对，加入输出接口。路由器为这个组查找组到 RP 的映射(如例 5-14 所示，**show ip pim rp mapping** 命令显示路由器组到 RP 的映射，这里所有组都映射到 RP 172.16.224.1)，并在单播路由表查找到某一特定 RP 的路由，把到 RP 的上游接口加入到输入接口(RPF)。

例 5-14　所有组均映射到 RP 172.16.224.1

```
Iron#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s): 224.0.0.0/4, Static
    RP: 172.16.224.1 (?)
Iron#
```

例 5-15 表示了图 5-55 中 Iron 的(*,G)路由条目，例中(*, G)条目表明组(236.82.134.23)共享树上的上游邻居是 172.16.224.1。与该条目相关的标志表明它是稀疏模式，并有连接的组员。

例 5-15　图 5-55 中 Iron 的(*,G)路由条目

```
Iron#show ip mroute 236.82.134.23
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 236.82.134.23), 00:08:58/00:02:59, RP 172.16.224.1, flags: SC
  Incoming interface: Serial1.708, RPF nbr 172.16.2.242
  Outgoing interface list:
    Ethernet0, Forward/Sparse, 00:08:59/00:02:47

Iron#
```

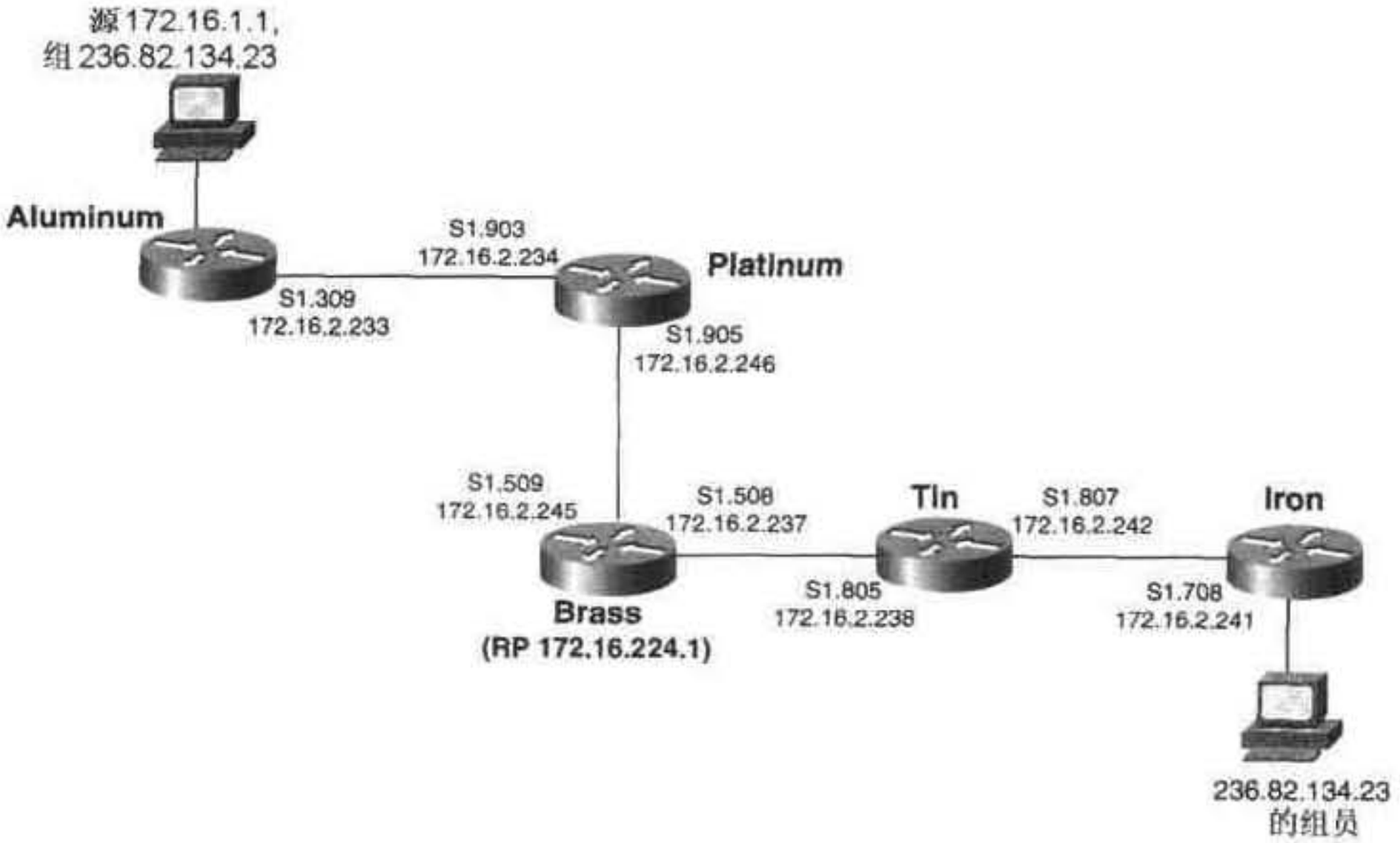


图 5-55　路由器 Brass 是这个 PIM-SM 域中的 RP，它的 RP 地址为其 loopback 地址

这时，路由器通过上游接口向组 224.0.0.13 发送 Join/Prune 消息，如图 5-56 所示。这个消息包括了要加入的组的地址和 RP 的地址，剪除部分的消息为空。这个消息中设置了两个标志通配比特(WC-比特)和 RP 树比特(RPT-比特)：

- WC-比特＝1 表示这个加入地址是一个 RP 地址，而不是源地址。
- RPT-比特＝1 表示这个消息顺着共享树传送到 RP。

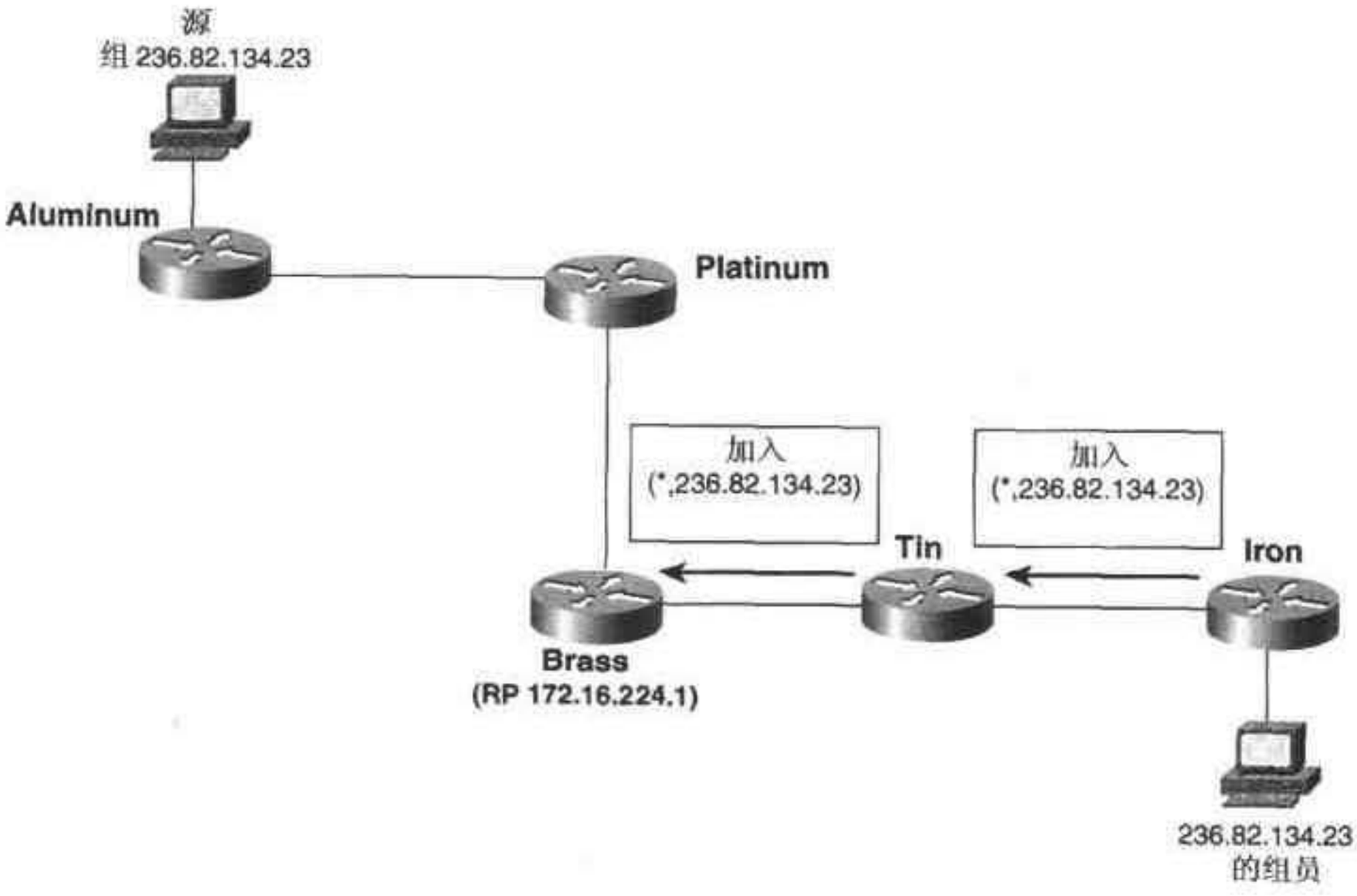


图 5-56　Join/Prune 消息经一跳接一跳传送到 RP

当上游路由器收到 Join/Prune 消息，下列 4 种情况之一及相关的行为为真：

- 这路由器不是 RP，它在多播树上。这台路由器把收到 Join/Prune 消息的接口加到这个组的输出接口列表中。
- 这路由器不是 RP，它也不在多播树上。这台路由器建立一个(*,G)条目，并向 RP 发出自己的 Join/Prune 消息。
- 这路由器是 RP，它对这个组已有一个条目。这台路由器把收到 Join/Prune 消息的接口加入到组的输出接口列表中。
- 这路由器是 RP，它没有这个组的条目。路由器建立一个(*,G)条目，并把接收到消息的接口加入到组的输出接口列表中。

最后一条暗示多播组从 RP 的成员请求建立的树不一定有多播源。

当建立了共享树，路由器向上游邻居周期性发送 Join/Prune 消息作为存活保持。Join/Prune 消息中列出了所有邻居是上一跳路由器的路由条目。默认的周期为 60s。这可以通过 Cisco IOS 软件命令 **ip pim message-interval** 来改变。保持的时间长度为 Join/Prune 间隔的 3 倍，或者是默认的 3 分钟，它也在 Join/Prune 消息中宣告。如果 PIM-SM 路由器在保持时间里没有从下游的 Join/Prune 消息中听到一个已知组的信息，那么它把下游路由器从这个组输出接口列表中删除。例 5-16 中显示了图 5-55 中路由器 Tin 中组 236.82.134.23 的条目，路由器 Tin 上组 236.82.134.23 的条目显示与下游路由器 Iron 相关的保持时间计时器。注意，这个条目没有设置 C 标志，因为 Tin 没有直连的组员。到路由器 Iron 的输出接口 S1.805 表明这个接口如果在 2 分钟 11 秒之内没收到 Iron 的 Join/Prune 消息，这个接口就要被剪除。

例 5-16　图 5-55 中路由器 Tin 上组 236.82.134.23 的条目

```
Tin#show ip mroute 236.82.134.23
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set
Timers: Uptime/Expires

(*, 236.82.134.23), 00:09:39/0:02:56, RP 172.16.224.1, flags: S
  Incoming interface: Serial1.805, RPF neighbor 172.16.2.237
  Outgoing interface list:
    Serial1.807, Forward state, Sparse mode, uptime 00:09:39, expires 0:02:11

Tin#
```

剪除会发生同样的情况。当一台路由器因为没有直连的组，也没有下游邻居，而希望从共享树中剪除时，它从 RPF 接口发送一个 Join/Prune 消息，设置其中的 WC-比特与 RPT-比特。上游路由器把收到这个消息的接口从组的输出接口列表中删除。如果路由器没有下游的邻居，也没有连接的组员，它也把自己剪除。

注：PIM-DM 中描述的剪除覆盖的机制也会在此用来保证多路访问的下游邻居不会被无意剪除

5.9.4　源的注册

共享树的基本概念前面已经多次提到，就是多播树根植于一个核心或会聚点，而不是源。

于是产生了一个问题，源如何向 RP 发送多播包，使其能沿树枝传送。在 CBT 中用双向树来解决这个问题——包可以从核心向下传送，也可以沿树枝向上传到核心。直连多播源的路由器加入到多播树中，沿树枝向核心发送流量。双向树的问题是它很难保证无环的拓扑，因为没有明确的上游和下游，所以不能进行 RPF 检查。

与 CBT 不同，PIM-SM 采用 RPF 检查。因此，它的树必须为单向——就是说，流量只能从 RP 沿树枝向下传送。单向的流量保证了明确的输入或 RPF 接口。如果流量只由 RP 向外发送，那么源将如何向 RP 发送多播流量。

当 PIM-SM 路由器首次收到直连的源发出的多播流量时，它通过查找组到 RP 的映射来找出这个组的正确 RP，如例 5-17 中显示的输出结果，例中为图 5-55 中路由器 Aluminum 的组到 RP 的映射：与例 5-14 比较，Iron 有 RP 静态映射，而 Aluminum 则动态得到 RP 地址。这个步骤与一个成员用 IGMP 表示一个组的加入相同。

例 5-17　路由器 Aluminum 的组到 RP 的映射

```
Aluminum#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/4, uptime: 00:02:39, expires: 00:02:17
    RP 172.16.224.1 (?), PIMv2 v1
    Info source: 172.16.2.245 (?)
Aluminum#
```

决定了组的 RP 后，路由器将多播包封装在 PIM Register 消息中，并将这个消息发往 RP，Register 消息不是用多播，而是用单播发到 RP 的地址，如图 5-57 所示。

当 RP 收到这个 Register 消息后，将多播包去除封装。如果多播路由表中已经有这个组的条目，则多播包的备份从所有输出端口上送出，如图 5-58 所示。

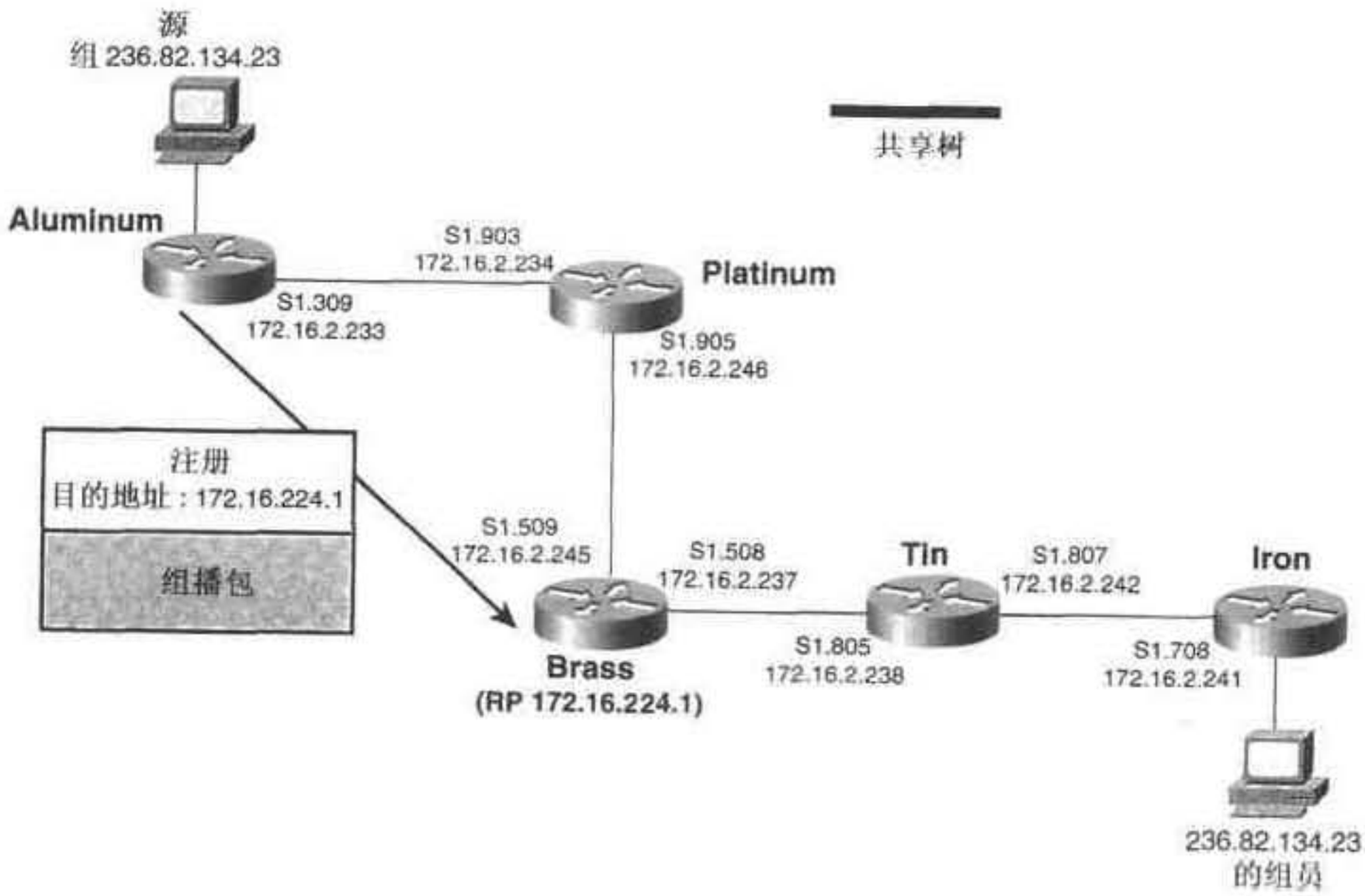


图 5-57　第一个多播包封装在 PIM Register 消息里，用单播送到 RP

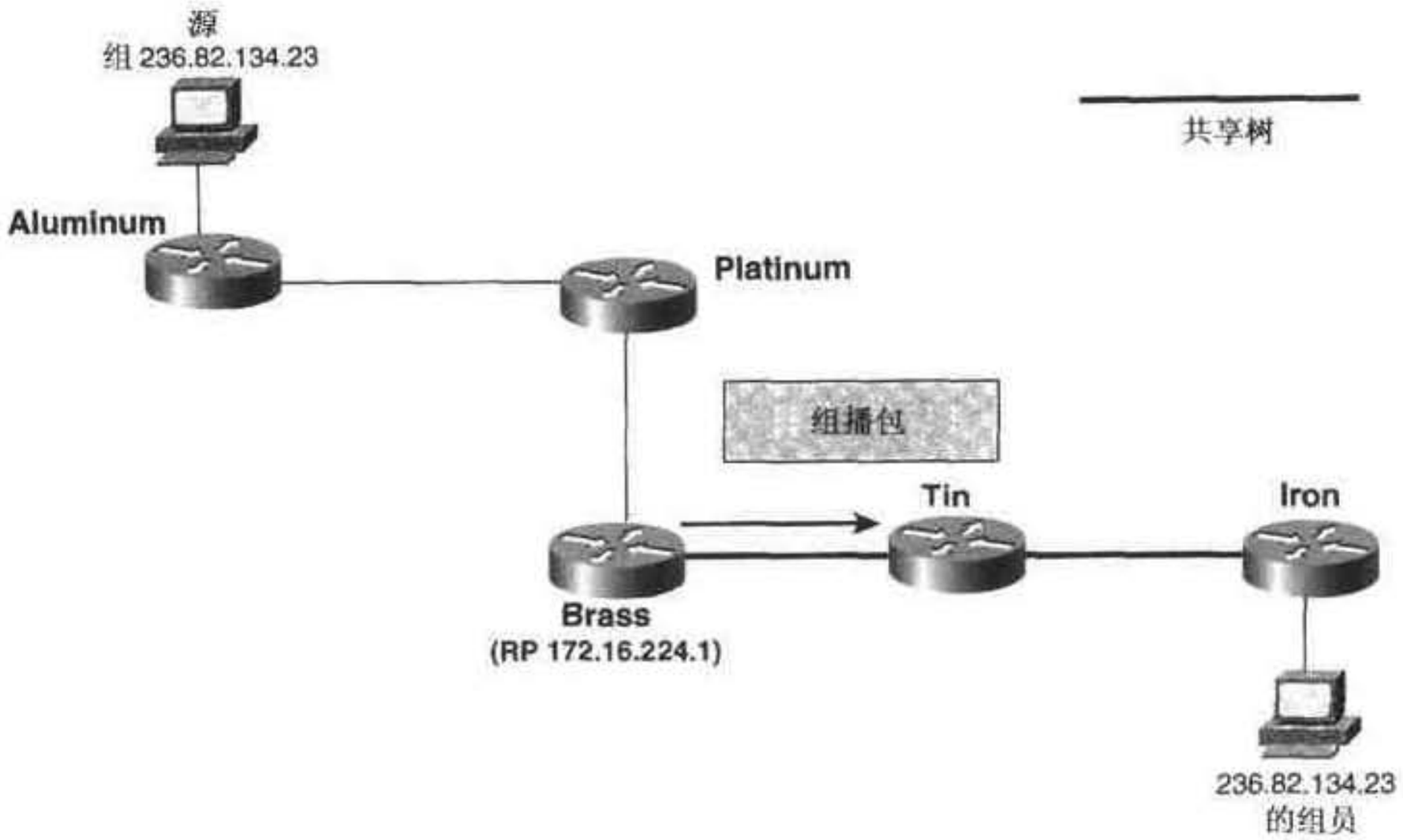


图 5-58　多播包不用 Register 消息封装，而从所有该组输出接口列表里的接口上前转

如果有相当多的多播流量要发向 RP，不断在 Register 消息里封装这些发往 RP 的包是很低效的。因些 RP 在多播表里建立一个(S,G)条目，并且 Join/Prune 消息与源 DR 间创建一条 SPT，如图 5-59 所示。这个消息包括多播源的地址，WC-比特＝0，RPT-比特＝0 来表示这条通路是基于源的 SPT，而不是共享 RPT。

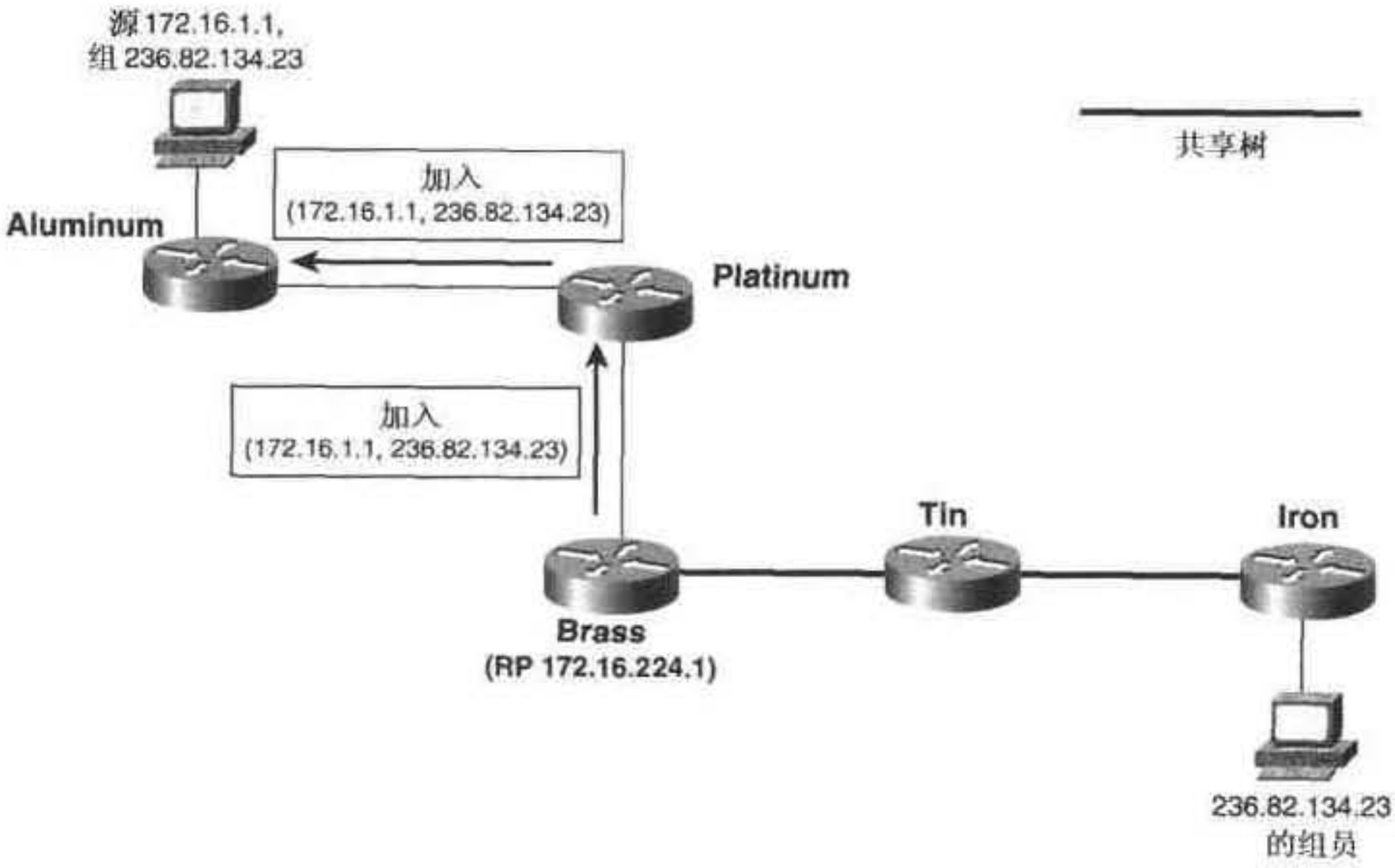


图 5-59　RP 建立一个到达 DR 的基于源的最短路径树

建立好 SPT 后，RP 能通过树收到组的流量。它向多播源 DR 发送 Register Stop 消息，告诉它停止在 Register 消息中发送多播包，如图 5-60 所示。

如果当多播源开始向 RP 发送多播流量，而组内还没有组员时，RP 不会建立 SPT。相反，

它会向多播源 DR 发送 Register Stop 消息，告诉它停止在 Register 消息中封装多播包。RP 对这个(*，G)对建立一个条目，当有组员加入后，RP 会发起建立 SPT。

有一个称为注册抑制的机制来保护 DR 继续向一个无效的 RP 发送包。当 DR 收到 Register Stop，启动一个 60s 的注册抑制计时器，当这个计时器超时，路由器再次用 Register 消息向 RP 发送多播包。不过在发送消息 5s 前 DR 发送一个设置了一个标识的 Register 消息，这个标识称为空注册比特，而且这个消息也不封装任何包。如果这个消息触发了 RP 发送一个 Register Stop 消息，则重新设置注册抑制计时器。

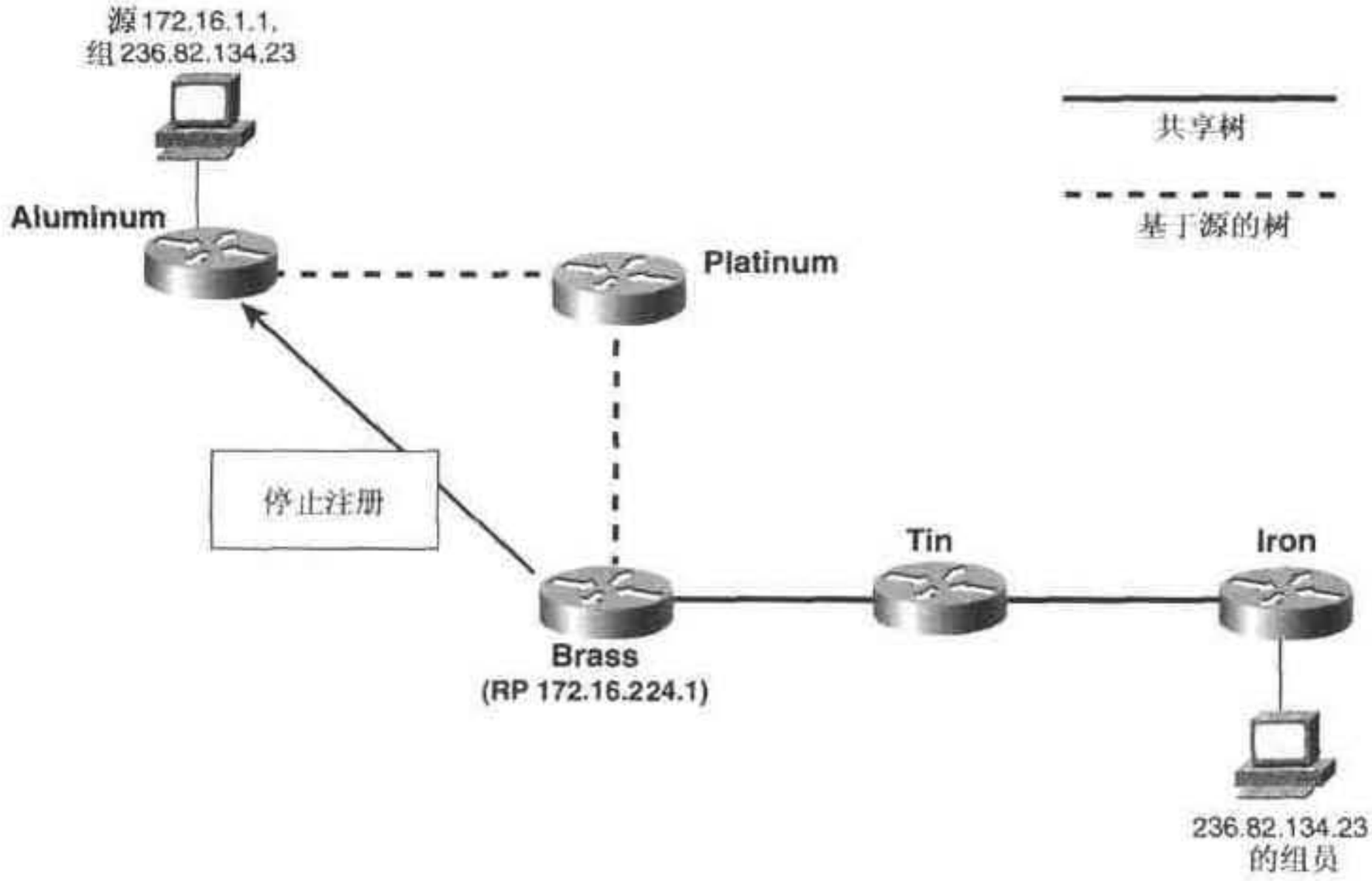


图 5-60　RP 发送 Register Stop 消息来停止 Register 消息，源的多播包通过 SPT 进行发送

例 5-18 中的 **debug** 的输出中显示了当路由器 Aluminum 向组 236.82.134.23 开始发送流量时发生的一系列事件，注意，所有的消息都是单播，而不是多播。在这特定的情况下，还没有组员加入到组里，因此 RP(Brass)马上向 Aluminum 发送 Register Stop 命令来响应 Register 消息。

例 5-18　当路由器向组 236.82.134.23 发送流量时，在 Aluminum 上 **debug** 的输出中显示

```
Brass#debug ip pim 236.82.134.23
PIM debugging is on
Brass#
PIM: Received Register on Serial1.509 from 172.16.2.233 for 172.16.1.1, group
236.82.134.23
PIM: Send Register-Stop to 172.16.2.233 for 172.16.1.1, group 236.82.134.23
```

例 5-19 显示了组的一个路由条目。注意到，这个组同时有(*,G)和(S,G)路由条目的存在。(*,G)条目显示了空的输入接口和 RPF 邻居为 0.0.0.0，表明这台路由器是共享树的根；(S,G)条目显示了到多播源的上游路由器 Platinum(172.16.2.246)是 RPF 邻居。由于没有输出接口，这个条目将被剪除。

例 5-19　RP 上的组 236.82.134.23 的路由条目(没有组员加入这个组)

```
Brass#show ip mroute 236.82.134.23
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 236.82.134.23), 00:07:38/00:02:59, RP 172.16.224.1, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial1.509, Forward/Sparse, 00:03:06/00:02:50

(172.16.1.1, 236.82.134.23), 00:07:38/00:01:21, flags: P
  Incoming interface: Serial1.509, RPF nbr 172.16.2.246
  Outgoing interface list: Null

Brass#
```

例 5-20 显示了作为多播源 DR 的 Aluminum 列出的这个组的路由条目，这里仍有(*,G)条目，有一个连接源的以太网接口列在输出接口表中。输入的接口列表为空。在(S,G)条目中，显示了在输入列表中的同一个以太网接口。这两个条目都有两个标识，一个表明这个源是直连的，另一个(F)表明这路由器必须用 Register 消息发送多播流量。

例 5-20　源的 DR 上相应的路由条目显示一个剪除的 SPT 条目

```
Aluminum#show ip mroute 236.82.134.23
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop, State/Mode

(*, 236.82.134.23), 00:15:30/00:02:59, RP 172.16.224.1, flags: SJCF
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse, 00:15:23/00:02:28

(172.16.1.1/32, 236.82.134.23), 00:00:29/00:02:30, flags: PCFT
  Incoming interface: Ethernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list: Null

Aluminum#
```

(S,G)条目中的 T 标识表明这个条目是 SPT，P 标识表示输出接口列表中没有任何接口。如果有一个 RPF 邻居，路由器会向它发送这个组的剪除消息。

最后一个关心的标识是(*,G)条目里的 J 标识，这一标识表明当在共享树上收到一个包时，这台路由器变成 SPT。PIM-SM 如何从共享树变成 SPT 是下一节的内容。

例 5-21 中的 **debug** 消息显示了当一台主机连接到路由器 Iron 上，加入一个组时发生的一系列事件。由 Iron 产生的 Join/Prune 消息经一跳接一跳传送到 RP，从 Tin 那里收到。这个到 Tin 的接口加入到(*,G)条目中，另外也加入到(S,G)条目中，因为要用到 Aluminum 的 SRP。下一步，向 Aluminum 发送 SPT Join 消息。

例 5-21　debug 消息显示连在路由器 Iron 上的成员加入组 236.82.134.23

```
Brass#debug ip pim 236.82.134.23
PIM debugging is on
Brass#
PIM: Received v2 Join/Prune on Serial1.508 from 172.16.2.238, to us
PIM: Join-list: (*, 236.82.134.23) RP 172.16.224.1, RPT-bit set, WC-bit set, S-bit
set
PIM: Add Serial1.508/172.16.2.241 to (*, 236.82.134.23), Forward state
PIM: Add Serial1.508/172.16.2.241 to (172.16.1.1/32, 236.82.134.23)
PIM: Building Join/Prune message for 236.82.134.23
PIM: For 172.16.2.246, Join-list: 172.16.1.1/32
PIM: Send periodic Join/Prune to 172.16.2.246 (Serial1.509)
```

例 5-22 显示了 RP 最后的路由信息。当组员加入时，它的接口加入到(*,G)条目中，也加到(S,G)条目中，因为有到 Aluminum 的 SPT 树。例 5-23 显示了多播源 DR 最后的路由条目，到 RP 的接口被加入到 Aluminum 的(S,G)输出接口列表中，而且这条目不再为剪除状态。

例 5-22　RP 最后的路由信息

```
Brass#show ip mroute 236.82.134.23
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 236.82.134.23), 00:29:58/00:03:05, RP 172.16.224.1, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial1.509, Forward/Sparse, 00:29:58/00:02:52
    Serial1.508, Forward/Sparse, 00:24:36/00:03:05

(172.16.1.1, 236.82.134.23), 00:24:54/00:02:59, flags: T
  Incoming interface: Serial1.503, RPF nbr 172.16.2.246
  Outgoing interface list:
    Serial1.508, Forward/Sparse, 00:24:36/00:02:35

Brass#
```

例 5-23　多播源 DR 最后的路由条目

```
Aluminum#show ip mroute 236.82.134.23
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop, State/Mode

(*, 236.82.134.23), 00:00:47/00:02:59, RP 172.16.224.1, flags: SJCF
  Incoming interface: Serial0/1.309, RPF nbr 172.16.2.245
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse, 00:00:01/00:02:58

(172.16.1.1/32, 236.82.134.23), 00:00:47/00:02:59, flags: CFT
  Incoming interface: Ethernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1.309, Forward/Sparse, 00:00:34/00:02:58

Aluminum#
```

5.9.5　PIM-SM 与最短路径树

在图 5-61 中，路由器 Lead 加入到 PIM-SM 域中，Lead 连接着一个组员。根据基本的共享树规则，Lead 要加入根植在路由器 Brass 的树中，图中能很明显看出到 Aluminum 的直连链路是从多播源到 Lead 的组员的最有效的路径。

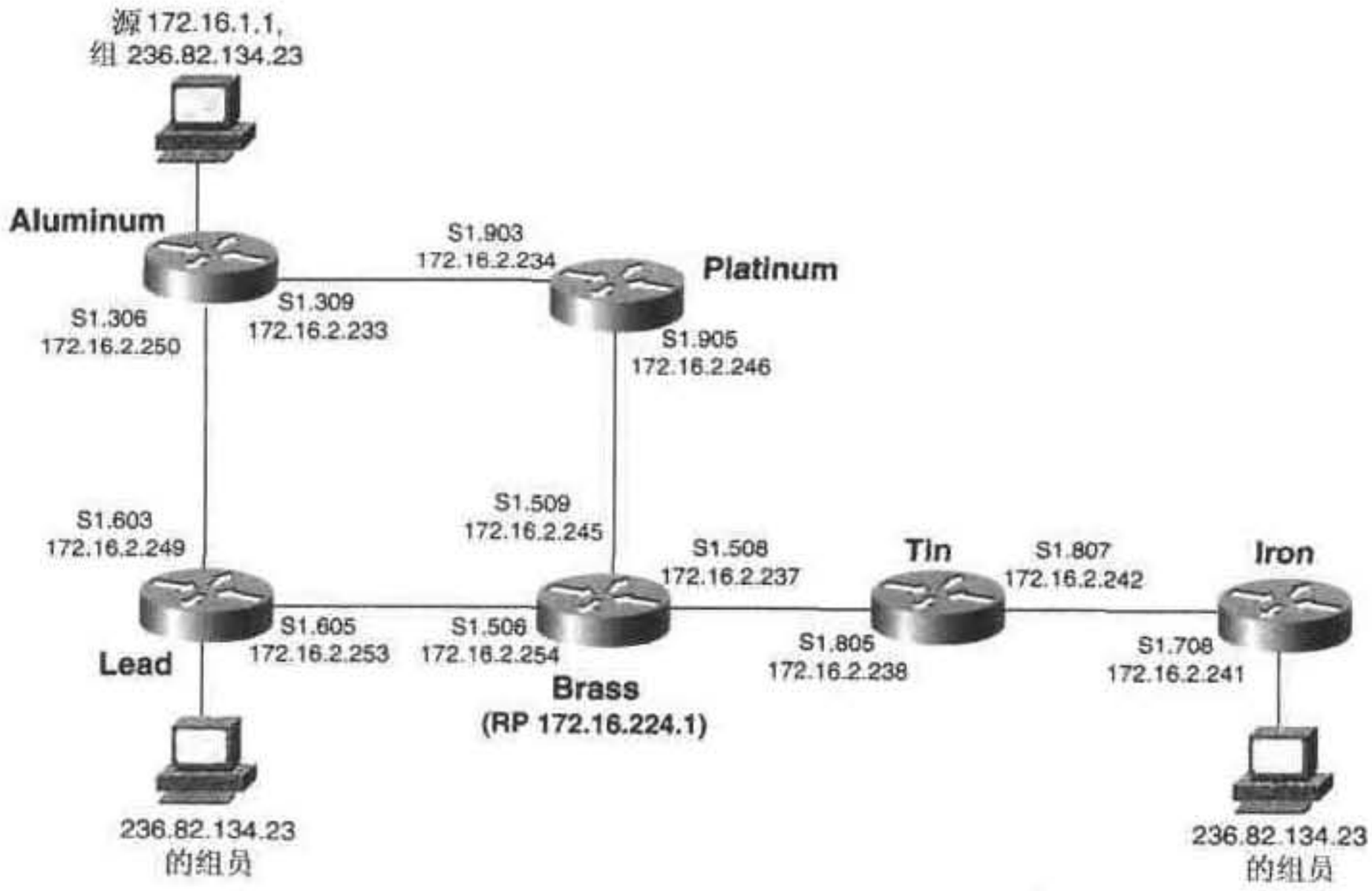


图 5-61　Lead 与 Aluminum 间的直连链路对于发往在 Lead 上的多播成员的多播包来说，比 Aluminum-Platinum-Brass-Lead 这一条通路更有效

你已经看到 PIM-SM 是如何在 RP 与源 DR 间建立一 SPT 的。这个协议也允许连接着组员的路由器建立到源 DR 的 SPT 来减少拓扑中的低效因素，就像图 5-61 中所示。图中，Lead 与 Aluminum 间的直连链路与 Aluminum-Platimun-Brass-Lead 通路相比，是到达 Lead 上连接的组员更经济的一条路由。

例 5-24 显示了 Lead 在组员通过 IGMP 请求加入一个组后建立一个 SPT 的过程。首先，路由器如所期望的一样，向 RP 发送一个 Join 消息(通过 S1.605)，当多播包开始到达时，路

例 5-24　Lead 加入到共享树中，随后收到多播流量，直接加入到源 DR

```
Lead#debug ip pim 236.82.134.23
PIM debugging is on
Lead#
PIM: Check RP 172.16.224.1 into the (*, 236.82.134.23) entry
PIM: Send v2 Join on Serial1.605 to 172.16.2.254 for (172.16.224.1/32,
236.82.134.23), WC-bit, RPT-bit, S-bit
PIM: Building batch join message for 236.82.134.23
PIM: Send Join on Serial1.603 to 172.16.2.250 for (172.16.1.1/32, 236.82.134.23),
S-bit
PIM: Send v2 Prune on Serial1.605 to 172.16.2.254 for (172.16.1.1/32,
236.82.134.23), RPT-bit, S-bit
Lead#
```

由器观察包的源 IP 地址。通过查询它的单播路由表，发现源 IP 地址通过另一个接口(S1.603)能够到达，Lead 向 Aluminum 发送一个 Join 消息，在这两个路由器间建成了一个 SPT。当 Lead 开始通过 SPT 收到(172.16.1.1,236.82.134.23)的多播流量时，它向 RP 发送一个剪除消息，从共享树中剪除。

例 5-25 显示了 Lead 中的组 236.82.134.23 的路由条目。共享树的(*,G)条目仍然存在，只要路由器还有这个组的组员或下游邻居，那么这条路由就会一直保持。不过要注意，(S,G)条目显示了不同的输入接口和不同的 RPF 邻居。

例 5-25　Lead 上组 236.82.134.23 的路由条目显示路由器从 RPT 切换到了 SPT

```
Lead#show ip mroute 236.82.134.23
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 236.82.134.23), 00:26:26/00:02:58, RP 172.16.224.1, flags: SJC
  Incoming interface: Serial1.605, RPF nbr 172.16.2.254
  Outgoing interface list:
    Ethernet0, Forward/Sparse, 00:26:26/00:02:12

(172.16.1.1, 236.82.134.23), 00:26:26/00:02:36, flags: CJT
  Incoming interface: Serial1.603, RPF nbr 172.16.2.250
  Outgoing interface list:
    Ethernet0, Forward/Sparse, 00:26:26/00:02:12

Lead#
```

例 5-26 显示了 Aluminum 的路由条目，以及到 Lead 与 Brass 的 SPT；例 5-27 显示 Brass 的路由条目，到 Lead 的接口(S1.506)仍在(*,G)的输出接口列表中，但不在(S,G)的输出接口列表中。你可以观察到，Aluminum 在 SPT 上同时向 Lead 与 Brass 前转流量。在 Brass 中，到 Brass 的接口不在(S,G)输出接口列表中，因为 RP 不向这台路由器前转流量。

例 5-26　Aluminum 中组 236.82.134.23 的多播路由条目

```
Aluminum#show ip mroute 236.82.134.23
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop, State/Mode

(*, 236.82.134.23), 00:08:17/00:02:59, RP 172.16.224.1, flags: SJCF
  Incoming interface: Serial0/1.309, RPF nbr 172.16.2.234
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse, 00:07:33/00:02:30

(172.16.1.1/32, 236.82.134.23), 00:08:17/00:02:59, flags: CFT
  Incoming interface: Ethernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1.309, Forward/Sparse, 00:08:07/00:02:48
    Serial0/1.306, Forward/Sparse, 00:06:55/00:02:59

Aluminum#
```

例 5-27　Brass 中组 263.82.134.23 的条目

```
Brass#show ip mroute 236.82.134.23
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 236.82.134.23), 00:13:13/00:03:20, RP 172.16.224.1, flags: S
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial1.508, Forward/Sparse, 00:13:04/00:03:20
    Serial1.509, Forward/Sparse, 00:12:30/00:02:18
    Serial1.506, Forward/Sparse, 00:11:52/00:02:33

(172.16.1.1, 236.82.134.23), 00:13:14/00:02:59, flags: T
  Incoming interface: Serial1.509, RPF nbr 172.16.2.246
  Outgoing interface list:
    Serial1.508, Forward/Sparse, 00:13:05/00:02:49

Brass#
```

RFC2362 中规定了路由器“在数据量很大时”，应从 RPT 切换到 SPT。可是什么是大的数据量？答案是非常随意的。它可能包括路由器可用带宽总和、路由器上的阻塞、路由器的性能或是其他一些因素。作为网络管理员，你必须根据网络的特点作出判断。

Cisco 采用了一个简单的默认方式。Cisco 路由器从共享树上收到某一个(S,G)组的第一个包，马上加入到 SPT。这个默认方式可以用命令 **ip pim spt-threshold** 来改变。在这个命令中，切换到 SPT 的门限用 *k*(bit/s)来定义(默认为 0kbit/s)。路由器每秒钟测量一次到达的包的速率。如果任意组的包或某一特定组的包的速率超过了这一门限，那么这个路由器将切换到 SPT。当路由器切换到 SPT 时，它监视源流量的到达速率。如果组流量的速率低于配置门限的时间超过了 60s，那么路由器将尝试切回组的共享树。

可以用关键词 **infinity** 来防止路由器切换到 SPT。

有意思的是，如果最短路由经过 RP，则路由器也会切换到 SPT。在前一个例子中，路由器 Iron 仍在 RPT 中，原因是为了简化对 PIM-SM 树行为的介绍，Iron 的配置里加了 **ip pim spt-threshold infinity**。例 5-28 显示了 Iron 对于组 236.82.134.23 的路由条目。这条配置命令随后删除。再次观察这条路由，你可以看到 SPT 的门限设为默认值后，路由器马上切换到 SPT 上。RP 上的路由表仍然同例 5-27 中一样，因为到 Iron 的接口已经存在于(S,G)路由的输出接口列表中。

在例 5-28 与前面几张图中，J 标识不是与(*,G)条目相关联，就是与(S,G)条目相关联，或者是与两者均相关联。这是加入 SPT 标识。当与(*,G)相关联时，它表明沿共享树向下游传送的流量已经超过了 SPT 门限。如果还没有加入 SPT，它会在收到这个组的下一个包后开始加入。当与(S,G)条目相关联时，J 标识表示因为 RPT 流量超过了 SPT 门限，所以加入了 SPT。

表 5-11 中列出了与 mroute 相关联的所有标识。这个列表直接从 Cisco IOS 软件命令参考中取得。

例 5-28　在 STP 切换门限设成默认值的前后，Iron 中组 236.82.134.23 的条目

```
Iron#show ip mroute 236.82.134.23
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode
(*, 236.82.134.23), 00:00:57/00:02:59, RP 172.16.224.1, flags: SC
  Incoming interface: Serial1.708, RPF nbr 172.16.2.242
  Outgoing interface list:
    Ethernet0, Forward/Sparse, 00:00:57/00:02:02

Iron#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Iron(config)#no ip pim spt-threshold infinity
Iron(config)#^Z
Iron#
2d01h: %SYS-5-CONFIG_I: Configured from console by console

Iron#show ip mroute 236.82.134.23
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 236.82.134.23), 00:01:23/00:02:59, RP 172.16.224.1, flags: SJC
  Incoming interface: Serial1.708, RPF nbr 172.16.2.242
  Outgoing interface list:
    Ethernet0, Forward/Sparse, 00:01:23/00:02:34

(172.16.1.1, 236.82.134.23), 00:00:11/00:02:59, flags: CJT
  Incoming interface: Serial1.708, RPF nbr 172.16.2.242
  Outgoing interface list:
    Ethernet0, Forward/Sparse, 00:00:12/00:02:47

Iron#
```

表 5-11　　**mroute 标识**

标　识	描　述
D—密集	这条目是密集模式里的
S—稀疏	这条目是稀疏模式里的
C—连接的	多播组的组员直连在接口上
L—本地的	路由器本身是组员
P—剪除的	路由被剪除了
R—设置 RP 比特	表示(S,G)条目指向 RP，这是沿一共享树到达一多播源的典型剪除状态
F—注册标识	表示软件为一个多播源注册
T—设置 SPT 比特	表明在最短路径上收到包

续表

标　识	描　述
J—加入 SPT	对于(*,G)条目，表示沿这个组共享树传送的流量的速率超过了 SPT 门限(默认 SPT 门限是 0kbit/s)，当设置了 J 标识，收到下一个(S,G)包会触发路由器加入到源为根的多播树中。 对于(S,G)条目，表示这个条目已经因为超过了 SPT 门限而建立。当一个(S,G)条目设置了 J 标识，路由器监视多播源树的流量，当流量速率低于组的 SPT 门限的时间小于 1 分钟，路由器会试图切换回共享树。

5.9.6　PIMv2 消息格式

PIMv2 消息封装在 IP 包头中，协议号为 103。除了某些情况下用单播的消息外，PIMv2 的 IP 目的地址是保留的多播地址 224.0.0.13，TTL 设为 1。多播地址与 TTL 值均保证消息只能传到相邻的路由器。

虽然版本 2 是当前的版本，但 PIMv1 也很常见。这个版本的 IP 协议号为 2，使它成为 IGMP 协议的一个子集。版本 1 使用多播地址 224.0.0.2。Cisco IOS 在 11.3(2)T 开始支持 PIMv2。它提供了对 PIMv1 的后向兼容，如果一个接口检测到邻居为版本 1，那么它会自动切换到版本 1。一个接口可以人工地通过命令 **ip pim version** 来设定 PIMv1 或 PIMv2。

因为篇幅的原故，本书只涉及 PIMv2 的消息格式。至于 PIMv1 的格式，可参阅相应的 Internet 草案。

你会注意到，有些消息类型的某些项的标签涉及到编码的地址。关于编码的格式与这些项的细节可以参阅 RFC 2362。

下述消息的所有保留项均设为全 0，或在接收时忽略。

1．PIMv2 消息头格式

所有 PIM 消息均有标准的包头，如图 5-62 所示。

图 5-62　PIMv2 消息头的格式

PIMv2 消息头中的各项如下所述：

- 版本定义了版本号，当前版本号为 2，虽然 PIMv1 也在广泛使用。
- 类型定义了包头后面的 PIM 消息类型，表 5-12 列出了 PIMv2 消息的类型。

表 5-12　　**PIMv2 消息类型**

类　型	消　息
0	Hello
1	Register(只在 PIM-SM 中使用)

续表

类　型	消　息
2	Register-Stop(只在 PIM-SM 中使用)
3	Join/Prune
4	Bootstrap(只在 PIM-SM 中使用)
5	Assert
6	Graft(只在 PIM-DM 中使用)
7	Graft-Ack(只在 PIM-DM 中使用)
8	Candidate-RP-Advertisement(只在 PIM-SM 中使用)

• 校验和是一个标准的 IP 风格的校验和，用 16bit PIM 消息反码和的反码，不包括 Register 消息中的数据部分。

2．PIMv2 Hello 消息格式

PIMv2 Hello 消息的格式如图 5-63 所示，用于对邻居的发现和邻居的生存保持。这个消息默认为每 30s 发送一个，可以用 **ip pim query-interval** 来改变这个周期值。

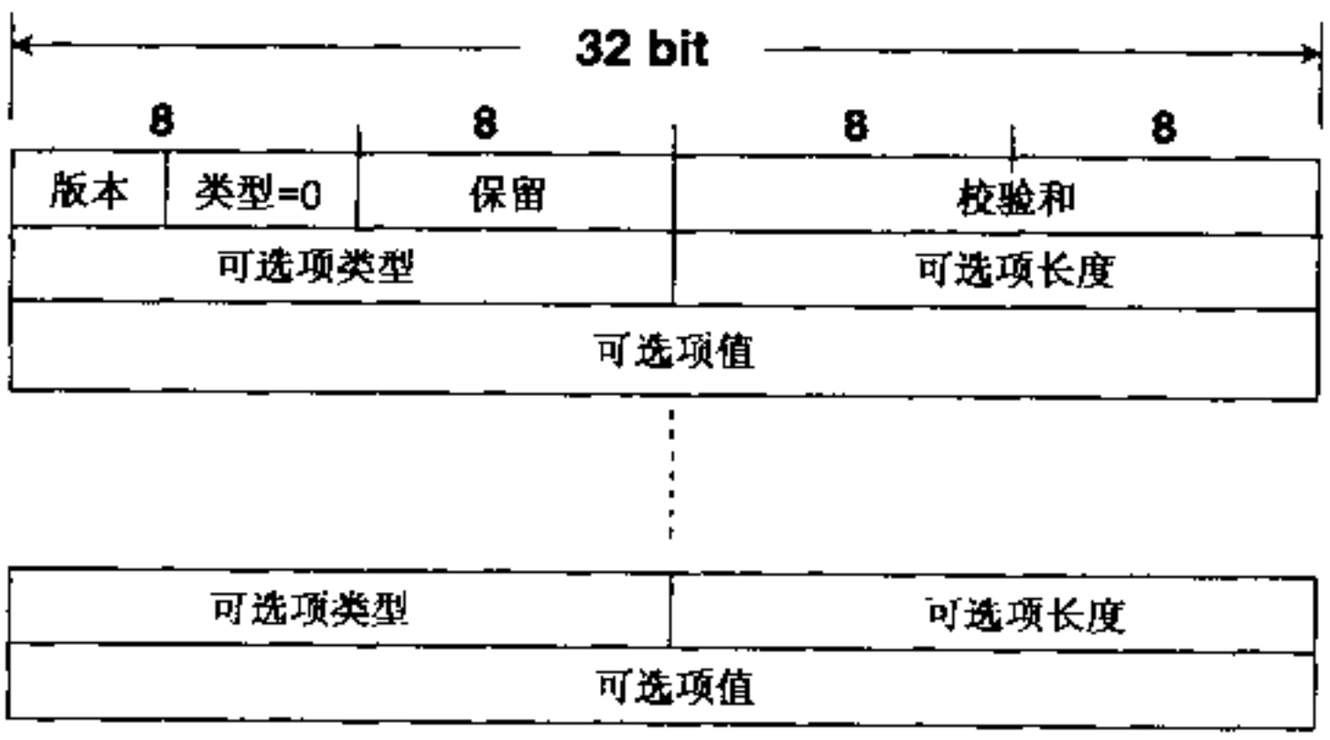


图 5-63　PIMv2Hello 消息格式

PIMv2 Hello 消息各项的定义如下：

• 可选项类型定义了可选项值中的可选项类型。当前，只有可选项类型 1，它定义可选项类型为保持时间，2～16 是保留的。

• 可选项长度定义了以字节为单位的可选项值的长度。当可选项值为保持时间(可选项类型为 1)时，可选项长度为 2。

• 可选项值为一可变长的项，能装下可选项类型中定义的任何值。保持时间(可选项类型＝1，可选项长度＝2)为路由器在宣布邻居无效前，等待从它发出的 Hello 消息的时间，保持时间为 Hello 间隔的 3.5 倍。

这个格式中可以看到多个 TLV(类型/长度/值)可以在一个 Hello 消息里承载。

3．PIMv2 Register 消息格式

Register 消息格式如 5-64 所示，这个消息只用于 PIM-SM，是从源 DR 向 RP 用单播发送的一个消息，它承载着从源发出的初始多播包。就是说，在还没有建立从源 DR 到 RP 的 SPT 之前，用 Register 消息把多播流量从源传送到 RP。

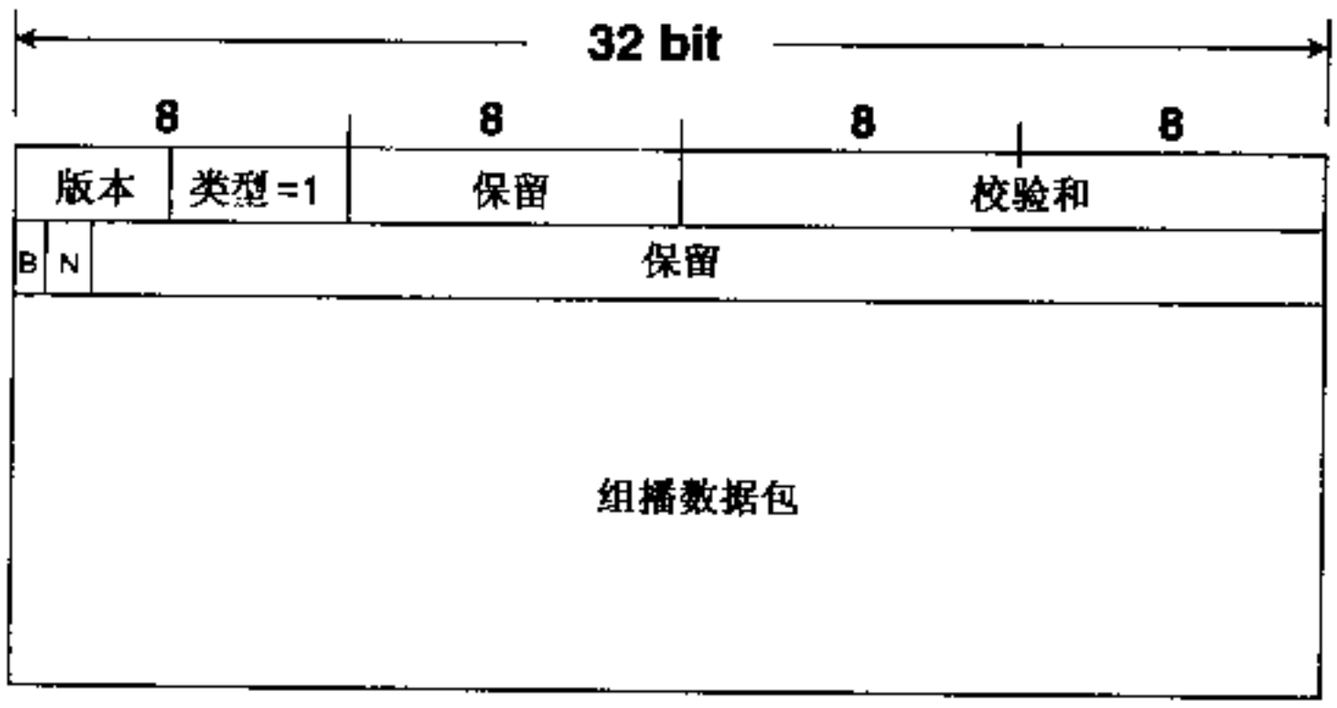


图 5-64　PIMv2 Register 消息格式

PIMv2 Register 消息各项定义如下：

- 校验和，在 Register 消息里，只计算消息头，而不包括数据部分。
- *B* 为边界比特。如果消息发起者为一个直连着源的 DR，那么这个比特设为 0；如果源为 PIM 多播边界路由器(PIM Multicast Border Router PMBR)，则这个比特设为 1。PMBR 与其他域间多播的问题将在第 7 章中讨论。
- *N* 为空注册比特。DR 在注册抑制计时器超时前查找 RP 时设置这个比特为 1。
- 多播数据包为来自源的一个多播包，它通过 Register 消息传到 RP。

4．PIMv2 Register Stop 消息格式

Register Stop 消息的格式如图 5-65 所示，这个消息用于 RP 响应 DR 产生的 Register 消息。这个消息在两种情况下使用：

- RP 通过 SPT 收到多播包，而不再需要收到 Register 消息封装的多播消息；
- 没有直连的，或连在 SPT、RPT 上的组员等待 RP 前转这个包。

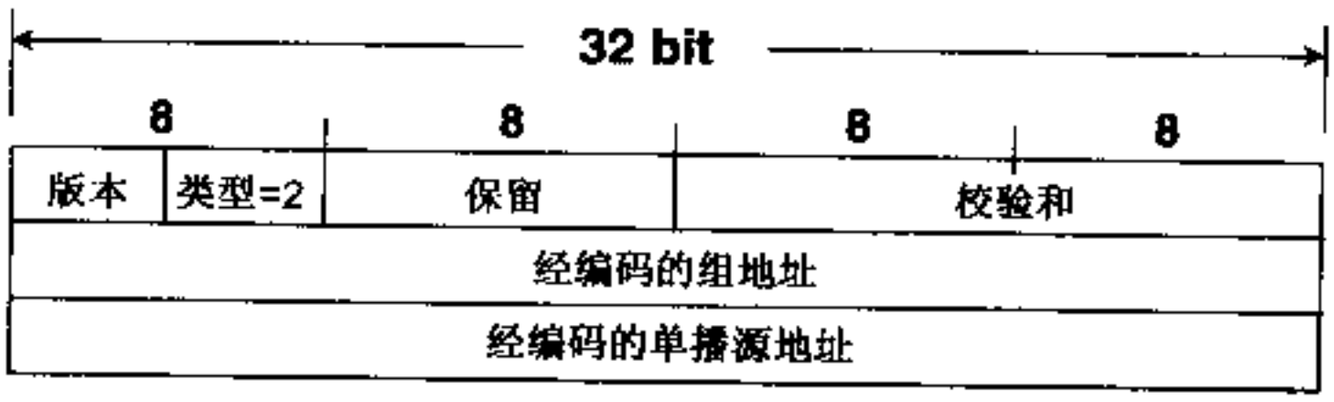


图 5-65　PIMv2 Register Stop 消息格式

PIMv2 Register Stop 消息各项定义如下：

- 经编码的多播地址是多播组的 IP 地址，对于这个地址，接收者要停止发送 Register 消息。
- 经编码的单播源地址为多播源的 IP 地址，这项也可被(*,G)路由条目定义为全 0，作

为通配的源地址。

5．PIMv2 Join/Prune 消息格式

Join/Prune 消息的格式如图 5-66 所示。这个消息将向上游 RP 或多播源发送，用于加入或退出 RPT 或 SPT。这个消息由多播地址列表组成。对于每一个多播地址，有一个或多个源地址列表。这些列表共同定义了要加入和退出的(S,G)与(*,G)路由条目。

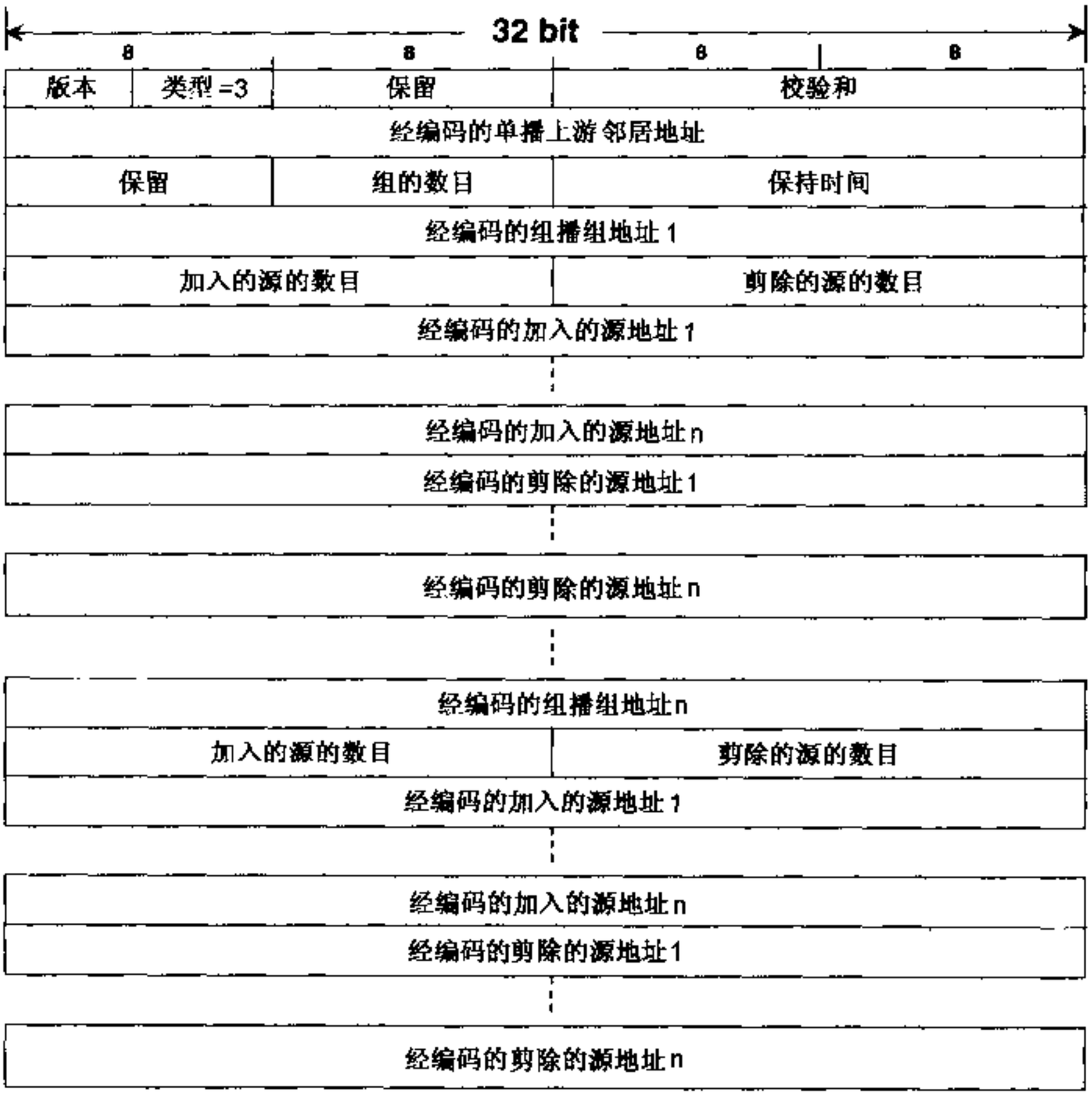


图 5-66　PIMv2 Join/Prune 消息格式

PIMv2 Join/Prune 消息中各项定义如下：

- 经编码的单播上游邻居地址是 RPF 或上游邻居的地址，这个消息将发送到这个地址上。
- 多播组数目定义了这个消息里包含的多播组的数目。
- 经编码的多播组地址定义了多播组的 IP 地址。
- 加入的多播源的数目定义了在这个多播组地址中列出的“经编码的加入多播源地址”的数目。
- 剪除的多播源的数目定义了这个多播地址中列出的“经编码的剪除的多播源地址”的数目。

• 经编码的加入多播源地址定义(S,G)对的源地址或(*,G)对的通配地址。(*,*,RP)三元组的两个通配符也可在此定义。除了源地址外，这一项中还规定了 3 个标识：

— S 为稀疏标识比特。这个比特设为 1 表示 PIM-SM，用于与版本 1 的兼容。

— W 为通配(WC)比特。如果它设为 1，则经编码的加入多播源地址代表(*,G)或(*,*,RP)路由条目中的通配符。当它设为 0 时，经编码的加入多播源地址代表(S,G)路由条目中的源地址。当向 RP 发送了 Join 消息，W 比特必须设为 1。

— R 为 RPT 比特。当这个比特设为 1 时，将向 RP 发送一个 Join 消息。当这个比特设为 0 时，向多播源发送 Join 消息。

• 经编码的剪除的多播源地址定义了被剪除的多播源地址。这个编码与经编码的加入多播源地址项相同，S、W、R 比特同样与加入的地址一样，适用于被剪除的地址。

6. PIMv2 Bootstrap 消息格式

Bootstrap 消息的格式如图 5-67 所示，它由自举路由器(BSR)每 60s 产生一个，扩散到整个 PIM-SM 域中，以保证一个组中的所有路由器判断的 RP 相同。这个消息包含了一个或多个多播地址的列表，对于每一个组地址，都有一个候选 RP(C-RP)及它们的优先权的列表。这个列表为这个组的 RP 集。收到消息的路由器用同样的算法从 C-CP 列表中算出这个组的 RP。

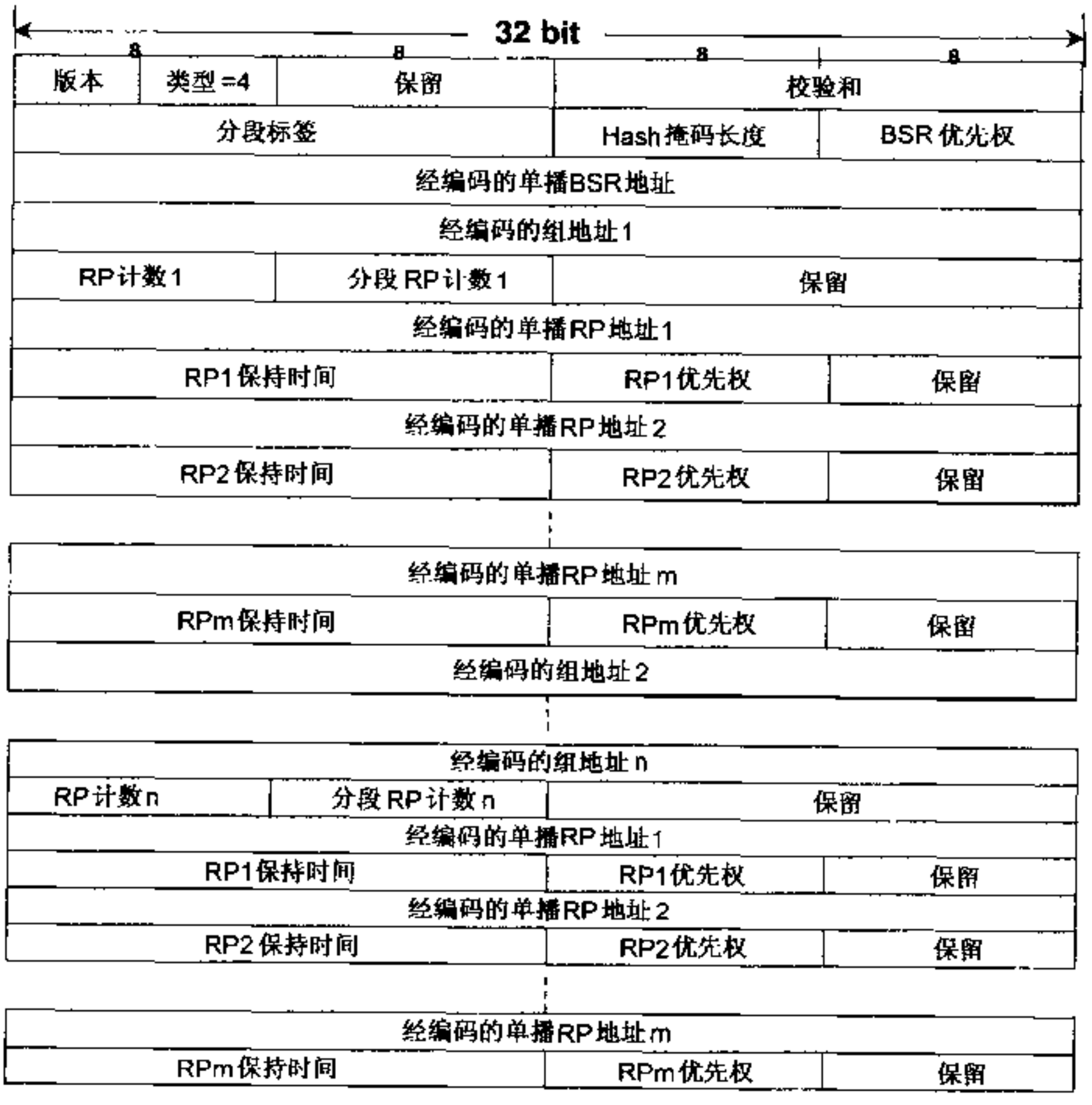


图 5-67　PIMv2 Bootstrap 消息格式

这个算法的设计保证了所有在 PIM 域中的路由器能得出相同的 RP 地址。Bootstrap 消息如同“自举协议”一节所述，也用于选举 BSR。

PIMv2 Bootstrap 消息中各项定义如下：

- 分段标签用于 Bootstrap 消息必须分为多个部分的情况，因为消息的长度超过了包的最大长度。这个分段标签为一个随机产生的数，相同消息的片段分配同一个标签。就是说，所有同一 Bootstrap 消息的片段分段标签项是相同的。
- hash 掩码的长度为用于 hash 算法的掩码的长度。这个掩码的长度可以用 **ip pim bsr-candidate** 来设置。
- BSR 优先权为一个 0～255 的值，它定义了发出这个消息的 BSR 的优先权。有最高优先权的 C-BSR 会成为 BSR。这个优先权可以用 **ip pim bsr-candidate** 命令设置。
- 经编码的单播 BSR 地址为这个域 BSR 的地址。
- 经编码的组地址为一个多播组的地址。
- RP 计数定义了对一个多播组列出的 C-RP 的总数，也就是 RP 集的大小。对 RP 集大小的描述是很重要的，因为如果 Bootstrap 消息是分段的，一个分段丢失了，那么在 PIM 域中对 RP 的判断可能会不一致。因此，如果收到的 RP 集中的 RP 数目与 RP 数目不符，则整个 RP 集会被丢弃。
- 分段 RP 计数定义了对于某个组在这个分段中的 C-RP 的数目。
- 经编码的单播 RP 地址为 C-RP 的 IP 地址。
- RP 保持时间为一个 BSR 在把一个 C-RP 从 RP 集中删除前等待 C-RP 发出 Candidate-RP-Advertisement 消息的时间，这个保持时间为 150s。
- RP 优先权为一个 0～255 的值，用于选择 RP 的算法中。最高的优先权为 0。

7．PIMv2 Assert 消息格式

PIMv2 Assert 消息的格式如图 5-68 所示，这个消息用于在多路访问网络中选举一个指定的前转器当一台 PIM 路由器在它的输出接口上收到同一个组的多播包时，路由器就认为一定有一台路由器连接到前转至那个组的数据链路。路由器发出 Assert 消息，这样，这个多路访问网络中的其他路由器可以决定它们中间谁来前转这个组的数据包。

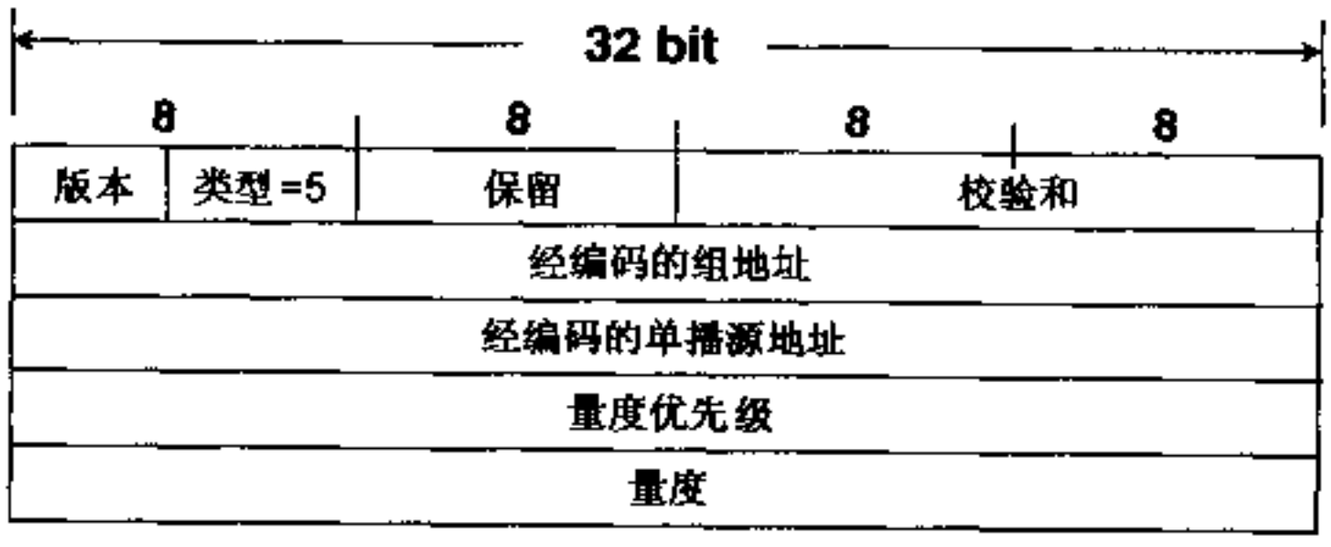


图 5-68　PIMv2 Assert 消息格式

PIMv2 Assert 消息各项定义如下：

- 经编码的组地址为这个包的多播 IP 地址，这个地址触发了 Assert 消息。
- 经编码的单播源地址为触发 Assert 消息的多播包的源地址。
- 量度优先值为分配给提供到源的路由的单播路由协议的优先权值。这个值与管理距离

的使用相同，在比较不同路由协议发现的路由时提供一致的量度。

- 量度是与发出消息的路由器的单播路由表到源的路由相关的量度。

8．PIMv2 Graft 消息格式

一台 PIM-DM 路由器向上游邻居发出 PIMv2 Graft 消息，请求再次加入到以前剪除的树中，它的格式与图 5-66 中 Join/Prune 消息相同，只是类型＝6。

9．PIMv2 Graft-Ack 消息格式

一台 PIM-DM 路由器向下游邻居发送 PIMv2 Graft-Ack 消息，以响应其 Graft 消息。Graft-Ack 消息的格式与图 5-66 中相同，只是类型＝7。

10．PIMv2 Candidate-RP-Advertisement 消息格式

候选 RP(C-RP)向 BSR 周期性单播发送 Candidate-RP-Advertisement 消息。BSR 用这个消息中的信息来建立 RP 集，这个 RP 集通过 Bootstrap 消息域中所有的 PIM-SM 路由器来对外宣告。图 5-69 显示了 Candidate-RP-Advertisemet 消息的格式。

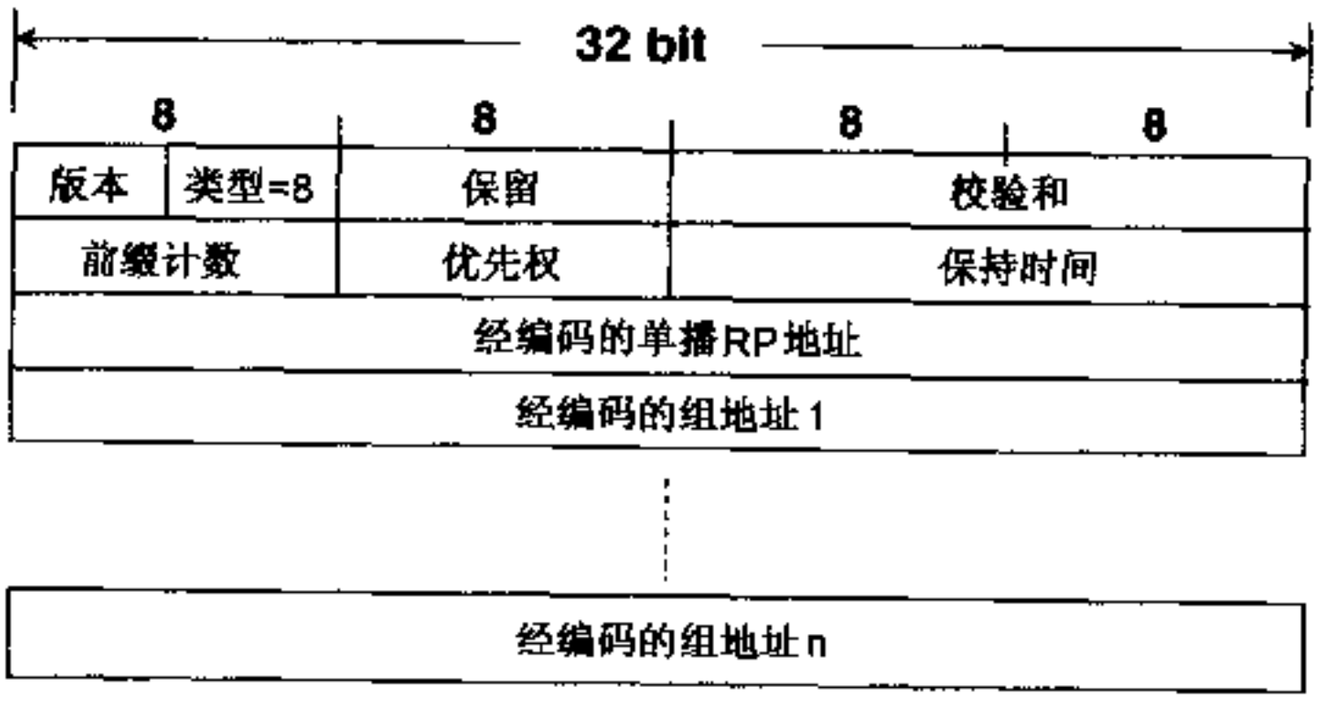


图 5-69　Candidate-RP-Advertisement 消息格式

Candidate-RP-Advertisemet 消息各项的定义如下：

- 前缀数目定义了包括在这个消息中的多播组地址的数目。如果发起消息的是一台多播域中所有多播组的 C-RP，则前缀数目为 0。
- 优先权为一个 0~255 的值，它定义了发起 C-RP 的优先权，这个值被用于选择 RP 的算法中。优先权与其值的大小相反：0 为最高的优先权，255 为最低。
- 保持时间为这个消息有效的时间。
- 经编码的单播 RP 地址为 C-RP 的地址，这个地址是路由器一个接口的 IP 地址，这个地址通常为 loopback 地址。
- 经编码的组地址定义了一个或多个多播组的地址。对于这个多播组，消息发起者为一个候选 RP。

5.10　尾注

[1]Steve E. Deering, "RFC 988: Host Extensions for IP Multicasting," RFC988, 1986 年 7 月。这一 RFC 文件已经作废，最新的版本是 RFC1112。

[2]Tomas Pusateri, "RFC 1469: IP Multicast over Token-Ring Local Area Networks," (工作正在进行中)。这个文件中实际推荐了 3 种支持多播的方法，但第 3 种并没有使用。

[3]Steve Deering, "RFC 1112: Host Extension for IP Multicasting," 1998 年 8 月。现已作废的 IGMPv0 在 RFC 998 中描述。

[4]William C. Fenner, "RFC 2236: Internet Group Management Protocol, Version 2," (工作正在进行中)。

[5]Brad Cain, Steve Deering, Ajit Thyagarajan, "Internet Group Management Protocol, Version 3"，<draft-ieft-idmr-igmp-v3-01.txt> 1999 年 2 月。

[6]Dave Katz, "RFC 2113: IP Router Alert Option," (工作正在进行中)。

[7]David Meyer, "RFC 2365: Administratively Scoped IP Multicst," (工作正在进行中)。

[8]D. Waitxman, C. Partridga, S. Deering, "RFC 1075: Distance Vector Multicast Routing Protocol," (工作正在进行中)。

[9]Thomas Pusateri, "Distance Vector Multicast Routing Protocol," draft-ietf- idmr-dvmrp-v3-09, 1999 年 9 月。

[10]John Moy, "RFC 1584: Multicast Extensions to OSPF," (工作正在进行中)。

[11]Tony Ballardie, "RFC 2189: Core Based Trees (CBT Version 2) Multicast Routing," (工作正在进行中)。

[12]Stephen Deering et al., "Protocol Independent Multicast Version 2 Dense Mode Specification," draft-ietf-pim-v2-dm-03.txt, 1999 年 3 月。

[13]Deborah Estrin et al., "RFC 2362: Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification," (工作正在进行中)。

5.11　展望

在这两卷关于所有讨论的路由协议中，多播对于大部分人来说可能是最不熟悉的。虽然这一章对 5 种协议和与其相关的支持协议进行了适度的介绍，但并不很详尽，还有很多 IP 多播的内容，本书没有覆盖。关于其他的内容，可以参阅“推荐读物”。

现在你对这 5 种协议如何工作有了一定的理解，第 6 章提供了具体的 Cisco 的例子，说明对 IP 多播路由如何配置，如何查错。

你可以认为本章中涉及的协议为多播 IGP。所有的协议都是在单一的域中进行操作的。在第 7 章中，你会去分析 AS 间的多播路由。

5.12　推荐读物

Beau Williamson, Developing IP Multicst Network, Indianapolis, IN, Cisco Press, 2000.

5.13　命令归纳

表 5-13 列出了本章中用到的命令。

表 5-13　　命令总结

命　令	描　述
clear ip cgmp [类型号]	使 CGMP 发送 Leave 消息，以清除所有从 Catalyst 交换机收到的多播组条目
debug ip igmp	显示收到和发送的 IGMP 消息，也包括 IGMP 与主机相关的事件
debug ip 包 [访问表号|detail]	显示收到和发送的 IP 包，可选用访问表在显示时过滤不关心的包
debug ip pim [组]	显示收到和发送的 PIM 消息，包括 PIM 相关的事件
ip igmp query-interval (s)	改变 IGMP 查询消息默认的每 60s 发送一次的时间间隔
ip igmp query-max-response-time (s)	改变 IGMP 查询消息中宣告的最大响应时间，这个时间间隔默认为 10s
ip igmp quiry-timeout (s)	改变路由器在担当 DR 或查询者前等待收到 DR 或查询者的 IGMP 查询消息的默认时间，这个值默认为 2 倍的查询间隔
ip igmp version {1|2}	设置一接口上 IGMP 的版本，默认为 2
ip multicast use-functional	将 IP 多播地址映射到令牌环的特定地址 0xC000.0004.0000
ip pim bsr-candidate 类型号 hash 码长度 [优先权]	配置路由器宣布自己为 C-BSR
ip pim message-interval (s)	改变稀疏模式中 Join/Prune 消息每 60s 发送一次的默认时间间隔
ip pim query-interval (s)	改变 PIMv2 发送 Hello 消息或 PIMv1 发送 Router Query 消息时默认的 30s 的时间间隔
ip pim spt-threshold {kbit/s|infinity} [**group-list** 访问表号]	定义 PIM-SM 路由器从 RPT 切换到 SPT 时的接收速率默认为收到第一个多播包时就切换
ip pim version [1|2]	设置一个接口使用 PIMv1 还是 PIMv2，默认为 2
show ip igmp groups [组名|组地址|类型号]	显示多播组直连的组员列表，这是通过 IGMP 知道的
show ip mroute [组名|组地址|类型号] [**summary**] [**count**] [**active** kbit/s]	显示 IP 多播路由的内容
show ip pim neighbor [类型号]	显示通过 PIMv1 Router Query 消息或 PIMv2 Hello 消息发现的邻居列表
show ip pim rp [组名|组地址|**mapping**]	显示与多播组或 mroute 有关的激活的 RP
show ip route [地址[掩码] [**longer-prefixes**]]|[协议 [进程 id]]	显示单播路由表里的路由

5.14　复习问题

1．给出几个原因，说明为什么复制的单播对于一个大型的网络不是真正的多播？

2．哪一个范围的地址保留给 IP 多播？

3．一个单独的 D 类前缀可以有多少个子网？

4．路由器处理目的地址在 224.0.0.1～224.0.0.225 范围内的包与其他多播地址的包有什么不同？

5．写出下面 IP 地址的正确的以太网 MAC 地址：

(a) 239.187.3.201

(b) 224.18.50.1

(c) 224.0.1.87

6．哪些多播 IP 地址可以用 MAC 地址 0100.5E06.2D54 来表示？

7．为什么令牌环对传送多播是一种不好的介质？

8．什么是加入时延？

9. 什么是退出时延？

10. 什么是多播 DR(或者说查询者)？

11. 什么设备发送 IGMP Query 消息？

12. 什么设备发送 IGMP Membership Report 消息？

13. 如何使用 IGMP Membership Report 消息？

14. General IGMP Query 与 Group-Specific IGMP Query 在功能上有什么区别？

15. IGMPv2 与 IGMPv1 兼容吗？

16. IGMP 的 IP 协议号是多少？

17. Cisco 组员资格协议(CGMP)的目的是什么？

18. IP 监听比 CGMP 有什么优点？可能的缺点是什么？

19. 什么设备发送 CGMP 消息：路由器、以太网交换机，还是两者都可以？

20. 什么是反向路径前转(RPF)?

21. 多少主机构成密集模式，多少主机构成稀疏模式？

22. 显式加入与隐式加入相比，最主要的优点是什么？

23. 基于源的多播树与共享多播树在结构上最主要的区别是什么？

24. 什么是多播的限制？

25. IP 多播限制的两种方法是什么？

26. 从多播路由器的角度看，什么是上游，什么是下游？

27. 什么是 RPF 检查？

28. 什么是剪除？什么是接入？

29. 什么是剪除生存期？当剪除生存期结束时会发生什么？

30. 什么是路由依赖？DVMPR 如何表示路由依赖？

31. DVMRP 是密集模式的协议，还是稀疏模式的协议？

32. MOSPF 是密集模式的协议，还是稀疏模式的协议？

33. MOSPF 专有的 LSA 名称和类型号是多少？

34. MOSPF 能不能与不支持 MOSPF 路由器建立邻接的关系？

35. 定义下列 MOSPF 路由器类型：

(a) 区域间多播前转器

(b) AS 间多播前转器

(c) 通配多播接收者

36. CBT 是密集模式的协议，还是稀疏模式的协议？

37. 什么是 CBT 父路由器和 CBT 子路由器？

38. 描述 CBT DR 从源向核心传送包的两种方法，在何种情况下用哪种方法？

39. 什么是 PIM 剪除覆盖？

40. 什么是 PIM 前转器？它是怎么选出来的？

41. PIM 用什么标准来选出 DR？

42. 什么是 PIM SPT？什么是 PIM RPT？

43. Cisco 路由器自动发现 PIM-SM RP 的两种方法是什么？

44. 问题 43 中的机制中，哪一个可以用在多厂商路由器的拓扑中？

45. 什么是 C-RP？

46. 什么是 BSR？

47. 什么是 RP 映射代理？

48. (S,G)多播路由条目与(*,G)多播路由条目的区别是什么？

49. CBT 在源与核心间的双向树与 PIM-SM 在 RP 和源间的单向树不同，其最大缺陷是什么？

50. 什么是 PIM-SM 源的注册？

51. 什么时候 Cisco 路由器从 PIM-SM RPT 切换到 SPT?

第 6 章　IP 多播路由的配置和故障排除

- **配置 IP 多播路由器**——在这一章中讲述用 Cisco IOS 软件配置 PIM-DM 和 PIM-SM，并且举出了案例研究来进一步理解这些协议的配置。
- **IP 多播路由协议的故障排除**——在这一章讨论用 Cisco IOS 软件进行 IP 多播的一些工具，并举出一些查错中的技巧。

你已经在第 5 章“IP 多播路由的介绍”中学到各种 IP 多播路由协议，你也了解到 Cisco 与大多数厂商所选择的是 PIM 协议，不论其是密集模式还是稀疏模式。现在你已经明白基本的 PIM 的操作，这一章将着重于用 Cisco IOS 软件进行 PIM-DM 与 PIM-SM 的配置与查错。

6.1　配置 IP 多播路由

在你可以配置某一种 IP 多播路由协议之前，你必须对路由器进行一般性地、中立于路由协议的设置。

例 6-1 显示了一个配置，在这个配置中需要 **ip multicast-routing** 命令来启动多播路由的支持；在这个基本的配置中可能需要根据具体的应用加入其他的命令。这个配置中的几个命令你可能已经用过。这些命令中，只有 **ip multicast-routing** 才是你所需要的。如同默认的 **ip routing**(因此没有显示)打开单播 IP 路由一样，这条命令启动了所有 IP 多播功能的支持。

> 注：“与协议无关”是比“中立于协议”更贴切的术语，可是它会与 PIM 的名称产生混淆。

例 6-1　ip multicast-routing 命令

```
version 12.1
!
hostname Stovepipe
!
ip multicast-routing
ip dvmrp route-limit 20000
!
interface Ethernet0
 ip address 172.17.1.1 255.255.255.0
 ip igmp version 1
!
interface Ethernet1
 ip address 172.17.2.1 255.255.255.0
 ip cgmp
!
interface Serial0
 ip address 172.18.1.254 255.255.255.252
 no ip mroute-cache
```

```
!
interface TokenRing0
 ip address 172.16.2.1 255.255.255.0
 ip multicast use-functional
 ring-speed 16
!
interface TokenRing1
 ip address 172.16.1.1 255.255.255.0
 ring-speed 16
```

这些配置命令中有趣的一点是，没有一个用于启动 IGMP 的命令。当 IP 多播路由在一台路由器上启动后，IGMPv2 会自动运行在 LAN 接口上。这些配置命令中，唯一一个 IGMP 命令是接口 E0 上 **ip igmp version** 命令，把默认的版本设成 IGMPv1。表 6-1 中列出了所有改变某一接口上 IGMP 默认值的命令。其他 IGMP 命令也会在本章稍后列出。

表 6-1　　IGMP 接口命令

命　令	默 认 值	描　述
ip igmp query-interval 秒	60	路由器查询一个接口上连接的组员的频率
ip igmp query-max-response-time 秒	10	IGMP 查询消息中的 Max-Response-Time 值，告诉主机，路由器在删除一个多播组前会等待多久。这个命令只对 IGMPv2 有效
ip igmp query-timeout 秒	2 倍查询的间隔	路由器成为查询者前等待接收其他路由器的查询消息的时间
ip igmp verion {1|2}	2	将相关的接口设为 IGMPv1 或 IGMPv2

例 6-1 中接口 E1 的配置包括 **ip cgmp** 命令，这使路由器为一台 Catalyst 交换机产生 CGMP 消息。另一个选项是 **ip cgmp proxy**，这个命令在子网中的其他路由器没有 CGMP 功能时使用。这个命令告诉路由器，在它的 CGMP 消息中宣告那些不支持 CGMP 的路由器。如果你配置一台 Cisco 路由器作为 CGMP 代理，你必须保证这台路由器被选为 IGMP 查询者。

例 6-1 中下一个有趣的命令是接口 S0 上的 **no ip mroute-cache**。这个命令就像 **no ip route-cache** 命令关闭快速单播 IP 包交换一样，关闭了快速多播包交换。你可以根据关闭快速单播包交换一样的原因关闭快速多播包交换，比如要实现基于包的负载均衡，而不是基于目的地的负载均衡。

接口 TO0 的配置包括 **ip multicast use-functional** 命令，但 TO1 上并没有使用。这一结果导致 TO0 多播 IP 包映射到令牌环的专用地址 0xC000.0004.0000。TO1 却将多播地址映射到广播地址 0xFFFF.FFFF.FFFF。

6.2　案例研究：配置与协议无关多播，密集模式(PIM-DM)

在 Cisco 路由器上启动了 IP 多播后，你可以简单地在所有接口上加上命令 **ip pim dense-mode** 来启动 PIM-DM。图 6-1 显示了一个简单的 PIM-DM 拓扑结构，例 6-2 中显示了

路由器 Porkpie 的配置，其他路由器的配置与 Porkpie 很相似。

当配置 PIM-DM 时，在例 6-2 中给出很重要的两点考虑，第一点显而易见，就是单播路由协议——这里是 OSPF——必须运行起来。否则，PIM 没有办法来判断反向路径前转(Reverse Path Forward　RPF)接口。第二点考虑可以将例 6-2 与拓扑图图 6-1 比较看出，当配置 PIM 时，这个协议要在每个接口上都启动。否则，可能会有使 RPF 检查失败的危险。

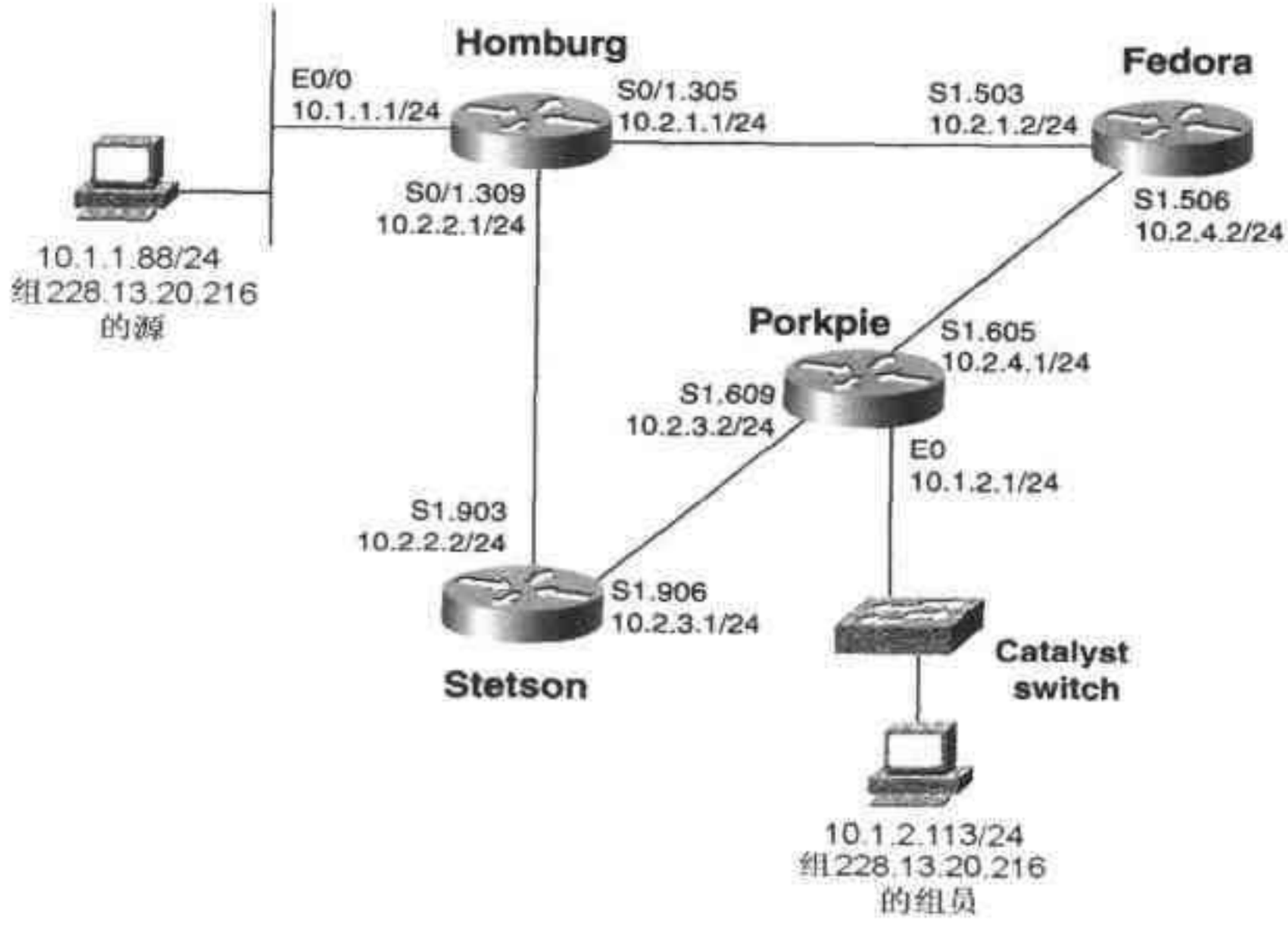


图 6-1　基本 PIM-DM 功能的演示

例 6-2　在一个接口上用 ip Pim dense-mode 命令启动 PIM-DM

```
hostname Porkpie
!
ip multicast-routing
!
interface Ethernet0
 ip address 10.1.2.1 255.255.255.0
 ip pim dense-mode
 ip cgmp
!
interface Serial1
 no ip address
 encapsulation frame-relay
 no ip mroute-cache
!
interface Serial1.605 point-to-point
 description PVC to Fedora
 ip address 10.2.4.1 255.255.255.0
 ip pim dense-mode
 no ip mroute-cache
 frame-relay interface-dlci 605
!
interface Serial1.609 point-to-point
 description PVC to Stetson
 ip address 10.2.3.2 255.255.255.0
```

```
 ip pim dense-mode
 no ip mroute-cache
 frame-relay interface-dlci 609
!
router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
!
```

例 6-3 中显示了当成员 10.1.2.113 加入多播组中和源 10.1.1.88 开始发送数据后，组 228.13.20.216 Porkpie 的 mroute 条目。第 5 章中 PIM-DM 部分为了清晰，只显示了(S,G)条目，实际上除了(S,G)外还会有(*,G)的条目，这个(*,G)条目不是 PIM-DM 规范的一部分，也不用于包的前转。但 Cisco IOS 软件创造这个条目作为(S,G)的“父”数据结构。所有连接着 PIM 邻居的接口和所有直连着组员的接口，都加到(*,G)条目的输出接口列表中，运行 PIM-DM 时，这个条目的输入列表总是空的。(S,G)条目的输入输出接口都是取自这一列表。

注：Cisco IOS Software Release 12.1 在这一章开始写作之前开始发行，并在演示路由器上安装，因此，你会注意到如 **show ip mroute** 和 **show ip route** 命令各项格式与较早章节有所不同。

例 6-3　Porkpie 上组 228.13.20.21 的条目

```
Porkpie#show ip mroute 228.13.20.216
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
       M - MSDP created entry, X - Proxy Join Timer Running
       A - Advertised via MSDP
Outgoing interface flags: H - Hardware switched
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 228.13.20.216), 20:06:06/00:02:59, RP 0.0.0.0, flags: DJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0, Forward/Dense, 20:05:25/00:00:00
    Serial1.609, Forward/Dense, 00:03:32/00:00:00
    Serial1.605, Forward/Dense, 00:03:32/00:00:00

(10.1.1.88, 228.13.20.216), 00:03:21/00:02:59, flags: CT
  Incoming interface: Serial1.605, RPF nbr 10.2.4.2
  Outgoing interface list:
    Ethernet0, Forward/Dense, 00:03:21/00:00:00
    Serial1.609, Prune/Dense, 00:03:21/00:00:03

Porkpie#
```

在例 6-3 中，你可以看到 E0，S1.609 和 S1.605 均在(*,G)输出接口列表中。S1.605 在(S,G)条目中做为 RPF 接口，包可从 E0 上前转。S1.609 也在输出列表中，但被剪除。

如第 5 章所讨论的一样，PIM(以及其他任何使用 RPF 检测的协议)只会有一个输入接口。例 6-4 中显示了 Porkpie 的单播路由表。表中有两条到子网 10.1.1.0/24 等开销的通路，所以

PIM 选择有较高 IP 地址值的接口作为 RPF 接口，来打破这个平局。在例 6-4 中，这个地址是 S1.605 的 10.2.4.2。查看一下例 6-3 证实了这个接口在输入接口列表中。

例 6-4　Porkpie 的单播路由表

```
Porkpie#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/24 is subnetted, 6 subnets
O       10.2.1.0 [110/128] via 10.2.4.2, 00:15:07, Serial1.605
C       10.1.2.0 is directly connected, Ethernet0
O       10.2.2.0 [110/128] via 10.2.3.1, 00:15:07, Serial1.609
O       10.1.1.0 [110/138] via 10.2.4.2, 00:15:07, Serial1.605
                 [110/138] via 10.2.3.1, 00:15:07, Serial1.609
C       10.2.3.0 is directly connected, Serial1.609
C       10.2.4.0 is directly connected, Serial1.605
Porkpie#
```

在图 6-2 中，另一路由器加入到因特网中，这台路由器 Bowler 连在一台以太网交换机上，并与 Porkie 共享一个多路访问链路。在第 5 章中讨论到的 IGMP 查询者、PIM 指定路由器、PIM 前转器的规则如下所示：

- 有最小 IP 地址值的路由器成为 IGMPv2 查询者。

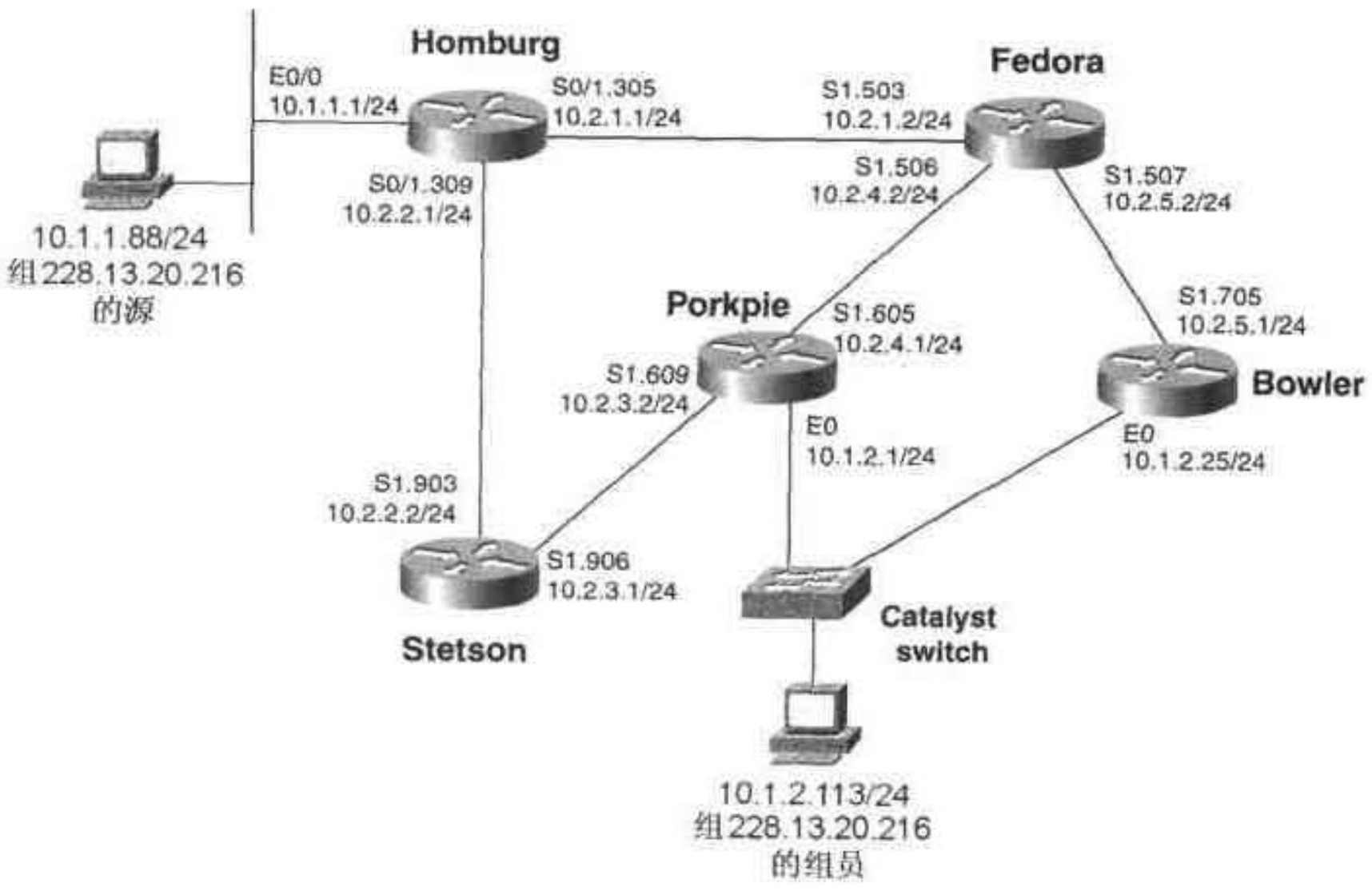


图 6-2　在图 6-1 中加入一台路由器 Bowler

- 有最高 IP 地址值的路由器成为 PIM 指定路由器(DR)。这个 DR 在运行 IGMPv1 的子网中才是很重要的。
- PIM 前转器是具有到达多播源的最低管理距离的路由器。如果管理距离相同，选择到达多播源的路由有最低量度的路由器。如果两者均相同，选择具有最高 IP 地址值的路由器做为前转器。

例 6-5 中显示了 IGMPv2 查询者和 PIM 指定路由器均已选出。Porkpie(10.1.2.1)在子网中有最小的 IP 地址值，所以成为 IGMP 查询者。Bowler(10.2.1.25)有最高的 IP 地址值，所以成为 DR。Porkpie 和 Bowler 均运行 IGMPv2，所以 DR 在此并不重要。

例 6-5　Porkpie(10.1.2.1)为 IGMP 查询者，Bowler 是 PIM DR

```
Bowler#show ip igmp interface ethernet 0
Ethernet0 is up, line protocol is up
  Internet address is 10.1.2.25/24
  IGMP is enabled on interface
  Current IGMP version is 2
  CGMP is enabled on interface
  IGMP query interval is 60 seconds
  IGMP querier timeout is 120 seconds
  IGMP max query response time is 10 seconds
  Last member query response interval is 1000 ms
  Inbound IGMP access group is not set
  IGMP activity: 6 joins, 2 leaves
  Multicast routing is enabled on interface
  Multicast TTL threshold is 0
  Multicast designated router (DR) is 10.1.2.25 (this system)
  IGMP querying router is 10.1.2.1
  No multicast groups joined
Bowler#
```

例 6-6 中显示了 Porkpie 与 Bowler 到子网 10.1.1.0/24 的单播路由，在 Porkpie 与 Bowler 中到多播源 10.1.1.0/24 的路由有相同的管理距离与量度；因此，有最高 IP 地址的路由器成为

例 6-6　Porkpie 与 Bowler 的单播路由

```
Porkpie#show ip route 10.1.1.0
Routing entry for 10.1.1.0/24
  Known via "ospf 1", distance 110, metric 138, type intra area
  Redistributing via ospf 1
  Last update from 10.2.3.1 on Serial1.609, 01:01:30 ago
  Routing Descriptor Blocks:
  * 10.2.4.2, from 10.1.1.1, 01:01:30 ago, via Serial1.605
      Route metric is 138, traffic share count is 1
    10.2.3.1, from 10.1.1.1, 01:01:30 ago, via Serial1.609
      Route metric is 138, traffic share count is 1

Porkpie#
```

```
Bowler#show ip route 10.1.1.0
Routing entry for 10.1.1.0/24
  Known via "ospf 1", distance 110, metric 138, type intra area
  Redistributing via ospf 1
  Last update from 10.2.5.2 on Serial1.705, 01:02:22 ago
  Routing Descriptor Blocks:
  * 10.2.5.2, from 10.1.1.1, 01:02:22 ago, via Serial1.705
      Route metric is 138, traffic share count is 1

Bowler#
```

子网 10.1.2.0/24 中的 PIM 前转器。已知图 6-2 中的网际网络中只运行 OSPF，所以两路由的管理距离均为 110 就不奇怪了。你可以看到这两条路由均有 OSPF 开销 138。因此，子网 10.1.2.0/24 中(10.1.1.88,228.13.20.216)的 PIM 前转器是 IP 地址值最高的 Bowler。例 6-7 中证实这一点，在这个例中比较了 Porkpie 与 Bowler 的 mroute，可以看出 Bowler 是子网 10.1.2.0/24 中多播组(10.1.1.88, 228.13.20.216)的前转器。Porkpie 的(S,G)条目与例 6-3 的相比，注意到接口 E0 没有被剪除，Bowler 的 E0 接口处于前转模式，表明，它正在将多播组的流量前转到子网中。

例 6-7　Porkpie 与 Bowler 的 mroute

```
Porkpie#show ip mroute 228.13.20.216
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
       M - MSDP created entry, X - Proxy Join Timer Running
       A - Advertised via MSDP
Outgoing interface flags: H - Hardware switched
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 228.13.20.216), 23:51:13/00:02:59, RP 0.0.0.0, flags: DJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial1.609, Forward/Dense, 03:48:39/00:00:00
    Serial1.605, Forward/Dense, 03:48:39/00:00:00
    Ethernet0, Forward/Dense, 01:18:18/00:00:00

(10.1.1.88, 228.13.20.216), 00:03:06/00:02:53, flags: PCT
  Incoming interface: Serial1.605, RPF nbr 10.2.4.2
  Outgoing interface list:
    Serial1.609, Prune/Dense, 00:03:06/00:00:18
    Ethernet0, Prune/Dense, 00:03:06/00:02:53

Porkpie#
```

```
Bowler#show ip mroute 228.13.20.216
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
       M - MSDP created entry, X - Proxy Join Timer Running
       A - Advertised via MSDP
Outgoing interface flags: H - Hardware switched
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 228.13.20.216), 01:47:12/00:02:59, RP 0.0.0.0, flags: DJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0, Forward/Dense, 01:26:34/00:00:00
    Serial1.705, Forward/Dense, 01:47:12/00:00:00

(10.1.1.88, 228.13.20.216), 01:27:43/00:02:59, flags: CTA
  Incoming interface: Serial1.705, RPF nbr 10.2.5.2
  Outgoing interface list:
    Ethernet0, Forward/Dense, 01:26:34/00:00:00

Bowler#
```

有趣的是 Porkpie 在子网中查询组员，Bowler 为 228.13.20.216 前转多播流量。回忆第 5 章中的 IGMPv2 的规则，并没有与之冲突。Porkpie 的查询会导致组员向组地址发出 IGMP Membership Report，Bowler 听到 Membership Report 后，开始前转多播流量。如果组员要退出组，它发出 IGMP Leave 地址消息到“所有多播路由器”地址 224.0.0.2，如例 6-8 所示，这是 Bowler 收听到的。例 6-8 例中显示虽然 Porkpie(10.1.2.1)为 IGMP 的查询者，Bowler 从连接的组员处收到了 IGMP Leave 消息；由于 Bowler 是这个组的前转器，它把这个接口从多播组的输出接口列表中删除。

例 6-8　Bowler 为多播组的前转器

```
Bowler#debug ip igmp
IGMP debugging is on
Bowler#
IGMP: Received Leave from 10.1.2.113 (Ethernet0) for 228.13.20.216
IGMP: Received v2 Query from 10.1.2.1 (Ethernet0)
IGMP: Received v2 Query from 10.1.2.1 (Ethernet0)
IGMP: Deleting 228.13.20.216 on Ethernet0
Bowler#
```

回头看一下例 6-5，**show ip igmp interface** 显示了 Bowler 的接口 E0 采用默认的 60 秒 IGMP 查询时间间隔和 120 秒查询超时间隔。Porkpie 也用同样的默认值。在例 6-9 中有时间戳的查错消息显示了这些激活的计时器。前 3 个消息显示 Porkpie 每 60 秒发送一个 IGMP 的查询消息。不过发生了一些情况，查询停止了。第四与第五条消息显示在 120 秒钟后，Bowler 成为查询者，并马上发出它自己的查询消息。下面的消息继续以每 60 秒的间隔发送。最后两条消息显示 Porkpie 又激活了，重新发送自己的查询。由于 Porkpie 有更低的 IP 地址值，Bowler 认出 Porkpie 是查询者，就不再发送查询消息了。

例 6-9　用查错手段来显示 IGMP 查询者失效又恢复过程中发生的情况

```
Bowler#debug ip igmp
IGMP debugging is on
Bowler#
*Mar  5 23:41:36.318: IGMP: Received v2 Query from 10.1.2.1 (Ethernet0)
*Mar  5 23:42:36.370: IGMP: Received v2 Query from 10.1.2.1 (Ethernet0)
*Mar  5 23:43:36.422: IGMP: Received v2 Query from 10.1.2.1 (Ethernet0)
*Mar  5 23:45:36.566: IGMP: Previous querier timed out, v2 querier for Ethernet0 is
this system
*Mar  5 23:45:36.570: IGMP: Send v2 Query on Ethernet0 to 224.0.0.1
*Mar  5 23:46:05.602: IGMP: Send v2 Query on Ethernet0 to 224.0.0.1
*Mar  5 23:47:05.654: IGMP: Send v2 Query on Ethernet0 to 224.0.0.1
*Mar  5 23:48:05.706: IGMP: Send v2 Query on Ethernet0 to 224.0.0.1
*Mar  5 23:48:36.698: IGMP: Received v2 Query from 10.1.2.1 (Ethernet0)
*Mar  5 23:49:36.742: IGMP: Received v2 Query from 10.1.2.1 (Ethernet0)
Bowler#
```

记得在第 5 章中 PIM 以默认的 30 秒为间隔，向邻居发送 hello 消息，保持时间为 hello

时间间隔的 3.5 倍。如果在保持时间里没有收到邻居的 hello 消息，这个邻居就被宣布死亡。在后一个例子中，开始时 Bowler 与 Porkpie 同时在线，Bowler 为组 228.13.20.216 前转数据包到以太网中。例 6-10 中显示了当 Bowler 故障时发生的情况。

在预先设定的时间内没有从 Bowler 听到 PIM hello 消息，Porkpie 担当多播组 228.13.20.216 的前转器。

例 6-10　Porkpie 担当组内的前转器

```
Porkpie#debug ip pim 228.13.20.216
PIM debugging is on
Porkpie#
PIM: Neighbor 10.1.2.25 (Ethernet0) timed out
PIM: Changing DR for Ethernet0, from 10.1.2.25 to 10.1.2.1 (this system)
PIM: Building Graft message for 228.13.20.216, Serial1.609: no entries
PIM: Building Graft message for 228.13.20.216, Serial1.605: no entries
PIM: Building Graft message for 228.13.20.216, Ethernet0: no entries
Porkpie#
```

Porkpie 在保持时间里没有听到 Bowler 的 hello 消息，它知道自己必须承担起 PIM 前转器的责任，它当起 DR 的角色，向它的邻居发送 PIM Graft 消息。将例 6-11 中 Porkpie(10.1.1.88, 228.13.20.216)的条目和例 6-7 的上部相比较，Porkpie 现在将多播流量前转到以太网口上，而这个接口在成为前转器前被剪除了。注意例 6-7 中条目的剪除标帜，在例 6-11 的条目中已经不存在了。

例 6-11　在 Bowler 失效后，Porkpie 把组的流量前转到以太网中

```
Porkpie#show ip mroute 228.13.20.216
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
       M - MSDP created entry, X - Proxy Join Timer Running
       A - Advertised via MSDP
Outgoing interface flags: H - Hardware switched
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 228.13.20.216), 1d01h/00:02:59, RP 0.0.0.0, flags: DJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial1.609, Forward/Dense, 05:16:35/00:00:00
    Serial1.605, Forward/Dense, 05:16:35/00:00:00
    Ethernet0, Forward/Dense, 00:06:14/00:00:00

(10.1.1.88, 228.13.20.216), 00:23:10/00:02:59, flags: CT
  Incoming interface: Serial1.605, RPF nbr 10.2.4.2
  Outgoing interface list:
    Serial1.609, Prune/Dense, 00:23:10/00:01:44
    Ethernet0, Forward/Dense, 00:06:14/00:00:00

Porkpie#
```

6.3　配置与协议无关多播，稀疏模式(PIM-SM)

看过启动一个接口上的 PIM-DM 的配置，对于如何启动 PIM-SM 可能是显而易见的。它的完成也非常简单，只要使用 **ip pim sparse-mode** 命令。PIM-SM 的很多配置不是让用户很感兴趣，也不需要独立的例子。PIM-SM 独特的要求与其配置有趣的方面是辨明会聚点(RP)。你在第 5 章中学到 RP 可以静态配置，或用 Cisco Auto-RP 或开放的标准自举协议来动态实现。在下面案例研究中显示了这三种方式。

6.3.1　案例研究：静态配置 RP

图 6-3 中是你在这一章中已经见到的网际网络，但是路由器配置成了可运行 PIM-SM。Stetson 被选为 RP，在所有的路由器中都静态配置了这个信息。这个图示中显示了 Stetson 的 RP 地址是 10.224.1.1。这个地址可以在任何一个接口上，因为它是由单播协议对外宣告的，所以其他路由器知道如何到达这个地址。实际中，你应使用 loopback 接口的地址。一个简单的原因是这个 RP 地址可以更容易管理，不过更重要的原因是 RP 地址不应与任何可能出故障的物理接口相连。同样的原因 Loopback 接口被推荐给 IBGP 对端终点。

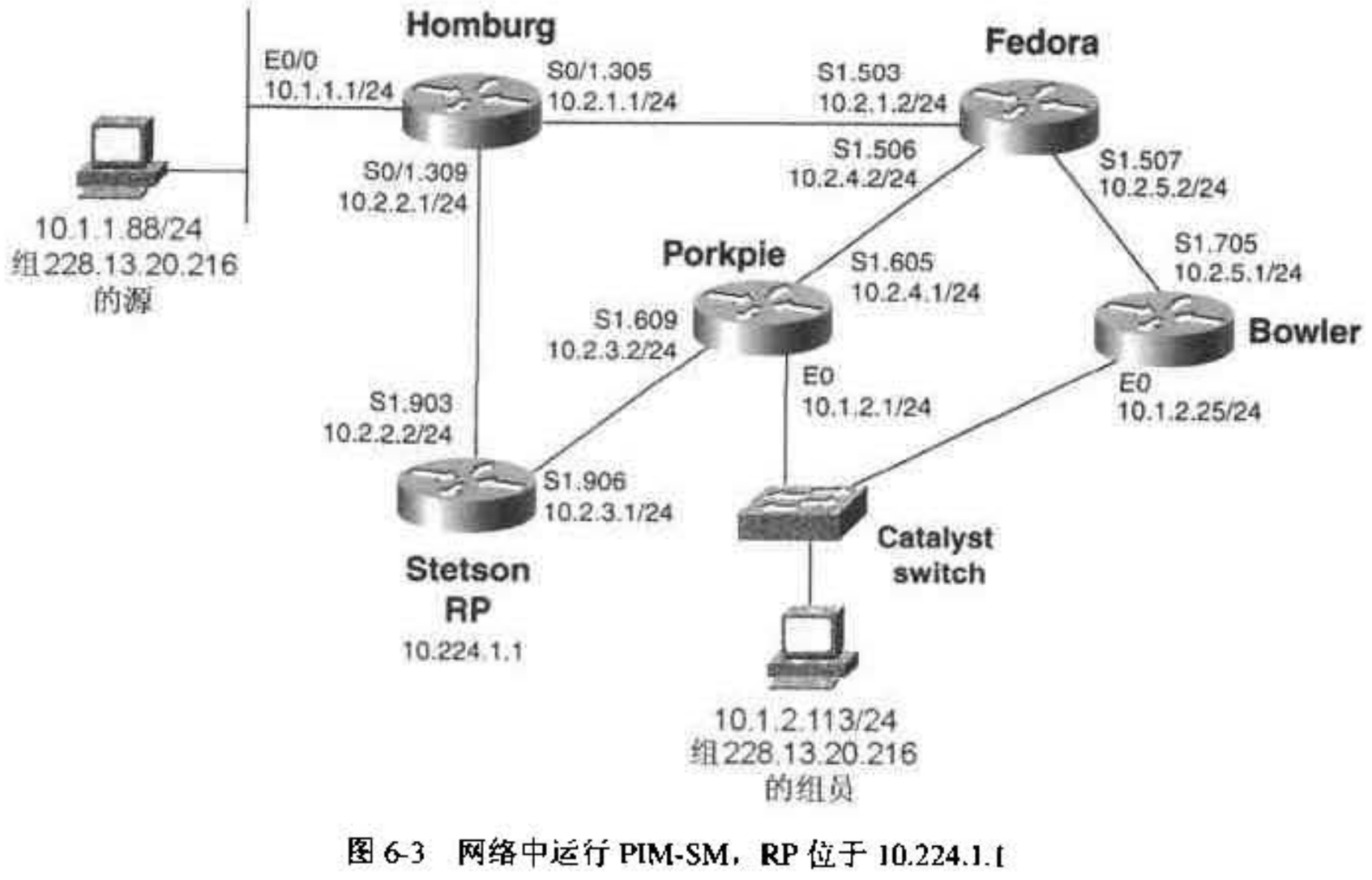


图 6-3　网络中运行 PIM-SM，RP 位于 10.224.1.1

例 6-12 显示了 Bowler 的配置，注意到这个接口原来是密集模式，现在为稀疏模式。

在例 6-12 中另有趣的一点是命令 **ip pim rp-address 10.224.1.1**,这个命令告诉路由器哪里发现 RP。当静态配置 RP 时，所有连接着组的源或成员的路由器必须有这条声明，因为它们必须知道 RP 在什么地方。注意到 Stetson 的 loopback 接口不一定像例 6-13 中说的那样要运行 PIM。这个接口除了提供 RP 地址外，不需要任何 PIM 功能。这个地址通过 OSPF 向整个网际网络宣告。不过 **ip pim rp-address 10.224.1.1** 语句在没有连接到源或组员时仍会在这个

配置中。在路由器中有这条语句的原因是路由器可以知道它自己是 RP。实际中，在所有路由器上静态配置 RP 地址是个很明智的做法。在不需要时，这也不会造成任何伤害，且会防止在需要这个语句时却被遗漏掉。

例 6-12　图 6-3 中 Bowler 的配置

```
hostname Bowler
!
ip multicast-routing
!
interface Ethernet0
 ip address 10.1.2.25 255.255.255.0
 ip pim sparse-mode
 ip cgmp
!
interface Serial1
 no ip address
 encapsulation frame-relay
!
interface Serial1.705 point-to-point
 description PVC to Fedora
 ip address 10.2.5.1 255.255.255.0
 ip pim sparse-mode
 no ip mroute-cache
 frame-relay interface-dlci 705
!
router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
!
ip pim rp-address 10.224.1.1
!
```

例 6-13　图 6-3 中的 RP，Stetson 的配置

```
hostname Stetson
!
ip multicast-routing
!
interface Loopback0
 ip address 10.224.1.1 255.255.255.255
!
interface Serial1
 no ip address
 encapsulation frame-relay
!
interface Serial1.903 point-to-point
 description PVC to R3
 ip address 10.2.2.2 255.255.255.0
 ip pim sparse-mode
 frame-relay interface-dlci 903
!
interface Serial1.906 point-to-point
 description PVC to 906
 ip address 10.2.3.1 255.255.255.0
 ip pim sparse-mode
 frame-relay interface-dlci 906
!
router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
!
ip pim rp-address 10.224.1.1
```

在 PIM-DM 部分，你比较 Porkpie 与 Bowler 中组 228.13.20.216 的 mroute 条目，这些条目中重要一点是路由器与组员共享一个以太网子网，所以 IGMP 查询和 PIM 前转会产生问题。例 6-14 中再一次比较了两个路由器的一个组的 mroute 条目，这些条目与例 6-7 的密集模式相比，有些含混。比如 Porkpie 的(*，G)条目中显示 E0 在输出列表中，并处于前转状态。它的(S,G)的输出接口列表却为空。但在 Bowler 中，E0 在(*,G)条目的输入接口列表中，而输出接口列表为空，E0 却在(S,G)条目输出接口列表中，并为前转状态。到底哪个路由器真正前转组的包呢？

例 6-14　比较 Porkpie 与 Bowler 中关于多播组 228.13.20.216 的 mroute 条目

```
Porkpie#show ip mroute 228.13.20.216
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
       M - MSDP created entry, X - Proxy Join Timer Running
       A - Advertised via MSDP
Outgoing interface flags: H - Hardware switched
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 228.13.20.216), 1d22h/00:02:59, RP 10.224.1.1, flags: SJC
  Incoming interface: Serial1.609, RPF nbr 10.2.3.1
  Outgoing interface list:
    Ethernet0, Forward/Sparse, 02:36:43/00:02:31

(10.1.1.88, 228.13.20.216), 03:08:42/00:02:02, flags: PCRT
  Incoming interface: Serial1.609, RPF nbr 10.2.3.1
  Outgoing interface list: Null

Porkpie#
```

```
Bowler#show ip mroute 228.13.20.216
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
       M - MSDP created entry, X - Proxy Join Timer Running
       A - Advertised via MSDP
Outgoing interface flags: H - Hardware switched
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 228.13.20.216), 1d00h/00:02:59, RP 10.224.1.1, flags: SJPC
  Incoming interface: Ethernet0, RPF nbr 10.1.2.1
  Outgoing interface list: Null

(10.1.1.88, 228.13.20.216), 02:38:20/00:02:59, flags: CT
  Incoming interface: Serial1.705, RPF nbr 10.2.5.2
  Outgoing interface list:
    Ethernet0, Forward/Sparse, 02:37:36/00:02:12

Bowler#
```

你仔细学习了第 5 章中的 PIM-SM 过程，你会知道哪一个路由器前转多播包。首先你知道 Bowler 是 DR，因为它的 IP 地址在 10.1.2.0/24 中有最高的值。你可以用 **show ip pim interface**

命令来验证 DR，如例 6-15 所示。

例 6-15　Bowler(10.1.2.25)为子网 10.1.2.0/24 中的 PIM DR

```
Porkpie#show ip pim interface

Address          Interface          Version/Mode    Nbr    Query      DR
                                                    Count  Intvl
10.1.2.1         Ethernet0          v2/Sparse        1     30      10.1.2.25
10.2.4.1         Serial1.605        v2/Sparse        1     30      0.0.0.0
10.2.3.2         Serial1.609        v2/Sparse        1     30      0.0.0.0
Porkpie#
```

当一个主机第一次请求加入一个组，DR 加入到共享 RP 树中(RPT)。检查例 6-16 中 Bowler 的单播路由表，从 Bolwer 到 RP 的路由通过子网 10.1.2.0/24 要经过 Porkpie。你知道为什么 Porkpie 的 E0 接口会在(*,G)的输出接口列表中。这个条目为连接 Bowler 到 Stetson 的 RPT。Bowler 的(*,G)条目中有一空的输出接口列表和一个已经设置的剪除标帜，因为它是 RPT 分枝的终点。

例 6-16　到 RP 的最短路径是通过连接的以太网到 Porkpie

```
Bowler#show ip route 10.224.1.1
Routing entry for 10.224.1.1/32
  Known via "ospf 1", distance 110, metric 75, type intra area
  Redistributing via ospf 1
  Last update from 10.1.2.1 on Ethernet0, 01:03:56 ago
  Routing Descriptor Blocks:
  * 10.1.2.1, from 10.224.1.1, 01:03:56 ago, via Ethernet0
      Route metric is 75, traffic share count is 1

Bowler#
```

下一步，你知道在默认情况下，收到第一个包，连接着组员的 PIM-SM 路由器会尝试切换到基于源的最短路径树(SPT)，不管这条路径是否经过 RP。Bowler 的单播路由表显示，到源网络 10.1.1.0/24 的最短路径是经过 Fedora，如例 6-17 所示。再看一下例 6-14 中的 mroute，Bowler 的(S,G)条目表示 Fedora 的 10.2.5.2 是上游或 RPF 邻居。接口 E0 在条目的输出列表中并处于前转状态，因为包当然要前转给组员。Porkpie 不为这个组前转数据，所以它的(S,G)条目的输出接口列表为空，并设置了剪除标帜。

例 6-17　Bowler 到达多播源子网 10.1.1.0/24 是通过接口 S1.705 路经 Fedora

```
Bowler#show ip route 10.1.1.0
Routing entry for 10.1.1.0/24
  Known via "ospf 1", distance 110, metric 138, type intra area
  Redistributing via ospf 1
  Last update from 10.2.5.2 on Serial1.705, 01:17:30 ago
  Routing Descriptor Blocks:
  * 10.2.5.2, from 10.1.1.1, 01:17:30 ago, via Serial1.705
      Route metric is 138, traffic share count is 1

Bowler#
```

你也可以用查错手段来看多播包是如何被前转的。例 6-18 显示了 Bowler 从 10.1.1.88 通过 S1.705 收到了 Fedora 发来 228.13.20.216 的多播包，例中使用了查错手段来捕捉 IP 多播包(mpackage)，你可以观察到 Bowler 在接口 S1.705 上收到(10.1.1.88, 228.13.20.216)的包，并从接口 E0 上将其转发出去。这个包从接口 E1 前转到所连接的组员。

例 6-18　用查错手段捕捉包

```
Bowler#debug ip mpacket 228.13.20.216
IP multicast packets debugging is on for group 228.13.20.216
Bowler#
IP: s=10.1.1.88 (Serial1.705) d=228.13.20.216 (Ethernet0) len 573, mforward
IP: s=10.1.1.88 (Serial1.705) d=228.13.20.216 (Ethernet0) len 573, mforward
IP: s=10.1.1.88 (Serial1.705) d=228.13.20.216 (Ethernet0) len 573, mforward
IP: s=10.1.1.88 (Serial1.705) d=228.13.20.216 (Ethernet0) len 573, mforward
IP: s=10.1.1.88 (Serial1.705) d=228.13.20.216 (Ethernet0) len 573, mforward
IP: s=10.1.1.88 (Serial1.705) d=228.13.20.216 (Ethernet0) len 573, mforward
IP: s=10.1.1.88 (Serial1.705) d=228.13.20.216 (Ethernet0) len 573, mforward
IP: s=10.1.1.88 (Serial1.705) d=228.13.20.216 (Ethernet0) len 573, mforward
IP: s=10.1.1.88 (Serial1.705) d=228.13.20.216 (Ethernet0) len 573, mforward
```

在 Porkpie 上使用同样查错的命令，可以显示出有趣的结果，如例 6-19 所示。查错的消息显示路由器没有从 RP 与 Fedora 处收到组 228.13.20.216 的包。但收到 Bowler 前转到以太网子网 10.1.2.0/24 中的包。Porkpie 的 *mroute* 条目如例 6-14 所示，显示了 RPF 对这个组的接口是 S1.609。因为这个包在 E0 上收到，RPF 检测失败，这个包给丢弃。

例 6-19　Porkpie 不会为多播组 228.13.20.216 前转任何包

```
Porkpie#debug ip mpacket 228.13.20.216
IP multicast packets debugging is on for group 228.13.20.216
Porkpie#
IP: s=10.1.1.88 (Ethernet0) d=228.13.20.216 len 583, not RPF interface
IP: s=10.1.1.88 (Ethernet0) d=228.13.20.216 len 583, not RPF interface
IP: s=10.1.1.88 (Ethernet0) d=228.13.20.216 len 583, not RPF interface
IP: s=10.1.1.88 (Ethernet0) d=228.13.20.216 len 583, not RPF interface
IP: s=10.1.1.88 (Ethernet0) d=228.13.20.216 len 583, not RPF interface
IP: s=10.1.1.88 (Ethernet0) d=228.13.20.216 len 583, not RPF interface
IP: s=10.1.1.88 (Ethernet0) d=228.13.20.216 len 583, not RPF interface
IP: s=10.1.1.88 (Ethernet0) d=228.13.20.216 len 583, not RPF interface
IP: s=10.1.1.88 (Ethernet0) d=228.13.20.216 len 583, not RPF interface
```

迄今为止，显示如此多的例子，都依赖于 Cisco 路由器收到第一个多播包后切换到组的 SPT 上。你在第 5 章中学到可以用学 **ip pim spt-threshold** 命令来改变这个默认值，一个门限值可能以 k 比特每秒定义，路由器在组流量到达速率超过这个门限后切换到 SPT。另外，你可以使用关键词 **infinity**，路由器将不会切换到 SPT。图 6-3 可使人明白 Bowler 的配置中加入了 **ip pim spt-threshold** 命令后会发生的情况。例 6-20 显示了在 Bowler 重新配置后，Porkpie 和 Bowler 的 mroute 条目。Bowler 的 RPT 经它的 E0，连接子网 10.1.2.0/24 并可通过 Porkpie。所以 Porkpie 必须从 RP 前转包。但是 Bowler 的 E0 接口也是它对这个组的 RPF 接口。PIM 路由器不能从 RPF 接口前转多播包。这是多播版的水平分割法的一个规则：包不能从接收到的接口上前转。所以，Bowler 的(*,G)加一个剪除的标帜。Porkpie 现在向组员前转包。有趣

的是，虽然 Porkpie 为组员前转包，因为 Bowler 必须使用 RPT，而 Porkpie 自己却没有这样的限制，它切换到了 SPT，通过 Fedora 而不是 RP 接收多播包。

例 6-20　Bowler 配置为不会切换到 SPT 后，多播组 228.13.20.216 前转的任务交给 Porkpie

```
Porkpie#show ip mroute 228.13.20.216
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
       M - MSDP created entry, X - Proxy Join Timer Running
       A - Advertised via MSDP
Outgoing interface flags: H - Hardware switched
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 228.13.20.216), 00:45:09/00:02:59, RP 10.224.1.1, flags: SJC
  Incoming interface: Serial1.609, RPF nbr 10.2.3.1
  Outgoing interface list:
    Ethernet0, Forward/Sparse, 00:44:11/00:02:54

(10.1.1.88, 228.13.20.216), 00:44:30/00:02:59, flags: CT
  Incoming interface: Serial1.605, RPF nbr 10.2.4.2
  Outgoing interface list:
    Ethernet0, Forward/Sparse, 00:44:11/00:02:24

Porkpie#
```

```
Bowler#show ip mroute 228.13.20.216
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
       M - MSDP created entry, X - Proxy Join Timer Running
       A - Advertised via MSDP
Outgoing interface flags: H - Hardware switched
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 228.13.20.216), 00:45:31/00:02:07, RP 10.224.1.1, flags: SPC
  Incoming interface: Ethernet0, RPF nbr 10.1.2.1
  Outgoing interface list: Null

Bowler#
```

有时，你可能需要为不同的组分配不同的 RP。特别在多播域里组的数目增加时，会这样做，你需要将 RP 的任务分摊到其他的路由器上，减轻内存和 CPU 的占用。图 6-4 与这一章中你一直在研究的网际网络一样，不过现在 Fedora 指定为 RP，其地址为 10.244.1.2。图中 Stetson 与 Fedora 均为 RP；访问表与静态 RP 地址一起使用来告诉每一台域中的路由器，对于某一组使用哪一个 RP。采用访问表，你可以配置多个 RP，规定哪一个组用哪一个 RP。

比如，考虑例 6-21。

访问表 5 定义了允许使用 RP 10.224.1.2(Fedora)的组，访问表 10 定义了允许使用 10.224.1.1(Stetson)的组。任何一个组如果它的地址不匹配这两个访问表之一，不会分配给 RP，因此不能加入到共享树中。这个配置也加到 Bowler 中，例 6-22 显示了这个结果。可

以看到列出的组(为路由器上激活的组)已经根据访问表 5 和 10 的限制映射到自己的 RP 上。

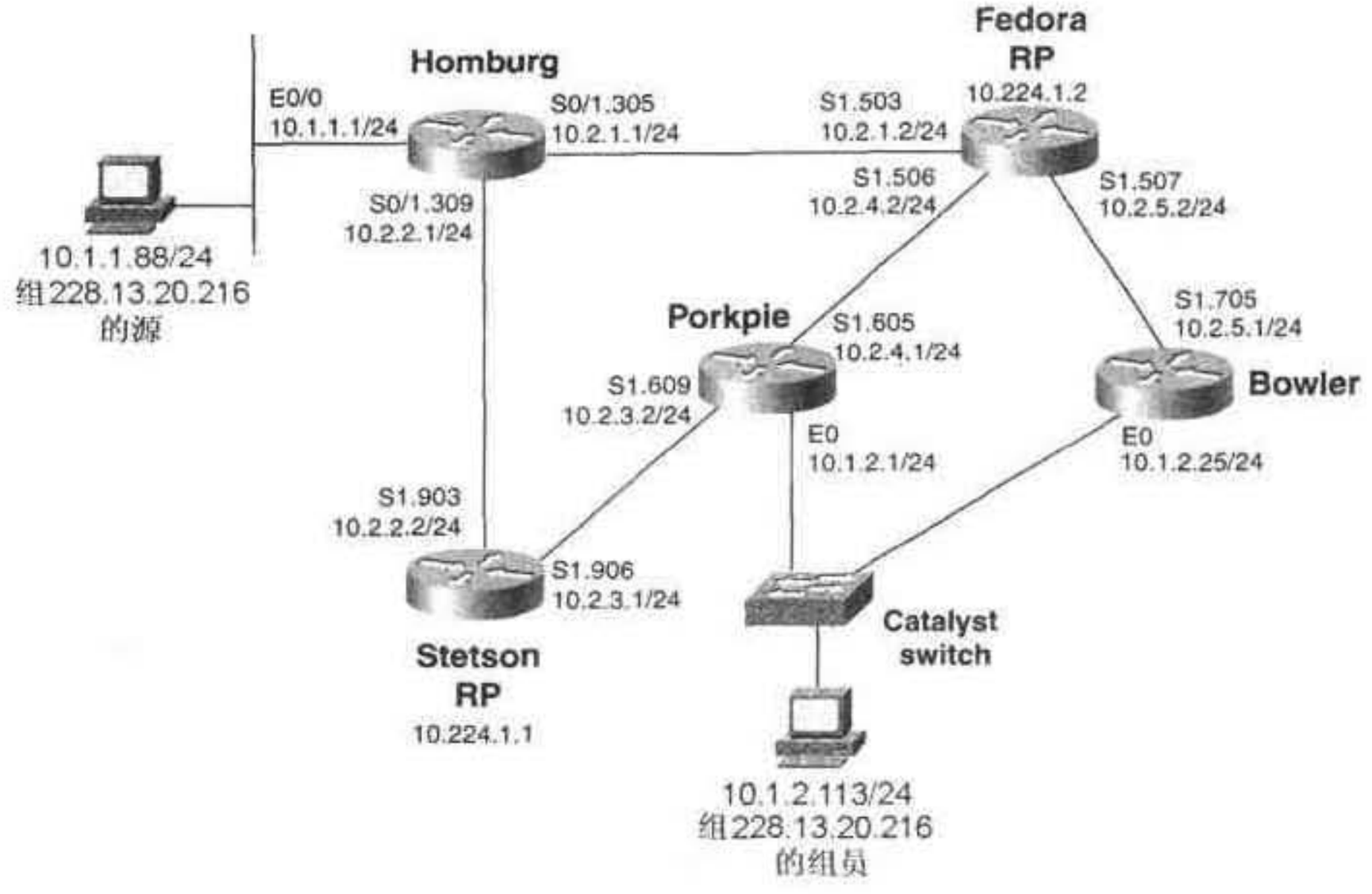


图 6-4　RP 任务的分摊

例 6-21　Bowler 的 RP 过滤配置

```
ip pim rp-address 10.224.1.1 10
ip pim rp-address 10.224.1.2 5
!
access-list 5 permit 239.0.0.0 0.255.255.255
access-list 5 permit 228.13.20.0 0.0.0.255
access-list 10 permit 224.2.127.254
access-list 10 permit 230.253.0.0 0.0.255.255
```

例 6-22　用 show ip pim rp 命令来显示路由器与其映射的 RP 上组的活动

```
Bowler#show ip pim rp
Group: 239.255.255.254, RP: 10.224.1.2, v2, uptime 01:20:13, expires 00:02:08
Group: 228.13.20.216, RP: 10.224.1.2, v2, uptime 01:19:30, expires never
Group: 224.2.127.254, RP: 10.224.1.1, v2, uptime 01:20:05, expires never
Group: 230.253.84.168, RP: 10.224.1.1, v2, uptime 01:20:06, expires 00:01:48
Bowler#
```

6.3.2　案例研究：配置 Auto-RP

在稳定的 PIM 域中，RP 的静态配置很直观。新的路由器加入后，给它配置上 RP 的位置。静态 RP 配置在两种环境中会产生问题。

- 因现有的 RP 的地址改变或安装了新的 RP，RP 的地址必须改变。网络管理员必须改变所有 PIM 路由器上的静态配置。这在一个较大的域中会引起相当的停机时间。
- RP 产生故障。静态配置的 PIM 域不会简单选择备用的 RP。

因此，在除了小型 PIM 域外，为了便于管理和较好的冗余备份，推荐使用一种或两种 Auto-RP 发现机制：Auto-RP 或自举协议。这一案例研究中主要针对 Auto-RP，下一案例研究是自举协议。

如在第 5 章中所讨论的一样，Auto-RP 是在自举协议做为 PIMv2 的一部分之前开发的，由 Cisco 拥有产权的协议。Auto-RP 必须和 Cisco IOS Software Release 11.3 一起使用，在这一版本中开始支持 PIMv2。

配置基本的 Auto-RP 有两个步骤：

1．所有的候选 RP 都必须配置。

2．所有的映射代理必须配置。

候选 RP(C-RP)用 **ip pim send-rp-annouce** 命令来配置，当你输入这条命令后，再定义一个接口，路由器采用这个接口的 IP 地址做为 RP 地址，然后定义一个 TTL 值，这个值加入到宣告消息中。TTL 提供了包不出多播域边界的范围。当路由器配置成候选 RP 时，它开始向保留地址 224.0.1.39 每 60 秒发送一个 RP-Announce 消息。

映射代理收听 C-RP 的 RP-Annouce 消息，并选择出 RP。它在 RP-Discovery 消息中向 PIM 域中的其他设备告知 RP 地址，这个消息每隔 60 秒发向地址 224.0.1.40 一次。

图 6-5 显示了一个简单的拓扑。这里路由器 Stetson 和 Fedora 为候选 RP，地址分别为 10.224.1.1 与 10.224.1.2。Porkpie 为映射代理，地址为 10.224.1.3。

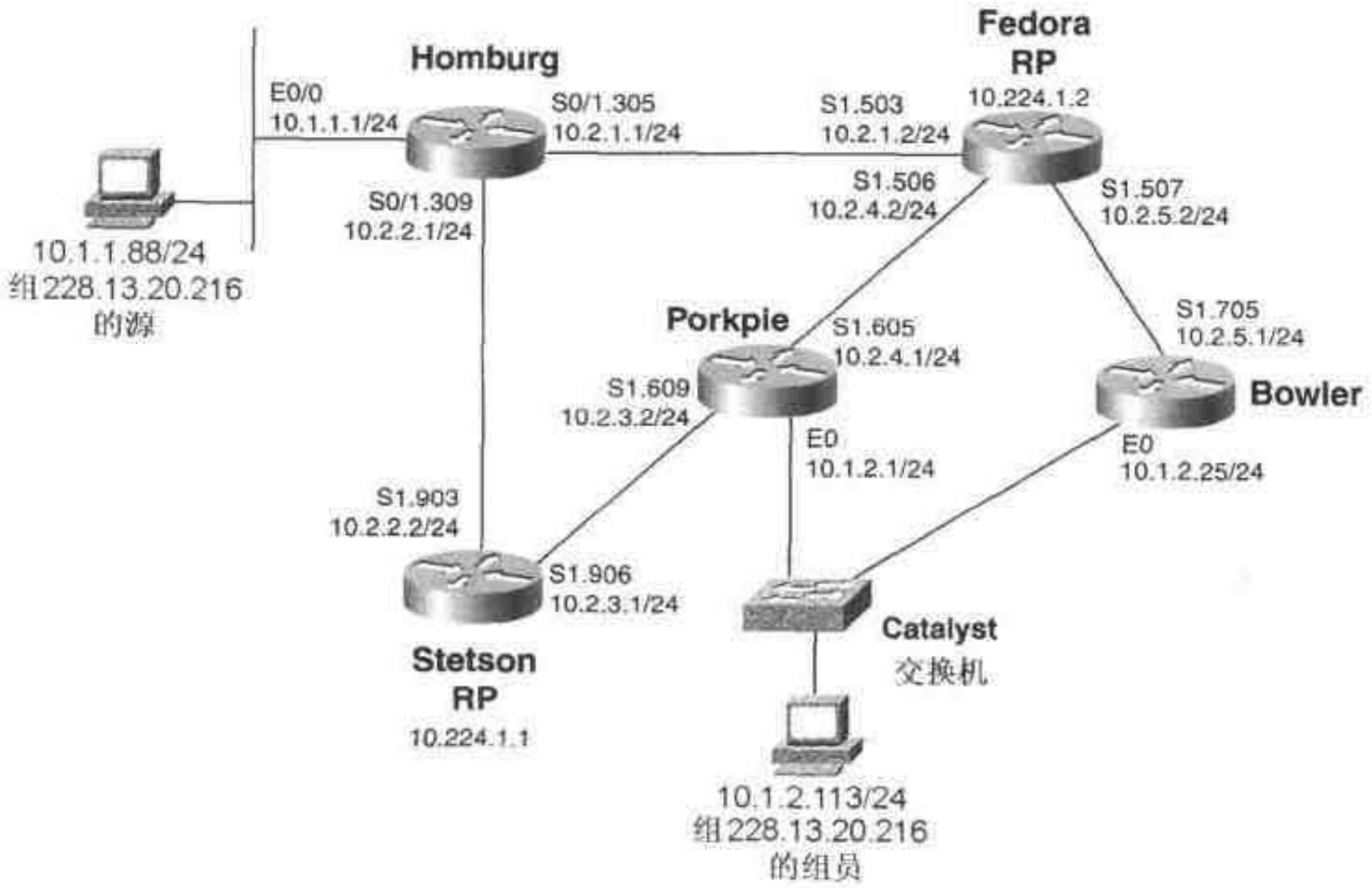


图 6-5　Stetson 与 Fedora 为候选 RP，Porkpie 为映射代理

例 6-23 显示了 Fedora 相关部分配置。

例 6-23　Fedora 配置成候选 RP

```
interface Loopback0
 ip address 10.224.1.2 255.255.255.255
!
ip pim send-rp-announce Loopback0 scope 5
```

Stetson 的配置是相似的。RP 的地址就是接口 L0 的地址，关键词 scope 设置了 RP-Announce 消息的 TTL 值。

例 6-24 显示了将 Porkpie 设为映射代理的配置。

例 6-24　把 Porkpie 设为映射代理

```
interface Loopback0
 ip address 10.224.1.3 255.255.255.255
 ip pim sparse-mode
!
ip pim send-rp-discovery Loopback0 scope 5
```

L0 的地址也做为映射代理的地址，TTL 设为 5。在例 6-24 的配置里，注意到 PIM-SM 必须在 loopback 接口上配置。这在映射代理中是必须设置的；如果没有在这个接口上设置 PIM-SM，你会看到如例 6-25 中所示的错误信息。

例 6-25　不在映射代理的 loopback 接口上启动 PIM-SM，会导致错误消息的产生

```
Porkpie(config)#ip pim send-rp-discovery Loopback0 scope 5
Non PIM interface ignored in accepted command.
Porkpie(config)#
```

这一配置的命令与下述命令很相似：

ip pim send-rp-discovery scope 5

这个命令中定义的接口是不可接受的，映射代理不能工作。与映射代理不同， C-RP 的 loopback 接口不一定要配置 PIM。当然，映射代理与 C-RP 上，所有连接着 PIM 邻居的物理接口都要配置 PIM-SM。

当一台 Cisco 路由器首次配置成 PIM-SM 后，它开始监听地址 224.0.1.40。如果 C-RP 或映射代理发生了变化，这些变化由相应的设备自动宣告，使整个域内的路由器都能记住这个变化。也许最重要的一点是你可以为任何或所有的组配置多个 RP。映射代理根据最高的 RP 地址值为一个组选择 RP。如果 RP 发生故障，映射代理选择有次高地址值的 RP，并向网内宣告。

例 6-26 中显示了一个 RP 切换的例子。在此，**debug ip pim auto-rp** 用于显示所有 Auto-RP 的行为。你可以看到图 6-5 中映射代理 Porkpie 收到 Stetson(10.224.1.1)和 Fedora(10.224.1.2)的 RP-Announce 消息。因为 Fedora 有最高的 IP 地址值，所以它被选为 RP 并向域内所有组(224.0.0.0/4)宣告。收到第一个 RP-Announce 消息后 Fedora 失效。当 Porkpie 在 180 秒内没

收到 Fedora 的 RP-Announce 消息(这个消息时间间隔的 3 倍)时，它宣布这个 RP 死亡，于是选择 Stetson 做为新的 RP，并开始宣告新的 RP。这一系列事件在调测信息中用亮光标了出来。

在例 6-27 中，Bowler 使用了 **show ip pim rp** 命令来显示路由器要接收的组和这个组所映射到的 RP，例中在 Fedora 失效前，所有的 Bowler 组都映射到 RP(10.224.1.2)；失效后 Bowler 的组根据映射代理的信息，重新映射到 Stetson。前面一个显示在 Fedora 故障前的多播组映射的 RP 地址，第二个显示了 Fedora 失效后映射代理宣告新的 RP，这时所有的多播组映射到 Stetson。

例 6-26　用查错来观察图 6-5 中的映射代理的错差倒换

```
Porkpie#debug ip pim auto-rp
PIM Auto-RP debugging is on
Porkpie#
Auto-RP: Received RP-announce, from 10.224.1.1, RP_cnt 1, ht 181
Auto-RP: Update (224.0.0.0/4, RP:10.224.1.1), PIMv2 v1
Auto-RP: Received RP-announce, from 10.224.1.1, RP_cnt 1, ht 181
Auto-RP: Update (224.0.0.0/4, RP:10.224.1.1), PIMv2 v1
Auto-RP: Received RP-announce, from 10.224.1.2, RP_cnt 1, ht 181
Auto-RP: Update (224.0.0.0/4, RP:10.224.1.2), PIMv2 v1
Auto-RP: Received RP-announce, from 10.224.1.2, RP_cnt 1, ht 181
Auto-RP: Update (224.0.0.0/4, RP:10.224.1.2), PIMv2 v1
Auto-RP: Build RP-Discovery packet
Auto-RP:  Build mapping (224.0.0.0/4, RP:10.224.1.2), PIMv2 v1,
Auto-RP: Send RP-discovery packet on Loopback0 (1 RP entries)
Auto-RP: Send RP-discovery packet on Serial1.605 (1 RP entries)
Auto-RP: Send RP-discovery packet on Serial1.609 (1 RP entries)
Auto-RP: Send RP-discovery packet on Ethernet0 (1 RP entries)
Auto-RP: Received RP-announce, from 10.224.1.1, RP_cnt 1, ht 181
Auto-RP: Update (224.0.0.0/4, RP:10.224.1.1), PIMv2 v1
Auto-RP: Received RP-announce, from 10.224.1.1, RP_cnt 1, ht 181
Auto-RP: Update (224.0.0.0/4, RP:10.224.1.1), PIMv2 v1
Auto-RP: Build RP-Discovery packet
Auto-RP:  Build mapping (224.0.0.0/4, RP:10.224.1.2), PIMv2 v1,
Auto-RP: Send RP-discovery packet on Loopback0 (1 RP entries)
Auto-RP: Send RP-discovery packet on Serial1.609 (1 RP entries)
Auto-RP: Send RP-discovery packet on Ethernet0 (1 RP entries)
Auto-RP: Received RP-announce, from 10.224.1.1, RP_cnt 1, ht 181
Auto-RP: Update (224.0.0.0/4, RP:10.224.1.1), PIMv2 v1
Auto-RP: Received RP-announce, from 10.224.1.1, RP_cnt 1, ht 181
Auto-RP: Update (224.0.0.0/4, RP:10.224.1.1), PIMv2 v1
Auto-RP: Build RP-Discovery packet
Auto-RP:  Build mapping (224.0.0.0/4, RP:10.224.1.2), PIMv2 v1,
Auto-RP: Send RP-discovery packet on Loopback0 (1 RP entries)
Auto-RP: Send RP-discovery packet on Serial1.609 (1 RP entries)
Auto-RP: Send RP-discovery packet on Ethernet0 (1 RP entries)
Auto-RP: Received RP-announce, from 10.224.1.1, RP_cnt 1, ht 181
Auto-RP: Update (224.0.0.0/4, RP:10.224.1.1), PIMv2 v1
Auto-RP: Received RP-announce, from 10.224.1.1, RP_cnt 1, ht 181
Auto-RP: Update (224.0.0.0/4, RP:10.224.1.1), PIMv2 v1
Auto-RP: Mapping (224.0.0.0/4, RP:10.224.1.2) expired,
Auto-RP: Build RP-Discovery packet
Auto-RP:  Build mapping (224.0.0.0/4, RP:10.224.1.1), PIMv2 v1,
Auto-RP: Send RP-discovery packet on Loopback0 (1 RP entries)
Auto-RP: Send RP-discovery packet on Serial1.609 (1 RP entries)
Auto-RP: Send RP-discovery packet on Ethernet0 (1 RP entries)
Porkpie#
```

例 6-27　RP 的重新映射

```
Bowler#show ip pim rp
Group: 239.255.255.254, RP: 10.224.1.2, v2, v1, uptime 00:08:07, expires 00:04:26
Group: 228.13.20.216, RP: 10.224.1.2, v2, v1, uptime 00:08:08, expires 00:04:26
Group: 224.2.127.254, RP: 10.224.1.2, v2, v1, uptime 00:08:07, expires 00:04:26
Group: 230.253.84.168, RP: 10.224.1.2, v2, v1, uptime 00:08:07, expires 00:04:26
Bowler#

Bowler#show ip pim rp
Group: 239.255.255.254, RP: 10.224.1.1, v2, v1, uptime 00:03:46, expires 00:02:56
Group: 228.13.20.216, RP: 10.224.1.1, v2, v1, uptime 00:03:46, expires 00:02:56
Group: 224.2.127.254, RP: 10.224.1.1, v2, v1, uptime 00:03:46, expires 00:02:56
Group: 230.253.84.168, RP: 10.224.1.1, v2, v1, uptime 00:03:46, expires 00:02:56
Bowler#
```

要改变C-RP发送RP-Announce默认的60秒钟的时间间隔，需在**ip pim send-rp-announce**命令中加入关键词 **interval**，比如下面一条命令使 Fedora 每 10 秒发送一个 RP一Announce 命令。

ip pim send-rp-announce Loopback0 scope 5 interval 10

映射代理等待 C-RP 发出的 RP-Announce 消息的保持时间，总是为 RP-Announce 消息时间间隔的 3 倍。所以前面一个命令将 Fedora 切换的时间缩短为 30 秒，同路由器产生 RP-Announce 消息一样，开销增加到 6 倍。

C-RP 在它的 RP-Announce 中宣告，它可以做为哪些组的 RP。这个消息默认发送到 224.0.0.0/4，这个地址代表所有多播组。如前面静态 RP 中学到的一样，你会在有些时候希望把不同的多播组映射到不同的 RP 上。比如，假设所有 224.0.0.0 到 231.255.255.255(224.0.0.0/5)的组映射到 Stetson，所有 232.0.0.0 到 239.255.255.255(232.0.0.0/5)的组映射到 Fedora。这两台路由器的 C-RP 配置如例 6-28 所示。

例 6-28　Stetson 与 Fedora 配置成 C-RP

Stetson

```
ip pim send-rp-announce Loopback0 scope 5 group-list 20
!
access-list 20 permit 224.0.0.0 7.255.255.255
```

Fedora

```
ip pim send-rp-announce Loopback0 scope 5 group-list 30
!
access-list 30 permit 232.0.0.0 7.255.255.255
```

关键词 **group-list** 把 **ip pim send-rp-announce** 语句与一个访问表联系起来。这个访问表描述了对于哪些组中的路由器可以成为 RP。例 6-29 中显示了映射代理 Porkpie 宣告了符合 RP-Announce 消息限制的 RP，路由器 Bowler 中结果。239.255.255.254 被映射到 Fedora，而其他 3 个组虽然也在 224.0.0.0/5 的范围内，但映射到 Stetson。

例 6-29　Bowler 的组到 RP 的映射，显示了 Stetson 与 Fedora 配置中的限制

```
Bowler#show ip pim rp
Group: 239.255.255.254, RP: 10.224.1.2, v2, v1, uptime 00:04:25, expires 00:02:56
Group: 228.13.20.216, RP: 10.224.1.1, v2, v1, uptime 00:11:05, expires 00:03:57
Group: 224.2.127.254, RP: 10.224.1.1, v2, v1, uptime 00:11:05, expires 00:03:57
Group: 230.253.84.168, RP: 10.224.1.1, v2, v1, uptime 00:11:05, expires 00:03:57
Bowler#
```

假设你也希望组 228.13.0.0 到 228.13.255.255 映射到 Fedora，对于路由器 Fedora 的配置就如例 6-30 显示的一样：

例 6-30　将 Fedora 配置成多播组 228.13.0.0 到 228.13.255.255 的 C-RP

```
ip pim send-rp-announce Loopback0 scope 5 group-list 30
!
access-list 30 permit 232.0.0.0 7.255.255.255
access-list 30 permit 228.13.0.0 0.0.255.255
```

例 6-31 显示了 Bowler 的结果，多播组 228.13.20.216，在例 6-29 中映射到 RP 10.224.1.1，这里映射到 RP 10.224.1.2。注意 Stetson 的配置并没有改变，C-RP 宣布 224.0.0.0/5 是它允许的组的范围，包括了 228.13.0.0/16。这个映射代理对于组 228.13.0.0/16 有两个 C-RP，因为 Fedora 有 IP 地址值较高，所以选择 Fedora 做为 C-RP。

例 6-31　多播组映射到相应的 RP

```
Bowler#show ip pim rp
Group: 239.255.255.254, RP: 10.224.1.2, v2, v1, uptime 00:01:43, expires 00:04:16
Group: 228.13.20.216, RP: 10.224.1.2, v2, v1, uptime 00:01:43, expires 00:04:16
Group: 224.2.127.254, RP: 10.224.1.1, v2, v1, uptime 00:36:05, expires 00:02:47
Group: 230.253.84.168, RP: 10.224.1.1, v2, v1, uptime 00:36:05, expires 00:02:47
Bowler#
```

show ip pim rp 命令中的几个变量可以使你观察组到 RP 的映射关系。这个命令在前面使用的基本型式里只显示了路由器上激活的组和匹配的多播组地址的 RP。要观察所有 RP 可能匹配的组，可以用命令 **show ip pim rp mapping**，如例 6-32 所示，例中 Bowler 收到映射代理 10.224.1.3 的 RP-Discover 消息，将三个范围的多播组地址映射到两个不同的 RP。

例 6-32　Bowler 将不同的多播组映射到不同的 RP 上

```
Bowler#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/5
  RP 10.224.1.1 (?), v2v1
    Info source: 10.224.1.3 (?), via Auto-RP
         Uptime: 01:14:37, expires: 00:02:42
Group(s) 228.13.0.0/16
  RP 10.224.1.2 (?), v2v1
    Info source: 10.224.1.3 (?), via Auto-RP
         Uptime: 00:43:15, expires: 00:02:37
Group(s) 232.0.0.0/5
  RP 10.224.1.2 (?), v2v1
    Info source: 10.224.1.3 (?), via Auto-RP
         Uptime: 00:43:15, expires: 00:02:41
Bowler#
```

一个相似的命令是 **show ip pim rp mapping in-use**，如例 6-33 所示。除了例 6-32 中所显示的信息外，当前路由器上使用的组的范围也显示出来。注意到例 6-32 与例 6-33 中的输出显示了映射代理的源 10.224.1.3。这个信息在有多个映射代理时很有用。

例 6-33　关键词 in-use 显示了路由器上当前使用的组地址范围

```
Bowler#show ip pim rp mapping in-use
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/5
  RP 10.224.1.1 (?), v2v1
    Info source: 10.224.1.3 (?), via Auto-RP
         Uptime: 01:21:24, expires: 00:02:50
Group(s) 228.13.0.0/16
  RP 10.224.1.2 (?), v2v1
    Info source: 10.224.1.3 (?), via Auto-RP
         Uptime: 00:50:02, expires: 00:02:49
Group(s) 232.0.0.0/5
  RP 10.224.1.2 (?), v2v1
    Info source: 10.224.1.3 (?), via Auto-RP
         Uptime: 00:50:02, expires: 00:02:48

RPs in Auto-RP cache that are in use:
Group(s): 224.0.0.0/5,   RP: 10.224.1.1
Group(s): 232.0.0.0/5,   RP: 10.224.1.2
Group(s): 228.13.0.0/16,   RP: 10.224.1.2
Bowler#
```

有些时候，某一特殊的组地址还没有在路由器上激活前，你会希望知道它映射到哪一个 RP。例如，假设你希望知道在 Bowler 上的组 235.1.2.3 会映射到哪一个 RP 上你可以用 **show ip pim rp-hash** 命令，结果如例 6-34 所示。这个结果表明组 235.1.2.3 会映射到 RP 10.224.1.2。这个结果与访问表中的限制相一致。

例 6-34　show ip pim rp-hash 命令可以使你判断某一多播组映射到的哪一个 RP

```
Bowler#show ip pim rp-hash 235.1.2.3
  RP 10.224.1.2 (?), v2v1
    Info source: 10.224.1.3 (?), via Auto-RP
         Uptime: 00:55:48, expires: 00:02:00
Bowler#
```

你可以设置 RP-Announce 消息过滤器，防止映射代理接收无意间配置成 C-RP 的路由器发出的消息。例 6-35 中显示了 Porkpie 上的简单的配置。

例 6-35　Porkpie 配置成一个 RP 公告过滤器

```
ip pim rp-announce-filter rp-list 1 group-list 11
ip pim send-rp-discovery Loopback0 scope 5
!
access-list 1 permit 10.224.1.2
access-list 1 permit 10.224.1.1
access-list 11 permit 224.0.0.0 15.255.255.255
```

在例 6-35 的配置中建立了 RP 公告过滤器，只接收访问表 1 中的 C-RP，还有由这些 C-RP 进行宣告，且在访问表 11 中定义的组。

整个案例研究中，为了清晰，在图 6-5 中的 Stetson 与 Fedora 为 C-RP，Porkpie 为映射代理，实际中，配置了冗余的 C-RP 而只配置一台映射代理是不合理的。如果映射代理失效，RP 不会向整个域中宣告，PIM-SM 也会失效。一个更实际的做法是将 Stetson 与 Fedora 同时设置成 C-RP 与映射代理。如果一台路由器有故障，另一台可以向域内宣告 RP。

6.3.3　案例研究：配置稀疏—密集模式

在前一个案例中使用了一点点“欺骗”。看一下图 6-5，注意到 C-RP 直连在映射代理上，映射代理直连在 Bowler 上。在图 6-6 中，Homburg 配置成 Auto-RP 映射代理。这个拓扑给出一个有趣的矛盾：Homburg 在 RP-Discovery 消息中向所有的路由器宣告 RP，使用了保留地址 224.0.1.40。所有的 PIM-SM 路由器都在监听这个地址。在稀疏模式中，多播包必须选从一个共享树上进行前转。这意味着收听地址 224.0.1.40 的路由器必须通知它们的 RP，它们要加入到这个组中，来接收 RP-Discovery 消息。不过它们没有收到 RP-Discovery 消息之前怎么会知道 RP 的位置呢？

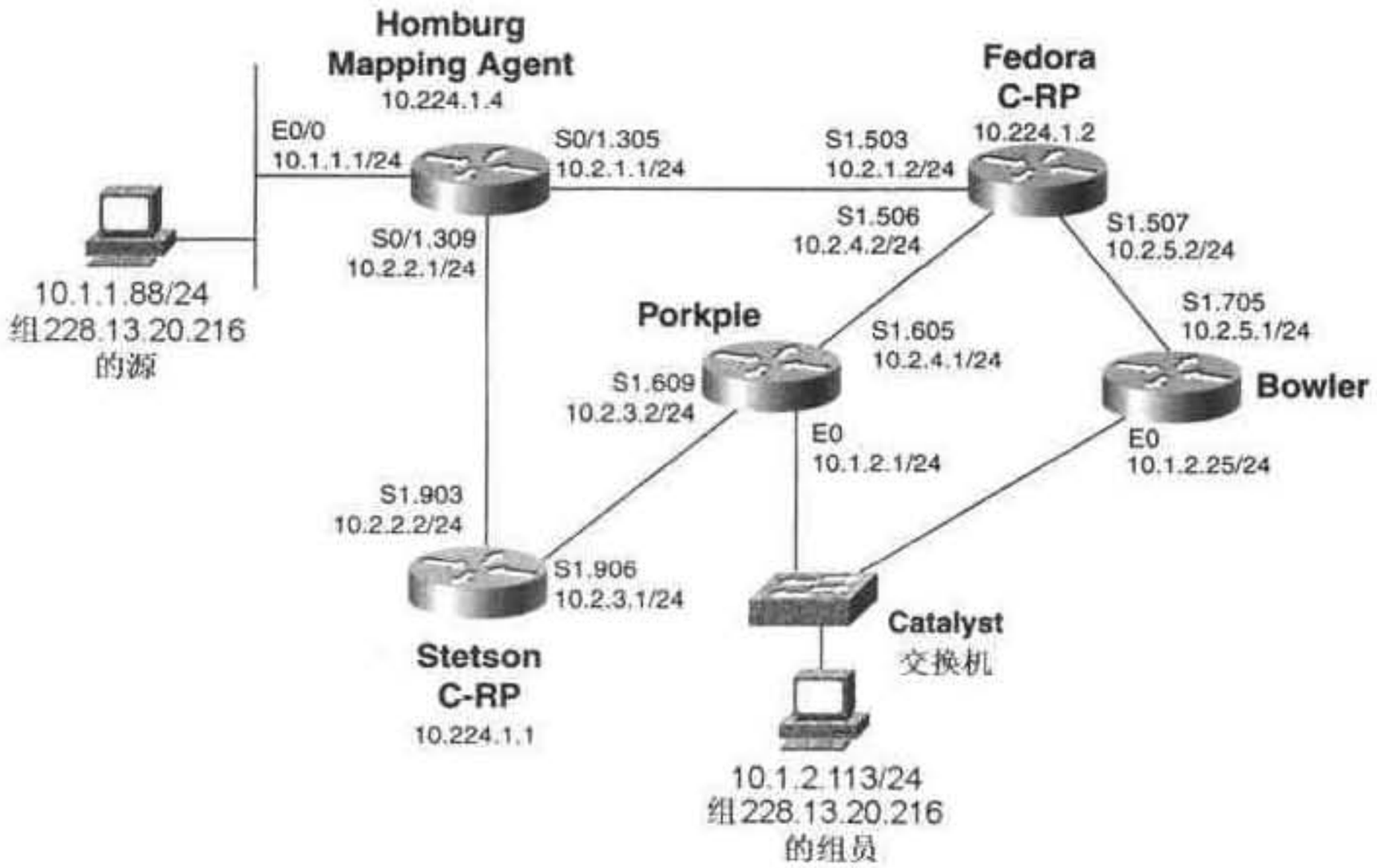


图 6-6　Homburg 是映射代理

这个“第 22 条军规”在 C-RP 没有直连在映射代理时，对 C-RP 也会起作用。映射代理为了选出 RP，必须收到 C-RP 的 RP-Announce 消息，为了完成这一目的，C-RP 必须加入组 224.0.1.39。不过，如果它不知道 RP 的位置，它就不能加入到这个组中，除非收到了 RP-Announce 消息。

PIM 稀疏—密集模式的创立解决了这个问题。当一个接口配置成这个模式时，它在知道组 RP 的位置时使用稀疏模式，如果不知道 RP 的位置，它使用密集模式。在 224.0.1.39 与

224.0.1.40 的情况中，多播组被认为是密集模式。例 6-36 显示了 Homburg 上稀疏—密集模式的配置。

例 6-36　路由器 Homburg 上的 PIM 密集稀疏模式

```
hostname Homburg
!
ip multicast-routing
!
interface Loopback0
 ip address 10.224.1.4 255.255.255.0
 ip pim sparse-mode
!
interface Ethernet0/0
 ip address 10.1.1.1 255.255.255.0
 ip pim sparse-dense-mode
 no ip mroute-cache
!
interface Serial0/1
 no ip address
 encapsulation frame-relay
 no ip mroute-cache
!
interface Serial0/1.305 point-to-point
 description PVC to R5
 ip address 10.2.1.1 255.255.255.0
 ip pim sparse-dense-mode
 no ip mroute-cache
 frame-relay interface-dlci 305
!
interface Serial0/1.309 point-to-point
 description PVC to R9
 ip address 10.2.2.1 255.255.255.0
 ip pim sparse-dense-mode
 no ip mroute-cache
 frame-relay interface-dlci 309
!
router ospf 1
 network 10.0.0.0 0.255.255.255 area 0
!
ip pim send-rp-discovery Loopback0 scope 5
!
```

命令 **ip pim sparse-dense-mode** 在所有物理接口上被配置。它与图 6-6 中的所有路由器中的所有物理接口的配置是相似的。Loopback 接口只在稀疏模式下使用，因为只有映射代理地址需要这个地址，不会在稀疏/密集模式的判断时使用。接口 E0/0 也可以设为稀疏模式，因为它不会因为有下游路由器而必须进行稀疏/密集模式的判断。不过实际中把所有的接口都配置成稀疏—密集模式来保持一致。事实上，在所有现在的 PIM 域中，只要路由器支持这种模式，这就是一种推荐的模式。

例 6-37 中显式了 Homburg 重新配置后的多播路由表。注意到(*,224.0.1.39)与(*,224.0.1.40)条目都有一个 D 标帜，表示它们运行在密集模式下。所有(*,G)条目都标识为稀疏模式。

除了两个 Auto-RP 组，有时你会希望有一些组运行在稀疏模式，另一些组运行在密集模式。像上一个案例中学到的，通过在 C-RP 上运行 **ip pim send-rp-annouce group-list** 命令可以规定哪些组可以映射到 RP，运行在稀疏模式下。没有映射到 RP 的组会运行在密集模式下。

例 6-37　Homburg 中组的工作模式

```
Homburg#show ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop, State/Mode
(*, 228.13.20.216), 00:20:42/00:02:59, RP 10.224.1.2, flags: SJCF
  Incoming interface: Serial0/1.305, RPF nbr 10.2.1.2
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse-Dense, 00:20:42/00:02:43
(10.1.1.88/32, 228.13.20.216), 00:20:42/00:02:59, flags: CFT
  Incoming interface: Ethernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1.305, Forward/Sparse-Dense, 00:20:04/00:02:47
(*, 224.2.127.254), 00:20:34/00:02:59, RP 10.224.1.2, flags: SJCF
  Incoming interface: Serial0/1.305, RPF nbr 10.2.1.2
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse-Dense, 00:20:34/00:02:42
(10.1.1.88/32, 224.2.127.254), 00:20:34/00:02:56, flags: CFT
  Incoming interface: Ethernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial0/1.305, Forward/Sparse-Dense, 00:20:06/00:02:44
(*, 224.0.1.39), 00:20:32/00:00:00, RP 0.0.0.0, flags: DJCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse-Dense, 00:20:32/00:00:00
    Serial0/1.305, Forward/Sparse-Dense, 00:20:32/00:00:00
    Serial0/1.309, Forward/Sparse-Dense, 00:20:32/00:00:00
(10.224.1.1/32, 224.0.1.39), 00:20:32/00:02:27, flags: CLT
  Incoming interface: Serial0/1.309, RPF nbr 10.2.2.2
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse-Dense, 00:20:32/00:00:00
    Serial0/1.305, Forward/Sparse-Dense, 00:20:32/00:00:00

(10.224.1.2/32, 224.0.1.39), 00:19:54/00:02:05, flags: CLT
  Incoming interface: Serial0/1.305, RPF nbr 10.2.1.2
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse-Dense, 00:19:54/00:00:00
    Serial0/1.309, Forward/Sparse-Dense, 00:19:54/00:02:08

(*, 224.0.1.40), 00:20:13/00:00:00, RP 0.0.0.0, flags: DJCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse-Dense, 00:20:14/00:00:00
    Serial0/1.305, Forward/Sparse-Dense, 00:20:14/00:00:00
    Serial0/1.309, Forward/Sparse-Dense, 00:20:14/00:00:00

(10.224.1.4/32, 224.0.1.40), 00:20:06/00:02:48, flags: CLT
  Incoming interface: Loopback0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Ethernet0/0, Forward/Sparse-Dense, 00:20:06/00:00:00
    Serial0/1.305, Forward/Sparse-Dense, 00:20:06/00:00:00
    Serial0/1.309, Forward/Sparse-Dense, 00:20:06/00:00:00

Homburg#
```

6.3.4　案例研究：配置自举协议

当PIMv2在RFC 2117中首先提出时，自举协议被定义为自动发现RP的一种机制。Cisco首次在Cisco IOS Software Release 11.3T中支持PIMv2，自举协议也包括在其中。

配置自举协议有两个步骤和Auto-RP的配置相似：

1．所有的候选RP必须配置

2．所有的候选自举路由器(C-BSR)必须配置

图6-7中显示与前两个案例相同的拓扑，不过现在它运行的是自举协议而不是Auto-RP协议。Stetson与Fedora再一次成为C-RP，但为了符合更强健的设计，它们也是C-BSR，在故障时进行RP与BSR的切换。

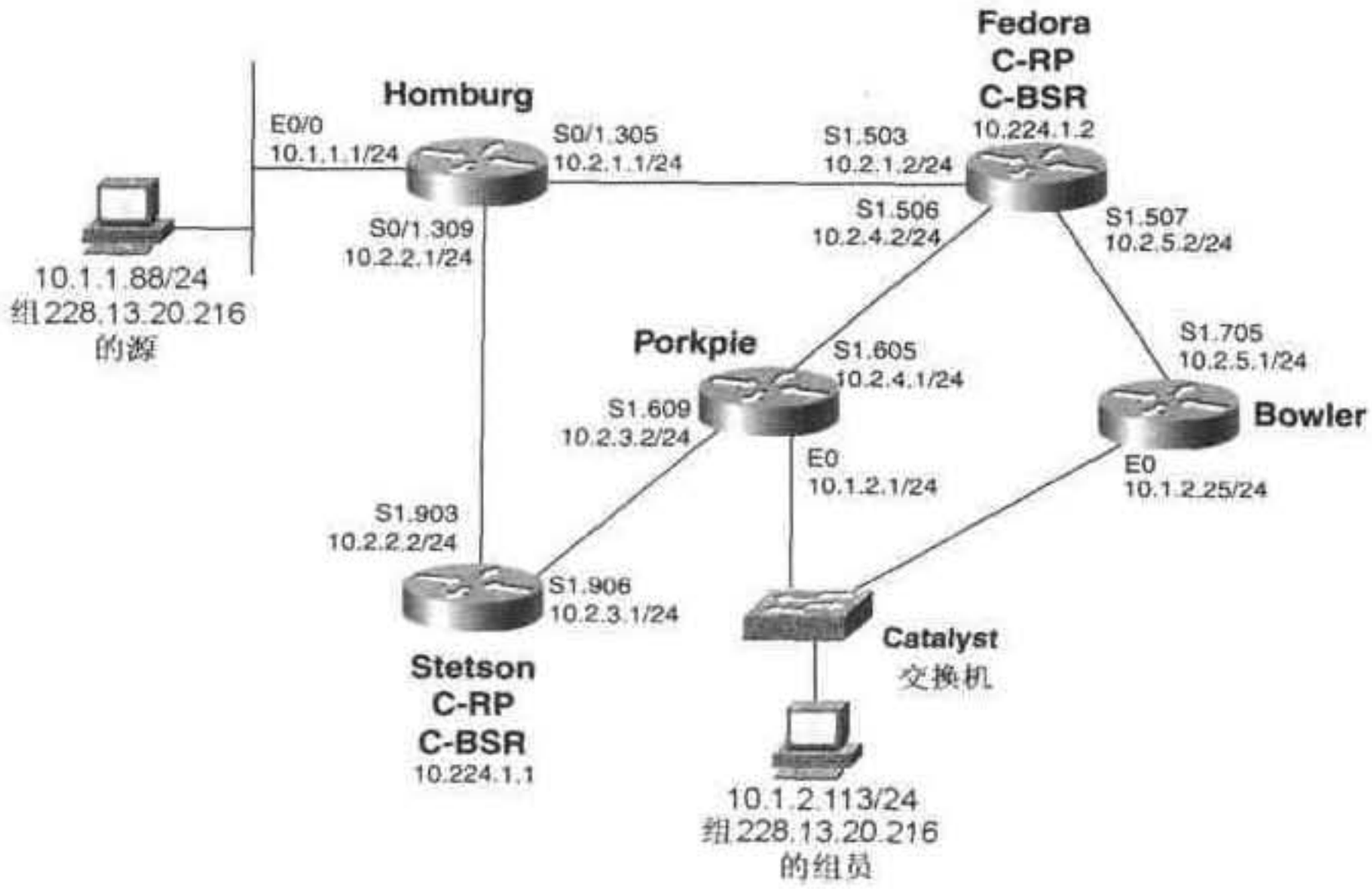


图6-7　Stetson与Fedora均既当候选RP也当候选BSR

例6-38中显示了Stetson与Fedora的相关配置。

例6-38　配置路由器Stetson与Fedora同时配置成候选RP与候选BSR

Stetson

```
interface Loopback0
 ip address 10.224.1.1 255.255.255.255
!
ip pim bsr-candidate Loopback0 0
ip pim rp-candidate Loopback0
```

Fedora

```
interface Loopback0
 ip address 10.224.1.2 255.255.255.255
!
ip pim bsr-candidate Loopback0 0
ip pim rp-candidate Loopback0
```

命令 **ip pim bsr-candidate** 设置路由器为 C-BSR，并规定了 BSR 的地址为接口 L0 的地址。命令结尾的 0 表示 hash 掩码的长度，在 Cisco 路由器上默认为 **0**。hash 掩码的使用在本案例稍后有演示。命令 **ip pim rp-candidate** 设置路由器为 C-RP 并设 RP 的地址也是接口 L0 的地址。

首先，一台 BSR 必须从 C-BSR 中选出。这些 C-BSR 在整个 PIM 域中发送 Bootstrap 消息，这个消息目的地址设为 224.0.0.13，并含有发起者 BSR 的地址与优先权。迄今为止，在配置中，默认的优先权 0，默认的 hash 掩码 0 还没有任何变化，因此在两个 C-BSR 上都是相等的。这样就采用高的 BSR 地址值来打破僵局。Fedora 的 BSR 地址值(10.224.1.2)比 Stetson(10.224.1.1)的值要高，所以 Fedora 成为 BSR。例 6-39 中证实了这一点。用 **show ip pim bsr-router** 命令可以看出在这个域中的任何路由器，你不仅可以观察激活的路由器，而且还包括 BSR 的地址、启动时间、优先权、hash 掩码长度和保持时间。

例 6-39　show ip pim bsr-router 命令显示了 PIMv2 域中的 BSR

```
Bowler#show ip pim bsr-router
PIMv2 Bootstrap information
  BSR address: 10.224.1.2 (?)
  Uptime:      00:17:35, BSR Priority: 0, Hash mask length: 0
  Expires:     00:01:56
Bowler#
```

当 C-RP 收到 Bootstrap 消息并判断了 BSR 的地址时，它用单播向 BSR 发送 Candidate-RP-Advertisement 消息。这个消息有 C-RP 的地址和优先权。BSR 收集 C-RP 构成 RP 集，在 Bootstrap 消息中包括了 RP 集。这是自举协议与 Auto-RP 协议完全不同的一点：与 Auto-RP 映射代理不同，BSR 不进行 RP 的选举。PIMv2 路由器收到 Bootstrap 消息，它们自己选择 RP。算法保证了所有的路由器对相同的组选择相同的 RP。

例 6-40 显示了 Bowler 中组到 RP 的映射，Bowler 中激活的组均映射到 Stetson；不同于 Auto-RP，有最低 RP 地址值的 C-RP 被选为 RP。你可以看到 RP 是 Stetson，因为它有较低的 RP 地址值(这个例子中 C-RP 的其他属性都相同)。

例 6-40　激活组的 RP

```
Bowler#show ip pim rp
Group: 239.255.255.254, RP: 10.224.1.1, v2, uptime 00:25:16, expires 00:02:40
Group: 228.13.20.216, RP: 10.224.1.1, v2, uptime 00:25:16, expires 00:02:40
Group: 224.2.127.254, RP: 10.224.1.1, v2, uptime 00:25:16, expires 00:02:40
Group: 230.253.84.168, RP: 10.224.1.1, v2, uptime 00:25:16, expires 00:02:40
Bowler#
```

例 6-41 中显示了映射到 RP 的完整的组地址范围。比较例 6-33 与这里显示的内容，特别之处是显示的映射是从自举过程中得到的，路由器从 RP 集中知道所有的 C-RP。

可以改变 BSR 与 RP 默认的行为。在例 6-39 中，BSR 为 Fedora，因为它的 IP 地址值更高，如果你要让 Stetson 成为 BSR，Fedora 成为备份，你可以改变 Stetson 的优先权，使其高于 0。要把 Stetson 的优先权改为 100，你需要按例 6-42 所示配置 Stetson。

例 6-41　Bowler 表明它知道 Stetson 与 Fedora 均为 C-RP

```
Bowler#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/4
  RP 10.224.1.1 (?), v2
    Info source: 10.224.1.2 (?), via bootstrap
         Uptime: 00:29:07, expires: 00:02:30
  RP 10.224.1.2 (?), v2
    Info source: 10.224.1.2 (?), via bootstrap
         Uptime: 00:29:07, expires: 00:02:17
Bowler#
```

例 6-42　将 Stetson 的优先权设为 100 使之成为 BSR

```
interface Loopback0
 ip address 10.224.1.1 255.255.255.255
!
ip pim bsr-candidate Loopback0 0 100
ip pim rp-candidate Loopback0
```

在例 6-43 中显示了新配置的结果，Bowler 现在显示 Stetson 为 BSR，优先权为 100。Fedora 在 Stetson 故障时承担其任务。

例 6-43　Stetson(10.224.1.1)，有优先权值 100，成为 BSR

```
Bowler#show ip pim bsr-router
PIMv2 Bootstrap information
  BSR address: 10.224.1.1 (?)
  Uptime:      00:10:27, BSR Priority: 100, Hash mask length: 0
  Expires:     00:02:02
Bowler#
```

同 Auto-RP 一样，你可以用访问表在多个 RP 间分配 RP 任务。比如，假设你要 Fedora 成为 228.13.0.0/16 地址范围内的 RP，Stetson 为其他所有组的 RP，其配置如例 6-44 所示。

例 6-44　Fedora 与 Stetson 间的 RP 任务分配

Stetson

```
interface Loopback0
 ip address 10.224.1.1 255.255.255.255
!
ip pim bsr-candidate Loopback0 0 100
ip pim rp-candidate Loopback0 group-list 20
!
access-list 20 deny   228.13.0.0 0.0.255.255
access-list 20 permit any
```

Fedora

```
interface Loopback0
 ip address 10.224.1.2 255.255.255.255
!
ip pim bsr-candidate Loopback0 0
```

```
ip pim rp-candidate Loopback0 group-list 10
!
access-list 10 permit 228.13.0.0 0.0.255.255
```

例 6-45 中显示了这些配置的结果，例中访问表限制了 Stetson 与 Fedora 中 RP 的映射，Bowler 将组 228.13.20.216 映射到 Fedora，而其他映射到 Stetson。BSR 把这些限制通过 Bootstrap 消息进行公告，Bowler 根据这些限制把它的组映射到不同的 RP。当然，这些配置不建议在真正的网络使用，因为如果一个 RP 失效，另一个不再担当备份的角色。一个更加实际的做法是用访问表在多个 C-RP 间分配组。对于一个组地址范围，访问表至少分配两个 C-RP。

例 6-45　访问表对组映射的限制

```
Bowler#show ip pim rp mapping
PIM Group-to-RP Mappings

Group(s) 224.0.0.0/4
  RP 10.224.1.1 (?), v2
    Info source: 10.224.1.1 (?), via bootstrap
         Uptime: 00:07:25, expires: 00:02:26
Group(s) 228.13.0.0/16
  RP 10.224.1.2 (?), v2
    Info source: 10.224.1.1 (?), via bootstrap
         Uptime: 00:07:25, expires: 00:02:54

Bowler#show ip pim rp
Group: 239.255.255.254, RP: 10.224.1.1, v2, uptime 00:07:30, expires 00:02:52
Group: 228.13.20.216, RP: 10.224.1.2, v2, uptime 00:07:30, expires 00:03:32
Group: 224.2.127.254, RP: 10.224.1.1, v2, uptime 00:07:30, expires 00:02:52
Group: 230.253.84.168, RP: 10.224.1.1, v2, uptime 00:07:30, expires 00:02:52
Bowler#
```

一个分配 RP 任务更好的方法是在 PIMv2 自举协议中使用 hash 掩码。hash 掩码是一个分配给 BSR 的 32 比特的数字，它的使用有些像 IP 地址掩码。BSR 在 Bootstrap 消息中宣告了它的 hash 掩码，接收路由器运行一个 hash 算法来为连续的组地址分配一个 C-RP，为下一组地址分配另一个 C-RP。

比如，如果 hash 掩码为 30 比特，它掩 IP 多播地址的前 30 位。最后两个比特所表示的 4 个组地址将分配给一个 RP。所以 225.1.1.0、225.1.1.1、225.1.1.2 与 225.1.1.3 是一个范围内的地址，分配到同一 RP。地址 225.1.1.4、225.1.1.5、225.1.1.6 与 225.1.1.7 是另一个范围内的地址，分配到另一个 RP 上。这样把组地址捆绑起来，完全覆盖整个 IP 多播地址范围，分配到所有可用的 C-RP 上。这个结果就是 IP 多播组地址在 C-RP 中分配。这个掩码给你一个判断用多少个连续地址组成一个地址范围，这样相关的地址更有可能共享一台 RP。如果这个掩码是 26 比特，一个地址范围分配 64 个连续地址。

hash 掩码长度在 **ip pim bsr-candidate** 命令的部分中定义。如你在前面的例子中已经观察到的，默认的掩码长度是 0，意味着整个多播地址段为一个单独的地址范围。例 6-46 中显示了为图 6-7 中的 Stetson 与 Fedora 分配 30 比特长的 hash 掩码长度的配置。

例 6-46　分配长度为 30 的掩码到路由器 Stetson 与 Fedora

```
Stetson
interface Loopback0
 ip address 10.224.1.1 255.255.255.255
!
ip pim bsr-candidate Loopback0 30
ip pim rp-candidate Loopback0
```

```
Fedora
interface Loopback0
 ip address 10.224.1.2 255.255.255.255
!
ip pim bsr-candidate Loopback0 30
ip pim rp-candidate Loopback0
```

在例 6-47 中，**show ip pim rp-hash** 命令用于显示这个结果，开始于 231.1.1.0，你可以看到它与下面 3 个连续的组地址映射到 Fedora。继续这个序列，下面 4 个地址映射到 Stetson。整个 IP 多播地址范围应在两个 RP 间平均分配。

例 6-47　hash 算法多播地址在所有可用 C-RP 间平均分配

```
Bowler#show ip pim rp-hash 231.1.1.0
  RP 10.224.1.2 (?), v2
    Info source: 10.224.1.2 (?), via bootstrap
         Uptime: 07:22:14, expires: 00:02:29
Bowler#show ip pim rp-hash 231.1.1.1
  RP 10.224.1.2 (?), v2
    Info source: 10.224.1.2 (?), via bootstrap
         Uptime: 07:22:19, expires: 00:02:24
Bowler#show ip pim rp-hash 231.1.1.2
  RP 10.224.1.2 (?), v2
    Info source: 10.224.1.2 (?), via bootstrap
         Uptime: 07:22:22, expires: 00:02:21
Bowler#show ip pim rp-hash 231.1.1.3
  RP 10.224.1.2 (?), v2
    Info source: 10.224.1.2 (?), via bootstrap
         Uptime: 07:22:28, expires: 00:02:15
Bowler#show ip pim rp-hash 231.1.1.4
  RP 10.224.1.1 (?), v2
    Info source: 10.224.1.2 (?), via bootstrap
         Uptime: 07:22:31, expires: 00:02:13
Bowler#show ip pim rp-hash 231.1.1.5
  RP 10.224.1.1 (?), v2
    Info source: 10.224.1.2 (?), via bootstrap
         Uptime: 07:22:35, expires: 00:02:10
Bowler#show ip pim rp-hash 231.1.1.6
  RP 10.224.1.1 (?), v2
    Info source: 10.224.1.2 (?), via bootstrap
         Uptime: 07:22:38, expires: 00:02:06
Bowler#show ip pim rp-hash 231.1.1.7
  RP 10.224.1.1 (?), v2
    Info source: 10.224.1.2 (?), via bootstrap
         Uptime: 07:22:43, expires: 00:02:02
```

6.4　案例研究：多播负荷分担

有时，为了充分利用带宽或防止一条通路有过重的多播流量造成拥塞，你会希望在平行

等开销的通路间进行多播流量的均摊。但 RPF 检测会防止在物理链路上直接进行负荷分担。

在图 6-8 中显示出来的问题是与前面案例研究中的 PIM 拓扑相同，除了没有 Bowler 外，Homburg 既是 Auto-RP 的映射代理也是 RP。

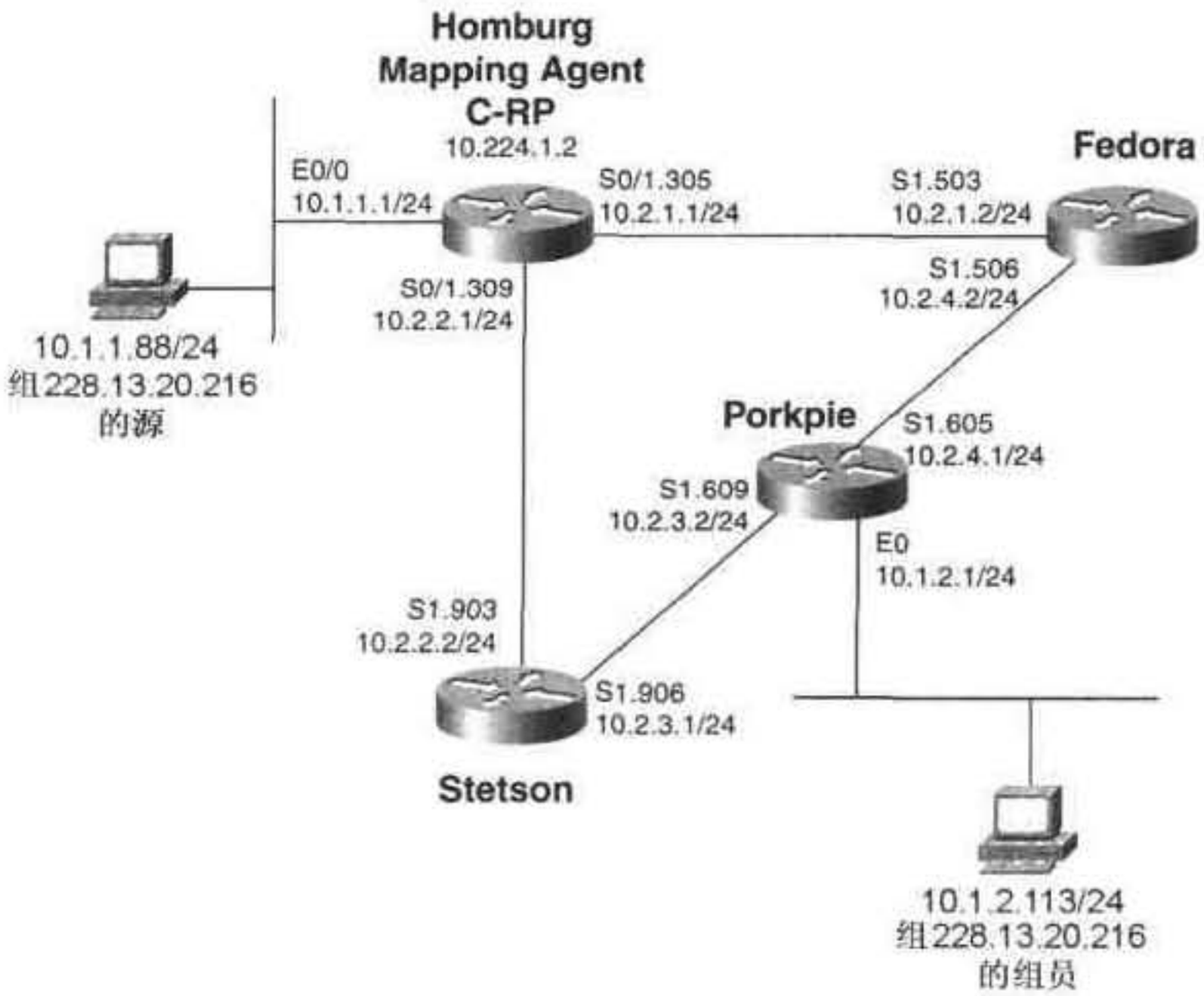


图 6-8　在多播源与组员间有等开销通路

从连接着源的 Homburg 到连接着组员的 Porkpie，有两条等开销的通路：一条经过 Fedora；另一条经过 Stetson。RPF 必须只有一条输入接口时才能正常工作，因此就产生了问题。如果 Fedora 被选为 RPF 邻居，来自 Stetson 的流量因为不是来自 RPF 接口上，会被丢弃。同样，如果 Stetson 被选为 RPF 邻居，来自 Fedora 的流量会因为 RPF 检测无效，进而被丢弃。RPF 要求所有的流量都在同一个上游接口上收到。

解决这个问题的办法是使用隧道，如图 6-9 所示。隧道是建立在 Homburg 与 Porkpie 两个 loopback 接口间，的所有从源到组员的流量都通过这个虚拟的隧道，而不是物理的链路。多播包经封装后，像一般的 IP 包一样进行前转。这一点来说，封装的包可以如第 1 卷中所述，用基于目的的均衡或基于包的均衡，在两条链路上实现负载均衡。

> 注：　基于包的负载均衡通过关闭快速交换实现，或用命令 no ip route-cache 也同样能实现。

当包到达 Porkpie 时，它是从 Fedora 收到的还从 Stetson 收到的并不重要，因为这些包的目的地址均为隧道的出口。在虚拟隧道的接口处，封装被去除。从 Porkpie 上 PIM 处理过程的角度来说，多播包好像是从同一个接口 TU0 上收到的，也均来自同一个上流邻居 Homburg。

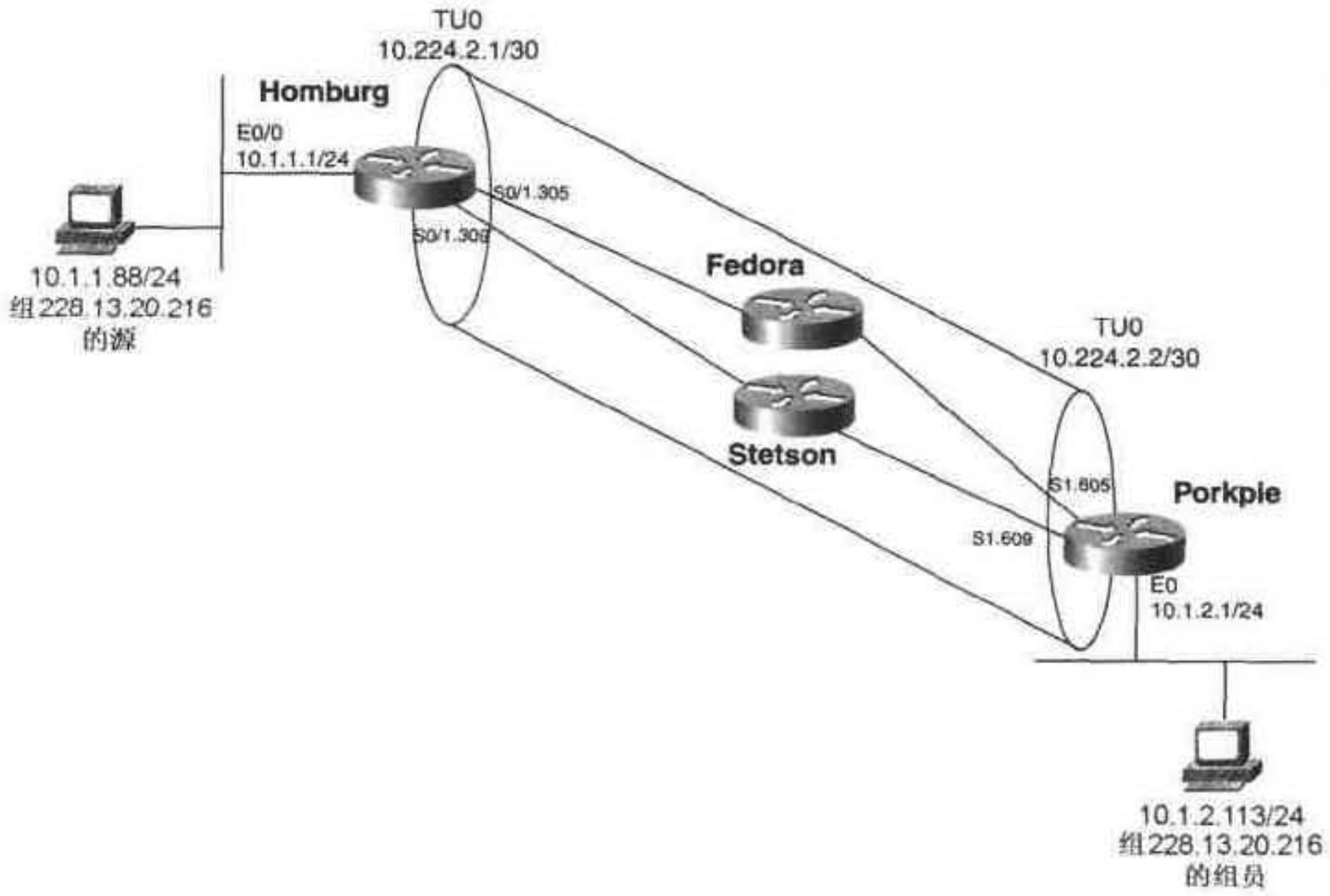


图 6-9　在等开销通路由的负荷分摊，在 Homburg 与 Porkpie 间建立一条隧道

例 6-48 显示了 Homburg 与 Porkpie 的配置。

例 6-48　在 Homburg 与 Porkpie 间配置隧道，在等开销通路上进行负载分摊

```
Homburg
hostname Homburg
!
ip multicast-routing
!
interface Loopback0
 ip address 10.224.1.4 255.255.255.0
 ip pim sparse-mode
!
interface Tunnel0
 ip address 10.224.2.1 255.255.255.252
 ip pim sparse-dense-mode
 tunnel source Loopback0
 tunnel destination 10.224.1.3
!
interface Ethernet0/0
 ip address 10.1.1.1 255.255.255.0
 ip pim sparse-dense-mode
!
interface Serial0/1
 no ip address
 encapsulation frame-relay
!
interface Serial0/1.305 point-to-point
 description PVC to R5
 ip address 10.2.1.1 255.255.255.0
 frame-relay interface-dlci 305
!
interface Serial0/1.309 point-to-point
 description PVC to R9
```

```
 ip address 10.2.2.1 255.255.255.0
 frame-relay interface-dlci 309
!
router ospf 1
 passive-interface Tunnel0
 network 10.0.0.0 0.255.255.255 area 0
!
ip pim send-rp-announce Loopback0 scope 5
ip pim send-rp-discovery scope 5
```

Porkpie

```
hostname Porkpie
!
ip multicast-routing
!
interface Loopback0
 ip address 10.224.1.3 255.255.255.255
!
interface Tunnel0
 ip address 10.224.2.2 255.255.255.252
 ip pim sparse-dense-mode
 tunnel source Loopback0
 tunnel destination 10.224.1.4
!
interface Ethernet0
 ip address 10.1.2.1 255.255.255.0
 ip pim sparse-dense-mode
 ip cgmp
!
interface Serial1
 no ip address
 encapsulation frame-relay
!
interface Serial1.605 point-to-point
 description PVC to R5
 ip address 10.2.4.1 255.255.255.0
 frame-relay interface-dlci 605
!
interface Serial1.609 point-to-point
 description PVC to R9
 ip address 10.2.3.2 255.255.255.0
 frame-relay interface-dlci 609
!
router ospf 1
 passive-interface Tunnel0
 network 10.0.0.0 0.255.255.255 area 0
```

两台路由器上，隧道接口配置成源地址为一台路由器的 loopback 地址，目的地址为另一台路由器的 loopback 地址。这条隧道采用 GRE(Generic route encapsulation)，并分配一个 IP 地址，这样这个虚拟接口在路由处理过程中为一个物理接口。最后在隧道接口上启动 PIM，注意不要在 Stetson 与 Fedora 的任何子接口上配置 PIM。在这两个路由器上，多播并没有启动。例 6-49 显示了 Porkpie 通过隧道与 Homburg 建立连接。

例 6-49　Porkpie 显示 Homburg 是通过 GRE 隧道连接的邻居

```
Porkpie#show ip pim neighbor
PIM Neighbor Table
Neighbor Address  Interface          Uptime    Expires   Ver  Mode
10.224.2.1        Tunnel0            04:09:21  00:01:11  v1   Sparse-Dense
Porkpie#
```

不过这里还有进一步的 RPF 问题要解决。当 Porkpie 从源 10.1.1.88 收到包时，它检查单播路由表来找到上游邻居。例 6-50 显示了路由器发现了什么。

例 6-50　单播路由表显示 10.2.3.1 或 10.2.4.2 到 10.1.1.88 的下一跳地址

```
Porkpie#show ip route 10.1.1.88
Routing entry for 10.1.1.0/24
  Known via "ospf 1", distance 110, metric 138, type intra area
  Redistributing via ospf 1
  Last update from 10.2.3.1 on Serial1.609, 01:13:30 ago
  Routing Descriptor Blocks:
  * 10.2.3.1, from 10.224.1.4, 01:13:30 ago, via Serial1.609
      Route metric is 138, traffic share count is 1
    10.2.4.2, from 10.224.1.4, 01:13:30 ago, via Serial1.605
      Route metric is 138, traffic share count is 1

Porkpie#
```

Porkpie 的 OSPF 配置将 TU0 设为被动模式来保证不会有单播的流量经过隧道，只会有多播流量流过。不幸的是 OSPF 仍然认为 Stetson(10.2.3.1)与 Fedora(10.2.4.2)为到 10.1.1.88 的下一跳。这样当来自 10.1.1.88 的包到达隧道接口时，RPF 检测会失败，如例 6-51 所示，例中 RPF 对从隧道上收到的 10.1.1.88 的包检测失败，因为单播路由表没有显示 TU0 为这个地址的上游地址。

例 6-51　TU0 的 RPF 检测

```
Porkpie#debug ip mpacket
IP multicast packets debugging is on
Porkpie#
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 len 569, not RPF interface
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 len 569, not RPF interface
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 len 569, not RPF interface
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 len 569, not RPF interface
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 len 569, not RPF interface
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 len 569, not RPF interface
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 len 569, not RPF interface
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 len 569, not RPF interface
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 len 569, not RPF interface
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 len 569, not RPF interface
```

为了克服这个问题，采用了静态多播路由。静态多播路由与静态单播路由相似，它们可以忽略单播路由表中的所有信息。ip mroute 命令用一个 IP 地址和掩码可以定义一个地址或一组地址的静态路由。一个 RPF 接口或 RPF 邻居地址也可定义，就像单播路由定义一个输出接口或者一个下一跳邻居。例 6-52 显示了有静态 mroute 的 Porkpie 配置。

例 6-52　Porkpie 配置一个静态 mroute

```
hostname Porkpie
!
ip multicast-routing
```

```
!
interface Loopback0
 ip address 10.224.1.3 255.255.255.255
!
interface Tunnel0
 ip address 10.224.2.2 255.255.255.252
 ip pim sparse-dense-mode
 tunnel source Loopback0
 tunnel destination 10.224.1.4
!
interface Ethernet0
 ip address 10.1.2.1 255.255.255.0
 ip pim sparse-dense-mode
 ip cgmp
!
interface Serial1
 no ip address
 encapsulation frame-relay
!
interface Serial1.605 point-to-point
 description PVC to R5
 ip address 10.2.4.1 255.255.255.0
 frame-relay interface-dlci 605
!
interface Serial1.609 point-to-point
 description PVC to R9
 ip address 10.2.3.2 255.255.255.0
 frame-relay interface-dlci 609
!
router ospf 1
 passive-interface Tunnel0
 network 10.0.0.0 0.255.255.255 area 0
!
ip mroute 10.1.1.88 255.255.255.255 Tunnel0
```

例 6-53 中再次使用查错方法来证实 Porkpie 上多播包通过了 RPF 检测，并前转给组员。

例 6-53　来自多播源 10.1.1.88 的包到达隧道接口通过了 RPF 检测并前转

```
Porkpie#debug ip mpacket
IP multicast packets debugging is on
Porkpie#
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 (Ethernet0) len 569, mforward
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 (Ethernet0) len 569, mforward
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 (Ethernet0) len 569, mforward
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 (Ethernet0) len 569, mforward
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 (Ethernet0) len 569, mforward
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 (Ethernet0) len 569, mforward
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 (Ethernet0) len 569, mforward
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 (Ethernet0) len 569, mforward
IP: s=10.1.1.88 (Tunnel0) d=228.13.20.216 (Ethernet0) len 569, mforward
```

例 6-54 中显示了小组 228.13.20.216 的 mroute 条目。你可以观察到 Homburg 在接口 E0/0 上，从 10.1.1.88 收到多播流量，并把这些流量前转到隧道上。Porkpie 从隧道上收到多播流量，并从它的 E0 接口上前转给组员。

例6-54　(10.1.1.88,228.13.20.216)的mroute条目表明这个组的流量经GRE隧道前转

```
Homburg#show ip mroute 228.13.20.216
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
Timers: Uptime/Expires
Interface state: Interface, Next-Hop, State/Mode

(*, 228.13.20.216), 04:48:39/00:02:59, RP 10.224.1.4, flags: SJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Tunnel0, Forward/Sparse-Dense, 01:35:18/00:02:01
    Ethernet0/0, Forward/Sparse-Dense, 04:48:39/00:02:59

(10.1.1.88/32, 228.13.20.216), 01:41:09/00:02:59, flags: CT
  Incoming interface: Ethernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Tunnel0, Forward/Sparse-Dense, 01:35:19/00:02:01

Homburg#
```

```
Porkpie#show ip mroute 228.13.20.216
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set, J - Join SPT
       M - MSDP created entry, X - Proxy Join Timer Running
       A - Advertised via MSDP
Outgoing interface flags: H - Hardware switched
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 228.13.20.216), 00:56:23/00:02:59, RP 10.224.1.4, flags: SJC
  Incoming interface: Tunnel0, RPF nbr 10.224.2.1, Mroute
  Outgoing interface list:
    Ethernet0, Forward/Sparse-Dense, 00:56:23/00:02:58


Porkpie#
```

6.5　IP多播路由的故障排除

在解决IP多播网络中的问题时，你最基本的工具是对IP多播协议扎实的理解。没有这一点，没有一个排错工具能帮助你在混乱有时甚至是很复杂的IP多播行为里找到一条查出基本问题的出路。仅对一个协议的理解是不够的，你必须理解PIM、IGMP和单播路由是如何相互作用的。

如果你已经仔细读过第1卷和这一卷中的每一章后的排错部分，你应该已经掌握了解决寻路网络中解决问题的技能和方法。所以这一章不再列出案例研究与排错的技巧，而演示几个分析多播网际网络的特殊工具。

经过这一章，你会看到各种**show**和**debug**命令，这些对于观察Cisco路由器上IP多播

路由行为是非常有用的。表 6-2 列出了对你有用的 **show** 命令，表 6-3 列出了重要的 **debug** 命令。如同 **show ip route** 是 IP 单播路由排错最基本的命令一样，**show ip mroute** 是 IP 多播路由排错的基本命令。

表 6-2　　IP 多播排错中重要的 show 命令

命　　令	描　　述
show ip igmp groups [*组名|组地址|类型号*]	显示这台路由器的接口上有组员的多播组地址
show ip igmp interface [*类型号*]	显示启动 IGMP 接口相关的详细信息
show ip mcache [*组*[*源*]]	显示快速交换缓存中的多播内容
show ip morute [*组名|组地址*] [*源*] [**summary**] [**count**] [**active** *kbit/s*]	显示多播路由表的内容
show ip pim bsr	显示 PIM 自举路由器的信息
show ip pim interface [*类型号*] [**count**]	显示启动 PIM 的接口的相关详细信息
show ip pim neighbor [*类型号*]	显示 PIM 邻居
show ip pim rp [*组名|组地址*|**mapping**]	显示已知的 RP，和映射到 RP 的组
show ip pim rp-hash *组*	显示这个组定义的 RP
show ip rpf [*源地址|名称*]	显示这台路由器如何判断 RPF 信息

表 6-3　　IP 多播排错中的重要 debug 命令

命　　令	描　　述
debug ip icmp [*主机名|组地址*]	显示 IGMP 协议的活动
debug ip mcache [*主机名|组地址*]	显示多播缓存的操作
dehug ip mpacket [*标准访问表|扩展访问表*] [*主机名|组地址*||**detail**]	显示经过这台路由器的多播包
debu ip mrouting [*主机名|组地址*]	显示多播路由表的活动
debug ip pim [*主机名|组地址*] [**auto-rp**] [**bsr**]	显示 PIM 活动与事件

6.5.1　使用 mrinfo

使用 **mrinfo** 命令可以观察路由器的多播连接和这些连接的细节。这个命令源于 MBone 中为测试路由器 mrouted 的组成工具中的一部分。因此这个命令在多域的环境中非常有用。看一下拓扑图 6-10。

在例 6-55 中，**mrinfo** 在路由器 Sombrero 上使用。输出的第一行显示了查询消息的源地址，在这台路由器上运行 Cisco IOS 软件版本和一些标帜。表 6-4 列出了可能的标帜与它们的

含义。输出的下两行显示了路由器上的多播接口和与本路由器可能建立关系的对端。在第二行，Sombrero 的接口 192.168.10.1 没有对端，用 0.0.0.0 来表示。1/0 表示这个接口的量度为 1 且没有 TTL 门限设置。PIM 在这个接口上运行，路由器对于连接的子网是一个 IGMP 查询者，子网为一个叶网络(就是说没有到别的网络的多播流量穿过这个网络)。第三行显示，Sombrero 的接口 192.168.200.1 有一个对端 192.168.200.2(路由器 Beret)，接口的量度是 1，没有 TTL 门限和运行了 PIM。

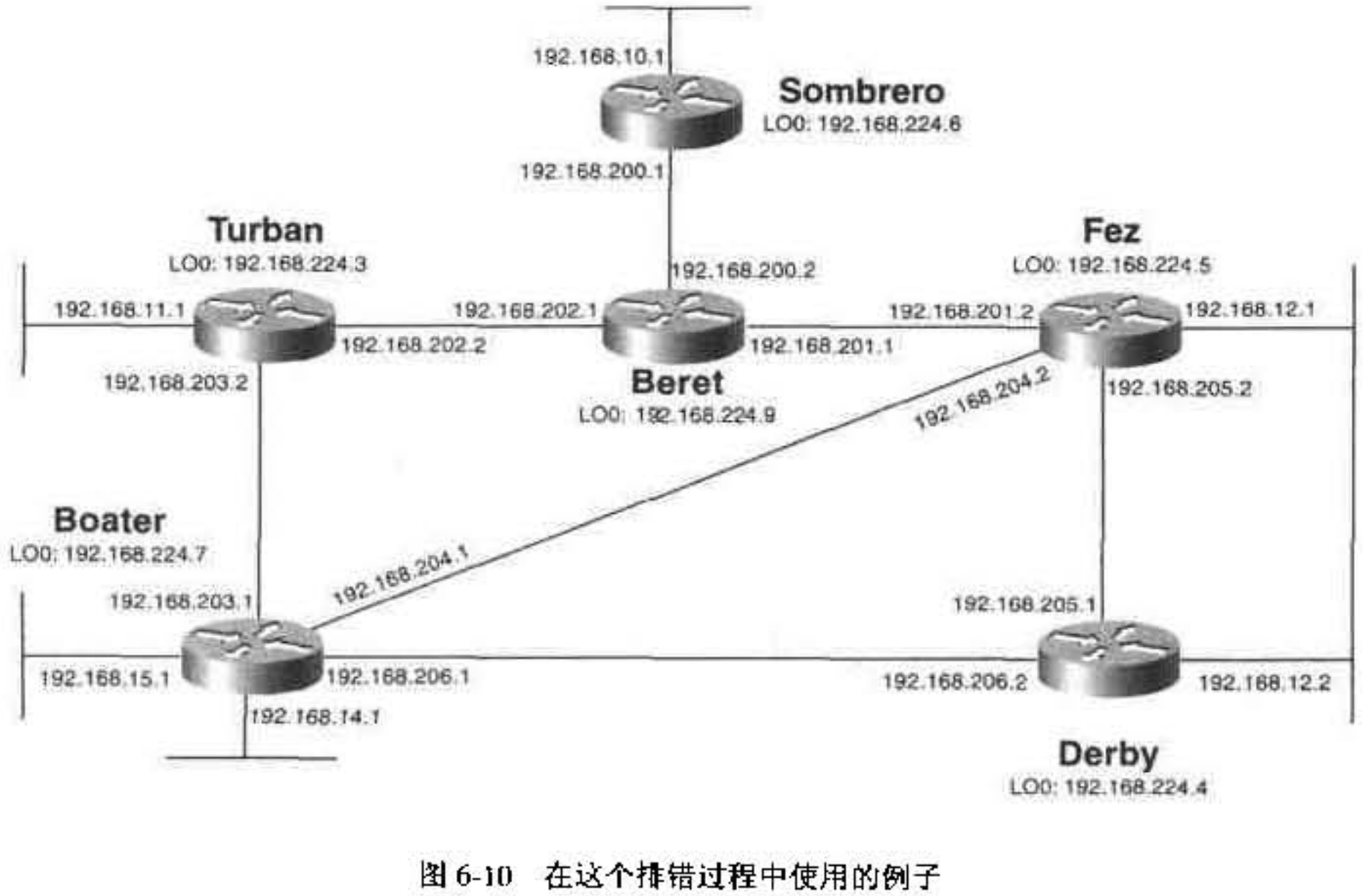


图 6-10　在这个排错过程中使用的例子

例 6-55　图 6-10 中 Soombrero 的 IP 多播连接信息

```
Sombrero#mrinfo
192.168.10.1 [version  12.1] [flags: PMA]:
  192.168.10.1 -> 0.0.0.0 [1/0/pim/querier/leaf]
  192.168.200.1 -> 192.168.200.2 [1/0/pim]

Sombrero#
```

表 6-4　　与 mrinfo 相关的标帜

标　帜	定　义
P	可以支持剪除
M	可以支持 mtrace
S	可以支持 SNMP
A	可以支持 Auto-RP

Mrinfo 的真正用处是你可以用命令来查询域内的其他路由器。在例 6-56 中，在 Sombrero

上用这个命令查询 Boater，在这个命令里指定了 Boater 的 IP 地址(本例中，是 Boater 的 loopback 地址)。注意到路由器的标帜表明它支持 SNMP，但是 Sombrero 并不支持。这台路由器有五个支持多播的接口，两个在叶网络上，三个有 PIM 对端。检查图 6-10 的拓扑结构，可以肯定这个信息是正确的。

例 6-56　在 Sombrero 用 **minfo** 查询 Boater 的多播对等端

```
Sombrero#mrinfo 192.168.224.7
192.168.224.7 [version  12.1] [flags: PMSA]:
  192.168.14.1 -> 0.0.0.0 [1/0/pim/querier/leaf]
  192.168.15.1 -> 0.0.0.0 [1/0/pim/querier/leaf]
  192.168.203.1 -> 192.168.203.2 [1/0/pim]
  192.168.206.1 -> 192.168.206.2 [1/0/pim]
  192.168.204.1 -> 192.168.204.2 [1/0/pim]

Sombrero#
```

在例 6-57 中，查询路由器 Derby 与 Fez。这两个路由器共享以太网连接。比较这两个查询结果，表明 Derby(192.168.224.4)是子网中的 IGMP 查询者。

例 6-57　从 Sombrero 查询 Derby(192.168.224.4)与 Fez(192.168.224.5)

```
Sombrero#mrinfo 192.168.224.4
192.168.224.4 [version  12.1] [flags: PMA]:
  192.168.12.2 -> 192.168.12.1 [1/0/pim/querier]
  192.168.205.1 -> 192.168.205.2 [1/0/pim]
  192.168.206.2 -> 192.168.206.1 [1/0/pim]

Sombrero#mrinfo 192.168.224.5
192.168.224.5 [version  12.1] [flags: PMA]:
  192.168.12.1 -> 192.168.12.2 [1/0/pim]
  192.168.205.2 -> 192.168.205.1 [1/0/pim]
  192.168.204.2 -> 192.168.204.1 [1/0/pim]
  192.168.201.2 -> 192.168.201.1 [1/0/pim]

Sombrero#
```

6.5.2　mtrace 与 mstat 的使用

另一个有用的工具是 **mtrace** 命令，它可以使你从一个目的到一个源去跟踪 RPF 通路。同 **mrinfo** 一样，**mtrace** 是基于 UNIX 的 MBone 工具，可以在多厂商的环境中使用；你也可以像 **mrinfo** 一样在域内的任何一台路由器上发出这个命令，而不必一定在 RPF 通路的一台路由器上。

当发出这个命令时，你规定了源地址和目的地址。一个跟踪请求会向目的地址发送，这里是向多播源发送单播的跟踪。通路上到多播源的第一跳路由器将跟踪的结果用单播转发给查询的路由器。

例 6-58 显示了一个例子，Sombrero 发出一个请求来跟踪从 Derby 的接口 192.168.12.2 到 Turban 的接口 192.168.11.1 的 RPF 通路。记住，因为这是反向路径的跟踪，所以 Turban 的接口是源，Derby 的接口是目的。这个输出从目的开始，直到源，并且显示了路径中每一台路

由器。从源开始的跳数也就是多播协议用到的跳数。

例 6-58　用 mtrace 来检查从目的 192.168.12.2 到源 192.168.11.1 的通路

```
Sombrero#mtrace 192.168.11.1 192.168.12.2
Type escape sequence to abort.
Mtrace from 192.168.11.1 to 192.168.12.2 via RPF
From source (?) to destination (?)
Querying full reverse path...
 0  192.168.12.2
-1  192.168.12.2 PIM  [192.168.11.0/24]
-2  192.168.206.1 PIM  [192.168.11.0/24]
-3  192.168.203.2 PIM  [192.168.11.0/24]
-4  192.168.11.1
Sombrero#
```

除了把多播路由故障隔离出来这一显而易见的作用外，**mtrace** 可以使你在网络中打开实时多播流量前检查多播行为。注意到图 6-10 中没有标出多播源和组员。假设你要打开一个连接在 Boater 上的多播源，其地址为 192.168.14.35。这个源会产生多播组 235.10.20.18 的流量，组员的地址在 192.168.12.15、192.168.10.8 与 192.168.11.102。例 6-59 显示了这一结果。

例 6-59　mtrace 可以用来对不在多播域中的源，目的，组地址测试 RPF

```
Sombrero#mtrace 192.168.14.35 192.168.12.15 235.100.20.18
Type escape sequence to abort.
Mtrace from 192.168.14.35 to 192.168.12.15 via group 235.100.20.18
From source (?) to destination (?)
Querying full reverse path...
 0  192.168.12.15
-1  192.168.201.2 PIM  [192.168.14.0/24]
-2  192.168.204.1 PIM  [192.168.14.0/24]
-3  192.168.14.35

Sombrero#mtrace 192.168.14.35 192.168.10.8 235.100.20.18
Type escape sequence to abort.
Mtrace from 192.168.14.35 to 192.168.10.8 via group 235.100.20.18
From source (?) to destination (?)
Querying full reverse path...
 0  192.168.10.8
-1  192.168.10.1 PIM  [192.168.14.0/24]
-2  192.168.200.2 PIM  [192.168.14.0/24]
-3  192.168.202.2 PIM  [192.168.14.0/24]
-4  192.168.203.1 PIM  [192.168.14.0/24]
-5  192.168.14.35

Sombrero#mtrace 192.168.14.35 192.168.11.102 235.100.20.18
Type escape sequence to abort.
Mtrace from 192.168.14.35 to 192.168.11.102 via group 235.100.20.18
From source (?) to destination (?)
Querying full reverse path...
 0  192.168.11.102
-1  192.168.202.2 PIM  [192.168.14.0/24]
-2  192.168.203.1 PIM  [192.168.14.0/24]
-3  192.168.14.35
Sombrero#
```

在例 6-59 中的跟踪消息规定了多播组的源与目的地址。虽然 RPF 通路一般对所有的组都是一样的，但在限制多播范围或选择的通路受 RP 过滤影响时，指定一个组是很有用的。当没指定一个组时，如例 6-58 所示，默认使用组地址 224.2.0.1(MBone 音频组地址)。

mstat 是 **mtrace** 的修改版，不仅能提供从源到组目的地址的跟踪，还能提供对通路的统计。例 6-60 显示了从源 192.168.14.35 到 192.168.10.8 对于组 235.100.20.18 的跟踪。比较例 6-60 的输出与例 6-59 相同的跟踪命令的输出，可以看出 **mstat** 不仅提供了包的统计还有整个通路的详细情况。

例 6-60　mstat 提供从源到目的更详细的组流量信息

```
Sombrero#mstat 192.168.14.35 192.168.10.8 235.100.20.18
Type escape sequence to abort.
Mtrace from 192.168.14.35 to 192.168.10.8 via group 235.100.20.18
From source (?) to destination (?)
Waiting to accumulate statistics......
Results after 10 seconds:

  Source          Response Dest    Packet Statistics For     Only For Traffic
192.168.14.35     192.168.200.1     All Multicast Traffic     From 192.168.14.35
     |         __/  rtt 47   ms   Lost/Sent = Pct  Rate     To 235.100.20.18
     v      /       hop 27   ms   ---------------------     ---------------------
192.168.14.1
192.168.203.1    ?
     |     ^       ttl   0
     v     |       hop 5    ms    0/0 = --%       0 pps    0/0 = --%  0 pps
192.168.203.2
192.168.202.2    ?
     |     ^       ttl   1
     v     |       hop 7    ms    0/0 = --%       0 pps    0/0 = --%  0 pps
192.168.202.1
192.168.200.2    ?
     |     ^       ttl   2
     v     |       hop 4    ms    0/0 = --%       0 pps    0/0 = --%  0 pps
192.168.200.1
192.168.10.1     ?
     |      \__    ttl   3
     v         \   hop 0    ms        0          0 pps            0    0 pps
192.168.10.8     192.168.200.1
  Receiver       Query Source
```

从底向上阅读，在例 6-60 中的输出显示了查询的源与响应的目的，在这个例子中均为 192.168.200.1(Sombrero)。注意到 ASCII 符号代表的箭头，显示 Sombrero 向 192.168.10.1(这个例中为它自己的一个接口)发出一个查询。反向通路跟踪到可能连接着源的 Boater，这时这个查询的响应发给 Sombrero。在最左边，ASCII 箭头显示，多播流量从源到目的的通路。在每一跳中，**ttl** 和 **hop** 的统计可能有点误导，**ttl** 实际上显示了从一点到源的跳数，而 **hop** 显示了一跳间的时延(以毫秒为单位)。注意，在响应的目的地址下面有一个往返时延(**rrt**)。对于所有的多播流量和命令中指定的(S,G)对都显示其统计的结果。第一个统计显示了丢弃包的数目与发送出包数目的比较；第二个统计显示，以包/秒为单位的总的流量速率。在例 6-59 中，所有这些统计为 0，因为没有从源到目的的流量，事实上，源和目的并不存在。

在例 6-61 中，已安装了主机，源开始发送流量，组员也已经加入。你可以观察包/秒的

速率与丢弃包的统计。当使用 **mstat** 时，要记住重要的一点，就是只有在路由器之间的时钟是同步时，时延才能有效。

例 6-61　在源与目的间有多播流量后，使用同一个 mstat 命令

```
Sombrero#mstat 192.168.14.35 192.168.10.8 235.100.20.18
Type escape sequence to abort.
Mtrace from 192.168.14.35 to 192.168.10.8 via group 235.100.20.18
From source (?) to destination (?)
Waiting to accumulate statistics......
Results after 10 seconds:

  Source         Response Dest    Packet Statistics For     Only For Traffic
192.168.14.35     192.168.200.1    All Multicast Traffic     From 192.168.14.35
     |         __/  rtt 48   ms   Lost/Sent = Pct  Rate     To 235.100.20.18
     v       /      hop 48   ms   ---------------------     ---------------------
192.168.14.1
192.168.203.1    ?
     |      ^       ttl   0
     v      |       hop 10   ms    0/82 = 0%      8 pps    0/81 = --%  8 pps
192.168.203.2
192.168.202.2    ?
     |      ^       ttl   1
     v      |       hop 6    ms    0/82 = 0%      8 pps    0/81 = 0%  8 pps
192.168.202.1
192.168.200.2    ?
     |      ^       ttl   2
     v      |       hop 4    ms    0/82 = 0%      8 pps    0/81 = 0%  8 pps
192.168.200.1
192.168.10.1     ?
     |       \__    ttl   3
     v          \   hop 0    ms       82          8 pps          81     8 pps
192.168.10.8     192.168.200.1
  Receiver       Query Source

Sombrero#
```

例 6-62 中显示了如果时钟不同步，显示会是什么样子。跟踪信息和包的速率仍为有效的，但单独一跳的时延看上去是荒谬的。在例 6-62 中，你可以看到 Turban 和 Beret 间丢了一个包。这可能是，也可能不是一个问题；唯一弄清的方法是运行几次 mstat 来观察这个包丢失是否是持续的。如果是，需要进一步的查错手段来研究。

例 6-62　路由器的时钟不同步，显示的路由器跳数的延迟没有任何意义

```
Sombrero#mstat 192.168.14.35 192.168.10.8 228.13.20.216
Type escape sequence to abort.
Mtrace from 192.168.14.35 to 192.168.10.8 via group 228.13.20.216
From source (?) to destination (?)
Waiting to accumulate statistics......
Results after 10 seconds:
  Source         Response Dest    Packet Statistics For     Only For Traffic
192.168.14.35     192.168.200.1    All Multicast Traffic     From 192.168.14.35
     |         __/  rtt 44   ms   Lost/Sent = Pct  Rate     To 228.13.20.216
     v       /      hop 44   ms   ---------------------     ---------------------
192.168.14.1
192.168.203.1    ?
     |      ^       ttl   0
     v      |       hop -222 s    0/82 = 0%      8 pps    0/81 = 0%  8 pps
```

```
192.168.203.2
192.168.202.2   ?
     |     ^      ttl   1
     v     |      hop 113  s     1/82 = 1%       8 pps     1/81 = 1%  8 pps
192.168.202.1
192.168.200.2   ?
     |     ^      ttl   2
     v     |      hop 108  s     0/80 = 0%       8 pps     0/80 = 0%  8 pps
192.168.200.1
192.168.10.1    ?
     |      \     ttl   3
     v       \    hop 0    ms        80          8 pps          80    8 pps
192.168.10.8    192.168.200.1
  Receiver      Query Source
```

最后，你可能碰到 **mstat** 显示丢包为负数，就像**-3/85**，的情况。这种负丢包事实上代表多了一个包。换句话讲，收到了其他的包。这表示有环的存在，需要进一步调查。

6.6　展望

你现在已经掌握了配置基本的 IP 多播路由。不过，同单播路由一样，随着多播域的扩展，你会碰到扩展性与控制的问题。第 7 章“大规模 IP 多播路由协议”把范围从域内扩展到域间，介绍了解决大规模多播路由中问题所需的工具与策略。

6.7　配置练习

1．Cisco IOS 启动 IP 多播路由的全局命令是什么？

2．显示在一个支持 PIM 的接口是密集模式、稀疏模式还是密集—稀疏模式的命令。

3．显示静态指定 RP 为 172.18.20.4 的命令。

4．写出把 239.1.2.3 和从 228.1.8.0 到 228.1.8.255 的组地址映射到 RP 192.168.15.5，把组 239.6.7.8 映射到 RP192.168.20.10，把其他的组映射到 192.168.25.1 的必要命令声明。

5．在图 6-11 中所有路由器的接口均运行在密集—稀疏模式下。写出如下相关配置：使

R1 只是 226.13.0.0/24 地址的 RP 的配置；R2 只是 239.0.0.0/8 的地址的 RP；R3 为映射代理；保证这个映射代理对于指定的组，只认为 R1 和 R2 为 RP。RP 所有 Auto-RP 消息的 TTL 值均为 20。

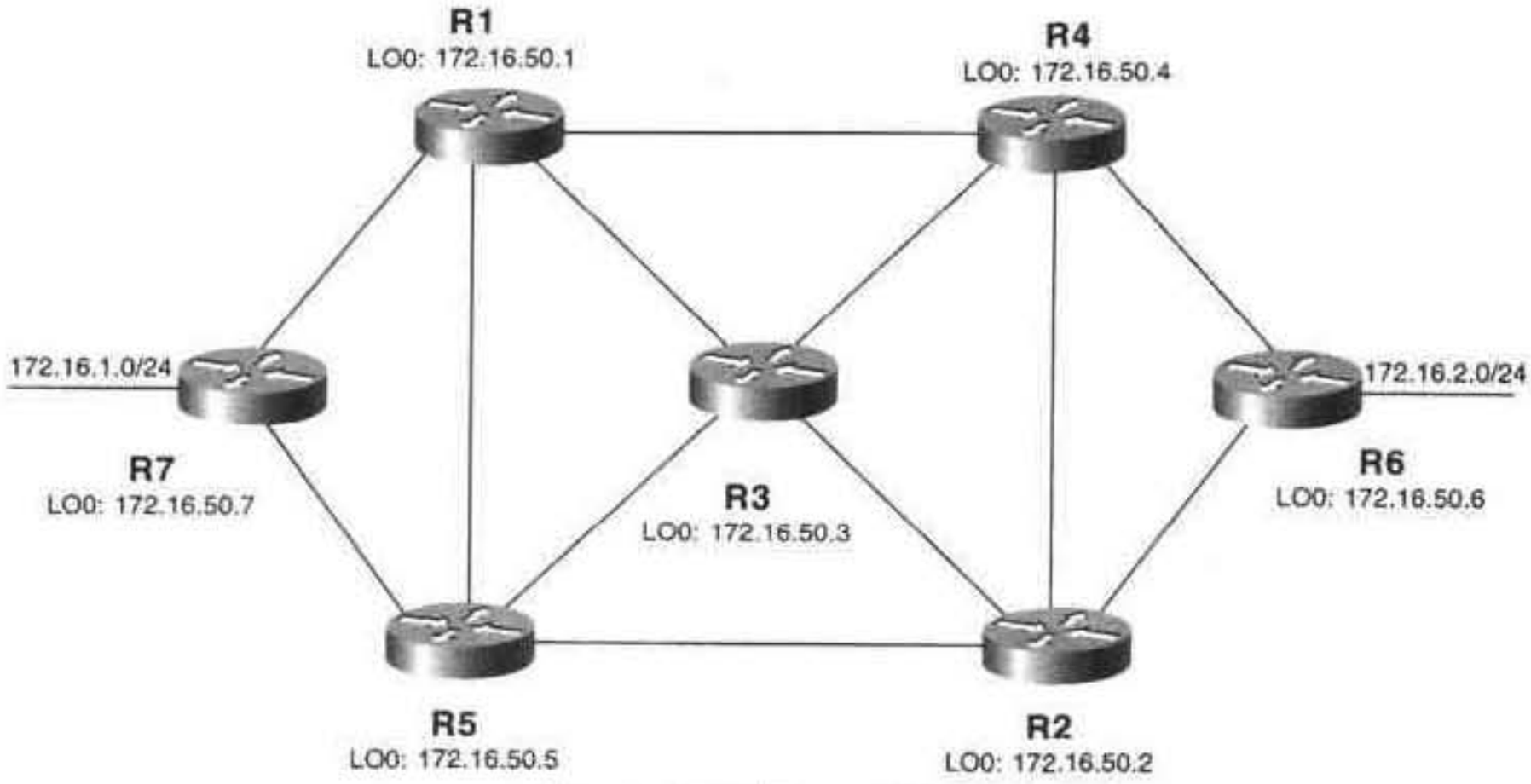


图 6-11　配置练习 5～8 的拓扑

6．已知配置练习 5 中的配置，假设一个源产生组 228.23.14.135 的流量，一个组员请求加入这个组，会有什么情况发生？

7．参考图 6-11，写出如下配置：启动自举协议，使 R1 与 R2 成为配置练习 5 中描述的组地址的 C-RP，R3 成为 BSR，R4 成为备份 BSR。

8．写下关于图 6-11 的配置：使源 172.16.1.75 与组员 172.16.2.100 间实现负载均衡。在隧道接口上使用未用的地址，参考 E0，假设 IGP 可以宣告这些地址。

9．检查案例研究“多播负荷分担”中 Homburg 与 Porkpie 的配置。哪一个路由器在隧道接口上运行 OSPF 的被动模式，为什么？

10. ip pim spt-threshold 100 group-list 25 这一命令的目的是什么？

6.8　排错练习

1. 例 6-63 的输出告诉你什么？

例 6-63　排错练习 1 的输出

```
R1#
Turban#debug ip mpacket
IP multicast packets debugging is on
R1#
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
```

2. 例 6-64 的输出告诉你什么？

例 6-64　排错练习 2 的输出

```
R2#
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
```

3．例 6-65 的输出告诉你什么？

例 6-65　排错练习 3 的输出

```
R3#debug ip mpacket
IP multicast packets debugging is on
R3#
IP: s=172.16.3.50 (Serial0.405) d=224.0.1.40 (Serial0.407) len 52, mforward
IP: s=172.16.3.50 (Ethernet0) d=224.0.1.40 len 62, not RPF interface
IP: s=172.16.3.50 (Ethernet0) d=224.0.1.39 len 62, not RPF interface
IP: s=172.16.3.50 (Serial0.405) d=224.0.1.39 (Serial0.407) len 52, mforward
```

4．在图 6-12 中，哪一个路由器是 PIM 指定路由器？

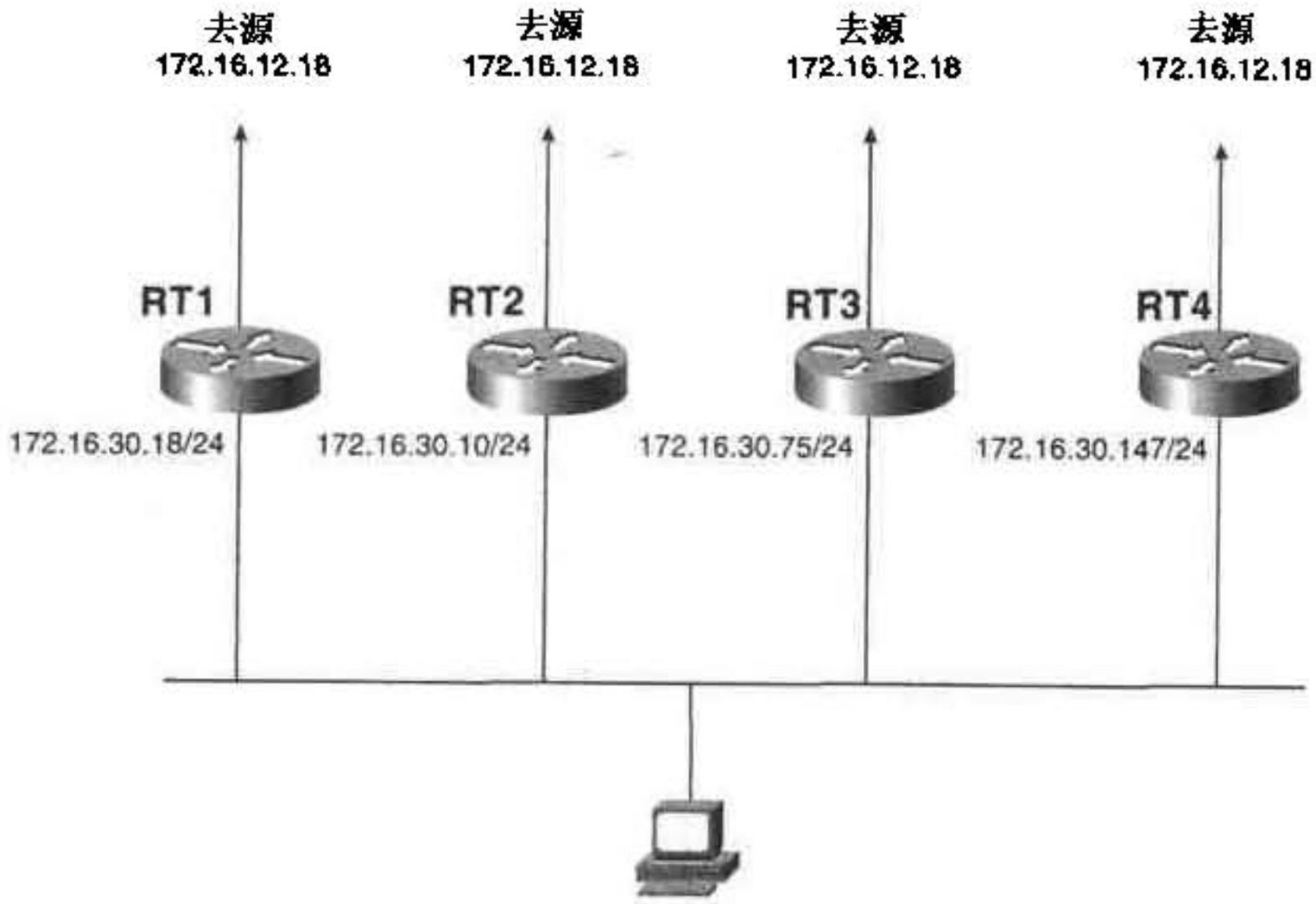


图 6-12　排错练习 4、5、6 的拓扑

5．在图 6-12 中，哪一个路由器向组员发送 IGMPv2 查询消息？

6．表 6-5 显示了图 6-12 中到源 172.16.12.18 的单播路由表。哪一台路由器是 PIM 前转

器？

表 6-5　　图 6-12 中 172.16.12.18 的单播路由

路　由　器	下　　跳	协　　议	量　　度
R1	172.16 50.5	OSPF	35
R2	172.16.51.80	EIGRP	307200
R3	172.16.13.200	EIGRP	2297856
R4	172..16.44.1	OSPF	83

7．例 6-66 中显示了图 6-10 中在一个 PIM 域中的 RPF 跟踪结果，这个域中运行 RIP-2 为作为它的单播 IGP，这个结果是否说明了可能存在的问题？

例 6-66　排错练习 7 中的 **mtrace**

```
Sombrero#mtrace 192.168.14.35 192.168.10.8 235.1.2.3
Type escape sequence to abort.
Mtrace from 192.168.14.35 to 192.168.10.8 via group 235.1.2.3
From source (?) to destination (?)
Querying full reverse path...
 0  192.168.10.8
-1  192.168.10.1 PIM  [192.168.14.0/24]
-2  192.168.200.2 PIM  [192.168.14.0/24]
-3  192.168.201.2 PIM  [192.168.14.0/24]
-4  192.168.204.1 PIM  [192.168.14.0/24]
-5  192.168.14.35
Sombrero#
```

第 7 章　大范围 IP 多播路由

前面两章介绍了 IP 多播路由协议的当前状态和在 Cisco IOS 软件中配置多播路由的基础。正如单播路由一样，当多播域增长时你必须采取措施来维护稳定性、可扩展性和可管理性。本章介绍了一些技术和协议来帮助达到上述目标。

- 多播范围限制——本节检验了限制多播流量范围的原因，以及限制多播范围的两种实现方式：TTL 限制和管理限制。
- 案例学习：多播穿过非多播域——该案例介绍了将 IP 多播流量穿过不支持 IP 多播的路由器的技术。
- 连接到 DVMRP 网络——本节展示了将 Cisco 路由器连接到 DVMRP 网络的方法，以及 Cisco 路由器 IOS 软件对 DVMRP 支持的部分和不支持的部分。
- AS 间多播——本节介绍了在 AS 间路由 IP 多播流量产生的问题并讨论了作为解决上述问题方案的 MBGP 和 MSDP 的运行。
- 案例学习：配置 MBGP——该案例学习展示了多协议 BGP 的配置。
- 案例学习：配置 MSDP——该案例学习展示了 MSDP 的配置。
- 案例学习：MSDP 全连接组——该案例学习展示了 MSDP 全连接组的配置。
- 案例学习：泛播 RP——该案例学习展示了 Anycast RP 的配置。
- 案例学习：MSDP 缺省对等体——该案例学习展示了 MSDP 缺省对等体的配置。

7.1　多播范围控制

构造大规模多播域首要考虑的问题是控制域的范围。你在第 5 章中已经读到了相关讨论，知道有两种方法可以控制多播域：

- TTL 范围控制
- 管理范围控制

使用 TTL 控制范围时，多播包的 TTL 值应当设置成：在 TTL 减为 0 被丢弃以前包可以传输的一定距离。你可以通过在边缘接口上配置 **ip multicast ttl-threshold** 命令通过 TTL 粗略地增加控制粒度。例如接口可以配制成 **ip multicast ttl-threshold 5**。只有 TTL 值大于 5 的包才能从该接口转发。任何 TTL 值是 5 或更小值的包都被丢弃。表 7-1 显示了 TTL 范围限制值的举例。表中的值是表 5-6 的重复，是 MBone 建议的一组 TTL 值。

在第 6 章中你已经看到了几个命令，例如对候选 RP 使能 Auto-RP 的命令和映射代理，通过设置协议消息的 TTL 值来作 TTL 范围控制的命令。你在这一章会碰到相同选项的更多命令。然而正如你在第 5 中所看到的：TTL 范围控制缺乏灵活性——接口上的 TTL 边界应用在所有多播包。对绝对多播边界来说没有问题，但有时你想要阻塞一些包但让另一些包通过。

表 7-1　　MBone TTL 门限

TTL 值	限　　制
0	限制在同一台主机
1	限制在同一子网
15	限制在同一场地
63	限制在同一区域
127	全球范围
191	全球范围，带宽受限
255	不受限

出于上述目的，管理范围限制提供了更多的灵活性。管理范围限制仅仅是一个程序，该程序将 224.0.0.0-239.255.255.255 的多播地址按地址块分配给不同的范围。可以通过过滤不同的地址块来建立不同的域边界。管理范围限制是 RFC 2365 的主要内容，表 7-2 显示了该 RFC 建议的部分内容。你已经看到了本链路范围 224.0.0.0/24 的使用。在上述多播地址范围内的地址例如 IGMP(224.0.0.1 和 224.0.0.2)、OSPF(224.0.0.5 和 224.0.0.6)、EIGRP(224.0.0.10)、PIM(224.0.0.13)，只能限制在产生上述包的链路范围内，不能由路由器转发。

表 7-2　　RFC2365 管理划分

前　　缀	范　　围
224.0.0.0/24	本链路范围
224.0.1.0-238.255.255.255	全球范围
239.0.0.0/10	未分配
239.64.0.0/10	未分配
239.128.0.0/10	未分配
239.192.0.0/14	本组织范围
239.255.0.0/16	未分配

在接口上使用 **ip multicast boundary** 命令来建立管理边界。该命令参考 IP 访问列表，由访问列表来规定接口上允许或拒绝的组地址范围。例 7-1 显示了一个范例。

例 7-1　在接口上增加 ip multicast boundary 命令建立管理边界

```
interface Ethernet0
 ip address 10.1.2.3 255.255.255.0
  ip multicast boundary 10
!
interface Ethernet1
 ip address 10.83.15.5 255.255.255.0
 ip multicast boundary 20
!
```

```
access-list 10 deny   239.192.0.0 0.3.255.255
access-list 10 permit 224.0.0.0 15.255.255.255
access-list 20 permit 239.135.0.0 0.0.255.255
access-list 20 deny   224.0.0.0 15.255.255.255
```

接口 E0 标记了边界，如表 7-2 所定义本组织范围的包被阻塞，全球范围的包可以通过。接口 E1 的边界允许目的地 239.135.0.0/16 范围内的包，拒绝其他多播包。上述地址范围落在表 7-2 中未指定范围内，由本地网络管理员赋予特定的含义。

7.2　案例学习：多播穿过非多播域

你将遇到的一个挑战是穿过不支持多播的域将多个多播域连接起来。上述情况在大范围路由域中仅有某些区域需要多播时很常见。你不希望仅为使相对少数的多播路由器的连接而将单播域中所有路由器上使能多播。第 2 个常见的例子是通过仅支持单播的互联网连接多个多播域。

在图 7-1 中，两个 PIM 域由单播 IP 域所分隔。单播域可能是企业网的骨干网，也可能是互联网本身。问题的重点是两个多播域必须通过单播域连接。解决方案很简单：在两个路由器间建立一个隧道来传输 PIM 流量。

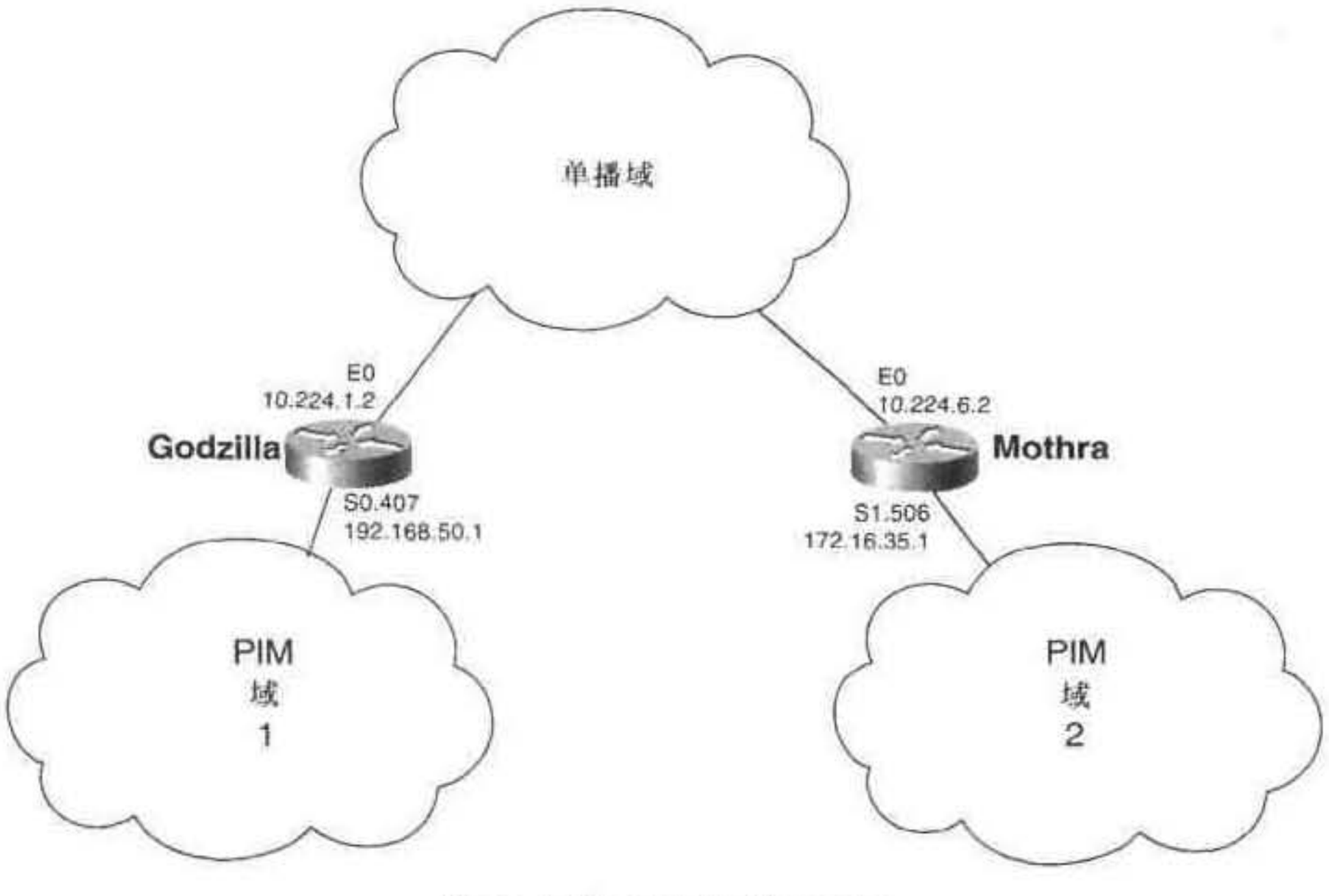


图 7-1　由单播 IP 域分割的 PIM 域

例 7-2 显示了图 7-1 中两个路由器的隧道配置。

你已经在第 6 章看到等开销路径本地共享隧道的使用。这里的配置与之类似。隧道源是每个路由器的以太网接口，但是 PIM 没有配置在物理接口上，仅配置在隧道上。使用缺省的隧道模式：GRE 封装。在 TU0 上的 OSPF 配置成被动运行，确保没有单播流

量穿过隧道。最后配置了静态多播路由，从对端域引用所有的源地址且显示上游接口是TU0。回忆第6章为防止RPF失败，该路由是必须的。如果没有该路由，RPF检查将使用OSPF路由，决定上游接口是路由器的E0接口。结果是TU0收到所有包的RPF检查都将失败。

例7-2　在穿过单播域的多播域间提供连接性的Godzilla和Mothra的配置

Godzilla

```
interface Tunnel0
 ip unnumbered Ethernet0
 ip pim sparse-dense-mode
 tunnel source Ethernet0
 tunnel destination 10.224.6.2
!
interface Ethernet0
 ip address 10.224.1.2 255.255.255.0
!
interface Serial0.407 point-to-point
 description PVC to R7
 ip address 192.168.50.1 255.255.255.0
 ip pim sparse-dense-mode
 frame-relay interface-dlci 407
!
router ospf 1
 passive-interface Tunnel0
 network 10.0.0.0 0.255.255.255 area 0
 network 192.168.0.0 0.0.255.255 area 0
!
ip mroute 172.16.0.0 255.255.0.0 Tunnel0
```

Mothra

```
interface Tunnel0
 ip unnumbered Ethernet0
 ip pim sparse-dense-mode
 tunnel source Ethernet0
 tunnel destination 10.224.1.2
!
interface Ethernet0
 ip address 10.224.6.2 255.255.255.0
!
interface Serial1.506 point-to-point
 description PVC to R6
 ip address 172.16.35.1 255.255.255.0
 ip pim sparse-dense-mode
 frame-relay interface-dlci 506
!
router ospf 1
 passive-interface Tunnel0
 network 0.0.0.0 255.255.255.255 area 0
!
ip mroute 192.168.0.0 255.255.0.0 Tunnel0
```

注：　如果DVMRP路由器不支持GRE封装，你可以使用IP-in-IP。

例7-3显示了配置的结果。

例 7-3　穿过 GRE 隧道建立的 PIM 邻接关系

```
Godzilla#show ip pim neighbor
PIM Neighbor Table
Neighbor Address  Interface          Uptime    Expires   Ver  Mode
192.168.50.2      Serial0.407        01:08:51  00:01:27  v2
172.16.35.1       Tunnel0            01:03:31  00:01:16  v2
Godzilla#
```

```
Mothra#show ip pim neighbor
PIM Neighbor Table
Neighbor Address  Interface          Uptime    Expires   Ver  Mode
172.16.35.2       Serial1.506        01:10:06  00:01:42  v2
192.168.50.1      Tunnel0            01:04:33  00:01:15  v2
Mothra#
```

7.3　连接到 DVMRP 网络

你可能偶尔需要将你的 PIM 路由器连接到 DVMRP 路由器。这并不是大范围多播网络的问题——只运行 DVMRP 的路由器可能在任何规模的互联网络中遇到。然而最可能出现的环境是你想连接到 MBone。

当你将 Cisco 路由器接口配置成运行 PIM 时，该接口开始收听 DVMRP Probe 消息。当收到 Probe 消息时，如例 7-4 中所示，Cisco IOS 软件自动在接口上使能 DVMRP。不需要特别的配置。PIM 路由器使用 DVMRP 的 Report 消息向 DVMRP 邻居宣告。从邻居处学到的 Report 消息包存在如例 7-5 所示单独的 DVMRP 路由表中，但是 Cisco 路由器上仍然由 PIM 作多播转发决定。DVMRP 的 Graft 消息被正常接收发送，只有 Prunes 和 Probe 的处理才使 Cisco IOS 软件对 DVMRP 的实现区别于完全实现。

例 7-4　该路由器在端口 E0 上从邻居 10.224.1.1 接收到 DVMRP Probe 消息

```
Godzilla#debug ip dvmrp detail
DVMRP debugging is on
Godzilla#
DVMRP: Received Probe on Ethernet0 from 10.224.1.1
DVMRP: Aging routes, 0 entries expired
DVMRP: Received Probe on Ethernet0 from 10.224.1.1
DVMRP: Aging routes, 0 entries expired
DVMRP: Received Probe on Ethernet0 from 10.224.1.1
DVMRP: Aging routes, 0 entries expired
```

例 7-5　**show ip dvmrp route** 显示的 DVMRP 特定路由信息

```
Godzilla#show ip dvmrp route
DVMRP Routing Table - 7 entries
10.224.2.0/24 [0/1] uptime 00:04:21, expires 00:02:38
    via 10.224.1.1, Ethernet0, [version mrouted 3.255] [flags: GPM]
10.224.3.0/24 [0/1] uptime 00:04:21, expires 00:02:38
    via 10.224.1.1, Ethernet0, [version mrouted 3.255] [flags: GPM]
10.224.4.0/24 [0/1] uptime 00:04:21, expires 00:02:38
    via 10.224.1.1, Ethernet0, [version mrouted 3.255] [flags: GPM]
10.224.5.0/24 [0/1] uptime 00:04:21, expires 00:02:38
```

```
    via 10.224.1.1, Ethernet0, [version mrouted 3.255] [flags: GPM]
10.224.6.0/24 [0/1] uptime 00:04:21, expires 00:02:38
    via 10.224.1.1, Ethernet0, [version mrouted 3.255] [flags: GPM]
172.16.70.0/24 [0/1] uptime 00:04:21, expires 00:02:38
    via 10.224.1.1, Ethernet0, [version mrouted 3.255] [flags: GPM]
192.168.50.0/24 [0/1] uptime 00:04:21, expires 00:02:38
    via 10.224.1.1, Ethernet0, [version mrouted 3.255] [flags: GPM]
```

DVMRP 的完全实现与基于 Cisco IOS 软件的 DVMRP 实现第一个不同之处在于对 Probe 的处理。正如前面提到的那样，Cisco 路由器依靠接收 Probe 消息来发现 DVMRP 邻居。如图 7-2 所示，假设 DVMRP 邻居位于多重访问的网络上且网络上还有其他 Cisco 路由器，如果其中一个 Cisco 路由器发送 Probe 消息，相邻的 Cisco 路由器会错误地认为上述 Cisco 路由器是 DVMRP 路由器而不是 PIM 路由器。所以 Cisco 路由器只接收 Probe 消息，并不产生该消息。如图 7-2 所示，如果 Cisco 路由器产生 DVMRP Probe 消息，下面的路由器会错误地认为发起者是 DVMRP 邻居；所以 Cisco 路由器不产生 Probe。

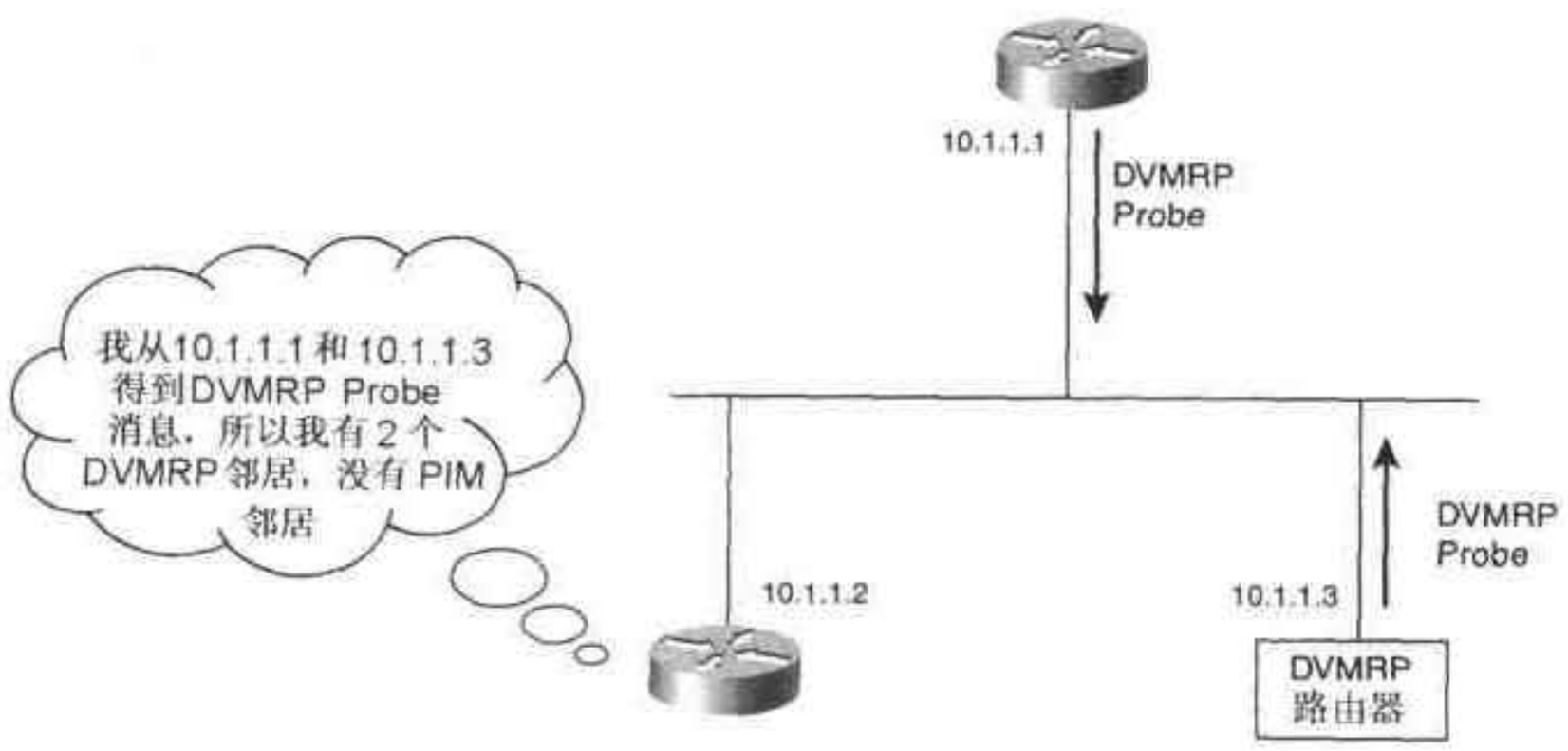


图 7-2　Cisco 路由器不发送 DVMRP Probe 消息

第 2 个不同之处是对 Prune 消息的处理。回忆第 5 章中 DVMRP 的讨论，DVMRP 路由器需要维护每个下游路由器的状态。如果下游邻居发送 Prune 消息，只剪除邻居的状态。除非所有的邻居都发送 DVMRP Prune，流量依然向端口发送。该机制是为解决多重访问网络上存在多个下游邻居时，防止一个邻居发送的 Prune 消息影响其他邻居。

注：回忆第 5 章 PIN-DM 使用 Prune 覆盖机制来解决该问题，不需要维护邻居的状态。

但是 Cisco 路由器不维护邻居的状态。所以为防止下游路由器发送的 Prune 消息影响其他下游邻居，Cisco 路由器忽略在多重访问接口上收到的 DVMRP Prune 消息。在点到点链路上由于只可能有一个下游邻居，所以 Prune 消息正常接收且处理。Cisco 路由器在存在 DVMRP 邻居的多重访问接口或点到点接口上正常发送 Prune 消息。

对你来说存在的问题已经很明显了。当 DVMRP 路由器连接到多重访问链路时，上游的 Cisco 路由器无法剪除。Cisco 路由器会向 DVMRP 域发送不需要的多播流量。解决的方法是使用隧道。

在图 7-3 中 Cisco 路由器通过多重访问网络连接到两个 DVMRP 路由器。通过建立到两个 DVMRP 路由器的隧道，Cisco 软件认为 DVMRP 邻居是在点到点链路上而不是在多重访问链路上。所以 Cisco 路由器可以正常剪除。

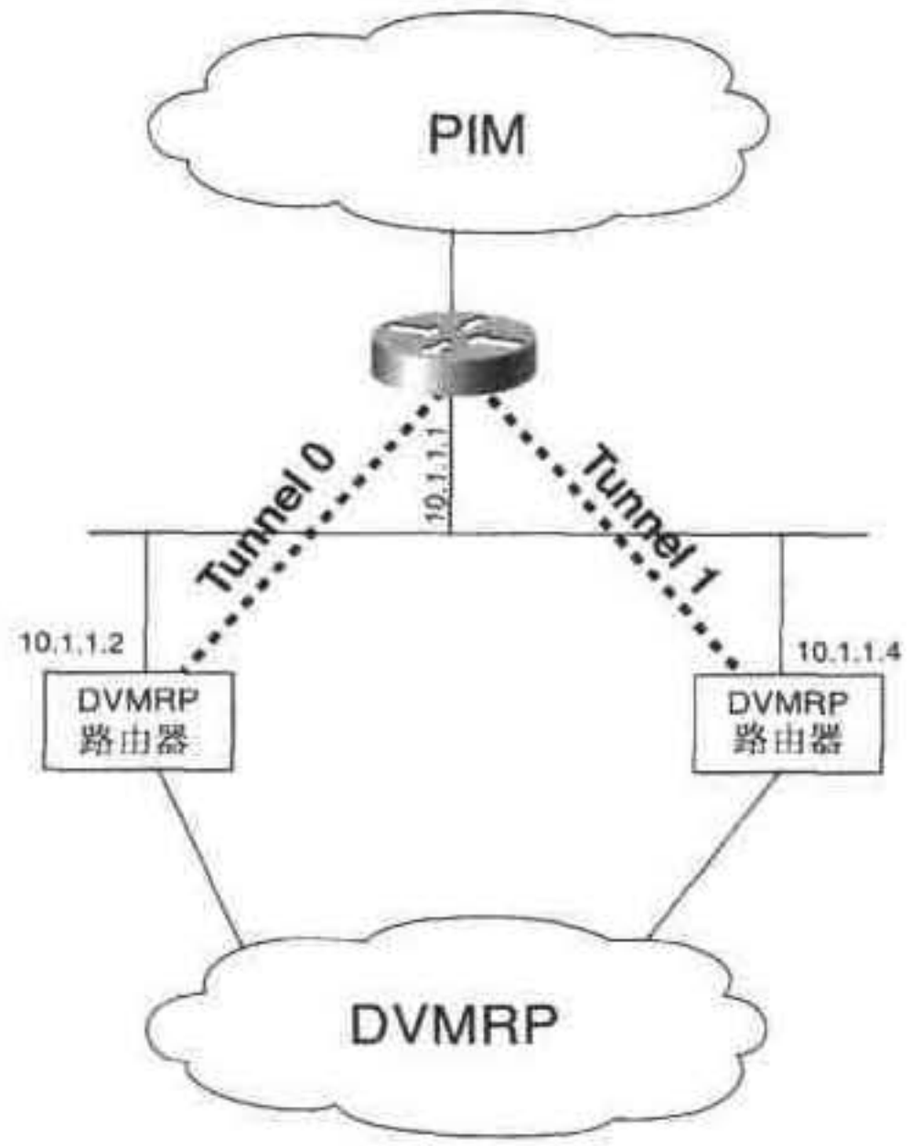


图 7-3　剪除使 DVMRP 剪除正常工作

例 7-6 显示了图 7-3 中 Cisco 路由器的配置。

例 7-6　图 7-3 中 Cisco 路由器通过点到点链路接受剪除的配置

```
interface Tunnel0
 ip unnumbered Ethernet0
 ip pim sparse-dense-mode
 tunnel source Ethernet0
 tunnel destination 10.1.1.2
 tunnel mode dvmrp
!
interface Tunnel1
 no ip address
 ip pim sparse-dense-mode
 tunnel source Ethernet0
 tunnel destination 10.1.1.4
 tunnel mode dvmrp
!
interface Ethernet0
 ip address 10.1.1.1 255.255.255.0
```

你可以看到与以前隧道配置唯一明显的不同是隧道模式设置成 DVMRP 而不是缺省的 GRE。和以前的隧道配置一样，PIM 配置在隧道上而不是在物理接口上。如果多重访问网络

上还有其他的 Cisco PIM 路由器，可以直接在以太网接口上配置 PIM，这样路由器在隧道上与 DVMRP 路由器连接，通过以太网与 PIM 路由器连接。

> 注：　谨记如果多播源通过 DVMRP 路由器可达，你必须配置静态多播路由防止 RPF 失败。

7.4　AS 间多播

多播路由协议面临的一个挑战(该问题对单播路由也存在)是如何对一组需要分发包的主机有效地扩展。你可以看到密集模式协议例如 PIM-DM 和 DVMRP 扩展性不好；按照定义，这些协议假设多播域中多数节点是组成员。PIM-SM 是稀疏模式协议，由于假设多播域中多数主机都不是组成员，所以扩展性较好。然而无论是稀疏模式还是密集模式，都假设在单一域范围内。换句话说，你所看到的 IP 多播路由协议都可以认为是多播 IGP。

然而如何才能在穿过 AS 边界分发多播包的同时还能维护每个 AS 的自治性？

PIM-SM Internet 草案试图通过定义 PIM 多播边缘路由器(PMBR)来解决上述问题。如图 7-4 所示，PMBR 位于 PIM 域的边缘为其中所有的 RP 建立分枝。每一个分枝由(*,*,RP)表示，通配符表示映射在该 RP 上所有的源和组地址。当 RP 从源收到流量时，将流量转发到 PMBR，由 PMBR 将流量转发到其他域。当邻接域因不需要流量，便向 PMBR 发送剪除，于是 PMBR 向 RP 发送剪除。

PMBR 关键的缺点是洪泛和剪除行为。事实上 PMBR 主要是为将 PIM-SM 域连接到 DVMRP 域而设计的。由于从上述内容遗留的扩展性问题，Cisco IOS 软件不支持 PMBR。如图 7-4 所示 PIM 多播边界路由器建立到每个 RP 的分支称为(*,*,RP)。RP 将来自源的流量顺上述分枝转发到 PMBR。

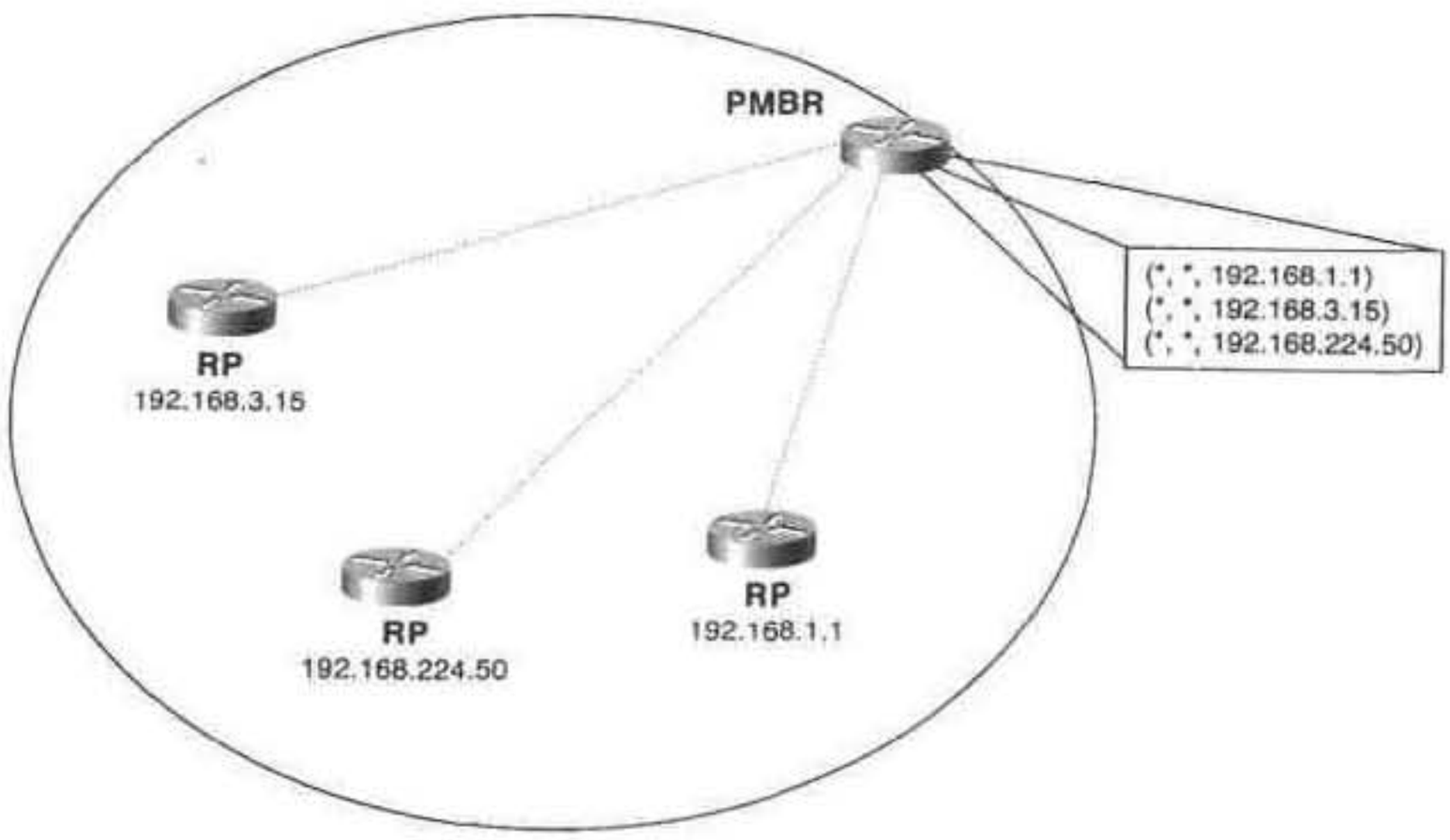


图 7-4　PIM 多播边界路由器建立到每个 RP 的分支

首先承认 PIM-SM 是 IP 多播路由协议的事实标准，如何在自治域间路由多播的问题就简化成如何在 PIM-SM 域间路由的问题。有两个问题必须解决：

- 当源在一个域而组成员在另一个域时，RPF 过程必须保持有效。
- 为保持自治性，域不能信赖另一个域中的 RP。

由于 PIM-SM 是协议无关的，所以第一个问题看上去容易解决。就像 PIM 使用单播 IGP 来决定域中的 RPF 接口，PIM 可以使用 BGP 路由来决定到达其他自治系统源的 RPF。在域间转发流量时，你可能希望单播流量与多播流量使用不同的链路如图 7-5 所示。如果多播包到达链路 A，BGP 指示到达该包源节点的单播路由是链路 B，则 RPF 检查失败。静态多播路由可以用来防止 RPF 问题，但是在大范围网络中显然不实用。所以必须扩展 BGP，使 BGP 可以指示宣告的路由是用作单播路由或多播 RPF 检查，还是两者皆可。

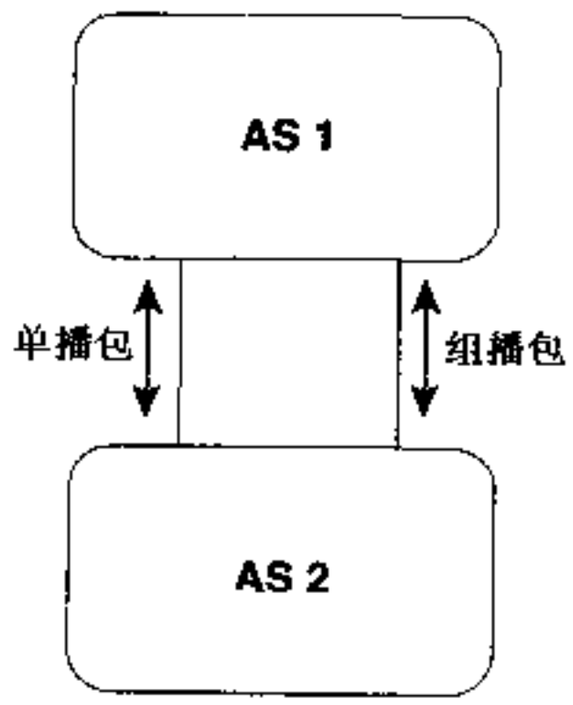


图 7-5　AS 间流量工程指出，多播流量与单播流量应使用不同链路

BGP 协议已被扩展，PIM 可以使用扩展的 BGP 带来的好处。BGP 的扩展版本称为多协议 BGP(MBGP)，由 RFC 2283 描述。尽管 BGP 扩展本意是为了携带其他协议例如 IPv6 和 IPX 等的可达性消息，广泛应用的 MBGP 也用作宣告多播源，结果是 MBGP 中的 M 更多地用作代表多播而不是多协议。

MBGP 最常见的应用是在 NAP 点同一交换多播信息的服务提供商组成的对等实体间。如图 7-6 所示，多个自治系统可以为单播流量建立对等关系，但必须为多播流量共享一个对等点。因为一些前缀必须在单播和多播 NAP 点上宣告，所以 MBGP 用作从单播路径上区分多播 RPF 路径。

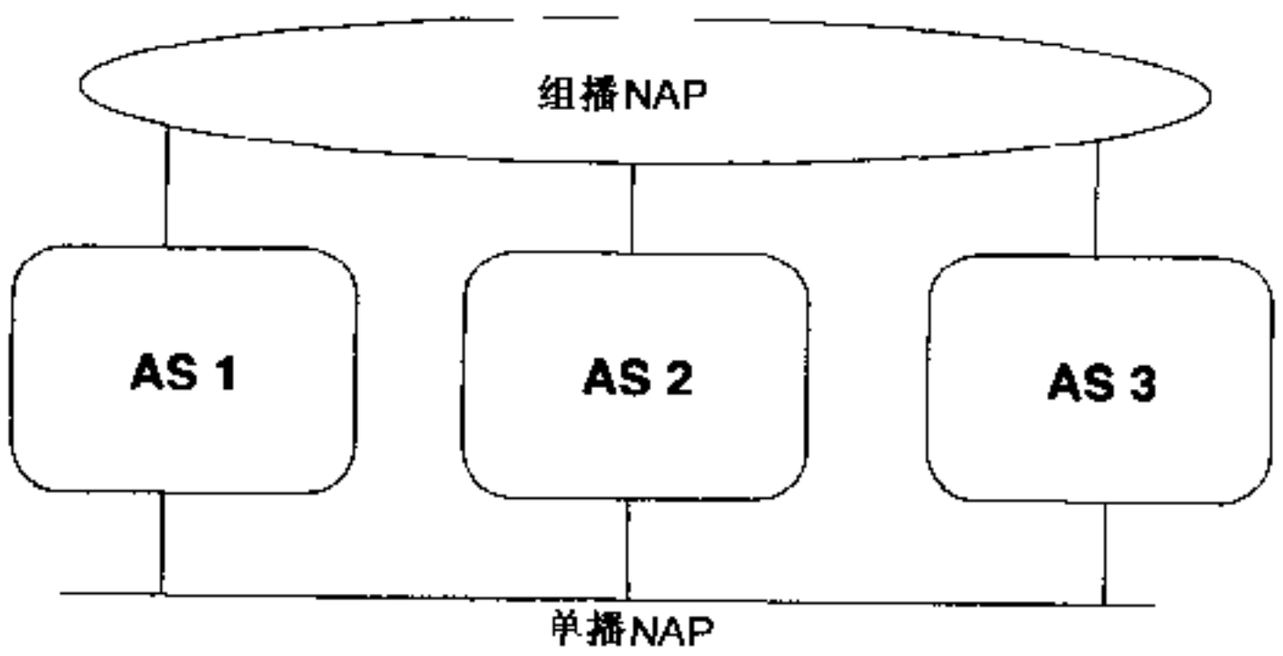


图 7-6　需要单播和多播对等点时使用 MBGP

AS 间 PIM 的第二个问题(为保持自治，域不能依赖域另一个域中的 RP)是因为 AS 不希望依赖于无法控制的 RP。如果每个 AS 设置自己的 RP，必须有一个协议来使多个 RP 穿过 AS 边界共享源信息，发现其他 RP 已知的源信息，如图 7-7 所示。多播源发现协议在 RP 间运行，允许每个 RP 发现其他 RP 已知的多播源。该协议是多播源发现协议(MSDP)。

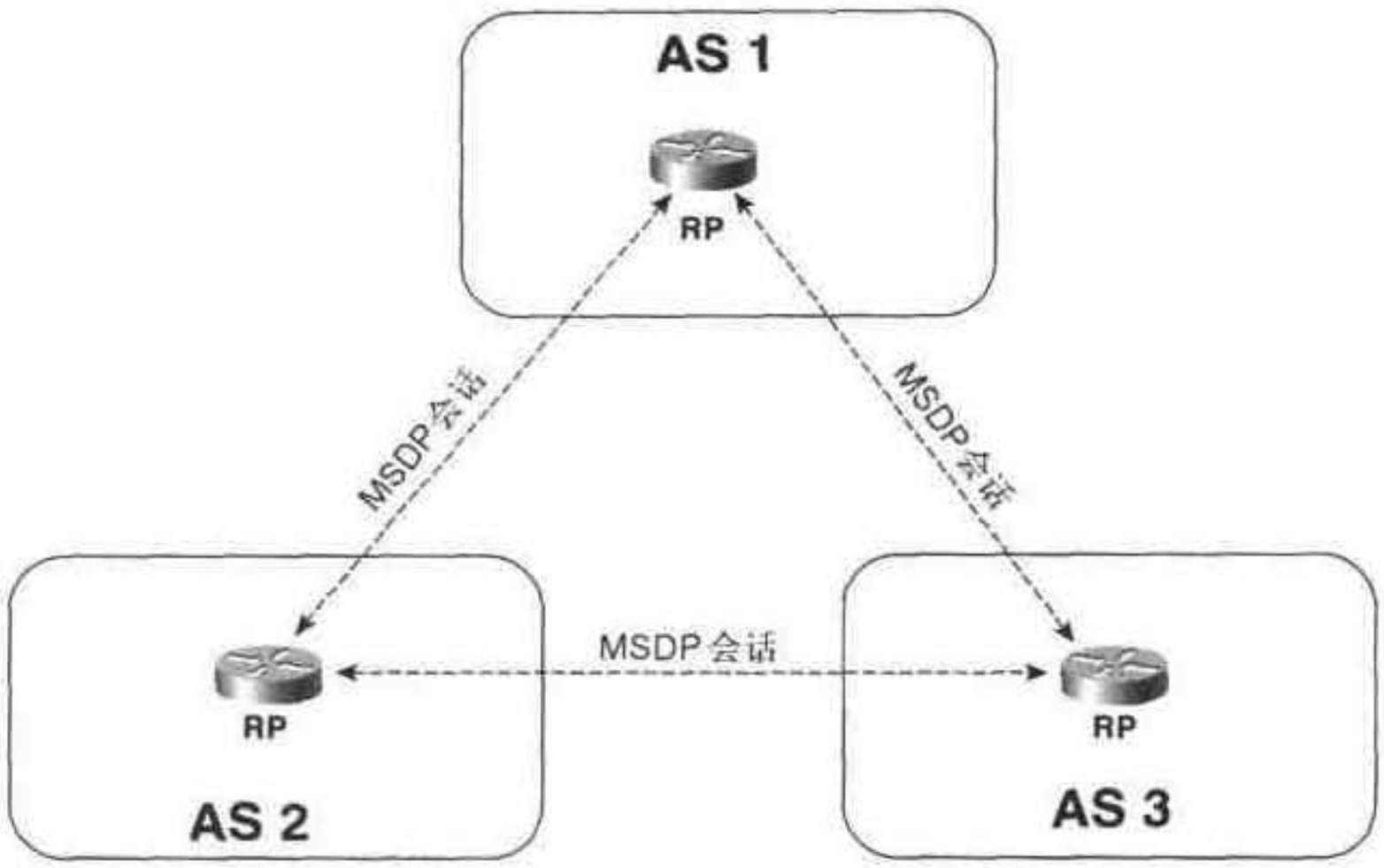


图 7-7　每个 RP 发现其他 RP 已知的多播源

下面两节描述 MBGP 的扩展和 MSDP 的运行。

7.4.1　BGP 的多协议扩展(MBGP)

RFC2283 通过定义如下两个属性，将 BGP 扩展为支持多协议：

- 多协议可达 NLRI，或 MP_REACH_NLRI(类型 14)
- 多协议不可达 NLRI，或 MP_UNREACH_NLRI(类型 15)

注：　BGP 属性类型编码项系列表参见第 2 章表 2-7。

上述两个属性都是任选非可透传的。回忆第 2 章“边界网关协议 4 简介”中内容，任选非可透传表示 BGP 运行并不是强制支持上述属性且不支持上述属性的路由器不需要将该属性传输到对端。

注：　MP_REACH_NLRI 的完成格式比这里指示的复杂，有一些字段与 IP 多播无关。详细的描述参见 RFC 2283。

MP_REACH_NLRI 属性用作宣告可达的路由，MP_UNREACH_NLRI 属性用作撤销可达路由。属性中包含的网络层可达消息(NLRI)明确协议的目的地消息。当 MBGP 用作 IP 多播时，NLRI 是一个或多个 IPv4 前缀描述的多播源。记住 PIM 路由器并不使用上述信息转发包，只用作对特定多播源确定 RPF 接口。上面两个新的属性提供了通知 BGP 对端特定的前缀是用作单播路由、多播 RPF 或两者皆可的能力。

MP_REACH_NLRI 包含一个或多个[地址族信息，下一跳信息，NLRI]三元组。

MP_UNREACH_NLRI 包含一个或多个[地址组信息，不可用路由长度，撤销的路由]三元组。

地址组信息包含地址组标识符(AFI)和一个子 AFI(Sub-AFI)。IPv4 的 AFI 为 1，IP 多播的 AFI 也是 1。子 AFI 用作描述 NLRI 是用作单播路由、多播 RPF 消息还是同时用作两者。如表 7-3 所示。

表 7-3　　子地址族标识符

子 地 址 族	描　　述
1	只用于是单播路由信息
2	只用于多播 RPF 信息
3	前缀可以用在单播路由信息与多播 RPF 信息

7.4.2　多播源发现协议(MSDP)运行

正如名字所描述，MSDP 的作用是在其他 PIM 域中发现多播源。运行该协议的好处是与其他域中的 RP 交换多播源信息的 RP 是属于你自己的；你的组成员不需要直接依赖于另一个域的 RP。

> 注：　你可以在下面几节的案例学习中看到在一个域中 MSPD 也能有效共享多播源信息。

MSDP 使用 TCP(端口 639)建立对等连接。与 BGP 一样，MSDP 使用点到点 TCP 连接所以必须明确配置。当 PIM DR 如图 7-8 所示在 RP 注册源时，RP 向所有的 MSDP 对等体发送源激活(SA)消息。

SA 包含：

- 多播源的地址
- 发送组地址
- 源 RP 的 IP 地址

接收到 SA 的每个 MSDP 对等体将 SA 洪泛到相对于源的下游。在某些例子中，例如图 7-8 中的 AS6 和 AS7，RP 可能从多个 MSDP 对等体收到一个 SA 的多个拷贝。为防止环回，RP 查询 BGP 的下一跳数据库来决定 SA 源的下一跳。如果同时配置了 MBGP 和单播 BGP，首先检查 MBGP 然后单播 BGP。如果下一跳邻居是 RPF 对等体且已从其他接口 RPF 对等体收到发起者的 SA，则该 SA 丢弃。所以 SA 洪泛过程称为对等 RPF 洪泛。由于对等 RPF 洪泛机制，BGP 和 MBGP 必须和 MSDP 联合使用。

RP 收到 SA 以后，将检查域内是否有该 SA 组的成员。上述检查是通过察看是否有接口属于该组的(*,G)实现。如果没有组成员，则 RP 什么也不做。如果存在组成员，RP 向多播域内发送加入消息。这样多播树的一个分支就穿过 AS 边界构造到 RP。当多播包到达 RP 以后，就从 RP 域中内部共享树转发到组成员。成员的 DR 可以选择是否使用标准的 PIM-SM 程序加入 RPT 树。

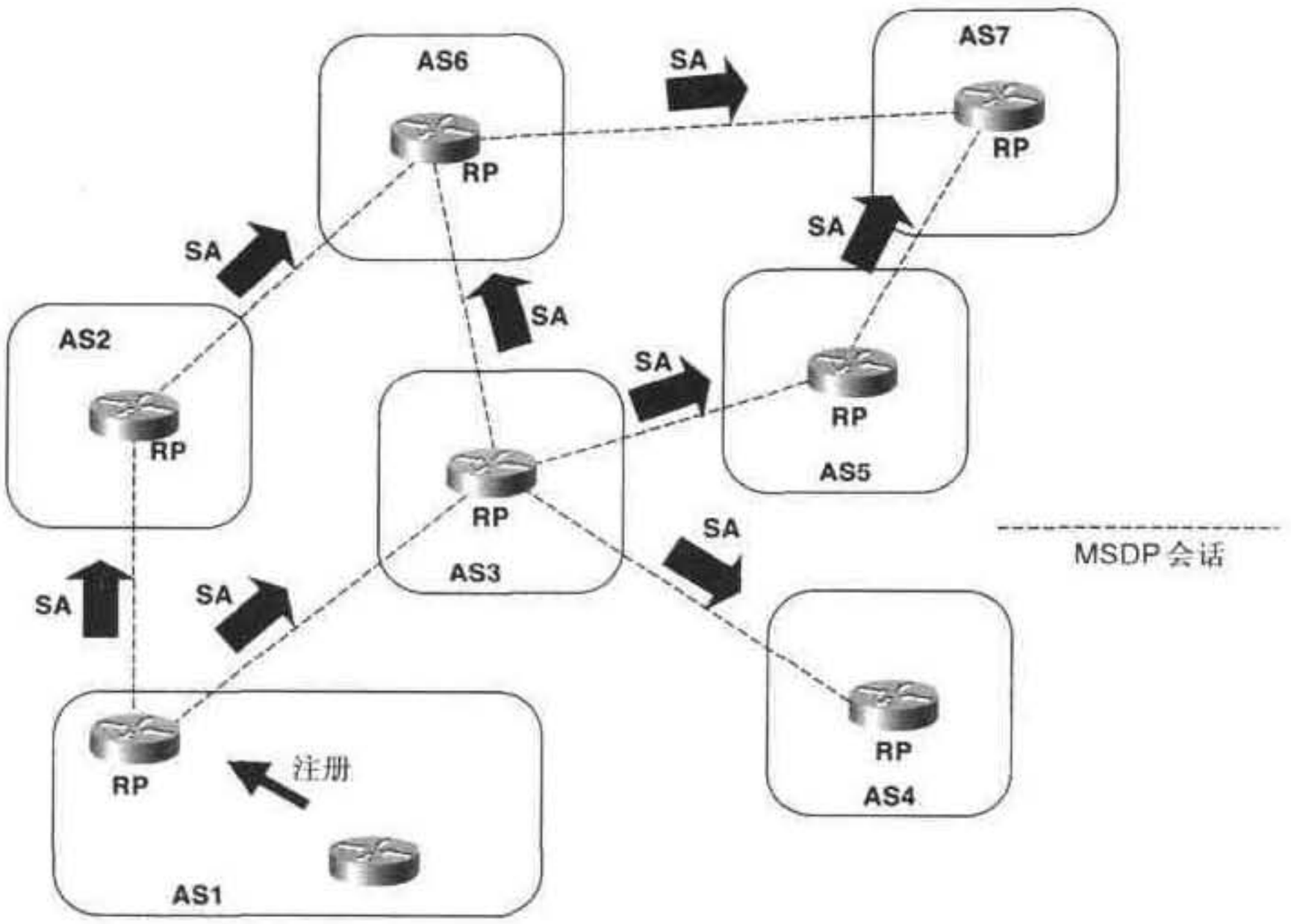


图 7-8　RP 使用源激活消息将多播源广告到所有 MSDP 邻居

在多播源向组成员连续发送数据时，源 RP 以 60 秒周期性持续向(S,G)组发送 SA。当 RP 收到 SA 以后，可以选择是否缓存该消息。例如假设 RP 从源 RP 10.5.4.3 收到(172.16.5.4,228.1.2.3)的 SA。RP 查询多播路由表后发现没有组 228.1.2.3 的激活成员，则将 SA 消息转发到下游的 10.5.4.3。如果域中的主机对 RP 发送组 228.1.2.3 发送加入消息，RP 将通往主机的接口加入到(*,228.1.2.3)的出口列表中。由于没有缓存上一个 SA，RP 不知道多播源。所以 RP 初始化到多播源的加入消息以前必须等待接收下一个 SA 消息。

另一方面如果 RP 缓存 SA，则路由器内存在表项(172.16.5.4,228.1.2.3)。在收到主机请求后能立刻加入到多播源树。这里的选择是是否需要缓存 SA 消息的内存与减少加入时延的折衷。如果 RP 属于很大的 MSDP 连接，并且 SA 数量较多，则需要内存的数量非常巨大。

缺省条件下，Cisco 路由器不缓存 SA。你可以使用 **ip msdp cache-sa-state** 命令使能缓存。为缓解可能的内存压力，你可以将上述命令连接到一个规定(S,G)组的访问列表。

如果 RP 拥有一个缓存 SA 的 MSDP 对等体，你可以通过使用 SA Request 和 SA Response 消息的办法在不打开 SA 缓存的条件下减少加入时延。当主机向一个特定组发送加入消息时，RP 向能够缓存的对等体发送 SA Request 消息。发送请求的 RP 使用 SA Response 中的消息，但是不向其他对等体转发。如果非缓存 RP 收到 SA Request 则会向发送者送出错消息。

为使能路由器发送 SA Request 消息，使用 **ip msdp sa-request** 命令来指定缓存对等体的名字或 IP 地址。你可以多次使用上述命令来指定多个缓存对等体。

7.4.3　MSDP 消息格式

MSDP 消息携带在 TCP 会话中。当两个路由器配置成 MSDP 对等体后，较高 IP 地址的路由器监听 639 端口，较低 IP 地址的路由器向 639 端口发起连接。

MSDP 消息使用 TLV(类型/长度/值)形式，可以是表 7-4 所示 5 种形式之一。下面几节详细介绍每种消息类型的格式。

表 7-4　MSDP 消息类型

类　型	消　息
1	Source Active
2	Source Active Request
3	Source Active Response
4	Keepalive
5	Notification

1．Source Active TLV

当 MSDP RP 从 IP 多播源收到 PIM 注册消息后，向对端发送 Source Active 消息。图 7-9 显示了 MSDP Source Active TLV 的格式。在多播源激活期间，SA 消息每 60 秒周期性发送。

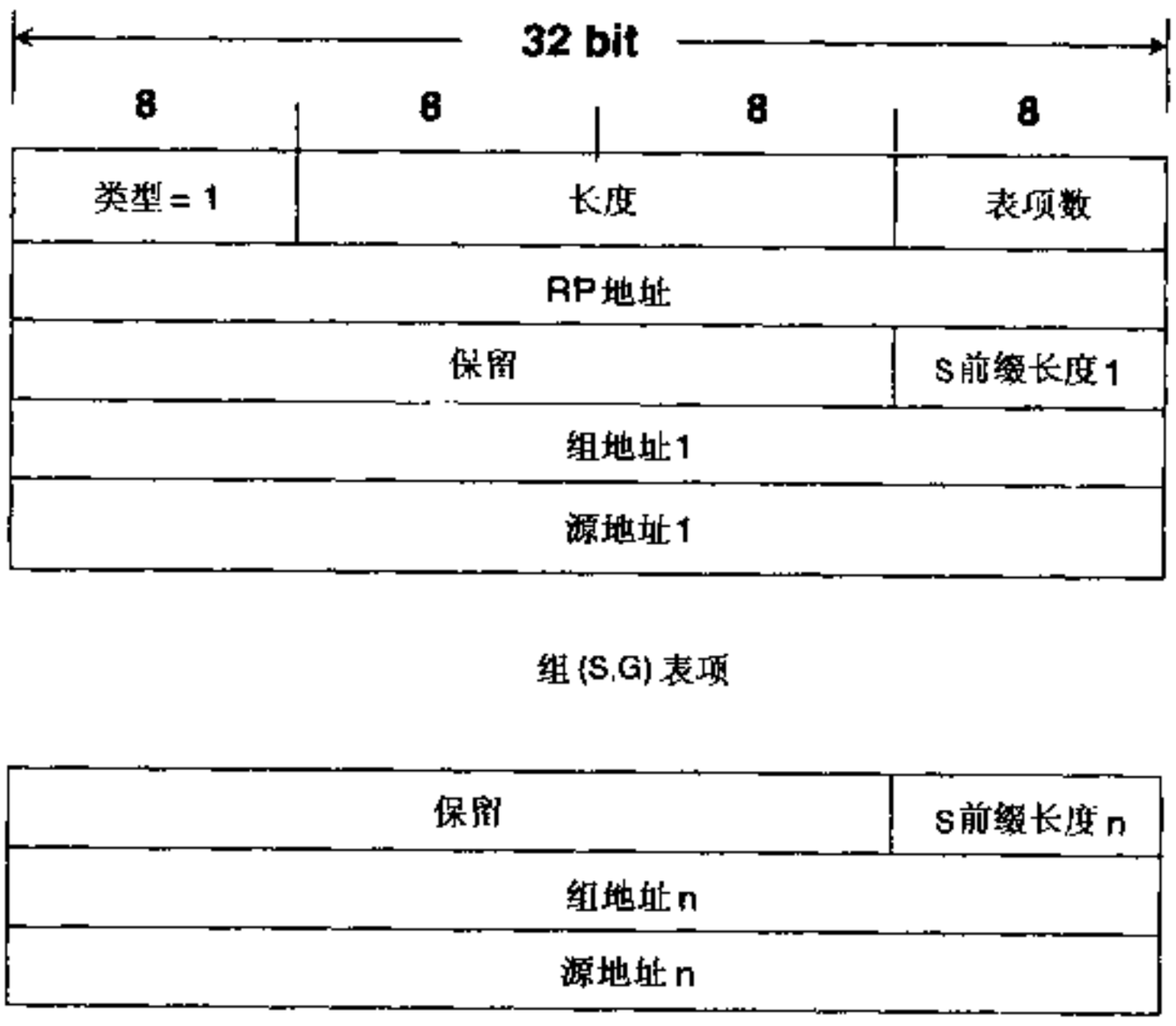


图 7-9　MSDP Source Active TLV 格式

MSDP Source Active TLV 格式中的字段定义如下：

- 条目计数字段是指定 RP 地址所宣告的(S,G)的数量。

- RP 地址是发起 RP 的 IP 地址。
- 保留字段设置为 0。
- 前缀长度是相关源地址的前缀长度。该长度总是 32。
- 多播地址是相关源发送多播包的多播地址。
- 源地址是活跃的源的 IP 地址。

2. Source Active Request TLV

SA Request 消息格式如图 7-10 所示，用于从缓存 SA 状态的 MSDP 对端请求(S,G)信息。SA Request 消息应仅向缓存的对端(非缓存对端将发送出错通告)发送，并仅在明确配置发送时发送。

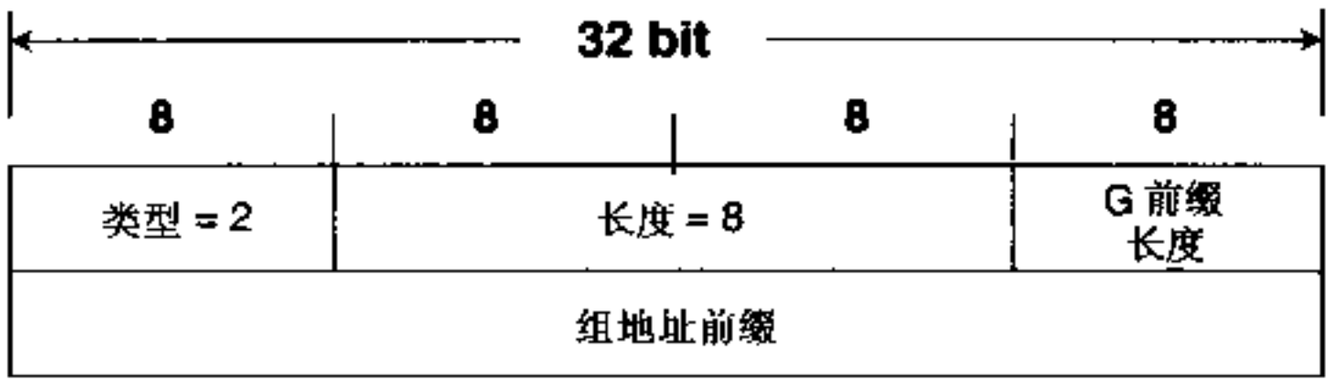


图 7-10　Source Active Request TLV 格式

MSDP Source Active Request TLV 格式中的字段定义如下：

- 组前缀长度是多播地址前缀长度。
- 组地址前缀是源信息被请求的组的组地址。

3. MSDP Source Response TLV

SA Response 消息格式如图 7-11 所示，由缓存对端用于应答 SA Request 消息。他们向请求的对端提供关联于指定组地址的源地址和 RP 地址。其格式与 SA 格式相同。

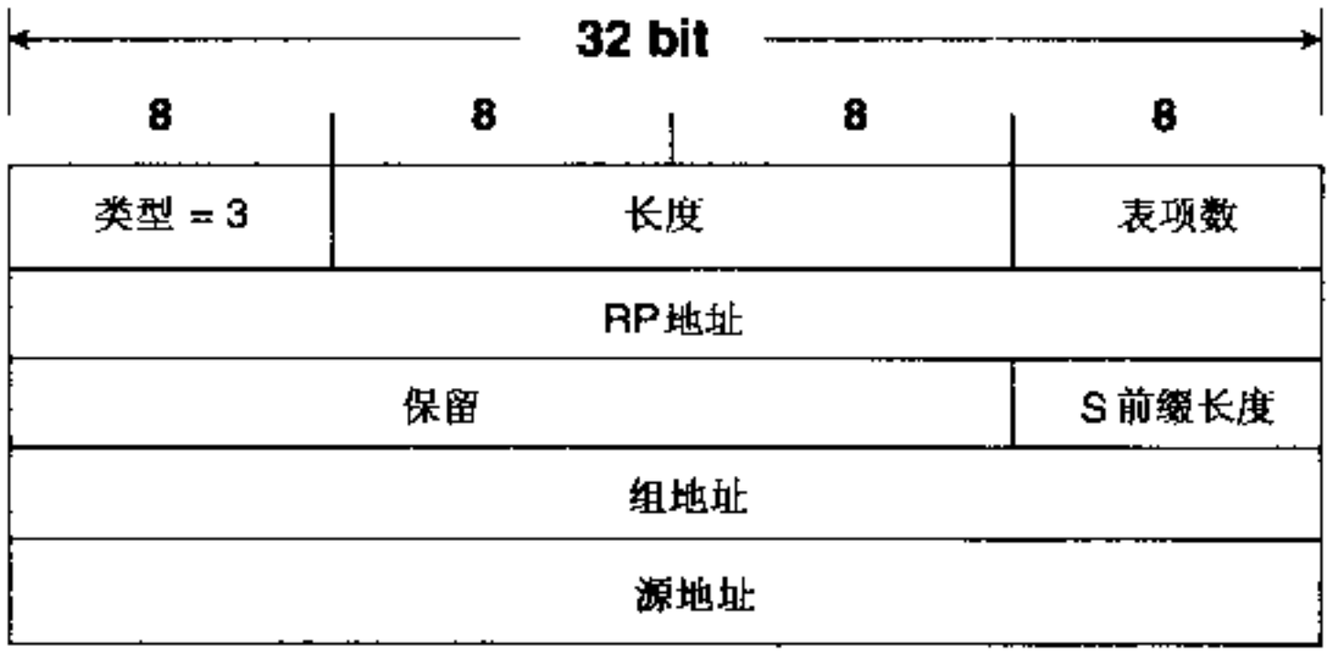


图 7-11　MSDP Source Response TLV 格式

4. Keepalive TLV

MSDP 连接的主动端(IP 地址较小的一端)使用 75 秒定时器来确认被动端的连接。如果在 Keepalive 定时器超时以前无法收到被动端的 MSDP 消息，则主动端将 TCP 连接复位。如果收到 MSDP 消息，则将定时器复位。如果被动端没有需要发送的 MSDP 消息，则发送 Keepalive 消息来防止主动端复位连接。如图 7-12 所示，Keepalive 消息是简单的包含类型和长度的 24

比特字段。

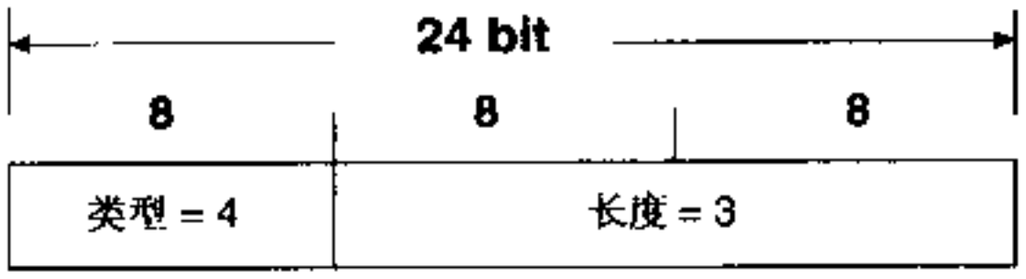


图 7-12　MSDP Keepalive TLV 格式

5．Notification TLV

当检测到错误时发送 Notification 消息。图 7-13 显示了 Notification 消息的格式。

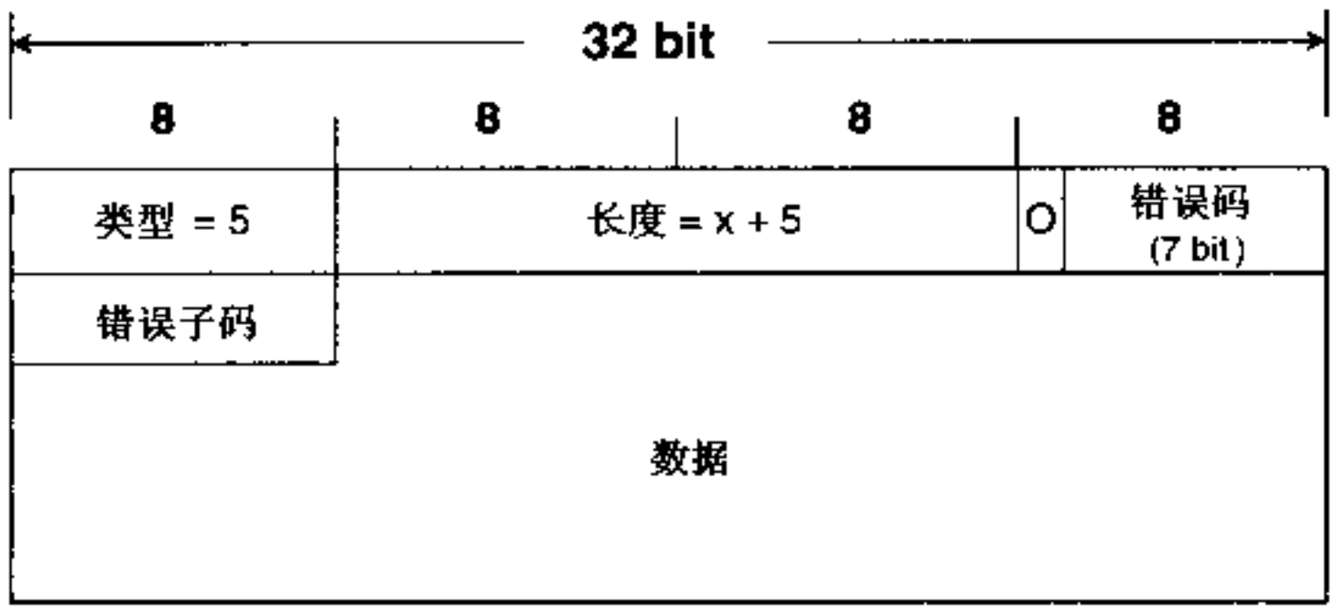


图 7-13　MSDP Notification TLV 格式

MSDP Notification TLV 格式字段定义如下：

- 长度=*x*+5 是 TLV 的长度。*x* 是数据字段的长度，5 是前面 5 个字节。
- O 是开放比特。当该比特为 0 时，收到该消息时连接必须关闭。表 7-5 显示了不同错误子代码中 O 比特的状态。MC 表示必须关闭连接，O 比特必须清 0；CC 表示可以关闭连接，O 比特可以清 0。
- 错误代码是 7 比特的无符号整数，指示通告的类型。表 7-5 罗列了错误代码。
- 错误子代码是 8 比特无符号整数，提供关于错误代码更详细的信息。如果错误代码没有子代码，该字段为 0。表 7-5 显示了每个错误代码相关的子代码。
- 数据是可变长度的字段包含相关错误代码和错误子代码的特定信息。本章中没有包含不同的数据字段；关于该字段内容的详细信息参见 MSDP 互联网草案。

表 7-5　　MSDN 错误代码和错误子代码

错 误 代 码	错误代码描述	错误子代码	错误子代码描述	O 比特状态
1	消息头错误	0	未指定	MC
		2	消息长度错误	MC
		3	消息类型错误	CC
2	SA 请求错误	0	未指定	MC

续表

错 误 代 码	错误代码描述	错误子代码	错误子代码描述	O 比特状态
		1	没有缓存 SA	MC
		2	无效的组	MC
3	SA 消息/SA 应答错误	0	未指定	MC
		1	无效的条目计数	CC
		2	无效的 RP 地址	MC
		3	无效的组地址	MC
		4	无效的源地址	MC
		5	无效的前缀长度	MC
		6	SA 环回(子集是 RP)	MC
		7	未知封装	MC
		8	管理范围边界冲突	MC
4	保持定时器超时	0	未指定	MC
5	有限状态机错误	0	未指定	MC
		1	未期待的消息类型 FSM 错误	MC
6	通告	0	未指定	MC
7	停止	0	未指定	MC

7.5　案例学习：配置 MBGP

图 7-14 表述了 3 个自治系统。AS200 穿过 AS100 宣告单播前缀 172.16.226.0/24 和 172.16.227.0/24，用作普通 AS 间路由。AS200 中还有几个多播源。他们是 172.16.224.1 和 172.16.225.50 上的主机。另外还有几个多播源在 172.16.227.0/24 上，该前缀同时作为单播前缀和多播前缀公告。

例 7-7 显示了图 7-14 中 Gorge 和 Radan 的配置。

例 7-7　图 7-14 中 Gorge 和 Radan 的 MBGP 配置

```
Gorge
router bgp 200
 no synchronization
 network 172.16.226.0 mask 255.255.255.0
 network 172.16.227.0 mask 255.255.255.0
 neighbor 192.168.1.2 remote-as 100
 no auto-summary
 !
 address-family ipv4 multicast
 neighbor 192.168.1.2 activate
```

```
 network 172.16.224.1 mask 255.255.255.255
 network 172.16.225.50 mask 255.255.255.255
 network 172.16.227.0 mask 255.255.255.0
 exit-address-family
```

Rodan

```
router bgp 100
 no synchronization
 neighbor 192.168.1.1 remote-as 200
 neighbor 192.168.254.2 remote-as 100
 neighbor 192.168.254.2 update-source Loopback0
 neighbor 192.168.254.2 next-hop-self
 !
 address-family ipv4 multicast
 neighbor 192.168.1.1 activate
 neighbor 192.168.254.2 activate
 neighbor 192.168.254.2 next-hop-self
 exit-address-family
```

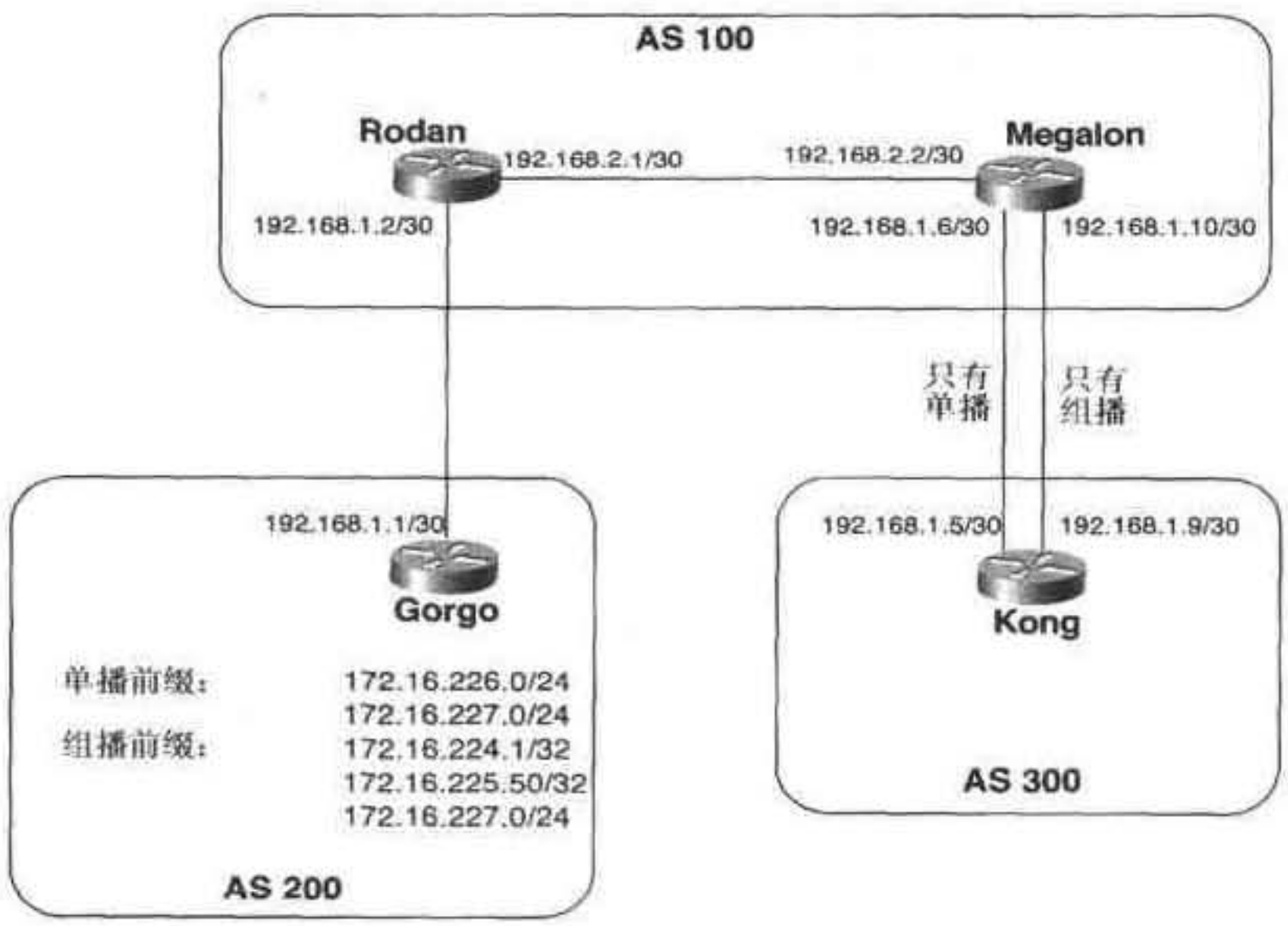


图 7-14　AS200 广告几个前缀和地址：由单播、多播和两者同时

两个路由器上单播部分的 BGP 配置与你在第 3 章“边缘网关协议 4 配置和故障排除”中的配置一样。定义了邻居和他们的 AS 号以及 Gorgo 中广告到 AS100 中的两个单播前缀。

> 注：　本章假设你已经熟悉单播 BGP 配置。如果你对一些 IBGP 工具例如 next-hop-self 或 update-source 不清楚，建议你复习第 3 章。

MBGP 由 **address-family ipv4 multicast** 命令激活。回忆“BGP 多协议扩展(MBGP)”，MBGP使用两个新的属性——MP_REACH_NLRI和MP_UNREACH_NLRI——属性的地址族指示器(AFI)编码是 1——IPv4。Multicast 关键字将属性的 Sub-AFI 设置成多播。在 **address-family** 命令之后，MBGP 的配置和单播 BGP 非常类似。MBGP 邻居被标识，被公告的前缀

被标识成多播。Activate 关键字显示 MBGP 由邻居所激活。对端的 AS 号在 BGP 下指定，不是在 MBGP 下指定。注意 IBGP 配置，例如 next-hop-self 在 MBGP 下的使用与在 BGP 下一样。你可以单独为 MBGP 邻居配置策略。最后一条，exit-address-family 由 Cisco IOS 自动键入，标记 MBGP 配置的结束。

使能 **address-family ipv4 multicast** 隐含地使能了 **address-family ipv4 unicast** 命令。尽管隐含使能的命令没有在配置中显示，该命令被应用到了单播 BGP 配置。应用的结果是该小节配置下指定的前缀被应用到 MP_REACH_NLRI 属性且分配了单播 Sun-AFI。注意 Gorge 配置中的前缀 172.16.227.0/24 出现在 BGP 和 MBGP 中。该前缀广告时即作为单播又作为多播前缀(Sub-AFI=3)。

在例 7-8 中使用 **show ip bgp ipv4** 来显示配置的结果。首先使用 **unicast** 关键字，然后使用 **multicast** 关键字，Sub-AFI 匹配前缀的关键字被显示。在此注意，由于 172.16.227.0/24 被配置成单播前缀和多播前缀，所以出现在两次显示中。

注：　**show ip bgp ipv4** 的输出与 **show ip bgp** 输出相同。

例 7-8　show ip bgp ipv4 按照 Sub-AFI 的前缀显示

```
Rodan#show ip bgp ipv4 unicast
BGP table version is 7, local router ID is 192.168.254.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 172.16.226.0/24  192.168.1.1               0             0 200 i
*> 172.16.227.0/24  192.168.1.1               0             0 200 i

Rodan#show ip bgp ipv4 multicast
BGP table version is 10, local router ID is 192.168.254.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 172.16.224.1/32  192.168.1.1               0             0 200 i
*> 172.16.225.50/32 192.168.1.1               0             0 200 i
*> 172.16.227.0/24  192.168.1.1               0             0 200 i
Rodan#
```

由于为 BGP 和 MBGP 使用不同的链路，图 7-14 中 Megalon 和 Kong 的配置稍显复杂。例 7-9 显示上述两个路由器的配置。

例 7-9　配置 Megalon 和 Kong，使单播和多播使用独立的数据链路

```
Megalon
router bgp 100
 no synchronization
 no bgp default ipv4-unicast
 neighbor 192.168.1.5 remote-as 300
 neighbor 192.168.1.5 activate
 neighbor 192.168.1.9 remote-as 300
 neighbor 192.168.254.1 remote-as 100
 neighbor 192.168.254.1 update-source Loopback0
```

```
 neighbor 192.168.254.1 activate
 neighbor 192.168.254.1 next-hop-self
 no auto-summary
 !
 address-family ipv4 multicast
 neighbor 192.168.1.9 activate
 neighbor 192.168.254.1 activate
 exit-address-family
```

Kong

```
router bgp 300
 no synchronization
 no bgp default ipv4-unicast
 neighbor 192.168.1.6 remote-as 100
 neighbor 192.168.1.6 activate
 neighbor 192.168.1.10 remote-as 100
 no auto-summary
 !
 address-family ipv4 multicast
 neighbor 192.168.1.10 activate
 exit-address-family
```

MBGP 配置说明只有 192.168.1.8/30 子网用作对等关系，并且在单播 BGP 会话下有一些新命令。记住使用 **address-family ipv4 multicast** 命令后，**address-family ipv4 unicast** 命令被自动隐含调用。在子网 192.168.1.8/30 上不需要单播 BGP 流量。所以使用 **no ip default ipv4-unicast** 命令来防止自动的行为。然后在需要的链路上使用 **neighbor active** 命令显式地启动单播 BGP。注意 192.168.2.1/30 和 192.168.1.4/30 子网可用作单播，192.168.1.8/30 不能用作单播。只有在 BGP 命令下指定的 AS 号才能建立邻接关系。

例 7-10 显示了例 7-9 的配置结果，AS 300 收到由 AS 200 宣告的前缀，在 Kong 和 Megalon 间为单播和多播选择正确的下一跳。输出看起来与例 7-8 相似，正确地区分单播多播前缀。但是在这个例子中，单播前缀的下一跳地址是 192.168.1.6，多播前缀的下一跳地址是 192.168.1.10。

例 7-10　AS300 在 Kong 和 Megalon 间为单播和多播选择正确的下一跳

```
Kong#show ip bgp ipv4 unicast
BGP table version is 7, local router ID is 10.254.254.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 172.16.226.0/24  192.168.1.6                           0 100 200 i
*> 172.16.227.0/24  192.168.1.6                           0 100 200 i

Kong#show ip bgp ipv4 multicast
BGP table version is 10, local router ID is 10.254.254.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 172.16.224.1/32  192.168.1.10                          0 100 200 i
*> 172.16.225.50/32 192.168.1.10                          0 100 200 i
*> 172.16.227.0/24  192.168.1.10                          0 100 200 i
Kong#
```

例 7-11 显示了 BGP 与 MBGP 公告的实际应用。使用同时作为单播和多播公告的前缀 172.16.227.0/24，为 172.16.227.1 查找路由。显示内容说明路由携带的下一跳地址是 192.168.1.6，该路由是图 7-14 中的单播链路。然后在 相同地址上执行 RPF 查找。查找返回的下一跳地址是 192.168.1.10，是多播链路的地址。根据地址不同的的使用功能，为同一地址选择不同的链路。

例 7-11　对 172.16.227.1 下一跳是 192.168.1.6，RPF 查找得到 192.168.1.10

```
Kong#show ip route 172.16.227.1
Routing entry for 172.16.227.0/24
  Known via "bgp 300", distance 20, metric 0
  Tag 100, type external
  Last update from 192.168.1.6 04:10:21 ago
  Routing Descriptor Blocks:
  * 192.168.1.6, from 192.168.1.6, 04:10:21 ago
      Route metric is 0, traffic share count is 1
      AS Hops 2

Kong#show ip rpf 172.16.227.1
RPF information for ? (172.16.227.1)
  RPF interface: Serial1
  RPF neighbor: ? (192.168.1.10)
  RPF route/mask: 172.16.227.0/24
  RPF type: mbgp
  RPF recursion count: 0
  Doing distance-preferred lookups across tables
Kong#
```

下面内容值的最后一次强调：MBGP 不影响多播流量的转发。在某些情况下需要进一步配置才能影响转发，例如图 7-14 中相同链路选择时强迫多播流量通过专门的组播链路。MBGP 只负责穿过 AS 边界分发 RPF 信息。

7.6　案例学习：配置 MSDP

图 7-15 显示了前面案例学习中的路由器。在这里 4 个路由器都是所代表 AS 的 RP，由 RP 地址说明。

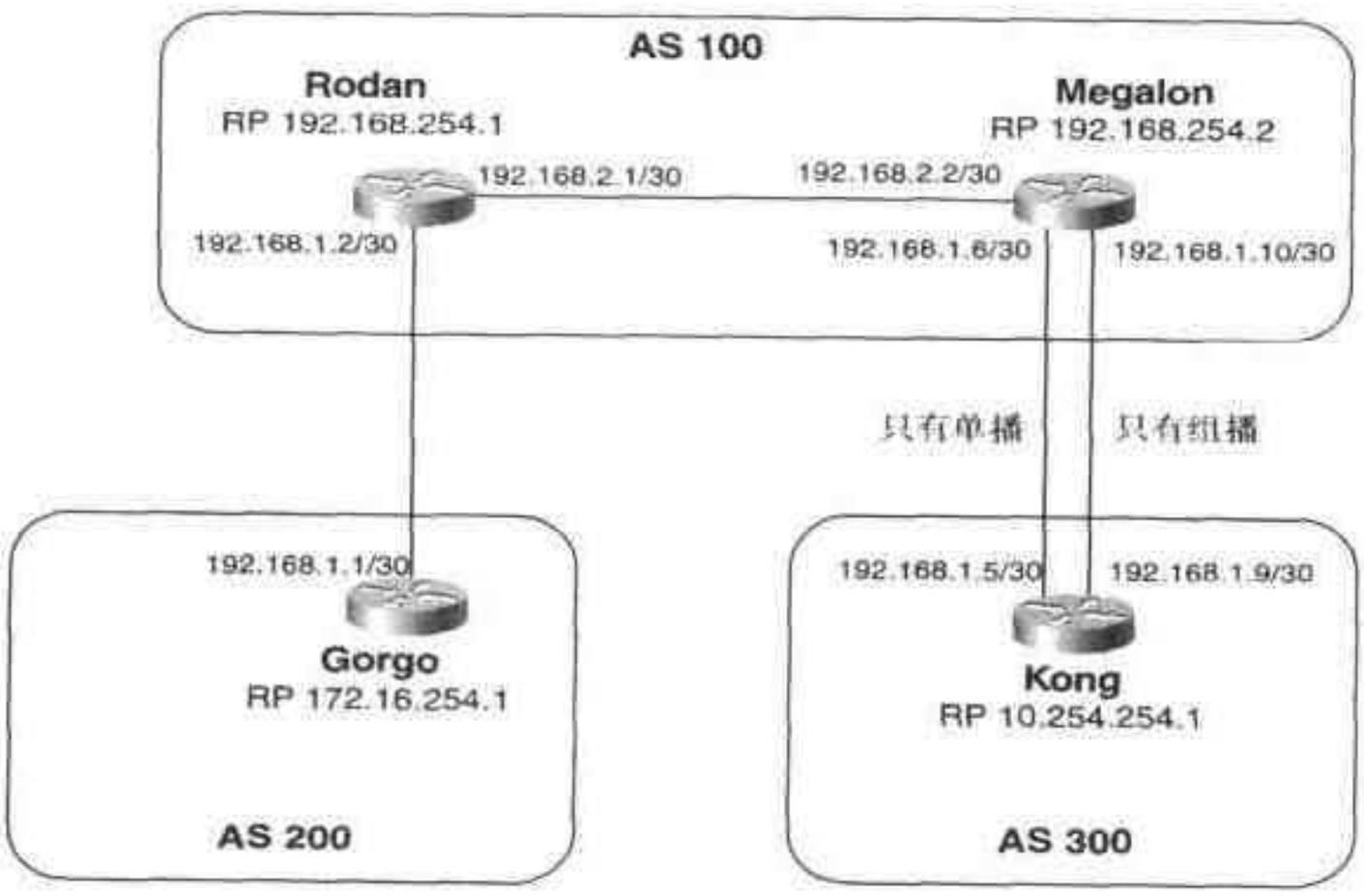


图 7-15　在 4 个路由器间配置 MSDP 会话

使用 **ip msdp peer** 命令且指定对端的地址即可启动 MSDP，非常简单。例 7-12 显示了图 7-15 中 4 个路由器的配置。

例 7-12　在图 7-15 中 4 个路由器上配置 MSDP

```
Gorgo
ip msdp peer 192.168.1.2
!
Kong
ip msdp peer 192.168.1.10
!
Rodan
ip msdp peer 192.168.1.1
ip msdp peer 192.168.254.2 connect-source Loopback0
!
Megalon
ip msdp peer 192.168.254.1 connect-source Loopback0
ip msdp peer 192.168.1.9
```

对等关系建立在 Gorgo 和 Rodan 间以及 Kong 和 Megalon 间，非常直观。路由器到对端之间都有单一的链路，所以会话配置在两个物理接口地址间。然而 Rodan 和 Megalon 间的对等关系建立在环回地址间。和 IBGP 对等关系一样，建立在环回地址上的 MSDP 会话能提供更高的可靠性。如果图 7-15 中显示的 Rodan 和 Megalon 间的链路故障并且路由器间存在其他链路(没有显示在图中)，则 TCP 会话可以重路由。缺省条件下，携带在 MSDP 会话 TCP 包中的源地址是发包的物理接口的地址。需要与非直连子网的地址建立对等关系时使用 connect-source 选项来改变缺省的源地址。

例 7-13 使用 **show ip msdp peer** 命令显示了 Megalon 两个 MSDP 会话的状态。可以想象的信息有连接状态，建立时间，收发的消息等。

例 7-13　使用 **show ip msdp peer** 命令显示 MSDP 对等会话的状态

```
Megalon#show ip msdp peer
MSDP Peer 192.168.254.1 (?), AS 100
Description:
  Connection status:
    State: Up, Resets: 0, Connection source: Loopback0 (192.168.254.2)
    Uptime(Downtime): 3d22h, Messages sent/received: 5683/5677
    Output messages discarded: 0
    Connection and counters cleared 3d22h    ago
  SA Filtering:
    Input filter: none, route-map: none
    Output filter: none, route-map: none
  SA-Requests:
    Input filter: none
    Sending SA-Requests to peer: disabled
  Peer ttl threshold: 0
  Input queue size: 0, Output queue size: 0
MSDP Peer 192.168.1.9 (?), AS 300
Description:
  Connection status:
    State: Up, Resets: 0, Connection source: none configured
    Uptime(Downtime): 3d22h, Messages sent/received: 5674/5694
```

```
    Output messages discarded: 0
    Connection and counters cleared 3d22h    ago
  SA Filtering:
    Input filter: none, route-map: none
    Output filter: none, route-map: none
  SA-Requests:
    Input filter: none
    Sending SA-Requests to peer: disabled
  Peer ttl threshold: 0
  Input queue size: 0, Output queue size: 0
Megalon#
```

例 7-13 显示内容中还可以看到可能为 SA 和 SA Request 消息配置的过滤器。你可以在 MSDP 路由器上使用过滤器选项来控制和限制 MSDP 行为。你可以：

- 控制本地的源在 RP 注册
- 控制 RP 向 MSDP 对端发送或从对端收到的 SA 消息
- 控制 RP 向 MSDP 对端发送或从对端收到的 SA Request 消息

大范围 MSDP 环境其他选项是对每个对端的附加描述和 MSDP 消息的 TTL 值。例 7-14 显示了对图 7-15 中 Megalon 路由器的更繁琐的配置。

注：　这里显示的是配置制作示范，没有对配置使用的讨论。

例 7-14　更详细的 MSDP 配置

```
ip pim rp-address 192.168.254.2
ip msdp peer 192.168.254.1 connect-source Loopback0
ip msdp description 192.168.254.1 Rodan in AS 100
ip msdp sa-filter out 192.168.254.1 list 101
ip msdp filter-sa-request 192.168.254.1 list 1
ip msdp sa-request 192.168.254.1
ip msdp ttl-threshold 192.168.254.1 5
ip msdp peer 192.168.1.9
ip msdp description 192.168.1.9 Kong in AS 300
ip msdp sa-filter in 192.168.1.9 list 101
ip msdp sa-filter out 192.168.1.9 list 103
ip msdp sa-request 192.168.1.9
ip msdp ttl-threshold 192.168.1.9 2
ip msdp cache-sa-state list 101
ip msdp redistribute list 102
!
access-list 1 permit 229.50.0.0 0.0.255.255
access-list 101 permit ip 10.254.0.0 0.0.255.255 224.0.0.0 31.255.255.255
access-list 102 permit ip 192.168.224.0 0.0.0.255 224.0.0.0 31.255.255.255
access-list 103 permit ip 172.16.0.0 0.0.255.255 230.0.0.0 0.255.255.255
access-list 103 permit ip 192.168.224.0 0.0.0.255 224.0.0.0 31.255.255.255
```

在例 7-12 中使能到 Rodan 和 Kong 的两条命令仍然保留。增加到配置中的是使用 **ip msdp description** 命令对每个对端的描述。该描述总是直接跟在相应的 **ip msdp peer** 命令之后，这在存在大量 MSDP 对端时非常有用。

使用 **ip msdp cache-sa-state** 命令打开了 SA 缓存，该配置还应用了可选的访问列表。访问列表 101 指定 Megalon 只缓存源地址是 10.254.0.0/16 的(S,G)对的 SA 消息。组可以是任何多播地址(224.0.0.0/3)。

对两个对等体配置的 **ip msdp sa-request** 命令进一步降低了加入时延。如果路由器收到对特定组的加入消息，则向邻居发送 SA Request 消息。前提是假设两个邻居都配置了缓存 SA 消息。

发往 Rodan(192.168.254.1)的 SA Request 由 **ip msdp filter-sa-request** 命令进一步限制。引用的过滤器是访问列表 1，只允许 229.50.0.0/16。结果是 Megalon 只能从 Rodan 请求地址在 229.50.0.0/16 前缀范围内的多播源信息。

下一步 Megalon 被配置成向发来 PIM 注册消息的可能的源所在的子网发送 SA 消息。**ip msdp redistribute** 命令引用了访问列表 102，该访问列表依次允许源地址前缀 192.168.224.0/24 和组地址前缀 224.0.0.0/3(所有多播组)。在 RP 的 PIM-SM 配置限制中的任何源都可以在 RP 上注册，但是只有源地址前 24 比特在 192.168.224 内才能在 SA 消息中宣告。

向 MSDP 对端的 SA 消息转发由 **ip msdp sa-filter out** 命令来规定。该过滤器应用在所有的 SA 消息上，无论消息是本地发起的还是从其他对端 MSDP 收到的。但是 **ip msdp redistribute** 命令只应用在本地发起的 SA 消息。Megalon 有两条这样的命令。对于邻居 Rodan，由访问列表 101 规定只转发源地址前缀是 10.254.0.0/16 的消息。Megalon 只向 Kong(192.168.1.9)发送访问列表 103 规定的 SA 消息。该访问列表允许下列消息：源地址前缀是 172.16.0.0/16 且组地址前缀是 230.0.0.0/8 的消息或源地址前缀是 192.168.224.0/24 组地址前缀任意的消息。

你还可以使用 **ip msdp sa-filter in** 命令过滤收到的 SA 消息。使用该命令，Megalon 只从 Kong 接收访问列表 101 所允许的 SA 消息。注意在发往 Rodan 的 SA 消息有同样的限制。

最后由 **ip msdp ttl-threshold** 规定了 MSDP 消息的 TTL 值。发往 Radon 消息的 TTL 设为 5，发往 Kong 的消息的 TTL 设为 2。

例 7-15 显示了配置的结果。将输出与例 7-13 的输出对比，你可以看到改变的描述、过滤器和 TTL 门限。

例 7-15　体现在 Megalon 上 MSDP 配置改变的输出

```
Megalon#show ip msdp peer
MSDP Peer 192.168.254.1 (?), AS 100
Description: Rodan in AS 100
  Connection status:
    State: Up, Resets: 0, Connection source: Loopback0 (192.168.254.2)
    Uptime(Downtime): 4d14h, Messages sent/received: 6624/6617
    Output messages discarded: 0
    Connection and counters cleared 4d14h    ago
  SA Filtering:
    Input filter: none, route-map: none
    Output filter: 101, route-map: none
  SA-Requests:
    Input filter: 1
    Sending SA-Requests to peer: enabled
  Peer ttl threshold: 5
  Input queue size: 0, Output queue size: 0
MSDP Peer 192.168.1.9 (?), AS 300
Description: Kong in AS 300
  Connection status:
    State: Up, Resets: 0, Connection source: none configured
    Uptime(Downtime): 4d14h, Messages sent/received: 6614/6634
    Output messages discarded: 0
    Connection and counters cleared 4d14h    ago
```

```
  SA Filtering:
    Input filter: 101, route-map: none
    Output filter: 102, route-map: none
  SA-Requests:
    Input filter: none
    Sending SA-Requests to peer: enabled
  Peer ttl threshold: 2
  Input queue size: 0, Output queue size: 0
Megalon#
```

在访问列表之外，你可以将出入的 SA 过滤器链接到路由映射来得到策略应用更好的控制粒度。你同样可以在 MSDP 再分发上联用路由映射，正如在 AS 路径访问列表上一样。

7.7　案例学习：MSDP 全连接组

在前面的案例学习中，路由器 Rodan 和 Megalon 是同一 AS 上的 RP。在一个大的多播域中通常有多个 RP 来做负荷分担或多播树分区。尽管 MSDP 主要用做共享 AS 间的源信息的工具，但是在用作单一域内多个 RP 间共享信息也被证明非常有效。多播源通常注册到某一个 RP，但是域内所有的成员都应当能找到任何多播源。

在域内的每一个 RP 通常与域内每一个其他 RP 拥有一个 MSDP 对等会话，用于冗余和健壮性的目的。图 7-16 显示了一个例子。在同一个 AS 内显示了 4 个路由器，每个路由器都和其他 3 个路由器建立了对等连接。4 个路由器可能直接连接，也可能在物理上远离。

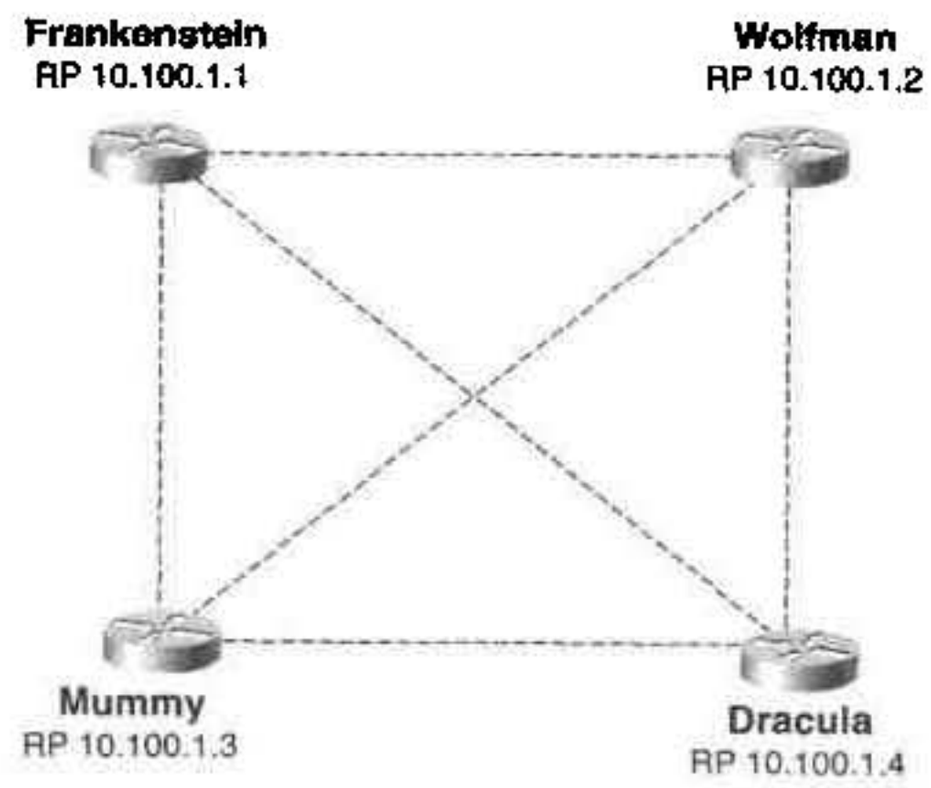


图 7-16　4 个路由器间的 MSDP 全连接

例 7-16 显示了图 7-16 中 4 个路由器的配置。

例 7-16　在图 7-16 中 4 个路由器上配置 MSDP

```
Frankenstein
ip pim rp-address 10.100.1.1
ip msdp peer 10.100.1.3 connect-source Loopback0
ip msdp description 10.100.1.3 to Mummy
ip msdp peer 10.100.1.2 connect-source Loopback0
ip msdp description 10.100.1.2 to Wolfman
ip msdp peer 10.100.1.4 connect-source Loopback0
ip msdp description 10.100.1.4 to Dracula
```

Wolfman

```
ip pim rp-address 10.100.1.2
ip msdp peer 10.100.1.1 connect-source Loopback0
ip msdp description 10.100.1.1 to Frankenstein
ip msdp peer 10.100.1.3 connect-source Loopback0
ip msdp description 10.100.1.3 to Mummy
ip msdp peer 10.100.1.4 connect-source Loopback0
ip msdp description 10.100.1.4 to Dracula
```

Mummy

```
ip pim rp-address 10.100.1.3
ip msdp peer 10.100.1.1 connect-source Loopback0
ip msdp description 10.100.1.1 to Frankenstein
ip msdp peer 10.100.1.2 connect-source Loopback0
ip msdp description 10.100.1.2 to Wolfman
ip msdp peer 10.100.1.4 connect-source Loopback0
ip msdp description 10.100.1.4 to Dracula
```

Dracula

```
ip pim rp-address 10.100.1.4
ip msdp peer 10.100.1.1 connect-source Loopback0
ip msdp description 10.100.1.1 to Frankenstein
ip msdp peer 10.100.1.2 connect-source Loopback0
ip msdp description 10.100.1.2 to Wolfman
ip msdp peer 10.100.1.3 connect-source Loopback0
ip msdp description 10.100.1.3 to Mummy
```

配置中的问题在于每一个路由器产生的 SA 消息被洪泛到所有其他路由器，引起大量的 RPF 洪泛失败产生 MSDP 通知消息。如果每一个 RP 都有到其他 RP 的连接，则不需要洪泛。每一个 RP 从发送者那里直接得到 SA 的一个拷贝。建立 MSDP 全连接组解决了洪泛问题。

MSDP 全连接组是如图 7-16 所示一组完全网状连接的 MSDP 对等体，对等体间不需要转发 SA 消息。即当 RP 从对端收到 SA 后，不需要将消息向其他对等体转发。

全连接组由 **ip msdp mesh-group** 命令配置。全连接组可以有一个任意的名字(所以如果需要的话，每个 RP 可以属于多个全连接组)，并指定组成员。例 7-17 中的配置将图 7-16 中的 RP 加入到一个名为 Boogeyman 的全连接组。

例 7-17　将图 7-16 中的 RP 加入到一个名为 Boogeyman 的全连接组

Frankenstein

```
ip pim rp-address 10.100.1.1
ip msdp peer 10.100.1.3 connect-source Loopback0
ip msdp description 10.100.1.3 to Mummy
ip msdp peer 10.100.1.2 connect-source Loopback0
ip msdp description 10.100.1.2 to Wolfman
ip msdp peer 10.100.1.4 connect-source Loopback0
ip msdp description 10.100.1.4 to Dracula
ip msdp mesh-group Boogeymen 10.100.1.3
ip msdp mesh-group Boogeymen 10.100.1.2
ip msdp mesh-group Boogeymen 10.100.1.4
```

Wolfman

```
ip pim rp-address 10.100.1.2
ip msdp peer 10.100.1.1 connect-source Loopback0
ip msdp description 10.100.1.1 to Frankenstein
ip msdp peer 10.100.1.3 connect-source Loopback0
ip msdp description 10.100.1.3 to Mummy
ip msdp peer 10.100.1.4 connect-source Loopback0
ip msdp description 10.100.1.4 to Dracula
ip msdp mesh-group Boogeymen 10.100.1.1
ip msdp mesh-group Boogeymen 10.100.1.3
ip msdp mesh-group Boogeymen 10.100.1.4
```

Mummy

```
ip pim rp-address 10.100.1.3
ip msdp peer 10.100.1.1 connect-source Loopback0
ip msdp description 10.100.1.1 to Frankenstein
ip msdp peer 10.100.1.2 connect-source Loopback0
ip msdp description 10.100.1.2 to Wolfman
ip msdp peer 10.100.1.4 connect-source Loopback0
ip msdp description 10.100.1.4 to Dracula
ip msdp mesh-group Boogeymen 10.100.1.1
ip msdp mesh-group Boogeymen 10.100.1.2
ip msdp mesh-group Boogeymen 10.100.1.4
```

Dracula

```
ip pim rp-address 10.100.1.4
ip msdp peer 10.100.1.1 connect-source Loopback0
ip msdp description 10.100.1.1 to Frankenstein
ip msdp peer 10.100.1.2 connect-source Loopback0
ip msdp description 10.100.1.2 to Wolfman
ip msdp peer 10.100.1.3 connect-source Loopback0
ip msdp description 10.100.1.3 to Mummy
ip msdp mesh-group Boogeymen 10.100.1.1
ip msdp mesh-group Boogeymen 10.100.1.2
ip msdp mesh-group Boogeymen 10.100.1.3
```

7.8　案例学习：泛播 RP

设计一个巨大的地理上分散的 PIM-SM 域通常需要在有效地放置 RP 的两难选择上斟酌。PIM-SM 只允许组-RP 的单一映射，在大的多播域中出现下列问题：[4]

- 可能的流量瓶颈
- 缺少可扩展的注册器解封装(使用共享树时)
- 当激活的 RP 故障时，故障恢复慢
- 对多播包可能次优转发
- 依赖于远端 RP

你已在第 5 第 6 章读到了一些缓解上述问题的不同机制，例如对 PIMv2 自举协议的 hash 算法和自动 RP 过滤。这些工具都不能提供完全解决问题的方案。泛播 RP 是允许单个组映射到多个 RP 的方法。这些 RP 可以分布在整个域范围，并使用相同的 RP 地址。结果产生了虚拟 RP。MSDP 是产生虚拟 RP 的基础。

> 注：　泛播表示一个包发送到单一地址时，多个设备中一个设备可以响应该地址。

图 7-17 显示了使用上一个案例学习中的路由器的例子，如图所示 4 个路由器组成虚拟 RP，发布单一 RP 地址 10.100.254.1，使用 MSDP 来交换注册到每个路由器上的源信息。但是所有的路由器都运行自动-RP 并发布 10.100.254.1 的 RP 地址。在域中的源 DR 只有一个 RP 地址信息，注册到最近的物理 RP。通常这会引起 PIM 域的分裂。但是使用 MSDP 全连接组可以使组内的泛播 RP 交换源信息。

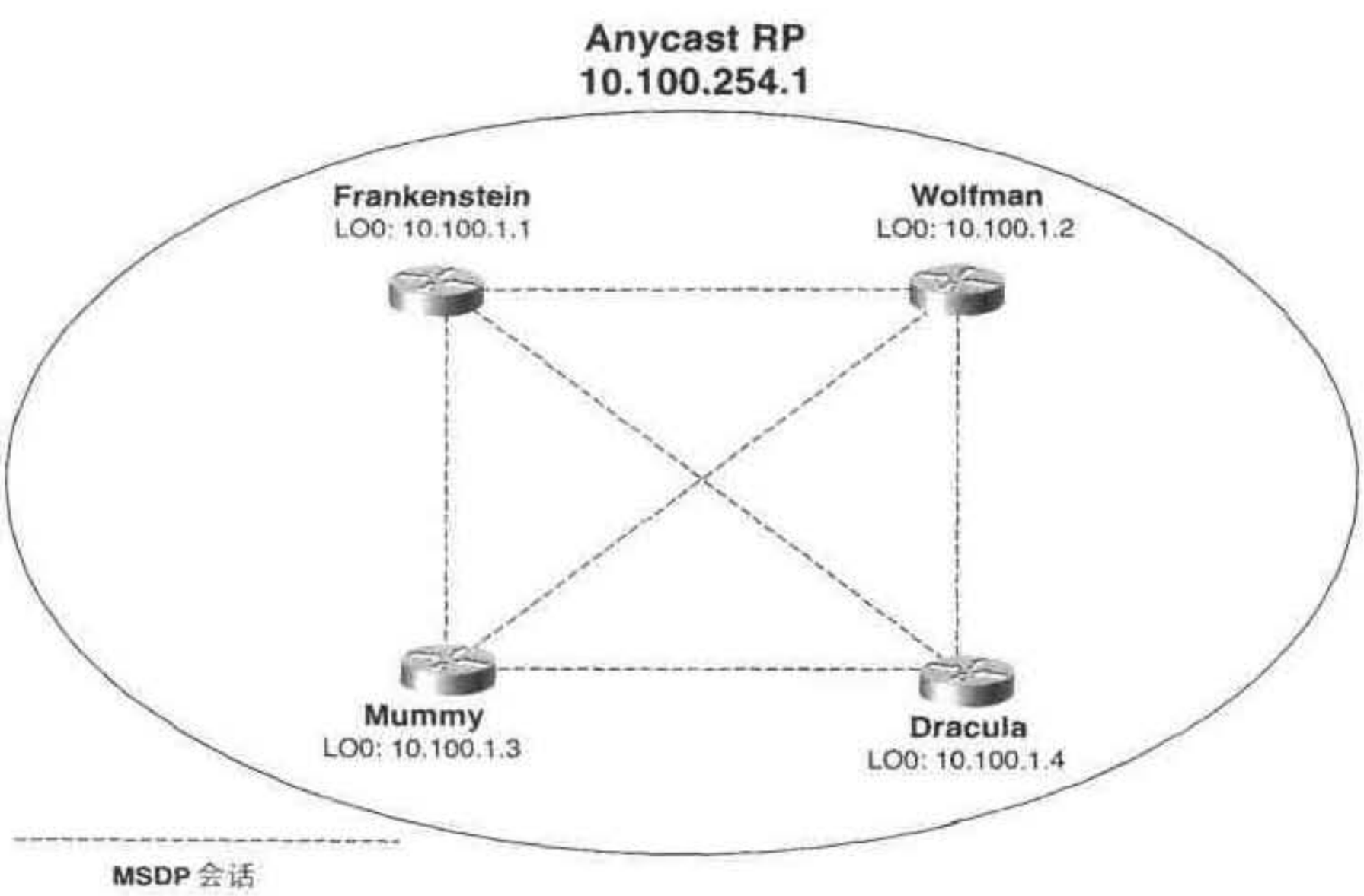


图 7-17　发布单一 RP 地址的虚拟 RP

每个泛播 RP 的单播路由协议宣告共同的 RP 地址。从源和组 DR 的观点看，该地址上只有一个单一 RP 以及多条路径。DR 选择一条最短的路径，该路径有效地到达泛播 RP。当某个泛播 RP 故障时，单播协议宣布到达 RP 的路由不可用。DR 看到不可用的路由，然后选择次优路由，该路由能到达次近的泛播 RP。结果是 RP 路由恢复链接与单播路由恢复一样快。

MSDP 对等关系像以前一样在 LO0 间建立；使用另一个 loopback 来配置路由器所宣布的 RP 地址。通常 MSDP 使用在 SA 消息中的 RP 地址。由于所有 4 个路由器发布相同的 RP 地址，MSDP 必须配置成使用 SA 消息中的唯一地址。可以使用 **ip msdp originator-id** 命令来达到上述目的。例 7-18 显示了 4 个路由器的相关配置，使用了全连接组和 **ip msdp originator-id**。

例 7-18　Frankenstein, Wolfman, Mummy 和 Dracula 的 anycast RP 配置

```
Frankenstein
interface Loopback0
 ip address 10.100.1.1 255.255.255.255
!
interface Loopback5
 ip address 10.100.254.1 255.255.255.255
 ip pim sparse-dense-mode
!
```

```
router ospf 1
 router-id 10.100.1.1
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 6500
 bgp router-id 10.100.1.1
 neighbor Boogeymen peer-group
 neighbor Boogeymen remote-as 6500
 neighbor Boogeymen update-source Loopback0
 neighbor 10.100.1.2 peer-group Boogeymen
 neighbor 10.100.1.3 peer-group Boogeymen
 neighbor 10.100.1.4 peer-group Boogeymen
 !
 address-family ipv4 multicast
 neighbor 10.100.1.2 activate
 neighbor 10.100.1.3 activate
 neighbor 10.100.1.4 activate
 exit-address-family
!
ip pim send-rp-announce Loopback5 scope 20
ip pim send-rp-discovery Loopback5 scope 20
ip msdp peer 10.100.1.3 connect-source Loopback0
ip msdp description 10.100.1.3 to Mummy
ip msdp peer 10.100.1.2 connect-source Loopback0
ip msdp description 10.100.1.2 to Wolfman
ip msdp peer 10.100.1.4 connect-source Loopback0
ip msdp description 10.100.1.4 to Dracula
ip msdp mesh-group Boogeymen 10.100.1.3
ip msdp mesh-group Boogeymen 10.100.1.2
ip msdp mesh-group Boogeymen 10.100.1.4
ip msdp cache-sa-state
ip msdp originator-id Loopback0
```

Wolfman

```
interface Loopback0
 ip address 10.100.1.2 255.255.255.255
!
interface Loopback5
 ip address 10.100.254.1 255.255.255.255
 ip pim sparse-dense-mode
!
router ospf 1
 router-id 10.100.1.2
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 6500
 bgp router-id 10.100.1.2
 neighbor Boogeymen peer-group
 neighbor Boogeymen remote-as 6500
 neighbor Boogeymen update-source Loopback0
 neighbor 10.100.1.1 peer-group Boogeymen
 neighbor 10.100.1.3 peer-group Boogeymen
 neighbor 10.100.1.4 peer-group Boogeymen
 !
 address-family ipv4 multicast
 neighbor 10.100.1.1 activate
 neighbor 10.100.1.3 activate
 neighbor 10.100.1.4 activate
 exit-address-family
!
```

```
ip pim send-rp-announce Loopback5 scope 20
ip pim send-rp-discovery Loopback5 scope 20
ip msdp peer 10.100.1.1 connect-source Loopback0
ip msdp description 10.100.1.1 to Frankenstein
ip msdp peer 10.100.1.3 connect-source Loopback0
ip msdp description 10.100.1.3 to Mummy
ip msdp peer 10.100.1.4 connect-source Loopback0
ip msdp description 10.100.1.4 to Dracula
ip msdp mesh-group Boogeymen 10.100.1.1
ip msdp mesh-group Boogeymen 10.100.1.3
ip msdp mesh-group Boogeymen 10.100.1.4
ip msdp cache-sa-state
ip msdp originator-id Loopback0
```

Mummy

```
interface Loopback0
 ip address 10.100.1.3 255.255.255.255
!
interface Loopback5
 ip address 10.100.254.1 255.255.255.255
 ip pim sparse-dense-mode
!
router ospf 1
 router-id 10.100.1.3
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 6500
 bgp router-id 10.100.1.3
 neighbor Boogeymen peer-group
 neighbor Boogeymen remote-as 6500
 neighbor Boogeymen update-source Loopback0
 neighbor 10.100.1.1 peer-group Boogeymen
 neighbor 10.100.1.2 peer-group Boogeymen
 neighbor 10.100.1.4 peer-group Boogeymen
 !
 address-family ipv4 multicast
 neighbor 10.100.1.1 activate
 neighbor 10.100.1.2 activate
 neighbor 10.100.1.4 activate
 exit-address-family
ip pim send-rp-announce Loopback5 scope 20
ip pim send-rp-discovery Loopback5 scope 20
ip msdp peer 10.100.1.1 connect-source Loopback0
ip msdp description 10.100.1.1 to Frankenstein
ip msdp peer 10.100.1.2 connect-source Loopback0
ip msdp description 10.100.1.2 to Wolfman
ip msdp peer 10.100.1.4 connect-source Loopback0
ip msdp description 10.100.1.4 to Dracula
ip msdp mesh-group Boogeymen 10.100.1.1
ip msdp mesh-group Boogeymen 10.100.1.2
ip msdp mesh-group Boogeymen 10.100.1.4
ip msdp cache-sa-state
ip msdp originator-id Loopback0
```

Dracula

```
interface Loopback0
 ip address 10.100.1.4 255.255.255.255
!
interface Loopback5
 ip address 10.100.254.1 255.255.255.255
 ip pim sparse-dense-mode
```

```
!
router ospf 1
 router-id 10.100.1.4
 network 0.0.0.0 255.255.255.255 area 0
!
router bgp 6500
 bgp router-id 10.100.1.4
 neighbor Boogeymen peer-group
 neighbor Boogeymen remote-as 6500
 neighbor Boogeymen update-source Loopback0
 neighbor 10.100.1.1 peer-group Boogeymen
 neighbor 10.100.1.2 peer-group Boogeymen
 neighbor 10.100.1.3 peer-group Boogeymen
 !
 address-family ipv4 multicast
 neighbor 10.100.1.1 activate
 neighbor 10.100.1.2 activate
 neighbor 10.100.1.3 activate
 exit-address-family
!
ip pim send-rp-announce Loopback5 scope 20
ip pim send-rp-discovery Loopback5 scope 20
ip msdp peer 10.100.1.1 connect-source Loopback0
ip msdp description 10.100.1.1 to Frankenstein
ip msdp peer 10.100.1.2 connect-source Loopback0
ip msdp description 10.100.1.2 to Wolfman
ip msdp peer 10.100.1.3 connect-source Loopback0
ip msdp description 10.100.1.3 to Mummy
ip msdp mesh-group Boogeymen 10.100.1.1
ip msdp mesh-group Boogeymen 10.100.1.2
ip msdp mesh-group Boogeymen 10.100.1.3
ip msdp cache-sa-state
ip msdp originator-id Loopback0
```

在例7-18中4个路由器的每一个都配置成自动-RP候选RP以及映射代理。你还可以对泛播RP使用静态映射或PIMv2自举。本例中所有4个路由器都配置成缓存SA消息。

每个路由器的LO5都被配置成虚拟RP地址，LO0是MSDP会话的终点。注意自动-RP配置命令引用了LO5，**ip msdp originator-id**命令引用了LO0。LO0配置非常重要，因为MSDP在对等会话终点必须使用唯一的IP地址。

OSPF和BGP章节中的配置被显示是因为重要的原因。回忆一下，OSPF和BGP使用配置在任何loopback上的最高IP地址作为路由器ID。不幸的是每个路由器上的LO5的地址高于LO0的地址。结果是每个路由器上OSPF和BGP进程试图使用10.100.254.1作为路由器ID。当路由器的LSA试图覆盖其他路由器的LSA时，一些不希望的结果会使OSPF数据库振荡。一个解决方案是选择数值上低于其他环回地址的虚拟RP地址，但是该方法在这里会对无意的配置错误防范过于脆弱，所以不可行。在这个例子中使用了更好的方法，在OSPF和BGP配置中使用router-id命令强迫路由器使用LO0地址作为路由器ID。

注意LO0接口没有运行PIM。该接口对PIM功能来说并不需要，路是作为MSDP对等关系的路由器特定IP地址。

7.9　案例学习：MSDP缺省对等实体

如果AS是末梢AS或非传输AS，特别是AS是非多宿时，没有理由与传输AS运行BGP

协议。在末梢 AS 中使用一个静态缺省路由，在传输 AS 时用静态路由指向末梢前缀通常就足够了。但是如果末梢 AS 同时是多播域，它的 RP 必须与相邻域建立对等关系时会怎样呢？对 MDSP 操作稍加观察就可以发现 MSDP 对对端 RPF 检查时依赖于 BGP 下一跳数据库。

你可以使用 **ip msdp default-peer** 命令禁止这种对 BGP 的依赖。MSDP 将接收缺省对端的所有 SA 消息。图 7-18 示意了一个简单例子，如图所示，这是末梢 AS 与传输 AS 间没有运行 BGP 的典型例子，但是可能对 MDSP 产生问题。这里末梢 AS 域传输 AS 在一条单一链路上建立对等关系。由于只有一条路径不可能出现环路，所以不需要 RPF 检查。

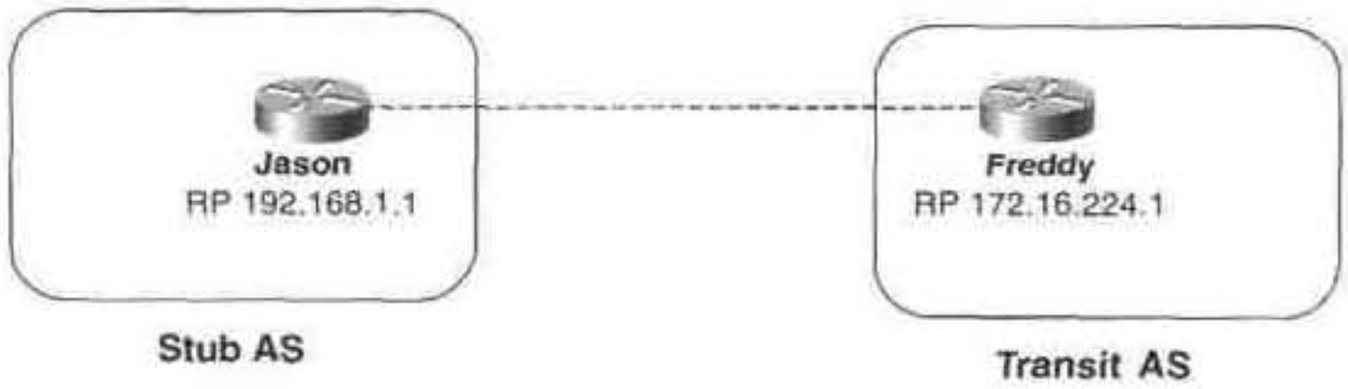


图 7-18　末梢 AS 与传输 AS 间没有运行 BGP 的典型例子

例 7-19 显示了两个路由器上的 MSDP 配置。

例 7-19　路由器 Jason 和 Freddy 的 MSDP 配置

```
Jason
ip msdp peer 172.16.224.1 connect-source Loopback0
ip msdp default-peer 172.16.224.1
```

```
Freddy
ip msdp peer 192.168.1.1 connect-source Loopback0
ip msdp default-peer 192.168.1.1
```

出于冗余考虑，末梢 AS 可能希望与多于一个 RP 建立 MSDP 对等关系。如图 7-19 所示。由于没有 RPF 检查机制，不能直接接收缺省对端的 SA 消息。代替的方案是 SA 消息只从一个对端接收。如果该对端故障，则从另一对端接收。这里有一个需要强调的假设，两个缺省对端发送的是否是相同的 SA 消息。

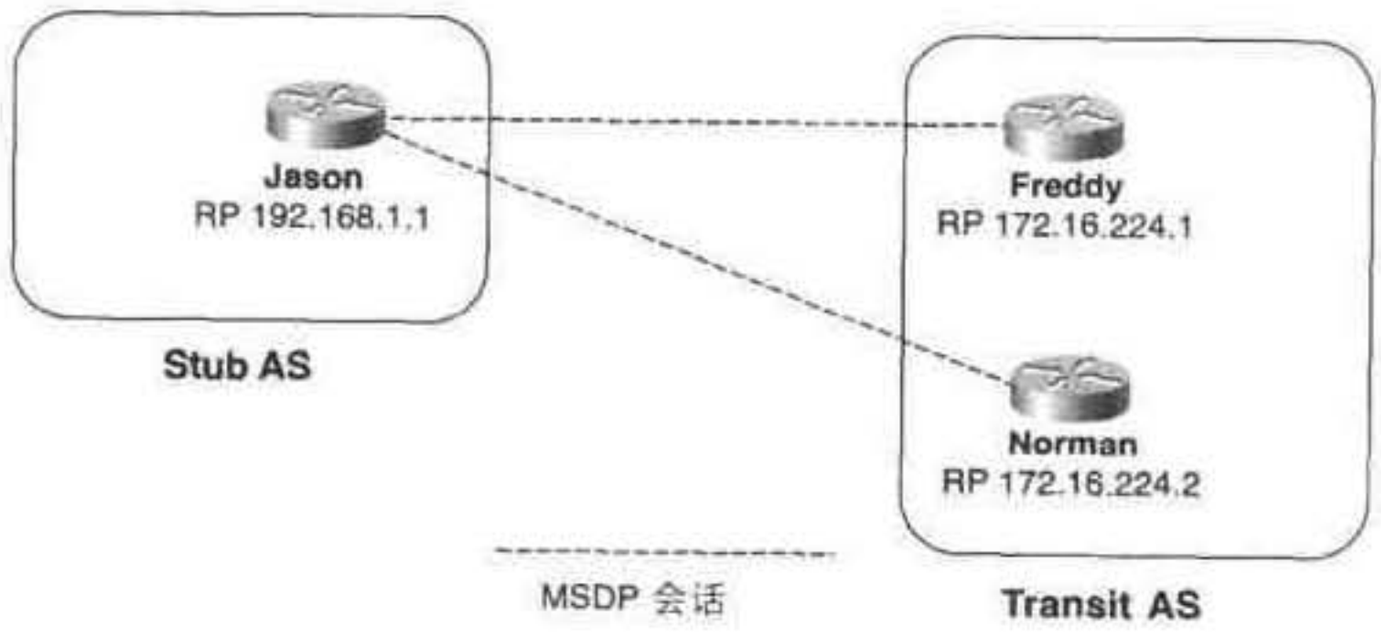


图 7-19　Jason 连接到多于 1 个 MDSP 对端

例7-20显示了Jason的配置。

例7-20　配置Jason以在Freddy和Norman间建立完全的对等关系

```
ip msdp peer 172.16.224.1 connect-source Loopback0
ip msdp peer 172.16.224.2 connect-source Loopback0
ip msdp default-peer 172.16.224.1
ip msdp default-peer 172.16.224.2
```

在一般环境下，激活的缺省对端是配置中先出现的对端。在这个例子里是172.16.224.1。除非172.16.224.1故障，Jason不会接受从172.16.224.2发来的SA消息。

如图7-20所示，传输AS中的RP更有可能拥有多个缺省MSDP对端。因为只能从一个RP接收SA消息，所以只是像上一个例子中简单罗列缺省RP将不起作用。使用BGP风格的前缀列表使RP能在忽略对端RPF检查条件下接收SA消息并防止环回。RP从所有的缺省对端接收SA消息，但只接收每个对端所关联的前缀列表所允许的源地址前缀。这里缺省的假设是每一个AS使用独特的前缀，这样可以确保防止环回。

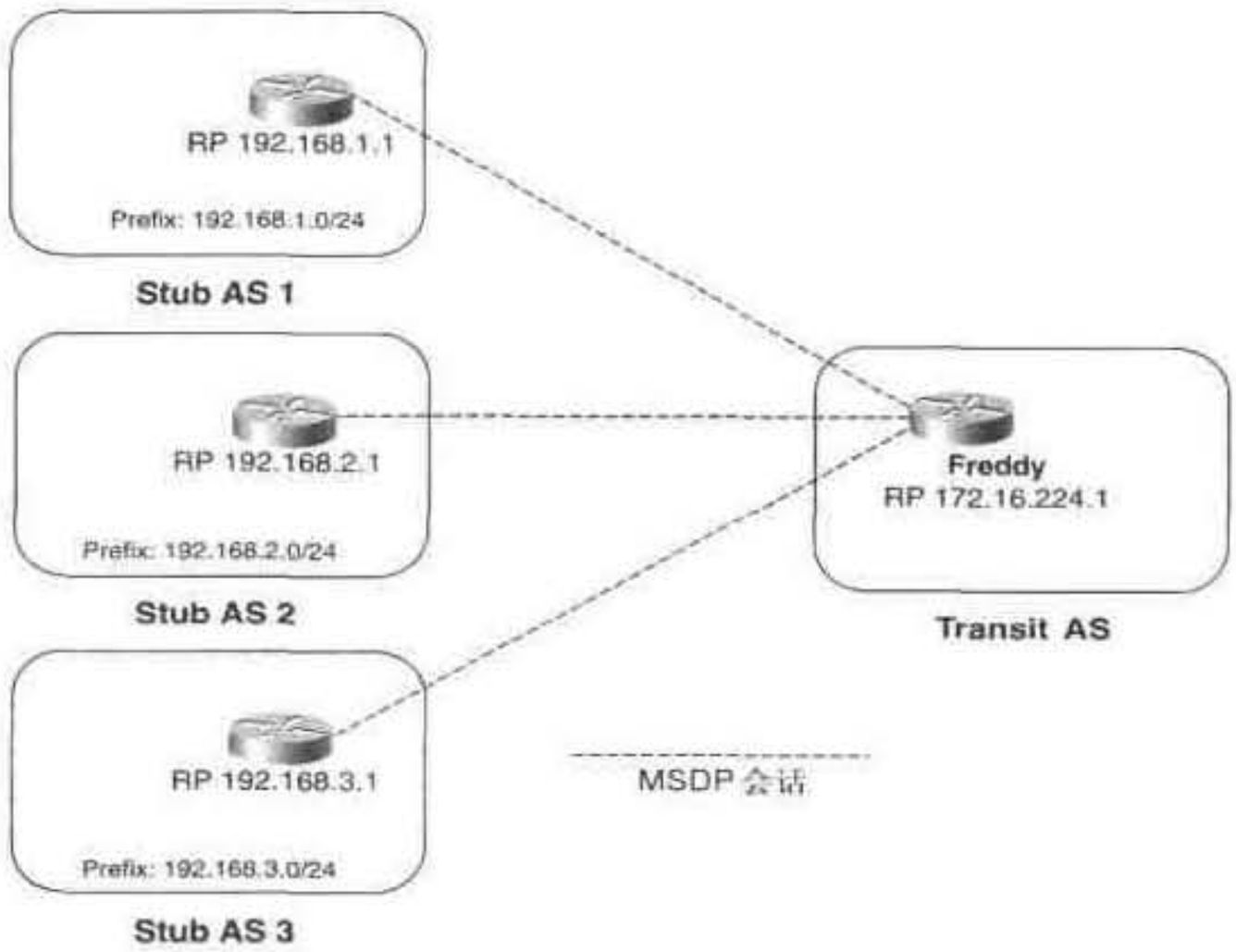


图7-20　传输AS中的RP拥有3个缺省MSDP对端

例7-21显示了Freddy的配置。

例7-21　配置RP从多个MSDP端接收SA消息

```
ip msdp peer 192.168.1.1 connect-source Loopback0
ip msdp peer 192.168.2.1 connect-source Loopback0
ip msdp peer 192.168.3.1 connect-source Loopback0
ip msdp default-peer 192.168.1.1 prefix-list AS1
ip msdp default-peer 192.168.2.1 prefix-list AS2
ip msdp default-peer 192.168.3.1 prefix-list AS3
!
ip prefix-list AS1 seq 5 permit 192.168.1.0/24 le 32
ip prefix-list AS2 seq 5 permit 192.168.2.0/24 le 32
ip prefix-list AS3 seq 5 permit 192.168.3.0/24 le 32
```

7.10　命令归纳

表 7-6 罗列和描述了本章所讨论的命令。

表 7-6　　命 令 汇 集

命　令	描　述
address-family {ipv4 | vpnv4} {multicast | nunicast | vrf}	使能 MBGP
exit-address-family	标记 MBGP 配置小节的结束
ip mroute *source mask [protocol as-number] {rpf-address | type number} [distance]*	配置 RPF 检查使用的静态多播路由
ip msdp cache-sa-state [list *access-list-number*]	使能 SA 缓存
ip msdp default-peer *ip-address | name* [**prefix-list** *list*]	指定不需要 RPF 检查就可以接受 SA 消息的 MSDP 对端
ip msdp description *{peer-name | peer-address } text*	对 MSDP 对端配置增加描述
ip msdp filter-sa-request *{ip-address | name }* **[list** *access-list-number*]	对发送的 SA Request 消息使能过滤器
ip msdp mesh-group *name {ip-address | name }*	指定某一对等体成为全连接组的成员．全连接组成员的 SA 消息在全连接组不做转发
Ip msdp originator-id *type number*	改变 MSDP 消息中使用的缺省 RP 地址
Ip msdp peer *{peer-name | peer-address}* **[connect-source** *type number*] **[remote-as** *as-number*]	指定路由器作为 MSDP 对等体
Ip msdp redistribute [list *access-list-name*] [**asn** *aspath-access-list-number*] [**route-map** *map*]	对由本路由器发起的 MSDP SA 消息使能过滤器
Ip msdp sa-filter *in {ip-address | name}* **[list** *access-list-name*] [**route-map** *map*]	对收到的 MSDP SA 消息使能过滤器
Ip msdp sa-filter *out {ip-address | name}* **[list** *access-list-name*] [**route-map** *map*]	对发送的 MSDP 消息使能过滤器
Ip msdp sa-request *{ip-address | name }*	使能 RP 向缓存对端发送 MSDP SA Request 消息
Ip msdp ttl-threshold *{ip-address | name }* **ttl**	对本路由器发起的 MSDP 消息设置 TTL 值
Ip multicast boundary *access-list-number*	指定接口作为管理范围的多播边界
Ip multicast ttl-threshold *ttl-value*	指定接口作为 TTL 范围的多播边界
Neighbor *{ip-address | name }* **activate**	指定激活成单播、多播或两者皆可的邻居
No bgp default ipv4-unicast	禁止将所有的 BGP 邻居缺省指定为单播，这样可以独立激活多播

续表

命　令	描　述
show ip msdp peer *peer-address* |*name*	显示 MSDP 对端的详细信息
Show ip msdp sa-cache [*group-address* | *source-address* | *group-name* | *source-name*] [*group-address* | *source-address* | *group-name* | *source-name*] [*autonomous-system-number*]	显示环存在本路由器中的 SA 状态信息
Show ip msdp summary	显示关于 MSDP 对端的总结信息

7.11　尾注

[1]David Meyer, "RFC 2365: Administratively Scoped IP Multicast" (工作正进行中)

[2]Tony Bates, Ravi Chandra, Dave Katz 与 Yakov Rekhter, "RFC 2283: Multiprotocol Extensions for BGP-4" (工作正进行中)

[3]Dino Farinacci et al., "Multicast Source Discovery Protocol (MSDP)" draft-ietf-msdp-spec-05.txt, 2000 年 2 月

[4]Dorain Kim, David Meyer, Henry Kilmer 与 Dino Farinacci, "Anycast RP Mechanism Using PIM and MSDP" draft-ietf-mboned-anycast-rp-05.txt, 2000 年 2 月

7.12　展望

到现在为止，你已将不短的时间不仅投入到是 IP 路由本身，而且涉及复杂性不断增长的现代 IP 网络路由所呈现的问题。这些问题的解决许多都涉及到 IPv4 以及相关路由协议本身的缺陷。下一章将展示最新版本的 IP 协议——IPv6。在设计 IPv6 时总结了许多 IPv4 的教训。大多数人看 IPv6 时仅看到的 128 比特的地址空间来缓解 IPv4 地址短缺。然而你将可以看到 IPv6 做了更多。IPv6 被设计成更好的安全性、更好的 AS 间质量以及更好的多播支持，同时减少了许多 IPv4 不必要的复杂性。

7.13　复习问题

1. 在多播范围限制一章中，有一个配置举例是通过管理的范围限制。在边界接口 E0 阻塞所有组织内部的包(目的地址前缀匹配 239.192.0.0/14)，允许全局范围包通过。组地址是 224.0.0.50 的包是否能穿过边界？

2. 在配置了运行 PIM 的点到点接口和多路访问接口上 Cisco IOS 软件如何处理 DVMRP 剪除消息？

3．为什么 Cisco IOS 软件接收 DVMRP 消息却不应答？

4．PIM (*,*,RP)条目是什么？

5．多协议 BGP(MBGP)与普通 BGP 有什么区别？

6．什么是 MBGP AFI？

7．下边的说法对吗？MSDP 在不同 PIM 域的 RP 间携带多播源和组成员信息。

8．MSDP 的传输协议是什么？

9．什么是 MSDP SA 消息？

10．MSDP RP 如何确认 RPF 接口上是否收到 FA？

11．什么是 SA 缓存？

12．不使用 SA 缓存时能否降低接入时延？

第 8 章　IPv6

本章覆盖下列关键主题

- **IPv6 设计目标**——本节覆盖 IPv6 设计目标：可扩展的网络，配置简单，维护安全。
- **当前 IPv6 状态**——本节覆盖可用的标准和草案，厂商支持和计划支持。客户驱动的应用的实现。
- **IPv6 包格式**——本节覆盖 IPv6 包格式，包含扩大、层次化的 IPv6 地址和允许路由器有效处理包的包头。
- **IPv6 功能**——本节覆盖 IPv6 功能，包括自动路由器发现、死亡路由器检测、自动主机配置和多播能力。
- **从 IPv4 向 IPv6 过渡**——IPv4 将存在很长时间。本节覆盖了过渡机制(包括双协议栈、隧道和地址/协议翻译)来帮助过渡和共存。

IPv6 被设计成继 IPv4 后的下一代 IP 协议。版本 5 已经被赋予另一个互联网协议：RFC 1190 中定义的互联网流协议版本 II。IPv6 协议本身的详细描述已超出本书范围，本章彻底概述 IPv6 网络设计目标以及该协议当前状态。本章还覆盖 IPv6 的包格式、IPv6 的功能以及从 IPv4 互联网络过渡到 IPv6 的互联网络的方法。本章中 Cisco 路由器的配置使用的是基于版本 IOS 12.0 的测试版软件。在最终版本的代码中，命令可能被增加、修改或删除，输出内容也可能被改变，尽管变化可能会很小。Cisco 宣称正式代码将在 Cisco IOS 软件版本 12.1 中正式发布。

8.1　IPv6 的设计目标

互联网是一个巨大的成功，成功来源于互联网络的合作精神。当前几乎所有公司都有 Web 站点(甚至可以在酒瓶的包装上找到 URL)。电子邮件已经和电话一样成为重要的商务工具。但是 IPv4 上的某些设计为互联网可以增长到多大设定了上限。32 比特的地址空间限定了全球范围内所连接的可路由主机数量以及网络层次的数量。正如你在本书其他章节所看到的，可扩展的互联网络需要层次化的路由。要使网络像应用开发者和互联网使用者所梦想的那样使用与规模无关，必须严格地维护路由分层。为维护层次结构，连接在互联网的站点必须符合寻址和汇聚规则。连接到 ISP 或交换点的站点必须使用分配给 ISP 或交换点的地址中分配出来的地址。这意味着重新分配地址，在第 2 章中表述的先天困难将继续存在。

互联网络的成功还提高了对数据完整性、认证和安全的要求。

IPv4 网络设计者使用一些不同的技术缓解了其中一些问题需求。如第 4 章所讨论：网络内部可以使用专用地址，使用地址翻译来与互联网上其他公司通信，允许大量的节点访问外部网络，从而缓和地址空间问题。然而 NAT 的实现和维护并不非常容易。某些应用对 NAT 设备处理有过多的要求，某些应用甚至无法工作。更糟的是一些未来应用例如个人数字助理、

家庭安全系统或汽车维护计算机等，可能要求分配全球可寻路的 IP 地址，这样他们才能从互联网站点上被访问。

基于类的地址所引入的 IPv4 分层的严重问题已经由第 2 章所讨论的 CIDR 技术的实现而缓和。CIDR 允许你更有效地分组或分类，但是 32 比特地址空间所引入的总层次限制还是存在。

IPv6 的地址空间如此之大，足够满足全球可路由地址快速增长的需求和网络更多的层次结构。地址空间增长到 128 比特。分层结构已设计在全球可路由地址格式内部。

ISP 为客户分配一系列地址，如果客户想要更换 ISP 则极有可能需要对网络重新编址。IPv4 网络设计者实现了动态主机配置协议(DHCP)来减少对 PC 的地址分配负担。DHCP 很成功，并且可能在 IPv6 中继续使用。IPv6 主机可以使用 DHCP 或者内建的自动配置机制来配置自身地址。两种方式都允许 IPv6 主机使用新地址建新连接同时保持旧地址的现存连接的能力。该能力维持两个地址来确保到新网络前过渡的平滑性。

8.1.1　提高可扩展性

你在第 2 章已看到 IPv4 地址如何限制互联网络的可扩展性。本节回顾这些可扩展性问题。IPv4 第一个问题是 32 比特地址空间限制的问题，该问题是设计新协议的主要驱动力。专家预测，如果不加干涉则 IPv4 地址将在 20 世纪 90 年代中耗尽。上述预测并没有成为现实。NAT 通过允许企业使用隐藏的专用地址延长了 IPv4 的使用。IPv6 的 128 比特地址允许更多的全球可路由地址连接到互联网。IPv6 中也定义了专用地址。

IPv4 的另一个问题是巨大的互联网路由表。CIDR 通过聚合地址来引入更多的层次，从而减小路由表。然而很多地址无法被聚合。例如在 CIDR 以前分配的地址以及由多个互联网连接的网络使用的地址无法聚合。

吸取从 IPv4 得到的教训，IP6 被设计成可扩展，易配置，适用安全的网络。IPv6 的设计没有试图解决互联网路由表爆炸的问题。因为从一开始就严格执行的分配规则和程序以及设计的层次格式，使路由表大小是可控制的。设计目标是尽可能聚合的全球可路由地址空间格式以帮助达到控制路由表大小的目的。

8.1.2　易于配置

IPv6 引入一种机制使主机与路由器间的通信管理以及主机的配置更容易。这些机制对 IPv6 的成功非常重要。由于越来越多的人、公司和学校希望连接到互联网并建立自己的内部互联网络，相关的工作必须被简化。并不是每个人都希望成为 CCIE，但必须使他们能够运行网络。他们只希望网络能够工作。IPv6 的自动配置机制能使主机获得 IP 地址、发现邻居和默认路由器，并有效地使用多个默认路由器达到冗余目的。

连到互联网上的大公司希望能够灵活地更换服务提供商并且不引起网络混乱。在 IPv6 中同样需要改变网络地址。但是由于所有的节点能够同时维护多个地址，并保持两个地址：一个新的地址和另一个逐步停止使用的地址，所以改变网络地址变得简单的多。另外，网络前缀是由路由器宣告到主机，使主机能够自动配置 IPv6 地址。如果公司由于更换 ISP 需要改变网络地址，可以将路由器配置成同时宣告新旧前缀。收到宣告信息的主机能使用新前缀信息自动配置，并在建立新连接时使用新的地址。现存的连接使用旧的地址。

8.1.3　安全性

个人和公司都不希望为安全性担心。他们希望数据是安全的，不需要多加考虑。IPv6 中内建了认证和加密。在网络协议中 IPv6 包可以在网络层受保护。

8.2　当前 IPv6 状态

对大多数组织来说，IPv6 只不过是他们谈到网络时提及的一组新的数字和字母。然而现在绝大多数规范已经完成了，一些是草案标准，更多的是提交的草案标准。IANA 将地址空间分配给地域性的互联网注册机构(RIR)，RIR 已经开始向互联网提供商分配地址。网络和终端设备厂商已开始发布支持 IPv6 的软件，或宣布在短期内支持的计划。存在一个叫 6bone 的实验网允许各个组织测试他们的 IPv6 实现、学习如何过渡网络以及习惯 IPv6 的网络管理。还有一个公众的产业网：6REN，为教研机构研究 IPv6 网络产品作支持。由于 IPv6 网络变得越来越可用并且易于实现，网络计划者开始考虑 IPv6 网络。CCIE 的候选者应当准备好解决 IPv6 问题。

8.2.1　IPv6 规范(RFC)

IPv6 规范现在是批准的草案标准。已经有公司已发布(预发布)支持该规范的产品。当前的草案标准包括：

- RFC2373 ：IP Version 6 Addressing Architecture
- RFC2374: An IPv6 Aggregatable Global Unicast Address Format
- RFC2460: Internet Protocol, Version 6 (IPv6) Specification
- RFC2461: Neighbor Discovery for IPv6
- RFC2462: IPv6Stateless Address Autoconfigureation
- RFC2463: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification

许多公司提出的待批准的草案标准有：

- RFC1886: DNS Extension to Support IP Version 6
- RFC1887: An Architecture for IPv6 Unicast Address Allocation
- RFC1981: Path MTU Discovery for IP Version 6
- RFC2080: RIPng for IPv6
- RFC2473: Generic Packet Tunneling in IPv6 Specification
- RFC2526: Reserved IPv6 Subnet Any cast Addresses
- RFC2529: Transmission of IPv6 over IPv4 Domain Without Explicit Tunnels
- RFC2545: Use of BGP-4 Multiprotocol Extension for IPv6 Inter-Domain Routing
- RFC2710: Multicast Listener Discovery(MLD)for IPv6
- RFC2740: OSPF for IPv6

由于存在许多其他提交的草案标准和发布的草案文档，另外还可能有更多的文档和标准

会出现在不远的未来，所以不可能将他们一一罗列。你可以在 www.isi.edu 或者其他 RFC 站点查找 RFC 文档。

8.2.2　厂商支持

相对于协议标准化进程，IPv6 的协议开发以及相关部件还做得很不够，所以许多厂商都承诺作开发和测试项目。Cisco 当前在一个基于 12.1 的测试版本 IOS 软件中支持 IPv6。他们宣称在稍后的 IOS 12.1 正式发布版本中支持 IPv6。微软和 SUN 公司已经有终端设备的 IPv6 协议栈。并不是所有的厂商都支持 IPv6 协议所有的部件。一些厂商在等待标准的成熟，另一些厂商在准备开发以前等待客户的需求。许多正在到来的大范围的应用例如手提计算机等，都需要 IPv6，所以厂商至少要有一个迅速实现 IPv6 的计划。当前 Cisco 的 IPv6 实现包括下列特性：

- RIPng
- IPv6 的 BGP-4+
- IPv6 静态路由
- 流量过滤
- 自动和静态隧道
- EUI-64 地址
- 邻居发现
- 以太网、FDDI、Cisco HDLC 和 ATM PVC 中封装 IPv6
- 对 Telnet、DNS 和 TFTP 的双协议支持
- ICMPv6 和 Ping
- trace route 和 debug 命令

8.2.3　实现

有两个供公共使用的 IPv6 实现。一个是 6bone，是 IPv6 的实验床。可以使用 6bone 来测试协议实现、IPv4 到 IPv6 的过渡和可运行的程序等。另一个网络：IPv6 研究和教育网(6REN)，为不同组织提供可运行的 IPv6 网络来连接其他的 IPv6 网络。上述两个网络实现都进入了 IPv6 的开发过程，为厂商和网络设计师提供一个大范围的平台来测试软件、网络配置、设计方案并且让他们熟悉和理解该协议。

1．6bone

6bone 是一个全球范围的 IPv6 网络，用来对 IPv6 设备和网络作测试和预实现。当前该网络支持着 39 个国家中的 260 个组织。6bone 被设计成全球性层次化的 IPv6 网络。它包含伪顶级(1 级)穿越服务提供商，伪二级(2 级)穿越服务提供商和伪站点级组织。伪顶级提供商是全世界互连的组织。顶级提供商使用 BGP 的 IPv6 扩展相互通信。二级提供商同样使用 BGP 连接到本地的顶级提供商，站点级组织连接到二级提供商。站点级组织可以使用静态路由或者 BGP-4 连接到二级提供商。最初的连接使用 IPv4 隧道在互联网上传输。渐渐地，非隧道的 IPv6 连接也在建立。6bone 已经被证明是 IPv6 标准和产品有效的实验床设施。现在 IPv6 也用作测试网络过渡和可运行的程序。图 8-1 显示了 6bone 骨干网。

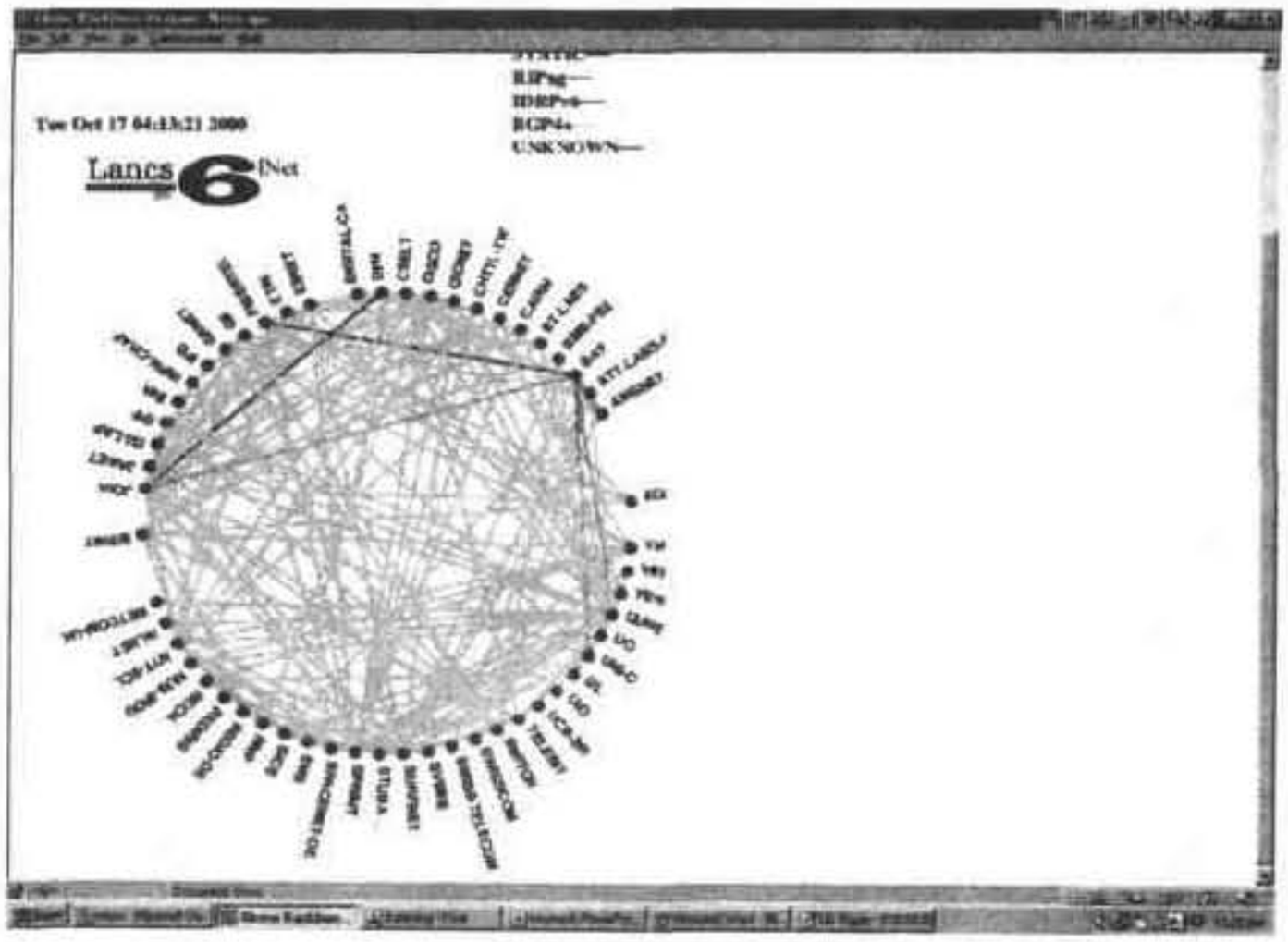


图 8-1　6bone 骨干网

2．6REN

6REN 是自发建立的，初衷是建立提供 IPv6 穿越服务的 IPv6 生产运营网。在 Web 站点 www.6ren.net/overview.htm 中 6ren 声明，6REN 的目的是：

……提供 IPv6 穿越服务产品来促进 IPv6 网络高质量、高性能、可操作的健壮性。

研究、教育机构、盈利组织和非盈利组织都可以获得穿越服务。网络通过 ATM 承载纯 IPv6。6REN 提供到 6bone 成员的连接。

8.3　IPv6 包格式

IPv6 包格式中包含一个巨大的层次化的地址和使路由器能有效处理该包的包头。即使考虑到互联网的增长以及多层的分级结构，上述地址也足以应付。

IPv4 头中的一些字段在 IPv6 中被删除。对可选字段以及所有路由器必须处理选项的需求的删除能使包的处理变得更有效率。网络选项仍然存在，然而在大多数情况下仅由目的地节点处理。IPv6 将选项移到扩展头，该扩展头仅由需要该信息的节点处理。

IPv6 地址类型、使用和结构以及 IPv6 的头将在下面章节详细讨论。

8.3.1　IPv6 地址

当 IP 在 20 世纪 70 年代发展时，互联网世界与现在完全不同。互联网被用作研究和教育。32 比特的地址看起来完全超过了全世界的需求，至少是在 IP 生存周期内。但互联网是如此成功，以至于 IP 被内嵌到许多商业和家庭中。IP 网络是许多成功的商业组织日复一日正常运行的完整一部分。随着组织的成长以及需要连入互联网的站点的增加，需要更可扩展的解决方案。有如此多的设备使用(或再不远的将来会使用)IP 地址接入互联网，以至于即使存在被开发用来延缓 IP 地址耗尽的方法例如 NAT 和 CIDR，IP 地址也将变得捉襟见肘。手机、PDA、E-mail 设备、家庭照明系统、汽车以及应用仪表都将拥有 IP 地址。IP 的初始设计已经

无法应付如此成功的应用。

IPv6 的地址解决了最紧迫的问题。它增加了地址空间并且定义了多个地址级别。为保持层次化聚合能力，IPv6 协议还定义了管理地址分配的规则。

8.3.2　地址空间

如此广泛使用的地址恰当的大小究竟是多少呢？地址空间应当是固定的还是可变的？太小的地址空间会限制可扩展性。太大的地址会使包头过大，使路由器很难管理。如果地址长度变长，则会使软件过于复杂，影响包处理速度。下一代 IP(IPng)中有一个提案建议使用网络服务接入点(NSAP)地址，NSAP 地址可以由 1 字节到 20 字节长。另一个提案建议使用 64 比特地址。尽管 64 比特的地址空间看起来对 IP 节点足够，还是加上的附加的比特来处理由增长的网络所引起的不断增长的附加的复杂性。为了将来网络的扩展以及地址分配的层次化，选择了 128 比特地址空间。

理论上说 128 比特的地址空间可以容纳 340，282，366，920，938，463，463，374，607，431，768，211，456 个节点。如果世界上人口总数是 10,000,000,000，大约每人分到 3.4×10^{27} 个地址。即使是未来的高度连接的远程工作，包括互联网无线电话，IP 手表，有路由的厨房用具和应用工具的家庭网络，配置网络的汽车，都用不完这么多地址。

8.3.3　地址的文字表示

IPv6 地址长 128 比特，可以书写成由冒号隔开的 8 个 16 比特段。每一段表达成 4 个 16 进制数字。下面有两个使用上述方法表达的地址：

- FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
- 1080:0000:0000:0000:0008:0800:200C:417A

IPv6 地址空间非常大，能为许多节点寻址并提供灵活的层次结构，但是很难写下来，更不用提记住了。有几种方法能压缩地址，为人工处理提供一点点方便性。

地址里通常会有很多 0。在每个 16 比特段中你可以省略前面的 0，每个 16 比特段至少包含一个 16 进制数(只有一个例外)。上面地址 1 没有可省略的 0，所以不能压缩。地址 2 可以压缩成：

- 1080:0:0:0:8:800:200C:417A

你甚至可以压缩多个连续为 0 的段。这就是上边所说的至少一个数字代表本段的例外。你可以使用两个冒号(::)来替代多个为 0 的段。表 8-1 显示了一些地址压缩。

表 8-1　　显示使用(::)压缩地址中多个为 0 段的举例

压缩前的地址	压缩后的地址
1080:0000:0000:0000:0008:200C:417A	1080::8:800:200C:417A
1080:0000:0000:3245:0000:0000:200C:417A	1080:0:0:3245::200C:417A
0000:0000:0000:0000:0000:0000:0000:0001	::1
0000:0000:0000:0000:0000:0000:0000:0000	::

连续或字段前的 0 可以被压缩。注意::只能替代一组连续的 0。多个::会使地址出现二义

性。例如地址 1080::3245::200C:417A 没有提供足够的信息来确定 0x3245 的位置。

在某些混合 IPv6 和 IPv4 的模式中，你可以使用另一种地址表达方式。该格式混合使用由冒号隔开 16 进制数的 IPv6 方式与由点隔开的十进制数的 IPv4 方式，如下所示：

X:X:X:X:X:X:d.d.d.d

XXXX 表示 IPv6 的十六进制数字，d.d.d.d 使用点隔开十进制数字的方式表示最后 32 比特。表 8-2 显示混合模式格式的一些例子：

表 8-2　　IPv6 和 IPv4 混合环境中的值的文字表述

压缩前的地址	压缩后的地址
0:0:0:0:0:0:13.1.68.3	::13.1.68.3
0:0:0:0:0:0:129.144.52.38	::129.144.52.38

混合 IPv4/IPv6 的表示提供一种 IPv6 节点与 IPv4 节点在同一网段共存的方式。在本章的“从 IPv4 转变成 IPv6”节中讨论了共存与转变方式。

8.3.4　地址前缀的文字表示

IPv6 地址前缀的表达方式类似于 IPv4 CIDR 表示方法。IPv6 前缀表示如下所示：

IPv6 地址/前缀长度

其中：

*IPv6 地址*是任何有效的地址

*前缀长度*构成前缀的连续比特数

下面的前缀是 56 比特前缀 200F00000000AB 的有效文字表示：

200F::AB00:0:0:0:0/56

200F:0:0:AB00::/56

注意作为 IPv6 主机地址，双冒号(：：)在每个表示中只能出现一次。

下面是 56 比特前缀 200F00000000AB 的无效表示：

200F:0:0:AB/56

200F::AB00/56

200F::AB/56

第 1 个表示无效是因为 16 比特段中后面比特的省略以及地址长度无效。在(/)左边的地址必须是有效的全长或压缩的 IPv6 地址。第 2，3 个表示是有效的 IPv6 压缩地址，但是扩展以后不是原来的地址：

源地址是 200F:0000:0000:AB00:0000:0000:0000:0000，扩展以后分别是

200F:0000:0000:0000:0000:0000:0000:AB00 和

200F:0000:0000:0000:0000:0000:0000:00AB。

8.3.5　地址类型分配

在 CIDR 以前，IPv4 地址的最高几个比特定义了类型——A、B、C、D 和 E 类。地址的

类型标识了一个固定长度的网络部分和固定长度用户可以随意使用的主机部分。这是 IPv4 地址唯一定义的结构。IPv6 地址定义了更多的结构。结构在下一章“地址结构”中详细讨论。地址中的高比特定义了 IPv6 的地址类型。组成这些比特的可变长度域称为格式前缀(FP)。表 8-3 显示了这些比特的初始分配。

表 8-3　　IPv6 前缀分配

分　　配	前缀(二进制)	地址空间比例
保留	0000 0000	1/256
未分配	0000 0001	1/256
为 NSAP 保留	0000 001	1/128
为 IPX 保留	0000 010	1/128
未分配	0000 011	1/128
未分配	0000 1	1/32
未分配	0001	1/16
可聚合全球单播地址	001	1/8
未分配	010	1/8
未分配	011	1/8
未分配	100	1/8
未分配	101	1/8
未分配	110	1/8
未分配	1110	1/16
未分配	1111 0	1/32
未分配	1111 10	1/64
未分配	1111 110	1/128
未分配	1111 1110 0	1/512
本链路单播地址	1111 1110 10	1/1024
本站点单播地址	1111 1110 11	1/1024
多播地址	1111 1111	1/256

地址空间被分配用作可聚合全球可路由地址，本地使用地址和多播地址。当前已分配大约 15%的地址空间。

有一些空间被保留。地址空间中还留有将来实现到 NSAP 与 IPX 的路由。一部分保留的空间，特别是 0x00 用作特殊地址。特殊地址有环回地址，未指定地址等，这些都将在下一节讨论。

许多地址空间还没有分配。这些都可以在将来用作现有地址的补充或新的用途。

可聚合全球单播地址类似于全球可路由 IP 地址。他们被成块分配给互联网提供商或交换点，由交换点或提供商将地址分配给公司或者用户。该类地址被定义成多级层次和聚合。这是已分配最大的地址空间，仅使用了 1/8 的可用地址空间。

由表 8-3，你可以看到多播地址由 0xFF 开始。由任何其他数开始的地址都是单播地址。表 8-4 是确定 IP 地址类型的快速索引。

表 8-4　　定义地址类型的数字

定义开始的数字	地 址 类 型
00	未指定，环回及 IPv4 兼容地址
2 3	可聚合全球单播地址
FE8	本链路
FEC	本站点
FF	多播

8.4　地址结构

IPv6 地址结构用作分配和指定地址。包转发是基于前缀的最长匹配，就和 IPv4 中一样。单播地址可能是可聚合全球地址、本链路地址、本站点地址或特殊格式地址。一个 IPv6 接口可以指定多个地址——单播地址，泛播地址或多播地址。下一节单独讨论地址结构类型。

8.4.1　可聚合全球地址格式

可聚合全球地址可以用作连接到公众互联网或者其他需要全球可路由的场合。地址结构支持现在的基于运营商的聚合以及一个新类型：基于交换点的聚合。基于交换点的聚合将地址空间分配给交换点，由交换点将地址子空间分配给他们的客户。交换点还有一个名字是 NAP，曾在第 2 章讨论。NAP 是一个二层交换机，将 ISP 和顶层服务提供商互连。NAP 可能使用路由仲裁器来收集路由信息并在服务提供商间提供对等点。基于交换点聚合的经验正在取得中。通过这些聚合点，可聚合全球地址所定义的格式能帮助互联网层次化且可扩展。层次化在下一节中详细讨论。

1．层次化

聚合地址可以组织成 3 个层次：公众、站点和接口。公众拓扑包括提供互联网传输服务的服务提供商和交换点。最高层公众拓扑组成称为无默认域——路由器的路由表中没有默认路由。这些站点明确知道如何到达其他网络前缀。站点拓扑是站点或组织内的本地概念，不提供公众传输服务，尽管可能提供小范围的内部传输服务。接口拓扑标识网络上的接口。图 8-2 显示了聚合地址的互联网层次结构。

IPS1，ISP2，ISP3 和 ISP4 代表公众空间。他们可以直接互连或通过交换点互连。小的 ISP——P1 和 P2——以及终端用户 S1 到 S5 组成站点空间。

在下一章讨论的地址格式定义体现了层次化结构。

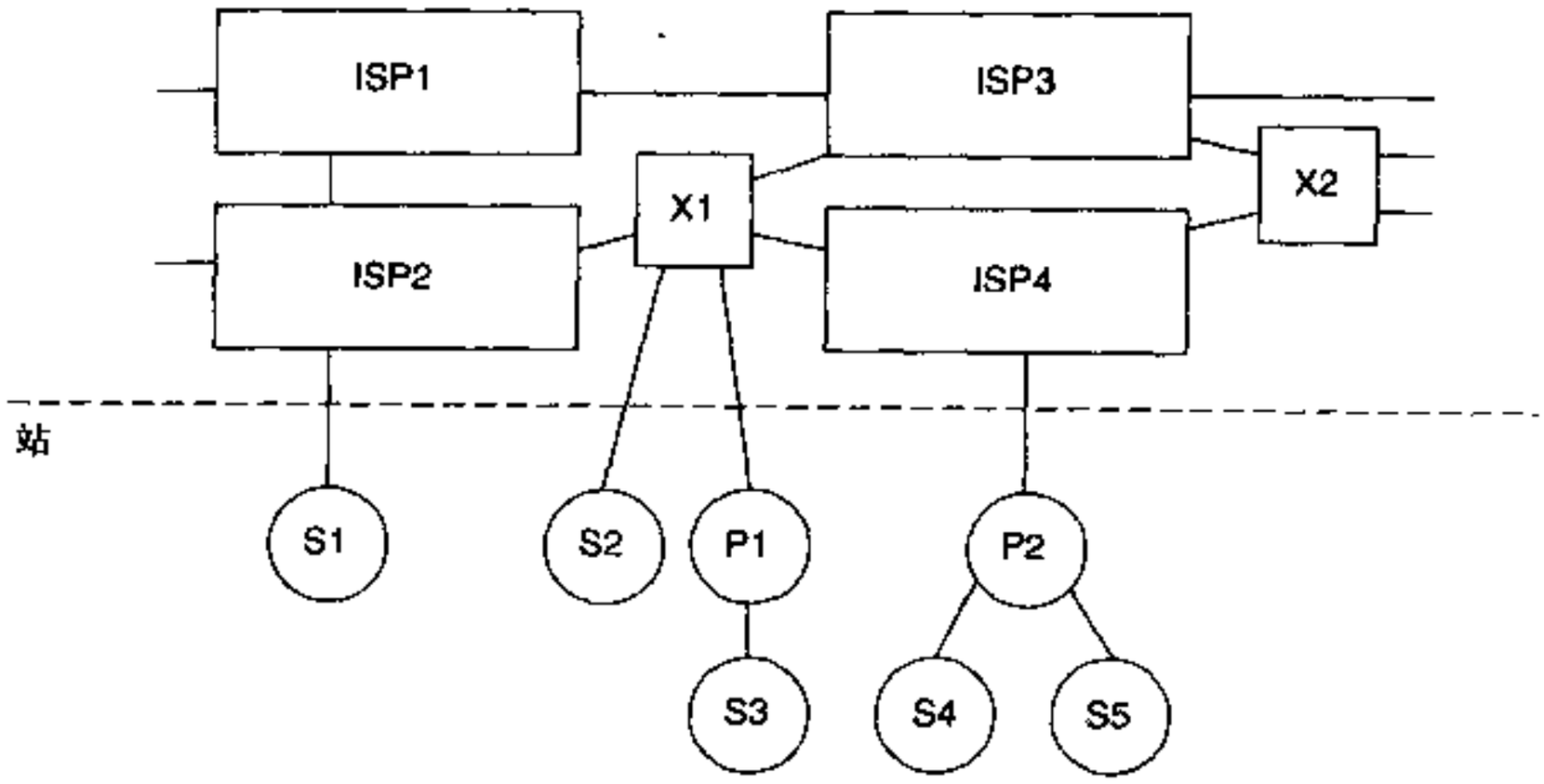


图 8-2　互联网分层结构

2．格式

图 8-3 显示了可聚合全球单播地址格式。

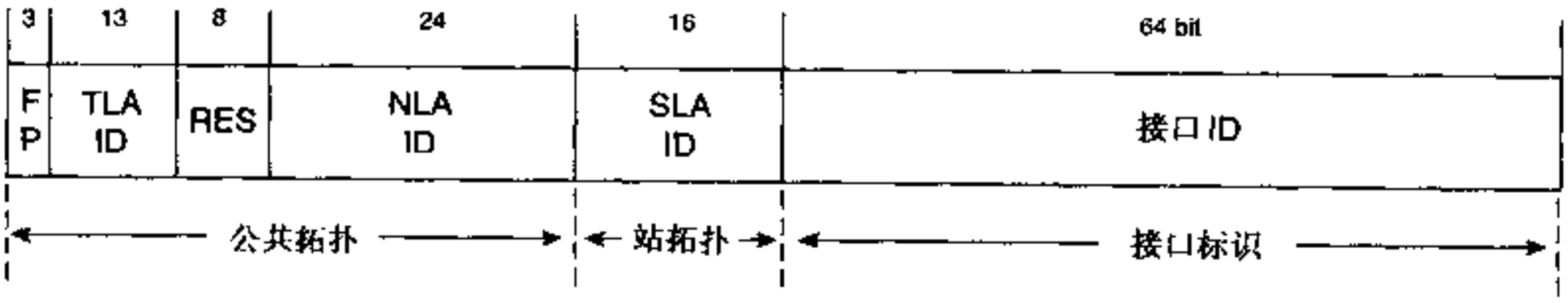


图 8-3　可聚合全球单播地址格式

组成公众层的字段是 FP,TLA,RES 和 NLA。SLA 是站点层，接口 ID 是接口层。地址的网络部分组成前 64 比特，节点部分组成后 64 比特。

地址的字段定义如下所示：

- FP 是格式前缀(001)。
- TLA 是顶级聚合标识符。
- RES 保留供将来使用。
- NLA ID 是下一级聚合标识符。
- SLA ID 是站点级聚合标识符。
- 接口 ID 是接口标识符。

格式前缀是二进制的 001，标识可聚合全球单播地址。图 8-3 显示了 FP，TLA ID，RES 和 NLA ID 组成了公众拓扑。SLA ID 组成站点拓扑，接口 ID 是接口标识。公众拓扑和站点拓扑可以再划分来产生更多的层次结构。FP,TLA,RES,NLA 和 SLA 构成地址的网络部分。接

口 ID 是节点部分。下面讨论图 8-3 中每一个字段。

顶级聚合标识符　顶级聚合标识(TLA ID)是路由层次结构中最高层。无默认域的路由器必须对每一个 TLA ID 有一个路由条目，可能还有对自身拓扑描述的路由条目。TLA ID 字段 13 个比特。如果需要更多的 TLA，可以分配另一个 FP，或者将 TLA 字段向右扩展到保留字段。

正如 IANA 将一系列地址分配给区域性的 IP 注册机构(RIR)，一系列 TLA 也被分配给全球的区域性 IP 注册机构。区域性 IP 注册机构将 TLA 顺序分配给大的 ISP，被分配 TLA 的 ISP 通常是传输服务提供商或交换点。

IANA 最初分配的 TLA 是 0x0001。该 TLA 被细分成多个子 TLA 块分配给 RIR。RIR 将分配到的块中的子 TLA 分配给提供 IPv6 服务的组织——TLA 注册者。TLA0x0001 有 13 比特的子 TLA，随后的是 6 个 TLA 保留比特，13 个 NLA 比特，16 个 SLA 比特和 64 个接口 ID 比特。每一个子 TLA 有 13 比特的 NLA ID 分配给 NLA 注册者。当 NLA 分配完以后，组织可以申请更多的地址。每一个组织都被保留有一个子 TLA 条目，但任何时候只能分配一个子 TLA 条目。如果需要更多 TLA 条目，该组织必须调整工程文档和配置计划。在子 TLA 中的 6 个保留比特可以提供更多的地址。

例如 IANA 为 ARIN 分配了 2001:0400::/23。ARIN 可以分配从 2001:0400::/29 到 2001:05FF::/29 的子 TLS。为鼓励慢启动地址申请，RIR 使用 35 比特的前缀分配地址，例如 2001:0408::/35。实际上子 TLA 是 29 比特外加 6 比特保留空间。VBSN 被分配 2001:0408::/35。VBSN 有 13 比特的 NLA 空间分配给客户。VBSN 可以使用任何可选择的方式分配空间。它可以将 2001:0408:0010::/40 分配给一个小的 ISP。上述 ISP 将 2001:0408:0010::/48 到 2001:0408:001F148 分配给客户。

保留　除了在前面章节所描述的 TLA 0x0001 外，保留字段(RES)当前没有使用。所有的比特设值为 0。将来可以用作 TLA ID 或者 NLA ID 的增长。

下一级聚合标识符　一个拥有 TLA ID 的组织可以建立地址层次结构，使用 NLA ID 来标识站点。接收 NLA 空间分配的组织称为 NLA 注册机构。上述机制类似于在 IPv4 中 CIDR 地址块分配给顶级 ISP，由顶级 ISP 将更长前缀的地址块分配给客户。

拥有 TLA 的组织有 24 比特，拥有子 TLA 的组织有 13 比特的 NLA 地址空间为客户作分配。他们的客户可以是 ISP(NLA 注册机构)或终端用户定购者。TLA 注册机构可以使用高比特的 NLA 建立层次，将低比特的 NLA 分配给 NLA 注册机构。NLA 注册机构将被分配足够多的地址，用于提供给他们的客户。NLA 注册机构的最小地址空间是 48 比特前缀，给客户至少 16 比特的 SLA 来建立自己的子网机制，一个站点可以请求多于 48 比特前缀的地址空间；但是它必须提供使用统计和工程文档或配置计划来证明对地址的需求。

站点级汇聚标识符　站点可以在所拥有的 SLA ID 中任意建立层次。它可以使用没有子网划分的扁平编号机制，或者将 SLA ID 划分成子网，将 SLA ID 地址建立层次结构。

接口标识符　接口标识符用作在指定链路上标识接口。它在链路上是唯一的。除多播之外，所有高比特是 001 到 111 间的地址必须拥有一个 EUI-64 格式的接口 ID。当前只有特殊格式地址，分配给 NSAP 的地址空间和 IPX 的地址空间不在此范围内。

如果存在接口 MAC 地址，EUI-64 地址可以由此得到。在公司 ID 和节点 ID 间插入 FFFE，并将全球/本地比特设置成 1 即可(见图 8-4)。

通过在 MAC 地址的公司标识符后插入 FFFE 并设置全球/本地比特[1]，可以将 MAC 地址

0000:0C0A:2C51 转变成 EUI-64 地址：0200:0CFF:FE0A:2C51。

MAC 地址：0000:0C0A:2C51

同一个二进制MAC地址：

00000000 00000000 00001100 00001010 00101100 01010001

在分配的公司和制造商标识值间插入FFFE

00000000 00000000 00001100 1111 1111 1111 1110 00001010 00101100 01010001

全球／本地比特设为1表示全球范围

00000010 00000000 00001100 1111 1111 1111 1110 00001010 00101100 01010001

EUI-64 标识：0200:0CFF:FE0A:2C51

图 8-4　MAC 到 EUI-64 的转换

3．特殊格式地址

一些地址使用特殊格式。这些地址从保留 FP0x00 中分配。特殊格式地址有未指定地址、环回地址和内嵌 IPv4 地址。

未指定地址

未指定地址由全 0 构成 0:0:0:0:0:0:0:0。该地址不能分配给任何节点。该地址实际上表示缺少地址。该地址的一个应用是作为正在初始化的主机的源地址：由于该主机还没有分配地址，才使用未指定地址。未指定地址绝对不能出现在 IPv6 的目标地址或 IPv6 的路由头中。

环回地址

环回地址是 0:0:0:0:0:0:1。该地址类似于 IPv4 中的 127.0.0.1。节点使用该地址给自己发包。该地址不能分配给任何物理接口。发往环回地址的流量绝对不能离开发送节点。

嵌入 IPv4 地址的 IPv6 地址

从 IPv4 到 IPv6 的一种过渡机制是使用自动隧道。在 IPv4 网络上传输时，IPv6 包自动封装到 IPv4 包中。该机制需要特殊格式的 IPv6 单播地址。支持自动隧道机制的 IPv4/IPv6 双协议栈节点必须分配一个 IPv6 地址，该地址的低 32 比特嵌入了一个 IPv4 地址。该地址其他比特都是 0。该类地址称为 IPv4 兼容的 IPv6 地址，格式如下所示：

:: d.d.d.d

d.d.d.d 是典型的 IPv4 地址表示方法。IPv4 兼容地址和自动隧道将在“从 IPv4 向 IPv6 过渡”一章中详细讨论。

4．本地使用的地址

有两种类型的地址具有本地意义。本链路地址只对节点在单一链路上有意义。本站点地址只对节点在站中有意义。这些地址并不是全球唯一的。他们只在相应范围内唯一。

本链路地址

本链路地址由节点在单一链路上使用。自动配置、邻居发现、没有路由器的链路上的节点、甚至路由协议都使用本链路地址。路由器不允许转发链路上源地址或目的地址是本链路

地址的包。所以任何需要将包发往单一链路上的设备和不希望包发出链路范围之外的协议都可以使用本链路地址。本链路地址由 FP 1111111010 定义，后面跟随 54 个 0 以及接口 ID。该地址不包含 TLA、NLA 或 SLA 信息。其中根本没有层次结构信息。图 8-5 中显示了本链路地址的结构。

10	54	64 bit
1111111010	0	接口 ID

图 8-5　本链路地址格式

每一个节点为节点上每一个激活的 IPv6 地址分配一个本链路地址。他们既可以自动配置，也可以手工配置。

下面是一些本链路地址的举例：

FE80::5ABC:01FF:FE01:1111

FE80::0060:08FF:FEB1:7EA2

FE80::200:CFF:FE0A:2C51

本站点地址

站点是一个组织或组织的一部分。站点可以是一个拓扑地点，也可以是通过某种方式互连的几个拓扑地点。一个配置本站点地址的网络不能从站点之外的地方访问。该点边缘路由器必须保证本站点的通信不出边界，并负责控制路由传播。除本站点 FP 和接口 ID 之外，本站点地址还有一个子网标识字段。注意，无论如何，该地址中不存在 TLA 或 NLA 标识。这些地址被设计成仅使用在站点内，不需要全球唯一的前缀。这些地址的使用相当于 IPv4 中的专用地址。这些地址不是全球唯一的。在路由器的接口上建议使用本站点地址，而不是全球可聚合地址。

本站点地址由 FP 1111111011 定义并后接 38 个 0，然后是 16 比特的子网字段和 64 比特的接口标识。该地址格式如图 8-6 所示。

10	38	16	64 bit
1111111011	0	子网 ID	接口 ID

图 8-6　本站点地址格式

子网字段可以用作在站点内建立多个网络并构造本地层次结构。该层次结构可用作在站点内聚合地址。本站点地址不允许广播到站点范围之外。

下面是一些本站点地址的举例：

FEC0::1:5ABC:1FF:FE01:1111

FEC0::CAB:60:8FF:FEB1:7EA2

5．泛播地址

泛播路由是对使用同一 IP 地址的多个接口的寻址机制，这些接口通常位于不同的节点。发往泛播的通信流量只路由到最近的节点。泛播功能将稍后在“泛播处理过程”一节中讨论。泛播地址与单播地址格式相同。没有为泛播地址定义特殊的 FP。

泛播地址从单播地址空间中分配。事实上，泛播地址是从接口 ID 字段中得到的。子网路由器泛播地址是预定义的，连接到某链路上的所有路由器接口必须分配该地址。上述地址是接口 ID 为全 0 的单播地址。图 8-7 显示了子网路由器泛播地址。

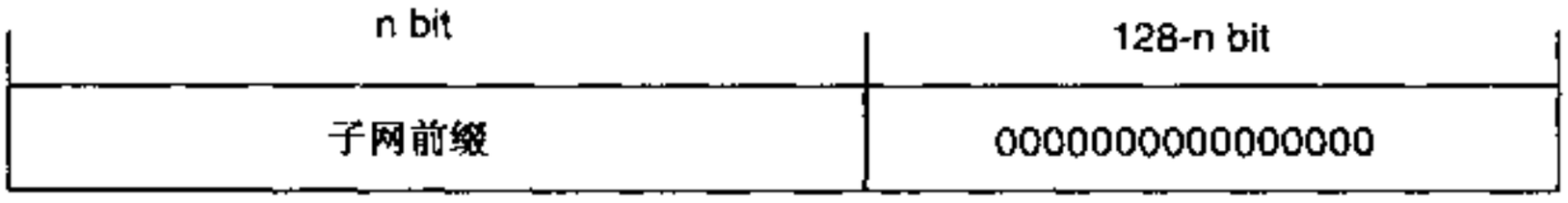


图 8-7　子网路由器 Anycast 地址

一个路由器接口上的子网路由器泛播的例子是 FEC0:0:0:A::，该接口的单播地址是 FEC0:0:0:A:200:CFF:FE0A:2C51。

最大的 128 个接口 ID 是为已分配的子网泛播地址所保留。保留的子网泛播地址是任何 IPv6 子网上都可以使用的泛播地址，无论前缀的类型或格式是什么。图 8-8 显示了保留的泛播地址的构造。

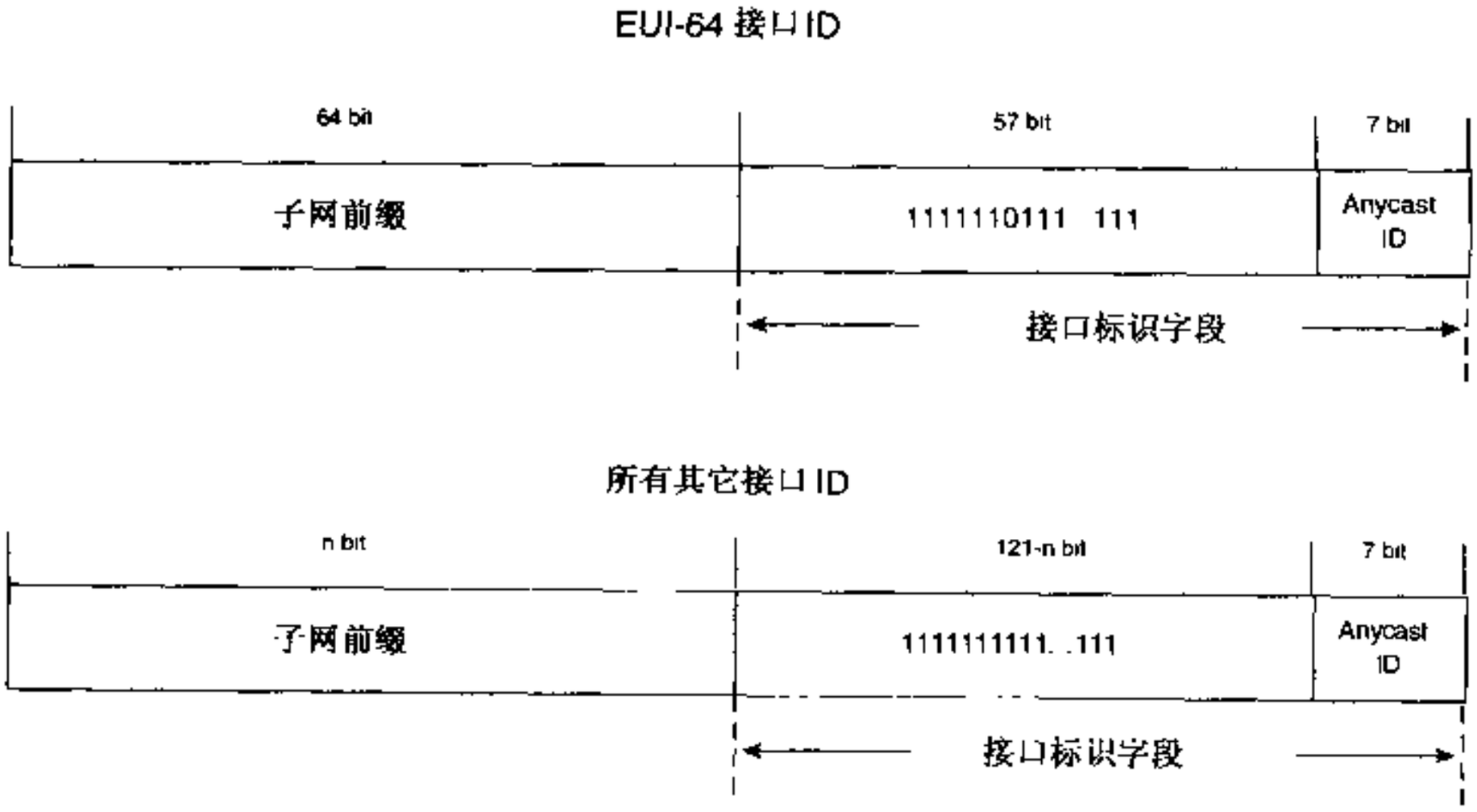


图 8-8　泛播地址构造

从图 8-8 中可以看到，保留的泛播地址空间完全在接口 ID 字段中得到。保留的泛播地址是为每一个子网所保留。最高 128 个接口 ID 保留用作泛播地址分配。地址的最后 7 比特标识了指定的泛播地址。

到本书完成为止，已定义的泛播地址是移动 IPv6 的归属代理。它的泛播标识符是二进制 1111110。即在每一个 IPv6 子网上，关联到 EUI-64 接口 ID：FDFF:FFFF:FFFF:FFFE 的 IPv6 地址都保留用作移动 IPv6 归属代理。前缀 FEC0:0:0:A/64 的移动 IPv6 归属代理使用泛播地址：FEC0:0:0:A: FDFF:FFFF:FFFF:FFFE。

所有其他的泛播标识：十六进制 0-7D 和 7F 都留作将来使用。

由于泛播地址在语法上与单播地址没有区别，所以接口必须被明确配置才能获悉该接口地址是泛播地址。

6．多播地址

多播地址标识一组接口，每一个接口都可以拥有多个多播地址。多播地址由 0xFF 开头，由此可以区别于单播地址。IPv6 的网络层上没有广播的概念。广播包对于不愿接受该广播包的节点来说附加了不必要的负担。接到广播包的所有 IP 接口必须处理该包，查看是否是所预期的接收者。节点通常都不是预期的接收者。每一个 IPv6 接口都知道自己所属的多播组。多播包仅由属于该多播组的接口处理。IPv6 使用多播来代替广播。

IPv6 多播地址可以是由官方的地址授权中心所分配(公认地址)，也可以是临时的——本地分配不作全球应用。最初的 IPv6 多播地址分配基于 IPv4 多播地址分配。所有相关的 IPv4 多播地址都转变成 IPv6 多播地址。你可以在 RFC 2375[2] 中发现当前已分配多播地址的完全列表。下面将出现的表 8-6 中列出一些例子。

IPv6 多播地址同样也有应用范围。多播地址有一个字移来标识范围是本节点、本链路、本站点、本组织或是全球。某一范围内的临时地址定义仅对该范围内节点有效。同一个地址可以在另一个范围，另一个网络定义成完全不同的含义。

图 8-9 显示了多播地址的格式。

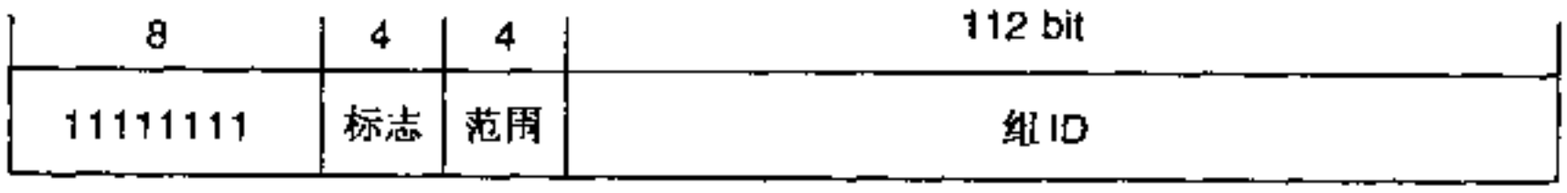


图 8-9　多播地址格式

前导的字节：二进制 11111111，标识了该地址是多播地址。

标志是 4 比特的集合。前 3 比特保留，必须置 0。最后一个比特指示该多播地址是由全球互联网编号权威机构永久分配的还是非永久分配，称为“临时”使用。第 4 比特为 0 表示该地址为公认多播地址，是全球互联网编号权威机构分配的。

范围是 4 比特的值，用作限制多播地址应用范围。表 8-5 列出了范围值。

表 8-5　　多播地址范围值

值	描　述
0	保留
1	本节点范围
2	本链路范围
5	本站点范围
8	本组织范围
E	全球范围
F	保留

所有的范围值都可以应用到公认地址或临时地址。特定应用范围的临时地址只在指定范围有效，在任何其他范围都无效。

组 ID 在给定范围内标识多播组，无论是公认还是临时。

表 8-6 列出一些常见的多播组以及他们的地址和范围。

表 8-6　　一些 IPv6 公认多播地址

IPv6 公认多播地址	IPv4 公认多播地址	多　播　组
本节点范围		
FF01:0:0:0:0:0:0:1	224.0.0.1	所有节点地址
FF01:0:0:0:0:0:0:2	224.0.0.2	所有路由器地址
本链路范围		
FF02:0:0:0:0:0:0:1	224.0.0.1	所有节点地址
FF02:0:0:0:0:0:0:2	224.0.0.2	所有路由器地址
FF02:0:0:0:0:0:0:5	224.0.0.5	OSPFIGP
FF02:0:0:0:0:0:0:6	224.0.0.6	OSPFIGP-指派路由器
FF02:0:0:0:0:0:0:9	224.0.0.9	RIP 路由器
FF02:0:0:0:0:0:0:D	224.0.0.13	所有 PIM 路由器
本站点范围		
FF05:0:0:0:0:0:0:2	224.0.0.2	所有路由器地址
所有范围有效		
FF0X:0:0:0:0:0:0:101	224.0.1.1	网络时间协议(NTP)
FF0X:0:0:0:0:0:0:127	224.0.1.39	Cisco-rp-announce
FF0X:0:0:0:0:0:0:128	224.0.1.40	Cisco-rp-discovery

在接口初始化过程中，多播协议和应用初始化以后，节点便加入到需要的多播组中。节点加入到所有节点多播组，地址为 FF01::1 和 FF02::1。地址格式指示，上述多播地址是公认的，范围分别是本节点和本链路范围。路由器加入到所有路由器多播组，地址为 FF01::2、FF02::2 和 FF05::2。他们是公认地址，范围相应是本节点、本链路和本站点。

你可以在表 8-6 中看到一个多播组可以在多个范围中运行，并拥有多个 IPv6 地址。这与 IPv4 中不同，在 IPv4 中公认地址没有范围。在第 5 章“IP 多播路由介绍”中讨论了两种将 IPv4 多播限定范围的方式。TTL 范围限定需要网络管理员在多播边界设定 TTL 门限。如果到达边界的多播包的 TTL 值小于某一设定的门限，则该多播包被丢弃。该方法的一个缺陷是灵活性较差——接口的门限对所有离开该接口的多播包都有效。该方法的另一缺陷是在一个大网中很难预期门限值。在 IPv4 中另一种类型是通过管理限值范围，定义一组只能在企业范围内使用的专用多播地址。保留的多播地址范围在 239.0.0.0-239.255.255.255。建议使用 239.255.0.0/16 作为本站点地址，使用 239.192.0.0/14 作为本组织范围。上述范围仅仅是建议，企业可以以任意方式使用保留地址。通过管理限值范围可以在组织内部工作，但是无法在全

球范围工作。上述地址仅用作内部使用。

多播范围限制已经内建在 IPv6 多播地址结构中。当前已经定义了 5 级范围限制。本链路范围在 IPv4 中可以通过将本链路多播包的 TTL 设置为 1 来得到。其他的 IPv6 范围限制为不同级别的多播限制提供一种潜在的能力。使用 IPv6 多播的应用可以限制在链路、站点或组织。在 IPv6 中为将来的范围限制提供了保留的地址。限定范围的公认多播地址在一定范围内应用，同时确保该多播地址不被用于另一个多播组。当两个公司在两个不同的多播组使用相同多播地址没有危害，但是两公司决定合并时上述情况会引起冲突。

一种特殊类型的多播地址是被请求节点(solicited-node)地址。该地址用不同的 IPv6 功能与 IPv6 节点通信。上述功能对该地址的使用在“IPv6 功能”一节中讨论。被请求节点多播地址应当由每一个分配了单播和泛播地址的接口创建及分配。该地址与本链路地址不同。被请求节点多播地址使用接口 ID 的第 24 比特与前缀 FF02:0:0:0:0:1:FF00::/104 组成。图 8-10 显示被请求节点多播地址如何形成。

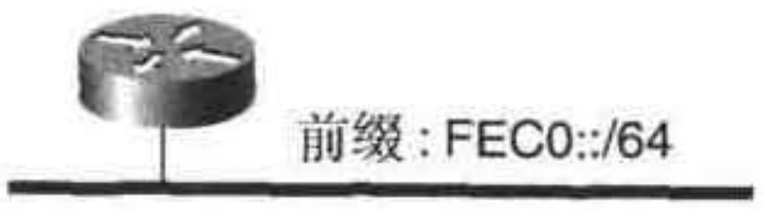


图 8-10　被请求节点多播地址格式

MAC 地址为 0000.0C0A.2C51 的接口形成的 EUI-64 接口 ID 是::200:CFF:FE0A:2C51，形成的本链路地址是 FE80::200:CFF:FE0A:2C51。在子网 0 上的本站点前缀形成本站点地址 FEC0::200:CFF:FE0A:2C51。由于接口有一个本站点地址，所以创建被请求节点地址。被请求节点地址使用接口 ID 的后 24 比特，加上被请求节点前缀，形成 FF01::1:FF0A:2C51。

每一个接口可能关联了多个前缀和 IPv6 地址。所有地址的接口 ID 都相同。使用接口 ID 的后 24 比特组成请求接口多播地址可以减少节点必须加入的多播地址。

7．节点需要的地址

节点在标识自身时需要识别多个地址。IPv6 机制要求节点只有维护这些地址才能正常工作。每一个地址的使用将在各 IPv6 功能中进一步讨论。

主机需要识别下列地址：

- 每个接口的本链路地址
- 所有分配的单播地址
- 环回地址
- 所有节点多播地址
- 对每个单播地址和泛播地址的被请求节点多播地址
- 该主机所属其他组的多播地址

在上述地址之外，路由器还需要识别：

- 每个路由接口的子网泛播地址
- 在路由器上配置的所有其他泛播地址
- 所有路由器多播地址
- 路由器所属其他组的多播地址

表 8-7 总结了地址类型。

表 8-7　　地址类型及举例

定义地址开头的数	举　　例	地 址 类 型
00	::1	未指定 环回 IPv4 兼容
2 3	2001:0608:0100::	可聚合全球单播
FE8	FE80:.200.CFF:FE0A:2C51	本链路
FEC	FEC0.:200:CFF:FE0A:2C51	本站点
FF	FF02::2	多播
FF02:0:0:0:0:1:FF00::	FF02:0:0:0:0:1:FF0A:2C51	被请求节点

8.4.2　IPv6 头

IPv6 的一个设计目标是改进 IPv4 的头。IPv6 的头更简单，灵活，使用选项时更有效。IPv6 对 IPv4 的选项部分删除，部分改名。地址长度变成原来的 4 倍，头仅变成原来 2 倍长。选项的编码的变化使处理效率高，并且对大小和增加都提供了更大的灵活性。

1．头格式

IPv6 的头很简单，包括源地址和目的地址共有 8 个字段。图 8-11 显示了 IPv6 的包头。

IPv6 头格式中各字段定义如下：

- *版本*指示 IP 的版本号(这里值为 6)。
- *净荷长度*是不包括头的 IP 包长度，以八位组计数。在下一节中讨论的扩展头被认为是净荷的内容，包含在净荷长度中。
- *下一包头*的值标识紧随在 IPv6 包头后面的头。下一包头或者是高层协议头(例如 ICMP、UDP、TCP)，或者是下一节要讨论的 IPv6 扩展头。
- *跳数限制*由包所经过的每一个节点递减。当跳数限制到达 0 时，该包被丢弃。一些 IPv6 功能例如路由器宣告，邻居宣告和请求，IPv6 重定向等只使用在同一链路上的不同设备间。一种 IPv6 用来确定包不是从不在本链路节点(可能是恶意重定向流量)发出的技术是将跳数限制设为 255，这是跳数限值的最大值。如果包从不在本链路的节点发出并穿过一个路由器，所收到包的跳数应当小于 255。收到该包的节点会认为该包无效并丢弃。

- *源/目的地址*是 128 比特的字段，用作容纳 IPv6 的源地址和目的地址。

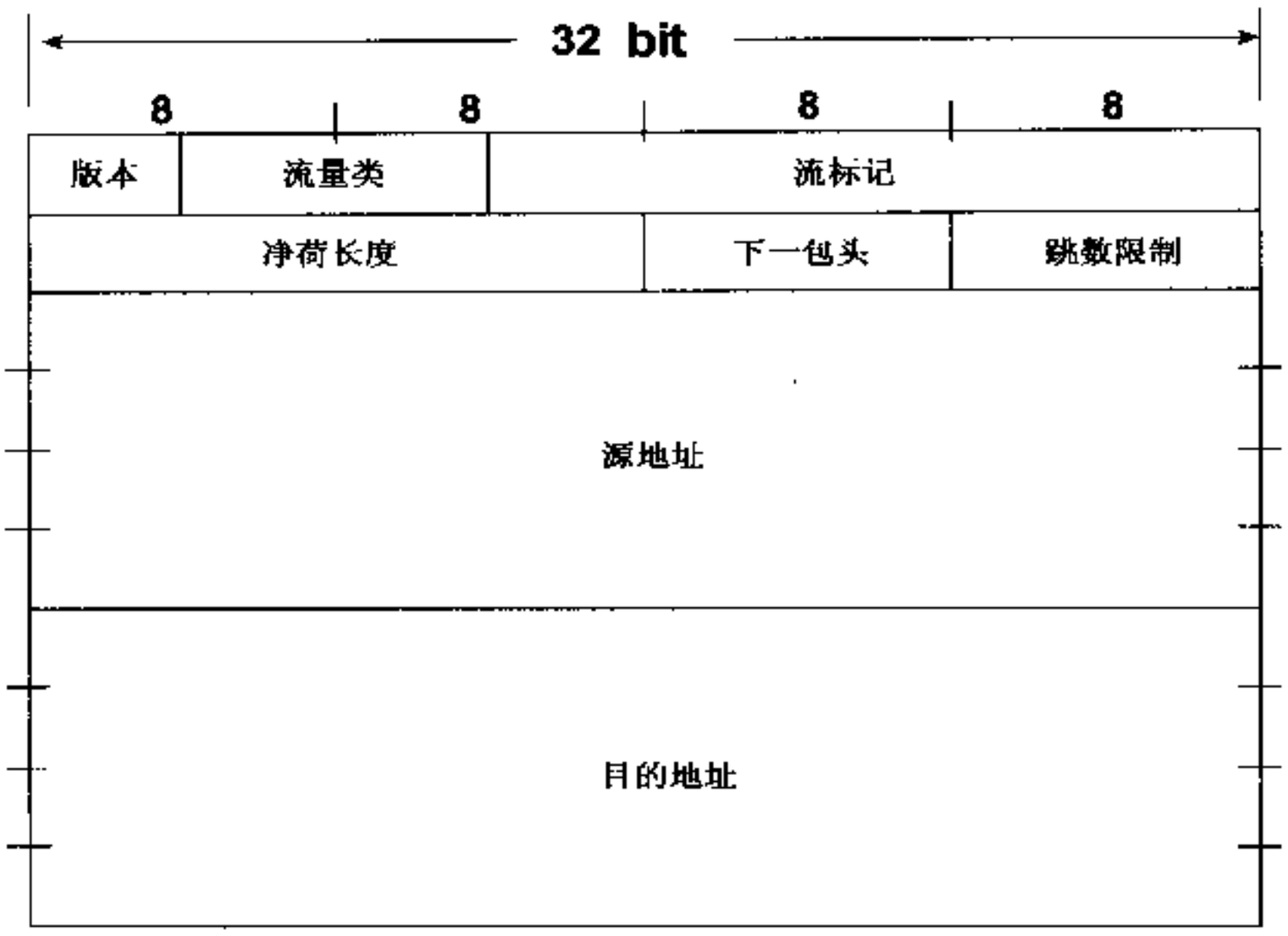


图 8-11　IPv6 头格式

流量类和流标记字段将在本章中稍后的“服务质量”一节中讨论。

图 8-12 中显示了 IPv4 的头作为比较。只有需要每个节点都处理的字段保留在 IPv6 头中。其他字段包含的信息与 IP 包可能相关，也可能不相关。这些信息被移到 IPv6 扩展头中。

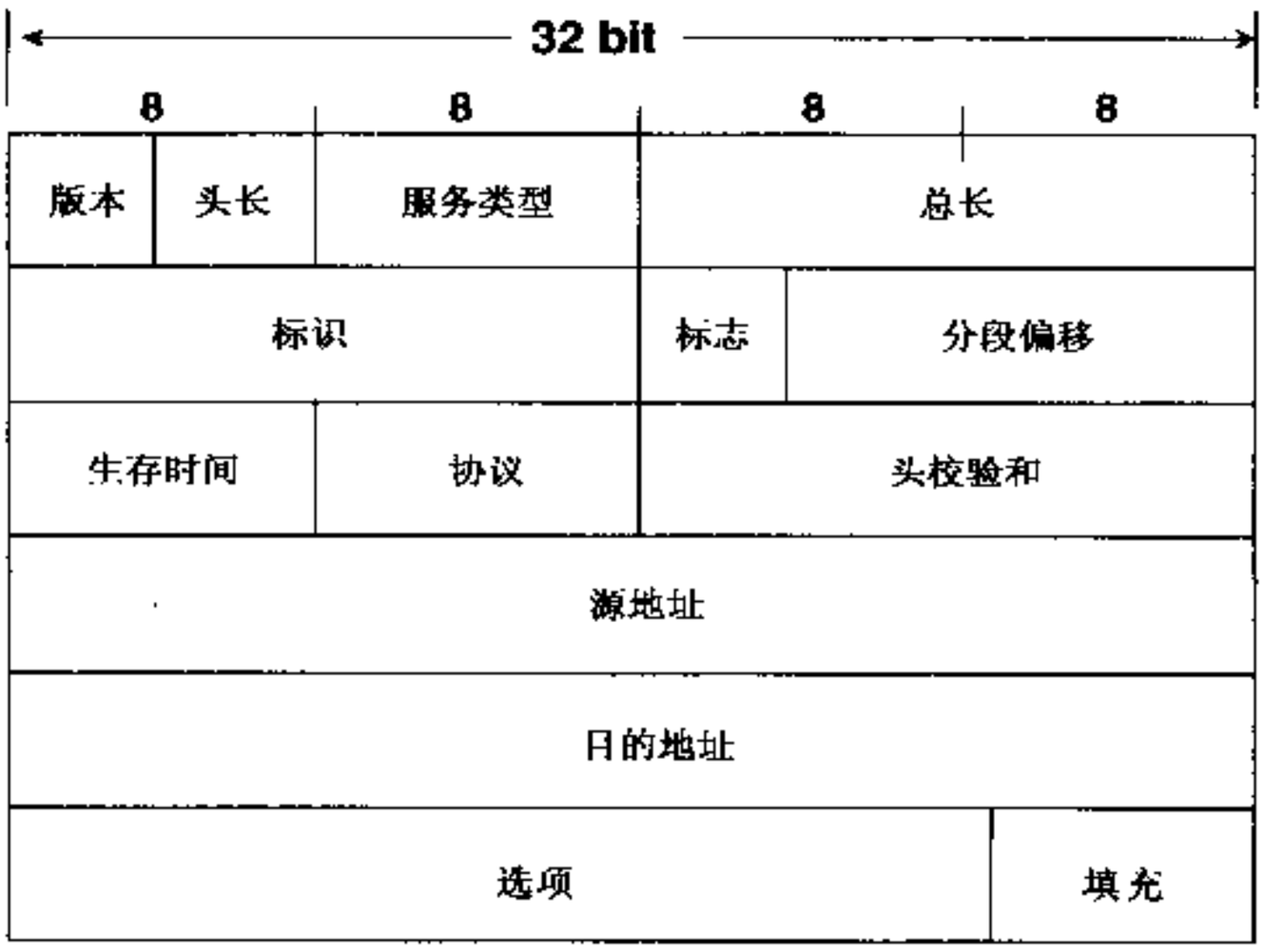


图 8-12　IPv4 头格式

2. 扩展头

IPv6 头中不包含可选的网络层信息。这些信息放在单独的头中，经编码后放置在 IPv6 头和上层协议头之间。除一个例外以外，扩展头不需要由包分发途径中每一个节点处理。他

们仅由包头中目的地址所标识的节点(或者多播目的节点)处理。这种机制使路由器不需要处理可能只与目的节点相关的信息，能够提高选项处理效率。上面所说的例外是每一跳选项。每一跳选项所包含的信息需要包分发路径上每一个路由器处理。

扩展头顺序

节点通过察看前一个头中的信息来决定是否需要检查处理扩展头。所以扩展头必须按出现在包中的顺序依次处理。如果所有扩展头都出现在包中，他们应当按照表 8-8 中的顺序。表中显示了表示扩展头的下一包头值。扩展头应当按照 8-8 中显示的顺序，但是除每一跳头外可以不按顺序。因为每一跳头如果出现则必须紧随在 IPv6 头之后。无论次序如何，节点必须处理包头。

表 8-8　　包头和下一包头值

包　　头	前一包头中下一包头值
每一跳选项	0
目的地选项	60
选路选项	43
分段选项	44
认证选项	51
目的地选项	60
IPv6 的 OSPF	89

目的地选项头在表 8-8 中出现了两次。在不同的位置含义不同。当出现在选路头之前时，该头由出现在 IPv6 目的地字段的第一个目的地址以及路由列表中后续地址的节点检查。如果单独出现或者在选路头之后出现，该选项仅由包的最终目的地处理。表 8-13 显示了扩展头如何使用。

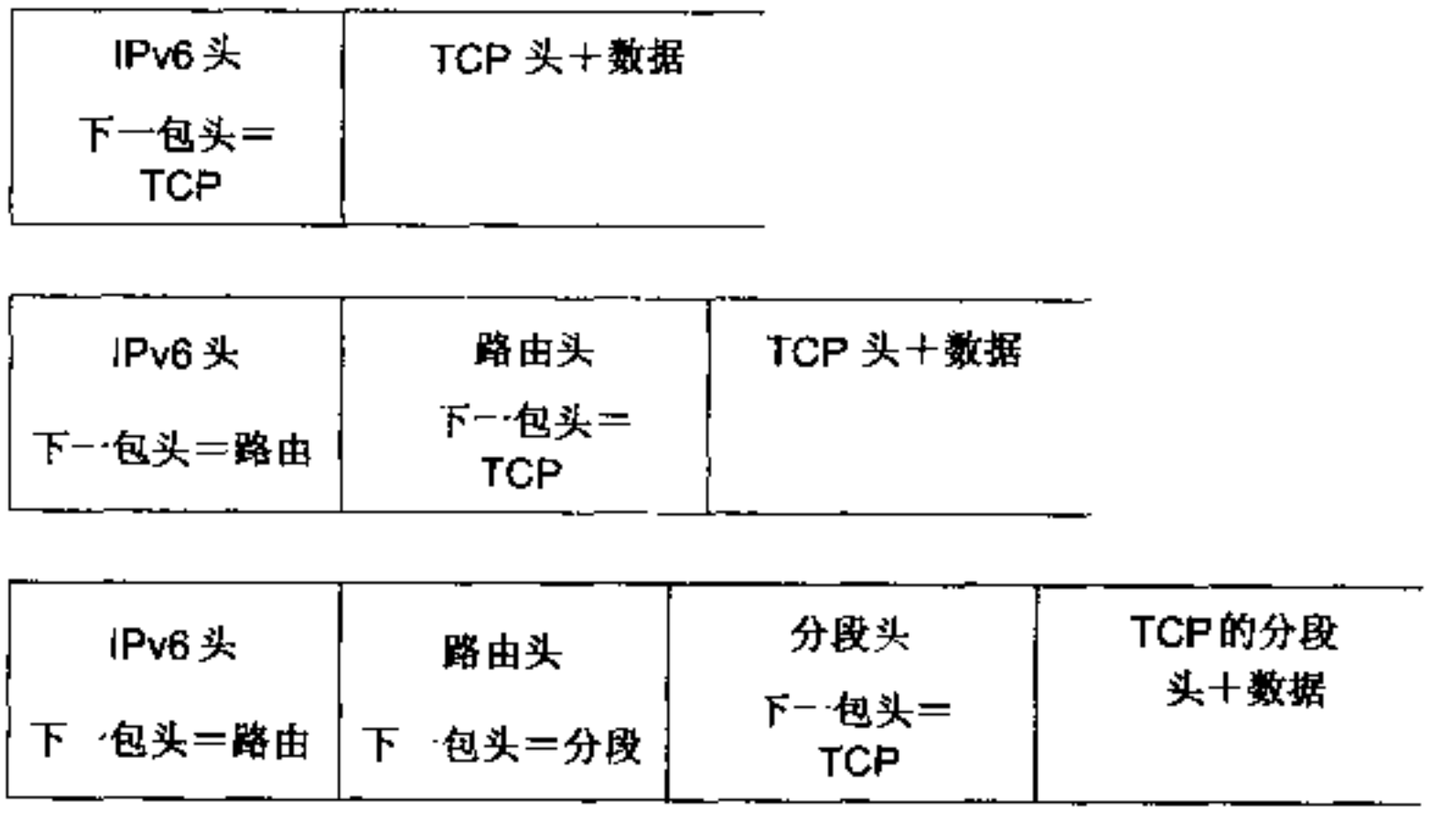


图 8-13　扩展头的使用

在每一个头中，下一包头值标识了紧随其后的包头。在读取 IPv6 头之后，如果该节点不是包的最终目的地并且下一包头不是每一跳包头，则该包被转发。如果该节点是最终目的地，则顺序处理每一个包头。

选项

当前定义的两个扩展包头——每一跳包头和目的地包头——包含很多类型长度值(TLV)选项。选项包含一个标志来指示传输中是否可以改变。对于认证包头，这一点非常重要。如果可以改变，认证包头在计算认证信息时将该选项作全零处理。当前指定了两个选项：Pad1 和 PadN。上述两个选项用来填充包头(使长度是 8 字节的整数倍)或者用来调整子选项。Pad1 在包头的选项区域增加 1 字节的填充。PadN 则增加两个或多个字节。

每一跳选项头

每一跳选项头中包含的信息必须由包到达目的地之前路径上每一个路由器检查。每一跳选项头必须紧随在 IPv6 头之后。这样在路径上每一个路由器可以可以检查该选项而不必处理其他扩展头。

选路头

选路头中列出的地址是包到达目的地之前必须访问的节点。IPv6 头中包含了必须访问的第一个节点，选路头中包含了一组这样的节点，包含最终目的地。

选路头中包含下一包头、长度、类型、剩余段和地址字段。类型字段只有一个已定义的值：类型 0。剩余段字段包含到达目的地之前明确列出必须访问的节点的数量。选路头由 IP 头的目的地址字段所标识的节点处理。该节点检查选路头。如果还有节点未访问，节点通过比较选路头中节点数与剩余段的数量确定下一访问的节点。将下一访问节点地址放置到 IPv6 包头中，递减剩余段的值，然后转发包。

分段包头

如果源节点想要发送一个大于到目的节点的 MTU 的包，则使用分段包头。如果路径上的链路 MTU 小于包长，由源节点负责将包分段。路由器不负责分段 IPv6 包。源节点负责将包分段，用多个包传输，到达目的地以后重组。源节点可以使用 MTU 路径发现过程来确定到目的地路径上的最小 MTU。当得到最小 MTU 以后，数据源就知道路径上可以发送的最大包长。如果数据源不运行 MTU 路径发现过程，则假设可以使用的最大 MTU 是 IPv6 最小 MTU:1280 字节。这一过程在“MTU 路径发现”一节中详细描述。同一数据包的所有分段由源节点产生的标识值来标记。

目的地选项头

目的地选项头包含必须由 IPv6 目的地检查的选项。当目的地选项头后紧接选路头时，该选项由包头中每一个节点处理。当目的地选项头在上层协议头之前时，该选项由最终目的地节点处理。

认证

IPv6 种增加了认证头(AH)。该包头试图为 IP 包提供认证以及完整性。IP 包中到达目的地之前传输过程中不改变的字段都将用作计算认证信息。在认证信息的计算中可能改变的字段例如跳数限制作为全零处理。

封装安全净荷

完整性与保密性由封装安全净荷(ESP)提供。你可以与 ESP 联用 AH 来提供认证。ESP

将需要加密保护的字段加密以后放入 ESP 头的数据部分。存在两种封装模式：隧道模式与传输模式。在隧道模式中 ESP 将整个 IPv6 数据包加密后放入封装字段。将 ESP 头用作新的未加密的 IPv6 头。在传输模式中，ESP 头仅将传输层会话(TCP、UDP 和 ICMP 等)加密后放入封装字段，ESP 头位于传输层协议之前。

安全机制在本书范围之外。关于 IPv6 安全的更多信息参见 RFC 2401,RFC 2402 和 RFC 2403。

8.5　IPv6 功能

一些作为 IPv6 设计的一部分的功能必须由所有宣称支持 IPv6 的节点实现，包括：

- ICMPv6
- 邻居发现
- 无类自动配置
- 泛播
- 多播
- MTU 通道发现(建议)

上述都是 IPv6 的基本功能，大多数情况下增强了 IPv4 的能力。

IPv6 的另一特性是为同一接口分配多个地址的能力，该能力消除了前缀重编号的问题。不但 IPv6 接口可以分配不同前缀的多个地址，而且同一链路上的两个节点无论是否使用同一前缀都可以直接通信。

该功能在本节中详细讨论。这里使用 Cisco 路由器的命令和输出来帮助你理解 IPv6 功能。

8.5.1　在 Cisco 路由器上使能 IPv6 能力

在 Cisco 路由器上的 IPv6(缺省禁止)可以由如下全局命令使能：

ipv6 unicast-routing [table-count *num***]**

Cisco 支持使能多个路由表。默认的情况下只使能一个路由表。多个路由表允许网络管理员对路由条目查找拥有更多的控制。最长匹配不再是唯一规则。如果使能多个路由表，转发算法顺序搜索路由表直到发现可用的路由。

配置 IPv6 的下一步是使能 IPv6 接口并配置地址或使能自动配置。自动配置在下一节讨论。

使能接口上 IPv6 并配置地址的接口子命令如下所示：

ipv6 address *ipv6address/profix-length* **[link-local]**

不配置地址且使能 IPv6 的接口子命令如下所示：

ipv6 enable

作为使能接口的一部分，路由器自动配置一个本链路单播地址。

位于同一以太网链路上的两个路由器 Falcon 和 Eagle 配置如例 8-1 所示。

注意例 8-1 中两个配置是相同的。

使用下面命令可以显示接口上的 IPv6 状态以及其他接口信息：

show ipv6 interface *interface-type number*

例 8-1　在位于同一以太网链路的两个路由器上使能 IPv6

```
Falcon
ipv6 unicast-routing
!
interface Ethernet0
 ipv6 enable
```

```
Eagle
ipv6 unicast-routing
!
interface Ethernet0
 ipv6 enable
```

例 8-2 显示了 **show ipv6 interface** 的部分输出结果，其中显示了以太网接口的 MAC 地址、IPv6 状态和接口上自动配置的本链路地址。

例 8-2　使用 **show ipv6 interface Ethernet 0** 来显示 IPv6 接口信息

```
Falcon#sh int e 0
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 0000.0c0a.2c51 (bia 0000.0c0a.2c51)

Falcon#show ipv6 interface ethernet 0
Ethernet0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::200:CFF:FE0A:2C51
```

```
Eagle#sh int e 0
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 0000.0c76.5b7c (bia 0000.0c76.5b7c)

Eagle#show ipv6 interface ethernet 0
Ethernet0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::200:CFF:FE76:5B7C
```

注意 Falcon 的 MAC 地址是 0000.0C0A.2C51，建立的本链路地址是 FE80::200:CFF:FE0A:2C51；Eagle 的 MAC 地址是 0000.0C76.5B7C，建立的本链路地址是 FE90::200:CFF:FE76:5B7C。节点上的 IPv6 都已使能。

8.5.2　ICMPv6

ICMPv6 是 IPv6 的一部分。实现 IPv6 的节点必须完全实现 ICMPv6。ICMPv6 是 ICMPv4 的改变版本。错误报告和许多 IPv6 功能例如 MTU 路径发现和邻居发现等都使用 ICMPv6。出错消息在这里讨论。

ICMPv6 包跟随在 IPv6 头后面或者下一包头值 58 的扩展包头之后(该值与 ICMPv4 中标识 ICMP 的值不同)。信息和出错消息由 ICMP 类型字段中高比特标识。出错消息的 ICMP 类型字段高比特为 0。ICMP 出错消息在不超过最小 IPv6 MTU 1280 字节的条件下，尽可能多包含引发该 ICMP 的包。

接着讨论下列 ICMP 出错消息：

- 目的地不可达
- 包太大
- 超时
- 参数问题

当节点因除拥塞外其他原因无法转发数据包时发送目的地不可达错误消息。节点向数据源发送出错消息，使用代码指示下列原因：

- 没有到目的地的路由(0)
- 因管理原因禁止访问(1)
- 地址不可达(3)
- 端口不可达(4)

当数据包超过链路 MTU 时节点发送包太大消息。与 IPv4 中不一样的是在 IPv6 中路由器不执行数据分段。只有源节点执行分段功能。引发该错误的链路上的 MTU 被包含在包中。包太大消息不区分目的地址是单播或多播。该消息用于 MTU 路径发现过程。

当 IPv6 跳数限制到 0 时，发送 ICMP 超时消息。跳数限制为 0 通常指示路由环路。

如果节点发现 IP 头或者扩展都的参数有问题，则发送 ICMP 参数问题消息。错误消息中包含指向错误参数的指针。使用下面错误代码标识所有问题类型：

- 遇到错误的包头字段(0)
- 遇到无法识别的下一包头类型(1)
- 遇到无法识别的 IPv6 选项(2)

8.5.3　邻居发现

邻居发现协议(ND)解决了同一链路上多个节点相关问题。ND 提供下列功能：无服务器自动配置、路由器发现、前缀发现、地址解析、邻居不可达检测、链路 MTU 发现、下一跳决定和重复地址检测等。在 IPv4 中需要许多协议包括 DHCP、ICMP 路由器发现、路由协议和 ARP 等才能提供上述功能的一部分。ND 使用 ICMPv6 来实现上述功能。ND 试图通过将所有功能集成到 IPv6 的必要部分 ICMPv6 中来改进 IPv4。

当节点初始化以后，必须了解下面内容才能通信：

- 了解自身 IP 地址。
- 了解自身前缀信息，这样才能知道如何向位于其他前缀的节点发包。
- 了解链路上的路由器。
- 了解如何决定到目的地路径上的下一跳地址。
- 了解如何得到关联在已知网络层地址的链路层地址。
- 了解所需发送数据包的大小。

为使通信更顺畅，节点还应了解如下内容：

- 当邻居不再可达时应当能检测到，不再向该邻居发包。
- 了解链路上的邻居。
- 应当知道正试图使用的地址是否已由链路上另一节点使用。
- 知道在同一链路上分配给该节点的其他前缀。
- 对任何目的地，如存在更好的下一跳路由，应能将流量重定向到该路由。

ND 定义了 5 种 ICMPv6 包。IPv6 节点在通信以前使用这些包来了解应该和必须了解的信息：

- 路由器请求(RS) ——当节点不愿等待下一次周期性路由器宣告，希望路由器立刻发送路由器宣告时发送的多播包。一个正在初始化的节点可以发送路由器请求，这样它可以马上得到链路上路由器的配置参数。
- 路由器宣告(RA) ——周期性发送或对请求作应答。路由器报告他们的存在，并提供节点配置自己所必要的信息。
- 邻居请求(NS) ——允许节点确定邻居的链路层地址，或者判断邻居是否还能从缓存的链路层地址可达。同时允许节点判断链路上是否存在重复的 IP 地址。
- 邻居宣告(NA) ——对邻居请求所发的应答或者在链路层地址改变时不经请求发送。
- 重定向——由路由器发送，将流量重定向到链路上更好的第一跳地址。

每一个消息都是特定类型的 ICMP 包。ICMP 包包含类型指定信息。每一类型的信息都包含一个或多个 TLV 选项。

ND 为无类自动配置提供基础：不需要配置服务器自动配置。路由器宣告为节点配置提供必要的信息。自动配置在“自动配置”一节中详细讨论。

1. 路由器请求

主机需要立即接收路由器宣告时发送路由器请求——不愿意等待周期性的宣告。在初始化时，主机发送路由器请求来尽快地了解配置信息。

RS 是类型 133 的 ICMP 包。源地址是分配给发送主机的地址。如果还没有分配地址，则使用未指定地址：0:0:0:0:0:0:0:0。典型的目的地址是所有路由器的多播地址。RS 可以包含一个发送者链路层地址的选项。如果源地址是未指定地址，则禁止包含链路层地址。

2. 路由器宣告

路由器在链路上宣告他们的存在并提供节点配置所必需的信息。RA 是多播包，发往链路范围内所有节点多播组。

RA 是类型 134 的 ICMP 包。源地址是发送路由器的本链路地址，目的地址是发送路由器请求包的节点地址或链路范围所有节点多播地址。跳数限制必须设置成 255。在这种情况下跳数限制不是限制路由器转发包。将跳数限制设为 1 才能限制包转发：因为路由器收到包后将跳数限制减 1，并丢弃跳数限制为 0 的包。跳数限制值 255 确保非本链路的设备不能通过发送路由器宣告来试图干扰通信流。如果非本链路设备向本链路发送 RA，经过路由器以后跳数限制减 1，使该包成为非法。接收节点确认 RA 是否有效中有一条确认跳数限制是否是 255。IPv4 不使用这种方式来确保包不能穿过路由器。

RA 中包含路由器生存时间。路由器生存时间告诉节点他们能将路由器作为默认路由器多久。该生存时间以秒为单位，最长 18.2 小时。其值为 0 表示该路由器不是候选的默认路由器，不能出现在节点的默认路由器表中。

收到 RA 的节点构造一个默认路由器列表。所有发送 RA 且生存时间不为 0 的路由器都出现在该列表。在默认路由器列表中生存时间表项由每一个收到的 RA 更新。如果一个列表中的路由器发出 RA 中包含生存时间为 0，主机立即将该路由器从缺省列表中删除(对 IPv4 的改进)。IPv4 路由器必须手工配置默认路由器列表。一些 IPv4 主机运行路由协议例如 RIP 来动态学习这些信息，一些主机运行 ICMP 路由器发现协议(IRDP)。然而无论是 RIP 还是 IRDP

都没有在所有的 IPv4 主机上实现。

RA 中还包含可达时间和重传定时器。可达时间告诉主机在收到邻居的可达确认以后多长时间内可以假设邻居仍然可达。该信息用在邻居不可达探测过程中。重传定时器是一个以毫秒为单位的时间，是连续两个邻居请求消息间的间隔。该定时器用在地址解析和邻居不可达探测过程中。

RA 包中有两个比特，管理地址比特(M)和另一个有状态配置比特(O)，告诉主机如何配置自己。如果设置 M 比特，则主机除使用无状态自动配置来配置地址外还使用基于状态的自动地址配置协议例如 DHCP 等配置地址。如果设置 O 比特，主机除使用基于状态地址配置协议配置地址之外还使用上述协议配置其他信息。IPv4 主机必须人工配置来指示是否通过 DHCP 来学习 IP 配置信息。通过路由器宣告在链路上自动提供这些信息能将包含在主机中的静态配置信息总量最小化，减少将来重配置的工作量。自动配置方式将在“自动配置”一节中讨论。

可能出现在 RA 中的可选项是源链路层地址、MTU 和前缀消息。在 RA 中包含源链路层地址省去主机在默认路由器执行地址解析协议。路由器可以选择不包括链路层地址。MTU 选项允许将主机在链路上使用 MTU 的控制集中化。该选项主要用作在可变 MTU 的链路上选择可用在其他链路的 MTU。该值在路由器中设置，然后配置到链路上所有的主机。前缀信息用作通知其他节点用作地址自动配置的在线前缀。知道链路上所有配置前缀的主机能够更智能地转发通信。一个多宿主机能对已知在线目标前缀选择最靠近的接口。非多宿主机能使用前缀信息帮助发现下一跳。

前缀信息选项包含本链路确定和无状态自动配置的数据。它包含实际前缀和前缀长度，长度从 1～128 比特。它还包含比特信息来指示前缀是否用在本链路确定或者地址配置。当 L 比特设置时，你可以使用前缀用作本链路确定。当没有设置时，你可以确定没有在线和非在线的信息。A 比特在设置时表示你可以使用前缀作无状态地址配置。

前缀选项还包含有效生存时间值和优选生存时间值。有效生存时间以秒为单位指示前缀用作本链路确定的有效时长。生存时间相关于包发送的时长。优选生存时间指使用该前缀配置的地址以秒为单位保持优选状态的时间。接口上的优选地址是节点可以主动用作通信的地址。优选生存时间为 0 表示不赞成使用拥有该前缀的地址。如果存在优选地址，则不赞成使用的地址只能用于维护现存的连接，不应当用作建立新连接。生存时间全 1 表示生存时间无限。你可以将前缀同时用作本链路确定和配置。两种类型的地址将在“自动配置”一节中详细讨论。

3. 邻居请求

邻居解析消息用来获得邻居的链路层地址、提供本机的链路层地址以及验证邻居的可达性。它是一个类型为 135 的 ICMP 消息。IP 包的源地址是发请求的节点的本链路地址。用作链路层地址确定时目的地址是关联在目标 IP 地址的被请求节点多播地址，用作可达性确认时目的地址是目标的单播地址。跳数限制是 255。和 RA 中一样，收到的 NS 中跳数 255 用于确保该包没有穿过路由器。如果该包穿过路由器，则跳数限制会小于 255。NS 中还包含指示目标地址的字段。

NS 中可以包含源链路层地址选项。如果 NS 试图得到目标的链路层地址，则 NS 是链路上的多播，包中还必须包含源链路层地址。包含源链路层地址能够减少链路上出现的地址解

析包。

4. 邻居宣告

邻居宣告用作应答 NS 或者在没有被请求时立即传播如链路层地址改变等新消息。NA 是类型为 136 的 ICMP 包。源地址是分配给发送接口的任意有效单播地址。当应答 NS 时目的地址是请求包的源地址，或者请求源地址是未指定地址时目的地址是所有节点地址。未经请求的宣告通常发往所有节点的多播地址。NA 包含被请求标志比特(S)。当 NA 应答 NS 时设置 S 比特。跳数限制是 255。目标地址是请求中同一个目标地址。它是链路层地址所查找的地址。对于未经请求的宣告，它是改变链路层地址的 IP 地址。NA 可以包含目标链路层选项。发送未经请求的 NA 通知节点宣告者新链路层地址包含该选项，值是新的链路层地址。被请求的 NA 类似于 IPv4 中的 ARP 应答。未经请求的 NA 是增加的特性。一个向所有节点地址发送的 NA 多播通知所有节点关于链路层地址改变，代替了 IPv4 网络上 ARP 缓存超时候请求使用设备新链路层地址的 ARP 请求和应答广播。

5. 重定向

路由器通过发送重定向消息通知主机到达目的地更好的第一跳。更好的第一跳可能是另一个路由器或者目的地本身。如果目的地是数据源的邻居，即使他们不属于同一个前缀，路由器可以重定向流，这样他们可以直接通信(IPv4 ICMP 的增强)。IPv4 ICMP 重定向消息由路由器在同一链路上存在替代路由器到目的主机或网络有更好的路径时发送。如果更好的第一跳是目的地本身时，上述 IPv4 机制无法重定向。重定向允许相同数据链路上拥有不同前缀的主机能直接通信，无需通过路由器。

重定向消息的源地址是路由器的本链路地址。目的地址是被重定向包的源地址。跳数限制是 255。

目标 IP 地址和目的地址同样包含在 ICMP 包中。如果更好的第一跳是路由器，目标地址是路由器的本链路地址。如果更好的下一跳是目的地本身，目标地址是目的地的 IP 地址。ICMP 的目的地址是被重定向流的目的 IP 地址。注意如果更好的下一跳是目的地本身，上述字段将包含相同的地址。

重定向消息可能包含目标链路层地址选项。该选项使主机不依赖于地址解析就能得到链路层地址。

引发重定向消息的部分 IP 包可能作为选项包含在重定向消息中。在重定向消息不超过 1280 字节的条件下，尽可能多地包含引发重定向的 IP 包。

6. 下一跳发现

需要发包的主机必须首先决定下一跳。如果曾经发往过该目的地，下一跳可能存储在目的地缓存中。如果这是发往目的地的第一个包，通过比较目的地址与主机的本链路前缀表发现下一跳。本链路目的地的包直接发往目的地节点。非本链路目的地的包发往默认路由器。IPv4 节点除发往本子网流量外，必须将所有流量发往默认路由器。如果目的地节点位于同一链路不同子网，路由器将通信流转发回链路，通信流在链路上传输了两次。

7. 地址解析

地址解析在节点已知某 IP 地址查找其链路层地址时使用。地址解析过程使用邻居解析与邻居宣告。节点向目的地 IP 地址发包前首先检查邻居缓存查找是否存在该项。如果不存在，节点建立一个该 IP 地址的表项，状态为 IMCOMPLETE。然后节点向所请求 IP 地址的被请

求节点多播地址发送邻居请求。请求的源地址是单播地址，或者是发起通信流的节点的源地址，或者是在远程链路上查找目的地的路由器的源地址。如果存在的话，该包还包含源链路层地址。

收到来源于单播地址、发往本机接口的邻居解析包的节点使用邻居宣告作应答，指示自身链路层地址。

当请求的节点收到被应答的邻居宣告时，将邻居缓存条目更新成目标的链路层地址，并将状态由 IMCOMPLETE 改为 REACHABLE。

注：　对不同可能反映的完整描述见 RFC 2461[3]。

8. 邻居不可达检测

如果与另一节点通信故障，在高层协议发现以前检测出故障并没有带来好处。但如果是路由器坏了并可能有一台替代的路由器，在高层协议发现以前检测出故障则非常有意义。

邻居可达性可以由下面两种方式之一确认：从高层协议的暗示或邻居请求的应答。前转方向的邻居一定是可达的。如果高层协议正在作前转，则邻居可达性已经被确认。例如前转过程由 TCP 连接实现，收到请求数据的确认或者发送确认后收到数据都可以证明邻居是可达的。如果转发过程是端到端的，经过下一跳路由器，则到路由器的可达性也得到了确认。

一些高层协议例如 UDP 不提供这些暗示。如果高层协议无法收到确认，则节点主动探测邻居来确定可达性状态。节点向所缓存的邻居的链路层地址发邻居请求并等待邻居宣告。节点只有收到邻居请求时才在邻居宣告中设置请求比特。当节点收到设置有请求比特的邻居宣告时，可以确认它的邻居收到了所发的邻居请求，所以转发方向通信存在。这些探测在通信流之后发送。如果节点不发送通信流则不向节点发送探测。

邻居缓存包含关于邻居的信息，包括 IP 地址、链路层地址和可达性状态。表 8-9 罗列了可能的可达性状态。

表 8-9　　邻居可达性状态

状　　态	描　　述
INCOMPLETE	正在作地址解析，已发送 NS，还没有收到应答
REACHABLE	在过去的 30 秒中转发方向的通信已被确认
STALE	在过去 30 秒中在邻居缓存中的表项没有被确认，在邻居缓存中加入未请求的邻居宣告的发送者，状态为 STALE，在需要向该表项的主机发送通信流前，不需要任何动作
DELAY	在过去的 30 秒钟没有收到可达性消息，并在过去 5 秒内已向该邻居发送包。如果进入 DELAY 状态 5 秒以后还没有收到被动确认，则发送 NS 并将状态转移为 PROBE
PROBE	已发送 NS 验证可达性，还未收到 NA

最初邻居缓存的表项在 INCOMPLETE 状态。当学习到该表项的链路层地址以后，转发方向的通信被确认，状态转入 REACHABLE。在转发方向的通信不断地被确认时，状态保持在 REACHABLE。

当无法从 REACHABLE 状态的主机收到可达性确认时，状态改变成 STALE。从节点收

到未经请求的 RA 或 NA 在邻居缓存中增加 INCOMPLETE 条目，然后立即转移到 STALE 状态。未经请求的宣告没有为转发通信提供任何信息。在向该表项节点发通信流之前表项保持 STALE 状态。

旦向邻居发包，状态转移成 DELAY，并且设置 5 秒定时器。即使主机处于 STALE 状态，包也发往缓存的链路层地址。如果在收到任何可达性确认以前定时器超时，则转入 PROBE 状态。如果确认可达性，则转入 REACHABLE 状态。

当转入 PROBE 状态以后，向邻居所缓存的链路层地址发送 NS。即使没有另外的包需要发送，在缺少应答条件下请求被每秒依次连续发送。在发送 3 次请求后 1 秒内没有收到应答，该表项从缓存内删除。

例 8-3 显示了 **debug ipv6 icmp** 和 **debug ipv6 nd** 命令的输出，并显示出邻居缓存由 INCOMPLETE 经过所有中间状态转移到 REACHABLE 状态。例 8-3 还显示了 **show ipv6 neighbor** 命令的的输出，该命令显示邻居缓存。**show ipv6 neighbor** 命令提供 IPv6 地址、存在时间、链路层地址(如果已知)、状态以及相应接口。

例 8-3　debug 输出显示邻居可达性状态变化

```
Falcon#debug ipv6 icmp
ICMP packet debugging is on
Falcon#debug ipv6 nd
ICMP Neighbor Discovery events debugging is on

10:58:08: ICMPv6-ND: Received RA from FE80::200:CFF:FE76:5B7C on
  Ethernet010:58:08: ICMPv6-ND: INCMP created: FE80::200:CFF:FE76:5B7C
10:58:08: ICMPv6-ND: INCMP -> STALE: FE80::200:CFF:FE76:5B7C

Falcon#show ipv6 nei
IPv6 Address                              Age MAC Address    State Interface
FE80::200:CFF:FE76:5B7C                     2 0000.0c76.5b7c STALE Ethernet0

11:01:13: ICMPv6: Received echo request from FE80::200:CFF:FE76:5B7C
11:01:13: ICMPv6: Sending echo reply to FE80::200:CFF:FE76:5B7C

11:01:13: ICMPv6-ND: STALE -> DELAY: FE80::200:CFF:FE76:5B7C

11:01:19: ICMPv6-ND: DELAY -> PROBE: FE80::200:CFF:FE76:5B7C
11:01:19: ICMPv6-ND: Sending NS for FE80::200:CFF:FE76:5B7C on Ethernet0
11:01:19: ICMPv6-ND: Received NA for FE80::200:CFF:FE76:5B7C on Ethernet0
  from FE80::200:CFF:FE76:5B7C
11:01:19: ICMPv6-ND: PROBE -> REACH: FE80::200:CFF:FE76:5B7C

Falcon#show ipv6 nei
IPv6 Address                              Age MAC Address    State Interface
FE80::200:CFF:FE76:5B7C                     0 0000.0c76.5b7c REACH Ethernet0
```

Falcon 从 Eagle 的链路层地址 FE80::200:CFF:FE76:5B7C 收到 RA。由于该宣告未经请求，所以在 Falcon 缓存中建立 INCOMPLETE 状态的条目，随后进入 STALE 状态。这时命令查询邻居缓存。表项显示状态在 STALE 且链路层地址已知。

几分钟以后，由应答请求消息可见 Eagle 向 Falcon 发 ping 包。Falcon 使用缓存的链路层地址向 Eagle 发应答。由于路由器向 STALE 状态的表项发包，所以状态转移成

DELAY，确认转发方向通信通道是否可用。路由器无法使用 ICMP 来确认可达性。所以将状态转移到 PROBE，通过发送 NS 来探测 Eagle 获得可达性确认。Eagle 发送 NA。Debug 命令无法显示 NA 中设置的请求比特。收到 NA 确认通信以后，Falcon 将 Eagle 表项状态转移成 REACH。

邻居不可达检测过程使主机能在默认路由器故障时将流量重定向到替代的路由器。该过程检测出默认路由器的故障，然后选择另一个路由器来转发流量。这些过程能在上层协议或应用超时前悄悄完成。IPv4 路由器永远无法检测到默认路由器故障。路由器故障时上层协议或应用将超时。IPv4 主机将试图使用无法使用的路由器重新建立连接。某些 IPv4 主机可能能配置多个默认路由器，选择第二路由器来建立连接。

9. 默认路由器选择

当目的地不在本链路并且目的地不存在缓存表项或现有默认路由器故障时，主机必须在默认路由器列表中选择一个路由器。通常在到特定目的地的流量第一次需要时选择默认路由器。该信息将被缓存以供后续流量使用。

默认路由器选择过程使用默认路由器列表和邻居缓存。在选择默认路由器时没有确认不可达的路由器，即不处于 INCOMLLATE 状态的路由器优先选择。如果存在多个路由器处于非 INCOMPLETE 状态，路由器选择过程按照实现方式或者永远返回同一个路由器或者以轮流的方式返回路由器。

当下一跳路由器故障时，邻居不可达检测过程将检测到故障。如果确实发生故障，该路由器表项将从缓存中删除。然后重复下一跳检测和地址解析，使用可用的下一跳路由器。

案例学习：默认路由器故障和通信恢复

一个主机使用 FTP 从远程服务器传输文件。主机向本链路默认路由器发流量。主机因数据发送不断收到 ACK 所以确认默认路由器可达。在会话过程中路由器发生故障，主机无法收到 ACK。由于主机无法从 TCP 层收到确认转发方向通信的暗示，所以将路由器状态转移成 STALE。由于主机仍试图发包，所以状态转移成 DELAY。5 秒以后主机仍无法收到路由器可达性状态的被动确认，所以将状态转移到 PROBE 并发送 NS。由于路由器无法应答所以从缓存中删除。

主机仍然试图发包，但是没有相应的下一跳条目。由于主机看到目的地前缀非本链路，所以在存储的列表中获取默认路由器。如果缓存中没有所得路由器，主机将该路由器以 INCOMPLETE 状态放入邻居缓存，并试图通过发送 NS 得到链路层地址。当新的路由器应答 NA 以后，可达性被被动确认，通信流量发往新的路由器。

10. 重复地址检测

所有节点在向接口分配单播地址以前实施重复地址检测。对 Anycast 地址不作上述检测。无论地址分配是通过有状态、无状态还是手工得到，都要进行上述检测。检测在接口初始化过程中地址分配以前实施。当重复地址检测过程运行时，将分配给接口的地址称为“尝试”的。

在发送请求以前，接口加入所有节点多播组确保能从使用该地址的任何节点收到邻居宣告，为尝试地址加入被请求节点多播组确保其他节点不试图使用该地址，所有节点将获知各自的存在。

节点使用尝试的 IP 地址作为目标发送邻居请求。源地址是未指定地址，目的地址是尝试地址的被请求节点多播地址。缺省条件下发送一次请求。

任何已使用所尝试地址的节点收到邻居请求后发送邻居宣告作应答。宣告中指定的目标是尝试的地址。目的地址是尝试地址的被请求节点地址。如果节点收到邻居宣告并且目标地址是接口所尝试地址，则所尝试地址重复且不能分配给接口。一些 IPv4 的主机在分配接口 IP 地址前作重复地址检测。但是并不是所有的 IPv4 主机都作检测，重复地址的分配可能会干扰现存的通信流。

8.5.4　自动配置

由于网络管理对网络的成功运行是如此重要，帮助网络管理的机制甚至有必要内建到协议中。在需要变化时，那些主机静态配置、人工输入的网络很难管理。很多工具能减轻 IPv4 网络管理负担，例如 DHCP 能使静态配置总量最小化，但是他们都不是协议的必要部分。无论是否存在 DHCP 服务器，IPv6 主机都能自动配置，使主机配置改变非常方便。

路由器宣告告诉主机如何配置自己。RA 包含两个比特告诉主机是否使用配置服务器，如果使用则是否获取地址以外的信息。如果设置管理配置(M)比特，主机应使用有状态的地址配置协议例如 DHCP。主机上还有无状态地址配置。其他有状态信息(O)比特告诉主机使用有状态配置协议配置地址以外的信息。另一方面 IPv4 的主机如果需要动态配置获取 IP 地址和其他信息，则需要静态配置成使用 DHCP 在特定的 DHCP 服务器得到。否则这些配置只能手工录入。

1. 无状态自动配置

节点通过混合节点所知道的(接口标识符)和路由器所知道的(分配给链路的前缀)可以配置自身的 IP 地址。不需要配置服务器来建立基本 IP 连接。上述机制可以在任何有多播能力的接口上工作。

在接口初始化时，节点为接口产生本链路地址。本链路地址由接口标识与公认的本地链路前缀 FE80::组成。本链路地址前缀的最右面的 0 由接口标识替代，构成 128 比特的地址。注意，接口标识通常是 64 比特，但不总是 64 比特。

本地链路前缀 FE80:0:0:0:0:0:0:0 和接口标识 200:CFF:FE0A:2C51 组成本链路地址 FE80:0:0:0:200:CFF:FE0A:2C51。如果接口 IS 大于 118 比特，则不能与 10 比特的本链路 FP 联用。自动配置会失败，只能使用人工配置。

节点不直接将产生的本链路地址分配给接口。首先确定是否存在重复地址。节点启动重复地址检测过程。

检测到地址重复的节点必须人工配置。一种方法是再配置一个替代的接口 ID。这样节点仍然能执行无状态自动配置过程自动配置所需要的地址加上分配的单播多播和地址。另一种方法是在接口人工配置 IPv6 地址。如果接口上需要配置多个地址，这样的配置需要巨大的管理工作量。

当节点确认不存在重复地址，再将地址分配给接口。

这时已存在基本 IP 层连接。即使没有路由器，链路上的主机已经能相互通信。主机不需要人工的网络层配置就可以实现这样的通信。

例 8-4 显示了 Falcon 与 Eagle 的最少基本配置。

例 8-4　Falcon 与 Eagle 使用 IPv6 通信的最小基本配置

```
Falcon
ipv6 unicast-routing
!
interface Ethernet0
 ipv6 enable
```

```
Eagle
ipv6 unicast-routing
!
interface Ethernet0
 ipv6 enable
```

从 Eagle Ping Falcon 的本链路地址显示出可以通信，正如例 8-5 输出所显示。

例 8-5　确认 Falcon 和 Eagle 本链路地址间可以通信

```
Eagle#ping ipv6 fe80::200:cff:fe0a:2c51
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to FE80::200:CFF:FE0A:2C51, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 60/68/80 ms
```

为实现基本 IP 连接，路由器和主机都执行上述无状态自动配置过程的每一步骤。每一个接口必须创建一个本链路地址。除下面讨论的情况之外，无论通过无状态自动配置、有状态自动配置还是人工配置，所有的单播地址在分配给接口之前必须执行重复地址检测。

主机继续自动配置(路由器不需要)。主机发送“所有路由器”多播请求来查找链路上的路由器。所有的路由器都使用路由器宣告作应答。RA 可能通知主机使用有状态自动配置来配置地址和其他信息。主机使用标记成地址配置的前缀信息来创建本站点地址。为创建本站点地址，需要将 FP、前缀和接口 ID 连接起来。当分配本站点地址时，主机不需要检查地址是否重复。原因是上述过程已验证本链路地址唯一。本链路地址唯一表示本链路上接口 ID 唯一。本站点地址只是为同一个接口 ID 分配不同的前缀，所以本站点地址一定唯一。这样就使用相同的方法产生且分配了全球可聚合地址。

RA 还提供本链路前缀信息。该信息是前缀和前缀长度的列表，标识成本链路前缀。主机使用上述信息建立自己的前缀列表。主机使用前缀列表来确认主机是在本链路还是其他链路，于是决定是否使用默认路由器来发送流量。

主机还可以基于 RA 中包含的信息来配置 MTU。

如需要配置其他信息例如 DNS 服务器等，则需要有状态自动配置。

配置路由器时使用下面的接口子命令可以使路由器使用特定值宣告前缀；设置管理配置标志，设置其他配置标志：

ipv6 nd prefix-advertisement 2001:ABAB::/48 3000 3000 online autoconfig

ipv6 nd managed-config-flag

ipv6 nd other-config-flag

前缀 2001:ABBA/48 使用生存时间 3000 秒，优选生存时间 3000 秒，可用作本链路宣告和自动配置。

只要接口使能，自动配置过程就会在节点的接口上执行。多宿节点在每个接口上独立执行自动配置。下面情况下接口被使能：

- 系统启动时接口的初始化
- 由系统管理员临时禁止接口或接口出现故障后的重新使能
- 接口第一次连接到链路
- 管理原因关闭以后重新使能

2. 有状态自动配置

有状态自动配置可以和无状态自动配置联合使用。DHCP 为 IPv4 提供有状态自动配置。DHCP 的 IPv6 实现能够使用 IPv6 的优越性，增强 DHCP 的能力。

注：　动态主机配置工作组已出版了 DHCP 的 IPv6 草案“draft-ietf-dhc-dhcpv6-15.txt”。

由地址服务器为提出请求的主机分配地址和其他信息，例如 DNS 服务器地址。和无状态自动分配的前缀一样，地址上关联了有效生存时间和优选有效生存时间。服务器可以使用有效生存时间要求所有主机将被分配的地址重新生效，从而达到对网络重新编号的目的。

3. 重新编号

即使拥有足够的地址，站点重新编号仍然会发生。一种情况是地址前缀严格维护，分配给站点的地址可能会取消。另一种情况是站点可能希望更换 ISP，和现在 IPv4 的公司更换 ISP 一样，需要改变前缀。IPv6 的设计并不能消除这种现象，只能使重新编号更简单。

地址可能处于两种状态：优选和非优选。主机总是试图使用优选的地址来通信。非优选地址只有在下边情况采用作源地址：如果使用优选的地址会干扰现有的连接。如果两个主机间有使用优选地址建立的 TCP 连接，当其中一个主机地址变成非优选时，如果该主机该用新的优选地址，则连接会中断。

使用优选和非优选地址可以使主机重编号变得简单。地址保持优选的时间由路由器宣告设置，该宣告在链路上周期性发送且由链路上所有节点处理。可以在路由器宣告中增加新的前缀，从而在接口上增加新的地址，旧的地址可以变成非优选然后删除。可以使用相应的机制利用配置服务器对主机重编号。服务器可以对所有节点发多播请求，要求重配置分配的地址。主机可以询问配置服务器关于改变生存时间值的地址，非优选现存地址或分配新的优选地址。这种重标号机制的健壮性依赖于路由器宣告和有状态消息能够到达链路上所有主机。考虑下面由 RFC 2641 得到的案例学习。

案例学习：网络重编号。

有一个被宣告的前缀，生存时间是 2 个月。8 月 1 日决定，该前缀必须改变，9 月 1 日停止使用。该前缀宣告的生存时间可以改变成两星期，随着 9 月 1 日的临近逐渐减少直到为 0，这时该地址非法。但是考虑到可能有一个主机 7 月 31 日从网络断开，当 9 月 1 日接入网络时，该主机仍然认为前缀的有效时间到 9 月 30 日。强制主机停止使用先前被宣告拥有很长的生存时间的前缀的唯一办法是发送很短生存时间的 RA。路由器必须连续发送生存时间为 0 的 RA 直到 10 月 1 日，来确保变化前从网络上断开的主机重新连入网络时不使用无效的前缀。

除非阻止重新连入网络的主机使用旧的前缀，一般来说路由器不应宣告生存时间为 0 的前缀。注意当主机可能频繁地从网络上断开连接时，路由器如果宣告生存时间无限的前缀可能在网络重编号时产生问题。

如果需要保持通信，路由器重编号需要大量的计划工作。路由器相互之间有通信，主机通过路由器的本链路地址与路由器通信。这些通信都独立于分配的前缀。无论链路上分配什么样的全球地址，节点将继续与路由器的本链路地址通信。

IPv6 中 DNS 的重编号工作与在 IPv4 中一样。或者在新地址可用之前将地址人工输入到 DNS 数据库，或者实现动态更新 DNS 服务器(DDNS)。

8.5.5　路由

前面的章节讨论了节点如何发现向邻居转发包所需要的信息和目标节点不在本链路时向下一跳路由器转发包所需要的信息。下面讨论路由问题，说明 IPv6 包在大网中路由的不同方式。

1．MTU 路径发现

在 IPv6 网络上的每一条链路上，MTU 必须至少 1280 字节长。然而建议的大小是 1500 字节或者更大。不能处理上述大小包的链路必须提供链路层的分段功能。IP 层的分段功能只由数据源实现，中间路由器不实现分段。节点可以不需要实现 MTU 路径发现，但是建议实现。没有实现 MTU 路径发现的节点使用 IPv6 最小 MTU，即 1280 字节。实现 MTU 路径发现的节点可以享受网络上可以提供的最大包的好处，可能获得更高的性能。单播目的地或多播目的地都由路径发现机制实现。

MTU 路径发现使用 ICMP 中“包太大”出错消息。节点最初发送流量时假设路径 MTU(PMTU)等于所连链路的 MTU。分发路径上任何节点发现无法在一个 MTU 较小的链路上分发包时发送“包太大”ICMP 出错消息并丢弃较大的包，上述消息中指示链路的 MTU。源节点收到出错消息以后将包减小到出错消息中指示的 MTU 值。该过程可能在分发路径上每一个更远的节点处重复。图 8-14 显示 PMTU 的发现。

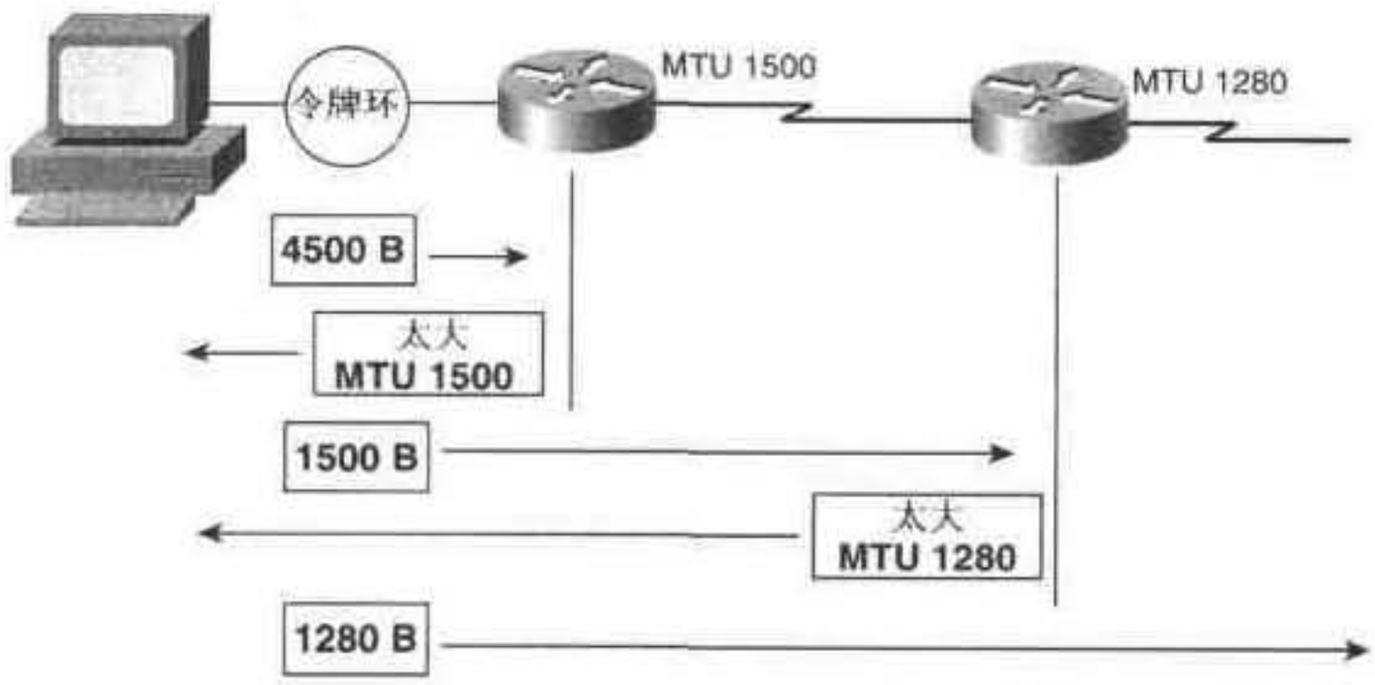


图 8-14　PMTU 发现过程

连接在令牌环上的 PC 机开始使用 4500 字节发送数据包。该包到达一个分发路径的 MTU 是 1500 字节的路由器。路由器向 PC 机发送 ICMP 包太大消息，包含 MTU 1500 字节并丢弃收到的包。PC 机建立一个较小的 1500 字节的包。在第一个路由器没有问题。下一个路由器分发路径上链路 MTU 为 1280 时该路由器丢弃所收到的包并向 PC 机发送 ICMP 包太大消息。

然后 PC 机发送大小为 1280 的包，该包能通过所有路由器。

无论对多播目的地址或单播目的地址，MTU 路径发现机制都能工作。多播包可能分枝到许多路径。任何路径上任何节点都可能发送包太大消息。该 PMTU 机中的最小值决定了所发送包的大小。

2．RIPng

RIPng(ng 代表“下一代”)基于 RIP 版本 2(RIP-2)。所有的操作过程、定时器或稳定过程都没有改变。RIPng 是将 RIP-2 改变成在 IPv6 接口上支持更大的 IP 地址和多播地址。RIPng 的 UDP 端口号是 521。RIPng 不能同时支持 IPv4 和 IPv6，所以不能后向兼容 RIP-2。

注：　路由 TCP/IP 第一卷第 7 章“路由信息协议版本 2”中讨论 RIP 版本 2。

图 8-15 说明 RIPng 消息格式。基本结构非常类似于 RIP-2。

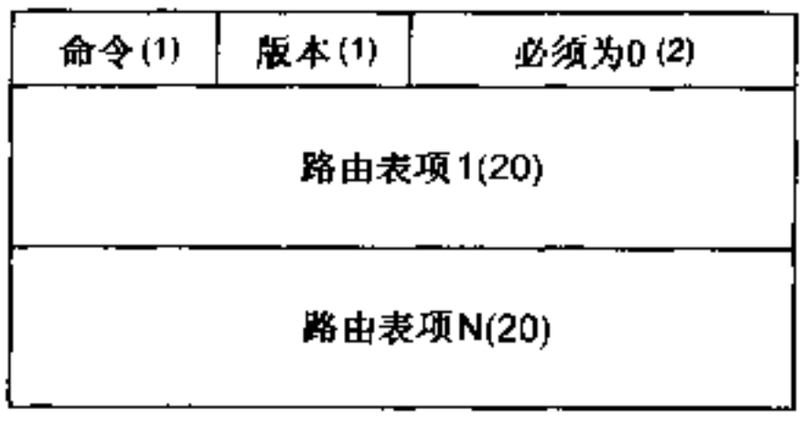


图 8-15　RIPng 消息格式

RIPng 消息字段定义如下(包括字节为单位的长度)：

- 命令设置成 1 表示请求，或者 2 表示应答。
- 版本当前为 1。

下面的消息包含路由表条目(RTE)列表。图 8-16 说明 RTE 的格式。

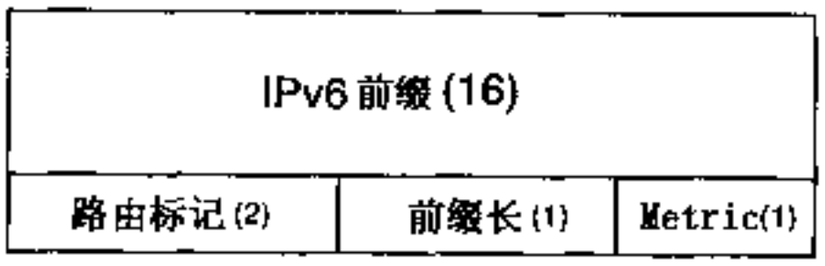


图 8-16　RIPng 路由表条目格式

RTE 格式中的字段定义如下：

- IPv6 前缀：128 比特 IPv6 地址前缀。
- 路由标记：与 RIP-2 相同，提供一个字段标记外部路由或被注入到 RIPng 中的路由。
- 前缀长度：指定地址前缀中有意义的部分。
- 量度：与 RIP_2 相同，跳数值为 1～15，不包括 15。

RIPng 更新中能包含的路由条数依赖于链路 MTU、在 RIPng 消息前面头信息的长度、RIPng 头的大小和路由表条目(RTE)的大小。下面公式决定在一个更新包中 RTE 的数量：

RTE 数量≈取整((MTU-IP 头大小-UDP 头长-RIPng 头长)/RTE 大小)

RTE 的数量与链路 MTU、IP 头长、UDP 头和 RIPng 头直接相关。

每一个 RIP-2 的 RTE 包含一个相关的下一跳字段，指定比宣告路由器更好的下一跳地址。IPv6 的地址如此之长，几乎将 RTE 的大小加倍。RIPng 指定一个单个的下一跳 RTE 应用到

下面所有的 RTE，直到消息的结束或另一个下一跳 RTE 的出现。图 8-17 中的下一跳 RTE 说明路由标记字段和前缀长度字段必须为 0。量度字段必须为 0xFF。地址字段 0:0:0:0:0:0:0:0 指示下一跳是该 RIPng 宣告的发送者。

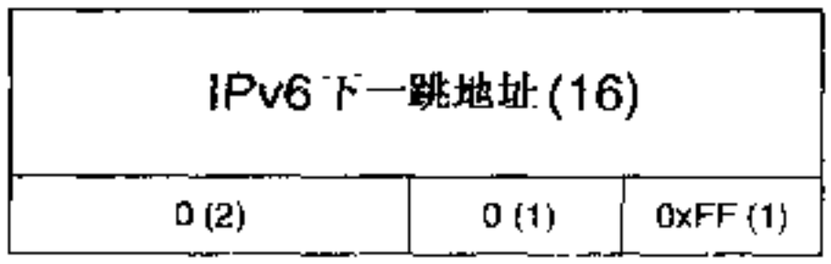


图 8-17　下一跳 RTE

下一跳地址必须是下一跳路由器的本链路地址。如果该地址不是本链路地址，宣告接收者将该地址作为 0:0:0:0:0:0:0:0 处理。

周期性的和触发的 RIPng 应答都应当保持在本地链路——它们不能穿过路由器。周期性更新触发更新的源地址必须是发起宣告的路由器的本链路地址，跳数限制等于 255。由于路由器会将每个包的跳数限制减 1，所以跳数限制 255 确保包没有穿过路由器。包的目标多播地址是所有 RIP 路由器的多播地址：FF01::9。

Cisco 路由器能运行多个 RIPng 进程。路由进程由下面的接口子命令使能：

ipv6 rip *tag* **enable**

该命令必须在所有配置了前缀需要在 RIP 更新中宣告的地址的接口上使能。多个进程由 tag 区分。当前支持最多 4 个进程。每一个进程必须使用不同的 UDP 端口号。如果路有一个进程则可以使用缺省端口 521。在其他进程中必须改变端口号，否则新进程无法启动。改变 RIPng 使用的端口号和多播地址的全局命令如下所示：

ipv6 rip tag port udp-port multicast-group multicast-address

多个进程可以是用同一个多播地址。如果没有配置该命令，则使用缺省端口号：521，和缺省多播地址：FF02::9。

与 RIP-2 不一样：RIP-2 中需要全局命令 **router rip** 来使能路由协议，在 IPv6 中不需要全局命令来使能协议。

一些可选的全局命令可以控制 RIPng 进程，影响所有的配置接口。可以使用全局命令来使能或禁止水平分割，反向毒化、改变 UDP 端口号和 RIPng 多播地址、改变缺省定时器、改变管理距离和注入静态路由。大多数功能可以在 RIPng 中使用。

表 8-10 罗列了可用的全局命令。

表 8-10　　RIPng 全局命令

命　令	描　述
[no] ipv6 rip *tag* **port** *udp-port* **multicast-group** *multicast-address*	配置 RIP 路由进程使用特定的 UDP 端口和多播地址
[no] ipv6 rip *tag* **table** *table-number*	将特定的路由表分配给某 RIP 路由进程，缺省是 0。注意，只有路由表 0 能用在 IPv6 单播包转发
[no] ipv6 rip *tag* **distance** *distance-value*	设置该进程的管理距离，缺省是 120

续表

命　令	描　述
[no] ipv6 rip *tag* **timers** *update expire holddown garbage-collect*	改变该 RIPng 进程定时器。数值单位是秒，缺省值 30，180，180，120
[no] ipv6 rip *tag* **reistribute static**	将静态路由注入到 IPv6，就好像他们是直连的一样
[no] ipv6 rip *tag* **split-horizon**	对更新过程使用水平分裂。是缺省值
[no] ipv6 rip *tag* **poison-reverse**	对更新过程使用毒化反转。缺省未设置

RIPng 还有一些接口子命令可以使用。这些接口子命令用作向特定接口发送更新消息中包含默认路由、总结接口上宣告的路由、对接口上收发的更新消息应用过滤器以及改变接口上收到路由的量度偏移量。这些功能在 RIP-2 中都有。表 8-11 罗列了接口子命令。

表 8-11　　RIPng 接口子命令

命　令	描　述
[no] ipv6 rip *tag* **enable**	在接口上配置 RIPng 路由
[no] ipv6 rip *tag* **default-information originate**	产生默认路由(0::0/0)，并将默认路由包含在该接口发出的更新消息中
[no] ipv6 rip *tag* **default-information only**	产生默认路由(0::0/0)，在接口上只发送默认路由
[no] ipv6 rip *tag* **summary-address** *prefix/length*	总结路由信息。如果路由前边指定长度比特匹配给定的前缀，则只宣告指定的前缀。这样多条路由可以使用一条单一路由替代，该路由的量度等于多条路由最小的量度。你可以多次使用该命令
[no] ipv6 rip *tag* **input-filter** *name*	在接口上收到的 RIP 更新消息上应用简单访问控制列表
[no] ipv6 rip *tag* **output-filter** *name*	在接口上发送的 RIP 更新消息上应用简单访问控制列表
[no] ipv6 rip *tag* **metric-offset** *number*	改变进入路由表的路由的量度偏移量。缺省是 1。该值可以是 1 和 16 之间

下面是一个简单网络图和路由器配置，用来帮助了解运行 RIPng 的最小路由器配置(见图 8-18)。

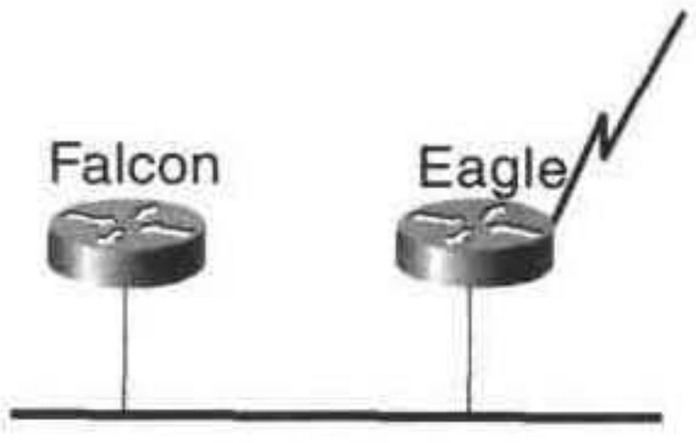


图 8-18　简单 RIPng 网络

在两台路由器的以太网接口和串行口上都配置了 RIPng。例 8-6 显示了路由器配置。

例 8-6　在路由器 Falcon 和 Eagle 上的 RIPng 配置

Falcon
```
ipv6 unicast-routing
no ipv6 rip birdbath split-horizon
!
!
interface Ethernet0
 no ip address
 no ip directed-broadcast
 ipv6 enable
 ipv6 address FEC0::/64 eui-64
 ipv6 address FEC0::1:0:0:0:0/64 eui-64
 ipv6 address FEC0::2:0:0:0:0/64 eui-64
 ipv6 rip birdbath enable
!
```

Eagle
```
ipv6 unicast-routing
no ipv6 rip birdbath split-horizon
!
!
interface Ethernet0
 no ip address
 no ip directed-broadcast
 ipv6 address FEC0::/64 eui-64
 ipv6 address FEC0::2:0:0:0:0/64 eui-64
 ipv6 address FEC0::3:0:0:0:0/64 eui-64
 ipv6 rip birdbath enable
!
interface Serial1
 ipv6 address FEC0::A:0:0:0:1/126
 ipv6 rip birdbath enable
!
```

两个路由器共享两个前缀：FEC0::/64 和 FEC0::2:0:0:0:0/64。每个路由器上都另外配置了第三个前缀。为了让路由器相互宣告不同的前缀，水平分裂被禁止。RIPng 在 Eagle 的以太网接口和串行接口 1 上使能。该进程名字为 birdbath。

例 8-7 显示了 Falcon 的路由表。

例 8-7　显示 RIPng 学习到的路由的 IPv6 路由表

```
Falcon#show ipv6 route
IPv6 Routing Table - 9 entries
Codes: C - Connected, L - Local, S - Static, R - RIP, B - BGP
Timers: Uptime/Expires

L FE80::/64 [0/0]
  via ::, Null0, 01:37:41/never
L FEC0::200:CFF:FE0A:2C51/128 [0/0]
  via FEC0::200:CFF:FE0A:2C51, Ethernet0, 01:20:58/never
C FEC0::/64 [0/0]
  via FEC0::200:CFF:FE0A:2C51, Ethernet0, 01:20:58/never
```

```
L FEC0::1:200:CFF:FE0A:2C51/128 [0/0]
  via FEC0::1:200:CFF:FE0A:2C51, Ethernet0, 01:01:36/never
C FEC0::1:0:0:0:0/64 [0/0]
  via FEC0::1:200:CFF:FE0A:2C51, Ethernet0, 01:01:36/never
L FEC0::2:200:CFF:FE0A:2C51/128 [0/0]
  via FEC0::2:200:CFF:FE0A:2C51, Ethernet0, 01:00:21/never
C FEC0::2:0:0:0:0/64 [0/0]
  via FEC0::2:200:CFF:FE0A:2C51, Ethernet0, 01:00:21/never
R FEC0::3:0:0:0:0/64 [120/2]
  via FE80::200:CFF:FE76:5B7C, Ethernet0, 00:00:08/00:02:51
R FEC0::A:0:0:0:0/126 [120/2]
  via FE80::200:CFF:FE76:5B7C, Ethernet0, 00:00:08/00:02:51
```

例 8-7 中的路由表显示出，Falcon 以太网接口上的前缀已经连接。Eagle 的以太网前缀 FEC0::3:0:0:0:0/64 和串口前缀 FEC0::A:0:0:0:0/126 都是通过 RIPng 进程学习到的。

RIPng 是一个非常容易实现的协议，引入多进程使 RIPng 比 RIP-2 更灵活；但是在第 1 卷第 7 章中表述的缺陷依然存在。例如它的跳数最大值依然很小，限制了运行该协议的网络规模。

3. 支持 IPv6 的 OSPF

OSPFv2 改变了许多设计来支持 IPv6 变大的地址，并且从 IPv4 到 IPv6 协议已有所改变。Cisco 的 IOS 还没有支持 IPv6 的 OSPF。OSPF 的基本机制：洪泛、DR 选举、区域支持、SPF 等没有改变。IPv6 OSPF 直接运行在 IPv6 上。标识 OSPF 的下一包头值是 89。在 IPv6 OSPF 中下面功能没有改变：

- 两个版本的协议都支持相同的包类型： Hello、数据库描述、链路状态请求、链路状态更新和链路状态确认包。其中一些类型包例如 Hello 包已被改变。
- 交换 Hello 包来交换邻居信息。
- 邻接关系选择和建立。
- 接口状态机，包括指派路由器选择过程中接口状态转移。
- 邻居状态机，包括成为邻接以前邻居状态转移。
- 链路状态数据器时间老化。

注：　路由 TCP/IP 第 1 卷第 9 章“开放最短路径优先”中讨论 OSPF 版本 2。

有一些机制已经作了改变。将 OSPF 变成网络层协议无关(变得更灵活)的努力引起了这些变化，新的地址格式，显式指出的洪泛范围，接口支持多个地址和前缀。OSPF 协议已变成网络协议无关。版本号由 2 改成 3，所以该协议在本章剩余部分称为 OSPFv3。本节指出协议中的变化。

1. 链路代替子网

IPv6 节点在链路上通信，而不是在子网上。IPv6 节点可以在连接到链路的接口上配置多个前缀和地址，可以不依赖于配置的子网在链路上相互通信。OSPFv3 关心链路，不像 OSPFv2 关心子网。由于 IPv6 OSPF 在每链路上运行而不是在每子网上，路由器发送的 OSPF 包的接口不再要求接收该包的接口在同一个子网上。

2. 删除地址符号

OSPFv3 从 OSPFv2 包和 LSA 中删除地址符号，建立一个网络协议无关的内核。该变化使 OSPF 走向将来的多协议 OSPF。许多 OSPFv2 包和 LSA 携带 IPv4 地址，用来表示路由器

ID、区域 ID 或 LSA 链路状态 ID。OSPFv3 的路由器 ID、区域 ID 和 LSA 链路状态 ID 仍然编码成 32 比特，所以不能使用 IP 地址表示(当然仍然能可以使用部分 IP 地址表示)。OSPFv2 广播和 NBMA 网络使用 IP 地址罗列邻居。OSPFv3 邻居由路由器 ID 表示。另外 OSPFv2 的 LSA 例如路由器 LSA 和网络 LSA 包含 IP 地址；使用 IP 地址表示链路状态数据库中的网络拓扑。OSPFv3 的路由器 LSA 和网络 LSA 路表示网络拓扑；他们使用网络协议无关的方式描述网络拓扑。IPv6 使用接口 ID 而不是来标识链路。路由器上每一个接口都分配一个唯一的接口 ID。某些实现可以使用 MIBII 的 ifIndex。MIBII ifIndex 在 RFC 2233 "the Interface Group MIB using SMIv2" 中讨论。邻居和指定路由器由路由器 ID 标识，不再使用 IP 地址。IPv6 地址仅包含在链路状态更新包所携带的 LSA 净荷中。

3. LSA 洪泛范围和未知 LSA

LSA 包的洪泛被推广了。LSA 的类型决定了 OSPFv2 的洪泛范围。每一种类型都关联自己的范围。在 OSPFv3 中洪泛范围在 LSA 头中明确配置。OSPFv3 路由器即使不认识 LSA 也知道如何洪泛。洪泛范围可能是本链路、区域或者 AS。OSPFv3 允许路由器拥有不同的能力。路由器不再被要求丢弃未知的 LSA。洪泛范围、未知类型的处理和 LSA 类型都编码在包头扩展的 LSA 类型字段。类型字段最高 3 个比特用于洪泛范围和未知类型处理的编码。处理比特告诉路由器对未知 LSA 是本链路洪泛还是正常存储洪泛，就好像该 LSA 是已知类型。由于类型中存在范围编码，所以路由器可以正常洪泛未知 LSA。表 8-12 和表 8-13 显示洪泛范围值和未知 LSA 处理相关值。

表 8-12　　洪泛范围值和描述

洪泛范围值(二进制)	描　　述
00	本链路范围，只在产生 LSA 的链路范围洪泛
01	本区域范围，只在产生 LSA 的区域范围洪泛
10	AS 范围，在 AS 内所有路由器洪泛
11	保留

表 8-13　　指示未知 LSA 处理的值

处理未知 LSA 值(二进制)	描　　述
0	作为本链路范围洪泛
1	假设该类型已知，存储且洪泛

对洪泛范围明确编码的机制是将新的 OSPF 特性集成到现存网络。

4. 每链路上多个 OSPF 实例

可以在单个链路上运行多个 OSPFv3 实例。当多个区域共享单一链路时，该方法被证明是有效的(见图 8-19)。OSPFv3 包头中的实例 ID 使该功能得以实现。图 8-19 表示两个路由器共享链路，两个区域需要运行在单一链路上；每链路上多个 OSPF 协议进程保障上述情况可行。

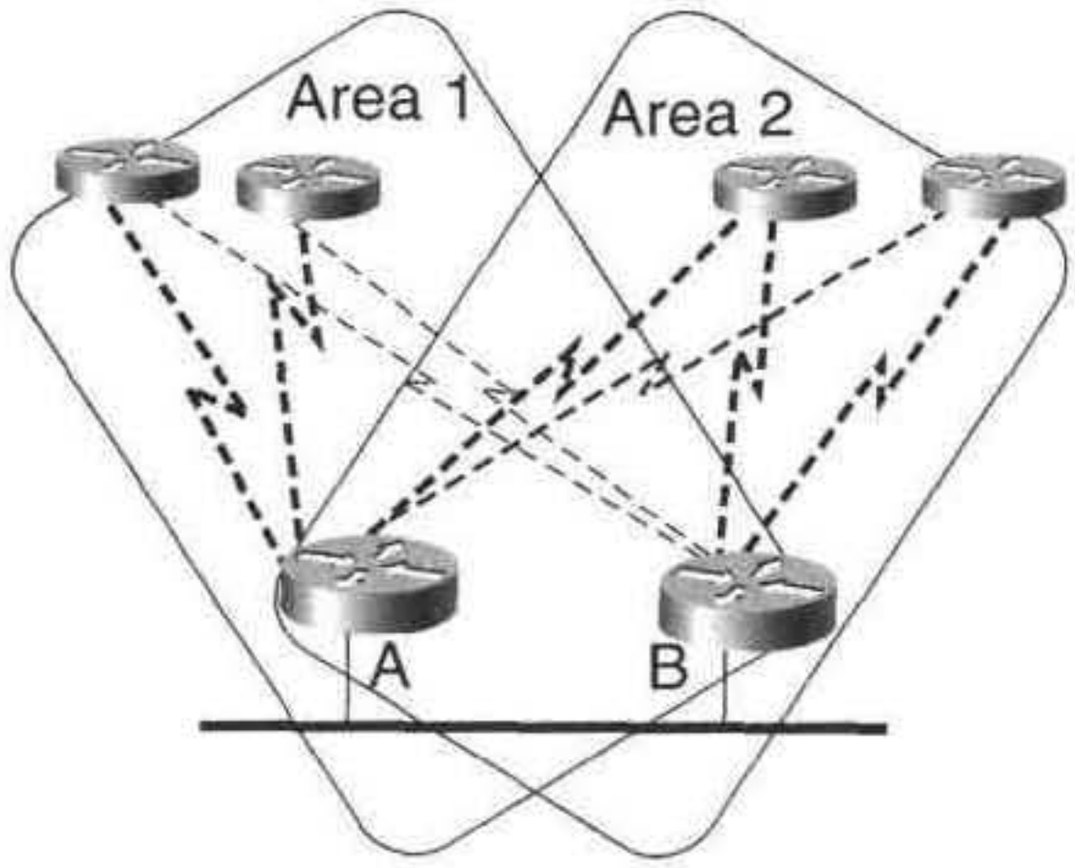


图 8-19　多个 OSPF 区域共享链路

在图 8-19 中，区域 1 和区域 2 都有 4 个路由器。区域 1 的两个远端路由器主用链路连接到路由器 A，备用链路连接到路由器 B。区域 2 的两个远端路由器与区域 1 相似，主用链路连接到 B，备用链路连接到 A。区域 1，2 必须运行在路由器 A，B 间的同一条链路上。你只能在 OSPFv3 中这样使用，不能在 OSPFv2 中这样使用。

另一个例子是多个公司或公司内多个独立部门在共享的单一链路上运行 OSPF，并且使用该链路相互通信。该链路可能属于其中一个公司，该公司使用上述链路连接各独立的组织。其中一个子集希望建立对等关系，不包含其他组织。一个常见的方法是使用 BGP 互连独立的组织。但是如果公司内只有 OSPF 专家，没有 BGP 专家，要求使用 OSPF 互连。图 8-20 显示了上述设定。

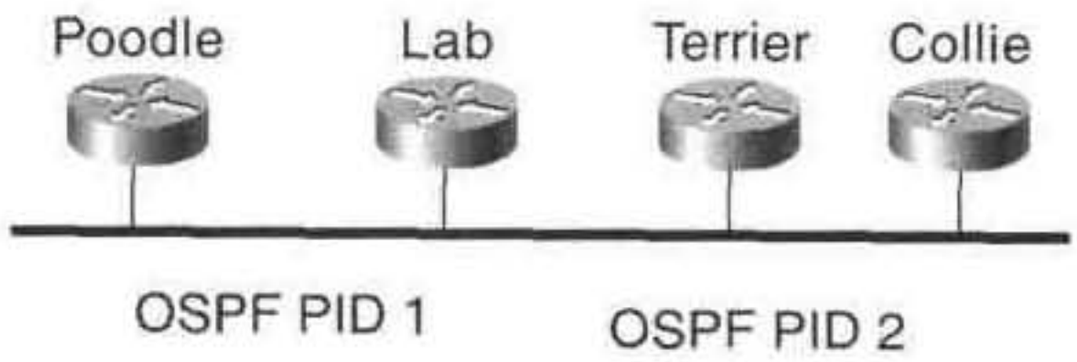


图 8-20　OSPF 路由器共享链路：路由器的子集共享 OSPF 进程

如果图 8-20 中路由器 Poodle 和 Lab 对等且共享共同的 OSPFv3 进程。路由器 Lab, Terrier 和 Collie 对等，共享另一个 OSPFv3 进程。Lab 的链路配置了两个 OSPFv3 进程标识。

5．OSPF 对本链路地址的使用

由于每一个 IPv6 路由器的活跃链路上都配置了本链路地址，OSPFv3 使用这些本链路地址作为协议包的源地址以及链路 LSA 的内容(在“新的 LSA 和 LSA 变化”一节中描述)。按照定义，所有的本链路地址都共享相同的 IPv6 前缀(FE80::/64)。这样所有的 OSPFv3 节点可以不管所分配的本站点地址或全球可聚合地址前缀是否相同，很方便地相互通信并建立邻接关系。在 LSA 内使用本链路地址来标识链路上的路由器能够避免将链路关联到特定的 IP 地址上，这样能保持拓扑信息与所使用的网络层协议无关。

6．删除认证

认证可以从 IPv6 OSPF 中删除。IPv6 对网络层协议内建了完整性、认证和保密机制。OSPFv3 直接在网络层以上运行。OSPFv3 从头中删除认证信息能够提高效率。不需要路由安全性的网络现在不需要处理这些头。需要路由安全性的节点可用 IP 层的 AH 和 ESP 扩展头。

7．新的 LSA 和 LSA 变化

尽管大多数功能保持不变，还是有一些 OSPFv2 的字段被改变，LSA 被改名。OSPF 中增加的新的 LSA 来携带 IPv6 地址和下一跳信息。

OSPFv2 LSA 中包含下列字段：使用期限、选项、类型、链路状态 ID、宣告路由器、序列号、校验和及长度。OSPFv3 将选项字段扩展到 24 比特，从头中移到路由器 LSA、网络 LSA、区域间路由器 LSA 和链路 LSA 内部。类型字段扩展到 16 比特，扩展使用选项字段留下的空间。头中其他内容保持不变。

LSA 类型字段中包含未知类型处理，洪泛范围和 LSA 类型比特。图 8-21 显示了 LSA 类型字段。

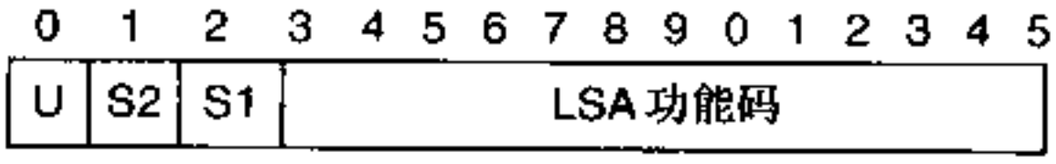


图 8-21　OSPFv3 LSA 类型字段

U 比特指定了未知 LSA 类型的处理。S2 和 S1 指示洪泛范围。

对未知 LSA 类型的处理方式有所改变。IPv4 OSPF 丢弃未知类型的 LSA。由于 OSPFv3 希望在单一链路上混合使用不同能力的路由器，所以不希望出现上述丢弃行为。如果指派路由器支持的选项比其他路由器少，则无法使用所有的功能。

表 8-14 罗列了每一个 LSA 的链路类型值。

表 8-14　　每个 OSPFv3 LSA 的链路类型值

LSA 功能代码	值	LSA 类型
1	0x2001	路由器 LSA
2	0x2002	网络 LSA
3	0x2003	区域间前缀 LSA
4	0x2004	区域间路由器 LSA
5	0x4005	AS 外部 LSA
6	0x2006	组成员资格 LSA
7	0x2007	类型 7LSA
8	0x0008	链路 LSA
9	0x2009	区域内前缀 LSA

从表 8-14 可以看到，由两个 OSPFv2 总结 LSA 被改名，增加了两个 LSA：链路 LSA 和区域内前缀 LSA。你还可以看到每一种类型的洪泛范围和处理方式。所有罗列 LSA 的 U 比特为 0，表示如果路由器不识别该类型则作为本链路洪泛范围。如果路由器识别该类型，则按照 S2 和 S1 比特规定的洪泛范围 LSA。例如路由器 LSA 的类型是 0x2001，S2,S1 比特值为 01(二进制)。该 LSA 将洪泛到区域内所有路由器。AS 外部 LSA 的范围值是 10(二进制)，表示洪泛到 AS 内所有路由器。

OSPFv2 类型 3 网络总结 LSA 改名为区域间前缀 LSA。记住这些 LSA 由区域边缘路由器使用，宣告区域外的网络。

类型 4 ASBR 总结 LSA 改名为区域间路由器 LSA。这些 LSA 由 AS 边缘路由器宣告，宣告区域外的 ASBR。

在 OSPFv3 中选项字段由 8 比特扩展到 24 比特。该字段出现在 Hello 包、数据库描述包和某些 LSA 中(路由器 LSA、网络 LSA、区域间路由器 LSA 和链路 LSA)。选项字段允许路由器相互通知他们所支持(或不支持)的能力，使不同能力的路由器能在同一 OSPF 路由域中共存。当路由器不支持某些能力时采用的行为由选项字段规定。

已定义如下 6 比特的选项字段：

- V6——如果为 0 表示路由器参加拓扑分发，但是不转发经过的 IPv6 包。
- E——和在 OSPFv2 中一样，E 比特设置表示发起路由器有能力接收 AS 外部 LSA。在末梢区域中发起的 LSA 中 E=0。该比特还用于 Hello 包中，指示接口收发 AS 外部 LSA 的能力。E 比特不匹配的路由器无法建立邻接关系，确保区域内所有路由器支持相同的末梢能力。
- MC——设置该比特表示发起路由器有能力转发多播包。MOSPF 使用该比特。
- N——仅用在 Hello 包中。设置该比特表示发起路由器支持 NSSA 外部 LSA。如果 N=0 表示发起路由器不接收或发送 NSSA 外部 LSA。E 比特不匹配的路由器无法建立邻接关系，确保区域中所有路由器支持相同的 NSSA 能力。如果 N=1，E 必须为 0。
- R——设置路由器比特(R)表示该路由器活跃。R 比特复位表示 OSPF 会话者愿意参加拓扑分发，但不愿意转发流量。该比特可以用在多宿主机，该主机希望加入路由工作，但是不希望作为路由器在多个接口间转发通信流。V6 比特是 R 比特的细化。如果 R 比特置位且 V6 比特复位，表示该节点不转发 IPv6 数据报但是转发其他协议的数据报。
- DC——该比特设置表示发起路由器有能力在按需建立的电路上支持 OSPF。

将这 6 个定义的比特与 OSPFv2 选项字段定义的比特(T,E,MC,N/P,EA,DC)比较，你可以看到发生的变化。在 OSPFv3 中不再支持服务类型(TOS)，所以 T 比特被替换。N 比特仍然使用，但只用于 Hello 中。P 比特是 OSPFv3 另一组选项的一部分，前缀选项关联在每一个宣告的前缀。OSPFv2 的 EA 比特指示是否支持外部属性 LSA。外部属性 LSA 用来替代内部 BGP(iBGP)，用作穿过 OSPF 与传输 BGP 信息。外部属性 LSA 没有被实现，也没有相关的草案或 RFC 出版。即使没有选项比特来定义 EA 能力，外部属性 LSA 仍然能作为附加 LSA 类型在规定洪泛范围和未知类型处理后由 OSPFv3 支持。

有一种新的链路 LSA，用在单一链路的路由器间交换 IPv6 前缀和地址信息。上述 LSA 还有路由器用作宣告一组有关本链路发起网络 LSA 的选项。链路 LSA 提供路由器本链路地址和相关该链路的前缀列表。该 LSA 作为多播发送到链路上所有路由器。由网络 LSA 宣告的选项是所有路由器发送的链路 LSA 中选项的逻辑或。

还有一种新的 LSA 称为区域间前缀 LSA。该 LSA 携带的 IPv6 前缀信息在 OSPFv2 中由路由器 LSA 和网络 LSA 携带。路由器使用上述 LSA 宣告分配给路由器自身的地址前缀，例如附加的末梢网络和附加的传输网络。

OSPFv3 LSA 中包含的前缀信息总是携带前缀长度，前缀选项和前缀地址。前缀选项是 8 比特字段，描述前缀相关能力。下面是已定义的 4 个选项：

- NU——“非单播”比特的设置将该前缀从单播路由计算中排除。
- LA——“本地地址”比特的设置表示该前缀是宣告路由器的真实地址。
- MC——“多播能力”比特的设置该前缀应当包含在多播路由计算中。
- P——NSSA 前缀上“传播”比特的设置表示该前缀应当在 NSSA 区域边界重新宣告。

每一个宣告的前缀都携带 8 比特前缀选项字段作为不同路由计算的输入。选项可以指示某些前缀应当排除，某些前缀应当传播。

4. BGP-4 多协议扩展

BGP-4 上的附加内容并不专门针对 IPv6。他们还包括对其他协议例如 IPX 的支持。BGP-4 的多协议支持在这里讨论是因为他们与 IPv6 相关。多协议 BGP(MBGP)在第 7 章“大范围 IP 多播路由”中讨论。

有 3 条 BGP-4 信息是针对 IPv4 的：

- 下一跳属性
- AGGREGATOR 属性
- 网络层可达性消息(NLRI)

在本书撰写时，假设每一个 BGP 会话者都维护至少一个 IPv4 地址。AGGREGATOR 属性仍使用上述地址。关于 AGGREGATOR 属性更多的信息请参考第 2 章。所以 BGP-4 的扩展只针对 NEXT-HOP 属性和 NLRI。另外由于下一跳属性用于向一组目的地转发包，仅用于增加 NLRI 不用于撤销路由，所以下一跳信息被增加到可达 NLRI 更新中。

为了使 BGP 支持多协议，定义了两个新的属性。他们是多协议可达 NLRI(MP-REACH-NLRI)和多协议不可达 NLRI(MP-UNREACH-NLRI)属性。他们都是可选、非传输属性，表示不识别上述属性的 BGP 程序可以快速忽略包含该属性的 Update 消息并不向对端宣告该信息。

正如名字的暗示，多协议可达 NLRI 属性描述的是可达的目的地。该属性包含信息有地址所属的网络层协议以及向下列目标前缀发包时所使用的下一跳地址。每一个 MP-REACH-NLRI Update 消息包含一个下一跳地址和相关 NLRI 的列表。NLRI 是<length/prefix>的二元组：length 是 prefix 的长度，prefix 是可达的 IPv6 地址前缀。

下一跳是向相关地址前缀转发包时 BGP 会话者所使用的地址。回顾第 2 章，下一跳属性缺省规则为：

- 如果正在进行路由宣告的路由器和接收的路由器在不同的自治系统中(外部对等)，NEXT_HOP 是正在宣告路由器接口的 IP 地址。
- 如果正在进行路由宣告的路由器和接收的路由器在同一个 AS(内部对等)内并且更新消息的 NLRI 指明目的地也在同一个 AS 内，那么 NEXT_HOP 就是已宣告路由的邻居的 IP 地址。
- 如果正在宣告的路由器和接收的路由器是内部对等实体并且更新消息的 NLRI 指明目的地在和它们在不同的 AS，则 NEXT_HOP 就是学习到路由的外部对等实体的 IP 地址。

对于 IPv6，由于 IPv6 地址范围的定义，规则变得更加明确。IPv6 BGP 路由器宣告下一

跳路由器的全球地址，可能后接一个本链路地址。只有在 BGP 会话者，NEXT_HOP 字段标识的节点，接收 Update 消息的对等体 3 者共享同一条数据链路时才会在全球地址后接一个本链路地址。在所有其他情况下，NEXT_HOP 字段只包含全球地址。

下面的网络图、路由器配置和命令输出说明 IPv6 的 MBGP 配置和输出与 IPv4 所使用的非常类似。

图 8-22 显示了简单的 BGP 路由器拓扑。

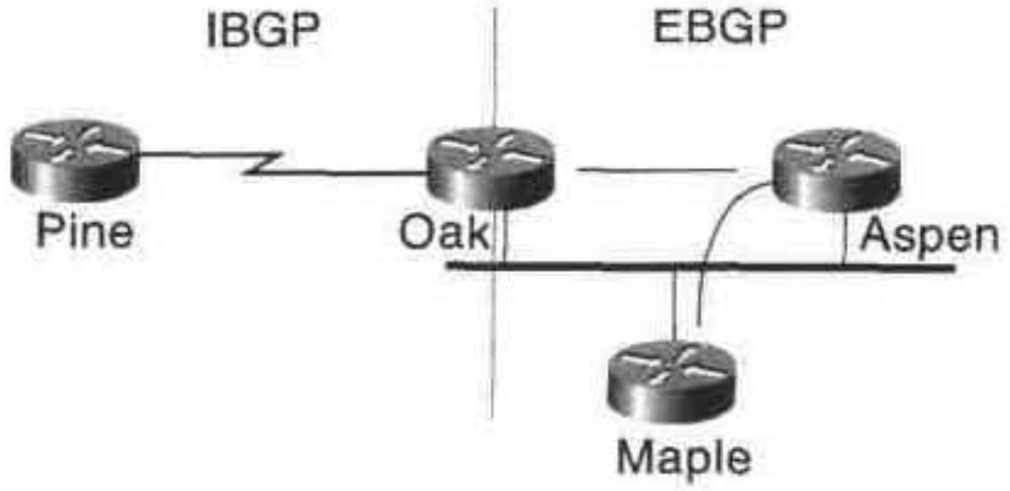


图 8-22　简单 BGP 网络

Maple 和 Aspen 是 E-BGP 对等体。Oak 和 Aspen 是 EBGP 对等实体。所有的路由器都在同一个快速以太网网段上。Oak 和 Pine 是 IBGP 对等实体。

Aspen 向 Oak 宣告一条从 Maple 学到的 NLRI。NLRI 包含 Maple 的地址作为下一条信息，所以 Oak 可以直接向 Maple 发送流量而不必通过 Maple 中转。由于上述 3 个路由器共享同一条快速以太网段，所以 Maple 的全球地址和链路层地址都包含在更新中。

Oak 向 Pine 宣告 Maple 的 NLRI。下一条地址是 Maple 的全球地址。本链路地址被删除。例 8-8 显示了 Oak 和 Aspen 的配置。

例 8-8　BGP 路由器配置

```
Oak
interface fastethernet 0
 description Oak to Aspen (e-bgp)
 ipv6 address 200A::2:0:0:0:1/64
!
interface serial 0
 description Oak to Pine (iBGP)
 ipv6 address 200A:0:0:10::1/124
!
interface serial 1
 description IGP link
 ipv6 address 200A:0:0:1::1/124
!
router bgp 100
 neighbor 200A::2:0:0:0:2 remote-as 300
 neighbor 200A:0:0:10::2 remote-as 100
!
 address-family ipv6
 neighbor 200A::2:0:0:0:2 activate
 neighbor 200A:0:0:10::2 activate
 network 200a:0:0:1::/124
 exit-address-family
```

```
Aspen
interface fastethernet 0
 description Oak to Aspen (e-bgp)
 ipv6 address 200A::2:0:0:0:2/64
!
router bgp 300
 neighbor 200A::2:0:0:0:1 remote-as 100
 !
 address-family ipv6
 neighbor 200A::2:0:0:0:1 activate
 exit-address-family
```

Oak 的快速以太网地址是 200A::2:0:0:0:1/64，正如你在接口子命令中所看到的。Oak 还有一个前缀是 200A:0:0:1::/124 的 IGP 链路。例 8-9 显示了 BGP 邻居的状态，从 Oak 向 Aspen 发送的关于 IGP 前缀的更新和 Aspen 路由表中的条目。

例 8-9　BGP 命令输出

```
Aspen#show bgp ipv6 nei
BGP neighbor is 200A::2:0:0:0:1,  remote AS 100, external link
  BGP version 4, remote router ID 172.16.255.1
  BGP state = Established, up for 00:00:18
  Last read 00:00:18, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received
    Address family IPv6 Unicast: advertised and received
  Received 40 messages, 0 notifications, 0 in queue
  Sent 51 messages, 0 notifications, 0 in queue
  Route refresh request: received 0, sent 0
  Minimum time between advertisement runs is 30 seconds

 For address family: IPv6 Unicast
  BGP table version 2, neighbor version 1
  Index 1, Offset 0, Mask 0x2
  1 accepted prefixes consume 64 bytes
  Prefix advertised 0, suppressed 0, withdrawn 0

  Connections established 4; dropped 3
  Last reset 00:00:43, due to User reset
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Local host: 200A::2:0:0:0:2, Local port: 11015
Foreign host: 200A::2:0:0:0:1, Foreign port: 179
```

你可以看到命令输出所显示的信息与 IPv4 中非常相似。事实上由于路由协议是 MBGP，不是为 IPv6 开发的新版本协议，我们唯一能期望看到的增加的输出是标注 IPv6 的地址族类型和 IPv6 地址格式。输出显示了这一点。地址族的值已经增加。地址类型不同。TCP 端口号相同：179。

8.5.6　泛播处理过程

Anycast 是用于域中将包路由到多个相同编址的节点中的一个节点的机制。相同编址的节点可能是一组向客户提供众所周知服务的服务器，或者属于同一 ISP 的一组路由器，要求

流量通过配置了 Anycast 地址路由器中的一个。节点将 IP 包寻址到组的单一 Anycast 地址。节点学习 Anycast 地址的下一跳地址，就好像使用单播地址一样。如果 Anycast 地址在本链路，则节点使用地址解析过程。所得到的第一个应答被加入到邻居缓存。如果地址不在本链路，包被按照路由协议的距离度量转发到最近的目的地。在域中应当有一个前缀包含一组 Anycast 节点。例如所有使用 Anycast 地址 FEC0::A:FDFF:FFFF:FFFF:FFFE/64 的节点在共同的前缀 FEC0:0:0:A::/64 中。所有 Anycast 节点必须被宣告成在域中使用上述前缀的主机路由。节点使用主机路由的量度来决定最近的 anycast 节点。你可以看到如果组内有许多 Anycast 组且组内有许多 Anycast 节点，并且包含这些组的域非常大的话，域内的路由表会变得非常大。

尽管 IPv6 规定了 Anycast 寻址，当前 Anycast 应用严格受限。在大范围使用 Anycast 的经验非常少，还存在一些因素使问题复杂化：例如如何确保会话中所有的包都到达同一的 Anycast 节点，要求所有的 Anycast 节点共享状态信息[4]等。在获得经验过程中需要解决许多问题。在所有 Anycast 子网路由器之外为已定义的 Anycast 组是移动 IPv6 归属代理地址。在问题的解决方案被通过以前，Anycast 寻址仅被限制在路由器。

8.5.7　多播

IPv6 使用且方便了使用多播。在有多播能力的链路上使用多播代替广播能够减少请求、宣告和更新等带来的影响。IPv6 通过下面机制促使多播广泛使用，支持区分范围的多播地址以及支持内建的数据链路组成员资格协议：收听者发现协议。协议无关多播(PIM)路由协议允许链路上的 IPv6 主机加入网络范围的多播组。

1．区分范围的地址

在 IPv6 的多播地址空间中加入了多播范围。应用和多播技术的使用可以在全球范围公众使用，也可以供本组织本站点或本链路使用。为有效应用多播范围，应当使用管理策略来标识站点或组织的边界。在定义的范围内可以包含公认的多播组，使多播应用的限制范围容易控制。

2．收听者发现

多播收听者发现(MLD)协议从 IGMPv2 发展而来，允许路由器发现链路上哪个主机希望接收多播包以及这些节点属于哪个多播组。然后这些信息被发送到网络上运行的多播协议例如 PIM。MLD 可以分解成两种功能：主机功能和路由器功能。

1．主机功能

主机功能类似于第 5 章“IP 多播路由介绍”中讨论的 IGMPv2 主机功能。定义了两类报告消息：

- 组成员报告(Membership Report)
- 完成报告(Done Report)

当主机开始收听链路上特定多播地址时，应当立刻报告路由器：链路上有收听者。主机向多播组发送报告，在报告包 MLD 多播地址字段包含多播组地址。报告的源地址是主机的本链路地址。源地址使本链路地址能够防止包传出本链路范围。

路由器通过周期性发送询问来决定链路上的主机属于哪个多播组。当主机听到一般询问(不涉及任何特定多播地址)后，为每个收听的多播组设置延时定时器，其中链路范围所有节

点地址和范围 0(保留)或范围 1(本节点)多播地址除外。主机听到对特定多播地址的询问时，路设置该地址的延时定时器。定时器设置成 0 到最大响应时间值间的随机值，最大响应时间值是询问中的一部分。对每一个地址的定时器都应设置不同的随机值。

如果主机在某地址定时器超时前没有听到本链路其他主机对该地址的报告，则主机自己发送报告。如果在超时前听到了报告，则不发送报告且重新设置定时器。这样链路上就不会充满所有组成员的报告，但是至少存在一个已知成员。

当路由器收到对特定多播地址的报告时，如果该多播地址不在路由器多播地址列表中，则路由器添加该地址且将添加通知运行的多播路由协议。如果该地址已存在，则路由器重置该地址的定时器为多播收听者间隔值。如果直到该定时器超时也没有收到对特定地址的报告，则将该地址从路由器列表中删除。

图 8-23 是 MLD 处理主机功能流程图。

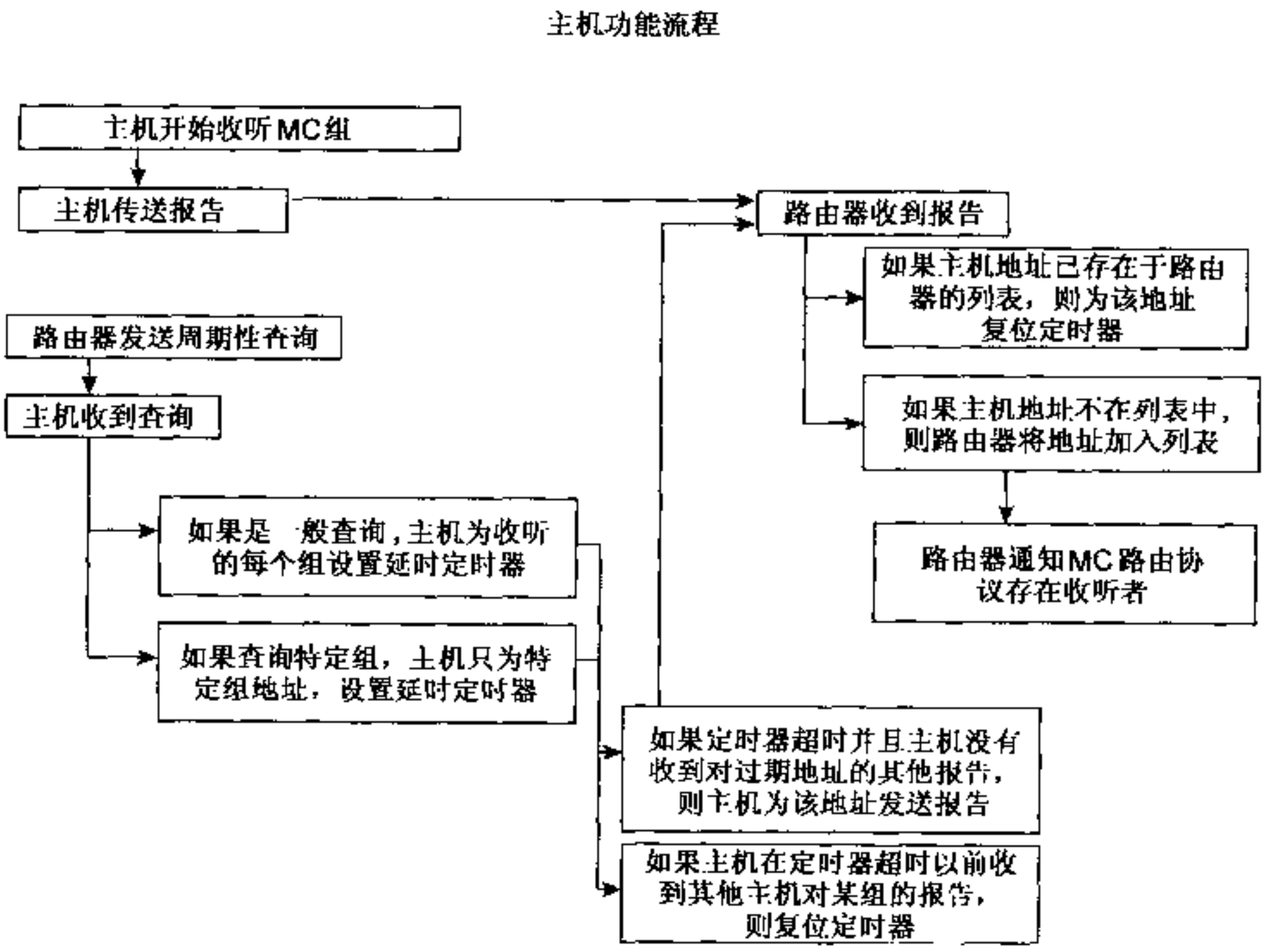


图 8-23　MLD 处理过程的主机功能

当主机结束收听某多播组时，应当发送 Done 消息。该消息相当于 IGMPv2 中的 Leave 消息。该消息发送给链路范围所有路由器多播组 FF02::2。消息的多播地址字段包含主机不再收听的多播地址。如果主机的最后一个报告由其他主机发出的报告所替代，则主机不需要发送 Done 消息，因为链路上可能还有另外一个节点收听该多播地址的数据。

2．路由器功能

MLD 的路由器功能也很类似于第 5 中讨论的 IGMPv2。术语只有一点不同。路由器发送多播收听者询问，该询问由两种子类型：

- 一般询问 (General Query)

- 特定多播地址询问 (Multicast-Address-Specific Query)

询问者与非询问者路由器的概念依然存在。路由器对每个多播链路设定询问者或非询问者状态。与 IGMPv6 中一样，初始时路由器假定自己是询问者并发送一般询问。如果路由器收到另一路由器的询问消息，则检查所收到消息的 IPv6 源地址。如果源地址数值小于自身地址则将变换角色，另一路由器作为询问者。如果自身地址较小则保持询问者状态。

询问者路由器在启动时和启动后周期性地发送一般询问来发现是否存在组成员。询问的源地址是路由器的本链路地址。询问发往链路范围所有节点多播地址 FF01::1。

当路由器收到完成消息时，如果完成消息所指地址存在于路由器列表中，则路由器发送特定多播地址询问来确定链路上是否还存在该地址收听者。如果在最大响应时延内没有主机响应，则路由器将该地址从列表中删除且通知多播路由部件。

图 8-24 显示了 MLD 路由器功能的处理流程。

路由器功能流程

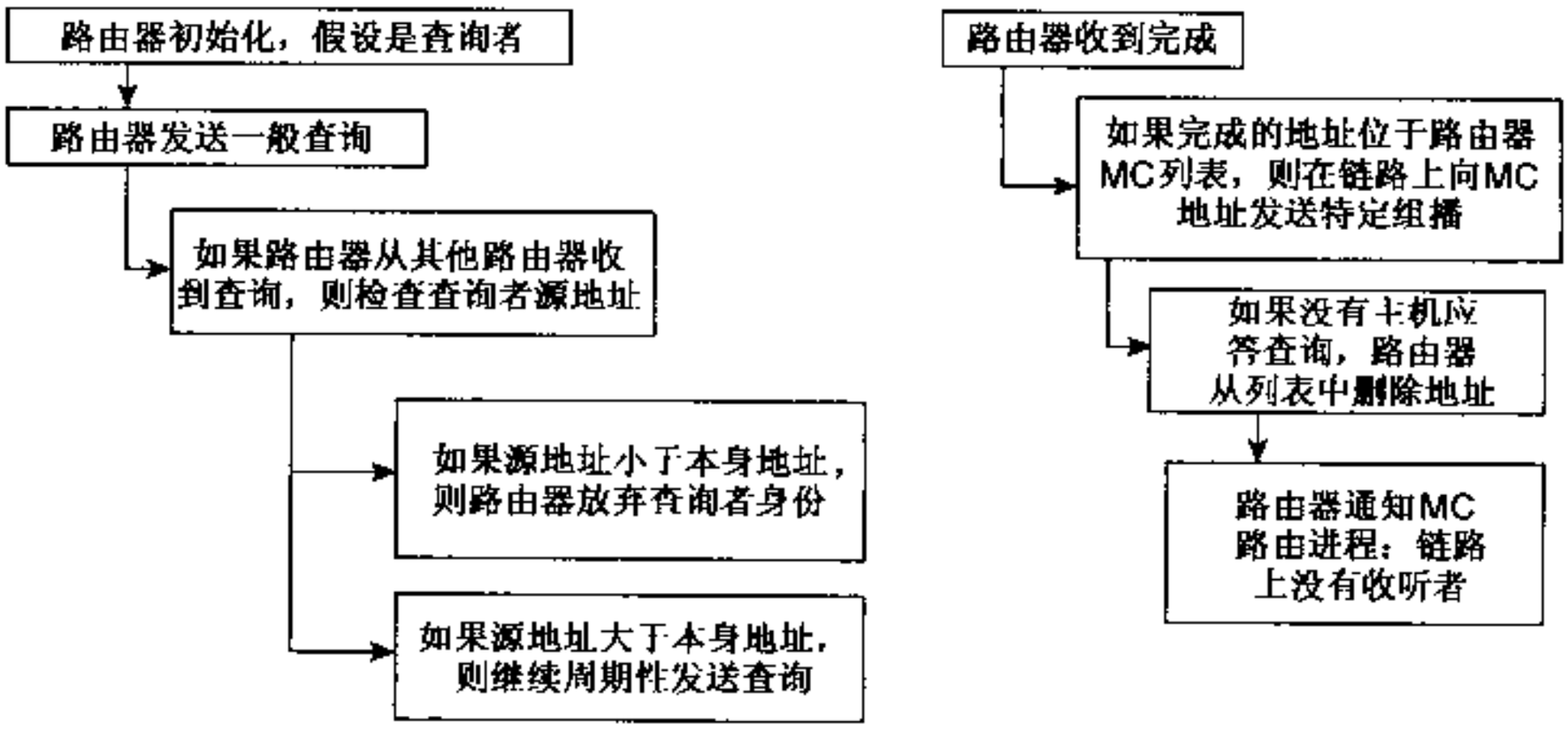


图 8-24　MLD 处理的路由器功能

3．PIM 多播路由

正如单播路由协议一样，多播路由协议被修改成支持 IPv6。从功能上说，协议还是以相同的方式运行。主要的修改是支持更大的地址空间。PIM 是当前唯一定义了 IPv6 修改的多播路由协议。PIM 和其他多播路由协议在第 5 章中曾详细讨论。

关于 IPv6 的修改定义了必须在 PIM 消息中使用的地址并标识了区域，该区域牵涉区分范围的多播地址和集中化的 bootstrap 机制。

在 IPv4 中，每一个不同的 PIM 消息使用多播或单播地址作为目的地址，分配的接口 IP 地址作为源地址。IPv6 地址将按地区分配，并且一条链路上可分配多个地址，选择哪一个地址使用还需进一步定义。

大多数消息使用全球 IPv6 所有 PIM 路由器多播地址：FF02::D 作为 IPv6 目的地址，发送接口的本链路地址作为源地址。其他消息使用需要与之通信的服务的全球 IPv6 单播地址作为目的地址，自身全球单播地址作为源地址。

在多播接口上使用 Hello 消息来发现 PIM 邻居。这些包的目的地址是所有 PIM 路由器多

播地址。源地址是接口的本链路地址。本链路地址用来建立邻居表以及选举指派路由器。

当路由器从某端口收到一多播包且路由器将该端口看作(源，组)或(S,G)的发送接口时，路由器发送 Assert 消息。回忆一下第 5 章，多播路由器为每个特定组((S,G)对)的每个特定源都维护一个上下游接口的多播转发表。如果路由器对某个(S,G)组从出接口(下游接口)收到多播包，该多播包是连接在下游链路的另一个路由器所转发的。图 8-25 说明了上面描述。

路由器 SJ 多播转发表特定(S,G)对显示 E0 是下游接口，所以是输出接口。SJ 从上述以太网接口收到上述(S,G)对的多播包。使用 Assert 消息来决定谁是该多路访问网络的 PIM 转发者。Assert 消息从接口的本链路地址发往所有 PIM 路由器多播地址。使用本链路地址值来打破选择过程平局，本链路地址数值较大的路由器称为转发者。下游路由器存储转发者的本链路地址用作解析未来的 RPF 请求。

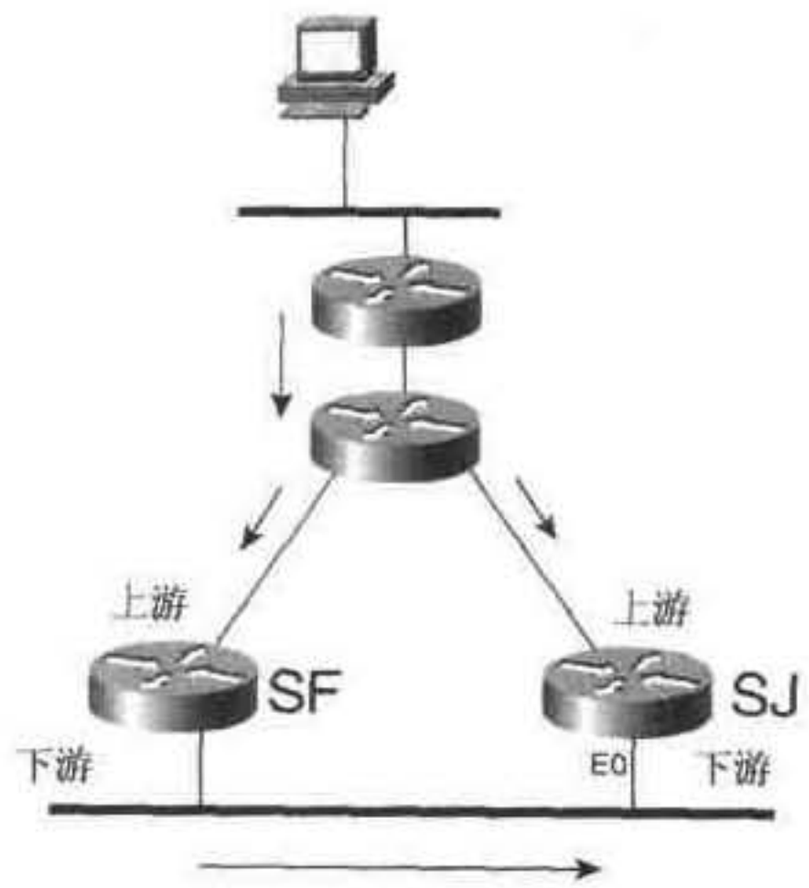


图 8-25　在下游接口收到多播包

用来建立和剪除多播路由器转发表的消息：Join/Prune, Graft 和 Graft-Ack 都使用本链路地址作源地址，所有 PIM 路由器多播地址作目的地址。所有消息都包含上游邻居的地址。上游邻居地址设置成邻居的本链路地址。使用 RPF 查找来得到上述地址。如果无法得到邻居的本链路地址，使用邻居已知的全球地址。

另一个使用本链路地址作源地址所有 PIM 路由器多播地址作目的地址的消息是 BootStrap 消息。BootStrap 消息是 BootStrap 路由器(BSR)对所有 PIM 路由器的多播。BootStrap 路由器的地址包含在消息中。由于该地址必须能由所有 PIM 路由器访问，所以该地址必须是 BootStrap 路由器能由全域范围可访问的地址。

Register 和 Register Stop 消息用在 PIM 稀疏模式中。希望向多播组发送流量的源指派路由器(DR)首先将多播包封装在 Register 消息中发往集合点(RP)。RP 向 DR 发送 Register Stop 消息告诉源停止在 Register 消息中封装多播包。这些事件不需要是连续的。第 5 章描述了这些事件的完整顺序。Register 和 Register Stop 消息的目的地都是集合点路由器的全域可访问单播地址。源地址是 DR 的全域可访问单播地址。源 DR 从 BootStrap 路由器多播到所有 PIM 路由器的 RP-set 消息中得到 RP 地址。RP 从 DR 发出的 Register 消息的源地址得到 DR 的全球 IPv6 地址。

所有的 RP 候选者都向 BootStrap 路由器发送一个单播的 Candidate-RP-Advertisement 消息。该消息包含发宣告的路由器作候选 RP 的多播组地址。该宣告还包含该路由器作为 RP 路由器的 IPv6 地址。Candidate-RP-Advertisement 消息的目的地址是 BSR 的全域可访问单播地址。源地址是候选 RP 的全域可访问单播地址。BSR 从这些宣告中得到 RP-set。

受限多播地址解决了多播范围限制的问题；但是他们对 PIM 和 BootStrap 机制产生影响。BootStrap 机制是在 PIM 域中的集中化过程。从中心的 BSR 发出的 BootStrap 消息希望能到达所有的 PIM 路由器。如果 PIM 域不是受限多播地址域的子集，则 BootStrap 机制无法工作。在一个受限地址域中的多播包无法传播到另一个受限地址域。结论是如果要使 BootStrap 机制工作，PIM 域必须是受限多播地址域的子集或者所有的多跳消息必须是全球可达 IPv6 地址。

8.5.8　服务质量

IPv6 中没有设计服务质量功能，例如描述通过路由器转发不同流量类的排队或转发方式，对多个流区分优先级等，但是存在允许这些协议与 IPv6 协同工作的机制。两个这样的机制是下面章节描述的 IPv6 头中的 Traffic Flow 和 Traffic Class 字段。

1．Traffic Flow

发送流量的节点可能希望对不同的通信流请求特殊处理。节点可以标记该流，要求 IPv6 路由器对该流提供非缺省的 QoS。例如呼叫中心应用要求迅速响应，呼叫中心典型使用的应用是在打电话的人说话时提供从服务器得到的信息。节点可以标识这样的流，请求获得与其他流不同的 QoS。

2．Traffic Class

为源节点和/或中间路由器提供的 IPv6 中的流量类比特能区分 IP 包不同的类或优先级。这些比特可以和 IPv4 中服务类型比特相同的方法使用，这些方法现在正实验中。区分服务(DiffServ)重定义了 Traffic Class 字段，称为 DS 字段。IPv6 和 IPv4 中 DS 字段的定义相同。最左面 6 比特作为区分服务码点。包在网络的边缘获得码点。码点决定了每个路由器对该包作排队和转发的行为。该行为称为每一跳行为(PHB)。

8.6　从 IPv4 向 IPv6 过渡

如果没有明确的过渡方式，新的路由协议无法实施。过渡程序越简单，新的协议越有可能被实施。IPv6 和 IPv4 互操作是必要的。至少在开始时 IPv6 节点需要与 IPv4 节点通信，但有可能是永远需要。IETF NGTRANS 工作组开发了一系列方法来方便过渡，确保兼容。

由许多不同的方法可以兼容 IPv4。节点可以运行双协议栈，完全实现 IPv4 和 IPv6。节点可以使用 IPv4 或 IPv6 通信。节点可以将 IPv6 封装到 IPv4 中，在现存的 IPv4 网络中建立一个隧道，允许两个 IPv6 节点通信。有两种隧道机制：

- 自动隧道
- 配置隧道

现已定义了一种兼容 IPv4 的 IPv6 地址：前 96 比特为 0，后 32 比特包含 IPv4 地址。例

如::172.69.1.1 是 IPv4 兼容地址。配置了 IPv4 兼容地址的节点使用自动隧道。

对 IPv4 和 IPv6 共存的网络，必须提供正确解析名字与 IP 地址的机制。定义了修改的 DNS 使 DNS 服务器能正确返回 IPv6 地址或 IPv4 地址(或所有地址)。上述能力是协议能共存的关键部分。

或者在 IPv6 网络和 IPv4 网络间放置网络地址翻译—协议翻译(NAT-PT)设备。首先讨论双协议栈。

8.6.1　双协议栈

对节点实现 IPv6 且保持兼容 IPv4 的方法是同时完全实现 IPv4 和 IPv6。完全实现两个协议站的节点称为 IPv6/IPv4 节点。IPv6/IPv4 节点可以使用 IPv6 包和 IPv6 节点通信，使用 IPv4 包和 IPv4 节点通信。

IPv6/IPv4 节点必须同时配置 IPv6 地址和 IPv4 地址。上述两个地址可能相关也可能不相关。IPv4 兼容地址可以看作一个地址既可以用作 IPv6 地址也可以用作 IPv4 地址。完全的 128 比特代表 IPv6 地址，低 32 比特代表 IPv4 地址。

你可以使用许多方式配置地址：

- 你可以使用无状态或基于状态(IPv6 DHCP)自动配置方式配置 IPv6 地址。所配置的地址可以是 IPv4 兼容地址或者纯 IPv6 地址。
- 你可以使用任何 IPv4 机制获得 IPv4 地址。
- 你可以使用 IPv4 机制获得的 IPv4 地址配置 IPv4 兼容地址。节点会通过添加 96 比特前缀 0:0:0:0:0:0 将 IPv4 地址映射为 IPv4 兼容地址。在 IPv6 路由器或地址配置服务器可用以前安装 IPv6/IPv4 节点时，该方法被证明非常有效。

一个同时拥有 IPv6 地址和 IPv4 地址的节点必须有一种机制来决定什么时候使用哪一个地址。DNS 提供这种机制。

8.6.2　DNS

为 IPv6 定义了一种新类型的资源记录——AAAA 记录。该记录提供从名字到 IPv6 地址的映射。IPv6/IPv4 节点的 DNS 解析器必须既能处理 IPv4 的 A 资源记录又能处理 IPv6 的 AAAA 资源记录。当节点向 DNS 服务器请求地址时，返回一个 A 记录或者 AAAA 记录。返回地址的类型决定所使用的协议。如果返回 A 记录，节点使用 IPv4 地址以及 IPv4 协议与请求的目标节点通信。如果返回 AAAA 记录，使用 IPv6。

当 IPv6/IPv4 节点分配 IPv4 兼容地址时，DNS 中同时定义 A 记录和 AAAA 记录。AAAA 记录列出了 IPv6 128 位地址，A 记录列出地址的后 32 比特。由于罗列了两种地址，IPv6 节点可以询问服务器得到 IPv6 地址，IPv4 节点可以得到 IPv4 地址。

当前如果对 IPv4 兼容地址同时罗列 A 记录和 AAAA 记录，DNS 解析器可以选择返回的内容，所返回的内容影响通信使用的协议：

- 仅向应用返回 IPv6 地址。
- 仅向应用返回 IPv4 地址。
- 向应用返回所有地址。

返回的地址和地址的顺序影响所产生 IP 流量的类型。

8.6.3　IPv4 中的 IPv6 隧道

许多 IPv6 网络都在 IPv4 网络边缘实现。IPv6 主机通常在 IPv4 网络上通信。在 IPv4 包中封装 IPv6 支持上述实现。你可以建立 4 种类型的通道：

- 路由器到路由器
- 主机到路由器
- 主机到主机
- 路由器到主机

IPv6/IPv4 路由器能将 IPv6 流量封装到 IPv4 网络上传输。你可以使用这种方式为路由器两端只支持 IPv6 的节点服务，也可以用于端到端的 IPv6 路径需要穿过 IPv4 网络的情况。数据源向 IPv6 路由器发送 IPv6 包。该路由器作为隧道源节点将 IPv6 包封装到 IPv4 中再将 IPv4 包发送到隧道终点。隧道终点路由器将包解封装以后将 IPv6 包转发到最终目的地。

IPv6/IPv4 主机可以发起一条隧道到达 IPv6/IPv4 路由器。该隧道是 IPv6 路径的第一段。发起节点将 IPv6 包封装到 IPv4，沿隧道将 IPv4 包转发到终点路由器。终点路由器解封装以后将 IPv6 包转发到最终目的地。

IPv6/IPv4 主机可以建立一条隧道到达另一 IPv6/IPv4 主机。这是完全端到端的隧道。IPv6/IPv4 源节点将 IPv6 包封装到 IPv4 中，将包通过纯 IPv4 网络转发到目标主机。目标主机收到 IPv4 包，解封装以后处理 IPv6 包。

路由器到主机的隧道是 IPv6 路径中最后一段。路由器收到 IPv6 包以后建立隧道，将包沿互联的 IPv4 网络转发到目标主机。目标主机接收 IPv4 包，接封装后处理 IPv6 包。

前面两种方式：路由器到路由器和主机到路由器隧道没有全程到达最终目的地。包的隧道终点和最终目的地不同。隧道终点与最终目的地址不同。隧道终点需要 IPv4 地址，但是无法从 IPv6 目的地址中得到上述信息。上述方法需要配置隧道。

后两种方法中隧道的终点与最终目的地相同。隧道终点的 IPv4 地址可以从 IPv4 兼容的 IPv6 地址的后 32 比特得到。所以在后两种方式中你可以使用自动隧道机制。

1．配置隧道

通过在 IPv6 和 IPv4 网络边缘的路由器上创建隧道接口，可以在 Cisco 路由器间建立隧道。两段路由器上都定义隧道的终点。创建了一个 IPv6 子网，并在两端路由器配置 IPv6 地址。如果使用 IPv6 动态路由协议例如 RIPng 或 BGP，将路由协议在隧道接口上使能。图 8-26 显示了通过 IPv4 网络互连的两个 IPv6 网络。两个 IPv6 网络间配置了一条隧道相互通信。

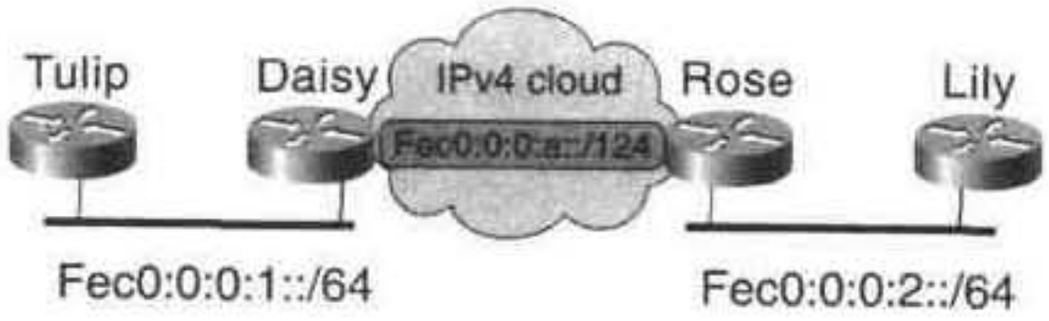


图 8-26　IPv6 网络通过隧道穿过 IPv4 网络的连接图

Tulip 和 Lily 是只支持 IPv6 的路由器。他们通过 Daisy 和 Rose 间的 IPv6-over-IPv4 隧道通信。

例 8-10 显示了 Daisy 和 Rose 的配置。

例 8-10　配置了隧道的路由器配置

```
Daisy
ipv6 unicast-routing
!
interface Tunnel0
 description tunnel Daisy -> Rose
 no ip address
 no ip directed-broadcast
 ipv6 address FEC0::A:0:0:0:1/124
 ipv6 rip flowerpot enable
 tunnel source Serial1.503
 tunnel destination 172.69.255.250
 tunnel mode ipv6ip
!
interface Ethernet0
 ipv6 address FEC0::1:0:0:0:0/64 eui-64
 ipv6 rip flowerpot enable
!
interface Serial1.503 point-to-point
 ip address 172.69.255.254 255.255.255.252
```

```
Rose
ipv6 unicast-routing
!
interface Tunnel0
 description tunnel Rose -> Daisy
 no ip address
 no ip directed-broadcast
 ipv6 address FEC0::A:0:0:0:2/124
 ipv6 rip flowerpot enable
 tunnel source Serial1.703
 tunnel destination 172.69.255.254
 tunnel mode ipv6ip
!
interface Ethernet1
 no ip address
 ipv6 address FEC0::2:0:0:0:0/64 eui-64
 ipv6 rip flowerpot enable
!
interface Serial1.703 point-to-point
 ip address 172.69.255.250 255.255.255.252
```

隧道接口是一般隧道，配置成 ipv6ip 模式。

图 8-11 中从 Tulip 到 Lily 的 **traceroute** 命令显示出 IPv6 包穿过隧道。

例 8-11　从 Tulip 到 Lily 的 **traceroute** 命令的显示

```
Tulip#traceroute ipv6 fec0:0:0:2:210:7bff:fe3a:ce8a

Type escape sequence to abort.
Tracing the route to FEC0::2:210:7BFF:FE3A:CE8A

  1 FEC0::1:200:CFF:FE0A:2AA9 8 msec *  4 msec
  2 FEC0::A:0:0:0:2 24 msec *  16 msec
  3 FEC0::2:210:7BFF:FE3A:CE8A 28 msec *  20 msec
```

第一个地址是 Daisy 的 IPv6 以太网地址。第二个地址是 Rose 的隧道接口地址。第三个地址是 Lily 的以太网地址。

配置隧道提供了一个通过 IPv4 网络连接 IPv6 网络的直接方法。

2. 自动隧道

配置成自动隧道的封装主机从目的地的 IPv4 兼容地址得到 IPv4 地址。该 IPv4 地址是隧道的终点。封装主机必须能与表示在 IPv4 兼容地址中的 IPv4 地址互连。源主机将包封装到 IPv4 中，使用得到的 IPv4 地址作目的地址，使用从自己 IPv4 兼容地址中得到的 IPv4 地址作源地址。主机之间的路由器对净荷中的 IPv6 内容一无所知。

8.6.4　网络地址翻译—协议翻译

另一种使 IPv6 和 IPv4 网络和主机共存的方式是使用网络地址翻译—协议翻译(NAT-PT)。IPv6/IPv4 路由器不在只支持 IPv4 和只支持 IPv6 的主机间翻译。主机通信时必须确保使用相同版本的 IP 协议。配置了 NAT-PT 的路由器能作所有的翻译工作。源地址和目的地址都在 IPv4 和 IPv6 间作翻译。

在 IPv4 NAT 中存在的问题同样存在于 IPv6 到 IPv4 的 NAT-PT。IPv4 和 IPv6 域间翻译的带内或带外的流量必须通过同一个地址翻译器。地址翻译器维护被翻译会话的状态信息。端到端的安全性无法保证。IPSec 无法穿过地址翻译器工作。如果应用在非 IP 头的位置携带 IP 地址，除非在翻译路由器上运行应用翻译网关，否则应用无法工作。对穿过协议域的 DNS 查询，DNS 包中请求和应答信息必须在 IPv4 和 IPv6 间翻译。

除地址以外，IPv4 和 IPv6 翻译的一个特别问题是头信息。正如你所知，IPv6 头中包含的字段与 IPv4 头包含的不一样。选项处理非常困难。在两个域间作翻译意义不大，你应当在实在没有其他办法时才使用地址翻译。

8.7　尾注

[1]IEEE, "Guideline for 64-bits Global Identifier (EUI-64) Registragion Authority," http://standards.ieee.org/regauth/oui/tutorials/EUI64.html (1997 年 3 月)

[2]R. Hinden 与 S. Deering, "RFC 2375: Ipv6 Multicast Address Assignments" (工作正在进行中)

[3]T. Narten, E.Nordmark 与 W. Simpson, "RFC 2461: Neighbor Discovery for IP Version 6 (IPv6)" (工作正在进行中)

[4]C. Partrdge, T. Mendez 与 W. Milliken, "RFC 1546: Host Anycasting Service" (工作正在进行中)

8.8　展望

IPv6 允许互联网扩展到非常大。它还简化了主机的配置管理。许多主机配置都可以从路由器配置得到。第 9 章“路由器管理”中将讨论的路由器管理能提供安全的配置来使路由器可靠优化地运行。

8.9　推荐书目

Huitema, C. IPv6: The New Internet Protocol. Upper Saddle River, New Jersey: Prentice Hall PTR; 1996.

尽管该书有一点过时(一些 RFC 成为了草案标准，一些 RFC 被创建，一些被改写)，但是该书提供了非常好的技术讨论及对设计考虑的真知灼见。

8.10　复习问题

1．对 60 比特前缀的地址 200A 0000 0000 0C00 0000 0000 0000 0000 下面哪儿种表示是正确的？

A．200A:0000:0000:0C/60

B．200A::0C00:0:0:0:0/60

C．200A:0000:0000:0C00::/60

D．200A::0c00::/60

E．200A:0:0:C00::/60

F．200A::0C/60

2．0:0:0:0:0:0:0:0 用在哪里？

3．你配置连接到 IPv6 公网的站点边缘路由器，广播所有的内部网络，包括 FEC0:0020:0:0100::/56。你接到网络管理员愤怒的电话。出什么问题了？

4．从源到目的地的路径中，哪一个扩展头各节点都要处理？

5．使用哪一种扩展头来指定到达目的地之前访问一组路由器，并由这些路由器处理该头？

6. 如果路由器收到一个包大于出口链路的 MTU。路由器是否将包分片后转发？

7. 在路由器宣告中，管理比特设置代表什么？

8. 如果路由器在 RA 中宣告前缀消息，这些消息如何使用？

9. 主机中的 IP 地址可以在那两种状态？这两种状态如何使用？

10. 路由器在 RA 中要求主机替你使用某个前缀发起会话时宣告什么消息？

11. 如果节点有一个邻居状态是 DELAY，能否向该邻居发包？

12. 如果主机上没有运行任何路由协议，路由器通过默认路由器向源端节点发数据。默认路由器故障时，节点会不会向故障路由器发数据直到 TCP 连接中断？

13. 多播包的范围是什么？如何使用？

14. Cisco 路由器使能 IPv6 路由的命令是什么？

15. 在接口上使能 IPv6 的接口子命令是什么？

16. 使能 RIPng 过程的命令是什么？

17. 在邻居间如何使能 BGP-for-IPv6 过程？

8.11 参考文献

Atkinson, R., and S. Kent, "IP Authentication Header," RFC 2402, November 1998.

Atkinson, R., and S. Kent, " IP Encapsulating Security Payload (ESP)," RFC 2403, November 1998.

Atkinson, R., and S. Kent, "Security Architecture for the Internet Protocol," RFC 2401, November 1998.

Bates, T., R. Chandra, D. Katz, and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 2283, February 1998.

Coltun, R., D. Ferguson, and J. Moy, "OSPF for IPv6," RFC 2740, December 1999.

Conta, A., and S. Deering, "Generic Packet Tunneling in IPv6 Specification," RFC 2473, December 1998.

Conta, A., and S. Deering. "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," RFC 1885, December 1995.

Deering, S., and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 1883, December 1995.

Deering, S., W. Fenner, and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6," RFC 2710, October 1999.

Gilligan, R., and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers," RFC 1993, April 1996.

Haberman, B., H. Sandick, and G. Kump, "Protocol Independent Multicast Routing in the Internet Protocol Version 6 (IPv6)." draft-ietf-pim-ipv6-03.txt, March 2000.

Hinden, R., M. O'Dell, and S. Deering, "An Aggregatable Global Unicast Address Format," RFC 2374, July 1998.

Hinden, R., and S. Deering, "IP Version 6 Addressing Architecture," RFC 2373, July 1998.

Hinden, R., and S. Deering, "IPv6 Multicast Address Assignments", RFC 2375, July 1998.

Hinden, R., S. Deering, R. Fink, and T. Hain, "Initial IPv6 Sub-TLA ID Assignments," <draft-ietf-ipngwg-iana-tla-03.txt>, January 13, 2000.

Hubbard, K., M. Kosters, D. Conrad, D. Karrenberg, and J. Postel, "Internet Registry IP Allocation Guidelines," BCP 12, RFC 1466, November 1996.

IAB and IESG, "IPv6 Address Allocation Management," RFC 1881, December 1995.

IEEE, "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority", http://standards.ieee.org/db/oui/tutorials/EUI64.html, March 1997.

Johnson, D., and S. Deering, "Reserved IPv6 Subnet Anycast Addresses," RFC 2526, March 1999.
McCann, J., S. Deering, and J. Mogul, "Path MTU Discovery for IP version 6," RFC 1981, August 1996.
Malkin, G., and R. Minnear, "RIPng for IPv6," RFC 2080, January 1997.
Marques, P., and F. Dupont, "Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing," RFC 2545, March 1999
Narten, T., Nordmark, E. and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.
Partridge, C., T. Mendez, and W. Milliken, "Host Anycasting Service," RFC 1546, November 1993.
Reynolds, J., and J. Postel, "Assigned Numbers," STD 2, RFC 1700, October 1994. (see also www.iana.org/iana/assignments.html)
Thomson, S., and C. Huitema, "DNS Extensions to support IP version 6," RFC 1886, December 1995.
Thompson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration," RFC 1971, August 1996.
Thomson, S., and T. Narten, "IPv6 Stateless Address Autoconfiguration," RFC 2462, December 1998.
Tsirtsis, G., and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)," RFC 2766, February 2000.

第 9 章　路由器管理

- **规则和程序定义**——清晰的规则和程序定义是维护一个良好运行的网络所必需的。服务等级协议改变管理规则，增加所需要的程序。
- **简单网络管理协议**——SNMP 协议提供网络管理协议的基础。如果你想在网络上使用网络管理应用，理解该协议非常重要。
- **RMON**——RMON 提供附加的网络管理能力。如果你想在网络上使用基于 RMON 的网络管理能力，理解该协议非常重要。
- **日志**——记录 Cisco 路由器上所发生事件的信息，当网络发生可能相关路由器的问题时提供有价值的资料。
- **系统日志**——将日志记录到 syslog 服务器，为路由器事件信息提供集中的存储，可以收集多个网络设备的信息。
- **网络时间协议**——网络时间协议同步相关网络设备的时钟，使分析事件更容易。
- **记账**——在网络设备上的记账时收集数据，你可以使用这些数据理解通信流或者按照通信流对网络使用者计费。
- **配置管理**——配置管理是维护配置文件有效及可用的一组工具、程序和规则。
- **故障管理**——故障管理系统通知网络管理员在网络某处发生了故障。故障可能是使某些网络服务不可用的事件，也可能是网络性能可能受影响的指示。
- **性能管理**——性能管理系统收集数据用作对网络趋势和能力的计划。
- **安全管理**——路由器安全管理确保路由器的完整性。存在不同的工具和配置参数防止路由器安全受威胁。
- **设计服务器来支持管理程序**——支持管理程序的服务器提供了网络上的眼睛，所以必须安全健壮，并且对管理的网络有接入有冗余。
- **网络健壮性**——网络健壮性要求包括 LAN 和从终端节点离开 LAN 到网络其他部分的路由。HSRP 提供了缺省路由的冗余，是配置单一缺省路由的终端节点能可靠地使用网络。
- **实验室**——网络实验室用来测试设计、计划、新硬件、新操作系统、新协议和新程序等。实验室应当尽可能体现生产网，以便进行有效的测试。

在本书中，您已经读到了通过网络和路由器转发 IP 包的方式。只有在网络和路由器正常、工作状态良好并且可用时，包才能成功转发。网络和路由器必须可管理才能确保正常，其中管理包括被动管理和主动管理。为保持网络和路由器持续地正常工作，下列内容同样很重要：实验室测试和监视转发机制的效果即路由协议、网络地址翻译、多播和服务质量。

本章节覆盖下列内容：明确的可操作规则和程序的必要性、为监视网络和路由器，路由器上 SNMP 和 RMON 的基本配置以及必要的性能、故障和安全管理。同时本章节还讨论维护路由器可用性的方法，实验室建设的一般概念和使用。

9.1 规则和程序定义

如果没有明确的规则和过程，正确管理路由器是不可能的。规则中必须有明确条文：谁来负责不同级别的管理以及他们什么时候负责。规则必须明确路由器配置什么时候能改变。程序记录配置如何改变，包括描述改变了什么、发生了什么、恢复了什么以及测试过程。程序指定当一个有经验的工程师开始解决问题或管理过程需要介入时需遵循的步骤。对分布式更新规则和程序的清晰计划也同样重要，这样的计划能使所牵涉的人员都明确所作的更新。

应当存在指定用户从网络上能期望得到的服务质量(QoS)的规则，QoS 包括例如往返时延、最小吞吐量和网络可用性，同样应当存在规定服务无法得到时所采取行动的规则。上述内容称为服务等级协议(Service Level Agreements)。本章节描述服务等级协议、改变管理规则、扩大程序以及保持规则和程序最新版本的必要性。

9.1.1 服务等级协议

服务等级协议协定(SLA)明确定义所提供服务的数量和质量以及这些服务在什么时候由谁来提供。SLA 指定服务质量，例如确保的响应时间和吞吐量以及最大的时延抖动。服务数量表达成网络可用性的百分比。SLA 确定网络什么时候可用、服务中断的最长时间、计划的服务中断以及服务由谁提供。为防止对责任范围的误解，确定由谁提供服务非常重要。通过规定哪一个组织负责网络的哪一部分以及每一个组织所提供的服务等级来防止随便指定。SLA 必须明确规定，这样提供 SLA 的组织以及接受 SLA 的组织才能理解和同意 SLA 中的条款。SLA 可以由为您的商务提供服务的外部组织提供，例如 ISP 或你外包网络的公司。或者 SLA 可以由公司内为商务提供服务的内部 IT 部门提供。

针对商务目标写成的 SLA 是最有效的。例如假设由下列 SLA 描述：

- 从 A 点到 B 点一小时内往返时延小于 50ms。
- 链路可用性不小于 95%。

如果商务目标要求可用性需要 99.9999%，往返时延只不需要在 400ms 以内，上述 SLA 就没有意义。任何对特定应用、终端或站点提供特定 QoS 的 SLA 提供者必须在 SLA 中包含所保证的 QoS 等级。

如果不监视服务，则 SLA 不能提供任何好处。提供者在 SLA 中保证的 QoS 需要确认用户或应用真正得到所保证的 QoS。提供端到端保证的 SLA 必须端到端监视。采用下边两种描述的 SLA 必须使用不同的方法测试：

从 A 点用户到 B 点服务器 1 小时内平均往返时延小于 200ms。

从 A 点边缘路由器到 B 点边缘路由器 1 小时内平均往返时延小于 100ms。

上述陈述可能同时包含在服务等级合同中，两者应当同时被监视。所收集到的数据应当同时报告到服务提供者和服务使用者。服务提供者和使用者都应当阅读报告来确保 SLA 被满足。

9.1.2 改变管理

没有改变管理的网络像一个处于混沌状态的网络。改变管理规则陈述什么时候作改变、

谁来做、如何记录和公布即将到来的改变、如何记录所完成的改变以及将改变记录到哪里。

改变管理规则规定当网络或系统变化将要发生时需要使用的程序。这些变化包括路由器配置的改变、新设计的实现、IOS 的更新、甚至新的网络应用的实现。应当存在一个电子表格来填写下面信息中的部分或全部：

- 谁要求改变
- 为什么作改变
- 改变的影响(无危害，可能危害，危害)
- 什么时候改变
- 改变需要多长时间
- 改变的结果持续多长时间
- 已完成什么样的测试过程来测试将发生的改变
- 谁做的测试
- 谁来做改变
- 实施改变的程序
- 改变后使用什么样的测试程序来验证改变的成功
- 恢复程序

应当有一个改变控制板(CCB)可以看到任何给定星期将要到来的改变以及批准或者没有批准的改变。CCB 应当拥有代表设计、运营、维护和管理网络的每一个组织的知识。CCB 应当拥有网络每一个使用组的代表，例如组织内不同的商务单位。如果网络是一个 ISP，网络用户代表应当是客户服务代表，负责客户组的福利利益。CCB 回顾过程确保所有的网络结构、运行操作者、监视管理员者、经理管理者、客户支持人员和网络使用者获悉计划的行为执行，明确可能的危害，有机会在改变实施前考虑其他的缓解因素因子。

请求的改变必须由 CCB 所有成员通过，由所有相关组织签字认可，表示他们明确要做的改变以及潜在的危害。

有时可能需要一个紧急改变——例如解决问题。改变规则还指出在紧急情况下怎么做。规则指定谁能作紧急改变，在什么情况下做，如何记录改变。完全记录所有的改变非常重要。应当由一张表格记录什么时间做的改变、做了什么样的改变、谁做的改变以及对改变描述文档的引用(例如表 9-1 中的例子)，这样其他人才能在未来路由器或网络出现问题时知道当时作了什么改变，如何恢复。

表 9-1　　改变规则管理文档

改变日期时间	改 变 描 述	改变实现人	紧　　急	改变文档位置，server.file
6/3/00，1:30 a.m	在路由器 Taos 上改变 BGP 对等实体	Joe Smith	No	Pluto:changes\RtrTaos060300
5/27/00，1:00 a.m	在路由器 Aspen 到站点 Denver 的链路上使能客户队列	Jane Anders	No	Pluto:changes\RtrAspen052700

如果 Denver 站点上的人向网络运行中心打电话报告该站点与远端站点的连接问题，说问

题出现在 5/29/00，星期一早晨，改变日志清楚地显示是星期六晚上的改变影响了 Denver 站。改变日志对问题的故障定位提供明确的时间起始点。

只有在企业网上才可以能不强制实施严格的改变控制规则。ISP 可以在实施这些规则上取得好处。在企业网上，不利的改变可能影响很多人，业务可能中断，可能会有很多恼火的终端用户。在 ISP 网络上中断业务的改变有可能影响许多公司，引起更多的业务中断。不但会出现更多恼火的终端用户，而且如果他们不满意网络运行方式则有可能改用竞争的对手 ISP。严格执行规则能使非计划的影响服务的改变最少化，而且在出现问题时得到更快的恢复。

9.1.3　扩大提交过程程序

明确定义的扩大程序提交过程指出在多长时间内一个具有某种技能的工程师无法解决一个问题则将问题转交给更有经验的工程师。它规定了程序指定将问题转交给谁以及如何转交。此外程序还规定在什么时候，用什么方式以什么样的频度将事件通知管理人员，规定问题在多长时间内无法解决则提交给管理员。

9.1.4　更新规则

无论规则指定得多完善，它总就会因新技术、新组织的出现而过时。规则需要被更新来反映变化。负责实现规则和程序的人员需要获悉这些更新，并且培训在更新的内容上被培训。规则所影响的人必须获悉所有的改变。

9.2　简单网络管理协议

网络管理软件，例如 Cisco Works，使用简单网络管理协议(SNMP)来管理网络设备。SNMP 是关于网络漂亮的图、表和曲线图的幕后实现者。SNMP 查询设备，收集建立图、表和曲线图所需要的数据。所有的 Cisco 路由器都支持 SNMPv1[1]。所有 IOS 版本 11.3 以上的 Cisco 路由器都支持 SNMPv2C。与 SNMPv1 相比，SNMPv2C 支持成块数据传输以及更详细的出错报告。

> 注：SNMPv2C 由 SNMPv2 组成，SNMPv2 在 RFC 1902-1907 中定义，SNMPv2C 在 RFC 1901[2] 中定义。

9.2.1　SNMP 概述

SNMP 包含管理器和代理。管理器收集数据；代理提供数据。

管理器可以是网络管理系统(NMS)例如 Cisco Works 的一部分。代理包含在被管理设备中，例如路由器中。

必须在管理器与代理之间建立一种联系，这样管理器可以在代理上获取或设置信息。管理器可以向代理发送 SNMP 消息来请求数据或者要求代理使用管理器指定的数据设置参数。这些消息被相应地称为 *gets* 和 *sets*。那些可以从代理请求数据、设置参数的管理器的社团团体被定义成使用访问控制列表和口令。口令被称为社团团体字串。管理器在所有的 get 和 set 请求中包含团体社团字串。被预先配置社团字串团体字串的代理通过判断社团团体字串的正

确性来验证请求的管理器是否有权执行 get 和 set 操作。管理器可以请求运行代理的平台所支持的管理信息库(MIB)中定义的任何参数。图 9-1 表示管理器从路由器请求管理状态信息。

管理工作站想要得到路由器串行接口 0 的运行状态。管理工作站发出一条 SNMP get 请求，请求 MIB 变量 ifEntry.ifOperStatus.1。ifEntry 是一个可以从代理任意接口轮询到的变量列表。IfOperStatus 是其中一个变量。0.1 是索引值，在这个例子中表示路由器串行接口 0。社团团体字串“restricted”包含在 get 请求中。路由器对该请求作应答。应答指示所请求的变量值为 1。在 MIB 定义中 ifOperStatus 值为 1 表示状态为 up；值为 2 表示状态为 down；值为 3 表示状态为 testing。图中的串口 0 链路为 up。图 9—1 中管理工作站发布 get 请求，查询路由器串行接口 0 的运行状态；路由器用 Get Reply 响应。

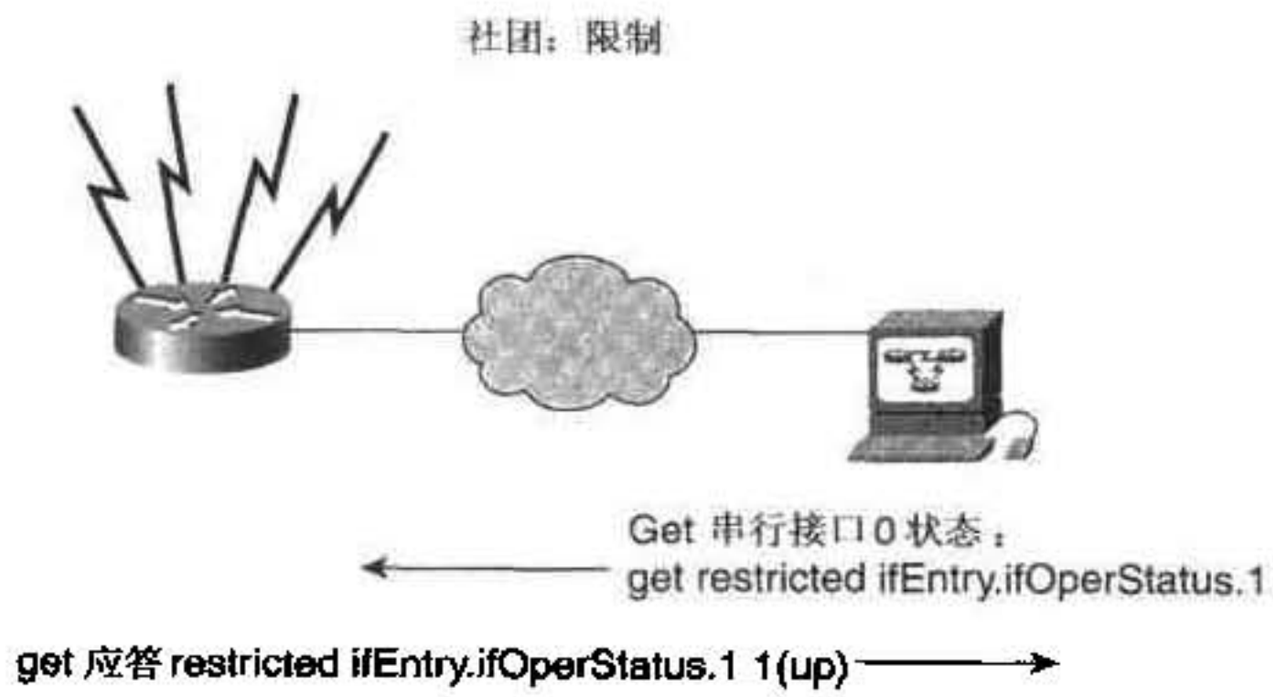


图 9-1　管理器从路由器请求管理状态信息管理工作站发布 get 请求，查询路由器串行接口 0 的运行状态；路由器用 Get Reply 响应

注：　所有有 SNMP 能力的路由器都支持 MIBII。Interface 变量在 MIBII 中定义。MIBII[3] 由 RFC 1213 定义。

SNMP 运行在 UDP 上。SNMP 运行的优先级低于路由器上其他进程。如果路由器忙于高优先级工作，SNMP 消息可能被丢弃。大多数管理器配置中指定在状态改变显示以前出现的多于一个丢失的轮询。

SNMP 允许您收集许多信息。通过 SNMP 收集的信息可以是每一个网络设备(以及设备中每一个部件)和协议栈中每一个协议的每一个统计信息。有时所收集的统计信息正在变化，管理器收集变化着的数据信息，例如在不正常的接口上的每分钟错误出现数量。这样的统计要求大量频繁地使用 SNMP get 消息来获得最准确的统计数值。大量的 SNMP 流量可能对网络性能有不利影响，然而(所以)SNMP 流量的总量应当被小心管理。在你使能管理应用之前，应当彻底了解应用所收集的 SNMP 流量总量(以及其他流量)。

Trap 消息由代理向管理工作站发送。Trap 是代理主动提供的，是某些事件发生的结果。这些事件可能是 link down 链路失效、BGP 连接失败、认证失败或者其他事件。当事件发生时，路由器向管理工作站发送 SNMP trap，通知事件的发生。管理工作站的配置规定了收到 trap 之后做什么。可能会是一个网络图像改变颜色，在屏幕上显示一条消息或者向网络管理员发送一个电子邮件或者寻呼消息。

SNMP 由为网络管理平台，例如 Cisco Works 提供基础。

9.2.2　CiscoWorks

Cisco 产品组成的网络可以在 CiscoWorks 的帮助下管理。CiscoWorks 在网管平台的最上层运行，这样的网关平台可以是 HP 的 OpenView、IBM 的 NewView 或者 Sun 的 NetManager 等。管理平台提供一般的图、表和曲线图功能，Cisco Works 在其上增加 Cisco 特定的实体，例如机架视图和设备配置管理。

CiscoView 是 CiscoWorks 中的一个应用。CiscoView 提供 Cisco 设备组成的网络的实时视图。上述视图提供设备配置和性能条件物理图像的实时更新。机架视图显示了 Cisco 设备的前后面板视图，包括 LED 状态指示灯。如果您点击机架视图中显示的端口，您可以得到关于该端口的一个统计数据表，表中包括利用率、输入和输出错误、队列丢包、碰撞和忽略的包等。CiscoView 同时还提供仪表盘风格的视图来显示 Cisco 设备的系统性能，包括内存利用率、缓存利用率和 CPU 利用率等。CiscoView 在一个集中的网络管理站点上运行，在 CiscoView 上你可以在一个统一的 GUI 里检查、重配置或监视特定的设备数据(GUI 中显示例如动态状态报告、性能统计和网络查询等信息)而不必要到每一个不同站点和远端站点去实际检查设备间物理连线、模块或端口。

网络管理平台的一个选择和 CiscoWorks 共同工作，实现故障管理、性能管理和配置管理。图、表和曲线图依靠 SNMP 来保持最新。

由 CiscoWorks 管理的代理必须配制成接受轮询并且能向运行 CiscoWorks 的工作站发送 trap。

9.2.3　路由器的 SNMP 配置

各种全局的 SNMP 命令允许路由器能被 CiscoWorks 管理。所有的全局 SNMP 配置由 **snmp-server** 命令配置开始。没有一个特别的命令来使能 SNMP。前面键入的 **snmp-server** 命令将使能路由器上所有版本的 SNMP。

路由器必须配制成使用与管理工作站版本相同的 SNMP。

创建管理社团团体的命令如下所示：

[no] snmp-server community *string* [**view** *view-name*] [**ro** | **rw**][*access-list number*]

社团字串团体字串被作为管理者与代理之间的口令。管理工作站对路由器可以有只读(RO)或者读写(RW)权限。CiscoWorks 管理工作站需要读写访问权限才能做完全管理，特别是设置参数，重启路由器软件或者更新配置。SNMP 是非常强大的工具。路由器上几乎所有的配置和状态信息都可以通过 SNMP MIB 读取。通过 SNMP 读访问得到的信息可以是路由表和 ARP 表，可以使管理人员获悉特定设备在特定区域的攻击。SNMP 写操作可以改变配置或者使路由器或链路复位。使用 **access-list** 选项来限制那些设备可以向路由器作读和/或写操作。该列表是一个简单的访问列表，指定管理工作站的一个地址或者地址范围。

使能 SNMP 唯一需要的命令是 **snmp-server community**。其他 SNMP 命令是可选的，用来调节可收集或可设置信息。

snmp-server community 命令中的 **view** 选项在下面命令中使用：

snmp-server view *view-name oid-tree* {**include** | **exclude**}

该命令限制 SNMP 管理者只能访问某一个 MIB 对象。其中的 *oid-tree* 即对象标识符树，标识该 MIB 子树被包含或者排除在外。为标识一棵子树，可以使用包含数字的字符串例如 1.3.6.2.4 或者一个单词例如“system”来指定需要子树的根。指定“system”表示系统子树中所有的 MIB 值被包含或者不包含。1.3.6.2.1.2 是 iso.org.dod.mgmt.mib2.interface 的数字表示形式。

注：可参照 RFC 定义的独立的 MIB 以及 RFC 1902，“简单网络管理协议版本 2(SNMPv2)的管理信息结构”来对所有对象标识符作数字化表示。

SNMP 管理值可以在虚拟终端或者控制台上向用户发送信息。发送消息的 SNMP 请求同时指定消息发送以后应采取的行动，例如关闭系统。这是一个非常强大的工具。为使能该功能，你必须配置 **snmp-server system-shutdown** 命令。如果你没有配置该命令，该功能禁止。

另一个强大的工具是 TFTP 服务器通过 SNMP 来存取配置文件的能力。你可以通过使用 **snmp-server tftp-server-list** *number* 命令指定访问列表对服务器作限制。其中 number 是访问列表数字。

规定向哪一个主机发送 trap 以及发送哪一种 trap 的命令如下所示：

snmp-server host *host* [**version** {**1** |**2c** }] *community-string* [**udp-port** *port*] [*trap-type*]

如果不指定 trap 类型，路由器上使能的所有 trap 类型都向主机发送。**Vversion** 定义管理工作站使用的 SNMP 版本号。**udp-port** 选项能改变缺省的端口号。

要向指定的主机发送 trap，必须先在全局条件下使能 trap。一些 trap 缺省使能。其余 trap 必须使用如下命令来使能：

snmp-server enable traps *trap-type trap-option*

snmp-server enable traps 命令对路由器上指定的 trap 使能自陷机制。该命令没有指定向哪一个主机发送 trap。使用该命令只能使能或禁止某些类型的 trap。当禁止 traps 时，只能对每一个主机指定 trap。

另一个命令可以用作基于接口控制 trap。在被配置的接口上用来禁止或使能链路状态 trap 的接口子命令是**[no] snmp trap link-status**。缺省情况下该 trap 在所有接口使能。

如果主机需要接收 trap，必须在路由器中使用 **snmp-server host** 命令指定主机的地址全局配置中使能 trap。使能 trap 可以通过 **snmp-server enable traps** 命令或其他命令例如 **snmp trap link-status** 或者仅使用缺省配置。一些配置实例如下所示。

例 9-1 中的配置允许任何 SNMP 管理只通过社团字串团体字串“access”作只读访问。使能 BGP traps，并且发送到主机 172.16.1.200 和 172.16.1.201。

例 9-1　允许任意 SNMP 管理者只读访问并且使能 BGP traps

```
snmp-server community access RO
snmp-server enable traps bgp
snmp-server host 172.16.1.200 access
snmp-server host 172.16.1.201 access
```

在路由器 10.1.2.1 和路由器 10.1.2.25 间建立 BGP 外部连接。当连接清除时产生 trap，如例 9-2 所配置。

例 9-2　清除 BGP 外部连接情况下产生 trap

```
Bowler#clear ip bgp *

SNMP: Queuing packet to 172.16.1.200
SNMP: V1 Trap, ent bgp, addr 10.1.2.25
 bgpPeerEntry.14.10.1.2.1 = 00 00
 bgpPeerEntry.2.10.1.2.1 = 1

SNMP: Queuing packet to 172.16.1.201
SNMP: V1 Trap, ent bgp, addr 10.1.2.25
 bgpPeerEntry.14.10.1.2.1 = 00 00
 bgpPeerEntry.2.10.1.2.1 = 1

SNMP: Packet sent via UDP to 172.16.1.200
SNMP: Packet sent via UDP to 172.16.1.201
```

版本 1 的 trap 被发送到两个 trap 主机。发送 trap 的路由器的地址是 10.1.2.25，该地址是向 trap 主机发送包所用的接口的地址。MIB OID bgpPeerEntry.14.10.1.2.1，表示该对等体看到的上一个 BGP 错误号，在与地址为 10.1.2.1 的对等体连接时的值为 00 00。00 00 表示没有看到错误。OID bgpPeerEntry.2.10.1.2.1 表示 BGP 对等体状态，由本对等体在连接到对等体 10.1.2.1 所看到的值。该值为 1 表示状态是 idle。

> 注：　站点 www.cisco.com/public/mibs/v1. 可以得到所有被支持的 Cisco MIB 的定义列表。

在例 9-3 的配置中，主机 172.16.1.201 已升级到 SNMP 版本 2c，该主机只接收 BGP trap。172.16.1.200 只接收 TTY trap。另外所有 SNMP trap 的源 IP 地址配制成使用路由器环回接口的 IP 地址：172.16.2.25。

例 9-3　使能 SNMPv2c Trap 并指定 SNMP 源 IP 地址

```
snmp-server community access RO
snmp-server enable traps bgp tty
snmp-server host 172.16.1.200 access tty
snmp-server host 172.16.1.201 version 2c access bgp
snmp-server trap-source loopback 1
```

当用户从路由器外登录时，路由器发送 TTY 连接 Trap，如例 9-4 所示：

例 9-4　当用户从路由器外登录时，路由器发送 TTY 连接 Trap

```
#Telnet Boxer
Trying 10.1.1.1 ... Open


User Access Verification

Password:
Boxer>logout

[Connection to Boxer closed by foreign host]
```

```
Boxer#
SNMP: Queuing packet to 172.16.1.200
SNMP: V1 Trap, ent enterprises.9, addr 172.16.2.25
 ltsLineSessionEntry.1.66.1 = 5
 tcpConnEntry.1.10.1.1.1.23.10.1.10.1.11000 = 5
 ltcpConnEntry.5.10.1.1.1.23.10.1.10.1.11000 = 958
 ltcpConnEntry.1.10.1.1.1.23.10.1.10.1.11000 = 45
 ltcpConnEntry.2.10.1.1.1.23.10.1.10.1.11000 = 87
 ltsLineEntry.18.66 =
```

trap 类型是 **enterprises.9**，该 trap 在路由器软件重启或 TCP 连接关闭时发生。

ltsLineSessionEntry.1 表示线路会话类型。该值为 5 表示远程登录会话产生的该 trap。

tcpConnEntry.1 表示 TCP 连接的状态。该值为 5 表示连接被关闭。下面的数字是本机的 IP 地址以及实施 telnet 操作的设备的 IP 地址。

ltcpConnEntry.5 表示 TCP 连接持续的时间，以百分之一秒为单位表示。所以连接持续了 9.58 秒。下面两个 OID 表示该 TCP 连接收发的字节数：收到 45 字节，发送 87 字节。如果 TACACS 使能，**ltsLineEntry.18** 显示 TACACS 用户名。例 9-5 所示发送到 172.16.1.201 的 SNMPv2C BGP trap。

例 9-5　与 SNMPv1 BGP trap 相比，SNMPv2C BGP trap 包含更多信息

```
SNMP: Queuing packet to 172.16.1.201
SNMP: V2 Trap
 sysUpTime.0 = 14423502
 internet.6.3.1.1.4.1.0 = bgp.7.2
 bgpPeerEntry.14.10.1.2.1 = 00 00
 bgpPeerEntry.2.10.1.2.1 = 1
```

版本 2c trap 比版本 1 trap 包含更多信息。system uptime 是从网络管理部分系统上一次初始化到现在所经过的时间，以百分之一秒为单位计数。OID **internet.6.3.1.1.4.1.0** 的值是 **bgp.7.2**，表示产生了指定的 trap，bgpBackwardTransition。上述 trap 在 BGP 连接从较高的状态值转移到较低的状态值时产生，例如从 establish 状态转变成 idle 状态。例 9-6 中 trap 显示 BGP 状态进入 ESTABLISHED 状态。

例 9-6　BGP 状态进入 ESTABLISHED

```
SNMP: V1 Trap, ent bgp, addr 172.16.2.25
 bgpPeerEntry.14.10.1.2.1 = 00 00
 bgpPeerEntry.2.10.1.2.1 = 6

SNMP: V2 Trap
 sysUpTime.0 = 14425396
 internet.6.3.1.1.4.1.0 = bgp.7.1
 bgpPeerEntry.14.10.1.2.1 = 00 00
 bgpPeerEntry.2.10.1.2.1 = 6
```

OID **internet.6.3.1.1.4.1.0** 值为 **bgp.7.1**，表示 bgpEstablished trap，相邻条目值为 6 表示与对等体 10.1.2.1 的连接是 ESTABLISHED 状态。

例 9-7 中的配置值允许访问列表 1 中规定 IP 地址的管理者作只读访问，使用社团字串团体字串为“restricted”。并且只允许主机访问部分 MIB，主要是接口实体。

例 9-7　允许访问列表上指定 IP 地址作只读访问

```
access-list 1 permit 172.16.1.200
snmp-server view interface_entries ifEntry included
snmp-server community restricted view interface_entries RO 1
```

其他 SNMP 管理者不能使用社团字串团体字串“restricted”来访问该设备上该 SNMP 代理。如果上述社团字串团体字串是路由器上配置的唯一社团字串团体字串，则 172.16.1.200 是唯一能读取 SNMP MIB 变量的设备；然而该管理者不能设置 MIB 变量。**view** 命令配置了一个名为 **interface_entrys** 的视图，并将该视图限制在 ifEntry。**community** 命令将所定义的视图关联到字串为“restricted”的社团团体，满足访问列表 1 的主机。

例 9-8 显示了读取 MIB 中 ifEntry 和 IP 分支的 SNMP 操作的部分结果。

例 9-8　在路由器上设置视图限制以前在 MIB 读取 ifEntry 和 IP 分支的 MIB 操作

```
ObiWan:~# snmpwalk 172.16.1.7 restricted ifEntry
interfaces.ifTable.ifEntry.ifIndex.1 = 1
interfaces.ifTable.ifEntry.ifIndex.2 = 2
interfaces.ifTable.ifEntry.ifIndex.3 = 3
interfaces.ifTable.ifEntry.ifIndex.4 = 4
interfaces.ifTable.ifEntry.ifDescr.1 = Ethernet0
interfaces.ifTable.ifEntry.ifDescr.2 = Ethernet1
interfaces.ifTable.ifEntry.ifDescr.3 = Serial0
interfaces.ifTable.ifEntry.ifDescr.4 = Serial1
interfaces.ifTable.ifEntry.ifOperStatus.1 = down(2)
interfaces.ifTable.ifEntry.ifOperStatus.2 = up(1)
interfaces.ifTable.ifEntry.ifOperStatus.3 = down(2)
interfaces.ifTable.ifEntry.ifOperStatus.4 = up(1)
interfaces.ifTable.ifEntry.ifInOctets.1 = 720250042
interfaces.ifTable.ifEntry.ifInOctets.2 = 283245
interfaces.ifTable.ifEntry.ifInOctets.3 = 0
interfaces.ifTable.ifEntry.ifInOctets.4 = 761771001
interfaces.ifTable.ifEntry.ifOutOctets.1 = 779888827
interfaces.ifTable.ifEntry.ifOutOctets.2 = 228281
interfaces.ifTable.ifEntry.ifOutOctets.3 = 0
interfaces.ifTable.ifEntry.ifOutOctets.4 = 10994586

ObiWan:~# snmpwalk 172.16.1.7 restricted ip
ip.ipRouteTable.ipRouteEntry.ipRouteDest.172.16.1.0 = IpAddress: 172.16.1.0
ip.ipRouteTable.ipRouteEntry.ipRouteIfIndex.172.16.1.0 = 2
ip.ipRouteTable.ipRouteEntry.ipRouteMetric1.172.16.1.0 = 0
ip.ipRouteTable.ipRouteEntry.ipRouteNextHop.172.16.1.0 = IpAddress: 172.16.1.7
ip.ipRouteTable.ipRouteEntry.ipRouteType.172.16.1.0 = direct(3)
ip.ipRouteTable.ipRouteEntry.ipRouteProto.172.16.1.0 = local(2)
ip.ipRouteTable.ipRouteEntry.ipRouteAge.172.16.1.0 = 0
ip.ipRouteTable.ipRouteEntry.ipRouteMask.172.16.1.0 = IpAddress: 255.255.255.0
ip.ipNetToMediaTable.ipNetToMediaEntry.ipNetToMediaPhysAddress.2.172.16.1.2 =
  0:10:5a:e5:e:e3
ip.ipNetToMediaTable.ipNetToMediaEntry.ipNetToMediaPhysAddress.2.172.16.1.7 =
  0:0:c:76:5b:7d
```

注：　**snmp get** 读取 MIB 中单一条目，**snmpwalk** 命令读取 MIB 分支所有内容。

例 9-9 显示引入视图限制后相同 snmpwalk 命令的执行。

例 9-9　路由器设置视图限制后在 MIB 读取 ifEntry 和 IP 分支

```
ObiWan:~# snmpwalk 172.16.1.7 restricted ifEntry
interfaces.ifTable.ifEntry.ifIndex.1 = 1
interfaces.ifTable.ifEntry.ifIndex.2 = 2
interfaces.ifTable.ifEntry.ifIndex.3 = 3
interfaces.ifTable.ifEntry.ifIndex.4 = 4
interfaces.ifTable.ifEntry.ifDescr.1 = Ethernet0
interfaces.ifTable.ifEntry.ifDescr.2 = Ethernet1
interfaces.ifTable.ifEntry.ifDescr.3 = Serial0
interfaces.ifTable.ifEntry.ifDescr.4 = Serial1
interfaces.ifTable.ifEntry.ifOperStatus.1 = down(2)
interfaces.ifTable.ifEntry.ifOperStatus.2 = up(1)
interfaces.ifTable.ifEntry.ifOperStatus.3 = down(2)
interfaces.ifTable.ifEntry.ifOperStatus.4 = up(1)
interfaces.ifTable.ifEntry.ifInOctets.1 = 720250042
interfaces.ifTable.ifEntry.ifInOctets.2 = 334364
interfaces.ifTable.ifEntry.ifInOctets.3 = 0
interfaces.ifTable.ifEntry.ifInOctets.4 = 761771405
interfaces.ifTable.ifEntry.ifOutOctets.1 = 779888827
interfaces.ifTable.ifEntry.ifOutOctets.2 = 268919
interfaces.ifTable.ifEntry.ifOutOctets.3 = 0
interfaces.ifTable.ifEntry.ifOutOctets.4 = 10995692
End of MIB

ObiWan:~# snmpwalk 172.16.1.7 restricted ip
End of MIB
ObiWan:~# logout
```

管理工作站无法读到没有在视图定义中明确包含的部分的任何 MIB。

9.3　RMON

远程监视(RMON)通过允许管理工作站获取关于节点以及节点与其他节点交互的更多信息来增强 SNMP 提供的能力。

9.3.1　RMON 概述

正如 SNMP，RMON 的功能的使用与管理工作站相关，可以是 RMON 控制台和被管理的代理。RMON 数据存储在路由器的表格中，当接到请求或由 send trap 触发时发送到 RMON 控制台。RMON 机制通过减少需要由一般的 SNMP 包轮询的数据总量来降低网络流量。在路由器中的 RMON 引擎本地轮询 SNMP MIB 变量。存在两种门限：上升门限与下降门限。当 MIB 变量值超过门限时，RMON 产生一个日志条目并且发送 SNMP trap。在对端门限超过以前，RMON 不会产生其他事件。当变量值增加超过上升门限时，触发一个事件。在该数值减少到下降门限以前，不会有其他事件触发。

没有定购 RMON 选件的路由器有告警和事件能力。RMON 选件只能在 2500 系列和 AS5200 系列路由器中存在，Cisco RMON 增加了其他组——ststistics、history、hosts、hostTopN、matrix、filter 和 capture 组。

包捕获只有在 2500 系列路由器和 AS5200 系列路由器以太网端口上可用，而且只能捕获

包头。包捕获可以有两种方式：

- **本地方式**——只捕获发往路由器本以太网接口的包
- **混合方式**——在该网段上的所有包

包捕获机制对数据和处理器非常敏感。使能包捕获时，必须对路由器性能和网络流量密切监视。

告警和事件加上已存在的 MIB 变量，允许你定义一些主动的监视。你可以在 MIB 变量上设置告警，这些告警处理整数类型的值、计数器、度量或时间。

注：　RFC 2819 完整地定义了所有的 RMON 组以及他们的交互。

9.3.2　路由器的 RMON 配置

定义告警表项和所监视的变量的命令如下所示：

rmon alarm *number variable interval* **{delta | absolute} rising-threshold** *value* [*event-number*] **falling-threshold** *value* [*event-number*] **[owner** *string*]

number 在告警表中唯一地标识该表项。*variable* 是一个 MIB OID。*interval* 是监视 MIB 对象的时间间隔。**delta** 和 **absolute** 两个关键字指定告警测试在指定时间间隔内 MIB 的变化还是测试实际 MIB 值。**rising-threshold** 是产生事件的门限值。如果上一个采样值小于该门限且当前采样值大于等于该门限则产生事件。如果指定事件值，则触发上升门限时产生指定事件号的事件。只有当采样值下降到小于 **falling-threshold** 时，才会产生另一事件。**falling-threshold** 同样也是另一事件产生的门限值。如果上一采样值大于该门限且当前采样值小于等于 falling-threshold 值则产生一事件。只有当采样值又大于 **rising-threshold** 时才会产生另一事件。

RMON 事件表中增删表项的命令如下所示：

rmon event *number* **[log] [trap** *community*] **[description** *string*] **[owner** *string*]

该命令所定义的事件只有当事件号所指定的门限到达时才触发。当事件触发时可能记录日志或者发送 SNMP trap 或者两者同时发生。在 SNMP trap 发送以前必须为 **rmon event** 命令中指定的社团团体配置 **snmp-server community** 和 **snmp-server host** 命令。

例 9-10 中为接口上大量输出错误以及路由器上高 CPU 利用率使能事件和告警的范例。MIB OID 可以被完整键入，例如 1.3.6.1.2.1.2.2.1.20.4，表示 ifOutError 索引号 4 的 MIBII 值；或者 1.3.6.1.4.1.9.2.1.58.0 代表 Cisoc 的 CUP 在 5 分钟内忙时百分比的 MIB 值。路由器自动将 OID 转变成例 9-10 中所示。

例 9-10　为路由器高接口输出错误和高 CPU 利用率时使能事件和告警

```
snmp-server community eventtrap RO
snmp-server enable traps
snmp-server host 172.16.1.2 eventtrap
snmp-server trap-source loopback 1
rmon event 1 log trap eventtrap description "High ifOutErrors"
rmon event 2 log trap eventtrap description "High 5-minute CPU" owner jsmith
rmon alarm 10 ifEntry.20.4 20 delta rising-threshold 15 1 falling-threshold 0
  owner jsmith
rmon alarm 11 lsystem.58.0 20 absolute rising-threshold 50 2 falling-threshold 25
  owner jsmith
```

RMON 事件 1 将事件描述成“**High ifOutError**”记录到日志，关联到所有人 jsmith。该事件还触发一个 SNMP trap 发往 **eventtrap** 社团团体。当相关告警出现时产生该事件。RMON 告警事件 10 配置在 MIB 变量 **ifEntry.20.4** 上，表示接口 4 上的输出错误。接口 4 在本例中是串行接口 1。该告警每 20 秒读取 MIB 变量。如果两次的采样值增加了 15 或者更多，则该告警触发，并触发事件 1。如果下一次 MIB OID 采样显示接口上没有输出错误，则该告警复位，准备下一次触发。

事件 2 描述为“**High 5-minite CPU**”记录到日志。相关的告警，告警 11 在 MIB OID **lsystem.58.0**(5 分钟平均忙)采样值大于等于 50 时产生事件。当 5 分钟平均 CPU 忙时百分比小于 25 时，告警复位，准备下一次触发。

例 9-11 显示事件 1 产生的 SNMP trap。

例 9-11　例 9-10 中定义的事件 1 产生的 SNMP trap

```
SNMP: Queuing packet to 172.16.1.2
SNMP: V1 Trap, ent rmon, addr 172.16.2.25
 alarmEntry.1.10 = 10
 alarmEntry.3.10 = ifEntry.20.4
 alarmEntry.4.10 = 2
 alarmEntry.5.10 = 20
 alarmEntry.7.10 = 15
```

RFC 2819[4] 中定义了告警条目。告警条目 1 表示告警索引。正如例 9-11 中指出，告警索引 10 产生了该 SNMP trap。告警条目 3 定义了被采样的对象标识。在这个 SNMP trap 中，OID 是 ifEntry.20.4，在接口 1 上输出的错误数量。告警条目 4 识采样类型。值 1 表示类型是绝对值，值 2 表示采样类型是增量。告警条目 5 表示上一个采样周期的采样值。告警条目 7 表示上升门限。在例 9-11 的 SNMP trap 中，告警值 20 超过 15 的门限值，所以才触发事件。

RMON 告警和事件可以通过使用 **show rmon alarms** 和 **show rmon events** 命令来浏览。例 9-12 显示了上述两个命令的输出。

例 9-12　使用 **show rmon alarms** 和 **show rmon events** 命令显示 RMON 告警和事件表

```
Bowler#show rmon event alarms
Event 1 is active, owned by jsmith
 Description is High ifOutErrors
 Event firing causes log and trap to community eventtrap, last fired 1d00h
 Current log entries:
      index       time   description
          1      1d00h   High ifOutErrors
          2      1d00h   High ifOutErrors
Event 2 is active, owned by jsmith
 Description is High 5-minute CPU
 Event firing causes log and trap to community eventtrap, last fired 1d00h
 Current log entries:
      index       time   description
          1      1d00h   High 5-minute CPU
Alarm 10 is active, owned by jsmith
 Monitors ifEntry.20.4 every 20 second(s)
 Taking delta samples, last value was 20
```

```
  Rising threshold is 15, assigned to event 1
  Falling threshold is 0, assigned to event 0
  On startup enable rising or falling alarm
Alarm 11 is active, owned by jsmith
  Monitors lsystem.58.0 every 20 second(s)
  Taking absolute samples, last value was 60
  Rising threshold is 50, assigned to event 2
  Falling threshold is 25, assigned to event 0
  On startup enable rising or falling alarm
```

事件和告警表中的时间表示事件发生时的 sysUpTime。SysUpTime 是从上一次路由器复位后所过的时间。例 9-12 输出的结果值是 1 天 0 小时。

9.4 记录日志

当路由器使能日志功能时，某些事件发生时路由器会产生消息并存储。日志可以记录在路由器上也可以记录在路由器外部，存储在网络上某个服务器内。

路由器将 **debug** 命令输出结果和系统出错信息发送到日志进程。日志进程按照路由器配置将消息分发到不同的日志记录设备和文件。消息被发送到日志记录缓存、终端行和/或 UNIX 系统日志服务器。日志记录缓存在路由器上维护。该缓存是环形缓存，最老的消息由最新的消息替代。当浏览日志时，最老的日志条目最先出现。

注：系统日志格式与 BSD UNIX4.3 兼容。

下列命令使能日志缓存、显示日志内容以及清除日志。

- **logging buffered** [*size*]
- **show logging**
- **clear logging**

发送到系统日志服务的消息存储在服务器的文件中。消息被直接发送到运行在服务器上的 syslog 进程，该进程将消息存储到相应的文件。

logging *host* 命令使日志能发送到服务器 IP 地址所指定的 syslog 服务器。

当远程登录到路由器时，通常你无法看到日志消息。如果要路由器将日志消息发送到 Telnet 会话，则需要键入 EXEC 命令 **terminal monitor**。如果你通过控制台端口直接连接到路由器，则无需键入上述命令。路由器缺省配置是将日志消息发送到控制台端口。

当遇到问题需要排除故障时，路由器将从调试等级到紧急消息所有的内容发送到 Telnet 会话的能力非常重要。如果记录所有通过 Telnet 会话收发的消息，你将得到一个良好的排除故障的记录以及在你连接期间路由器上发生的事件。

注：可以通过 Telnet 工具的日志记录或者捕获功能记录日志消息。

必须使能时戳并设置时钟，才能使日志中的消息有意义。

例 9-13 显示了 **show logging** 命令的输出结果(没有使能时戳)。

例 9-13　show logging 命令的输出

```
Seattle#show logging
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level debugging, 9 messages logged
    Monitor logging: level debugging, 0 messages logged
    Buffer logging: level debugging, 9 messages logged
    Trap logging: level informational, 13 message lines logged

Log Buffer (4096 bytes):

%LINK-5-CHANGED: Interface TokenRing0, changed state to administrative
ly down
%LINK-5-CHANGED: Interface Serial0, changed state to administratively
down
%LINK-3-UPDOWN: Interface Serial1, changed state to down
%SYS-5-CONFIG_I: Configured from console by console
```

上述输出中没有关于事件发生的时间。

使用 **service timestamps log uptime** 能显示从系统上一次重起后经过的时间。例 9-14 显示了使能 **service timestamps log uptime** 后相同的日志输出。

例 9-14　使能 service timestamps log uptime 后 show logging 命令的输出

```
Seattle#show logging
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level debugging, 9 messages logged
    Monitor logging: level debugging, 0 messages logged
    Buffer logging: level debugging, 9 messages logged
    Trap logging: level informational, 13 message lines logged

Log Buffer (4096 bytes):

00:00:39: %LINK-5-CHANGED: Interface TokenRing0, changed state to administratively
down
00:00:39: %LINK-5-CHANGED: Interface Serial0, changed state to administratively
down
00:00:39: %LINK-3-UPDOWN: Interface Serial1, changed state to down
1d07h: %SYS-5-CONFIG_I: Configured from console by console
```

所有的事件都带有时戳。你可以看到最前面 3 个事件同时发生。事实上他们都发生在系统启动后 39 秒。第 4 个事件发生在系统启动后 1 天零 7 小时。

要显示路由器上的实际日期和时间，需要键入下面命令：

service timestamps log datetime [msec] [loaltime] [show-timezone]

上述时间可以包含微秒，并可以显示路由器本地时间。时区同样可以显示。一些路由器不维护日历，所以这些路由起重启以后时钟复位。如果不使用网络时间协议，你必须使用下面 EXEC 命令人工设置时钟：

clock set *hh:mm:ss day month year*

当你从多个路由器记录日志信息时，一致的时间信息能使事件间的关系容易理解。当故

障排除时，要求从多个路由器报告的消息中查找特定的事件，这时所有的路由器都使用单一时区(不指定本地时间)非常重要。如果所由路由器消息中的时戳使用本地时间，你必须使用命令**show-timezone** 来区分。Syslog 守护进程将消息到达服务器本地时钟日期和时间写入日志文件。你可以指定几个重要级别的消息来限制日志消息的记录。表 9-2 列出日志消息的级别。

表 9-2　　日志消息级别

消息重要等级	值	翻　译
Emergencies	0	系统不稳定
Alert	1	需要立即采取行动
Critical	2	临界情况
Errors	3	错误情况
Warnings	4	警告情况
Notifications	5	正常但重要的情况
Informational	6	仅信息性消息
Debugging	7	调试消息

如果你在特定的日志中指定一个级别的消息，你可以得到该级别以及更高级别的所有消息。如果你指定调试级别，则立即得到所有的消息。如果你指定 warning 级别，你同时得到 errors、critical、alert 和 emergencies 级别的消息。

使用下列配置命令指定消息级别：

- **logging console** *level*——限制记录到控制台上的消息
- **logging monitor** *level*——限制记录到终端上的消息
- **logging trap** *level*——限制记录到 syslog 服务器上的消息

软件和硬件的故障显示在 warning 级别到 emergencies 级别。接口的 up/down 变化和系统的重启显示在 notification 级别。重启请求和底层堆栈消息显示在 information 级别，**debug** 输出显示在 debugging 级别。

例 9-15 显示位于 Seattle 的路由器的配置，位于太平洋时区。

例 9-15　在 Seattle 的路由器(位于太平洋时区)的日志配置

```
service timestamp debug datetime localtime show-timezone
service timestamp log datetime localtime show-timezone
clock timezone PST -8
clock summer-time PDT recurring
logging buffered
```

路由器的系统时钟是基于同等全球时间(UTC)，与格林尼治时间相同。太平洋标准时间比 UTC 早 8 小时。

例 9-16 显示了完成例 9-15 配置后路由器日志显示。

例 9-16　实现例 9-15 配置后日志记录缓存显示

```
Seattle>show logging
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level debugging, 8 messages logged
    Monitor logging: level debugging, 0 messages logged
    Buffer logging: level debugging, 8 messages logged
    Trap logging: level informational, 12 message lines logged

Log Buffer (4096 bytes):

*Nov 28 16:00:39 PST: %LINK-5-CHANGED: Interface TokenRing0, changed state to
administratively down
*Nov 28 16:00:39 PST: %LINK-3-UPDOWN: Interface Serial1, changed state to down
*Nov 28 16:00:39 PST: %LINK-5-CHANGED: Interface Serial0, changed state to
administratively down
*Nov 30 12:00:39 PST: %SYS-5-CONFIG_I: Configured from console by console
```

尽管当没有 syslog 服务器时日志缓存非常有用，它也有缺陷。查找一个事件需要在整个文件内翻页查找或者存储到文件中使用系统搜索机制。如果你查找一个最新的事件，因为最老的事件在最前面，你必须翻遍整个文件(或者将文件存储到另一个操作系统中)。将日志记录到 syslog 能自动在 UNIX 服务器上产生文件，该方法能提供较友好的查找和文件维护机制。

9.5　系统日志(Syslog)

Syslog 是一个运行在 UNIX 服务器上的进程或者守护进程。该进程收集信息并存储到日志文件。可以使用服务器上的文件管理系统维护日志文件。

9.5.1　Syslog 概述

日志消息从运行在 UNIX 服务器上的不同服务以及其他网络节点发来。发送消息的服务指示其设备类型。Syslog 守护进程决定如何记录所收到消息时使用所指示的设备类型。表 9-3 列出了不同的设备类型。

表 9-3　　**Syslog 设备类型**

设 备 类 型	服　　务
Auth	认证系统
Cron	时钟守护进程设备
Daemon	系统守护进程
Kern	内核
Local0-7	本地定义的消息
Lpr	打印机系统
Mail	邮件系统

续表

设 备 类 型	服　　务
News	USENET 新闻
Sys9-14	系统使用
Syslog	系统日志
User	用户进程
Uucp	Unix 到 Unix 复制系统

Syslog 守护进程的配置可以通过修改服务器/etc 目录下 syslog.conf 文件来实现。在启动时 Syslog 守护进程读取上述文件决定如何处理得到的消息。文件包含例如下列内容：

Local7.debugging　/usr/adm/logs/cisco.log

上面配置内容指定 debugging 或者更高级别的 local7 设备消息应当记录到文件 cisco.log 中，该文件位于目录/usr/adm/logs。文件中上述条目是对大小写敏感的。

任何配置的改变都要求 UNIX 系统管理员强制 Syslog 守护进程重新读取该配置文件。

9.5.2　路由器上 Syslog 的配置

Cisco 路由器向 Syslog 发送消息时缺省使用 **local7** 设备。如果上述设备已经由其他进程占用，你可以在 Cisco 路由器上使用下面命令改变设备类型：

logging facility *facility-type*

例 9-17 中的路由器配置使 Syslog 日志记录到指定主机。日志的内容是 notifications 以及以上级别的消息。

例 9-17　使用 Syslog 将日志记录到指定主机

```
logging 172.16.1.2
logging trap notifications
```

在 UNIX 服务器/etc/syslog.conf 文件中加入下面一行内容：

local7.notice　/usr/adm/logs/cisco.log

注：　参看你使用的 UNIX 服务器的用户手册，关于 syslog 配置文件和 syslog 守护进程的相关信息。

Local7 指定了日志设备，**notifications** 指定了日志记录级别。所有的信息存储在/usr/adm/logs 目录中 cisco.log 文件中。该文件必须已经存在，并且 syslog 守护进程必须有写该文件的权限。确信 syslog 守护进程正在运行，并确认改变配置以后该守护进程重新读取了配置文件。

注：　使用 UNIX 中的 **kill** 以及恰当的信号命令可以使进程重新读取配置文件。请参考 UNIX 系统用户手册中关于 **kill** 命令的详细内容。

一些路由器支持通过 SNMP 向 SNMP 网管发送 Syslog 消息的能力。可以通过使用路由器命令 **snmp-server enable traps syslog** 使能上述能力，使用命令 **logging history** level 来指定日志记录级别。

例 9-18 显示了路由器发出的 SNMP 包中的 Syslog 消息。

例 9-18　作为改变配置的结果所产生的 Syslog 消息，并通过 SNMP 发送

```
Cascade#conf t
Cascade(config)#snmp-server enable traps syslog
Cascade(config)#logging history notification
Cascade(config)#^Z

SNMP: V1 Trap, ent ciscoSyslogMIB.2, addr 10.2.1.1
 clogHistoryEntry.2.65 = SYS
 clogHistoryEntry.3.65 = 6
 clogHistoryEntry.4.65 = CONFIG_I
 clogHistoryEntry.5.65 = Configured from console by console
 clogHistoryEntry.6.65 = 30249161
```

上述 Syslog 消息因路由器改变配置产生。HistoryEntry 行中最后的 **65** 表示特定事件的索引。该 Syslog 历史条目范围从 1-6。条目 2 指示 OID 的值，是产生该消息的设备。例 9-18 中的设备是 SYS。条目 3 表示消息严重程度，该值为 6 表示 notification。条目 4 是该消息类型标识的文字描述。条目 5 是消息文字内容。条目 6 是消息产生时 SysUpTime 的值。

注：　Syslog MIB 的完全定义在 www.cisco.com/public/mibs/v1/CISCO-SYSLOG-MIB-V1SMI.my。

9.6　网络时间协议(NTP)

网络时间协议允许你使用集中时间源同步系统时钟。在网络上排除故障很少只牵涉单一系统。搜索日志文件——查找在特定时间内特定事件产生的错误消息——如果全网所有系统使用同一个时钟对错误消息打时戳将会使上述搜索简单得多。

9.6.1　NTP 概述

NTP 在分布的设备间同步时钟。每一个设备与时钟源建立对等关系。时钟源的可靠性由级别决定。1 级服务器直接连接在可靠的时钟源上，例如无线电接收的时钟、GPS 时钟卫星接收器、原子钟等。2 级服务器从 1 级时钟源获取时钟。2 级服务器可以通过互联网连接到对公众开放的可用的 1 级服务器。你可以在 www.eecis.udel.edu/~ntp/(从 Delaware 大学)找到许多公开的 NTP 服务器以及使用它们的信息。

在组织内配置的一组可靠的 NTP 服务器可以连接到公众的 2 级 NTP 服务器，为组织内所有的路由器提供时钟服务。当然路由器也可以做 NTP 服务器。事实上，如果无法连接到互联网或者防火墙不允许 NTP 协议通过，路由器可以配制成有效的 NTP 服务器。路由器

使用自身系统时钟作为参考时间。只有复位后能维护时钟的路由器-带日历的路由器-才能配制成有效的时钟源。任何其他路由器的参考时钟无效。如果 NTP 运行在由日历系统的路由器上并且路由器可以通过 NTP 获得时间，该日历系统可能由 NTP 更新来补偿内部时钟的漂移。如果网络上所有路由器时间同步的话，能简化故障排除，所以上述时钟并非需要可靠的原子钟。

NTP 非常有效。每分钟一个包能使两个设备同步误差在 10ms 以内。

9.6.2 路由器的 NTP 配置

配置 NTP 首先需要创建一个联接。使用下列命令初始化联接的创建：

ntp server *ip_address* [**version** *number*] [**key** *key_id*] [**source** *interface*] [**prefer**]

ntp peer *ip_address* [**version** *number*] [**key** *key_id*] [**source** *interface*] [**prefer**]

如果路由器将和另一个 NTP 时钟源同步，则建立一个服务器联接。如果路由器不但与另一个设备同步，而且允许其他设备和该路由器同步，则创建一个对等联接。

缺省的版本号是 3。缺省情况下，没有配置认证 key id，源 IP 地址时发送端口的 IP 地址。**Prefer** 关键字告诉 IOS 该同步对等体的优先级。

为控制对路由器 NTP 服务的访问，使用下列命令：

ntp access-group {query-only | serve-only | serve | peer } *access-list-number*

使用 **query-only** 选项值允许从列出的 IP 地址发出 NTP 控制查询。控制查询用在监视 NTP 进程的 SNMP 网络管理工作站。

serve-only 选项值允许访问控制列表上的 IP 地址请求时间。路由器不向远程系统同步时间。

serve 选项允许时间请求和控制查询，路由器同样不向远程系统同步时间。

peer 选项允许时间请求和控制查询，并且允许路由器从远程系统同步时间。

例 9-19 中的配置允许路由器：Seattle 从一个公共的二级时钟源同步，在与 Seattle 同一个网络上的另一个路由器 Tacoma 被允许从 Seattle 获得同步。

例 9-19 Seattle 获得同步时间；路由器 Tacoma 从 Seattle 获得同步

Seattle

```
access-list 1 permit 172.16.0.0 0.0.255.255
access-list 2 permit 128.105.39.11
ntp access-group peer 2
ntp access-group serve 1
ntp server 128.105.39.11
```

Tacoma

```
ntp server 172.16.1.5
```

Seattle 被允许从 128.105.39.11 同步时钟。任何地址在 172.16.0.0/16 范围内的路由器可以从 Seattle 同步时钟。

如果没有可用的外部时钟源，路由器可以配置成主时钟。配置成主时钟的路由器必须由一个内值的日历，用于在重启或掉电时维持日期和时间。可以配置 **clock calendar-valid** 命令

来使能日历，成为路由器合法的时间源。

为将 Cisco IOS 软件配制成其他对等实体可以同步的 NTP 主时钟，需要使用命令 **ntp master** [*stratum*]。要将路由器配置一个较高的等级数，防止路由器的时钟不会取代一个更低等级的时钟将时钟改写一个较低等级数(是更可靠的时钟源)。路由器缺省的等级数是 8。配置成 NTP 主时钟的路由器仍试图发现较低等级数的服务器。如果无法找到，路由器向配置的等级数同步。当路由器同步以后，其他路由器可以获得同步。

NTP 时间是 UTC。如果希望路由器保持在另一个时区，你可以使用下列命令保持本地时间：

clock timezone pst –8

clock summer-time PDT recurring

使用 **ntp update-calendar** 命令使路由器用 NTP 获得的时间更新内部日历。

NTP 协议可以使用认证。下列配置命令使能认证：

ntp authenticate

ntp authentication-key *number* **md5** *key*

ntp trusted-key *number*

ntp server *ip-address* **key** *number*

ntp authenticate 命令必须配置在 NTP 服务器和请求时间同步的路由器。该命令全局使能认证。**ntp authentication-key** 命令同样需要配置在所有路由器。该命令定义了一个认证串并将其赋值。

请求时间同步的路由器使用 **ntp trusted-key** 命令配置。该命令列出了 **ntp authentication-key** 命令定义的认证串的数值，该数值必须包含在 NTP 同步请求包中。所以 **ntp trusted-key** 命令只需配置在客户路由器端。

key *number* 选项必须包含在客户的 **ntp server** 命令中。该命令将密码包含到客户发往服务器的 NTP 包中。当路由器看到密码，如果该密码在服务器中有定义，则将密码包含到发往客户的 NTP 包中。

在例 9-20 的配置中，Seattle 与 Tacoma 间的认证已使能。

例 9-20　在 Seattle 与 Tacoma 间使能认证

```
Seattle
ntp authenticate
ntp authentication-key 10 md5 ntpkey
```

```
Tacoma
ntp authenticate
ntp authentication-key 10 md5 ntpkey
ntp trusted-key 10
ntp server seattle key 10
```

Tacoma 的 **ntp server seattle key 10** 命令指定在 Tacoma 与 Seattle 时钟同步以前，服务器 -Seattle 必须在 NTP 包中提供 10 号密码。从 Tacoma 的 NTP 包中，Seattle 看到 10 号密码。Seattle 中的 **ntp authentication-key 10 md5 ntpkey** 命令使能 Seattle 在对 Tacoma 的应答包中

包含认证密码 10。为举例说明认证，在服务器的配置中去掉 **authentication-key 命令**。例 9-21 显示了 **debug ntp packet** 命令的输出。起初，时间服务器 Seattle 没有配置认证。客户路由器 Tacoma 在从时间服务器同步时钟以前要求认证。

例 9-21 在 NTP 服务器上没有认证配置时的 NTP 包交换

```
Seattle
ntp clock-period 17179873
ntp server 128.105.39.11
```

```
NTP: xmit packet to 172.16.1.105:
leap 3, mode 3, version 3, stratum 0, ppoll 64
rtdel 1813 (94.040), rtdsp 3E25 (242.752), refid AC100169 (172.16.1.105)
ref BDD13136.BEAD46C0 (15:04:06.744 Eastern Thu Nov 30 2000)
org BDD13580.2FF266EC (15:22:24.187 Eastern Thu Nov 30 2000)
rec BDD13580.272011D4 (15:22:24.152 Eastern Thu Nov 30 2000)
xmt BDD13580.BD318A8C (15:22:24.739 Eastern Thu Nov 30 2000)
Authentication key 10

NTP: rcv packet from 172.16.1.105 to 172.16.1.106 on Ethernet0:
leap 0, mode 4, version 3, stratum 3, ppoll 64
rtdel 0FD1 (61.783), rtdsp 080C (31.433), refid 8069270B (128.105.39.11)
ref BDD1355B.9F07E00F (15:21:47.621 Eastern Thu Nov 30 2000)
org BDD13580.BD318A8C (15:22:24.739 Eastern Thu Nov 30 2000)
rec BDD13580.CB4EBD34 (15:22:24.794 Eastern Thu Nov 30 2000)
xmt BDD13580.CB6623DA (15:22:24.794 Eastern Thu Nov 30 2000)
inp BDD13580.C5C23E0A (15:22:24.772 Eastern Thu Nov 30 2000)
```

从 Tacoma 发往 Seattle 的调试包显示出包含了认证密码 10。Tacoma 期望认证密码 10 包含在从 Seattle 发来的 NTP 包中。但是没有包含。所以在例 9-22 中显示 NTP 状态保持未同步。

例 9-22 认证错误情况下 **show ntp status** 和 **show ntp association detail 命**令的显示

```
Tacoma#show ntp status
Clock is unsynchronized, stratum 16, no reference clock
nominal freq is 250.0000 Hz, actual freq is 249.9999 Hz, precision is 2**19
reference time is BDD13136.BEAD46C0 (15:04:06.744 Eastern Thu Nov 30 2000)
clock offset is 5.8229 msec, root delay is 94.04 msec
root dispersion is 242.74 msec, peer dispersion is 70.86 msec

Tacoma#show ntp association detail
172.16.1.105 configured, insane, invalid, unsynced, stratum 16
```

这里只显示了 **show ntp association detail** 命令的第一行输出。显示出对端是人工配置，对端没有通过基本安全检查(认证域)。由于对端是非认证的，时间无效，当前时钟等级维持缺省值 16。

然后在 Seattle 配置中增加 **ntp authentication-key 10 md5 ntpkey** 和 **ntp authenticate** 命令。例 9-23 和例 9-24 显示了 **debug ntp packet**、**show ntp status** 和 **show ntp association detail** 的相应输出。

例 9-23　在 NTP 服务器和客户端正确配置认证时 NTP 包交换

```
Seattle(config)#ntp authentication
Seattle(config)#ntp authentication-key 10 md5 ntpkey
Seattle(config)#^Z
```

```
Tacoma#
NTP: xmit packet to 172.16.1.105:
leap 3, mode 3, version 3, stratum 0, ppoll 64
rtdel 1813 (94.040), rtdsp 3E25 (242.752), refid AC100169 (172.16.1.105)
ref BDD13136.BEAD46C0 (15:04:06.744 Eastern Thu Nov 30 2000)
org BDD13600.B85D38DF (15:24:32.720 Eastern Thu Nov 30 2000)
rec BDD13600.B5E44B24 (15:24:32.710 Eastern Thu Nov 30 2000)
xmt BDD13640.CED09281 (15:25:36.807 Eastern Thu Nov 30 2000)
Authentication key 10

NTP: rcv packet from 172.16.1.105 to 172.16.1.106 on Ethernet0:
leap 0, mode 4, version 3, stratum 3, ppoll 64
rtdel 10CE (65.643), rtdsp 0821 (31.754), refid 8069270B (128.105.39.11)
ref BDD1361B.9EC9B021 (15:24:59.620 Eastern Thu Nov 30 2000)
org BDD13640.CED09281 (15:25:36.807 Eastern Thu Nov 30 2000)
rec BDD13640.DAE3EC4F (15:25:36.855 Eastern Thu Nov 30 2000)
xmt BDD13640.DB1FC317 (15:25:36.855 Eastern Thu Nov 30 2000)
inp BDD13640.D7686AD0 (15:25:36.841 Eastern Thu Nov 30 2000)
Authentication key 10
NTP: 172.16.1.105 reachable
NTP: sync change
NTP: peer stratum change
```

Seattle 到 Tacoma 的 NTP 包中包含了所期望的认证密码。对端在 NTP 协议下可达，Tacoma 的非认证状态发生变化，时钟等级也从缺省值发生变化。例 9-24 显示了新的 NTP 状态。

例 9-24　认证正确后 **show ntp status** 和 **show ntp association detail** 命令的输出显示

```
Tacoma#show ntp status
Clock is synchronized, stratum 4, reference is 172.16.1.105
nominal freq is 250.0000 Hz, actual freq is 249.9999 Hz, precision is 2**19
reference time is BDD13640.D7686AD0 (15:25:36.841 Eastern Thu Nov 30 2000)
clock offset is 30.8433 msec, root delay is 98.30 msec
root dispersion is 15937.61 msec, peer dispersion is 15875.02 msec

Tacoma#show ntp association detail
172.16.1.105 configured, authenticated, our_master, sane, valid, stratum 3
```

现在，时钟是同步的，等级从 16 改为 4，对端成为已认证，对端提供的时间被认为是有效的。

9.7　记账

有时为对网络使用计费，收集关于流量的统计信息非常有用。这些信息不但对基于流量的网络收费有用，而且对流量工程也很有用。

你可以在路由器接口上使能基本 IP 记账。记账内容包括列出的 IP 源和目的地址，还有

两节点之间传输的字节数和包数。NetFlows 提供了更强大的记账功能。除源和目的地址、字节数、包数外，还提供协议和 AS 信息。NetFlows 数据可以按照不同方式分类，例如按照自治系统、子网前缀或协议类型。NetFlows 将在“NetFlows”章节中详细讨论。

9.7.1　IP 记账

IP 记账提供基本的记账服务。经过路由器的数据包按照源/目的地址计数。源于路由器或发往路由器的包不作计数。记账工作在发送端口完成。IP 记账禁止接口上的自治交换和 SSE 交换。作为可选，没有通过访问控制的包也能计数。出现大量违反接入控制的数据包可能由于路由器误配置或者网络攻击。

你可以在输出接口上使用 **ip accounting** 命令来使能记账。

为显示在输出接口使能记账的结果，可以使用 **show ip accounting [checkpoing] [access-violation]**命令。

例 9-25 在以太网接口上使能 IP 记账。

例 9-25　显示了在以太网接口上收集到的 IP 记账数据，**show ip accounting** 命令显示接口上发送了许多多播包

```
Bowler(config)#int e 0
Bowler(config-if)#ip accounting
Bowler(config-if)#^Z
Bowler#show ip accounting
   Source            Destination                Packets                Bytes
 10.1.1.88         228.13.20.216                   45                24611

Accounting data age is 0
Bowler#show ip accounting
   Source            Destination                Packets                Bytes
 10.1.1.88         224.2.127.254                    1                  229
 10.1.1.88         228.13.20.216                  133                73689

Accounting data age is 0
Bowler#show ip accounting
   Source            Destination                Packets                Bytes
 10.1.1.88         224.2.127.254                    1                  229
 10.1.1.88         228.13.20.216                  173                95952

Accounting data age is 0
```

路由器 Bowler 的以太网接口 Ethernet 0 上使能了 IP 记账。包在流出以太网接口时被计数。几张表上 3 个显示子序列显示出源地址 10.1.1.88 发往 224.2.127.254 和 228.13.20.216 的多播包。

你可以使用 **clear ip accounting** 命令清除记账表。

IP 记账可提供接口上有价值的信息。注意，当你实现 IP 记账时可能会影响性能。由于使用 IP 记账会禁止自治交换和 SSE 交换，包在接口交换时会使用比设计网络更低效率的交换机制。另外维护记账数据库会消耗路由器的内存。当路由器内存利用率很高时不要使能 IP 记账。

ip accounting-threshold *threshold* 能为记账数据库中存储的帐务条目定义门限。缺省值是

512 个源/目的地址对。这样的缺省配置下每一个数据库、行为和检查点可能最多使用 12,928 字节内存。如果你将门限值配置的过高，可能会消耗所有可用内存。

在接口上使能 IP 记账能以最快的方式按照源/目的地址浏览流量，但是没有内建的机制将记账数据传输到服务器供其将来使用。NetFlows 在提供附加信息之外，还提供了上述 IP 记账功能未提供的功能。

9.7.2 NetFlow

NetFlow 交换标识流量并处理路由器内的访问列表和交换功能。另外由于流量已标识，按照流量的统计可以输出到计费服务器。当数据流活跃激活时，关于流的数据维护在 NetFlow 的缓存中。当数据流结束后，上述数据能增加到聚合缓存，并且输出到管理工作站。NetFlow 缺省条件下包含 64k 个数据流缓存条目。

> 注：　NetFlow 交换模式比其他交换模式占用更多的 CPU 与内存。在使能 NetFlow 以前，必须清楚理解路由器上需要的资源。

为使能 NetFlow 交换，需要使用接口子命令 **ip route-cache flow**。

使用下列全局命令定义接收所采集数据的 IP 地址和 UDP 端口号：

ip flow-export destination *ip-address udp-port*

ip flow-export [version 1| version 5 [origin-as | peer-as]]

版本号必须匹配流量采集者所期望的版本号。缺省情况下是版本 1。

origin-as 选项指定输出的数据携带源和目的的 BGP 起源 AS 号。

peer-as 选项指定输出的数据在源/目的地之上携带接收数据的路由器的 BGP AS 号，而不是流量源和目的真实 AS 号。

图 9-2 显示运行 BGP 得简单网络，并且在路由器 Hammer 上收集 NetFlows 数据。

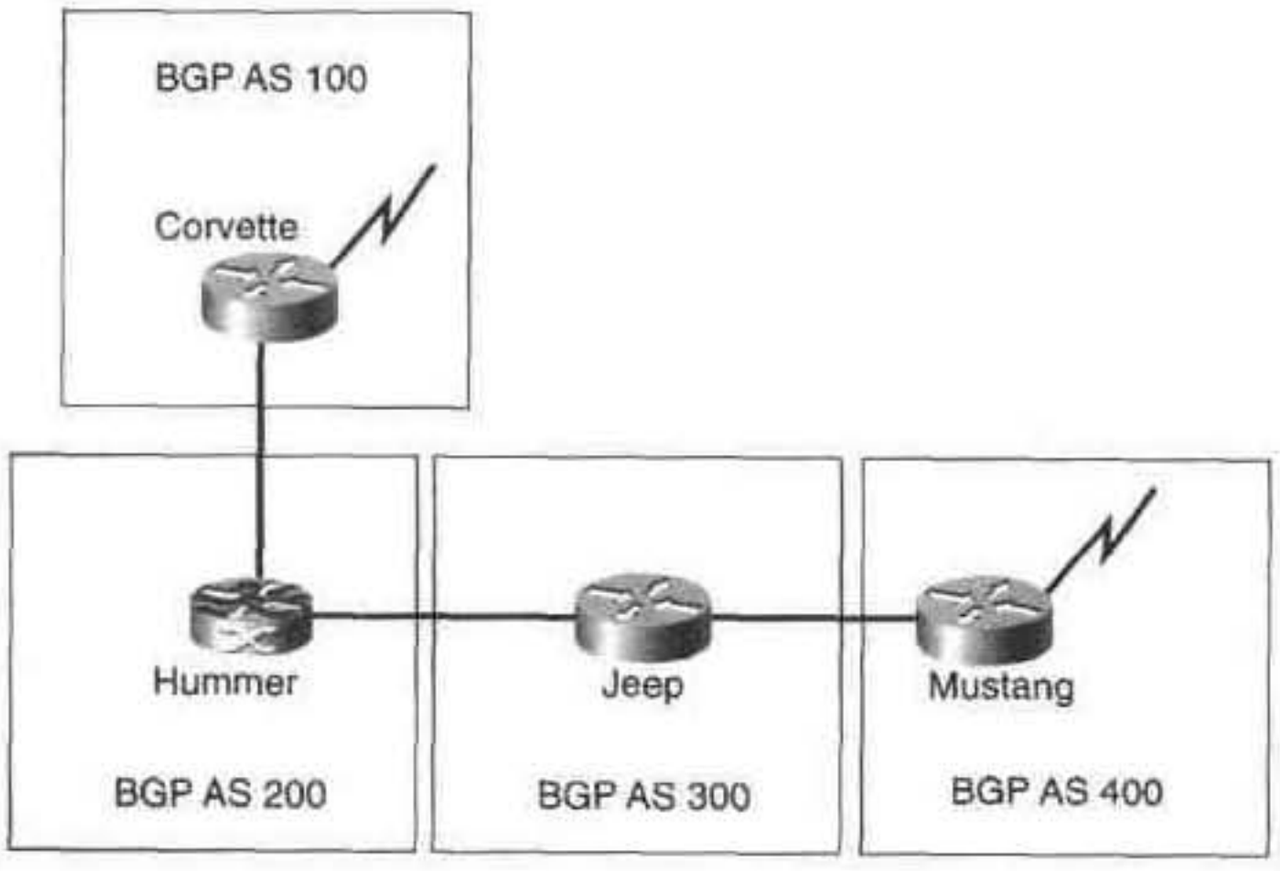


图 9-2　运行 NetFlow 对流量记账的 BGP 网络

在 Hummer 路由器使能 NetFlows。路由器被配制成在所有两个端口上采集信息。例 9-26 显示了 Hummer 路由器的配置。

例 9-26　图 9-2 中 Hummer 路由器的配置，该配置在两个以太网接口上收集流量信息

```
interface Ethernet1/2
 ip address 1.1.7.5 255.255.255.0
 ip route-cache flow
!
interface Ethernet1/3
 ip address 1.1.5.5 255.255.255.0
 ip route-cache flow
!
ip flow-export version 5 peer-as
ip flow-export destination 1.1.3.250 125
```

在例 9-26 的配置中，所收集数据包括 peer 对端 AS 而不是 origin 发起 AS。**show ip flow export** 命令显示数据的输出参数，**show ip cache flow** 命令显示数据流缓存。例 9-27 显示数据流输出参数和路由器 Hummer 数据流缓存的样本。

例 9-27　show ip flow export 和 show ip cache flow 命令显示的 NetFlow 数据流信息

```
Hummer#show ip flow export
Flow export is enabled
  Exporting flows to 1.1.3.250 (125)
  Exporting using source IP address 1.1.7.5
  Version 5 flow records, peer-as
  527 flows exported in 18 udp datagrams
  0 flows failed due to lack of export packet
  0 export packets were sent up to process level
  0 export packets were dropped due to no fib
  0 export packets were dropped due to adjacency issues

Hummer#show ip cache flow
IP packet size distribution (51719 total packets):
   1-32   64  96  128  160  192  224  256  288  320  352  384  416 448  480
   .131 .000 .034 .000 .000 .000 .490 .000 .000 .000 .000 .000 .000 .000 .000

    512  544  576 1024 1536 2048 2560 3072 3584 4096 4608
   .000 .000 .000 .000 .343 .000 .000 .000 .000 .000 .000

IP Flow Switching Cache, 4456704 bytes
  68 active, 65468 inactive, 1080 added
  22140 ager polls, 0 flow alloc failures
  Active flows timeout in 30 minutes
  Inactive flows timeout in 15 seconds
  last clearing of statistics 00:08:35
Protocol         Total    Flows   Packets Bytes  Packets Active(Sec) Idle(Sec)
--------         Flows     /Sec     /Flow  /Pkt     /Sec     /Flow    /Flow
TCP-Telnet          19      0.0        90    70      3.3       0.0   15.5
TCP-WWW            596      1.1        40   220     46.7       0.0   15.4
UDP-DNS            397      0.7         3    28      2.3       0.0   15.5
Total:            1012      1.9        26   201     52.4       0.0   15.4

SrcIf         SrcIPaddress    DstIf         DstIPaddress    Pr SrcP  DstP  Pkts
Et1/3         1.1.2.13        Et1/2         1.1.3.13        06 0403 0015   17K
Et1/3         1.1.2.12        Et1/2         1.1.3.12        06 042D 0017   90
Et1/3         1.1.2.11        Et1/2         1.1.3.11        06 099E 0050   40
Et1/3         1.1.2.21        Et1/2         1.1.3.21        11 07D3 0035   3
Et1/3         1.1.2.21        Et1/2         1.1.3.21        11 07D2 0035   3
Et1/3         1.1.2.21        Et1/2         1.1.3.21        11 07D1 0035   3
Et1/3         1.1.2.21        Et1/2         1.1.3.21        11 07D0 0035   3
```

你可以看到，与 IP 记账相比，NetFlows 包含了更多的信息-每一个流的包大小分布，摘要信息、协议信息。

还可以用不同的方法将数据流信息聚合分组：按照自治系统号分组、按照源和目的地前缀和按照协议端口。聚合缓存能够在输出流量数据之前聚合数据。流量数据在 NetFlows 主缓存过期之前进入到每一个使能的聚合缓存。

流量聚合只能使用 NetFlows 版本 8。版本 8 允许聚合缓存输出。版本 5 的主缓存需要使用 **peer-as** 和 **origin-as** 命令选项指定。

在配置流量聚合以前，必须使能 Cisco 快速转发(CEF)和 NetFlow 交换。使能 CEF 按照包的源和目的地址转移转发缓存，上述转移用于数据聚合。

> 注：　参见配置指南 12.1Cisco 快速转发概述，CCO 上交换服务配置指南，更多的信息在 CEF 上。

使用全局命令 **ip cef** 配置 CEF，使能所有支持 CEF 接口上的 CEF 路由-缓存。下面的全局命令定义　个聚合缓存：

ip flow-aggregation cache {*autonomous_system* | *destination_prefix* | *prefix* | *protocol-port* | *source-prefix*}

表 9-4 记录了一些可以用在缓存上的命令。所有这些命令应当在聚合缓存配置模式下输入。

表 9-4　　缓 存 命 令

命　　令	功　　能
cache entries *number_of_entries*	设置缓存条目限制，范围从 1024 到 524288，缺省是 4096
cache timeout inactive *seconds*	定义不活跃激活的条目超时以前保持在缓存的时间，范围从 10-600 秒，缺省 15 秒
cache timeout active *minutes*	定义活跃激活的条目保持活跃激活的分钟数，范围从 1-60 分钟。缺省 30 分钟
export destination *ip_address udp_port*	在聚合缓存配置模式下指定输出聚合缓存流数据目的地 Ip 地址和 UDP 端口。该数据采集器应接收版本 8 流记录
enabled	使能聚合缓存

AS 聚合组将相同源 BGP AS、目的 BGP AS、接收端口、输出端口的流聚合。输出数据中的聚合记录摘要了流的数量、包数和字节数。

例 9-28 显示了 Hammer 路由器的 AS 聚合配置。

例 9-28　图 9-2 中 Hammer 路由器的 AS 聚合配置

```
ip cef
!
ip flow-export version 5 origin-as
ip flow-export destination 1.1.3.250 125
ip flow-aggregation cache as
 cache entries 2048
 cache timeout inactive 200
 cache timeout active 45
 export destination 1.1.3.250 9991
 enabled
!
```

例 9-29 使用 **show ip cache flow aggregation as** 命令显示了 AS 聚合缓存。

例 9-29　使用 **show ip cache flow aggregation as** 命令察看 AS 聚合缓存内容

```
Hummer#show ip cache flow aggregation as

IP Flow Switching Cache, 135048 bytes
  1 active, 2043 inactive, 3 added
  167 ager polls, 0 flow alloc failures
  Active flows timeout in 45 minutes
  Inactive flows timeout in 200 seconds

Src If       Src AS  Dst If       Dst AS  Flows   Pkts  B/Pk  Active
 Et1/3        400    Et1/2        100     357     42K   848   407.6
```

存在 357 个流关联在源以太网接口 Ethernet1/3，源 AS 400，目的接口以太网 Ethernet1/2，目的 AS 100。

使能前缀聚合能够得到更多。前缀聚合除了与 AS 聚合一样将流量基于源和目的 BGP AS、输入输出接口分组，还可以基于源和目的的前缀和掩码分组。

目的-前缀聚合组将数据流按照相同的目的前缀、目的前缀掩码、目的 BGP AS 和输出端口分组。使用上述分组方式来检查穿过 NetFlows 路由器的流量的目的地信息。

例 9-30 中的配置信息被增加到路由器 Hammer。

例 9-30　在图 9-2 中的 Hammer 路由器配置目的前缀聚合

```
ip flow-aggregation cache destination-prefix
 cache entries 2046
 cache timeout inactive 200
 cache timeout active 45
 export destination 1.1.3.250 9991
 enabled
```

例 9-31 显示了目的前缀聚合缓存，例中使用 **show ip cache flow aggregation destination-prefix** 命令察看前缀聚合缓存内容。

例 9-31　目的前缀聚合缓存

```
Hummer#show ip cache flow aggregation destination-prefix

IP Flow Switching Cache, 135048 bytes
  1 active, 2045 inactive, 1 added
  240 ager polls, 0 flow alloc failures
  Active flows timeout in 45 minutes
  Inactive flows timeout in 200 seconds

Dst If       Dst Prefix      Msk  AS    Flows   Pkts B/Pk  Active
Et1/2        1.1.3.0         /24  100   324      11K  442   239.5
```

存在 324 个流关联在目的接口以太网 Ethernet1/2、目的前缀 1.1.3.0、掩码/24 和目的 AS 100。

你同样可以使用源前缀聚合机制使用源信息检查流量。该机制通过源前缀、源前缀掩码、

源 BGP AS 和源接口将数据分组。

将例 9-32 中的配置增加到路由器 Hammer。

例 9-32　图 9-2 中的路由器 Hammer 配置源-前缀聚合

```
ip flow-aggregation cache source-prefix
 cache entries 2046
 cache timeout inactive 200
 cache timeout active 45
 export destination 1.1.3.250 9991
 enabled
```

例 9-33 显示源前缀聚合流。

例 9-33　使用 **show ip cache flow aggregation source-prefix** 命令察看前缀聚合缓存内容

```
Hummer#show ip cache flow aggregation source-prefix

IP Flow Switching Cache, 135048 bytes
  2 active, 2044 inactive, 3 added
  440 ager polls, 0 flow alloc failures
  Active flows timeout in 45 minutes
  Inactive flows timeout in 200 seconds

Src If         Src Prefix      Msk  AS    Flows   Pkts B/Pk  Active
Et1/3          1.1.2.0         /24  400     181   4813   200    42.0
Et1/2          1.1.7.0         /24  0         1      1    44     0.0
```

存在 181 个流关联在源接口以太网 Ethernet1/3、源前缀 1.1.2.0、掩码/24 和源 AS 400。

如果你需要根据流量类型检查，则使能协议-端口聚合。流量将根据相同的 IP 协议、源端口号和目的端口号分组。

将例 9-34 中的配置增加到 Hammer 来配置协议-端口聚合。

例 9-34　将图 9-2 中的 Hammer 路由器配置协议-端口聚合

```
ip flow-aggregation cache protocol-port
 cache entries 2046
 cache timeout inactive 200
 cache timeout active 45
 export destination 1.1.3.250 9991
 enabled
```

例 9-35 显示了协议-端口聚合缓存。

例 9-35　**show ip cache flow aggregation protocol-port** 命令察看协议端口聚合缓存内容

```
Hummer#show ip cache flow aggregation protocol-port

IP Flow Switching Cache, 135048 bytes
  14 active, 1972 inactive, 74 added
  882 ager polls, 0 flow alloc failures
  Active flows timeout in 45 minutes
  Inactive flows timeout in 200 seconds
```

Protocol	Source Port	Dest Port	Flows	Packets	Bytes/Packet	Active
0x06	0x0401	0x0017	1	90	70	0.0
0x06	0x0400	0x0017	1	90	70	0.0
0x11	0x0404	0x0035	1	3	28	0.0
0x11	0x0405	0x0035	1	3	28	0.0
0x11	0x0406	0x0035	1	3	28	0.0
0x11	0x0407	0x0035	1	3	28	0.0
0x11	0x0400	0x0035	1	3	28	0.0
0x11	0x0414	0x0035	1	3	28	0.0
0x11	0x0415	0x0035	1	3	28	0.0
0x06	0x040B	0x0050	1	40	220	0.0
0x06	0x0408	0x0050	1	40	220	0.0
0x06	0x0409	0x0050	1	40	220	0.0
0x06	0x0436	0x0050	1	40	220	0.0
0x06	0x0437	0x0050	1	40	220	0.0

存在 14 个不同的协议端口流。他们按照 IP 协议、源端口和目的端口分组。

不同的聚合缓存为数据流流量聚合方式提供了灵活性。这些信息可以用作流量分析或者计费。

表 9-5 中列出了对于每种聚合机制 UDP 包中的最多流记录数以及相应 UDP 包大小。

表 9-5　　对每种聚合机制 UDP 包中的最多流记录数以及相应 UDP 包大小的列表

聚 合 机 制	每个 UDP 包中最多流记录	最大 UDP 包大小
BGP 自治系统	51	1456
目的地址前缀	44	1436
前缀	35	1428
协议端口	51	1456
源地址前缀	44	1436

Cisco NetFlows 流收集器是对 NetFlows 数据采集和报告的应用。流收集器将多个 Cisco 路由器(交换机)输出的数据聚合。你可以将收集的数据过滤或聚合以满足网管需要。

注：　你可以在 www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/nfc/nfc_3_0/nfc_ug/index.htm 上 CCO 中发现 Cisco 流收集器的详细信息。

9.8　配置管理

从不同的网络设备周期性(每周或每晚)下载的配置维护在一个数据库中，确保路由器需要最新配置时使用。当现存路由器被替换，配置丢失或者配置错误时，会需要恢复配置文件。

CiscoWorks 提供通过 TFTP 从路由器下载配置的能力。事实上路由器配置可以下载到任

何 TFTP 服务器。许多组织使用 UNIX TFTP 服务器和 Perl 脚本，周期性执行脚本来实施配置管理。组织的策略可能是将配置存储 7 天。文件名可以是例如 routernme.current 和 routername.1、routername.2、rautername.3 等代表 7 天的配置。一个每晚运行的简单脚本可以将文件 routername.N 复制到 routername.N+1(N 从 1 到 6)，然后将文件 routername.current 复制到 routername.1，连接到路由器使用 TFTP 下载配置到 routername.current，最后断开路由器。脚本可以在网络的一组路由器上循环执行。

注：　你可以在下面两本书上得到 Perl 的详细信息：由 Randel L. Schwortz，Tom Christiansen 和 Steve Talbot(编辑)撰写的 Learning Perl 第二版(Q'Reilly & Associate，Inc，July 1997)，和由 Larry Wall，Jon Orwant 和 Tom Christiansen 撰写的 Programming Perl(Q'Reilly & Associate，Inc，July 2000)。

你应当存储一段时间内的配置文件，至少一个星期，一个月的最理想，用作对路由器恢复已知可用的配置或者对可能发生在以前的配置改变查找故障。使用每天的配置文件和改变管理日志，可以很容易地在路由器上恢复可用的配置。

9.9　故障管理

一个可靠的网络需要故障管理系统。只有尽快地检查出潜在的和存在的问题，你才能迅速采取行动来解决问题。故障管理系统有希望在终端用户发现之前探测出系统或链路的问题。

当配置 SNMP 的路由器检查到故障时向管理工作站发送 traps。由于 SNMP 采用 UDP 发送 trap，所以无法确保描述故障的消息到达管理工作站。故障管理系统不能只依赖 trap，还应该轮询路由器的链路状态、接口和路由器部件。在轮询路由器部件信息之外，管理工作站还轮询路由器本身，有时使用 ICMP Ping 来确认路由器可访问。为确认路由器可以通过任何一个活跃激活的端口可访问，可以在路由器上配置一个 Loopback 环回地址来标识路由器的 IP 地址。管理工作站轮询或 Ping 环回地址可以从任何可能的路径到达路由器。如果管理工作站轮询或 Ping 路由器的非环回 Loopback 接口地址，可能出现下面情况：当所轮询接口出现故障时，即使路由器可以通过其他接口访问，管理工作站还是认为该路由器不可访问。你应当将 trap 的源地址配制成环回地址，将管理工作站配制成轮询环回地址。

正如许多其他协议，存在一个所引发网络流量多少与管理工作站检测异常迅速程度的折衷。如果管理工作站没有收到 trap，则必须依靠 Ping 或轮询来检测异常，这种情况下可能有很长一段时间异常没有被发现。如果路由器发生故障时管理工作站配制成每 5 分钟 ping 一次且 3 次 ping 不到就认为路由器故障，则需要 15 分钟才能发现路由器故障。链路或其他部件的故障发现则快得多。管理工作站并不是依靠这些部件没有反应来检查异常，而是向路由器查询这些部件的状态。路由器用状态信息作响应。

在图 9-3 中管理工作站向路由器轮询接口状态。

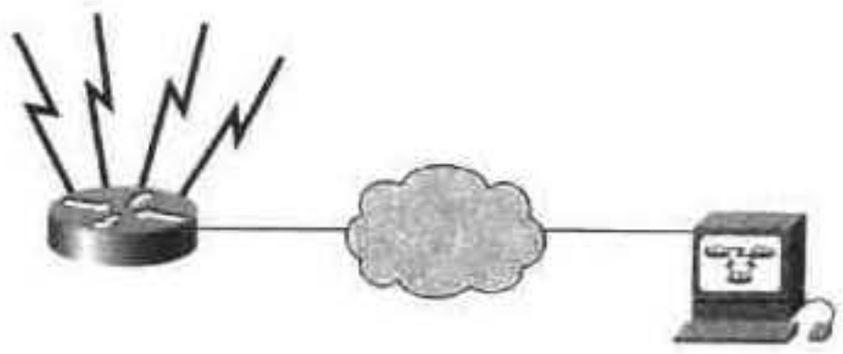


图 9-3 管理工作站向路由器轮询接口状态

管理工作站使用 MIB 中的 ifEntry.ifOperStatues 对象 ID 向路由器查询接口状态。路由器响应 3 个接口 up，一个接口 down。

故障管理系统检测故障。故障会以可见或声音方式可闻或者通过电子邮件或寻呼等方式通知网络管理员。发送告警的方式由用户环境自行定义。如果网管控制台每周 7 天每天 24 小时有人值守，则可见或声音可闻方式告警就足够了。如果控制台并不总有人值守，则无人值守时发送电子邮件或寻呼告警。故障表示链路、路由器或路由器部件中断。故障之后的告警非常常见。故障管理工作站同样也试图在故障发生之前向管理员发告警。

在很多时候，特定的事件是由故障部件引起的。例如串行线路在完全无法使用之前会报告大量错误或载波传输。路由器在故障前会报告内存问题。故障管理工作站维护门限信息，当门限被超过时向网管人员发送告警。你可以对任意数量的变量配置门限。为配置这些门限值，首先需要得到网络基线。获得基线需要网络正常运行一段时间，例如一星期。在一段时间的正常运行中得到变量正常值。然后你可以配置门限，例如超过正常值 20%。

许多 MIB 变量提供有用的门限信息，包括：

- 空闲内存总量
- 平均 CPU 利用率
- 丢失的缓存
- 接口输入输出速率
- 接口输入输出错误
- 接口输入输出队列丢弃
- 接口忽略的包数量
- 接口复位
- 串行接口 CRC、中断和帧错误
- 帧中继 FECN 和 BECN
- 串行接口载波传输
- 以太网碰撞
- 以太网超小包、超长常包和帧错误
- 令牌环线路和突发错误

- 令牌环内部错误
- 令牌环令牌和软件错误
- 令牌环信号丢失

某些上述事件发生在正常运行的网络中。只有超出门限时才会影响性能，甚至可能孕育更严重的问题。管理工作站向路由器周期性地轮询上述变量的值。如果轮询期间内变化的值超过了门限，则产生告警。

你也可以使用 RMON 管理门限。使用 RMON 时管理工作站不需要轮询变量的值。路由器上的 RMON 代理会本地轮询变量值，在超出门限时向管理工作站发 trap。管理工作站收到 trap，然后产生告警。在这里是网络利用率与路由器处理能力的折衷。使能 RMON 会减少网络流量，但是增加路由器的处理负担。

9.10　性能管理

性能管理用在趋势和能力的计划。采集数据然后分析。网络工程师和管理员浏览这些数据，寻找指示增加或减少网络能力的趋势。链路利用率、帧中继的 FECN 和 BECN 以及平均 CPU 利用率可能指示网络能力需要的变化。在网络边缘(尽可能靠近用户)路由器间定期测试的响应时间直接显示了网络部件超利用率或低利用率的效果。响应时间也显示网络能力改变以后的改进。

性能管理工作站连续地收集数据。该工作站通过 SNMP 收集数据：周期性短时间间隔(例如 5 分钟)轮询一组变量。所得数据存储在原始表格中以备研究。性能管理工作站也计算并报告基于小时的最小最大和平均值。上述数据可以在白天处理(对前几个小时作计算)，提供准实时的报告和图像；也可以晚上处理，提供前几天数据的详细报告。

系统应当报告每天所轮询值低于某范围的总时间数。

例如考虑链路利用率。能够得到每天的时间在下列利用率范围：0～20%，20%～40%，40%～60%，60%～80%，80%～90%和 90%～100%的分布非常有用。

经过一些天的数据收集和处理，系统可以对数据进一步处理，系统可以报告每天的最小最大和平均值以及一段时间内的范围分布。如果时间段是一星期，应当报告每天的最小最大和平均值以及整个星期内所分析值在整个范围内的总时间。

性能管理系统需要不间断地收集数据。只有收集完所期望的一段时间内的数据，才能作有意义的趋势分析。将趋势外延到未来能够帮助确定突破点什么时候发生。

灵活性能使性能管理系统更有价值。配置察看数据的时间段，尽可能实时的报告能够使性能管理系统称为发现能力变化需求的有效工具。

9.11　安全管理

要保证网络安全，路由器本身必须是安全的。口令是控制访问路由器的一种方式。你可以在路由器上配置口令，或者使用例如 TACACS+或 RADIUS 等认证服务器。在口令保护之外，你应当将对路由器的交互访问限制到必须的用户和协议。应当只使用能对路由器恰当的管理和功能必要的协议，并且将访问权限严格限制在你所知道安全的 IP 子网。即使存在访问

控制，仍可能有恶意用户试图恶意影响路由器正常工作。应采取行动减少路由器受拒绝服务攻击的机会。下面几个章节讨论提供路由器安全正确工作所需要的配置参数。

9.11.1 口令类型和加密

你应当使用一些认证机制来控制所有的访问。如果无法使用 TACACS+或 RADIUS，你应当使用 **enable secret** 口令类型保护特权 EXEC 模式的路由器访问。不要使用旧的 **enable password**，因为旧的方式使用弱加密算法。**enable secret** *password* 命令将口令使用不可逆加密功能存储，能提供更好的安全性。在配置文件和口令存储在 TFTP 服务器，需要穿过网络传输的环境中，增加的安全加密层被证明非常有用。使用 **service password-encryption** 命令将所有的口令加密能防止显示路由器配置时旁观者看到口令，这些口令包括用户名口令、认证密钥口令、特权命令口令、控制台和虚拟终端线路访问口令以及 BGP 邻居口令。

9.11.2 控制交互式访问

你应当控制对路由器的交互式访问。你可以使用下面的命令来只允许特定网络号的访问：

access-class *access-list_1-199_or_1300-2699* {**in** | **out**}

access-list 参数指定允许连接到线路的源网络号(使用关键字 **in**)，或者指定可以建立连接的网络号(使用关键字 **out**)。

确保没有只允许想要远程访问协议的访问漏洞，例如：

transport input telnet ssh

列出允许的协议，拒绝其他所有协议。

在登录到控制台和辅助端口所需要之外，你还应当使用口令保护连接到路由器上的调制解调器。

如果调制解调器只用作播入(管理员从家中访问路由器)，应当禁止从网络上反向远程登录到调制解调器向外拨号的能力。反向远程登录提供使用 IP 地址和端口号连接到路由器异步端口上设备的能力。在连接调制解调器的线路上使用 **transport input none** 命令能使所有连接到不用作播出的异步终端或调制解调器的端口上禁止反向远程登录。

例 9-36 显示配置了所有讨论的介入控制方式的路由器。

例 9-36 在路由器上控制交互式访问

```
access-list 1 permit 172.16.0.0 0.0.255.255
line con 0
 transport input none
line aux 0
 transport input none
line vty 0 4
 access-class 1 in
 transport input telnet ssh
```

远程登录和安全的命令解释器访问是唯一允许的远程协议，只有源 IP 地址是 172.16.0.0/16 范围内的人才能够使用上述协议连接到路由器。

9.11.3　减少拒绝服务攻击的危险

拒绝服务(DoS)对某些资源作拒绝访问攻击。某些人可以利用许多途径实施 DoS 攻击。你可以采取一些行动，减少对路由器攻击产生的危险。

在路由器上只有有限的 vty 端口可用。当所有的端口都占用时其他远程会话无法连接到路由器，为 DoS 攻击提供潜在目标。入侵者可能堵塞所有的 vty 接口，对管理员拒绝服务。可以在最后一个 vty 端口上配置限制性的 **access-class** 命令。只允许指定的管理工作站访问。这样至少有一个端口可访问。同时还应该配置 **exec-timeout** 命令，防止空闲的会话无限制占用 vty。**service tcp-keepalive-in** 命令在收到的连接上配置 TCP keepalive 消息，防止恶意攻击和远端系统崩溃引起的“orphaned”会话。

一种 DoS 攻击使用直接广播。使用一个伪造的源地址向一个直接广播地址发送 ICMP 包，局域网上所有的机器都将作应答，向所伪造的源地址发送大量的数据。拥有真的该源地址的节点将收到大量的数据。在局域网接口上配置 **no ip directed-broadcast** 命令能够阻碍这样的攻击。**no ip directed-broadcast** 命令在 IOS 12.0 及后续版本中作为缺省配置。

带有 **source-route** 选项的 IP 包在从源到目的地的传输中指定了路由。返回的包也必须使用相同的路由。源地址路由中虚假的源地址能够使节点忽略路由表，将数据送到虚假的地址。网络上很少有有效的源地址路由包。可以使用 **no ip source-route** 命令能使路由器丢弃带有 **source-route** 选项的包。

快速涌来的包会使路由器花费太多的时间处理接口中断，没有时间处理其他事件。**scheduler interval** *milliseconds* 命令高速路由器在规定的时间间隔停止处理中断，改为处理其他事务。新的平台可能使用 **scheduler allocate** *interrupt-time process-time* 命令来替代。

interrupt-time 参数表示路由器在网络中断上下文中用于快速交换的最多微秒数。*process-time* 参数表示网络中断禁止时路由器运行的最少微秒数。

路由器运行一些小的服务用作诊断目的，这些服务很少被使用。路由器的TCP服务是Echo、Chargen、Discard 和 DayTime。路由器的 UDP 服务是 Echo、Chargen 和 Discard。攻击者可能会向这些服务洪泛大量的流量，损害路由器的选路能力。下列命令可以禁止这些服务：

no service tcp-small-servers

no service udp-small-servers

Cisco 软件版本 12.0 及更高版本缺省情况下禁止上述服务。

如果不使用 Finger 和异步线路上的 BOOTP 服务器服务的话，这些服务也应当禁止。使用下面的命令来禁止上述服务：

no service finger

no ip bootp server

Finger 服务允许 Finger 协议向路由器作请求。Finger 协议请求相当于远程执行 **show users** 命令，显示路由器上激活线路的信息。

路由器为连接在异步线路上的主机提供 BOOTP 服务。**no ip bootp server** 命令禁止 BOOTP 服务。

例 9-37 中的配置显示使用上面讨论的命令使遭受 DoS 攻击的危险最小化。

例 9-37 最小化 DoS 攻击危险

```
enable secret jj150p
service tcp-keepalives-in
scheduler interval 500
no server tcp-small-servers
no server udp-small-servers
no service finger
no ip bootp server
no ip source-route
int e 0
 no ip directed-broadcast
access-list 10 permit 172.16.1.2
line vty 4
 access-class 10 in
 transport input telnet
```

可以配置例如 TACACS+的远程认证和认证服务器来使路由器更安全。

9.11.4 TACACS+

增强的终端访问控制器访问控制系统(TACACS+)提供对试图访问路由器或网络访问服务器的用户的集中认证。在服务器上有 TACACS+的应用作为守护进程运行，在数据库中存储访问特权信息。当用户登录到配置 TACACS+的路由器上时，路由器上的 TACACS+客户与 TACACS+守护进程通信发送登录和口令提示，并交换认证与授权信息。路由器还发送记账信息到 TACACS+服务器。尽管路由器与用户间的通信可能没有加密，路由器与 TACACS+访问期间所有的通信都是加密的。

TACACS+提供认证、授权和记账信息：

- TACACS+认证需要用户输入登录 ID 和口令。认证服务器可以向登录用户发送消息，例如请求改变口令。
- TACACS+授权调整登录用户能干什么。授权可以在登录后自动运行命令，提高接入控制或限制会话持续时间。授权还可以限制登录在路由器上的用户可使用的命令。
- TACACS+记账收集信息用作计费、审计或报告，并发往 TACACS+服务器。记账所记录的信息包括用户 ID、起始和结束时间、运行的命令、包的数量和字节的数量。上述信息对安全审计和计费非常有用。当 TACACS+用作控制网络接入服务器用户访问控制时，网络接入服务器允许用户使用网络服务，计费信息非常有用。

1. TACACS+认证配置

路由器使用 AAA 来使能 TACACS+。**aaa new-model** 命令使能 AAA。

定义了一组认证方式的命令如下所示：

aaa authentication login {**default** | *list_name*} **group** *auth_type* [*auth_type* ...]

认证方式可以是 **tacacs+**、**radius**、**kerberos**、**local**、**line password**、**enable password** 和 **none**。

认证列表定义以后，被应用到线路上。关键字 **default** 将列表自动应用到所有线路上。当你第一次配置认证时，指定列表名并手工应用到线路上是一个较好的方法。这样你可以在一种可控制的方式下测试你的配置，不至于将你自己锁在路由器之外。

使用下列命令将列表应用到线路上：

line *type number*

login authentication *list_name*

你还需要指定 TACACS+服务器的位置。使用 **tacacs-server host** *ip_address* 命令来指定服务器的 IP 地址。

例 9-38 示范了路由器配置 TACACS+认证。

例 9-38　路由器的 TACACS+认证配置

```
aaa new-model
aaa authentication login tac tacacs+  enable
tacacs-server host 172.16.1.2

line vty 0 1
 login authentication tac
!line vty 2 4
! login authentication tac
!line con 0
! login authentication tac
!line aux 0
! login authentication tac
```

认证列表中 **tac** 试图使用 TACACS+服务器 172.16.1.2 认证。如果服务器不可达，则使用第二种认证方式 **enable password**。第二种方式允许在 TACACS+服务器不可访问时访问路由器。只有在某种方式不可用时才使用下一种方式，而不是某种方式失败时使用下一种。例如当 TACACS+服务器宕机时使用 **enable password**。如果 TACACS+服务器正常工作而并且输入的用户 ID 不正确，则发送失败消息并不再尝试下一种方式。

注意例 9-38 中的认证列表只应用在线路 vty0 和 1。上述配置是以测试为目的的。即使 TACACS+配置错误，还可以通过其他 vty 线路或控制台进入路由器。在将配置应用到所有线路之前，应当确认配置正确并且按照您所期望的方式工作。当配置正确时，将认证应用到所有的线路。

你可以使用 **tacacs-server key** key 定义 TACACS+使用共享密钥加密。

你必须在服务器 TACACS+配置文件中定义相同的密钥。

你可以在路由器上进入特权(**enable**)模式时使用 TACACS+。通过下面命令得到：**aaa authentication enable default group** *auth_type* [*auth_type*]。

例 9-39 在路由器 Seattle 中增加下面配置来指定特权模式认证和 TACACS+共享密钥。

例 9-39　在路由器 Seattle 中增加特权模式认证和 TACACS+共享密钥

```
aaa new-model
aaa authentication login tac tacacs+  enable
aaa authentication enable default group  tacacs+ enable
tacacs-server host 172.16.1.2
tacacs-server key mykey
line vty 0 4
 login authentication tac
```

例 9-40 显示对应例 9-39 中路由器配置的 TACACS+服务器配置文件。

例 9-40　对例 9-39 中 Seattle 配置的 TACACS+服务器配置文件

```
Key = "mykey"
User = agnes
{
                    login = cleartext 'agnes password"
}
user = admin
{
                    login = cleartext "encrypted"
}
user = $enab15$
{
                    login = cleartext "secret"
}
```

有两个一般用户，**agnes** 和 **admin**。用户**$enab15$**用作特权级别的认证，缺省情况下特权级别 15。Enable 认证的负效应是创建了一个名为**$enab15$**的用户。如果有人知道口令的话，可以使用该用户 ID 登录路由器。

如果使用 9-39 中的配置，访问路由器需要用户名和口令。只有当 TACACS+服务器不可访问时才使用通常的路由器登录 ID。最后用户名会出现在某些日志信息中。

下列日志信息显示了用户名、用户的 IP 地址和改变配置的日期和时间：

Jun 20 16:42:32 UTC: %SYS-5-CONFIG_I: Configured from console by agnes on vty0 (10.1.2.25)

前面显示的是非常基本的配置。你可以用 TACACS+完成更多的工作，例如定义用户组、基于用户组指定特权、使用 DES 加密口令或使用 UNIX 口令文件。

> 注：　参见 TACACS+用户手册中完全的实现说明。Passwd(5)是 UNIX 口令文件支持的类型。

2．TACACS+授权配置

你可以配置 TACACS+来授权用户可以在路由器上作什么。可以应用访问列表，限制命令或允许 PPP 和 SLIP 接入。可以在 TACACS+服务器上为每个用户设置轮廓简档。在轮廓简档中指定用户被授权作什么。用户登录到路由器以后，所有的行为必须由用户简档轮廓内容授权。

下列命令定义了授权方式列表并将列表应用到路由器的线路上：

aaa authorization {**network** | **exec** | **commands** *level* | **reverse-access**} {**default** | *list-name*} [*method1* [*method2*...]]

line *type number*

authorization {**arap** | **exec** | **commands** *level* | **reverse-access**} {**default** | *list-name*}

用户简档轮廓必须在 TACACS+服务器上配置。认证配置中的警告也使用于授权配置。确认先测试授权配置，由于 **default** *list* 默认的列表会应用到所有的线路和接口，所以必须在使用默认的列表 default list 之前或者将命名的列表应用到所有线路之前配置一个不受限访问的用户。当命令键入以后，授权就应用到路由器上，所键入的内容甚至会影响当前的连接。同时 TACACS+配置文件缺省不使能授权。如果在路由器上配置 TACACS+授权并且应用到路由器和所有线路，同时没有在配置文件中指定允许任何事，你将发现无法执行任何命令。

例 9-41 中服务器配置文件限制 Agnes 执行的命令，对用户 Admin 没有限制。

例 9-41　TACACS+服务器配置文件，提供基于用户的权限

```
Key = "mykey"
User = agnes
{
            login = cleartext "agnes password"
            cmd = show {
                        permit .*
            }
}
user = admin
{
            default service = permit
            login = cleartext "encrypted"
}
user = $enab15$
{
            login = cleartext "secret"
}
```

Agnes 被允许使用任何 **show** 命令，不允许作其他任何事。

例 9-42 显示了使用例 9-41 中 TACACS+配置的路由器命令。

例 9-42　关于例 9-41 中 TACACS+服务器授权的路由器授权配置

```
aaa authorization commands 1 restrict group tacacs+
aaa authorization commands 15 restrict group tacacs+
line vty 0 1
            authorization commands 1 restrict
            authorization commands 15 restrict
! line vty 2 4
!            authorization commands 1 restrict
!            authorization commands 15 restrict
```

谨记在完全测试以前不要将授权列表应用到所有 vty 端口。

正如 TACACS+认证，你可以使用授权做许多其他事。我有 TACACS+授权特性简单应用的举例。可以使用访问列表。可以强制命令自动执行。可以应用远程登录限制。

TACACS+记账显示连接到路由器上的用户在做什么的信息。

3．TACACS+记账配置

TACACS+记账用作记录关于用户连接的信息，包括连接的时长、键入的命令和带外连接的时长及目的地。上述信息在计费和安全审计中非常重要。您可以使能记账，通过使用路由器上的 AAA 记账命令和在服务器的 TACACS+配置文件中指定记账文件名。

下面命令定义了记账类型关联在指定的列表：

aaa accounting {**system** | **network** | **exec** | **connection** | **commands** *level*} {**default** | *list-name* } {**start-stop** | **wait-start** | **stop-only** | **none** } {*method1* [*method2...*]}

line *type number*

accounting {**arap** | **exec** | **connection** | **commands** *level*} {**default** | *list-name*}

关于系统级事件的信息例如系统重启或记账配置由系统记账使能。关于 PPP，SLIP 或 ARAP 的信息包括包和字节数由网络记账提供。EXEC 记账提供关于路由器上 EXEC 终端的信息。上述信息包含用户名数据和会话的起始结束时间。连接记账提供了路由器发起的连接的相关信息。连接可以是 **telnet**，**LAT**，**tn3270**，**PAD** 或 **rlogin**。信息包括目标地址、协议、起始结束时间、用户名和传输的包和字节数。命令记账提供关于输入命令的信息。命令通常是 1 级或 15 级。你在登录后使用的命令是 1 级命令。15 级命令只能在使能后特权级才能使用。实际输入的命令也被记录，甚至记录配置命令。

使用线路子命令 **accounting** ［**type**］ *list-name* 可以将列表应用到线路。

在 TACACS+服务器配置记账，需要增加命令 **accounting file** =*filename*。如例 9-43 所示。

例 9-43　TACACS+服务器上的记账配置

```
Key = "mykey"
Accounting file = tacacs.acct
User = agnes
{
          login = cleartext "agnes password"
          cmd = show {
                    permit .*
          }
}
user = admin
{
          default service = permit
          login = cleartext "encrypted"
}
user = $enab15$
{
          login = cleartext "secret"
}
```

例 9-44 显示使能命令和 EXEC 记账的路由器命令。

例 9-44　命令的 EXEC 记账

```
aaa accounting commands 1 default stop-only group tacacs+
aaa accounting commands 15 default stop-only group tacacs+
aaa accounting exec default start-stop group tacacs+
```

由于使用默认 default 列表，在上述配置中不需要线路 vty 子命令。Default 默认列表自动应用到所有的线路和接口。

例 9-45 显示了记账日志的内容。

例 9-45　记账日志显示 Commands 和 EXEC 记录

```
ObiWan:/tacacs# more tacacs.acct
Tue Jun 20 10:33:06 2000        172.16.1.7      agnes   tty2  10.1.2.25
stop    task_id=2       start_time=961520711    timezone=UTC service=shell
  priv-lvl=15     cmd=debug aaa accounting <cr>
Tue Jun 20 10:33:57 2000        172.16.1.7      agnes   tty8  10.1.2.25
stop    task_id=4       start_time=961520761    timezone=UTC    service=shell
  priv-lvl=15     cmd=write terminal <cr>
```

```
Tue Jun 20 10:34:08 2000        172.16.1.7      agnes   tty3  10.1.2.25
stop    task_id=5       start_time=961520773    timezone=UTC    service=shell
  priv-lvl=15     cmd=configure terminal <cr>
Tue Jun 20 10:34:22 2000        172.16.1.7      agnes   tty3  10.1.2.25
stop    task_id=6       start_time=961520786    timezone=UTC    service=shell
  priv-lvl=15     cmd=interface Serial 1 <cr>
Tue Jun 20 10:34:24 2000        172.16.1.7      agnes   tty3  10.1.2.25
stop    task_id=7       start_time=961520789    timezone=UTC    service=shell
  priv-lvl=15     cmd=shutdown <cr>

Tue Jun 20 10:34:42 2000        172.16.1.7      agnes   tty3  10.1.2.25
stop    task_id=3       start_time=961520734    timezone=UTC    service=shell
  disc-cause=1    disc-cause-ext=1020     elapsed_time=73 nas-rx-speed=0
  nas-tx-speed=0
```

记账日志显示了输入的命令和一个 EXEC 会话记录。在记录中你可以看到 Agnes 在 6 月 20 日星期二的 10:34:24 关闭了串行接口 1。她在同一天的 10:34:42 终断她的 EXEC 会话。她的会话持续了 73s。

当你需要知道谁在路由器上作了什么的时候，可以将记账信息用作安全审计。你同样可以将记账用作计费。假设这些信息来自接入服务器(NAS)，用户可以通过接入服务器访问网络，上面的数据显示 Agnes 连接了 73s。你可以向她收取 73s 的网络使用费。

9.11.5 RADIUS

远程访问拨号接入用户服务(RADIUS)提供了与 TACACS+相同的功能，但是略有差别。RADIUS 是为网络拨号接入设计的认证、授权和计费服务器。RADIUS 客户位于路由器或接入服务器，与网络上的 RADIUS 服务器通信。RADIUS 与 TACACS+的区别在于 RADIUS 无法控制用户在路由器上执行的命令。如果网络管理员希望对不同的用户严格控制可使用的命令，TACACS+对接入控制是一个更好的选择。

RADIUS 是 Livingstone Enterprise 设计的。源代码公开，没有任何使用限制。由许多服务器实现了 RADIUS，许多厂商的设备都支持客户端程序。

在 Cisco 路由器上的 RADIUS 配置类似于 TACACS+的使能，通过 AAA 命令。网络上可能同时存在 RADIUS 和 TACACS+服务器，为同一台路由器或接入服务器提供认证授权和计费服务。例如，希望使用 RADIUS 服务器来认证用户。如果 RADIUS 服务器不可访问，则使用 TACACS+服务器认证。例 9-46 显示将 RADIUS 作为主认证服务器，TACACS+作为备用认证服务器的路由器配置。

例 9-46　在路由器上使能 RADIUS 和 TACACS+

```
aaa new-model
aaa authentication login remoteauth radius tacacs+  enable
tacacs-server host 172.16.1.2
radius-server host 172.16.1.2
tacacs-server key mytackey
radius-server key myradkey
line vty 0 4
 login authentication remoteauth
```

你可以用使能 TACACS+的方式同样使能 RADIUS。

9.11.6　安全的命令解释器

安全的命令解释器(SSH)允许用户与路由器建立安全的，加密的连接。连接功能类似于带内的远程登录会话。与远程登录不同，连接使加密，比远程登录提供更巨大的优越性。远程登录时客户与服务器(在这里是路由器)间的通信是明文传输，在流量路径上的任何网络分析仪都可以截取数据并解读。Telnet 模式表示如果你使用远程登录访问路由器，你在路由器上刻意加密的口令将在网络上明文传输。SSH 将连接加密，没有数据以明文形式在路由器与客户间交换。尽管 RAS 认证可以在一些客户端支持，路由器不支持对 SSH 连接的 RAS 认证。认证只对用户 ID 和口令实施。

SSH 只在 7200、75000 和 12000 系列路由器支持，并且只在 DES 或 triple DES 数据加密软件映像支持。IOS 仅支持 SSH 版本 1。

为在路由器上使能 SSH，必须实施以下步骤：

步骤 1　在路由器上配置主机名和域名。

步骤 2　产生 RAS 密钥对。

步骤 3　使能本地或 AAA 认证。

步骤 4　如果使用 AAA 认证，在控制台端口禁止 AAA 认证。

步骤 5　配置可选的 SSH 参数。

主机名和域名使用下面命令配置：

hostname *hostname*

ip domain-name *domainname*

如果你没有配置上述命令，在产生 RAS 密钥对时会报错。

在 RAS 密钥对产生以后，SSH 自动使能。当 RAS 密钥对删除以后，SSH 自动禁止。

命令 **crypto key generate rsa** 能产生 RAS 密钥对，并使能 SSH。SSH 可以在使用命令 **username** 以后用作本地认证，也可以用作 AAA 认证。AAA 必须在控制台端口上禁止。

SSH 参数能更改缺省的连接行为。你可以改变应用在 SSH 协商阶段的超时值，也可以指定认证重试的次数。超时值必须不超过 120 秒。缺省值是 120 秒。重试次数不超过 5 次，缺省值 5 次。可以使用下面命令来改变上述参数：

ip ssh {[**timeout** *seconds*] | [**authentication-retries** *integer*]}

例 9-47 示范了基本 SSH 配置。

例 9-47　SSH 配置

```
hostname Seattle
ip domain-name thecompany.com
crypto key generate rsa
aaa authentication login tacauth tacacs+ local enable
aaa authentication login aaanone none
username agnes password 0 agnespassword
ip ssh time-out 60
ip ssh authentication-retries 2
tacacs-server host 172.16.1.2
tacacs-server key secret
```

```
line con 0
 login authentication aaanone
 transport input none
line aux 0
 login authentication tacauth
line vty 0 4
 login authentication tacauth
```

注意 AAA 认证列表是 **aaanone**，没有定义 AAA 认证方式应用到控制台端口。其他认证列表 **tacauth** 被应用到所有的 vty 端口和辅助端口。

9.12　设计支持管理程序的服务器

支持管理程序的服务器应当是健壮并且安全的。管理服务器被设置，用作收集和处理维护网络完整性所需要的数据。服务器必须放置在物理上安全的地点，能连续运行不被中断。服务器的操作系统应当被保护安全，远程访问应当严格限制。谨记网管工作站能访问到网络上所有的路由器，网管工作站必须非常安全。

服务器的规模应当基于厂商的建议。使用保守的数字来确保即使所收集的数据总量增加，还会有足够的处理能力、内存和磁盘空间服务到未来。

服务器需要冗余的方式接入到网络设备。即使在路由器或链路故障情况下，服务器也应当能收集数据。管理一个看不到运行状态的网络将是对经验的考验。

9.13　网络健壮性

健壮的网络可以经受故障，保持应用平滑运行。

在路由器故障时保持通讯需要局域网上的路由器冗余。IPv4 局域网上的主机很容易因为路由器(缺省路由器)故障而无法与远端网端上的主机通信。当缺省路由器故障时，主机无法获悉，会向一个黑洞发送流量。路由器冗余协议例如虚拟路由器冗余协议(VRRP)或 Cisco 的热备份路由协议(HSRP)能解决上述问题。上述两个协议都能使多个路由器共享一个 IP 地址。然后，主机将上述共享的 IP 地址配制成缺省网关，共享使用地址。只有一个路由器的共享地址是活跃激活的。当激活的路由器故障时，备份的路由器继续收发流量。主机对故障一无所知，甚至不知道有多个路由器转发局域网段上的流量。VRRP 是基于 Cisco 的 HSRP 的开放标准。Cisco 的 IOS 软件不支持 VRRP，所以本书不讨论 VRRP。HSRP 将在下一章节进一步讨论。

注：　VRRP 由 RFC 2338 规范。

9.13.1　HSRP

HSRP 允许在一个局域网(以太网、令牌环或 FDDI)上或 ISL 封装的 VLAN 上的多个路由器共享一个 IP 地址和 MAC 地址。共享地址的一组路由器被配制成 HSRP 组。组中

每一个路由器配置一个组 IP 地址和优先级。存在一个路由器是活跃激活的，接收所有发往组 IP/MAC 地址的包。如果激活的路由器发生故障，组中另一个路由器激活并接收包。

激活的路由器是优先级最高的路由器。缺省的优先级是 100。如果多个路由器优先级相同，HSRP 接口上 IP 地址数值最大的路由器将激活。优先级其次的路由器作为备份路由器。当激活的路由器无法宣告自身的存在或者发布一个较低的优先级，则备份路由器激活。图 9-4 举例说明 HSRP。

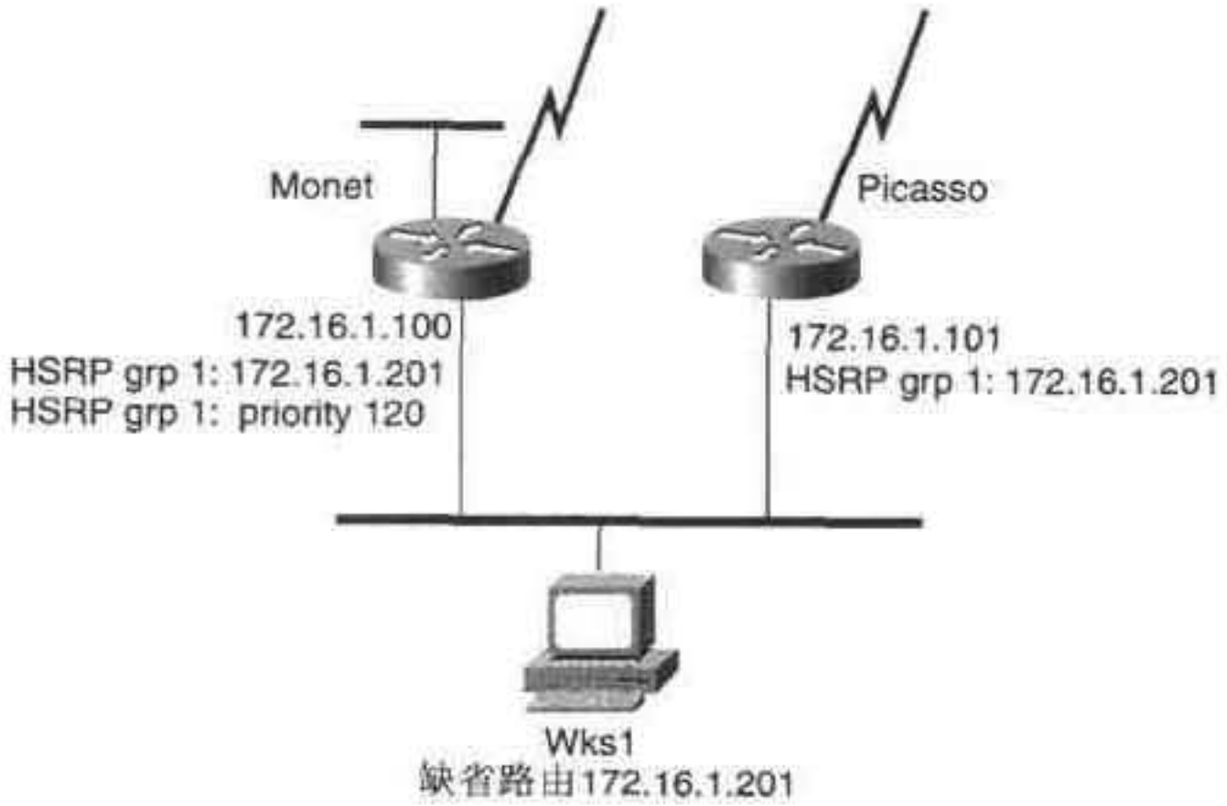


图 9-4 HSRP

路由器 Monet 配置了一个接口 IP 地址 172.16.1.100 和一个 HSRP 组 1 的 IP 地址 172.16.1.201。路由器 Monet 对 HSRP 组 1 发布优先级 120。该优先级高于 Picasso 的缺省优先级。所以 Monet 是组 1 的激活路由器。当 Wks1 需要向缺省网关发包时，发送 ARP 查找 SHRP 组 1 地址。Monet 使用 HSRP 组 1 的 MAC 地址应答。Wks1 向 SHSRP 组 1 的 MAC 地址发送包，这些包由 Monet 接收。

HSRP 组中的路由器通过交换多播 Hello 包来发布优先级。Hello 消息在配置 HSRP 组的链路上交换。缺省条件下，路由器每 3 秒发送一个 Hello 消息。如果活跃激活的路由器在一个称为保持时间(缺省条件下 10 秒)的可配置时间段内无法发送 Hello，拥有最高优先级的备份路由器激活，开始接收发往组 MAC 地址的包。

9.13.2 多组 HSRP

多组 HSRP(MHSRP)允许在一个接口上配置多个 HSRP 组。你希望将激活活跃路由器的功能分布到同一局域网上多个路由器时，可以使用 MHSRP。部分终端节点将一个组的 IP 地址作为缺省路由；其他节点将另一个组的 IP 地址作为缺省路由。任何一个缺省路由器故障时，其他路由器继续数据转发。不允许关联多个 MAC 地址的以太网接口不支持 MHSRP(使用 Lance Ethernet 硬件[1000，2500，3000 和 4000]不支持单一以太网上多个组)。以太网和 FDDI 接口支持多达 255 个 MHSRP 组。令牌环支持最多 3 个组(组 0，1，2)。MSHRP 在交换机间链路(ISL)封装上支持。图 9-5 举例说明 MHSRP。

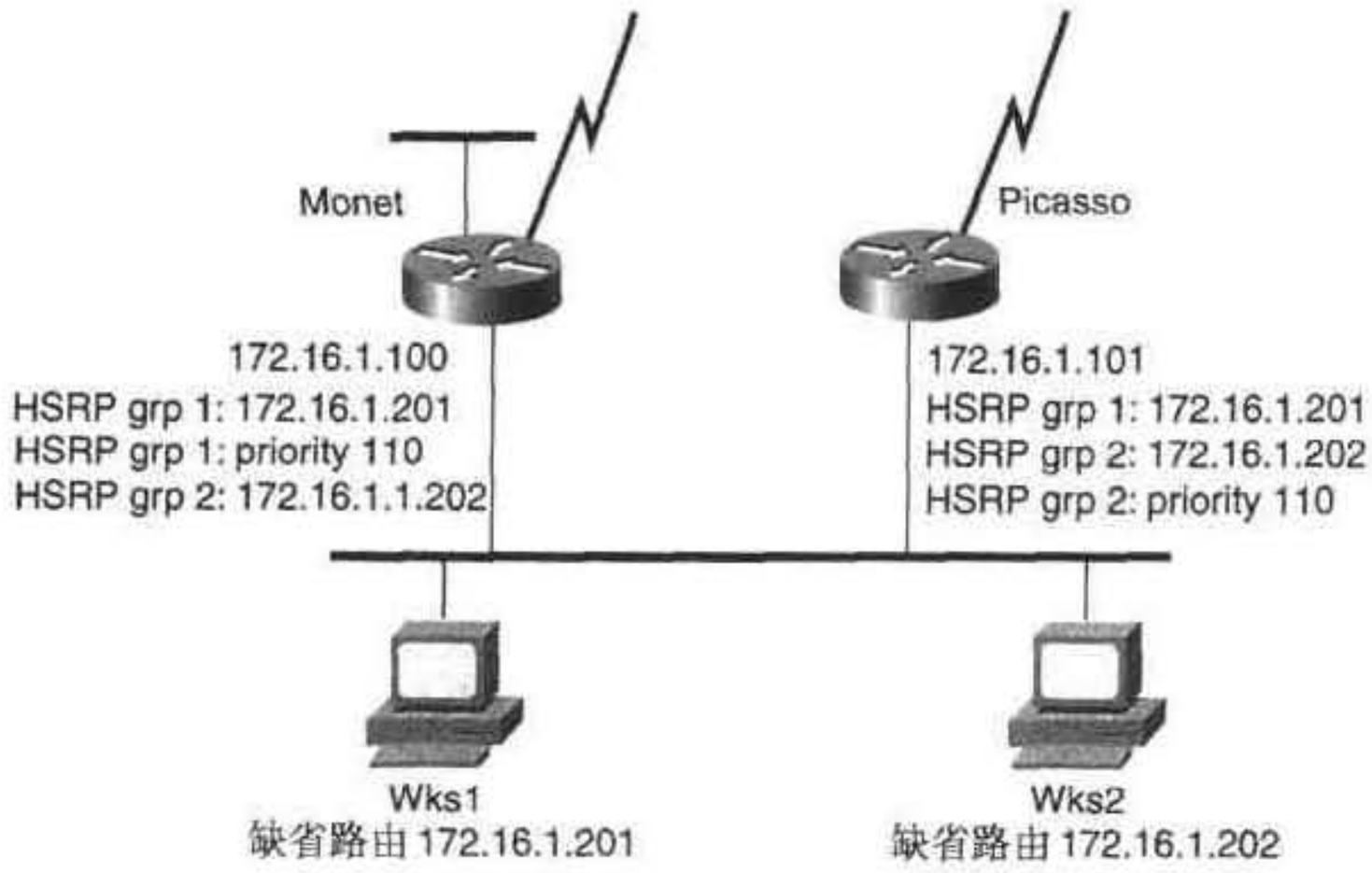


图 9-5　路由器接口上可以配置 MHSRP 组均衡流量

在图 9-5 中，Monet 是组 1 的激活路由器；Picasso 是组 2 的激活路由器。Wks1 缺省路由到 172.16.1.201，组 1；Wks2 缺省路由到 172.16.1.202，组 2。如果 Monet 无法收到来自组 2 的 HSRP Hello 消息，Monet 将为在组 1 激活路由器之外成为组 2 的激活路由器。

9.13.3　配置 HSRP

输入下列接口子命令使能 HSRP：

standby [group-number] **ip** [ip-address [secondary]]

你必须至少在 HSRP 组的一个路由器上指定 IP 地址。如果你不在路由器上指定 IP 地址，该地址将通过 HSRP Hello 消息学习到。

下列命令影响路由器在 HSRP 中的角色：

standby [*group-number*] **timers** *hellotime holdtime*

standby [*group-number*] **priority** priority [**preempt** [**delay** *delay*]]

standby [*group-number*] [**priority** priority] **preempt** [**delay** *delay*]

standby [*group-number*] **track** *type number* [*interface-priority*]

standby use-bia [**scope interface**]

timers 命令改变 hello 包时间间隔和备份路由器判定激活路由器故障需要的时间。缺省的 hello time 是 3 秒，缺省 hold time 是 10 秒。

priority 和 **preempt** 命令改变 HSRP 路由器的优先级。即使当前的激活路由器没有故障，**preempt** 使最高优先级的路由器成为激活路由器。**delay** 选项是因 **preempt** 选项成为激活路由器前的延时。延时范围 0-3600 秒。缺省 0 秒。

有时路由器的局域网接口没有问题，路由器本身也没有问题，但是路由器转发包的对外接口出现故障。在这种情况下转发到路由器的包必须重定向到另一个路由器，如图 9-6 所示。

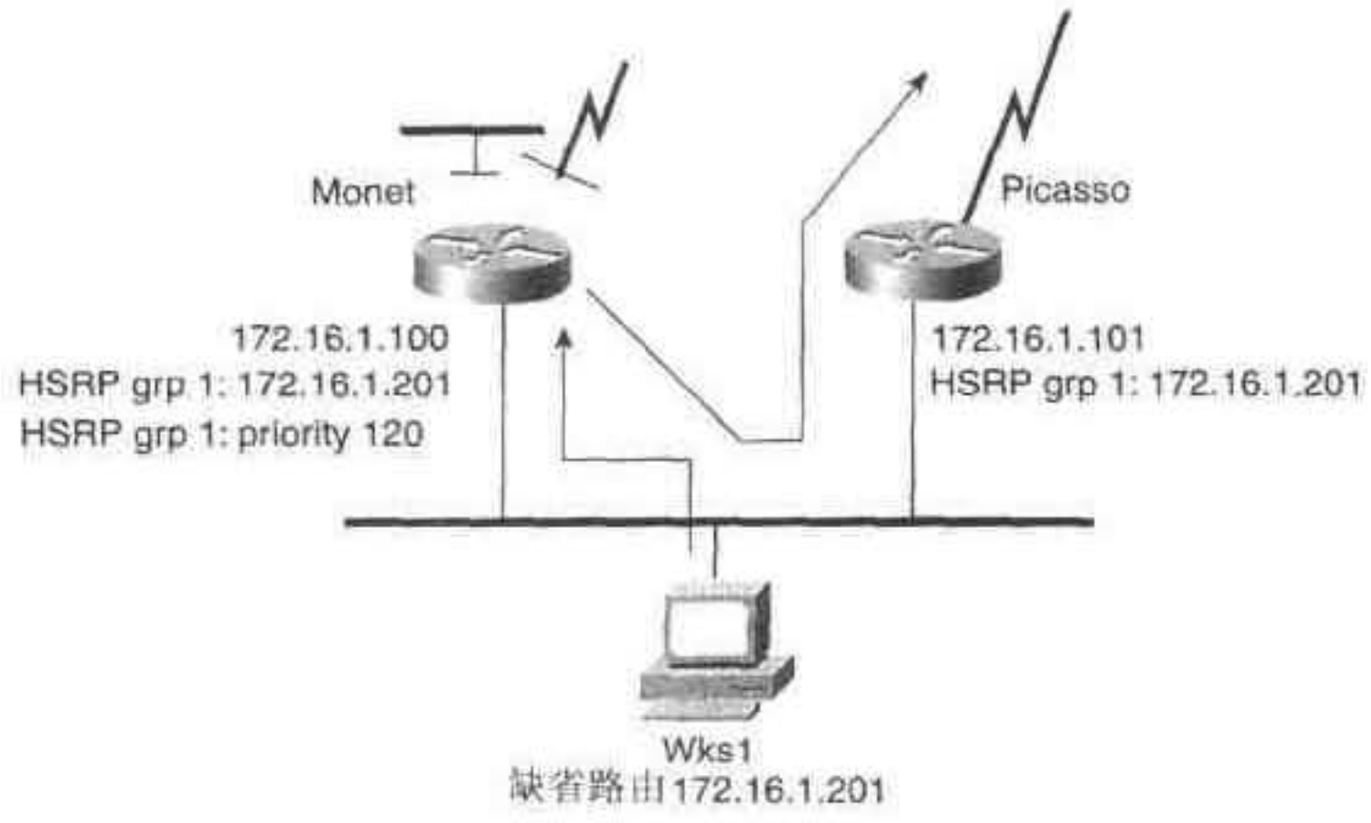


图 9-6　没有接口追踪监视 track 的 HSRP

工作站向缺省网关发包，即激活路由器 Monet。Monet 的两个对外接口都发生故障。Monet 参考自身路由表，将包从以太网发往 Picasso 作进一步转发。

track 命令能使 HSRP 追踪对外接口的状态，当对外接口故障时降低优先级并退出活跃激活状态。当被追踪的接口发生故障时，路由器改变发布的优先级。如果新的优先级比备分路由器的优先级低并且备份路由器配置了抢占较低优先级的激活状态，则备份路由器成为该组的激活路由器。故障路由器的优先级将减少 *interface-priority* 所指定的数量。缺省值是 10。可以追踪多个接口。如果多个接口被追踪且都配置了 *interface-priority* 参数，当多个接口故障使所降低的优先级数将累计。如果在被追踪接口上没有配置 *interface-priority* 值并且多个接口故障，优先级只降低缺省的 10 并不累计。

authentication 命令允许路由器在 HSRP 消息中包含认证串。你必须将组内所有路由器配置相同的认证串或者都不配置。第一个路由器使能 HSRP 成为活跃激活路由器。如果新激活的路由器的认证串与激活路由器不匹配，则新激活得路由器保持在学习状态。没有路由器成为备份路由器。

例 9-48 显示了图 9-7(单组 HSRP)中路由器 Monet 与 Picasso 的 HSRP 配置。

例 9-48　图 9-7 中路由器 Monet 与 Picasso 的 HSRP 配置

Router Monet

```
interface Ethernet 1
 ip address 172.16.1.100 255.255.255.0
 standby 1 priority 120 preempt delay 10
 standby 1 authentication secret
 standby 1 ip 172.16.1.201
```

Router Picasso

```
interface Ethernet 0
 ip address 172.16.1.101 255.255.255.0
 standby 1 authentication secret
 standby 1 ip
```

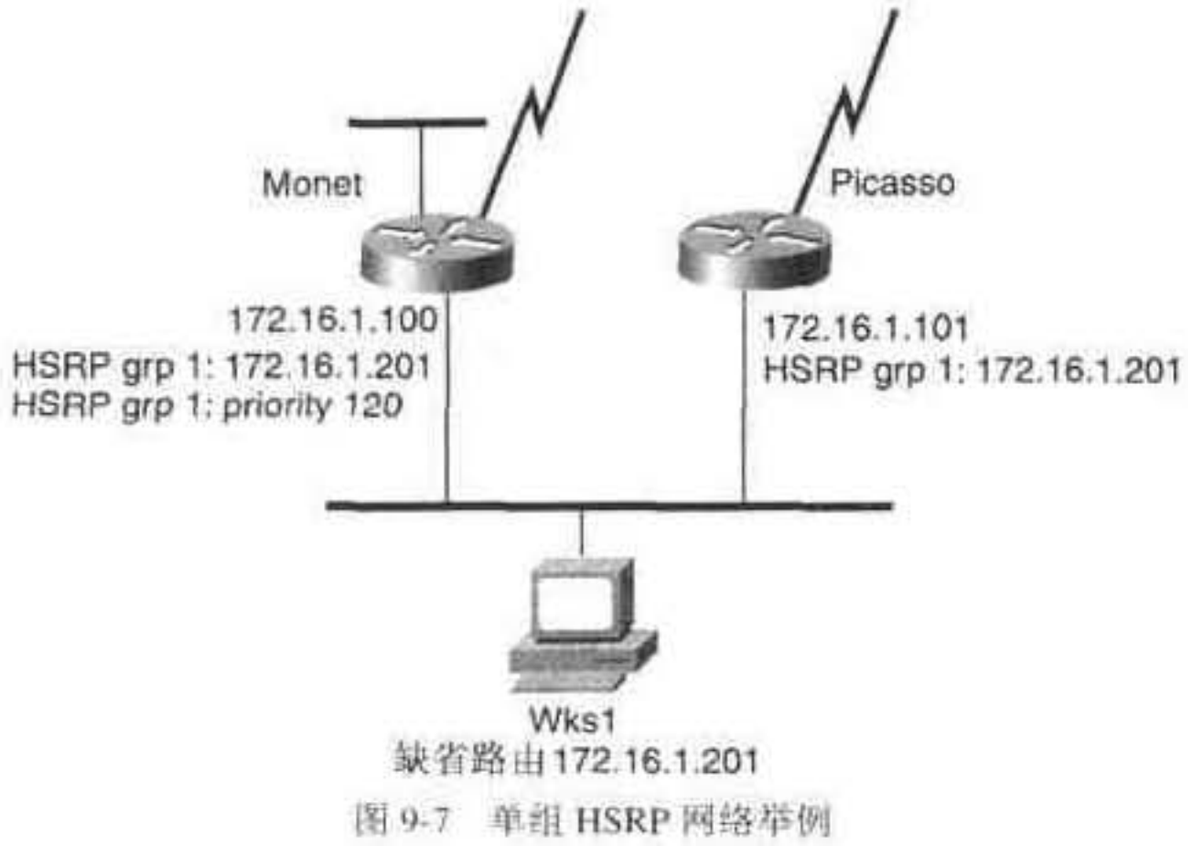


图 9-7　单组 HSRP 网络举例

Picasso 从 HSRP 功能得到 IP 地址和定时器。

在例 9-49 中 Picasso 路由器 **show standby** 命令的输出显示学习到的信息。可以看到 Picasso 从 HSRP 功能学到的 IP 地址和定时器信息。

注意 Picasso 的状态是备份，优先级 100。激活路由器的地址是 Monet：172.16.1.100。HSRP 地址是 172.16.1.201。关联在 HSRP 地址上的 MAC 地址是 0000.0c07.ac01。

例 9-49　show standby 命令的输出

```
Picasso#show standby
Ethernet0 - Group 1
  Local state is Standby, priority 100
  Hellotime 3 holdtime 10
  Next hello sent in 00:00:00.340
  Hot standby IP address is 172.16.1.201
  Active router is 172.16.1.100 expires in 00:00:10
  Standby router is local
  Standby virtual mac address is 0000.0c07.ac01
  4 state changes, last state change 00:02:57
```

在例 9-50 中从工作站向 HSRP 地址的 ping 命令显示了激活路由器故障，由备份路由器恢复。

例 9-50　向 HSRP 地址的 ping 命令显示了激活路由器故障，由备份路由器恢复

```
ObiWan:-# ping 172.16.1.201
PING 172.16.1.201 (172.16.1.201): 56 data bytes
64 bytes from 172.16.1.201: icmp_seq=0 ttl=255 time=5.7 ms
64 bytes from 172.16.1.201: icmp_seq=1 ttl=255 time=3.5 ms
64 bytes from 172.16.1.201: icmp_seq=2 ttl=255 time=3.5 ms
64 bytes from 172.16.1.201: icmp_seq=3 ttl=255 time=3.5 ms
64 bytes from 172.16.1.201: icmp_seq=4 ttl=255 time=3.5 ms
64 bytes from 172.16.1.201: icmp_seq=5 ttl=255 time=3.4 ms
64 bytes from 172.16.1.201: icmp_seq=6 ttl=255 time=3.5 ms
64 bytes from 172.16.1.201: icmp_seq=17 ttl=255 time=3.5 ms
64 bytes from 172.16.1.201: icmp_seq=18 ttl=255 time=3.5 ms
64 bytes from 172.16.1.201: icmp_seq=19 ttl=255 time=3.5 ms
64 bytes from 172.16.1.201: icmp_seq=20 ttl=255 time=3.4 ms
64 bytes from 172.16.1.201: icmp_seq=21 ttl=255 time=3.4 ms
```

工作站每秒发一个 ping 包。包 1~6 成功返回。包 7~16 没有响应。正如您所见，备份路由器需要 10~11 秒才能开始从 HSRP MAC 地址接收包。当激活路由器的局域网接口故障以后，备份路由器无法收到 hello 消息。备份路由器等待保持时间 10 秒以后开始接收包。

如例 9-51 所示，在 Monet 的配置之上附加使能 HSRP 接口追踪。图 9-8 显示了 HSRP 接口追踪带来的好处。Monet 的串口 1 和以太网 0 连接到远程资源，该资源同时可以通过 Picasso 访问。设计的目的是当 Monet 多个对外端口(串口 1 或以太网 0)up 时，允许在 LAN 上工作站将缺省路由设到 Monet。当 Monet 两个对外端口都故障时，工作站缺省使用 Picasso。

例 9-51 在路由器 Monet 上使能 HSRP 接口追踪

Monet

```
interface Ethernet1
 standby 1 track Ethernet0 15
 standby 1 track Serial1 15
```

Picasso

```
interface Ethernet0
 standby 1 priority 100 preempt delay 10
```

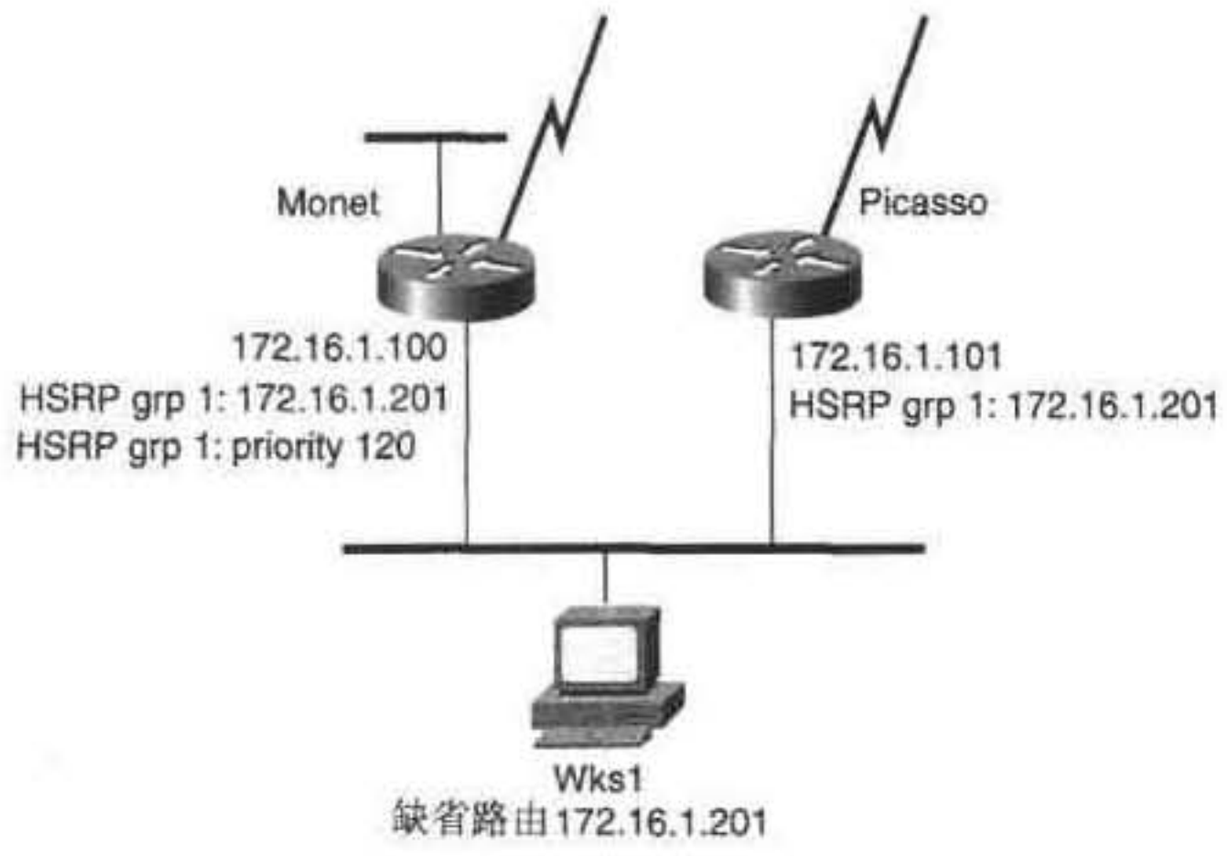


图 9-8 HSRP 接口追踪网络举例

注意 Monet 同时追踪串口 1 和以太网接口 0。当其中一个端口 down 时，Monet 优先级为 105，仍然高于 Picasso 的优先级，所以 Monet 保持激活状态。如果两个端口都故障，Monet 的优先级从 120 降低到 90。当等待 **preempt** 时延以后，Picasso 发送 HSRP coup 消息，通知 Monet 它将成为激活路由器。Monet 退出激活状态，通过倾听其他路由器的 HSRP 消息来决定是否成为备份路由器。你必须在 Picasso 路由器配置 **preempt** 命令才能使 Picasso 夺取激活状态。当 Monet 的一个接口恢复以后，优先级上升为 105。由于 Monet 有 **preempt** 和 **delay** 配置，10 秒以后 Monet 成为组 1 的激活路由器。

9.13.4 配置 MHSRP

图 9-9 举例说明多组 HSRP。

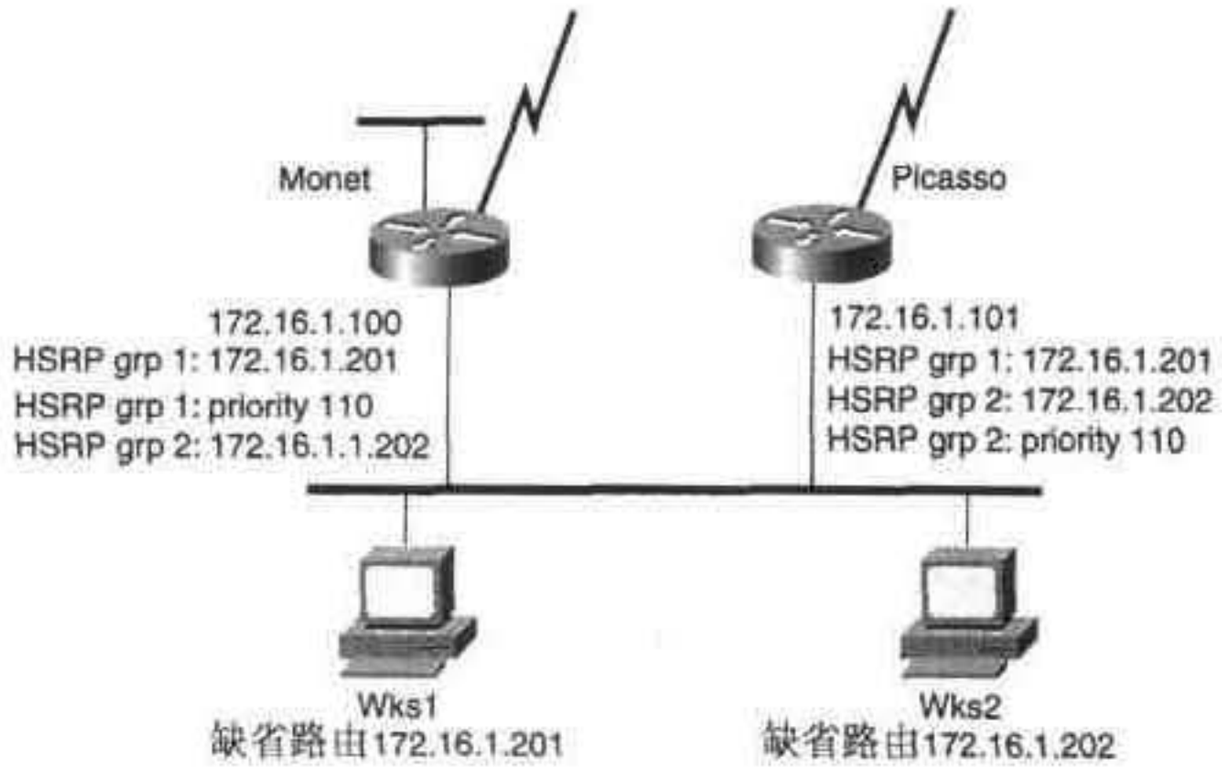


图 9-9　MSHRP 网络示意

例 9-52 中是路由器 Monet 和 Picasso 的 MSHRP 配置。

例 9-52　路由器 Monet 和 Picasso 的 MSHRP 配置

```
Router Monet
interface Ethernet 1
 ip address 172.16.1.100 255.255.255.0
 standby 1 priority 120 preempt delay 10
 standby 1 authentication secret
 standby 1 ip 172.16.1.201
 standby 2 authentication secret
 standby 2 ip
```

```
Router Picasso
interface Ethernet 0
 ip address 172.16.1.101 255.255.255.0
 standby 1 authentication secret
 standby 1 ip
 standby 2 priority 120 preempt delay 10
 standby 2 authentication secret
 standby 2 ip 172.16.1.202
```

Monet 是组 1 的活跃激活路由器，HSRP IP 地址是 172.16.1.201；Picasso 是组 2 的活跃激活路由器，HSRP IP 地址是 172.16.1.202。为实现负载均衡，将局域网上一半工作站的缺省网关配置成 172.16.1.201，另一半的缺省网关配制成 172.16.1.202。

9.14　实验室

网络实验室能提供一个测试新配置，IOS 版本和特性的平台。

由于实验室的目的是预先测试任何实现在实际网上的新东西，所以实验室的构造应当反映实际网络。一个有效的实验室不需要是实际网络的完全复制，但需要包含与实际网络相同类型的路由器、界面和 Cisco IOS 软件。在实验室中应当运行相同的路由协议和路由特性。任何在实际网络的实现都可以复制到实验室。

实验室应当与生产产业网隔离，但是要求需要使用的人能方便地访问。一种将实验室隔离于生产产业网又允许从网络接入实验室的方法是使用终端服务器。终端服务器的局域网接口连接到生产网。终端服务器的异步接口连接到实验室路由器的控制台端口。多数终端服务器允许对异步端口所连接设备反向远程登录。每一个异步端口和协议端口数相关联。如果你远程登录终端服务器的 IP 地址并指定所需异步端口的正确协议端口，你可以连接到该端口所连的异步设备。图 9-10 示例的使用终端服务器连接生产网和实验室。

图 9-10 的信息显示从生产网远程登录 172.16.1.254 端口 2001，终端服务器能将你连接到异步端口 1 所连的设备，路由器 R1。

终端服务器上的配置条目将主机 R1 关联到 IP 地址与端口号 172.16.1.254 2001。从终端服务器发起的到 R1 的会话通过串行端口 1 连接到路由器 R1。

实验室用作测试网络设计和网络将出现的变动，包括配置变化、路由器增加、IOS 升级和新特性增加。实验室测试是改变规则一个完整的组成部分。成功的测试能保证网络变化的成功，不出现计划外的负面影响。测试假设由网络工程师来完成商业单位预期的努力，尽所有努力保持网络优化运行。

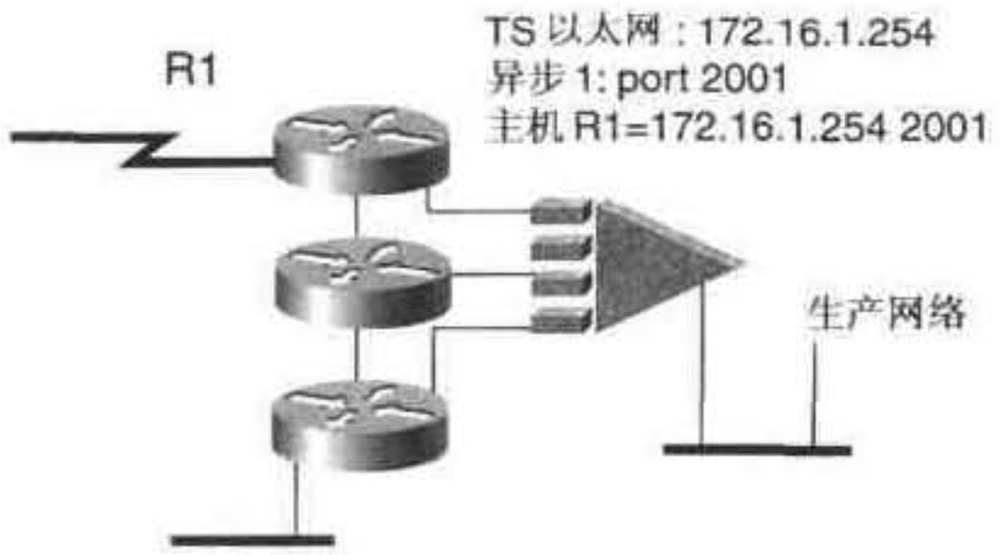


图 9-10 从生产网络使用终端服务器接入隔离的实验室

彻底测试新的设计、增加的特性和 Cisco IOS 版本升级非常重要。上述 3 点是网络升级的主要内容。

如果没有一个优秀的测试计划，测试可能无法按照预期完成，结果也可能无效。测试计划描述待测项目并定义如何测试。在测试开始以前写一个清晰的测试计划能节约您的时间。你应当明确需要测什么并定义完成测试需要的步骤。对每个步骤都应当精确定义。事实上计划应当定义的如此清楚，以至于任何人都可以按照步骤测试，任何两个人如果实施计划的同一步骤都应当得到相同结果。

实验室同时还提供一个环境，在那里您可以随便试用命令、尝试错误配置的结果以及练习故障查找排除。实验室可以用作培训或 CCIE 练习。只有在实验室里你才能彻底练习配置、搞错配置看会发生什么以及确定什么样的现象确定配置错误。CCIE 实验室考试所需要的知识深度要求您对 Cisco IOS 软件有这样类型的练习。

9.15 推荐书目

Marshal T.Rose，一本简单的书：An Introduction to Networking Management：第二次修订

版(NewYork, NY: Simon & Schuster Trade, 1995).该书详尽解释了 SNMP。

Randal L. Schwartz, Tom Christiansen, Steve Talbot (编辑), Learning Perl, 第二版(O'Reilly & Associate, Inc., July 1997).

Larry Wall, Jon Orwant, Tom Christiansen, Programming PERL(O'Reilly & Associates, Inc., July 2000).

9.16　尾注

[1]J. Case et al., "RFC 1157: A Simple Network Management Protocol (SNMP)" (工作正进行中)

[2]J. Case et al., "RFC 1901: Introduction to Community-based SNMPv2" (工作正进行中)

[3]K. McCloghrie 与 M. Rose, "RFC 1213: Management Information Base for Network Management of TCP/IP-based internets: MIB-II" (工作正进行中)

[4]S. Waldbusser, "RFC 2819: Remote Network Monitoring Management Information Base" (工作正进行中)

9.17　展望

本章节深入总结了 TCP/IP 选路中选路协议之外用于域间选路的其他技术。您必须在 CCIE 考试之前彻底理解这些内容。然而 TCP/IP 选路并不是考试中唯一的主题。您需要学习其他协议例如 SNA、IPX 和 AppleTalk。如果你还没有做这些，你还需要学习 LAN 和 WAN 交换。

9.18　命令归纳

表 9-6 提供了一个命令列表，该列表包含本章节内讨论过的所有命令的描述。

表 9-6　　命 令 汇 总

命　　令	描　　述
snmp-server community *community-string*[**view** *view-name*] [ro|rw]*access-list number*]	定义团体字串，使用该字串能看到的视图，该字串的访问类型(ro 或 rw)和关联的访问列表来指定允许使用该字串的设备
snmp-server view *view-name oid-tree* {**indluded| excluded**}	限制 SNMP 管理器能访问哪一个 MIB 对象
snmp-server system-shutdown	允许 SNMP 管理器通过 SNMP 向路由器上的登录用户发送消息，然后重新启动路由器
snmp-server tftp-server-list *access-list_number*	将能通过 SNMP 下载配置文件的 TFTP 服务器限制在指定的访问列表

续表

命　令	描　述
snmp-server host *host* [**version** {1|2c}] *community-string* [**udp-port** *port*] [*trap-type*]	执行发送 traps 的目标主机
snmp-server enable traps *trap-type trap-option*	允许发送列出的 trap 类型
snmp trap link-status	使能接口上的 up/down traps
show snmp	显示 SNMP 统计
rmon alarm *number variable interval* {**deltalabsolute**} **rising-threshold** *value* [*event-number*]**falling-threshold** *value* [*event-number*] [**owner** *string*]	定义一个告警，指定告警什么时候触发和清除以及告警触发什么事件
rmon event *number* [**log**] [**trap** *community*] [**description** *string*] [**owner** *string*]	定义一个 RMON 事件，指定被告警触发时将事件记录到哪里
show rmon alarms	显示所定义告警的相关信息
show rmon events	显示事件表
logging buffered [*size*]	在路由器中使能缓存日志，并指定日志大小
show logging	显示缓存日志
clear logging	清除缓存日志
logging *host*	指定接收 syslog 消息的主机的名字或地址
terminal monitor	向当前终端线路发送日志消息
service timestamps log uptime	在日志中增加时戳
service timestamps log datetime [**msec**] [**localtime**] [**show-timezone**]	在日志中增加时戳
logging console *level*	限制发送到控制台的日志消息
logging monitor *level*	限制发送到终端线路的日志消息
logging trap *level*	限制发送到 syslog 服务器的日志消息
logging facility *facility-type*	定义将日志消息发到 syslog 服务器时使用的设备类型
snmp-server enable traps syslog	对 syslog 消息使能 SNMP traps
logging history *leverl*	指定通过 SNMP 发送的 syslog 消息的级别
ntp server *ip_address* [**version** *number*] [**key** *keyid*] [**source** *interface*] [**perfer**]	创建服务器关联，路由器可以从另一个 NTP 时钟源获得同步
nep peer *ip_address* [**version** *number*] [**key** *keyid*] [**source** *interface*] [**perfer**]	创建一个对等关联，路由器可以从其他设备时钟获得同步，其他设备可以从路由器获得同步

续表

命　令	描　述
ntp access-group {**query-only**|**serve-only**|**server**|**peer**} *access-list-number*	对路由器 NTP 服务的访问控制
clock calendar-valid	将路由器的日历作为授权的时钟源
ntp master [*stratum*]	将 IOS 配制成其他设备获取同步的 NTP 主时钟
ntp update-calendar	使用 NTP 得到的时间/日期更新路由器的日历
ntp authenticate	全局使能 NTP 认证
ntp authenticate-key *number* **md5** *key*	定义 NTP 认证口令
nep trusted-key *number*	列出口令的号，该口令由 **ntp authentication-key** 命令定义必须包含在该服务器与其他路由器同步之前的 NTP 包中
ip accounting	在接口上使能 IP 记账
ip accounting-threshold *threshold*	设置可以存储在记账表中最多条目数
show ip accounting [checkpoint] [access-violations]	显示 IP 记账数据
chear ip accounting	清除 IP 记账数据
ip route-cache flow	在接口上使能 NetFlow
ip flow-export destination *ip-address udp-port*	指定接收 NetFlow 数据主机的 IP 地址和 UDP 端口号
ip flow-export [version 1|version 5 [origin-as|peer-as]]	指定向流收集器发送的 NetFlow 的版本以及发送哪一个 AS 号，流量的源 AS 还是路由器对端的 AS
show ip flow export	显示关于数据输出的信息
show ip cache flow	显示将要输出的数据
ip cef	全局使能 CEF，在所有支持 CEF 的接口生效
ip flow-aggregation cache {*as|destination-prefix|prefix| protocol-port|source-prefix*}	定义 NetFlow 缓存汇聚
cache entries *number_of_entries*	指定汇聚缓存中最多条目数
chahe timeout inactive *seconds*	指定在汇聚缓存中不活跃激活条目的超时值
cache timeout active *minutes*	改变汇聚缓存活跃激活条目保持活跃激活的分钟数
export destination *ip_address udp_port*	指定汇聚缓存的输出目的地
enabled	使能汇聚缓存
show ip cache flow aggregation as	显示 AS 缓存数据
show ip cache flow aggregation destination-prefix	显示目标前缀缓存数据

续表

命　　令	描　　述
show ip cache flow aggregation source-prefix	显示源地址前缀缓存数据
show ip cache flow aggregation protocol-port	显示协议端口缓存数据
enable-secret *password*	定义 enable 层口令
service password-encryption	显示配置时对口令加密
access-class *access-list_1-199_or_1300-2699* **[in|out]**	在允许出入终端会话以前指定访问列表
transport input telnet ssh	限制建立终端会话的协议
transport input none	在被配置线路上禁止所有协议
exec-timeout	定义交互终端会话的超时值
service tcp-keepalives-in	在到来的连接上使能 TCP Keepalive 消息
no ip directed-broadcast	在接口上禁止 IP 直接广播
no ip source-route	全局禁止转发携带源地址路由消息的包
scheduler interval *milliseconds*	配置路由器停止处理中断，处理其他事务的间隔
scheduler allocate *interrupt-time process-time*	定义路由器在任何网络中断上下文处理快速交换的最长时间以及禁止网络中断路由器处理其他事务的最短时间
no service tcp-small-servers	禁止 TCP 小服务器
no service udp-small-servers	禁止 UDP 小服务器
no service finger	禁止 Finger 服务器
no ip bootp server	禁止 BOOTP 服务器
aaa new-model	使能 AAA
aaa authentication login {default|*list_name***} group** *auth type* [*auth type*...]	定义 AAA 认证方式列表
login authentication *list_name*	指定认证连接用户时使用哪一个定义的 AAA 认证方式列表
tacacs-server host *ip address*	指定 TACACS 服务器
radius-server host *ip_address*	指定 RADIUS 服务器
tacacs-server key *key*	在路由器与 TACACS 服务器间定义共享密钥
radius-server key *key*	在路由器与 RADIUS 服务器间定义共享密钥
aaa authentication enable default group *auth type* [*auth_type*]	定义 enable 级访问的认证类型
aaaauthorization {network|execlcommands*level***|reverse-access}** **{default|***list-name***}** [*method1* [*method2*...]]	定义 AAA 授权方式列表

续表

命　令	描　述
authorization {**arap**|**exec**|**commands** *level*|**reverse-access**} {**default**|*list-name*}	指定对连接的用户使用哪一个定义的 AAA 授权列表
aaa accounting {**system**|**network**|**exec**| **connection**|**commands** *level*} {**default**| *list-name*} {**start-stop**|**wait-start**|**stop-only**| **none**} [*method1*[*method2*…]]	定义一个 AAA 记账方式列表，并定义对什么信息记账
accounting {**arap**|**exec**|**connection**| **commands** *level*} {**default**|*list-name*}	指定对连接的用户会话使用哪一个 AAA 记账方式列表
hostname *hostname*	定义路由器主机名，该名字在产生 RAS 加密时需要
ip domain-name *domainname*	定义路由器域名，该域名在产生 RAS 加密时需要
crypto key generate　rsa	产生 RAS 密钥(并使能 SSH)
ip ssh {[**timeout** *seconds*]|[**authentication-retries** *integer*]}	改变 SSH 参数
standby [*group-number*]**ip** [*ip-address* [**secondary**]	为指定的热备份组定义 HSRP 地址
standby {*group-number*] **timers** *hellotime holdtime*	改变指定热备份组的定时器
standby {*group-number*} **priority** *priority* {**preempt** {**delay** *delay*}}	改变路由器在指定热备份组的 HSRP 优先级。并指定优先权特性
standby [*group-number*] [**priority** *priority* | **preempt** [**delay** *delay*]	指定优先权特性。并改变路由器在指定热备份组中的 HSRP 优先级
standby [*group-number*] **stack** *type number* [*interface-priority*]	标识热备份的追踪接口
standby [*group-number*] **authentication** *string*	定义热备份的认证串
standby use-bia [**scope interface**]	指定内建的 MAC 地址作为热备份地址
show standby	显示当前的 HSRP 特性

9.19　复习问题

1. 解释 SNMP 轮询与 traps 间的不同。

2. 如果希望将 severity 级消息记录日志成为 error，哪些其他级别的消息会被记录？

3. 如果您浏览路由器接口，发现异常的流量模式。一般情况下流量是带内的，现在有许多带外流量。您如何最快地确定流量的源和目的地址？

9.20 配置练习

1. 配置一个路由器，只接受 172.16.1.2 和 172.16.1.3 管理工作站的轮询。不允许工作站的写操作。只允许工作站 SNMP MIBII 接口条目读操作。允许 172.16.1.4 工作站读所有的 MIB 变量，并允许使用 SNMP 上载下载配置。通过 SNMP 发送 Notification 级的日志信息到 172.16.1.4。

2. 配置路由器在 5 分钟平均 CPU 利用率超过 90%时发送 SNMP trap 到 172.16.1.4。当路由器 CPU 利用率在 60 秒间隔内从低于 85%到高于 90%时发送 trap。

3. 配置路由器使用 NTP 从路由器 172.16.100.100 获取时钟信息更新内部的时间和日期。不允许其他路由器从你所配置的路由器获取时钟信息。

4. 配置 NetFlow 汇集缓存，基于源地址和目的地址前缀对数据分组。在数据中包含 peer-AS，将数据输出到 172.16.1.4。

5. 配置两个路由器在以太网段上相互备份。路由器 A 为主用，A 故障时 B 接替。A 恢复以后重新成为主用路由器。路由器 A 有两条串行链路，串口 0 和串口 1，可以将流量转发到不同的目的地。如果两个链路都故障，路由器 B 接替成为主用路由器。

9.21 参考文献

Monitoring the Router and Network. Cisco Systems – Documentation. [Web page] June. 1998; www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/fun_c/fcprt4/fcmonitr.htm [Accessed 24 Oct. 2000]

Troubleshooting the Router. Cisco Systems – Documentation. [Web page] October, 2000; www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/fun_c/fcprt4/fctroubl.htm [Accessed 24 Oct. 2000]

Synchronizing Clocks with the NTP Service. Cisco Systems – Documentation [Web page] December 1997; www.cisco.com/univercd/cc/td/doc/product/iaabu/cddm/cddm111/adguide/ntp.htm

Performing Basic System Management. Cisco Systems – Documentation [Web page] August 2000; www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/fun_c/fcprt3/fcd303.htm#34418 [Accessed 24 Oct. 2000]

Configuring NetFlow Switching. Cisco Systems – Documentation [Web page] July 2000; www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/switch_c/xcprt3/xcdnfc.htm#4727 [Accessed 24 Oct. 2000]

Fault Management. Cisco Systems – Documentation [Web page] December 1997; www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/cwparent/cw32/cw32ug/faultmgt.htm [Accessed 24 Oct. 2000]

Performance Management. Cisco Systems – Documentation [Web page] December 1997; www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/cwparent/cw32/cw32ug/perfmgmt.htm [Accessed 24 Oct. 2000]

Improving Security on Cisco Routers. Cisco Systems – Documentation [Web page] August 2000; www.cisco.com/warp/public/707/21.html, [Accessed 24 Oct. 2000]

Secure Shell Version 1 Support. Cisco Systems – Documentation [Web page] www.cisco.com/univercd/cc/td/doc/product/software/ios121/121newft/121t/121t1/sshv1.htm [Accessed 24 Oct. 2000]

Configuring TACACS+. Cisco Systems – Documentation [Web page] July 2000; www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/secur_c/scprt2/scdtplus.htm#4586 [Accessed 24 Oct. 2000]

TACACS+ and RADIUS Comparison. Cisco Systems – Technical Tips [Web page] March 1999; www.cisco.com/warp/public/480/10.html [Accessed 24 Oct. 2000]

Configuring Authentication. Cisco Systems – Documentation [Web page] October 2000; www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/secur_c/scprt1/scathen.htm [Accessed 24 Oct. 2000]

IP Services Commands. Cisco Systems – Documentation [Web page] September 2000; www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/ip_r/iprprt1/1rdip.htm [Accessed 24 Oct. 2000]

Using HSRP for Fault-Tolerant IP Routing. Cisco Systems – Documentation [Web page] February 2000; www.cisco.com/univercd/cc/td/doc/cisintwk/ics/cs009.htm [Accessed 24 Oct. 2000]

Configuring IP Services. Cisco Systems – Documentation [Web page] September 2000; www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/ip_c/ipcprt1/1cdip.htm [Accessed 24 Oct. 2000]

NetFlow Aggregation. Cisco Systems – Documentation [Web page] August 1999; www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/120t3/netflow.htm [Accessed 24 Oct. 2000]

Cisco Network Monitoring and Event Correlation Guidelines. Cisco Systems – Documentation [Web page] July 2000; www.cisco.com/warp/public/cc/pd/wr2k/tech/cnm_rg.htm [Accessed 24 Oct. 2000]

Guidelines for Polling MIB Variables. Cisco Systems – Documentation [Web page] September 2000; www.cisco.com/warp/public/477/Gen_NMS/38.html

第三部分

附　　录

附录 A　show ip bgp neighbors 的显示

本附录详细解释由 **show ip bgp neighbores** 命令返回的大量信息。有些信息是不言自喻的，大多数信息已在本书不同章节讨论了。

例 A-1 显示了典型的邻居输出，加入行号是为了便于说明。表 A-1 每一行分析了 A-1 的每一行输出。

例 A-1　show ip bgp neighbores 命令的典型输出

```
TeddyBear#show ip bgp neighbors 10.100.1.1
1. BGP neighbor is 10.100.1.1,  remote AS 6500, internal link
2.  Member of peer-group Pooh for session parameters
3.   BGP version 4, remote router ID 10.100.1.1
4.   BGP state = Established, up for 00:04:06
5.   Last read 00:00:07, hold time is 180, keepalive interval is 60 seconds
6.   Neighbor capabilities:
7.     Route refresh: advertised and received
8.     Address family IPv4 Unicast: advertised and received
9.     Address family IPv4 Multicast: advertised and received
10.   Received 7 messages, 0 notifications, 0 in queue
11.   Sent 7 messages, 0 notifications, 0 in queue
12.   Route refresh request: received 0, sent 0
13.   Minimum time between advertisement runs is 5 seconds

14.  For address family: IPv4 Unicast
15.   BGP table version 1, neighbor version 1
16.   Index 1, Offset 0, Mask 0x2
17.   Pooh peer-group member
18.   0 accepted prefixes consume 0 bytes
19.   Prefix advertised 0, suppressed 0, withdrawn 0

20.  For address family: IPv4 Multicast
21.   BGP table version 1, neighbor version 1
22.   Index 1, Offset 0, Mask 0x2
23.   0 accepted prefixes consume 0 bytes
24.   Prefix advertised 0, suppressed 0, withdrawn 0

25.   Connections established 1; dropped 0
26.   Last reset 00:04:17, due to Address family activated
27. Connection state is ESTAB, I/O status: 1, unread input bytes: 0
28. Local host: 10.100.1.2, Local port: 11012
29. Foreign host: 10.100.1.1, Foreign port: 179

30. Enqueued packets for retransmit: 0, input: 0  mis-ordered: 0 (0 bytes)

31. Event Timers (current time is 0x3ABFA00):
32. Timer          Starts    Wakeups            Next
33. Retrans             8          0             0x0
34. TimeWait            0          0             0x0
35. AckHold             7          5             0x0
36. SendWnd             0          0             0x0
37. KeepAlive           0          0             0x0
38. GiveUp              0          0             0x0
39. PmtuAger            0          0             0x0
40. DeadWait            0          0             0x0
```

```
41. iss: 2227710177  snduna: 2227710341  sndnxt: 2227710341     sndwnd:  16221
42. irs: 1632859231  rcvnxt: 1632859395  rcvwnd:       16221  delrcvwnd:    163

43. SRTT: 540 ms, RTTO: 3809 ms, RTV: 1364 ms, KRTT: 0 ms
44. minRTT: 8 ms, maxRTT: 300 ms, ACK hold: 200 ms
45. Flags: higher precedence, nagle

46. Datagrams (max data segment is 536 bytes):
47. Rcvd: 11 (out of order: 0), with data: 7, total data bytes: 163
48. Sent: 14 (retransmit: 0), with data: 7, total data bytes: 163
TeddyBear#
```

表 A-1　　对 show ip bgp neighbores 命令输出的逐行解释

行　号	原　句	解　释
1	**BGP neighbor is 10.100.1.1, remote AS 6500, internal link**	在所有输出中，该行结果最不言自喻。IP 地址是 BGP 对等体间 TCP 连接的对端地址。对端实体 AS 号 6500，对等实体是内部的，所以是 IBGP 连接
2	**Member of peer-group Pooh for sesseion parameters**	在路由器上有一个名为 Pooh 的对等体组，该邻居是 Pooh 组成员
3	**BGP version 4, remote router ID 10.100.1.1**	在对等会话中运行的是 BGP-4。回想一下第 2 章“边缘网关协议 4”，BGP 从所支持的最新版本开始与对端自动协商版本。对端的 BGP 路由器 ID 是 10.100.1.1，与对端 IP 地址相同。这是因为对等连接在环回接口间实现，该接口地址用作决定路由器 ID。如果对等连接使用物理接口地址，通常会与路由器 ID 不同
4	**BGP state=Established, up for 00:04:06**	邻居的状态是 established，表示邻接关系完全建立。会话已经存在 4 分钟 6 秒
5	**Last read 00:00:07, hold time is 180, keepalive interval is 60 seconds**	BGP 最近一次从对端读取信息在 7 秒以前。Holdtime 示 180 秒，keepalive 间隔 60 秒。上述值都是缺省值；回想第 2 章，BGP 邻居间可以协商这些定时器
6	**Neighbors capabilities:**	该行指出，下面行是描述对端的能力
7	**Route refresh: advertised and received**	本地路由器和远端都支持增强的路由器软件复位(该能力由本地路由器发布，远端路由器接收)。上述特性在 IOS 12.0 中发布，允许不需要配置 **neighbor soft-reconfiguration** 就可以将改变的规则自动发布到对端
8	**Address family IPv4 Unicast: advertised and received**	本地路由器和对端都运行多协议 BGP (MBGP)。对端已激活支持单播 NLRI。查阅第 7 章“大规模 IP 多播选路”中 MBGP 和地址族指示器的讨论。与上面行一样，该能力由本地发布，对端接收

续表

行　号	原　句	解　释
9	**Address family IPv4 Multicast: advertised and received**	两端都支持 BGP 多播 NLRI
10	**Received 7 messages, 0 notifications, 0 in queue**	包括 keepalive，共收到 7 个消息。没有收到 notification 消息，消息队列中没有消息待接收
11	**Sent 7 messages, 0 notifications, 0 in queue**	包括 keepalive，供发送 7 个消息。没有发送 notification 消息，消息队列中没有消息待发送
12	**Route refresh request: received 0, sent 0**	没有收到或发送路由更新请求(对增强的 BGP 软件复位)
13	**Minimum time between advertisement runs is 5 seconds**	最小的更新间隔是 5 秒
14	**For address family: IPv4 Unicast**	该行指示下面行适用于单播 BGP 路由表
15	**BGP table version 1, neighbor version 1**	邻居已升级为 BGP 单播路由表版本 1，本地路由器使用对端的 BGP 单播路由表版本 1 升级
16	**Index1, Offset 0, Mask 0x2**	引用指定对端的内部索引。该字段仅由能访问源代码的 Cisco 人员使用
17	**Pooh peer-group member**	该邻居是单播对等组 Pooh 的成员
18	**0 accepted prefixes consume 0 bytes**	该行指出从对端接收了多少个单播前缀以及内存中填充了多少字节的前缀。在这里，没有接到前缀
19	**Prefix advertised 0, suppressed 0, withdrawn 0**	从对端没有单播前缀被发送、抑制或撤销
20	**For address family: IPv4 Multicast**	该行指示下边行适用于多播路由表
21	**BGP table version 1, neighbor version 1**	邻居已升级为 BGP 多播路由表版本 1，本地路由器使用对端的 BGP 多播路由表版本 1 升级
22	**Index 1, Offset 0, Mask 0x2**	引用指定对端的内部索引。该字段仅由能访问源代码的 Cisco 人员使用
23	**0 accepted prefixes consume 0 bytes**	该行指出从对端接收了多少个多播前缀以及内存中填充了多少字节的前缀。在这里，没有接到前缀
24	**Prefix advertised 0, suppressed 0, withdrawn 0**	从对端没有多播前缀被发送、抑制或撤销
25	**Connections established 1; dropped 0**	与对端的 BGP 连接和邻接关系之间建立过 1 次，并且连接没有中断过
26	**Last reset 00:04:17, due to Address family activated**	与对端的上一次复位发生在 4 分 17 秒以前，MBGP 被激活时

续表

行　　号	原　　句	解　　释
27	**Connection state is ESTAB, I/O status: 1, unread input bytes: 0**	Connection status 是对端连接的状态-第 4 行显示结果的重复。I/O status 描述的是内部状态。Unread input byte 是 BGP 位处理的字节数
28	**Local host: 10.100.1.2, Local port: 11012**	该行是本地 Ip 套接字，包含 IP 地址和本地 TCP 端口号
29	**Foreign host: 10.100.1.1, Foreign port: 179**	这是对端的 Ip 套接字，包含对端 IP 地址和 TCP 端口号。与 28 行结果比较，你将发现是本地路由器发起的 TCP 连接。原因是对端使用的是公开的 BGP 端口号 179 而本地不是
30	**Enqueued packets for retransmit: 0, input: 0 mis-ordered: 0 (0 bytes)**	在队列中等待重传，输入或次序错误的包
31	**Event Timers (current time is 0x3ABFA00):**	该行是下面所跟随的时间定时器的头
32	**Timer, Starts, Wakeups, Next**	时间定时器列的头
33	**Retrans**	指定一个传输的帧保持未被确认到 Cisco IOS 软件查询确认的时间
34	**TimeWait**	指定本地 TCP 连接等待确认远程 TCP 主机接收到连接中断请求通知的时间
35	**AckHold**	系统无法承载 TCP 确认的时间，承载确认能够有效地减少网络流量
36	**SendWnd**	发送 0 窗口探测的定时器。该字段本质上指出用户是远程主机超载的频率以及用户需要多少时间发送它。对大多数 Cisco IOS 软件应用，该值为 0
37	**KeepAlive**	指定 Cisco IOS 软件向自身(以太网或令牌环)或对端(串口)发送消息确认网络端口正常的频率。**keepalive interface configuration** 命令用作配置该定时器
41	**iss:2227710177 snduna: 2227710341 sndnxt: 2227710341 sndwnd:16221**	TCP 连接中使用的序列号 iss：最初发送的序列号 snduna：未被应答的发送序列号；最后一个本地主机发送但是未接到确认的序列号 sndnxt：本地路由器下一次发送的序列号 sndwnd：远端主机的 TCP 窗口大小

续表

行　号	原　句	解　释
42	**irs 1632859231 rcvnxt: 1632859395 rcvwnd: 16221 delrcvwnd: 163**	远端 TCP 连接中使用的序列号 irs：最初接收的序列号 rcvnxt：本地路由器已确认的最后一个的序列号 rcvwnd：本地主机的 TCP 窗口大小 delrcvwnd：被延后的接收窗口-本地主机已经从窗口中读取但是还没有从通知远程主机的接收窗口中减去的数据。该值逐渐递增直到大于最大尺寸的包，这时再应用到 rcvwnd 字段
43	**SRTT: 540ms, RTTO: 3809ms, RTV: 1364ms, KRTT: 0ms**	这些数值关于对等连接的时延 SRTT：计算得到的平均往返时间 RTTO：往返超时 RTV：往返时间的变化 KRTT：Karn 往返时间；新的往返超时(使用 Karn 算法)。该字段单独跟踪重传包的往返时间
44	***minRTT*: 8ms, *maxRTT*: 300ms, ACK hold: 200ms**	这些值是 43 行性能值的延续 *minRTT*：记录到最小的往返时间 maxRTT：记录到最大的往返时间 ACK hold：本地路由器将确认延时以便承载的延时时间
45	**Flags:higher precedence, *nagle***	BGP 包的 IP 优先级
46	**Datagrams (max data segment is 536 bytes):**	是下两行的表头，提供 BGP 从对端收发 Update 的统计。同时显示最大 TCP 分段是 536 字节
47	**Rcvd: 11 (out of order: 0), with data:7, total data bytes: 163**	收到 Update 的统计，包含所有收到的 update，数据的报文数和总字节数
48	**Sent: 14 (retransmit: 0), with data: 7, total data bytes: 163**	发送 Update 的统计，包含所有发出的 update，数据的报文数和总字节数

附录 B　正则表达式指南

本指南遵从 Jeffrey E.F.Friedl 在著作 Mastering Regular Expression 中的完美表述。尽管运行 Cisco IOS 所需要的正则表达式(regex)知识仅包含在上述著作中最前面，该书还是罗列在本附录最后推荐阅读的书目中。因为您你将发现正则表达式在数据通信和数据处理行业有非常广泛的应用，所以建议阅读该书。Friedl 对主体的表述不但清楚，而且轻松幽默。

1. 字和元字符特殊字符

一个典型的 AS_PATH 过滤器如下所示：

ip as-path access-list 83 permit ^1_701_(_5646_|_1240).*

在关键字 **permit** 后面的字符串是一个正则表达式。正则表达式包含字和元字符特殊字符。字是正则表达式试图匹配的一些正文字符。在上述例子中 **1**，**701**，**5646** 和 **1240** 都是描述自治系统号的字。

元字符特殊字符是用作运算符的特殊的正则表达式符号，告诉正则表达式如何匹配。表 B-1 显示了 Cisoc IOS 会用到的元字符特殊字符；本附录的剩余部分表述每一个元字符特殊字符的用法。

表 B-1　　关于 AS_PATH 访问列表的正则表达式元字符特殊字符

元字符特殊字符	匹 配 内 容
.	任何单一字符，包括空格
[]	在方括弧中罗列的任何字符
[^]	除了在方括弧中所罗列字符外任何字符(^必须放置在字符列表之前)
-	(连字符)在由连字符所分隔的两个字符之间的任意字符
?	字符或模式出现 0 次或 1 次
*	字符或模式出现 0 次或多次
+	字符或模式出现 1 次或多次
^	一行的开始
$	一行的结束
|	由元字符特殊字符分隔的字之一
_	(下划线)一个逗号，行的开始，行的结束或空格

2. 描述：匹配行的起始和结束

考虑下面的 AS_PATH 过滤器：

ip as-path access-list 20 permit 850

该过滤器匹配任何包含字符串 **850** 的 AS_PATH。匹配的 AS_PATH 例如：**(850)**，(23，5，**850**，155)和(3568，**5850**，310)等。无论所匹配的串是属性中唯一的串或者属性中多个 AS 号中的一个甚至是属性中一个很大的 AS 号中的一部分，该匹配都成功。

假设你只想匹配包含唯一 AS 号 850 的 AS_PATH，您你必须描述行的开始和结束。使用补字号(^)匹配行的开始，美元符($)匹配行的结束。于是

ip as-path access-list 20 permit ^850$

这样告诉正则：表达式行的开始紧接一个字符串 850，然后紧接行的结束。

你还可以使用两个描述来匹配一个空的 AS_PATH：

ip as-path access-list 21 permit ^$

在上述情况，正则表达式匹配行的开始紧接行的结束；如果行的开始与行的结束间存在任何字符则匹配不成功。

3. 括弧：匹配字符集和

括弧是你能指定单字符的范围。例如：

ip as-path access-list 22 permit ^85[0123459]$

上述过滤器匹配包含单一 AS 号 850，851，852，853，854，855 或 859 的 AS_PATH。

如果字符的范围连续，你可以只指定序列中开始和结束的字符：

ip as-path access-list 23 permit ^85[0-5]$

该过滤器匹配上一个过滤器中除 859 以外的其他 AS 号。

4. 否定：匹配除字符集和外的其他任何字符

当在括弧中使用加字符时，将否定括弧中指定的范围。结果是正则表达式将匹配范围外的任何内容。例如：

ip as-path access-list 24 permit ^85[^0-5]$

该过滤器除了多一个加字符以外与上一个过滤器类似，指定了“不是 0~5”。该正则表达式将匹配 856~859 范围内的单个 AS 号。

5. 通配符：匹配任何单一字符

点(.)匹配任何单一字符。单一字符可能是一个空的。考虑下面过滤器：

ip as-path access-list 24 permit ^85.

该过滤器匹配一个由 AS 号 850-859 开始的 AS_PATH。由于‘.’可以匹配一个空格，AS 号 85 也能匹配成功。

6. 替代：匹配字符串集合中的一个

(|)用作表示一个“或”操作。即(|)任何一边的字都可以匹配。例如：

ip as-path access-list 25 permit ^(851|852)$

该过滤器匹配单个 AS 号：851 或 852 的 AS_PATH。你可以扩展“或”功能来作多于两个可能性的选择：

ip as-path access-list 26 permit ^(851|852|6341|53)$

7. 选择字符：匹配一个可能存在也可能不存在的字符

问号(？)匹配字的 0 个或一个实例。例如

ip as-path access-list 27 permit ^(850)?$

该过滤器匹配一个单一 AS 号 850 的 AS_PATH 或者匹配一个空的列表。注意这里的园括弧，表示元字符特殊字符‘？’应用在整个 AS 号上。如果表达式使用 850？，元字符特殊字符只应用在最后一个字符上。该表达式匹配 85 或者 850。

8. 重复：匹配许多字符的重复

你可以使用两个元字符特殊字符来匹配重复的词：星号(*)匹配 0 次或多次重复，加号(+)匹配 1 次或多重复。例如

ip as-path access-list 28 permit ^(850)*$

该过滤器 AS-PATH 匹配空的 AS 列表或包含 1 个或多个的 AS 号 850 的列表。即 AS 路径可以是(850)，(850，850)，(850，850，850)等。

下面的过滤器功能类似，只是要求列表中至少包含 1 个 AS 号 850。

ip as-path access-list 29 permit ^(850)+$

9. 分界线：描述多个字

分界线(_)用作描述分隔分别指定的一串字。例如假设你想匹配指定的 AS_PATH(5610，148，284，13)。过滤器可以如下所示：

ip as-path access-list 30 permit ^5610_148_284_13$

下划线匹配行的开始、行的结束、逗号或空格。注意下面过滤器与上述过滤器的区别：

ip as-path access-list 31 permit-*5610*_148_284_13_

由于第一个过滤器指定了行的开始与结束，只有 AS_PATH(5610，148，284，13)才能匹配。在第二个过滤器中，只要求指定的序列包含在 AS_PATH 中，并不要求是唯一的属性。

所以 AS_PATH(5610，148，284，13)，(23，15，5610，148，284，13)和(5610，148，284，13，3005)都能匹配。

10. 放到一起：一个复杂的例子

将多个元字符特殊字符混合是用来匹配一些复杂的字符串才能显示正则表达式的真正威力。考虑下面的过滤器：

ip as-path access-list 10 permit ^(550)+_[880|2304]?_1805_.*

上述过滤器寻找一个 AS_PATH，该路径路由最后一个 AS 号是 550。550 前面的加字符表示 550 是列表的第一个数。550 后的加号表示 550 至少出现一次，也可能出现多次。通过允许出现多于 1 个实例，过滤器允许出现第 3 章“边缘网关协议 4 的配置和故障查找排除”中讨论的情况：AS550 在实现路径计划。

在一个或多个 550 之后，可能存在 880 或者 2304。接下来必须有一个 1805。最后的部分指定 1805 以后，AS-PATH 可能存在任何 AS 号序列，也可能什么也没有。

11. 推荐书目

Friedl,Jeffrey E.F.Mastering Regular Expression.Sebastopol, California:O'Reilly & Associates; 1997.

附录 C　保留的多播地址

下边的列表是本书完成之前最新的保留多播地址。该列表是直接从 ftp://ftp.isi.edu/in-notes/iana/assignments/multicast-addresses 下载的。如果需要最新的列表，请查询该网站。下面是这篇文档的原文。

Internet Mnlticast Addresses

Host Extensions for IP Multicasting [RFC1112] specifies the extensions required of a host implementation of the Internet Protocol (IP) to support multicasting. The multicast addressess are in the range 224.0.0.0 through 239.255.255.255. Current addresses are listed in the following text.

The range of addresses between 224.0.0.0 and 224.0.0.255, inclusive, is reserved for the use of routing protocols and other low-level topology discovery or maintenance protocols, such as gateway discovery and group membership reporting. Multicast routers should not forward any multicast datagram with destination addresses in this range, regardless of its TTL.

224.0.0.0	Base Address (Reserved)	[RFC1112,JBP]
224.0.0.1	All Systems On this Subnet	[RFC1112,JBP]
224.0.0.2	All Routers On this Subnet	[JBP]
224.0.0.3	Unassigned	[JBP]
224.0.0.4	DVMRP Routers	[RFC1075,JBP]
224.0.0.5	OSPFIGP OSPFIGP All Routers	[RFC2328,JXM1]
224.0.0.6	OSPFIGP OSPFIGP Designated Routers	[RFC2328,JXM1]
224.0.0 7	ST Routers	[RFC1190,KS14]
224.0 0.8	ST Hosts	[RFC1190,KS14]
224.0.0.9	RIP2 Routers	[RFC1723,GSM11]
224.0.0.10	IGRP Routers	[Farinacci]
224.0.0.11	Mobile-Agents	[Bill Simpson]
224.0.0.12	DHCP Server/ Relay Agent	[RFC1884]
224.0.0.13	All PIM Routers	[Farinacci]
224.0.0.14	RSVP-ENCAPSULATION	[Braden]

224.0.0.15	all-cbt-routers	[Ballardie]
224.0.0.16	designated-sbm	[Baker]
224.0.0.17	all-sbms	[Baker]
224.0.0.18	VRRP	[Hinden]
224.0.0.19	IPAllL1ISs	[Przygienda]
224.0.0.20	IPAllL2ISs	[Przygienda]
224.0.0.21	IPAllIntermediate Systems	[Przygienda]
224.0.0 22	IGMP	[Deering]
224.0 0.23	GLOBECAST-ID	[Scannell]
224.0.0.24	Unassigned	[JBP]
224.0.0.25	router-to-switch	[Wu]
224.0.0.26	Unassigned	[JBP]
224.0.0.27	Al MPP Hello	[Martinicky]
224.0 0.28	ETC Control	[Polishinski]
224.0.0.29	GE-FANUC	[Wacey]
224.0.0.30	indigo-vhdp	[Caughie]
224.0 0.31	shinbroadband	[Kittivatcharapong]
224.0.0.32	digistar	[Kerkan]
224.0.0.33	ff-system-management	[Glanzer]
224.0.0.34	pt2-discover	[Kammerlander]
224.0.0.35	DXCLUSTER	[Koopman]
224.0 0.36-224.0.0.250	Unassigned	[JBP]
224.0.0.251	mDNS	[Cheshire]
224.0.0.252-224.0.0.255	Unassigned	[JBP]
224.0.1.0	VMTP Managers Group	[RFC1045,DRC3]
224 0.1.1	NTP Network Time Protocol	[RFC1119,DLM1]
224.0.1.2	SGI-Dogfight	[AXC]
224 0.1.3	Rwhod	[SXD]
224.0.1.4	VNP	[DRC3]
224.0 1.5	Artificial Horizons—Aviator	[BXF]
224.0.1.6	NSS—Name Service Server	[BXS2]

224.0.1.7	AUDIONEWS－Audio News Multicast	[MXF2]
224.0.1.8	SUN NIS+ Information Service	[CXM3]
224.0.1.9	MTP Multicast Transport Protocol	[SXA]
224.0.1.10	IETF-1-LOW-AUDIO	[SC3]
224.0.1.11	IETF-1-AUDIO	[SC3]
224.0.1.12	IETF-1-VIDEO	[SC3]
224.0.1.13	IETF-2-LOW-AUDIO	[SC3]
224.0.1.14	IETF-2-AUDIO	[SC3]
224.0.1.15	IETF-2-VIDEO	[SC3]
224.0.1.16	MUSIC-SERVICE	[Guido van Rossum]
224.0.1.17	SEANET-TELEMETRY	[Andrew Maffei]
224.0.1.18	SEANET-IMAGE	[Andrew Maffei]
224.0.1.19	MLOADD	[Braden]
224.0.1.20	any private experiment	[JBP]
224.0.1.21	DVMRP on MOSPF	[John Moy]
224.0.1.22	SVRLOC	[Veizades]
224.0.1.23	XINGTV	[Gordon]
224.0.1.24	microsoft-ds	<arnoldm@microsoft.com>
224.0.1.25	nbc-pro	<bloomer@birch.crd.ge.com>
224.0.1.26	nbc-pfn	<bloomer@birch.crd.ge.com>
224.0.1.27	lmsc-calren-1	[Uang]
224.0.1.28	lmsc-calren-2	[Uang]
224.0.1.29	lmsc-calren-3	[Uang]
224.0.1.30	lmsc-calren-4	[Uang]
224.0.1.31	ampr-info	[Janssen]
224.0.1.32	mtrace	[Casner]
224.0.1.33	RSVP-encap-1	[Braden]
224.0.1.34	RSVP-encap-2	[Braden]
224.0.1.35	SVRLOC-DA	[Veizades]
224.0.1.36	rln-server	[Kean]
224.0.1.37	proshare-mc	[Lewis]

224.0.1.38	dantz	[Zulch]
224.0.1.39	cisco-rp-announce	[Farinacci]
224.0.1.40	cisco-rp-discovery	[Farinacci]
224.0.1.41	gatekeeper	[Toga]
224.0.1.42	iberiagames	[Marocho]
224.0.1.43	nwn-discovery	[Zwemmer]
224.0.1.44	nwn-adaptor	[Zwemmer]
224.0.1.45	isma-1	[Dunne]
224.0.1.46	isma-2	[Dunne]
224.0.1.47	telerate	[Peng]
224.0 1.48	ciena	[Rodbell]
224.0.1.49	dcap-servers	[RFC2114]
224.0.1.50	dcap-clients	[RFC2114]
224.0.1.51	mcntp-directory	[Rupp]
224.0.1.52	mbone-vcr-directory	[Holfelder]
224.0 1.53	heartbeat	[Mamakos]
224.0.1.54	sun-mc-grp	[DeMoney]
224.0.1.55	extended-sys	[Poole]
224.0.1.56	pdmcs	[Wissenbach]
224.0.1.57	tns-adv-multi	[Albin]
224.0.1.58	vcals-dmu	[Shindoh]
224.0.1.59	zuba	[Jackson]
224.0.1.60	hp-device-disc	[Albright]
224.0.1.61	tms-production	[Gilani]
224.0.1.62	sunscalar	[Gibson]
224.0.1.63	mmtp-poll	[Costales]
224.0.1.64	compaq-peer	[Volpe]
224.0.1.65	iapp	[Meier]
224.0.1.66	multihasc-com	[Brockbank]
224.0.1.67	serv-discovery	[Honton]
224.0.1.68	mdhcpdisover	[RFC2730]

224.0.1.69	MMP-bundle-discovery1	[Malkin]
224.0.1.70	MMP-bundle-discovery2	[Malkin]
224.0.1.71	XYPOINT DGPS Data Feed	[Green]
224.0.1.72	GilatSkySurfer	[Gal]
224.0.1.73	SharesLive	[Rowatt]
224.0.1.74	NorthernData	[Sheers]
224.0.1.75	SIP	[Schulzrinne]
224.0.1.76	IAPP	[Moelard]
224.0.1.77	AGENTVIEW	[Iyer]
224.0.1.78	Tibco Multicast1	[Shum]
224.0.1.79	Tibco Multicast2	[Shum]
224.0.1.80	MSP	[Caves]
224.0.1.81	OTT(One-way Trip Time)	[Schwartz]
224.0.1.82	TRACKTICKER	[Novick]
224.0.1.83	dtn-mc	[Gaddie]
224.0.1.84	jmi-announcement	[Scheifler]
224.0.1.85	jmi-request	[Scheifler]
224.0.1.86	sde-discovery	[Aronson]
224.0.1.87	DirecPC-SI	[Dillon]
224.0.1.88	B1RMonitor	[Purkiss]
224.0.1.89	3Com-AMP3 dRMON	[Banthia]
224.0.1.90	imFtmSvc	[Bhatti]
224.0.1.91	NQDS4	[Flynn]
224.0.1.92	NQDS5	[Flynn]
224.0.1.93	NQDS6	[Flynn]
224.0.1.94	NLVL12	[Flynn]
224.0.1.95	NTDS1	[Flynn]
224.0.1.96	NTDS2	[Flynn]
224.0.1.97	NODSA	[Flynn]
224.0.1.98	NODSB	[Flynn]
224.0.1.99	NODSC	[Flynn]

224.0.1.100	NODSD	[Flynn]
224.0.1.101	NQDS4R	[Flynn]
224.0.1.102	NQDS5R	[Flynn]
224.0.1.103	NQDS6R	[Flynn]
224.0.1.104	NLVL12R	[Flynn]
224.0.1.105	NTDS1R	[Flynn]
224.0.1.106	NTDS2R	[Flynn]
224.0.1.107	NODSAR	[Flynn]
224.0.1.108	NODSBR	[Flynn]
224.0.1.109	NODSCR	[Flynn]
224.0.1.110	NODSDR	[Flynn]
224.0.1.111	MRM	[Wei]
224.0.1.112	TVE-FILE	[Blackketter]
224.0.1.113	TVE-ANNOUNCE	[Blackketter]
224.0.1.114	Mac Srv Loc	[Woodcock]
224.0.1.115	Simple Multicast	[Crowcroft]
224.0.1.116	SpectraLinkGW	[Hamilton]
224.0.1.117	dieboldmcast	[Marsh]
224.0.1.118	Tivoli Systems	[Gabriel]
224.0.1.119	pq-lic-mcast	[Sledge]
224.0.1.120	HYPERFEED	[Kreutzjans]
224.0.1.121	Pipesplatform	[Dissett]
224.0.1.122	LiebDevMgmg-DM	[Velten]
224.0.1.123	TRIBALVOICE	[Thompson]
224.0.1.124	UDLR-DTCP	[Cipiere]
224.0.1.125	PolyCom Relay1	[Coutiere]
224.0.1.126	Infront Multi1	[Lindeman]
224.0.1.127	XRX DEVICE DISC	[Wang]
224.0.1.128	CNN	[Lynch]
224.0.1.129	PTP-primary	[Eidson]
224.0.1.130	PTP-alternate1	[Eidson]

224.0.1.131	PTP-alternate2	[Eidson]
224.0.1.132	PTP-alternate3	[Eidson]
224.0.1.133	ProCast	[Revzen]
224.0.1.134	3Com Discp	[White]
224.0.1.135	CS-Multicasting	[Stanev]
224.0.1.136	TS-MC-1	[Sveistrup]
224.0.1.137	Make Source	[Daga]
224.0.1.138	Teleborsa	[Strazzera]
224.0.1.139	SUMAConfig	[Wallach]
224.0.1.140	Unassigned	
224.0.1.141	DHCP-SERVERS	[Hall]
224.0.1.142	CN Router-LL	[Armitage]
224.0.1.143	EMWIN	[Querubin]
224.0.1.144	Alchemy Cluster	[O'Rourke]
224.0.1.145	Satcast One	[Nevell]
224.0.1.146	Satcast Two	[Nevell]
224.0.1.147	Satcast Three	[Nevell]
224.0.1.148	Intline	[Sliwinski]
224.0.1.149	8x8 Multicast	[Roper]
224.0.1.150	Unassigned	[JBP]
224.0.1.151	Intline-1	[Sliwinski]
224.0.1.152	Intline-2	[Sliwinski]
224.0.1.153	Intline-3	[Sliwinski]
224.0.1.154	Intline-4	[Sliwinski]
224.0.1.155	Intline-5	[Sliwinski]
224.0.1.156	Intline-6	[Sliwinski]
224.0.1.157	Intline-7	[Sliwinski]
224.0.1.158	Intline-8	[Sliwinski]
224.0.1.159	Intline-9	[Sliwinski]
224.0.1.160	Intline-10	[Sliwinski]
224.0.1.161	Intline-11	[Sliwinski]

224.0.1.162	Intline-12	[Sliwinski]
224.0.1.163	Intline-13	[Sliwinski]
224.0.1.164	Intline-14	[Sliwinski]
224.0.1.165	Intline-15	[Sliwinski]
224.0.1.166	marratech-cc	[Parnes]
224.0.1.167	EMS-InterDev	[Lyda]
224.0.1.168	itb301	[Rueskamp]
224.0.1.169	rtv-audio	[Adams]
224.0.1.170	rtv-video	[Adams]
224.0.1.171	HAVI-Sim	[Wasserroth]
224.0.1.172-224.0.1.255	Unassigned	[JBP]
224.0.2.1	"rwho" Group(BSD) (unofficial)	[JBP]
224.0.2.2	SUN RPC PMAPPROC_CALLIT	[BXE1]
224.0.2.064-224.0.2.095	SIAC MDD Service	[Tse]
224.0.2.096-224.0.2.127	CoolCast	[Ballister]
224.0.2.128-224.0.2.191	WOZ-Garage	[Marquardt]
224.0.2.192-224.0.2.255	SIAC MDD Market Service	[Lamberg]
224.0.3.000-224.0.3.255	RFE Generic Service	[DXS3]
224.0.4.000-224.0.4.255	RFE Individual Conferences	[DXS3]
224.0.5.000-224.0.5.127	CDPD Groups	[Bob Brenner]
224.0.5.128-224.0.5.191	SIAC Market Service	[Cho]
224.0.5.192-224.0.5.255	Unassigned	[IANA]
224.0.6.000-224.0.6.127	Cornell ISIS Project	[Tim Clark]
224.0.6.128-224.0.6.255	Unassigned	[IANA]
224.0.7.000-224.0.7.255	Where-Are-You	[Simpson]
224.0.8.000-224.0.8.255	INTV	[Tynan]
224.0.9.000-224.0.9.255	Invisible Worlds	[Malamud]
224.0.10.000-224.0.10.255	DLSw Groups	[Lee]
224.0.11.000-224.0.11.255	NCC.NET Audio	[Rubin]
224.0.12.000-224.0.12.063	Microsoft and MSNBC	[Blank]
224.0.13.000-224.0.13.255	UUNET PIPEX Net News	[Barber]

224.0.14.000-224.0.14.255	NLANR	[Wessels]
224.0.15.000-224.0.15.255	Hewlett Packard	[van der Meulen]
224.0.16.000-224.0.16.255	XingNet	[Uusitalo]
224.0.17.000-224.0.17.031	Mercantile& Commodity Exchange	[Gilani]
224.0.17.032-224.0.17.063	NDQMD1	[Nelson]
224.0.17.064-224.0.17.127	ODN-DTV	[Hodges]
224.0.18.000-224.0.18.255	Dow Jones	[Peng]
224.0.19.000-224.0.19.063	Walt Disney Company	[Watson]
224.0.19.064-224.0.19.095	Cal Multicast	[Moran]
224.0.19.096-224.0.19.127	SIAC Market Service	[Roy]
224.0.19.128-224.0.19.191	IIG Multicast	[Carr]
224.0.19.192-224.0.19.207	Metropol	[Crawford]
224.0.19.208-224.0.19.239	Xenoscience, Inc.	[Timm]
224.0.19.240-224.0.19.255	HYPERFEED	[Felix]
224.0.20.000-224.0.20.063	MS-IP/TV	[Wong]
224.0.20.064-224.0.20.127	Reliable Network Solutions	[Vogels]
224.0.20.128-224.0.20.143	TRACKTICKER Group	[Novick]
224.0.20.144-224.0.20.207	CNR Rebroadcast MCA	[Sautter]
224.0.21.000-224.0.21.127	Talarian MCAST	[Mendal]
224.0.22.000-224.0.22.255	WORLD MCAST	[Stewart]
224.0.252.000-224.0.252.255	Domain Scoped Group	[Fenner]
224.0.253.000-224.0.253.255	Report Group	[Fenner]
224.0.254.000-224.0.254.255	Query Group	[Fenner]
224.0.255.000-224.0.255.255	Border Routers	[Fenner]
224.1.0.0-224.1.255.255	ST Multicast Groups	[RFC1190,KS14]
224.2.0.0-224.2.127.253	Multimedia Conference Calls	[SC3]
224.2.127.254	APv1 Announcements	[SC3]
224.2.127.255	SAPv0 Announcements (deprecated)	[SC3]
224.2.128.0-224.2.255.255	SAP Dynamic Assignments	[SC3]

224.252.0.0-224.255.255.255	DIS transient groups	[Joel Snyder]
225.0.0.0-225.255.255.255	MALLOC(temp-renew 1/d)	[Handley]
232.0.0.0-232.255.255.255	VMTP transient group	[DRC3]
	see single-source-multicast file	
233.0.0.0-233.255.255.255	Static Allocations (temp—renew6101)	[Meyer2]
239.000.000.000-239.255. 255.255	Administratively Scoped	[IANA,RFC2365]
239.000.000.000-239.063.255.255	Reserved	[IANA]
239.064.000.000-239.127.255.255	Reserved	[IANA]
239.128.000.000-239.191.255.255	Reserved	[IANA]
239.192.000.000-239.251.255.255	Organization-Local Scope	[Meyer,RFC2365]
239.252.000.000-239.252.255.255	Site-Local Scope (reserved)	[Meyer,RFC2365]
239.253.000.000-239.253.255.255	Site-Local Scope (reserved)	[Meyer,RFC2365]
239.254.000.000-239.254.255.255	Site-Local Scope (reserved)	[Meyer,RFC2365]
239.255.000.000-239.255.255.255	Site-Local Scope	[Meyer,RFC2365]
239.255.002.002	rasadv	[Thaler]

There is a concept of relative addresses to be used with the scoped multicast addresses. These relative addresses are listed here:

Relative	Description	Reference
0	SAP Session Announcement Protocol	[Handley]
1	MADCAP Protocol	[RFC2730]
2	SLPv2 Discovery	[Guttman]
3	MZAP	[Thaler]
4	Multicast Discovery of DNS Services	[Manning]
5	SSDP	[Goland]
6	DHCP v4	[Hall]
7	AAP	[Hanna]
8-252	Reserved - To be assigned by the IANA	
253	Reserved	
254-255	Reserved - To be assigned by the IANA	

These addresses are listed in the Domain Name Service under MCAST.NET and 224.IN-ADDR.ARPA.

Note that when used on an Ethernet or IEEE 802 network, the 23 low-order bits of the IP Multicast address are placed in the low-order 23 bits of the Ethernet or IEEE 802 net multicast address 1.0.94.0.0.0. See the section on " "ANA ETHERNET ADDRESS BLOCK."

References

[RFC1045] Cheriton, D., "VMTP: Versatile Message Transaction Protocol Specification", RFC 1045, Stanford University, February 1988.

[RFC1075] Waitzman, D., C. Partridge, and S. Deering,"Distance Vector Multicast Routing Protocol", RFC-1075, BBN STC, Stanford University, November 1988.

[RFC1112] Deering, S., "Host Extensions for IP Multicasting", STD 5, RFC 1112, Stanford University, August 1989.

[RFC1119] Mills, D., "Network Time Protocol (Version 1), Specification and Implementation", STD 12, RFC 1119, University of Delaware, July 1988.

[RFC1190] Topolcic, C., Editor, "Experimental Internet Stream Protocol, Version 2 (ST-II)", RFC 1190, CIP Working Group, October 1990.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, Ascend Communications, April 1998.

[RFC1723] Malkin, G., "RIP Version 2: Carying Additional Information", RFC 1723, Xylogics, November 1994.

[RFC1884] Hinden, R., and S. Deering, "IP Version 6 Addressing Architecture", RFC 1884, Ipsilon Networks, Xerox PARC, December 1995.

[RFC2114] Chiang, S, J. Lee and H. Yasuda, "Data Link Switching Client Access Protocol", RFC 2114, Cisco, Mitsubishi February 1997.

[RFC2365] Meyer, D., "Administratively Scoped IP Multicast", RFC 2365, University of Oregon, July 1998.

[RFC2730] Hanna, S., B.Patel, and M. Shah, "Multicast Address Dynamic Client Allocation Protocol (MADCAP)", December 1999.

People

[Adams] Chris Adams, <jc.adams@reuters.com>, July 2000.

[Albin] Jerome Albin, <albin@taec.enet.dec.com>, June 1997.

[Albright] Shivaun Albright, <shivaun_albright@hp.com>, July 1997.

[Armitage] Ian Armitage <ian@coactive.com>, August 1999.

[Aronson] Peter Aronson, <paronson@esri.com>, August 1998.

<arnoldm@microsoft.com>

[AXC] Andrew Cherenson <arc@SGI.COM>

[Baker] Fred Baker, <fred@cisco.com>, June 1997.

[Ballardie] Tony Ballardie <A.Ballardie@cs.ucl.ac.uk>, February 1997.

[Ballister] Tom Ballister, <tballister@starguidedigital.com>, July 1997.

[Banthia] Prakash Banthia, <prakash_banthia@3com.com>, September 1998.

[Barber] Tony Barber, <tonyb@pipex.com>, January 1997.

[Bhatti] Zia Bhatti, <zia@netright.com>, September 1998.

[Blackketter] Dean Blackketter, <dean@corp.webtv.net>, November 1998.

[Blank] Tom Blank, <tomblank@microsoft.com>, November 1996.

[Braden] Bob Braden, <braden@isi.edu>, April 1996.

[Bob Brenner]

[Brockbank] Darcy Brockbank <darcy@hasc.com>, December 1997.

<bloomer@birch.crd.ge.com>

[BXE1] Brendan Eic <brendan@illyria.wpd.sgi.com>

[BXF] Bruce Factor <ahi!bigapple!bruce@uunet.UU.NET>

[BXS2] Bill Schilit, <schilit@parc.xerox.com>

[Carr] Wayne Carr <Wayne_Carr@ccm.intel.com>, December 1997.

[Casner] Steve Casner, <casner@isi.edu>, January 1995.

[Caughie] Colin Caughie, <cfc@indigo-avs.com>, May 2000.

[Caves] Evan Caves, <evan@acc.com>, June 1998.

[Cheshire] Stuart Cheshire, <cheshire@apple.com>, April 2000.

[Chiang] Steve Chiang, <schiang@cisco.com>, January 1997

[Cho] Joan Cho/SIAC, <jcho@siac.com>, October 1998.

[Cipiere] Patrick Cipiere, <Patrick.Cipiere@ sophia.inria.fr >, February 1999.

[Tim Clark]

[Costales] Bryan Costales <bcx@infobeat.com>, September 1997.

[Crawford] James Crawford, <jcrawford@metropol.net>, May 1998.

[Crowcroft] Jon Crowcroft, <jon@hocus.cs.ucl.ac.uk>, November 1998.

[CXM3] Chuck McManis, <cmcmanis@sun.com>

[Daga] Anthony Daga ,<anthony@mksrc.com>, June 1999.

[Deering] Steve Deering, <deering@cisco.com>, October 1999.

[DeMoney] Michael DeMoney, <demoney@eng.sun.com>, April 1997.

[Dillonn] Doug Dillon, <dillon@hns.com>, August 1998.

[Dissett] Daniel Dissett, <ddissett@peerlogic.com>, December 1998.

[DLM1] David Mills, <Mills@HUEY.UDEL.EDU>

[DRC3] Dave Cheriton, <cheriton@DSG.STANFORD.EDU>

[Dunne] Stephen Dunne, <sdun@isma.co.uk>, January 1997.

[DXS3] Daniel Steinber, <Daniel.Steinberg@Eng.Sun.COM>

[Eidson] John Eidson, <eidson@hpl.hp.com>, April 1999.

[Fenner] Bill Fenner, <fenner@parc.xerox.com>, December 1997.

[Farinacci] Dino Farinacci, <dino@cisco.com>, February, March 1996.

[Felix] Ken Felix, <kfelix@pcquote.com>, August 1999.

[Flynn] Edward Flynn, <flynne@nasdaq.com>, September 1998.

[Gabriel] Jon Gabriel, <grabriel@tivoli.com>, December 1998.

[Gaddie] Bob Gaddie, <bobg@dtn.com>, August 1998.

[Gal] Yossi Gal, <yossi@gilat.com>, February 1998.

[Gibson] Terry Gibson, <terry.gibson@sun.com>, August 1997.

[Gilani] Asad Gilani, <agilani@nymex.com>, July 1997.

[Glanzer] Dave Glanzer, <dglanzer@fieldbus.org>, June 2000.

[GSM11] Gary S. Malkin <GMALKIN@XYLOGICS.COM>

[Goland] Yaron Goland, <yarong@microsoft.com>, August 1999.

[Gordon] Howard Gordon, <hgordon@xingtech.com>

[Green] Cliff Green, <cgreen@xypoint.com>, February 1998.

[Guttman] Erik Guttman, <Erik.Guttman@eng.sun.com>, March 1998.

[Hall] Eric Hall, <ehall@ntrg.com>, August 1999, October 1999.

[Hamilton] Mark Hamilton, <mah@spectralink.com>, November 1998.

[Handley] Mark Handley, <mjh@ISI.EDU>, December 1998.

[Hanna] Stephen Hanna, <steve.hanna@sun.com>, July 2000.

[Hinden] Bob Hinden, <hinden@Ipsilon.com>, November 1997.

[Hodges] Richard Hodges, <rh@source.net>, March 1999.

[Holfelder] Wieland Holdfelder, <whd@pi4.informatik.uni-mannheim.de>,
January 1997.

[Honton] Chas Honton, <chas@secant.com>, December 1997.

[IANA] IANA, <iana@iana.org>

[Iyer] Ram Iyer <ram@aaccorp.com>, March 1998.

[Jackson] Dan Jackson, <jdan@us.ibm.com>, September 1997.

[Janssen] Rob Janssen, <rob@pe1chl.ampr.org>, January 1995.

[JBP] Jon Postel, <postel@isi.edu>

[JXM1] Jim Miner, <miner@star.com>

[Kammerlander] Ralph Kammerlander, <ralph.kammerlander@khe.siemens.de>,
June 2000.

[Kean] Brian Kean, <bkean@dca.com>, August 1995.

[Kerkan] Brian Kerkan, <brian@satcomsystems.com>, May 2000.

[Kittivatcharapong] Sakon Kittivatcharapong, <sakonk@cscoms.net>, May 2000.

[Koopman] Dirk Koopman, <djk@tobit.co.uk>, July 2000.

[Kreutzjans] Michael Kreutzjans, <mike@pcquote.com>, December 1998.

[KS14] Karen Seo, <kseo@bbn.com>

[Lamberg] Mike Lamberg, <mlamberg@siac.com>, February 1997.

[Lee] Choon Lee, <cwl@nsd.3com.com>, April 1996.

[Lewis] Mark Lewis, <Mark_Lewis@ccm.jf.intel.com>, October 1995.

[Lindeman] Morten Lindeman, <Morten.Lindeman@os.telia.no>, March 1999.

[Lyda] Stephen T. Lyda, <slyda@emsg.com>, February 2000.

[Lynch] Joel Lynch, <joel.lynch@cnn.com>, April 1999.

[Andrew Maffei]

[Malamud] Carl Malamud, <carl@invisible.net>, April 1998.

[Malkin] Gary Scott Malkin <gmalkin@baynetworks.com>, February 1998.

[Mamakos] Louis Mamakos, <louie@uu.net>, March 1997.

[Manning] Bill Manning, <bmanning@isi.edu>, August 1999.

[Marocho] Jose Luis Marocho, <73374.313@compuserve.com>, July 1996.

[Marquardt] Douglas Marquardt, <dmarquar@woz.org>, February 1997.

[Marsh] Gene Marsh, <MarshM@diebold.com>, November 1998.

[Martinicky] Brian Martinicky, <Brian_Martinicky@automationintelligence.com>,
March 2000.

[Meier] Bob Meier, <meierb@norand.com>, December 1997.

[Mendal] Geoff Mendal, <mendal@talarian.com>, January 1999.

[Meyer] David Meyer, <meyer@ns.uoregon.edu>, January 1997.

[Meyer2] David Meyer, <dmm@cisco.com>, June 1999 - MALLOC assignment
for temp use: renew 06/2000

[Moelard] Henri Moelard, <HMOELARD@wcnd.nl.lucent.com>, March 1998.

[Moran] Ed Moran, <admin@cruzjazz.com>, October 1997.

[John Moy] John Moy, <jmoy@casc.com>

[MXF2] Martin Forssen, <maf@dtek.chalmers.se>

[Nelson] Gunnar Nelson, <nelsong@nasd.com>, March 1999.

[Nevell] Julian Nevell, <JNEVELL@vbs.bt.co.uk>, August 1999.

[Novick] Alan Novick, <anovick@tdc.com>, August 1998.

[O'Rourke] Stacey O'Rourke, <stacey@network-alchemy.com>, August 1999,

[Parnes] Peter Parnes, <peppar@marratech.com> February 2000.

[Peng] Wenjie Peng, <wpeng@tts.telerate.com>, January 1997.

[Polishinski] Steve Polishinski, <spolishinski@etcconnect.com>, March 2000.

[Poole] David Poole, <davep@extendsys.com>, April 1997.

[Przygienda] Tony Przygienda, <prz@siara.com>, October 1999.

[Purkiss] Ed Purkiss, <epurkiss@wdmacodi.com>, September 1998.

[Querubin] Antonio Querubin, <tony@lava.net>, August 1999.

[Revzen] Shai Revzen, <shrevz@nmcfast.com>, April 1999.

[Rodbell] Mike Rodbell, <mrodbell@ciena.com>, January 1997.

[Roper] Mike Roper, <mroper@8x8.com>, September 1999.

[Guido van Rossum]

[Rowatt] Shane Rowatt, <shane.rowatt@star.com.au>, March 1997.

[Roy] George Roy, <c/o Bill Owens owens@appliedtheory.com>, October 1997.

[Rubin] David Rubin, <drubin@ncc.net>, August 1996.

[Rupp] Heiko Rupp, <hwr@xlink.net>, January 1997.

[Rueskamp] Bodo Rueskamp, <br@itchigo.com>, March 2000.

[Sautter] Robert Sautter, <rsautter@acdnj.itt.com>, August 1999.

[SC3] Steve Casner, <casner@precept.com>

[Scannell] Piers Scannell, <piers@globecastne.com>, March 2000.

[Scheifler] Bob Scheifler, <Bob.Scheifler@sun.com>, August 1998.

[Schwartz] Beverly Schwartz, <bschwart@BBN.COM>, June 1998.

[Shindoh] Masato Shindoh, <j111456@yamato.ibm.co.jp>, August 1997.

[Shum] Raymond Shum, <rshum@ms.com>, April 1998.

[Simpson] Bill Simpson, <bill.simpson@um.cc.umich.edu> November 1994.

[Sledge] Bob Sledge, <bob@pqsystems.com>, December 1998.

[Sliwinski] Robert Sliwinski, <sliwinre@mail1st.com>, February 2000.

[Joel Snyder]

[Stanev] Nedelcho Stanev, <nstanev@csoft.bg>, May 1999.

[Stewart] Ian Stewart, <iandbige@yahoo.com>, June 1999.

[Strazzera] Paolo Strazzera, <p.strazzera@telematica.it>, June 1999.

[Sveistrup] Darrell Sveistrup, <darrells@truesolutions.net>, June 1999.

[SXA] Susie Armstrong, <Armstrong.wbst128@XEROX.COM>

[SXD] Steve Deering, <deering@PARC.XEROX.COM>

[Thaler] Dave Thaler, <dthaler@microsoft.com>, March 1999, June 2000.

[Thompson] Nigel Thompson, <nigelt@tribal.com>, January 1999.

[Timm] Mary Timm, <mary@xenoscience.com>, July 1998.

[tynan] Dermot Tynan, <dtynan@claddagh.ie>, August 1995.

[Toga] Jim Toga, <jtoga@ibeam.jf.intel.com>, May 1996.

[Tse] Geordie Tse, <gtse@siac.com>, April 1996.

[Uang] Yea Uang, <uang@force.decnet.lockheed.com> November 1994.

[Uusitalo] Mika Uusitalo, <msu@xingtech.com>, April 1997.

[van der Muelen] Ron van der Muelen, <ronv@lsid.hp.com> February 1997.

[Veizades] John Veizades, <veizades@tgv.com>, May 1995.

[Velten] Mike Velten, <mike_velten@liebert.com>, January 1999.

[Vogels] Werner Vogels, <vogels@rnets.com>, August 1998.

[Volpe] Victor Volpe, <vvolpe@smtp.microcom.com>, October 1997.

[Wacey] Ian Wacey, <iain.wacey@gefalbany.ge.com>, May 2000.

[Wallach] Walter Wallach, <walt@sumatech.com>, July 1999.

[Wang] Michael Wang, <Michael.Wang@usa.xerox.com>, March 1999.

[Wasserroth] Stephan Wasserroth, <wasserroth@fokus.gmd.de>, July 2000.

[Watson] Scott Watson, <scott@disney.com>, August 1997.

[Wei] Liming Wei, <lwei@cisco.com>, October 1998.

[Wessels] Duane Wessels, <wessels@nlanr.net>, February 1997.

[White] Peter White, <peter_white@3com.com>, April 1999.

[Wissenbach] Paul Wissenbach, <paulwi@vnd.tek.com>, June 1997.

[Wong] Tony Wong, <wongt@ms.com>, July 1998.

[Woodcock] Bill Woodcock, <woody@zocalo.net>, November 1998.

[Wu] Ishan Wu, <iwu@cisco.com>, March 2000.

[Zulch] Richard Zulch, <richard_zulch@dantz.com>, February 1996.

[Zwemmer] Arnoud Zwemmer, <arnoud@nwn.nl>, November 1996.

附录 D　复习问题的答案

第 1 章　复习问题答案

1．EGP 的当前版本是多少？

答案：当前 EGP 版本为 2。

2．什么是 EGP 内部邻居？什么是 EGP 外部邻居？

答案：在同一 AS 的邻居是内部邻居，不同 AS 的邻居是外部邻居。

3．EGP 末梢网关与 EGP 核心网关的主要区别是什么？

答案：末梢网关只广告 AS 内部的网络，核心网关同时广告内部和外部的网络。

4．为什么 EGP 采用核心或者骨干 AS 的概念？

答案：EGP 没有发现环回的机制。所以必须设计物理上无环的拓扑，这样内部流量才能够穿过骨干网。

5．主动 EGP 邻居和被动 EGP 邻居之间的区别是什么？

答案：主动邻居发器对端关系并且发送 Hello 包来维护该邻接关系。被动邻居只使用包含 I-Heared-You 的 Hello 消息作应答。

6．EGP 轮询(Poll)消息的目的是什么？

答案：轮询消息请求邻居更新。

7．什么是间接的，或者说是第三方邻居？

答案：非直接邻居是一个网关，该网关于另一网关共享一个数据链路并且能通过上述网关到达某些网络，但是没有与上述网关建立直接对等关系。该网关从数据链路上的另一网关获得可达性消息。

8．EGP 如何使用它的度量来计算到一个目的地的最佳路径？

答案：尽管 EGP 有 metric，但是不使用量度来决定最佳路径。所以量度只用于指示不可达网络。

第 2 章　复习问题答案

1．BGP-4 和早期版本的 BGP 之间最重要的区别是什么？

答案：BGP-4 是无类的，低版本是基于类的。

2．开发 CIDR 是为了减轻哪两方面的问题？

答案：设计 CIDR 能够缓解互联网路由表的爆炸和 B 类地址的消耗速度。

3．有类别和无类别 IP 路由器之间的区别是什么？

答案：有类别的 IP 路由器执行选路时首先查找主类的网络地址，然后匹配子网。无类别的路由器忽略目标地址的类信息，直接作最长匹配地址前缀。

4．有类别和无类别 IP 路由协议之间的区别是什么？

答案：有类别的路由协议只宣告网络地址和子网地址，没有前缀长度信息。作为结果，这样收到宣告的路由器必须对前缀作一定的假设。无类别路由协议包含了允许接收路由器区别前缀长度的信息。作为结果，在无类协议中可能实现 VLSM 和地址总结。

5．给出 4 个地址 172.17.208.0/23，172.17.210.0/23，172.17.212.0/23，172.17.214.0/23，用一个聚合地址来归纳以上地址，使用可能的最长的地址掩码。

答案：172.17.208.0/21。

6．什么是地址前缀？

答案：地址前缀是 IP 地址中的一部分，路由器基于地址前缀作路由选择。在一个分类的环境中，前缀是一个上类网络地址或者其子网。在无类环境中，前缀可能是 32 比特地址中从前开始任何数量的比特。

7．例 2-16 中的路由表来自一个无类别路由器。路由器会将带有下列地址的数据包转发到哪一个下一跳地址？

172.20.3.5

172.20.1.67

172.21.255.254

172.16.50.50

172.16.0.224

172.16.51.50

172.17.40.1

172.17.41.1

172.30.1.1

例 2-16　复习问题 7 的路由表

```
Stratford#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

     172.20.0.0 is variably subnetted, 6 subnets, 2 masks
D       172.20.0.0 255.255.0.0 [90/409600] via 172.20.5.2, 00:01:50, Ethernet0
D       172.20.2.0 255.255.255.0
           [90/409600] via 172.20.6.2, 00:01:50, Ethernet1
D       172.20.3.0 255.255.255.0
           [90/5401600] via 172.20.6.2, 00:01:50, Ethernet1
C       172.20.5.0 255.255.255.0 is directly connected, Ethernet0
C       172.20.6.0 255.255.255.0 is directly connected, Ethernet1
C       172.20.7.0 255.255.255.0 is directly connected, Ethernet2
     172.16.0.0 is variably subnetted, 3 subnets, 2 masks
D       172.16.50.0 255.255.255.0
           [90/409600] via 172.20.6.2, 00:01:50, Ethernet1
D       172.16.0.0 255.255.255.0
           [90/460800] via 172.20.6.2, 00:01:51, Ethernet1
D       172.16.0.0 255.255.0.0 [90/409600] via 172.20.7.2, 00:01:51, Ethernet2
     172.17.0.0 is subnetted (mask is 255.255.255.0), 1 subnets
D       172.17.40.0 [90/2841600] via 172.20.7.2, 00:01:52, Ethernet2
D   172.16.0.0 (mask is 255.240.0.0) [90/409600] via 172.20.5.2, 00:01:52, Ethernet0
Stratford#
```

答案：

目的地地址	下一条地址
172.20.3.5	172.20.6.2
172.20.1.67	172.20.5.2
172.21.255.254	172.20.5.2
172.16.50.50	172.20.6.2
172.16.0.224	172 20.6.2
172.16.51.50	172.20.7.2
172.17.40.1	172.20.7.2
172.17.41.1	172.20.5.2
172.30.1.1	丢弃

8．解释为什么路由总结可以帮助隐藏网络的不稳定性。

答案：由一个汇聚地址总结的成员地址，或者说目标地址，不会广告到总结点之外。如果成员地址的状态发生变化，该变化不会广告到总结地址之外。

9．解释为什么路由总结会引起不对称的业务量类型。

答案：路由总结能隐藏总结点之后的互联网络。如果有多个路由器广告总结地址，在总结节点之外的路由器只选择其中一个。

10．不对称的业务量是不希望出现的吗？

答案：该答案太主观。不对称流量会使设置流量基线和故障查找排除更困难，在地理范围很大时可能影响时延敏感的通信。另一方面，路由总结带来的优点可以平衡这些问题。

11．什么是 NAP?

答案：网络访问点(NAP)是一个服务提供商可以相互连接的局域网或交换机。观察互联网商的通信流，NAP 是拓扑层次最高的点。

12．什么是路由服务器？

答案：路由服务器是一种可以通过路由协议与之建立对等关系的服务器。每一个路由器向路由服务器发送更新消息，而不是向其他对等实体。路由服务器在应用了恰当的路由策略以后将更新消息发到其他对等体。当许多路由器需要通过一条公共的数据链路建立对等关系，例如在 NAP 点时非常有用，能够有效地减少每个路由器需要建立的对等会话数量。这在单播协议例如 BGP 中特别重要，在 BGP 中必须独立地向每个对等提发数据。路由服务器不作数据转发，所以不是路由器。

13．独立于供应商的地址空间是什么，为什么有一个独立的地址空间会带来一定的好处？

答案：运营商无关的地址空间不是服务提供商 CIDR 地址块的一部分，而是由地区性地址注册机构分配的。这样的地址空间在 AS 连接到多个服务提供商时非常有用。还有一个优点是这样的地址空间是可携带的，即地址拥有者可以在不改变地址的条件下更换服务提供商。

14. 有一个独立于供应商的/21 的地址空间，会带来什么样的问题？

答案：由于一些国家级的服务提供商不接收长于/19 的前缀，所以/21 可能无法广播到整个互联网。

15. 什么是路由策略？

答案：路由策略是一组用来处理出入路由的预定义规则。典型设置路由策略的工具有路由再分发，路由过滤，路由图。

16. 提供到邻居可靠连接的 BGP 下层协议是什么？

答案：BGP 使用 TCP，端口 179。

17. BGP 的四个消息类型是什么，它们是如何使用的？

答案：4 种类型是 Open, Keepalive, Update 和 Notification。Open 消息用作向对端标识 BGP 运行者并开始对等会话。Keepalive 消息用作保持对等连接。Update 消息用作广告路由，Notification 消息向对端报错。

18. 在什么状态下，BGP 对等实体之间可以交换 Update 消息？

答案：只有在 BGP 对等体都在 established 状态时才能交换 Update 消息。

19. 什么是 NLRI?

答案：网络层可达性消息是 BGP Update 中 IP 地址前缀或广告的前缀。

20. 什么是路径属性？

答案：路径属性是 BGP 路由的特点。

21. BGP4 类路径属性是什么？

答案：路径属性有公认必尊属性，公认自选属性，任选可透传属性和任选非透传属性。

22. AS_PATH 属性的目的是什么？

答案：AS_PATH 属性描述了收到的 Update 包在离开源路由器以后穿过的 AS。该属性能够帮助选择 AS 间最端路径以及探测路由环路。

23. AS_PATH 的不同类型是什么？

答案：AS_PATH 的属性有 AS_SEQUENCE，AS_CONFED_SEQUENCE，AS_SET 和 AS_CONFED_SET。AS_SEQUENCE 是一组有序的 AS 号，AS_SET 是一组无序的 AS 号。AS_CONFED_SEQUENCE 和 AS_CONFED_SET 与 AS_SEQUENCE 和 AS_SET 类似，只是存在与 BGP 联盟中。

24. NEXT_HOP 属性的作用是什么？

答案：NEXT_HOP 描述了按照 BGP Update 中广告的 NLRI 将包发往目的地所需要到达的下一跳路由器的 IP 地址。

25. LOCAL_PREF 属性的作用是什么？

答案：在 AS 内如果有多个 IBGP 宣告同一条路由，LOCAL_PREF 用来决定选择的倾向性。该值越大表示越倾向选择该路由。

26. MULTI_EXIT_DISC 属性的作用是什么？

答案：如果两个 AS 间存在多条链路，EBGP 使用 MULTI_EXIT_DISC 来通知邻接 AS，倾向于从那一条链路接收流量。

27. 如果运行 BGP 的路由器发起一个聚合路由，什么属性是有用的？

答案：ATOMIC_AGGREGATE 告诉下游路由器，由于聚合所以会丢失路由信息。

ATOMIC_AGGREGATE 属性标识了发起聚合的路由器。

28. BGP 管理权值是什么？

答案：BGP 管理权重是 Cisco 规定的参数，该参数在路由器内给每条路由赋值。权值越高，越优先选择。该数值是路由器本地决定的，不会广告到对端。

29. 到同一个目的地，有一条 EBGP 路由和一条 IBGP 路由，BGP 路由器会优先选择哪一个？

答案：如果权值、LOCAL_PREF、AS_PATH 长度、ORIGIN 编码和 MED 都一样，路由器会选择 EBGP 的路由。

30. 一个路由器有两条到同一个目的地的路由。路径 A 在 AS_PATH 中有一个值为 300 的 LOCAL_PREF 和 3 个 AS 号。路径 B 在 AS_PATH 中有一个值为 200 的 LOCAL_PREF 和 2 个 AS 号。假设没有其他的不同，该路由器会选择哪条路径？

答案：LOCAL_PREF 比 AS_PATH 由更高的优先级，所以选 A。

31. 什么是路由抑制？

答案：路由抑制是 BGP 路由器为变化的状态指定一个惩罚值的机制。状态变化得越频繁(路由抖动)，累积的惩罚值越大。如果惩罚值超过一定门限，路由被抑制一段时间。这样不稳定的路由能较少地影响 BGP 网络互连。

32. 定义应用于路由抑制的惩罚、抑制界限、重新使用界限和半衰期。

答案：惩罚是路由抑制机制在每次路由状态变化时分配的惩罚值。抑制界限是一个门限值，路由的惩罚值超过门限值时该路由将不被宣告。重新使用界限是一个门限值，当路由惩罚值低于该门限时路由重新被宣告。半衰期是路由所累积的惩罚值减少的速率。每次到达半衰期时，惩罚值被减半。

33. 什么是 IGP 同步，为什么它很重要？

答案：IGP 同步是一个规则，该规则规定除非某穿越路由在 IGP 路由表中出现，路由器不能将该路由宣告到 EBGP 对端。当 BGP 路由器将一个穿越 AS 的包转发给 IGP 路由器而 IGP 路由器没有该路由时，该数据包将被丢弃。

34. 在什么环境下您你可以安全地关闭 IGP 同步功能？

答案：当 AS 内的 IBGP 对等实体完全网状连接或者 AS 不是供穿越的 AS 时，你可以安全地关闭 IGP 同步。

35. BGP 对等组是什么？

答案：BGP 对等组是标识成单一路由器，共享相同的路由策略的一组 BGP 对等体。对等组的配置能够简单地使路由策略应用到组，而不是单一路由器。

36. 什么是 BGP 社团团体？

答案：BGP 社团团体是共享相同路由策略的一组路由器。通过在路由器设置 COMMUNITY 属性来定义社团团体；这些路由器的对端能都辨认 COMMUNITY 属性并应用正确的策略。

37. 什么是路由反射器，什么是路由反射客户，什么是路由反射组簇？

答案：路由反射器类似于路由服务器。允许 IBGP 路由器不用相互连接，只需连接到反射器。从一个对等体发出的广告经过反射器到达所有的对等体。结果是减少了 IBGP 完全连接所需要的大量对等会话。路由反射器本身是路由器，这一点区别于路由服务器。路由反射

器客户是与路由反射器对等连接的 IBGP 路由器。路由反射器簇时路由反射器和它的客户。路由反射器簇中可以存在多个路由反射器，但是簇中的所有客户必须与簇中所有路由反射器建立对等连接。

38．ORIGINATOR_ID 和 CLUSTER_LIST 路径属性的作用是什么？

答案：当使用路由反射器时，ORIGINATOR_ID 和 CLUSTER_LIST 防治路由环路。

39．什么是 BGP 联盟？

答案：为便于管理划分成一组子 AS 的一个大 AS 成为 BGP 联盟。

40．可以在联盟内使用路由反射器吗？

答案：可以。

41．**next_hop_self** 功能的目的是什么？有什么合理的替代该功能的办法吗？

答案：next-hop-self 告诉路由器将从外部对等体收到的路由的 NEXT_HOP 属性改成本机地址。该功能用在 IGP 不知道外部下一跳地址的情况。替代的方式是在外部连路上被动地运行 IGP，这样 IGP 可以获知外部下一跳地址所在的子网。

第 5 章　复习问题答案

1．给出几个原因说明为什么复制的单播对于一个大型的网络不是真正的多播？

答案：单播复制位数据源产生巨大的处理负荷，可能在源端口、数据链路或者连接的路由器产生严重的瓶颈。数据源必须保持状态来记住包复制的地址，必须存在复杂的机制允许成员注册或离开该数据元。最后复制可能产生队列问题，使包间的时延无法接受。

2．哪一个范围的地址保留给 IP 多播？

答案：前 4 个比特是 1110 的 D 类地址。范围是 224.0.0.0-239.255.255.255。

3．一个 D 类前缀可以有多少个子网？

答案：D 类前缀没有子网。多播地址都单个使用，没有子网。

4．路由器处理目的地址在 224.0.0.1-224.0.0.225 范围的包与其他多播地址的包有什么不同？

答案：目的地址在 224.0.0.1-224.0.0.255 范围的包路由器不转发。

5．写出下面 IP 地址的正确以太网 MAC 地址：

(a) 239.187.3.201

(b) 224.18.50.1

(c) 224.0.1.87

答案：

(a) 0100.5E3B.03C9

(b) 0100.5E12.3201

(c) 0100.5E00.0157

6．哪些多播 IP 地址可以用 MAC 地址 0100.5E06.2D54 来表示？

答案：MAC 地址 0100.5E06.2D54 所代表的 32 比特 IP 地址中，第一个字节是 1～15 个数字从 224～239，第二个字节是 134 或者 6，第 3 个字节是 45，最后一个字节是 87。

7．为什么令牌环对传送多播是不好的一种介质？

答案：由于令牌环的帧格式是小端格式的，没有一种简单的方法将多播地址编码到 MAC 地址，所以不适合多播分发。如果使用保留 MAC 地址或广播地址会极大地降低数据链路的效率。

8. 什么是加入时延？

答案：主机发出信号希望加入组到开始接受组流量的时间间隔成为加入时延。

9. 什么是退出时延？

答案：从主机开始离开组到该组将该主机删除的时间间隔成为退出时延。

10. 什么是多播 DR(或者说查询者)？

答案：多播查询者是在子网上询问所连接主机组成员资格的路由器。

11. 什么设备发送 IGMP Query 消息？

答案：路由器发送 IGMP Query 消息。如果了网上存在多个路由器，地址最小的路由器成为询问者。

12. 什么设备发送 IGMP Membership Report 消息？

答案：　主机发送 IGMP Membership Report 消息

13. 如何使用 IGMP Membership Report 消息？

答案：IGMP Membership Report 消息由主机发送，通知路由器该主机希望加入组。

14. General IGMP Query 与 Group-Specific IGMP Query 功能上有什么具别？

答案：路由器发送 General IGMP Query 来发现任意组和所有组的成员。发送 Group-Specific IGMP Query 来发现特定组的成员，通常在收到离开组请求消息之后。

15. IGMPv2 与 IGMPv1 兼容吗？

答案：尽管如果网络上有 IGMPv1 路由器，则应当将所由路由器都设置成 IGMPv1，IGMPv2 还是在很大程度上兼容 IGMPv1。

16. IGMP 的 IP 协议号是多少？

答案：IGMP 使用协议号 2。

17. Cisco 组员资格协议(CGMP)的目的是什么？

答案：CGMP 是以太网交换机使用的协议，用来发现组成员连接在哪个端口上从而避免向所有端口转发多播帧。

18. IP 监听比 CGMP 有什么优点？可能的缺点是什么？

答案：与 CGMP 不一样，IP 监听不是运营者专用的，可以用在多厂商网络上。缺陷是如果交换机仅软件支持 IP 偷窥，运行时交换机性能会受影响。

19. 什么设备发送 CGMP 消息：路由器、以太网交换机，还是两者都可以？

答案：只有路由器发送 CGMP 消息，交换机倾听 CGMP 消息。

20. 什么是反向路径前转(RPF)？

答案：RPF 识别 IP 多播路由的基本转发机制。由于路由器只能发现到源的最短路径，而不是到目的地的最短路径。当多播包向目的地转发(更精确地说，离开数据源)时，他们转发在最短路径的反方向上。

21. 多少主机构成密集模式，多少主机构成稀疏模式？

答案：密集与稀疏拓扑没有数量区别。

22. 显式加入与隐式加入相比，最主要的优点是什么？

答案：好处在于路由器不需要维护不是任何组成员上游的接口状态。

23．基于源的多播树与共享多播树结构上最主要的区别是什么？

答案：基于源的树的树根在源子网或源路由器。共享树树根在公共集合点或核，可以由多个源共享。

24．什么是多播的限制？

答案：多播范围限制是在指定拓扑区域限制特定多播包范围的行动。

25．IP 多播限制的两种方法是什么？

答案：IP 多播范围限制两种方式是 TTL 范围限制和管理范围限制。

26．从多播路由器的角度看，什么是上游，什么是下游？

答案：上游是到多播源的发项，下游是离开多播源的方向。

27．什么是 RPF 检查

答案：RPF 检查是用来确认从特定源的多播包到达上游接口，而不是其他接口。

28．什么是剪除，什么是接入？

答案：剪除是将路由器从多播树中删除的行为。接入是将路由器加入到多播树的行为。

29．什么是剪除生存期？当剪除生存期结束会发生什么？

答案：剪除生存期用于隐性加入协议，是路由器将接口保持在剪除状态的总时间。当剪除生存期超时后，路由器重新向接口转发包直到邻近的下行再次发出剪除请求。

30．什么是路由依赖？DVMPR 如何表示路由依赖？

答案：路由依赖是路由器为特定组转发包对上游邻居的依赖。DVMRP 路由器信号适用反向毒化路由指示路由依赖，这时量度是到源的跳数加上 32。

31．DVMRP 是密集模式的还是稀疏模式的协议？

答案：DVMRP 是密集模式协议。

32．MOSPF 是密集模式的还是稀疏模式的协议？

答案：MOSPF 是密集模式协议。

33．MOSPF 专有的 LSA 名称和类型号是多少？

答案：MOSPF 的 LSA 专用的名字是组成员资格 LSA，类型为 6。

34．MOSPF 能不能与不支持 MOSPF 路由器建立邻接的关系？

答案：可以。只有数据库描述包中 MC 比特置位的邻居才交换组成员资格 LSA。

35．定义下列 MOSPF 路由器类型

(a) 区域间多播转发器

(b) AS 间多播转发器

(c) 通配多播接收器

答案：

(a) 区域间多播转发器转发区域间 IP 多播包，与单播 OSPF ABR 类似。

(b) AS 间多播转发器转发 MOSPF 外 IP 多播包，与单播 OSPF ASBR 类似。

(c) 通配多播接收器是一个路由器，所有的多播包将转发到该路由器。

36．CBT 是密集模式的还是稀疏模式的协议？

答案：CBT 是稀疏模式协议。

37．什么是 CBT 父路由器和 CBT 子路由器？

答案：CBT 父路由器是上游路由器，CBT 子路由器是下游路由器。

38. 描述 CBT DR 从源向核心传送包的两种方法，在何种情况下用哪种方法？

答案：如果只连的数据源是成员数据源，包将直接在树上转发；如果不是，将建立一条到核的隧道，包沿隧道转发。

39. 什么是 PIM 剪除覆盖？

答案：在多播访问网络上向上由路由器发送加入消息，用作取消同一网端上另一路由器发出的取消请求称为 PIM 剪除覆盖。

40. 什么是 PIM 前转器？它怎么选出来的？

答案：当多个上游路由器连接在同一个多播网络并接受同一个组的数据时，向网络转发包的路由器称为 PIM 前转器。前转器由 Assert 消息中广告的管理决定距离选举，最低的距离当选。如果该距离相同，使用量度最小的路由器。如果量度相同，使用较小的 IP 地址打破平局。

41. PIM 用什么标准来选出 DR？

答案：使用最大的 IP 地址的路由器(按照 PIM Hello 消息)作为 DR。

42. 什么是 PIM SPT？什么是 PIM RPT？

答案：最短路径数是基于源的树。SPT 是共享数，跟在一个指定点。

43. Cisco 路由器自动发现 PIM-SM RP 的两种方法是什么？

答案：可以使用 Auto-RP 或者自举协议自动发现 PIM-SM RP。

44. 问题 43 中的机制中哪一个可以用在多厂商路由器的拓扑中？

答案：其他厂商不支持 Auto-RP，所以只能使用自举。

45. 什么是 C-RP？

答案：C-RP 是候选 RP，是指一个可以被选举成所有组或者指定组 RP 的路由器。

46. 什么是 BSR？

答案：当使用自举协议时，通过 RP-Set 在全 PIM-SM 范围内广播 C_RP。

47. 什么是 RP 映射代理？

答案：使用 Auto-RP 时，由 RP 映射代理广告组到 RP 的映射。

48. (S,G)多播路由条目与(*,G)多播路由条目的区别是什么

答案：(S,G)是关于 SPT，(*,G)关于 RPT。

49. CBT 在源与核心间的双向树，与 PIM-SM 在 RP 和源间单向树不同，其最大缺陷是什么？

答案：由于没有明显的上游与下游，在一个双向树中很难保证一条无环的路径。

50. 什么是 PIM-SM 源的注册？

答案：源注册是一种机制，使用这种机制路由器在 PIM 注册消息中将包从多播源传输到 RP。如果由大量数据从源发送，则由 RP 构造一个 SPT 然后发送注册停止。

51. 什么时候 Cisco 路由器从 PIM-SM RPT 切换到 SPT？

答案：Cisco 路由器在两种情况下游 PIM-SM RPT 变成 SPT：(1)在 RPT 收到给特殊(S,G)的第一个包；(2)(S,G)收到包的速率超过了由命令 **ip pim spt-threshold** 配置的门限。

第 7 章　复习问题答案

1．在“多播范围限制”一章中，有一个配置举例是通过管理的范围限制。在边界接口 E0 阻塞所有组织内部的包(目的地址前缀匹配 239.192.0.0/14)，允许全局范围包通过。组地址是 224.0.0.50 的包是否能穿过边界？

答案：只有在本地路由器发出目的地址是 224.0.0.50 的包时，该包才能穿过边界。尽管访问接表 10 允许 224.0.0.50，由于该地址是本地链路范围，不会转发到下一跳路由器。

2．在配置了运行 PIM 的点到点接口和多路访问接口上 Cisco IOS 软件如何处理 DVMRP 剪除消息？

答案：DVMRP 剪除消息在多路访问接口上被忽略，在点到点接口上正常处理。

3．为什么 Cisco IOS 软件接收 DVMRP 消息却不应答？

答案：接收 DVMRP 消息是探测 DVMRP 邻居所必需的。如果发送探测消息，在多路访问接口上的其他 Cisco 路由器会误以为该路由器只有 DVMRP 能力。

4．PIM (*,*,RP)条目是什么？

答案：(*，*，RP)条目到 PIM 多播边缘路由器。Cisco IOS 不支持 MBR。

5．多协议 BGP(MBGP)与普通 BGP 有什么区别？

答案：MBGP 扩展了两个路由属性：MP_REACH_NLRI 和 MP_UNREACH_NLRI。

6．什么是 MBGP AFI？

答案：AFI 是地址组标识。当 MBGP 用在多播时，AFI 总是设置成 1(IPv4)，由于 AFI 指示相关的 NLRI 使用作单播、多播还是两者皆有。

7．下边的说法对吗？MSDP 在不同 PIM 域的 RP 间携带多播源和组成员信息。

答案：错误。MSDP 只传达多播源信息，不传达组内成员信息。

8．MSDP 的传输协议是什么？

答案：MSDP 使用 TCP 端口号 639。

9．什么是 MSDP SA 消息？

答案：SA 是数据源活跃激活消息。当数据源的 DR 使用 RP 注册时，如果 RP 运行 MSDP 则在 SA 消息中向对端广告(S,G)对。

10．MSDP RP 如何确认 RPF 接口上是否收到 SA？

答案：MSDP RP 检查下一跳数据库(先找 MBGP，然后单播 BGP)查找正确的上游接口。

11．什么是 SA 缓存？

答案：SA 缓存是存储在 SA 中消息学到的(S,G)状态信息。SA 缓存使用路由器的内存占用换取减少加入时延。缺省时，在 Cisco IOS 软件禁止 SA 缓存。

12．不使用 SA 缓存时能否降低接入时延？

答案：是的。如果 MSDP 对端已缓存，您可以配置 RP 在收到加入消息时使用 SA 请求消息从对端请求(S,G)信息。

第 8 章　复习问题答案

1．对 60 比特前缀的地址 200A 0000 0000 0C00 0000 0000 0000 0000 下面哪几种表示是正确的？

A．200A:0000:0000:0C/60

B．200A::0C00:0:0:0:0/60

C．200A:0000:0000:0C00::/60

D．200A::0c00::/60

E．200A:0:0:C00::/60

F．200A::0C/60

答案：B、C 和 E。A 地址不完整。D 中有两个：：引起混淆。F 扩展出的地址不对。

2．地址 0:0:0:0:0:0:0:0 用在哪里？

答案：这是非指定地址。用来表示地址缺席。如果用在源地址，表示该接口还没有分配地址。该主机试图发现它想用的地址是否已被其他节点使用。

3．您配置连接到 IPv6 公网的站点边缘路由器，广播所有的内部网络，包括 FEC0:0020:0:0100::/56。你接到网络管理员愤怒的电话。出什么问题了？

答案：FEC0:0020:0:0100::/56 是站点内部地址。该地址不能被广播到站点边界以外。

4．从源到目的地的路径中，哪一个扩展头各节点都处理？

答案：Hop-by-Hop 扩展头。

5．使用哪一种扩展头来指定到达目的地之前访问一组路由器，并由这些路由器处理该头？

答案：路由头后面跟随的目的地选项头。

6．如果路由器收到一个包大于出口链路的 MTU。路由器是否将包分片后转发？

答案：不会。路由器会丢弃该包，并向数据源发送数据包过大的 ICMP 消息。数据源使用这些 ICMP 消息来发现路径的 MTU。对数据分片是数据源的责任。

7．在路由器广告中，管理比特设置代表什么？

答案：路由器向链路上所有的主机发送 RA。如果设置管理比特，主机由基于状态的地址服务器获得地址。

8．如果路由器在 RA 中广告前缀消息，这些消息如何使用？

答案：RA 中的前缀信息告诉主机，哪一个前缀可以使用和/或自动配置地址时使用哪一个前缀。

9．主机中的 IP 地址可以在那两种状态？这两种状态如何使用？

答案：倾向与不倾向。倾向使用的地址可以用作发起 IP 会话。如果存在倾向的地址，不倾向的地址应当只用于维持现有连接，不应发起新的连接。

10．路由器在 RA 中要求主机替换使用某个前缀发起会话时广告什么消息？

答案：发送一个值为 0 的 Valid Lifetime 或者 Preffered Lifetime。Valid Lifetime 值为 0 表使该前缀不再有效。Preffered Lifetime 为 0 表示该前缀为不倾向前缀。

11．如果节点有一个邻居状态是 DELAY，能否向该邻居发包？

答案：可以。状态为 DELAY 的邻居没有被确认可达，但是节点可以向缓存的该邻居的链路层地址发送数据。

12．如果主机上没有运行任何路由协议，路由器通过缺省路由器向源端节点发数据。缺省路由器故障时，节点会不会向故障路由器发数据直到 TCP 连接中断？

答案：不会。邻居不可达处理、缺省路由器列表和地址解析处理能帮助发现不能工作的路由器并找到一个新的路由器作为替代。

13．多播包的范围是什么？如何使用？

答案：节点、链路、站点、组织和全球。这些范围只用来限制多播组的含义并控制多播包的传输距离。

14．Cisco 路由器使能 IPv6 路由的命令是什么？

答案：**ipv6 unicast routing**。

15．在接口上使能 IPv6 的接口子命令是什么？

答案：**ipv6 enable**，**ipv6 address address prefix** *[eui-64]*

16．使能 RIPng 过程的命令是什么？

答案：接口子命令：**ipv6 rip process-name enabled**。

17．在邻居间如何使能 BGP-for-IPv6 过程？

答案：

```
router bgp local-AS
   neighbor neighbor-ipv6-address remote-as remote-as-number
   !
   address-family ipv6
   neighbor neighbor-ipv6-address activate
```

第 9 章　复习问题

1．解释 SNMP 轮询与 trap 间的不同。

答案：管理工作站向路由器请求信息称为轮询路由器。路由器发送关于所发生事件的未经请求的信息称为发送 trap。

2．如果希望将 severity 级消息记录日志成为 error，那些其他级别的消息会被记录？

答案：Emergencise，Alerts 和 Critical。

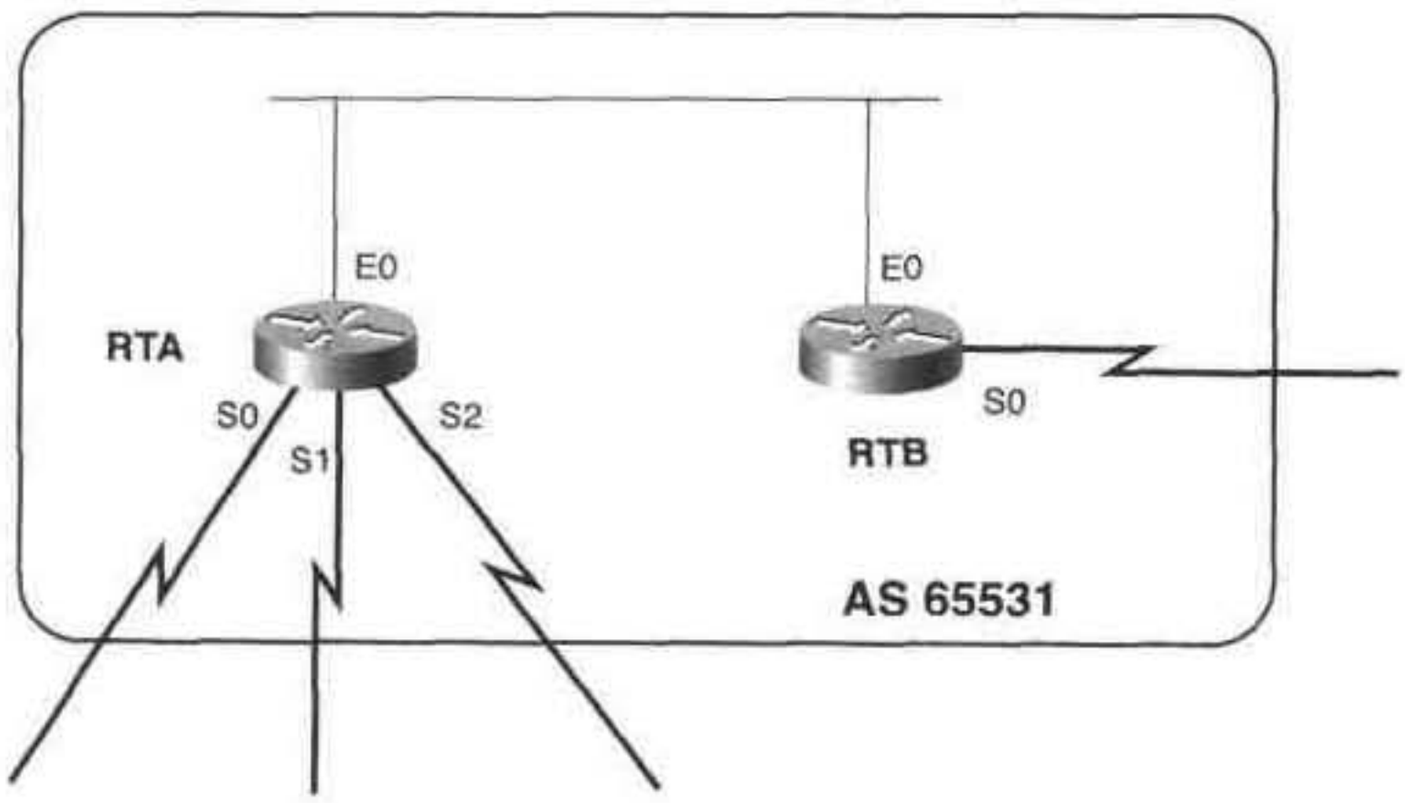
3．如果您浏览路由器接口发现异常的流量模式。一般情况下流量是带内的，现在由许多带外流量。您如何最快地确定流量的源和目的地址？

答案：在接口上使能 IP 记账。重复使用 show ip accounting 来发现所发送通信流的源和目的地址对。

附录 E　配置练习的答案

第 1 章　配置练习的答案

1．图 1-14 中 AS 65531 是一个核心 AS。



RTA 接口	地址
E0	192.168.1.1/24
S0	192.168.2.1/24
S1	192.168.3.1/24
S2	192.168.4.1/24

RTB 接口	地址
E0	192.168.1.2/24
S0	192.168.5.1/24

图 1-14　配置练习 1 的网络

根据下面的条件，在 RTA 和 RTB 上配置 EGP：

AS 的内部链路不能公布给任何一个外部邻居。

RTA 将与它的 S1 接口相连的网络公布给 RTB；除此以外，在 RTA 和 RTB 之间没有公布其他 AS 之间的链路。

除了从其他 AS 学习网络以外，RTA 和 RTB 向它们的外部邻居公布了一条缺省路由。没有网关向内部的邻居公布缺省路由。

答案：RTA 与 RTB 的配置如下。

```
hostname RTA
!
interface Ethernet0
 ip address 192.168.1.1 255.255.255.0
!
```

```
interface Serial0
 ip address 192.168.2.1 255.255.255.0
!
interface Serial1
 ip address 192.168.3.1 255.255.255.0
!
interface Serial2
 ip address 192.168.4.1 255.255.255.0
!
autonomous-system 65531
!
router egp 0
 network 192.168.3.0
 neighbor 192.168.1.2
 neighbor any
 default-information originate
 distribute-list 1 out Ethernet0
!
access-list 1 deny   0.0.0.0
access-list 1 permit any
```

```
hostname RTB
!
interface Ethernet0
 ip address 192.168.1.2 255.255.255.0
!
interface Serial0
 ip address 192.168.5.1 255.255.255.0
!
autonomous-system 65531
!
router egp 0
 neighbor any
 default-information originate
 distribute-list 1 out Ethernet0
!
access-list 1 deny   0.0.0.0
access-list 1 permit any
```

2. 例1-26给出了图1-15中RTC的路由表。

例1-26 图1-15中RTC的路由表

```
RTC#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
Gateway of last resort is not set

I    192.168.105.0 [100/8976] via 192.168.6.2, 00:01:00, Serial1
I    192.168.110.0 [100/8976] via 192.168.6.2, 00:01:00, Serial1
I    192.168.100.0 [100/8976] via 192.168.10.2, 00:01:00, Serial2
I    192.168.120.0 [100/8976] via 192.168.10.2, 00:01:01, Serial2
C    192.168.2.0 is directly connected, Serial0
C    192.168.6.0 is directly connected, Serial1
C    192.168.10.0 is directly connected, Serial2
RTC#
```

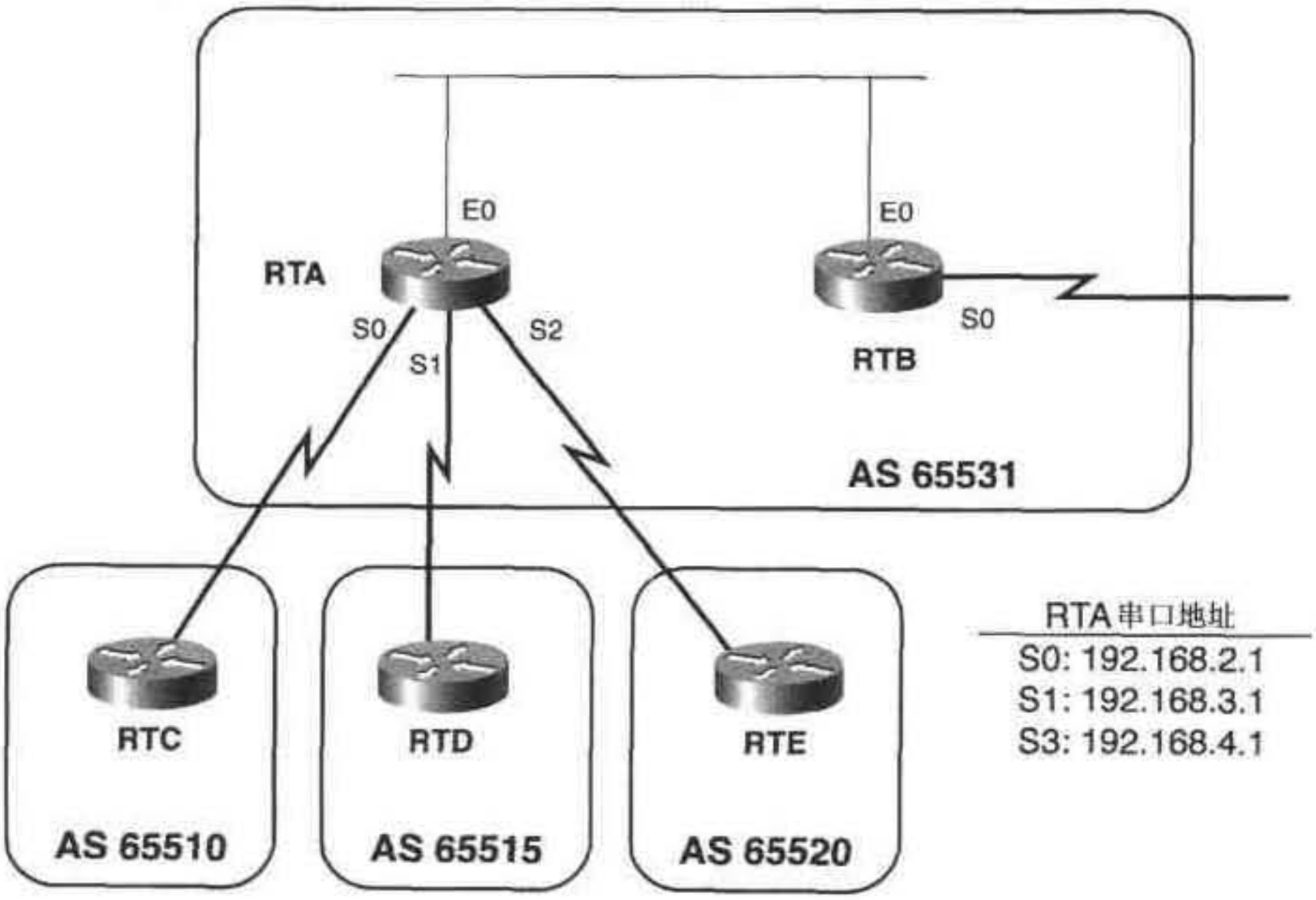


图 I-15　配置练习 2 的网络图

通过再分发，配置 RTC，让它将所有学习到的 EGP 网络公布给 AS 65510，将除了 192.168.105.0 之外的所有内部网络公布给核心 AS。保证 AS 65510 内部的网络不会通过 EGP 公布回来从而防止路由环路。这个配置的进程 ID 与本地 AS 号相同。

答案：相关 RTC 的配置如下。

```
autonomous-system 65510
!
router igrp 65510
 redistribute egp 65531 metric 1544 100 255 1 1500
 network 192.168.6.0
!
router egp 65531
 redistribute igrp 65510
 neighbor 192.168.2.1
 distribute-list 10 out Serial0
 distribute-list 20 in Serial0
!
access-list 10 deny   192.168.105.0
access-list 10 permit any
access-list 20 deny   192.168.105.0
access-list 20 deny   192.168.110.0
access-list 20 deny   192.168.100.0
access-list 20 deny   192.168.120.0
access-list 20 deny   192.168.10.0
access-list 20 deny   192.168.6.0
access-list 20 permit any
```

注意到量度没有再分发到 EGP 中；EGP 加上了默认的量度 3。在这个练习中 distribute-list 这个命令用来过滤路由，虽然用路由图也可以达到同样的目的。这个过滤器有

趣的一点是如果内部网络地址在收到的 EGP 更新消息里，这些地址会被过滤掉。即使 192.168.105.0 没有宣告到 AS 外，这个地址也包括在访问表 20 中。这保证了不会有网络用其他方法找到到 EGP 域内的路由，然后再转发到 AS 65510 可能性。它也保证了不会有重覆的网络地址进入 AS。

3. 例 1-27 给出了图 1-15 中 RTD 的路由表。

例 1-27　图 1-15 中 RTD 的路由表

```
RTD#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is not set

C    192.168.3.0 is directly connected, Serial0
C    192.168.7.0 is directly connected, Serial1
R    192.168.230.0 [120/1] via 192.168.7.2, 00:00:14, Serial1
R    192.168.200.0 [120/2] via 192.168.7.2, 00:00:15, Serial1
R    192.168.220.0 [120/1] via 192.168.7.2, 00:00:15, Serial1
R    192.168.210.0 [120/2] via 192.168.7.2, 00:00:15, Serial1
RTD#
```

用下面的参数来配置 RTD：

只将 192.168.220.0 和 192.168.230.0 公布给 AS 65531。

没有路由协议再分发给 EGP。

将 EGP 再分发给 AS 65515 的 IGP。

将 192.168.3.0 公布给 AS 65515，它的度量是 1。

将来自 RTC 的 192.168.100.0 公布给 AS 65515，度量是 1。

将来自 RTC 的 192.168.120.0 公布给 AS 65515，度量是 3。

所有其他的路由公布给 AS 65515，度量是 5。

答案：RTD 相关的配置如下。

```
autonomous-system 65515
!
router rip
 redistribute egp 65531 route-map EXTERNAL
 network 192.168.7.0
 network 192.168.3.0
 default-metric 5
!
router egp 65531
 network 192.168.220.0
 network 192.168.230.0
 neighbor 192.168.3.1
!
access-list 10 permit 192.168.100.0
access-list 20 permit 192.168.120.0
access-list 30 permit any
!
```

```
route-map EXTERNAL permit 10
 match ip address 10
 set metric 1
!
route-map EXTERNAL permit 20
 match ip address 20
 set metric 3
!
route-map EXTERNAL permit 30
 match ip address 30
```

4. 例 1-28 给出了图 1-15 中 RTE 的路由表。

例 1-28　图 1-15 中 RTE 的路由表

```
RTE#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR


Gateway of last resort is not set

O    192.168.125.0/28 [110/74] via 192.168.130.6, 00:01:03, Serial1
C    192.168.4.0/24 is directly connected, Serial0
     192.168.225.0/28 is subnetted, 1 subnets
O E2    192.168.225.160 [110/50] via 192.168.130.18, 00:01:04, Ethernet0
     192.168.215.0/24 is variably subnetted, 3 subnets, 3 masks
O       192.168.215.161/32 [110/65] via 192.168.130.6, 00:01:04, Serial1
O E2    192.168.215.192/26 [110/50] via 192.168.130.18, 00:01:04, Ethernet0
O E1    192.168.215.96/28 [110/164] via 192.168.130.6, 00:01:04, Serial1
     192.168.130.0/24 is variably subnetted, 7 subnets, 4 masks
D       192.168.131.192/27 [90/2195456] via 192.168.130.6, 00:16:49, Serial1
D       192.168.131.96/27 [90/409600] via 192.168.130.18, 00:16:49, Ethernet0
O       192.168.131.97/32 [110/11] via 192.168.130.18, 00:01:05, Ethernet0
D       192.168.131.64/27 [90/409600] via 192.168.130.18, 00:15:01, Ethernet0
D       192.168.131.8/30 [90/2195456] via 192.168.130.6, 00:16:49, Serial1
C       192.168.131.4/30 is directly connected, Serial1
C       192.168.131.16/28 is directly connected, Ethernet0
RTE#
```

用下面的参数配置 RTE：

没有 IGP 再分发给 EGP。

EGP 不会再分发给任何的 IGP。

将 AS 65520 内部所有的网络都公布给 AS 65531。

AS 65520 内部所有的路由器都能够向由 RTA 公布的网络转发数据包。

所有进程 ID 都与 AS 号相同。

所有 OSPF 的接口都在区域 0 内。

答案：RTE 的相关配置如下。

```
autonomous-system 65520
!
router eigrp 65520
 redistribute static
 network 192.168.130.0
 default-metric 1000 100 255 1 1500
 no auto-summary
!
router ospf 65520
 redistribute static metric 10 subnets
 network 192.168.130.4 0.0.0.3 area 0
 network 192.168.130.16 0.0.0.15 area 0
!
router egp 65531
 network 192.168.125.0
 network 192.168.131.0
 network 192.168.215.0
 network 192.168.225.0
 neighbor 192.168.4.1
!
ip route 0.0.0.0 0.0.0.0 192.168.4.1
```

5. 在图 I-16 中，在前一个练习的网络中加入了 AS 65525。RTF 以太网接口的 IP 地址为 192.168.1.3/24。

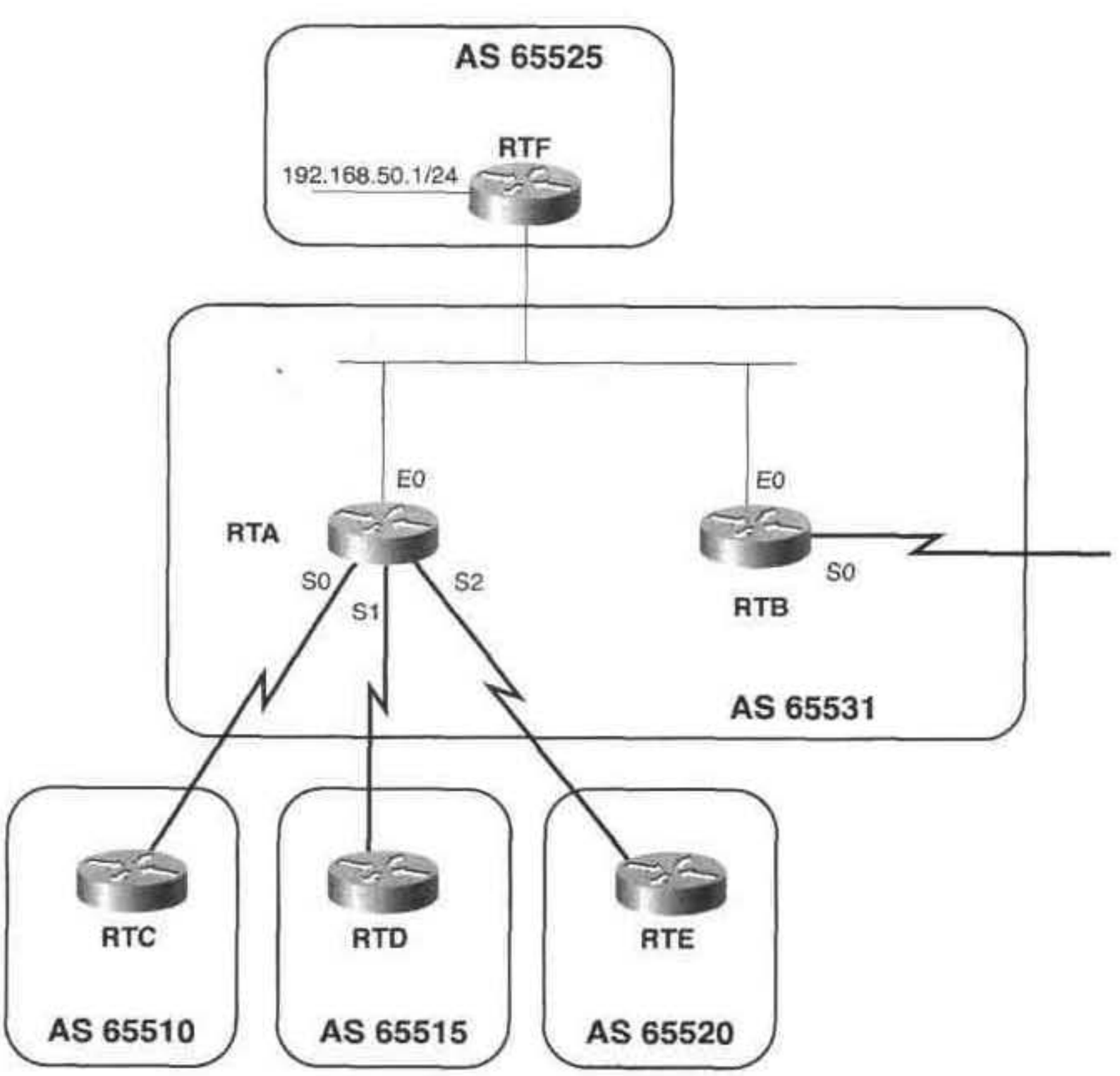


图 I-16 配置练习 5 的网络

配置新加入的路由器，让它只是 RTB 的对端，同时对配置进行必要的修改从而使它支持第三方邻居。

答案：RTF 与 RTB 相关的配置如下。

```
RTF
autonomous-system 65525
!
router egp 65531
 network 192.168.50.0
 neighbor 192.168.1.2
```

```
RTB
autonomous-system 65531
!
router egp 0
 neighbor 192.168.1.1
 neighbor 192.168.1.1 third-party 192.168.1.3 external
 neighbor 192.168.1.3
 neighbor 192.168.1.3 third-party 192.168.1.1
 neighbor any
 default-information originate
 distribute-list 1 out Ethernet0
!
access-list 1 deny   0.0.0.0
access-list 1 permit any
```

第 3 章　配置练习的答案

表 3-4 给出了用于配置练习 1~13 的路由器和地址。

表 3-4　　用于配置练习 1~13 的路由器/地址

自治系统	路由器	接口	IP 地址/掩码
1	R1	L0	10.255.255.1/32
		S0	192.168.100.1/30
		E0	192.168.100.5/30
		E1	192.168 100.13/30
	R2	L0	10.255.255.2/32
		S0	192.168.100.9/30
		S1	192.168.100.57/30
		E0	192.168.100.6/30
		E1	192.168.100.17/30
	R3	L0	10.255.255.3/32

续表

自治系统	路由器	接口	IP地址/掩码
		S0	192.168.100.25/30
		E0	192.168.100.18/30
		E1	192.168.100.21/30
	R4	L0	10.255.255.4/32
		S0	192.168.100.29/30
		S1	192.168.100.33/30
		E0	192.168.100.22/30
		E1	192.168.100.14/30
2	R5	S0	192.168.100.2/30
		E0	192.168.1.129/26
	R6	S0	192.168.100.10/30
		E0	192.168.1.130/26
3	R7	L0	10.255.255.7/32
		S0	192.168.100.26/30
		S1	192.168.100.41/30
		E0	192.168.100.37/30
		E1	172.16.1.1/24
4	R8	L0	10.255.255.8/32
		S0	192.168.100.30/30
		S1	192.168.100.45/30
		E0	192.168.100.38/30
		E1	172.16.2.1/24
5	R9	L0	10.255.255.9/32
		S0	192.168.100.42/30
		E0	192.168.9.1/24
		E1	192.168.150.1/24
	R10	L0	10.255.255.10/32
		S0	192.168.100.46/30

续表

自治系统	路由器	接口	IP 地址/掩码
		E0	192.168.10.1/24
		E1	192.168.100.53/30
		E2	192.168.150.2/24
	R11	L0	10.255.255.11/32
		S0	192.168.100.34/30
		E0	192.168.100.54/30
		E1	192.168.11.1/24
6	R12	L0	192.168.255.1/32
		S0	192.168.100.58/30
		E0	192.168.16.83/27

表 3-4 列出了配置练习 1~13 所需要的 AS、路由器、接口以及地址，给出了路由器的所有接口。对于每个练习来讲，如果表中显示出路由器有 Loopback 接口，那么该接口一定是所有 IBGP 连接的源。除非在练习中有明确地规定，否则通常应该在物理接口地址间建立 EBGP 连接。提示：在做练习之前，先根据表中给出的子网情况，把网络图画出来。

1．表 3-4 中的 AS 1 是一个转接 AS，它采用的 IGP 是 OSPF。区域 0 跨越整个 AS。没有把 AS 内部的网络公布到 AS 外。运行 EBGP 的子网不能公布给 AS 1。写出 AS 1 内路由器的 BGP 配置，将所有的内部邻居放到一个名为 LOCAL 的对等组中。只有 R3 是在环回接口上执行 EBGP 对等。通过口令 **ExeRcise1** 来验证所有的 IBGP 连接。

答案：

R1

```
router ospf 1
 network 10.255.255.1 0.0.0.0 area 0
 network 192.168.100.5 0.0.0.0 area 0
 network 192.168.100.13 0.0.0.0 area 0
!
router bgp 1
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 1
 neighbor LOCAL password 7 153713693609223720B2
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.2 peer-group LOCAL
 neighbor 10.255.255.3 peer-group LOCAL
 neighbor 10.255.255.4 peer-group LOCAL
 neighbor 192.168.100.2 remote-as 2
```

R2

```
router ospf 1
 network 10.255.255.2 0.0.0.0 area 0
```

```
 network 192.168.100.6 0.0.0.0 area 0
 network 192.168.100.17 0.0.0.0 area 0
!
router bgp 1
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 1
 neighbor LOCAL password 7 153713093609223720G2
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.1 peer-group LOCAL
 neighbor 10.255.255.3 peer-group LOCAL
 neighbor 10.255.255.4 peer-group LOCAL
 neighbor 192.168.100.10 remote-as 2
 neighbor 192.168.100.58 remote-as 6
```

R3

```
router ospf 1
 network 10.255.255.3 0.0.0.0 area 0
 network 192.168.100.18 0.0.0.0 area 0
 network 192.168.100.21 0.0.0.0 area 0
!
router bgp 1
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 1
 neighbor LOCAL password 7 153713093609223720G2
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor REMOTE peer-group
 neighbor REMOTE ebgp-multihop 2
 neighbor REMOTE update-source Loopback0
 neighbor 10.255.255.1 peer-group LOCAL
 neighbor 10.255.255.2 peer-group LOCAL
 neighbor 10.255.255.4 peer-group LOCAL
 neighbor 192.168.100.26 peer-group REMOTE
 neighbor 192.168.100.26 remote-as 3
```

R4

```
router ospf 1
 network 10.255.255.4 0.0.0.0 area 0
 network 192.168.100.14 0.0.0.0 area 0
 network 192.168.100.22 0.0.0.0 area 0
!
router bgp 1
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 1
 neighbor LOCAL password 7 153713093609223720G2
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.1 peer-group LOCAL
 neighbor 10.255.255.2 peer-group LOCAL
 neighbor 10.255.255.4 peer-group LOCAL
 neighbor 192.168.100.30 remote-as 4
 neighbor 192.168.100.34 remote-as 5
```

2. 表 3-4 中的 AS 2 是一个末梢 AS(非转接)，它的 IGP 是 EIGRP。对 AS 2 内的路由器进行配置，使它们对任何一个外部对等都运行 EBGP 而且它们会将任何一个 EIGRP 路由再分发给 BGP。将学习到的 BGP 路由再分发给 EIGRP。使用必要的过滤手段阻止再分发不正确的路由。

答案：

R5

```
router eigrp 2
 redistribute bgp 2 route-map External_Routes metric 10000 100 255 1 1500
 passive-interface Serial0
 network 192.168.1.0
 network 192.168.100.0
 no auto-summary
!
router bgp 2
 redistribute eigrp 2 route-map Internal_Routes
 neighbor 192.168.100.1 remote-as 1
!
ip as-path access-list 1 deny _2_
ip as-path access-list 1 permit .*
ip as-path access-list 2 permit ^$
!
route-map External_Routes permit 10
 match as-path 1
!
route-map Internal_Routes permit 10
 match as-path 2
```

R6

```
router eigrp 2
 redistribute bgp 2 route-map External_Routes metric 10000 100 255 1 1500
 passive-interface Serial0
 network 192.168.1.0
 network 192.168.100.0
 no auto-summary
!
router bgp 2
 redistribute eigrp 2 route-map Internal_Routes
 neighbor 192.168.100.9 remote-as 1
!
ip as-path access-list 1 deny _2_
ip as-path access-list 1 permit .*
ip as-path access-list 2 permit ^$
!
route-map External_Routes permit 10
 match as-path 1
!
route-map Internal_Routes permit 10
 match as-path 2
```

3．网络 192.168.1.0、192.168.2.0、192.168.3.0、192.168.4.0 和 192.168.5.0 在 AS 2 内，这个 AS 的管理者希望相邻 AS 在向 192.168.1.0 和 192.168.3.0 发送业务量时，优选 R5；在向 192.168.2.0 和 192.168.4.0 发送业务量时，优选 R6。在这种情况下，非优选链路作为优选链路的备份链路。192.168.5.0 是一个专用网络，因此不能将它公布给任何 EBGP 对端。修改练习 2 中的配置让 AS 2 执行这个策略。

答案：注意到在这个配置中，练习 2 中的 AS_PATH 过滤器仍然存在。虽然对于访问表中过滤的前缀并不是全都需要，在实际网络中它们可以对宣告的错误路由进行额外的保险。

R5

```
router eigrp 2
 redistribute bgp 2 route-map External_Routes metric 10000 100 255 1 1500
 passive-interface Serial0
 network 192.168.1.0
```

```
 network 192.168.100.0
 no auto-summary
!
router bgp 2
 redistribute eigrp 2 route-map Internal_Routes
 neighbor 192.168.100.1 remote-as 1
!
ip as-path access-list 1 deny _2_
ip as-path access-list 1 permit .*
ip as-path access-list 2 permit ^$
!
access-list 1 permit 192.168.1.0
access-list 1 permit 192.168.3.0
access-list 2 permit 192.168.2.0
access-list 2 permit 192.168.4.0
!
route-map External_Routes permit 10
 match as-path 1
!
route-map Internal_Routes permit 10
 match ip address 1
 match as-path 2
 set metric 50
!
route-map Internal_Routes permit 20
 match ip address 2
 match as-path 2
 set metric 150
```

R6

```
router eigrp 2
 redistribute bgp 2 route-map External_Routes metric 10000 100 255 1 1500
 passive-interface Serial0
 network 192.168.1.0
 network 192.168.100.0
 no auto-summary
!
router bgp 2
 redistribute eigrp 2 route-map Internal_Routes
 neighbor 192.168.100.9 remote-as 1
!
ip as-path access-list 1 deny _2_
ip as-path access-list 1 permit .*
ip as-path access-list 2 permit ^$
!
access-list 1 permit 192.168.2.0
access-list 1 permit 192.168.4.0
access-list 2 permit 192.168.1.0
access-list 2 permit 192.168.3.0
!
route-map External_Routes permit 10
 match as-path 1
!
route-map Internal_Routes permit 10
 match ip address 1
 match as-path 2
 set metric 50
!
route-map Internal_Routes permit 20
 match ip address 2
 match as-path 2
```

```
 set metric 150
```

4. 配置 R5 和 R6 的 EBGP 邻居，让它们公布一条缺省路由给 AS 2，不再公布其他路由。

答案：

```
R1
router ospf 1
 network 10.255.255.1 0.0.0.0 area 0
 network 192.168.100.5 0.0.0.0 area 0
 network 192.168.100.13 0.0.0.0 area 0
!
router bgp 1
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 1
 neighbor LOCAL password 7 153713093609223720 62
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.2 peer-group LOCAL
 neighbor 10.255.255.3 peer-group LOCAL
 neighbor 10.255.255.4 peer-group LOCAL
 neighbor 192.168.100.2 remote-as 2
 neighbor 192.168.100.2 default-originate
 neighbor 192.168.100.2 distribute-list 1 out
!
access-list 1 permit 0.0.0.0
access-list 1 deny any
```

```
R2
router ospf 1
 network 10.255.255.2 0.0.0.0 area 0
 network 192.168.100.6 0.0.0.0 area 0
 network 192.168.100.17 0.0.0.0 area 0
!
router bgp 1
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 1
 neighbor LOCAL password 7 153713093609223720 62
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.1 peer-group LOCAL
 neighbor 10.255.255.3 peer-group LOCAL
 neighbor 10.255.255.4 peer-group LOCAL
 neighbor 192.168.100.10 remote-as 2
 neighbor 192.168.100.10 default-originate
 neighbor 192.168.100.10.distribute-list 1 out
 neighbor 192.168.100.58 remote-as 6
!
access-list 1 permit 0.0.0.0
access-list 1 deny any
```

5. AS 2 的相邻 AS 的管理者对练习 2 中的策略不是完全同意。他希望他所有的路由器在向 192.168.3.0 发送业务量时，都选择 R6，而把 R5 作为备份；所有发送给 192.168.4.0 的业务量都经过 R5，而把 R6 作为备份。练习 2 中的其他策略是可以接受的。写出执行这个策略的配置。

答案：记住 LOCAL_PREF 在 BGP 判断过程中先于 MED 前考虑。因此，在 AS 2 中的相应路由器上改变默认的 LOCAL_PREF 属性会覆盖 MED 属性。

R1

```
router bgp 1
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 1
 neighbor LOCAL password 7 153713093609223720562
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.2 peer-group LOCAL
 neighbor 10.255.255.3 peer-group LOCAL
 neighbor 10.255.255.4 peer-group LOCAL
 neighbor 192.168.100.2 remote-as 2
 neighbor 192.168.100.2 route-map SET_PREF in
 neighbor 192.168.100.2 default-originate
 neighbor 192.168.100.2 distribute-list 1 out
!
access-list 1 permit 0.0.0.0
access-list 1 deny any
access-list 2 permit 192.168.4.0
access-list 2 deny any
!
route-map SET_PREF permit 10
 match ip address 2
 set local-preference 200
!
route-map SET_PREF permit 20
```

R2

```
router bgp 1
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 1
 neighbor LOCAL password 7 153713093609223720562
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.1 peer-group LOCAL
 neighbor 10.255.255.3 peer-group LOCAL
 neighbor 10.255.255.4 peer-group LOCAL
 neighbor 192.168.100.10 remote-as 2
 neighbor 192.168.100.10 route-map SET_PREF in
 neighbor 192.168.100.10 default-originate
 neighbor 192.168.100.10 distribute-list 1 out
 neighbor 192.168.100.58 remote-as 6
!
access-list 1 permit 0.0.0.0
access-list 1 deny any
access-list 2 permit 192.168.3.0
access-list 2 deny any
!
route-map SET_PREF permit 10
 match ip address 2
 set local-preference 200
!
route-map SET_PREF permit 20
```

6. 表3-4中的AS 3是一个末梢AS，AS 4是一个转接AS。这两个AS的IGP都是OSPF，R7和R8的内部接口都在区域0内。写出R7和R8的BGP和OSPF配置，将表3-5中给出的内部地址公布给所有的EBGP对端并且保证在OSPF域内的路由器能够到达任何一个外部地址。在两个方向都不要进行路由再分发。同时，还要保证R7的BGP路由器ID为192.168.3.254。

答案：R7 上的路由图 STUB 可以防止从一 EBGP 对端收到的路由宣告到其他的 EBGP 对端。这样就使 AS 为非转接的。R8 没有这样的设置，所以 AS 4 为转接 AS。

R7

```
router ospf 3
 network 10.255.255.7 0.0.0.0 area 0
 network 172.16.1.1 0.0.0.0 area 0
 default-information originate
!
router bgp 3
 bgp router-id 192.168.3.254
 network 172.16.1.0 mask 255.255.255.0
 network 172.16.3.0 mask 255.255.255.0
 network 172.17.0.0
 network 192.168.6.128 mask 255.255.255.128
 neighbor 192.168.100.25 remote-as 1
 neighbor 192.168.100.25 ebgp-multihop 2
 neighbor 192.168.100.25 update-source Loopback0
 neighbor 192.168.100.25 route-map STUB out
 neighbor 192.168.100.38 remote-as 4
 neighbor 192.168.100.38 route-map STUB out
 neighbor 192.168.100.42 remote-as 5
 neighbor 192.168.100.42 route-map STUB out
 no auto-summary
!
ip route 0.0.0.0 0.0.0.0 Null0
!
ip as-path access-list 1 permit ^$
!
route-map STUB permit 10
 match as-path 1
```

R8

```
router ospf 4
 network 10.255.255.8 0.0.0.0 area 0
 network 172.16.2.1 0.0.0.0 area 0
 default-information originate
!
router bgp 4
 network 172.16.2.0 mask 255.255.255.0
 network 172.16.4.0 mask 255.255.255.0
 network 172.18.0.0
 network 192.168.6.0 mask 255.255.255.128
 neighbor 192.168.100.29 remote-as 1
 neighbor 192.168.100.37 remote-as 3
 neighbor 192.168.100.46 remote-as 5
 no auto-summary
!
ip route 0.0.0.0 0.0.0.0 Null0
```

表 3-5　　AS3 与 AS4 的内部目的子网

AS 3	AS 4
172.16.1.0/24	172.16.2.0/24
172.16.3.0/24	172.16.4.0/24
172.17.0.0/24	172.18.0.0/16
192.168.6.128/25	192.168.6.0/25

7. 调整练习 6 的配置，从而使 R7 和 R8 在它们直连的链路上运行 OSPF；去掉链路上的 BGP。子网 172.16.3.0/24 和 172.16.4.0/24 之间的业务量应当优选这条直连链路，而把任何的 EBGP 链路作为备份。AS 3 内部和 AS 4 内部其他地址之间的业务量应该使用 EBGP 链路而把直连链路作为备份。而且，来自其他 AS 的业务量可以把直连链路作为备份路由。例如，如果到 AS 4 的 EBGP 链路出现了故障，相邻 AS 可以把目的地是 AS 4 的业务量发送给 AS 3，然后在由 AS 3 通过直连链路将这些业务量转发给 AS 4。

答案：

R7

```
router ospf 3
 network 10.255.255.7 0.0.0.0 area 0
 network 172.16.1.1 0.0.0.0 area 0
 network 192.168.100.37 0.0.0.0 area 0
 default-information originate
!
router bgp 3
 bgp router-id 192.168.3.254
 network 172.16.1.0 mask 255.255.255.0
 network 172.16.3.0 mask 255.255.255.0 backdoor
 network 172.17.0.0
 network 192.168.6.128 mask 255.255.255.128
 neighbor 192.168.100.25 remote-as 1
 neighbor 192.168.100.25 ebgp-multihop 2
 neighbor 192.168.100.25 update-source Loopback0
 neighbor 192.168.100.25 route-map STUB out
 neighbor 192.168.100.42 remote-as 5
 neighbor 192.168.100.42 route-map STUB out
 no auto-summary
!
ip route 0.0.0.0 0.0.0.0 Null0
!
ip as-path access-list 1 permit ^$
!
route-map STUB permit 10
 match as-path 1
```

R8

```
router ospf 4
 network 10.255.255.8 0.0.0.0 area 0
 network 172.16.2.1 0.0.0.0 area 0
 network 192.168.100.38 0.0.0.0 area 0
 default-information originate
!
router bgp 4
 network 172.16.2.0 mask 255.255.255.0
 network 172.16.4.0 mask 255.255.255.0 backdoor
 network 172.18.0.0
 network 192.168.6.0 mask 255.255.255.128
 neighbor 192.168.100.29 remote-as 1
 neighbor 192.168.100.46 remote-as 5
 no auto-summary
!
ip route 0.0.0.0 0.0.0.0 Null0
```

8. 表 3-4 中的 AS 5 是一个转接 AS，它的 IGP 是 IS-IS。二层区域 47.0001 扩展到整个 AS。内部的网络是 192.168.9.0、192.168.10.0、192.168.11.0 以及 192.168.12.0。写出 R9 和

R10 和 R11 的 IS-IS 和 BGP 配置。保证 IS-IS 域内的路由器能够了解所有外部的路由而且将内部所有的网络都公布给所有的 EBGP 对等。不要将 IS-IS 路由再分发给 BGP。

答案：虽然可以在外部接口上运行 IS-IS 被动模式，这个配置使用了 **next-hop-self**。

R9

```
router isis
 net 47.0001.0000.1234.abcd.00
 is-type level-2-only
 redistribute bgp 5 metric 0 metric-type external level-2
!
router bgp 5
 network 192.168.9.0
 network 192.168.10.0
 network 192.168.11.0
 network 192.168.12.0
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 5
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.10 peer-group LOCAL
 neighbor 10.255.255.11 peer-group LOCAL
 neighbor 192.168.100.41 remote-as 3
```

R10

```
router isis
 net 47.0001.0000.5678.ef01.00
 is-type level-2-only
 redistribute bgp 5 metric 0 metric-type external level-2
!
router bgp 5
 network 192.168.9.0
 network 192.168.10.0
 network 192.168.11.0
 network 192.168.12.0
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 5
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.9 peer-group LOCAL
 neighbor 10.255.255.11 peer-group LOCAL
 neighbor 192.168.100.45 remote-as 4
```

R11

```
router isis
 net 47.0001.0000.4321.dcba.00
 is-type level-2-only
 redistribute bgp 5 metric 0 metric-type external level-2
!
router bgp 5
 network 192.168.9.0
 network 192.168.10.0
 network 192.168.11.0
 network 192.168.12.0
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 5
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.9 peer-group LOCAL
```

```
 neighbor 10.255.255.10 peer-group LOCAL
 neighbor 192.168.100.33 remote-as 1
```

9. 修改练习8中的配置，只让AS 4知道网络192.168.12.0,其他所有的AS都不知道。

答案：对于192.168.12.0的网络声明从R9与R11中删除，这样它们不会宣告这个网络。在R10中，NO_EXPORT团体加到了192.168.12.0中，这样，它不会宣告到AS外。

R9

```
router isis
 net 47.0001.0000.1234.abcd.00
 is-type level-2-only
 redistribute bgp 5 metric 0 metric-type external level-2
!
router bgp 5
 network 192.168.9.0
 network 192.168.10.0
 network 192.168.11.0
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 5
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.10 peer-group LOCAL
 neighbor 10.255.255.11 peer-group LOCAL
 neighbor 192.168.100.41 remote-as 3
```

R10

```
router isis
 net 47.0001.0000.5678.ef01.00
 is-type level-2-only
 redistribute bgp 5 metric 0 metric-type external level-2
!
router bgp 5
 network 192.168.9.0
 network 192.168.10.0
 network 192.168.11.0
 network 192.168.12.0
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 5
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.9 peer-group LOCAL
 neighbor 10.255.255.11 peer-group LOCAL
 neighbor 192.168.100.45 remote-as 4
 neighbor 192.168.100.45 send-community
 neighbor 192.168.100.45 route-map EXPORT_COMMUNITY out
!
access-list 1 permit 192.168.12.0
!
route-map EXPORT_COMMUNITY permit 10
 match ip address 1
 set community no-export
!
route-map EXPORT_COMMUNITY permit 20
```

R11

```
router isis
 net 47.0001.0000.4321.dcba.00
 is-type level-2-only
```

```
 redistribute bgp 5 metric 0 metric-type external level-2
!
router bgp 5
 network 192.168.9.0
 network 192.168.10.0
 network 192.168.11.0
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 5
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.9 peer-group LOCAL
 neighbor 10.255.255.10 peer-group LOCAL
 neighbor 192.168.100.33 remote-as 1
```

10. 调整练习 9 中的配置，使 AS 3 和 AS 4 优选通过 AS 1 的路径到达网络 192.168.11.0。

答案：网络 192.168.11.0 由 R11 正常宣告，但它由 R9 与 R10 附加了 AS 号。

R9

```
router isis
 net 47.0001.0000.1234.abcd.00
 is-type level-2-only
 redistribute bgp 5 metric 0 metric-type external level-2
!
router bgp 5
 network 192.168.9.0
 network 192.168.10.0
 network 192.168.11.0
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 5
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.10 peer-group LOCAL
 neighbor 10.255.255.11 peer-group LOCAL
 neighbor 192.168.100.41 remote-as 3
 neighbor 192.168.100.41 route-map PREPEND out
!
access-list 1 permit 192.168.11.0
!
route-map PREPEND permit 10
 match ip address 1
 set as-path prepend 5 5
!
route-map PATH permit 20
```

R10

```
router isis
 net 47.0001.0000.5678.ef01.00
 is-type level-2-only
 redistribute bgp 5 metric 0 metric-type external level-2
!
router bgp 5
 network 192.168.9.0
 network 192.168.10.0
 network 192.168.11.0
 network 192.168.12.0
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 5
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
```

```
 neighbor 10.255.255.9 peer-group LOCAL
 neighbor 10.255.255.11 peer-group LOCAL
 neighbor 192.168.100.45 remote-as 4
 neighbor 192.168.100.45 send-community
 neighbor 192.168.100.45 route-map EXPORT_COMMUNITY out
!
access-list 1 permit 192.168.12.0
access-list 2 permit 192.168.11.0
!
route-map EXPORT_COMMUNITY permit 10
 match ip address 1
 set community no-export
!
route-map EXPORT_COMMUNITY permit 20
 match ip address 1
 set as-path prepend 5 5
!
route-map EXPORT_COMMUNITY permit 30
```

R11

```
router isis
 net 47.0001.0000.4321.dcba.00
 is-type level-2-only
 redistribute bgp 5 metric 0 metric-type external level-2
!
router bgp 5
 network 192.168.9.0
 network 192.168.10.0
 network 192.168.11.0
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 5
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.9 peer-group LOCAL
 neighbor 10.255.255.10 peer-group LOCAL
 neighbor 192.168.100.33 remote-as 1
```

11. 表 3-4 中 AS 6 的内部网络是 192.168.16.0、192.168.17.0、192.168.18.0 和 192.168.19.0。写出 R12 的 BGP 配置，让它把这些网络公布给邻居 AS 并且公布一个网络的归纳路由。而邻居 AS 应该只将这个归纳路由公布给其他的 AS。

答案：

```
router bgp 6
 network 192.168.16.0
 network 192.168.17.0
 network 192.168.18.0
 network 192.168.19.0
 aggregate-address 192.168.16.0 255.255.252.0
 neighbor 192.168.100.57 remote-as 1
 neighbor 192.168.100.57 send-community
 neighbor 192.168.100.57 route-map AGGREGATE out
!
access-list 101 permit ip host 192.168.16.0 host 255.255.252.0
!
route-map AGGREGATE permit 10
 match ip address 101
 set community none
!
route-map AGGREGATE permit 20
 set community no-export
```

12. 调整 R12 的 EBGP 邻居最近的配置，使该邻居不接受不属于 R12 公布的聚合地址的前缀，不接受长于 24 比特的前缀，不接受多余 5 个的前缀。

答案：

```
R2
router bgp 1
 neighbor LOCAL peer-group
 neighbor LOCAL remote-as 1
 neighbor LOCAL password 7 153713093609223720 62
 neighbor LOCAL update-source Loopback0
 neighbor LOCAL next-hop-self
 neighbor 10.255.255.1 peer-group LOCAL
 neighbor 10.255.255.3 peer-group LOCAL
 neighbor 10.255.255.4 peer-group LOCAL
 neighbor 192.168.100.10 remote-as 2
 neighbor 192.168.100.10 route-map SET_PREF in
 neighbor 192.168.100.10 default-originate
 neighbor 192.168.100.10 distribute-list 1 out
 neighbor 192.168.100.58 remote-as 6
 neighbor 192.168.100.58 maximum-prefix 5
 neighbor 192.168.100.58 route-map PREFIX_LIMIT in
!
access-list 1 permit 0.0.0.0
access-list 1 deny any
access-list 2 permit 192.168.3.0
access-list 2 deny any
!
ip prefix-list AS6 seq 5 permit 192.168.16.0/22 le 24
!
route-map SET_PREF permit 10
 match ip address 2
 set local-preference 200
!
route-map SET_PREF permit 20
!
route-map PREFIX_LIMIT permit 10
 match ip address prefix-list AS6
```

13. 例 3-164 给出了表 3-4 中 R7 的 BGP 配置。表 3-5 中给出了内部前缀由 OSPF 进行公布。

例 3-164　路由器 R7 的 BPG 配置

```
router bgp 3
 redistribute ospf 1
 neighbor NEIGHBORS peer-group
 neighbor NEIGHBORS ebgp-multihop 2
 neighbor NEIGHBORS update-source Loopback0
 neighbor NEIGHBORS route-map EX13 out
 neighbor 10.255.255.8 remote-as 4
 neighbor 10.255.255.8 peer-group NEIGHBORS
 neighbor 10.255.255.9 remote-as 5
 neighbor 10.255.255.9 peer-group NEIGHBORS
 neighbor 10.255.255.3 remote-as 1
 neighbor 10.255.255.3 peer-group NEIGHBORS
 no auto-summary
!
ip classless
```

```
ip as-path access-list 1 permit ^1 2$
!
access-list 1 permit 172.16.1.0
access-list 2 permit 172.16.3.0
!
route-map EX13 permit 10
 match ip address 1
 set as-path prepend 2
!
route-map EX13 permit 20
 match ip address 2
 set as-path prepend 1
!
route-map EX13 permit 30
 match as-path 1
 set as-path prepend 4 5
!
route-map EX13 deny 40
```

解释路由图 EX13 的作用。

答案：10 号路由图匹配前缀 172.16.1.0 并在 AS_PATH 中加入 2。因此，AS 2 中的路由器会拒绝这个前缀。20 号路由图匹配 172.16.2.0 并在 AS_PATH 中加入 1，这样这条路由会被 AS 1 中的路由器拒绝。30 中路由图 AS_PATH 为[1,2]，意味着路由由 AS 2 产生由 AS 1 宣告。这个路由图给 AS_PATH 中加入 AS 号 4 与 5。所以 AS 4 与 AS 5 会拒绝这些路由。40 号路由图抑制宣告的其他路由。

14. 图 3-36 中的路由器 R1 是路由器 R2、R3 和 R4 的路由反射器并且通过 FR PVC 与这些邻居互连。写出 R1 的 BGP 配置，为与这 4 个路由器相连的网络提供所有的连接性。簇 ID 是 6500。

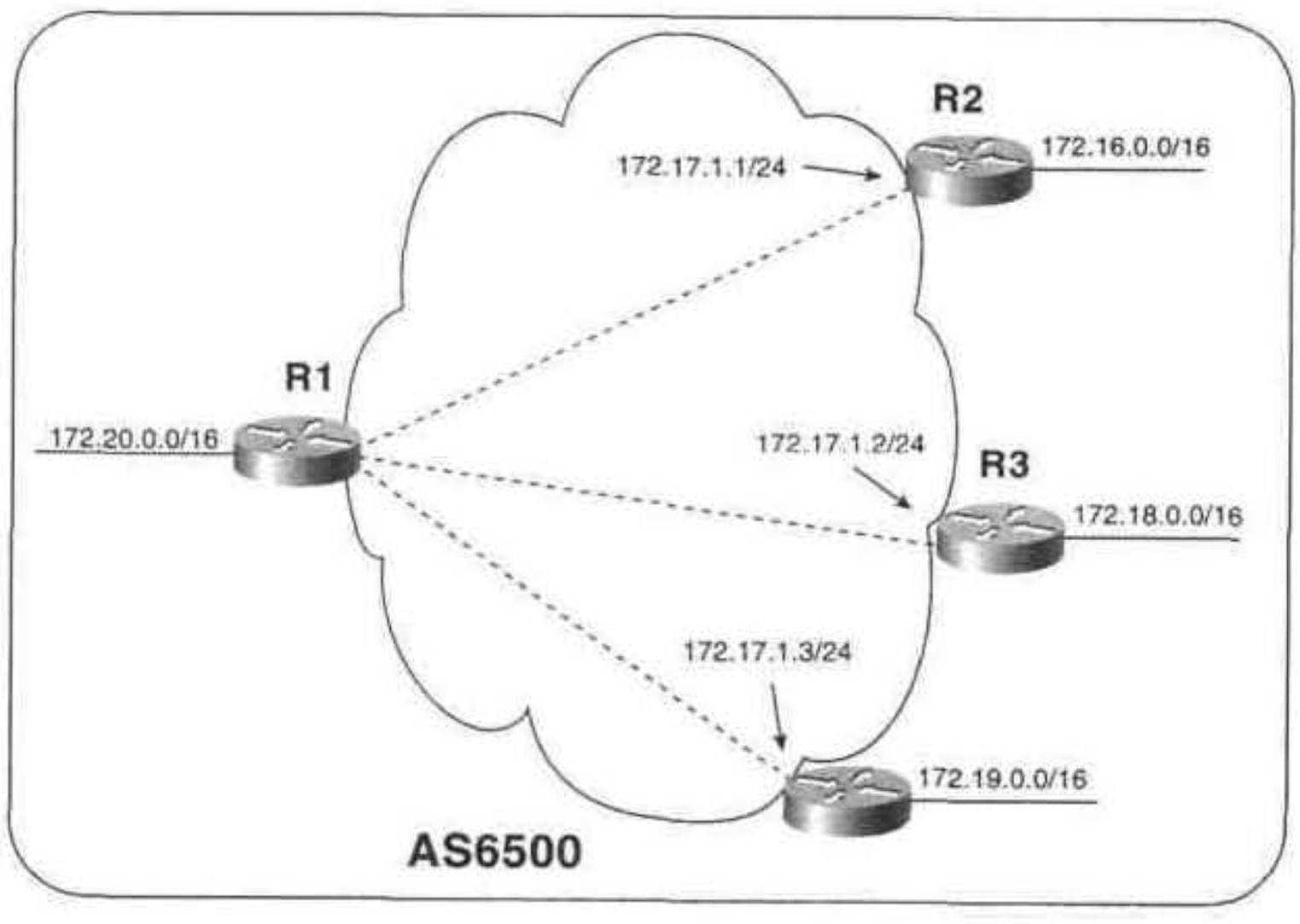


图 3-36 配置练习 14 的路由反射簇

答案：

```
router bgp 6500
 no synchronization
 bgp cluster-id 6500
 network 172.20.0.0
 neighbor 172.16.1.1 remote-as 6500
 neighbor 172.16.1.1 route-reflector-client
 neighbor 172.16.1.2 remote-as 6500
 neighbor 172.16.1.2 route-reflector-client
 neighbor 172.16.1.3 remote-as 6500
 neighbor 172.16.1.3 route-reflector-client
```

第 4 章　配置练习的答案

配置练习 1~5 参考图 4-28。

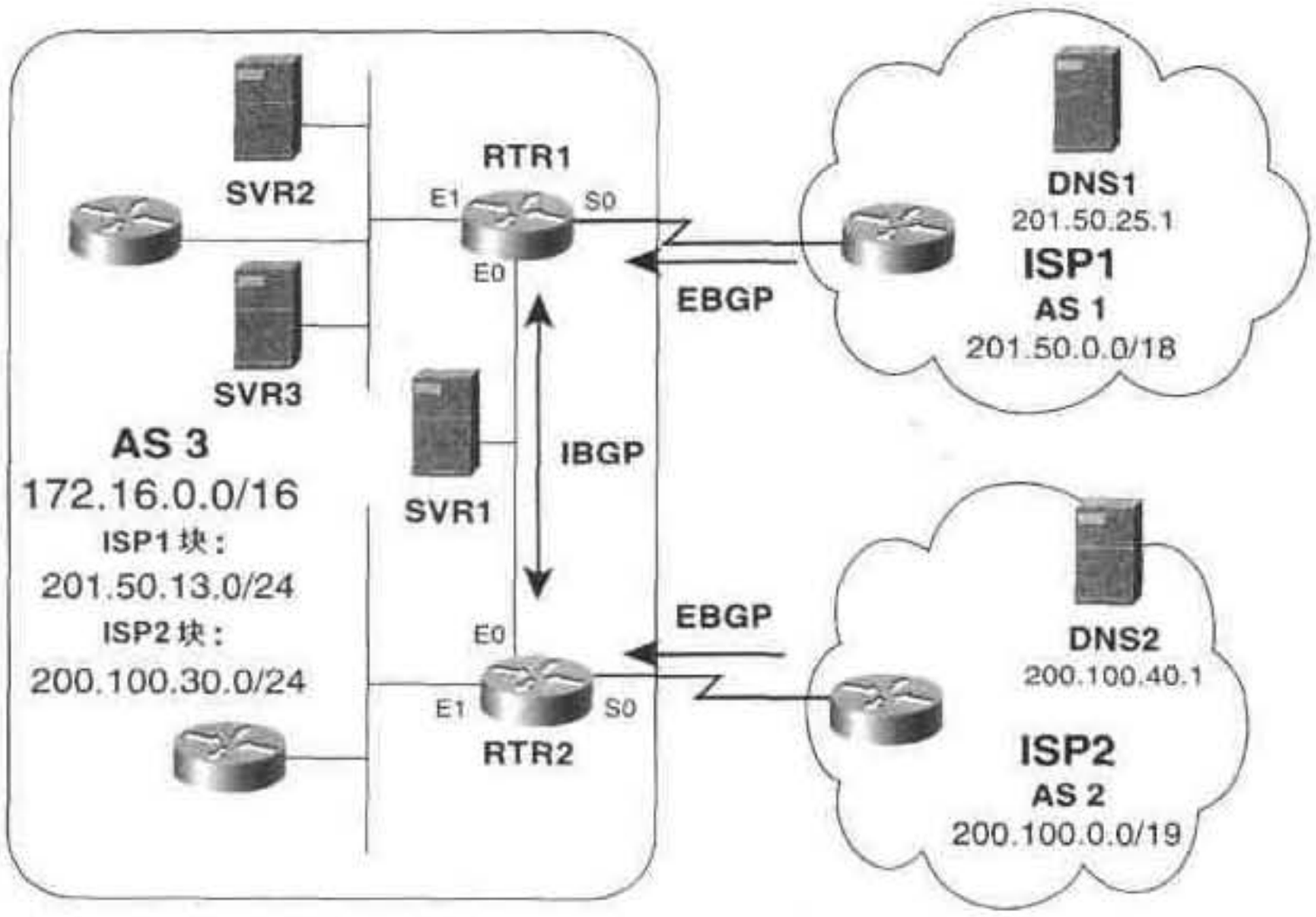


图 4-28　配置练习 1-5 中的网络拓扑

1．图 4-28 中的 ISP1 把地址块 201.50.13.0/24 分给了 AS 3。ISP2 把地址块 200.100.30.0/24 也分给了 AS 3。RTR1 与 RTR2 从 ISP 的路由器上收到了完整的 BGP 路由，但不能向 ISP 传送任何路由。两台路由器间运行 IBGP，所有的以太网接口上运行 OSPF。BGP 与 OSPF 间没有路由再分发。路由器的接口地址如下：

RTR1，E0：172.16.3.1/24

RTR1，E1：172.16.2.1/24

RTR1，S0：201.50.26.13/30

RTR2，E0：172.16.3.2/24

RTR2，E1：172.16.1.1/24

RTR2，S0：200.100.29.241/30

SVR1 是 AS 3 中受权的 DNS 服务器；它的地址为 172.16.3.3。DNS1 访问 SVR1 用的地址

为201.50.13.1，而DNS2访问同一台服务器用的地址是200.100.30.254。写出RTR1与RTR2中路由与NAT的配置，把内部地址正确地翻译成每一个ISP分配的地址块中的地址。任何一个内部设备必须能访问每一个ISP，不过，这种情况下，不能让源地址为有专用地址的包离开AS 3。

答案：

RTR1

```
interface Loopback0
 ip address 172.16.255.2 255.255.255.255
!
interface Ethernet0
 ip address 172.16.3.1 255.255.255.0
 ip nat inside
!
interface Ethernet1
 ip address 172.16.2.1 255.255.255.0
 ip nat inside
!
interface Serial0
 description to ISP1
 ip address 201.50.26.13 255.255.255.252
 ip access-group 101 out
 ip nat outside
!
autonomous-system 3
!
router ospf 1
 redistribute static
 network 172.16.0.0 0.0.255.255 area 0
 default-information originate
!
router bgp 3
 neighbor 172.16.255.1 remote-as 3
 neighbor 172.16.255.1 update-source Loopback0
 neighbor 201.50.26.14 remote-as 1
!
ip nat pool ISP1Pool 201.50.13.2 201.50.13.254 netmask 255.255.255.0
ip nat inside source list 1 pool ISP1Pool
ip nat inside source static 172.16.3.3 201.50.13.1
!
ip route 0.0.0.0 0.0.0.0 201.50.26.14
ip route 201.50.0.0 255.255.192.0 201.50.26.14
!
access-list 1 permit 172.16.0.0 0.0.255.255
access-list 101 deny   ip 172.16.0.0 0.0.255.255 any
access-list 101 permit ip any any
```

RTR2

```
interface Loopback0
 ip address 172.16.255.1 255.255.255.255
!
interface Ethernet0
 ip address 172.16.3.2 255.255.255.0
 ip nat inside
!
interface Ethernet1
 ip address 172.16.1.1 255.255.255.0
 ip nat inside

!
```

```
interface Serial0
 description to ISP2
 ip address 200.100.29.241 255.255.255.252
 ip access-group 101 out
 ip nat outside
!
autonomous-system 3
!
router ospf 1
 redistribute static
 network 172.16.0.0 0.0.255.255 area 0
 default-information originate
!
router bgp 3
 neighbor 172.16.255.2 remote-as 3
 neighbor 172.16.255.2 update-source Loopback0
 neighbor 200.100.29.242 remote-as 2
!
ip nat pool ISP2Pool 200.100.30.1 200.100.30.253 netmask 255.255.255.0
ip nat inside source list 1 pool ISP2Pool
ip nat inside source static 172.16.3.3 200.100.30.254
!
ip route 0.0.0.0 0.0.0.0 200.100.29.242
ip route 200.100.0.0 255.255.224.0 200.100.29.242
!
access-list 1 permit 172.16.0.0 0.0.255.255
access-list 101 deny ip 172.16.0.0 0.0.255.255 any
access-list 101 permit ip any any
```

2. 图 4-28 中 SVR2 的地址为 172.16.2.2，SVR3 的地址为 172.16.2.3。改变配置练习 1 中的配置，使 ISP 的 AS 中的设备访问这些设备时，访问地址 201.50.13.3，而实际对这些服务器轮流进行访问。

答案：注意除了新的命令外，ISP1Pool 中不再包括地址 201.50.13.3。

```
RTR1
ip nat pool ISP1Pool 201.50.13.4 201.50.13.254 netmask 255.255.255.0
ip nat pool SVRs 172.16.2.2 172.16.2.3 netmask 255.255.0.0 type rotary
ip nat inside source list 1 pool ISP1Pool
ip nat inside source static 172.16.3.3 201.50.13.1
ip nat inside destination list 2 pool SVRs
!
access-list 1 permit 172.16.0.0 0.0.255.255
access-list 2 permit 201.50.13.3
```

3. 从 ISP2 中向 200.100.30.50 发送 HTTP 包，实际发送到了图 4-28 中的 SVR2 处。从 ISP2 中发往 200.100.30.500 的 SMTP 包，实际发送到了 SVR3。修改前一个练习中的配置，来实现这种翻译。

答案：IG 地址在 ISP2Pool 的范围内，除了静态的映射外，ISP2Pool 还必须重新配置。

```
RTR2
ip nat pool ISP2Pool netmask 255.255.255.0
 address 200.100.30.1 200.100.30.49
 address 200.100.30.51 200.100.30.253
ip nat inside source list 1 pool ISP2Pool
ip nat inside source static tcp 172.16.2.3 24 200.100.30.50 25 extendable
```

```
ip nat inside source static tcp 172.16.2.2 80 200.100.30.50 80 extendable
ip nat inside source static 172.16.3.3 200.100.30.254
!
access-list 1 permit 172.16.0.0 0.0.255.255
```

4. 图 4-28 中，五个外部设备 201.50.12.67-201.50.12.71，必须在 AS 3 看来分别具有地址 192.168.1.1-192.168.1.5。在前一个配置中加入相应的 NAT 配置。

答案：

```
RTR1
ip nat pool ISP1Pool 201.50.13.2 201.50.13.254 netmask 255.255.255.0
ip nat pool SVRs 172.16.2.2 172.16.2.3 netmask 255.255.255.0 type rotary
ip nat inside source list 1 pool ISP1Pool
ip nat inside source static 172.16.3.3 201.50.13.1
ip nat inside destination list 2 pool SVRs
ip nat outside source static 201.50.12.71 192.168.1.5
ip nat outside source static 201.50.12.70 192.168.1.4
ip nat outside source static 201.50.12.69 192.168.1.3
ip nat outside source static 201.50.12.68 192.168.1.2
ip nat outside source static 201.50.12.67 192.168.1.1
!
access-list 1 permit 172.16.0.0 0.0.255.255
access-list 2 permit 201.50.13.1
```

5. 图 4-28 中 AS 3 里的设备具有地址子网 172.16.100.0/24，在有包发往 ISP2 时，应具有 IG 地址 200.100.30.75。修改前一练习的配置，来实现这一点。

答案：解决的方案是配置 PAT。不同于本章例中所示的 PAT，这里使用的地址不是外部接口的地址。所以在 RTR2 上配置一个地址池，其中只有一个地址。注意到访问表 1 已经过了修改，所以 PAT 使用的 IL 地址不在 ISP2Pool 范围中。

```
RTR2
ip nat pool ISP2Pool netmask 255.255.255.0
 address 200.100.30.1 200.100.30.49
 address 200.100.30.51 200.100.30.253
ip nat pool PATPool 200.100.30.75 200.100.30.75 netmask 255.255.0.0
ip nat inside source list 1 pool ISP2Pool
ip nat inside source list 3 pool PATPool overload
ip nat inside source static tcp 172.16.2.3 24 200.100.30.50 25 extendable
ip nat inside source static tcp 172.16.2.2 80 200.100.30.50 80 extendable
ip nat inside source static 172.16.3.3 200.100.30.254
!
access-list 1 deny 172.16.100.0 0.0.0.255
access-list 1 permit 172.16.0.0 0.0.255.255
access-list 3 permit 172.16.100.0 0.0.0.255
```

6. 图 4-29 中，RTR1 与 RTR2 上加了冗余链路，每一台路由器都与两个 ISP 相连，每一台路由器都从两个 ISP 接收完整的 BGP 路由。RTR1 的 S1 地址是 200.100.29.137/30，RTR2 的 S1 地址为 201.50.26.93/30。写出这两台路由器的配置，保证上述各要求都仍能正确工作。

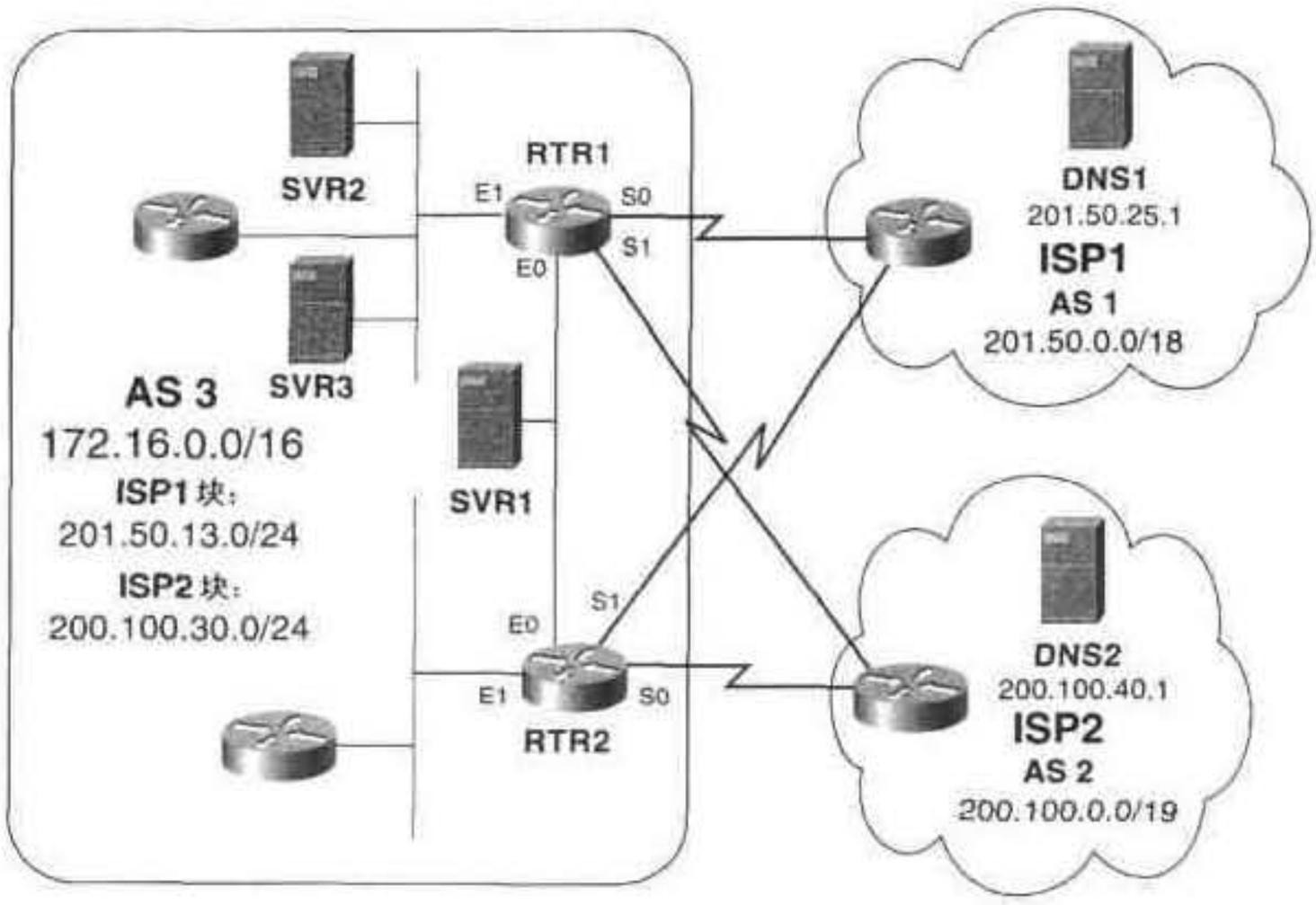


图 4-29　配置练习 6 的网络拓扑

答案：

RTR1

```
interface Loopback0
 ip address 172.16.255.2 255.255.255.255
!
interface Ethernet0
 ip address 172.16.3.1 255.255.255.0
 ip nat inside
!
interface Ethernet1
 ip address 172.16.2.1 255.255.255.0
 ip nat inside
!
interface Serial0
 description to ISP1
 ip address 201.50.26.13 255.255.255.252
 ip access-group 101 out
 ip nat outside
!
interface Serial1
 description to ISP2
 ip address 200.100.29.137 255.255.255.252
 ip access-group 101 out
 ip nat outside
!
autonomous-system 3
!
router ospf 1
 redistribute static
 network 172.16.0.0 0.0.255.255 area 0
 default-information originate
!
router bgp 3
 neighbor 172.16.255.1 remote-as 3
 neighbor 172.16.255.1 update-source Loopback0
 neighbor 200.100.29.138 remote-as 2
 neighbor 201.50.26.14 remote-as 1
```

```
!
ip nat pool ISP1Pool 201.50.13.2 201.50.13.254 netmask 255.255.255.0
ip nat pool ISP2Pool netmask 255.255.255.0
 address 200.100.30.1 200.100.30.49
 address 200.100.30.51 200.100.30.253
ip nat pool PATPool 200.100.30.75 200.100.30.75 netmask 255.255.0.0
ip nat pool SVRs 172.16.2.2 172.16.2.3 netmask 255.255.255.0 type rotary
ip nat inside source route-map ISP1 pool ISP1Pool
ip nat inside source route-map ISP2 pool ISP2Pool
ip nat inside source list 3 pool PATPool overload
ip nat inside source static tcp 172.16.2.3 24 200.100.30.50 25 extendable
ip nat inside source static tcp 172.16.2.2 80 200.100.30.50 80 extendable
ip nat inside source static 172.16.3.3 201.50.13.1
ip nat inside destination list 2 pool SVRs
ip nat outside source static 201.50.12.71 192.168.1.5
ip nat outside source static 201.50.12.70 192.168.1.4
ip nat outside source static 201.50.12.69 192.168.1.3
ip nat outside source static 201.50.12.68 192.168.1.2
ip nat outside source static 201.50.12.67 192.168.1.1
!
access-list 1 deny 172.16.100.0 0.0.0.255
access-list 1 permit 172.16.0.0 0.0.255.255
access-list 2 permit 201.50.13.1
access-list 3 permit 172.16.100.0 0.0.0.255
access-list 4 permit 200.100.29.138
access-list 5 permit 201.50.26.14
access-list 101 deny   ip 172.16.0.0 0.0.255.255 any
access-list 101 permit ip any any
!
route-map ISP1 permit 10
 match ip address 1
 match ip next-hop 5
!
route-map ISP2 permit 10
 match ip address 1
 match ip next-hop 4
```

RTR2

```
interface Loopback0
 ip address 172.16.255.1 255.255.255.255
!
interface Ethernet0
 ip address 172.16.3.2 255.255.255.0
 ip nat inside
!
interface Ethernet1
 ip address 172.16.1.1 255.255.255.0
 ip nat inside

!
interface Serial0
 description to ISP2
 ip address 200.100.29.241 255.255.255.252
 ip access-group 101 out
 ip nat outside
!
interface Serial1
 description to ISP1
 ip address 201.50.26.93 255.255.255.252
 ip access-group 101 out
 ip nat outside
autonomous-system 3
```

```
!
router ospf 1
 redistribute static
 network 172.16.0.0 0.0.255.255 area 0
 default-information originate
!
router bgp 3
 neighbor 172.16.255.2 remote-as 3
 neighbor 172.16.255.2 update-source Loopback0
 neighbor 200.100.29.242 remote-as 2
 neighbor 201.50.26.94 remote-as 1
!
ip nat pool ISP1Pool 201.50.13.2 201.50.13.254 netmask 255.255.255.0
ip nat pool ISP2Pool netmask 255.255.255.0
 address 200.100.30.1 200.100.30.49
 address 200.100.30.51 200.100.30.253
ip nat pool PATPool 200.100.30.75 200.100.30.75 netmask 255.255.0.0
ip nat pool SVRs 172.16.2.2 172.16.2.3 netmask 255.255.255.0 type rotary
ip nat inside source route-map ISP1 pool ISP1Pool
ip nat inside source route-map ISP2 pool ISP2Pool
ip nat inside source list 3 pool PATPool overload
ip nat inside source static tcp 172.16.2.3 24 200.100.30.50 25 extendable
ip nat inside source static tcp 172.16.2.2 80 200.100.30.50 80 extendable
ip nat inside source static 172.16.3.3 200.100.30.254
ip nat inside destination list 2 pool SVRs
ip nat outside source static 201.50.12.71 192.168.1.5
ip nat outside source static 201.50.12.70 192.168.1.4
ip nat outside source static 201.50.12.69 192.168.1.3
ip nat outside source static 201.50.12.68 192.168.1.2
ip nat outside source static 201.50.12.67 192.168.1.1
!
access-list 1 deny 172.16.100.0 0.0.0.255
access-list 1 permit 172.16.0.0 0.0.255.255
access-list 2 permit 201.50.13.1
access-list 3 permit 172.16.100.0 0.0.0.255
access-list 4 permit 200.100.29.242
access-list 5 permit 201.50.26.94
access-list 101 deny ip 172.16.0.0 0.0.255.255 any
access-list 101 permit ip any any
!
route-map ISP1 permit 10
 match ip address 1
 match ip next-hop 5
!
route-map ISP2 permit 10
 match ip address 1
 match ip next-hop 4
```

第 6 章　配置练习答案

1．Cisco IOS 启动 IP 多播路由的全局命令是什么？

答案：**ip multicast-routing**。

2．显示在一个支持 PIM 的接口是密集模式、稀疏模式还是密集稀疏模式的命令。

答案：

```
ip pim dense-mode
ip pim sparse-mode
ip pim sparse-dense mode
```

3．显示静态指定 RP 为 172.18.20.4 的命令。

答案：ip pim rp-address 172.18.20.4。

4．写出把 239.1.2.3 和从 228.1.8.0 到 228.1.8.255 的组地址映射到 RP 192.168.15.5，把组 239.6.7.8 映射到 RP192.168.20.10，把其他的组映射到 RP192.168.25.1 的必要命令声明。

答案：

```
ip pim rp-address 192.168.15.5 1
ip pim rp-address 192.168.20.10 2
ip pim rp-address 192.168.25.1
!
access-list 1 permit 239.1.2.3 0.0.0.0
access-list 1 permit 228.1.8.0 0.0.0.255
access-list 2 permit 239.6.7.8 0.0.0.0
```

5．在图 6-11 中所有路由器的接口均运行在密集稀疏模式下。写出如下相关配置：使 R1 只是 226.13.0.0/24 地址的 RP 的配置；R2 只是 239.0.0.0/8 的地址的 RP；R3 为映射代理；保证这个映射代理对于指定的组，只认为 R1 和 R2 为 RP。RP 所有 Auto-RP 消息 TTL 值均为 20。

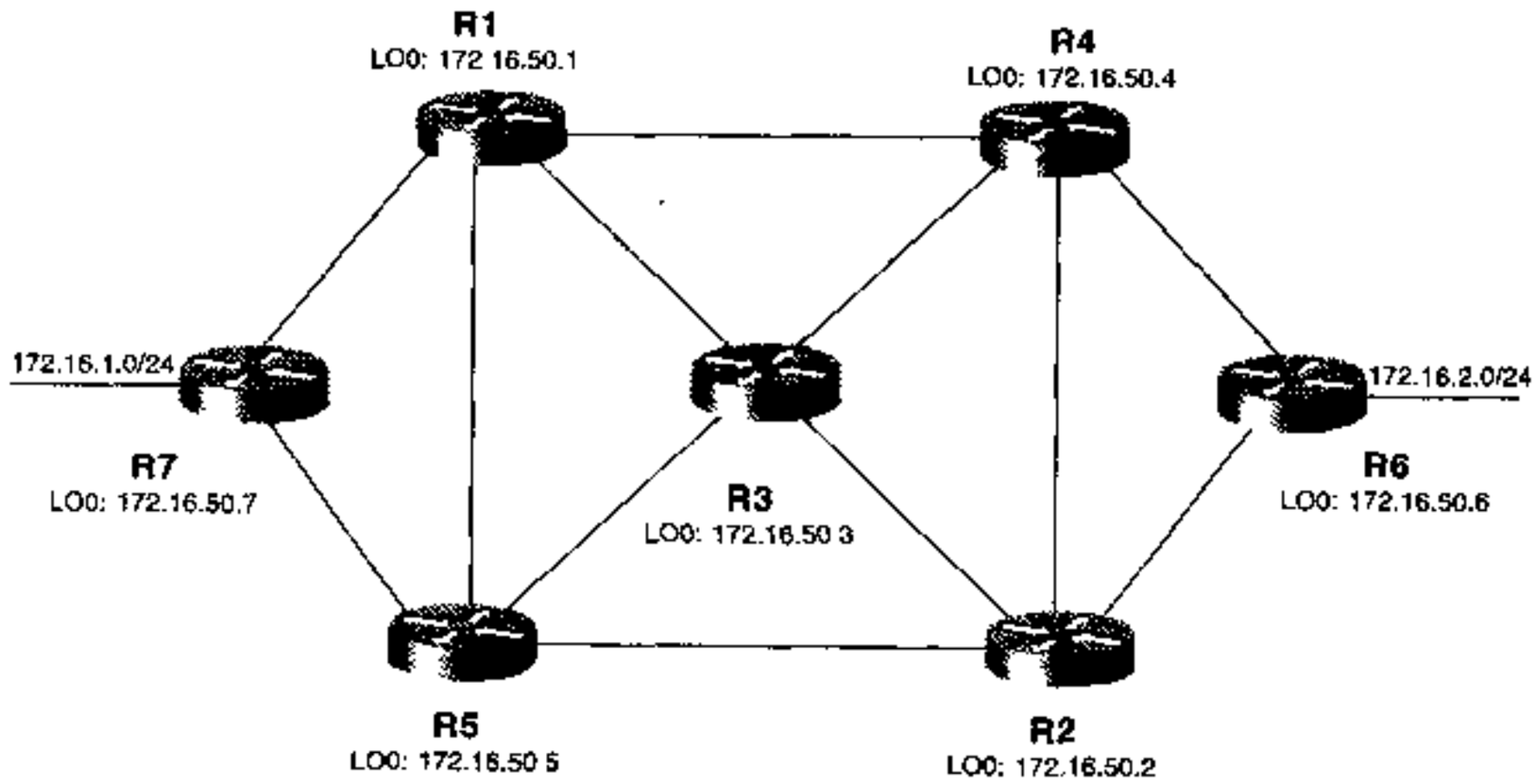


图 6-11　配置练习 5-8 的拓扑

答案：

R1

```
ip pim send-rp-announce Loopback0 scope 20 group-list 1
!
access-list 1 permit 226.13.0.0 0.0.0.255
```

R2

```
ip pim send-rp-announce Loopback0 scope 20 group-list 1
!
```

```
access-list 1 permit 239.0.0.0 0.255.255.255
```

R3

```
ip pim rp-announce-filter rp-list 10 group-list 11
ip pim rp-announce-filter rp-list 20 group-list 21
ip pim send-rp-discovery Loopback0 scope 20
!
access-list 10 permit 172.16.50.1
access-list 11 permit 226.13.0.0 0.0.0.255
access-list 20 permit 172.16.50.2
access-list 21 permit 239.0.0.0 0.255.255.255
```

6. 已知配置练习 5 中的配置，假设一个源产生组 228.23.14.135 的流量，一个组员请求加入这个组，会有什么情况发生？

答案：无论 R1 还是 R2 都没有配置成为这个组的 RP。因为所有的接口运行在密集稀疏模式下，但是这个组已经调用了密集模式，在源与组员间建立了 SPT。

7. 参考图 6-11，写出如下配置：启动自举协议，使 R1 与 R2 成为配置练习 5 中描述的组地址的 C-RP，R3 成为 BSR，R4 成为备份 BSR。

答案：

R1

```
ip pim rp-candidate Loopback0 group-list 1
!
access-list 1 permit 226.13.0.0 0.0.0.255
```

R2

```
ip pim rp-candidate Loopback0 group-list 1
!
access-list 1 permit 239.0.0.0 0.255.255.255
```

R3

```
ip pim bsr-candidate Loopback0 50
```

R4

```
ip pim bsr-candidate Loopback0 0
```

8. 写下关于图 6-11 的配置：使源 172.16.1.75 与组员 172.16.2.100 间实现负载均衡。在隧道接口上使用未用的地址，参考 E0，假设 IGP 可以宣告这些地址。

答案：

R6

```
interface Tunnel0
 ip unnumbered Ethernet0
 ip pim sparse-dense mode
 tunnel source Loopback0
 tunnel destination 172.16.50.7
!
ip mroute 172.16.1.75 255.255.255.255 Tunnel0
```

R7

```
interface Tunnel0
 ip unnumbered Ethernet0
 ip pim sparse-dense mode
 tunnel source Loopback0
 tunnel destination 172.16.50.6
```

9. 检查案例研究“多播负荷分担”中 Homburg 与 Porkpie 的配置。哪一个路由器在隧道

接口上运行 OSPF 的被动模式，为什么？

答案：把单播协议(本案例中为 OSPF)设为被动模式，协议知道多播 RPF 功能所需的接口地址，同时防止单播游量使用隧道。

10. **ip pim spt-threshold 100 group-list 25** 这一命令的目的是什么？

答案：当对于一个组地址或在访问表 25 中规定地址的多播包的到达速率超过了 100kbit/s，PIM-SM 路由器从共享树切换到最短路径树。

第 9 章 配置练习答案

1. 配置一个路由器，只接受 172.16.1.2 和 172.16.1.3 管理工作站的轮询。不允许工作站的写操作。只允许工作站 SNMP MIBII 接口条目读操作。允许 172.16.1.4 工作站读所有的 MIB 变量，并允许使用 SNMP 上载下载配置。通过 SNMP 发送 Notification 级的日志信息到 172.16.1.4。

答案：

```
access-list 1 permit 172.16.1.2 0.0.0.1
access-list 2 permit 172.16.1.4
snmp-server view interface_entries ifEntry included
snmp-server community anystring view interface_entries RO 1
snmp-server community restricted RO 2
snmp-server tftp-server-list 2
snmp-server enable traps syslog
logging history notification
```

2. 配置路由器在 5 分钟平均 CPU 利用率超过 90%时发送 SNMP trap 到 172.16.1.4。当路由器 CPU 利用率在 60 秒间隔内从低于 85%到高于 90%时发送 trap。

答案：

```
snmp-server community eventtrap RO
snmp-server enable traps
snmp-server host 172.16.1.4 eventtrap
rmon event 1 trap eventtrap description "High 5-minute CPU" owner smith
rmon alarm 10 lsystem.58.0 60 absolute rising-threshold 90 1 falling-threshold 85
owner smith
```

3. 配置路由器使用 NTP 从路由器 172.16.100.100 获取时钟信息更新内部的时间和日期。不允许其他路由器从你所配置的路由器获取时钟信息。

答案：

```
ntp server 172.16.100.100
```

4. 配置NetFlow汇集缓存，基于源地址目的地址前缀对数据分组。在数据中包含 peer-AS，将数据输出到 172.16.1.4。

答案：

```
ip cef
!
ip flow-export version 5 peer-as
ip flow-export destination 172.16.1.4 125
```

```
ip flow-aggregation cache prefix
 cache entries 2046
 cache timeout inactive 200
 cache timeout active 45
 export destination 172.16.1.4 9991
 enabled
!
```

5. 配置两个路由器在以太网段上相互备份。路由器 A 为主用，A 故障时 B 接替。A 恢复以后重新成为主用路由器。路由器 A 由两个串行链路，串口 0 和串口 1，可以将流量转发到不同的目的地。如果两个链路都故障，路由器 B 接替成为主用路由器。

答案：

Router A

```
interface Ethernet 0
 ip address 172.16.1.100 255.255.255.0
 standby 1 priority 120 preempt
 standby 1 ip 172.16.1.201
 standby 1 track Serial0 25
 standby 1 track Serial1 25
```

Router B

```
interface Ethernet 0
 ip address 172.16.1.101 255.255.255.0
 standby 1 ip
 standby 1 priority 100 preempt
```

附录 F　故障排除练习答案

第 1 章　故障排除练习答案

1．如图 1-17 所示，在网络中加入了路由器 RTG。

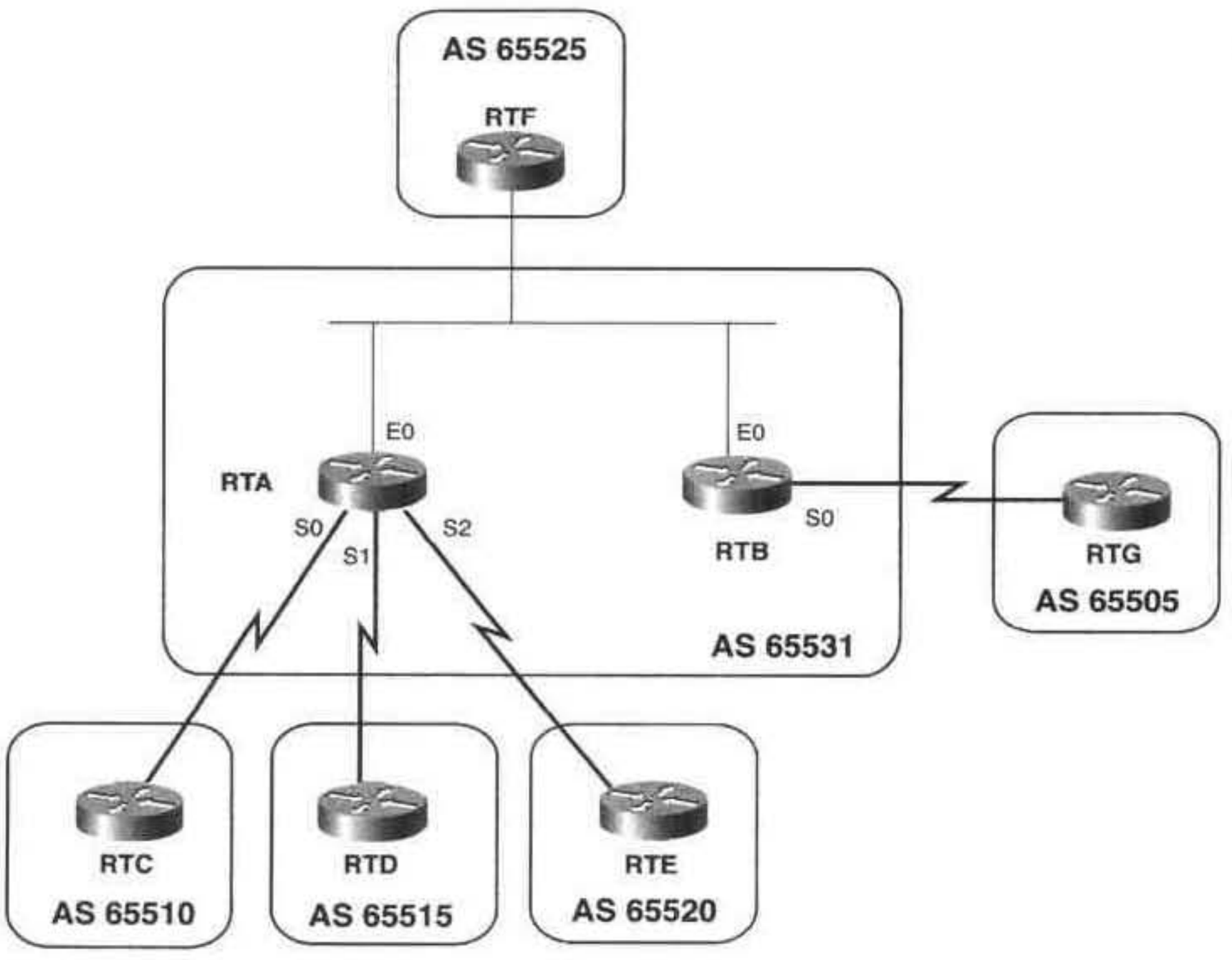


图 1-17　故障排除练习 1 的网络图

虽然它和 RTB 对等并且相互交换可到达信息，但是这里存在着一个配置错误。根据例 1-29 给出来的信息，指出错误是什么？

例 1-29　图 1-17 中 RTB 和 RTG 的 EGP 表

```
RTB#show ip egp
Local autonomous system is 65531

 EGP Neighbor       FAS/LAS  State     SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.1.1      65531/65531 UP      4      2      6     60    180   2 Perm, Pass
*192.168.1.3      65525/65531 UP      4      2    492     60    180   2 Perm, Pass
*192.168.5.2      65505/65531 UP      3      2     33     60    180   3 Temp, Pass
 EGP Neighbor     Third Party
```

```
*192.168.1.1       192.168.1.3(e)
*192.168.1.3       192.168.1.1
RTB#
```

```
RTG#show ip egp
Local autonomous system is 65505

 EGP Neighbor        FAS/LAS  State     SndSeq RcvSeq Hello  Poll j/k Flags
*192.168.5.1     65505/65505 UP      9     36      3    60   180   4 Perm, Act
RTG#
```

答案：路由器 RTG 的 EGP 配置是 router egp 65505 而不是 router egp 65531。

第 3 章　故障排除练习答案

图 3-37 给出了故障排除练习 1~6 的网络互连图。

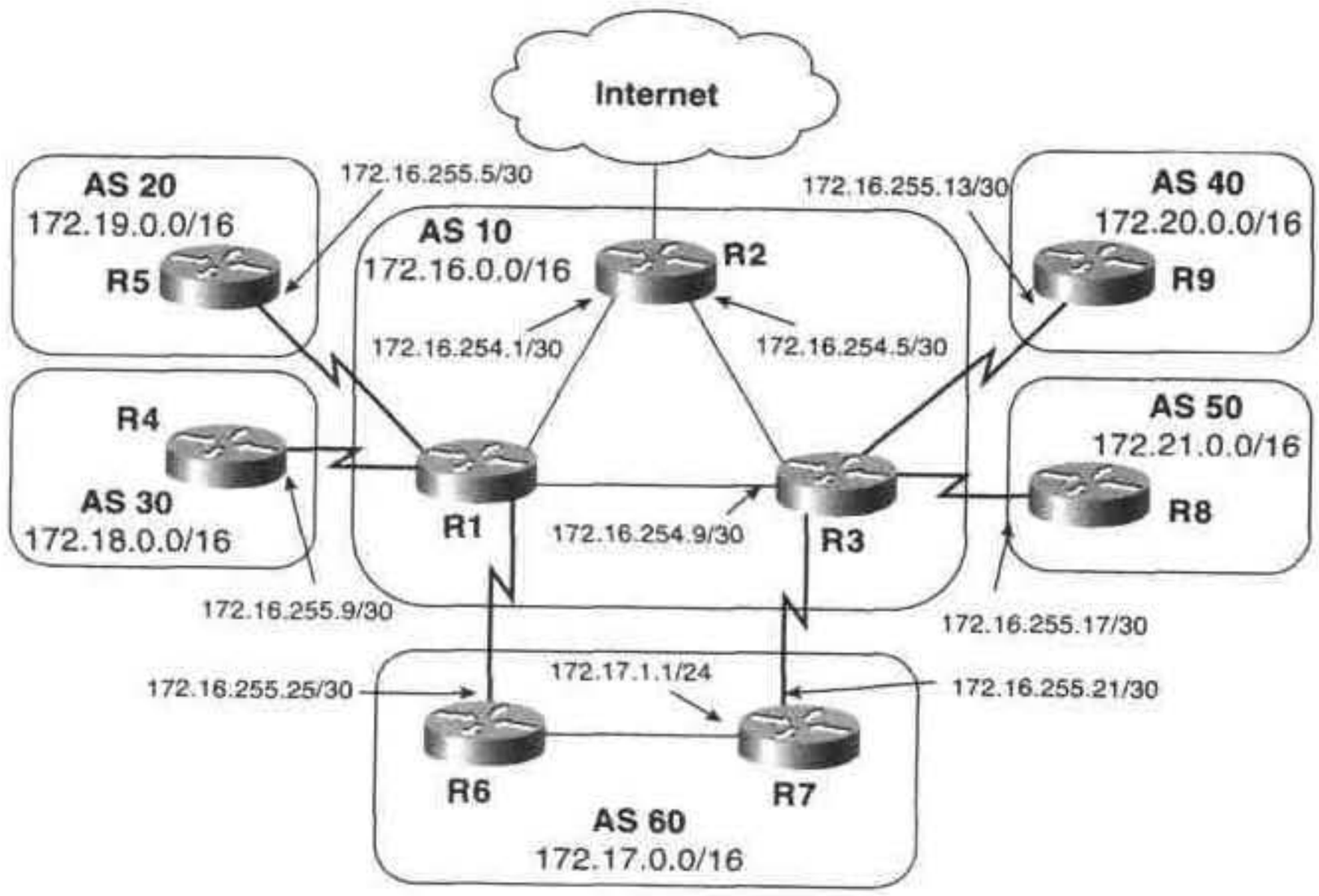


图 3-37　解决问题练习 1~6 的网络互连图

1．例 3-165 给出了图 3-37 中路由器 R2 的 BGP 配置。

例 3-165　路由器 R2 的 BGP 配置

```
router bgp 10
 no synchronization
 network 0.0.0.0
 neighbor 172.16.254.2 remote-as 10
 neighbor 172.16.254.2 next-hop-self
 neighbor 172.16.254.6 remote-as 10
 neighbor 172.16.254.6 next-hop-self
 no auto-summary
!
ip classless
ip route 0.0.0.0 0.0.0.0 Ethernet10
```

例 3-166 给出了 R2 的 BGP 表和路由表。在图 3-37 中，虽然有路由到 AS 内的目的地，但是 Ping 这些目的地却失败了。为什么？

例 3-166　图 3-37 中 R2 的 BGP 和路由表

```
R2#show ip bgp
BGP table version is 7, local router ID is 10.1.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 0.0.0.0          0.0.0.0                  0         32768 i
*>i172.17.0.0       172.16.255.21            0    100      0 60 i
*>i172.18.0.0       172.16.255.9             0    100      0 30 i
*>i172.19.0.0       172.16.255.5             0    100      0 20 i
*>i172.20.0.0       172.16.255.13            0    100      0 40 i
*>i172.21.0.0       172.16.255.17            0    100      0 50 i

R2#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default

Gateway of last resort is 0.0.0.0 to network 0.0.0.0

     10.0.0.0 255.255.255.0 is subnetted, 1 subnets
C       10.1.1.0 is directly connected, Ethernet11
B    172.20.0.0 [200/0] via 172.16.255.13, 00:01:15
B    172.21.0.0 [200/0] via 172.16.255.17, 00:01:16
     172.16.0.0 255.255.255.252 is subnetted, 2 subnets
C       172.16.254.0 is directly connected, Ethernet12
C       172.16.254.4 is directly connected, Ethernet13
B    172.17.0.0 [200/0] via 172.16.255.21, 00:01:16
B    172.18.0.0 [200/0] via 172.16.255.9, 00:00:59
B    172.19.0.0 [200/0] via 172.16.255.5, 00:00:59
S*   0.0.0.0 0.0.0.0 is directly connected, Ethernet10
R2#ping 172.17.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.17.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
R2#
```

答案：R2 没有到达 BGP 路由表中的下一跳地址的路由。R1 与 R3 必须宣告到达这些地址的路由，或使用 neighbor next-hop-solf 命令。

2. 例 3-167 给出了图 3-37 中路由器 R1 和 R5 的 **debug** 输出。这些信息暗示了什么问题？

例 3-167　图 3-37 中 R1 与 R5 的 debug 输出

```
R1#debug ip bgp
BGP debugging is on
R1#
BGP: 172.16.255.5 open active, local address 172.16.255.6
BGP: 172.16.255.5 sending OPEN, version 4
BGP: 172.16.255.5 received NOTIFICATION 2/2 (peer in wrong AS) 2 bytes 000A
BGP: 172.16.255.5 closing
```

```
R5#
6d08h: BGP: 172.16.255.6 open active, delay 28272ms
6d08h: BGP: 172.16.255.6 open active, local address 172.16.255.5
6d08h: BGP: 172.16.255.6 sending OPEN, version 4
6d08h: BGP: 172.16.255.6 OPEN rcvd, version 4
```

```
6d08h: BGP: 172.16.255.6 bad OPEN, remote AS is 10, expected 30
6d08h: BGP: 172.16.255.6 sending NOTIFICATION 2/2 (peer in wrong AS) 2 bytes 000A
6d08h: BGP: 172.16.255.6 remote close, state CLOSEWAIT
6d08h: BGP: 172.16.255.6 closing
```

答案：R5 的 BGP 配置包含了 neighbor 172.16.255.6 remote-as 30 的声明，这个声明应为 neighbor 172.16.255.6 remote-as 10

3．例 3-168 给出了图 3-37 中 R1 和 R3 的 BGP 表。第一个表指示通过 R6(172.16.255.25)或者 R3(172.16.254.9)都可以到达 172.17.0.0/24。R1 会选择哪条路径，为什么？

例 3-168　图 3-37 中 R1 和 R3 的 BGP 表

```
R1#show ip bgp
BGP table version is 8, local router ID is 172.20.7.1
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i10.0.0.0         172.16.254.1             0    100      0 i
* i172.17.0.0       172.16.254.9             0    100      0 60 i
*>                  172.16.255.25            0             0 60 i
*> 172.18.0.0       172.16.255.9             0             0 30 i
*> 172.19.0.0       172.16.255.5             0             0 20 i
*>i172.20.0.0       172.16.254.9             0    100      0 40 i
*>i172.21.0.0       172.16.254.9             0    100      0 50 i
R1#
```

```
R3#show ip bgp
BGP table version is 5, local router ID is 172.16.255.22
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i0.0.0.0          172.16.254.5             0    100      0 i
* i172.17.0.0       172.16.254.10            0    100      0 60 i
*>                  172.16.255.21            0             0 60 i
* i172.18.0.0       172.16.254.10            0    100      0 30 i
* i172.19.0.0       172.16.254.10            0    100      0 20 i
*> 172.20.0.0       172.16.255.13            0             0 40 i
*> 172.21.0.0       172.16.255.17            0             0 50 i
R3#
```

答案：R1 使用了通过 R6 的路径，因为 EBGP 通路优于 IBGP 通路。

4．例 3-169 给出了图 3-37 中 R1、R3、R6 和 R7 的 BGP 和 IGP 配置。

例 3-169　路由器 R1、R3、R6 和 R7 的 BGP 和 IGP 配置

```
R1
router bgp 10
 neighbor 172.16.254.1 remote-as 10
 neighbor 172.16.254.1 next-hop-self
 neighbor 172.16.254.9 remote-as 10
 neighbor 172.16.254.9 next-hop-self
 neighbor 172.16.255.5 remote-as 20
 neighbor 172.16.255.9 remote-as 30
 neighbor 172.16.255.25 remote-as 60
```

```
R3
router bgp 10
 neighbor 172.16.254.5 remote-as 10
 neighbor 172.16.254.5 next-hop-self
```

```
 neighbor 172.16.255.13 remote-as 40
 neighbor 172.16.255.17 remote-as 50
 neighbor 172.16.255.21 remote-as 60
 neighbor 172.16.255.21 next-hop-self
```

R6

```
router eigrp 60
 redistribute bgp 60 metric 1000 100 255 1 1500
 network 172.17.0.0
!
router bgp 60
 network 172.17.0.0
 neighbor 172.16.255.26 remote-as 10
```

R7

```
router eigrp 60
 redistribute bgp 60 metric 1000 100 255 1 1500
 network 172.17.0.0
!
router bgp 60
 network 172.17.0.0
 neighbor 172.16.255.22 remote-as 10
```

例3-168给出了R1和R3的BGP表。对于下面给出的每一个目的地，R6会选用哪个下一跳地址？解释一下R6选用这些地址的原因。

目的地:

172.20.7.102

172.18.58.35

10.53.12.6

答案:

172.20.7.102：下一跳为172.17.1.1

172.18.58.35：下一跳为172.16.255.26

10.53.12.6：包被丢弃

R1与R3都没有同步关闭。因此，每次只公布从EBGP邻居处学到的路由。R6从R1学到172.18.0.0/24，但是R1没有宣告172.20.0.0/24路由，这是从IBGP邻居处学到的。R3宣告了到R7的路由，这条路由是通过EIGRP学到的。R1与R3从IBGP邻居R2处学到了默认路由，所以这两个路由器均不对外宣告这个默认路由。

5．例3-170给出了图3-37中R1和R3的BGP配置。

例3-170 路由器R1和R3的BGP配置

R1

```
router bgp 10
no synchronization
aggregate-address 172.16.0.0 255.255.248.0 summary-only
neighbor 172.16.254.1 remote-as 10
neighbor 172.16.254.1 next-hop-self
neighbor 172.16.254.9 remote-as 10
neighbor 172.16.254.9 next-hop-self
neighbor 172.16.255.5 remote-as 20
neighbor 172.16.255.9 remote-as 30
neighbor 172.16.255.25 remote-as 60
```

```
R3
router bgp 10
 no synchronization
 aggregate-address 172.16.0.0 255.255.248.0 summary-only
 neighbor 172.16.254.5 remote-as 10
 neighbor 172.16.254.5 next-hop-self
 neighbor 172.16.254.10 remote-as 10
 neighbor 172.16.254.10 next-hop-self
 neighbor 172.16.255.13 remote-as 40
 neighbor 172.16.255.17 remote-as 50
 neighbor 172.16.255.21 remote-as 60
 neighbor 172.16.255.21 next-hop-self
```

该配置的目的是抑制所有的更具体路由而只公布聚合路由。但是在例 3-171 R8 的 BGP 表中还是显示了更具体路由。问题出在哪里？

例 3-171　图 3-37 中 R8 的 BGP 表

```
R8#show ip bgp
BGP table version is 163, local router ID is 172.21.1.1
Status codes: s suppressed, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 0.0.0.0          172.16.255.18                          0 10 i
*> 172.17.0.0       172.16.255.18                          0 10 60 i
*> 172.18.0.0       172.16.255.18                          0 10 30 i
*> 172.19.0.0       172.16.255.18                          0 10 20 i
*> 172.20.0.0       172.16.255.18                          0 10 40 i
*> 172.21.0.0       0.0.0.0                  0         32768 i
R8#
```

答案：在 aggregate-address 命令中规定的掩码应为 255.248.0.0。这个规定的聚合不匹配 R1 与 R3 路由表中的任何路由，所以不会对外宣告。

6. 在图 3-37 中，来自 AS 60，目的地是任何其他 AS 的数据包都应该通过 R6 和 R1 之间的链路进行转发，R7 和 R3 之间的链路应该只用做这些业务量的备份路由，但是目的地是 Internet 的业务量可以使用该链路。为了执行这个策略，R3 应该只公布缺省路由以及聚合路由 172.16.0.0/13 而 R1 应该公布更具体路由。例 3-172 给出了路由器 R1、R3、R6 和 R7 的配置。例 3-173 给出了 R7 的路由表。本例中提出的要求能够达到吗？如果不能，为什么？

例 3-172　路由器 R1、R3、R6 和 R7 的配置

```
R1
router bgp 10
no synchronization
neighbor 172.16.254.1 remote-as 10
neighbor 172.16.254.1 next-hop-self
neighbor 172.16.254.9 remote-as 10
neighbor 172.16.254.9 next-hop-self
neighbor 172.16.255.5 remote-as 20
neighbor 172.16.255.9 remote-as 30
neighbor 172.16.255.25 remote-as 60
```

R3

```
router bgp 10
 no synchronization
 aggregate-address 172.16.0.0 255.248.0.0 summary-only
 neighbor 172.16.254.5 remote-as 10
 neighbor 172.16.254.5 next-hop-self
 neighbor 172.16.254.10 remote-as 10
 neighbor 172.16.254.10 next-hop-self
 neighbor 172.16.255.13 remote-as 40
 neighbor 172.16.255.17 remote-as 50
 neighbor 172.16.255.21 remote-as 60
 neighbor 172.16.255.21 next-hop-self
```

R6

```
redistribute bgp 60 metric 1000 100 255 1 1500
 network 172.17.0.0
!
router bgp 60
 network 172.17.0.0
 neighbor 172.16.255.26 remote-as 10
```

R7

```
router eigrp 60
 redistribute bgp 60 metric 1000 100 255 1 1500
 network 172.17.0.0
!
router bgp 60
 network 172.17.0.0
 neighbor 172.16.255.22 remote-as 10
```

例 3-173　故障排除练习 6 中 R7 的路由表

```
R7#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
       U - per-user static route, o - ODR
       T - traffic engineered route

Gateway of last resort is 172.16.255.22 to network 0.0.0.0

     172.17.0.0/24 is subnetted, 3 subnets
C       172.17.1.0 is directly connected, Ethernet0
D       172.17.3.0 [90/409600] via 172.17.1.2, 09:18:50, Ethernet0
C       172.17.2.0 is directly connected, Ethernet1
     172.16.0.0/30 is subnetted, 1 subnets
C       172.16.255.20 is directly connected, Serial0
D EX 172.19.0.0/16 [170/2611200] via 172.17.1.2, 00:19:08, Ethernet0
D EX 172.18.0.0/16 [170/2611200] via 172.17.1.2, 00:19:08, Ethernet0
B*   0.0.0.0/0 [20/0] via 172.16.255.22, 00:18:37
B    172.16.0.0/13 [20/0] via 172.16.255.22, 00:18:09
R7#
```

答案：这个目标没有完全实现。到 172.18.0.0/24 与 172.19.0.0/24 的路由是正确的。但是

到 172.20.0.0/24 与 172.21.0.0/24 的路由不在路由表中，R3 处 aggregate-addrss 命令不仅抑制了向 R3 EBGP 对端，也抑制了对 IBGP 对端，宣告更具体路由。因此，R1 不会知道 172.20.0.0/24 与 172.21.0.0/24。

7．让我们重新看一下图 3-19 和例 3-98 以及相关的讨论。Meribel 将它的本地路由 172.17.0.0 公布给它的 EBGP 对等，公布路由的 ORIGIN 为 Incomplete，然而 Lillehammer 以 ORIGIN 为 IGP 的形式将该路由再次公布给了 Meribel,这样会导致路由环路吗？

答案：否。虽然在 BGP 判断过程中，ORIGIN 为 IGP 优于 ORIGIN 为 Incomplete，但是管理权重优于 ORIGIN。缺省情况下 Meribel 对本地产生的路由分配权重为 32768，对于学到的路由分配权重为 0，所以本地的路由更优先。

8．例 3-174 给出了图 3-24 中路由器 Colorado 的配置。图 3-24 中所有的路由器 ID 都是在环回接口上配置的，而且路由器上除了 BGP 以外没有运行其他的路由协议。假设图中所有的链路工作都正常，其他五个路由器都是 Colorado 的 EBGP 对等吗？如果不是，为什么？

例 3-174　图 3-24 中路由器 Colorado 的配置

```
router bgp 100
 network 10.1.11.0 mask 255.255.255.0
 network 10.1.12.0 mask 255.255.255.0
 neighbor CLIENTS peer-group
 neighbor CLIENTS ebgp-multihop 2
 neighbor CLIENTS update-source Loopback2
 neighbor CLIENTS filter-list 2 in
 neighbor CLIENTS filter-list 1 out
 neighbor 10.1.255.2 remote-as 200
 neighbor 10.1.255.2 peer-group CLIENTS
 neighbor 10.1.255.3 remote-as 300
 neighbor 10.1.255.3 peer-group CLIENTS
 neighbor 10.1.255.4 remote-as 400
 neighbor 10.1.255.4 peer-group CLIENTS
 neighbor 10.1.255.5 remote-as 500
 neighbor 10.1.255.5 peer-group CLIENTS
 neighbor 10.1.255.6 remote-as 600
 neighbor 10.1.255.6 peer-group CLIENTS
 no auto-summary
!
ip classless
ip route 10.1.255.2 255.255.255.255 Serial0/1.305
ip route 10.1.255.3 255.255.255.255 Serial0/1.306
ip route 10.1.255.4 255.255.255.255 Serial0/1.307
ip route 10.1.255.5 255.255.255.255 Serial0/1.308
!
ip as-path access-list 1 permit ^$
ip as-path access-list 2 permit ^[2-6]00$
```

答案：否。路由器 NewHampshire 不是一个对端，因为没有静态路由条目到达 Colorado 的地址 10.1.255.6/32。

9．参考故障排除练习 8 中对图 3-24 中路由器 Colorado 的配置。如果在配置中将 **no atuo-summary** 命令去掉，会产生什么结果？

答案：去掉这个声明对拓扑没有影响如图 3-82，因为所有的路由器 ID 与自治域的地址是 10.0.0.0 的子网。

10. 参考故障排除练习 8 中的配置，入站路由过滤器允许的路由有哪些？

答案：输入路由过滤涉及 AS_PATH 列表 2，凡是路由 AS_PATH 满足下述准则的均被允许：

- 路由的 AS_PATH 只有一个 AS 号。
- 10 进制表示的 AS 号必须为 3 位数字。
- 第 1 位数字必须在 2 和 6 间，包括 2 与 6。
- 第 2 位与第 3 位数字必须为 0。

11. 参考图 3-24 以及故障排除练习 8 中路由器 Colorado 的配置。除了它自己 AS 的本地子网或者 AS 间的链路以外，子网 10.1.3.0/24 上的一个主机可以 ping 通哪个子网？

答案：只有子网 10.1.11.0/24，10.1.12.0/24 与 10.1.255.1/32。在 Colorado 的出站路由过滤防止它的 EBGP 对端学习不到本地的路由。

第 4 章　排除故障练习答案

1. 指出例 4-33 配置中的错误。

例 4-33　排错练习 1 的配置

```
ip nat pool EX1 192.168.1.1 192.168.1.254 netmask 255.255.255.0 type match-host
ip nat pool EX1A netmask 255.255.255.240
 address 172.21.1.33 172.21.1.38
 address 172.21.1.40 172.21.1.46
ip nat inside source list 1 pool EX1
ip nat inside source static 10.18.53.210 192.168.1.1
ip nat outside source list 2 pool EX1A
!
access-list 1 permit 10.0.0.0 0.255.255.255
access-list 2 permit 192.168.2.0 0.0.0.255
```

答案：静态映射 IG 地址与池 EX1 中的地址重叠。

2. 图 4-30 中的 RTR1 连接到有重叠地址的两个网络。路由器上的 NAT 配置如例 4-34 所示，但设备不能经路由器相互通信，有什么错误？

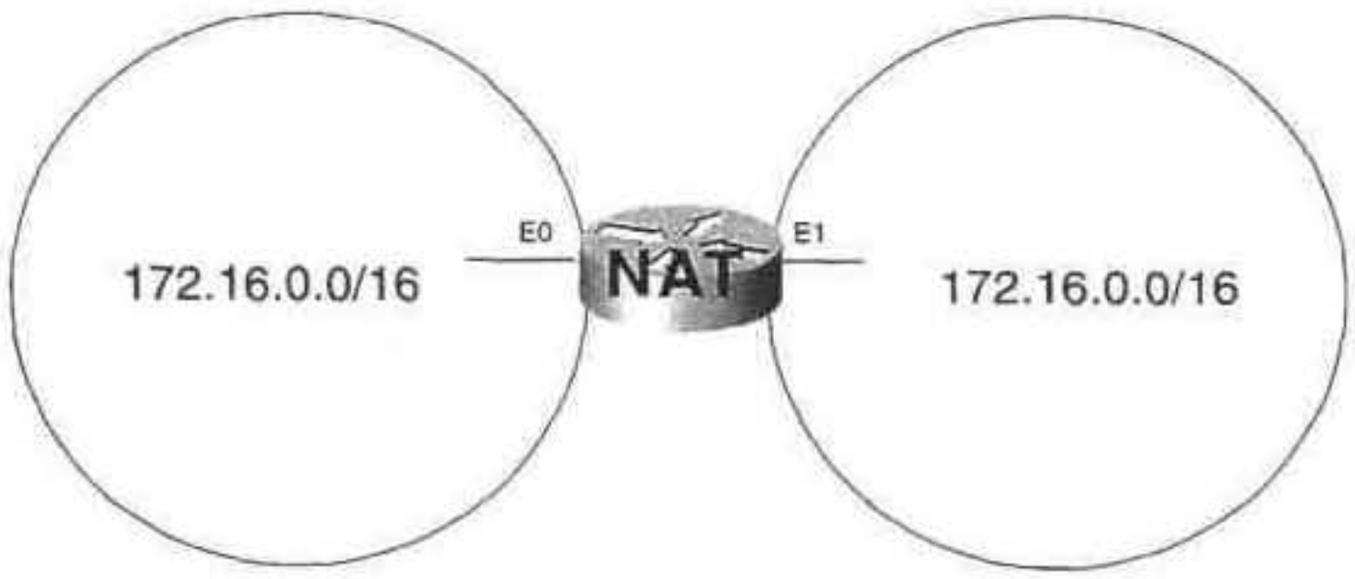


图 4-30　排错练习 2 的网络拓扑

例 4-34　排错练习 2 中的配置

```
interface Ethernet0
 ip address 172.16.10.1 255.255.255.0
 ip nat inside
!
interface Ethernet1
 ip address 172.16.255.254 255.255.255.0
 ip nat outside
!
router ospf 1
 redistribute static metric 10 metric-type 1 subnets
 network 10.0.0.0 0.255.255.255 area 0
!
ip nat translation timeout 500
ip nat pool NET1 10.1.1.1 10.1.255.254 netmask 255.255.0.0
ip nat pool NET2 192.168.1.1 192.168.255.254 netmask 255.255.0.0
ip nat inside source list 1 pool NET1
ip nat outside source list 1 pool NET2
!
ip classless
!
ip route 10.1.0.0 255.255.0.0 Ethernet0
ip route 192.168.0.0 255.255.0.0 Ethernet1
!
access-list 1 permit 172.16.0.0 0.0.255.255
```

答案：问题不是在于 NAT 本身，而是路由的问题。所有的地址翻译是动态的，对于两边的主机没有办法判断向哪一个地址发送包可以到达另一边。

3．参考图 4-21 中的 Cozumel 与 Guaymas 的配置。如果访问表的第一行去掉，会有什么结果？Guaymas 与 Cozumel 能不能 ping 通对方？

答案：当任一台路由器发从接口 E1 发送一个包，源地址翻译成一个 IG 地址池中的地址。这两台路由器可以仍 ping 对方，即使源地址已经被翻译了。如果 Cozumel ping Guaymas，它的源地址 10.255.13.254 会被翻译成 206.100.176.50。虽然 Guaymas 不认识这个地址在它直连的子网上，它有一条 206.100.176.0/20 的路由指向 Cozumel。当它响应这个 ping，响应前转到 Cozumel，Cozumel 把这目地地址翻译成 10.255.13.254。

第 6 章　故障排除练习答案

1．例 6-63 说明了什么？

例 6-63　排错练习 1 的输出

```
R1#
Turban#debug ip mpacket
IP multicast packets debugging is on
R1#
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
```

```
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
IP: s=192.168.14.35 (Serial0/1.307) d=228.13.20.216 len 573, mrouting disabled
```

答案：因为多播路由没有在路由器上使用，故多播包会被丢弃。

2．例 6-64 的输出告诉你什么？

例 6-64　排错练习 2 的输出

```
R2#
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
IP: s=192.168.13.5 (Ethernet0) d=227.134.14.26 len 583, not RPF interface
```

答案：组 227.134.14.26 的包，源于 192.168.13.5，它在接口 E0 上收到。但是，这个接口明显不是上游接口，因此它不是 RPF 接口，这个包 RPF 检测失败，被丢弃。

3．例 6-65 的输出告诉你什么？

例 6-65　排错练习 3 的输出

```
R3#debug ip mpacket
IP multicast packets debugging is on
R3#
IP: s=172.16.3.50 (Serial0.405) d=224.0.1.40 (Serial0.407) len 52, mforward
IP: s=172.16.3.50 (Ethernet0) d=224.0.1.40 len 62, not RPF interface
IP: s=172.16.3.50 (Ethernet0) d=224.0.1.39 len 62, not RPF interface
IP: s=172.16.3.50 (Serial0.405) d=224.0.1.39 (Serial0.407) len 52, mforward
```

答案：一台地址 172.16.3.50 的路由器同时是 C-RP(224.0.1.39)与映射代理(224.0.1.40)。这个 Auto-RP 消息在接口 S0.405 上收到，从接口 S0.407 前转出去。这个消息也在接口 E0 上收到，没有通过 RPF 检测，因此接口 S0.405 是 172.16.3.50 的上游接口。

4．在图 6-12 中，哪一个路由器是 PIM 指定路由器？

答案：PIM DR 是有最高 IP 地址值的路由器，因此 RT4 为 PIM DR。

5．在图 6-12 中，哪一个路由器向组员发送 IGMPv2 查询消息？

答案：IGMPv2 查询者为有最小 IP 地址值的路由器，因此 RT2 成为查询者。

6．表 6-5 显示了图 6-12 中到源 172.16.12.18 的的单播路由表。哪一台路由器是 PIM 前转器？

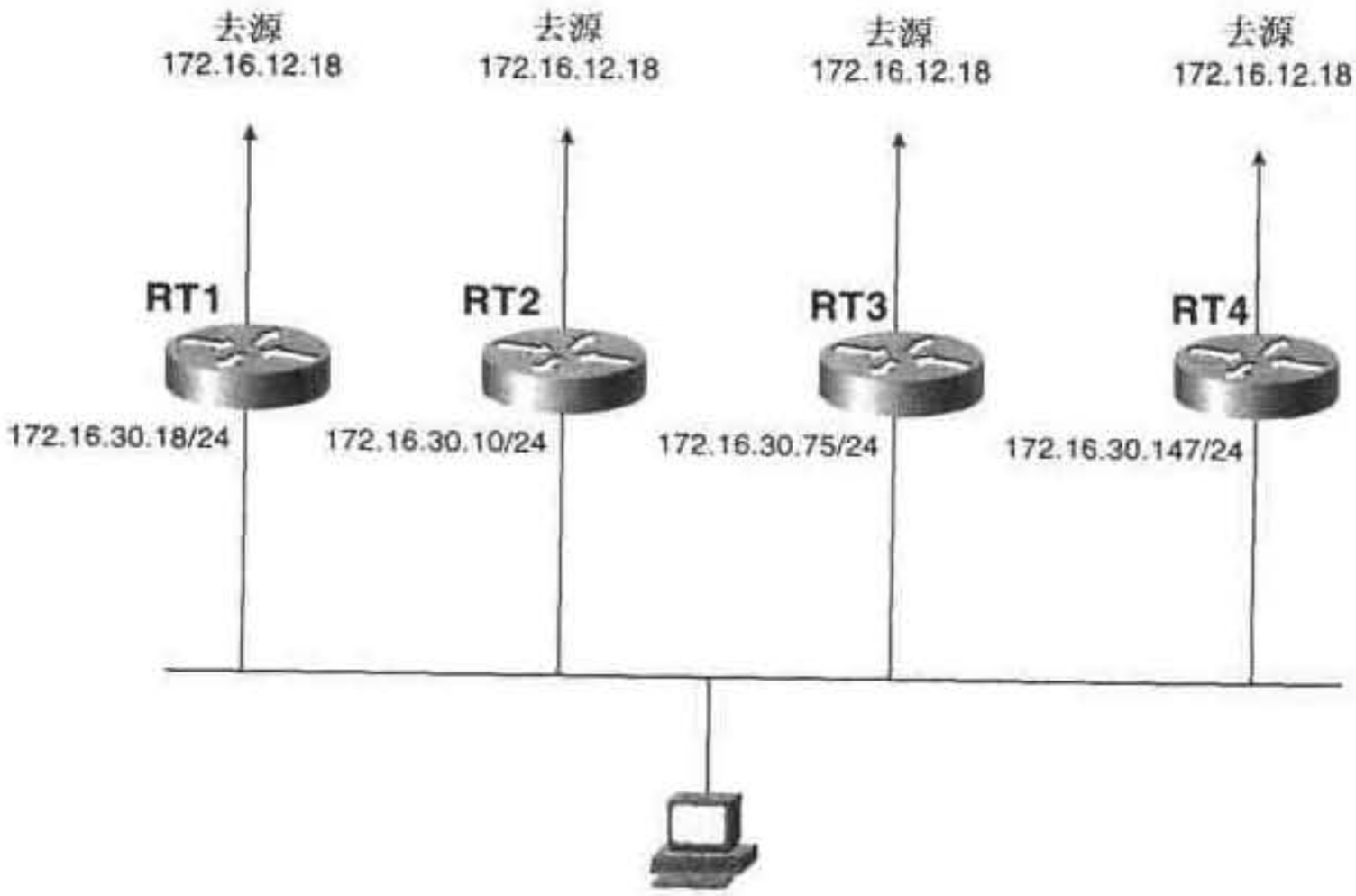


图 6-12　排错练习 4、5、6 的拓扑

表 6-5　图 6-12 中 172.16.12.18 的单播路由

路　由　器	下　一　跳	协　　议	量　　度
R1	172.16 50.5	OSPF	35
R2	172.16.51.80	EIGRP	307200
R3	172.16.13.200	EIGRP	2297856
R4	172..16.44.1	OSPF	83

答案：PIM 前转器是有最低管理距离的路由器。在管理距离相同的情况下，前转器为有最近量度的路由器。EIGRP 的管理距离为 90，OSPF 的为 110，所以 EIGRP 最低。在两个 EIGRP 间，R2 的路由有最低度量，所以 R2 为 PIM 前转器。

7．例 6-66 中显示了图 6-10 中在一个 PIM 域中的 RPF 跟踪结果，这个域中运行 RIP-2 作为它的单播 IGP，这个结果是否说明了可能的问题？

例 6-66　排错练习 7 中的 mtrace

```
Sombrero#mtrace 192.168.14.35 192.168.10.8 235.1.2.3
Type escape sequence to abort.
Mtrace from 192.168.14.35 to 192.168.10.8 via group 235.1.2.3
From source (?) to destination (?)
Querying full reverse path...
 0  192.168.10.8
-1  192.168.10.1 PIM  [192.168.14.0/24]
-2  192.168.200.2 PIM  [192.168.14.0/24]
-3  192.168.201.2 PIM  [192.168.14.0/24]
```

```
-4   192.168.204.1 PIM  [192.168.14.0/24]
-5   192.168.14.35
Sombrero#
```

答案：是。在 Beret 与 Boater 间有多条经过 Turban 或 Fez 的路径。Beret 只可以有一个 RPF 邻居，所以它选出有最高 IP 地址值的邻居。在这个案例中，Turban 有最高的 IP 地址值。但跟踪显示选用的路径是通过 Fez 的。因此，Beret 与 Turban 间有明显的问题。

224.0.0.18	VRRP	[Hinden]
224.0.0.19	IPAllL1ISs	[Przygienda]
224.0.0.20	IPAllL2ISs	[Przygienda]
224.0.0.21	IPAllIntermediate Systems	[Przygienda]
224.0.0.22	IGMP	[Deering]
224.0.0.23	GLOBECAST-ID	[Scannell]
224.0.0.24	Unassigned	[JBP]
224.0.0.25	router-to-switch	[Wu]
224.0.0.26	Unassigned	[JBP]
224.0.0.27	Al MPP Hello	[Martinicky]
224.0.0.28	ETC Control	[Polishinski]
224.0.0.29	GE-FANUC	[Wacey]
224.0.0.30	indigo-vhdp	[Caughie]
224.0.0.31	shinbroadband	[Kittivatcharapong]
224.0.0.32	digistar	[Kerkan]
224.0.0.33	ff-system-management	[Glanzer]
224.0.0.34	pt2-discover	[Kammerlander]
224.0.0.35	DXCLUSTER	[Koopman]
224.0.0.36-224.0.0.250	Unassigned	[JBP]
224.0.0.251	mDNS	[Cheshire]
224.0.0.252-224.0.0.255	Unassigned	[JBP]
224.0.1.0	VMTP Managers Group	[RFC1045,DRC3]
224.0.1.1	NTP Network Time Protocol	[RFC1119,DLM1]
224.0.1.2	SGI-Dogfight	[AXC]
224.0.1.3	Rwhod	[SXD]
224.0.1.4	VNP	[DRC3]

224.0.1.5	Artificial Horizons - Aviator	[BXF]
224.0.1.6	NSS - Name Service Server	[BXS2]
224.0.1.7	AUDIONEWS - Audio News Multicast	[MXF2]
224.0.1.8	SUN NIS+ Information Service	[CXM3]
224.0.1.9	MTP Multicast Transport Protocol	[SXA]
224.0.1.10	IETF-1-LOW-AUDIO	[SC3]
224.0.1.11	IETF-1-AUDIO	[SC3]
224.0.1.12	IETF-1-VIDEO	[SC3]
224.0.1.13	IETF-2-LOW-AUDIO	[SC3]
224.0.1.14	IETF-2-AUDIO	[SC3]
224.0.1.15	IETF-2-VIDEO	[SC3]
224.0.1.16	MUSIC-SERVICE	[Guido van Rossum]
224.0.1.17	SEANET-TELEMETRY	[Andrew Maffei]
224.0.1.18	SEANET-IMAGE	[Andrew Maffei]
224.0.1.19	MLOADD	[Braden]
224.0.1.20	any private experiment	[JBP]
224.0.1.21	DVMRP on MOSPF	[John Moy]
224.0.1.22	SVRLOC	[Veizades]
224.0.1.23	XINGTV	[Gordon]
224.0.1.24	microsoft-ds	<arnoldm@microsoft.com>
224.0.1.25	nbc-pro	<bloomer@birch.crd.ge.com>
224.0.1.26	nbc-pfn	<bloomer@birch.crd.ge.com>
224.0.1.27	lmsc-calren-1	[Uang]
224.0.1.28	lmsc-calren-2	[Uang]
224.0.1.29	lmsc-calren-3	[Uang]
224.0.1.30	lmsc-calren-4	[Uang]

224.0.1.31	ampr-info	[Janssen]
224.0.1.32	mtrace	[Casner]
224.0.1.33	RSVP-encap-1	[Braden]
224.0.1.34	RSVP-encap-2	[Braden]
224.0.1.35	SVRLOC-DA	[Veizades]
224.0.1.36	rln-server	[Kean]
224.0.1.37	proshare-mc	[Lewis]
224.0.1.38	dantz	[Zulch]
224.0.1.39	cisco-rp-announce	[Farinacci]
224.0.1.40	cisco-rp-discovery	[Farinacci]
224.0.1.41	gatekeeper	[Toga]
224.0.1.42	iberiagames	[Marocho]
224.0.1.43	nwn-discovery	[Zwemmer]
224.0.1.44	nwn-adaptor	[Zwemmer]
224.0.1.45	isma-1	[Dunne]
224.0.1.46	isma-2	[Dunne]
224.0.1.47	telerate	[Peng]
224.0.1.48	ciena	[Rodbell]
224.0.1.49	dcap-servers	[RFC2114]
224.0.1.50	dcap-clients	[RFC2114]
224.0.1.51	mcntp-directory	[Rupp]
224.0.1.52	mbone-vcr-directory	[Holfelder]
224.0.1.53	heartbeat	[Mamakos]
224.0.1.54	sun-mc-grp	[DeMoney]
224.0.1.55	extended-sys	[Poole]
224.0.1.56	pdrncs	[Wissenbach]
224.0.1.57	tns-adv-multi	[Albin]
224.0.1.58	vcals-dmu	[Shindoh]

224.0.1.59	zuba	[Jackson]
224.0.1.60	hp-device-disc	[Albright]
224.0.1.61	tms-production	[Gilani]
224.0.1.62	sunscalar	[Gibson]
224.0.1.63	mmtp-poll	[Costales]
224.0.1.64	compaq-peer	[Volpe]
224.0.1.65	iapp	[Meier]
224.0.1.66	multihasc-com	[Brockbank]
224.0.1.67	serv-discovery	[Honton]
224.0.1.68	mdhcpdisover	[RFC2730]
224.0.1.69	MMP-bundle-discovery1	[Malkin]
224.0.1.70	MMP-bundle-discovery2	[Malkin]
224.0.1.71	XYPOINT DGPS Data Feed	[Green]
224.0.1.72	GilatSkySurfer	[Gal]
224.0.1.73	SharesLive	[Rowatt]
224.0.1.74	NorthernData	[Sheers]
224.0.1.75	SIP	[Schulzrinne]
224.0.1.76	IAPP	[Moelard]
224.0.1.77	AGENTVIEW	[Iyer]
224.0.1.78	Tibco Multicast1	[Shum]
224.0.1.79	Tibco Multicast2	[Shum]
224.0.1.80	MSP	[Caves]
224.0.1.81	OTT (One-way Trip Time)	[Schwartz]]
224.0.1.82	TRACKTICKER	[Novick]
224.0.1.83	dtn-mc	[Gaddie]
224.0.1.84	jini-announcement	[Scheifler]
224.0.1.85	jini-request	[Scheifler]
224.0.1.86	sde-discovery	[Aronson]

224.0.1.87	DirecPC-SI	[Dillon]
224.0.1.88	B1RMonitor	[Purkiss]
224.0.1.89	3Com-AMP3 dRMON	[Banthia]
224.0.1.90	imFtmSvc	[Bhatti]
224.0.1.91	NQDS4	[Flynn]
224.0.1.92	NQDS5	[Flynn]
224.0.1.93	NQDS6	[Flynn]
224.0.1.94	NLVL12	[Flynn]
224.0.1.95	NTDS1	[Flynn]
224.0.1.96	NTDS2	[Flynn]
224.0.1.97	NODSA	[Flynn]
224.0.1.98	NODSB	[Flynn]
224.0.1.99	NODSC	[Flynn]
224.0.1.100	NODSD	[Flynn]
224.0.1.101	NQDS4R	[Flynn]
224.0.1.102	NQDS5R	[Flynn]
224.0.1.103	NQDS6R	[Flynn]
224.0.1.104	NLVL12R	[Flynn]
224.0.1.105	NTDS1R	[Flynn]
224.0.1.106	NTDS2R	[Flynn]
224.0.1.107	NODSAR	[Flynn]
224.0.1.108	NODSBR	[Flynn]
224.0.1.109	NODSCR	[Flynn]
224.0.1.110	NODSDR	[Flynn]
224.0.1.111	MRM	[Wei]
224.0.1.112	TVE-FILE	[Blackketter]
224.0.1.113	TVE-ANNOUNCE	[Blackketter]
224.0.1.114	Mac Srv Loc	[Woodcock]

224.0.1.115	Simple Multicast	[Crowcroft]
224.0.1.116	SpectraLinkGW	[Hamilton]
224.0.1.117	dieboldmcast	[Marsh]
224.0.1.118	Tivoli Systems	[Gabriel]
224.0.1.119	pq-lic-mcast	[Sledge]
224.0.1.120	HYPERFEED	[Kreutzjans]
224.0.1.121	Pipesplatform	[Dissett]
224.0.1.122	LiebDevMgmg-DM	[Velten]
224.0.1.123	TRIBALVOICE	[Thompson]
224.0.1.124	UDLR-DTCP	[Cipiere]
224.0.1.125	PolyCom Relay1	[Coutiere]
224.0.1.126	Infront Multi1	[Lindeman]
224.0.1.127	XRX DEVICE DISC	[Wang]
224.0.1.128	CNN	[Lynch]
224.0.1.129	PTP-primary	[Eidson]
224.0.1.130	PTP-alternate1	[Eidson]
224.0.1.131	PTP-alternate2	[Eidson]
224.0.1.132	PTP-alternate3	[Eidson]
224.0.1.133	ProCast	[Revzen]
224.0.1.134	3Com Discp	[White]
224.0.1.135	CS-Multicasting	[Stanev]
224.0.1.136	TS-MC-1	[Sveistrup]
224.0.1.137	Make Source	[Daga]
224.0.1.138	Teleborsa	[Strazzera]
224.0.1.139	SUMAConfig	[Wallach]
224.0.1.140	Unassigned	
224.0.1.141	DHCP-SERVERS	[Hall]
224.0.1.142	CN Router-LL	[Armitage]

224.0.1.143	EMWIN	[Querubin]
224.0.1.144	Alchemy Cluster	[O'Rourke]
224.0.1.145	Satcast One	[Nevell]
224.0.1.146	Satcast Two	[Nevell]
224.0.1.147	Satcast Three	[Nevell]
224.0.1.148	Intline	[Sliwinski]
224.0.1.149	8x8 Multicast	[Roper]
224.0.1.150	Unassigned	[JBP]
224.0.1.151	Intline-1	[Sliwinski]
224.0.1.152	Intline-2	[Sliwinski]
224.0.1.153	Intline-3	[Sliwinski]
224.0.1.154	Intline-4	[Sliwinski]
224.0.1.155	Intline-5	[Sliwinski]
224.0.1.156	Intline-6	[Sliwinski]
224.0.1.157	Intline-7	[Sliwinski]
224.0.1.158	Intline-8	[Sliwinski]
224.0.1.159	Intline-9	[Sliwinski]
224.0.1.160	Intline-10	[Sliwinski]
224.0.1.161	Intline-11	[Sliwinski]
224.0.1.162	Intline-12	[Sliwinski]
224.0.1.163	Intline-13	[Sliwinski]
224.0.1.164	Intline-14	[Sliwinski]
224.0.1.165	Intline-15	[Sliwinski]
224.0.1.166	marratech-cc	[Parnes]
224.0.1.167	EMS-InterDev	[Lyda]
224.0.1.168	itb301	[Rueskamp]
224.0.1.169	rtv-audio	[Adams]
224.0.1.170	rtv-video	[Adams]

224.0.1.171	HAVI-Sim	[Wasserroth]
224.0.1.172- 224.0.1.255	Unassigned	[JBP]
224.0.2.1	"rwho" Group (BSD) (unofficial)	[JBP]
224.0.2.2	SUN RPC PMAPPROC_CALLIT	[BXE1]
224.0.2.064- 224.0.2.095	SIAC MDD Service	[Tse]
224.0.2.096- 224.0.2.127	CoolCast	[Ballister]
224.0.2.128- 224.0.2.191	WOZ-Garage	[Marquardt]
224.0.2.192- 224.0.2.255	SIAC MDD Market Service	[Lamberg]
224.0.3.000- 224.0.3.255	RFE Generic Service	[DXS3]
224.0.4.000- 224.0.4.255	RFE Individual Conferences	[DXS3]
224.0.5.000- 224.0.5.127	CDPD Groups	[Bob Brenner]
224.0.5.128- 224.0.5.191	SIAC Market Service	[Cho]
224.0.5.192- 224.0.5.255	Unassigned	[IANA]
224.0.6.000- 224.0.6.127	Cornell ISIS Project	[Tim Clark]
224.0.6.128- 224.0.6.255	Unassigned	[IANA]
224.0.7.000- 224.0.7.255	Where-Are-You	[Simpson]
224.0.8.000- 224.0.8.255	INTV	[Tynan]
224.0.9.000- 224.0.9.255	Invisible Worlds	[Malamud]
224.0.10.000- 224.0.10.255	DLSw Groups	[Lee]

224.0.11.000- 224.0.11.255	NCC.NET Audio	[Rubin]
224.0.12.000- 224.0.12.063	Microsoft and MSNBC	[Blank]
224.0.13.000- 224.0.13.255	UUNET PIPEX Net News	[Barber]
224.0.14.000- 224.0.14.255	NLANR	[Wessels]
224.0.15.000- 224.0.15.255	Hewlett Packard	[van der Meulen]
224.0.16.000- 224.0.16.255	XingNet	[Uusitalo]
224.0.17.000- 224.0.17.031	Mercantile & Commodity Exchange	[Gilani]
224.0.17.032- 224.0.17.063	NDQMD1	[Nelson]
224.0.17.064- 224.0.17.127	ODN-DTV	[Hodges]
224.0.18.000- 224.0.18.255	Dow Jones	[Peng]
224.0.19.000- 224.0.19.063	Walt Disney Company	[Watson]
224.0.19.064- 224.0.19.095	Cal Multicast	[Moran]
224.0.19.096- 224.0.19.127	SIAC Market Service	[Roy]
224.0.19.128- 224.0.19.191	IIG Multicast	[Carr]
224.0.19.192- 224.0.19.207	Metropol	[Crawford]
224.0.19.208- 224.0.19.239	Xenoscience, Inc.	[Timm]
224.0.19.240- 224.0.19.255	HYPERFEED	[Felix]
224.0.20.000- 224.0.20.063	MS-IP/TV	[Wong]
224.0.20.064- 224.0.20.127	Reliable Network Solutions	[Vogels]

224.0.20.128- 224.0.20.143	TRACKTICKER Group	[Novick]
224.0.20.144- 224.0.20.207	CNR Rebroadcast MCA	[Sautter]
224.0.21.000- 224.0.21.127	Talarian MCAST	[Mendal]
224.0.22.000- 224.0.22.255	WORLD MCAST	[Stewart]
224.0.252.000- 224.0.252.255	Domain Scoped Group	[Fenner]
224.0.253.000- 224.0.253.255	Report Group	[Fenner]
224.0.254.000- 224.0.254.255	Query Group	[Fenner]
224.0.255.000- 224.0.255.255	Border Routers	[Fenner]
224.1.0.0- 224.1.255.255	ST Multicast Groups	[RFC1190,KS14]
224.2.0.0- 224.2.127.253	Multimedia Conference Calls	[SC3]
224.2.127.254	APv1 Announcements	[SC3]
224.2.127.255	SAPv0 Announcements (deprecated)	[SC3]
224.2.128.0- 224.2.255.255	SAP Dynamic Assignments	[SC3]
224.252.0.0- 224.255.255.255	DIS transient groups	[Joel Snyder]
225.0.0.0- 225.255.255.255	MALLOC (temp - renew 1/01)	[Handley]
232.0.0.0- 232.255.255.255	VMTP transient group see single-source-multicast file	[DRC3]
233.0.0.0- 233.255.255.255	Static Allocations (temp - renew 6/01)	[Meyer2]
239.000.000.000- 239.255.255.255	Administratively Scoped	[IANA,RFC2365]
239.000.000.000- 239.063.255.255	Reserved	[IANA]

239.064.000.000- 239.127.255.255	Reserved	[IANA]
239.128.000.000- 239.191.255.255	Reserved	[IANA]
239.192.000.000- 239.251.255.255	Organization-Local Scope	[Meyer,RFC2365]
239.252.000.000- 239.252.255.255	Site-Local Scope (reserved)	[Meyer,RFC2365]
239.253.000.000- 239.253.255.255	Site-Local Scope (reserved)	[Meyer,RFC2365]
239.254.000.000- 239.254.255.255	Site-Local Scope (reserved)	[Meyer,RFC2365]
239.255.000.000- 239.255.255.255	Site-Local Scope	[Meyer,RFC2365]
239.255.002.002	rasadv	[Thaler]

There is a concept of relative addresses to be used with the scoped multicast addresses. These relative addresses are listed here:

Relative	Description	Reference
0	SAP Session Announcement Protocol	[Handley]
1	MADCAP Protocol	[RFC2730]
2	SLPv2 Discovery	[Guttman]
3	MZAP	[Thaler]
4	Multicast Discovery of DNS Services	[Manning]
5	SSDP	[Goland]
6	DHCP v4	[Hall]
7	AAP	[Hanna]
8-252	Reserved - To be assigned by the IANA	
253	Reserved	
254-255	Reserved - To be assigned by the IANA	

These addresses are listed in the Domain Name Service under MCAST.NET and 224.IN-ADDR.ARPA.

Note that when used on an Ethernet or IEEE 802 network, the 23 low-order bits of the IP Multicast address are placed in the low-order 23 bits of the Ethernet or IEEE 802 net multicast address 1.0.94.0.0.0. See the section on "IANA ETHERNET ADDRESS BLOCK."

References

[RFC1045] Cheriton, D., "VMTP: Versatile Message Transaction Protocol Specification," RFC 1045, Stanford University, February 1988.

[RFC1075] Waitzman, D., C. Partridge, S. Deering, "Distance Vector Multicast Routing Protocol," RFC-1075, BBN STC, Stanford University, November 1988.

[RFC1112] Deering, S., "Host Extensions for IP Multicasting," STD 5, RFC 1112, Stanford University, August 1989.

[RFC1119] Mills, D., "Network Time Protocol (Version 1), Specification and Implementation," STD 12, RFC 1119, University of Delaware, July 1988.

[RFC1190] Topolcic, C., Editor, "Experimental Internet Stream Protocol, Version 2 (ST-II)," RFC 1190, CIP Working Group, October 1990.

[RFC2328] Moy, J., "OSPF Version 2," STD 54, RFC 2328, Ascend Communications, April 1998.

[RFC1723] Malkin, G., "RIP Version 2: Carying Additional Information," RFC 1723, Xylogics, November 1994.

[RFC1884] Hinden, R. S.Deering, "IP Version 6 Addressing Architecture," RFC 1884, Ipsilon Networks, Xerox PARC, December 1995.

[RFC2114] Chiang, S.J. Lee, H.Yasuda, "Data Link Switching Client Access Protocol," RFC 2114, Cisco, Mitsubishi, February 1997.

[RFC2365] Meyer, D., "Administratively Scoped IP Multicast," RFC 2365, University of Oregon, July 1998.

[RFC2730] Hanna, S. B.Patel, M.Shah, "Multicast Address Dynamic Client Allocation Protocol (MADCAP)," December 1999.

People

[Adams] Chris Adams, <jc.adams@reuters.com>, July 2000.

[Albin] Jerome Albin, <albin@taec.enet.dec.com>, June 1997.

[Albright] Shivaun Albright, <shivaun_albright@hp.com>, July 1997.

[Armitage] Ian Armitage, <ian@coactive.com>, August 1999.

[Aronson] Peter Aronson, <paronson@esri.com>, August 1998.

<arnoldm@microsoft.com>

[AXC] Andrew Cherenson, <arc@SGI.COM>

[Baker] Fred Baker, <fred@cisco.com>, June 1997.

[Ballardie] Tony Ballardie, <A.Ballardie@cs.ucl.ac.uk>, February 1997.

[Ballister] Tom Ballister, <tballister@starguidedigital.com>, July 1997.

[Banthia] Prakash Banthia, <prakash_banthia@3com.com>, September 1998.

[Barber] Tony Barber, <tonyb@pipex.com>, January 1997.

[Bhatti] Zia Bhatti, <zia@netright.com>, September 1998.

[Blackketter] Dean Blackketter, <dean@corp.webtv.net>, November 1998.

[Blank] Tom Blank, <tomblank@microsoft.com>, November 1996.

[Braden] Bob Braden, <braden@isi.edu>, April 1996.

[Bob Brenner]

[Brockbank] Darcy Brockbank, <darcy@hasc.com>, December 1997.

<bloomer@birch.crd.ge.com>

[BXE1] Brendan Eic, <brendan@illyria.wpd.sgi.com>

[BXF] Bruce Factor, <ahi!bigapple!bruce@uunet.UU.NET>

[BXS2] Bill Schilit,<schilit@parc.xerox.com>

[Carr] Wayne Carr, <Wayne_Carr@ccm.intel.com>, December 1997.

[Casner] Steve Casner, <casner@isi.edu>, January 1995.

[Caughie] Colin Caughie, <cfc@indigo-avs.com>, May 2000.

[Caves] Evan Caves, <evan@acc.com>, June 1998.

[Cheshire] Stuart Cheshire, <cheshire@apple.com>, April 2000.

[Chiang] Steve Chiang, <schiang@cisco.com>, January 1997

[Cho] Joan Cho/SIAC, <jcho@siac.com>, October 1998.

[Cipiere] Patrick Cipiere, <Patrick.Cipiere@sophia.inria.fr>, February 1999.

[Tim Clark]

[Costales] Bryan Costales, <bcx@infobeat.com>, September 1997.

[Crawford] James Crawford, <jcrawford@metropol.net>, May 1998.

[Crowcroft] Jon Crowcroft, <jon@hocus.cs.ucl.ac.uk>, November 1998.

[CXM3] Chuck McManis, <cmcmanis@sun.com>

[Daga] Anthony Daga, <anthony@mksrc.com>, June 1999.

[Deering] Steve Deering, <deering@cisco.com>, October 1999.

[DeMoney] Michael DeMoney, <demoney@eng.sun.com>, April 1997.

[Dillonn] Doug Dillon, <dillon@hns.com>, August 1998.

[Dissett] Daniel Dissett, <ddissett@peerlogic.com>, December 1998.

[DLM1] David Mills, <Mills@HUEY.UDEL.EDU>

[DRC3] Dave Cheriton, <cheriton@DSG.STANFORD.EDU>

[Dunne] Stephen Dunne, <sdun@isma.co.uk>, January 1997.

[DXS3] Daniel Steinber, <Daniel.Steinberg@Eng.Sun.COM>

[Eidson] John Eidson, <eidson@hpl.hp.com>, April 1999.

[Fenner] Bill Fenner, <fenner@parc.xerox.com>, December 1997.

[Farinacci] Dino Farinacci, <dino@cisco.com>, February, March 1996.

[Felix] Ken Felix, <kfelix@pcquote.com>, August 1999.

[Flynn] Edward Flynn, <flynne@nasdaq.com>, September 1998.

[Gabriel] Jon Gabriel, <grabriel@tivoli.com>, December 1998.

[Gaddie] Bob Gaddie, <bobg@dtn.com>, August 1998.

[Gal] Yossi Gal, <yossi@gilat.com>, February 1998.

[Gibson] Terry Gibson, <terry.gibson@sun.com>, August 1997.

[Gilani] Asad Gilani, <agilani@nymex.com>, July 1997.

[Glanzer] Dave Glanzer, <dglanzer@fieldbus.org>, June 2000.

[GSM11] Gary S. Malkin, <GMALKIN@XYLOGICS.COM>

[Goland] Yaron Goland, <yarong@microsoft.com>, August 1999.

[Gordon] Howard Gordon, <hgordon@xingtech.com>

[Green] Cliff Green, <cgreen@xypoint.com>, February 1998.

[Guttman] Erik Guttman, <Erik.Guttman@eng.sun.com>, March 1998.

[Hall] Eric Hall, <ehall@ntrg.com>, August 1999, October 1999.

[Hamilton] Mark Hamilton, <mah@spectralink.com>, November 1998.

[Handley] Mark Handley, <mjh@ISI.EDU>, December 1998.

[Hanna] Stephen Hanna, <steve.hanna@sun.com>, July 2000.

[Hinden] Bob Hinden, <hinden@Ipsilon.com>, November 1997.

[Hodges] Richard Hodges, <rh@source.net>, March 1999.

[Holfelder] Wieland Holdfelder, <whd@pi4.informatik.uni-mannheim.de>, January 1997.

[Honton] Chas Honton, <chas@secant.com>, December 1997.

[IANA] IANA, <iana@iana.org>

[Iyer] Ram Iyer <ram@aaccorp.com>, March 1998.

[Jackson] Dan Jackson, <jdan@us.ibm.com>, September 1997.

[Janssen] Rob Janssen, <rob@pe1chl.ampr.org>, January 1995.

[JBP] Jon Postel, <postel@isi.edu>

[JXM1] Jim Miner, <miner@star.com>

[Kammerlander] Ralph Kammerlander, <ralph.kammerlander@khe.siemens.de>, June 2000.

[Kean] Brian Kean, <bkean@dca.com>, August 1995.

[Kerkan] Brian Kerkan, <brian@satcomsystems.com>, May 2000.

[Kittivatcharapong] Sakon Kittivatcharapong, <sakonk@cscoms.net>, May 2000.

[Koopman] Dirk Koopman, <djk@tobit.co.uk>, July 2000.

[Kreutzjans] Michael Kreutzjans, <mike@pcquote.com>, December 1998.

[KS14] Karen Seo, <kseo@bbn.com>

[Lamberg] Mike Lamberg, <mlamberg@siac.com>, February 1997.

[Lee] Choon Lee, <cwl@nsd.3com.com>, April 1996.

[Lewis] Mark Lewis, <Mark_Lewis@ccm.jf.intel.com>, October 1995.

[Lindeman] Morten Lindeman, <Morten.Lindeman@os.telia.no>, March 1999.

[Lyda] Stephen T. Lyda, <slyda@emsg.com>, February 2000.

[Lynch] Joel Lynch, <joel.lynch@cnn.com>, April 1999.

[Andrew Maffei]

[Malamud] Carl Malamud, <carl@invisible.net>, April 1998.

[Malkin] Gary Scott Malkin, <gmalkin@baynetworks.com>, February 1998.

[Mamakos] Louis Mamakos, <louie@uu.net>, March 1997.

[Manning] Bill Manning, <bmanning@isi.edu>, August 1999.

[Marocho] Jose Luis Marocho, <73374.313@compuserve.com>, July 1996.

[Marquardt] Douglas Marquardt, <dmarquar@woz.org>, February 1997.

[Marsh] Gene Marsh, <MarshM@diebold.com>, November 1998.

[Martinicky] Brian Martinicky, <Brian_Martinicky@automationintelligence.com>, March 2000.

[Meier] Bob Meier, <meierb@norand.com>, December 1997.

[Mendal] Geoff Mendal, <mendal@talarian.com>, January 1999.

[Meyer] David Meyer, <meyer@ns.uoregon.edu>, January 1997.

[Meyer2] David Meyer, <dmm@cisco.com>, June 1999 - MALLOC assignment

for temp use: renew 06/2000

[Moelard] Henri Moelard, <HMOELARD@wcnd.nl.lucent.com>, March 1998.

[Moran] Ed Moran, <admin@cruzjazz.com>, October 1997.

[John Moy] John Moy, <jmoy@casc.com>

[MXF2] Martin Forssen, <maf@dtek.chalmers.se>

[Nelson] Gunnar Nelson, <nelsong@nasd.com>, March 1999.

[Nevell] Julian Nevell, <JNEVELL@vbs.bt.co.uk>, August 1999.

[Novick] Alan Novick, <anovick@tdc.com>, August 1998.

[O'Rourke] Stacey O'Rourke, <stacey@network-alchemy.com>, August 1999.

[Parnes] Peter Parnes, <peppar@marratech.com> February 2000.

[Peng] Wenjie Peng, <wpeng@tts.telerate.com>, January 1997.

[Polishinski] Steve Polishinski, <spolishinski@etcconnect.com>, March 2000.

[Poole] David Poole, <davep@extendsys.com>, April 1997.

[Przygienda] Tony Przygienda, <prz@siara.com>, October 1999.

[Purkiss] Ed Purkiss, <epurkiss@wdmacodi.com>, September 1998.

[Querubin] Antonio Querubin, <tony@lava.net>, August 1999.

[Revzen] Shai Revzen, <shrevz@nmcfast.com>, April 1999.

[Rodbell] Mike Rodbell, <mrodbell@ciena.com>, January 1997.

[Roper] Mike Roper, <mroper@8x8.com>, September 1999.

[Guido van Rossum]

[Rowatt] Shane Rowatt, <shane.rowatt@star.com.au>, March 1997.

[Roy] George Roy, <c/o Bill Owens owens@appliedtheory.com>, October 1997.

[Rubin] David Rubin, <drubin@ncc.net>, August 1996.

[Rueskamp] Bodo Rueskamp, <br@itchigo.com>, March 2000.

[Rupp] Heiko Rupp, <hwr@xlink.net>, January 1997.

[Sautter] Robert Sautter, <rsautter@acdnj.itt.com>, August 1999.

[SC3] Steve Casner, <casner@precept.com>

[Scannell] Piers Scannell, <piers@globecastne.com>, March 2000.

[Scheifler] Bob Scheifler, <Bob.Scheifler@sun.com>, August 1998.

[Schwartz] Beverly Schwartz, <bschwart@BBN.COM>, June 1998.

[Shindoh] Masato Shindoh, <jl11456@yamato.ibm.co.jp>, August 1997.

[Shum] Raymond Shum, <rshum@ms.com>, April 1998.

[Simpson] Bill Simpson, <bill.simpson@um.cc.umich.edu> November 1994.

[Sledge] Bob Sledge, <bob@pqsystems.com>, December 1998.

[Sliwinski] Robert Sliwinski, <sliwinre@mail1st.com>, February 2000.

[Joel Snyder]

[Stanev] Nedelcho Stanev, <nstanev@csoft.bg>, May 1999.

[Stewart] Ian Stewart, <iandbige@yahoo.com>, June 1999.

[Strazzera] Paolo Strazzera, <p.strazzera@telematica.it>, June 1999.

[Sveistrup] Darrell Sveistrup, <darrells@truesolutions.net>, June 1999.

[SXA] Susie Armstrong, <Armstrong.wbst128@XEROX.COM>

[SXD] Steve Deering, <deering@PARC.XEROX.COM>

[Thaler] Dave Thaler, <dthaler@microsoft.com>, March 1999, June 2000.

[Thompson] Nigel Thompson, <nigelt@tribal.com>, January 1999.

[Timm] Mary Timm, <mary@xenoscience.com>, July 1998.

[Toga] Jim Toga, <jtoga@ibeam.jf.intel.com>, May 1996.

[Tse] Geordie Tse, <gtse@siac.com>, April 1996.

[tynan] Dermot Tynan, <dtynan@claddagh.ie>, August 1995.

[Uang] Yea Uang, <uang@force.decnet.lockheed.com> November 1994.

[Uusitalo] Mika Uusitalo, <msu@xingtech.com>, April 1997.

[van der Muelen] Ron van der Muelen, <ronv@lsid.hp.com> February 1997.

[Veizades] John Veizades, <veizades@tgv.com>, May 1995.

[Velten] Mike Velten, <mike_velten@liebert.com>, January 1999.

[Vogels] Werner Vogels, <vogels@rnets.com>, August 1998.

[Volpe] Victor Volpe, <vvolpe@smtp.microcom.com>, October 1997.

[Wacey] Ian Wacey, <iain.wacey@gefalbany.ge.com>, May 2000.

[Wallach] Walter Wallach, <walt@sumatech.com>, July 1999.

[Wang] Michael Wang, <Michael.Wang@usa.xerox.com>, March 1999.

[Wasserroth] Stephan Wasserroth, <wasserroth@fokus.gmd.de>, July 2000.

[Watson] Scott Watson, <scott@disney.com>, August 1997.

[Wei] Liming Wei, <lwei@cisco.com>, October 1998.

[Wessels] Duane Wessels, <wessels@nlanr.net>, February 1997.

[White] Peter White, <peter_white@3com.com>, April 1999.

[Wissenbach] Paul Wissenbach, <paulwi@vnd.tek.com>, June 1997.

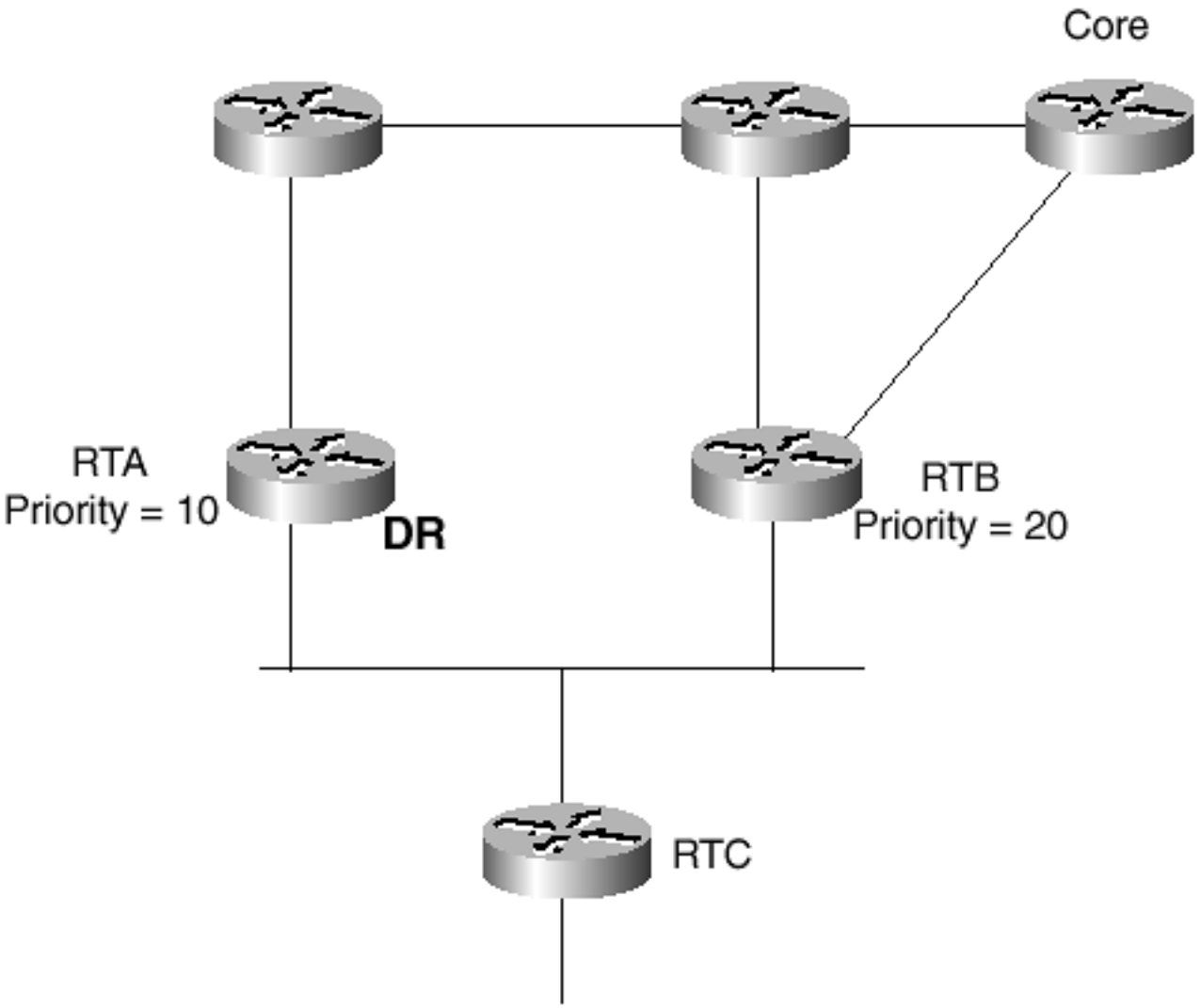
[Wong] Tony Wong, <wongt@ms.com>, July 1998.

[Woodcock] Bill Woodcock, <woody@zocalo.net>, November 1998.

[Wu] Ishan Wu, <iwu@cisco.com>, March 2000.

[Zulch] Richard Zulch, <richard_zulch@dantz.com>, February 1996.

[Zwemmer] Arnoud Zwemmer, <arnoud@nwn.nl>, November 1996.



Each CBT interface is configured with a preference value between 0 and 255, and this value is carried in the HELLO message. A value between 1 and 254 indicates that the router is eligible to become the DR, with the lower number indicating a higher preference—that is, a router with a preference of 10 is "more eligible" than a router with a preference of 20. A preference of 0 indicates that the router is the DR.

When a CBT router first becomes active on a multiaccess link, it sends two HELLO messages in succession to advertise its presence and its preference value. The router then listens for HELLOs, with one of the following three results:

- A HELLO with a lower preference value is heard from another router on the network.
- All HELLOs heard on the network have a higher preference value.
- No other HELLOs are heard on the network.

In the first case, the new router knows that the router with the lower preference value is elected as the DR. In the other two cases, the new router assumes the role of DR and advertises that fact by setting the preference to 0 in its HELLOs. If all HELLOs have equal preference values, the router with the lowest IP address is elected as the DR.

In steady state, the DR sends a HELLO every 60 seconds both as an advertisement of its status and as a keepalive. The DR also sends a HELLO in response to a HELLO from a new router. Other routers do not send HELLOs or respond to HELLOs from new routers.

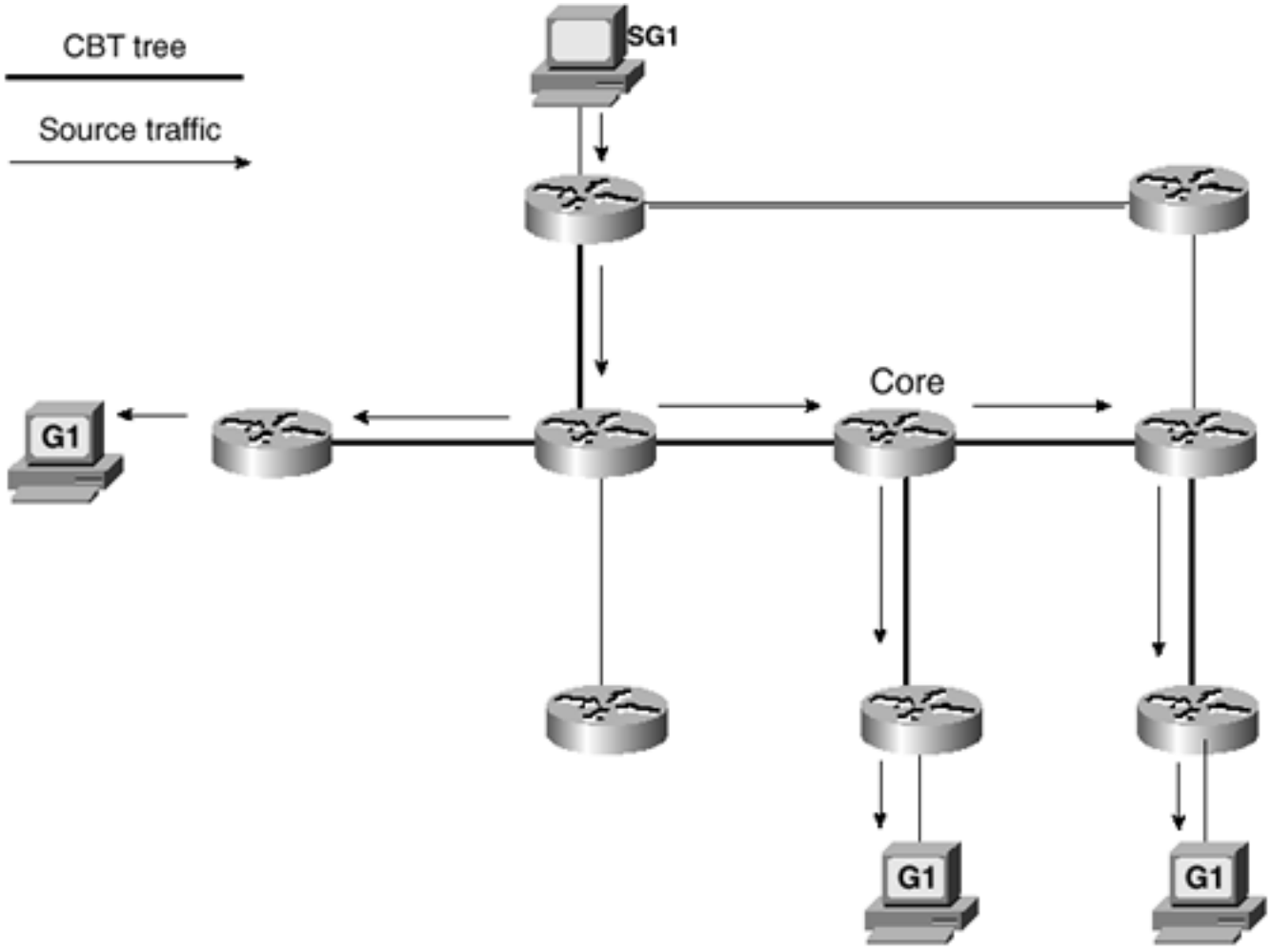
In some cases, the elected DR may not be on the path to the core. Suppose that RTA in Figure 5-39

is elected as the DR, but RTB is the best next-hop router to the core. In this case, when RTC forwards a JOIN_REQUEST to RTA, RTA unicasts the JOIN_REQUEST back across the multiaccess link to RTB. This redirection occurs only with JOIN_REQUESTs; when RTB sends a JOIN_ACK, the message is sent directly to RTC.

Member and Nonmember Sources

You might have noticed that so far nothing has been said about how sources deliver their traffic to the core. In many multicast applications, a sender also is a group member. CBT takes advantage of this fact, so a sender that is also a group member—a *member source*—can reach the core by virtue of the fact that its directly connected router is on-tree. Figure 5-40 illustrates this concept. Here, the host labeled SG1 is a member source of group 1. Because the host is a group member, its local router has already joined the CBT tree for group 1. Therefore, when SG1 sources packets for group 1, the local router can forward the packets up the tree.

Figure 5-40. SG1 Is a Member Source for Group 1. Its Local Router Has Joined the Group 1 Tree and Forwards Packets up the Tree Toward the Source

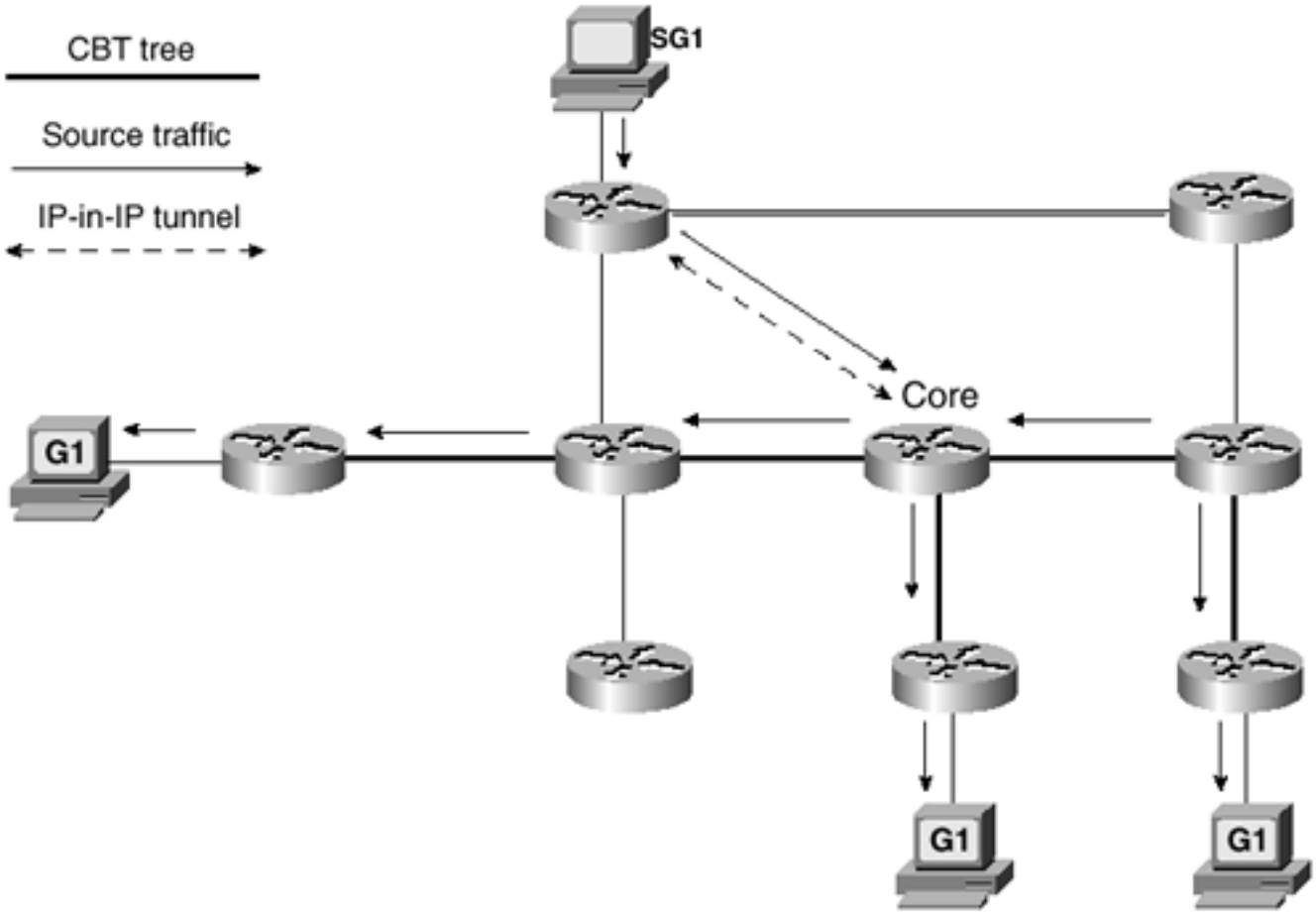


A fundamental characteristic of CBT is described in this behavior. Namely, CBT uses *bidirectional trees*. In other words, multicast traffic can not only travel downstream on the tree from the core to group members, but it also can travel upstream on the tree from a member source to the core. This is in contrast to the other shared-tree protocol, PIM-SM, which uses unidirectional trees.

Of course, not all sources are group members. Therefore, CBT also must have a mechanism for accommodating these *nonmember sources*. The mechanism is a simple IP-in-IP tunnel, as shown in

Figure 5-41. Here, the same host is originating multicast traffic for group 1, but the host itself is not a member of the group. When its local router receives the traffic, it creates a tunnel to the core (assuming the router is running CBT and therefore knows the address of the core). The multicast traffic is then unicast to the core, which passes the traffic onto the group tree.

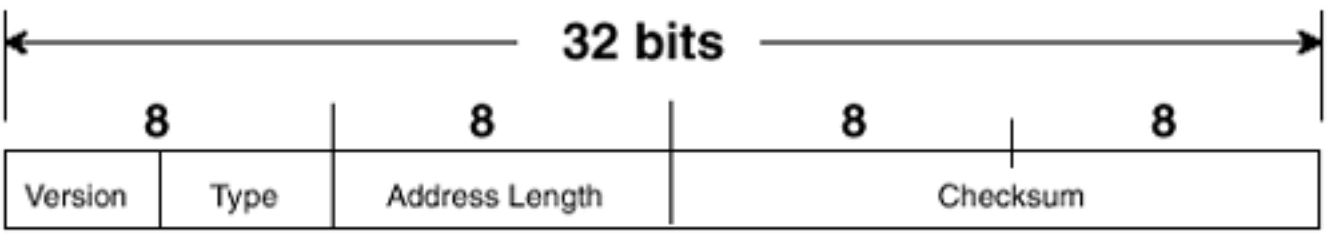
Figure 5-41. If the Source Host Is Not a Group Member, Its Local CBT Router Encapsulates the Source Traffic in an IP-in-IP Tunnel and Unicasts the Traffic to the Core



CBT Message Formats

CBT messages are encapsulated in IP headers with a protocol number of 7. With the unicast exceptions documented earlier in this section, the packets are transmitted with a destination address of 224.0.0.15 and a TTL of 1. Figure 5-42 shows the format of the common header shared by all CBT messages.

Figure 5-42. The CBT Message Header Format



The fields for the CBT message header are defined as follows: