作译者简介

James F. Kurose是美国马萨诸塞大学阿默斯特分校计算机科学系教授。

Kurose博士的教育工作已经得到了广泛认可，其中包括国家理工大学（8次）、马萨诸塞大学和研究生院东北联合会授予的杰出教师奖。他获得了IEEE Taylor Booth教育奖章，确立了在马萨诸塞共同体信息技术促进会的领导地位。他还获得了通用电气公司研究基金、IBM教职员发展奖和Lilly研究基金。

Kurose博士是《IEEE通信学报》和《IEEE/ACM网络学报》的前任主任编辑。多年来，他一直在IEEE Infocom、ACM SIGCOMM、ACM Internet Measurement Conference和ACM SIGMETRICS程序委员会中工作，并担任这些会议的技术程序联合主席。他是IEEE和ACM的会员。他的研究兴趣包括网络协议和体系结构、网络测量、传感器网络、多媒体通信以及建模和性能评价。他拥有哥伦比亚大学计算机科学的博士学位。

Keith W. Ross是美国纽约理工大学（布鲁克林校区）计算机科学系的Leornard J. Shustek教授。1985～1998年，他是宾夕法尼亚大学系统工程系的教授。1998～2003年，他是法国Eurecom学院多媒体通信系的教授。Keith Ross也是Wimba的主要创立者和首任CEO，该公司为电子学习市场研发了IP话音和流技术。

Ross教授的研究兴趣包括对等网络、因特网测量、视频流、Web缓存、内容分发网络、网络安全、IP话音和随机建模。他是IEEE的会员，目前是《IEEE/ACM网络学报》的副编辑。他是联邦贸易委员会P2P文件共享的顾问。他一直在IEEE Infocom、ACM SIGCOMM、ACM Multimedia、ACM Internet Measurement Conference和ACM SIGMETRICS程序委员会中工作。他拥有密歇根大学计算机、信息和控制工程的博士学位。

陈鸣，江苏无锡人，分别于1982年、1988年和1991年在解放军信息工程大学和理工大学获得学士、硕士和博士学位，现任解放军理工大学首席教授、博士生导师；1999～2000年为美国哥伦比亚大学访问科学家，任中国计算机学会和中国通信学会等多个学术团体的委员和IEEE会员；长期从事计算机网络、网络工程设计、分布式系统、网络管理等课程本科生、硕士生和博士生的教学工作，研究方向包括网络测量、网络管理和网络体系结构等；承担了国家九五重点科技攻关项目、国家自然科学基金项目、国家高技术研究发展计划（863）项目和多项军队、省部级研究及工程任务，撰写多本网络著作，发表多篇科技论文，拥有多项国家发明专利。

译 者 序

Kurose和Ross两位教授的这本书是目前国际上最为流行的计算机网络教科书之一。本书第4版最显著的特点是：

1）新颖的“自顶向下”教学法。由于计算机网络的复杂性，长期以来按分层体系结构自底向上讲授网络课程内容是一种定式。本书特别强调应用层，期望尽早激发学生们的学习热情，并尽早强调自主开发网络应用程序。

2）着眼原理。计算机网络领域的许多基础性的重要问题已经研究得较为清楚，重点研究这些原则，将使学生获得长“保质期”的知识，并且在飞速发展的网络研究开发中保持判断力和启发创造力。

3）以因特网为研究对象。本书以因特网体系结构的5层模型来组织材料，为学生们的学习热情提供了原动力。

4）及时更新教学内容。从2001年的第1版到2003年的第2版、2004年的第3版，再到现在的第4版，本书及时引人重要的最新知识并抛弃过时的内容。例如，本版各章全面关注了网络安全问题，更新并扩展了无线网络的覆盖范围，增强了P2P应用程序的内容，更新了局域网和多媒体网络的章节，增加了有关端到端吞吐量分析的新材料，全面修订并增加了新的课后习题等。

5）注重教学法。为帮助学生们理解关键的技术概念，本书包括了许多类比、幽默和实例，引人入胜的历史事件和实践中的原则，对网络领域著名专家的专访，循序渐进的Ethereal实验，以及网站上翔实的教学资料和实验内容等。

本书已经被世界上数以百计的学院和大学采用，被数以万计的学生和专业人士使用；在国内已经被解放军理工大学和许多其他著名高校的计算机、通信等专业采用为本科生、研究生课程教材多年。本书的中译本为缓解在有限时间内有效地学习计算机网络知识（而不是英文）的矛盾起到重要作用。我们对使用本书进行教学的一个建议是：前5章内容可作为本科“计算机网络原理”课程的教材，而后4章内容可作为硕士研究生“高级计算机网络”课程的部分内容。

在本书的翻译过程中，译者收到并参考了多名专家教授为本书翻译提出的很好建议；第4版原文中的多处错漏得到了原书作者的确认；谢希仁教授、李兵副教授和贾永兴博士为第4版的翻译提供了帮助。博士研究生许博、魏祥林制作的网站链接了与本书相关的教学资源，该网站的URL是：http://www.plaust.edu.cn/networks。希望使用这本教材的学人们以此为平台，交流网络教学经验，丰富教学辅导和实验材料，共同提高我国计算机网络课程的教学水平。如果您有教学资源可供共享的话，可与我们联系；一旦选用，我们会将这些材料连同其作者（或提供者）的姓名/单位放在该网站上。

限于时间和学识，译文错漏难免；如有识者，望不吝赐教。译者的联系方式是：cm@plaust.edu.cn 或mingchen@public1.ptt.js.cn。

前　言

欢迎阅读本书的第4版。自本书第1版于7年前问世以来，本书已经被数以百计的大学和学院采用，被译为10多种语言并被世界上数以万计的学生和专业人士使用。我们倾听了许多读者的意见，并感谢众多的赞扬。

第4版的新颖之处

我们认为本书成功的一个重要原因是，本书为计算机网络教学提供了一种崭新的方法。第4版中做了一些改变，但是保持了本书中我们认为（并且得到了使用本书的教师和学生的认可）最为重要的方面——自顶向下方法，关注因特网和计算机网络的现代分析方法，注重原则和实践，以及易于理解的风格和学习计算机网络的方法。

第4版进行了许多重要的改变：由于网络安全的极端重要性，我们增加了对网络安全的关注，从第1章用一个新节介绍了网络安全性问题开始，后面各章中都增加了全新的与安全性相关的材料，并且更新和扩展了第8章中网络安全的覆盖范围（自第1版以来，第8章一直是有关网络安全的主题）；更新并扩展了无线网络的覆盖范围，增加了有关802.11（WiFi）、802.16（WiMAX）和蜂窝网络的新内容；我们涉及的P2P应用程序（这是应用程序协议的一个日趋重要的家族）不仅包括文件共享协议，而且包括BitTorrent等文件分发协议以及使用Skype的IP话音等新型对等多媒体应用；有关局域网和多媒体网络的章节也已经被改进与更新，以反映这些领域中理论与实践的变化；我们还改进了第1章，增加了有关端到端吞吐量分析的新材料；在整本书中，增加了新的课后习题，以及循序渐进的Ethereal实验。

读者对象

本书适合作为计算机网络课程的入门教材，既适用于计算机科学系的学生，也适用于电子工程系的学生。至于编程语言，使用本书的学生需要有C、C++或Java的编程经验（尽管如此，也仅在一些地方需要）。与许多其他入门性的计算机网络教材相比，本书表述更为精确，分析更为细致，而且本书中很少用到学生在高中阶段没有学过的数学概念。我们有意避免使用任何高等微积分、概率论或随机过程的概念（但为具有这种背景的学生准备了某些课后习题）。因此，本书适合作为本科生课程和一年级研究生课程的教科书，对于电信业的从业人员也是有用的。

本书的特色

计算机网络这门学科的内容极为复杂，涉及以错综复杂的方式彼此交织的许多概念、协议和技术。为了处理这种大的跨度和复杂性，许多计算机网络教科书都围绕计算机网络体系结构的“层次”来组织内容。借助于这种分层的组织结构，学生们能够透过计算机网络的复杂性看到其内部，他们在学习整个体系结构的某个部分中的独特概念和协议的同时，也看清了所有这些部分是如何整合在一起的全貌。从教学法的角度来看，我们的经验是这种分层的教学方法的确是非常必要的。然而，我们也发现那种自底向上的传统教学方法，即从物理层

到应用层逐层进行讲解的方法，对于现代的计算机网络课程并非是最佳方法。

自顶向下方法

本书于7年前首次以自顶向下的方式来处理网络，这就是说从应用层开始向下一直讲到物理层。这种自顶向下方法有几个重要的好处。首先，它特别强调应用层（它是计算机网络中的“高增长领域”）。实际上，计算机网络领域中的许多革命性创新都发生在应用层，其中包括Web、对等文件共享和媒体流。及早强调应用层的方法不同于多数其他教科书中所采取的方法，那些教科书只有少量有关网络应用及其需求、应用层范式（例如，客户机/服务器和对等方到对等方）以及应用编程接口方面的内容。

其次，我们（和使用本书的许多教师）作为教师的经验是，在课程开始后就教授网络应用的内容，会有力地调动学生的学习积极性。学生们渴望了解电子邮件和Web等网络应用是如何工作的，这些是多数学生每天都在使用的应用。一旦理解了网络应用，学生们就能够理解支持这些应用的网络服务。学生们接下来会饶有兴致地研究在较低层次中可能提供和实现这些服务的各种方式。因此，及早地学习应用能够激发学生们学习本书其余部分的积极性。

最后，自顶向下方法使得教师能够在教学的早期阶段介绍网络应用程序的开发。学生们不仅能够了解流行的应用程序和协议的工作原理，还能体会到创造自己的网络应用程序和应用级协议有多么容易。采用自顶向下的方法后，学生们能够及早搞清应用编程接口（API）、服务模型和协议的概念，这些概念是后继讨论的各层中重新出现的重要概念。本书通过提供用Java语言编写的套接字编程的例子来强调主要思想，而不会使学生们困惑于复杂代码。电气工程和计算机科学系的本科生理解这些代码应当不会有困难。

以因特网为研究目标

对于第4版，书名中去掉了“描述因特网特色”短语，这是否意味着我们不再关注因特网了呢？确实不是。相反，因特网已经变得无所不在，任何网络教科书都必须极大地关注因特网，因此该短语在某种程度上已经没有必要了。如前3版中那样，本书继续以因特网的体系结构和协议为基本载体来学习基本的计算机网络概念。当然，我们也能将概念和协议放入其他网络体系结构中讲解。因为我们是以因特网为重点来讲解计算机网络，所以围绕因特网体系结构的5层模型来组织材料，这5个层次是应用层、运输层、网络层、链路层和物理层。

强调因特网的另一个好处是，大多数计算机科学和电气工程系的学生急切地希望学习因特网及其协议。他们听说因特网是一种革命性和突破性的技术，正极度改变着我们的世界。有了对因特网广泛的认识后，学生们自然而然会对学习其原理有了求知欲。因此，一旦用因特网作为引导的目标，教师就易于调动学生们学习计算机网络基本原理的积极性。

着眼原理

本书的两个特色是它的自顶向下方法和关注因特网，这可以从本书前3版的副标题看出。如果要在该副标题中加进第3个词的话，这将可能是原理一词。计算机网络领域已经发展得相当成熟，许多基础性的重要问题已经研究得较为清楚。例如，运输层的基础性问题包括：建立在不可靠的网络层上的可靠通信、连接建立/拆除与握手、拥塞和流量控制以及多路复用。网络层的两个基础性问题是：在两台路由器之间找到“好的”路径，处理大量异构网络的互联。数据链路层的基础性问题是：共享多路访问信道。在网络安全中，提供机密性、鉴别和

报文完整性的技术都基于密码学基本理论。本书在指明基础性网络问题的同时，还研究了解决这些问题的方法。掌握了这些原则的学生将获得具有长“保质期”的知识，在今天的网络标准和协议已经过时很长时间后，其中的原则将仍然重要且中肯。我们相信，因特网将学生引入网络之门后，再结合强调基础性问题及其解决方案的方法，会使他们迅速地理解几乎任何网络技术。

Web站点

这本教科书的配套Web站点的URL是：http://www.aw-bc.com/kurose-ross。该站点包括以下内容：

- *交互式学习材料*。该站点包含了交互式Java小程序，以动画方式显示了重要的网络概念。该站点也提供交互测试，这些测试允许学生们检查他们对该专题内容的理解。教师可以将这些交互式学习材料纳入讲稿或用作小实验。
- *附加的技术材料*。因为我们在本书的每个版本中都增加了新材料，所以不得不去除某些现有主题以控制本书的篇幅。例如，为了加入有关交换LAN的新材料，我们去除了有关集线器和网桥的材料；为了加入有关安全的新材料，我们去除了有关安全的旧主题（例如，Kerberos和密钥分发方案）。本书之前版本上的旧材料仍然是有用的，可以在本书的Web网站上找到它们。
- *编程作业*。该Web网站也提供了一些详细的编程作业。编程作业包括了构建一个多线程Web服务器，构建一个具有GUI接口的电子邮件客户机，编制可靠数据传输协议的发送方和接收方的程序，以及编写分布式选路算法等。
- Ethereal实验。通过观察网络协议的动作，可以大大加强对它们的理解。该Web站点提供了一些Ethereal作业，使学生们能够实际观察两个协议实体之间交换的报文顺序。其中包括有关HTTP、DNS、TCP、UDP、IP、ICMP、以太网、ARP、WiFi和SSL的Ethereal实验。

教学特色

我们每个人都教了20多年的计算机网络课程。这本书结合了我们45年来教授数千名学生的教学经验。在这个时期，我们也成为计算机网络领域活跃的研究人员。（事实上，James和Keith于1979年在哥伦比亚大学初次见面时还是硕士研究生，共同选修了由Mischa Schwartz执教的计算机网络课程。）我们认为所有这些都给了我们对于网络现状和网络未来可能发展的良好洞察力。在组织这本书的材料时，我们抵御住了偏向自己钟爱研究项目的诱惑。如果你对我们的研究感兴趣的话，可以访问我们的个人网站。因此，这是一本关于现代计算机网络的书，该书包含了当代协议和技术以及这些协议和技术背后的基本原理。我们也认为学习（和讲授）网络能够乐在其中。本书中包括的幽默、类比方法和实例为这些材料增加了趣味性。

历史事件、实践原则和关注安全

计算机网络领域具有丰富而引人入胜的历史。我们在本书讲解计算机网络历史时做了特别的尝试。我们在第1章中特地安排一节的篇幅介绍了历史，并在其余的各章中分插了10余个历史事件。在这些历史事件的片段中，我们谈及了分组交换技术的发明、因特网的演化、Cisco和3Com等网络巨人的崛起以及许多其他重要事件。学生们将会受到这些历史事件的激励。我们包括一个特殊的补充说明，重点指出计算机网络中的重要原则。这些补充说明将有

助于学生们欣赏应用于计算机网络中的某些基本概念。另外，本版增加的网络安全的某些内容以“关注安全”的形式出现在本书的核心各章中。

人物专访

本书还包括另一个原创性特色，用于启发学生们的灵感，激发学生们的学习热情，这就是专访网络领域声名卓著的创新家们。其中包括对Len Kleinrock、Bram Cohen、Sally Floyd、Vint Cerf、Simon Lam、Charlie Perkins、Henning Schulzrinne、Steven Bellovin和Jeff Case的专访。

各章间的相关性

本书的第1章提供了对计算机网络的概述。该章介绍了许多重要的概念与术语，为本书的其余部分奠定了基础。其他所有的章节都直接依赖于第1章的内容。在讲解完第1章之后，我们推荐按顺序讲解第2章到第5章，这样就在教学中体现了自顶向下的思想。这几章中任何一章都用到前面各章的内容。

在完成前5章的教学工作后，教师就有了相当大的灵活性。最后4章之间没有任何相关性，因此可以以任何顺序进行教学。然而，最后4章中的每一章都依赖于前5章。许多教师教授前5章，然后讲授后4章之一作为点睛之笔。

我们乐于听取您的意见

我们鼓励教师和学生们编写新的Java小程序来阐明本书中的概念和协议。如果你有了你认为适合于本书的Java小程序，请将它发送给作者。如果该Java小程序（包括注释和术语）合适的话，我们将乐于将它放在本书的网站上，并附上该Java小程序作者的适当信息。如前所述，我们也鼓励教师们向我们发送新的课后习题（及其解答），这将完善当前的课后习题。我们将把这些习题放在该Web网站的只有教师才能访问部分。

我们还鼓励学生和教师们向我们发送电子邮件，对本书发表任何评论。对我们而言，能够听到来自世界各地的教师和学生们对本书前3版的反响，的确是件令人愉快的事。请大家毫无保留地告诉我们有趣的相关URL，指出排版错误，不赞成我们的哪些主张，告诉我们怎样做效果好以及怎样做效果不好。告诉我们你认为在下一版中应当补充哪些内容及应当删除哪些内容。我们的电子邮件地址是kurose@cs.umass.edu和ross@poly.edu。

致谢

自1996年我们开始撰写本书以来，许多人就如何组织和讲授网络课程方面给出了极具价值的建议。在此，我们对那些帮助过我们的人，致以由衷的谢意。另外还要感谢成千上万来自世界各地的读者们，包括学生、教职员和从业人员们，他们给了我们对于前面版本的看法和评论，以及对未来版本的建议。除此之外，还要特别感谢下列这些人：

Al Aho（哥伦比亚大学）

Hisham Al-Mubaid（休斯顿净湖大学）

Pratima Akkunoor（亚利桑那州立大学）

Paul Amer（特拉华大学）

Shamiul Azom（亚利桑那州立大学）

Paul Barford（威斯康星大学）
Bobby Bhattacharjee（马里兰大学）
Steven Bellovin（哥伦比亚大学）
Pravin Bhagwat（Wibhu）
Supratik Bhattacharyya（前Sprint公司人员）
Ernst Biersack（Eurécom研究所）
Shahid Bokhari（工程技术大学Lahore分校）
Jean Bolot（Sprint公司）
Daniel Brushteyn（前宾夕法尼亚大学学生）
Ken Calvert（肯塔基大学）
Evandro Cantu（Santa Catarina联合大学）
Jeff Case（SNMP国际研究院）
Jeff Chaltas（Sprint公司）
Vinton Cerf（Google公司）
Byung Kyu Choi（密歇根技术大学）
Bram Cohen（BitTorrent公司）
Constantine Coutras（培斯大学）
John Daigle（密西西比大学）
Edmundo A. de Souza e Silva（Rio de Janiero联合大学）
Philippe Decuetos（Eurécom研究所）
Christophe Diot（汤姆森研究所）
Michalis Faloutsos（加利福尼亚大学Riverside分校）
Wu-chi Feng（俄勒冈研究生院）
Sally Floyd（ICIR，加利福尼亚大学伯克利分校）
Paul Francis（康奈尔大学）
Lixin Gao（马萨诸塞大学）
JJ Garcia-Luna-Aceves（加利福尼亚大学圣克鲁兹分校）
Mario Gerla（加利福尼亚大学洛杉矶分校）
David Goodman（工业大学）
Tim Griffin（剑桥大学）
Max Hailperin（Gustavus Adolphus学院）
Bruce Harvey（佛罗里达A&M大学，佛罗里达州立大学）
Carl Hauser（华盛顿州立大学）
Rachelle Heller（乔治华盛顿大学）
Phillipp Hoschka（INRIA/W3C）
Wen Hsin（Park大学）
Albert Huang（前宾夕法尼亚大学学生）
Esther A. Hughes（弗吉尼亚联邦大学）
Jobin James（加利福尼亚大学Riverside分校）

Sugih Jamin（密歇根大学）
Shivkumar Kalyanaraman（Rensselaer工艺学院）
Jussi Kangasharju（Darmstadt大学）
Sneha Kasera（犹他大学）
Hyojin Kim（前宾夕法尼亚大学学生）
Leonard Kleinrock（加利福尼亚大学洛杉矶分校）
David Kotz（达特茅斯学院）
Beshan Kulapala（亚利桑那州立大学）
Rakesh Kumar（工业大学）
Miguel A. Labrador（南佛罗里达大学）
Steve Lai（俄亥俄州立大学）
Tim-Berners Lee（万维网研究所）
Lee Leitner（Drexel大学）
Brian Levine（马萨诸塞大学）
William Liang（前宾夕法尼亚大学学生）
Willis Marti（得克萨斯A&M大学）
Nick Mckeown（斯坦福大学）
Josh Mckinzie（Park大学）
Deep Medhi（密苏里大学堪萨斯市分校）
Bob Metcalfe（国际数据组）
Sue Moon（KAIST）
Erich Nahum（IBM研究院）
Christos Papadopoulos（科罗拉多州立大学）
Craig Partridge（BBN技术）
Radia Perlman（Sun公司）
Jitendra Padhye（微软研究院）
Kevin Phillips（Sprint公司）
George Polyzos（雅典经济和商业大学）
Sriram Rajagopalan（亚利桑那州立大学）
Ramachadran Ramjee（微软研究院）
Ken Reek（罗切斯特理工学院）
Martin Reisslein（亚利桑那州立大学）
Jennifer Rexford（普林斯顿大学）
Leon Reznik（罗切斯特理工学院）
Sumit Roy（华盛顿大学）
Avi Rubin（约翰斯霍普金斯大学）
Dan Rubenstein（哥伦比亚大学）
Douglas Salane（John Jay学院）
Despina Saparilla（朗讯贝尔实验室）

Henning Schulzrinne（哥伦比亚大学）
Mischa Schwartz（哥伦比亚大学）
Harish Sethu（Drexel大学）
K. Sam Shanmugan（堪萨斯大学）
Prashant Shenoy（马萨诸塞大学）
Clay Shields（乔治顿大学）
Subin Shrestra（宾夕法尼亚大学）
Mihail L. Sichitiu（NC州立大学）
Peter Steenkiste（卡内基－梅隆大学）
Tatsuya Suda（加利福尼亚大学艾尔温分校）
Kin Sun Tam（纽约州立大学Albany分校）
Don Towsley（马萨诸塞大学）
David Turner（加州州立大学圣贝纳迪诺分校）
Nitin Vaidya（伊利诺伊大学）
Michele Weigle（Clemson大学）
David Wetherall（华盛顿大学）
Ira Winston（宾夕法尼亚大学）
Raj Yavatkar（Intel公司）
Yechiam Yemini（哥伦比亚大学）
Ming Yu（纽约州立大学Binghamton分校）
Ellen Zegura（佐治亚理工学院）
Hui Zhang（卡内基－梅隆大学）
Lixia Zhang（加利福尼亚大学洛杉矶分校）
Shuchun Zhang（前宾夕法尼亚大学学生）
Xiaodong Zhang（俄亥俄州立大学）
ZhiLi Zhang（明尼苏达大学）
Phil Zimmermann（独立顾问）
Cliff C. Zou（中佛罗里达大学）

我们也要感谢Addison-Wesley出版公司的整个团队，他们非常出色，还容忍了两位要求苛刻的作者！他们是：Michael Hirsch、Marilyn Lloyd和Lindsey Triebel。感谢Janet Theurer和Patrice Rossi Calkin两位艺术家为第2版、第3版和第4版设计的优美插图，感谢Nancy Kotary、Alicia Williams和Scott Harris对本版的出色运作。最后，特别感谢Addison-Wesley出版公司本版的编辑Michael Hirsch和前面版本的编辑Susan Hartman。没有他们有效的管理、积极的鼓励，近乎无限的耐心、幽默和坚定不移，本书不会如此出色。

第1章　计算机网络和因特网

从蜂窝电话中的Web浏览器到具有公共无线接入功能的咖啡店，从具有高速宽带接入的家庭网络到每张办公桌上都有联网PC的传统办公场所，到联网的汽车，到联网的环境传感器，到星际因特网……计算机网络基本上无所不在。令人兴奋的新应用不断研发出来，扩展了今天乃至未来网络的疆界。本书将介绍计算机网络这个动态领域的最新知识，使读者深入地理解网络的原则和实践，使他们不仅能理解今天的网络，而且能理解明天的网络。

本章概述计算机网络和因特网，目的是从整体上勾画出计算机网络的概貌。其中包括大量的背景知识，讨论大量的计算机网络构件，而且将它们放在整个网络的大环境中进行讨论。本章奠定了本书其他部分的基础。

本章在介绍了某些基本术语和概念后，将首先看看构成网络的基本硬件和软件组件。我们从网络的边缘开始，考察在网络中运行的端系统和网络应用；接下来研究计算机网络的核心，探讨传输数据的链路和交换机，以及连接端系统与网络核心的接入网和物理媒体。我们将了解因特网是“网络的网络”，以及这些网络是怎样彼此连接起来的。

在浏览完计算机网络的边缘和核心之后，在本章的后半部分我们将从更广泛、更抽象的角度来考察计算机网络。我们将介绍计算机网络中的时延、丢包和吞吐量，给出一个简单的端到端吞吐量和时延的定量模型，并且给出考虑了传输、传播和排队时延的模型。接下来，我们将介绍计算机联网时的一些关键的体系结构上的原则，如协议分层和服务模型。最后，我们将以计算机网络的简要历史结束本章的学习。

1.1　什么是因特网

在本书中，我们使用一种特定的计算机网络（即公共因特网）作为讨论计算机网络及其协议的主要载体。但什么是因特网？我们希望能用一句话给出因特网的定义，给出一个值得带回家与家人和朋友共享的定义。然而，因特网非常复杂，并且在不断变化，无论是对硬件和软件组件，还是对它所提供的服务而言，情况都是如此。

我们试着用一种更具描述性的方法来描绘因特网，而不是对它给出一个一句话的定义。描述的方法有两种：一种方法是描述因特网的具体构成，即构成因特网的基本硬件和软件组件；另一种方法是根据为分布式应用提供服务的网络基础设施来描述因特网。我们首先描述因特网的具体构成，并用图1-1举例说明我们的讨论。

1.1.1　具体构成描述

公共因特网是一个世界范围的计算机网络，即它是一个互联了遍及全世界的数以百万计的计算设备的网络。这些计算设备多数是传统的桌面PC、基于Linux的工作站以及所谓的服务器（它们用于存储和传输Web页面和电子邮件报文等信息）。然而，越来越多的非传统的因特网端系统，如个人数字助手（PDA）、TV、移动计算机、蜂窝电话、Web相机、汽车、环境传感设备、数字像框、家用电器和安全系统，正在与因特网相连。的确，在许多非传统设备被联向因特网的情况下，计算机网络这个术语开始听起来有些过时了。用因特网术语来说，所

有这些设备都称为**主机**（host）或**端系统**（end system）。到2006年7月为止，有4亿台端系统使用因特网，并且该数目持续迅速增长着[ISC 2007]。

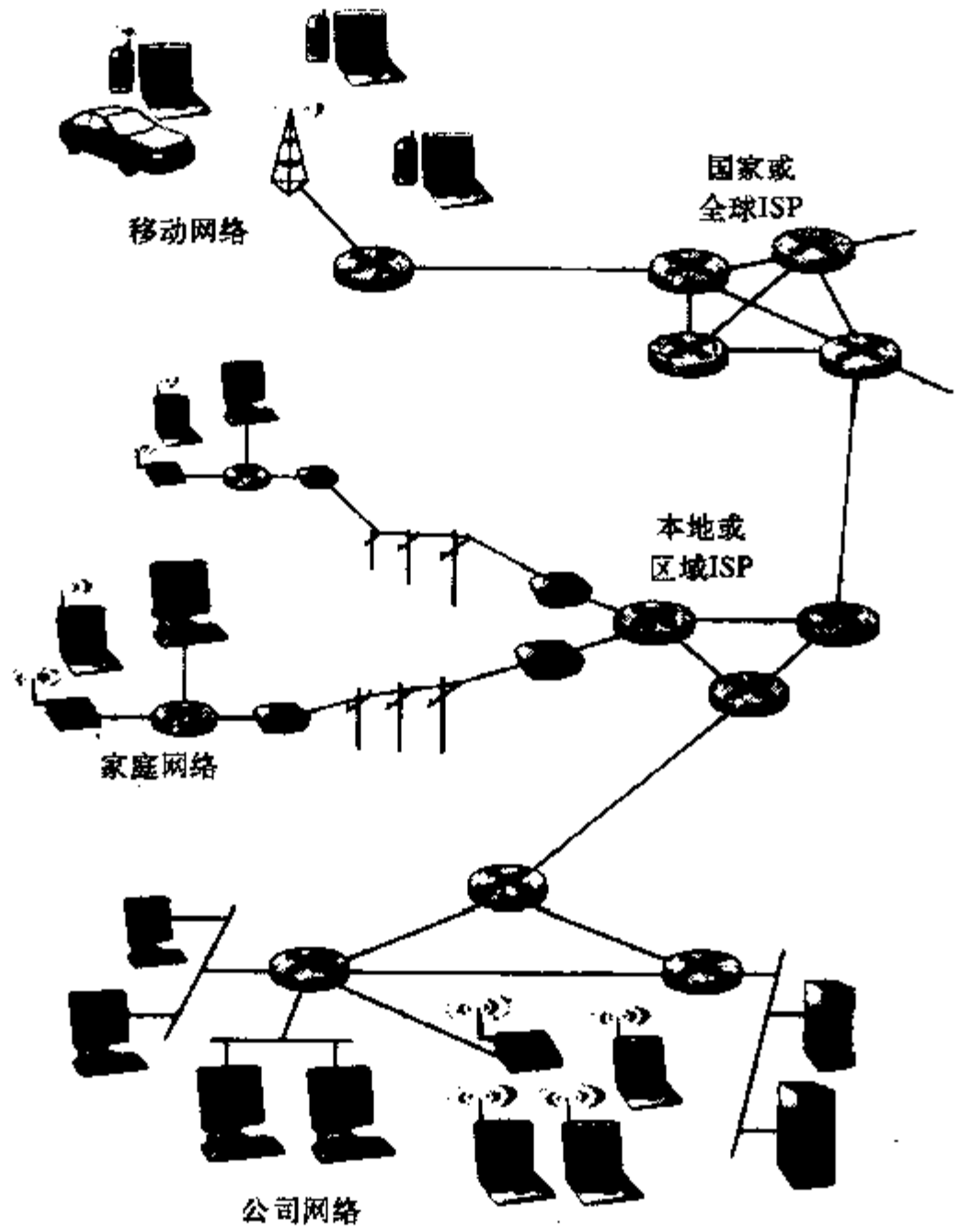


图1-1 因特网的一些部件

端系统通过**通信链路**（communication link）和**分组交换机**（packet switch）连接到一起。在1.2节中，我们将介绍许多类型的通信链路，它们由不同类型的物理媒体组成，这些物理媒体包括同轴电缆、铜线、光纤和无线电频谱。不同的链路以不同的速率传输数据，链路的传输速率是以bps度量的。当一台端系统有数据要向另一台端系统发送时，发送端系统将数据分段，并为每段加上首部字节。用计算机网络的术语来说，由此形成的信息包称为**分组**（packet）。这些分组通过网络发送到目的端系统，在那里被装配成初始数据。

分组交换机从它的一条入通信链路接收到达的分组，并从它的一条出通信链路转发该分组。市面上流行着各种不同类型且各具特色的分组交换机，但在当今的因特网中，两种最著名的类型是**路由器**（router）和**链路层交换机**（link-layer switch）。这两种类型的交换机朝着

最终目的地转发分组。我们将在第4章详细研究路由器，在第5章详细研究链路层交换机。从发送端系统到接收端系统，一个分组所经历的一系列通信链路和分组交换机称为通过该网络的路径（route或path）。第一个分组交换网络产生于20世纪70年代，它是今天因特网的“最早祖先”。今天的因特网中所承载的精确通信量是难以估算的[Odylsko 2003]，而在2005年一份因特网服务提供商（如AOL）的报告说，进入因特网的流量达每秒250吉（10^9）比特的速率[Gill 2005]。PriMetrica [PriMetrica 2007]估计在2006年由公共运营商所使用的国际流量为5太拉（10^{12}）比特，并且使用的带宽大约每两年翻一番。

用于传送分组的分组交换网络在许多方面类似于承载车辆的公路、铁路和立交桥。例如，考虑下列情况，一个工厂需要将大量货物搬运到位于数千公里以外的某个目的地仓库。每辆卡车则独立地通过公路、铁路和立交桥的网络向该仓库运送货物。在目的地的仓库，卸载下这些货物，并且与一起装载的同一批货物的其余部分分成一个组。因此，在许多方面，分组类似于卡车，通信链路类似于公路和铁路，分组交换机类似于立交桥，而端系统类似于建筑物。就像卡车选取运输网络的一条路径前行一样，分组则选取计算机网络的一条路径前行。

端系统通过**因特网服务提供商**（Internet Service Provider, ISP）接入因特网，包括像AOL（美国在线）那样的住宅区ISP、本地电缆或电话公司、公司ISP、大学ISP，以及像T-Mobile那样在机场、旅馆、咖啡店和其他公共场所提供无线接入的ISP。每个ISP是一个由多个分组交换机和多段通信链路组成的网络。不同的ISP为端系统提供了各种不同类型的网络接入，包括56 kbps拨号调制解调器接入、如线缆调制解调器或DSL那样的住宅宽带接入、高速局域网接入和无线接入。ISP也对内容提供者提供因特网接入服务，将Web站点直接接入因特网。为了允许因特网用户之间相互通信，允许用户访问世界范围的因特网内容，这些低层ISP通过国家的、国际的高层ISP（如AT&T和Sprint）互联起来。高层ISP是由通过高速光纤链路互联的高速路由器组成的。无论是高层还是低层ISP网络，它们每个都是独立管理的，运行IP协议（参见下文），遵从一定的命名和地址习惯。我们将在1.3节中更为详细地考察ISP和它们的互联的情况。

端系统、分组交换机和其他因特网部件，都要运行控制因特网中信息接收和发送的一系列**协议**（protocol）。TCP（Transmission Control Protocol，传输控制协议）和IP（Internet Protocol，网际协议）是因特网中两个最为重要的协议。IP协议定义了在路由器和端系统中发送和接收的分组的格式。因特网主要的协议统称为TCP/IP。我们在本章中就开始接触这些协议，但这仅仅是个开始，本书的许多地方与计算机网络协议有关。

指出了因特网协议的重要性后，重要的是每个人都能就各个协议的作用达成共识。这正是标准发挥作用的地方。**因特网标准**（Internet standard）由**因特网工程任务组**（Internet Engineering Task Force，IETF）研发[IETF 2007]。IETF的标准文档被称为**请求评论**（Request For Comment，RFC）。RFC最初是作为普通的请求评论（因此而得名），以解决因特网先驱者们面临的体系结构问题。RFC文档往往是技术性很强并相当详细的。它们定义了诸如TCP、IP、HTTP（用于Web）和SMTP（用于电子邮件的开放标准）这样的协议。目前已经有将近5000个RFC。其他组织也在制定用于网络组件的标准，其中最引人注目的是针对网络链路的标准。例如，IEEE 802 LAN/MAN标准化委员会[IEEE 802 2007]制定了以太网和无线WiFi的标准。

公共因特网（即上面讨论的全球性“网络的网络”）是一个特定的网络，通常被特指为**因特网**。还有许多专用网络，如许多公司和政府的网络，这些网络内的主机不能与专用网络外部的主机交换信息（除非这些信息通过了所谓的防火墙，否则防火墙一般会限制报文进入和流出网络）。这些专用网络常被称为**内联网**（intranet），因为它们与公共因特网采用同样类型

的主机、路由器、链路和协议。

1.1.2 服务描述

前面的讨论已经标识了构成因特网的许多部件。我们还能从一个完全不同的角度，即从*为应用程序提供服务的基础设施*的角度来描述因特网。这些应用程序包括电子邮件、Web冲浪、即时讯息、IP上的话音（VoIP）、因特网广播、流式视频、分布式游戏、对等（peer-to-peer，P2P）文件共享、因特网上的电视、远程注册等。这些应用程序称为**分布式应用程序**(distributed application)，因为它们涉及多台相互交换数据的端系统。重要的是，因特网应用程序运行在端系统上，即它们并不运行在网络核心中的分组交换机之中。随着本书的讨论我们将愈发清楚，尽管分组交换机促进了端系统之间的数据交换，但是它们并不关心作为数据的源或宿的应用程序。

我们再深入探讨一下前面所说的为应用程序提供服务的基础设施所表达的意思。假定你最终对分布式因特网应用程序有一个激动人心的新思想，它可能大大地造福于人类，或者它可能直接使你富有和出名。那么，如何将这种思想转换成为一种实际的因特网应用程序呢？因为应用程序运行在端系统上，你需要编写运行在端系统上的一些软件。例如，你可能用Java、C或C++编写你的软件。此时，因为你研发了一种分布式因特网应用程序，运行在不同端系统上的软件将需要互相发送数据。这里我们接触了一个核心问题，这导致将描述因特网的另一种方法用于应用程序的平台。运行在一个端系统上的应用程序怎样才能指示因特网向运行在另一个端系统上的软件发送数据呢？

与因特网相连的端系统提供了一个**应用程序编程接口**(Application Programming Interface，API)，API规定了运行在一个端系统上的软件请求因特网基础设施向运行在另一个端系统上的特定目的地软件交付数据的方式。因特网API是一套发送软件必须遵循的规则集合，因此因特网将向目的地软件交付数据。我们将在第2章详细讨论因特网API。现在，我们给出一个简单的类比，本书中会经常使用这个类比。假定Alice使用邮政服务向Bob发一封信。当然，Alice不能只是写了这封信（相关数据）然后把该信丢出窗外。邮政服务要求Alice将信放入一个信封中；在信封的中央写上Bob的全名、地址和邮政编码；封上信封；在信封的右上角贴上邮票；最后将该信封投进一个邮局的邮政服务邮箱中。因此，该邮政服务有其自己的“邮政服务API”或一套规则，Alice必须遵循该规则，才能通过邮政服务将自己的信件交付给Bob。类似地，因特网具有一个发送软件必须遵循的API，以使因特网向目的地软件交付数据。

当然，邮政服务向它的顾客提供了多种服务。它提供了特快专递、挂号、普通服务等。类似地，因特网向它的应用程序提供了多种服务。当你研发一种因特网应用程序时，你也必须为你的应用程序选择一种因特网服务。我们将在第2章中描述因特网服务，并在第3章中描述因特网是如何提供那些服务的。

因特网的第二种描述方法（根据基础设施向分布式应用程序提供的服务）是很重要的。因特网具体构成部件的发展日益由新应用程序的需求所驱动。因此，请记住：因特网是一种基础设施，新应用程序正在其上不断地被发明和设置。

我们已经给出了因特网的两种描述方法：一种是根据它的硬件和软件组件来描述，另一种是根据基础设施向分布式应用程序提供的服务来描述。但是你也许还是对因特网是什么感到困惑。什么是分组交换？什么是TCP/IP？什么是API？什么是路由器？因特网中正在使用什么样的通信链路？什么是分布式应用程序？一个烤箱或天气传感器怎样与因特网相连？如果现在对这些心存疑惑，请不要担心，这本书除了介绍因特网的架构外，还要介绍因特网的工

作方式和工作原理。我们将在后续章节中解释这些重要的术语和问题。

1.1.3 什么是协议

既然我们已经对因特网是什么有了一点印象，下面考虑计算机网络中另一个重要的时髦用语：协议。什么是协议？协议是干什么的？如果遇到一个协议，如何识别它？

1.人类活动的类比

要理解计算机网络协议的概念，也许最容易的办法是：先与某些人类活动进行类比，因为我们人类无时无刻不在执行协议。例如，当你想要向某人询问一天的时间时，将怎样做？图1-2中显示了一种典型的交互过程。人类协议（或者至少说是好的行为方式）要求一方首先进行问候（图1-2中的第一个“你好”），以开始与另一个人的通信。对这个“你好”的典型响应是返回一个“你好”报文。此人用一个热情的“你好”进行响应，这隐含着能够继续向那人询问时间了。对最初的“你好”的不同响应（例如，“不要烦我！”，或“我不会说英语”，或某些不合时宜的回答）可能表明：能勉强与之通信或不能与之通信。在此情况下，按照人类协议，发话者将不能够询问时间了。有时，一个人询问的问题根本得不到任何回答，在此情况下通常是放弃向该人询问时间的想法。注意，在人类协议中，有我们发送的特定报文，有我们根据接收到的应答报文或其他事件（例如，在某些给定的时间内没有应答）采取的动作。显然，传输的和接收的报文，以及当这些报文被发送和接收或其他事件出现时所采取的动作，在一个人类协议中起到了核心作用。如果人们执行不同的协议（例如，一个人讲礼貌，而另一人不讲礼貌，或一个人明白时间这个概念，而另一人却不知道），该协议就不能互动，因而不能完成有用的工作。在网络中该道理同样成立，为了完成一项工作，要求两个（或多个）通信实体运行相同的协议。

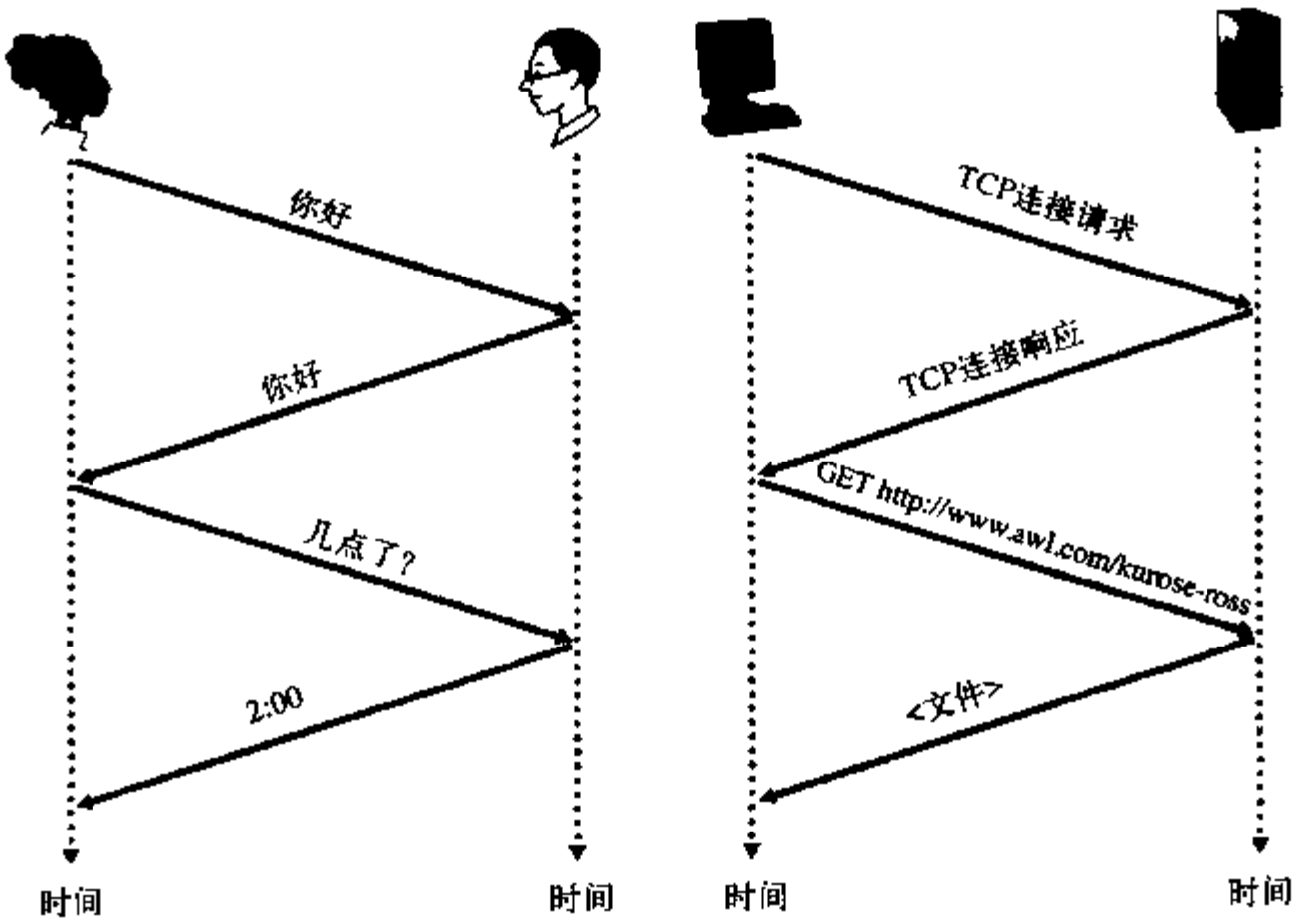


图1-2 人类协议和计算机网络协议

我们再来考虑第二个人类类比的例子。假定你正在一个大学课堂里上课（例如，上的是计算机网络课程），教师正在唠唠叨叨地讲述协议，而你困惑不解。这名教师停下来问：“同

学们有什么问题吗?”(教师发送出一个报文，该报文被所有没有睡觉的学生接收到了。)你举起了手(向教师发送了一个隐含的报文)，这位教师面带微笑地示意你“请讲……”(教师发出的这个报文鼓励你提出问题，教师喜欢被问问题。)接着你就问了问题(即向该教师传输了你的报文)，教师听取了你的问题(即接收了你的问题报文)并加以回答(向你传输了应答报文)。我们再一次看到了报文的发送和接收，以及当这些报文发送和接收时所采取的一系列约定俗成的动作，这些都是这个“提问和回答”协议的核心。

2. 网络协议

网络协议类似于人类协议，只不过交换报文和采取动作的实体是某些设备(如计算机、PDA、蜂窝电话、路由器或其他具有网络能力的设备)的硬件或软件组件。因特网中的所有活动，凡是涉及两个或多个通信的远程实体都受协议的制约。例如，在两台物理连接的计算机的网络接口卡中，硬件实现的协议控制了两块网络接口卡间的“线上”比特流；端系统中的拥塞控制协议控制了发送方和接收方之间传输的分组的速率。因特网中到处运行着协议，因此本书的大量篇幅与计算机网络协议有关。

以大家可能熟悉的一个计算机网络协议为例，比如，当你向一个Web服务器发出请求时，即你在Web浏览器中键入一个Web网页的URL时，将会发生什么情况呢？图1-2右边部分显示了一种情形。首先，计算机将向Web服务器发送一条“连接请求”报文，并等待回答。Web服务器最终将能接收到该连接请求报文，并返回一条“连接响应”报文。得知请求该Web文档正常以后，计算机则在一条“GET”报文中发送要从这台Web服务器上取回的网页的名字。最后，Web服务器向该计算机返回该Web网页(文件)。

从上述的人类活动和网络例子中可见，报文的交换以及当发送和接收这些报文时所采取的动作是一个协议的关键定义元素：

> 一个协议定义了在两个或多个通信实体之间交换的报文格式和次序，以及在报文传输和/或接收或其他事件方面所采取的动作。

一般来说，因特网和计算机网络广泛地使用了协议。不同的协议用于完成不同的通信任务。当你阅读完这本书后将会知道，一些协议简单而直截了当，而另一些协议则复杂且难懂。掌握计算机网络领域知识的过程就是理解网络协议的构成、原理和工作的过程。

1.2 网络边缘

前面给出了因特网和网络协议的总体概述，现在我们将更深入地研究一下计算机网络(特别是因特网)的部件。在本节中，我们从网络边缘开始，观察一下我们非常熟悉的部件，即计算机、PDA、蜂窝电话和其他设备。在接下来的一节中，我们将从网络边缘向网络核心推进，探讨计算机网络中的交换和选路。

令人眼花缭乱的因特网端系统

不久以前，与因特网相连的端系统设备主要还是传统的计算机，如桌面机器和强大的服务器。从20世纪90年代后期开始到今天，各种各样差异很大的有趣设备不断地连入因特网。这些设备都具有向其他设备发送和接收数字数据的共同特性。考虑到因特网具

有如下特性：无所不在、定义良好（标准）的协议和因特网适用的商品硬件，使用因特网技术将这些设备连接在一起是自然而然的事。

这些设备中，有些看起来纯粹是为了娱乐而创造的。支持IP的数字像框[Ceiva 2007]能从远程服务器下载数字相片，并将它们显示在一个设备上——该设备看起来像一个传统的像框；因特网烤箱能从某服务器下载气象信息，并将今日的天气预报（例如，多云天气）[BBC 2001]的图像烙在要烤的面包片上。其他设备还能提供别的有用信息，如Web相机可以显示当前交通流量和气象情况或监视相关的位置；与因特网相连的家用电器（包括洗衣机、电冰箱和炉具）具有Web浏览器界面，以进行远程监视和控制[Internet Home Alliance 2007]。支持IP的蜂窝电话将Web浏览器、电子邮件和发送消息"放在了指尖"。一类新型联网传感器系统使得我们观察和交互环境的方式发生了革命性变化。嵌入物理环境的联网传感器允许对如下东西进行监视：建筑物、桥梁、地震活动、野生动植物习性、河流江口和大气层的较低层[CENS 2007, Culler 2004, CASA 2007]。生物医学传感器可能也是嵌入式的、联网的[Schwiebert 2001]。感应的数据将对远程用户实时可用。将一个RFID标签或一个极小的嵌入式传感器粘贴到任何物体上，能够使得有关或来自该物体的信息在因特网上可用，从而使其成为一个"因特网的东西"[ITU 2005]。

回想前一节计算机网络的术语，与因特网相连的计算机等设备通常称为**端系统**（end system）。如图1-3所示，因为它们位于因特网的边缘，故而被称为端系统。因特网的端系统包括桌面计算机（例如，桌面PC、Mac和基于Linux的工作站）、服务器（例如，Web和电子邮件服务器）和移动计算机（例如，便携式计算机、PDA和采用无线因特网连接的电话）。此外，越来越多的其他类型的设备正被作为端系统与因特网相连（参见补充说明）。

端系统也称为**主机**，因为它们容纳（即运行）诸如Web浏览器程序、Web服务器程序、电子邮件阅读程序或电子邮件服务器程序等应用程序。本书通篇将交替使用主机和端系统这两个术语，即*主机 = 端系统*。主机有时又被进一步划分为两类：**客户机**（client）和**服务器**（server）。客户机非正式地等同于桌面PC、移动PC和PDA等，而服务器非正式地等同于更为强大的机器，用于存储和发布Web页面、流视频以及转发电子邮件等。

1.2.1 客户机和服务器程序

在网络软件的上下文中，客户机和服务器有另一种定义，本书通篇都采用这种定义。**客户机程序**（client program）是运行在一个端系统上的程序，它发出请求，并从运行在另一个端系统上的**服务器程序**（server program）接收服务。这种客户机-服务器模式无疑是因特网应用程序的最为流行的结构，我们将在第2章中详细研究。Web、电子邮件、文件传输、远程注册（例如Telnet）、新闻组和许多其他流行的应用程序采用了客户机-服务器模式。因为通常客户机程序运行在一台计算机上，而服务器程序运行在另一台计算机上，所以根据定义，客户机-服务器因特网应用程序是**分布式应用程序**（distributed application）。客户机程序和服务器程序通过因特网互相发送报文而进行交互。在这个层次的抽象下，路由器、链路和因特网服务的其他具体构件可作为一个"黑盒子"，该黑盒子在因特网应用程序的分布式的、通信的部件之间传输报文。图1-3中显示了这种级别的抽象。

今天的因特网应用程序并非全都是由与纯服务器程序交互的纯客户机程序组成的。越来

越多的应用程序是对等（P2P）应用程序，其中的端系统互相作用并运行执行客户机和服务器功能的程序。例如，在P2P文件共享应用程序（如Limewire、eDonkey和Kazaa）中，用户端系统中的程序起着客户机程序和服务器程序的双重作用。当它向另一个对等方请求文件时，起着客户机的作用；当它向另一个对等方发送文件时，起着服务器的作用。在因特网电话中，通信双方作为对等方交互，即通信会话是对称的，双方都在发送和接收数据。在第2章中，我们将详细比较和对照客户机-服务器和P2P体系结构。

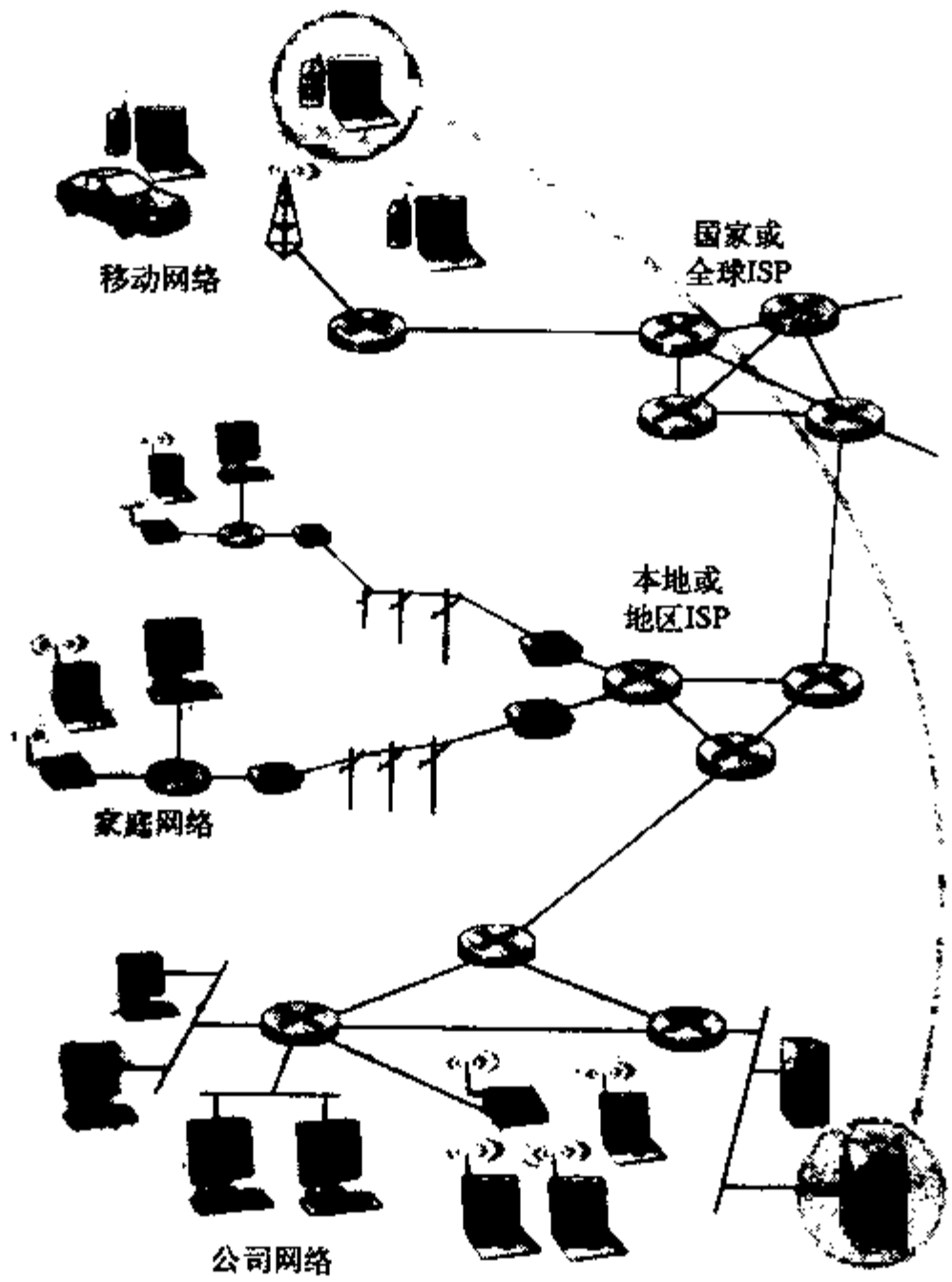


图1-3 端系统交互

1.2.2 接入网

讨论了位于“网络边缘”的应用程序和端系统后，我们接下来讨论**接入网**（access network），即将端系统连接到其**边缘路由器**（edge router）的物理链路。边缘路由器是端系统到任何其他远程端系统的路径上的第一台路由器。图1-4显示了从端系统到边缘路由器的几种类型的接入链路。图中的接入链路是用粗线突出标示的。网络接入大致可以分为以下三种类型：

- 住宅接入（residential access），将家庭端系统与网络相连。
- 公司接入（company access），将商业或教育机构中的端系统与网络相连。
- 无线接入（wireless access），将移动端系统与网络相连。

这些分类并不严格，例如，某些公司端系统可能使用我们认为属于住宅接入的接入技术，

反之亦然。下列描述在通常情况下是成立的。

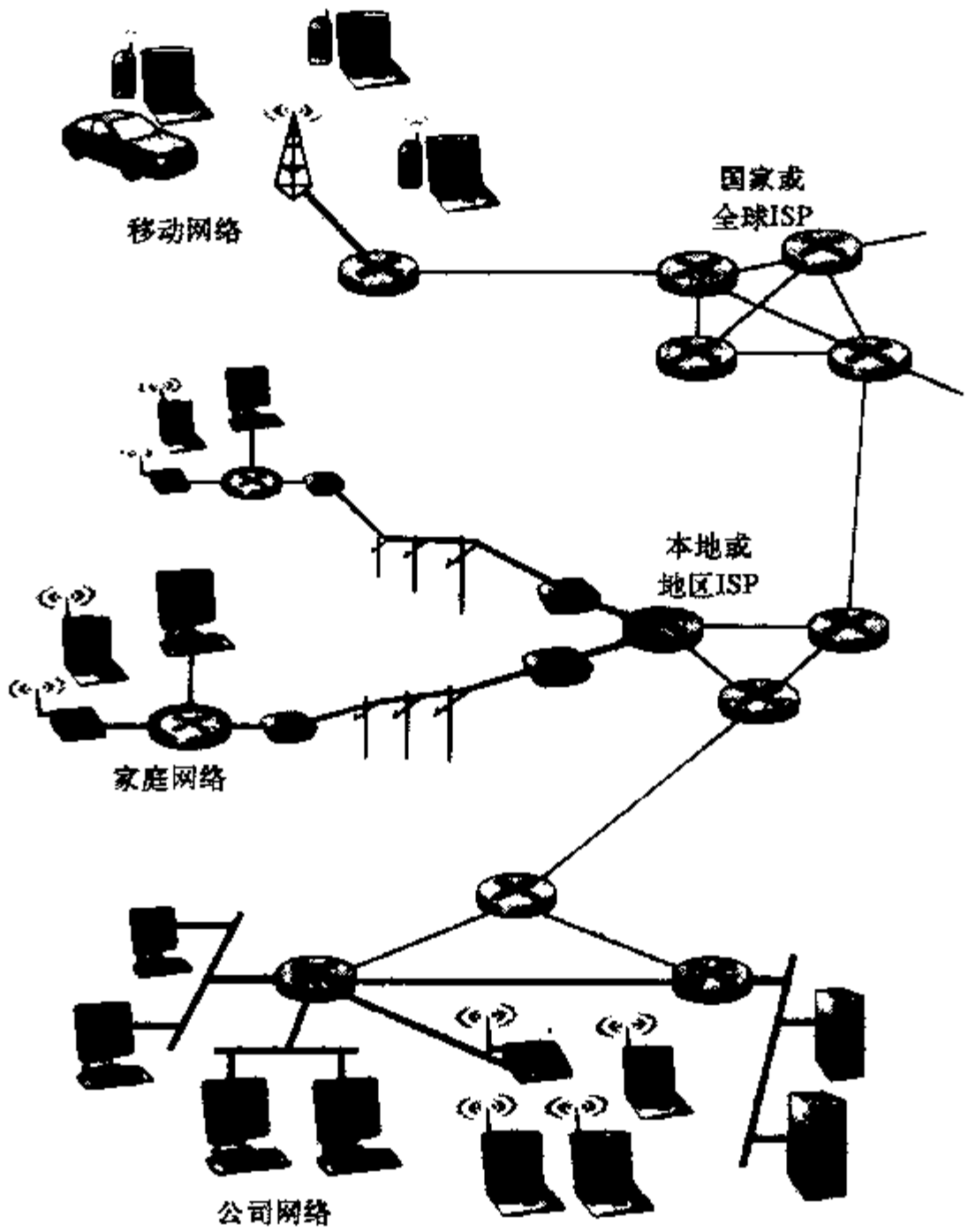


图1-4 接入网

1. 住宅接入

住宅接入是指将家庭端系统（PC或家庭网络，参见下文）与边缘路由器相连接。一种住宅接入形式是通过普通模拟电话线用**拨号调制解调器**（dial-up modem）与住宅ISP（如美国在线）相连。家用调制解调器将PC输出的数字信号转换为模拟形式，以便在模拟电话线上传输。模拟电话线由双绞铜线构成，就是与用于打普通电话相同的电话线。（本节稍后我们将讨论双绞线。）在模拟电话线的另一端，ISP的调制解调器再将模拟信号转换回数字形式，作为ISP路由器的输入。因此，该接入网络仅是连同一条点对点拨号电话线一起的一对调制解调器。今天的调制解调器的速率允许高达56 kbps的拨号接入。然而，由于在许多家庭和ISP之间的双绞线质量较低，所以许多用户获得的有效速率大大低于56 kbps。

许多住宅用户发现拨号调制解调器的56 kbps接入缓慢得令人难以接受。例如，通过56 kbps拨号调制解调器下载一首3分钟的MP3歌曲大约需要8分钟。此外，拨号调制解调器接入占用了用户的普通电话线，当住宅用户使用拨号调制解调器在Web上冲浪时，该用户就不

能用该电话线接收和拨打普通电话了。值得庆幸的是，新型宽带接入技术为住宅用户提供了更高的比特速率，也为用户提供了一种接入因特网的同时还能打电话的手段。宽带住宅区接入有两种常见类型：**数字用户线**（digital subscriber line, DSL）[DSL 2007]和**混合光纤同轴电缆**（hybrid fiber-coaxial cable, HFC）[Cable Labs 2007]。

到2006年3月为止，在许多发达国家中超过50%的家庭拥有了宽带线。美国和中国在宽带线总数方面领先，每个国家都超过4千万条线路[Point Topic 2006]。到2006年3月为止，美国和加拿大大约60%的宽带线是HFC，其余的宽带线则是DSL。除美国与加拿大以外，其他国家是DSL占有主导地位，特别是在欧洲许多国家拥有超过90%的DSL。

DSL接入一般是由电话公司（例如Verizon或法国电信）提供，有时由与独立的ISP合伙的公司提供。DSL概念上类似于拨号调制解调器，它是一种新型调制解调器技术，也运行在现有的双绞线电话线上。通过限制用户和ISP调制解调器之间的距离，DSL能够以高得多的速率传输和接收数据。其数据传输速率通常在两个方向上是不对称的，从ISP路由器到家庭的速率比从家庭到ISP路由器的要更高。数据传输速率的这种不对称性反映了这样一种观念，即家庭用户更可能是信息的消费者（将信息取回家），而不太可能是信息的生产者。

DSL在家庭和ISP之间将通信链路划分为3个不重叠的频段：

- 高速下行信道，位于50 kHz到1 MHz频段。
- 中速上行信道，位于4 kHz到50 kHz频段。
- 普通的双向电话信道，位于0到4 kHz频段。

这种方法使得单根DSL线路看起来就像有3条单独的线路，因此一个电话呼叫和一个因特网连接能够同时共享DSL链路。（我们将在1.3.1节中描述这种频分多路复用技术。）用户可用的实际下载和上载传输速率是家庭调制解调器和ISP调制解调器之间的距离、双绞线规格、电磁干扰强度及其他需考虑的因素的函数。与拨号调制解调器不同的是，工程师明确地将DSL设计为用于住宅和ISP调制解调器之间的短距离的传输，因此允许比拨号接入高得多的传输速率。对于靠近ISP调制解调器的用户，通常以不同的价格使用不同的传输速率。例如，Verizon提供从大约768 kbps到1.5 Mbps之间的下行传输速率。另外，还有各种各样的高速DSL技术目前在数以百计的国家得到了突飞猛进的发展。例如，甚高速DSL（VDSL）目前在韩国和日本得到了最高速度的发展，提供了12～55 Mbps下载速率和1.6～20 Mbps的上行速率[DSL 2007]。

虽然DSL和拨号调制解调器使用普通的电话线，但HFC接入网扩展了当前用于广播电缆电视的电缆网络。在传统的电缆系统中，电缆头端（head end）广播通过同轴电缆和放大器的分配网络传向住宅。如图1-5所示，光缆连接了该电缆头端到相邻域级的连接点，从这里再使用传统的同轴电缆到达各个家庭住宅。每个相邻域连接点通常支持500～5000户家庭。

与DSL类似，HFC需要特殊的调制解调器，称为**电缆调制解调器**（cable modem）。提供电缆因特网接入的公司要求其用户购买或租用一个调制解调器。电缆调制解调器通常是一种外部设备，通过一个以太网端口与家庭PC相连。（我们将在第5章非常详细地讨论以太网。）电缆调制解调器将HFC网络划分为两个信道，即下行信道和上行信道。与DSL类似，下行信道通常分配了更大的带宽，因而有更快的传输速率。

HFC的一个重要特征是它共享广播媒体。特别是由头端发送的每个分组向下行经每段链路到每个家庭，每个家庭发送的每个分组经上行信道向头端传输。由于这个原因，如果几个用户同时经下行信道下载不同的MP3，每个用户接收MP3的实际速率将大大低于下行速率。而另一方面，如果仅有很少的活动用户在进行Web冲浪，则这些用户中的每一个都可以以下

行速率的全部速率接收Web网页，因为用户们很少在完全相同的时刻请求网页。因为上行信道也是共享的，需要一个分布式多路访问协议，以协调传输和避免碰撞。（我们将在第5章讨论以太网时更为详细地讨论碰撞问题。）DSL的拥护者一针见血地指出，DSL在家庭和ISP之间建立了一条点对点连接，因此所有的DSL带宽是专用的而不是共享的。电缆的拥护者却声辨说，覆盖范围合理的HFC网络提供了比DSL更大的带宽。DSL和HFC之间的高速住宅接入之战正在拉开序幕，特别是在北美。在乡村的既没有DSL也没有HFC可用的区域，可以使用卫星链路以超过1 Mbps的速率将住宅与因特网相连；StarBand和HughesNet是两个这样的卫星接入提供商。

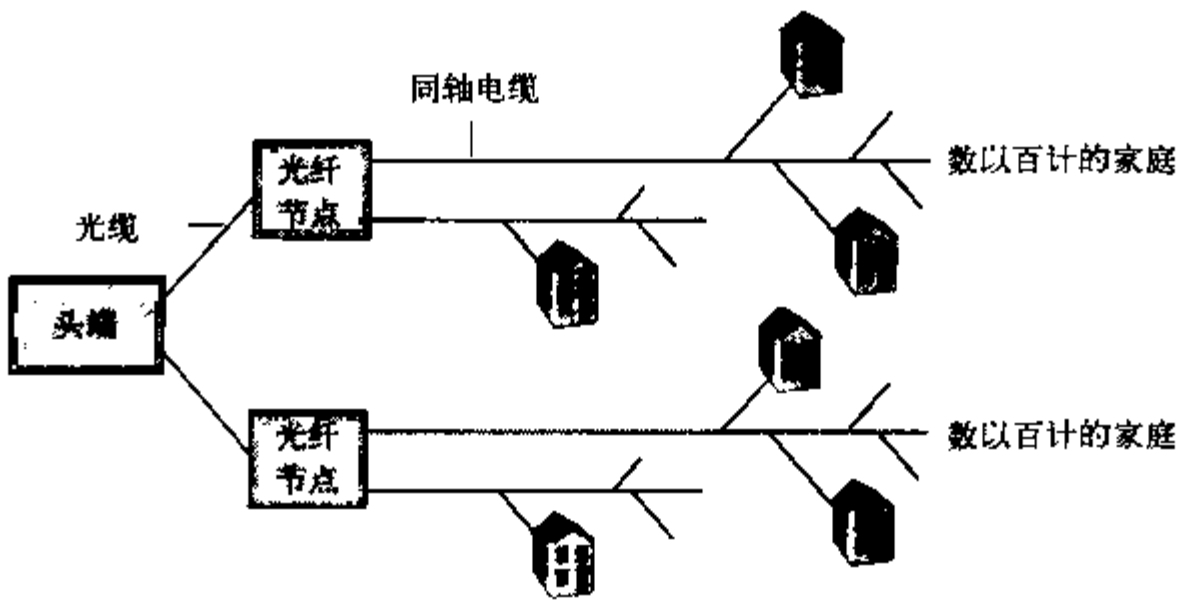


图1-5 一个混合光纤同轴接入网络

DSL、HFC和卫星接入的一个吸引人之处是服务总是在线（always on），即用户能够开着计算机，使这些计算机一直与ISP保持连接，并能够同时拨打和接收普通电话。

2. 公司接入

在公司和大学校园，局域网（LAN）通常被用于连接端用户与边缘路由器。如我们将在第5章所见，局域网技术有许多不同的类型。然而，到目前为止，以太网技术是当前公司网络中最为流行的接入技术。目前，以太网的运行速率是100 Mbps或1 Gbps（甚至到10 Gbps）。它使用双绞铜线或同轴电缆，将一些端系统彼此连接起来，并与边缘路由器连接。边缘路由器负责为目的地不在本局域网的分组选路。与HFC类似，以太网使用共享媒体，因此端系统共享LAN的传输速率。最近，共享以太网技术已经向着交换以太网技术迁移。交换以太网使用星型拓扑使主机直接与一台“交换机”相连，以允许所有主机以LAN的全部速率同时发送和接收。我们将在第5章详细研究共享以太网和交换以太网。

3. 无线接入

伴随着当前因特网技术的革命，无线技术的革命也对人们的工作和生活方式产生了深远影响。今天，欧洲拥有移动电话的人比拥有PC或汽车的人更多。无线化的趋势还在继续，许多分析家预测，全世界的无线（并且通常是移动）手持设备，如移动电话和PDA等，将超过有线联网计算机成为因特网占主导地位的接入设备。目前，有两大类无线因特网接入方式。在**无线局域网**（wireless LAN）中，无线用户与位于几十米半径内的基站（也称为无线接入点）之间传输/接收分组。这些基站通常与有线的因特网相连接，因而为无线用户提供连接到有线网络的服务。在**广域无线接入网**（wide-area wireless access network）中，分组经用于**蜂窝电话**的相同无线基础设施进行发送，基站由电信提供商管理，为数万米半径内的用户提供无线

接入服务。

基于IEEE 802.11技术的无线局域网（也被称为无线以太网和WiFi），当前正在大学的系、商务办公室、咖啡厅和家庭中迅速设置。许多大学在其校园中安装了IEEE 802.11基站，这样学生们就可以在校园的任何地方（例如，图书馆、宿舍、教室或户外校园椅子），发送和接收电子邮件或在Web上冲浪。在许多城市里，人们站在街头，就会处于10或20个基站覆盖中（可以浏览802.11基站的全局图，这些基站已被那些乐于此道的人们发现并在某Web网站上注册，参见[wiggle.net 2007]）。设置最多的802.11技术提供了54 Mbps的共享传输，这些将在第6章中详细讨论。

今天许多家庭正在将宽带住宅接入（即电缆调制解调器或DSL）与廉价的无线局域网技术结合起来，以产生强大的家用网络。图1-6显示了典型的家庭网络的示意图。这个家庭网络组成如下：一台漫游的便携机和1台有线的PC；一个与无线PC通信的基站（无线接入点）；一个提供了与因特网宽带接入的电缆调制解调器；一台互联了基站及固定PC（带有电缆调制解调器）的路由器。该网络允许家庭成员宽带接入因特网，其中一个成员可以从厨房、院子或卧室等处上网。这样一个网络的总固定成本少于150美元（包括电缆/DSL调制解调器）。

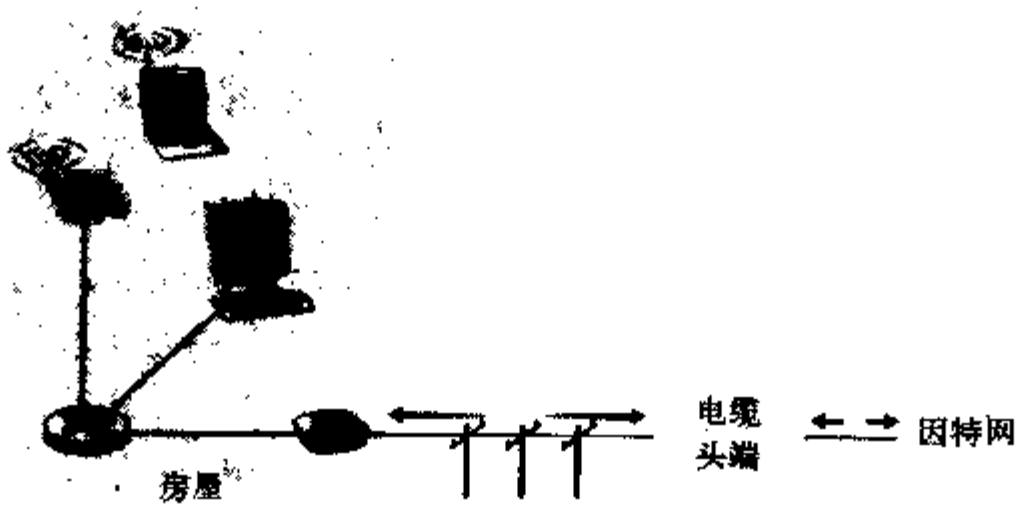


图1-6 一个典型的家庭网络的示意图

当通过无线局域网技术访问因特网时，通常需要位于基站的几十米半径之内。对于家庭接入、咖啡店接入，或更为一般的是围绕一座建筑物或在其内的接入，这是可行的。但是当你坐在海滩上或位于汽车中而需要因特网接入时，应当怎么办呢？对于这种广域接入，漫游的因特网用户利用移动电话基础设施，可接入能够覆盖数万米范围的基站。从概念上讲，这类似于用有线电话线通过拨号连接到因特网的家庭用户，只是现在使用的是蜂窝电话基础设施，而不是有线的电话基础设施。

电信公司已经在第三代（3G）无线中进行了大量投资，3G为分组交换广域无线因特网接入提供了超过1 Mbps的速率。广域无线因特网的两个重要标准是EVDO（Evolution-Data Optimized，被质疑为最糟糕的首字母缩写词之一）和HSDPA（High-Speed Downlink Packet Access），还有许多蜂窝电话网络运营商提供（或打算提供）这两个标准之一或其他标准。与往常一样，有一种潜在的“杀手级”技术在等待着废黜这些标准。WiMAX[Intel WiMAX 2007, WiMAX Forum 2007]也被称为IEEE 802.16，它是前面所讨论的802.11 WiFi协议的一种长距离的变种协议。WiMAX独立于蜂窝网络运行，并承诺跨越数万米的距离且速率为5～10 Mbps或更高。尽管到2006年底WiMAX还没有得到广泛应用，但是Sprint-Nextel已经承诺在

2007年及以后会投资数十亿美元来部署WiMAX。我们将在第6章中详细讨论WiFi、WiMAX和3G技术。

1.2.3 物理媒体

在前面的内容中，我们概述了因特网中的某些最为重要的网络接入技术。在描述这些技术的同时，我们也指出了所使用的物理媒体。例如，我们说过HFC使用了光缆和同轴电缆相结合的技术；拨号56 kbps调制解调器和ASDL使用了双绞铜线；移动接入网络使用了无线电频谱。在这一小节中，我们简要概述一下常在因特网中使用的传输媒体。

为了定义物理媒体所表示的内容，我们仔细考虑一下一个比特的短暂历程。考虑一个比特从一个端系统传输，通过一系列链路和路由器，到达另一个端系统。这个比特被传输许许多多次！源端系统首先传输这个比特，不久后其中的第一台路由器接收该比特；这第一台路由器传输该比特，接着不久后第二台路由器接收该比特，等等。因此，这个比特在从源到目的地传输时，通过一系列“传输接收”对。对于每个传输接收对，通过跨越一种**物理媒体**（physical medium）传播电磁波或光脉冲来发送该比特。该物理媒体能够具有多种形状和形式，并且对沿途的每个传输接收对而言不必具有相同的类型。物理媒体的例子包括双绞铜线、同轴电缆、多模光纤缆、地面无线电频谱和卫星无线电频谱。物理媒体划分为两类：**导引型媒体**（guided media）和**非导引型媒体**（unguided media）。对于导引型媒体，电波沿着固体媒体（如光缆、双绞铜线或同轴电缆）被导引。对于非导引型媒体，电波在空气或外层空间（例如，在无线局域网或数字卫星频道）中传播。

在深入讨论各种媒体类型的特性之前，我们简要地讨论一下它们的成本。物理链路（铜线、光缆等）的实际成本与其他网络成本相比通常是相当小的。特别是安装物理链路相关的劳动力成本比材料成本高几个数量级。正因为这个原因，许多建筑者在一个建筑物的每个房间中同时安装了双绞线、光缆和同轴电缆。即使最初仅使用了一个媒体，在不久的将来也很有可能会使用另一种媒体，这样做就不必在将来铺设另外的线缆，从而节省了经费。

1. 双绞铜线

最便宜并且使用最为普遍的导引型传输媒体是双绞铜线。一百多年来，它一直用于电话网。事实上，从电话机到本地电话交换机有99%以上的连线使用的是双绞铜线。多数人在自己家庭内和工作环境中都已经看到过双绞线。双绞线由两根隔离的铜线组成，每根大约1mm粗，以规则的螺旋形式排列着。这两根线被绞合起来，以减少对邻近双绞线的电气干扰。通常许多双绞线捆扎在一起形成一根电缆，并在这些双绞线外面覆盖上保护性防护层。一对电线构成了一个通信链路。**非屏蔽双绞线**（Unshielded Twisted Pair, UTP）常用在建筑物内的计算机网络中，即用于局域网（LAN）中。目前局域网中的双绞线的数据传输速率从10 Mbps到1 Gbps。所能达到的数据传输速率取决于线的厚度以及传输方和接收方之间的距离。

当20世纪80年代出现光纤技术时，许多人因为双绞线比特速率低而轻视双绞线。某些人甚至认为光纤技术将完全代替双绞线。但双绞线不是那么容易被抛弃的。现代的双绞线技术（例如5类UTP线）能够达到100 Mbps的数据传输速率，距离长达几百米，甚至在更短的距离内传输速率更高。双绞线最终已经作为高速LAN连网的主要方式。

正如在1.2.2节讨论的那样，双绞线也经常用于住宅因特网接入。我们看到拨号调制解调器技术通过双绞线，能以高达56 kbps的速率接入。我们也看到DSL（数字用户线）技术通过双绞线，使住宅用户以超过6 Mbps的速率接入因特网（当用户靠近ISP的调制解调器居住时）。

2. 同轴电缆

与双绞线类似，同轴电缆由两个铜导体组成，但是这两根导体是同心的而不是并行的。借助于这种结构与特殊的绝缘体和保护层，同轴电缆能够具有高比特速率。同轴电缆在电缆电视系统中相当普遍。我们前面已经看到，电缆电视系统最近与电缆调制解调器结合起来，为住宅区用户提供1 Mbps或更高速率的因特网接入。在电缆电视和电缆因特网接入中，发送设备将数字信号调制到一种特定的频段，产生的模拟信号从发送设备传送到一个或多个接收方。同轴电缆能被用作导引式**共享媒体**（shared medium）。特别是，许多端系统能够直接与该电缆相连，而且所有端系统都能接收由其他端系统发送的东西。

3. 光缆

光缆是一种细而柔软的、能够导引光脉冲的媒体，其中每个脉冲表示一个比特。一根光纤能够支持极高的比特速率，高达数十甚至数百Gbps。它们不受电磁干扰，长达100 km的光缆信号衰减极低，并且很难接头。这些特征使得光纤成为长途导引型传输媒体，特别是成为跨海链路的首选媒体。在美国和别的地方，许多长途电话网络现在完全使用光纤。光纤也广泛用于因特网的主干。然而，高成本的光设备，如发射器、接收器和交换机，阻碍光纤在短途传输中的应用，例如在LAN或在家庭接入网络中就不使用它们。光载波（Optical Carrier, OC）标准链路速率的范围从51.8 Mbps到39.8 Gbps；这些规格参数常被称为OC-n，其中的链路速率等于$n \times 51.8$ Mbps。目前正在使用的标准包括OC-1、OC-3、OC-12、OC-24、OC-48、OC-96、OC-192、OC-768。[IEC Optical 2007]、[Goralski 2001]、[Ramaswami 1998]和[Mukherjee 1997]提供了光纤网络的深入知识。

4. 陆地无线电信道

无线电信道承载电磁频谱中的信号。因为不需要安装物理线路，并具有穿透墙壁、提供与移动用户的连接以及长距离承载信号的能力，因而成为一种有吸引力的媒体。无线电信道的特性极大地依赖于传播环境和传输信号的距离。环境上的考虑取决于路径损耗和遮挡衰落（即当信号跨越距离和绕过/通过阻碍物体时，信号强度降低）、多径衰落（由于干扰对象的信号反射）以及干扰（由于其他无线电信道或电磁信号）。

陆地无线电信道可以大致地划分为两类：一类是运行在本地区域，通常跨越数十到几百米；另一类是运行在广域，跨越数万米。1.2.2节中描述的无线LAN技术使用了局域无线电信道；蜂窝接入技术使用了广域无线电信道。我们将在第6章中详细讨论无线电信道。

5. 卫星无线电信道

一颗通信卫星连接两个或多个位于地球的微波发射方/接收方，它们被称为地面站。该卫星在一个频段上接收传输，使用一个转发器（下面讨论）再生信号，并在另一个频率上传输信号。卫星能够提供Gbps的带宽。通信中常使用两类卫星：**同步卫星**（geostationary satellite）和**低地球轨道卫星**（low-earth orbiting satellite）。

同步卫星永久地停留在地球上方相同的点上。这种静止的存在是通过将该卫星放置在地球表面上方的36 000 km的轨道上而取得的。从地面站到卫星再回到地面站的巨大距离引入了280 ms的信号传播时延。不过，能以数百Mbps速率运行的卫星链路，经常用于电话网和因特网的主干。如同1.2.2节讨论的那样，卫星链路在那些无法使用DSL或基于电缆的因特网接入区域，也越来越多地用作高速住宅因特网接入。

低地球轨道卫星较为靠近地球，并且不是永久地停留在地球上方的某个点。它们围绕地

球旋转（就像月球围绕地球旋转那样），彼此通信。为了提供对一个区域的连续覆盖，需要在轨道上放置许多卫星。当前有许多低轨道通信系统在研制中。Lloyd的卫星星群Web页[Wood 2007]提供和收集了有关用于通信的卫星星座系统的信息。未来低地球轨道卫星技术也许能用于因特网接入。

1.3 网络核心

在考察了因特网的边缘后，我们现在更深入地研究网络核心，即互联了因特网端系统的分组交换机和链路的网状网络。图1-7用黑色阴影线勾画出网络核心部分。

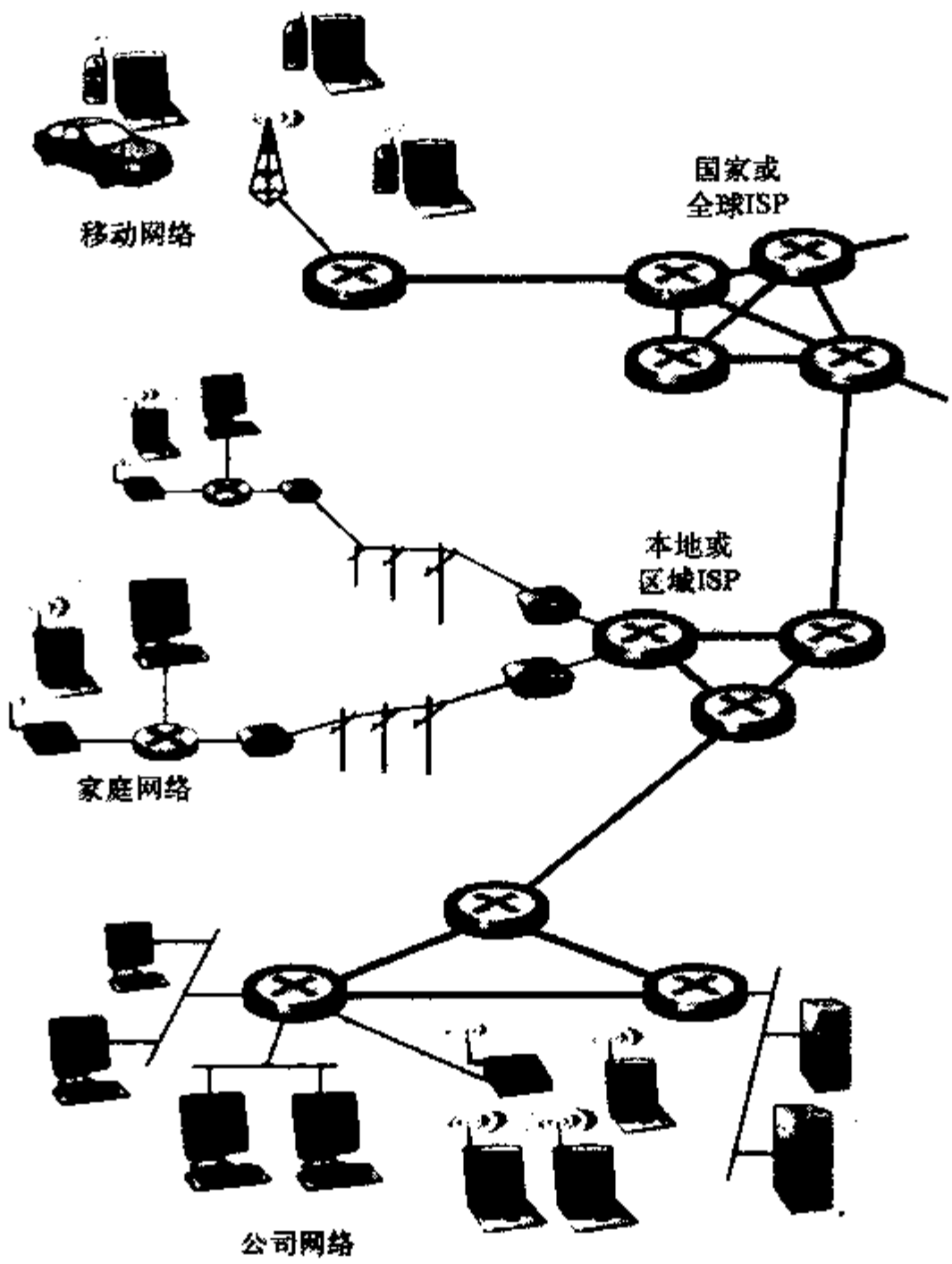


图1-7 网络核心

1.3.1 电路交换和分组交换

通过网络链路和交换机移动数据有两种基本方法：**电路交换**（circuit switching）和**分组交换**（packet switching）。在电路交换网络中，沿着端系统通信路径，为端系统之间通信所提供的资源（缓存、链路传输速率）在通信会话期间会被预留。在分组交换网络中，这些资源则不被预留；会话的报文按需使用这些资源，这将导致可能不得不等待（即排队）接入通信

线路。一个简单的类比是，考虑两家餐馆，一家需要预订，而另一家不需要预订但不保证能安排。对于需要预订的那家餐馆，我们在离开家之前要承受必须先打电话预订的麻烦。但当我们到达该餐馆时，原则上我们能够立即与服务员联系并点菜。对于不需要预订的那家餐馆，我们没有预订餐桌的麻烦，但也许不得不先等到有空闲餐桌后才能找服务员点菜。

无处不在的电话网络是电路交换网络的例子。考虑当一个人通过电话网向另一个人发送信息（话音或传真）时所发生的情况。在发送方能够发送信息之前，该网络必须在发送方和接收方之间建立一个连接。这是一个真正意义上的连接，因为此时沿着发送方和接收方之间路径上的交换机都将为该连接维护连接状态。用电话的术语来说，该连接被称为一条**电路**(circuit)。当网络创建这种电路时，它也在该网络链路上预留了恒定的连接期间的传输速率。因为已经为该发送方到接收方的连接预留了带宽，所以发送方能够以确保的恒定速率向接收方传送数据。

今天的因特网是分组交换网络的典范。考虑当一台主机通过因特网向另一台主机发送分组的情况。如同电路交换一样，分组通过一系列通信链路传输。但对于分组交换而言，分组被送往网络而不必预留任何带宽。如果因为其他分组需要同时经过某链路发送，使该链路之一变得拥塞，则分组将不得不在传输链路的发送侧的缓存中等待，从而形成时延。因特网*尽力而为*（best effort）地以适时的方式传递分组，但它不作任何确保。

并非所有的电信网络都能够被明确地归类为电路交换网络或分组交换网络。然而，这种分组交换和电路交换的基本分类法是理解电信网络技术的一个极好起点。

1. 电路交换

本书的内容是有关计算机网络、因特网和分组交换的，而非电话网络和电路交换的。但是，理解因特网和其他计算机网络为什么使用分组交换而非更为传统的用于电话网的电路交换是很重要的。基于这个原因，我们现在简要概述一下电路交换。

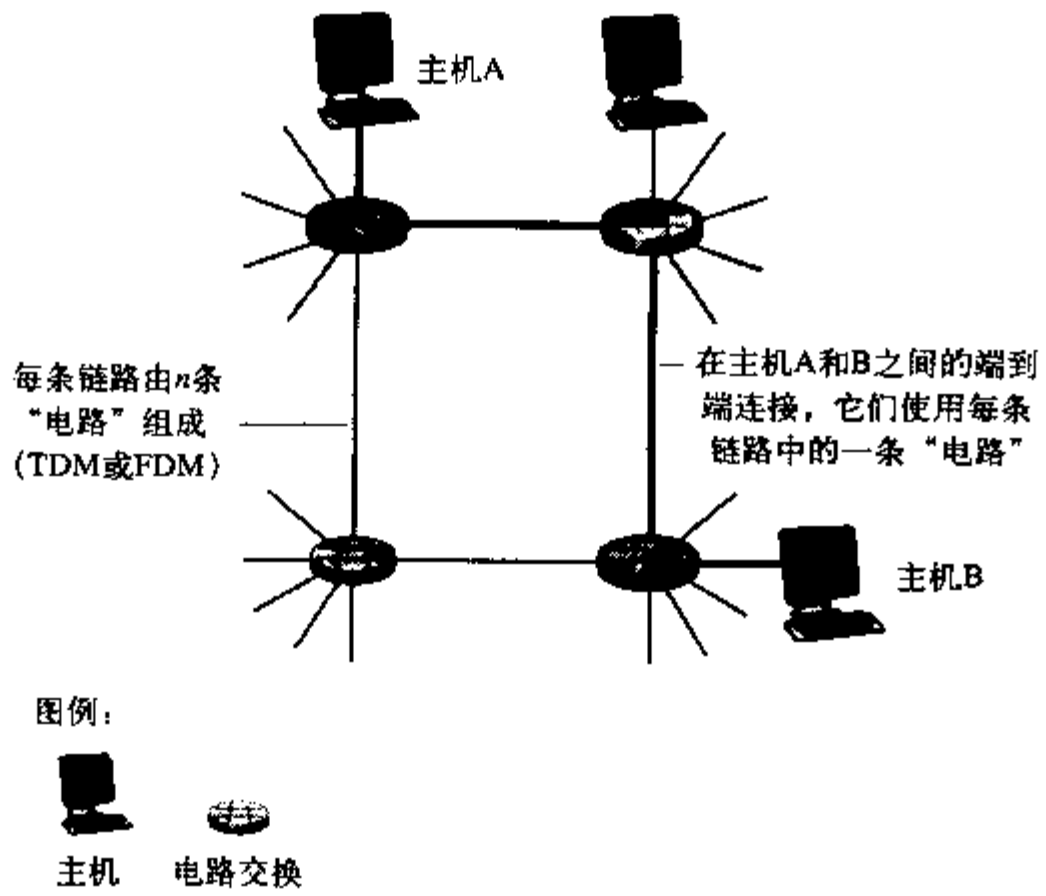


图1-8 由4台交换机和4条链路组成的简单电路交换网络

图1-8显示了一个电路交换网络。在这个网络中，用4条链路互联4台电路交换机。这些链

路的每条都有n条电路，因此每条链路能够支持n条电路同时连接。每台主机（例如PC和工作站）都与一台交换机直接相连。当两台主机要通信时，该网络在两台主机之间创建一条专用的**端到端连接**（end-to-end connection）。（多个设备之间的会议呼叫当然也是可能的。为了简化起见，我们假定每个连接仅有两台主机。）因此，主机A为了向主机B发送报文，该网络必须在两条链路之一上先预留一条电路。因为每条链路具有n条电路，每条链路由端到端连接使用，该连接在连接期间获得该链路带宽的$1/n$部分。

2. 电路交换网络中的多路复用

链路中的电路要么通过**频分多路复用**（Frequency-Division Multiplexing, FDM）实现，要么通过**时分多路复用**（Time-Division Multiplexing, TDM）实现。对于FDM，链路的频谱由跨越链路创建的所有连接所共享。特别是，该链路在连接期间为每条连接专用一个频段。在电话网络中，这个频段通常具有4 kHz（即每秒4000赫兹或4000周）。该频段的宽度被称为**带宽**(bandwidth)。调频无线电台也使用FDM来共享88～108 MHz的频谱，其中每个电台被分配一个特定的频带。

对于一条TDM链路，时间被划分为固定区间的帧，并且每帧又被划分为固定数量的时隙。当网络跨越一条链路创建一条连接时，该网络在每个帧中为该连接指定一个时隙。这些时隙专门由该连接单独使用，一个时隙可用于传输该连接（在每个帧内）的数据。

图1-9显示了支持多达4条电路的特定网络链路的FDM和TDM。对于FDM，其频率域被分割为4个波段，每个波段的带宽是4kHz。对于TDM，其时域被分割为帧，在每个帧中具有4个时隙，每个电路在循环的TDM帧中被分配相同的专用时隙。对于TDM，一条电路的传输速率等于一个时隙中的比特数乘以该帧的速率。例如，如果链路每秒传输8000个帧，每个帧由8个比特组成，则一条电路的传输速率是64 kbps。

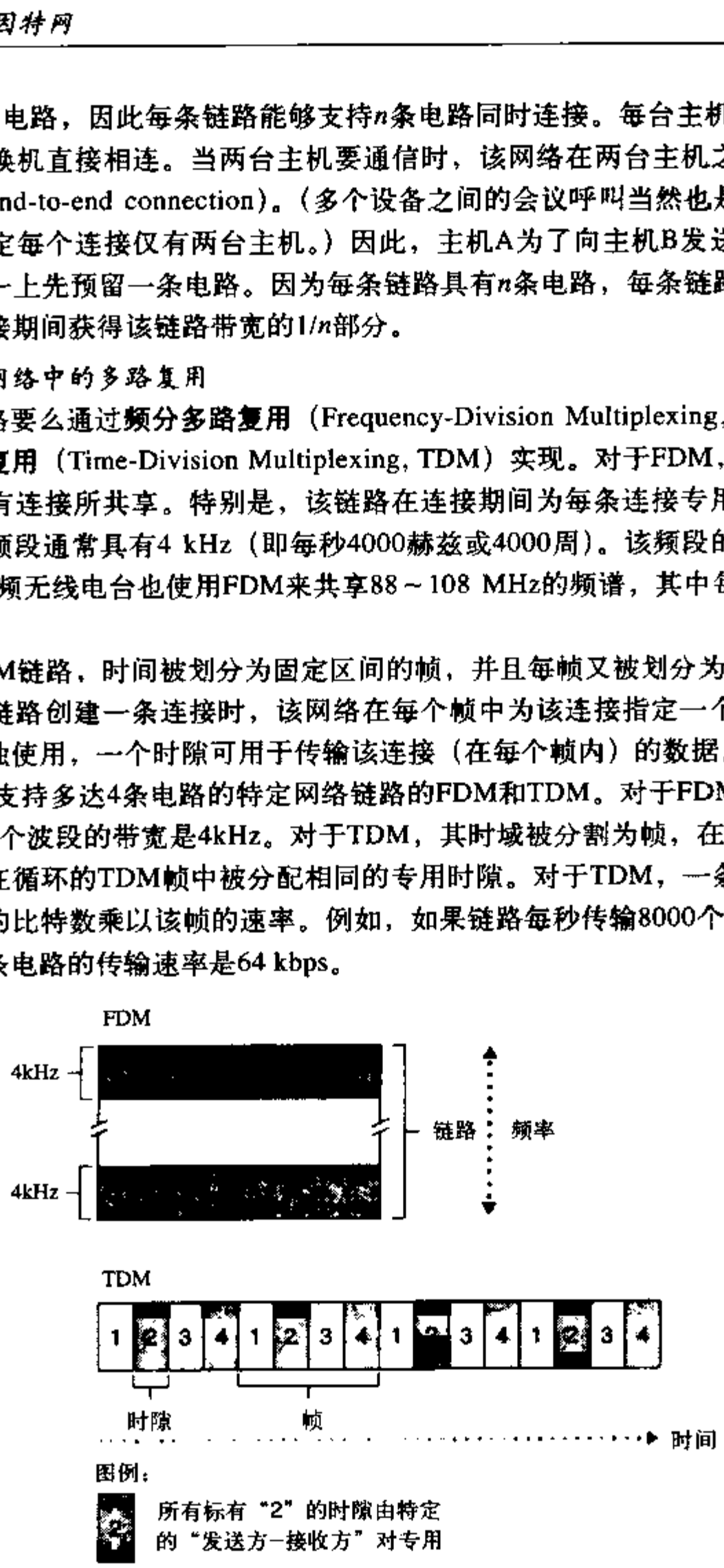


图1-9 对于FDM，每条电路连续地得到部分带宽。对于TDM，每条电路在简短时间间隔（即时隙）中周期性地得到所有带宽

分组交换的支持者总是争辩说，电路交换效率较低，因为在**静默期**（silent period）专用电路空闲。例如，打电话的一个人停止讲话，空闲的网络资源（即沿该连接路径的链路中的

频段或时隙）不能被其他进行中的连接所使用。作为这些资源被无效利用的另一个例子，考虑一名放射科医师使用电路交换网络远程访问一系列X射线图像的情形。该放射科医师建立一条连接，请求一幅图像后，判读该图像，然后请求一幅新图像。在放射科医师判读图像期间，网络资源被浪费了。分组交换的支持者还乐意指出，创建端到端电路和预留端到端带宽是很复杂的，需要复杂的信令软件来协调沿端到端路径的交换机的操作。

在结束电路交换讨论之前，我们讨论一个用数字表示的例子，它更能说明问题的实质。考虑从主机A到主机B经一个电路交换网络需要多长时间发送一个640 kb的文件。假定该网络中的所有链路使用时隙数为24的TDM，并具有1.536 Mbps的比特速率。同时假定在主机A能够开始传输该文件时需要500 ms创建一条端到端电路。那么，它需要多长时间才能发送该文件？每条电路的传输速率是1.536 Mbps/24 = 64 kbps，因此传输该文件需要 (640 kb)/(64kbps) = 10 s。对于这个10 s，再加上电路的创建时间，这样就需要10.5 s发送该文件。注意到该传输时间与链路的数量无关：即使该端到端电路通过一条链路或100条链路，传输时间也是10 s。（实际的端到端时延还包括传播时延，参见1.4节）。

3. 分组交换

各种应用程序在完成其任务时要交换**报文**（message）。报文能够包含协议设计者需要的任何东西。报文可以执行一种控制功能（例如，握手例子中的“你好”报文），或能够包含数据（例如电子邮件数据、JPEG图像或MP3音频文件）。在现代计算机网络中，源主机将长报文划分为较小的数据块，并称之为**分组**（packet）。在源和目的地之间，这些分组中的每个都通过通信链路和**分组交换机**（packet switch）（交换机主要有两类：路由器和链路层交换机）传送。分组以该链路的最大传输速率在通信链路上传输。

多数分组交换机在链路的输入端使用**存储转发传输**（store-and-forward transmission）机制。存储转发传输机制是指在交换机能够开始向输出链路传输该分组的第一个比特之前，必须接收到整个分组。因此，存储转发式分组交换机沿着该分组的路径在每条链路的输入端引入了存储转发时延。考虑从一台主机经分组交换网络向另一台主机发送一个L比特分组需要多长时间。假定在这两台主机之间有Q段链路，每段链路的速率为R bps。假定这是该网络中的唯一分组。从主机A发出的该分组必须首先传输到第一段链路上，这需要L/R s。然后它要在余下的$Q-1$段链路上传输，即它必须存储和转发$Q-1$次，每次都有L/R的存储转发时延，因此总时延为QL/R。

每个分组交换机有多条链路与之相连。对于每条相连的链路，该分组交换机具有一个**输出缓存**（output buffer）（也称为**输出队列**（output queue）），它用于存储路由器准备发往那条链路的分组。该输出缓存在分组交换中起着重要的作用。如果到达的分组需要跨越链路传输，但发现该链路正忙于传输其他分组，该到达分组必须在输出缓存中等待。因此，除了存储转发时延以外，分组还要承受输出缓存的**排队时延**（queue delay）。这些时延是变化的，变化的程度取决于网络中的拥塞水平。因为缓存空间的大小是有限的，所以一个到达的分组可能发现该缓存被等待传输的分组完全充满了。在此情况下，将出现**分组丢失或丢包**（packet lost）——可能是到达的分组也可能是已经排队的分组之一将被丢弃。回想一下本节前面餐馆类比的例子，该排队时延与你在餐馆吧台等待餐桌空闲下来所花费的时间相类似。分组丢失类比于你被服务员告知，已经有太多的其他人在吧台等待桌子就餐，你必须离开这家餐馆。

图1-10显示了一个简单的分组交换网络。在这张图和下面的图中，分组被表示为3维切片。切片的宽度表示分组的长度。在这张图中，所有分组具有相同的宽度，因此有相同的长

度。假定主机A和B向主机E发送分组。主机A和B先通过10 Mbps的以太网链路向第一个分组交换机发送分组，该分组交换机将这些分组导向到一条1.5 Mbps的链路。如果分组到达该交换机的速率超过了该交换机跨越1.5 Mbps的输出链路转发分组的速率，该链路有拥塞，这些分组在通过链路传输之前将在链路输出缓存中排队。我们将在1.4节中更为详细地研究这种排队时延。

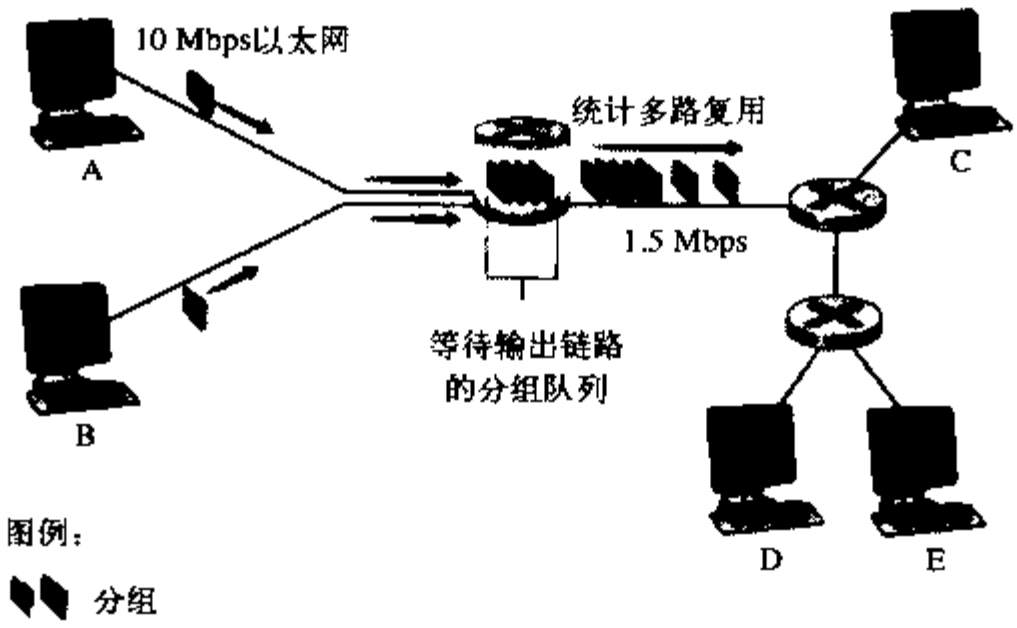


图1-10 分组交换

4. 分组交换与电路交换对比：统计多路复用

在描述了电路交换和分组交换之后，我们来对比一下这两者。分组交换的批评者经常争辩说，分组交换因其端到端时延是变动的和不可预测的（主要是因为排队时延的变动和不可预测所致），故不适合实时服务（例如，电话和视频会议）。分组交换的支持者却争辩道，①它提供了比电路交换更好的带宽共享；②它比电路交换更简单、更有效，实现成本更低。分组交换与电路交换之争的有趣讨论参见[Molinero-Fernandez 2002]。概括而言，嫌餐馆预订麻烦的人宁可要分组交换而不愿意要电路交换。

分组交换为什么更有效呢？我们看一个简单的例子。假定多个用户共享一条1 Mbps链路。再假定每个用户的活动性周期是变化的，即某用户时而以100 kbps恒定速率产生数据，时而静止（这时用户不产生数据）。进一步假定该用户仅有10%的时间活动（在余下的90%的时间空闲下来喝咖啡）。对于电路交换，在所有的时间内必须为每个用户预留100 kbps。例如，对于电路交换的TDM，如果一个1秒的帧被划分为10个时隙，每个时隙为100 ms，则每个用户每帧将被分配一个时隙。

因此，该链路仅能支持10（= 1 Mbps/100 kbps）个同时上网的用户。对于分组交换，一个特定用户活动的概率是0.1（即10%）。如果有35个用户，有11或更多个同时上网活动的用户的概率大约是0.0004。（课后习题7概述了如何得到这个概率值。）当有10个或更少的同时上网活动的用户时（发生概率为0.9996），到达的数据率总和小于或等于该链路的输出速率1 Mbps。因此，当有10个或更少的活动用户时，通过该链路的分组流基本上没有时延，与电路交换的情况一样。当同时活动的用户超过10个时，分组的到达率总和超过该链路的输出容量，则输出队列将开始变长（一直增长到输入速率总和重新低于1 Mbps，此后该队列才会减少长度）。因为本例中超过10个同时活动用户的概率极小，所以分组交换差不多总是具有与电路交换相同的性能，并且用户数量是所支持的数量的3倍多时也是如此。

我们现在考虑第二个简单的例子。假定有10个用户，某个用户突然产生1000个1 kb的分

组，而其他用户则保持静默，不产生分组。在具有每帧10个时隙并且每个时隙包含1 kb的TDM电路交换情况下，活动用户仅能使用每帧中的一个时隙来传输数据，而每个帧中剩余的9个时隙保持空闲。活动用户的所有1 Mb数据要传输完，需要10 s的时间。在分组交换情况下，活动用户能够连续地以1 Mbps的全部链路速率发送其分组，因为没有其他用户产生的分组需要与该活动用户的分组进行多路复用。在此情况下，该活动用户的所有数据将在1 s内发送完毕。

上面的例子从两个方面表明了分组交换的性能能够优于电路交换的性能。这些例子也强调了在多个数据流之间共享链路的两种形式，其传输速率的关键差异。电路交换不考虑要求而预先分配传输链路的使用，这使得已分配但不需要的链路时间未被利用。另一方面，分组交换使用按需的方式分配链路。链路传输能力将只在所有的其分组要在链路上传输的用户中，逐分组地被共享。这样的按需（而不是预分配）共享资源有时被称为资源的**统计多路复用**(statistical multiplexing)。

虽然分组交换和电路交换在今天的电信网络中都是普遍采用的方式，但趋势无疑是朝着分组交换方向发展。甚至许多今天的电路交换电话网正在缓慢地向分组交换迁移。特别是，电话网经常在电话昂贵的海外电话部分使用分组交换。

1.3.2 分组是怎样通过分组交换网形成其通路的

前面我们说过，路由器从与它相连的一条通信链路得到分组，将分组转发到与它相连的另一条通信链路。但是，路由器怎样确定它应当向哪条链路进行转发呢？不同的计算机网络实际上是以不同的方式完成的。在这里，我们将描述一种流行的方法，即因特网所采用的方法。

在因特网中，每个通过该网络传输的分组在它的首部包含了其目的地址。就像邮政地址一样，该地址是一种层次结构。当分组到达网络中的一台路由器时，该路由器检查分组的目的地址的一部分，并向相邻路由器转发该分组。更特别的是，每台路由器具有一个转发表，用于将目的地址（或目的地址的一部分）映射到输出链路。当分组到达一台路由器时，该路由器检查目的地址，并用这个目的地址搜索转发表，以找到合适的输出链路。然后，路由器将该分组导向输出链路。

我们刚刚学习了路由器使用分组的目的地址来索引转发表并决定合适的输出链路。但是这里还回避了另一个问题：转发表是如何设置的？是通过人工对每台路由器逐台进行配置的，还是因特网使用更自动的过程进行设置的呢？第4章将深入地探讨这个问题。但为了激发你的求知欲，我们现在将告诉你因特网具有一些特殊的选路协议，它们用于自动地设置**转发表**(forwarding table)。例如，选路协议可以决定从每台路由器到每个目的地的最短路径，并使用这些最短路径来配置路由器中的转发表。

端到端选路过程也与一个不使用地图而喜欢问路的汽车驾驶员相类似。例如，假定Joe驾车从费城到佛罗里达州奥兰多市Lakeside Drive街156号。Joe先驾车到附近的加油站，询问怎样才能到达佛罗里达州奥兰多市Lakeside Drive街156号。该加油站的服务员从该地址中抽取了佛罗里达州部分，告诉Joe他需要上I-95南州际公路，该公路恰有一个与该加油站临近的入口。他又告诉Joe，一旦到了佛罗里达后应当再问当地人。Joe然后上了I-95南州际公路，一直到达佛罗里达的Jacksonville，在那里他向另一个加油站服务员问路。该服务员从地址中抽取了奥兰多市部分，告诉Joe他应当继续沿I-95公路到Daytona海滩，然后再问其他人。Daytona海滩的另一个加油站服务员也抽取了该地址的奥兰多市部分，告诉Joe应当走I-4公路直接前往奥兰多市。Joe走了I-4公路，并从奥兰多市出口下来。Joe又向另一个加油站的服务员询问，

这时该服务员抽取了该地址中的Lakeside Drive部分，告诉Joe到Lakeside Drive必须要走的路。当Joe到达了Lakeside Drive街时，他向一个骑自行车的小孩询问如何到达他的目的地。这个孩子抽取了该地址的156号部分，并指出了房屋的方向。Joe最后到达了他的最终目的地。

在上述类比中，加油站服务员和骑自行车的孩子就好比是路由器，他们大脑中的转发表已经由多年积累起来的经验进行了配置。

怎样才能实际看到分组在因特网中所采用的端到端路径？现在请你自己用一下Traceroute程序，有关情况请访问站点http://www.traceroute.org（参见1.4节有关Traceroute的讨论）。

1.3.3 ISP和因特网主干

我们在前面看到，端系统（用户PC、PDA、Web服务器、电子邮件服务器等）通过接入网与因特网相连。前面也讲过，接入网可以是有线的或无线的局域网（例如，在一个公司、学校或图书馆），也可以是住宅电缆调制解调器或DSL网络，还可以是通过拨号调制解调器接入的住宅ISP（例如，AOL或MSN）。但是，将端用户和内容提供商连接到接入网仅是解决难题的很小的一部分，因为因特网是由数以亿计的用户和几十万个网络构成的。因特网是网络的网络，理解这个术语是解决该难题的关键。

在公共因特网中，坐落在因特网边缘的接入网络通过分层的ISP层次结构与因特网的其他部分相连，如图1-11所示。接入ISP（例如，AOL这样的住宅电缆和DSL网络、拨号接入网络，无线接入网络，使用LAN的公司和大学ISP）位于该层次结构的底部。该层次结构的最顶层是数量相对较少的**第一层ISP**（tier-1 ISP）。第一层ISP在许多方面与其他网络相同，它有链路和路由器，并与其他网络相连。然而，在另外一些方面，第一层ISP是特殊的。它们的链路速率通常是622 Mbps或更高，对于大型第一层ISP，其链路速率的范围是2.5～10 Gbps，相应地其路由器也必须能够以极高的速率转发分组。第一层ISP的特性可以表示为：

- 直接与其他每个第一层ISP相连。
- 与大量的第二层ISP和其他客户网络相连。
- 覆盖国际区域。

第一层ISP也被称为**因特网主干**（Internet backbone）网络，包括Sprint、Verizon、MCI（以前的UUNet/WorldCom）、AT&T、NTT、Level3、Qwest和Cable & Wireless。有趣的是，没有任何组织正式批准第一层的状态。如名言所说，如果必须问你是否是一个组织的成员的话，你可能不是。

第二层ISP通常具有区域性或国家性覆盖规模，并且非常重要地仅与少数第一层ISP相连接（参见图1-11）。因此，为了到达全球因特网的大部分区域，第二层ISP需要引导流量通过它所连接的第一层ISP。第二层ISP被称为是它所连接的第一层ISP的**客户**（customer），第一层ISP相对该客户而言是**提供商**（provider）。许多大公司和机构将它们的企业网直接与第一层或第二层ISP相连，因而成为该ISP的客户。一个提供商ISP向它的客户收费，费用通常根据连接两者的带宽而定。一个第二层网络也可以选择与其他第二层网络直接相连，在这种情况下，流量能够在两个第二层网络之间流动，而不必流经某第一层网络。在第二层ISP之下是较低层的ISP，这些较低层ISP经过一个或多个第二层ISP与更大的因特网相连。在该层次结构的底部是接入ISP。在更为复杂的情况下，某些第一层提供商也是第二层提供商（即垂直集成），它们除了向较低层ISP出售因特网接入外，也直接向端用户和内容提供商出售因特网接入。当两个ISP彼此直接相连时，它们被称为彼此是**对等**（peer）的。一个有趣的研究[Subramanian

2002]是根据客户—提供商和对等关系，通过研究因特网的拓扑来寻求更为精确地定义因特网的层次结构的方式。

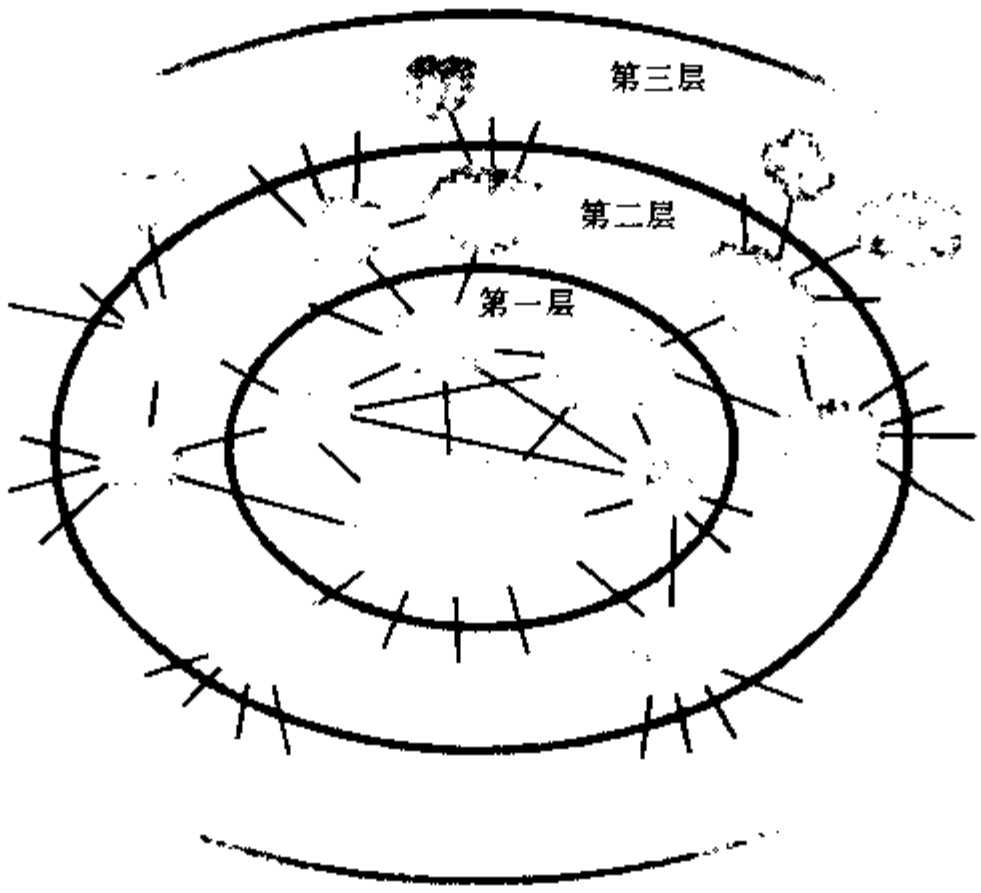


图1-11 ISP的互联

在一个ISP的网络中，某ISP与其他ISP（无论它在该层次结构的下面、上面或相同层次）的连接点被称为到**汇集点**（Point of Presence, POP）。POP就是某ISP网络中的一台或多台路由器组，通过它们能够与其他ISP的路由器连接。一个第一层提供商通常具有许多POP，这些POP分散在其网络中不同的地理位置，每个POP与多个客户网络和其他ISP相连。对于一个与提供商POP连接的客户网络而言，该客户通常从第三方电信提供商租用一条高速链路，并且直接将它的一台路由器与该提供商位于POP的一台路由器相连。两个第一层ISP也可以将一对POP连接在一起，形成彼此对等，这一对POP的每端分别属于这两个ISP之一。此外，两个ISP可以具有多个对等点，将两个或更多POP对彼此相连起来。

总之，因特网的拓扑是很复杂的，它由几十个第一层ISP和第二层ISP与数以千计的较低层ISP组成。ISP覆盖的区域不同，有些跨越多个大洲和大洋，有些限于世界的很小区域。较低层的ISP与较高层的ISP相连，较高层ISP彼此互联。用户和内容提供商是较低层ISP的客户，较低层ISP是较高层ISP的客户。

1.4 分组交换网中的时延、丢包和吞吐量

回想在1.1节中我们讲过，可以将因特网看成是为运行在端系统上的分布式应用提供服务的基础设施。理想情况下，我们希望因特网服务能够在任意两个端系统之间瞬间移动大量数据，而没有任何数据丢失。然而，这是一个极高的目标，现实中很难做到。与之相反，计算机网络必定要限制端系统之间的吞吐量（每秒能够传输的数据量），在端系统之间引入时延，并能够丢包。一方面，实际中引入时延、丢包和对吞吐量的限制是不好的。而另一方面，因为计算机网络存在这些问题，所以围绕着如何处理这些问题有许多令人着迷的问题，问题多得足以开设一门有关计算机网络方面的课程，可以做上百篇博士论文！在本节中，我们将开

始研究计算机网络中的时延、丢包和吞吐量等问题，并使之量化。

1.4.1 分组交换网中的时延概述

前面讲过，分组从一台主机（源）出发，通过一系列路由器传输，在另一台主机（目的地）结束它的历程。当分组从一个节点（主机或路由器）沿着这条路径到后继节点（主机或路由器）时，该分组在沿途的每个节点都经受了几种不同类型的时延。这些时延中最为重要的是**节点处理时延**（nodal processing delay）、**排队时延**（queuing delay）、**传输时延**（transmission delay）和**传播时延**（propagation delay），这些时延总体累加起来是**节点总时延**（total nodal delay）。为了深入理解分组交换和计算机网络，我们必须理解这些时延的性质和重要性。

时延的类型

我们来探讨一下图1-12中列出的这些时延。作为源和目的地之间的端到端路径的一部分，一个分组从上游节点通过路由器A向路由器B发送。我们的目的是在路由器A刻画出节点时延。注意到路由器A具有通往路由器B的输出链路。该链路前面有一个队列（也称为缓存）。当该分组从上游节点到达路由器A时，路由器A检查该分组的首部以决定该分组的适当的输出链路，并将该分组导向该链路。在这个例子中，该分组的输出链路是通向路由器B的那条链路。仅当该链路没有其他分组在传输并且没有其他分组排在该队列前面时，才能在这条链路上传输该分组；如果该链路当前繁忙或有其他分组已经在该链路上排队，则新到达的分组将参与排队。

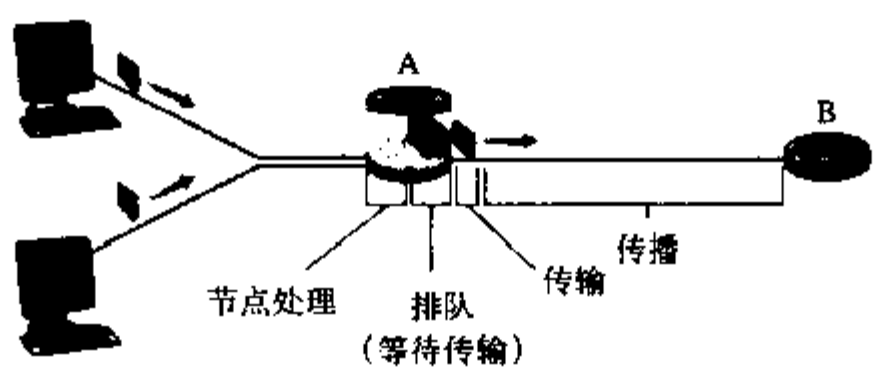


图1-12 在路由器A的节点时延

1. 处理时延

检查分组首部和决定将该分组导向何处所需要的时间是处理时延的一部分。处理时延也包括其他因素，如检查比特级差错所需要的时间，该差错出现在这些分组比特从上游节点向路由器A传输的过程中。高速路由器的处理时延通常是微秒或更低的数量级。在这种节点处理之后，路由器将该分组引向通往路由器B链路之前的队列。（在第4章中，我们将研究路由器运行的细节。）

2. 排队时延

在队列中，当分组在链路上等待传输时，它经受排队时延。一个特定分组的排队时延将取决于先期到达的、正在排队等待向链路传输的分组的数量。如果该队列是空的，并且当前没有其他分组在传输，则该分组的排队时延为0。另一方面，如果流量很大，并且许多其他分组也在等待传输，该排队时延将很大。我们将很快看到，到达组的分组数量是到达该队列的流量的强度和性质的函数。实际的排队时延通常在毫秒到微秒级。

3. 传输时延

假定分组以先到先服务方式传输——这在分组交换网中是常见的方式，仅当所有已经到达的分组被传输后，才能传输我们的分组。用L比特表示分组的长度，用R bps表示从路由器A到

路由器B的链路传输速率。例如，对于一条10 Mbps的以太网链路，速率R = 10 Mbps；对于100 Mbps的以太网链路，速率R=100 Mbps。传输时延（又称为存储转发时延，在1.3节中讨论过）是L/R。这是将所有分组的比特推（传输）向链路所需要的时间。实际的传输时延通常在毫秒到微秒级。

4. 传播时延

一旦一个比特被推向链路，该比特需要向路由器B传播。从该链路的起点到路由器B传播所需要的时间是**传播时延**。该比特以该链路的传播速率传播。该传播速率取决于该链路的物理媒体（即光纤、双绞铜线等），其速率范围是$2 \times 10^8 \sim 3 \times 10^8$ m/s，这等于或略小于光速。传播时延等于两台路由器之间的距离除以传播速率，即传播时延是d/s，其中d是路由器A和路由器B之间的距离，s是该链路的传播速率。一旦该分组的最后一个比特传播到节点B，该比特及前面的所有比特都被存储于路由器B。整个过程将随着路由器B执行转发而持续下去。在广域网中，传播时延在毫秒的量级。

5. 传输时延和传播时延的比较

计算机网络领域的新手有时难以理解传输时延和传播时延之间的差异。该差异虽说细小但很重要。传输时延是路由器将分组推出所需要的时间，它是分组长度和链路传输速率的函数，而与两台路由器之间的距离无关。另一方面，传播时延是一个比特从一台路由器向另一台路由器传播所需要的时间，它是两台路由器之间距离的函数，但与分组的长度或链路的传输速率无关。

下面通过一个类比来阐明传输时延和传播时延的概念。考虑一条公路每100 km有一个收费站，如图1-13所示。你可认为收费站间的公路段是链路，收费站是路由器。假定汽车以每小时100 km（100 km/h）的速度在该公路上行驶，即传播（也就是说，当一辆汽车离开一个收费站时，它立即加速到100 km/h并在收费站间维持该速度）。假定有10辆汽车组成车队在行驶，并且这10辆汽车以固定的顺序互相跟随。可以将每辆汽车看成是一个比特，该车队是一个分组。我们也假定每个收费站以每12 s通过一辆汽车的速度提供服务（即传输），由于时间是深夜，因此该车队是该公路上唯一的一批汽车。最后，假定无论该车队的第一辆汽车何时到达收费站，它都在入口处等待直到其他9辆汽车到达并整队依次前行。（因此，整个车队在它能够“转发”之前，必须存储在收费站。）收费站将整个车队推向公路所需要的时间是（10辆车）/（5辆车/分钟）= 2分钟。该时间类似于路由器中的传输时延。因此，对于一辆汽车来说，从一个收费站出口行驶到下一个收费站所需要的时间是100 km/（100 km/h）= 1h。这个时间类似于传播时延。因此，从该车队存储在收费站前到该车队存储在下一个收费站前的时间是“传输时延”加“传播时间”，在本例子中为62分钟。

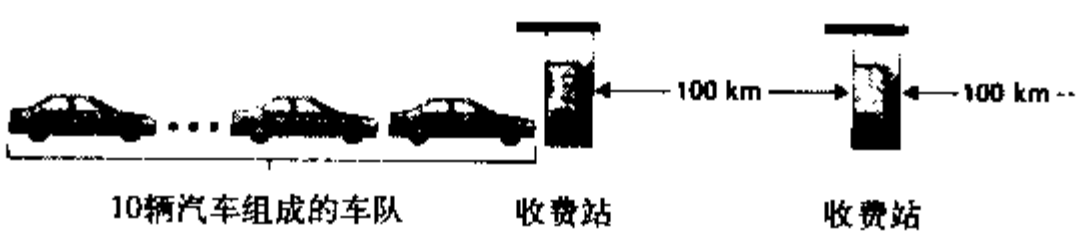


图1-13 车队的类比例子

我们更深入地来探讨一下这个类比例子。如果收费站对车队的服务时间大于汽车在收费站之间行驶的时间，将会发生什么情况呢？例如，假定现在汽车是以1000 km/h的速度行驶，收费站是以每分钟一辆汽车的速度为汽车提供服务，则汽车在两个收费站之间的行驶时延是6

分钟，收费站服务车队的时间是10分钟。在此情况下，该车队中的最后几辆汽车离开第一个收费站之前，前面的几辆汽车将会到达第二个收费站。这种局面在分组交换网中也会发生。一个分组中的前几个比特到达了一台路由器，而该分组中还有余下的比特仍然在前面的路由器中等待传输。

如果说一幅图胜过千言万语的话，则一个动画必定胜过上百万言语。本书配套的Web网站提供了一个交互式Java小程序，它很好地展现了传输时延和传播时延。我们强烈推荐读者访问该Java小程序。

如果令d_{proc}、d_{queue}、d_{trans}和d_{prop}分别表示处理时延、排队时延、传输时延和传播时延，则节点的总时延由下式给定：

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

这些时延成分所起的作用可能变化很大。例如，d_{prop}对于连接两台位于同一校园内的路由器的链路而言可能是微不足道的（例如，几个微秒）；然而，d_{proc}对于由同步卫星链路互联的两台路由器是几百毫秒，可能是d_{nodal}中的主要成分。类似地，d_{trans}的影响可能是微不足道的，也可能是很大的。通常对于10 Mbps和更高的传输速率（例如，对于LAN）的信道而言，d_{trans}的影响是微不足道的；然而，对于通过低速拨号调制解调器链路发送的大的因特网分组而言，d_{trans}可能是数百毫秒。处理时延d_{proc}通常是微不足道的；然而，它对一台路由器的最大吞吐量有重要影响，最大吞吐量是一台路由器能够转发分组的最大速率。

1.4.2 排队时延和丢包

节点时延的最为复杂和令人感兴趣的成分是排队时延d_{queue}。事实上，排队时延在计算机网络中的重要程度和人们感兴趣的程度，从发表的数以千计的论文和大量的专著可见一斑[Bertsekas 1991; Daigle 1991; Kleinrock 1975, 1976; Ross 1995]。我们这里仅给出有关排队时延的总体的、直觉的讨论，求知欲强的读者可以浏览其他书籍（或者最终写有关这方面的博士论文）。与其他三项时延（即d_{proc}、d_{trans}和d_{proc}）不同的是，排队时延对不同的分组是不同的。例如，如果10个分组同时到达空队列，传输的第一个分组没有排队时延，而传输的最后一个分组将经受相对大的排队时延（这时它要等待其他9个分组被传输）。因此，当表征排队时延时，人们通常使用统计量测度，如平均排队时延、排队时延的方差和排队时延超过某些特定值的概率。

什么时候排队时延大，什么时候又不大呢？该问题的答案很大程度上取决于流量到达该队列的速率、链路的传输速率和到达流量的性质，即流量是周期性到达还是以突发形式到达。为了深入理解这些，令a表示分组到达队列的平均速率（a的单位是每秒分组，即pkt/s）。前面定义了R是传输速率，即比特从队列中推出的速率（以bps为单位）。为了简单起见，也假定所有分组都是由L比特组成的，则比特到达队列的平均速率是La bps。最后，假定该队列非常大，因此它基本能容纳无限数量的比特。比率La/R被称为**流量强度**（traffic intensity），它在估计排队时延的影响程度方面经常起着重要的作用。如果$La/R > 1$，则比特到达队列的平均速率超过从该队列传输出去的速率。在这种不幸的情况下，队列的增加将趋于无界，并且排队时延将趋向无穷大！因此，流量工程中的一条金科玉律是：*设计系统时流量强度不能大于1*。

现在考虑$La/R \leqslant 1$时的情况。这时，到达流量的性质影响排队时延。例如，如果分组周期性到达，即每L/R s到达一个分组，则每个分组将到达一个空队列中，因此不会有排队时延。另一方面，如果分组以突发形式到达而不是周期性到达，则可能有很大的平均排队时延。例

如，假定每L/R s同时到达N个分组，则传输的第一个分组没有排队时延；传输的第二个分组就有L/R s的排队时延；更为一般地，第n个传输的分组具有$(n-1)$ L/R s的排队时延。我们将该例子中计算平均排队时延的问题留给读者作为练习。

以上描述周期性到达的两个例子有些学术味。到达队列的过程通常是随机的，即并不遵循任何模式到达，分组之间的时间间隔是随机的。在这种更为真实的情况下，量La/R通常不能全面地、充分地表征时延的统计量。不过，直观地理解排队时延的影响程度很有用。特别是，如果流量强度接近于0，则几乎没有分组到达并且到达间隔很大，那么到达的分组将不可能在队列中发现别的分组。因此，平均排队时延将接近于0。另一方面，当流量强度接近于1时，将存在到达率超过传输能力的时间间隔（由于到达的突发性），从而将形成队列。随着流量强度接近于1，平均排队长度变得越来越长。平均排队时延与流量强度的定性关系如图1-14所示。

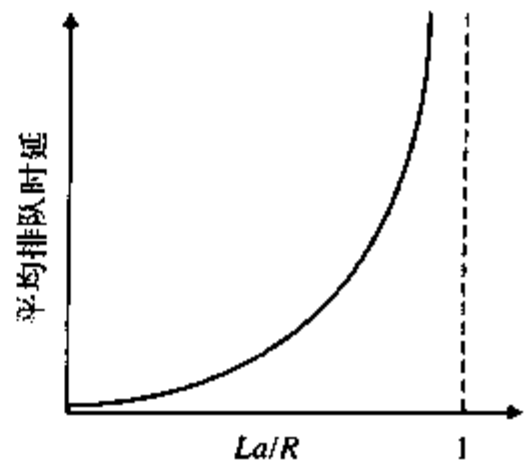


图1-14 平均排队时延与流量强度的关系

对图1-14的一个重要理解是，随着流量强度接近于1，平均排队时延迅速增加。流量强度的少量增加将导致时延的大量增加。也许你在公路上见到过这种现象。如果你规律地行驶在通常拥塞的公路上，那么这条经常拥塞的道路意味着它的流量强度接近于1。此时如果某些事件引起一个甚至稍微大于平常量的流量，那么你经受的时延将会很大。

为了真实地感受一下排队时延的情况，我们再次鼓励你访问本书的Web网站，该网站提供了一个有关排队的交互式Java程序。如果你将分组到达率设置得足够大，使流量强度超过1，你将看到经过一段时间队列慢慢地建立起来。

丢包

在上述讨论中，我们已经假设队列能够保持数量无穷多的分组。在现实中，虽说排队容量极大地依赖于路由器的设计和成本，但排在一条链路前的队列只有有限的容量。因为排队容量是有限的，所以流量强度接近于1时排队时延也不会趋向无穷大。相反，到达的分组将发现一个满的队列。由于没有地方存储这个分组，路由器将丢弃（drop）该分组，即该分组将会丢失（lost）。当流量强度大于1时，队列中的这种溢出也能够在有关队列的Java小程序中看到。

从端系统的角度看，上述现象看起来是一个分组已经传输到网络核心，但它绝不会出现在目的地。丢失分组的数量随着流量强度的增加而增加。因此，节点的性能常常不仅要根据时延来度量，而且要根据分组丢失的概率来度量。正如我们将在后面各章中讨论的那样，丢失的分组可能要基于端到端的原则重传，以确保所有的数据最终从源传送到了目的地。

1.4.3 端到端时延

到目前为止，我们的讨论一直集中在节点时延上，即单台路由器的时延。下面考虑从源到目的地的总时延。为了便于讨会这个概念，假定在源主机和目的主机之间有$N-1$台路由器，并且该网络是无拥塞的（因此排队时延是微不足道的），在每台路由器和源主机的处理时延是d_{proc}，每台路由器和源主机的输出速率是R bps，每条链路的传播时延是d_{prop}。节点时延累加起来得到端到端时延：

$$d_{end\text{-}end} = N\ (d_{proc} + d_{trans} + d_{prop})$$

式中$d_{trans} = L/R$，L是分组长度。在各节点具有不同的时延和每个节点存在平均排队时延的情

况下，需要对该公式进行一般化处理，我们将这项工作留给读者。

1. Traceroute

为了对计算机网络中的时延有一个第一手认识，我们可以利用Traceroute程序。Traceroute是一个简单的程序，它能够在任何因特网主机上运行。当用户指定一个目的主机名字时，源主机中的该程序朝着该目的地发送多个特殊的分组。当这些分组向着目的地传送时，它们通过一系列路由器。当路由器接收到这些特殊分组之一时，它向源回送一个短报文，该报文包括该路由器的名字和地址。

更具体地，假定在源和目的地之间有$N-1$台路由器，则源将向网络发送N个特殊的分组，其中每个分组地址指向最终目的地。这N个特殊分组标识为1到N，第一个分组标识为1，最后的分组标识为N。当第n台路由器接收到第n个标识为n的分组时，该路由器不是向它的目的地转发该分组，而是向源回送一个报文。当目的主机接收第N个分组时，它也会向源返回一个报文。该源记录了从它发送一个分组到它接收到对应返回报文所经受的时间，它也记录了返回该报文的路由器（或目的地主机）的名字和地址。以这种方式，源能够重建分组从源到目的地所采用的路由，并且该源能够决定到所有中间路由器的往返时延。Traceroute实际上对刚才描述的实验重复了3次，因此该源实际上向目的地发送了$3 \times N$个分组。RFC 1393详细地描述了Traceroute。

这里有一个Traceroute程序输出的一个例子，其中追踪的路由从源主机gaia.cs.umass.edu（位于马萨诸塞大学）到cis.poly.edu（位于布鲁克林的理工大学）。输出具有6列：第一列是前面描述的n值，即沿着路径上的路由器号码；第二列是路由器的名字；第三列是路由器地址（格式为xxx.xxx.xxx.xxx）；最后3列是3次实验的往返时延。如果源从任何给定的路由器接收到少于3条报文（由于网络中的丢包），则Traceroute在该路由器号码后面放一个星号，并向那台路由器报告少于3次往返时间。

```
1 cs-gw (128.119.240.254) 1.009 ms 0.899 ms 0.993 ms
2 128.119.3.154 (128.119.3.154) 0.931 ms 0.441 ms 0.651 ms
3 border4-rt-gi-1-3.gw.umass.edu (128.119.2.194) 1.032 ms 0.484 ms 0.451 ms
4 acr1-ge-2-1-0.Boston.cw.net (208.172.51.129) 10.006 ms 8.150 ms 8.460 ms
5 agr4-loopback.NewYork.cw.net (206.24.194.104) 12.272 ms 14.344 ms 13.267 ms
6 acr2-loopback.NewYork.cw.net (206.24.194.62) 13.225 ms 12.292 ms 12.148 ms
7 pos10-2.core2.NewYork1.Level3.net (209.244.160.133) 12.218 ms 11.823 ms 11.793 ms
8 gige9-1-52.hsipaccess1.NewYork1.Level3.net (64.159.17.39) 13.081 ms 11.556 ms 13.297 ms
9 p0-0.polyu.bbnplanet.net (4.25.109.122) 12.716 ms 13.052 ms 12.786 ms
10 cis.poly.edu (128.238.32.126) 14.080 ms 13.035 ms 12.802 ms
```

在上述跟踪中，在源和目的地之间有9台路由器。这些路由器中的多数有一个名字，所有都有地址。例如，路由器3的名字是`border4-rt-gi-1-3.gi-1-3.gw.umass.edu`，它的地址是`128.119.2.194`。看看为相同的路由器提供的数据，可以看到在源和路由器之间的往返时延：3次试验中的第一次是1.03 ms，后继两次试验的往返时延是0.48 ms和0.45 ms。这些往返时延包括刚才讨论的所有时延，即传输时延、传播时延、路由器处理时延和排队时延。因为排队时延随时间变化，所以分组n发送到路由器n的往返时延实际上比分组$n+1$发送到路由器$n+1$的往返时延更大。的确，我们在上述例子中观察到了这种现象：到路由器6的时延比到路由器7的时延大！

你想自己试试Traceroute程序吗？我们强烈推荐你访问http://www.traceroute.org，它的Web界面提供了有关路由跟踪的广泛的源列表。你选择一个源，并为任何目的地提供主机名，该Traceroute程序则会完成所有工作。有一些为Traceroute提供图形化界面的免费软件程序，

我们喜爱的一个程序是PingPlotter[PingPlotter 2007]。

2. 端系统、应用程序和其他时延

除了处理时延、传输时延和传播时延外，端系统中还有一些其他重要的时延。例如，拨号调制解调器引入的调制/编码时延，其量级在几十毫秒。（对于以太网、电缆调制解调器和DSL等接入技术，这种调制/编码时延是不太多的，通常可忽略不计。）向共享媒体（例如，在WiFi或以太网情况下）传输分组的端系统可以将有意地延迟传输作为其协议的一部分，以便与其他端系统共享媒体；我们将在第5章中详细地讨论这样的一些协议。另一个重要的时延是媒体分组化时延，这出现在IP话音（VoIP）应用中。在VoIP中，发送方在向因特网传递分组之前必须首先用编码的数字化语音填充分组。这种填充分组的时间就是分组化时延，它可能较大，从而影响用户感受到的VoIP呼叫的质量。这个问题将在本章末的课后作业中进一步探讨。

1.4.4 计算机网络中的吞吐量

除了时延和丢包外，计算机网络中另一个必不可少的性能测度是端到端吞吐量。为了定义吞吐量，考虑从主机A到主机B通过计算机网络传送一个大文件。例如，也许是从一个P2P文件共享系统中的一个对等方向另一个对等方传送一个大视频片段。任何瞬间的**瞬时吞吐量**（instantaneous throughput）是主机B接收到该文件的速率（以bps计）。（许多应用程序，包括许多文件共享系统，在下载期间其用户界面显示了瞬时吞吐量，也许你以前已经注意到了！）如果该文件由F比特组成，而主机B接收到所有F比特用了T秒，则文件传送的**平均吞吐量**（average throughput）是F/T bps。对于某些应用程序（如因特网电话），希望它们具有低时延，并保持高于某阈值的一致的瞬时吞吐量（例如，一般来说，要求因特网电话超过24 kbps，实时视频应用程序超过256 kbps）。对于其他应用程序（包括涉及文件传送的那些应用程序），时延不是至关重要的，但是希望它们具有尽可能高的吞吐量。

为了深入理解吞吐量这个重要概念，我们考虑几个例子。图1-15a显示了由两条通信链路和一台路由器相连的服务器和客户机这两个端系统。考虑从服务器传送一个文件到客户机的吞吐量。令R_s表示服务器与路由器之间的链路速率；R_c表示路由器与客户机之间的链路速率。假定在整个网络中只有从该服务器到该客户机的比特在传送。在这种理想的情况下，该服务器到该客户机的吞吐量是多少？为了回答这个问题，可以将比特看成是流体，将通信链路看成是管道。显然，该服务器不能以快于R_s bps的速率从其链路抽取比特；该路由器也不能以快于R_c bps的速率转发比特。如果$R_s < R_c$，则由该服务器抽取的比特将顺畅地通过路由器“流动”，并以速率R_s bps到达客户机，从而得到了R_s bps的吞吐量。另一方面，如果$R_c < R_s$，则该路由器将不能够以接收的速率来转发比特。在这种情况下，比特将以速率R_c离开该路由器，从而得到端到端吞吐量为R_c。（还要注意，如果比特持续以速率R_s到达该路由器，持续以R_c离开路由器的话，在该路由器中等待传输给客户机的比特将不断增加，这是一种非常不希望的情况！）因此，对于这种简单的两链路网络，其吞吐量是$\min\{R_c, R_s\}$，也就是说，是**瓶颈链路**（bottleneck link）的传输速率。在确定了吞吐量之后，我们现在近似地得到从服务器到客户机传输一个F比特的大文件所需要的时间是$F/\min\{R_c, R_s\}$。具体来说，假定你正在下载一个$F = 32 \times 10^6$比特的MP3文件，服务器的传输速率为$R_s = 2$ Mbps，接入链路的传输速率为$R_c = 1$ Mbps，则传输该文件所需的时间是32 s。当然，这些关于吞吐量和传送时间的表达式仅是近似的，因为它们并没有考虑分组层次和协议的问题。

图1-15b显示了在服务器和客户机之间具有N条链路的网络，这N条链路的传输速率分别是

$R_1, R_2, \cdots, R_N$。采用与两条链路网络相同的分析，可以发现从服务器到客户机传输文件的吞吐量是$\min\{R_1, R_2, \cdots, R_N\}$，仍然是沿着服务器和客户机之间路径的瓶颈链路的速率。

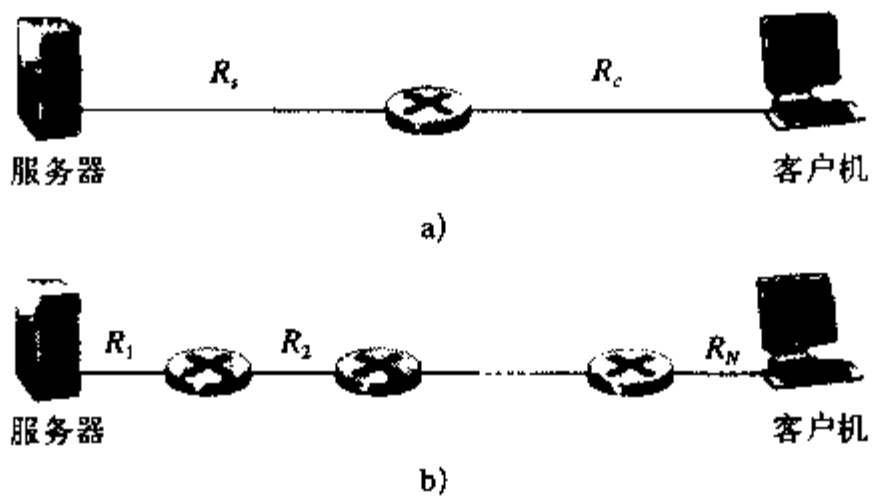
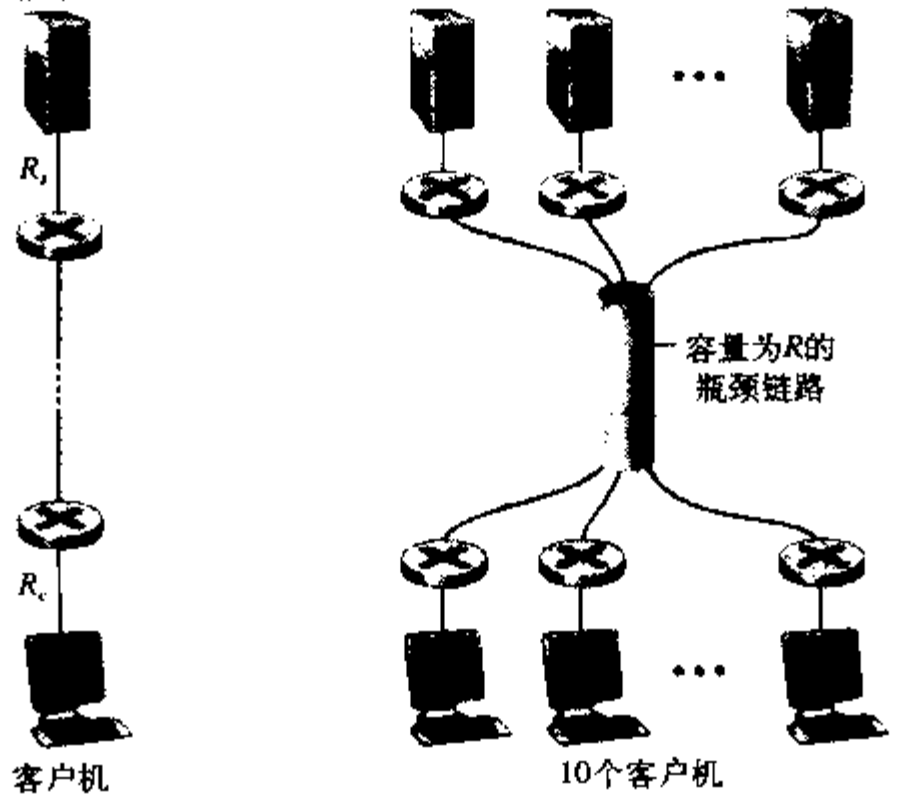


图1-15 从服务器传送一个文件到客户机的吞吐量

现在考虑由当前因特网所引发的另一个例子。图1-16a显示了与计算机网络相连的服务器和客户机这两个端系统。考虑从服务器到客户机传送文件的吞吐量。服务器以接入速率R_s与网络相连，客户机以接入速率R_c与网络相连。现在假定计算机网络核心中的所有链路具有非常高的传输速率，远远超过R_s和R_c。实际上，目前因特网的核心装备了非常高速率的链路，所以很少出现拥塞[Akella 2003]。还假定在整个网络中发送的比特都是从该服务器到该客户机。在这个例子中，因为计算机网络的核心就像一个宽大的管子，所以比特从源向目的地的流动速率还是R_s和R_c中的较小者，即$\min\{R_s, R_c\}$。目前，因特网中对吞吐量的限制因素通常是接入网。



a）客户机从服务器下载文件　　b）10个客户机从10个服务器下载文件

图1-16 端到端吞吐量

作为最后一个例子，考虑图1-16b，其中有10个服务器和10个客户机与计算机网络核心相连。在这个例子中，同时有10个下载，涉及10个客户机/服务器对。假定这10个下载是网络中的唯一流量。如该图所示，在核心中有一条所有10个下载通过的链路。假设这条链路R的传输速率为R。再假定所有服务器接入链路的速率都是R_s，所有客户机接入链路的速率都是R_c，并

且核心中除了速率为R的一条公共链路之外，所有其他链路的传输速率都比R_s、R_c和R大得多。这种下载的吞吐量是多少呢？显然，如果公共链路的速率R很大，比如说比R_s和R_c大100倍，则每个下载的吞吐量将仍然是$\min\{R_s, R_c\}$。但是，如果公共链路的速率与R_s和R_c量级相同会怎样呢？在这种情形下，吞吐量将是多少呢？具体来说，假定R_s = 2 Mbps，R_c = 1 Mbps，R = 5 Mbps，并且公共链路对10个下载平分其传输速率。这时，每个下载的瓶颈不再是接入网了，而是核心中的公共链路了，公共链路仅能为每个下载提供500 kbps的吞吐量。因此，每个下载的端到端吞吐量现在减少到500 kbps。

图1-15和图1-16a中的例子说明吞吐量取决于数据流过的链路的传输速率。可以看到当没有其他干扰流量时，吞吐量近似为沿着源和目的地之间路径的最小传输速率。图1-16b中的例子更一般地说明了吞吐量不仅取决于沿着路径的传输速率，而且取决于干扰流量。特别是，如果许多数据流都通过一条链路流动，那么即使这条链路具有高传输速率，也可能成为文件传输的瓶颈链路。我们将在本章课后习题和后继章节中更仔细地研究计算机网络中的吞吐量。

1.5 协议层次和它们的服务模型

从我们目前的讨论来看，因特网是一个极为复杂的系统。我们已经看到，因特网有许多部分：大量的应用程序和协议、各种类型的端系统、分组交换机和各种类型的链路级媒体。对于这种巨大的复杂性，存在着组织网络体系结构的希望吗？或者至少存在着我们对网络体系结构进行讨论的希望吗？幸运的是，对这两个问题的回答都是肯定的。

1.5.1 分层的体系结构

在我们试图构建因特网体系结构之前，先看一个人类社会中与之类比的例子。实际上，在日常生活中我们一直都在处理复杂系统。想象一下有人请你描述航线系统的情况吧。你怎样用一个结构来描述这样一个复杂的系统呢？该系统具有票务代理、行李检查、登机口人员、飞行员、飞机、空中航行控制和世界范围的导航系统。描述这种系统的一种方式是，描述你乘某个航班时，你（或其他人为你）将采取的一系列动作。你要购买机票，托运行李，寻找登机口，并最终登上这次航班。该飞机起飞，飞行到目的地。当飞机着陆后，你从登机口离机并认领行李。如果这次行程不理想，你向票务机构投诉这次航班（你的努力一无所获）。图1-17显示了相关情况。

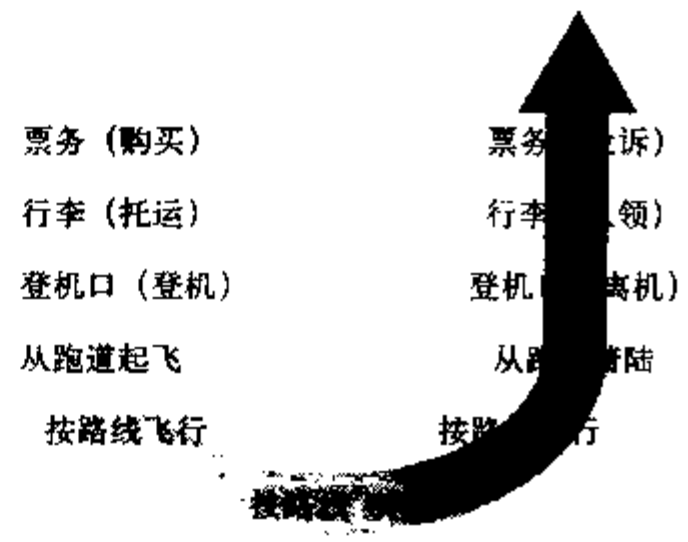


图1-17 乘飞机旅行的一系列动作

我们已经能从这里看出了与计算机网络的类似：航空公司把你从源送到目的地；分组被从因特网中的源主机送到目的主机。但这不是我们要找的理想类似。我们看看图1-17中的某些结构。观察图1-17，我们注意到每层都有票务功能；还对已经检票的乘客有行李功能，对已经检票并已经检查过行李的乘客有登机口功能。对于那些已经通过登机口的乘客（即已经经过检票、行李检查和通过登机的乘客）有起飞和着陆的功能，并且在飞行中有飞机按预定路线飞行的功能。这提示我们，可以以水平的方式看待这些功能，如图1-18所示。

图1-18将航线功能划分为一些层次，为我们提供了讨论航线旅行的框架。注意到每个层

次与其下面的层次结合在一起，实现了某些功能、服务。在票务层及以下，完成一个人从航线柜台到航线柜台的转移。在行李层及以下，完成某人的行李检查到行李认领和手提行李的转移。注意到行李层仅对已经完成票务的人进行。在登机口层，完成了人手提行李离开登机口到到达登机口的转移。在起飞/着陆层，完成了一个人及其行李的跑道到跑道的转移。每个层次通过以下方式提供了它的服务：①在这层中执行了某些动作（例如，在登机口层，某航线乘客的登机和离机）；②使用直接下层的服务（例如，在登机口层，使用起飞/着陆层的跑道到跑道的旅客转移服务）。

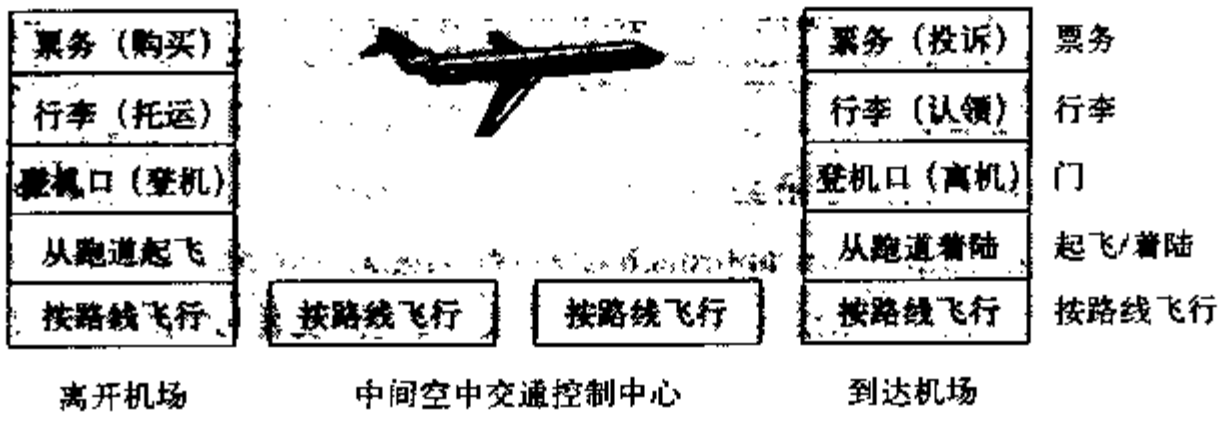


图1-18 航线功能的水平分层

利用分层的体系结构，我们可以讨论一个定义良好的、大而复杂的系统的特定部分。这种简化本身由于提供模块化而具有很高价值，这使得由层所提供的服务的实现易于改变。只要该层对其上面的层提供相同的服务，并且使用来自下面层次的相同服务，当某层的实现变化时，该系统的其余部分就可以保持不变。（注意，一个服务改变实现方式与改变服务本身是极为不同的！）例如，如果登机口功能被改变了（例如，让人们按身高登机和离机），航线的其余部分将保持不变，因为登机口仍然提供相同的功能（人们登机和离机）；改变后，它仅是以不同的方式实现了该功能。对于大而复杂且需要不断更新的系统，改变服务的实现而不影响该系统其他部分的能力是分层的另一个重要优点。

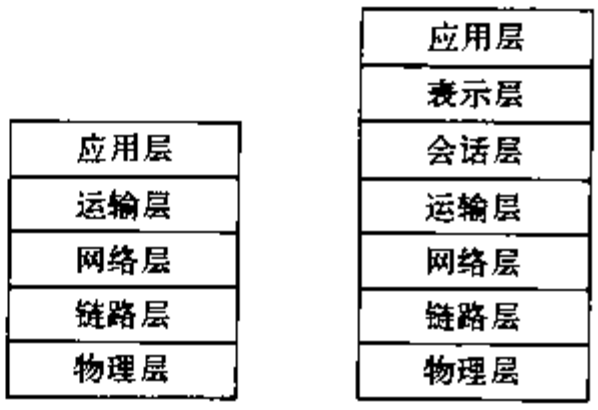
协议分层

我们对航线已经进行了充分的讨论，现在将注意力转向网络协议。为了给网络协议的设计提供一个结构，网络设计者以**分层**（layer）的方式组织协议以及实现这些协议的网络硬件和软件。每个协议属于一层，就像图1-18所示的航线体系结构中每种功能属于某一层一样。我们再次关注某层向其上一层提供的**服务**（service），即所谓的层的**服务模型**（service model）。就像前面航线例子中的情况一样，每层通过在该层中执行某些动作，或使用直接下层的服务，来提供它的服务。例如，层n提供的服务可能包括报文从网络的边缘到另一边缘的可靠传送。这可能是通过使用层$n-1$的“边缘到边缘”的不可靠报文传送服务，加上层n的检测和重传丢失报文的功能来实现的。

一个协议层能够用软件、硬件或两者的结合来实现。HTTP和SMTP等应用层协议通常都是在端系统中用软件实现的，运输层协议也是如此。因为物理层和数据链路层负责处理跨特定链路的通信，它们通常在与给定链路相关的网络接口卡（例如以太网或WiFi接口卡）中实现。网络层经常是硬件和软件的混合体。还要注意到，在分层的航线体系结构中，功能分布在构成该系统的各机场和飞行控制中心，与此相同，层n协议也分布在构成该网络的端系统、分组交换机和其他组件中。这就是说，层n协议的不同部分常常位于这些网络组件的各部分中。

协议分层具有概念化和结构化的优点。正如我们所见，分层提供了一种结构化方式来讨论系统组件。模块化使得更新系统组件更为容易。然而，需要提及的是，某些研究人员和网络工程师激烈反对分层[Wakeman 1992]。分层的一个潜在缺点是某层可能重复其较低层的功能。例如，许多协议栈都基于链路和端到端这两种情况提供了差错恢复。第二种潜在的缺点是某层的功能可能需要仅在其他某层才出现的信息（如时间戳值），这违反了层次分离的目标。

将这些综合起来，各层的所有协议被称为**协议栈**（protocol stack）。因特网的协议栈由5个层次组成：物理层、链路层、网络层、运输层和应用层（如图1-19a所示）。从本书目录中，可以发现我们大致是按因特网协议栈的层次来组织本书的。我们采用了**自顶向下方法**（top-down approach），首先讨论应用层，然后依次向下讨论。



a）五层因特网协议栈 b）七层ISO OSI参考模型

图1-19 因特网协议栈和OSI参考模型

1. 应用层

应用层是网络应用程序及其应用层协议存留的地方。因特网的应用层包括许多协议，例如HTTP（它为Web文档提供了请求和传送）、SMTP（它提供了电子邮件报文的传输）和FTP（它提供了两个端系统之间的文件传送）。我们将看到，某些网络功能，如将像www.ietf.org这样的对人友好的端系统名字转换为32比特网络地址，也是借助于应用层协议——域名系统（DNS）完成的。我们将在第2章中看到，创建并部署自己的新应用层协议是非常容易的。

应用层协议分布在多个端系统上，一个端系统中的应用程序使用协议与另一个端系统中的应用程序交换信息分组。我们将这种位于应用层的信息分组称为**报文**（message）。

2. 运输层

运输层提供了在应用程序端点之间传送应用层报文的服务。在因特网中，有两个运输层协议，即TCP和UDP，利用其中的任何一个都能传输应用层报文。TCP向它的应用程序提供了面向连接的服务。这种服务包括了应用层报文向目的地的确保传递和流量控制（即发送方/接收方速率匹配）。TCP也将长报文划分为短报文，并提供拥塞控制机制，因此当网络拥塞时，源抑制其传输速率。UDP协议向它的应用程序提供无连接服务。这是一种不提供不必要服务的服务，不提供可靠性，没有流量控制，也没有拥塞控制。在本书中，我们将运输层分组称为**报文段**（segment）。

3. 网络层

因特网的网络层负责将称为**数据报**（datagram）的网络层分组从一台主机移动到另一台主机。源主机中的因特网运输层协议（TCP或UDP）向网络层递交运输层报文段和目的地址，就像你向邮政信件提供目的地址一样。

因特网的网络层包括著名的IP协议，该协议定义了数据报中的各个字段以及端系统和路由器如何作用于这些字段。仅有一个IP协议，所有具有网络层的因特网组件都必须运行IP协议。因特网的网络层也包括决定路由的选路协议，数据报根据该路由从源传输到目的地。因特网具有许多选路协议。如我们在1.3节所见，因特网是一个网络的网络，在一个网络中，其网络管理者能够运行所希望的任何选路协议。尽管网络层包括了IP协议和一些选路协议，但它经常只被称为IP层，这反映了IP是将因特网连接在一起的粘合剂这样一个事实。

4. 链路层

因特网的网络层通过一系列路由器在源和目的地之间发送分组。为了将分组从一个节点（主机或路由器）移动到路径上的下一个节点，网络层必须依靠链路层的服务。特别是在每个节点，网络层将数据报下传给链路层，链路层沿着路径将数据报传递给下一个节点。在该下个节点，链路层将数据报上传给网络层。

链路层提供的服务取决于应用于该链路的特定链路层协议。例如，某些协议基于链路提供可靠传递，即从传输节点跨越一条链路到接收节点。注意，这种可靠传递服务不同于TCP的可靠传递服务，TCP是为从一个端系统到另一个端系统提供可靠传递。链路层的例子包括以太网、WiFi和点对点协议（PPP）。因为数据报从源到目的地传送通常需要经过几条链路，所以它可能被沿途不同链路上的不同链路层协议处理。例如，某个数据报可能被一条链路上的以太网和下一条链路上的PPP所处理。网络层将接收来自每个不同的链路层协议的不同服务。在本书中，我们将链路层分组称为**帧**（frame）。

5. 物理层

链路层的任务是将整个帧从一个网络元素移动到邻近的网络元素，而物理层的任务是将该帧中的一个一个比特从一个节点移动到下一个节点。该层中的协议仍然是链路相关的，并且进一步与链路（例如，双绞铜线、单模光纤）的实际传输媒体相关。例如，以太网具有许多物理层协议：关于双绞铜线的，关于同轴电缆的，关于光纤的，等等。在每种情况下，跨越这些链路移动一个比特的方式不同。

6. ISO模型

讨论过因特网协议栈后，我们应当知道它不是唯一的协议栈。特别是在20世纪70年代后期，国际标准化组织（ISO）提出计算机网络应组织为大约七层，称为开放系统互连（OSI）模型[ISO 2007]。当将OSI模型具体化时，就要使之成为因特网协议，而这些协议仍处于襁褓之中，是许多研发中的不同协议栈之一；事实上，初始发明者在创建OSI模型时，并没有考虑到因特网。但是，自20世纪70年代后期开始，许多培训课程和大学课程围绕七层模型挑选主题。由于受到早期网络教育的影响，某些网络教科书和培训课程中仍有七层模型的身影。

图1-19b中OSI参考模型的七层是：应用层、表示层、会话层、运输层、网络层、链路层和物理层。其中与因特网协议栈的五层名字相同的层，其功能也基本对应相同。所以，我们考虑OSI参考模型中另外的两层，即表示层和会话层。表示层的作用是使通信的应用程序能够解释交换数据的含义，它所提供的服务包括数据压缩、数据加密（它们是自解释的）以及数据描述（如我们将在第9章所见，这使得应用程序不必担心在不同的计算机中数据的表示/存储的内部格式不同的问题）。会话层提供了数据交换的定界和同步功能，包括建立检查点和恢复方案的方法。

因特网中没有OSI参考模型中建立的这两层，这引起了一些有趣的问题：这两层提供的服务不重要吗？如果某应用程序需要这些服务之一，将会怎样呢？因特网对这两个问题的回答是相同的：这留给应用程序开发者处理。因此，应用程序开发者应决定这样的服务是否重要，如果重要，就应该在应用程序中构建该功能。

1.5.2 报文、报文段、数据报和帧

图1-20显示了这样一条物理路径：数据从发送端系统的协议栈向下，上下中间的链路层交换机和路由器的协议栈，进而向上到达接收端系统的协议栈。在本书后面将讨论，路由器和

链路层交换机都是分组交换机。与端系统类似，路由器和链路层交换机以层的方式组织它们的网络硬件和软件。而路由器和链路层交换机并不实现协议栈中的所有层次。如图1-20所示，链路层交换机实现了第一层和第二层；路由器实现了第一层到第三层。例如，这意味着因特网路由器能够实现IP协议（一种第三层协议），而链路层交换机则不能。我们将在后面看到，尽管链路层交换机不能识别IP地址，但它们能够识别第二层地址，如以太网地址。可以看到主机实现了所有5个层次，这与因特网体系结构将它的复杂性放在网络边缘的观点是一致的。

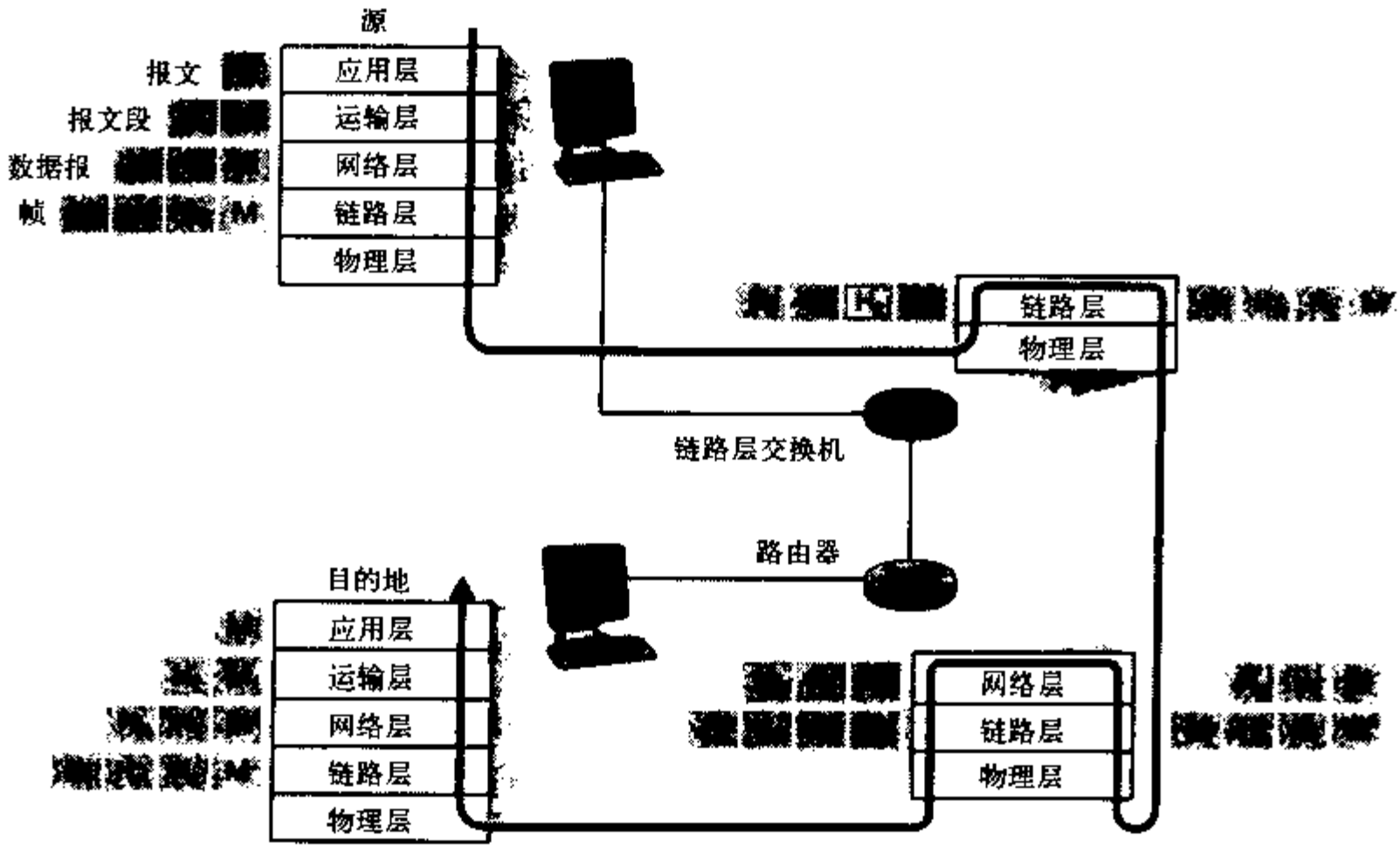


图1-20 主机、路由器和链路层交换机，每个包含了不同的层，反映了不同的功能

图1-20也举例说明了**封装**（encapsulation）这一重要概念。在发送主机，**应用层报文**（application-layer message）（图1-20中的M）被传送给运输层。在最简单的情况下，运输层收取报文并附上附加信息（即运输层首部信息，图1-20中的H_t），该首部将被接收端的运输层使用。应用层报文和运输层首部信息共同构成了**运输层报文段**（transport-layer segment）。运输层报文段因此封装了应用层报文。附加的信息可能包括下列信息，如允许接收端运输层向上向适当的应用程序交付报文的信息；差错检测比特信息，利用该信息接收方能够判断报文中的比特是否在途中已被改变。运输层则向网络层传递该报文段，网络层增加了如源和目的端系统地址等网络层首部信息（图1-20中的H_n），形成了**网络层数据报**（network-layer datagram）。该数据报接下来被传递给链路层，链路层当然也增加它自己的链路层首部信息并创建了**链路层帧**（link-layer frame）。于是，我们看到在每一层，分组具有两种类型的字段：首部字段和**有效载荷字段**（payload field）。有效载荷通常来自上一层的分组。

这里有一个有用的类比，即通过公共邮政服务发送一封办公室之间的备忘录。假定一个分支机构办公室的Alice要向另一个分支机构办公室的Bob发送一封备忘录。备忘录类比于应用层报文。备忘录被放入办公室间的公函信封中，并在公函信封上方写上了Bob的名字和部门。办公室间的公函信封类比于运输层报文段，即它包括了首部信息（Bob的名字和部门）并封装了应用层报文（备忘录）。发送分支机构办公室的收发室拿到该办公室间的备忘录，将其放入适

合在公共邮政服务发送的信封中，在邮政信封上写上发送和接收分支机构办公室的邮政地址。此处，邮政信封类比于数据报，它封装了运输层的报文段（办公室间的公函信封），该报文段封装了初始报文（备忘录）。邮局将该邮政信封交付给接收分支机构办公室的收发室，在此处信封被打开，得到办公室间的公函信封并转给Bob。最后，Bob打开该信封并得到该备忘录。

封装的过程能够比上面描述的更为复杂。例如，一个大报文可以被划分为多个运输层报文段，而这些报文段自身又可能被划分为多个网络层数据报。在接收端，这些数据报则必须要重新还原成报文段。

1.6 攻击威胁下的网络

目前，对于许多机构（包括大大小小的公司、大学和政府机关）而言，因特网已经成为不可或缺的一部分。同时，许多人也依赖因特网进行许多职业的、社会的和个人的活动。但是在这一切的背后，有一些“坏家伙”（攻击者）试图侵扰我们的日常生活，如损坏我们与因特网相连的计算机，侵犯我们的隐私以及使我们使用的因特网服务无法运行[Skoudis 2006]。

网络安全领域主要探讨以下问题：坏家伙如何攻击计算机网络，以及我们（计算机网络专家）如何防御网络免受攻击，或者最好是事先设计免除攻击的新型体系结构。面对经常发生的、各种各样的现有攻击以及新型、更具摧毁性的未来攻击的威胁，网络安全已经成为近年来计算机网络领域的中心主题。本版的特色之一就是将网络安全问题放在首要位置。在本节中，我们开始网络安全的开创性尝试，简要地描述在今天的因特网中较常见和危害较大的某些攻击。然后，在后继章节中详细地学习各种计算机网络技术和协议时，再探讨与这些技术和协议相关的各种安全性问题。最后，在第8章中，根据我们学习的计算机网络和因特网协议方面的专门技术，我们将深入学习在计算机网络中防御攻击的方法，或者事先设计免除攻击的体系结构的方法。

因为我们现在还不具备计算机网络和因特网协议方面的专业知识，所以这里我们将纵观较流行的安全相关问题，以便在后续章节中进行更为充实的讨论。这里，先提出以下问题：哪些东西会出现问题？计算机网络是怎样受到攻击的？今天流行的攻击类型是什么？

1. 坏家伙能够经因特网将恶意软件放入你的计算机

为了从因特网接收数据或向因特网发送数据，我们将设备与因特网相连。接收或发送的数据包括各种好的东西，例如Web页面、电子邮件报文、MP3、电话、视频实况、搜索引擎结果等。但是，随之而来的还有不好的东西（这些不好的东西统称为**恶意软件**（malware）），它们能够影响我们的设备。一旦恶意软件感染了我们的设备，它就能够做各种不正当的事情，包括：删除文件；安装间谍软件来收集隐私信息，如社会保险号、口令和按键，然后将这些信息经因特网发送给坏家伙。受害主机还可能征召网络上数以千计的类似受害设备，它们被统称为**僵尸网络**（botnet），坏家伙可以控制僵尸网络，有效地对目标主机展开垃圾邮件分发或分布式拒绝服务攻击（很快将讨论）。

今天，大多数恶意软件是**自我复制**（self-replicating）的：一旦它感染了一台主机，就会从那台主机进入更多的主机。按照这种方式，自我复制的恶意软件能够以指数级快速扩散。例如，2003 Saphire/Slammer蠕虫感染的设备数量在它爆发后的前几分钟内每8.5秒翻一番，在10分钟内感染了90%易受攻击的主机[Moore 2003]。恶意软件能够以病毒、蠕虫或特洛伊木马的形式扩散[Skoudis 2004]。**病毒**（virus）是一种需要某种形式的用户交互来感染用户设备的恶意软件。典型的例子是包含恶意可执行代码的电子邮件附件。如果用户接收并打开这样的附件，

就在不经意中运行了该恶意软件。通常，这种电子邮件病毒是自我复制的：例如，一旦执行，该病毒就可能向用户地址簿上的每个接收方发送一个带有相同恶意附件的相同报文。**蠕虫**（worm）是一种无需任何明显用户交互就能进入设备的恶意软件，如Slammer蠕虫。例如，用户也许运行了一个攻击者能够发送恶意软件的网络应用程序。在某些情况下，无需用户的任何干预，该应用程序就可能从因特网接收恶意软件并运行它，从而生成蠕虫。新近感染设备中的蠕虫能扫描因特网，搜索其他运行相同易受感染的网络应用程序的主机。当它发现其他易受感染的主机时，它向其他主机发送一个它自身的拷贝。**特洛伊木马**（Trojan horse）是隐藏在有用软件中的恶意软件。今天，恶意软件无所不在，造成的损失惨重。例如，据估计在2005年仅病毒对金融的影响就超过140亿美元[Malware 2006]。当你学习这本书时，建议你思考下列问题：计算机网络设计者能够做些什么来防御与因特网连接的设备免受恶意软件的攻击？

2. 坏家伙能够攻击服务器和网络基础设施

拒绝服务（Denial-of-Service, DoS）**攻击**是一种宽泛类型的安全性威胁。顾名思义，DoS攻击使得合法用户不能使用网络、主机或其他基础设施部分。Web服务器、电子邮件服务器、DNS服务器（在第2章中讨论）和机构网络都可以成为DoS攻击的目标。因特网DoS攻击极为常见，每年会出现数以千计的DoS攻击[Moore 2001; Mirkovic 2005]。大多数因特网DoS攻击属于下列三种类型之一：

- *弱点攻击*。这涉及向目标主机上运行的易受攻击的应用程序或操作系统发送制作精细的报文。如果多个分组以适当的顺序发送给一个易受攻击的应用程序或操作系统，该服务可能停止运行，甚至导致主机崩溃。
- *带宽洪泛*。攻击者向目的主机发送大量的分组，导致目标的接入链路变得拥塞，从而使合法的分组无法到达服务器。
- *连接洪泛*。攻击者在目标主机中创建大量的半开或全开TCP连接（TCP连接将在第3章中讨论）。目标主机因这些伪造的连接而陷入困境，从而停止合法的连接。

我们现在更详细地研究带宽洪泛攻击。在1.4.2节中我们讨论过时延和丢包问题，显然，如果服务器的接入速率为R bps，则攻击者需要以大约R bps的速率发送数据才能产生危害。如果R非常大的话，单一攻击源可能无法产生足够大的流量来危害服务器。此外，如果从单一源发出所有流量的话，上游路由器可以检测出该攻击并在该流量靠近服务器前就将其阻挡下来。在图1-21显示的**分布式DoS**（distributed DoS, DDoS）中，攻击者控制多个源并让每个源向目标猛烈发送流量。使用这种方法，为了损坏服务器，所有受控源的流量加起来需要产生大约R的能力。DDoS攻击充分利用数以千计的受害主机组成的僵尸网络在今天是屡见不鲜的[Mirkovic 2005]。与来自单一主机的DoS攻击相比，DDoS攻击更加难以检测和防御。

当你学习这本书时，建议你思考下列问题：计算机网络设计者可以采取哪些措施来防止DoS攻击？我们将看到对于3种不同类型的DoS攻击需要不同的防御方法。

3. 坏家伙能够嗅探分组

今天，许多用户经无线设备（例如，WiFi连接的膝上机或使用蜂窝因特网连接的手持设备，将在第6章中讨论）接入因特网。当无所不在的因特网接入极为便利且令人惊奇的新应用程序为移动用户所用时，也产生了重大的安全弱点，即在无线传输设备的附近放置一台被动的接收机，该接收机就能得到传输的每个分组的拷贝！这些分组包含各种敏感信息，包括口令、社会保险号、商业秘密和隐秘的个人信息。记录每个流经的分组拷贝的被动接收机被称为**分组嗅探器**（packet sniffer）。

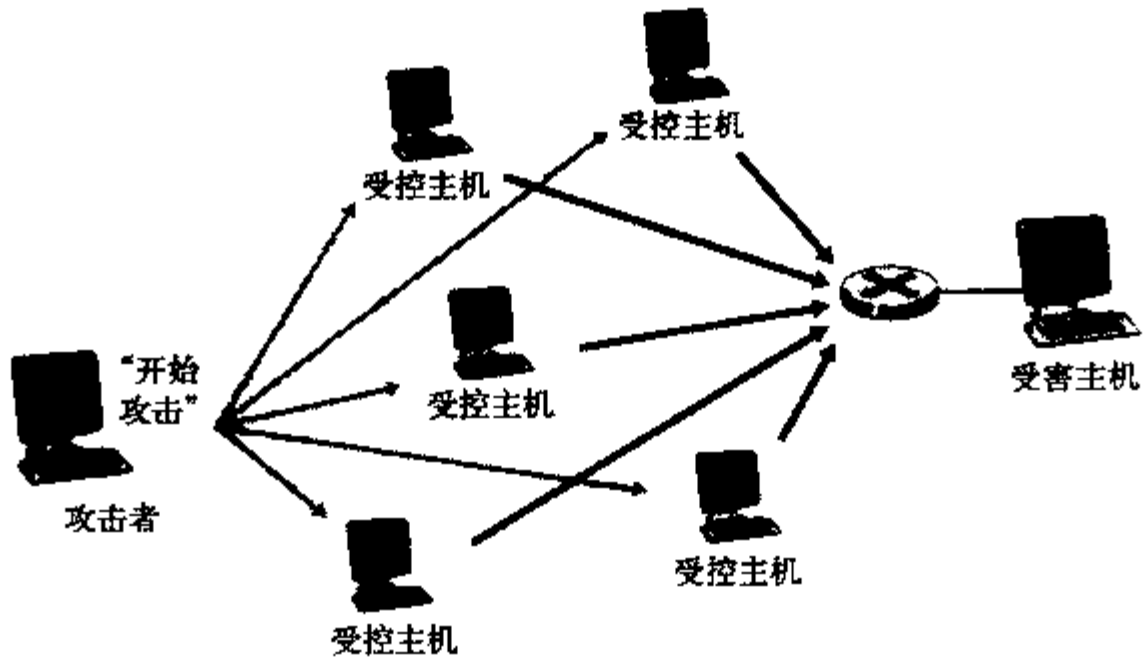


图1-21 分布式拒绝服务攻击

嗅探器也可以用于有线的环境中。在有线广播中，如在许多以太网LAN中，分组嗅探器能够获得经该LAN发送的所有分组。如1.2节中描述的那样，电缆接入技术也广播分组，因此易于受到嗅探攻击。此外，获得某机构与因特网连接的接入路由器或接入链路访问权的坏家伙能够放置一台嗅探器来得到从该机构出入的每个分组的拷贝，再对嗅探到的分组进行离线分析，就可以得出敏感信息。

在各种Web站点上可免费得到分组嗅探软件，也可通过购买获得。网络课程的实验作业就包括写一个分组嗅探器和应用层数据重构程序。实际上，本章末的Ethereal[Ethereal 2007]实验使用的正是一种分组嗅探器！

因为分组嗅探器是被动的，即它们不向信道中注入分组，从而难以检测它们的存在，所以当我们向无线信道发送分组时，必须知道某些坏家伙可能记录了我们分组的拷贝。如你所猜想的那样，最好的防御嗅探的方法基本上都与密码学有关，第8章中将涉及这方面的内容。

4. 坏家伙能够伪装成你信任的人

生成具有任意源地址、分组内容和目的地址的分组，然后将这个人工制作的分组传输到因特网中，这是非常容易的（当你学完这本教科书后，你就会具有这方面的能力！），当然因特网将忠实地将该分组转发到目的地。想象一下，一个不可信的接收方（比如说因特网上的一台路由器）接收了分组，用（虚假的）源地址伪装真实的源地址，进而执行某些嵌入在该分组中的命令（比如说修改它的转发表）。将具有虚假源地址的分组注入因特网的能力被称为IP哄骗（IP spoofing），它只是一个用户能够冒充另一个用户的多种方式之一。

为了解决这个问题，需要采用端点鉴别（end-point authentication）机制，即确保报文源自我们认为它应当来自的地方的机制。当你继续学习本书各章时，建议你思考一下怎样为网络应用程序和协议做这件事。我们将在第8章研究端点鉴别机制。

5. 坏家伙能够修改或删除报文

我们通过描述中间人攻击（man-in-middle attack）来终止这个有关网络攻击的概述。在这类攻击中，坏家伙插入到两个通信实体之间的通信路径中。我们不妨将通信实体称为Alice和Bob，Alice和Bob可能是实际的人，也可能是网络实体（如两台路由器或两台电子邮件服务器）。例如，坏家伙可能是在通信路径中受害的路由器，也可能是驻留在协议栈较低层的一台端主机中的一个软件模块。在中间人攻击中，坏家伙不仅能嗅探所有在Bob和Alice之间传递

的分组，而且能够危及Alice和Bob之间发送的数据的完整性。如我们将在第8章所见，机密性（防止嗅探）和端点鉴别（使接收方确保报文的源地址）机制并不能保证数据完整性。因此，还需要另一套技术来提供数据完整性。

在本节结束时，有必要思考一下因特网是如何从一开始就陷入不安全的境地的。大体上讲，答案是因特网最初就是基于“一群相互信任的用户连接到一个透明的网络上”这样的模型[Blumenthal 2001]进行设计的，在这样的模型中，安全性是没有必要的。初始的因特网体系结构在许多方面都深刻地反映了这种相互信任的观念。例如，一个用户向任何其他用户发送分组的能力是默认的，而不是一种请求/准予的能力。此外，在默认情况下，用户身份也不需要鉴别。

但是，今天的因特网并不涉及“相互信任的用户”。而当今天的用户无法相互信任时，他们仍然需要通信，可能是匿名通信，可能是间接地通过第三方通信（例如，我们将在第2章中学习的Web缓存，将在第6章学习的移动帮助代理），也可能不信任通信的硬件、软件甚至经过的天空。在进一步学习本书之前，已经有许多安全性相关的挑战：我们应当对抗嗅探、端点假冒、中间人攻击、DDoS攻击、恶意软件等。一定要牢记：相互信任的用户之间的通信是一种例外。欢迎来到现代计算机网络世界！

1.7 计算机网络和因特网的历史

1.1节到1.6节概述了计算机网络和因特网的技术。你现在应当有足够多的知识来给家人和朋友留下深刻印象了。然而，如果你真的想在下次鸡尾酒会上一鸣惊人，你应当在你的演讲中插进一些有关因特网引人入胜的历史轶闻[Segaller 1998]。

1.7.1 分组交换的发展：1961～1972

计算机网络和因特网的开端可以追溯到20世纪60年代早期，那时电话交换网是世界上占统治地位的通信网络。1.3节讲过，电话网使用电路交换将信息从发送方传输到接收方，这种适当的选择使得话音以一种恒定的速率在发送方和接收方之间传输。随着20世纪60年代早期计算机的重要性（和极高的成本）的提高，以及分时计算机的出现，考虑如何将计算机连接在一起，并使它们能够被地理上分散的用户所共享的问题，也许就成了一件自然的事（至少是事后诸葛亮！）。这些用户所产生的流量很可能具有突发性，即活动的间断性，例如向远程计算机发送一个命令，接着由于在等待回答或在对接收到的响应进行思考而有静止的时间段。

全世界有3个研究组首先发明了分组交换的概念，作为一种有效的、健壮的电路交换的替代技术。这3个研究组彼此并不知道其他人的工作[Leiner 1998]。有关分组交换技术的首次公开发表是由Leonard Kleinrock完成的[Kleinrock 1961；Kleinrock 1964]，那时他是麻省理工学院（MIT）的一名研究生。Kleinrock使用排队论完美地体现了使用分组交换方法处理突发性流量源的有效性。1964年，兰德公司的Paul Baran [Baran 1964]已经开始研究分组交换的应用，即在军用网络上传输安全语音，同时在英国的国家物理实验室（NPL），Donald Davies和Roger Scantlebury也在研发分组交换技术。

在MIT、兰德和NPL所进行的工作奠定了今天的因特网的基础。但是因特网也有很长的一种“让我们边构建边示范”（let's-build-it-and-demonstrate-it）态度的历史，这可追溯到20世纪60年代早期。J. C. R. Licklider [DEC 1990]和Lawrence Roberts都是Kleinrock在MIT的同事，他们转而去领导美国高级研究计划署（Adveanced Research Projects Agency, ARPA）的计算

机科学计划。Roberts公布了一个号称ARPAnet[Roberts 1967]的总体计划，它是第一个分组交换计算机网络，是今天的公共因特网的直接祖先。早期的分组交换机被称为**接口报文处理机**(Interface Message Processor, IMP)，BBN公司得到了建造这些交换机的合同。1969年的国际劳动节，第一台IMP在Kleinrock的监管下安装在UCLA（美国加州大学洛杉矶分校），其他3台IMP不久后安装在斯坦福研究院（Stanford Research Institute，SRI）、美国加州大学圣芭芭拉分校（UC Santa Barbara）和犹他大学（University of Utah），参见图1-22。羽翼未丰的因特网祖先到1969年底有了4个节点。Kleinrock回忆说，该网络的最先应用是从UCLA到SRI执行远程注册，但却导致了该系统的崩溃[Kleinrock 2004]。

图1-22 L. Kleinrock与一台早期的接口报文处理机（Mark J. Terrill AP/Wide World Photos）

到了1972年，ARPAnet已经成长为约15个节点，Robert Kahn于1972年的计算机通信国际会议上首次对它进行了公众演示。第一个在ARPAnet端系统之间的主机到主机协议，被称为网络控制协议（NCP），就是此时实现的[RFC 001]。随着端到端协议的可供使用，这时能够写应用程序了。第一个电子邮件程序由BBN的Ray Tomlinson于1972年所写。

1.7.2 专用网络和网络互联：1972～1980

最初的ARPAnet是一个单一的、封闭的网络。为了与一台ARPAnet的主机通信，一台主机必须与另一台ARPAnet IMP实际相连。20世纪70年代早期和中期，除ARPAnet之外的其他分组交换网络问世：

- ALOHAnet是一个微波网络，它将夏威夷岛上的大学[Abramson 1970]以及DARPA的分组卫星[RFC 829]和分组无线电网 [Kahn 1987]连接到一起。

- Telenet是BBN的商用分组交换网，基于ARPAnet技术。
- Cyclades是一个法国的分组交换网，它由Louis Pouzin所倡导[Think 2007]。
- 如Tymnet和GE信息服务网这样的分时网络，以及20世纪60年代后期和70年代初期的类似网络[Schwartz 1977]。
- IBM的SNA（1969～1974），它与ARPAnet并行工作[Schwartz 1977]。

网络的数目开始增加。人们看到（当然又是事后诸葛亮），研制将网络连接到一起的体系结构的时机已经成熟。互联网络的先驱性工作（得到了美国国防部高级研究计划署（DARPA）的支持），本质上就是创建一个网络的网络，由Vinton Cerf和Robert Kahn [Cerf 1974]完成；术语网络互联（internetting）就是用来描述该项工作的。

这些体系结构的原则被具体表达在TCP协议中。然而，TCP的早期版本与今天的TCP相当不同。TCP的早期版本与数据可靠的顺序传递相结合，经过有转发功能（今天该功能由IP执行）的端系统的重传（今天仍是TCP的一部分）。TCP的早期实验以及认识到对于分组话音等应用程序，一个不可靠的、非流量控制的、端到端传递服务的重要性，导致IP从TCP中分离出来，并研制了UDP协议。我们今天看到的3个重要因特网协议——TCP、UDP和IP，到20世纪70年代末从概念上说已经完成。

除了与因特网相关的DARPA的研究外，许多其他重要的网络活动也在进行中。在夏威夷，Norman Abramson正在研制ALOHAnet，这是一个基于分组的无线电网络，它允许夏威夷岛上的多个远程站点互相通信。ALOHA协议[Abramson 1970]是第一个多路访问协议，它允许地理上分散的用户共享单一的广播通信媒体（一个无线电频率）。Abramson的多路访问协议建立在Metcalfe和Boggs研制的以太网协议[Metcalfe 1976]之上，以太网是有线共享广播网络（参见图1-23）。令人感兴趣的是，Metcalfe和Boggs的以太网协议是由将多台PC、打印机和共享磁盘连接在一起的需求所激励的[Perkins 1994]。在PC革命和网络爆炸的25年之前，Metcalfe和Boggs就奠定了今天PC LAN的基础。以太网技术也代表了网络互联的重要一步。每个以太网局域网自身是一个网络，随着LAN的激增，将这些LAN互联起来的需求变得日益重要。我们将在第5章中详细地讨论以太网、ALOHA和其他LAN技术。

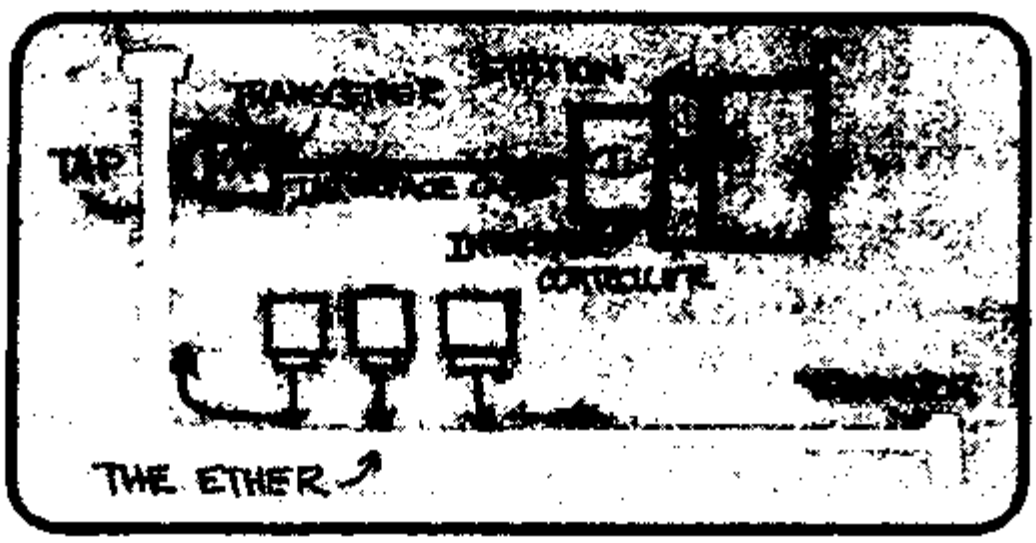


图1-23 Metcalfe的以太网最初概念

1.7.3 网络的激增：1980～1990

到了20世纪70年代末，大约200台主机与ARPAnet相连。到了20世纪80年代，连到公共因特网的主机数量达到100 000台。20世纪80年代是连网的主机数量急剧增长的时期。

这种增长导致了创建将大学连接到一起的计算机网络。BITNET为位于美国东北部的几个

大学之间提供了电子邮件和文件传输服务。建立了CSNET（计算机科学网），以将还没有接入ARPAnet的大学研究人员连接在一起。1986年，建立了NSFNET，以提供对NSF（美国国家科学基金会）资助的超级计算中心的接入。NSFNET最初具有56 kbps的主干速率，到了20世纪80年代末，它的主干运行速率是1.5 Mbps，并成为连接区域网络的基本主干。

在ARPAnet中，许多构成当今因特网体系结构的内容变得清晰起来。1983年1月1日，TCP作为新的用于ARPAnet的标准主机协议正式实施，替代了NCP协议。从NCP到TCP/IP的迁移[RFC 801]是一个重大事件，所有主机都要求在这一天转移到TCP/IP上去。20世纪80年代后期，TCP进行了重要扩展，以实现基于主机的拥塞控制[Jacobson 1988]。研制出了DNS（域名系统），该系统用于把人可读的因特网名字（例如gaia.cs.umass.edu）映射到它的32比特IP地址[RFC 1034]。

20世纪80年代，在ARPAnet（这绝大多数是美国努力的成果）发展的同时，法国开始了Minitel项目，这个雄心勃勃的计划是让数据网络进入每个家庭。在法国政府的支持下，Minitel系统由公共分组交换网络（基于X.25协议集，使用的是虚电路）、Minitel服务器和具有内置低速调制解调器的廉价终端组成。Minitel于1984年取得了巨大的成功，当时法国政府向每个需要的住户免费分发一个Minitel终端。Minitel站点包括免费站点（如电话目录站点）以及一些专用站点。这些专用站点根据每个用户的使用情况来收取费用。在20世纪90年代中期到达其顶峰时，它提供了20 000多种不同的服务，从家庭银行业务到特殊的研究数据库。超过20%的法国人使用该系统，每年产生了超过10亿美元的收入并创造了10 000个工作机会。Minitel在大量法国家庭中存在了10年后，大多数美国人才听说了因特网。

1.7.4 因特网爆炸：20世纪90年代

20世纪90年代出现了许多标志因特网的持续革命和不久后到来的商业化的事件。作为因特网祖先的ARPAnet已不复存在。MILNET和国防数据网在20世纪80年代发展起来，承载了大多数与美国国防部相关的流量，NSFNET已经开始作为连接美国境内的区域网络和海外国家网络的主干网络。1991年，NSFNET放宽了对NSFNET用于商业目的的限制。NSFNET于1995年完成了使命，这时因特网主干流量则由商业因特网服务提供商负责承载。

然而，20世纪90年代的主要事件是万维网（World Wide Web）的出现，它将因特网带入世界上数以百万计的家庭和企业中。作为一个平台，Web也引入并设置了数百个新的应用程序，我们今天对这些应用程序已经习以为常了。Web早期的简史可参见[W3C 1995]。

Web是由Tim Berners-Lee于1989～1991年期间在CERN发明的[Berners-Lee 1989]，它基于20世纪40年代Vannevar Bush [Bush 1945]和自20世纪60年代以来Ted Nelson [Xanadu 2007]在超文本方面的早期工作。Berners-Lee和他的同事研制了HTML、HTTP、Web服务器和浏览器的初始版本，这是Web系统的4个关键部分。到了1992年末，大约有200个Web服务器在运行，而这些只是正在出现的Web服务器的冰山一角。就在这个时候，几个研究人员研制了具有GUI接口的Web浏览器，其中的Marc Andreesen后来领导了流行的GUI浏览器Mosaic的开发。1994年，Marc Andreesen和Jim Clark创办了Mosaic通信公司，该公司后来改名为Netscape通信公司[Cusmano 1998; Quittner 1998]。到了1995年，大学生们每天都使用Mosaic和Netscape浏览器在Web上冲浪。大约在这个时候，大大小小的公司都开始运行Web服务器，并在Web上处理业务。1996年，微软公司开始开发浏览器，这导致了Netscape和微软之间的“浏览器之战”，并以微软公司在几年后获胜而告终[Cusmano 1998]。

20世纪90年代的后5年是因特网飞速增长和变革的时期，伴随着主流公司和数以千计的后

起之秀创造着因特网产品和服务。因特网电子邮件继续演化，提供了功能丰富的电子邮件读取器，提供了地址簿、附件、热链接和多媒体传送。到2000年末，因特网已经支持数百流行的应用程序，包括以下4种招人喜爱的应用程序（killer application）：

- 电子邮件，包括附件和Web可访问的电子邮件。
- Web，包括Web浏览和因特网商务。
- 即时讯息（instant messaging），具有“联系人列表”，由ICQ所倡导。
- MP3的对等（peer-to-peer）文件共享，由Napster所倡导。

值得一提的是，前两个招人喜爱的应用程序出自专业研究机构，而后两个却由一些年轻创业者所发明。

1995～2001年，这段时间也是因特网在金融市场上急转突变的时期。在成为有利可图的公司之前，数以百计的因特网后起之秀靠首次公开发行股票并在股票市场上交易起家。许多公司身价数十亿美元，却没有任何主要的收入渠道。与因特网有关的股票在2000～2001年期间崩盘，导致许多创业公司倒闭。不过，一些公司仍成为因特网世界的大赢家，包括微软、Cisco、AOL、Yahoo、e-Bay、Google和Amazon。

1.7.5 最新发展

计算机网络中的变革是持续不断的。所有前沿研究领域均在取得进展，包括新型应用程序的设置、内容分发、因特网电话、LAN中更高的传输速率和更快的路由器。但有三个进展特别值得关注：高速接入网激增（包括无线接入）、安全和对等（P2P）网络。

如1.2节所讨论的，宽带住宅因特网接入使用的电缆调制解调器和DSL技术正在为大量的新型多媒体应用建造一个舞台，这些多媒体应用包括经IP的音频和视频[Skype 2007]、视频共享[YouTube 2007]和经IP的电视[PPLive 2007]。高速（11Mbps及更高）公共WiFi网络越来越无所不在，蜂窝电话网的中速（几百kbps）因特网接入不仅使得保持持续连接成为可能，也使得启用各种令人激动的新型特定位置服务成为可能。我们将在第6章中讨论无线网络和移动网络。

20世纪90年代后期，一些著名的Web服务器遭到了一系列拒绝服务攻击，之后，感染端系统和用流量阻塞网络的蠕虫攻击（如Blaster蠕虫）大幅度增加，网络安全自此成为一个极为重要的课题。这些攻击导致了入侵检测系统的研制，该系统可对攻击提供早期警告。此外，还使用防火墙来过滤不必要的流量。我们将在第8章中讨论一些与安全相关的重要专题。

我们特别关注的最后一个创新是P2P网络的应用。P2P网络应用充分开采了用户计算机中的资源，如存储、内容、CPU周期和人的存在，并与中心服务器有很大的自治性。用户的计算机（即对等方）通常具有间歇性的连接。在过去的几年里，有许多P2P成功的案例，包括P2P文件共享（Napster、Kazaa、Gnutella、eDonkey、LimeWire等）、文件分发（BitTorrent）、经IP的话音（Skype）和IPTV（PPLive、ppStream）。我们将在第2章中讨论一些P2P应用程序。

1.8 小结

在本章中，我们讨论了大量的材料！我们已经看到各种构成特别的因特网和普通的计算机网络的硬件和软件。我们从网络的边缘开始，观察端系统和应用程序，以及运行在端系统上为应用程序提供的运输服务。接着，我们也观察了接入网中常见的链路层技术和物理媒体。然后我们进入网络核心更深入地钻研网络，指出通过电信网络传输数据的两种基本方法，即

分组交换和电路交换，并且探讨了每种方法的长处和短处。我们也研究了全球性因特网的结构，知道了因特网是网络的网络。我们看到因特网的由较高层和较低层ISP组成的层次结构，允许该网络扩展为包括数以千计的网络。

在本章的后半部分，我们研究了计算机网络领域的几个重要主题。我们首先研究了分组交换网中的时延、吞吐量和丢包的原因。我们研究了传输、传播和排队时延以及用于吞吐量的简单定量模型，我们将在本书的课后习题中广泛使用这些时延模型。接下来，我们研究了协议分层和服务模型、网络中的关键体系结构原则，书中多处会用到它们。我们还概述了当今因特网中一些流行的安全攻击。我们用计算机网络的简要历史结束我们对网络的概述。第1章本身就构成了计算机网络的小型课程。

因此，第1章中的确包括了大量的背景知识！如果你有些不知所云，请不要着急。在后继几章中我们将重新回顾这些概念，更为详细地研究它们。此时，我们希望读者学完本章内容时，对构建一个网络的许多东西有正在发展的直觉，深知网络词汇正在发展（应经常回过头来查阅本章），更多地学习网络的愿望与日俱增。这本书的其余部分是我们面临的任务。

本书路线图

在开始任何旅行之前，我们总要先察看路线图，以便更熟悉前面的主要道路和交界处。对于我们着手从事的这个“旅行”，最终目的地是深入理解有关计算机网络的怎么、什么和为什么。我们的路线图是本书各章的顺序：

第1章计算机网络和因特网
第2章应用层
第3章运输层
第4章网络层
第5章链路层和局域网
第6章无线网络和移动网络
第7章多媒体网络
第8章计算机网络中的安全
第9章网络管理

第2章到第5章是本书的4个核心章。你应当注意到，其中的每一章都对应于因特网协议栈上面4层中的一层。进一步要注意的是，我们的旅行将从因特网协议栈的顶部（即应用层）开始，然后向下学习。这种自顶向下旅行的基本原理是，一旦我们理解这些应用程序，就能够理解支持这些应用程序所需求的服务。然后依次研究由网络体系结构所可能实现的服务的各种方式。较早地涉及应用程序，也能够为学习本课程其余部分提供动力。

第6章到第9章关注现代计算机网络中的4个极为重要的（并且在某种程度上是独立的）主题。在第6章中，我们研究无线网络和移动网络，包括无线LAN（含有WiFi、WiMAX和蓝牙）、蜂窝电话网（含有GSM）和移动网络（含有IP网络和GSM网络）。在第7章中，我们研究音频和视频应用，例如因特网电话、视频会议和流式存储媒体。此外，还讨论如何设计分组交换网络，以对音频和视频应用程序提供一致的服务质量。在第8章中，我们首先学习加密和网络安全的基础知识，然后研究基础理论如何应用于因特网环境。最后一章（第9章）研究网络管理中的关键问题以及网络管理中使用的主要因特网协议。

课后习题和问题

复习题

1.1节

1. “主机”和“端系统”之间有什么不同？列举不同类型的端系统。Web服务器是端系统吗？
2. 新闻媒体讲到外交关系时经常使用“协议”一词。给出外交协议的一个例子。

1.2节

3. 什么是客户机程序？什么是服务器程序？服务器程序请求和接收来自客户机程序的服务吗？
4. 列出6种不同的接入技术。它们分别属于住宅接入、公司接入或无线接入中的哪一类？
5. HFC带宽是专用的还是在用户间共享的？在下行HFC信道中，有可能发生冲突吗？为什么？
6. 列出你所在城市中可供使用的住宅接入技术。对于每种接入方式，给出所宣称的下行速率、上行速率和每月的价格。
7. 以太LAN的传输速率有多高？对于给定的传输速率，LAN上的每个用户能够持续以该速率传输吗？
8. 能够运行以太网的一些物理媒体是什么？
9. 拨号调制解调器、HFC和DSL都用于住宅接入。对于每种技术，给出传输速率的范围并讨论有关带宽是共享的还是专用的。
10. 描述现在最流行的无线因特网接入技术。对它们进行比较。

1.3节

11. 与分组交换网络相比，电路交换网络有哪些优点？在电路交换网络中，TDM比FDM有哪些优点？
12. 为什么说分组交换应用了统计多路复用？将统计多路复用与TDM中使用的多路复用技术进行对比。
13. 假定在发送主机和接收主机间只有一个分组交换机。发送主机和交换机间以及交换机和接收主机间的传输速率分别是R_1和R_2。假设该交换机使用存储转发分组交换方式，发送一个长度为L的分组的端到端总时延是什么？（忽略排队时延、传播时延和处理时延。）
14. 第一层ISP和第二层ISP之间的关键差异是什么？
15. 假定用户共享一条2 Mbps的链路。同时假定每个用户持续以1 Mbps速率传输，但每个用户仅有20%的时间在传输。（参见1.3节中关于统计多路复用的讨论。）
    a. 当使用电路交换时，能够支持多少用户？
    b. 对于下面的问题，假定使用分组交换。为什么如果两个以下的用户同时传输的话，链路前面基本上没有排队时延？为什么如果3个用户同时传输的话，将有排队时延？
    c. 求某指定用户传输的概率。
    d. 假定现在有3个用户。求出在任何给定时刻，所有3个用户在同时传输的概率。求出排队增长的时间比率。

1.4节

16. 考虑从某源主机跨越一条固定路由向某目的主机发送一分组。列出分组的端到端时延中的时延组成。这些时延中，哪些是固定的？哪些是变化的？
17. 访问本书配套网站上有关传输时延与传播时延的Java小程序。在可用速率、传播时延和分组长度之中，找出一种组合，使得该分组的第一个比特到达接收方之前，发送方完成了传输。找出另一种组合，使得发送方完成传输之前，该分组的第一个比特到达了接收方。
18. 一个长度为1000字节的分组经距离为2500 km的链路传播，传播速率为2.5×10^8 m/s，传输速率为

2 Mbps，它需要用多长时间？更为一般地，一个长度为L的分组经距离为d的链路传播，传播速率为s，传输速率为R bps，它需要用多长时间？该时延与传输速率相关吗？

19. 假定主机A要向主机B发送一个大文件。从主机A到主机B的路径上有3段链路，其速率分别为R_1 = 500 kbps，R_2 = 2 Mbps，R_3 = 1 Mbps。
    a. 假定该网络中没有其他流量，该文件传送的吞吐量是什么？
    b. 假定该文件大小为4000 kB。将该文件传输到主机B大致需要多长时间？
    c. R_2减小到100 kbps，重复（a）和（b）。
20. 假定端系统A要向端系统B发送一个大文件。在一个非常高的层次上，描述端系统怎样为该文件创建分组。当这些分组之一到达某分组交换机时，该交换机使用分组中的什么信息来确定将该分组转发到哪一条链路上？因特网中的分组交换为什么可以与驱车从一个城市到另一个城市并沿途询问方向相类比？
21. 访问本书配套网站的排队和丢包Java小程序。什么是最大发送速率和最小传输速率？对于这些速率，流量强度是什么？用这些速率运行该Java小程序并确定到出现丢包要花费多长时间。然后重复该实验，再次确定到出现丢包要花费多长时间。这两个值有什么不同？为什么会出现这种现象？

1.5节

22. 列出一个层次能够执行的5个任务。这些任务中的一个（或多个）能够由两个（或多个）层次执行吗？
23. 因特网协议栈中的5个层次是什么？在这些层次中，每层的主要任务是什么？
24. 什么是应用层报文？什么是运输层报文段？什么是网络层数据报？什么是链路层帧？
25. 路由器处理因特网协议栈中的哪些层次？链路层交换机处理的是哪些层次？主机处理的是哪些层次？

1.6节

26. 病毒、蠕虫和特洛伊木马之间有什么不同？
27. 描述如何产生僵尸网络，僵尸网络是如何用于DDoS攻击的？
28. 假定Alice和Bob经计算机网络互相发送分组。假定Trudy将其自己安置在网络中，她能够捕获Alice发送的所有分组，并能够向Bob发送任何东西；她也能够捕获Bob发送的所有分组，并能够向Alice发送任何东西。在这种情况下，列出Trudy能够做的一些恶意事情。

习题

1. 设计并描述在自动柜员机和银行的中央计算机之间使用的应用层协议。你的协议应当允许验证用户卡和口令，查询账目结算（这些都在中央计算机系统中进行维护），支取账目（即向用户支付钱）。有关协议实体应当能够处理在取钱时账目中钱不够时的常见问题。通过列出自动柜员机和银行的中央计算机在报文传输和接收过程中交换的报文和采取的动作，定义该协议。使用类似于图1-2所示的图，拟定在简单无差错取钱情况下该协议的操作。明确地阐述在该协议中有关底层端到端运输服务所做的假设。
2. 考虑一个应用程序以稳定的速率传输数据（例如，发送方每k个时间单元产生一个N比特的数据单元，其中k较小且固定）。另外，当这个应用程序启动时，它将持续运行相对长的一段时间。回答下列问题，简要论证你的答案：
    a. 分组交换网还是电路交换网更适合这种应用？为什么？
    b. 假定使用了分组交换网，并且该网中的所有流量都来自如上所述的这种应用程序。同时，假定该应用程序数据传输速率的总和小于每条链路的各自容量。需要某种形式的拥塞控制吗？为什么？
3. 考虑图1-8中的电路交换网。其中在每条链路上有n条电路。
    a. 在该网络中，任何时候能够进行同时连接的最大数量是多少？

b. 假定所有连接位于左上角的交换机和右下角的交换机之间。能够进行同时连接的最大数量是多少?

4. 回顾1.4节中车队的类比。假定传播速度还是100 km/h。
   a. 假定车队旅行200km：在收费站前面开始，通过第二个收费站，并且在第三个收费站前面结束。端到端时延是多少?
   b. 重复（a），现在假定车队中有7辆汽车，而不是10辆。

5. 这个基本问题开始研究传播时延和传输时延，这是数据网络中的两个重要概念。考虑两台主机A和B由一条速率为R bps的链路相连。假定这两台主机相隔m米，沿该链路的传播速率为s m/s。主机A向主机B发送长度为L比特的分组。
   a. 用m和s表示传播时延d_{prop}。
   b. 用L和R确定分组的传输时间d_{trans}。
   c. 忽略处理时延和排队时延，得出端到端时延的表达式。
   d. 假定主机A在时刻$t = 0$开始传输该分组。在时刻$t = d_{trans}$，该分组的最后一个比特在什么地方?
   e. 假定d_{prop}大于d_{trans}。在时刻$t = d_{trans}$，该分组的第一个比特在什么地方?
   f. 假定d_{proc}小于d_{trans}。在时刻$t = d_{trans}$，该分组的第一个比特在什么地方?
   g. 假定$s = 2.5 \times 10^8$，$L = 100$比特，$R = 28$ kbps。求出d_{proc}等于d_{trans}的距离m。

6. 在这个问题中，我们考虑从主机A向主机B通过分组交换网发送话音（VoIP）。主机A将模拟话音转换为传输中的64 kbps数字比特流。主机A然后将这些比特分为48字节的分组。A和B之间有一条链路：它的传输速率是1 Mbps，传播时延是2 ms。一旦A收集了一个分组，就将它向主机B发送。一旦主机B接收到一个完整的分组，就将该分组的比特转换成模拟信号。从比特产生（从位于主机A的初始模拟信号起）的时刻起，到该比特被解码（在主机B上作为模拟信号的一部分）花了多少时间?

7. 假定用户共享一条1 Mbps的链路。又设每个用户传输时要求100 kbps，但是每个用户仅有10%的时间在传输。（参见1.3节中关于统计多路复用的讨论。）
   a. 当使用电路交换时，能够支持多少用户?
   b. 对于下面的问题，假定使用分组交换。求某给定用户传输的概率。
   c. 假定有40个用户。求出在任何给定时刻，实际有n个用户在同时传输的概率。（提示：使用二项式分布。）
   d. 求出有11个或更多用户同时传输的概率。

8. 考虑在1.3节关于统计多路复用的讨论中，给出了一个具有一条1 Mbps链路的例子。用户在忙时以100 kbps速率产生数据，但忙于产生数据的概率仅是$p = 0.1$。假定用1 Gbps链路替代1 Mbps的链路。
   a. 当采用电路交换技术时，能被同时支持的最大用户数量N是多少?
   b. 现在考虑分组交换和有M个用户的情况。给出多于N用户发送数据的概率公式（用p、M、N表示）。

9. 考虑一个长度为L的分组从端系统A开始，经一段链路传送到一台分组交换机，并从该分组交换机经第二段链路传送到目的端系统。令d_i、s_i和R_i分别表示链路i（$i = 1, 2$）的长度、传播速度和传输速率。该分组交换机对每个分组的时延为d_{proc}。假定没有排队时延，根据d_i、s_i、R_i（$i = 1, 2$）和L，该分组总的端到端时延是什么？现在假定该分组是1000字节，分组交换机的处理时延是1 ms，第一段链路的长度是4000 km，并且最后一段链路的长度是1000 km。对于这些值，端到端时延为多少?

10. 在上述习题中，假定$R_1 = R_2 = R$且$d_{proc} = 0$。进一步假定该分组交换机不存储-转发分组，而是在等待分组到达前立即传输它收到的每个比特。这时端到端时延为多少?

11. 一台分组交换机接收一个分组并决定该分组应当转发的出链路。当某分组到达时，另一个分组正在该出链路上被发送到一半，还有3个其他分组在等待传输。这些分组以到达的次序传输。假定所有分

组是1000字节并且链路速率是1 Mbps。该分组的排队时延是多少？在更一般的情况下，当所有分组的长度都是L，传输速率是R，当前正在传输的分组已经传输了x比特，并且已经在队列中有n个分组时，其排队时延是多少？

12. 假定有N个分组同时到达一条当前没有分组传输或排队的链路。每个分组的长度为L，链路传输速率为R。对N个分组而言，其平均排队时延是多少？

13. 考虑在路由器缓存中的排队时延（在输出链路的前端）。假定所有分组有L比特，传输速率是R bps，每隔LN/R s有N个分组同时到达缓存。求出分组的平均排队时延。（提示：第一个分组的排队时延是0，第二个分组的时延是L/R；第三个分组的排队时延是$2L/R$。当第二批分组到达时，第N个分组已经传输。）

14. 考虑路由器缓存中的排队时延。令I表示流量强度，即$I = La/R$。假定排队时延的形式为$IL/R\ (1-I)$，其中$I < 1$。

    a. 写出总时延公式，即排队时延加上传输时延。

    b. 以L/R为函数画出总时延的图。

15. a. 对于不同的处理速率、传输速率和传播速率，给出1.4.3节中端到端时延公式的一般表达式。

    b. 重复（a），但现在假定在每个节点有d_{queue}的平均排队时延。

16. 在一天的3个不同的小时内，在同一个大陆上的源和目的地之间执行Traceroute。

    a. 在这3个小时的每个小时中，求出往返时延的均值和方差。

    b. 求出在这3个小时中各自的路由器数量。在这些时段中，路径发生变化了吗？

    c. 试图根据源到目的地Traceroute分组通过的情况，辨明ISP网络的数量。具有类似名字和/或类似的IP地址的路由器应当被认为属于相同ISP。在你的实验中，在相邻的ISP间的对等接口处出现了最大的时延了吗？

    d. 对位于不同大陆上的源和目的地重复上述内容。比较大陆内部和大陆之间的这些结果。

17. 考虑对应于图1-16b吞吐量的例子。现在假定有M对客户机/服务器而不是10对。用R_s、R_c和R分别表示服务器链路、客户机链路和网络链路的速率。假设所有的其他链路都有充足容量，并且除了这M对客户机/服务器产生的流量外，网络中没有其他流量。推导出用R_s、R_c、R和M表示的通用吞吐量表达式。

18. 假定两台主机A和B相隔10 000 km，由一条直接的R = 1 Mbps的链路相连。假定跨越该链路的传播速率是2.5×10^8m/s。

    a. 计算“带宽时延”积$R \cdot t_{prop}$。

    b. 考虑从主机A向主机B发送一个400 kb的文件。假定该文件作为一个大的报文连续发送。在任何给定的时间，在链路上具有的比特数量最大值是多少？

    c. 给出带宽时延积的一种解释。

    d. 该链路上一个比特的宽度（以米计）是多少？它比一个足球场更长吗？

    e. 根据传播速率s、带宽R和链路m的长度，推导出比特宽度的一般表示式。

19. 对于问题18，假定我们能够修改R。对什么样的R值，一个比特的宽度能与该链路的长度一样长？

20. 考虑问题18，但现在链路的速率是R = 1 Gbps。

    a. 计算带宽时延积$R \cdot t_{prop}$。

    b. 考虑从主机A向主机B发送一个400 kb的文件。假定该文件作为一个大的报文连续发送。在任何给定的时间，在链路上具有的比特数量最大值是多少？

    c. 该链路上的一个比特的宽度（以米计）是多少？

21. 再次考虑问题18。

a. 假定连续发送，发送该文件需要多长时间？

b. 假定现在该文件被划分为10个分组，每个分组包含40 kb。假定每个分组被接收方确认，确认分组的传输时间可忽略不计。最后，假定在前一个分组被确认后，发送方才能发送分组。发送该文件需要多长时间？

c. 比较（a）和（b）的结果。

22. 假定在同步卫星和它的地球基站之间有一条10 Mbps的微波链路。每分钟该卫星拍摄一幅数字照片，并将它发送到基站。假定传播速率是2.4×10^8 m/s。

a. 该链路的传播时延是多少？

b. 带宽时延积$R \cdot t_{prop}$是多少？

c. 令x表示该照片的长度。对于这条微波链路，能够连续传输的x最小值是多少？

23. 考虑1.5节中我们讨论分层时航线旅行的类比，以及随着协议数据单元向协议栈底层流动时增加的首部。随着旅客和行李移动到航线协议栈底部，有与首部信息等价的概念吗？

24. 在现代分组交换网中，源主机将长的应用层报文（如一个图像或音乐文件）分段为较小的分组并向网络发送。接收方则将这些分组重新装配为初始报文。我们称这个过程为*报文分段*。图1-24说明了一个报文的端到端传输是否具有报文分段。考虑一个大小为7.5Mb的报文，它在图1-24中从源发送到目的地。假定该图中的每段链路是1.5 Mbps。忽略传播时延、排队时延和处理时延。

a. 考虑从源到目的地无报文分段地发送该报文。从源主机到第一台分组交换机移动报文需要多长时间？记住，每台交换机使用“存储转发”分组交换，从源主机到目的主机移动该报文需要多长时间？

b. 现在假定该报文被分段为5000个分组，每个分组1500比特长。从源主机到第一台交换机移动第一个分组需要多长时间？第一个分组从第一台交换机发送到第二台交换机，第二个分组从源主机发送到第一台交换机各需要多长时间？什么时候第二个分组能被第一台交换机全部收到？

c. 当使用报文分段时，从源主机向目的主机移动该文件需要多长时间？将该结果与（a）部分答案进行比较并解释之。

d. 讨论报文分段的缺点。

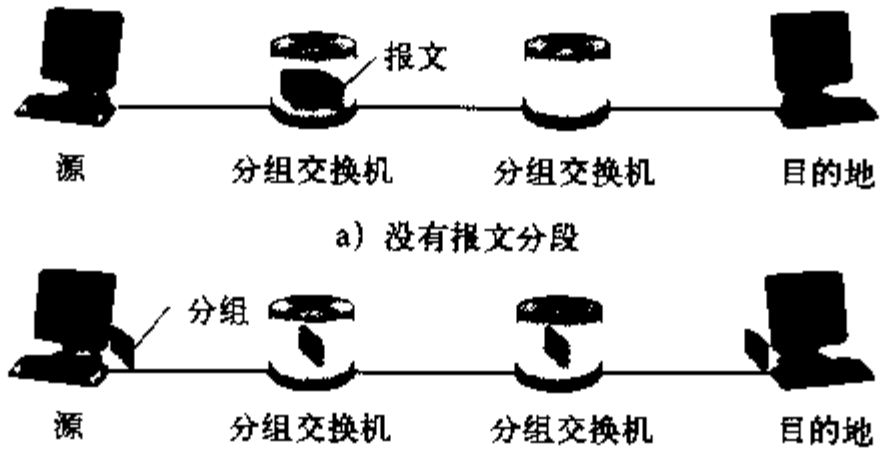


图1-24 端到端报文传输

25. 用本书配套网站上的报文分段Java小程序进行实验。该Java小程序中的时延与前面一个问题中的时延相当吗？链路传播时延是怎样影响分组交换（有报文分段）和报文交换的总端到端时延的？

26. 考虑从主机A向主机B发送一个F比特的大文件。A和B之间有两段链路（和一台交换机），并且该链路不拥塞（即没有排队时延）。主机A将该文件分为每个为S比特的报文段，并为每个报文段增加一个40比特的首部，形成$L = 40 + S$比特的分组。每条链路的传输速率为R bps。求出从A到B移动该文

件，使时延最小的S值。忽略传播时延。

讨论题

1. 在你所在的地区中，有什么样的无线蜂窝服务可供使用？
2. 使用802.11无线LAN技术，为你的家庭或你父母家设计一个家庭网络。列出你的家庭网络的特定产品模型和价格。
3. 描述PC到PC的Skype服务。试验Skype的PC到PC视频服务，事后写出报告。
4. Skype提供一种服务，使你能够从PC拨电话给普通电话。这意味着话音电话必须通过因特网和电话网传输。讨论这是怎样做到的。
5. 什么是短消息服务（Short Message Service, SMS）？这个服务在哪些国家/大洲流行？可以从一个Web站点向移动电话发送一条SMS吗？
6. 什么是存储流式视频？列举目前提供流式视频的一些Web站点。
7. 什么是P2P流式实况视频？列举目前提供该服务的一些Web站点。
8. 找出5个提供点对点文件共享的公司。对于这些公司，它们各自处理什么类型的文件（即内容）？
9. 谁发明了第一个即时讯息服务——ICQ？什么时候发明的？发明者当时多大？类似地，谁发明了Napster？什么时候发明的？发明者当时多大？
10. 比较和对照WiFi无线因特网接入和3G无线因特网接入。这两种服务的比特速率是多少？它们的成本有多大？讨论漫游和接入无所不在。
11. 为什么初始的Napster P2P文件共享服务不复存在？什么是RIAA？它采用何种措施来限制有版权内容的P2P文件共享？直接和间接违反版权有什么区别？
12. 什么是BitTorrent？它与P2P文件共享服务（如eDonkey、LimeWire或Kazaa）本质上有什么不同？
13. 你认为从现在起的10年中，在计算机网络上仍会有版权文件的广泛共享吗？为什么？请详细叙述。

Ethereal实验

“不闻不若闻之，闻之不若见之，见之不若知之，知之不若行之。”

——中国谚语

一个人对网络协议的理解通过以下方法通常能够得到极大的深化：观察它们的工作和使用它们，即观察两个协议实体之间交换的报文序列，钻研协议运行的细节，使协议执行某些动作，观察这些动作及其后果。这能够在仿真环境下或在因特网等真实网络环境下完成。本书配套网站上的Java小程序采用的是第一种方法。在Ethereal实验中，我们将采用后一种方法。使用桌面上的计算机在各种情况下运行网络应用程序。在计算机上观察网络协议与在因特网别处执行的协议实体进行交互和交换报文。因此，你与你的计算机将是这些实际实验的组成部分。通过实际动手来观察和学习。

观察在执行协议的实体之间交换的报文的基本工具被称为**分组嗅探器**（packet sniffer）。顾名思义，一个分组嗅探器被动地拷贝（嗅探）由你的计算机发送和接收的报文；它也能显示出这些被捕获报文的各个协议字段的内容。图1-25显示了Ethereal分组嗅探器的屏幕快照。Ethereal是一个运行在Windows、Linux/UNIX和Mac计算机上的免费分组嗅探器。贯穿全书，你将发现Ethereal实验帮助你探索在该章中学习的一些协议。在这第一个Ethereal实验中，你将获得并安装一个Ethereal的拷贝，访问一个Web站点，捕获并检查在你的Web浏览器和Web服务器之间交换的协议报文。

可以从Web站点http://www.awl.com/kurose-ross上找到有关该第一个Ethereal实验的全部材料（包括

如何获得并安装Ethereal的指导)。

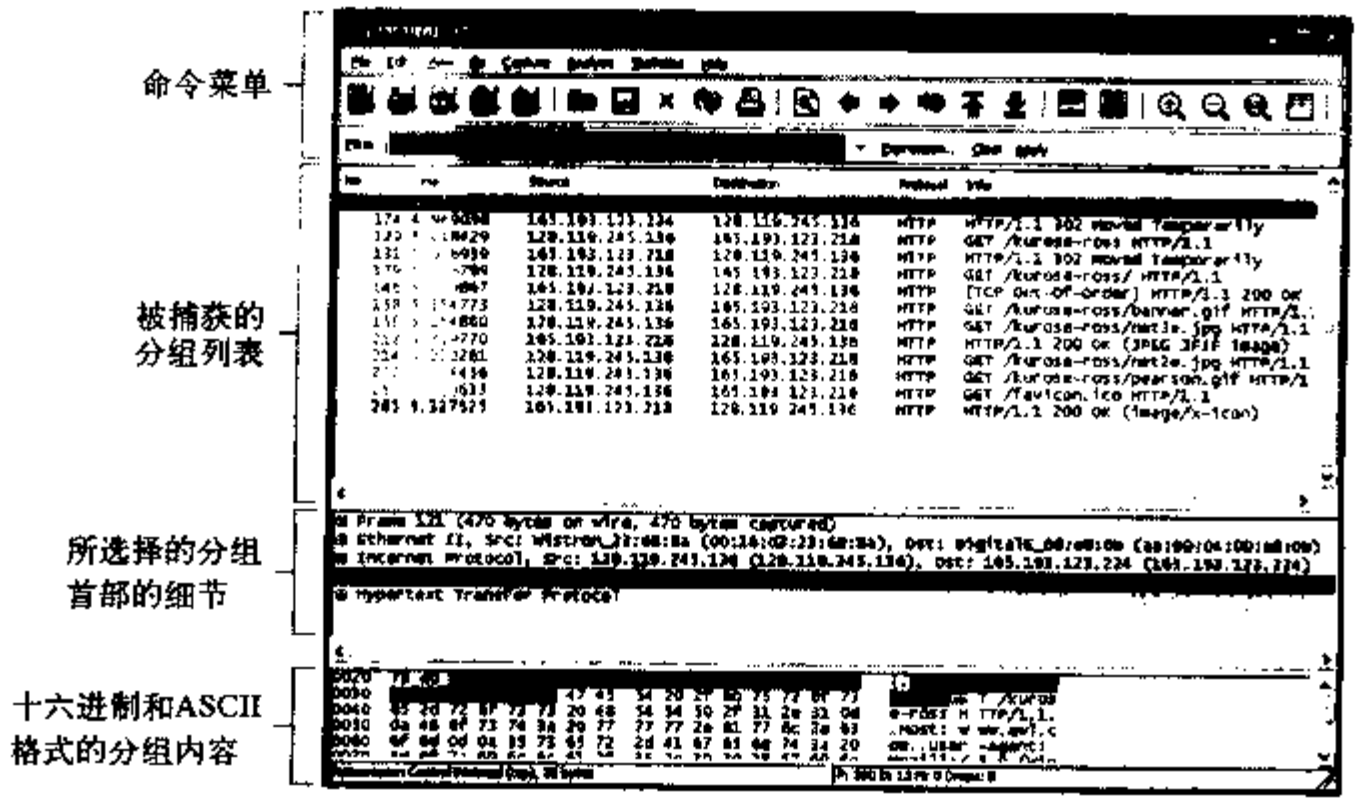


图1-25 一个Ethereal屏幕快照

人物专访

Leonard Kleinrock

Leonard Kleinrock是加州大学洛杉矶分校（UCLA）的计算机科学教授。1969年，他在UCLA的计算机成为因特网的第一个节点。1961年，他创造的分组交换原理成为因特网的支撑技术。Leonard也是Noamdix有限公司的主席和创始人。这家公司具有提供更大的宽带因特网服务接入能力的技术。他获得了纽约城市大学（City College of New York, CCNY）的电子工程学士学位，并获得了麻省理工学院（MIT）的电子工程硕士和博士学位。

• 是什么使得您决定专门研究网络/因特网技术的?

当我于1959年在MIT读博士时，我发现我周围的大多数同学正在研究信息理论和编码理论领域。在MIT，那时有伟大的研究者Claude Shannon，他开创了这些领域，并且已经解决了许多重要的问题。留下来的研究问题既难又不太重要。因此我决定开始新的研究领域，该领域其他人还没有想到。回想那时在MIT我的周围有许多计算机，我很清楚很快这些计算机将有相互通信的需求。在那时，却没有有效的方式来做到这一点，因此我决定研发能创建有效的数据网络的技术。

• 您在计算机产业的第一份工作是什么？它使您得到了什么收益?

1951～1957年，我为了获得电子工程学士学位在CCNY读夜大。在那段时间里，我在一家称为Photobell的小工业电子公司工作，先是作技术员，然后作工程师。在那里，我在它们的产品线上引入了数字技术。我们主要使用光电子设备来检测某些物体（盒子、人等）的存在，一种称为双稳态多频振荡器的电路的使用正是我们需要的技术类型，这可以将数字处理引入检测领域。这些电路恰好是计算机的基本模块，用今天的话说就是触发电路或交换机。

• 当您发送第一个主机到主机报文（从UCLA到斯坦福研究院）时，您的心中想到了什么?

第一个主机到主机报文有些虎头蛇尾。在我的心中，印象更为深刻的第一个事件发生在1969年9月2

日，那是第一个网络设备（IMP）与外面的（位于UCLA的我的主机）第一个运行的系统连接起来。那是因特网诞生的时刻。那年的早些时候，UCLA新闻稿引用我的话说，一旦该网络建立并运行起来，将能够从家中和办公室访问计算机设施，就像我们获得电力和电话连接那样容易。因此那时我的美好愿望是，因特网将无所不在、总是运行、总是可用，任何人从任何地方用任何设备都能够与之相连，并且它将是不可见的。然而，我从没有期待我99岁的母亲今天能够上因特网，她的确做到了这一点。

• 您对未来网络的展望是什么?

我的展望中最清晰的部分是移动计算和智能空间。轻量级的、不昂贵的、高性能的、便携式的计算设备的可用性加上因特网的无所不在，将能够使我们移动。移动计算是指使端用户从一个地方移到另一个地方时能够以透明方式访问因特网服务，而无论他们移动到哪里。然而，移动计算仅是迈出的一步。下一步将使我们从计算机的虚拟空间走向智能的物理世界。我们的环境（办公桌、墙壁、运输工具、手表、皮带等）将通过调节器、传感器、逻辑、处理、存储、照相机、扩音器、话筒、显示器和通信，随着技术发展而具有生命力。嵌入式技术将允许我们的环境提供我们所需的IP服务。当我走进房间时，这间房子就会知道我进来了。我将与我的环境自然地进行通信，就像说英语那样；我的请求将从墙上显示器中的Web网页得到回答，通过眼镜、声音、全息摄影等方式送给我。

再向前看一点，我能看到未来的网络将包括下列附加的关键组件。我看到在网络各处设置的智能软件代理，它的功能是挖掘数据，根据数据采取动作，观察趋势，并能动态地、自适应地执行我们的任务。我看到大量网络流量并不是由人产生的，而是由这些嵌入式设备和这些智能软件代理产生的。我看到巨量的信息瞬间通过网络立即得到大量的处理和过滤。因特网最终将是一个无所不在的全球性神经系统。当我们在21世纪急速前进时，我将看到这些东西和更多的东西。

• 哪些人激发了您的专业灵感?

到目前为止，是MIT的Claude Shannon。他是一名卓越的研究者，具有以高度直觉的方式将他的数学理念与物理世界关联起来的能力。他是我博士论文答辩委员会的成员。

• 您对进入网络/因特网领域的学生们有什么忠告吗?

因特网以及与它相关的东西是一个巨大的战场，充满了令人惊奇的挑战，为众多创新提供了广阔空间。不要受今天技术的束缚，开动大脑，想象能够做些什么，并去实现它。

第2章 应 用 层

网络应用是计算机网络存在的理由，如果我们不能构想出任何有用的应用，也就没有任何必要去设计支持它们的网络协议了。在过去的40年里，创造出了无数有影响力而奇妙的网络应用，这些应用程序既包括20世纪70年代和80年代开始流行的、经典的基于文本的几种应用（如文本电子邮件、计算机远程访问、文件传输、新闻组以及文本聊天），也包括20世纪90年代中期开始出现的招人喜爱的应用程序——万维网（包含了Web冲浪、搜索和电子商务）。此外，还包括20世纪末引人的两个招人喜爱的应用，即具有好友列表的即时讯息和对等（P2P）文件共享。当然，还有很多成功的音频和视频应用，如因特网电话、视频共享和流式视频、因特网收音机和IP电视（IPTV）。此外，宽带住宅接入和无线接入的日益普及和不断突破，为未来更多的新型应用提供了舞台。

在本章中，我们学习有关网络应用的原理和实现方面的知识。我们从定义几个关键的应用层概念开始，其中包括应用程序所需要的网络服务、客户机和服务器、进程和运输层接口。然后，详细讨论几种网络应用程序，包括Web、电子邮件、DNS、对等文件分发和P2P因特网电话。接下来，将探讨开发运行在TCP和UDP上的网络应用程序的方法，特别是学习套接字API，并概要学习用Java实现的几个简单的客户机/服务器应用。在本章后面，我们也将提供几个有趣而令人好奇的套接字编程作业。

应用层是我们学习协议时非常好的起点，因为它是我们最熟悉的领域。我们熟悉的很多应用就是建立在这些将要学习的协议基础上的。通过对应用层的学习，将有助于我们认知协议的有关知识，将使我们学习到进行运输层、网络层及链路层协议学习时也会碰到的很多相同的问题。

2.1 应用层协议原理

现在假定你对某种新型网络应用有许多想法。也许这种应用将对人类提供伟大的服务，或者将使你的教授高兴，或者将带给你大量的财富，或者只是开发得有趣。无论你的动机是什么，我们现在考察一下如何将你的想法转变为一种真实世界的网络应用。

研发网络应用程序的核心是写出能够运行在不同的端系统和通过网络彼此通信的程序。例如，在Web应用程序中，有两个互相通信的不同的程序：一个是运行在用户主机（桌面机、膝上机、PDA、蜂窝电话等）上的浏览器程序；另一个是运行在Web服务器主机上的Web服务器程序。又比如P2P文件共享系统，在参与文件共享的每台主机中都有一个程序。在这种情况下，每台主机中的这些程序可能都是类似的或相同的。

因此，当研发新应用程序时，你需要编写将在多台端系统上运行的软件。例如，该软件可以用C、C++或Java语言来编写。重要的是，你不需要写在网络核心设备（如路由器或链路层交换机）上运行的软件。即使你想为网络核心设备写应用程序软件，你也不能够做到这一点。如图1-20所显示的那样，网络核心设备并不在应用层起作用，而是在较低层起作用，具体来说是在网络层及下面层次。这种基本设计的方法，即将应用软件限制在端系统（如图2-1所示），促进了大量的因特网应用程序的研发和部署。

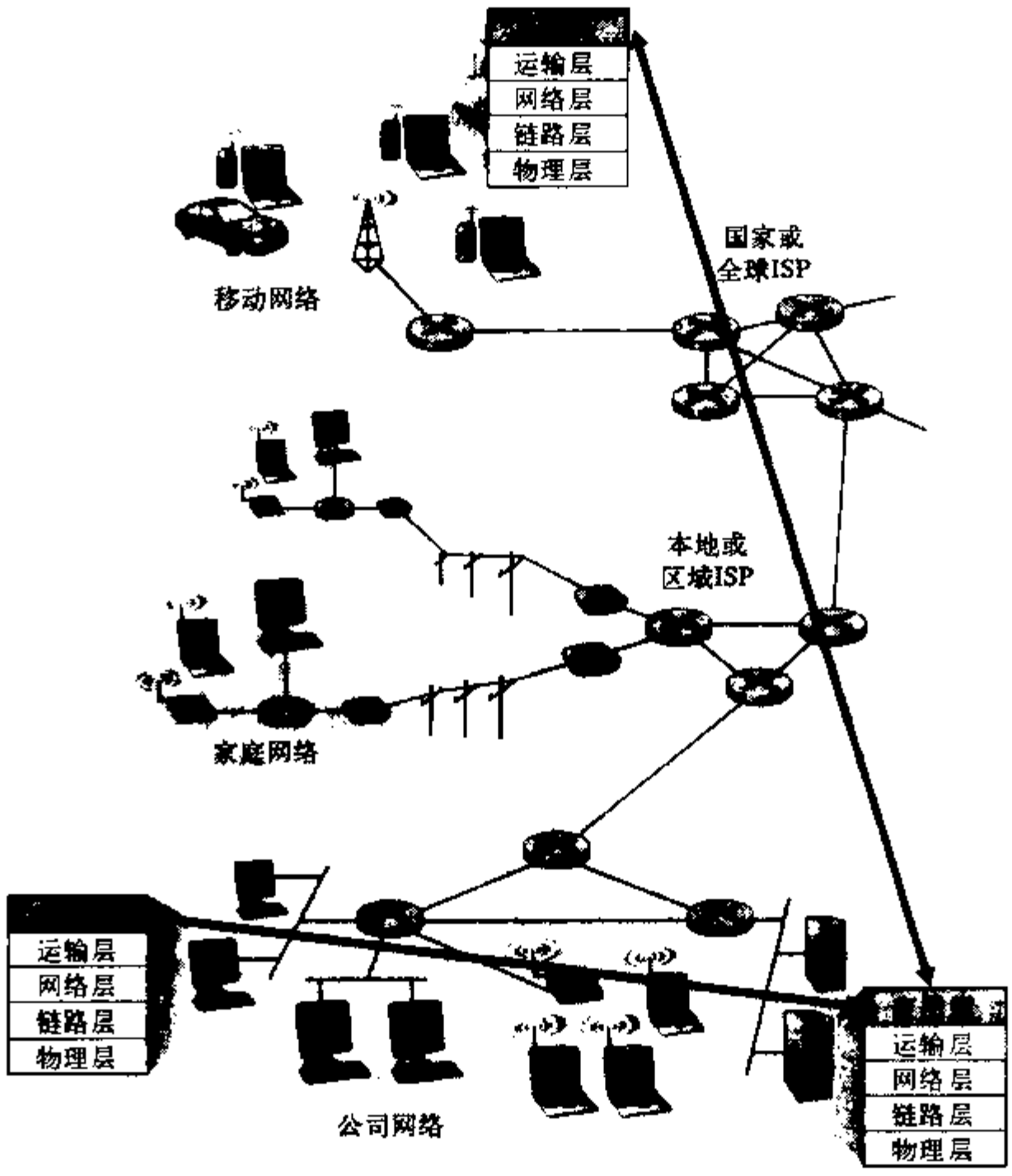


图2-1 在应用层的端系统之间的网络应用的通信

2.1.1 网络应用程序体系结构

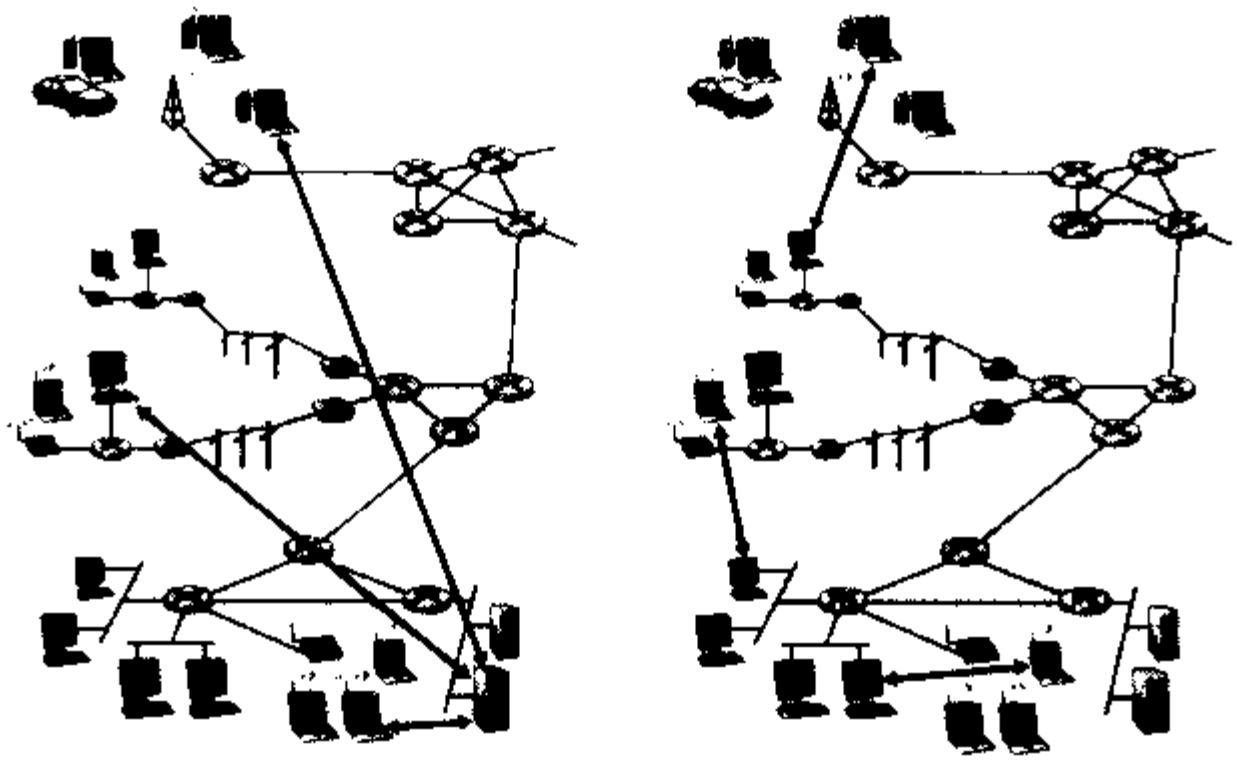
当进行软件编码之前，应当对应用程序有一个宽泛的体系结构计划。记住，应用程序的体系结构明显不同于网络的体系结构（例如，在第1章中所讨论的5层因特网体系结构）。从应用程序研发者的角度看，网络体系结构是固定的，并为应用程序提供了特定的服务集合。另一方面，**应用程序体系结构**（application architecture）由应用程序研发者设计，规定了如何在各种端系统上组织该应用程序。在选择应用程序体系结构时，应用程序研发者很可能利用现代网络应用程序中所使用的两种主流体系结构之一：客户机/服务器体系结构或对等（P2P）体系结构。

在**客户机/服务器体系结构**（client-server architecture）中，有一个总是打开的主机称为服务器，它服务于来自许多其他称为客户机的主机请求。客户机主机既可能有时打开，也可能总是打开。一个典型的例子是Web应用程序，其中总是打开的Web服务器服务于运行在客户机主机上的浏览器的请求。当Web服务器接收到来自某客户机对某对象的请求时，它向该客户机发送所请求的对象作为响应。注意到客户机/服务器体系结构中，客户机相互之间不直接通信，例如，在Web应用中两个浏览器并不直接通信。客户机/服务器体系结构的另一个特征是

服务器具有固定的、周知的地址，称为IP地址（我们将很快讨论之）。因为服务器具有固定的、周知的地址，并且总是处于打开状态，所以客户机总是能够通过向该服务器的地址发送分组来与其联系。某些具有客户机/服务器体系结构的更为著名的应用程序包括Web、FTP、Telnet和电子邮件。图2-2a中显示了这种客户机/服务器体系结构。

在客户机/服务器应用中，常会出现一台服务器主机跟不上其所有客户机请求的情况。例如，一个流行的社会联网站点如果仅有一台服务器来处理所有请求，将很快变得不堪重负。为此，在客户机/服务器体系结构中，常用主机群集（有时称为**服务器场**（server farm））创建强大的虚拟服务器。基于客户机/服务器体系结构的应用服务通常是**基础设施密集的**(infrastrucuture intensive)，因为它们要求服务提供商购买、安装和维护服务器场。此外，服务提供商必须支付为了发送和接收到达/来自因特网的数据而不断出现的互联和带宽费用。搜索引擎（如Google）、因特网商务（如Amazon和e-Bay）、基于Web的电子邮件（如Yahoo Mail）、社会联网（如MySpace和Facebook）和视频共享（如YouTube）等流行服务是基础设施密集的，需要投入巨额费用。

在**P2P体系结构**（P2P architecture）中，对总是打开的基础设施服务器有最小的（或者没有）依赖。相反，任意间断连接的主机对——称为对等方，直接相互通信。对等方并不为服务提供商所有，而是为用户控制的桌面机和膝上机所有，大多数对等方驻留在家庭、大学和办公室。因为这种对等方通信不必通过专门的服务器，所以该体系结构被称为对等方到对等方(简称为对等)。目前，大多数流行的流量密集型应用程序都是P2P体系结构的，包括文件分发(如BitTorrent)、文件搜索/共享（如eMule和LimeWire）、因特网电话（如Skype）和IPTV（如PPLive）。图2-2b中给出了P2P体系结构的例子。需要提及的是，某些应用具有混合的体系结构，由客户机/服务器和P2P元素结合而成。例如，对于许多即时讯息应用而言，服务器场用于跟踪用户的IP地址，但用户到用户的报文在用户主机之间直接发送（无需通过中间服务器）。



a) 客户机/服务器体系结构　　b) 对等体系结构

图 2-2

P2P体系结构的最突出特性之一是它的**自扩展性**（self-scalability）。例如，在一个P2P文件共享应用中，尽管每个对等方都由请求文件产生负载，但每个对等方向其他对等方分发文

件也为系统增加了服务能力。P2P体系结构也是成本有效的，因为它们通常不需要庞大的服务器基础设施和服务器带宽。为了降低成本，服务提供商（MSN、Yahoo等）对于将P2P体系结构用于应用的兴趣越来越大。另一方面，由于P2P应用程序具有高度分布和开放的性质，因此要格外关注系统的安全[Doucer 2002; Yu 2006; Liang 2006; Naoumov 2006]。

2.1.2 进程通信

在构建网络应用程序前，也需要对运行在多个端系统上的程序的互相通信情况有一个基本了解。在操作系统的术语中，进行通信的实际上是**进程**（process）而不是程序。进程可以被认为是运行在端系统中的程序。当进程运行在相同的端系统上的时候，它们使用进程间通信机制相互通信。进程间通信的规则由端系统上的操作系统确定。在本书中，我们对运行在同一台主机上的进程间的通信不感兴趣，而只关注运行在不同端系统（可能具有不同的操作系统）上的进程间的通信。

不同端系统上的进程通过跨越计算机网络交换**报文**（message）而相互通信。发送进程创建并向网络中发送报文，接收进程接收这些报文并可能负责回送报文。图2-1举例说明了进程是如何通过使用5层协议栈的应用层而互相通信的。

1. 客户机和服务器进程

网络应用程序是由成对的进程组成，这些进程通过网络相互发送报文。例如，在Web应用程序中，一个客户机浏览器进程与一台Web服务器进程交换报文。在一个P2P文件共享系统中，文件从一个对等方中的进程传输到另一个对等方中的进程。对每对通信进程，我们通常将这两个进程之一标示为**客户机**（client），而另一个进程标示为**服务器**（server）。在Web中，浏览器是一个客户机进程，Web服务器是一个服务器进程。对于P2P文件共享，下载文件的对等方被标示为客户机，上载文件的对等方被标示为服务器。

你或许已经观察到，在P2P文件共享等应用中，一个进程可以既是客户机又是服务器。实际上，在P2P文件共享系统中，一个进程既能上载文件又能下载文件。无论如何，在给定的一对进程之间的通信会话中，我们仍能标示一个进程为客户机，标示另一个进程为服务器。我们定义客户机和服务器进程如下：

> 在给定的一对进程之间的通信会话中，发起通信（即在该会话开始时与其他进程联系）的进程被标示为**客户机**，在会话开始时等待联系的进程是**服务器**。

在Web中，一个浏览器进程向某Web服务器进程发起联系，因此该浏览器进程是客户机，而该Web服务器进程是服务器。在P2P文件共享中，当对等方A请求对等方B发送一个特定的文件时，在这个特定的通信会话中对等方A是客户机，而对等方B是服务器。在不会混淆的情况下，我们有时也使用术语“应用程序的客户机端和服务器端”。在本章的结尾，我们将逐步讲解网络应用程序的客户机端和服务器端的简单代码。

2. 进程与计算机网络之间的接口

如上所述，多数应用程序由通信进程对组成，每对中的两个进程互相发送报文。从一个进程向另一个进程发送的报文必须通过下面的网络。进程通过一个称为**套接字**（socket）的软件接口在网络上发送和接收报文。为了有助于理解进程和套接字，我们来打个比方。进程可类比于一座房子，而它的套接字可以类比于它的门。当一个进程想向位于另外一台主机上的另一个进程发送报文时，它把报文推出门（套接字）。该发送进程假定门到另一侧之间有运输

的基础设施，该设施将把报文传送到目的进程的门口。一旦报文抵达目的主机，它通过报文接收进程的门（套接字）传递，然后接收进程对报文进行相应的处理。

图2-3显示了两个经过因特网通信的进程之间的套接字通信情况（图2-3中假定该进程使用的下面的运输层协议是因特网的TCP协议）。如图所示，套接字是同一台主机内应用层与运输层之间的接口。由于该套接字是在网络上建立网络应用程序的可编程接口，因此也将套接字称为应用程序和网络之间的**应用程序编程接口**（Application Programming Interface，API）。应用程序开发者可以控制套接字在应用层端的所有东西，但是对该套接字的运输层端几乎没有控制。应用程序开发者对于运输层的控制仅限于：①选择运输层协议；②也许能设定几个运输层参数，如最大缓存、最大报文段长度等（在第3章中讨论）。一旦应用程序开发者选择了一个运输层协议（如果可供选择的话），则应用程序就建立在由该协议提供的运输层服务之上。我们将在2.7节和2.8节中对套接字进行更为详细的探讨。

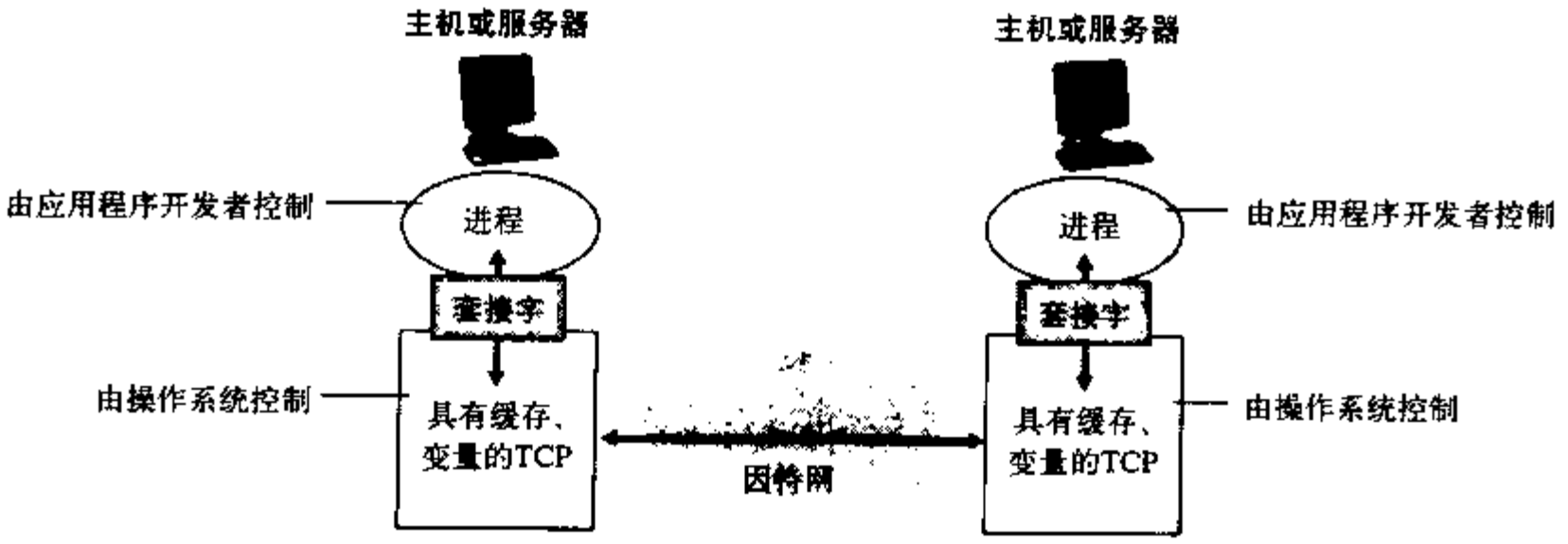


图2-3 应用进程、套接字和下面的运输层协议

2.1.3 可供应用程序使用的运输服务

前面讲过，套接字是应用程序进程和运输层协议之间的接口。发送端的应用程序通过套接字发送报文。在套接字的另一侧，运输层协议负责将该报文传输到接收套接字的“门”中。

包括因特网在内的很多网络提供了不止一种运输层协议。当开发一个应用时，必须选择一种可用的运输层协议。如何做出这种选择呢？最可能的方式是，通过研究这些可用的运输层协议所提供的服务，选择一个能为你的应用提供恰当服务的协议。这种情况类似于在两个城市间旅行时，选择飞机或火车作为交通工具。你必须选择其中一种，而且不同的交通方式为你提供不同的服务。（例如，火车可以直到市区上下，而飞机提供了更短的旅行时间。）

运输层协议能为调用它的应用程序提供什么样的服务呢？我们大体可以从四个方面对应用程序服务要求进行分类：可靠数据传输、吞吐量、定时和安全性。

1. 可靠数据传输

如第1章讨论的那样，分组能在计算机网络中丢失。例如，分组能够使路由器中的缓存溢出，或者当分组中的某些比特损坏后可能被丢弃。像电子邮件、文件传输、远程主机访问、Web文档传输以及金融应用等应用，数据丢失可能会造成灾难性的后果。（特别是在金融应用中，对银行和顾客都有影响！）因此，对于这些应用，必须确保由应用程序的一端发送的数据正确地、完全地交付给该应用程序的另一端。如果一个协议提供了这样的确保数据交付服务，就提供了**可靠数据传输**（reliable data transfer）。运输层协议能够潜在地向应用程序提供的一

个重要服务是进程到进程的可靠数据传输。当一个运输层协议提供这种服务时，发送进程只要将其数据传递到套接字，就可以相信该数据将能无差错地到达接收进程。

当一个运输层协议不提供可靠数据传输时，由发送进程发送的数据可能不能到达接收进程。对于容忍丢失的应用（loss-tolerant application）来说这是可以接受的。最值得注意的是多媒体应用，如实时音频/视频或存储的音频/视频，它们能承受一定程度的数据丢失。在这些多媒体应用中，丢失的数据引起播放的音频/视频出现小干扰，而不是致命的损伤。

2. 吞吐量

在第1章中我们引入了可用吞吐量的概念，两个进程在一条网络路径上进行通信会话时，可用吞吐量就是发送进程能够向接收进程交付比特的速率。因为其他会话将共享沿着该网络路径上的带宽，并且这些其他会话将会到达和离开，所以可用吞吐量将随时间波动。自然就有了另一种服务，即运输层协议能够以某种特定的速率提供确保的可用吞吐量。使用这种服务，应用程序可以请求r bps的确保吞吐量，并且运输层协议能够确保可用吞吐量总是至少为r bps。这种确保吞吐量的服务将对众多应用程序有吸引力。例如，如果因特网电话应用程序对话音以32 kbps的速率进行编码，那么它也必须以这个速率向网络发送数据，并向接收应用程序交付数据。如果运输层协议不能提供这种吞吐量，那么该应用程序或以较低速率（并且接收带宽也必须足以维持这种较低的编码速率）进行编码，或应当放弃发送，这是因为对于这种因特网电话应用而言，只能达到所需吞吐量的一半是几乎没用的。具有吞吐量要求的应用程序称为**带宽敏感的应用**（bandwidth-sensitive application）。现在许多多媒体应用是带宽敏感的，但是未来的多媒体应用可能采用自适应编码技术，以便能用与当前可用吞吐量相匹配的速率进行编码。

带宽敏感的应用需要提供一定的吞吐量，而**弹性应用**（elastic application）能够根据需要充分利用可供使用的吞吐量。电子邮件、文件传输以及Web传输都属于弹性应用。当然，吞吐量是越多越好，有一句格言说，钱越多越好，人越瘦越美，人们永远不会嫌吞吐量太多！

3. 定时

运输层协议也能提供定时保证。如同吞吐量保证一样，定时保证可以以多种形式实现。例如，可以设置发送方注入进套接字中的每个比特到达接收方的套接字不迟于100 ms。这种服务对交互式实时应用程序非常适用，例如，对于因特网电话、虚拟环境、电视会议和多方游戏，为了有效性而对数据交付有严格的时间限制（参见第7章，[Gauthier 1999; Ramjee 1994]）。例如，因特网电话中较长的时延会导致会话中不自然的停顿；在多方游戏和虚拟互动环境中，在做出动作及看到来自环境（如来自位于端到端连接中另一端点的玩家）的响应之间，较长的时延会使游戏失去真实感。在非实时的应用中，较低的时延总要比较高的时延好，但对端到端的时延没有严格的约束。

4. 安全性

最后，运输层协议能够为应用程序提供一种或多种安全性服务。例如，在发送主机中，运输层协议能够加密由发送进程传输的所有数据；在接收主机中，运输层协议能够在将数据交付给接收进程之前解密这些数据。这种服务将对发送进程和接收进程保密，以防发送进程和接收进程以某种方式观察到数据。运输层协议也能提供除了机密性以外的其他安全性服务，包括数据完整性和端点鉴别，我们将在第8章中详细讨论这些主题。

2.1.4 因特网提供的运输服务

至此，我们已经讨论了计算机网络能够提供的一般运输服务。下面要更为具体地考察因

特网上的应用程序支持的类型。因特网（更一般地说是TCP/IP网络）上的应用使用了两个运输层协议：UDP和TCP。当你（作为一个软件开发者）创建一个新的因特网应用时，首先要做出的决定是，选择UDP还是选择TCP。每个协议为调用它们的应用程序提供了不同的服务集合。图2-4 显示了某些应用程序的服务要求。

应用	数据丢失	带宽	时间敏感
文件传输	不能丢失	弹性	不
电子邮件	不能丢失	弹性	不
Web文档	不能丢失	弹性（几 kbps）	不
因特网电话/视频会议	容忍丢失	音频（几 kbps~1 Mbps）	是，100 ms
		视频（10 kbps~5 Mbps）	
存储音频/视频	容忍丢失	同上	是，几秒
交互式游戏	容忍丢失	几 kbps~10 kbps	是，100 ms
即时讯息	不能丢失	弹性	是和不是

图2-4 部分网络应用的要求

1. TCP服务

TCP服务模型包括面向连接服务和可靠数据传输服务。当应用程序调用TCP协议作为它的运输层协议时，该应用程序就能获得TCP协议提供的这两种服务。

- 面向连接服务：使用TCP协议时，在应用层数据报文开始流动之前，其客户机程序和服务器程序之间互相交换运输层控制信息。这个所谓的握手过程提示客户机和服务器做好传输分组的准备。在握手阶段后，就在两个进程的套接字之间建立了一个TCP连接（TCP connection）。这个连接是全双工的，即连接双方的进程可以在此连接上同时进行报文收发。当应用程序结束报文发送时，必须拆除该连接。我们之所以称之为“面向连接”的服务，而不是“连接”服务，是因为两个进程间是以一种非常松散的方式进行连接的。在第3章中我们将详细讨论面向连接的服务，并分析它是如何实现的。
- 可靠数据传输服务：进行通信的进程依靠TCP协议，无差错、按适当顺序交付发送的数据。当应用程序的一端通过套接字传送一个字节流时，它能够依靠TCP协议将相同的字节流交付给接收方的套接字，而没有字节的丢失和冗余。

TCP协议还具有拥塞控制机制，这种服务不一定能为通信进程带来直接好处，但能为因特网带来整体好处。当发送方和接收方之间的网络出现拥塞时，TCP协议的拥塞控制机制会抑制发送进程（客户机或服务器）。如我们将在第3章中所见，TCP协议的拥塞控制试图限制每个TCP连接，使它们达到公平共享网络带宽的目的。对有最低要求带宽限制的实时音频/视频应用而言，抑制传输速率具有非常有害的影响。况且，实时应用是可以容忍数据丢失的，并不需要完全可靠的传输服务。出于这些原因，实时应用的开发者们通常将他们的应用放在UDP协议而不是TCP协议之上。

关注安全性

安全TCP

TCP和UDP都没有提供任何加密机制，这就是说发送进程传送至其套接字的数据，与经网络传送到目的进程的数据相同。因此，如果某发送进程以明文方式（即没有加密）将一个口令传送至它的套接字，该明文口令将经过发送方与接收方之间的所有链路传送，这就可能在任何中间链路被嗅探和发现。因为隐私和其他安全问题对许多应用而言已经成为至关重要的问题了，所以人们已经研制了TCP的加强版本，称为**安全套接字层**(Secure Socket Layer, SSL)。用SSL加强后的TCP不仅能够做传统的TCP所能做的一切，而且还提供了关键的进程到进程的安全性服务，包括加密、数据完整性和端点鉴别。需要注意的是，SSL不是独立于TCP和UDP的第三种因特网运输层协议，而只是对TCP的加强，这种加强是在应用层上实现的。特别是，如果一个应用程序要使用SSL的服务，它需要在其客户机端和服务器端中包括SSL代码（利用现有的、高度优化的库和类）。SSL有它自己的类似于传统的TCP套接字API的套接字API。当一个应用使用SSL时，发送进程向SSL套接字传递明文数据；发送主机中的SSL则加密该数据并将加密后的数据传递给TCP套接字。加密后的数据经因特网传送到接收进程中的TCP套接字。该接收套接字将加密后的数据传递给SSL，由其进行解密。最后，SSL通过其SSL套接字将明文数据传递给接收进程。我们将在第8章中讨论SSL的某些细节。

2. UDP服务

UDP是一种不提供不必要服务的轻量级运输层协议，它仅提供最小服务。UDP是无连接的，因此在两个进程通信前没有握手过程。UDP协议提供的是不可靠数据传输服务，也就是说，当进程通过UDP套接字发送报文时，UDP协议并不保证该报文能够被接收进程接收到。不仅如此，接收进程收到的报文也可能是乱序到达的。

UDP没有拥塞控制机制，所以发送端可以以任何速率向其下面的层（网络层）注入数据。(然而，注意到实际的端到端吞吐量可能小于这一速率，这可能是因为中间链路的带宽受限或因为拥塞而导致的)。因为实时应用通常可以忍受一定的数据丢失，同时有最低速率的要求，所以开发者有时在这种应用程序中会选择使用UDP协议，以避开TCP协议的拥塞控制机制和分组开销。另一方面，因为许多防火墙被配置为阻塞（多数类型的）UDP流量，所以设计者在多媒体和实时应用程序中越来越多地选择使用TCP[Sripanidkulchai 2004]。

3. 因特网运输层协议所不提供的服务

我们已经从4个方面组织了可能的运输层协议服务：可靠数据传输、吞吐量、定时和安全性。TCP和UDP提供了哪些服务呢？我们已经注意到TCP提供了可靠的端到端数据传输服务，并且我们也知道TCP在应用层可以很容易地通过SSL来提供安全服务。但在对TCP和UDP的描述中，缺少对吞吐量和定时保证的讨论，即目前的因特网运输层协议并没有提供这两种服务。这是否意味着因特网电话等时间敏感应用就不能运行在今天的因特网上呢？答案显然不是，因为在因特网上时间敏感应用已经存在多年了。这些应用运行得很好是因为设计时尽最大可能弥补了这些保证的缺乏。我们将在第7章中研究这些设计的技巧。然而，聪明的设计对时延过大仍是有限制的，就像我们在因特网中常遇到的情况那样。总之，现在的

因特网通常能够为时间敏感应用提供使之满意的服务，但是不能提供任何有关定时或带宽的保证。

图2-5指出了一些流行的因特网应用所使用的运输层协议。可以看到，电子邮件、远程终端访问、Web、文件传输都是使用TCP协议，它们选择TCP协议的最主要原因是TCP协议提供了可靠数据传输服务，能够保证所有数据最终到达目的地。我们也看到因特网电话通常运行在UDP协议上。每个因特网电话的站点在网络上发送数据有最低速率的要求（参见图2-4中的实时音频），因此，用UDP协议实现的可能性大于用TCP协议实现的可能性。另外，因特网电话应用可以容忍一定的数据丢失，因此它也不需要TCP协议提供的可靠数据传输服务。

应用	应用层协议	支撑的运输层协议
电子邮件	SMTP [RFC 2821]	TCP
远程终端访问	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
文件传输	FTP [RFC 959]	TCP
流媒体	HTTP（如YouTube）、RTP	TCP或UDP
因特网电话	SIP、RTP或专用（如Skype）	通常用UDP

图2-5 流行的因特网应用及其应用层协议和支撑的运输层协议

4. 进程寻址

上面讨论了两个通信进程之间的运输服务。但是，一个进程如何使用这些服务来指示它要与哪个进程进行通信呢？例如，位于美国马萨诸塞州阿默斯特的一台主机上的一个进程如何指定它要与位于泰国曼谷的一台主机上的一个特定进程进行通信呢？为了识别接收进程，需要定义两种信息：①该主机的名称或地址，②用来指定目的主机上接收进程的标识。

在因特网中，主机是用IP地址（IP address）进行标识的。我们将在第4章深入讨论IP地址。此时，我们知道IP地址是用来唯一标识主机的32比特数就足够了（然而，如第4章要讲到的，网络地址转换（NAT）的广泛部署意味着实际中32比特IP地址不仅仅唯一地用于寻址一台主机）。

除了知道报文去往的目的主机的IP地址外，发送程序也必须识别运行在主机上的接收进程。之所以需要该信息，是因为通常在一台主机上能够运行许多网络应用程序。目的地端口号（port number）就是服务于这个目的的。已经给流行的应用程序分配了特定的端口号。例如，Web服务进程用的是80号端口。邮件服务进程（使用SMTP协议）用的是25号端口。所有因特网标准协议所使用的周知端口的列表可在http://www.iana.org找到。当开发者创建了一个新的网络应用程序时，必须为该应用程序分配一个新的端口号。我们将在第3章中详细分析端口号。

2.1.5 应用层协议

我们刚刚学习了通过把报文发送到套接字中来使网络进程间相互通信。但是如何构造这些报文？这些报文中各个字段的含义是什么？这些问题将我们带进应用层协议的范围。**应用层协议**（application-layer protocol）定义了运行在不同端系统上的应用程序进程如何相互传递报文。特别是应用层协议定义了：

- 交换的报文类型，如请求报文和响应报文。

- 各种报文类型的语法，如报文中的各个字段及其详细描述。
- 字段的语义，即包含在字段中的信息的含义。
- 进程何时、如何发送报文及对报文进行响应的规则。

有些应用层协议是由RFC文档定义的，因此它们位于公共领域。例如，Web的应用层协议HTTP（超文本传输协议[RFC 2616]）就作为一个RFC供大家使用。如果浏览器开发者遵从HTTP RFC规则，所开发出的浏览器就能访问任何遵从该文档标准的Web服务器并获取相应的Web页面。还有很多其他应用层协议是专用的，不能随意应用于公共领域。例如，很多现有的P2P文件共享系统使用的是专用应用层协议。

区分网络应用和应用层协议是很重要的。应用层协议只是网络应用的一部分。我们来看一些例子。Web应用是一种客户机/服务器应用程序，它允许客户机按照需求从Web服务器获得文档。Web应用有很多组成部分，包括文档格式的标准（即HTML）、Web浏览器（如Firefox和Microsoft Internet Explorer）、Web服务器（如Apache、 Microsoft服务器程序），以及一个应用层协议。Web的应用层协议是HTTP，它定义了在浏览器和Web服务器之间传输的报文格式和序列。因此，HTTP只是Web应用的一个部分。另外一个例子是因特网电子邮件应用，它也有很多组成部分，包括能容纳用户邮箱的邮件服务器程序，允许用户读取和生成邮件的邮件阅读器，定义电子邮件报文结构的标准，定义如何在服务器之间及服务器与邮件阅读器间传送报文的应用层协议，定义如何解释邮件报文的某个部分（如邮件报文的首部）的内容的应用层协议。用于电子邮件的主要应用层协议就是SMTP（简单邮件传输协议[RFC 2821]）。因此，SMTP也只是电子邮件应用的一个部分（虽然是较大的一部分）。

2.1.6 本书涉及的网络应用

每天都有新的公共领域或者专用领域的因特网应用被开发出来。我们不愿像百科全书一样涉及大量的因特网应用，而是选择其中的几种重要而流行的应用。本章我们详细讨论5种重要的应用：Web、文件传输、电子邮件、目录服务和P2P。我们首先讨论Web应用，不仅因为它极为流行，而且因为它的应用层协议HTTP相对比较简单并且易于理解。讨论完Web，我们简要地讨论FTP，因为它与HTTP形成了很好的对照。接下来讨论电子邮件应用，这是因特网上最早的招人喜爱的应用程序。说电子邮件比Web更复杂，是因为它使用了多个而不是一个应用层协议。在电子邮件后面，我们讨论DNS，它为因特网提供目录服务。大多数用户不直接与DNS打交道，而是通过其他的应用（包括Web、文件传输和电子邮件）间接使用它。DNS很好地说明了一种核心的网络功能（网络名字到网络地址的转换）是怎样在因特网的应用层实现的。最后，讨论P2P应用，包括文件分发、分布式搜索和IP电话。

2.2 Web应用和HTTP协议

20世纪90年代以前，因特网的主要使用者是研究人员、学者和大学生，他们利用因特网登录远程主机，在本地主机和远程主机之间传输文件，收发新闻，收发电子邮件。尽管这些应用非常有用（并且继续如此），但是本质上因特网只是在科研领域内使用。到了20世纪90年代初期，一个主要的新型应用（即万维网，World Wide Web）登上了舞台[Berners-Lee 1994]。Web是一个引起公众注意的因特网应用，它极大地改变了人们与工作环境内外交流的方式。它将因特网从只是很多数据网之一的地位提升为仅有的一个数据网。

也许对大多数用户来说，最具有吸引力的就是Web的按需操作。当用户需要时，就能得

到他想要的内容。这不同于广播和电视，用户只能收听、收看内容提供者提供的节目。除了可以按需操作以外，Web还有很多让人们喜爱的特性。任何人要在Web上发布信息都非常简单，只需要极低的费用就能成为出版人。超链接和搜索引擎帮助我们在Web站点的海洋里畅游。图形化的界面刺激着我们的感官。表单、Java小程序和很多其他的装置，使我们可以与Web页面和Web站点进行交互。越来越多的Web站点为存储在因特网上的大量视频和音频素材提供了菜单界面，这样一来，多媒体也可以实现按需访问了。

2.2.1 HTTP概况

Web的应用层协议是**超文本传输协议**（HyperText Transfer Protocol，HTTP），它是Web的核心，在[RFC 1945]和[RFC 2616]中进行了定义。HTTP协议由两部分程序实现：一个客户机程序和一个服务器程序，它们运行在不同的端系统中，通过交换HTTP报文进行会话。HTTP定义了这些报文的格式以及客户机和服务器是如何进行报文交换的。在详细介绍HTTP协议之前，有必要先了解一下Web中常用的术语。

Web页面（Web page，也叫文档）是由对象组成的。**对象**（object）简单来说就是文件，如HTML文件、JPEG图形文件、Java小程序或视频片段文件，这些文件可通过一个URL地址寻址。多数Web页面含有一个**基本HTML文件**（base HTML file）以及几个引用对象。例如，如果一个Web页面包含HTML文本和5个JPEG图形文件，那么这个Web页面有6个对象：一个基本HTML文件加5个图形。在基本HTML文件中通过对象的URL地址对对象进行引用。每个URL地址由两部分组成：存放对象的服务器主机名和对象的路径名。例如，URL地址http://www.someSchool.edu/someDepartment/picture.gif中的www.someSchool.edu就是主机名，/someDepartment/picture.gif就是路径名。因为Web浏览器（Web browser，如Internet Explorer和Firefox）实现了HTTP的客户机端，所以在Web环境中，我们经常交替使用“浏览器”和“客户机”这两个术语。流行的Web浏览器包括Netscape Communicator和Microsoft Internet Explorer。**Web服务器**（Web server）用于存储Web对象，每个对象由URL寻址。Web服务器实现了HTTP的服务器端，流行的Web服务器程序有Apache和Microsoft Internet Information Server。

HTTP定义了Web客户机是如何向Web服务器请求Web页面，以及服务器如何将Web页面传送给客户机的。我们下面详细地讨论客户机和服务器之间的交互过程，其基本思想在图2-6中进行了描述。当用户请求一个Web页面（如点击一个超链接）时，浏览器向服务器发出对该页面中所包含对象的HTTP请求报文，服务器接受请求并用包含这些对象的HTTP响应报文进行响应。

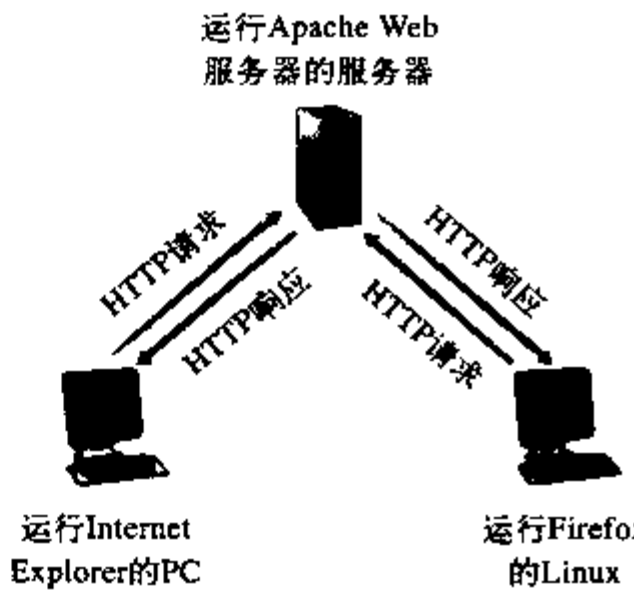


图2-6 HTTP的请求-响应行为

HTTP使用TCP（而不是UDP）作为它的支撑运输层协议。HTTP客户机发起一个与服务器的TCP连接，一旦连接建立，浏览器和服务器进程就可以通过套接字接口访问TCP。就像2.1节中描述的那样，客户机端的套接字接口是客户机进程与TCP连接之间的门，服务器端的套接字接口则是服务器进程与TCP连接之间的门。客户机从套接字接口发送HTTP请求报文和接收HTTP响应报文。类似地，服务器也是从套接字接口接收

HTTP请求报文和发送HTTP响应报文。一旦客户机发送了一个请求报文，该报文就“脱离客户机控制”并“进入TCP的控制”。2.1节讲过，TCP为HTTP提供可靠数据传输服务。这意味着，一个客户机进程发出的每个HTTP请求报文最终都能完整地到达服务器；类似地，服务器进程发出的每个HTTP响应报文最终也都能完整地到达客户机。从这里我们看到了分层体系结构最大的优点，即HTTP协议不用担心数据丢失，也不用担心TCP是如何从网络的数据丢失和乱序故障中恢复的，那是TCP以及协议栈底层协议的工作。

注意到下列现象是很重要的：服务器向客户机发送被请求的文件时，并不存储任何关于该客户机的状态信息。假如某个特定的客户机在短短的几秒钟内两次请求同一个对象，服务器并不会因为刚刚为该用户提供了该对象就不再做出反应，而是重新发送该对象，就像该服务器已经完全忘记不久之前所做过的事一样。因为一个HTTP服务器并不保存关于客户机的任何信息，所以我们说HTTP是一个**无状态协议**（stateless protocol）。我们还注意到，Web使用了客户机/服务器应用程序体系结构（如2.1节所述）。Web服务器总是打开的，具有一个固定的IP地址，它服务于数以百万计的不同浏览器。

2.2.2 非持久连接和持久连接

在许多因特网应用中，客户机和服务器进行长时间通信，其中客户机发出一系列请求，服务器对每个请求进行响应。根据不同的应用程序以及应用程序使用的方式，这一系列请求可以周期性地一个接一个发出，也可以间断性地发出。当这种客户机/服务器的交互运行于TCP协议之上时，应用程序的研制者需要确定每个请求/响应对是经一个单独的TCP连接发送，还是所有的请求及相应的响应经相同的TCP连接发送。如果采用前一种方法，该应用程序被称为使用**非持久连接**（non-persistent connection）；如果采用后一种方法，该应用程序被称为使用**持久连接**（persistent connection）。为了深入地理解这种设计问题，我们研究在特定的应用程序（即HTTP）中持久连接的优点和缺点。HTTP既可以使用非持久连接，也可以使用持久连接，默认方式下HTTP使用持久连接。

1. 非持久连接

我们看看在非持久连接情况下，从服务器向客户机传送一个Web页面的步骤。假设该页面含有一个基本HTML文件和10个JPEG图形，并且这11个对象位于同一台服务器上。该HTML文件的URL为：http://www.someSchool.edu/someDepartment/home.index。

我们看看发生了什么：

1）HTTP客户机进程在端口号80发起一个到服务器www.someSchool.edu的TCP连接，该端口号是HTTP的默认端口。客户机和服务器上分别有一个套接字与该连接相关联。

2）HTTP客户机经它的套接字向服务器发送一个HTTP请求报文。请求报文中包含了路径名/someDepartment/home.index（后面我们会详细讨论HTTP报文）。

3）HTTP服务器进程经它的套接字接收该请求报文，从其存储器（RAM或磁盘）中检索出对象someDepartment/home.index，在一个HTTP响应报文中封装对象，并通过其套接字向客户机发送响应报文。

4）HTTP服务器进程通知TCP断开该TCP连接。（但是直到TCP确认客户机已经完整地收到响应报文为止，它才会真正中断连接。）

5）HTTP客户机接收响应报文，TCP连接关闭。报文指出封装的对象是一个HTML文件，客户机从响应报文中提取出该文件，检查该文件，得到对10个JPEG图形的引用。

6）对每个引用的JPEG图形对象重复前4步。

当浏览器收到Web页面后，把它显示给用户。两个不同的浏览器也许会以不同的方式解释（即向用户显示）该页面。HTTP协议并不管客户机如何解释一个Web页面。HTTP规约（[RFC 1945]和[RFC 2616]）仅定义了在HTTP客户机程序与HTTP服务器程序之间的通信协议。

上面的步骤举例说明了非持久连接的使用，每个TCP连接在服务器返回对象后关闭，即该连接并不为其他的对象而持续下来。注意到每个TCP连接只传输一个请求报文和一个响应报文。因此在本例中，用户请求该Web页面，要建立11个TCP连接。

在上面描述的步骤中，我们有意没有明确客户机获得这10个JPEG图形对象，是使用10个串行的TCP连接还是使用并行的TCP连接。事实上，用户可以设置浏览器以控制并行度。默认方式下，大部分浏览器打开5～10个并行的TCP连接，而每个连接处理一个请求-响应事务。还可以将最大并行连接数设置为1，这样10个连接就会以串行方式建立。我们在下一章会看到，使用并行连接可以缩短响应时间。

在继续讨论之前，我们来估算一下从客户机请求基本HTML文件开始到用户收到整个文件为止所花费的时间。为此，我们给出往返时间（Round-Trip Time, RTT）的定义，即一个小分组从客户机到服务器再回到客户机所花费的时间。RTT包括分组传播时延、分组在中间路由器和交换机上的排队时延以及分组处理时延（这些在1.4节已经讨论过）。接下来，我们看看当用户点击超链接时会发生什么。如图2-7所示，浏览器在浏览器和Web服务器之间发起一个TCP连接，这涉及一个“三次握手”过程，即客户机向服务器发送一个小TCP报文段，服务器用一个小TCP报文段做出确认和响应，最后，客户机向服务器返回确认。一个RTT等于三次握手中前两个部分所耗费的时间。完成了三次握手的前两个部分后，客户机将三次握手的第三个部分（确认）与一个HTTP请求报文结合起来发送到该TCP连接。一旦该请求报文到达服务器，服务器向该TCP连接发送HTML文件。该HTTP请求/响应又耗掉一个RTT。因此，粗略地讲，总的响应时间就是两个RTT加上服务器传输HTML文件的时间。

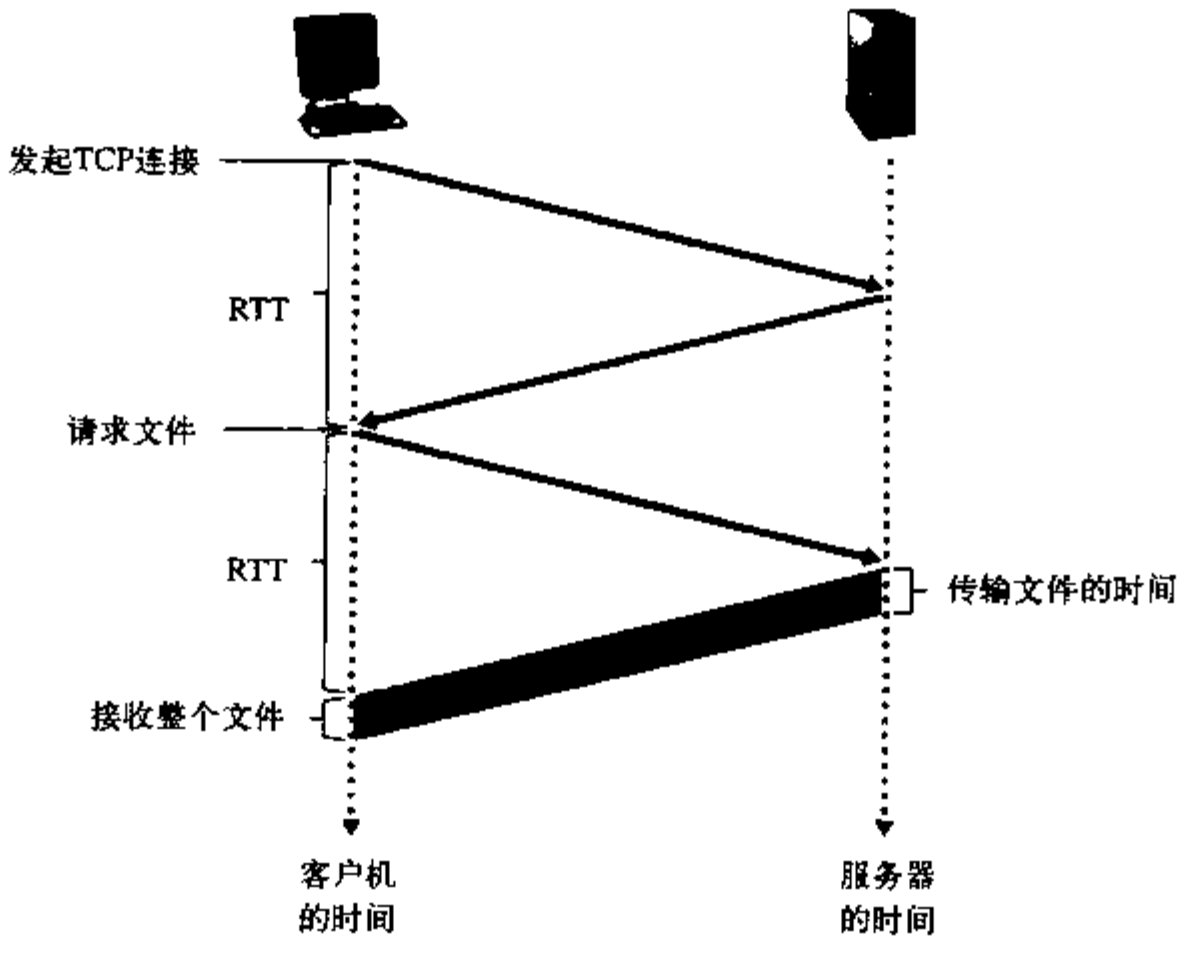


图2-7 请求并接收一个HTML文件所需的时间估算

2. 持久连接

非持久连接有一些缺点。首先，必须为每一个请求的对象建立和维护一个全新的连接。对于每个这样的连接，在客户机和服务器都要分配TCP的缓冲区和变量，这给服务器带来了严重的负担，因为一台Web服务器可能同时服务于数以百计的客户机请求。其次，就像我们刚描述的那样，每一个对象的传输时延为两个RTT，即一个RTT用于建立TCP，另一个RTT用于请求和接收一个对象。

在持久连接的情况下，服务器在发送响应后保持该TCP连接打开。在相同的客户机与服务器之间的后续请求和响应报文可通过相同的连接进行传送。特别是一个完整的Web页面（如上例中的基本HTML文件加上10个图形）可以用单个持久TCP连接进行传送。更有甚者，位于同一台服务器的多个Web页面在从该服务器发送给同一个客户机时，可以在单个持久TCP连接上进行。对这些对象的请求可一个接一个地发出，而不必等待未决请求的回答（流水线）。一般来说，如果一个连接经过一定时间间隔（一个可配置的超时间隔）仍未被使用，HTTP服务器就关闭该连接。HTTP的默认模式使用了流水线方式的持久连接。我们把量化比较持久连接和非持久连接性能的任务留作第2、3章的课后习题，有兴趣的读者可参考阅读[Heidemann 1997；Nielsen 1997]。

2.2.3 HTTP报文格式

HTTP规约[RFC 2616]包含了对HTTP报文格式的定义。HTTP报文有两种：请求报文和响应报文，下面讨论这两种报文。

1. HTTP请求报文

下面是一个典型的HTTP请求报文：

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language: fr
```

通过仔细观察这个简单的请求报文就能知道很多东西。首先，我们看到该报文是用普通的ASCII文本书写的，这样有一定计算机知识的人就能够阅读它。其次，我们看到该报文含有5行，每行用一个回车换行符结束，最后一行后跟有附加的回车换行符。该报文只有5行，而实际的请求报文可以有更多行或仅有一行。HTTP请求报文的第一行叫做**请求行**（request line），其后继的行叫做**首部行**（header line）。请求行有3个字段：方法字段、URL字段和HTTP协议版本字段。方法字段可以取值GET、POST、HEAD、PUT和DELETE。绝大部分的HTTP请求报文使用GET方法。当浏览器请求一个对象时，使用GET方法，在URL字段填写该对象的URL地址。在本例中，浏览器请求对象/somedir/page.html。其版本字段是自解释的，在本例中，浏览器实现的是HTTP/1.1版本。

现在我们看看本例的首部行。首部行Host:www.someschool.edu定义了目标所在的主机。你也许认为该首部行是不必要的，因为在该主机中已经有一个TCP连接了。但是，如我们将在2.2.5节中所见，该首部行提供的信息是Web代理高速缓存所要求的。通过包含Connection: close首部行，浏览器告诉服务器不希望麻烦地使用持久连接，它要求服务器在发送完被请求的对象后就关闭连接。User-agent:首部行用来定义用户代理，即向服务器发送请求的浏览器的类型。这里浏览器的类型是Mozilla/4.0，即Netscape浏览器。这

个首部行是有用的，因为服务器可以正确地为不同类型的用户代理实际发送相同对象的不同版本。（每个版本都由相同的URL处理。）最后，Accept-language:首部行表示用户想得到该对象的法语版本（如果服务器中有这样的对象），否则，使用服务器的默认版本。Accept-language:首部行仅是HTTP中众多可选内容协商首部之一。

在看过一个例子之后，我们再来看看请求报文的通用格式。如图2-8所示，我们看到该通用格式与前面的例子密切对应。你可能已经注意到，在首部行（和附加的回车换行符）后有一个“实体主体”(entity body)。使用GET方法时实体主体为空，而使用POST方法时才使用。HTTP客户机常常在用户提交表单时使用POST方法，例如，用户向搜索引擎提供搜索关键词。在使用POST方法的报文中，用户仍可以向服务器请求一个Web页面，但Web页面的特定内容依赖于用户在表单字段中输入的内容。当方法字段的值为POST时，实体主体中包含的就是用户在表单字段中输入的值。

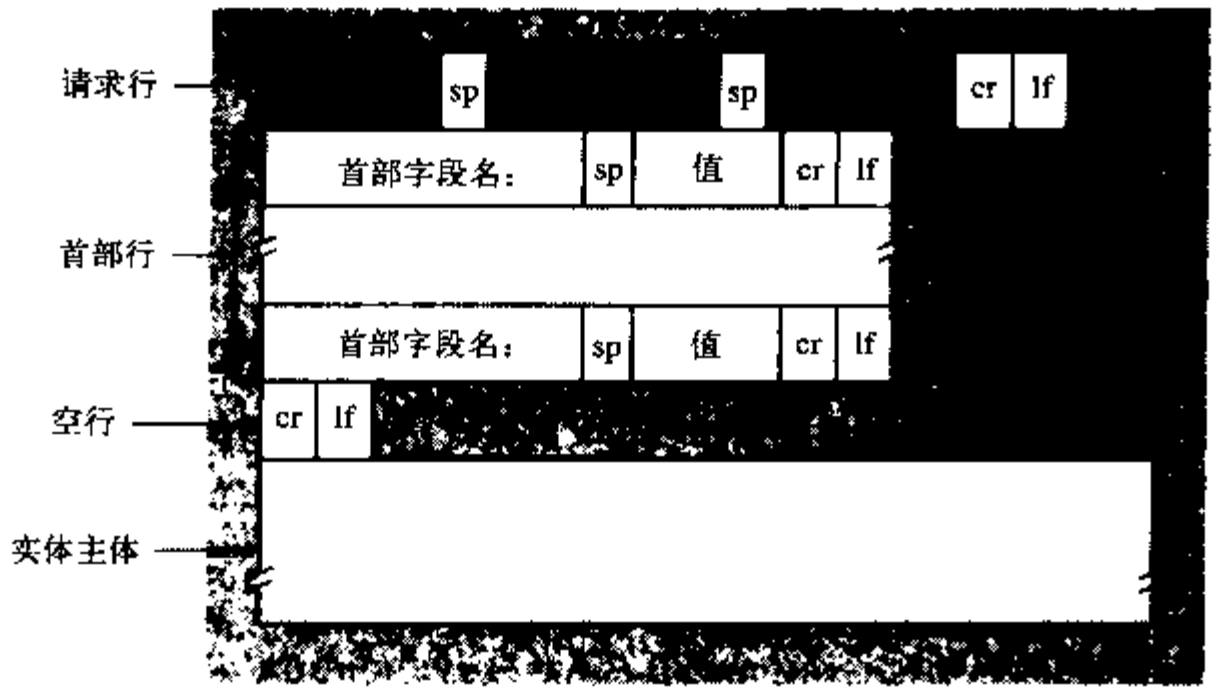


图2-8 HTTP请求报文的通用格式

当然，如果不提“用表单生成的请求报文不需要使用POST方法”这一点，那将是失职。HTML表单经常使用GET方法，将输入数据（在表单字段中）传送到正确的URL。例如，一个表单使用GET方法，它有两个字段，分别填写的是monkeys和bananas，那么得到的URL结构为www.somesite.com/animalsearch?monkeys&bananas。在日复一日的网上冲浪中，你也许已经留意到了这种扩展的URL。

HEAD方法类似于GET方法。当服务器收到使用HEAD方法的请求时，会用一个HTTP报文进行响应，但是并不返回请求对象。应用程序开发者常用HEAD方法进行故障跟踪。PUT方法常与Web发行工具联合使用，用户利用它来将对象上传到指定的Web服务器上指定的路径（目录）。PUT方法也被应用程序用来向Web服务器上传对象。利用DELETE方法，用户或者应用程序可以删除Web服务器上的对象。

2. HTTP响应报文

我们再来看看下面这个典型的HTTP响应报文。可以说，该响应报文是对上面讨论的例子中请求报文的响应。

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 03 Jul 2003 12:00:15 GMT
```

```
Server: Apache/1.3.0 (Unix)
Last-Modified: Sun, 6 May 2007 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```

我们仔细看一下这个响应报文。它分成三个部分：一个初始**状态行**（status line）、6个**首部行**（header line），然后是**实体主体**（entity body）。实体主体部分是报文的主体，即它包含了所请求的对象本身（表示为data data data data data...）。状态行有3个字段：协议版本、状态码和相应状态信息。在这个例子中，状态行指示服务器使用的协议是HTTP/1.1，并且一切正常（即服务器已经找到并正在发送所请求的对象）。

我们现在来看看首部行。服务器用Connection: close首部行告诉客户机在报文发送完后关闭了该TCP连接。Date:首部行指示服务器产生并发送该响应报文的日期和时间。值得一提的是，这个时间不是指对象创建或者最后修改的时间，而是服务器从它的文件系统中检索到该对象、插入到响应报文并发送该响应报文的时间。Server:首部行表明该报文是由一个Apache Web服务器产生的，它类似于HTTP请求报文中的User-agent:首部行。Last-Modified:首部行指示了对象创建或者最后修改的日期和时间。Last-Modified:首部行对既可能在客户机也可能在网络缓存服务器上的对象缓存来说非常重要。我们将详细讨论缓存服务器（也叫代理服务器）。Content-Length:首部行表明了被发送对象的字节数。Content-Type:首部行指示了实体主体中的对象是HTML文本。（对象类型应该正式地用Content-Type:首部行而不是文件扩展名来指示。）

看过这个例子后，我们再来查看响应报文的通用格式（如图2-9所示）。该通用格式与前面例子中的响应报文相对应。我们补充说明一下状态码和它们对应的短语，它们指示了请求的结果。一些常见的状态码和相关的短语有：

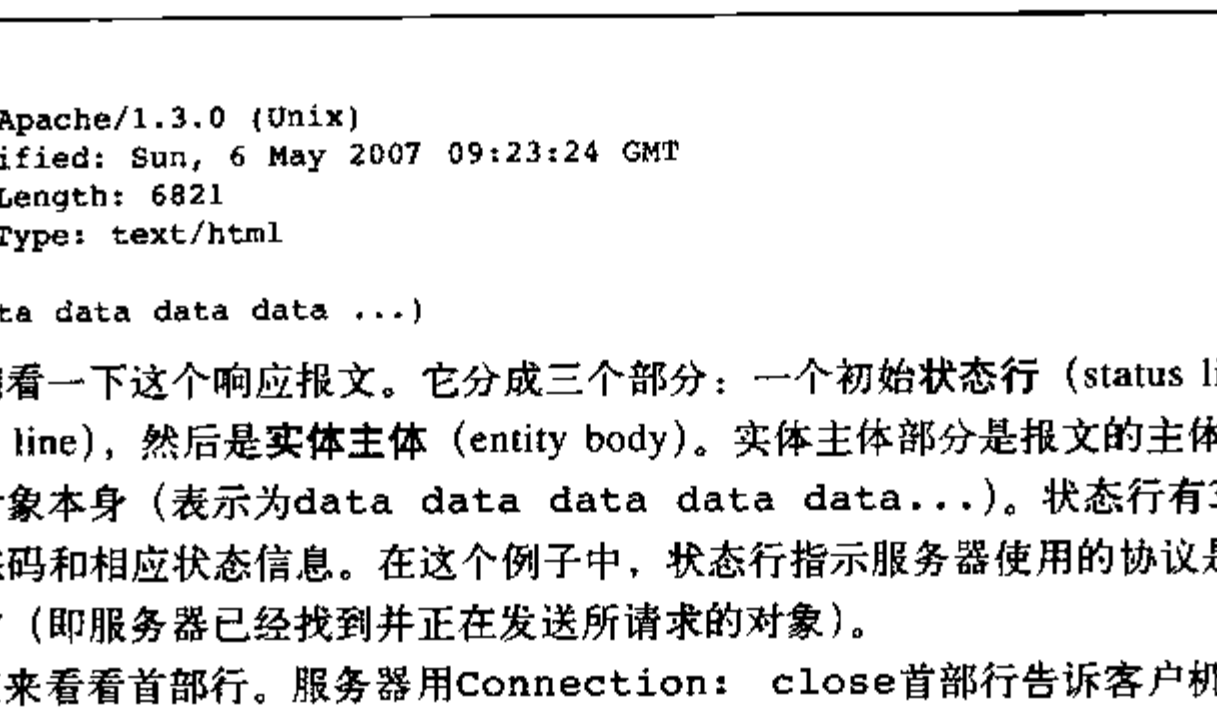


图2-9 HTTP响应报文的通用格式

- 200 OK：请求成功，信息包含在返回的响应报文中。
- 301 Moved Permanently：请求的对象已经被永久转移了，新的URL定义在响应报文的Location:首部行中指定。客户机软件自动用新的URL获取该对象。
- 400 Bad Request：一个通用差错代码，指示该请求不能被服务器理解。
- 404 Not Found：被请求的文档不在服务器上。

• 505 HTTP Version Not Supported：服务器不支持请求报文使用的HTTP协议版本。

你想看一下真实的HTTP响应报文吗？这正是我们所推荐的，而且也很容易做到。首先用Telnet工具登录到你最喜欢的Web服务器上，接下来输入一个只有一行的请求报文去请求放在该服务器上的某些对象。例如，假设你在命令行提示下输入：

```
telnet cis.poly.edu 80

GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

（在输入第二行后连续敲两次回车。）这会打开一个到主机cis.poly.edu的80端口的TCP连接，并发送一个HTTP请求报文。你将会看到一个携带Ross教授主页的基本HTML文件的响应报文。如果你只是想看一下HTTP协议的报文行，而不是获取对象本身，那么可以用HEAD代替GET。最后，用/~banana/代替/~ross/，看看你得到的响应报文是什么样的。

在本节中，我们讨论了HTTP请求报文和响应报文的部分首部行。HTTP规约中定义了很多可以被浏览器、Web服务器和Web缓存服务器插入的首部行，我们只提到了其中的小部分，在2.2.5节中我们讨论网络Web缓存时还会提到几个。一本可读性很强的文献是[Krishnamurty 2001]，它对HTTP协议（包括它的首部行和状态码）进行了广泛讨论。开发人员可参阅[Luotonen 1998]。

浏览器是如何决定在一个请求报文中包含哪些首部行的呢？Web服务器又是如何决定在一个响应报文中包含哪些首部行呢？浏览器产生哪些首部行与很多因素有关，包括浏览器的类型和协议版本（例如，用HTTP/1.0协议实现的浏览器将不会产生HTTP/1.1中定义的首部行）、用户对浏览器的配置（如喜好的语言）、浏览器当前是否有一个缓存着的但是可能超期的对象版本以及配置，所有这些都会影响响应报文中包含的首部行。

2.2.4 用户与服务器的交互：cookie

我们前面提到了HTTP服务器是无状态的。这简化了服务器的设计，并且允许工程师们去开发可以同时处理数以千计TCP连接的高性能的Web服务器。然而一个Web站点通常希望能够识别用户，既可能是因为服务器想限制用户的访问，又可能是因为它想把内容与用户身份关联起来。为此，HTTP使用了cookie。cookie在RFC 2109中进行定义，它允许站点跟踪用户。目前，大多数商务Web站点都使用了cookie。

如图2-10所示，cookie技术有4个组成部分：①在HTTP响应报文中有一个cookie首部行；②在HTTP请求报文中有一个cookie首部行；③在用户端系统中保留有一个cookie文件，由用户的浏览器管理；④在Web站点有一个后端数据库。根据图2-10，我们通过一个典型的例子来看看cookie的工作过程。假设Susan总是从她的家用PC机使用Internet Explorer上网，她第一次访问Amazon.com。当请求报文到达Amazon Web服务器时，该Web站点将产生一个唯一识别码，并以此作为索引在它的后端数据库中产生一个表项。接下来Amazon Web服务器用一个包含Set-cookie:首部行的HTTP响应报文对Susan的浏览器进行响应，其中Set-cookie:首部行含有识别码。例如，该首部行可能是

```
Set-cookie: 1678
```

当Susan的浏览器收到了该HTTP响应报文时，它会看到该Set-cookie:首部。该浏览器在它管理的特定cookie文件中添加一行，其中包含该服务器的主机名和Set-cookie:首部中的识别码。注意到该cookie文件已经有了用于eBay的表项，因为Susan曾访问过该站点。当

Susan继续浏览Amazon网站时，每请求一个Web页面，其浏览器就会从它的cookie文件中获取这个网站的识别码，并放到HTTP请求报文中含有识别码的cookie首部行中。特别是，发往Amazon服务器的每个HTTP请求报文都包括以下首部行：

```
Cookie: 1678
```

在这种方式下，Amazon服务器可以跟踪Susan在该站点的活动。Amazon Web站点并不需要知道Susan的名字，但它确切地地知道用户1678访问了哪些页面，按照什么顺序，在什么时间！Amazon使用cookie来提供购物车服务，即它为Susan维护所有想购买物品的列表，这样在Susan结束会话时可以一起付费。

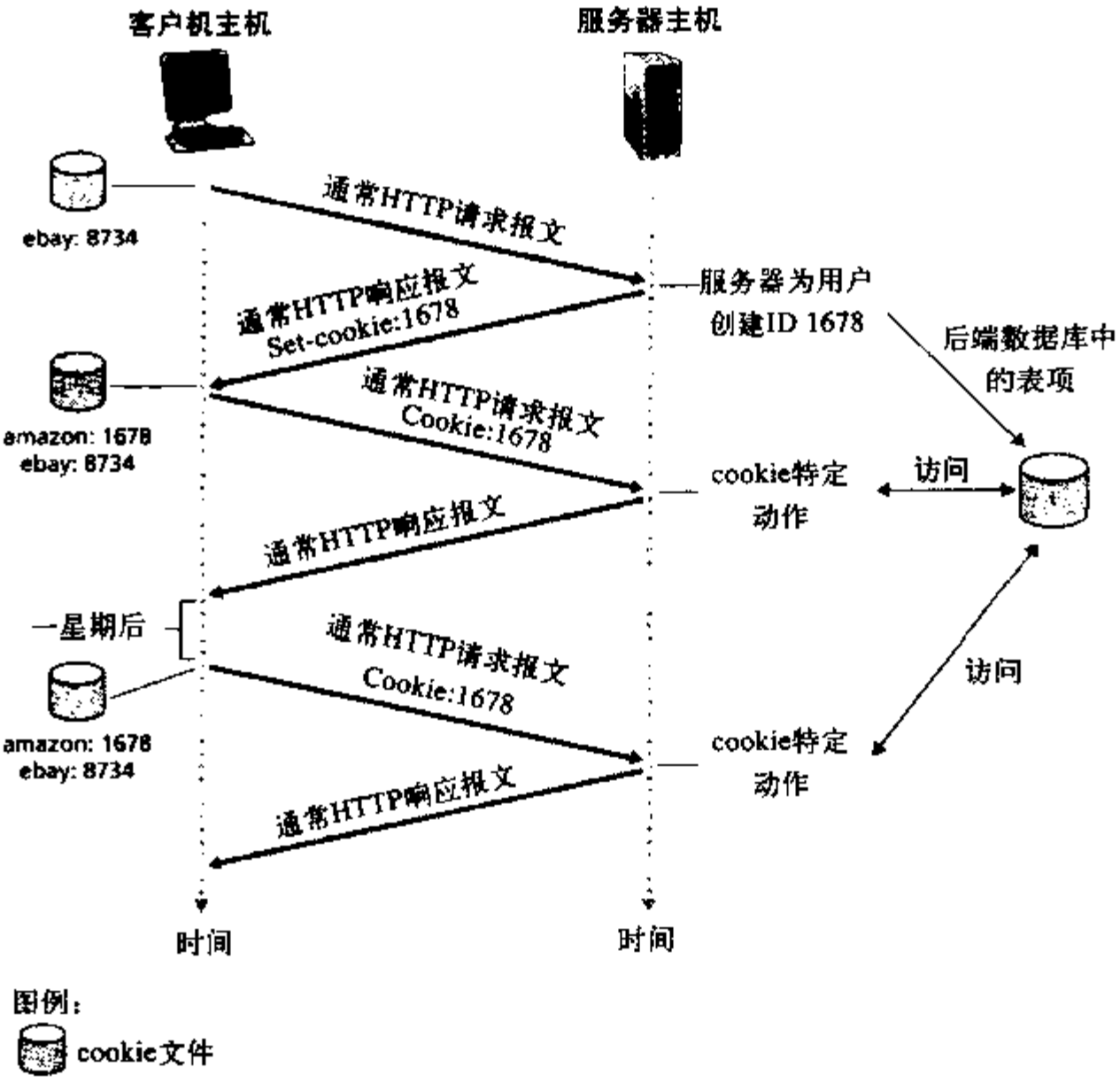


图2-10 用cookie保持用户状态

如果一个星期后Susan再次访问Amazon站点，她的浏览器会在其请求报文中继续使用首部行Cookie:1678。Amazon将根据Susan过去访问Amazon的记录向她推荐产品。如果Susan在Amazon注册过，即提供了她的全名、电子邮件地址、邮政地址和信用卡账号，则Amazon在其数据库中记录了这些信息，将她的全名与识别码相关联（以及过去访问过的所有页面）。这就解释了Amazon等电子商务网站如何实现“点击购物”（one-click shopping），即当Susan在后继的访问中购买物品时，她不用重新输入姓名、信用卡账号或者地址等信息。

从上述讨论中我们看到，cookie可以用于标识用户。用户首次访问站点时，可能需要提供一个用户标识（可能是名字）。在后继访问中，浏览器向服务器传递一个cookie首部，供服务器识别该用户。因此，cookie可以在无状态的HTTP上建立一个用户会话层。例如，当用户登录一个基于Web的电子邮件系统（如Hotmail）时，浏览器向服务器发送cookie信息，允许该

服务器通过用户与应用程序之间的会话对用户进行验证。

尽管cookie常常能简化用户的因特网购物活动，但是它的使用仍有很大的争议，因为它们被看作是对用户隐私的一种侵害。就像上面所讨论的，结合cookie和用户提供的账户信息，Web站点可以知道许多有关用户的信息，并可能将这些信息出卖给第三方。Cookie Central [Cookie Central 2007]包括了很多对cookie的争论。

2.2.5 Web缓存

Web缓存器（Web cache）也叫**代理服务器**（proxy server），它是能够代表初始Web服务器来满足HTTP请求的网络实体。Web缓存器有自己的磁盘存储空间，并在该存储空间中保存最近请求过的对象的拷贝。如图2-11所示，可以配置用户的浏览器，使得用户的所有HTTP请求首先指向Web缓存器。一旦配置了浏览器，每个浏览器对一个对象的请求首先被定向到Web缓存器。举例来说，假设浏览器正在请求对象`http://www.someschool.edu/campus.gif`，将会发生如下情况：

1）浏览器建立一个到Web缓存器的TCP连接，并向Web缓存器中的对象发送一个HTTP请求。

2）Web缓存器检查本地是否存储了该对象拷贝。如果有，Web缓存器就用HTTP响应报文向客户机浏览器返回该对象。

3）如果Web缓存器没有该对象，它就与该对象的初始服务器（如`www.someschool.edu`）打开一个TCP连接。Web缓存器则在TCP连接上发送获取该对象的HTTP请求。在收到请求后，初始服务器向Web缓存器发送具有该对象的HTTP响应。

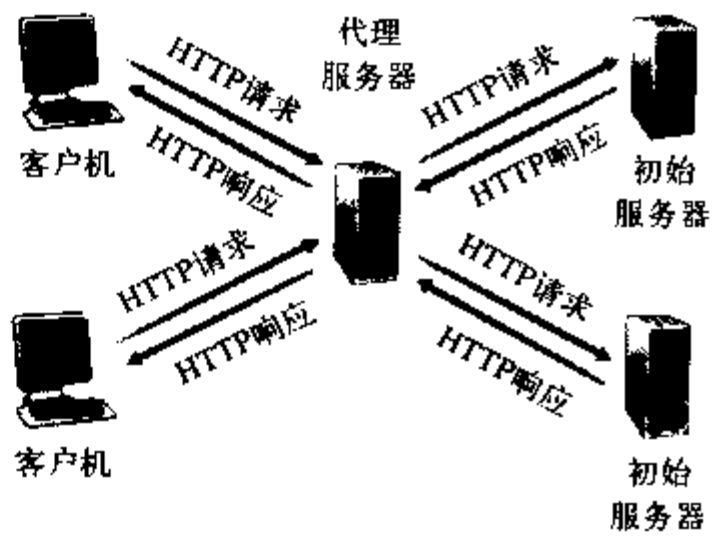


图2-11 客户机通过Web缓存器请求对象

4）当Web缓存器接收该对象时，它在本地存储空间存储了一份拷贝，并用HTTP响应报文向客户机的浏览器发送该拷贝（通过已经建立在客户机浏览器和Web缓存器之间的TCP连接）。

注意到Web缓存器既是服务器又是客户机。当它接收浏览器的请求并发回响应时，它是服务器；当它向初始服务器发出请求并接收响应时，它是客户机。

一般而言，Web缓存器由ISP购买并安装。例如，一所大学可能在它的校园网上安装一台缓存器，并且将所有校园网上的用户浏览器配置为指向它。或者，一个主要的居民区ISP（例如AOL）可能在它的网络上安装一台或多台Web缓存器，并且预先配置其配套的浏览器，使之指向这些缓存器。

在因特网上部署Web缓存器有两个原因。首先，Web缓存器可以大大地减少对客户机请求的响应时间，特别是当客户机与初始服务器之间的瓶颈带宽远低于客户机与Web缓存器之间的瓶颈带宽时更是如此。如果在客户机与Web缓存器之间有一个高速连接，并且用户所请求的对象在Web缓存器上（实际情况通常是这样的），则Web缓存器可以迅速将该对象交付给用户。其次，如下面的例子说明的那样，Web缓存器可以大大减少一个机构内部网与因特网接入链路上的通信量。通过减少通信量，该机构（如一家公司或者一所大学）就不必急于增加带宽，因此会降低费用。此外，Web缓存器能从整体上大大降低因特网上的Web流量，从而改善了所有应用的性能。

为了深入理解缓存器带来的好处，我们考虑图2-12中的例子。该图显示了两个网络，即机构内部网络和公共因特网的一部分。机构内部网络是一个高速局域网。机构内部网络的路由器与因特网上的路由器通过一条15 Mbps的链路连接。初始服务器与因特网相连，但遍布全球。假设对象的平均长度为1Mb，机构内的浏览器对初始服务器的平均访问速率为每秒15个请求。假设HTTP请求报文小到可以忽略，因而不会在网络中以及接入链路（从机构内部路由器到因特网路由器）上产生任何通信量。我们还假设，从图2-12中接入链路的因特网一侧的路由器转发HTTP请求报文（在一个IP数据报中）开始，到它收到其响应报文（通常在多个IP数据报中）的时间平均为2s。我们非正式地将该持续时延称为“因特网时延”。

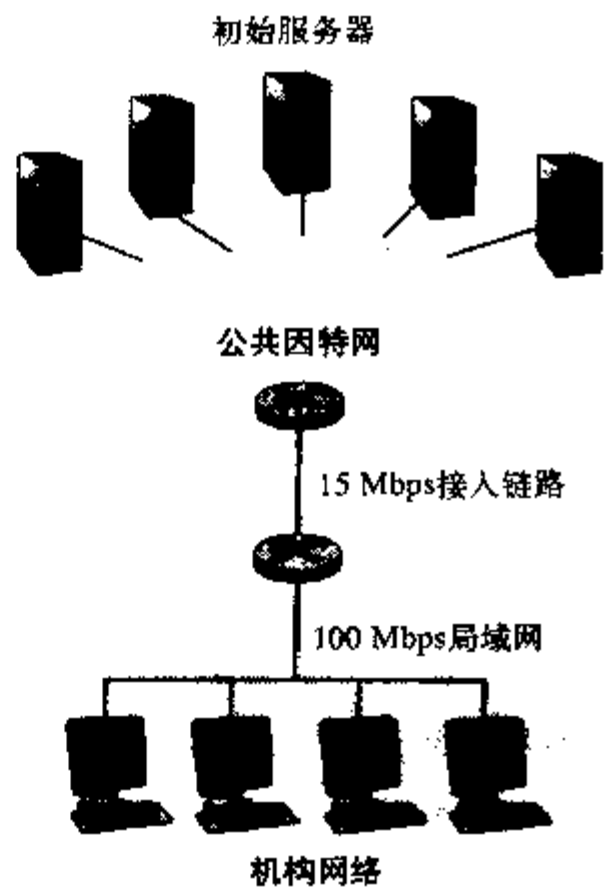


图2-12 机构网络与因特网之间的瓶颈

总的响应时间（即从浏览器请求一个对象到接收到该对象为止的时间）是局域网时延、接入时延（即两个路由器之间的时延）和因特网时延之和。我们来粗略地估算一下这个时延。局域网上的流量强度（参见1.4.2节）为

$$(15\text{个请求/s}) \cdot (1\ \text{Mb/请求}) / (100\ \text{Mbps}) = 0.15$$

而接入链路上的流量强度（从因特网路由器到机构路由器）为

$$(15\text{个请求/s}) \cdot (1\ \text{Mb/请求}) / (15\ \text{Mbps}) = 1$$

局域网上强度为0.15的通信量最多导致数十毫秒的时延，因此我们可以忽略局域网时延。然而，如1.4.2节所讨论的那样，如果流量强度接近1（就像图2-12中接入链路的情况那样），链路上的时延会变得非常大并且无限增长。因此，满足请求的平均响应时间将以分钟计。显然，必须做点什么来改进时间响应特性。

一个可能的解决办法就是增加接入链路的速率，如从15 Mbps增加到100 Mbps。这可以将接入链路上的流量强度减少到0.15，这样一来，两个路由器之间的链路时延也可以忽略了。这时，总的响应时间将大约为2s，即为因特网时延。但这种解决方案也意味着该机构必须用较大投资，将接入链路由15 Mbps升级为100 Mbps。

现在来考虑另一种解决方案，即不升级链路带宽，而是在机构网络中安装一个Web缓存器。这种解决方案如图2-13所示。实际的命中率（即由一个缓存器满足的请求的比率）通常在0.2～

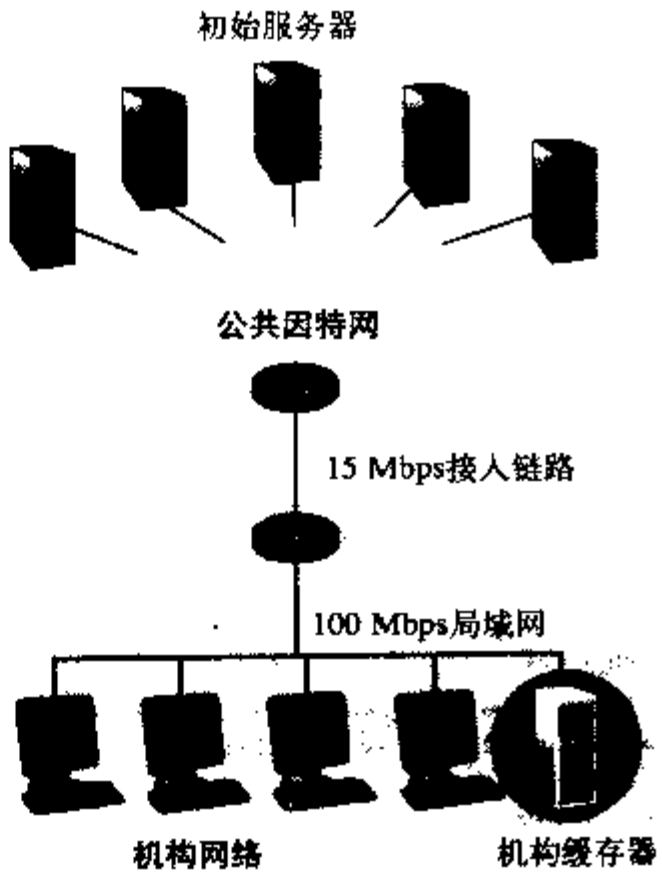


图2-13 为机构网络添加一台缓存器

0.7之间。为了阐述方便，我们假设该机构缓存器的命中率为0.4。因为客户机和缓存器位于同一个高速LAN上，这40%的请求将几乎立即会得到响应，时延约在10 ms以内。毫无疑问，剩下的60%的请求仍然需要通过访问初始服务器才能满足。但是只有60%的被请求对象通过接入链路传送，其流量强度从1.0减小到0.6。一般而言，在15 Mbps链路上，当流量强度小于0.8时时延很小，约为几十毫秒。这个值与2 s的因特网时延相比，可以忽略不计。考虑这些之后，平均时延因此为

$$0.4 \cdot (0.010\text{s}) + 0.6 \cdot (2.01\text{s})$$

略大于1.2 s。因此，第二种解决方案提供的响应时延甚至比第一种解决方案更低，也不需要该机构升级它到因特网的接入链路。该机构理所当然是购买和安装Web缓存器。除此之外，它的成本也低，很多缓存器就是使用在廉价PC机上运行的公共领域软件。

2.2.6 条件GET方法

尽管高速缓存能减少用户感受到的响应时间，但也引入了一个新的问题，即存放在缓存器中的对象拷贝可能是陈旧的。换句话说，保存在服务器中的对象在该拷贝缓存在客户机后可能已经被修改了。幸运的是，HTTP协议有一种机制，允许缓存器证实它的对象是最新的。这种机制就是**条件GET**（conditional GET）方法。如果①请求报文使用GET方法；②请求报文中包含一个`If-modified-since:`首部行，那么这个HTTP请求报文就是一个条件GET请求报文。

为了描述GET方法是如何操作的，我们来看一个例子。第一，一个代理缓存器代表一个请求浏览器，向某Web服务器发送一个请求报文：

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
```

第二，Web服务器向缓存器发送具有被请求的对象的响应报文：

```
HTTP/1.1 200 OK
Date: Thu, 7 Jul 2007 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 4 Jul 2007 09:23:24
Content-Type: image/gif

(data data data data data ...)
```

缓存器在将对象转发到请求的浏览器的同时，也将对象保存在本地缓存器中。重要的是，缓存器在存储对象时也存储了最后修改时间。第三，一个星期后，另一个用户通过该缓存器请求同一个对象，该对象仍在这个缓存器中。由于在过去的一个星期中位于Web服务器上的该对象可能已经被修改了，该缓存器通过发送一个条件GET，执行最新检查。具体来说，该缓存器发送：

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 4 Jul 2007 09:23:24
```

注意到`If-modified-since:`首部行的值正好等于一星期前服务器发送的响应报文中的`Last-modified:`首部行的值。该条件GET报文告诉服务器，仅当自指定日期之后修改过该对象才发送该对象。假设该对象自2007年7月4日09:23:24后没有被修改。接下来，第四步，Web服务器向该缓存器发送一个响应报文：

```
HTTP/1.1 304 Not Modified
Date: Sat , 14 Jul 2007 15:39:29
Server: Apache/1.3.0 (Unix)
(实体主体为空)
```

我们看到，作为对该条件GET方法的响应，Web服务器发送一个响应报文，但并没有包含所请求的对象。包含该对象只是对带宽的浪费，并增加用户感受到的响应时间，特别是当该对象很大的时候。应当注意到在最后的响应报文中，状态行中状态码和相应状态信息的值为304 Not Modified，它告诉缓存器可以使用该对象，向请求的浏览器转发该对象的拷贝。

至此，我们已经详细讨论了第一个因特网协议（应用层协议）——HTTP。我们学习了HTTP报文的格式，以及发送和接收HTTP报文时Web客户机和服务器所采取的动作。我们还学习了一些与HTTP协议有关的Web应用程序基础设施，包括缓存、cookie和后端数据库。

2.3 文件传输协议：FTP

在典型的FTP会话中，用户坐在一台主机（本地主机）前面，向一台远程主机上传文件或从远程主机下载文件。为使用户能访问远程主机的账户，用户必须提供一个用户标识和口令。在提供了授权信息后，用户就能从本地文件系统向远程主机文件系统传送文件，反之亦然。如图2-14所示，用户通过一个FTP用户代理与FTP交互。该用户首先提供远程主机的主机名，使本地主机的FTP客户机进程建立一个到远程主机FTP服务器进程的TCP连接。然后，该用户提供用户标识和口令，作为FTP命令的一部分在该TCP连接上传送。一旦该服务器向该用户授权，用户就可以向远程文件系统拷贝存放在本地文件系统中的一个或者多个文件（反之亦然）。

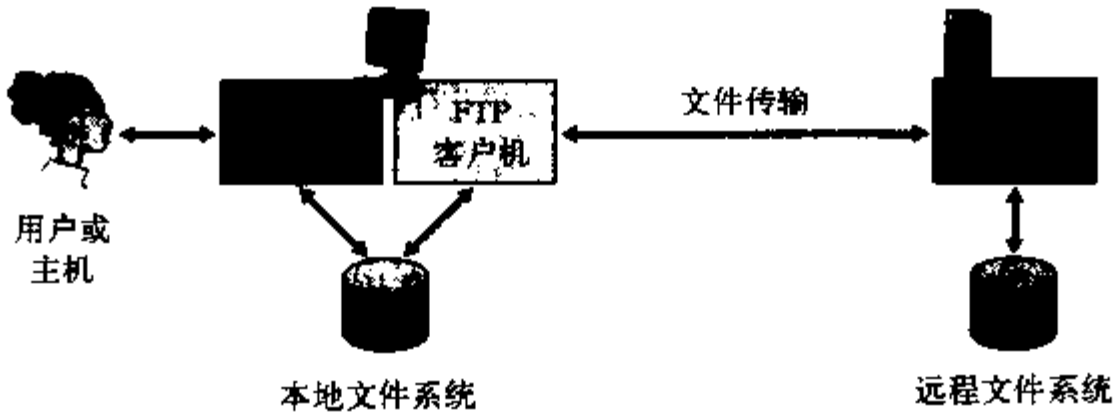


图2-14 使用FTP协议在本地文件系统和远程文件系统间移动文件

HTTP和FTP都是文件传输协议，并且有很多共同的特点。例如，它们都运行在TCP上。然而，这两个应用层协议也有一些重要的区别。其中最显著的就是FTP使用两个并行的TCP连接来传输文件，一个是**控制连接**（control connection），一个是**数据连接**（data connection）。控制连接用于在两个主机之间传输控制信息，如用户标识、口令、改变远程目录的命令以及“put”和“get”文件的命令。数据连接用于实际传输一个文件。因为FTP协议使用一个分离的控制连接，所以我们也称FTP的控制信息是**带外**（out-of-band）传送的。在第7章中我们将看到，用于控制音频和视频等连续媒体传输的RTSP协议的控制信息也是带外传送的。正如你所知道的，HTTP协议是在传输文件的TCP连接中发送请求和响应首部行的。因此，HTTP也可以说是**带内**（in-band）发送控制信息的。FTP协议的控制连接和数据连接如图2-15所示。

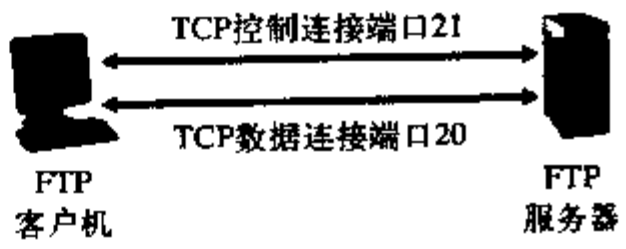


图2-15 控制连接和数据连接

当用户主机与远程主机开始一个FTP会话前，FTP的客户机（用户）首先在21号端口上发起一个用于控制的与服务器（远程主机）的TCP连接。FTP的客户机通过该控制连接发送用户的标识和口令，也发送改变远程目录的命令。当FTP的服务器端从该连接上收到一个文件传输的命令后（无论是到远程主机还是从远程主机），就发起一个到客户机的数据连接。FTP在该数据连接上准确地传送一个文件并关闭该连接。如果在同一个会话期间，用户还需要传输另一个文件，FTP则打开另一个数据连接。因而对FTP传输而言，控制连接贯穿了整个用户会话期间，但是针对会话中的每一次文件传输都需要建立一个新的数据连接（即数据连接是非持久的）。

FTP服务器必须在整个会话期间保留用户的状态（state）信息。特别是，服务器必须把特定的用户账户与控制连接联系起来，随着用户在远程目录树上移动，服务器必须追踪用户在远程目录树上的当前位置。对每个活动着的用户会话的状态进行追踪，可以对FTP会话总数进行限制。另一方面，HTTP是无状态的，即它不必对任何用户状态进行追踪。

FTP命令和回答

下面简要地讨论几个常见的FTP命令和回答。从客户机到服务器的命令和从服务器到客户机的回答，都是按照7位ASCII格式在控制连接上传送。因此，和HTTP协议的命令一样，FTP协议的命令也是人可读的。为了区分连续的命令，每个命令后跟回车换行符。每个命令由4个大写字母组成，有些还具有可选参数。一些常见的命令如下：

- `USER username`：用于向服务器传送用户标识。
- `PASS password`：用于向服务器传送用户口令。
- `LIST`：用于请求服务器返回远程主机当前目录的所有文件列表。文件列表是在（新建的非持久连接）数据连接上传送，而不是在控制TCP连接上传送。
- `RETR filename`：用于从远程主机的当前目录检索（即get）文件。该命令触发远程主机发起一个数据连接，并在该数据连接上发送所请求的文件。
- `STOR filename`：用于向远程主机的当前目录存放（即put）文件。

在用户发出的命令和FTP协议在控制连接上发送的命令之间，一般有一一对应关系。每个命令都对应着一个从服务器返回到客户机的回答。回答是一个3位数字，后跟一个可选信息。这与HTTP响应报文状态行的状态码和状态信息的结构相同。一些典型的回答以及它们可能的报文如下：

- `331 Username OK, Password required`
- `125 Data connection already open; transfer starting`
- `425 Can't open data connection`
- `452 Error writing file`

要想学习其他FTP命令和回答，有兴趣的读者可阅读RFC 959。

2.4 因特网中的电子邮件

历史事件

Hotmail

1995年12月，Sabeer Bhatia和Jack Smith拜访了因特网风险投资人Draper Fisher

Jurvetson，建议他开发一个免费的基于Web的电子邮件系统。其基本想法是为任何想要的人分配一个免费的电子邮件账户，并且使得这个账户可以在Web上使用。使用这种基于Web的电子邮件，任何访问Web的人无论是在学校还是在社区图书馆都能阅读和发送邮件。此外，基于Web的邮件系统给它的用户提供了很大的移动性。以15%的公司股份作为交换条件，Draper Fisher Jurvetson向Bhatia和Smith提供了资金，后者组建了一家公司，叫做Hotmail。在3个全职员工和12～14个兼职人员（为自己所拥有的股份而工作）的共同努力下，在1996年7月，他们研发并提供了该服务。之后的一个月内，他们就拥有了100 000名用户。用户的数目不断地迅速增长，所有的用户在阅读邮件时都会看到展现在他们眼前的广告条。1997年12月，在启动该服务不到18个月，Hotmail就超过了1200万个用户，并且以4亿美元的价格被微软公司收购。

Hotmail公司的成功常被归结于它的“先行者优势”（first-mover advantage）和它固有的电子邮件“病毒行销”（viral marketing）策略。Hotmail公司具有的“先行者优势”是因为它是第一家提供基于Web的电子邮件服务的公司。其他的公司当然只是模仿它的想法，但是Hotmail具有6个月的领先发展优势。令人垂涎的先行者优势来自于有一个初始的创意，以及接下来秘密进行的快速开发。如果一项服务或者一件产品能够自我行销，那么它就可以被称之为病毒行销。电子邮件是具有病毒行销特征的典型例子，即发送方向一个或多个收件人发送邮件，使得所有的接收方都知道了该项服务。Hotmail公司的成功表明，结合“先行者优势”和“病毒行销”策略就能开发出招人喜爱的应用程序。也许正在阅读本书的某些学生将会成为这种新人之一——构思并且开发具有“先行者优势”和“病毒行销”策略特征的因特网服务。

自从有了因特网，电子邮件就在因特网上流行起来。当因特网还在襁褓之中时，电子邮件已经成为最为流行的应用程序[Segaller 1998]，随着时间的推移，它变得越来越精细，越来越强大，并且一直还在发展着。它是当今因特网上最重要、最实用的应用程序之一。

与普通邮件一样，电子邮件是一种异步通信媒介，当人们方便时就可以收发邮件，不必与他人的计划进行协调。与普通邮件相比，电子邮件更为快速并且易于分发，而且价格便宜。现代电子邮件具有许多强大的特性。使用邮件列表，电子邮件报文和垃圾邮件可以一次性发送给数以千计的接收方。另外，现代电子邮件报文常常包含附件、超链接、HTML格式文本和图片。

在本节中，我们将讨论处于因特网电子邮件的核心地位的应用层协议。在深入讨论这些应用层协议之前，我们先总体上看看因特网电子邮件系统和它的关键组件。

图2-16是因特网电子邮件系统的总体情况，从该图中我们可以看出它有3个主要组成部分：**用户代理**（user agent）、**邮件服务器**（mail server）和**简单邮件传输协议**（Simple Mail Transfer Protocol, SMTP）。下面我们结合发送方Alice发电子邮件给接收方Bob的例子，对每个组成部分进行描述。用户代理允许用户阅读、回复、转发、保存和撰写报文。（电子邮件的用户代理有时也叫做*邮件阅读器*，尽管我们在本书中通常避免使用该术语。）当Alice完成邮件撰写时，她的邮件代理向其邮件服务器发送邮件，并且该邮件被放在邮件服务器发送报文队列中。当Bob想读取一条报文时，其邮件代理从他的位于邮件服务器的邮箱中获取该报文。20世纪90年代末，具有GUI（图形用户界面）的用户代理开始流行起来，它允许用户阅读和编写多媒体邮件。当前，Microsoft Outlook、Apple Mail和Mozilla Thunderbird都是流行的具有

GUI的电子邮件用户代理。同时，还有很多免费的基于文本的电子邮件用户代理（包括mail、pine和elm），以及基于Web的界面。

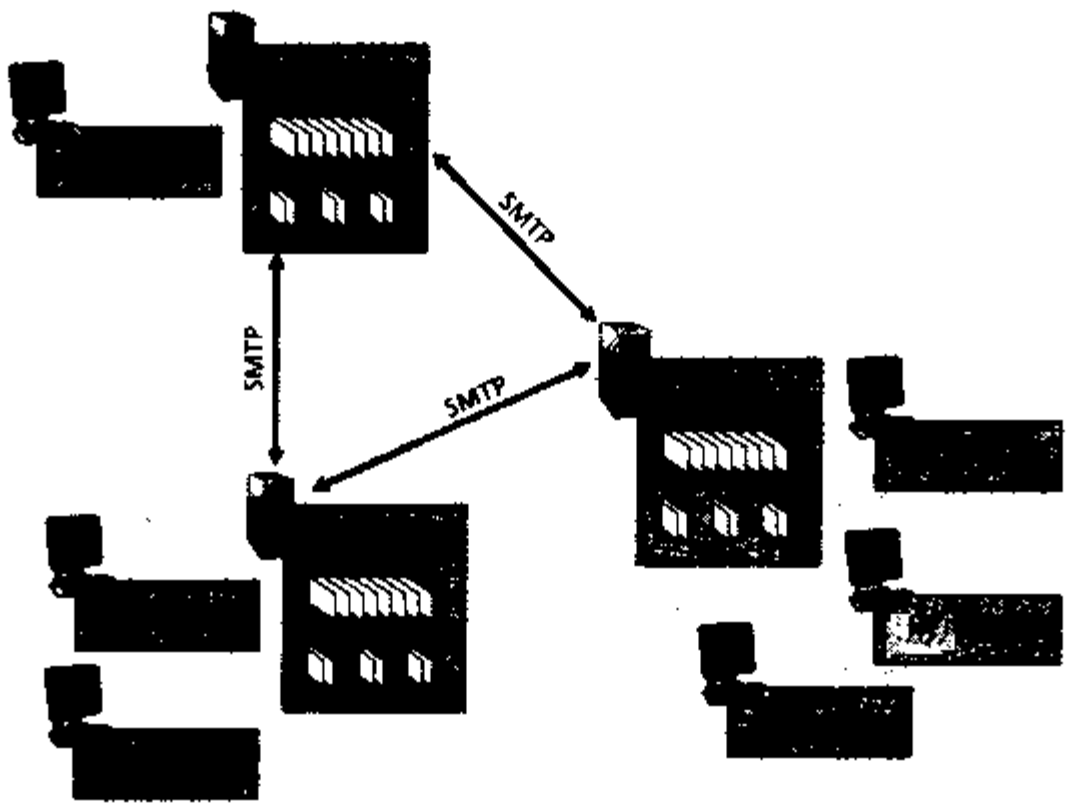


图2-16 因特网电子邮件系统的总体描述

邮件服务器组成了电子邮件体系结构的核心。每个接收方（如Bob）在其中的某个服务器上有一个**邮箱**（mailbox）。Bob的邮箱管理和维护发送给他的报文。一个典型的邮件发送过程是：从发送方的用户代理开始，传输到发送方的邮件服务器，再传输到接收方的邮件服务器，然后在这里被分发到接收方的邮箱中。当Bob在他的邮箱中访问该报文时，包含他的邮箱的邮件服务器鉴别Bob的身份（使用用户名和口令）。Alice的邮箱还必须处理Bob邮件服务器中的故障。如果Alice的服务器不能将邮件交付到Bob的服务器，Alice的邮件服务器在一个**报文队列**（message queue）中保持该报文并在以后尝试再次发送。通常每30分钟左右进行一次尝试，如果几天后仍不能成功，服务器删除该报文并以电子邮件的形式通知发送方（Alice）。

SMTP是因特网电子邮件中主要的应用层协议。它使用TCP可靠数据传输服务，从发送方的邮件服务器向接收方的邮件服务器发送邮件。像大多数应用层协议一样，SMTP也有两个部分：运行在发送方邮件服务器的客户机端和运行在接收方邮件服务器的服务器端。每个邮件服务器上既有SMTP的客户机端运行也有SMTP的服务器端运行。当一个邮件服务器向其他邮件服务器发送邮件时，它就表现为SMTP的客户机；当邮件服务器从其他邮件服务器上接收邮件时，它就表现为一个SMTP的服务器。

2.4.1 SMTP

RFC 2821给出了SMTP的定义。SMTP是因特网电子邮件应用的核心。如前所述，SMTP用于从发送方的邮件服务器发送报文到接收方的邮件服务器。SMTP比HTTP问世的时间要长得多（最初关于SMTP协议的RFC可追溯到1982年，而SMTP在此之前很长一段时间就已经出

现了)。尽管电子邮件应用在因特网上的独特地位可以证明SMTP有着众多非常出色的性质，但它所具有的某种“陈旧”特征表明它仍然是一种遗留下来的技术。例如，它限制所有邮件报文的主体部分（不只是其首部）只能采用简单的7位ASCII码表示。在20世纪80年代早期，这种限制很有意义，因为当时带宽不足，没有人会通过电子邮件发送大的附件或大的图片、声音、视频文件。然而，在今天的多媒体时代，7位ASCII码的限制的确有点痛苦，在用SMTP传送邮件之前，需要将二进制多媒体数据编码为ASCII码，并且在使用SMTP传输后需要将相应的ASCII码邮件解码还原为多媒体数据。在2.2节我们讲过，使用HTTP传送前不需要进行多媒体数据与ASCII码之间的转换。

为了描述SMTP的基本操作，我们看一下常见的情景。假设Alice想给Bob发送一封简单的ASCII报文。

1）Alice调用她的邮件代理程序并提供Bob的邮件地址（如bob@someschool.edu），撰写邮件，然后通过用户代理发送该邮件。

2）Alice的用户代理把报文发给Alice的邮件服务器，在那里该报文被放在报文发送队列中。

3）运行在Alice邮件服务器上的SMTP客户机端发现了报文发送队列中的这个报文，它就创建一个到运行在Bob的邮件服务器上的SMTP服务器的TCP连接。

4）在经过一些初始SMTP握手后，SMTP客户机通过该TCP连接发送Alice的报文。

5）在Bob的邮件服务器上，SMTP的服务器端接收该报文，Bob的邮件服务器然后将该报文放入Bob的邮箱中。

6）在Bob方便的时候，他调用用户代理阅读该报文。

图2-17总结了上述这个情况。

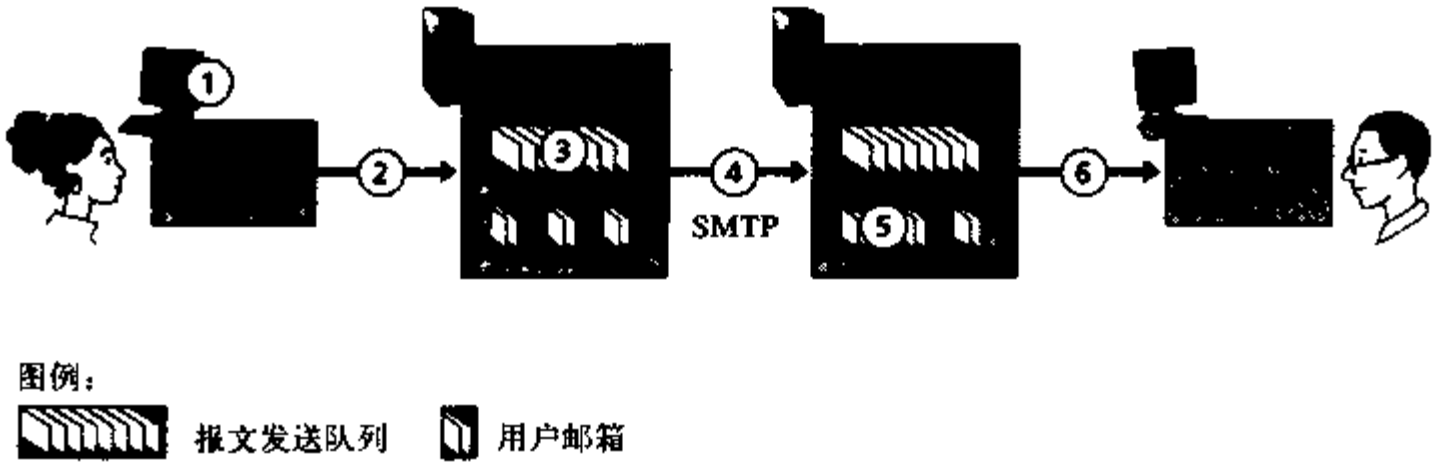


图2-17 Alice向Bob发送一条报文

观察到下述现象是非常重要的：SMTP一般不使用中间邮件服务器发送邮件，即使这两个邮件服务器位于地球的两端也是这样。假设Alice的邮件服务器在中国香港，而Bob的邮件服务器在美国St. Louis（圣路易斯），那么这个TCP连接也是从中国香港到圣路易斯服务器之间的直接相连。特别是，如果Bob的邮件服务器没有开机，该报文会保留在Alice的邮件服务器上并在稍后进行新的尝试，这意味着邮件并不在中间的某个邮件服务器上存留。

我们现在仔细观察一下，使用SMTP如何将一个报文从发送邮件服务器传送到接收邮件服务器。我们将会看到，SMTP与人类面对面交往的行为方式有许多类似性。首先，SMTP客户机（运行在发送邮件服务器上）在25号端口建立一个到SMTP服务器（运行在接收邮件服务器上）的TCP连接。如果某服务器没有开机，客户机会在稍后继续尝试连接。一旦连接建立，服务器和客户机就执行一些应用层的握手，就像人们在互相交流前先进行自我介绍一样。

SMTP的客户机和服务器在传输信息前先相互介绍。在SMTP握手的阶段，SMTP客户机指明发送方的邮件地址（产生报文的那个人）和接收方的邮件地址。一旦该SMTP客户机和服务器彼此介绍之后，客户机就发送该报文。SMTP可以利用TCP提供的可靠数据传输无差错地将邮件投递到接收服务器。该客户机如果有另外的报文要发送到该服务器，就在该相同的TCP连接上重复这种处理；否则，它指示TCP关闭连接。

接下来我们分析SMTP客户机（C）和SMTP服务器（S）之间交换报文脚本的例子。客户机的主机名为crepes.fr，服务器的主机名为hamburger.edu。以C:开头的ASCII码文本行就是客户机交给其TCP套接字的那些行，以S:开头的ASCII码则是服务器发送给其TCP套接字的那些行。一旦创建了TCP连接，就开始了下列过程。

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr ... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

在上例中，客户机程序从邮件服务器crepes.fr向邮件服务器hamburger.edu发送了一个报文（Do you like ketchup?How about pickles?）。作为对话的一部分，该客户机发送了5条命令：HELO（是Hello的缩写）、MAIL FROM、RCPT TO、DATA以及QUIT。这些命令都是自解释的。该客户机通过发送一个只包含一个句点的行，告诉服务器该报文结束了。（按照ASCII码的表示方法，每个报文以CRLF.CRLF结束，其中的CR和LF分别表示回车和换行。）服务器对每条命令做出回答，其中每个回答含有一个回答码和一些（可选的）英文解释。在这里我们指出SMTP用的是持久连接：如果发送邮件服务器有几个报文发往同一个接收邮件服务器，它可以通过同一个TCP连接发送所有这些报文。对每个报文，客户机都用一个新的MAIL FROM:crepes.fr开始，用一个独立的句点指示该邮件的结束，并且仅当所有邮件发送完后才发送QUIT。

我们强烈推荐使用Telnet与SMTP服务器进行直接对话。使用的命令是

```
telnet serverName 25
```

其中，serverName是本地邮件服务器的名称。当你这么做时，就会在本地主机与邮件服务器之间建立一个TCP连接。输完上述命令后，你立即会从该服务器收到220回答。接下来，在适当的时机发出HELO、MAIL FROM、RCPT TO、DATA、CRLF.CRLF以及QUIT等SMTP命令。强烈推荐读者完成本章后面的编程作业2。在该作业中，你将建立一个简单的用户代理以实现SMTP的客户机端，它允许你通过本地邮件服务器向任意的接收方发送电子邮件报文。

2.4.2 与HTTP的对比

我们简单地比较一下SMTP和HTTP。这两个协议都用于从一台主机向另一台主机传送文件：HTTP从Web服务器向Web客户机（通常是浏览器）传送文件（也称为对象），SMTP从一

个邮件服务器向另一个邮件服务器传送文件（即电子邮件报文）。当进行文件传送时，持久HTTP和SMTP都使用持久连接。因此，这两个协议有一些共同特征。然而，两者之间也有一些重要的区别。第一，HTTP主要是一个拉协议（pull protocol），即人们可以在方便的时候装载Web服务器上的信息，也就是说，用户使用HTTP从该服务器拉取信息。特别是，TCP连接是由想获取文件的机器发起的。另一方面，SMTP基本上是一个推协议（push protocol），即发送邮件服务器把文件推向接收邮件服务器。特别是，这个TCP连接是由要发送文件的机器发起的。

第二个区别就是我们前面间接提到过的，SMTP要求每个报文（包括它们的主体）都使用7位ASCII码格式。如果某报文包含了非7位ASCII字符（如具有重音的法文字符）或二进制数据（如图形文件），则该报文必须按照7位ASCII码进行编码。HTTP数据则没有这个限制。

第三个重要区别在于如何处理一个既包含文本又包含图形（也可能是其他媒体类型）的文档。如2.2节所述，HTTP把每个对象封装到它自己的HTTP响应报文中，而因特网电子邮件则把所有报文对象放在一个报文之中，我们后面将会详细讨论这一点。最后一个重要区别是我们前面已经讨论过的，HTTP使用带内控制，而FTP使用带外控制。

2.4.3 邮件报文格式和MIME

当Alice给Bob写一封邮寄时间很长的普通信件时，她可能要在信的上部包含所有各种环境首部信息，如Bob的地址、她自己的回复地址以及日期等。同样，当从一个人给另一个人发送电子邮件时，在报文主体前面也附有环境信息。这些环境信息包括在一系列首部行中，这些行由RFC 822定义。首部行和该报文的主体用空行（即回车换行）进行分隔。RFC 822定义了邮件首部行及其语义解释的精确格式。如同使用HTTP协议一样，每个首部行都包含了可读的文本，它们是由关键词后跟冒号再后跟值组成的。某些关键词是必需的，有些则是可选的。每个首部都必须含有一个`From:`首部行和一个`To:`首部行，可以包含一个`Subject:`首部行或者其他可选的首部行。注意到下列事实是很重要的：这些首部行不同于我们在2.4.1节所学到的SMTP命令（即使那里也包含了某些相同的词汇，如from和to）。2.4.1节中的命令是SMTP握手协议的一部分，而首部行则是邮件报文的一部分。

一个典型的报文首部如下：

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Searching for the meaning of life.
```

在报文首部之后，紧接着是一个空白行，然后是以ACSII格式表示的报文主体。可以用Telnet向邮件服务器发送包含一些首部行的报文，其中含有`Subject:`首部行。为此，输入命令`telnet serverName 25`，如2.4.1节中讨论的那样。

1. 非ASCII码数据的MIME扩展

虽然在RFC 822中描述的报文首部适合于发送普通ASCII文本，但它们不能充分地满足多媒体报文（如带有图片、音频和视频的报文）或携带有非ASCII文本格式（例如，非英语语言所使用的字符）的报文的需求。为发送非ASCII文本的内容，发送方的用户代理必须在报文中使用附加的首部行。这些额外的首部行定义在RFC 2045和RFC 2046中，多用途因特网邮件扩展（Multipurpose Internet Mail Extension, MIME）是对RFC 822的扩展。

支持多媒体的两个关键MIME首部是`Content-Type:`和`Content-Transfer-Encoding:`。`Content-Type:`首部允许接收用户代理对报文采取适当的动作。例如，通过指出报文主体包含一个JPEG图形，接收用户代理可以为报文主体启用一个JPEG图形的解压缩

程序。为了理解为什么需要Content-Transfer-Encoding:首部行，回想一下，为了不致扰乱SMTP的正常工作，非ASCII报文必须编码成ASCII格式。Content-Transfer-Encoding:首部行提示接收用户代理该报文主体已经使用了ASCII编码，并指出了所用的编码类型。因此，当用户代理接收到包含这两个首部行的报文时，就会根据Content-Transfer-Encoding:的值将报文主体还原成非ASCII的格式，然后根据Content-Type:首部行决定它应当采取何种动作来处理报文主体。

我们看一个具体的例子。假设Alice想发送一个JPEG图形给Bob。为此，Alice调用她的电子邮件用户代理，指定Bob的邮件地址，定义该报文的主题，并在该报文的报文主体中插入该JPEG图形。（具体操作方法取决于她所使用的用户代理，她可能将该图形作为一个“附件”插入。）写完她的报文后，Alice点击“发送”。Alice的用户代理则产生了一个MIME报文，该报文看起来具有如下形式：

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

(base64 encoded data .....
.........................
......base64 encoded data)
```

从上述MIME报文可以看到，Alice的用户代理使用base64编码对该JPEG图形进行了编码。这是在MIME [RFC 2045]中标准化的几种编码技术之一，用于转换为可接受的7位ASCII码格式。另外一个流行的编码技术是引用可打印内容转换编码（quoted-printable content-transfer-encoding），该编码常用于将一个8位ASCII报文（可能包含非英文字符）转换成7位ASCII码格式。

当Bob用他的用户代理阅读该邮件时，他的用户代理对相同的MIME报文进行操作。当Bob的用户代理发现了Content-Transfer-Encoding: base64首部行时，它会对base64编码的报文主体执行解码操作。该报文同时包含Content-Type: image/jpeg首部行，用于提示Bob的用户代理程序，该报文主体应当进行JPEG文件解压缩。最后，该报文中包含了MIME-Version:首部行，它用于指出MIME的版本号。注意，该报文的其他部分遵从的是RFC 822/SMTP格式。特别是，在报文首部后有一个空白行，接下来是报文主体。

2. 接收的报文

如果不提及由SMTP接收服务器插入的另一类首部行，将是不负责任。接收服务器一旦接收到具有RFC 822和MIME首部行的报文，就在该报文的顶部添加一个Received:首部行，该首部行定义了发送该报文的SMTP服务器的名称（from），接收该报文的SMTP服务器的名称（by），以及接收服务器接收到的时间。这样，目的用户看到的邮件具有如下所示的格式：

```
Received: from crepes.fr by hamburger.edu; 12 Oct 98
15:27:39 GMT
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
```

```
base64 encoded data .....
.....
.....base64 encoded data
```

几乎每个用过电子邮件的用户都在电子邮件报文的前面看到过Received:首部行（连同其他首部行）。（该行通常在屏幕上或者通过打印机可以直接看到。）你可能注意到，一个邮件有时有多个Received:行和一个更为复杂的Return-Path:首部行。这是因为有的邮件在发送方和接收方之间的路径上要经过不止一个SMTP服务器转发。例如，如果Bob指示他的邮件服务器hamburger.edu向sushi.jp转发他的所有报文，则Bob的用户代理阅读的报文中都有类似于下面的行：

```
Received: from hamburger.edu by sushi.jp; 3 Jul 01
15:30.01 GMT
Received: from crepes.fr by hamburger.edu; 3 Jul 01
15:17:39 GMT
```

这些首部行给接收用户代理提供了所访问的SMTP服务器的踪迹，以及访问发生的时间戳。

2.4.4 邮件访问协议

一旦SMTP将邮件报文从Alice的邮件服务器交付给Bob的邮件服务器，该报文就被放入了Bob的邮箱中。在讨论中，我们假定Bob是通过登录到服务器主机并直接在该主机上运行一个邮件阅读程序来阅读他的邮件的。直到20世纪90年代早期，这都是一种标准方式。而在今天，邮件访问使用了一种客户机/服务器体系结构，即用户一般通过在用户端系统上运行一个用户代理来阅读电子邮件，这里的端系统可能是办公室的PC、便携机或者PDA。通过运行在本地主机上的用户代理，用户可以享受一系列丰富的特性，包括查看多媒体报文和附件的能力。

假设Bob（接收方）在他的本地PC上运行用户代理程序，我们自然而然地也可以考虑在他的本地PC上放置一个邮件服务器。在这种情况下，Alice的邮件服务器就直接与Bob的PC机进行对话。然而这种方法会带来一个问题。前面讲过，邮件服务器管理用户的邮箱，并且运行SMTP的客户机和服务器端。如果Bob的邮件服务器位于他的PC上，那么为了能够及时接收可能在任何时候到达新的邮件，他的PC机必须总是不间断地运行着并一直保持在线。这对于大多数因特网用户而言是不现实的。相反，用户通常在本地PC上运行一个用户代理程序，而它访问位于共享邮件服务器上的邮箱，该共享邮件服务器总是保持开机并一直保持在线。该邮件服务器与其他用户共享，并且通常由用户的ISP（如大学或公司）进行维护。

现在我们考虑当从Alice向Bob发送电子邮件报文时所取的路径。我们刚才已经知道，在沿着该路径的某些点上，电子邮件报文存放在Bob的邮件服务器上。通过让Alice的用户代理直接向Bob的邮件服务器发送报文，就能够做到这一点。这可以由SMTP来完成：实际上，SMTP被设计成将电子邮件从一台主机推到另一台主机。然而，通常Alice的用户代理和Bob的邮件服务器之间并没有一个直接的SMTP对话。相反，如图2-18所示，Alice的用户代理用SMTP将电子邮件报文推入她的邮件服务器，接着她的邮件服务器（作为一个SMTP客户机）再用SMTP将邮件转发到Bob的邮件服务器。为什么要将该过程分成两步呢？主要是因为不借助于Alice的邮件服务器进行转发，Alice的用户代理将无法到达一个不可达的目的地接收服务器。通过首先将邮件存放在自己的邮件服务器中，Alice的邮件服务器可以重复地尝试向Bob的邮件服务器发送该报文，如每30分钟尝试一次，直到Bob的邮件服务器运行为止。（并且，如果Alice的邮件服务器关机，她能向系统管理员进行申告！）SMTP RFC定义了如何使用SMTP命令经过多个SMTP服务器进行报文转发。

但是这里仍然有一个疏漏的环节！像Bob这样的接收方是如何通过运行在他本地PC上的用户代理，获得位于某ISP的邮件服务器上的他的邮件的呢？注意到Bob的用户代理不能使用SMTP取回邮件，因为取邮件是一个拉操作，而SMTP协议是一个推协议。通过引入一个特殊的邮件访问协议来解决这个难题，该协议将Bob邮件服务器上的邮件传送给他的本地PC。目前有多个流行的邮件访问协议，包括**第三版的邮局协议**（Post Office Protocol－Version 3，POP3）、**因特网邮件访问协议**（Internet Mail Access Protocol，IMAP）以及HTTP。

图2-18总结了因特网电子邮件应用中使用的一些协议：SMTP用来将邮件从发送方的邮件服务器传输到接收方的邮件服务器；SMTP也用来将邮件从发送方的用户代理传送到发送方的邮件服务器。像POP3这样的邮件访问协议用来将邮件从接收方的邮件服务器传送到接收方的用户代理。

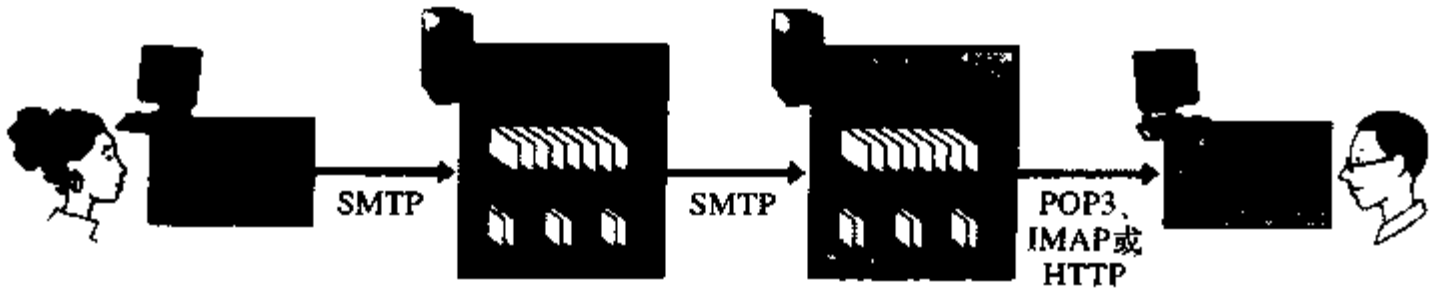


图2-18 电子邮件协议及其通信实体

1. POP3

POP3是一个非常简单的邮件访问协议，由RFC 1939进行了定义。该文档简短且可读性强。因为该协议非常简单，故其功能相当有限。当用户代理（客户机）打开了一个到邮件服务器（服务器）端口110上的TCP连接后，POP3就开始工作了。随着TCP连接的创建，POP3按照三个阶段进行工作：特许（authorization）、事务处理以及更新。在第一个阶段，即特许阶段，用户代理发送（以明文形式）用户名和口令以鉴别用户。在第二个阶段，即事务处理阶段，用户代理取回报文；在这个阶段，用户代理还能进行如下操作：对报文做删除标记，取消报文删除标记，以及获取邮件的统计信息。在第三个阶段，即更新阶段，它出现在客户机发出了quit命令之后，目的是结束该POP3会话；这时，邮件服务器删除那些被标记为删除的报文。

在POP3的事务处理过程中，用户代理发出一些命令，服务器对每个命令做出回答。回答可能有两种：+OK（有时后面还跟有服务器到客户机的数据），服务器用它来指示前面的命令是正常的；-ERR，服务器用它来指示前面的命令出现了某些差错。

特许阶段有两个主要的命令：user <user name>和pass <password>。为了举例说明这两个命令，我们建议读者直接用Telnet登录到POP3服务器的110端口，然后发出这两个命令。假设你的邮件服务器的名字为mailServer，那么你将看到类似下面的过程：

```
telnet mailServer 110
+OK POP3 server ready
user bob
+OK
pass hungry
+OK user successfully logged on
```

如果你的命令拼写错误了，该POP3服务器将返回一个-ERR报文。

现在我们来看一下事务处理过程。使用POP3的用户代理通常由用户配置为“下载并删除”或者“下载并保留”方式。POP3用户代理发出的命令序列取决于用户代理程序被配置为这两种工作方式中的哪一种。在下载并删除方式下，用户代理发出list、retr和dele命令。举

例来说，假设用户在他的邮箱里有两个报文。在下面的对话中，C：（代表客户机）是用户代理，S：（代表服务器）是邮件服务器。事务处理的过程将类似于如下过程：

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: (blah blah ...
S: ..................
S: ..........blah)
S: .
C: dele 1
C: retr 2
S: (blah blah ...
S: ..................
S: ..........blah)
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

用户代理首先请求邮件服务器列出所有存储报文的长度，接着用户代理从邮件服务器取回每封邮件并删除它们在服务器上的备份。注意在特许阶段以后，用户代理仅使用四个命令：list、retr、dele和quit，这些命令的语法在RFC 1939中定义。在处理quit命令后，POP3服务器进入更新阶段，从用户的邮箱中删除邮件1和2。

下载并删除方式带来这样一个问题，即邮件接收方Bob可能是移动的，希望从多个不同的机器访问他的邮件报文，如从办公室的PC、家里的PC或他的便携机来访问邮件。下载并删除方式将对Bob的邮件报文根据这3台机器进行划分，如果Bob最先是在办公室的PC机上收取了一个邮件，那么晚上当他在家里时，通过他的便携机将不能再次收取该邮件。使用下载并保留方式，用户代理下载某邮件后，该邮件仍保留在邮件服务器上。这时，Bob就能通过不同的机器重新读取这些邮件；他可以在上班时收取一封邮件，而在回家后再次访问它。

在用户代理与邮件服务器使用POP3进行会话期间，该POP3服务器保留了一些状态信息，特别是记录了哪些用户报文被标记为删除了。然而，POP3服务器并不在POP3会话过程中携带状态信息。会话中不包括状态信息大大简化了POP3服务的实现。

2. IMAP

使用POP3访问时，一旦Bob将邮件下载到本地主机后，他能建立邮件文件夹，并且将下载的邮件放入该文件夹中。然后，Bob可以删除报文，在文件夹间移动报文，也可以查询报文（通过发送方的名字或报文主题）。但是这种文件夹和报文存放在本地机上的方法，会给移动用户带来问题，因为他更喜欢使用一个远程服务器上的层次文件夹，这样他可以从任何一台机器上对所有报文进行访问。POP3不可能做到这一点，POP3协议没有给用户提供任何创建远程文件夹以及为报文指派文件夹的方法。

为了解决这个或其他一些问题，由RFC 3501定义的因特网邮件访问协议（IMAP）应运而生。和POP3一样，IMAP是一个邮件访问协议，但是它比POP3具有更多的特色，不过也比POP3复杂得多。（因此客户机和服务器端的实现也都复杂得多。）

IMAP服务器把每个报文与一个文件夹联系起来，当报文第一次到达服务器时，它是放在收件人的收件箱文件夹里。收件人则可以把邮件移到一个新的、用户创建的文件夹中，或阅读邮件、删除邮件等。IMAP协议为用户提供了创建文件夹以及在文件夹之间移动邮件的命令。

IMAP还为用户提供了在远程文件夹中查询邮件的命令，按指定条件去查询匹配的邮件。注意与POP3不同的是，IMAP服务器维护了IMAP会话的用户状态信息，例如，文件夹的名字以及哪个报文与哪个文件夹相关联。

IMAP的另一个重要特性是它具有允许用户代理获取报文组件的命令。例如，用户代理可以只读取一个报文的报文首部，或只是一个多方MIME报文的一部分。当用户代理和其邮件服务器之间使用低带宽连接（如低速调制解调器链路）的时候，这个特性非常有用。在这种低带宽连接的情况下，用户可能并不想取回其邮箱中的所有邮件，尤其要避免可能包含音频或视频片段的大邮件。读者可以在IMAP的官方网站全面了解IMAP的情况 [IMAP 2007]。

3. 基于Web的电子邮件

今天越来越多的用户使用他们的Web浏览器收发电子邮件。20世纪90年代中期，Hotmail引入了基于Web的访问；今天，Yahoo、Google以及重要的大学或者公司都提供了基于Web的电子邮件。使用这种服务，用户代理就是普通的浏览器，用户和其远程邮箱之间的通信则通过HTTP进行。当一个收件人（如Bob）想从他的邮箱中取一个报文时，该电子邮件报文从Bob的邮件服务器发送到他的浏览器，使用的是HTTP而不是POP3或者IMAP协议。当发件人（如Alice）要发送一封电子邮件报文时，该电子邮件报文从Alice的浏览器发送到她的邮件服务器，使用的是HTTP而不是SMTP。然而，Alice的邮件服务器在与其他的邮件服务器之间发送和接收邮件时，仍然使用SMTP。

2.5 DNS：因特网的目录服务

人类能以很多方式来识别。例如，可以通过出生证上的名字来识别，可以通过社会保险号码来识别，也可以通过驾驶执照上的号码来识别。尽管这些办法都可以用来识别一个人，但是在特定环境下，某种识别方法可能比另一种方法更为适合。例如，IRS（美国的一个税务征收机构）的计算机更喜欢使用定长的社会保险号码而不是出生证上的姓名。另一方面，普通人喜欢使用更好记的出生证上的姓名而不是社会保险号。（实际上，你能想象人们之间以这种方式说话吗？例如，“你好，我叫132-67-9875。请找一下我的丈夫，178-87-1146”。）

因特网上的主机和人类一样，也可以使用多种方式进行识别。主机的一种识别方法是用它的**主机名**（hostname），如`cnn.com`、`www.yahoo.com`、`gaia.cs.umass.edu`以及`cis.poly.edu`等，这些名字便于记忆，也乐于被人们接受。然而，主机名提供了很少（如果有的话）关于主机在因特网中位置的信息。（一个名为`www.eurecom.fr`的主机以国家码`.fr`结束，告诉我们该主机很可能在法国，仅此而已。）此外，因为主机名可能由不定长的字母数字组成，所以路由器很难处理。基于这些原因，主机也可以使用所谓**IP地址**（IP address）进行识别。

我们将在第4章对IP地址进行详细讨论，但现在简略地介绍一下还是有必要的。一个IP地址由4个字节组成，并有着严格的层次结构。例如，像`121.7.106.83`这样一个IP地址，其中的每个字节都被句点分隔开来，表示了0～255的十进制数字。我们说IP地址具有层次结构，是因为当我们从左至右扫描它时，会得到越来越详细的关于主机位于因特网何处的信息（即在哪个网络里，在众多网络的哪个网络里）。类似地，当我们从下向上查看邮政地址时，能够获得该地址位于何处的越来越具体的信息。

2.5.1 DNS提供的服务

实践原则

DNS：通过客户机/服务器模式提供的重要网络功能

与HTTP、FTP和SMTP协议一样，DNS协议是应用层协议，其原因在于：①使用客户机/服务器模式在通信的端系统之间运行，②在通信的端系统之间通过下面的端到端运输层协议来传送DNS报文。然而，在某种意义上，DNS的作用非常不同于Web应用、文件传输应用以及电子邮件应用。不同之处在于，DNS并不直接和用户打交道。相反，DNS为因特网上的用户应用程序以及其他软件提供一种核心功能，即将主机名转换为它们下面的IP地址。我们在1.2节就提到，因特网体系结构的复杂性大多数位于网络的“边缘”。DNS通过采用客户机和服务器在网络边缘实现了关键的名字地址转换功能，它仍是这种设计原理的一个范例。

我们刚刚看到了有两种方式识别主机：通过主机名或者IP地址。人们喜欢便于记忆的主机名标识，而路由器则喜欢定长的、有着层次结构的IP地址。为了折中这些不同的偏好，我们需要一种能进行主机名到IP地址转换的目录服务。这就是**域名系统**（Domain Name System, DNS）的主要任务。DNS是：①一个由分层的DNS服务器（DNS server）实现的分布式数据库；②一个允许主机查询分布式数据库的应用层协议。DNS服务器通常是运行BIND (Berkeley Internet Name Domain) 软件[BIND 2007]的UNIX机器。DNS协议运行在UDP之上，使用53号端口。

DNS通常由其他应用层协议（包括HTTP、SMTP和FTP）所使用，用于将用户提供的主机名解析为IP地址。举一个例子，考虑当某个用户主机上的一个浏览器（即一个HTTP客户机）请求URL `www.someschool.edu/index.html`页面时，会发生什么现象。为了使用户的主机能够将一个HTTP请求报文发送到Web服务器`www.someschool.edu`，该用户主机必须获得`www.someschool.edu`的IP地址。其做法如下：

1）同一台用户主机上运行着DNS应用的客户机端。

2）该浏览器从上述URL中抽取出主机名`www.someschool.edu`，并将这个主机名传给DNS应用的客户机端。

3）该DNS客户机向DNS服务器发送一个包含主机名的请求。

4）该DNS客户机最终会收到一份回答报文，其中含有对应于该主机名的IP地址。

5）一旦该浏览器接收到来自DNS的IP地址，它就可以向由该IP地址定位的HTTP服务器发起一个TCP连接。

从这个例子中，我们可以看到DNS给使用它的因特网应用带来了额外的时延，有时还相当可观。幸运的是，如我们下面所讨论的那样，想获得的IP地址通常就缓存在一个“附近的”DNS服务器中，这有助于减少DNS的网络流量和DNS的平均时延。

除了进行主机名到IP地址的转换外，DNS还提供了一些重要的服务：

- **主机别名**（host aliasing）。有着复杂主机名的主机可以拥有一个或者多个别名。例如，一台名为`relay1.west-coast.enterprise.com`的主机，可能还有两个别名——`enterprise.com`和`www.enterprise.com`。在这种情况下，`relay1.west-`

coast.enterprise.com也叫**规范主机名**（canonical hostname）。主机别名（如果有的话）比主机规范名更容易记忆。应用程序可以调用DNS来获得主机别名对应的规范主机名以及主机的IP地址。

- **邮件服务器别名**（mail server aliasing）。显而易见，人们也非常希望电子邮件地址好记忆。例如，Bob在Hotmail上有一个账户，Bob的邮件地址就像bob@hotmail.com这样简单。然而，Hotmail邮件服务器的主机名可能更为复杂，不像hotmail.com那样简单好记（例如，规范主机名可能是relay1.west-coast.hotmail.com）。电子邮件应用程序调用DNS，对提供的邮件服务器别名进行解析，以获得该主机的规范主机名及其IP地址。事实上，MX记录（参见后面）允许一个公司的邮件服务器和Web服务器使用相同的（别名化的）主机名；例如，一个公司的Web服务器和邮件服务器都可以叫做enterprise.com。
- **负载分配**（load distribution）。DNS也用于在冗余的服务器（如冗余的Web服务器等）之间进行负载分配。繁忙的站点（如cnn.com）被冗余分布在多台服务器上，每台服务器均运行在不同的端系统上，有着不同的IP地址。对于这些冗余的Web服务器，一个IP地址集合对应于同一个规范主机名。DNS数据库中存储着这些IP地址集合。当客户机为映射到这个IP地址集合的名字发出一个DNS请求时，该服务器用包含全部这些地址的报文进行回答，但在每个回答中旋转这些地址排放的顺序。因为客户机通常总是向IP地址排在最前面的服务器发送HTTP请求报文，所以DNS就在所有这些冗余的Web服务器之间旋转分配负载。DNS的旋转同样可以用于邮件服务器，因此，多个邮件服务器可以具有相同的别名。近来，Akamai [Akamai 2007]等公司以更加复杂的方式使用DNS，以提供Web内容分布服务（参见第7章）。

DNS是由RFC 1034和RFC 1035定义的，并在另外几个RFC中进行了更新。DNS是一个复杂的系统，我们在这里只是就其运行的主要方面进行学习。感兴趣的读者可以参考这些RFC文档以及Abitz和Liu写的书[Abitz 1993]，也可参阅回顾性的文章[Mockapetris 1988]（该文提供了DNS组成和工作原理的良好描述）和[Mockapetris 2005]。

2.5.2 DNS工作机理概述

下面给出一个DNS工作过程的总体概括，我们的讨论将集中在主机名到IP地址转换服务方面。

假设运行在用户主机上的某些应用程序（如Web浏览器或邮件阅读器）需要将主机名转换为IP地址。这些应用程序将调用DNS的客户机端，并指明需要被转换的主机名。（在很多基于UNIX的机器上，应用程序为了执行这种转换需要调用函数gethostbyname()。在2.7节，我们将显示一个Java程序是如何调用DNS的。）用户主机上的DNS接收到后，向网络中发送一个DNS查询报文。所有的DNS请求和回答报文使用UDP数据报经端口53发送。经过若干毫秒到若干秒的时延后，用户主机上的DNS接收到一个提供所希望映射的DNS回答报文。这个查询结果则被传递到调用DNS的应用程序。因此，从用户主机上调用应用程序的角度看，DNS是一个提供简单、直接的转换服务的黑盒子。但事实上，实现这个服务的黑盒子非常复杂，它由分布于全球的大量DNS服务器以及定义了DNS服务器与查询主机通信方式的应用层协议组成。

DNS的一种简单设计方式是在因特网上只使用一个DNS服务器，该服务器包含所有的映

射。在这种集中式设计中，客户机直接将所有查询直接发往单一的DNS服务器，同时该DNS服务器直接对所有的查询客户机做出响应。尽管这种设计方式非常具有吸引力，但它不适用于当今的因特网，因为因特网有着数量巨大（并持续增长）的主机。这种集中式设计的问题包括：

- **单点故障**（a single point of failure）。如果该DNS服务器崩溃，整个因特网将随之瘫痪！
- **通信容量**（traffic volume）。单个DNS服务器不得不处理所有的DNS查询（用于为上亿台主机产生的所有HTTP请求报文和电子邮件报文服务）。
- **远距离的集中式数据库**（distant centralized database）。单个的DNS服务器不可能“邻近”所有查询客户机。如果我们将单台DNS服务器放在纽约市，那么所有来自澳大利亚的查询必须传播到地球的另一边，中间也许还要经过低速和拥塞的链路，这将导致严重的时延。
- **维护**（maintenance）。单个DNS服务器将不得不为所有的因特网主机保留记录。这不仅将使这个中央数据库非常庞大，而且它还不得不为解决每个新添加的主机而频繁更新。

总的来说，在单一DNS服务器上运行集中式数据库完全没有可扩展能力。因此，DNS采用了分布式的设计方案。事实上，DNS是一个在因特网上实现分布式数据库的精彩范例。

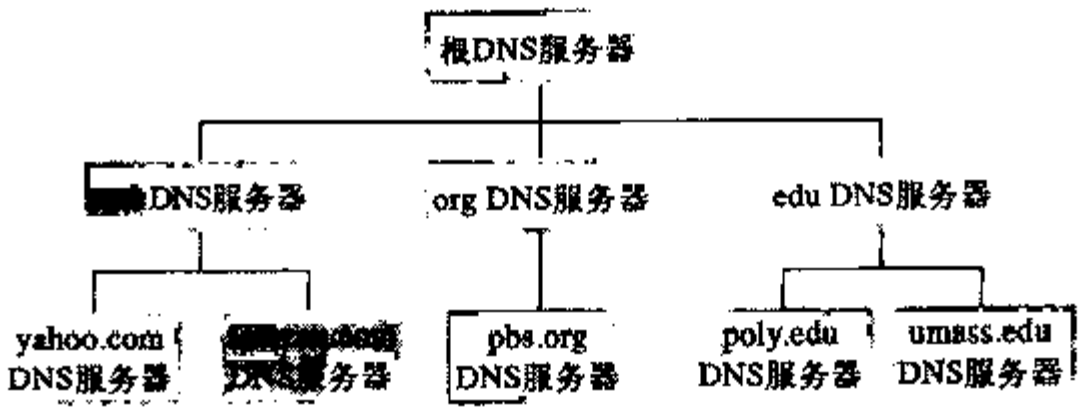


图2-19 DNS服务器的部分层次结构

1. 分布式、层次数据库

为了处理规模问题，DNS使用了大量的DNS服务器，它们以层次方式组织，并且分布在全世界范围内。没有一台DNS服务器具有因特网上所有主机的映射，相反，该映射分布在所有的DNS服务器上。大致说来，有3种类型的DNS服务器：根DNS服务器、顶级域（Top-Level Domain, TLD）DNS服务器和权威DNS服务器。这些服务器以图2-19中所示的层次结构组织起来。为了理解这3种类型的DNS服务器是如何交互的，假定一个DNS客户机要确定主机名www.amazon.com的IP地址。粗略来说，将发生下列事件。该客户机首先与根服务器之一联系，它将返回顶级域名com的TLD服务器的IP地址。该客户机则与这些TLD服务器之一联系，它将为amazon.com返回权威服务器的IP地址。最后，该客户机为amazon.com联系权威服务器之一，它为主机名www.amazon.com返回IP地址。我们很快将详细地研究DNS查找过程，但我们先仔细看一下这三种类型的DNS服务器。

- **根DNS服务器**。在因特网上有13个根DNS服务器（标号为A到M），其中大部分位于北美洲。图2-20中显示的是2006年10月的根DNS服务器分布图，通过[Root-servers 2007]可查看当前可用的根DNS服务器列表。尽管我们将这13个根DNS服务器中的每个都视为单个的服务器，但每台“服务器”实际上是冗余服务器的群集，以提供安全性和可靠性。
- **顶级域（TLD）服务器**。这些服务器负责顶级域名（如com、org、net、edu和gov）和所

有国家的顶级域名（如uk、fr、ca和jp）。在本书写作的时候（2007年春），Network Solutions公司维护com顶级域的TLD服务器，Educause公司维护edu顶级域的TLD服务器。

- **权威DNS服务器**。在因特网上具有公共可访问主机（如Web服务器和邮件服务器）的每个组织机构必须提供公共可访问的DNS记录，这些记录将这些主机的名字映射为IP地址。由组织机构的权威DNS服务器负责保存这些DNS记录。组织机构可以选择实现它自己的权威DNS服务器来保持这些记录，另一种方法是支付费用将这些记录存储在某个服务提供商的权威DNS服务器中。多数大学和大公司实现和维护它们自己基本和辅助（备份）权威DNS服务器。

图2-20 2007年的DNS根服务器（名称、组织和位置）

根、TLD和权威DNS服务器都处在DNS服务器的层次结构中，如图2-19所示。还有另一类重要的DNS，称为本地DNS服务器（local DNS server）。本地DNS服务器严格来说并不属于DNS服务器的层次结构，但它对DNS层次结构是很重要的。每个ISP（如大学、系、公司或居民区的ISP）都有一台本地DNS服务器（也叫默认名字服务器）。当主机与某个ISP连接时，该ISP提供一台主机的IP地址，该主机具有一台或多台其本地DNS服务器的IP地址（通常通过DHCP，将在第4章中讨论）。通过访问Windows或UNIX的网络状态窗口，可以很容易地确定本地DNS服务器的IP地址。主机的本地DNS服务器通常“邻近”本主机。对机构ISP而言，本地DNS服务器可能就与主机在同一个局域网中；对于居民区ISP来说，本地DNS服务器通常与主机相隔不超过几个路由器。当主机发出DNS请求时，该请求被发往本地DNS服务器，它起着代理的作用，并将该请求转发到DNS服务器层次结构中，我们下面将更为详细地讨论。

我们来看一个简单的例子，假设主机`cis.poly.edu`想知道主机`gaia.cs.umass.edu`的IP地址。同时假设理工大学（Polytechnic）的本地DNS服务器为`dns.poly.edu`，并且`gaia.cs.umass.edu`的权威DNS服务器为`dns.umass.edu`。如图2-21所示，主机`cis.poly.edu`首先向它的本地DNS服务器`dns.poly.edu`发送一个DNS查询报文。该查询报文含有被转换的主机名`gaia.cs.umass.edu`。本地DNS服务器将该报文转发到根DNS服务器。该根DNS服务器注意到其edu前缀并向本地DNS服务器返回负责edu的TLD的IP地址列表。该本地DNS服务器则再次向这些TLD服务器发送查询报文。该TLD服务器注意到`umass.edu`前缀，并用权威DNS服务器的IP地址进行响应，该权威DNS服务器是负责马萨诸

塞大学的dns.umass.edu。最后，本地DNS服务器直接向dns.umass.edu重发查询报文，dns.umass.edu用gaia.cs.umass.edu的IP地址进行响应。注意到在本例中，为了获得一个主机名的映射，共发送了8份DNS报文：4份查询报文和4份回答报文!我们将很快看到DNS缓存是如何减少这种查询流量的。

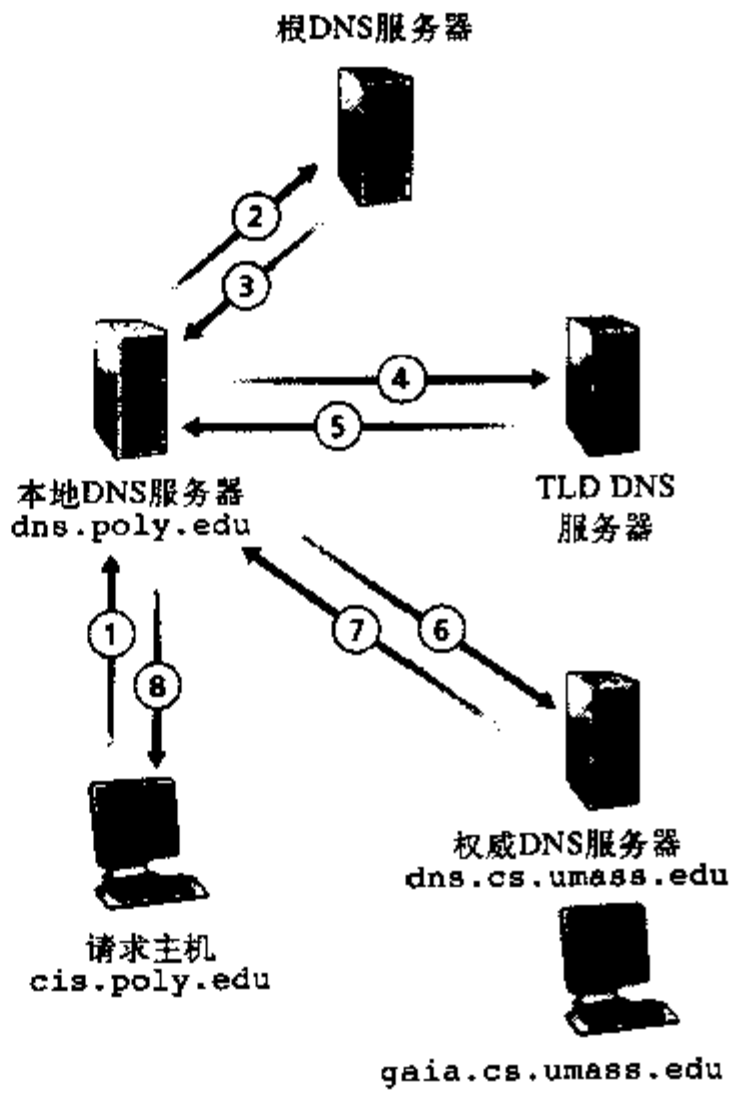


图2-21 各种DNS服务器的交互

前面的例子假设TLD服务器知道用于每台主机的权威DNS服务器的IP地址。一般而言，这种假设并不总是正确的。相反，TLD服务器只是知道中间的某个DNS服务器，该中间DNS服务器依次才能知道用于该主机的权威DNS服务器。例如，再次假设马萨诸塞大学有一台用于本大学的DNS服务器，称为dns.umass.edu。同时假设该大学的每个院系都有自己的DNS服务器，每个院系DNS服务器是本系所有主机的权威服务器。在这种情况下，当中间DNS服务器dns.umass.edu收到了对某主机的请求时，该主机名是以cs.umass.edu结尾的，它向dns.poly.edu返回dns.cs.umass.edu的IP地址，后者是所有主机名以cs.umass.edu结尾的主机的权威服务器。本地DNS服务器dns.poly.edu则向权威DNS服务器发送查询，该权威DNS服务器将请求的映射发送给本地DNS服务器，该本地服务器依次向请求主机返回该映射。在这个例子中，共发送了10份DNS报文!

图2-21所示的例子利用了**递归查询**（recursive query）和**迭代查询**（iterative query）。从cis.poly.edu到dns.poly.edu发出的查询是递归查询，因为该查询请求dns.poly.edu以自己的名义获得该映射。而后继的3个查询是迭代查询，因为所有的回答都是直接返回给dns.poly.edu。从理论上讲，任何DNS查询既可以是迭代的也可以是递归的。例如，图2-22显示了一条DNS查询链，其中的所有查询都是递归的。实际中，查询通常遵循图2-21中的模式：从请求主机到本地DNS服务器的查询是递归的，其余的查询是迭代的。

2. DNS缓存

迄今为止我们的讨论还没有涉及DNS系统的一个重要特征：**DNS缓存**（DNS caching）。实际上，为了改善时延性能并减少在因特网上到处传输的DNS报文数量，DNS广泛使用了缓存技术。DNS缓存的原理非常简单。在请求链中，当一个DNS服务器接收一个DNS回答（例如，包含主机名到IP地址的映射）时，DNS服务器能将回答中的信息缓存在本地存储器。例如，在图2-21中，每当本地DNS服务器dns.poly.edu从某个DNS服务器接收到一个回答，它就缓存包含在该回答中的任何信息。如果在DNS服务器中缓存了一个主机名/IP地址对，另一个对相同主机名的查询到达该DNS服务器时，该服务器能够提供所要求的IP地址，即使它不是该主机名的权威服务器。由于主机和主机名与IP地址间的映射决不是永久的，所以DNS服务器在一段时间后（通常设置为两天）将丢弃缓存的信息。

举一个例子，假定主机apricot.poly.edu向dns.poly.edu查询主机名cnn.com的

IP地址。此后，假定几个小时后，理工大学的另外一台主机kiwi.poly.fr也向dns.poly.edu查询相同的主机名。因为有了缓存，所以本地DNS服务器可以立即返回cnn.com的IP地址，而不必查询任何其他DNS服务器。本地DNS服务器也可以缓存TLD服务器的IP地址，因而允许本地DNS绕过查询链中的根DNS服务器（这经常发生）。

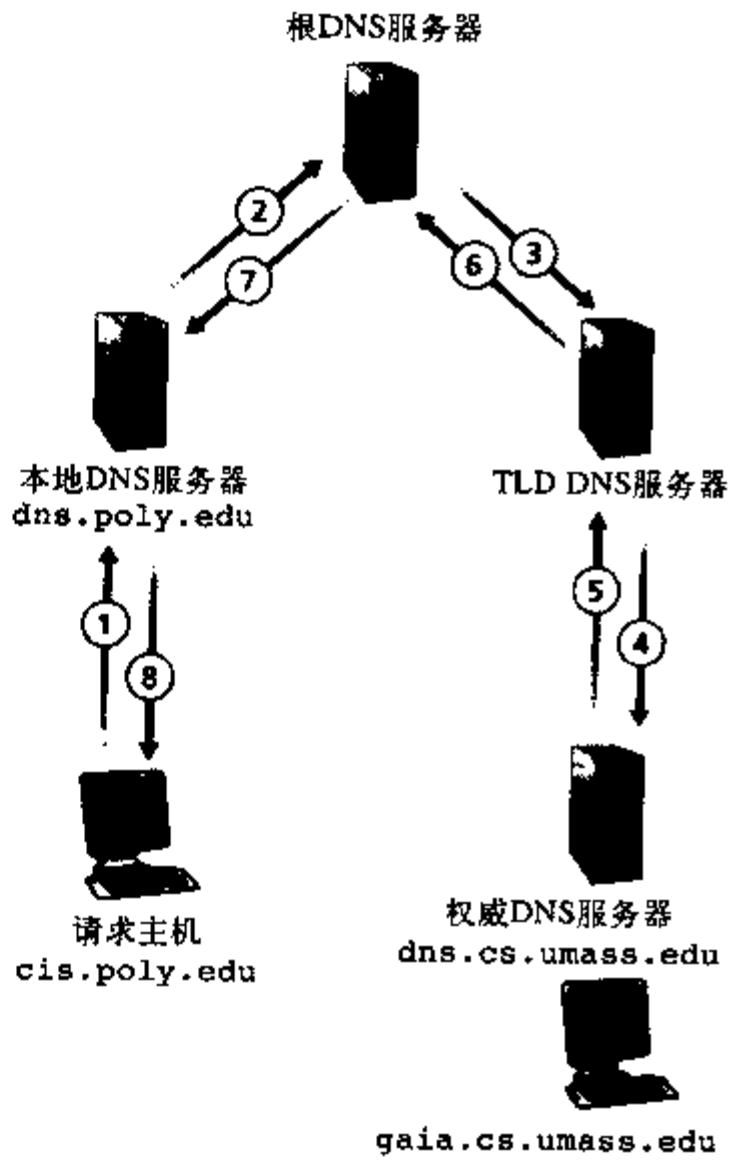


图2-22 DNS中的递归查询

2.5.3 DNS记录和报文

实现DNS分布式数据库的所有DNS服务器共同存储着资源记录（Resource Record, RR），RR提供了主机名到IP地址的映射。每个DNS回答报文包含了一条或多条资源记录。在本小节以及后续小节中，我们概要地介绍DNS资源记录和报文，更详细的信息可以在[Abitz 1993]或有关DNS的RFC文档[RFC 1034；RFC 1035]中找到。

资源记录是一个包含了下列字段的4元组：

(Name,Value,Type,TTL)

TTL是该记录的生存时间，它决定了资源记录应当从缓存中删除的时间。在下面给出的资源记录例子中，我们忽略掉TTL字段。Name和Value的值取决于Type：

- 如果Type=A，则Name是主机名，Value是该主机名的IP地址。因此，一条类型为A的资源记录提供了标准的主机名到IP地址的映射。例如，(relay1.bar.foo.com, 145.37.93.126, A) 就是一条类型A的记录。
- 如果Type=NS，则Name是域（如foo.com），而Value是知道如何获得该域中主机IP地址的权威DNS服务器的主机名。这个记录用于沿着查询链路进一步路由DNS查询。例如，(foo.com, dns.foo.com, NS) 就是一条类型NS的记录。
- 如果Type=CNAME，则Value是别名为Name的主机对应的规范主机名。该记录能够向请求主机提供一个主机名对应的规范主机名，例如，(foo.com, relay1.bar.foo.com, CNAME) 就是一条CNAME类型的记录。
- 如果Type=MX，则Value是别名为Name的邮件服务器的规范主机名。例如，(foo.com, mail.bar.foo.com, MX) 就是一条MX记录。MX记录允许邮件服务器的主机名具有简单的别名。注意，通过使用MX记录，一个公司的邮件服务器和其他服务器（如它的Web服务器）可以使用相同的别名。为了获得邮件服务器的规范主机名，DNS客户机应当请求一条MX记录；而为了获得其他服务器的规范主机名，DNS客户机应当请求一条CNAME记录。

如果一台DNS服务器是某特定主机名的权威DNS服务器，那么该DNS服务器会有一条包含该主机名的类型A记录。（即使该DNS服务器不是其权威DNS服务器，它也可能在缓存中含

有一条类型A记录。）如果DNS服务器不是某个主机名的权威DNS服务器，那么该服务器将包含一条类型NS记录，该记录对应于包含主机名的域；它还将包括一条类型A记录，该记录提供了在NS记录的Value字段中DNS服务器的IP地址。举例来说，假设一台edu TLD服务器不是主机gaia.cs.umass.edu的权威DNS服务器，则该服务器将包含一条包括主机cs.umass.edu的域记录，如（umass.edu，dns.umass.edu，NS）。该edu TLD服务器还将包含一条类型A记录，如（dns.umass.edu，128.119.40.111，A），该记录将名字dns.umass.edu映射为一个IP地址。

1. DNS报文

在本节前面，我们提到了DNS查询和回答报文。DNS只有这两种报文。并且，查询和回答报文有着相同的格式，如图2-23所示。

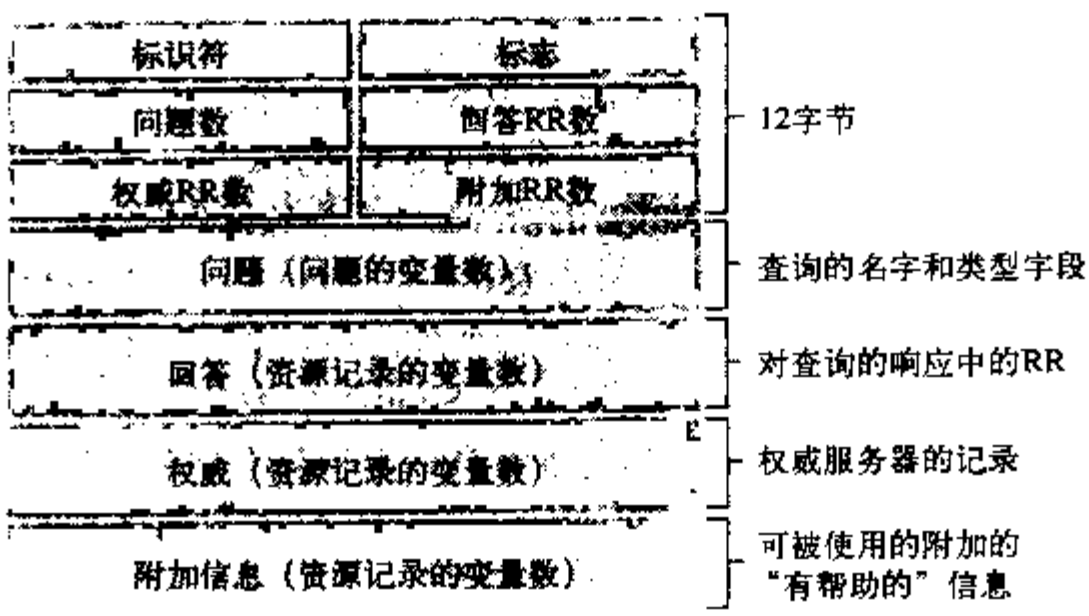


图2-23 DNS报文格式

DNS报文中各字段的语义如下：

- 前12个字节是首部区域，其中有几个字段。第一个字段是一个16 比特的数，用于标识该查询。这个标识符会被复制到对查询的回答报文中，以便让客户机用它来匹配发送的请求和接收到的回答。标志字段中含有若干标志。1比特的“查询/回答”标志位指出报文是查询报文（0）还是回答报文（1）。当某DNS服务器正好是被请求主机的权威DNS服务器时，1比特的“权威的”标志位被置在回答报文中。如果客户机（主机或者DNS服务器）希望DNS服务器执行递归查询，将设置1比特的“希望递归”标志位。如果该DNS服务器支持递归查询，在它的回答报文中会对1比特的“递归可用”标志位置位。在该首部中，还有4个“数量”字段，这些字段指出了在首部后4类数据区域出现的数量。
- 问题区域包含着正在进行的查询信息。该区域包括：①名字字段，用于指出正在被查询主机名字；②类型字段，用于指出正被询问的问题类型，例如是与一个名字相联系的主机地址（类型A）还是某个名字的邮件服务器（类型MX）。
- 在来自DNS服务器的回答报文中，回答区域包含了对最初请求的名字的资源记录。前面讲过每个资源记录中有Type（如A、NS、CNAME和MX）字段、Value字段和TTL字段。在一个回答报文的回答区域中可以包含多条RR，因为一个主机名可以对应多个IP地址（如本节前面讨论的冗余Web服务器）。
- 权威区域包含了其他权威DNS服务器的记录。
- 附加区域包含了其他一些有帮助的记录。例如，对于一个MX请求的回答报文的回答区

域中包含了一条资源记录，该记录提供了邮件服务器的规范主机名。该附加区域就可以包含一个类型A记录，该记录提供了对于该邮件服务器的规范主机名的IP地址。

你喜欢从正在工作的主机直接向某些DNS服务器发送一个DNS查询报文吗？使用nslookup程序（nslookup program）能够容易地做到这一点，nslookup程序适用于多数Windows和UNIX平台。例如，在一台Windows主机中打开命令提示符界面，直接键入“nslookup”即可调用nslookup程序。在调用nslookup后，可以向任何DNS服务器（根、TLD或权威）发送DNS查询。在接收到来自DNS服务器的回答后，nslookup将显示包括在该回答中的记录（以人可读的格式）。从你自己的主机运行nslookup还有一种方法，即访问允许你远程使用nslookup的许多Web站点之一。（在一个搜索引擎中键入“nslookup”就能够得到这些站点之一。）

2. 在DNS数据库中插入记录

上面的讨论只是关注如何从DNS数据库中取数据。你可能想知道，这些数据最初是怎样进入数据库中的？我们举一个特定的例子，看看这是如何完成的。假定你刚刚创建一个称为网络乌托邦（Network Utopia）的令人兴奋的新兴公司，当然你要做的第一件事是在注册登记机构注册域名networkutopia.com。**注册登记机构**（registrar）是一个商业实体，它验证域名的唯一性，将域名输入DNS数据库（如下面所讨论的那样），对所提供的服务收取少量费用。1999年以前，唯一的注册登记机构是Nework Solution，它独家经营对com、net和org域名的注册。但是现在有许多注册登记机构竞争客户，因特网名字和地址分配机构（Internet Corporation for Assigned Names and Numbers，ICANN）向各种注册登记机构授权。http://www.internic.net上列出了授权的注册登记机构。

当向某些注册登记机构注册域名networkutopia.com时，需要向该机构提供你的基本权威DNS服务器和辅助权威DNS服务器的名字和IP地址。假定该名字和IP地址是dns1.networkutopia.com和dns2.networkutopia.com及212.212.212.1和212.212.212.2。对这两个权威DNS服务器的每一个，该注册登记机构确保将一个类型NS和一个类型A的记录输入TLD com服务器。特别是对于用于networkutopai.com的基本权威服务器，该注册登记机构将下列两条资源记录插入该DNS系统中：

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

你也必须确保用于Web服务器www.networkutopia.com的类型A资源记录和用于邮件服务器mail.networkutopia.com的类型MX资源记录被输入你的权威DNS服务器中。（直到最近，每个DNS服务器中的内容都是静态配置的，例如来自系统管理员创建的配置文件。最近，在DNS协议中添加了一个更新（UPDATE）选项，允许通过DNS报文对数据库中的内容进行动态地添加或者删除。RFC 2136和RFC 3007定义了DNS动态更新。）

一旦完成所有这些步骤，人们将能够访问你的Web站点，并向你公司的雇员发送电子邮件。我们通过验证该说法的正确性，来总结DNS的讨论。这种验证也有助于充实我们已经学到的DNS知识。假定在澳大利亚的Alice要浏览www.networkutopia.com的Web页面。如前面所讨论，她的主机将首先向其本地DNS服务器发送请求。该本地服务器接着则联系一个TLD com服务器。（如果TLD com服务器的地址没有缓存，该本地DNS服务器也将必须与根DNS服务器相联系。）该TLD服务器包含前面列出的类型NS和类型A的资源记录，因为注册登记机构将这些资源记录插入了所有的TLD com服务器。该TLD com服务器向Alice的本地DNS

服务器发送一个回答，该回答包含了这两条资源记录。该本地DNS服务器则向212.212.212.1发送一个DNS查询，请求对应于www.networkutopia.com的类型A记录。该记录提供了所希望的Web服务器的IP地址，如212.212.71.4，本地DNS服务器将该地址回传给Alice主机。Alice的浏览器能够向主机212.212.71.4发起一个TCP连接，并在该连接上发送一个HTTP请求。当人们在网上冲浪时，有比满足眼球更多的事情在进行！

关注安全

DNS攻击

我们已经知道DNS是因特网基础设施的一个至关重要的组件，如果没有它，Web、电子邮件等许多重要的服务将不能正常工作。于是，我们自然要问，DNS能被攻击吗？DNS是一个易受攻击的目标吗？DNS遭到破坏后，大多数因特网应用会随它一起无法工作吗？

第一种针对DNS服务的攻击类型是DDoS带宽洪泛攻击（参见1.6节）。例如，攻击者能够向每个DNS根服务器连续不断地发送大量的分组，从而使大多数合法DNS请求得不到回答。2002年10月21日，就曾发生过这种大规模针对DNS根服务器的DDoS攻击。在这次攻击中，攻击者利用一个僵尸网络向13个DNS根服务器发送了大批的ICMP ping报文。(ICMP报文将在第4章中进行讨论。这里，知道ICMP分组是特殊类型的IP数据报就可以了。)幸运的是，这种大规模的攻击所带来的损害很小，对用户上网几乎没有影响。攻击者的确成功地将大量的分组发向了根服务器，但许多DNS根服务器都配置了分组过滤器，从而阻挡了所有发向根服务器的ICMP ping报文，这些服务器因此未受到损害。此外，大多数本地DNS服务器缓存了顶级域名服务器的IP地址，使得这样的请求过程绕过DNS根服务器。

对DNS更有效的DDoS攻击是向顶级域名服务器（例如，向所有处理.com域的顶级域名服务器）发送大量的DNS请求。这是因为，更难过滤指向DNS服务器的DNS请求，并且顶级域名服务器不像根服务器那样容易绕过。但是，这种攻击的严重性也可以通过本地DNS服务器中的缓存部分地被缓解。

DNS还会遭受其他方式的攻击。在中间人攻击中，攻击者截获来自主机的请求并返回伪造的回答。在DNS毒害攻击中，攻击者向DNS服务器发送伪造的回答，诱使服务器在其缓存中保存伪造的记录。这些攻击都能将Web用户重定向到攻击者的Web站点。然而，这些攻击难以实现，因为它们要求截获分组或遏制服务器[Skoudis 2006]。

另一种重要的DNS攻击本质上并不是一种针对DNS服务的攻击，而是充分利用DNS基础设施来对目标主机（例如，你所在大学的邮件服务器）发起DDoS攻击。在这种攻击中，攻击者向许多权威DNS服务器发送DNS请求，每个请求带有目标主机的假冒源地址。这些DNS服务器则直接向目标主机发送它们的回答。如果这些请求能够精心制作成响应比请求（字节数）大得多（所谓放大）的话，则攻击者不用产生大量的流量就能攻击目标主机。这种利用DNS的反射攻击至今为止少有成功[Mirkovic 2005]。

总之，DNS已经展示了其自身对抗攻击的健壮性。至今为止，还没有一个攻击成功地妨碍了DNS服务。虽然有成功的反射攻击，但是可以通过适当地配置DNS服务器来对抗它。

2.6 P2P应用

回顾2.1.1节中我们说过，粗略地讲，应用程序可以设计成采用客户机/服务器体系结构或者对等体系结构。在目前为止，本章中描述的应用程序（包括Web、电子邮件和DNS）都应用了客户机/服务器体系结构，它们要求有总是打开的基础设施服务器。前面讲过，使用P2P体系结构，对总是打开的基础设施服务器有最小的（或者没有）依赖。相反，任意间断连接的主机对（称为对等方）直接通信。对等方并不为服务提供商所有，而是为用户控制的桌面机和膝上机所有。

本节我们将研究三种特别适合用P2P设计的应用程序。第一种应用是文件分发，其中应用程序从单个源向大量的对等方分发文件。文件分发是研究P2P的良好起点，因为它清晰地揭示了P2P体系结构的自我扩展性。在文件分发中，我们将描述流行的BitTorrent协议。第二种应用是在对等方社区中组织并搜索信息。对于这个应用，我们将探讨一些不同的方案，其中每一种方案都已经在大规模P2P文件共享系统中得到了部署。第三种应用是Skype，它是一个相当成功的P2P因特网电话应用。

2.6.1 P2P文件分发

我们通过讨论一个非常自然的应用来开始研究P2P，这个应用是从单一服务器向大量主机（称为对等方）分发大文件。该文件可以是一个新版的Linux操作系统、现有操作系统或应用程序的一个补丁、一个MP3音乐文件或一个MPEG视频文件。在客户机/服务器文件分发中，服务器必须向每个对等方发送该文件的一个拷贝，即服务器承受了极大的负担，并且消耗了大量的服务器带宽。在P2P文件分发中，每个对等方都能够重新分发其所有的该文件的任何部分，从而协助服务器进行分发。在本书写作时（2007年春），最流行的P2P文件分发协议是BitTorrent [BitTorrent 2007]。据估计，BitTorrent占据了因特网主干流量的30%[CacheLogic 2007]。BitTorrent最初是由Bram Cohen研发的（参见本章末对Bram Cohen的专访），现在有许多遵从BitTorrent协议的BitTorrent客户机，就像有许多遵从HTTP协议的Web浏览器客户机一样。在本小节中，我们首先研究在文件分发环境中P2P体系结构的自我扩展性，然后详细地描述BitTorrent，介绍其重要的特征。

1. P2P体系结构的扩展性

为了比较客户机/服务器体系结构与对等体系结构，阐述P2P的扩展性，我们现在考虑一个用于两种体系结构类型的简单定量模型，将一个文件分发给一个固定集合。如图2-24所示，服务器和对等方使用接入链路与因特网相连。其中，u_s表示服务器接入链路的上载速率，u_i表示第i个对等方接入链路的上载速率，d_i表示第i个对等方接入链路的下载速率。此外，F表示被分发的文件长度（以比特计），N表示要获得文件拷贝的对等方的数量。分发时间（distribution time）是N个对等方得到文件拷贝所需要的时间。在下面关于分发时间的分析中，对

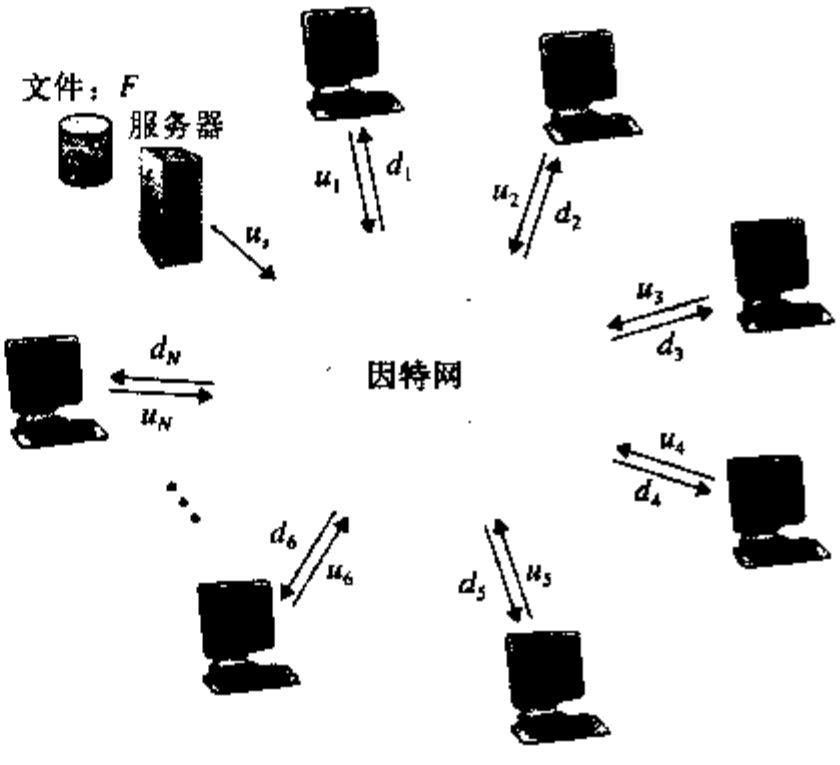


图2-24 文件分发问题示例

两种体系结构都作了简化（并且一般是准确的[Akella 2003]）的假设，即因特网核心具有大量的带宽，这意味着所有瓶颈都在网络接入链路上。我们还假设服务器和客户机没有运行任何其他网络应用，因此它们的所有上载和下载访问带宽都用于分发文件。

我们首先来确定客户机/服务器体系结构中的分发时间，设其为D_{cs}。在客户机/服务器体系结构中，没有对等方来帮助分发文件。我们进行下列观察：

- 服务器必须向N个对等方的每个都传输一个文件拷贝，因此该服务器必须传输NF比特。因为该服务器的上载速率是u_s，所以分发该文件的时间至少是NF/u_s。
- 令$d_{\min}$表示下载速率最小的对等方的下载速率，即$d_{\min} = \min\{d_1, d_2, \cdots, d_N\}$。下载速率最小的对等方不可能在$F/d_{\min}$秒之内获得该文件的所有$F$比特。因此，最小分发时间至少为$F/d_{\min}$。

将这两个观察结合起来，我们得到

$$D_{cs} \geqslant \max\left\{\frac{NF}{u_s}, \frac{F}{d_{\min}}\right\}$$

上式给出了客户机/服务器体系结构中的最小分发时间的下界。课后习题中要求读者给出服务器调度它的传输使其实际取得该下界的方法。取该下界作为实际分发时间，即

$$D_{cs} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{\min}}\right\} \tag{2-1}$$

从式（2-1）看到，对足够大的N，客户机/服务器体系结构中的分发时间由NF/u_s确定。于是，分发时间随着对等方的数量N线性地增加。因此，如果一个星期对等方的数量从1000增加到了100万，则将该文件分发到所有对等方需要的时间就要增加1000倍。

下面简单地分析一下P2P体系结构，其中每个对等方都能够帮助服务器来分发文件。也就是说，当一个对等方接收到文件数据时，它可以利用自己的上载能力重新将数据分发给其他对等方。计算P2P体系结构的分发时间在某种程度上比计算客户机/服务器体系结构的分发时间更复杂，因为分发时间取决于每个对等方如何向其他对等方分发部分该文件。但不管怎样，都可以得到对该最小分发时间的一个简单表达式[Kumar 2006]。我们先作下列观察：

- 在分发的开始，只有服务器拥有文件。为了使这些对等方得到该文件，服务器必须经其接入链路至少发送一次该文件的每个比特。因此，最小分发时间至少是F/u_s。（与客户机/服务器方案不同，服务器发送过一次的比特可能就不用再次发送了，因为对等方可以互相重新分发这些比特。）
- 与客户机/服务器体系结构相同，下载速率最小的对等方不可能在$F/d_{\min}$秒之内获得所有F比特。因此，最小分发时间至少为$F/d_{\min}$。
- 最后，系统的总上载能力等于服务器的上载速率加上每个对等方的上载速率，即$u_{total} = u_s + u_1 + \ldots + u_N$。系统必须向$N$个对等方的每个交付（上载）$F$比特，因此总共交付$NF$比特。这不可能以快于$u_{total}$的速率完成。所以，最小分发时间至少是$NF/(u_s + u_1 + \ldots + u_N)$。

将这三个观察结合起来，我们获得了P2P的最小分发时间，记为D_{P2P}。

$$D_{P2P} \geqslant \max\left\{\frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\} \tag{2-2}$$

式（2-2）给出了P2P体系结构中的最小分发时间的下界。如果假定每个对等方接收到一个比特就重新分发一个比特的话，则该重新分发方案可以实际取得这个下界[Kumar 2006]。（课后习题中将针对特殊情形证明该结果。）实际上，重新分发的是文件块而不是一个个比特。式（2-2）可以作为实际最小分发时间的很好近似。因此，取式（2-2）提供的下界作为实际的最小分发时间，即

$$D_{P2P} = \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\} \tag{2-3}$$

图2-25比较了客户机/服务器和P2P体系结构的最小分发时间，其中假定所有的对等方具有相同的上载速率u。在图2-25中，我们设置了$F/u = 1$小时，$u_s = 10u$，$d_{min} \geq u_s$。因此，在1小时中一个对等方能够传输整个文件，该服务器的传输速率是对等方上载速率的10倍，并且（为了简化起见）对等方下载速率被设置得足够大，使之不会产生影响。我们从图2-25中看到，对于客户机/服务器体系结构，随着对等方数量的增加，分发时间呈线性增长并且没有界。对于P2P体系结构，最小分发时间不仅总是小于客户机/服务器体系结构的分发时间，而且对任何多的对等方其总是小于1小时。因此，采用P2P体系结构的应用程序是可以自我扩展的，因为对等方除了是比特的消费者外还能进行重新分发。

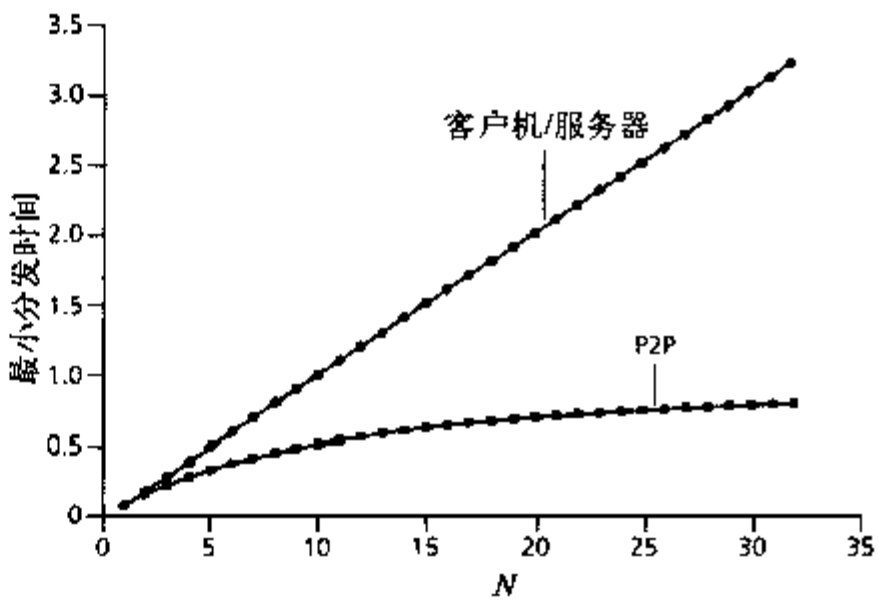


图2-25 P2P和客户机/服务器体系结构的分发时间

2. BitTorrent

BitTorrent是一种用于文件分发的流行P2P协议[BitTorrent 2007]。用BitTorrent的术语来讲，参与一个特定文件分发的所有对等方的集合称为一个洪流（torrent）。在一个洪流中，对等方彼此下载等长度的文件块，块长度通常为256 KB。当一个对等方开始加入一个洪流时，它没有文件块。随着时间的推移，它将累积越来越多的文件块。当它下载文件块时，也为其他对等方上载了多个文件块。对等方一旦获得了整个文件，它也许（自私地）离开洪流，或（大公无私地）留在洪流中并继续向其他对等方上载文件块。同时，任何对等方可以在任何时候（这时，它还没有获得整个文件）离开洪流，以后也可以重新加入洪流。

我们现在更为仔细地研究BitTorrent是如何运行的。因为BitTorrent是一个相当复杂的协议，所以这里我们仅描述它最重要的机制，浏览某些内部的细节，这将使得我们能够通过树木看森林。每个洪流具有一个基础设施节点，称为追踪器（tracker）。当一个对等方加入洪流时，它向追踪器注册，并周期性地通知追踪器它仍在洪流中。追踪器以这种方式跟踪洪流中的对等方。一个特定的洪流可能在任何时刻拥有数以百计或数以千计的对等方。

如图2-26所示，当一个新的对等方Alice加入洪流时，追踪器随机地从参与对等方集合中选择一些对等方（为了具体起见，比如说50个对等方），并将这50个对等方的IP地址发送给Alice。Alice持有对等方的这张列表，试图与该列表上的50个对等方创建并行的TCP连接。我们称所有与Alice成功地创建TCP连接的对等方为“邻近对等方”。（在图2-26中，Alice仅有三

个邻近对等方。通常，她应当有更多的对等方。）随着时间的推移，其中的一些对等方可能离开，而其他对等方（最初50个对等方以外的）可能试图与Alice创建TCP连接。因此，邻近对等方将随着时间而改变。

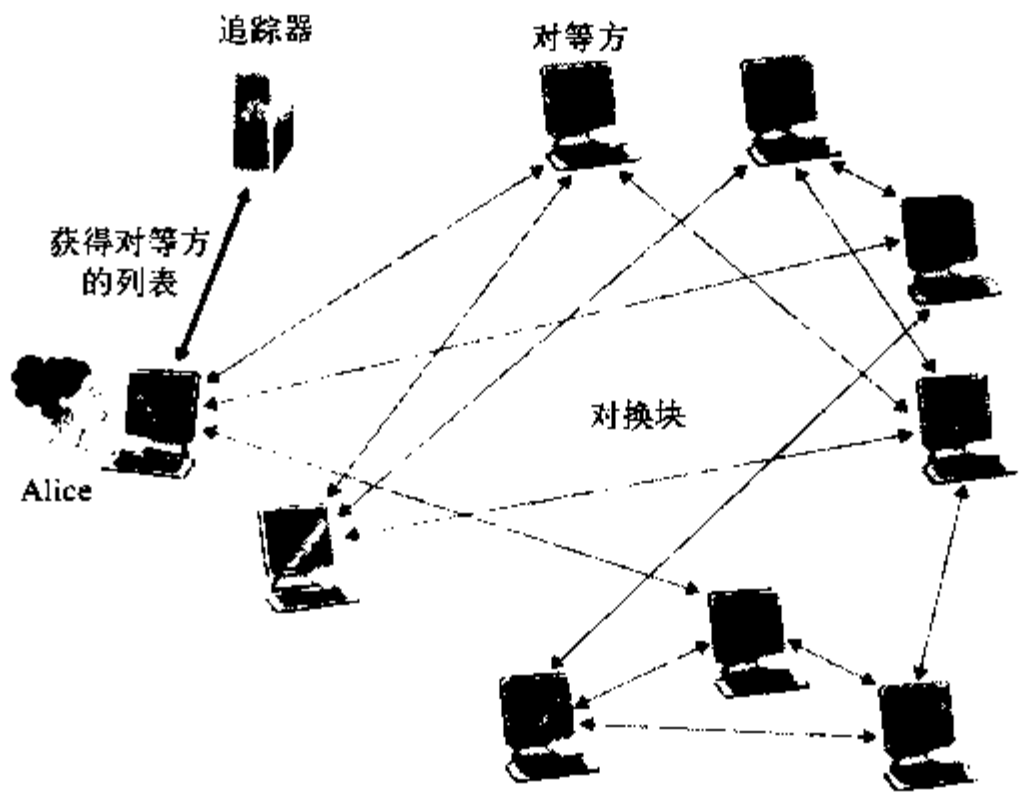


图2-26 用BitTorrent分发文件

在任何时刻，每个对等方都具有来自某文件块的子集，且不同的对等方具有不同的文件块子集。Alice周期性地（经TCP连接）询问每个邻近对等方它们所具有的块列表。如果Alice有L个邻居，那么她将获得L个块列表。因此，Alice将对她当前还没有的块发出请求（仍通过TCP连接）。

因此，在任何给定的时刻，Alice将具有块的子集并知道其邻居具有哪些块。于是，Alice将作出两个重要决定。第一，她应当向她的邻居请求哪些块呢？第二，她请求的块应当发送给她的哪些邻居？在决定请求哪些块的过程中，Alice使用一种称为**最稀罕优先**（rarest first）的技术。这种技术的思路是，根据她没有的块从她的邻居中确定最稀罕的块（最稀罕的块就是在她的邻居中拷贝数量最少的那些块），并优先请求那些最稀罕的块。按照此方式，最稀罕的块更迅速地重新分发，其目标（大致）是均衡每个块在洪流中的拷贝数量。

为了决定她响应哪个请求，BitTorrent使用了一种机灵的对换算法。其基本想法是Alice确定其邻居的优先权，这些邻居是那些当前能够以最高的速率供给她数据的。特别是，Alice对于她的每个邻居都持续地测量接收到比特的速率，确定以最高速率流入的4个邻居。然后，她将数据块发给这4个邻居。每过10秒，她重新计算该速率并可能修改这4个对等方。更重要的是，每过30秒，她要随机地选择一个另外的邻居并向它发送块。我们将这个随机选择的对等方称为Bob。因为Alice在向Bob发送数据，所以她可能成为Bob前4位上载者之一，这样的话Bob将开始向Alice发送数据。如果Bob向Alice发送数据的速率足够高，那么他也能成为Alice的前4位上载者。换言之，每过30秒Alice将随机地选择一名新的对换伙伴并开始与那位伙伴进行对换。如果这两个对等方都满足此对换要求，那么它们会将对方放入其前4位列表中并继续与对方进行对换，直到对等方之一发现了一个更好的伙伴为止。这样，对等方就能够以趋于满意的速率上载。而随机选择邻居是为了让新的对等方得到块，因此它们能够有对换的东西。除了这5个对等方（4个“前面”对等方和1个探测对等方），所有其他相邻对等方均被“阻止”，即它们不能从Alice接收到任何块。

在P2P文件共享中，**搭免费车**（free-riding）是一个常见的问题，这是指对等方从文件共

享系统中下载文件而不上载文件。BitTorrent的对换算法有效地消除了这种搭免费车问题，因为Alice为了能在一段较长的时间内以较快的速率从Bob下载比特，就必须同时以一种较快的速率向Bob上载比特。BitTorrent还具有很多其他有趣的机制，这里不再讨论了，这些机制包括部分（小块）、管道、随机优先选择、残局模型和反怠慢[Cohen 2003]。

2.6.2 在P2P区域中搜索信息

许多P2P应用程序中的一个重要部分是信息索引，即信息到主机位置的映射。在这些应用程序中，对等方动态地更新和搜索索引。由于“信息到主机位置的映射”这一说法听起来有点抽象，所以我们来看几个具体的例子。

- 在P2P文件共享系统中，通常有大量参与对等方，每个对等方都有文件（包括MP3、视频、图像和软件）可供共享。P2P文件共享系统有一个索引，它动态地跟踪这些对等方可供共享的文件。对于该对等方区域可供共享的每个文件的拷贝，该索引维护了一个记录，该记录将有关拷贝的信息（例如，如果它是一首MP3歌曲，则是该歌曲的曲名、作曲家等）映射到具有该拷贝对等方的IP地址。随着对等方进入和退出以及对等方获得文件的新拷贝，该索引进行动态的更新。例如，当一个对等方加入系统时，它通知系统它所拥有的文件索引。当某用户（如Alice）希望获得一个文件时，她搜索索引以定位该文件的拷贝位置。在定位了具有该文件拷贝的对等方以后，她就能够从这些对等方处下载该文件。一旦她具有了该完整文件，就会更新索引以将她具有的该文件新拷贝算在内。
- 在即时讯息应用程序中，有一个映射用户名与位置（IP地址）的索引。为了理解索引在这个应用程序中的重要性，考虑都在对方的好友列表中的两个用户BeautifulAlice和HandsomeBob。当HandsomeBob在具有IP地址X的主机上启动他的即时讯息客户机时，他的客户机将通知该索引HandsomeBob正以IP地址X在线。随后，当BeautifulAlice启动她的即时讯息客户机时，因为HandsomeBob位于她的好友列表中，所以她的客户机在该索引上搜索HandsomeBob并发现了他以IP地址X在线。然后，BeautifulAlice可以与具有地址X的主机创建一个直接的TCP连接，并与HandsomeBob开始即时讯息通信。除了即时讯息外，许多其他应用程序也使用索引来进行跟踪，包括因特网电话系统（参见2.6.3节）。

前面简要地讨论了BitTorrent协议，该协议只是一个文件分发协议，并没有提供任何索引和搜索文件的功能。

下面我们讨论在对等方区域中组织和搜索索引的3种方法。为了具体起见，我们假设在P2P文件共享系统中搜索一个文件。但是，这里的讨论适用于在P2P区域中搜索任何信息。

1. 集中式索引

定位文件的一个最直接的办法是提供一个**集中式索引**（centralized index），就像Napster那样。Napster是第一家大规模部署P2P文件共享应用程序的商业公司。在这种设计中，由一台大型服务器（或服务器场（farm））来提供索引服务。如图2-27所示，当用户启动P2P文件共享应用程序时，该应用程序将它的IP地址以及可供共享的文件名称（如它存储的所有MP3标题）通知索引服务器。该索引服务器从每个活动的对等方那里收集这些信息，从而建立一个集中式的动态索引，将每个文件拷贝映射到一个IP地址集合。注意，具有集中式索引的P2P文件共享系统实际上是一种P2P和客户机/服务器混合体系结构。文件分发是P2P的，但搜索是客户机/服务器的。目前很多应用程序都采用了这样的混合体系结构，如许多即时讯息应用程序。

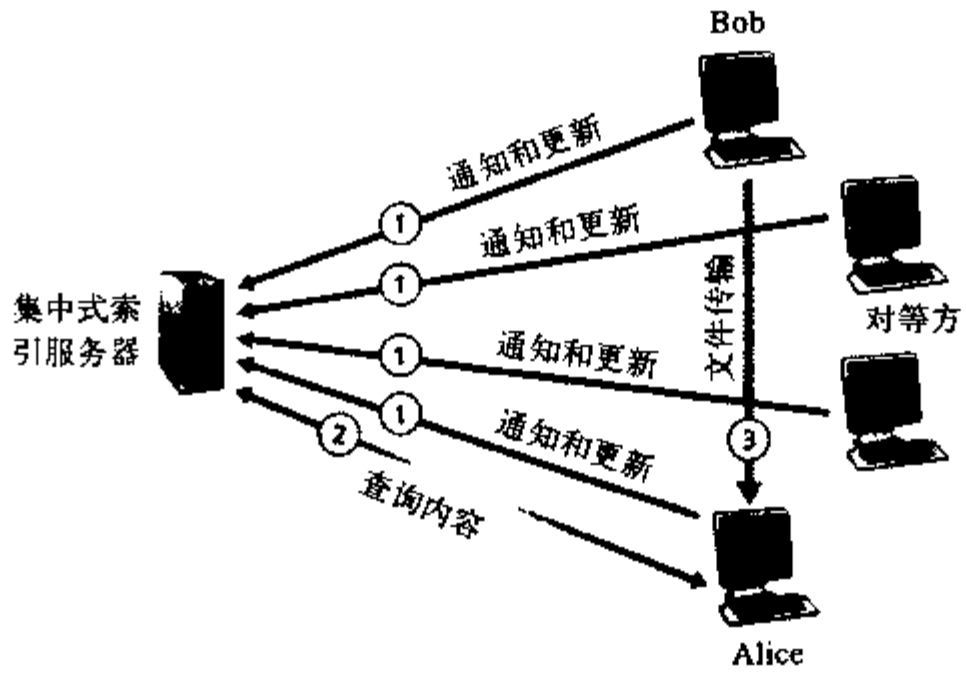


图2-27 集中式索引

使用集中式索引定位内容从概念上讲是很直接的，但是它也有几个缺点：

- 单点故障。如果索引服务器崩溃，则整个P2P应用也就随之崩溃。即使服务器场还有多余的服务器可以使用，因特网与服务器场之间的连接也可能失效，从而导致整个应用崩溃。
- 性能瓶颈和基础设施费用。在一个大型的P2P系统中，有着成千上万条的连接用户，集中式服务器必须维护一个庞大的索引，每秒钟必须对数千次查询做出响应。事实上，Napster作为2000年最流行的P2P应用，其中央服务器承受了巨大通信量问题的折磨。
- 侵犯版权。尽管这个主题超出了本书的范围，但我们还是要简要地提一下唱片录音行业所关注的问题（最低程度这样说），即P2P文件共享系统允许用户免费获取受版权保护的内容。（对于版权法和P2P的一个很好的讨论参见[von Lohmann 2003]。）当一个P2P文件共享公司有一台集中式索引服务器时，法律程序将迫使该索引服务器不得不关闭。不过关闭具有更分散的体系结构的系统要困难得多。

2. 查询洪泛

与集中式索引对立的方法是查询洪泛的完全分布式方法。查询洪泛是建立在Gnutella协议基础上的。在查询洪泛中，索引全面地分布在对等方的区域中。每个对等方索引可供共享的文件而不索引其他文件。

在Gnutella中，对等方形成了一个抽象的逻辑网络，该网络被称为**覆盖网络**（overlay network）。用图论的术语来说，如果对等方X与另一个对等方Y维护了一个TCP连接，那么我们就说在X和Y之间有一条**边**（edge）。图由所有活跃的对等方和连接的边（持续的TCP连接）组成，它定义了覆盖网络。注意，一条边不是一条物理通信链路，而是一条抽象链路，该链路可能由下面的许多物理链路组成。例如，某覆盖网络中的一条边可能表示在立陶宛的一个对等方与在巴西的一个对等方之间的TCP连接。

尽管一个覆盖网络可能有成千上万参与对等方，但一个给定的对等方通常与该覆盖网络中的少量（通常少于10个）节点连接，如图2-28所示。后面我们将解释当对等方加入和离开网络时，覆盖网络是如何构建和维护的。此时我们假定覆盖网络已经存在，先关注某对等方定位和检索内容。

在这种设计中，对等方通过已经存在的TCP连接，向覆盖网络中的相邻对等方发送报文。当Alice要定位“Network Love”时，她的客户机向她的所有邻居发送一条查询报文，该报文

包括关键词“Network Love”。Alice的所有邻居向它们的所有邻居转发该报文，这些邻居又接着向它们的所有邻居转发该报文等。显示在图2-28中的这个过程被称为**查询洪泛**（query flooding）。当一个对等方接收到一条查询报文时，它将检查该关键词是否与可供共享的任何文件相匹配。如果存在一个匹配，它向Alice回送一条“查询命中”（QueryHit）报文，该报文包含了匹配文件名和文件长度。该“查询命中”报文遵循“查询”报文的反向路径，因而使用预先存在的TCP连接。Alice以这种方式，发现了具有她想要的文件拷贝的对等方。

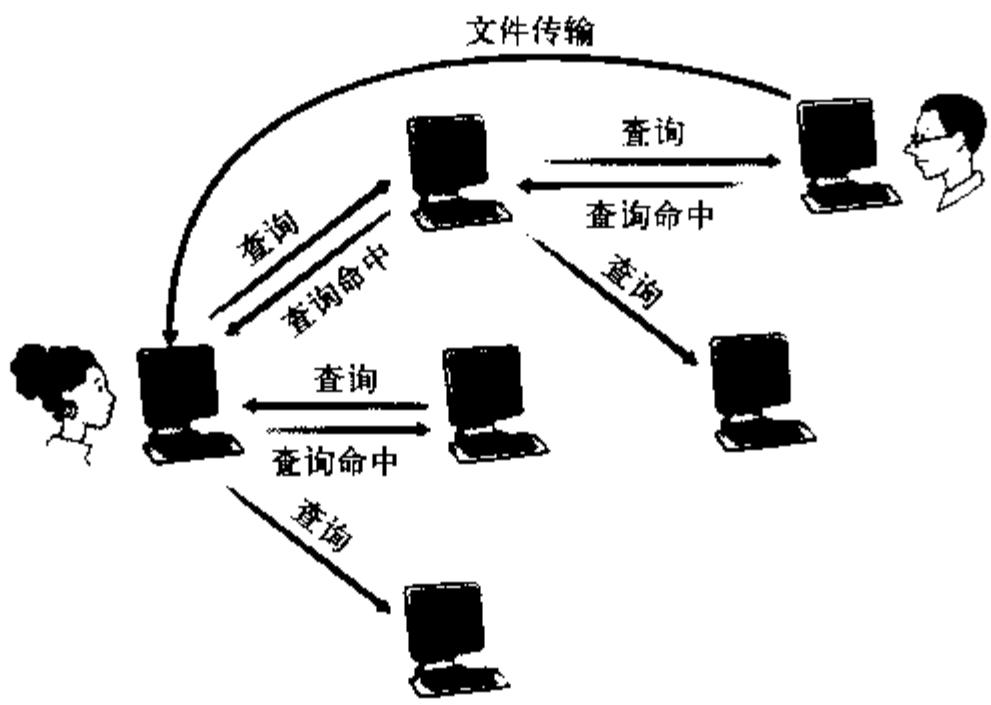


图2-28 查询洪泛

尽管这种分布式设计简单而优美，但也常因其扩展性差而受到批评。特别是对于洪泛查询，只要某对等方发起查询，该查询就会传播到整个覆盖网络中的每个其他对等方，从而在连接对等方的支撑网络（如因特网）中产生大量流量。Gnutella的设计者为解决该问题使用了**范围受限查询洪泛**（limited scope query flooding）。具体说来，当Alice发出她的初始查询报文时，该报文中的对等方计数字段被设置一个特定值（比如说7）。每当该查询报文到达一个新对等方时，该对等方在向其覆盖邻居转发该请求之前就将对等方计数字段减1。当某对等方接收到一个对等方计数字段为0的查询时，它就停止转发该查询。用这种方式，洪泛被局限于覆盖网络的某个区域。显然，这种范围受限查询洪泛减少了查询流量。然而，它也减少了对等方的数量。因此，即使你希望查找的内容在对等方区域的某个地方，你也可能不能定位到它。

覆盖网络中的一个基本问题是如何处理对等方的加入和离开。我们以初始的Gnutella设计为例，描述当新对等方加入时覆盖网络所采取的动作。假定新对等方X要加入覆盖网络。

1）对等方X必须首先发现某些已经位于覆盖网络中的其他对等方。解决这种**引导跨接问题**（bootstrap problem）的一种方法是，让X维护一张对等方的列表（IP地址），这些对等方经常在该覆盖网络中开机；另一种方法是，X能够联系维护这种列表的跟踪站点（像BitTorrent中那样）。

2）一旦访问了这样一张列表，X接下来试图与该列表上的对等方建立一个TCP连接，直到与某个对等方Y创建了一个连接为止。

3）在X和Y之间创建一个TCP连接以后，对等方X向Y发送一个ping报文。该ping报文也包括对等方计数字段。一旦接收到该ping报文，Y向它在该覆盖网络中的所有邻居转发之。这些对等方继续转发该ping报文直到该对等方计数字段为0。

4）只要一个对等方Z接收到一个ping报文，它通过该覆盖网络向X回发一个pong报文，该pong报文包括Z的IP地址。

5）当X接收到该pong报文后，它知道了该覆盖网络中的许多对等方的IP地址。然后，它能够与某些其他对等方建立TCP连接，从而从它自己向该覆盖网络中创建多条边。

我们将在课后作业中探讨当对等方离开时，覆盖网络所采取的动作。

我们现在已经讨论了查询洪泛和动态覆盖架构。总而言之，查询洪泛是一种简单的分布式P2P方案，它允许一个用户查询位于临近对等方的信息（其中“临近”是指对等方位于该覆盖网络中的少量跳数之内）。初始的Gnutella设计实现了如上所述的查询洪泛。经过多年的发展，Gnutella协议经历了重大演化，并且目前在P2P文件共享系统中充分利用了对等方的多样性。现在，Gnutella仍然非常流行，流行的P2P客户机LimeWire就采用了该协议。

3. *层次覆盖*

我们已经学习了集中式索引和洪泛查询，它们使用了完全相反的方法来定位内容。现在我们描述第三种方法——**层次覆盖设计**（hierarchical overlay design），该方法结合了上述两种方法的优秀特征。层次覆盖设计由FastTrack首创，多年来许多客户机实现了这种P2P文件共享协议，其中包括Kazaa和Morpheus。现代Gnutella也采用了这里所描述的层次覆盖设计的一个变种。

与洪泛查询类似，层次覆盖设计不使用专用的服务器（或服务器场）来跟踪和索引文件。然而，与洪泛查询不同的是，在层次覆盖设计中并非所有对等方都是平等的。特别是，与因特网高速连接并具有高可用性的对等方被指派为超级对等方，它承担更多的任务。如图2-29所示，如果某对等方不是超级对等方，则它就是一个普通对等方，并被指派为一个超级对等方的子对等方。一个超级对等方可能有几百个普通对等方作为其子对等方。

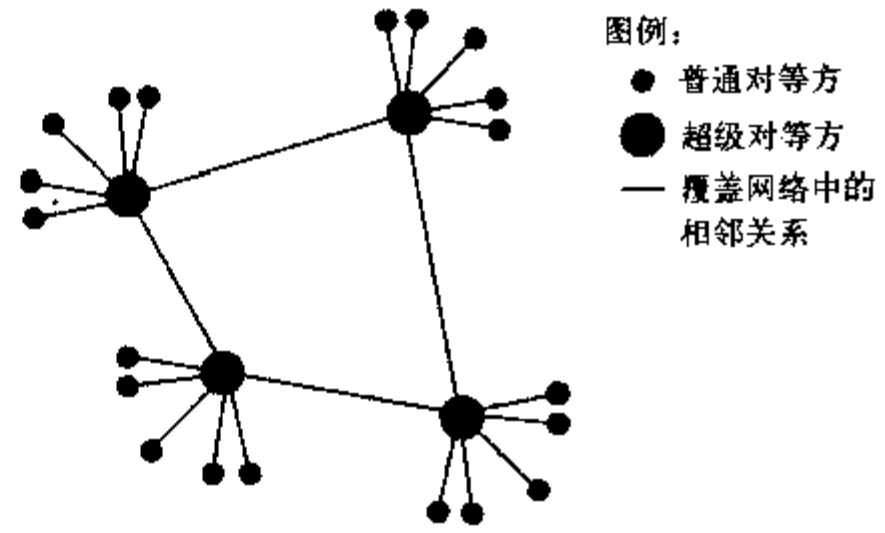


图2-29 层次覆盖网络

一个新的对等方与超级对等方之一创建一个TCP连接。然后，新对等方将它可供共享的所有文件告诉超级对等方。这样，超级对等方就维护着一个索引，该索引包括了其子对等方正在共享的所有文件的标识符、有关文件的元数据和保持这些文件的子对等方的IP地址。这样，每个超级对等方成为一个“袖珍型”索引。但与前面讨论的集中式索引相比，超级对等方不是一台专用服务器，而是普通的对等方，通常位于一个住宅中或一个大学校园中。

如果每个超级对等方及其子对等方被隔离，那么任何对等方可用的内容数量将严重受限。为了处理这种限制，超级对等方之间相互建立TCP连接，从而形成一个覆盖网络。借助于这个覆盖网络，超级对等方可以向其相邻超级对等方转发查询。该方法类似于查询洪泛，但覆盖网络中的超级对等方仅使用了范围受限查询洪泛。

当某对等方进行关键词匹配时，它向其超级对等方发送带有该关键词的查询。超级对等方则用其具有相关文件的子对等方的IP地址进行响应，这些文件的描述符与关键词（连同这些文件的标识符）相匹配。该超级对等方还可能向一个或多个相邻超级对等方转发该查询。如果某相邻对等方收到了这样一个请求，它也会用具有匹配文件的子对等方的IP地址进行响

应。超级对等方的响应遵循该覆盖网络中的反向路径。

通过将少量更强大的对等方指派为超级对等方，层次覆盖设计充分利用了对等方的异质性（heterogeneity of the peers），形成了一个层次覆盖网络的顶层，如图2-29所示。与范围受限查询洪泛设计（像初始的Gnutella设计那样）相比，层次覆盖设计允许数量多得多的对等方检查匹配，而不会产生过量的查询流量[Liang 2005]。

在结束对P2P应用程序中信息搜索的讨论之前，我们简要地提一下另一种重要的设计方法，即**分布式散列表**（Distributed Hash Table, DHT）[Stoica 2001; Rowstron 2001; Ratnasamy 2001; Zhao 2004; Maymounko 2002; Garces-Erce 2003]。对DHT的全面讨论超出了本书的范围。我们这里仅指出DHT的特点：①产生一个全分布式索引，该索引将文件标识符映射到文件位置；②允许用户（原则上）确定文件的所有位置，而不会产生过量的搜索流量。DHT得到研究界的广泛关注，流行的文件共享应用程序电骡（eMule）的核心组件Overnet就应用了DHT[Liang 2006]。

2.6.3 案例学习：Skype的P2P因特网电话

Skype是一种极为流行的P2P应用程序，在任何时候通常有七八百万用户与之连接。除了提供PC到PC的因特网电话服务外，Skype还提供PC到电话的电话服务、电话到PC的电话服务和PC到PC的视频会议服务。Skype是由创建Fastrack和Kazaa的那些人开发的，并于2005年被eBay公司以26亿美元的价格收购。

Skype以多种创新的方式使用了P2P技术，很好地展示了可以以不同于内容分发和文件共享的方式应用P2P。就像即时讯息一样，PC到PC的因特网电话本质上是P2P的，因为在应用程序的核心，成对的用户（即对等方）实时互相通信。而Skype还针对两个其他重要功能应用了P2P技术，这两个重要功能是用户定位和NAT遍历。

不仅Skype协议是专用的，而且所有的Skype分组传输（话音和控制分组）都是加密的。尽管如此，研究人员还是通过Skype网站和一些测量研究知道了Skype的一般工作方式[Baset 2006; Guha 2006; Chen 2006; Suh 2006; Ren 2006]。像FasTrack一样，Skype中的节点被组织成层次覆盖网络，其中每个节点分类为超级对等方或普通对等方。Skype包括了一个用于将Skype用户名映射到当前IP地址（和端口号）的索引。该索引跨越超级对等方进行分布。当Alice要呼叫Bob时，她的Skype客户机搜索该索引以确定Bob的当前IP地址。因为Skype协议是专用的，所以目前还不清楚该索引映射是怎样跨越超级对等方组织起来的，尽管它非常有可能使用了某种形式的DHT组织。

P2P技术也被用于Skype中继（relay）中，归属网络中的主机之间使用中继来创建呼叫。许多归属网络被配置成通过路由器（通常是无线路由器）提供对因特网的访问。这些路由器实际上不仅仅是普通路由器，通常还包括网络地址转换（NAT）。我们将在第4章中学习NAT。这里，我们仅需要知道NAT阻止位于归属网络以外的主机向位于归属网络中的主机发起通信。如果两个Skype主叫方均具有NAT，则它们都不能接受由他人发起的呼叫，从而导致无法呼叫。巧妙地使用超级对等方和中继就可以很好地解决这个问题。假定Alice向系统注册时，她被指派到一个非NAT的超级对等方。Alice可以向她的超级对等方发起一个会话，因为她的NAT只是不接受她的归属网络之外发起的会话。这样，Alice和她的超级对等方就可以通过该会话交换控制报文。当Bob向系统注册时也会发生同样的事情。当Alice要呼叫Bob时，她通知其超级对等方，该超级对等方接着通知Bob的超级对等方，Bob的超级对等方再通知Bob有来自Alice的入呼叫。如果

Bob接受该呼叫，这两个超级对等方选择一个非NAT第三方超级对等方（即中继节点），该中继节点的工作是在Alice和Bob之间转发数据。Alice和Bob的超级对等方则分别指示Alice和Bob与该中继发起会话。接下来，Alice通过Alice到中继的连接（该连接由Alice发起）向中继发送语音分组，然后该中继通过中继到Bob的连接（该连接由Bob发起）转发这些分组；从Bob到Alice的分组则反向流经这两个相同的中继连接。瞧！即使Bob和Alice都不能够接受源于它们的LAN之外的会话，他们也具有了一条能满足需求的端到端连接。中继的使用展示了P2P系统日益复杂的设计，其中对等方为其他部分（在这两个例子中为索引服务和中继）执行核心系统服务，而同时它们自己又使用了P2P系统提供的端用户服务（如文件下载、IP电话）。

Skype已经成为相当成功的因特网应用，数千万用户都采用了它。Skype的惊人发展和广泛应用，连同之前的P2P文件共享、Web和即时讯息，都在切实地证明着因特网总体体系结构设计的智慧，该设计不可能预见到未来的30年中会研制的丰富的、不断扩展的各种因特网应用。为因特网应用程序提供的网络服务已经充分证明了可以研发数以千计的应用程序，其中网络服务包括无连接数据报传输（UDP）、面向连接的可靠数据报传输（TCP）、套接字接口、寻址和命名（DNS）等。因为这些应用程序都位于因特网协议栈的现有4个较低层次之上，所以它们仅涉及研发与在端系统中使用的P2P软件一样的新式客户机/服务器。这也使得可以迅速地部署和采用这些应用程序。

2.7 TCP套接字编程

我们已经看到了一些重要的网络应用，下面就探讨一下网络应用程序是如何实际编写的。在本节中，我们将编写使用TCP的应用程序；在下节中我们将编写使用UDP的程序。

在2.1节讲过，网络应用程序的核心是由一对程序（即客户机程序和服务器程序）组成的，它们位于两个不同的端系统中。当运行这两个程序时，创建了一个客户机进程和一个服务器进程，同时它们彼此之间通过从套接字读出和写入数据进行通信。开发者开发一个网络应用时，其主要任务就是编写客户机程序和服务器程序的代码。

网络应用程序有两类。一类是网络应用程序，它们由如RFC所定义的标准协议的实现。对于这类实现，客户机程序和服务器程序必须满足该RFC所定义的规则。例如，某客户机程序可能是FTP客户机端的一种实现，如2.3节所描述，并明确由RFC 959定义；而其服务器程序可能是FTP服务器的一种实现，也明确由RFC 959定义。如果一个开发者编写客户机程序的代码，另一个开发者编写服务器程序的代码，并且两者都完全遵从该RFC的规则，那么这两个程序将能够交互操作。实际上，今天大多数网络应用程序涉及客户机和服务器程序间的通信，这些程序都是由不同的程序员单独开发的（例如， 与Apache Web服务器通信的Firefox浏览器，或PC机上用于向Linux FTP服务器上传文件的FTP客户机）。当客户机程序或服务器程序实现了由某RFC定义的协议时，应该使用与协议相关的端口号。（端口号已在2.1节中进行了简要的讨论，第3章将进行更详细的讨论。）

另一类网络应用程序是专用的网络应用程序。在这种情况下，由客户机程序和服务器程序使用的应用层协议不必符合任何现有RFC。一个开发者（或开发团队）创建了客户机程序和服务器程序，并且该开发者用他的代码完全控制着程序的功能。但是因为这些代码并没有实现公共的协议，其他独立的开发者将不能开发出和该应用程序交互的代码。当研发一个专用应用程序时，开发者必须小心，不要使用已在RFC中定义的周知端口号。

在本节和下节中，我们将讨论研发一个专用客户机/服务器应用程序中的一些关键问题。在研发阶段，开发者必须最先做的一个决定是，应用程序是运行在TCP上还是运行在UDP上。

前面讲过TCP是面向连接的，并且为两个端系统之间的数据流动提供可靠的字节流通道。UDP是无连接的，从一个端系统向另一个端系统发送独立的数据分组，不对交付提供任何保证。

在本节中，我们开发一个简单的、运行在TCP之上的客户机程序；在下一节中，我们开发一个简单的、运行在UDP之上的客户机应用程序。我们用Java语言写这些简单的TCP和UDP程序。也可以用C或C++语言来编写这些程序，而我们选择用Java有以下几个原因：用Java写的程序更为清晰简洁，Java代码行数更少，并且向新程序员解释每一行代码不会有太大困难。如果你不熟悉Java，也用不着担心，只要你有过一些用其他语言编程的经验，就应该能看得懂下面的代码。

愿意用C语言编制客户机/服务器程序的读者，可以参考以下优秀的资料[Donahoo 2000; Stevens 1997; Frost 1994; Kurose 1996]。

2.7.1 TCP套接字编程

2.1节讲过，运行在不同机器上的进程彼此通过向套接字发送报文来进行通信。我们说过每个进程好比是一座房子，进程的套接字就好比是一个门。如图2-30所示，套接字是应用进程和TCP之间的门。应用程序开发者在套接字的应用层一侧可以控制所有东西，但是，它不能控制运输层一侧。（充其量，应用程序开发者能设置一些TCP参数，如最大缓冲长度和最大报文段长度等。）

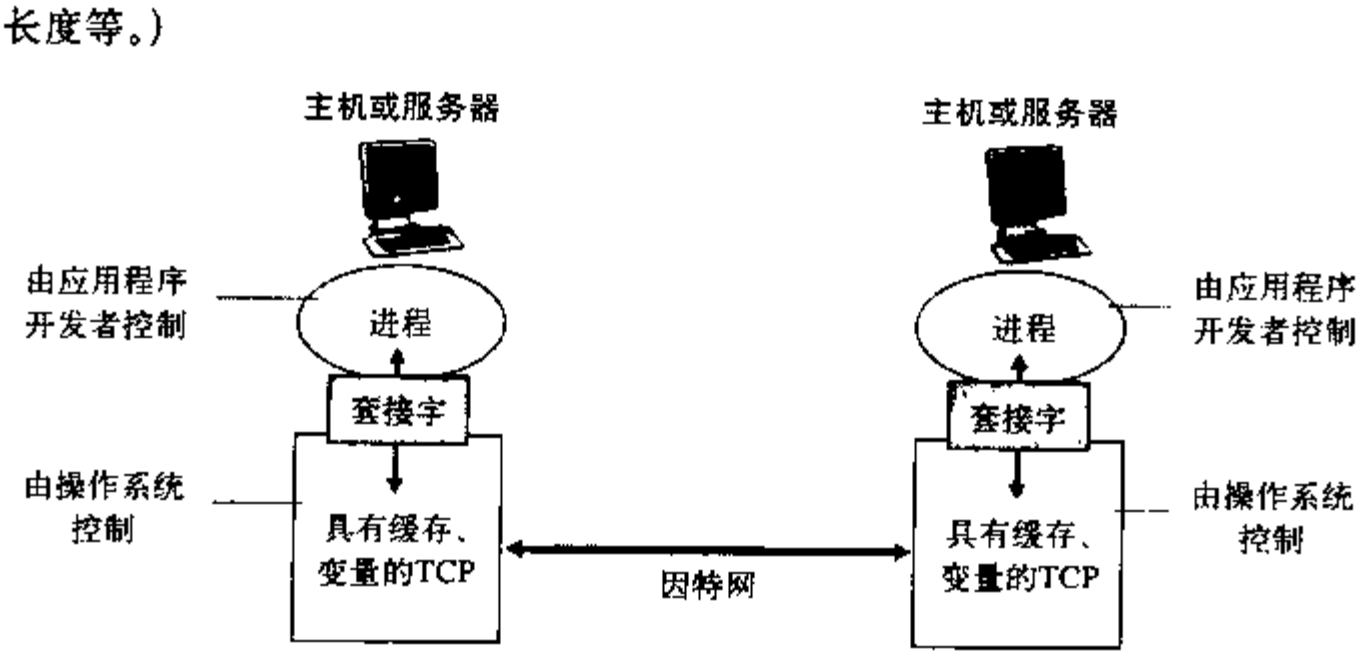


图2-30 进程通过TCP套接字通信

现在我们更仔细地观察客户机程序和服务器程序之间的交互。客户机负责发起与服务器的联系。服务器为了能对客户机程序发起的连接做出响应，必须先准备好。这意味着两件事。第一，服务器程序不能处于睡眠状态；在客户机程序试图发起连接之前，它必须已经作为一个进程在系统中运行。第二，服务器程序必须有某种类型的门（更为精确的说是套接字），欢迎来自运行在任意机器上的客户机程序发起的连接。通过使用房子/门来比喻进程/套接字，我们有时将客户机发起连接称为“敲欢迎之门”。

当服务器进程运行时，客户机进程可以向服务器发起一个TCP连接。在客户机程序中，这可以通过创建一个套接字来完成。当客户机创建它的套接字时，它指定服务器进程的地址，即服务器的IP地址和进程的端口号。一旦在客户机程序中生成套接字，客户机的TCP与服务器的TCP发起三次握手并建立一个TCP连接。这个三次握手过程发生在运输层，对于客户机程序和服务器程序是完全透明的。

在三次握手期间，客户机进程敲服务器进程的欢迎之门。当服务器“听”到敲门时，它

将创建一个新门（更精确地讲是一个新套接字），为某个特定的客户机程序服务。在我们下面的例子中，欢迎之门是一个ServerSocket对象，我们称之为welcomeSocket。当一个客户机程序“敲门”时，服务器程序调用welcomeSocket中的accept()方法，这将为该客户机程序创建一个新门。在握手的最后阶段，客户机套接字和服务器套接字之间已存在一个TCP连接。因此，我们称新的套接字为服务器的**连接套接字**（connection socket）。

从应用程序的观点来看，TCP连接是客户机套接字和服务器连接套接字之间的一个直接的虚拟管道。客户机进程可以向它的套接字发送任意字节的数据，TCP保证服务器进程能够按发送的顺序接收（通过连接套接字）到每个字节的数据。TCP因此在客户机进程和服务器进程之间提供了**可靠字节流服务**(reliable byte-stream service)。此外，就像人们可以从同一个门进出一样，客户机进程也能从它的套接字中接收字节，类似地，服务器进程也能向它的连接套接字发送字节。这个过程如图2-31所示。由于套接字在客户机/服务器应用程序中起着核心作用，因此客户机/服务器应用程序开发也称为套接字编程。

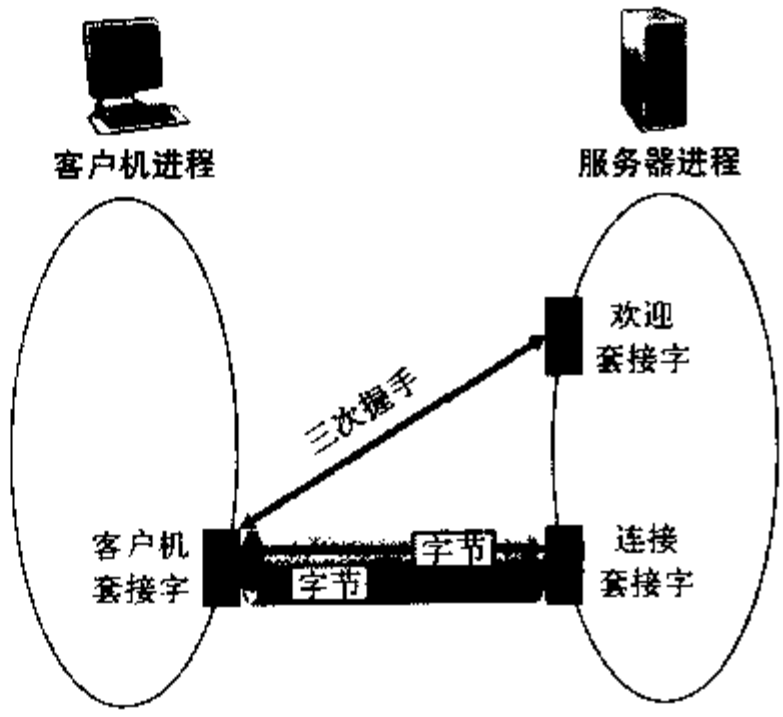


图2-31 客户机套接字、欢迎套接字和连接套接字

在给出客户机/服务器应用程序例子之前，有必要讨论一下流的概念。**流**（stream）是流入和流出进程的字符序列。对一个进程来说，每条流或者是**输入流**（input stream），或者是**输出流**（output stream）。如果流是一条输入流，则它与该进程的某个输入源相连，例如标准输入（键盘）或一个套接字（来自因特网的数据流入该套接字）。如果流是一条输出流，则它与该进程的某个输出源相连，例如标准输出（监视器）或者一个套接字（数据由此流入因特网）。

2.7.2 一个Java客户机/服务器应用程序例子

我们将使用下面简单的客户机/服务器应用程序，来演示如何使用TCP和UDP套接字编程：

1）一台客户机从其标准输入（键盘）读取一行字符，并通过其套接字将该行发送到服务器。

2）服务器从其连接套接字读取一行字符。

3）服务器将该行字符转换成大写。

4）服务器将修改后的行通过其连接套接字再回发给客户机。

5）客户机从其套接字中读取修改后的行，然后将该行在其标准输出（监视器）上打印出来。

图2-32示意了客户机和服务器的与套接字相关的主要活动。

下面我们提供该应用程序的客户机/服务器程序对的TCP实现。在每个程序后面我们提供了详细的、逐行的分析。客户机程序称为TCPClient.java，服务器程序称为TCPServer.java。为了强调一些重要问题，我们有意提供一些能说明问题却并不完美的代码。“优秀的代码”无疑还要有更多辅助代码行。

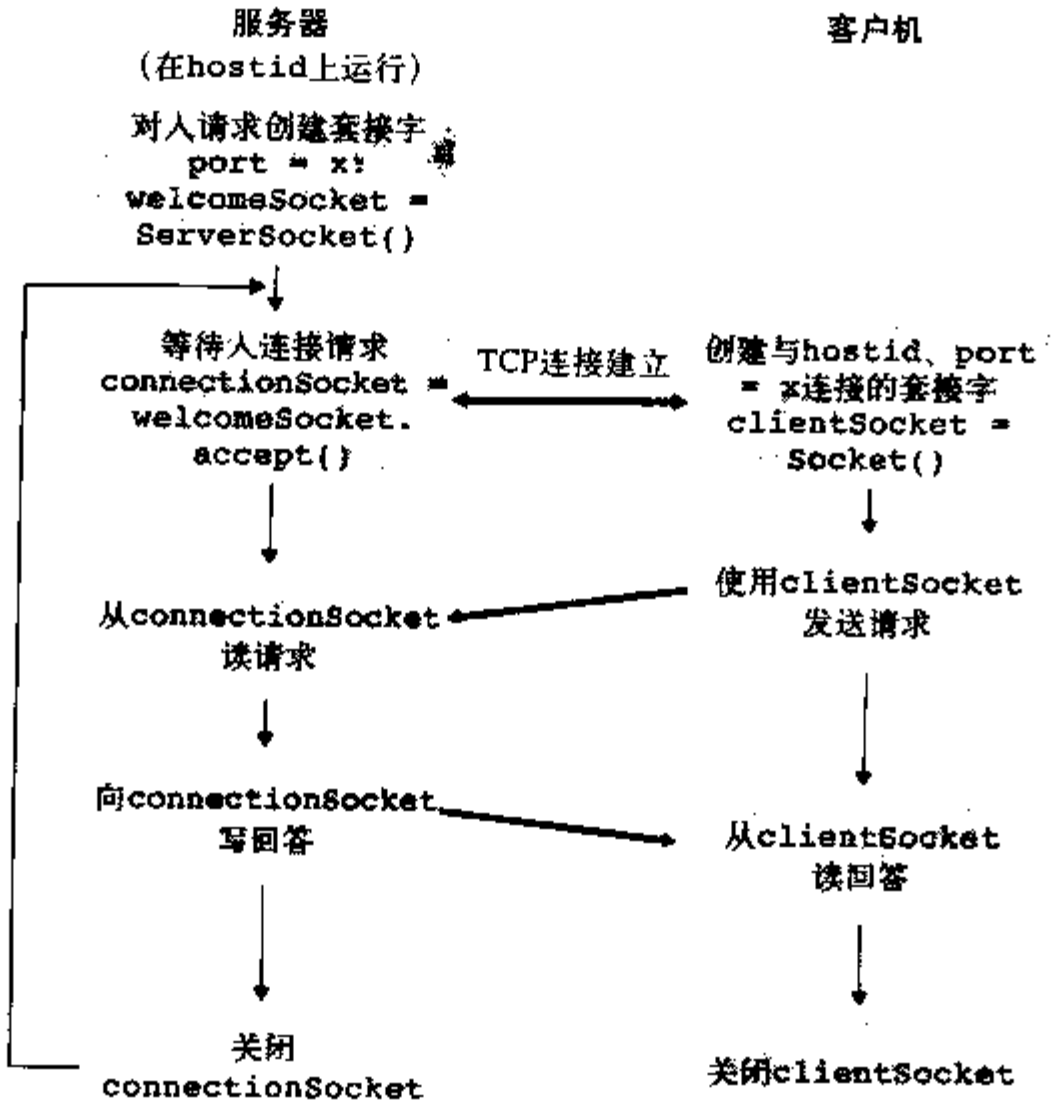


图2-32 客户机/服务器应用程序使用面向连接的运输服务

一旦这两个程序在各自的主机上编译后，服务器程序就先在服务器主机上运行，这将在服务器主机上生成一个进程。如上面所讨论的那样，服务器进程等待由客户机进程发起的连接。当客户机程序执行时，在客户机上创建一个进程，该进程与服务器联系并与它建立一个TCP连接。客户机上的用户然后使用该应用程序发送一行字符，然后接收经过大写转换的行。

1. TCPClient.java

下面是应用程序的客户机端的代码：

```
import java.io.*;
import java.net.*;
class TCPClient {
   public static void main(String argv[]) throws Exception
   {
      String sentence;
      String modifiedSentence;
      BufferedReader inFromUser = new BufferedReader(
               new InputStreamReader(System.in));
      Socket clientSocket = new Socket("hostname", 6789);
      DataOutputStream outToServer = new DataOutputStream(
               clientSocket.getOutputStream());
      BufferedReader inFromServer =
         new BufferedReader(new InputStreamReader(
               clientSocket.getInputStream()));
      sentence = inFromUser.readLine();
      outToServer.writeBytes(sentence + '\n');
      modifiedSentence = inFromServer.readLine();
      System.out.println("FROM SERVER: " +
                         modifiedSentence);
```

```
        clientSocket.close();
    }
}
```

程序TCPClient创建了3条流和一个套接字，如图2-33所示。我们称这个套接字为clientSocket。inFromUser流是程序的输入流，它连接到标准输入（即键盘）。当用户在键盘上输入字符时，这些字符流入inFromUser流中。inFromServer流是程序的另一个输入流，它连接到套接字。从网络来的字符流入inFromServer流。最后，outToServer是程序的输出流，它也连接到套接字。客户机程序发送到网络的字符流入到outToServer流中。

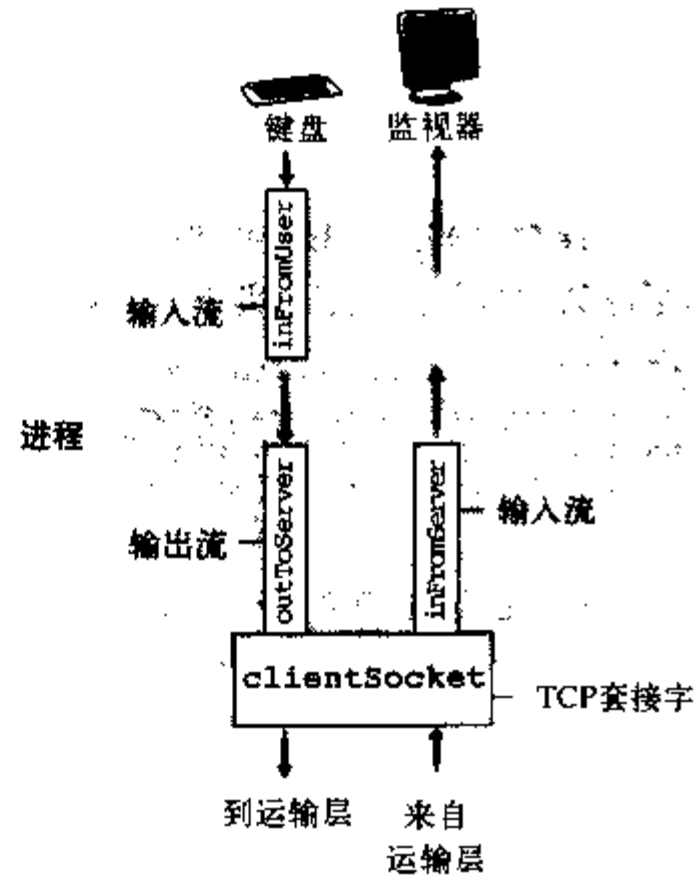


图2-33 TCPClient具有3条流，字符通过这些流流动

我们现在来看看以下几行代码。

```
import java.io.*;
import java.net.*;
```

java.io和java.net是Java包。java.io包中包含输入流类和输出流类，特别是，java.io包包含了BufferedReader类和DataOutputStream类，程序使用这些类创建上述3个流。java.net包提供了支持网络功能的类，特别是，它包含了Socket类和ServerSocket类。该程序的clientSocket对象衍生于Socket类。

```
class TCPClient {
   public static void main(String argv[]) throws Exception
         {......}
}
```

目前为止，我们所见到的这些代码是大多数Java程序开始的标准部分。第3行是一个类定义块的开始。关键词class表明TCPClient类的类定义的开始。类包含了变量和方法。花括号将类定义的变量和方法括起来，该花括号标志着类定义块的开始和结束。TCPClient类没有类变量，仅有一个方法，即main()方法。方法类似于C语言中的函数或过程，Java语言中的main方法同样类似于C和C++中的main函数。当Java解释器执行一个应用程序时（通过调用应用程序的控制类），从调用类的main()方法开始执行。该main方法再调用运行该应用程序所需的所有其他方法。作为Java套接字编程的入门介绍，我们忽略public、static、void main和throws Exceptions等关键字（虽然在代码中必须包含它们）。

```
String sentence;
String modifiedSentence;
```

以上两行声明了两个String类型的对象。sentence对象包括用户输入和送到服务器的字符串。modifiedSentence对象是从服务器得到并送到用户标准输出的字符串。

```
BufferedReader inFromUser = new BufferedReader(
      new InputStreamReader(System.in));
```

上面一行创建了一个类型为BufferedReader的inFromUser流对象，输入流用

System.in初始化，System.in将流连接到标准输入。该命令允许客户机从其键盘读入文本。

```
Socket clientSocket = new Socket("hostname", 6789);
```

这一行创建了一个类型为Socket的clientSocket对象。它也发起客户机和服务器之间的TCP连接。字符串"host-name"必须用服务器的主机名代替（例如，"apple.poly.edu"）。在TCP连接实际发起之前，客户机必须先进行一次DNS查询，以便通过主机名获得主机的IP地址。6789是一个端口号，你可以使用不同的端口号，但必须保证所使用的端口号与应用程序的服务器端所使用的端口号相同。如前所述，主机IP地址连同该应用程序的端口号标识了该服务器进程。

```
DataOutputStream outToServer =
   new DataOutputStream(clientSocket.getOutputStream());
BufferedReader inFromServer =
   new BufferedReader(new inputStreamReader(
                    clientSocket.getInputStream()));
```

上面两行创建了两个连接到套接字的流对象。outToServer流为进程提供了到套接字的输出。InFromServer流为进程提供了来自套接字的输入（参见图2-33）。

```
sentence = inFromUser.readLine();
```

这一行将用户输入的一行读到sentence字符串中。字符串sentence一直收集字符，直到用户敲下回车键终止这一行为止。这一行将来自标准输入的内容通过inFromUser流传递到sentence字符串中。

```
outToServer.writeBytes(sentence + '\n');
```

上面这一行将增加了回车符的sentence字符串发送到outToServer流中。增加了回车符的sentence字符串流经客户机套接字并进入TCP管道中。客户机然后等待接收来自服务器的字符。

```
modifiedSentence = inFromServer.readLine();
```

当字符到达服务器时，它们流过inFromServer流，并进入modifiedSentence字符串中。modifiedSentence中的字符不断累积，遇到回车符后结束该行。

```
System.out.println("FROM SERVER " + modifiedSentence);
```

上面一行将服务器返回来的modifiedSentence字符串打印到监视器上。

```
clientSocket.close () ;
```

最后一行关闭了套接字，因此客户机和服务器之间的TCP连接也同时被关闭。这引起客户机的TCP向服务器的TCP发送一个TCP报文（参见3.5节）。

2. TCPServer.java

现在我们看一下服务器程序。

```
import java.io.*;
import java.net.*;
class TCPServer {
   public static void main(String argv[]) throws Exception
      {
         String clientSentence;
         String capitalizedSentence;
         ServerSocket welcomeSocket = new ServerSocket
            (6789);
         while(true) {
```

```
                Socket connectionSocket = welcomeSocket.
                   accept();
                BufferedReader inFromClient =
                   new BufferedReader(new InputStreamReader(
                      connectionSocket.getInputStream()));
                DataOutputStream outToClient =
                   new DataOutputStream(
                      connectionSocket.getOutputStream());
                clientSentence = inFromClient.readLine();
                capitalizedSentence =
                      clientSentence.toUpperCase() + '\n';
                outToClient.writeBytes(capitalizedSentence);
             }
          }
       }
```

TCPServer和TCPClient有很多相似之处。现在我们来看看TCPServer.java中的一些代码行。我们将不再对与TCPClient.java中相同或类似的行进行说明。

在TCPServer中，第一行明显不同于TCPClient中的行：

```
ServerSocket welcomeSocket = new ServerSocket(6789);
```

该行创建了一个类型为ServerSocket的welcomeSocket对象。welcomeSocket是一种能监听某客户机“敲门”声的门。welcomeSocket在6789端口上监听。下一行是：

```
Socket connectionSocket = welcomeSocket.accept();
```

当某客户机敲击welcomeSocket时，该行创建一个新套接字，称为connectionSocket。这个套接字也使用6789号端口。（在第3章，我们将解释为什么两个套接字具有同样的端口号。）之后，TCP在客户机的clientSocket和服务器端的connectionSocket之间建立了一条直接的虚拟管道。该客户机和服务器则可以通过该管道彼此发送字节，并且发送的所有字节将按序到达对方。随着connectionSocket的建立，该服务器能继续使用welcomeSocekt监听来自应用程序的其他客户机的连接请求。（这个版本的程序实际上并不能监听更多的连接请求，但是通过修改为使用多线程就可以做到。）该程序创建几个流对象，与在clientSocket中创建流对象类似。现在来考虑：

```
capitalizedSentence = clientSentence.toUpperCase() + '\n';
```

这条命令是该应用程序的核心。它将客户机发送过来的行中的字符转换为大写，然后加上回车符后发送回去。它使用了toUpperCase()方法。该程序中的所有其他指令都是配套的，它们用来和客户机通信。

为了测试这一对程序，你在一台主机上安装并编译TCPClient.java，在另一台主机上安装并编译TCPServer.java。需要确认在TCPClient.java中包括了服务器的适当主机名。然后，你在服务器上运行编译好的服务器程序TCPServer.class。这将在服务器端创建一个进程，它处于空闲状态，直到某个客户机与之连接。接下来，在客户机上运行编译好的客户机程序TCPClient.class。这将在客户机创建一个进程，并在客户机进程和服务器进程之间建立一个TCP连接。最后，使用这个程序输入一行字符并按回车键。

如果想开发自己的客户机/服务器应用程序，可以稍微修改一下这个程序。例如，不是将所有字符都转换为大写，而是让服务器统计字母“s”出现的次数，然后返回统计的个数。

2.8 UDP套接字编程

在前面一节中，我们学习了何时两个进程通过TCP进行通信，通信时好像在进程之间有

一条管道。这条管道一直保持，除非其中一个进程关闭它。当两进程之一想向另一进程发送字节时，它直接将这些字节插入管道即可。发送进程不必在这些字节上附加目的地址，因为这条管道逻辑上已与目的地址连接了。此外，该管道提供了一条可靠的字节流通道，即接收进程接收到的字节顺序和发送进程插入管道的字节顺序完全一致。

UDP也允许运行在不同机器上的两个（或多个）进程彼此通信。然而，UDP和TCP在许多基本方式上有不同之处。首先，UDP是一种无连接的服务，即在两个进程间没有创建管道时所需的初始握手阶段。因为UDP没有管道，所以当一个进程需要向另一个进程发送一批字节时，该发送进程需要为这批字节附上目的进程地址。并且，该过程对于每批由发送进程所发送的字节都必须重复做。打个比方，考虑20个人分乘5辆出租汽车到相同目的地，当这些人上车时，他们都必须分别告诉司机目的地址。因此，UDP类似于出租汽车服务。目的地址由二元组组成：目的主机的IP地址和目的进程的端口号。我们将带有IP目的地址和端口号的一批字节数据称为“分组”。UDP提供了一种不可靠面向报文的服务模型，它尽力而为地向目的地交付这批字节，但不担保这批字节的确能被交付。UDP是面向报文的，是指在发送方单次操作所发送的一批字节在接收方作为一个批次来交付，这与TCP的字节流语义形成对照。UDP是尽力而为的，是指UDP不能确保这批字节确实能被交付。因此UDP服务在几个方面与TCP的可靠字节流服务模型形成鲜明对比。

创建一个分组后，发送进程通过套接字将分组送到网络。我们继续使用出租车的类比，在发送套接字的另一端，有一辆出租车在等待这个分组。随后，出租车带着这个分组向目的地址方向开去。然而，该出租车并不能保证这个分组到最终目的地：出租车可能中途抛锚或遇到了某些其他不可预见的问题。换句话说，UDP为通信进程提供了不可靠的运输服务，它并不确保数据报能到达它的最终目的地。

在本节中，我们将通过重新开发上一节的相同应用程序来举例说明套接字编程，但这次是在UDP之上。我们也将看到用Java编制UDP程序和用它编制TCP程序有很多不同之处。特别是我们应当注意以下几点：① 两个进程之间没有进行初始握手，因此不需要欢迎套接字；② 没有流与套接字相联系；③ 发送主机通过将IP目的地址和端口号与它发送每批字节相联系，生成分组；④ 接收进程必须拆开每个所接收到的分组，获得该分组的信息字节。再次回想一下我们前面的简单应用：

1）一台客户机从其标准输入（键盘）读取一行字符，并通过其套接字将该行发送到服务器。

2）服务器从其套接字读取一行数据。

3）服务器将该行字符转换成大写。

4）服务器将修改后的行通过其连接套接字再回发给客户机。

5）客户机从其套接字中读取修改后的行，然后将该行在其标准输出（监视器）上打印出来。

图2-34突出显示了客户机和服务器的与套接字相关的主要活动，两者通过UDP无连接运输服务通信。

1. UDPClient.java

下面是该应用程序客户机端的代码：

```
import java.io.*;
import java.net.*;
class UDPClient {
   public static void main(String args[]) throws Exception
   {
      BufferedReader inFromUser =
         new BufferedReader(new InputStreamReader
                  (System.in));
```

```
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress =
                        InetAddress.getByName("hostname");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket =
           new DatagramPacket(sendData, sendData.length,
                        IPAddress, 9876);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket =
           new DatagramPacket(receiveData,
                        receiveData.length);
        clientSocket.receive(receivePacket);
        String modifiedSentence =
           new String(receivePacket.getData());
        System.out.println("FROM SERVER:" +
                        modifiedSentence);
        clientSocket.close();
    }
}
```

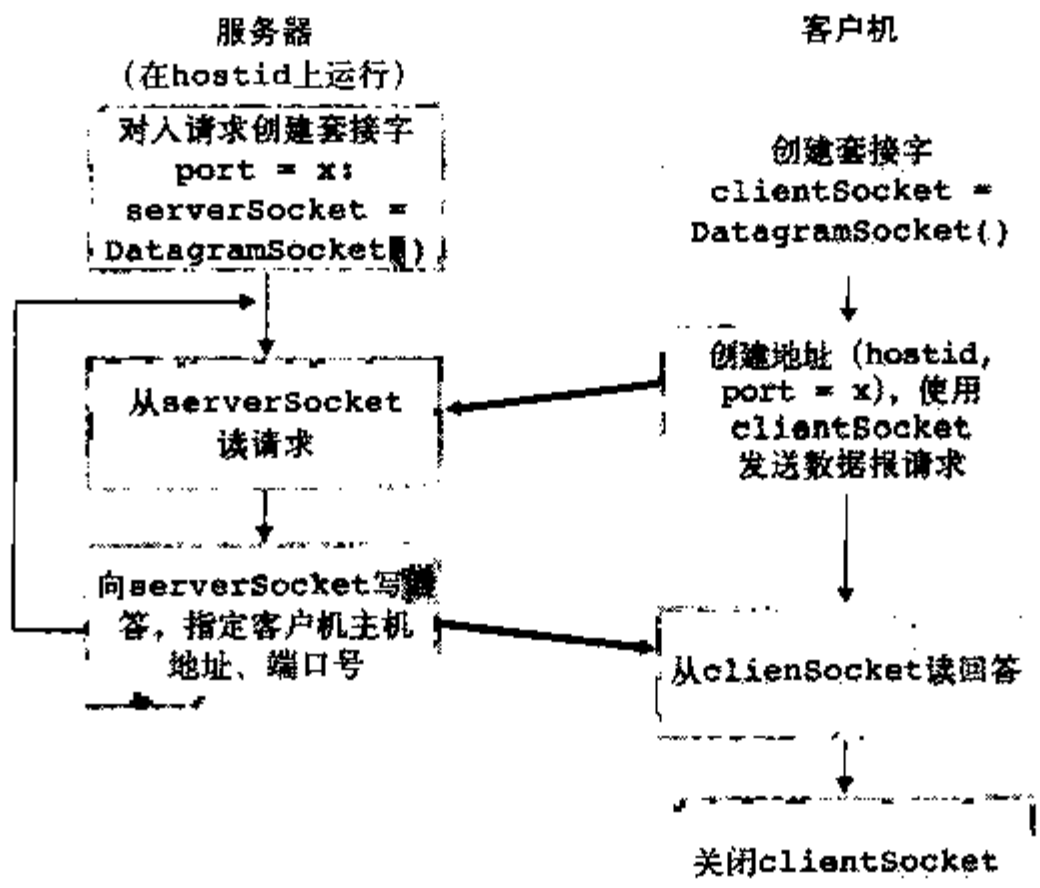


图2-34 使用无连接运输服务的客户机/服务器应用程序

UDPClient.java程序构造了一条流和一个套接字，如图2-35所示。该套接字被称为clientSocket，它的类型为DatagramSocket。注意在客户机UDP使用了与TCP不同类型的套接字。尤其是，UDP客户机使用的是DatagramSocket，而TCP客户机使用的是Socket。inFromUser流是该程序的一条输入流，它与标准输入相联系，即与键盘相联系。在前面的TCP版本的程序中也有一条等价的流。当用户从键盘输入字符时，这些字符进入inFromUser流中。但和TCP相反的是，没有流（输入或输出）与套接字相联系。相反，UDP不是将字节送入与Socket对象相联系的流，而是将一个个分组通过DatagramSocket对象直接发送出去。

我们现在看一下明显不同于TCPClient.java中的代码行。

```
DatagramSocket clientSocket = new DatagramSocket();
```

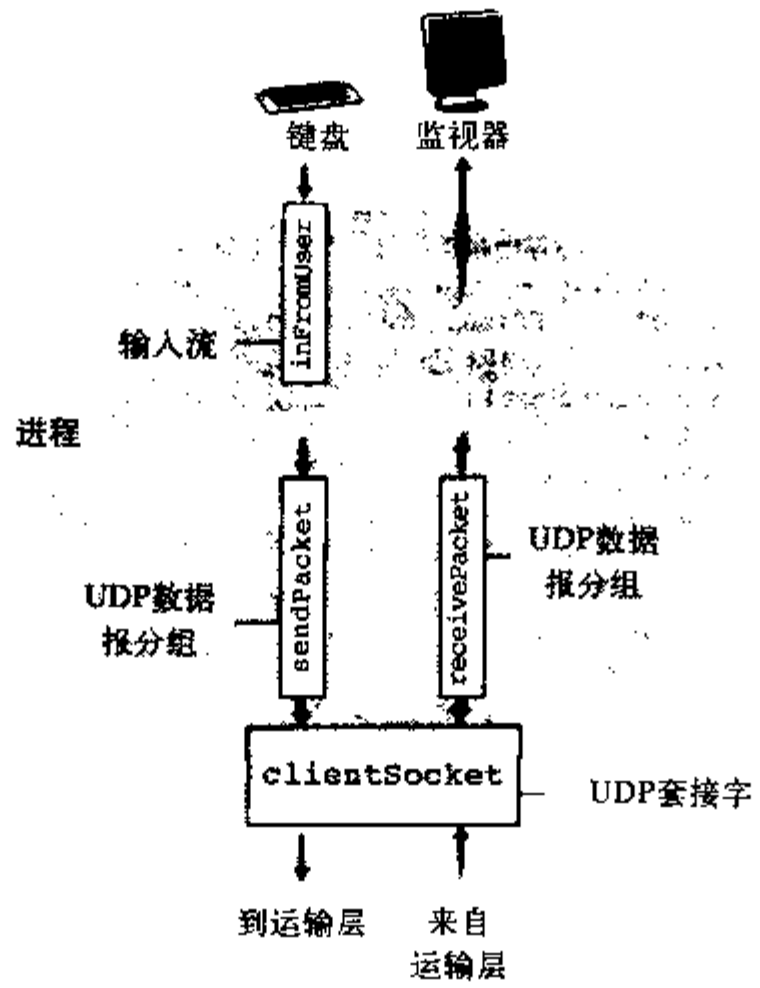


图2-35 UDPClient有一条流，其套接字接受来自进程的分组并向进程交付分组

该行创建了一个DatagramSocket类型的clientSocket对象。和TCPClient.java不同的是，该行没有发起一个TCP连接。尤其是在执行这一行的时候，客户机主机并没有与服务器联系。由于这个原因，构造函数DatagramSocket()并没有把服务器主机名或端口号作为参数。使用我们的门/管道类比的话，上述这行的执行为客户机进程创建了门，但并没有在进程之间创建一个管道。

```
InetAddress IPAddress = InetAddress.getByName("hostname");
```

为了向目的进程发送数据，我们需要获得目的进程的地址。该地址的一部分是目的主机的IP地址。上面这行调用了DNS查询，将主机名（在本例中，由开发人员在代码中提供）转换成IP地址。TCP版本的客户机程序中也调用了DNS查询，尽管那时是隐式而不是显式完成的。getByName()方法将服务器的主机名作为参数，返回该服务器的IP地址，并将该地址的值赋予类型为InetAddress的IPAddress对象。

```
byte[] sendData = new byte[1024];
byte[] receiveData = new byte[1024];
```

sendData和receiveData这两个字节数组分别存放客户机发送和接收的数据。

```
sendData = sentence.getBytes();
```

这一行本质上是进行一次类型转换。它获取字符串内容并将它重命名为sendData，sendData是一个字节数组。

```
DatagramPacket sendPacket = new DatagramPacket(
   sendData, sendData.length, IPAddress, 9876);
```

该行构造了一个分组sendPacket，客户机通过套接字将它发送到网络。这个分组包含sendData分组中的数据、数据的长度、服务器的IP地址和该应用程序的端口号（我们设它为9876）。注意到sendPacket的类型是DatagramPacket。

```
clientSocket.send(sendPacket);
```

在上面这一行中，对象clientSocket的send()方法得到了刚构造的分组，再通过clientSocket向网络发送。需再次注意，UDP发送一行字符的方式和TCP又有很大不同。TCP是直接将字符串插到流中，该流与服务器之间有一条直接的逻辑连接。UDP创建一个包含服务器地址的分组。发送完该分组后，客户机等待接收来自服务器的分组。

```
DatagramPacket receivePacket =
   new DatagramPacket(receiveData, receiveData.length);
```

在上面这一行中，在等待来自服务器的分组时，该客户机创建了放置分组的地方，即receivePacket对象，该对象的类型为DatagramPacket。

```
clientSocket.receive(receivePacket);
```

在接收到一个分组之前该客户机一直空闲；当它接收到一个分组时，将其放到receivePacket中。

```
String modifiedSentence =
   new String(receivePacket.getData());
```

上面一行从receivePacket中提取了数据，并进行类型转换，将字节数组转换成modifiedSentenc字符串。

```
System.out.println("FROM SERVER:" + modifiedSentence);
```

在TCPClient中也存在该行，它将modifiedSentence中的内容打印到客户机的监视器上。

```
clientSocket.close();
```

最后一行关闭套接字。因为UDP是无连接的，这一行将不会引起客户机向服务器发送运输层报文（这与TCPClient不同）。

2. UDPServer.java

现在来看看应用程序的服务器端的代码：

```
import java.io.*;
import java.net.*;
class UDPServer {
   public static void main(String args[]) throws Exception
      {
      DatagramSocket serverSocket = new
                           DatagramSocket(9876);
      byte[] receiveData = new byte[1024];
      byte[] sendData = new byte[1024];
      while(true)
         {
         DatagramPacket receivePacket =
               new DatagramPacket(receiveData,
                        receiveData.length);
         serverSocket.receive(receivePacket);
         String sentence = new String(
                     receivePacket.getData());
         InetAddress IPAddress =
                     receivePacket.getAddress();
         int port = receivePacket.getPort();
         String capitalizedSentence =
                     sentence.toUpperCase();
         sendData = capitalizedSentence.getBytes();
         DatagramPacket sendPacket =
         new DatagramPacket(sendData,
                  sendData.length, IPAddress, port);
         serverSocket.send(sendPacket);
      }
```

```
    }
}
```

UDPServer.java程序构造了一个套接字，如图2-36所示。该套接字被称为serverSocket。它是类型为DatagramSocket的对象，与应用程序的客户机端的套接字相同。这里仍没有与套接字相联系的流。

我们现在来看一下与TCPServer.java不同的代码行。

```
DatagramSocket serverSocket = new DatagramSocket(9876);
```

上面一行在端口9876构造了DatagramSocket serverSocket，所有发送和接收的数据都将通过该套接字。因为UDP是无连接的，因此当接收时，我们没有必要像TCPServer.java那样产生一个新的套接字并继续监听新的请求。如果有多个客户机同时访问服务器，它们都将向这个单一的门serverSocket发送分组。

```
String sentence = new String(receivePacket.getData());
InetAddress IPAddress = receivePacket.getAddress();
int port = receivePacket.getPort();
```

以上三行对来自客户机的分组进行拆分。其中，第一行从分组中提取出数据，并将这些数据放入String sentence中；在UDPClient中也有类似行。第二行提取了IP地址。第三行提取了客户机端口号，该端口号是由客户机选择的，它与服务器端口9876是不同的。（在下一章中我们将更详细地讨论客户机端口号。）对于服务器来说，获得客户机的地址（IP地址和端口号）是有必要的，只有这样才能将转换为大写的行发送回客户机。

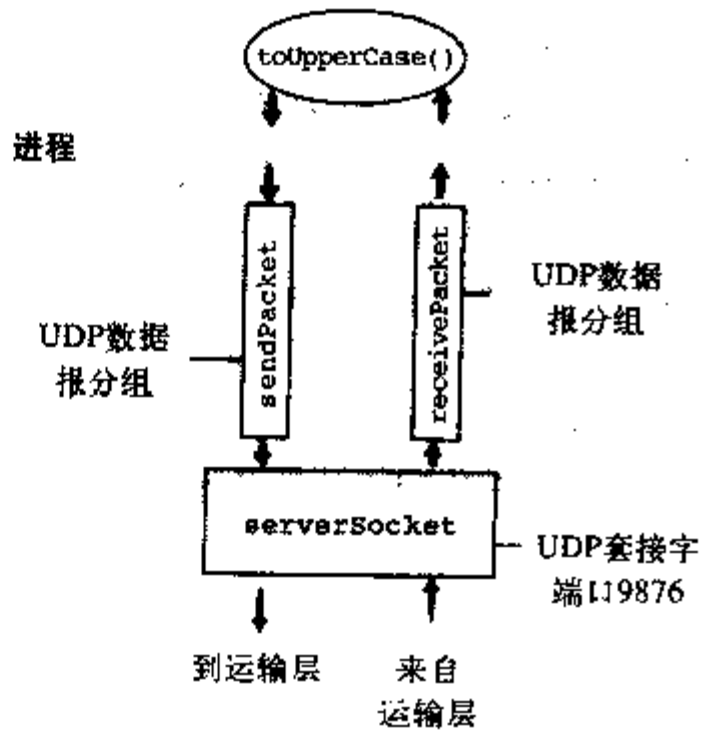


图2-36 UDPServer没有流：套接字接收来自进程的分组并向进程交付分组

至此，我们分析完了两个UDP程序。为了测试该应用程序，你在一台主机上安装并编译UDPClient.java，在另一台主机上安装并编译UDPServer.java。（确信UDPServer.java中使用了适当的服务器主机名。）然后在各自主机上分别运行这两个程序。和TCP不同的是，可以先运行客户机，然后再运行服务器。这是因为当你执行客户机端时，客户机进程并没有试图和服务器发起连接。当客户机和服务器运行起来后，你可以使用该应用程序在客户机上输入一行。

2.9 小结

在本章中，我们从概念和实现两方面学习了网络应用方面的知识。我们学习了被因特网应用普遍采用的客户机/服务器模式，并且知道了该模式在HTTP、FTP、SMTP、POP3和DNS等协议中的使用。我们更为详细地学习了这些重要的应用层协议以及相关应用（Web、文件传输、电子邮件和DNS）。我们也学习了日益流行的P2P体系结构及其在许多应用程序中的应用。我们还探讨了如何使用套接字API编写网络应用程序。我们考察了面向连接的（TCP）和无连接的（UDP）端到端传输服务中的套接字应用。至此，我们在分层的网络体系结构中的

自顶向下之旅，已经完成了“向下”的第一步。

在1.1节中，我们对协议给出一个相当含糊和框架性的定义：“在两个或多个通信实体之间交换的报文格式和次序，以及在报文传输和/或接收或其他事件方面所采取的动作。”本章中的内容，特别是我们对HTTP、FTP、SMTP、POP3和DNS协议进行的细致研究，已经为这个定义加入了相当可观的实质性的内容。协议是网络中的核心概念，我们对应用层协议的学习使我们对“协议是什么”有了更为直观的认识。

在2.1节中，我们描述了TCP和UDP为调用它们的应用所提供的服务模型。当我们在2.7和2.8节中开发运行在TCP和UDP之上的简单应用时，我们对该服务模型进行了更加深入的观察。然而，我们没有介绍TCP和UDP是如何提供这种服务模型的。例如，我们没有介绍TCP是如何提供可靠数据服务的。在下一章中我们将不仅关注运输层协议是什么，而且还关注它如何工作以及为什么要这么做。

有了因特网应用程序结构和应用层协议的知识之后，我们现在准备继续沿该协议栈向下，在第3章中探讨运输层。

课后习题和问题

复习题

2.1节

1. 列出5种非专用的因特网应用及它们所使用的应用层协议。
2. 网络体系结构与应用程序体系结构之间有什么区别？
3. 对两进程之间的通信会话而言，哪个进程是客户机，哪个进程是服务器？
4. 对P2P文件共享应用，你同意“一个通信会话不存在客户机端和服务器端的概念”这种说法吗？为什么？
5. 运行在一台主机上的一个进程使用什么信息来标识运行在另一台主机上的进程？
6. 假定你想尽快地处理从远程客户机到服务器的事务，应使用UDP还是TCP？为什么？
7. 参见图2-4，我们在该图中看到所列出的应用程序没有一种同时既要求“无数据丢失”又要求“定时”。你能设想一种既要求无数据丢失又要求高度时间敏感的应用程序吗？
8. 列出运输协议能够提供的4种宽泛类型的服务。对于每种服务类型，指出是UDP还是TCP（或这两种协议）提供这样的服务。
9. 前面讲过TCP能用SSL来强化，以提供进程到进程安全性服务，包括加密。SSL运行在运输层还是应用层？如果某应用程序研制者想要用SSL来强化UDP，该研制者应当做些什么工作？

2.2～2.5节

10. 握手协议的作用是什么？
11. 为什么HTTP、FTP、SMTP、POP3都运行在TCP而不是UDP之上？
12. 考虑一个电子商务网站需要保留每一个客户的购买记录。描述如何使用cookie来完成该功能。
13. 描述Web缓存器如何减少接收被请求的对象的时延。Web缓存器将减少用户请求的所有对象的时延还是其中的某些对象？为什么？
14. 用Telnet向Web服务器注册并发送一个多行的请求报文。在该请求报文中包含`If-modified-since:`首部行，迫使响应报文中出现`304 Not Modified`状态代码。
15. 为什么说FTP在“带外”发送控制信息？
16. 假定Alice使用一个基于Web的电子邮件账户（如Hotmail或gmail）向Bob发报文，而Bob使用POP3访

问他的邮件服务器来获取自己的邮件。讨论报文是怎样从Alice主机到达Bob主机的。列出在两台主机间移动该报文时所使用的各种应用层协议。

17. 将你最近收到的报文首部打印出来。其中Received:首部行有多少行？分析该报文中的报文首部行中的每一行。
18. 从用户的观点看，POP3协议中“下载并删除”模式和“下载并保留”模式有什么区别？
19. 一个机构的Web服务器和邮件服务器可以有完全相同的主机名别名（如foo.com）吗？包含邮件服务器主机名的RR有什么样的类型？

2.6节

20. 在BitTorrent中，假定Alice以30s间隔向Bob发送文件块。Bob将必须回应，以相同的间隔向Alice发送文件块吗？为什么？
21. 考虑一个新对等方Alice加入BitTorrent，但她没有任何文件块。由于没有任何文件块，没有什么可上载所以她不能成为任何其他对等方的前4位上载者。那么，Alice将怎样得到她的第一个文件块呢？
22. 什么是覆盖网络？它包括路由器吗？在覆盖网络中什么是边？查询洪泛覆盖网络是怎样创建和维护的？
23. 具有集中式索引的即时讯息以何种方式采用客户机/服务器和P2P体系结构的混合结构？
24. 今天大多数即时讯息系统使用集中式索引来定位用户。考虑使用查询洪泛覆盖网络（如Gnutella）来定位用户。描述如何实现这种设计，并讨论其优点与缺点。
25. Skype针对两个重要功能使用了P2P技术。它们是什么？
26. 至少列出4个不同的应用，它们本质上适合采用P2P体系结构。（提示：文件分发和即时讯息就是两个。）

2.7～2.8节

27. 2.8节所描述的UDP服务器仅需要一个套接字，而2.7节所描述的TCP服务器需要两个套接字。为什么？如果TCP服务器支持n个并行连接，每个连接来自不同的客户机主机，TCP服务器将需要多少个套接字？
28. 对于2.7节所描述的运行在TCP之上的客户机/服务器应用程序，服务器程序为什么必须先于客户机程序运行？对于2.8节所描述的运行在UDP之上的客户机/服务器应用程序，客户机程序为什么可以先于服务器程序运行？

习题

1. 是非判断题。
   a. 假设用户请求由某些文本和两幅图片组成的Web页面。对于这个页面，客户机将发送一个请求报文并接收三个响应报文。
   b. 两个不同的Web页面（例如，www.mit.edu/research.html及www.mit.edu/students.html）可以通过同一个持久连接发送。
   c. 在浏览器和初始服务器之间使用非持久连接的话，一个TCP报文段可能携带两个不同的HTTP服务请求报文。
   d. HTTP响应报文中的Date:首部指出了该报文中的对象最后一次修改的时间。
2. 阅读有关FTP的RFC 959，列出这个RFC中所支持的所有FTP客户机命令。
3. 考虑一个HTTP客户机要以给定的URL获取一个Web页面。开始时并不知道该HTTP服务器的IP地址，在这种情况下，除了HTTP外，还需要什么运输层和应用层协议？
4. 考虑当浏览器发送一个HTTP GET报文（即HTTP GET报文的实际内容）时通过Ethereal捕获到下列ASCII字符串。字符<cr><lf>是回车换行符（即文本中的斜体字符串<cr>表示单个回车符，该回车

符包含在HTTP首部中)。回答下列问题，指出你在下面HTTP GET报文中找到答案的地方。

```
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai
a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (
Windows;U; Windows NT 5.1; en-US; rv:1.7.2) Gec
ko/20040804 Netscape/7.2 (ax) <cr><lf>Accept:ex
t/xml,application/xml,application/xhtml+xml,text
/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
<cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept-
Encoding: zip,deflate<cr><lf>Accept-Charset: ISO
-8859-1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive: 300<cr>
<lf>Connection:keep-alive<cr><lf><cr><lf>
```

a. 浏览器请求的文档的URL是什么?

b. 该浏览器运行的是何种版本的HTTP?

c. 该浏览器请求的是一条非持久连接还是持久连接?

d. 运行该浏览器的主机的IP地址是什么?

5. 下面文本中显示的是来自服务器的回答，以响应上述问题中的HTTP GET报文。回答下列问题，指出你在下面报文中找到答案的地方。

```
HTTP/1.1 200 OK<cr><lf>Date: Tue, 07 Mar 2006
12:39:45GMT<cr><lf>Server: Apache/2.0.52 (Fedora)
<cr><lf>Last-Modified: Sat, 10 Dec2005 18:27:46
GMT<cr><lf>ETag: "526c3-f22-a88a4c80"<cr><lf>Accept-
Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf>
Keep-Alive: timeout=max=100<cr><lf>Connection:
Keep-Alive<cr><lf>Content-Type: text/html; charset=
ISO-8859-1<cr><lf><cr><lf><!doctype html public "-
//w3c//dtd html 4.0 transitional//en"><lf><html><lf>
<head><lf>  <meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1"><lf> <meta
name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT
5.0; U) Netscape]"><lf>  <title>CMPSCI 453 / 591 /
NTU-ST550A Spring 2005 homepage</title><lf></head><lf>
<much more document text following here (not shown)>
```

a. 服务器能够成功地找到那个文档吗? 该文档提供的回答是什么时间?

b. 该文档最后一次修改是什么时间?

c. 已被返回的文档有多少字节?

d. 被返回的文档的前5个字节是什么? 该服务器认可这是一条持久连接吗?

6. 获得HTTP/1.1规范 (RFC 2616)。回答下面的问题:

a. 解释在客户机和服务器之间用于指示持久连接被关闭的信令机制。客户机、服务器或两者都能通知连接关闭了吗?

b. HTTP提供了什么加密服务?

7. 假定你在浏览器中点击一个超链接获得Web页面。假定相关的URL的IP地址没有缓存在本地主机上，因此必须进行DNS查询从而获得IP地址。如果主机从DNS得到IP地址之前，已经访问了n个DNS服务器，相继产生的RTT依次为RTT_1，…，RTT_n。进一步假定与链路相关的Web页面只包含一个对象，即少量的HTML文本。令RTT_0表示本地主机和包含对象的服务器之间的RTT值。假定该对象传输时间为0，则从客户机点击该超链接到它接收到该对象需要多长时间?

8. 参照习题7，假定在同一服务器上某HTML文件引用了三个非常小的对象。忽略发送时间，在下列情况下需要多长时间?

a. 没有并行TCP连接的非持久HTTP。

b. 有并行连接的非持久HTTP。

c. 有流水线的持久HTTP。

9. 考虑图2-12，其中有一个机构的网络和因特网相连。假定对象的平均长度为900kb，从这个机构网的浏览器到初始服务器的平均请求率是每秒15个请求。还假定从访问链路的因特网一侧的路由器转发一个HTTP请求开始，到接收到其响应的平均时间是2s（参见2.2.5节）。将总的平均响应时间建模为平均访问时延（即从因特网路由器到机构路由器的时延）和平均因特网时延之和。对于平均访问时延，使用$\Delta/(1-\Delta\beta)$，式中Δ是跨越访问链路发送一个对象所需的平均时间，β是对象对该访问链路的平均到达率。

a. 求出总的响应时间。

b. 现在假定在这个机构的局域网中安装了缓存器。假定命中率为0.4，求出总的响应时间。

10. 考虑一条10米短链路，某发送方可以通过它以150 bps速率双向传输。假定包含数据的分组是100 kb长，仅包含控制（如ACK或握手）的分组是200 b长。假定N个并行连接的每个都获得$1/N$的链路带宽。现在考虑HTTP协议，并且假定每个下载对象是100 Kb长，这些初始下载对象包含10个来自相同发送方的引用对象。在这种情况下，经非持久HTTP的并行实例的并行下载有意义吗？现在考虑持久HTTP，你期待这比非持久情况有很大增益吗？评价并解释你的答案。

11. 写一个简单的TCP程序，使服务器接收来自客户机端的输入行并将其打印在标准输出上（可以通过修改书中的`TCPServer.java`程序实现上述任务）。编译并执行你的程序。在另一台有浏览器的机器上，设置浏览器的代理服务器为你的服务器程序正在运行的机器，同时配置适当的端口号。这时你的浏览器向服务器发送GET请求报文，服务器应当在其标准输出上显示该报文。使用这个平台，确定你的浏览器是否对本地缓存的对象产生了条件GET报文。

12. SMTP中的`MAIL FROM:`与邮件消息本身中的`From:`之间有什么区别？

13. 阅读POP3 的RFC，即RFC 1939。UIDL POP3命令的目的是什么？

14. 考虑用POP3接入你的电子邮件。

a. 假定你已经配置了你的POP邮件客户机以在“下载并删除”模式下运行。完成下列事务：

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: blah blah ...
S: ..........blah
S: .
?
?
```

b. 假定你已经配置了你的POP邮件客户机，以运行在“下载并保持”模式下。完成下列事务：

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: blah blah ...
S: ..........blah
S: .
?
?
```

c. 假定你已经配置了你的POP邮件客户机，以运行在“下载并保持”模式下。使用（b）部分中的记录，假定你检索报文1和2，退出POP，5分钟以后，你再访问POP以检索新电子邮件。假定在这5分钟间隔内，没有新报文发送给你。给出第二种POP会话的记录。

15. a. 什么是whois数据库？

b. 使用因特网上的各种whois数据库，获得两台DNS服务器的名字。指明你使用的是哪个whois数据库。

c. 使用你本地机器上的nslookup向3台DNS服务器发送DNS查询：本地DNS服务器和两台你在(b)部分发现的DNS服务器。尝试对类型A、NS和MX报告的查询，总结你的发现。

d. 使用nslookup找出一台具有多个IP地址的Web服务器。你所在的机构（学校或公司）的Web服务器具有多个IP地址吗？

e. 使用ARIN whois数据库，确定你所在大学使用的IP地址范围。

f. 描述一个攻击者在发动攻击前，如何利用whois数据库和nslookup工具来执行对一个机构的侦察。

g. 讨论为什么whois数据库应当为公众所用。

16. 考虑向N个对等方分发一个F=10 Gb的文件。该服务器的上载速率为u_s=20 Mbps，每个对等方的下载速率为d_i= 1 Mbps、上载速率为u。对于N = 10 100和1000并且u = 200 kbps, 600 kbps和1 Mbps的每种组合，绘制出确定最小分发时间的示意图，需要针对客户机/服务器分发和P2P分发两种情况来进行。

17. 考虑采用客户机/服务器体系结构向N个对等方分发一个文件。假定使用的是流体模型，即服务器能够同时向多个对等方传输，只要组合速率不超过u_s，就以不同的速率向每个对等方传输。

a. 假定$u_s/N \leqslant d_{\min}$。定义一个具有NF/u_s分发时间的分发方案。

b. 假定$u_s/N \geqslant d_{\min}$。定义一个具有$F/d_{\min}$分发时间的分发方案。

c. 得出最小分发时间通常由$\max\{NF/u_s, F/d_{\min}\}$所确定的结论。

18. 考虑采用P2P体系结构向N个用户分发一个F比特的文件。假定使用的是流体模型。为了简化起见，假定$d_{\min}$很大，因此对等方下载带宽不会成为瓶颈。

a. 假定$u_s \leqslant (u_s+u_1+\cdots+u_N)/N$。定义一个具有$F/u_s$分发时间的分发方案。

b. 假定$u_s \geqslant (u_s+u_1+\cdots+u_N)/N$。定义一个具有$NF/(u_s+u_1+\cdots+u_N)$分发时间的分发方案。

c. 得出最小分发时间通常由$\max\{F/u_s, NF/(u_s+u_1+\cdots+u_N)\}$所确定的结论。

19. 假定覆盖网络中有N个活跃的对等方，每对对等方有一个活跃的TCP连接。此外，假定该TCP连接通过总共M个路由器。在对应的覆盖网络中，有多少节点和边？

20. 在2.6节中讨论使用查询洪泛的覆盖网络时，我们较为详细地描述了一个新的对等方是如何加入覆盖网络的。在这个问题中，我们要探讨当一个对等方离开覆盖网络时发生的情况。假定每个参与对等方维护TCP连接，同时至多有4个不同的对等方。假定对等方X与其他对等方有5个TCP连接，它要离开。

a. 先考虑正常离开的情况，即对等方X明确地关闭了它的应用，因此正常地关闭了它的5个TCP连接。这5个先前连接的对等方中的每个将采取哪些动作？

b. 现在假定X突然从因特网断开连接，而没有通知它的5个邻居关闭其TCP连接。这会发生什么情况？

21. 在这个问题中，我们探讨查询洪泛中“查询命中”报文的反向路径选路。假定Alice发出一个“查询”报文。进一步假定Bob接收到该“查询”报文（这可能由几个中间对等方转发），并有一个文件匹配该查询。

a. 回想当一个对等方有一个匹配文件时，它沿着对应的“查询”报文的反向路径发送一个“查询命中”报文。另一种设计是Bob与Alice创建一条直接的TCP连接，经该连接发送“查询命中”报文。这种设计方法有什么优点和缺点？

b. 当对等方Alice产生一条“查询”报文时，它在该报文的MessageID字段中插入一个独特的ID。当对等方Bob有匹配时，它产生“查询命中”报文，该报文使用与查询报文相同的MessageID。描述对等方如何使用MessageID字段和本地选路表来完成反向路径选路。

c. 另一种不使用报文标识符的方法如下：当一条查询报文到达某对等方时，在转发该报文之前，该对等方用它的IP地址扩充该查询报文。描述对等方如何使用这种机制来完成反向路径选路。

22. 在本题中我们探讨设计一种层次覆盖网络，它具有普通对等方、超级对等方和超超级对等方。

a. 假设每个超超级对等方大约负责200个超级对等方，每个超级对等方大约负责200个普通对等方。对于一个有400万对等方的网络，有必要设多少个超超级对等方？

b. 每个超级对等方可能存储什么信息？每个超超级对等方存储什么信息？在这样的3层设计中，怎样执行搜索？

23. 考虑P2P文件共享中的查询洪泛（如2.6节所讨论的）。假定每个对等方在覆盖网络中与至多N个邻居相连，同时假定节点计数字段初始被置为K，再假定Alice进行查询。求出被发送进覆盖网络的查询报文数量的上界。

24. 在一台主机上安装编译用Java写的TCPClient和UDPClient程序，在另一台主机上安装编译TCPServer和UDPServer程序。

a. 如果你在运行TCPServer之前运行TCPClient，将发生什么现象？为什么？

b. 如果你在运行UDPServer之前运行UDPClient，将发生什么现象？为什么？

c. 如果你对客户机端和服务器端使用了不同的端口，将发生什么现象？

25. 假定在UDPClient.java中，我们使用

```
DatagramSocket clientSocket = new DatagramSocket (5432) ;
```

代替

```
DatagramSocket clientSocket = new DatagramSocket () ;
```

在UDPServer.java中是否有必要进行修改？UDPClient和UDPServer中的套接字端口号是多少？在变化之前它们是多少？

讨论题

1. 你认为P2P文件共享应用为什么这样流行？是因为它们（有争议地非法）发布免费的音乐和视频吗？是因为它们的大量服务器有效地响应对几百万字节的大量要求吗？还是因为所有这些原因？
2. 阅读由Biddle、England、Peinado和Willman [Biddle 2003]撰写的论文The Darknet and the Future of Content Distribution（内容分发的黑网和未来）。你同意作者们的所有观点吗？为什么？
3. 电子商务网站和其他Web站点通常有后端数据库。HTTP服务器怎样和这些后端服务器进行通信？
4. 你怎样配置浏览器的本地缓存？你有哪些缓存设置选项？
5. 你可以配置浏览器，使其可以同时打开到一个Web站点的多个连接吗？具有大量并行的TCP连接有什么优点和缺点？
6. 我们已经发现TCP套接字将发送的数据看成是字节流，而UDP套接字识别报文边界。面向字节的API与明确识别和保留应用程序定义的报文格式的API相比，有什么优缺点？
7. Apache Web服务器是什么？它的价格是多少？目前它具有哪些功能？
8. 假定Web标准化组织决定改变命名规则，使得每个对象都有一个独一无二的名字，且与位置无关（即所谓URN）。讨论由这种变化所引发的相关问题。
9. 目前有公司经因特网分发实况电视吗？如果有的话，这些公司使用的是客户机/服务器体系结构还是P2P体系结构？
10. 目前有公司经因特网使用P2P体系结构提供按需视频服务吗？
11. Skype如何向许多不同的国家提供PC到电话的服务？
12. 目前最为流行的BitTorrent客户机是什么？

套接字编程作业

作业1：多线程Web服务器

当本编程作业结束时，你将用Java语言开发了一个多线程的Web服务器，它能并行服务于多个请求。你将实现由RFC 1945定义的HTTP 1.0版。

HTTP/1.0为每一次请求/响应对创建一个单独的TCP连接。一个单独的线程将处理这些连接中的每一个。存在一个主线程，服务器使用该主线程监听希望创建连接的客户机。为了简化编程任务，我们分两阶段开发代码。在第一个阶段，你将写一个多线程服务器，用于简单地显示接收到的HTTP请求报文的内容。在这个程序正常运行后，再添加产生适当响应所需的代码。

当开发这个程序时，可以使用Web浏览器来测试该服务器。但是记住不能通过标准的80端口提供服务，因此你需要在浏览器的URL中指定端口号。例如，如果主机名为host.someschool.edu，服务器监听端口为6789，并且需要获取的文件名为index.html，应在浏览器中指定下列URL：

```
http://host.someschool.edu:6789/index.html
```

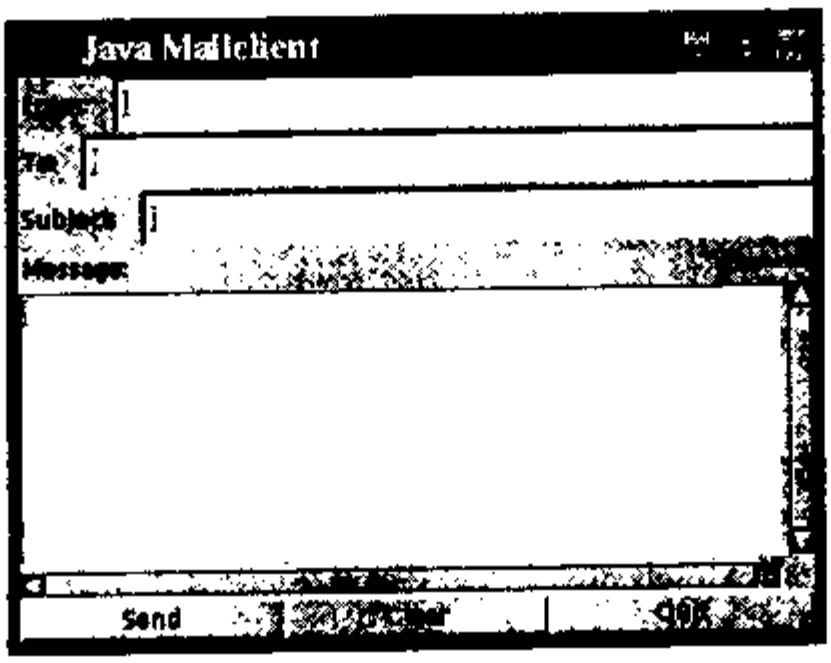
当服务器遇到差错时，它应该发送一个具有适当HTML源的响应报文，从而可以在浏览器窗口中显示差错信息。在Web站点http://www.awl.com/kurose-ross中可以找到本作业的所有细节及重要的Java代码片段。

作业2：邮件客户机

在这个作业中，你将用Java开发一个具有如下特点的用户邮件代理：

- 为发送者提供一个图形界面，其中具有用于本地邮件服务器的以下字段：发送者电子邮件地址、接收者电子邮件地址、报文主题及报文本身。
- 在邮件客户机和本地邮件服务器之间创建一个TCP连接。向本地邮件服务器发送SMTP命令。从本地邮件服务器接收和处理SMTP命令。

你开发的界面看起来如下图所示：



你将开发用户代理，使它每次至多给一个接收者发送电子邮件。并且，该用户代理将假定接收者电子邮件地址的字段部分有SMTP服务器的规范名字。（该用户代理不对MX记录执行DNS查找，因此该发送者必须提供实际的邮件服务器的名字。）在Web站点http://www.awl.com/kurose-ross中可以找到本作业的所有细节及重要的Java代码片段。

作业3：UDP Pinger实验

在本实验中，你将实现一个简单的基于UDP的Ping客户机和服务器。由这些程序提供的功能类似于

现代操作系统中使用的标准Ping程序。标准的Ping工作时发送因特网控制报文协议（ICMP）ECHO报文，远程机器对发送方返回响应。该发送方则能够决定它自己和被探测的计算机之间的往返时延。

Java不提供发送和接收ICMP报文的任何功能，这是在本实验中你要在应用层用标准的UDP套接字和报文实现Ping的原因。在Web站点http://www.awl.com/kurose-ross中可以找到本作业的所有细节及重要的Java代码片段。

作业4：Web代理服务器

在本实验中，你将研发一个简单的Web代理服务器，该服务器也能够缓存Web页面。该服务器将接收来自浏览器的GET报文，向目的地Web服务器转发GET报文，从目的地服务器接收HTTP响应报文，向浏览器转发HTTP响应报文。这是一个非常简单的代理服务器，它仅能理解简单的GET请求。然而，该服务器能够处理各种对象，并不仅仅是HTML页面，而且也包括图像。在Web站点http://www.awl.com/kurose-ross中可以找到本作业的所有细节及重要的Java代码片段。

Ethereal实验

Ethereal实验：HTTP

在实验1中，我们已经初步使用了Ethereal分组嗅探器，我们现在准备使用Ethereal研究运行中的协议。在本实验中，我们将研究HTTP协议的几个方面：基本的GET/回答交互，HTTP报文格式，检索大HTML文件，检索嵌有URL的 HTML文件，持久和非持久连接，HTTP鉴别和安全性。

如同所有的Ethereal实验一样，对该实验的全面描述可查阅本书的Web站点http://www.awl.com/kurose-ross。

Ethereal实验：DNS

在本实验中，我们仔细观察DNS的客户机端（DNS是用于将因特网主机名转换为IP地址的协议）。2.5节讲过，DNS中的客户机角色是相当简单的：客户机向它的本地DNS服务器发送一个请求，并接收返回的响应。在此过程中发生的很多事情均不为DNS客户机所见，如层次DNS服务器互相通信，以递归地或迭代地解析该客户机的DNS请求。然而，从DNS客户机的角度来看，该协议是相当简单的，即向本地DNS服务器发送一个请求，从该服务器接收一个响应。在本实验中我们观察DNS。

在本书的Web站点http://www.awl.com/kurose-ross可以找到本实验的完整描述。

人物专访

Bram Cohen

Bram Cohen是BitTorrent公司的首席执行官和奠基人之一，也是BitTorrent对等（P2P）文件分发协议的创建者。Bram也是CodeCon的奠基人之一、Codeville的作者之一。在创建BitTorrent之前，Bram在MojoNation公司工作。该公司允许人们将机密的文件分割为多个加密块，并将这些块分发到其他运行MojoNation软件的计算机上。这个概念激发了Bram研制BitTorrent的灵感。在加入MojoNation公司之前，Bram是一名电子商务精英，20世纪90年代中后期曾在几个因特网公司工作。Bram生长于纽约城，毕业于Stuyvesant高中，并在布法罗大学上学。

• 你是如何想到研制BitTorrent的？

我那时在网络（TCP/UDP之上的协议）方面具有丰富的工作经验，并且正在实现云集（swarming），云集在当时看起来是最令人感兴趣的未解问题，因此我决定攻克它。

• 研制BitTorrent时最富有挑战性的方面是什么？

最为基础的部分是使整体设计和协议完整、结构健全。这些方面就绪后，充实工作就是“简单的编程”了。就编程实现而言，至今最困难的部分是实现一种可靠的系统。当处理不可信的对等方时，你必须假设他们中的任何人在任何时间能做任何事，同时建立起应对所有极端事件的措施。在最初创建BitTorrent时，随着新问题的出现以及总体设计的逐渐清晰，我不得不重写大部分代码。

• 人们最初是如何开始使用BitTorrent的？

人们发现BitTorrent可用作下载工具。有一些他们想要的内容使用BitTorrent是最为合适的，因此他们就下载下来使用了。发行人决定使用BitTorrent，只是因为他们并没有宽带来以任何其他方式分发他们的内容。

• RIAA（美国唱片产业联合会）和MPAA（美国电影联合会）提起诉讼，反对人们使用BitTorrent等文件共享工具分发电影和音乐，请谈谈你的想法。你是否因研制能非法分发具有版权的资料的技术而被起诉过？

违反版权是非法的，而技术并不违法。我从来没有被起诉过，因为我从没有参与过任何违反版权的活动。

• 您认为在不久的将来，其他文件分发系统将取代BitTorrent吗？例如，Microsoft公司也许会在将要发布的操作系统中包括它自己的专用文件分发协议。

未来也许有其他通用协议，但云集数据的基本原则（如BitTorrent协议中阐述的那样）是不可能改变的。除非是某些基本量之间的比率随着速率增加而迅速改变，从而导致因特网的总体结构发生了根本的变化。但是，对未来几年的规划只能是加强当前的模型。

• 更一般而言，您认为因特网的发展方向在何方？您认为最重要的技术挑战是什么？您能设想将会出现何种新型的“招人喜爱的应用程序”吗？

因特网（更一般地说是计算机）越来越无处不在了。随着价格的下降，iPod nano也许会越来越受到人们的喜爱，当前最令人兴奋的技术挑战是从所有连接的设备中收集尽可能多的数据，并且以一种可访问和有用的形式使用这些数据。例如，几乎每个可携带的设备都包含一个GPS，利用GPS，当你丢失了自己的物件（包括衣服、玩具、器具和家具等）时，可以知道它们在哪里，并了解其变化情况（包括必要的维护、期待的未来效用、滥用的检测等）。你不仅能够得到有关你自己物品的信息，而且能够相当精确地收集一个特定产品的通用生命周期等。另外，人们之间的协作也变得更容易了，当两个人都有移动电话时，极大地方便了人们之间的沟通。

• 最近您成立了一家公司，该公司将如何使用BitTorrent来获得收益？

我们打算帮助发行人将其内容上线并使之货币化，同时也向其他公司提供数据分发服务。

• 有什么激发了您的专业灵感吗？以什么样的方式？

没有想起特别的格言，但是硅谷后起之秀的神话是我一直以来追随的目标。

• 对于即将进入网络/因特网领域工作的学生们，您有什么忠告吗？

找出目前并不热门但你认为激动人心的东西，从你最感兴趣的部分开始做起。当然，需要在你希望工作的领域努力获得专业经验。实际经验会教给你现实世界中什么是重要的，但是从纯学术的观点来看，某些东西总是不现实的。

第3章 运 输 层

运输层位于应用层和网络层之间，是分层的网络体系结构的重要部分。该层为运行在不同主机上的应用进程提供直接的通信服务起着至关重要的作用。我们在本章采用的教学方法是，交替地讨论运输层的原理和这些原理在现存的协议中是如何实现的；与往常一样，我们将关注因特网协议，特别是TCP和UDP运输层协议。

首先，我们将讨论运输层和网络层的关系。这就进入了研究运输层第一个关键功能的阶段，即将网络层的在两个端系统之间的交付服务，扩展到运行在两个不同端系统上的应用层进程之间的交付服务。我们将在讨论因特网的无连接运输协议UDP时，举例说明该功能。

其次，我们重新回到原理学习上，面对计算机网络中最基本的问题之一，即两个实体怎样才能在一种会丢失或损坏数据的介质上可靠地通信。通过一系列复杂性不断增加的情况（即更真实的情况），我们将逐步建立起一系列被运输协议用来解决这些问题的技术。然后，我们将说明这些原理是如何体现在因特网面向连接的运输协议TCP中的。

第三，我们讨论网络中的第二个最基本的重要问题，即控制运输层实体的传输速率以避免网络中的拥塞，或从拥塞中恢复过来。我们将考察拥塞的原因和后果，以及常用的拥塞控制技术。在透彻地理解了拥塞控制问题之后，我们将研究TCP的拥塞控制方法。

3.1 概述和运输层服务

在前两章中，我们已对运输层的作用及其所提供的服务有所了解。现在我们快速地回顾一下前面介绍过的有关运输层的知识。

运输层协议为运行在不同主机上的应用进程之间提供了**逻辑通信**（logic communication）功能。从应用程序的角度看，通过逻辑通信，运行不同进程的主机好像直接相连一样；实际上，这些主机也许位于地球的两侧，通过很多路由器及多种不同类型的链路相连。应用进程使用运输层提供的逻辑通信功能彼此发送报文，而无需考虑承载这些报文的物理基础设施的细节。图3-1图示了逻辑通信的概念。

如图3-1所示，运输层协议是在端系统中而不是在网络路由器中实现的。在发送方，运输层将接收到的来自发送应用进程的报文转换成运输层分组，用因特网术语称其为运输层**报文段**（segment）。可能的方法是，将应用报文划分为较小的块，并为每块加上一个运输层首部来创建运输层报文段。然后，在发送方端系统中，运输层将这些报文段传递给网络层，网路层将其封装成网络层分组（一个数据报）并向目的地发送。注意到下列事实是重要的：网络路由器仅作用于该数据报的网络层字段，即它们不检查封装在该数据报的运输层报文段的字段。在接收方，网络层从数据报中提取运输层报文段，并将该报文段向上交给运输层。运输层则处理接收到的报文段，使得接收方应用进程可应用该报文段中的数据。

网络应用程序可以使用多种运输层协议。例如，因特网有两种协议，即TCP和UDP，这两种协议都能为其调用的应用程序提供一组不同的运输层服务。

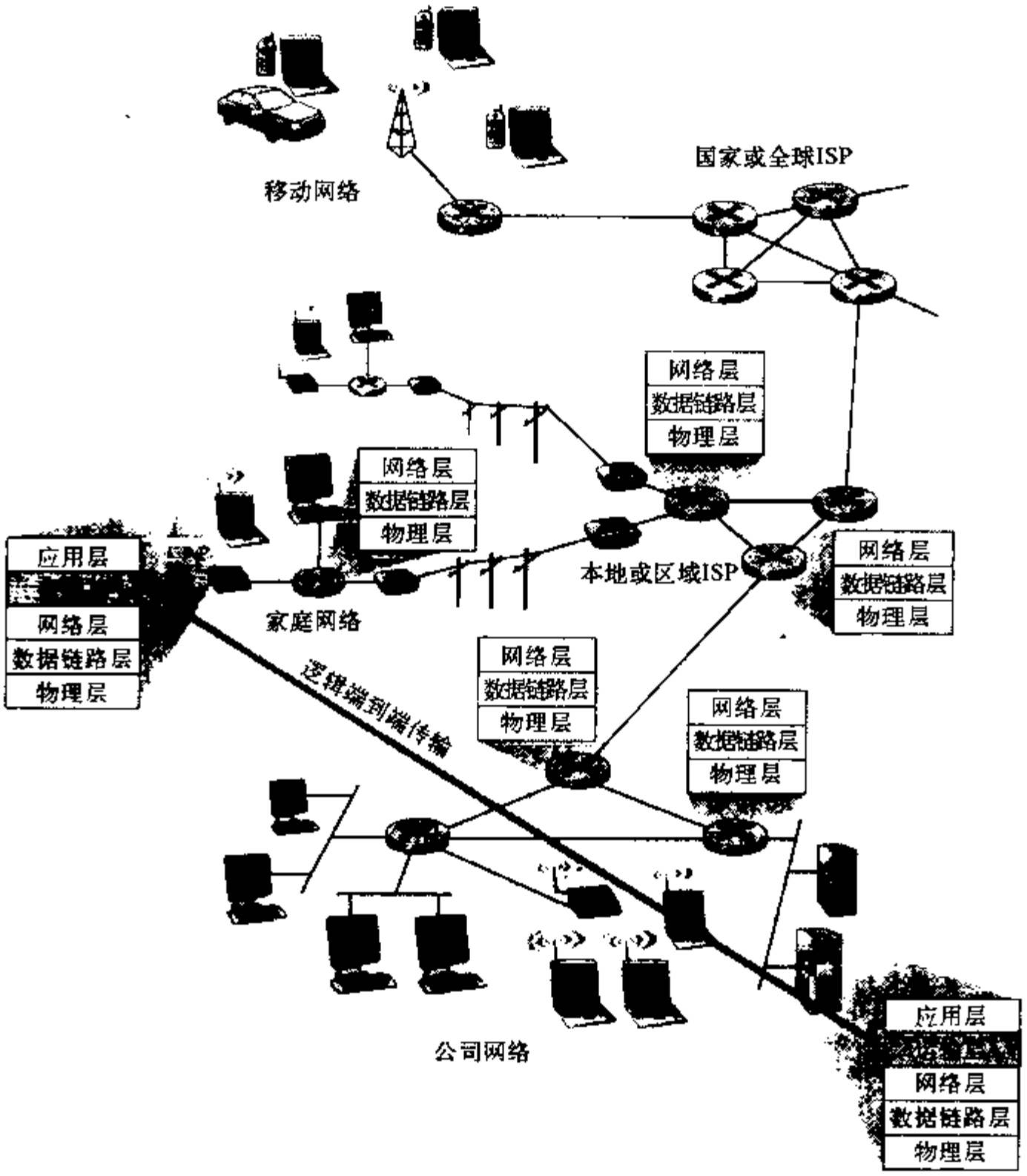


图3-1 运输层在应用程序进程间提供逻辑的而非物理的通信

3.1.1 运输层和网络层的关系

前面讲过，在协议栈中，运输层刚好位于网络层之上。运输层为运行在不同主机上的进程之间提供了逻辑通信，而网络层则提供了主机之间的逻辑通信。这种差别虽然细微，但很重要。下面我们用一个家庭打比方，来分析这种差别。

考虑有两个家庭，一家位于美国东海岸，一家位于美国西海岸，每家有12个孩子。东海岸家庭的孩子们是西海岸家庭孩子们的堂兄弟姐妹。这两个家庭的孩子们喜欢彼此通信，每个人每周要给每个堂兄弟姐妹写一封信，每封信都用单独的信封通过传统的邮政服务发送。因此，每个家庭每周向另一家庭发送144封信。（如果他们有电子邮件的话，这些孩子可以省不少钱！）每一个家庭有一个孩子负责收发邮件，西海岸家庭是Ann而东海岸家庭是Bill。每周Ann去她的所有兄弟姐妹那里收集信件，并将这些信件交到每天到家门口来的邮政运输车上。当信件到达西海岸家庭时，Ann也负责将信件分发到她的兄弟姐妹手上。东海岸家庭中的

Bill也负责类似的工作。

在这个例子中，邮政服务为两个家庭间提供逻辑通信，邮政服务将信件从一家送往另一家，而不是从一个人送往另一个人。另一方面，Ann和Bill为堂兄弟姐妹之间提供了逻辑通信，Ann和Bill从兄弟姐妹那里收取或到兄弟姐妹那里交付信件。从堂兄弟姐妹们的角度来看，Ann和Bill就是邮件服务，尽管他们只是端到端交付过程的一部分（即端系统部分）。在解释运输层和网络层之间的关系时，这个家庭的例子是很形象的。

应用层报文 = 信封上的字

进程 = 堂兄弟姐妹

主机（又称为端系统） = 家庭

运输层协议 = Ann和Bill

网络层协议 = 邮政服务（包括邮政运输车）

我们继续观察这个类比。注意到Ann和Bill都是在各自家里进行工作，例如，他们并没有参与任何一个中间邮件中心对邮件进行的分拣，或者将邮件从一个邮件中心送到另一个邮件中心之类的工作。类似地，运输层协议只工作在端系统。在端系统中，运输层协议将来自应用进程的报文移动到网络边缘（即网络层），反之亦然，但是对有关这些报文在网络核心如何移动并不作任何规定。事实上，如图3-1所示，中间路由器既不处理也不识别运输层加在应用层报文的任何信息。

我们继续讨论这两家的情况。现在假定Ann和Bill外出度假，另外一对堂兄妹Susan和Harvey接替他们的工作，即在家庭内部进行信件的收集和交付工作。不幸的是，Susan和Harvey的收集和交付工作与Ann和Bill所做的不完全一样。由于年龄小，Susan和Harvey收发邮件的次数更少，而且偶尔还会丢失邮件（假设是被家里的狗咬坏了）。因此，堂兄妹Susan和Harvey并没有提供与Ann和Bill一样的服务集合（即相同的服务模型）。以类似方式，计算机网络中可以安排多种运输层协议，每种协议为应用程序提供不同的服务模型。

Ann和Bill能够提供的可能服务显然受到了邮政服务能够提供的可能服务的限制。例如，如果邮政服务不能提供在两家之间传递邮件所需时间的最长期限（比如3天），那么Ann和Bill就不可能保证邮件在堂兄弟姐妹之间传递信件的最长期限。类似地，运输层协议所能提供的服务也受到了底层网络层协议的服务模型的限制。如果网络层协议不能为两主机之间发送的运输层报文段提供时延和带宽保证，那么运输层协议也不能为两进程之间发送的报文提供时延和带宽保证。

然而，即使底层网络协议在网络层不提供相应服务，运输层协议也能提供某些服务。例如，如本章后面所要介绍的，即使底层网络协议是不可靠的，即网络层协议会使分组丢失、混乱和重复，运输层也能为应用程序提供可靠的传输服务。另一个例子是（我们在第8章中讨论网络安全时将会深入研究），即使网络层不能保证运输层报文段的机密性，运输层也能使用加密来确保应用层报文不被入侵者读取。

3.1.2 因特网运输层概述

前面讲过，更一般而言因特网是一个TCP/IP网络，为应用层提供了两种截然不同的运输层协议。其中，一种是UDP（用户数据报协议），它为调用它的应用程序提供了一种不可靠的、无连接的服务。另一种是TCP（传输控制协议），它为调用它的应用程序提供了一种可靠的、面向连接的服务。当设计一个网络应用程序时，该应用程序的开发人员必须指定使用这两种

运输层协议中的哪一种。如2.7节和2.8节所述，开发人员在创建套接字时必须指定是选择UDP还是选择TCP。

为了简化术语，在与因特网有关的环境中，我们将运输层报文段称为报文段（segment）。然而，需要指出的是，因特网文献（如RFC文档）也将TCP的运输层分组称为报文段，而将UDP的运输层分组称为数据报。而同样是这些因特网文献也将网络层分组称为数据报！本书作为计算机网络的入门书籍，我们认为将TCP和UDP的分组统称为报文段，而将数据报术语保留给网络层分组比较不容易混淆。

在对UDP和TCP进行简要介绍之前，先简短介绍一下因特网的网络层（第4章将详细讨论之）。因特网网络层协议有一个名字叫IP，全称是网际协议。IP为主机之间提供了逻辑通信。IP的服务模型是**尽力而为交付服务**（best-effort delivery service）。这意味着IP尽它最大的努力在通信的主机之间交付报文段，但它并不做任何确保。特别是，它不确保报文段的交付，不保证报文段的按序交付，更不保证报文段中数据的完整性。由于这些原因，IP被称为**不可靠服务**（unreliable service）。在此还要指出的是，每台主机至少有一个网络层地址，即所谓的IP地址。我们在第4章将详细讨论IP地址；在这一章中，我们只需要知道每台主机有一个IP地址。

在对IP服务模型有了初步了解后，我们总结一下UDP和TCP所提供的服务模型。UDP和TCP最基本的任务是，将两个端系统间IP的交付服务扩展为运行在两个端系统上的进程之间的交付服务。将主机间交付扩展到进程间交付，称为**运输层的多路复用**（transport-layer multiplexing）与**多路分解**（demultiplexing）。我们将在下一节讨论运输层的多路复用与多路分解。UDP和TCP还可以通过在其报文段的首部中添加差错检测字段而提供完整性检查。进程间数据交付和差错检测是两种最低限度的运输层服务，也是UDP所能提供的仅有的两种服务。特别是，和IP一样，UDP也是一种不可靠的服务，即不能保证一个进程所发送的数据能够完整无损地到达目的进程。在3.3节中将更详细地讨论UDP。

另一方面，TCP为应用程序提供了几种附加服务。首先，它提供**可靠数据传输**（reliable data transfer）。通过使用流量控制、序号、确认和定时器等技术（本章将详细介绍这些技术），TCP确保正确地、按序地将数据从发送进程交付给接收进程。这样，TCP就将两个端系统间不可靠的IP服务转换成了一种可靠的进程间数据传输服务。TCP还提供**拥塞控制**（congestion control）。总体来看，拥塞控制与其说是一种提供给调用它的应用程序的服务，不如说是一种提供给整个因特网的服务，这是一种带来广泛好处的服务。不太严格地说，TCP拥塞控制防止任何一条TCP连接用过多流量来淹没通信主机之间的链路和交换设备。从原理上讲，TCP允许TCP连接通过一条拥塞的网络链路，平等地共享网络链路带宽。这可以通过调节发送方TCP发送到网络的流量的速率来实现。另一方面，UDP流量是不可调节的。使用UDP传输的应用程序可以根据其需要以任何速率发送数据。

提供可靠数据传输和拥塞控制的协议是相当复杂的。我们将用几节来介绍可靠数据传输和拥塞控制的原理，用另外几节介绍TCP协议本身。这些主题将在3.4节至3.8节中研究。本章采取基本原理和TCP协议交替阐述的方法。例如，我们首先讨论一般环境下的可靠数据传输，然后讨论TCP是怎样提供可靠数据传输的。类似地，先讨论一般环境下的拥塞控制，然后讨论TCP是怎样实现拥塞控制的。在全面地介绍这些内容之前，我们先来学习运输层的多路复用与多路分解。

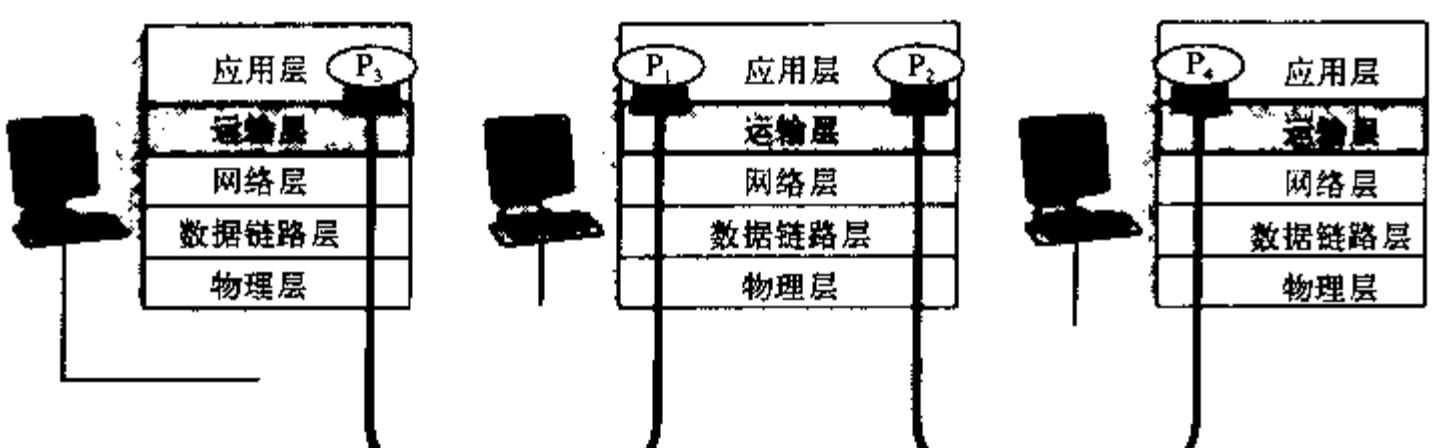
3.2 多路复用与多路分解

本节我们讨论运输层的多路复用与多路分解，也就是将网络层所提供的主机到主机交付

服务扩展到为在主机上运行的应用程序所提供的进程到进程交付服务。为了具体起见，我们将讨论因特网中的这种基本运输层服务。然而需要强调的是，多路复用与多路分解服务是所有计算机网络都需要的。

在目的主机，运输层从紧邻其下的网络层接收报文段。运输层负责将这些报文段中的数据交付给在主机上运行的合适的应用进程。我们来看一个例子。假定你正坐在计算机前下载Web页面，同时还在运行一个FTP会话和两个Telnet会话。此时你就有4个网络应用进程在运行，即两个Telnet进程、一个FTP进程和一个HTTP进程。当你的计算机中的运输层从底层的网络层接收数据时，它需要将所接收到的数据定向到这4个进程中的一个。现在我们来研究这是怎样实现的。

首先回想2.7节和2.8节的内容，进程（作为网络应用的一部分）有一个或多个**套接字**(socket)，它相当于从网络向进程传递数据和从进程向网络传递数据的门户。因此，如图3-2所示，接收主机中的运输层实际上并没有直接将数据交付给进程，而是通过一个中间的套接字来传递。由于在任何一个时刻接收主机上可能有多个套接字，所以每个套接字都有唯一的标识符。标识符的格式取决于它是UDP套接字还是TCP套接字，稍后将对它们进行讨论。



图例：
进程 套接字

图3-2 运输层的多路复用与多路分解

现在我们考虑接收主机怎样将一个收到的运输层报文段定向到合适的套接字。为达到这一目的，在每个运输层报文段中设置了几个字段。在接收端，运输层检查这些字段并标识出接收套接字，然后将报文段定向到该套接字。将运输层报文段中的数据交付到正确的套接字的工作称为**多路分解**（demultiplexing）。从源主机的不同套接字中收集数据块，并为每个数据块封装上首部信息（这将在多路分解时使用）从而生成报文段，然后将报文段传递到网络层的工作称为**多路复用**（multiplxing）。注意到图3-2的中间那台主机的运输层必须将从其下的网络层收到的报文段多路分解后交给其上的P_1或P_2进程，这一过程是通过将到来的报文段数据定向到相应进程的套接字来实现的。中间主机上的运输层也必须收集这些套接字输出的数据，形成运输层报文段，然后将其传递给下面的网络层。尽管我们在因特网运输层协议的环境下引入了多路复用和多路分解，但认识到以下事实是很重要的：它们与在某层（在运输层或其他层）的单一协议何时由位于次较高层的多个协议的使用有关。

为了说明多路分解的工作过程，考虑前一节中家庭的例子。每个孩子通过其名字来标识。当Bill从邮递员处收到一批信件，并通过查看收信人名字而将信件亲手交付给他的兄弟姐妹们时，他执行的就是一个多路分解操作。当Ann从兄弟姐妹们那里收集信件并将它们交给邮递员

时，她执行的就是一个多路复用操作。

既然我们理解了运输层多路复用与多路分解的作用，那就再来看看在主机中它们实际上是怎样工作的。通过上面的讨论，我们知道了运输层多路复用的要求：①套接字有唯一标识符，②每个报文段有特殊字段来指示该报文段所要交付的套接字。如图3-3所示，这些特殊字段是**源端口号字段**（source port number field）和**目的端口号字段**（destination port number field）。（UDP报文段和TCP报文段还有其他的一些字段，这些将在本章后继几节中阐述。）端口号是一个16 比特的数字，其大小在0～65 535之间。0～1023范围的端口号称为**周知端口号**（well-known port number），是受严格限制的，也就是说它们是保留给诸如HTTP（它使用端口号80）和FTP（它使用端口号21）之类的周知应用层协议的。周知端口的列表在RFC 1700中给出，同时在网站http://www.iana.org上有更新文档[RFC 3232]。当我们开发一个新的应用程序（如在2.7节和2.8节开发的一个应用程序）时，必须为其分配一个端口号。

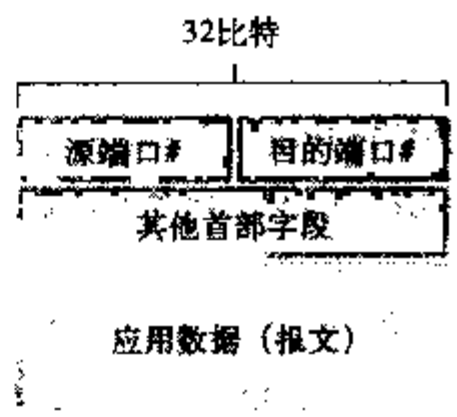


图3-3 运输层报文段中的源端口与目的端口字段

现在应该清楚运输层是怎样实现多路分解服务的了：主机上的每个套接字被分配一个端口号，当报文段到达主机时，运输层检查报文段中的目的端口号，并将其定向到相应的套接字。然后报文段中的数据通过套接字进入其所连接的进程。如我们将看到的那样，UDP基本上是这样做的。然而，也将如我们所见，TCP中的多路复用与多路分解更为复杂。

1. 无连接的多路复用与多路分解

2.8节讲过，在主机上运行的一个Java程序使用下面一行代码创建了一个UDP套接字：

```
DatagramSocket  mySocket = new DatagramSocket();
```

当用这种方式创建一个UDP套接字时，运输层自动地为该套接字分配一个端口号。特别是，运输层从范围1024～65 535内分配一个主机尚未使用的UDP端口号。Java程序也可以使用下面这行代码来创建套接字：

```
DatagramSocket  mySocket = new DatagramSocket(19157);
```

在这种情况下，应用程序为UDP套接字指派了一个特定的端口号19157。如果应用程序开发者所编写的代码实现的是一个“周知协议”的服务器端，那么他就必须为其分配一个相应的周知端口号。典型的情况是，应用程序的客户机端让运输层自动地（并且透明地）分配端口号，而服务器端则分配一个特定的端口号。

通过为UDP套接字分配端口号，我们现在能准确地描述UDP的多路复用与多路分解了。假定主机A中的一个进程具有的UDP端口号为19157，它要发送一个应用程序数据块给主机B中的另一进程，该进程具有的UDP端口号为46428。主机A中的运输层创建一个运输层报文段，其中包括应用程序数据、源端口号（19157）、目的端口号（46428）和两个其他值（将在后面讨论，这对当前的讨论并不重要）。然后，运输层将生成的报文段传递到网络层。网络层将该报文段封装到一个IP数据报中，并尽力而为地将报文段交付给接收主机。如果该报文段到达接收主机B，接收主机运输层就检查该报文段中的目的端口号（46428）并将该报文段传递给端口号46428所标识的套接字。注意到主机B能够运行多个进程，每个进程有自己的UDP套接字及相应的端口号。当从网络到达UDP报文段时，主机B通过检查该报文段中的目的端口号，

将报文段定向（多路分解）到相应的套接字。

注意到下述事实是重要的：一个UDP套接字是由一个包含目的IP地址和目的端口号的二元组来全面标识的。因此，如果两个UDP报文段有不同的源IP地址和/或源端口号，但具有相同的目的IP地址和目的端口号，那么这两个报文段将通过相同的目的套接字定向到相同的目的进程。

你也许现在想知道源端口号的用途是什么。如图3-4所示，在A到B的报文段中，源端口号作为“返回地址”的一部分，即当B需要发回一个报文段给A时，B到A的报文段中的目的端口号便从A到B的报文段的源端口号中取值。（完整的返回地址是A的IP地址和源端口号。）举个例子，回想2.8节研究过的UDP服务器程序。在UDPServer.java中，服务器通过一种方法从客户机发来的报文段中提取出源端口号，然后，它将所提取的源端口号作为目的端口号，向客户机发送一个新的报文段。

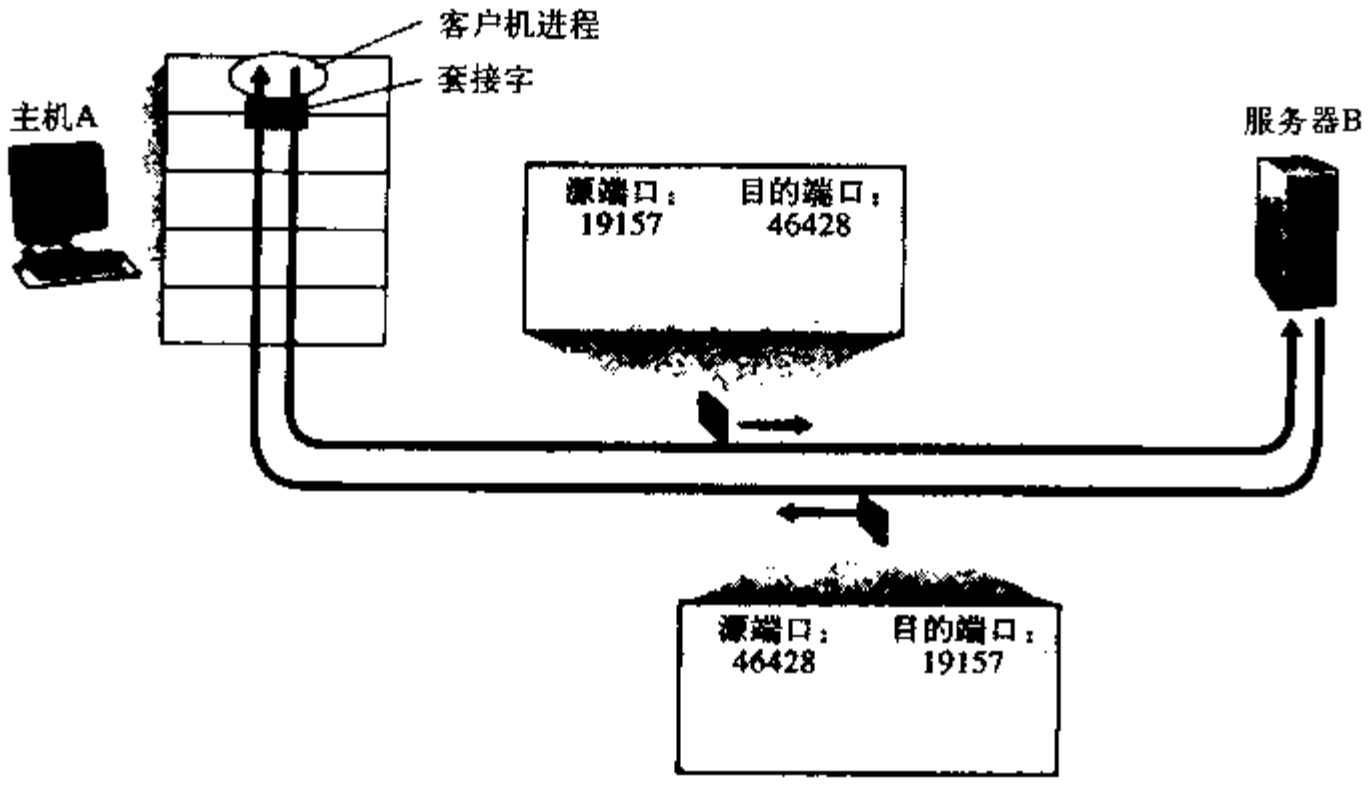


图3-4 源端口号与目的端口号的反转

2. 面向连接的多路复用与多路分解

为了理解TCP多路分解，我们必须更为仔细地研究TCP套接字和TCP连接创建。TCP套接字和UDP套接字之间的一个细微差别是，TCP套接字是由一个四元组（源IP地址，源端口号，目的IP地址，目的端口号）来标识的。这样，当一个TCP报文段从网络到达一台主机时，主机使用全部4个值来将报文段定向（多路分解）到相应的套接字。特别地，与UDP不同的是，两个具有不同源IP地址或源端口号的到达的TCP报文段将被定向到两个不同的套接字，除非TCP携带了初始创建连接的请求。为了深入地理解这一点，我们再来重新考虑2.7节中的TCP客户机/服务器编程例子。

- TCP服务器应用程序有一个“welcoming socket”，它在6789号端口上等待TCP客户机（见图2-31）的连接建立请求。
- TCP客户机使用下面这行代码产生一条连接建立报文段：

```
Socket  clientSocket = new Socket("serverHostName", 6789);
```

- 一个连接建立请求只不过是一个目的端口号为6789、对TCP首部的特定连接建立位置位的TCP报文段（在3.5节进行讨论）。这个报文段也包含一个源端口号，它是由客户机选择的。上面的程序行还为客户机进程创建了一个TCP套接字，通过该套接字客户机进程

可以发送和接收数据。

- 当运行服务器进程的计算机的主机操作系统接收到目的端口号为6789的入连接请求报文段后，它就定位服务器进程，该进程正在端口号6789等待接受连接。该服务器进程则创建一个连接套接字：

```
Socket connectionSocket = welcomeSocket.accept();
```

- 该服务器还关注连接请求报文段中的下列4个值：①该报文段中的源端口号，②源主机IP地址，③该报文段中的目的端口号，④自身的IP地址。新创建的连接套接字通过这4个值来标识。所有后续到达的报文段，如果它们的源端口号、源主机IP地址、目的端口号和目的IP地址都与这4个值匹配，则被多路分解到这个套接字。TCP连接完成以后，客户机和服务器便可以相互发送数据了。

服务器主机可同时支持很多TCP套接字，每个套接字与一个进程相联系，并由其四元组来标识每个套接字。当一个TCP报文段到达主机时，所有4个字段（源IP地址，源端口号，目的IP地址，目的端口号）用来定向（多路分解）报文段到相应的套接字。

关注安全

端口扫描

我们已经看到一个服务器进程潜在地在一个打开的端口等待远程客户机的请求。某些端口为周知应用程序（例如，Web、FTP、DNS和SNMP服务器）所预留，其他端口依照惯例运行流行的应用程序（例如，微软2000 SQL服务器在UDP 1434端口上监听请求）。因此，如果我们确定一台主机上打开了一个端口，就能够将在该主机上运行的一个特定的应用程序映射到该端口上。系统管理员通常对知晓什么样的网络应用程序运行在其网络主机上感兴趣，所以这对于系统管理员非常有用。而攻击者为了“寻找突破口”，也要知道目标主机上打开了哪些端口。如果发现一台主机正在运行具有已知安全缺陷的应用程序（例如，在端口1434上监听的一个SQL服务器会遭受缓存溢出，使得远程用户在易受攻击的主机上执行任意代码，这是一种由Slammer蠕虫所利用的缺陷[CERT 2003-04]），那么该主机就成为攻击者的囊中之物了。

确定哪个应用程序正在监听哪些端口是一件相对容易的事情。事实上，有许多公共域程序（称之为端口扫描器）做得正是这种事情。其中使用最广泛的是nmap，该程序可以从http://insecure.org/nmap上免费下载，并且包括在大多数Linux分布软件中。对于TCP，nmap顺序地扫描端口，寻找能够接受TCP连接的端口。对于UDP，nmap也是顺序地扫描端口，寻找对传输的UDP报文段进行响应的UDP端口。在这两种情况下，nmap返回打开的、关闭的或不可达的端口列表。运行nmap的主机能够扫描因特网中任何地方的目的主机。我们将在3.5.6节中讨论TCP连接管理时介绍nmap。

在图3-5中对这种情况进行了说明，图中主机C向服务器B发起了两个HTTP会话，主机A向服务器B发起了一个HTTP会话。主机A与主机C及服务器B都有自己的唯一IP地址，分别是A、C、B。主机C为其两个HTTP连接分配了两个不同的源端口号（26145和7532）。因为主机A选择源端口号时与主机C互不相干，因此它也可以将源端口号26145分配给其HTTP连接。然而，服务器B仍然能够正确地多路分解这两个具有相同源端口号的连接，因为这两个连接有不同的源IP地址。

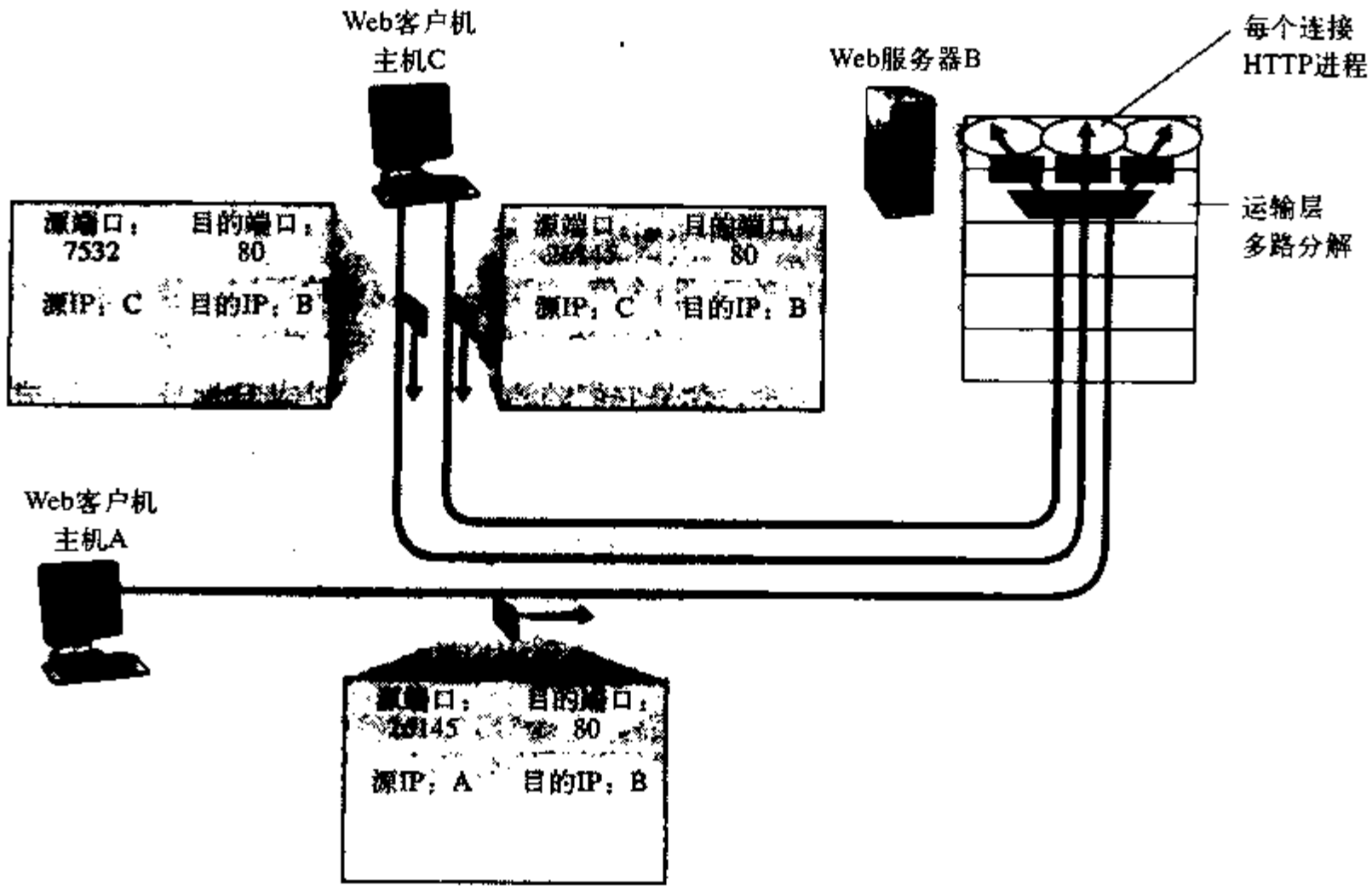


图3-5 两个客户机使用相同的目的端口号（80）与同一个Web服务器应用通信

3. Web服务器与TCP

在结束本节讨论之前，简述一下Web服务器以及它们如何使用端口号。假设有一台运行Web服务器的主机，如在端口80上运行一个Apache Web服务器。当客户机（如浏览器）向服务器发送报文段时，所有报文段的目的端口都为80。特别是初始连接建立报文段和承载HTTP请求的报文段的目的端口都为80。如前所述，服务器能够根据源IP地址和源端口号来区分来自不同客户机的报文段。

图3-5显示了为每个连接产生一个新进程的一台Web服务器。如图3-5所示，每个这样的进程都有自己的连接套接字，通过这些套接字可以收到HTTP请求和发送HTTP响应。然而，我们要指出的是，连接套接字与进程之间并非总是有着一一对应的关系。事实上，当今的高性能Web服务器通常只使用一个进程，但是为每个新的客户机连接创建一个具有新连接套接字的新线程。（线程可以看作是一个轻量级的子进程。）如果你做了第2章的第一个编程作业，你所构建的Web服务器就是这样工作的。对于这样的服务器，在任意给定的时间内都可能有许多连接套接字（具有不同的标识）连接到相同的进程。

如果客户机与服务器使用持久HTTP，则在整个连接持续期间，客户机与服务器之间经由同一个服务器套接字交换HTTP报文。然而，如果客户机与服务器使用非持久HTTP，则对每一对请求/响应，都有一个新的TCP连接被创建，请求/响应完成后随即被关闭。这种频繁地创建和关闭套接字会严重地影响一个繁忙的Web服务器的性能（尽管有许多操作系统技巧可用来减轻这个问题的影响）。读者若对与持久和非持久HTTP有关的操作系统问题感兴趣的话，可参见[Nielsen 1997；Nahum 2002]。

现在我们已经讨论了运输层多路复用与多路分解问题，下面我们来讨论因特网运输层协议之一——UDP。在下一节中，我们将知道UDP无非就是对网络层协议增加了一点多路复用/

多路分解服务。

3.3 无连接运输：UDP

在本节中，我们要仔细地研究一下UDP，看它是怎样工作的，能做些什么。读者可以参考一下2.1节，其中包括了UDP服务模型的概述，再看看2.8节，其中讨论了UDP上的套接字编程。

为了激发我们讨论UDP的热情，假如你对设计一个只提供必要服务的最简化的主干运输层协议感兴趣。你将打算怎样做？你也许会首先考虑使用一个无所事事的运输层协议。特别是在发送方，你可能会考虑将来自应用进程的数据直接交给网络层；在接收方，你可能会考虑把来自网络层的报文直接交给应用进程。而正如我们在前一节所学的那样，我们必须做一点事，而不是什么都不做！至少运输层必须提供一个多路复用/多路分解服务，以便在网络层与正确的应用级进程之间传递数据。

由RFC 768定义的UDP只是做了运输协议能够做的最少工作。除了多路复用/多路分解功能及一些轻型的差错检测外，它几乎没有对IP增加别的东西。实际上，如果应用程序开发人员选择UDP而不是TCP，则应用程序几乎是直接与IP打交道。UDP从应用进程得到数据，附加上多路复用/多路分解服务所需的源端口号和目的端口号字段，及两个其他的小字段，然后将形成的报文段交给网络层。网络层将该运输层报文段封装到一个IP数据报中，然后尽力而为地将此报文段交付给接收主机。如果该报文段到达接收主机，则UDP使用目的端口号来将报文段中的数据交付给正确的应用进程。注意到使用UDP时，在发送报文段之前，发送方和接收方的运输层实体之间没有进行握手。正因如此，UDP被称为*无连接的*。

DNS是一个通常使用UDP的应用层协议的例子。当一台主机中的DNS应用程序想要进行一次查询时，它构造了一个DNS查询报文并将其交给UDP。无须执行任何与运行在目的端系统中的UDP实体之间的握手，主机端的UDP为此报文添加首部字段，然后将形成的报文段交给网络层。网络层将此UDP报文段封装进一个IP数据报中，然后将其发送给一个名字服务器。查询主机中的DNS应用程序便等待对该查询的响应。如果它没有收到响应（可能是由于底层网络丢失了查询或响应），则要么试图向另一个名字服务器发送该查询，要么通知调用的应用程序它不能获得响应。

现在你也许想知道，为什么应用开发人员宁愿在UDP之上构建应用，而不选择TCP呢？既然TCP提供了可靠的数据传输服务，而UDP不能提供，那么TCP是否总是首选的呢？答案是否定的，因为有许多应用更适合用UDP，这主要有以下原因：

- *应用层能更好地控制要发送的数据和发送时间。*采用UDP时，只要应用进程将数据传递给UDP，UDP就会将此数据打包成UDP报文段并立即将其传递给网络层。另一方面，TCP有一个拥塞控制机制以便当源和目的主机间的一条或多条链路变得非常拥塞时，遏制运输层TCP发送方。TCP仍将继续重新发送数据报文段直到目的主机收到此报文并加以确认，而不管可靠交付需要用多长时间。实时应用通常要求最快的发送速率，不想过分地延迟报文段的传送，且能容忍一些数据丢失，而TCP服务模型并不是特别适合这些应用的需求。如下面要讨论的那样，这些应用可以使用UDP，并作为应用的一部分来实现所需的、超出UDP的不提供不必要服务的报文段交付之外的额外功能。
- *无需连接建立。*如我们将讨论的那样，TCP在开始数据传输之前要经过三次握手。UDP却不需要任何准备即可进行数据传输。因此UDP不会引入建立连接的时延。这可能是

DNS运行在UDP之上而不是运行在TCP之上的主要原因（如果运行在TCP上，则DNS会慢得多）。HTTP使用TCP而不是UDP，是因为对于具有文本数据的Web网页来说，可靠性是至关重要的。但是，如2.2节中简述的那样， HTTP中的TCP连接建立时延是与下载Web文档相关的时延的重要因素。

- *无连接状态*。TCP需要在端系统中维护连接状态。此连接状态包括接收和发送缓存、拥塞控制参数、序号与确认号的参数。我们将在3.5节看到，要实现TCP的可靠数据传输服务并提供拥塞控制，这些状态信息是必需的。另一方面，UDP不维护连接状态，也不跟踪这些参数。因此，某些专门应用服务器使用UDP而不是TCP，以便能支持更多的活动客户机。
- *分组首部开销小*。每个TCP报文段都有20 字节的首部开销，而UDP仅有8 字节的开销。

图3-6列出了目前流行的因特网应用及其所使用的运输层协议。如我们所期望的那样，电子邮件、远程终端访问、Web及文件传输都是运行在TCP之上。因为所有这些应用都需要TCP的可靠数据传输服务。然而，有很多重要的应用是运行在UDP上而不是TCP上。UDP被用于RIP选路表的更新（参见4.6.1节），因为RIP更新被周期性地发送（通常每5分钟一次），以便丢失的更新能被最近的更新所替代，因此更新丢失或过时的RIP是毫无意义的。UDP也用于承载网络管理数据（SNMP，参见第9章）。在这种场合下，UDP要优于TCP，因为网络管理应用程序通常必须在该网络处于重压状态时运行，而正是在这时可靠的、拥塞受控的数据传输变得难以实现。此外，如我们在前面提到的那样，DNS运行在UDP之上，从而避免了TCP的连接创建时延。

应用	应用层协议	下面的运输层协议
电子邮件	SMTP	TCP
远程终端访问	Telnet	TCP
Web	HTTP	TCP
文件传输	FTP	TCP
远程文件服务器	NFS	通常UDP
流式多媒体	通常专用	UDP或TCP
因特网电话	通常专用	UDP或TCP
网络管理	SNMP	通常UDP
选路协议	RIP	通常UDP
名字转换	DNS	通常UDP

图3-6 流行的因特网应用及其采用的运输层协议

如图3-6所示，UDP和TCP现在都用于多媒体应用，如因特网电话、实时视频会议、流式存储音频与视频。我们将在第7章仔细研究这些应用。我们刚说过，既然所有这些应用都能容忍少量的分组丢失，那么可靠数据传输对于这些应用的成功并不是至关重要的。此外，TCP的拥塞控制会导致如因特网电话、视频会议之类的实时应用性能变得很差。由于这些原因，多媒体应用开发人员通常将这些应用运行在UDP之上而不是TCP之上。然而，TCP被越来越多地应用于流式多媒体传输中。例如，[Sripanidkulchai 2004]发现接近75%的按需和实况流式多媒体使用了TCP。当分组丢包率低，并且为了安全某些机构阻塞UDP流量（参见第8章）时，

TCP变得对于流式媒体传输越来越具有吸引力了。

虽然目前通常这样做，但在UDP之上运行多媒体应用是有争议的。如前面所述，UDP没有拥塞控制。但是，需要拥塞控制来预防网络进入一种拥塞状态，此时只能做很少有用的工作。如果每个人都启动流式高比特率视频而无任何拥塞控制，则会使路由器中有大量的分组溢出，以至于几乎没有UDP分组能成功地通过源到目的的路径传输。此外，由无控制的UDP发送方引入的高丢包率将引起TCP发送方（如我们将看到的那样，发送方遇到拥塞将减小它们的发送速率）显著减小它们的速率。因此，UDP中缺乏拥塞控制能够导致UDP发送方和接收方之间的高丢包率，并挤垮了TCP会话，这是一个潜在的严重问题[Floyd 1999]。很多研究人员已提出了一些新的机制，以促使所有的数据源（包括UDP源）执行自适应的拥塞控制[Mahdavi 1997；Floyd 2000; Kohler 2006；RFC 4340]。

在讨论UDP报文段结构之前，我们要提一下，使用UDP的应用是可以实现可靠数据传输的。这可通过在应用程序自身中建立可靠性机制来完成（例如，可通过增加确认与重传机制来实现，如采用我们将在下一节学习的一些机制）。但这并不是无足轻重的任务，它会使应用开发人员用一段较长的时间进行调试。然而，将可靠性直接构建于应用程序中可以使其“得失兼顾”。也就是说，应用进程可以进行可靠通信，而无需受制于由TCP拥塞控制机制而引起的传输速率约束。

3.3.1 UDP报文段结构

UDP报文段结构如图3-7所示，它由RFC 768定义。应用层数据占用了UDP报文段的数据字段。例如，对于DNS应用，数据字段要么包含一个查询报文，要么包含一个响应报文。对于流式音频应用，音频抽样数据填充到数据字段。UDP首部只有4个字段，每个字段由两个字节组成。前一节已经讨论过，通过端口号可以使目的主机将应用数据交给运行在目的端系统中的相应进程（即执行多路分解功能）。接收主机使用检验和来检查报文段中是否存在差错。事实上，计算检验和时，除了UDP报文段以外还使用了IP首部的一些字段。但是我们忽略这些细节，以便从整体上看问题。下面，我们讨论检验和的计算。5.2节中将描述差错检测的基本原理。长度字段指明了包括首部在内的UDP报文段长度（以字节为单位）。

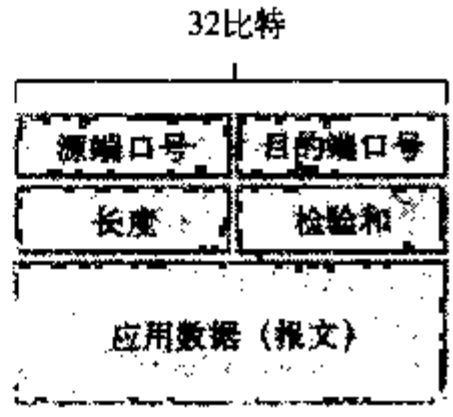


图3-7 UDP报文段结构

3.3.2 UDP检验和

UDP检验和提供了差错检测功能，即检验和用于确定当UDP报文段从源到达目的时，其中的比特是否发生了改变（例如，由于链路中或者路由器中存储数据时的噪声干扰）。发送方的UDP对报文段中的所有16 比特字的和进行反码运算，求和时遇到的任何溢出都被回卷。得到的结果放在UDP报文段中的检验和字段。下面给出一个计算检验和的简单例子。你可以在RFC 1071中找到有效实现的细节，还可在[Stone 1998; Stone 2000]中找到它对实时数据的性能。作为一个例子，假定我们有下面3个16 比特字：

0110011001100000
0101010101010101
1000111100001100

前两个16 比特字之和是：

```
  0110011001100000
  0101010101010101
  ----------------
  1011101110110101
```

再将上面的和与第三个字相加，得到：

```
  1011101110110101
  1000111100001100
  ----------------
  0100101011000010
```

注意到最后一次加法有溢出，这要被回卷。反码运算是将所有的0换成1，所有的1换成0。因此，值0100101011000010的反码运算结果是1011010100111101，这就是检验和。在接收方，全部的4个16 比特字（包括检验和）一起相加。如果分组中无差错，则显然在接收方这个和将是1111111111111111。如果有1个比特是0，那么我们就知道分组中出现了差错。

你可能想知道为什么UDP首先提供了检验和，如同许多链路层协议（包括流行的以太网协议）也提供差错检测那样。原因是由于不能保证源和目的之间的所有链路都提供差错检测，即其中也许有一条链路使用没有差错检测的协议。此外，即使报文段经链路正确地传输，当报文段存储在某台路由器的内存中时，也可能引入比特差错。既未确保逐段链路的可靠性，又未确保内存中的差错检测，UDP必须在端到端基础上在运输层提供差错检测。这是一个在系统设计中被赞扬的端到端原则（end-end princeple）[Saltzer 1984]，该原则表述为，因为某种功能（如本例中的差错检测）必须基于端到端实现，“与在较高级别提供这些功能的代价相比，在较低级别上设置的功能可能是冗余的或几乎没有价值。”

因为假定IP是可以运行在任何第二层协议之上的，所以运输层提供差错检测作为一种保险措施是非常有用的。虽然UDP提供差错检测，但它并不能进行差错恢复。某些UDP实现只是丢弃受损的报文段，其余的实现是将受损的报文段交给应用程序并告警。

至此结束了关于UDP的讨论。我们将很快看到TCP为应用提供了可靠数据传输及UDP所不能提供的其他服务。很显然，TCP要比UDP复杂得多。然而，在讨论TCP之前，我们后退一步，先讨论一下可靠数据传输的基本原理。

3.4 可靠数据传输的原理

在本节中，我们考虑在一般情况下可靠数据传输的问题。因为可靠数据传输的实现问题不仅在运输层出现，也在链路层以及应用层出现，这时讨论它是恰当的。因此，一般性的问题对网络来说更为重要。实际上，如果排名网络中最重要的“前10个”问题，可靠数据传输无疑是名列榜首的候选者。在下一节中，我们将学习TCP，尤其要说明TCP所采用的许多原理。

图3-8举例说明了我们学习可靠数据传输的框架。提供给上层实体的服务抽象是，数据可以通过一条可靠的信道进行传输。有了可靠信道，就不会有传输数据比特受到损坏（由0变为1，或者相反）或丢失，而且所有数据都是按照其发送顺序进行传送。这恰好就是TCP为调用它的因特网应用所提供的服务模型。

实现这种服务抽象是**可靠数据传输协议**（reliable data transfer protocol）的责任。由于事实上可靠数据传输协议的下层协议也许是不可靠的，因此这项任务是较困难的。例如，TCP是在不可靠的端到端网络层协议（IP）之上实现的可靠数据传输协议。更一般的情况是，两个可靠通信端点的下层可能是一条物理链路（如在链路层传输协议的场合）或是一个全球互联网络（如在运输层协议的场合）。然而，为便于学习，我们可将较低层视为不可靠的点对点信道。

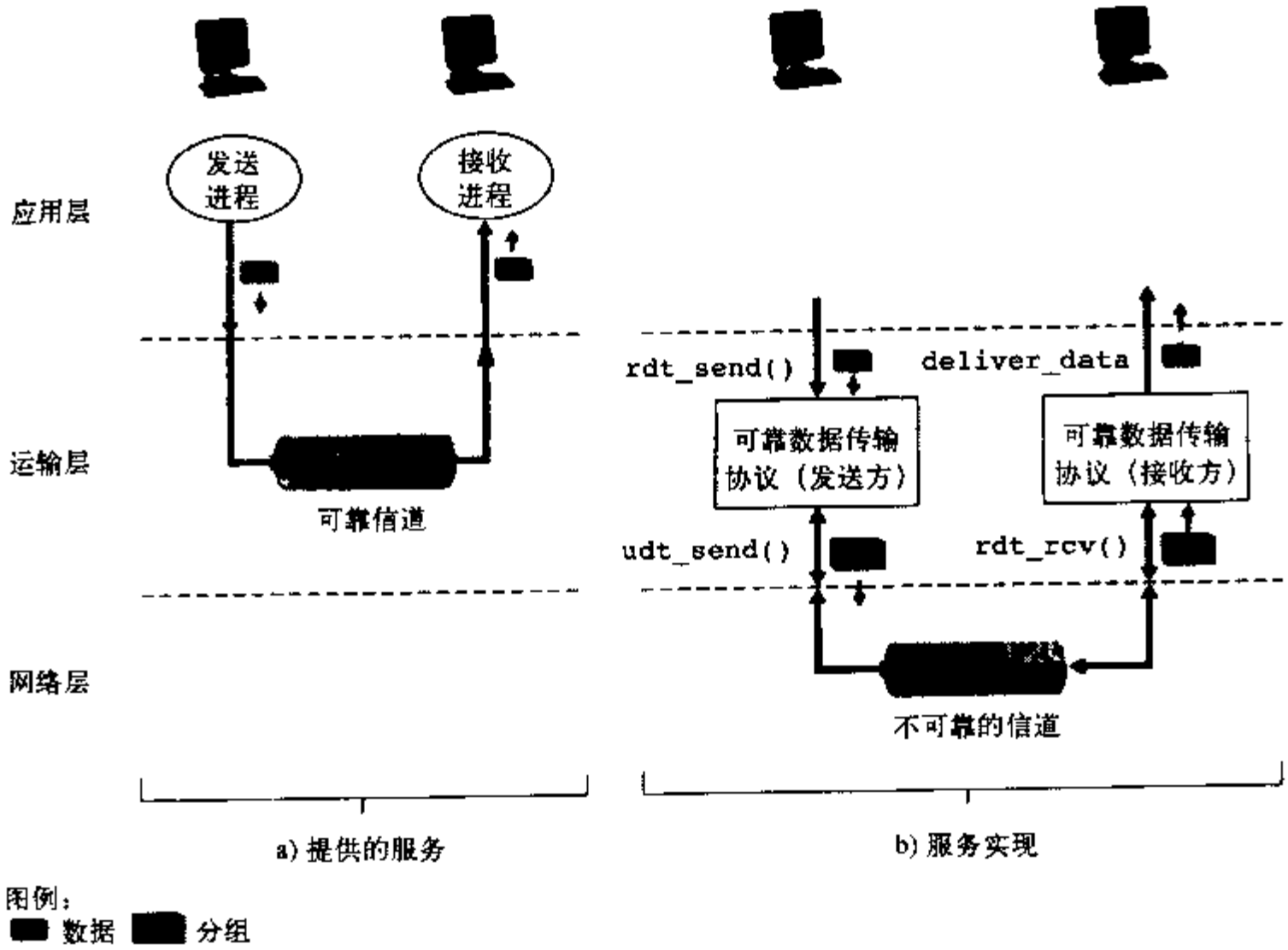


图3-8 可靠数据传输：服务模型与服务实现

在本节中，考虑到底层信道模型的复杂性越来越高，我们将逐步地开发可靠数据传输协议的发送方和接收方。图3-8b举例说明了我们的数据传输协议的接口。通过rdt_send()函数，可以调用数据传输协议的发送方。它将要发送的数据交付给接收方的上层。（这里rdt表示可靠数据传输协议，_send表明rdt的发送方正在被调用。开发任何协议的第一步就是要选择一个好的名字！）在接收方，当分组从信道的接收端抵达时，将调用rdt_rcv()。当rdt协议想向较高层交付数据时，通过调用deliver_data()完成。后面，我们将使用术语“分组”而不用运输层“报文段”。因为本节研讨的理论适用于一般的计算机网络，而不仅是用于因特网运输层，所以通用术语“分组”更为合适。

在本节中，我们仅考虑**单向数据传输**（unidirectional data transfer）的情况，即数据传输是从发送方到接收方的。**可靠的双向数据传输**（bidirectional data transfer）（即全双工数据传输）的情况从概念上讲不会更难，但解释起来更为单调乏味。虽然我们只考虑单向数据传输，但也要注意我们的协议需要在发送方和接收方两个方向上传输分组，如图3-8所示。我们很快会看到，除了交换含有待发数据的分组之外，rdt的发送方和接收方还需来回交换控制分组。rdt的发送方和接收方都要通过调用udt_send()发送分组给对方（其中udt表示不可靠数据传输）。

3.4.1 构造可靠数据传输协议

我们现在要研究一系列协议，它们一个比一个更复杂，最后得到一个完整的可靠数据传输协议。

1. 完全可靠信道上的可靠数据传输：rdt1.0

首先，我们考虑最简单的情况，即底层信道是完全可靠的。我们称该协议为rdt1.0，该

协议本身是微不足道的。图3-9显示了rdt1.0发送方和接收方的有限状态机（finite-state machine，FSM）定义。图3-9a中的FSM定义了发送方的操作，图3-9b中的FSM定义了接收方的操作。注意到下列问题是重要的，即发送方和接收方有各自的FSM。图3-9中发送方和接收方FSM都只有一个状态。FSM描述图中的箭头指示了协议从一个状态变迁到另一个状态。（因为图3-9中的每个FSM都只有一个状态，因此变迁必定是从原状态回到自身。我们很快会看到更复杂的状态图。）引起变迁的事件显示在表示变迁的横线上方，事件发生时所采取的动作显示在横线下方。如果对一个事件没有采取动作，或没有就事件发生而采取了一个动作，我们将在横线上方或下方使用符号^，以分别明确地表示缺少动作或事件。FSM的初始状态用虚线表示。尽管图3-9中的FSM只有一个状态，但很快我们就将看到多状态的FSM，因此标识每个FSM的初始状态是非常重要的。

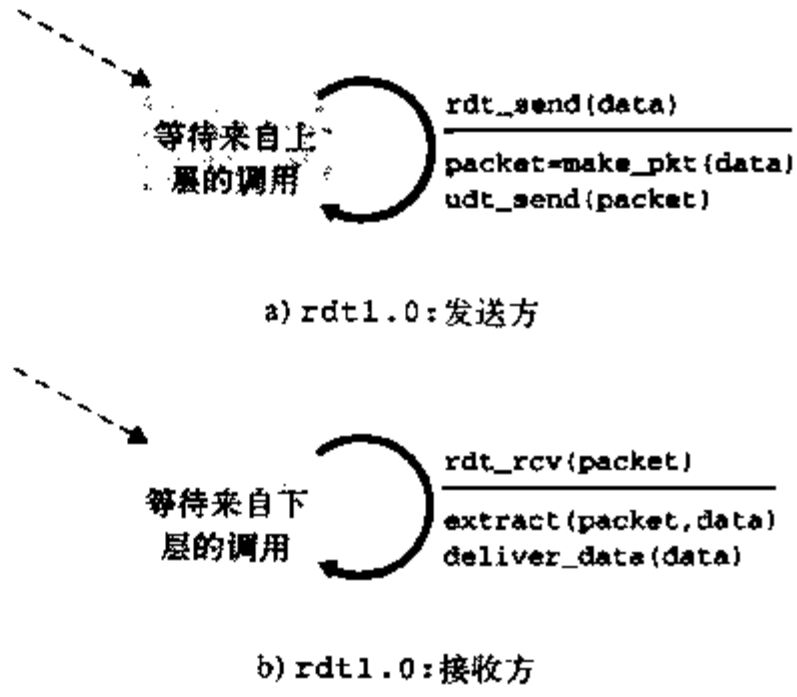


图3-9 rdt1.0：用于完全可靠信道的协议

rdt的发送方只通过rdt_send(data)从较高层接收的数据，产生一个包含该数据的分组（经由make_pkt(data)动作），并将分组发送到信道中。实际上，rdt_send(data)事件是由较高层应用的过程调用（例如，rdt_send()）产生的。

在接收方，rdt通过rdt_rcv(packet)事件从底层信道接收一个分组，从分组中取出数据（经由extract(packet, data)动作），并将数据上传给较高层（通过deliver_data(data)动作）。实际上，rdt_rcv(packet)事件是由较低层协议的过程调用（例如，rdt_rcv()）产生的。

在这个简单的协议中，一个单元数据与一个分组没什么差别，而且所有的分组是从发送方流向接收方。有了完全可靠的信道，接收方就不需要提供任何反馈信息给发送方，因为不会发生任何差错！注意，我们假定了接收方接收数据的速率和发送方发送数据的速率一样快。因此，接收方没有必要请求发送方放慢发送速度！

2. 具有比特差错信道上的可靠数据传输：rdt2.0

更现实的底层信道模型是分组中的比特可能受损。在分组的传输、传播或缓存的过程中，这种比特差错通常会出现在网络的物理部件中。我们继续假定所有传输的分组（虽然有些比特可能受损）将按其发送的顺序被接收。

在开发在这种信道上实现可靠通信的协议之前，首先考虑一下人们会怎样处理这类情形。考虑一下你自己是怎样通过电话口述一条长消息的。通常情况下，报文接收者在听到、明白、记下每句话后可能会说“OK”。如果消息接收者听到一句含糊不清的话，他可能要求你重复刚才那句话。这种口述消息协议使用了**肯定确认**（positive acknowledgment）（“OK”）与**否定确认**（negative acknowledgment）（“请重复一遍”）。这些控制报文使得接收方可以让发送方知道哪些内容被正确接收，哪些内容接收有误从而需要重传。在计算机网络环境中，基于这种重传机制的可靠数据传输协议称为**自动重传请求**（Automatic Repeat reQuest，ARQ）协议。

一般来说，ARQ协议中还需要另外三种协议来处理存在的比特差错：

- 差错检测。首先，需要一种机制以使接收方检测到何时出现了比特差错。前一节讲到，UDP使用互联网检验和字段正是为了这个目的。在第5章中，我们将更详细地学习差错检测和纠错技术。这些技术使接收方可以进行差错检测并可能纠正分组中的比特差错。现在，我们只需知道这些技术要求有额外的比特（除了待发送的初始数据比特之外的比特）从发送方发送到接收方，这些比特将被汇集在rdt2.0数据分组的分组检验和字段中。
- 接收方反馈。因为发送方和接收方通常在不同端系统上执行，可能相隔数千英里，发送方要了解接收方情况（即分组是否被正确接收）的唯一途径就是让接收方提供明确的反馈信息给发送方。口述消息中回答的肯定确认（ACK）和否定确认（NAK）就是这种反馈的例子。类似地，我们的rdt2.0协议将从接收方向发送方回送ACK与NAK分组。理论上，这些分组只需要一个比特长度，如用0表示NAK，用1表示ACK。
- 重传。接收方收到有差错的分组时，发送方将重传该分组。

图3-10给出了表示rdt2.0的FSM，该数据传输协议采用了差错检测、肯定确认与否定确认。

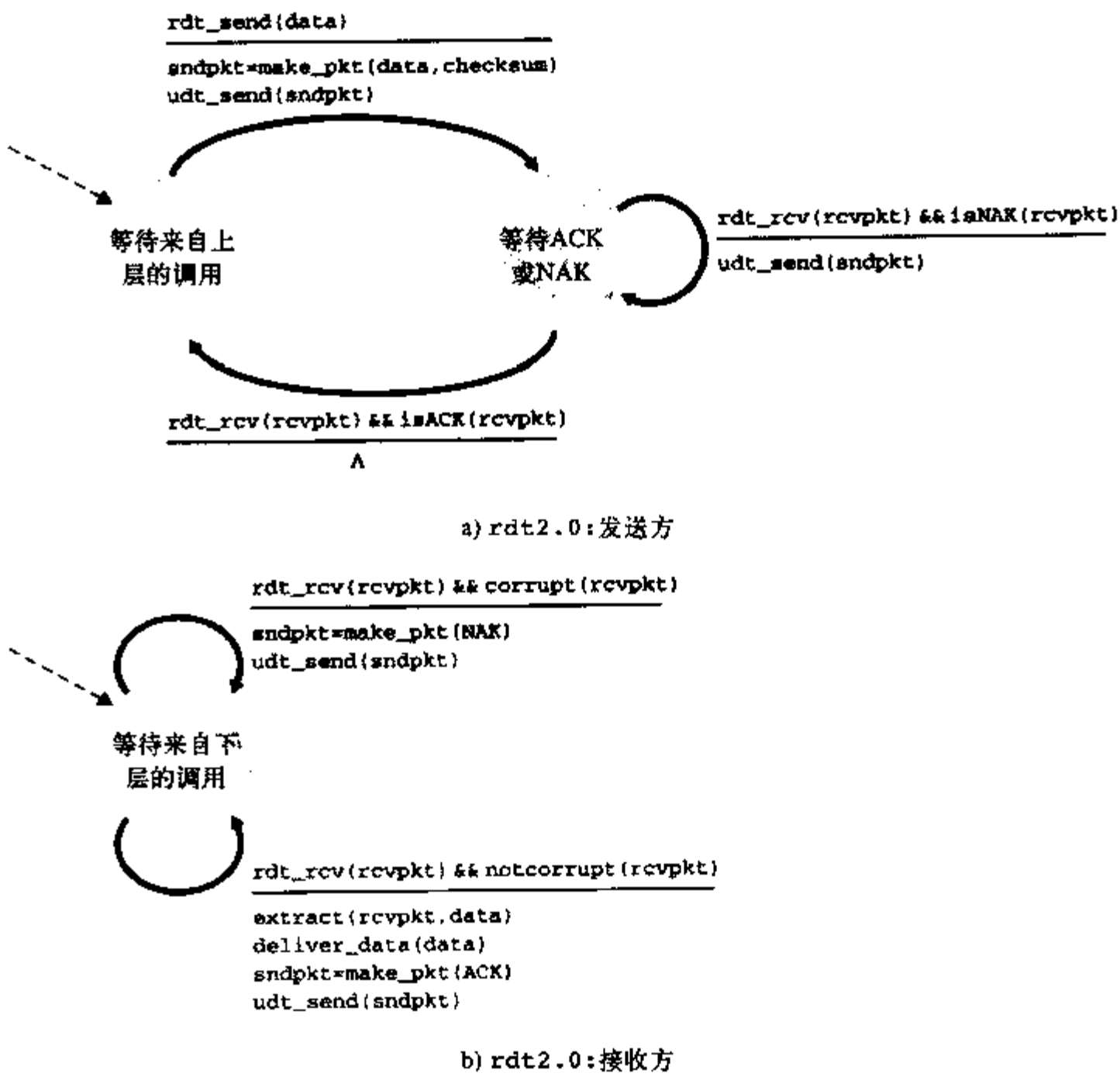


图3-10 rdt2.0：用于信道有比特差错的协议

rdt2.0的发送方有两个状态。在最左边的状态中，发送方协议正等待来自上层的数据。当rdt_send(data)事件发生时，发送方将产生一个包含待发送数据的分组（sndpkt），

计算出分组检验和（例如，就像3.3.2节讨论的UDP报文段使用的方法），然后经由udt_send(pkt)操作发送该分组。在最右边的状态中，发送方协议等待接收方的ACK或NAK分组。如果收到一个ACK分组（图3-10中符号rdt_rcv(rcvpkt)&&isACK(rcvpkt)对应该事件），则发送方知道最近传输的分组已被正确接收，因此协议返回到等待来自上层数据的状态。如果收到一个NAK分组，该协议重传最后一个分组并等待接收方返回的响应重传分组的ACK或NAK。注意，当发送方在wait-for-ACK-or-NAK状态时，它不能从上层获得更多的数据；这就是说，rdt_send()事件不可能出现；仅当接收到ACK并离开该状态时接收方才能继续获取数据。因此，发送方将不会发送一块新数据，直到发送方确信接收方已正确接收当前分组为止。由于这种行为，类似于rdt2.0的协议被称为**停等**（stop-and-wait）协议。

rdt2.0接收方的FSM仍然只有一个状态。当分组到达时，接收方要么回答一个ACK，要么回答一个NAK，这取决于收到的分组是否受损。在图3-10中，符号rdt_rcv(rcvpkt)&&corrupt(rcvpkt)对应于收到一个分组并发现有错的事件。

rdt2.0协议看起来似乎可以运行了，但遗憾的是，它存在一个致命的缺陷。尤其是我们没有考虑到ACK或NAK分组受损的可能性！（在继续研究之前，你应该考虑怎样解决该问题。）遗憾的是，我们细小的疏忽并非像它看起来那么无关紧要。至少，我们需要在ACK/NAK分组中添加检验和比特以检测这样的差错。更难的问题是，协议应该怎样纠正ACK或NAK分组中的差错。这里的难点在于，如果一个ACK或NAK分组受损，发送方无法知道接收方是否正确接收了上一块发送的数据。

处理受损ACK或NAK时需要考虑以下3种可能性：

- 第一种可能是，考虑在口述消息情况下人们的做法。如果说话者不明白受话者的“OK”或“请重复一遍”，说话者可能问“你说什么？”（因此在我们的协议中引入了一种新型发送方到接收方的分组）。说话者则复述该回复。但是如果说话者的“你说什么？”产生了差错，情况又会怎样呢？接收者不明白那句混淆的话是口述内容的一部分还是一个要求重复上次回答的请求，很可能回一句“你说什么？”于是，该回答可能含糊不清了。显然，我们走上了一条困难重重之路。
- 第二种可能是，增加足够的检验和比特，使发送方不仅可以检测比特差错，还可恢复比特差错。对于会产生差错但不丢失分组的信道，这就可以直接解决问题。
- 第三种可能是，当发送方收到含糊不清的ACK或NAK分组时，只需地重发当前数据分组即可。然而，这种方法在发送方到接收方的信道中引入了**冗余分组**（duplicate packet）。冗余分组的根本困难在于接收方不知道它上次所发送的ACK或NAK是否被发送方正确地收到，因此它无法事先知道接收到的分组是新的还是一次重传！

解决这个新问题的一个简单方法（几乎所有现存的数据传输协议，包括TCP，都采用了这种方法）是，在数据分组中添加一新字段，让发送方对其数据分组编号，即将发送的数据分组的**序号**（sequence number）放在该字段。于是，接收方只需要检查序号即可确定收到的分组是否是一次重传。对于停等协议这种简单情况，1比特序号就足够了，因为它可让接收方知道发送方在重传前一个发送分组（接收到的分组序号与最近收到的分组序号相同），还是一个新分组（序号变化了，用模2运算“前向”移动）。因为目前我们假定信道不丢失分组，ACK和NAK分组本身不需要指明它们要确认的分组序号。发送方知道所接收到的ACK和NAK分组（无论是否含糊不清）是为响应其最近发送的数据分组而生成的。

图3-11和图3-12给出了对rdt2.1的FSM描述，这是对rdt2.0的修订版。rdt2.1的发送

方和接收方FSM的状态数都是以前的两倍。这是因为协议状态此时必须反映出目前（由发送方）正发送的分组或希望（在接收方）接收的分组的序号是0还是1。注意到发送或期望接收0号分组的状态中的动作与发送或期望接收1号分组的状态中的动作是相似的，唯一的不同是序号处理方法不同。

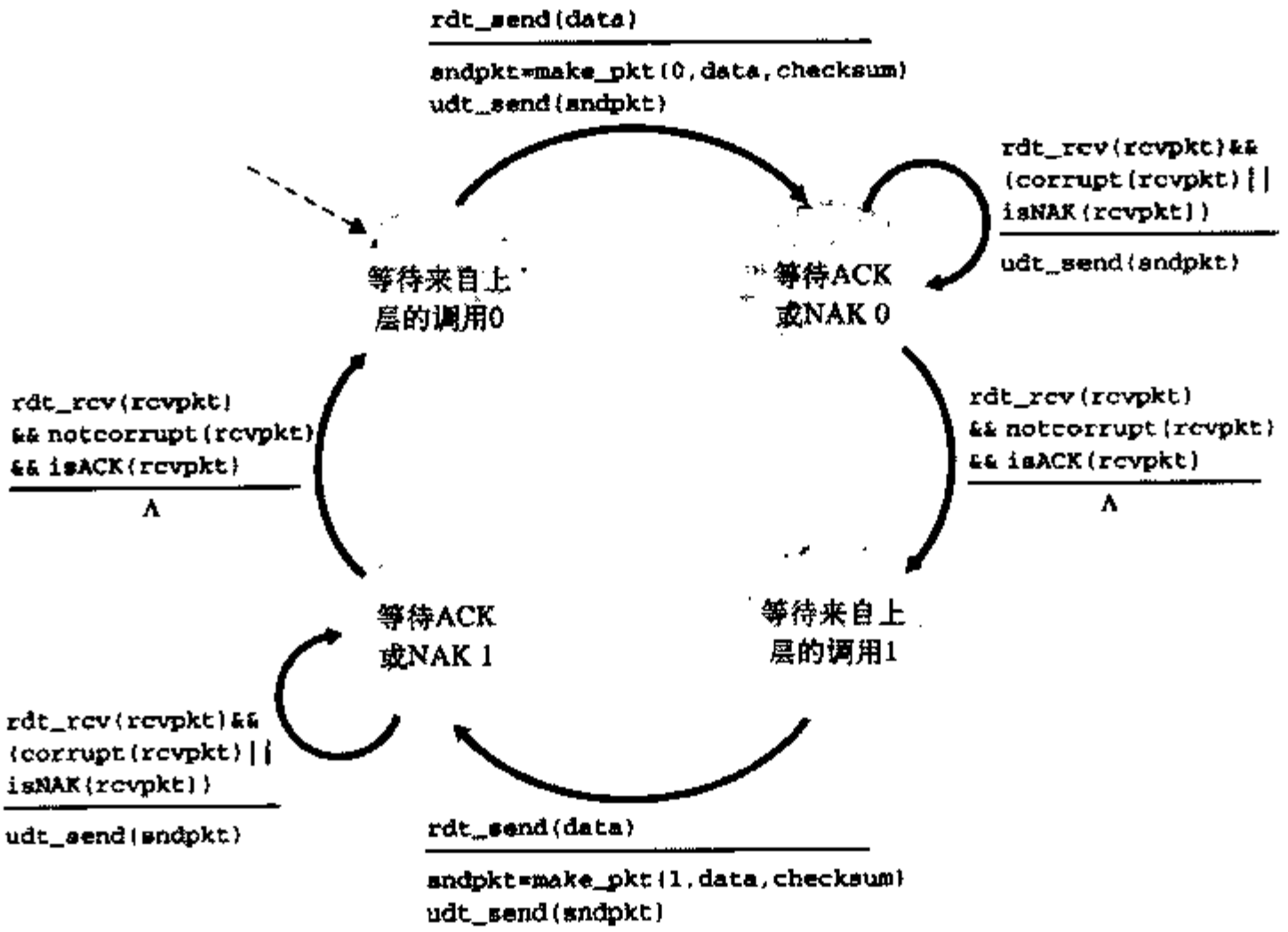


图3-11 rdt2.1发送方

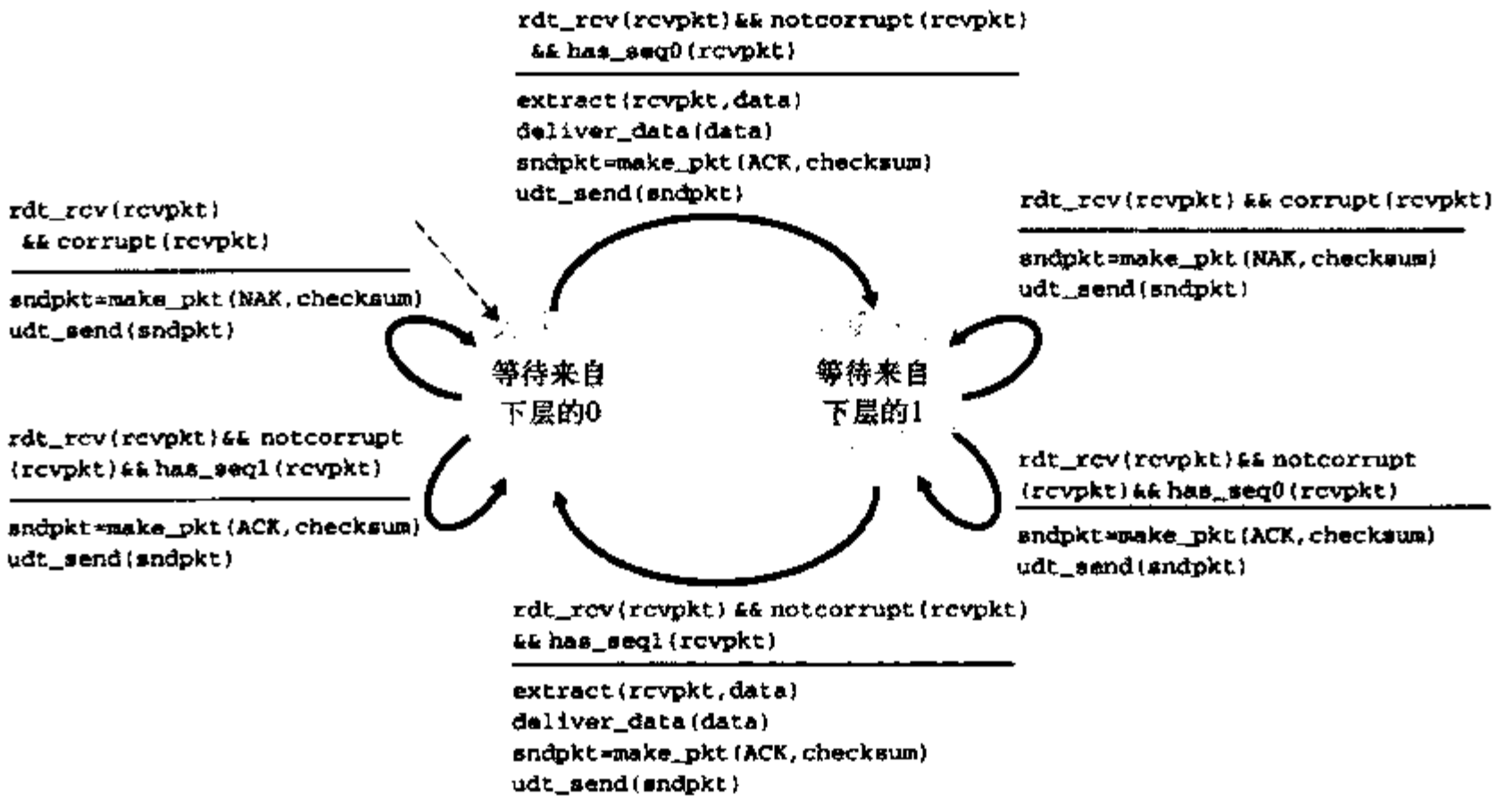


图3-12 rdt2.1接收方

协议rdt2.1使用了从接收方到发送方的肯定确认和否定确认。当接收到失序的分组时，

接收方对所接收的分组发送一个肯定确认。如果收到受损的分组，接收方将发送一个否定确认。如果不发送NAK，而是发送一个对上次正确接收的分组的ACK，我们也能实现与NAK一样的效果。发送方接收到对同一个分组的两个ACK（即接收冗余ACK，duplicate ACK）后，就知道接收方没有正确接收到跟在被确认两次的分组后面的分组。rdt2.2是在具有比特差错信道上实现的一个无NAK的可靠数据传输协议，如图3-13和图3-14所示。rdt2.1和rdt2.2之间的细微变化在于，接收方必须包括由一个ACK报文确认的分组序号（这可以通过在接收方FSM中，包括make_pkt()中的参数ACK，0或ACK，1来实现），发送方必须检查接收到的ACK报文中所确认的分组序号（这可通过在发送方FSM中，包括isACK()中的参数0或1来实现）。

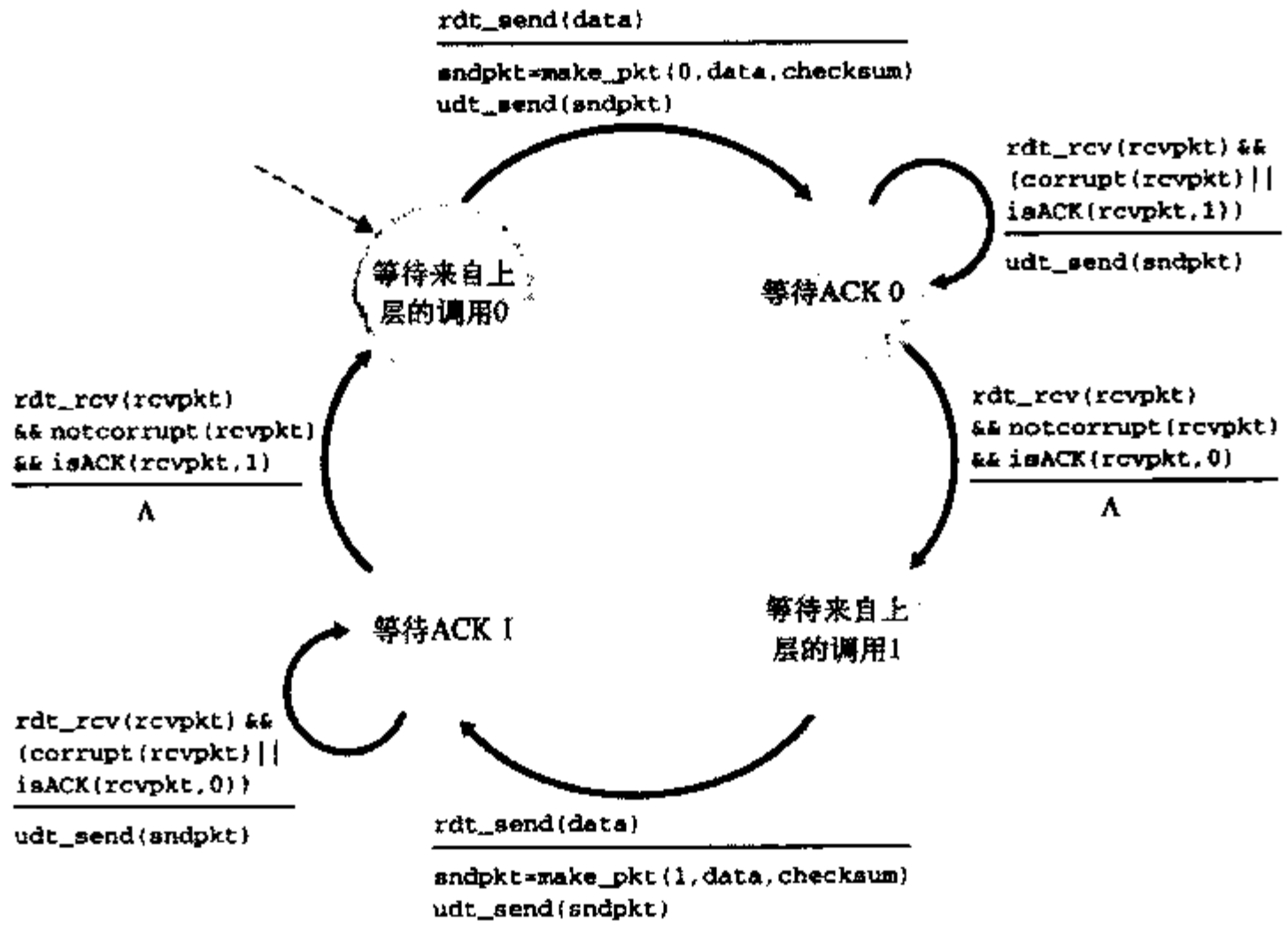


图3-13 rdt2.2发送方

3. 具有比特差错的丢包信道上的可靠数据传输：rdt3.0

现在假定除了比特受损外，底层信道还会丢包，这在今天的计算机网络（包括因特网）中并不罕见。该协议现在必须解决另外两个广泛关注的问题：怎样检测丢包以及发生丢包后该做些什么。利用rdt2.2中已经研发的技术，如检验和、序号、ACK分组和重传等，可以解决后一个问题。为解决广泛关注的前一个问题，还需增加一种新的协议机制。

有很多方法可以解决丢包问题（在本章后面的习题中探讨了几种其他方法）。这里，我们让发送方负责检测和恢复丢包。假定发送方传输一个数据分组，或者该分组或者接收方对该分组的ACK发生了丢失。在这两种情况下，发送方都收不到应当到来的接收方的响应。如果发送方愿意等待足够长的时间以便确定分组已丢失，则只需重传该数据分组即可。你应该相信该协议确实有效。

但是发送方需要等待多久才能确定分组或ACK已丢失了呢？很明显，发送方至少需要等待发送方与接收方之间的一个往返时延（可能会包括在中间路由器的缓冲时延）加上接收方处理一个分组所需的时间。在很多网络中，最坏情况下的最大时延是很难估算的，能确定的

因素非常少。此外，理想的协议应尽可能快地从丢包中理想地恢复出来，而等待一个最坏情况的时延可能意味着要等待一段较长的时间，直到启动差错恢复为止。因此，实际中发送方采取的方法是“明智地”选择一个时间值，以判定虽然不能确保但可能发生了丢包。如果在这个时间值内没有收到ACK，则重传分组。注意到如果一个分组经历了一个特别大的时延，发送方可能会重传该分组，即使该数据分组或ACK都没有丢失。这就在发送方到接收方的信道中引入了**冗余数据分组**（duplicate data packet）的可能性。幸运的是，rdt2.2协议已经有足够的功能（即序号）来处理冗余分组情况。

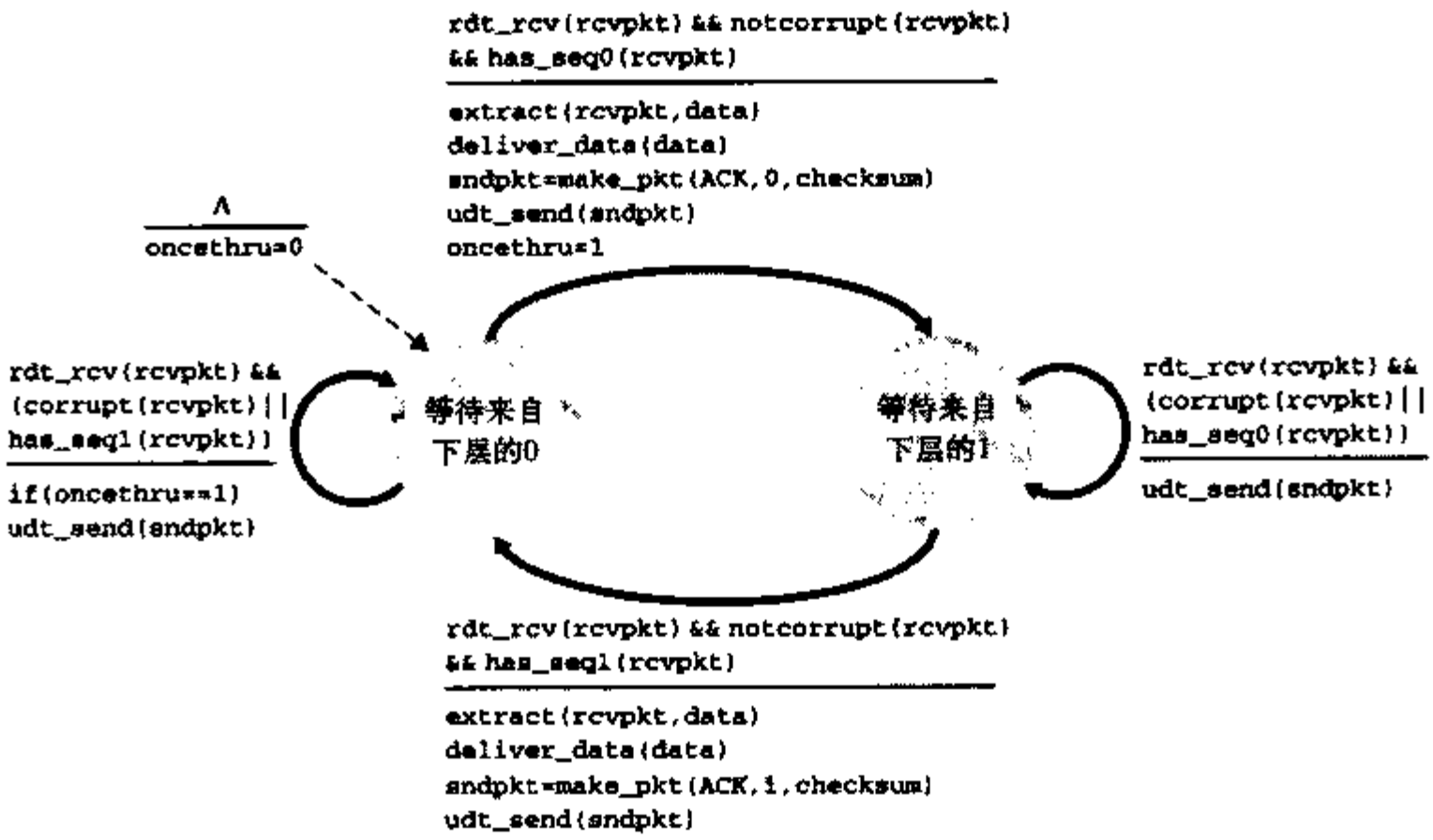


图3-14 rdt2.2接收方

从发送方的观点来看，重传是一种万能灵药。发送方不知道是一个数据分组丢失、一个ACK丢失，还是该分组或ACK只是过度时延。在所有情况下，采取的动作是同样的：重传。为了实现基于时间的重传机制，需要一个**倒计数定时器**（countdown timer），在一个给定的时间过期后，可中断发送方。因此，发送方需要能做到：①每次发送一个分组（即第一次分组和重传分组）时，便启动一个定时器；②响应定时器中断（采取适当的动作）；③终止定时器。

图3-15给出了rdt3.0的发送方FSM，这是一个在可能出错和丢包的信道上可靠传输数据的协议。在课后习题中，将请你提供rdt3.0的接收方FSM。图3-16显示了在没有丢包和分组延迟的情况下，协议是如何运作的以及它是如何处理数据分组丢失的。在图3-16中，时间从图的顶部朝底部移动；注意到一个分组的接收时间肯定迟于一个分组的发送时间（由于发送时延与传播时延之故）。在图3-16b～d中，发送方括弧表明了定时器的设置时刻以及随后的超时。本章后面的习题探讨了该协议几个更细微的方面。因为分组序号在0和1之间交替，因此rdt3.0有时被称为**比特交替协议**（alternating-bit protocol）。

现在我们归纳一下数据传输协议的要点。在检验和、序号、定时器、肯定确认和否定确认分组这些技术中，每种机制都在协议的运行中起到了必不可少的作用。至此，我们得到了一个有效的可靠数据传输协议！

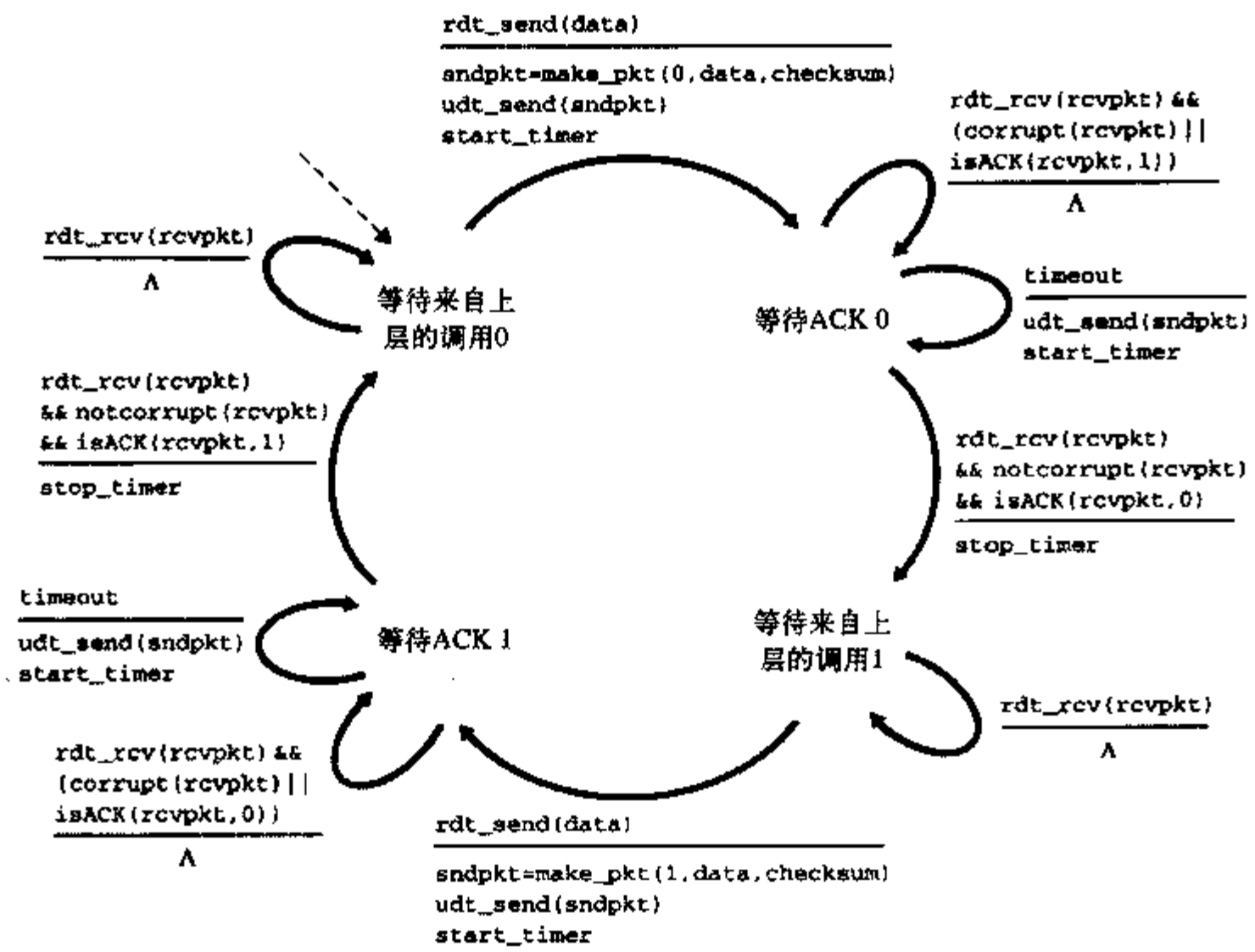


图3-15 rdt3.0发送方

3.4.2 流水线可靠数据传输协议

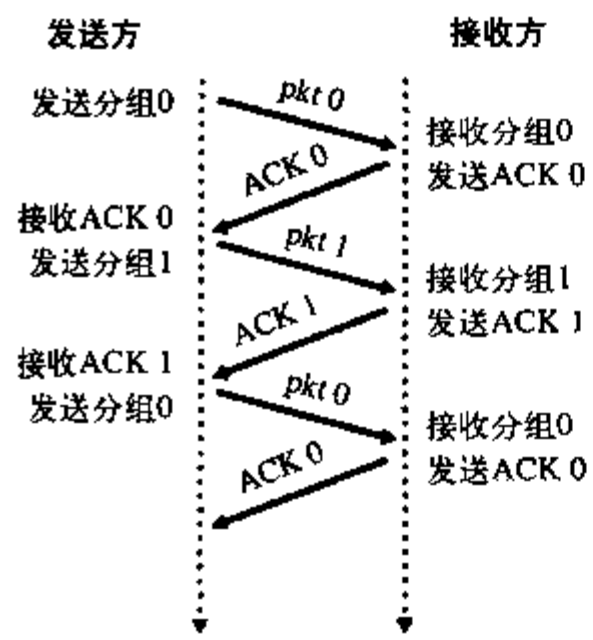
rdt3.0是一个功能正确的协议，但并非人人都满意其性能，特别是在今天的高速网络环境下更是如此。rdt3.0性能问题的核心在于它是一个停等协议。

为了评估停等方法对性能的影响，可考虑一种两台主机的理想化情况，一台位于美国西海岸，另一台位于美国东海岸，如图3-17所示。这两个端系统之间的光速往返传播时延（RTT）大约为30ms。假定彼此通过一条传输速率为$R = 1$ Gbps（10^9 bps）的信道相连。包括首部字段和数据的分组长$L = 1000$字节（8000比特），实际发送一个分组到1 Gbps链路中所需时间是：

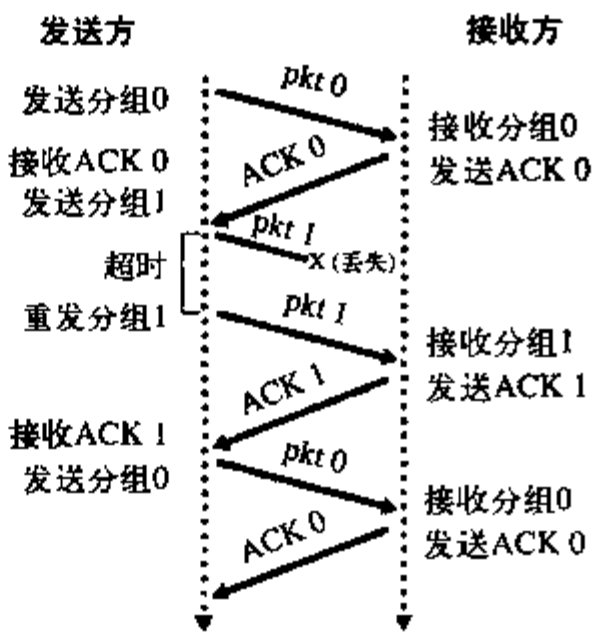
$$t_{\text{trans}} = \frac{L}{R} = \frac{8000 \text{ bit/packet}}{10^9 \text{ bit/s}} = 8 \ \mu\text{s}$$

图3-18a显示了对于停等协议，如果发送方在$t = 0$时刻开始发送分组，则在$t = L/R = 8$ μs后，最后1 比特数据进入发送方信道。该分组经过15 ms跨美国的旅行后到达接收方，该分组的最后1比特在时刻t = RTT/2 + L/R = 15.008 ms时到达接收方。为了简化起见，假设ACK分组很小（以便我们可以忽略其发送时间），接收方收到一个数据分组的最后1 比特后立即发送ACK，ACK在时刻t = RTT+L/R = 30.008 ms时到达发送方。此时，发送方可以发送下一个报文。因此，在30.008 ms内，发送方的发送只用了0.008 ms。定义发送方（或信道）的**利用率**（utilization）为：发送方实际忙着将发送比特送进信道的那部分时间与发送时间之比。图3-18a中的分析表明了停等协议有着非常低的发送方利用率U_{sender}：

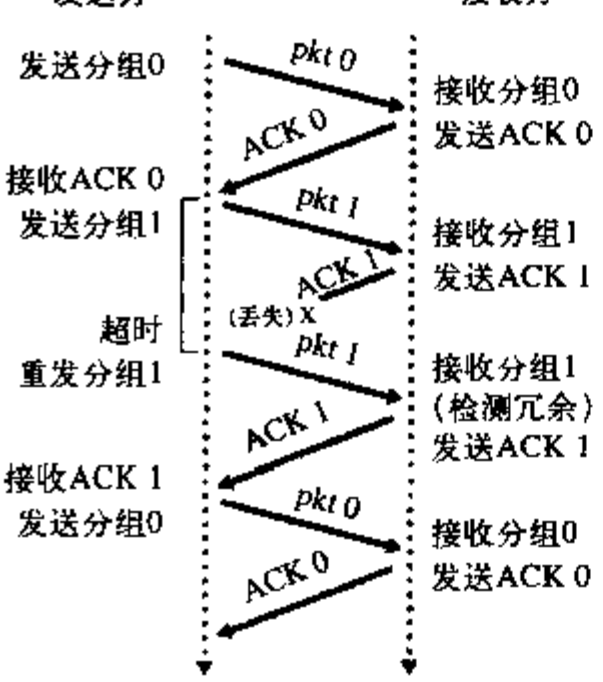
$$U_{\text{sender}} = \frac{L/R}{\text{RTT} + L/R} = \frac{0.008}{30.008} = 0.000\ 27$$



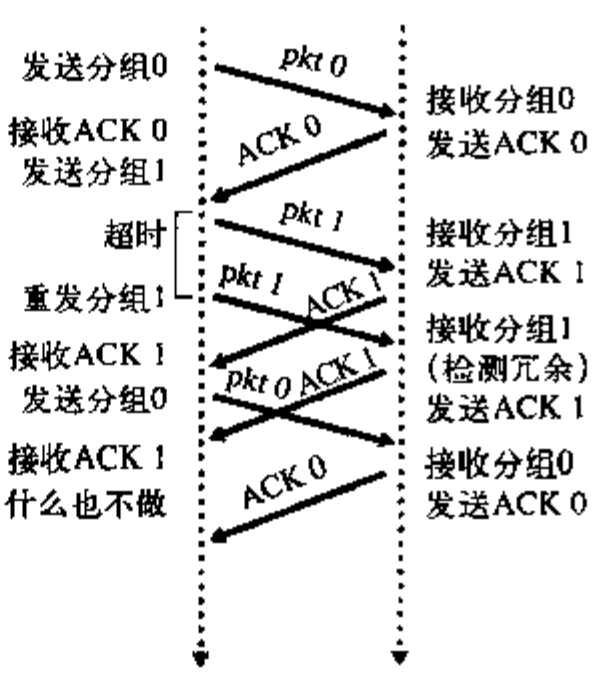
a) 无丢包操作



b) 分组丢失

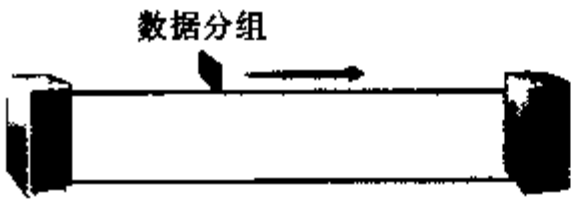


c) 丢失ACK



d) 过早超时

图3-16 rdt3.0的运行，比特交替协议



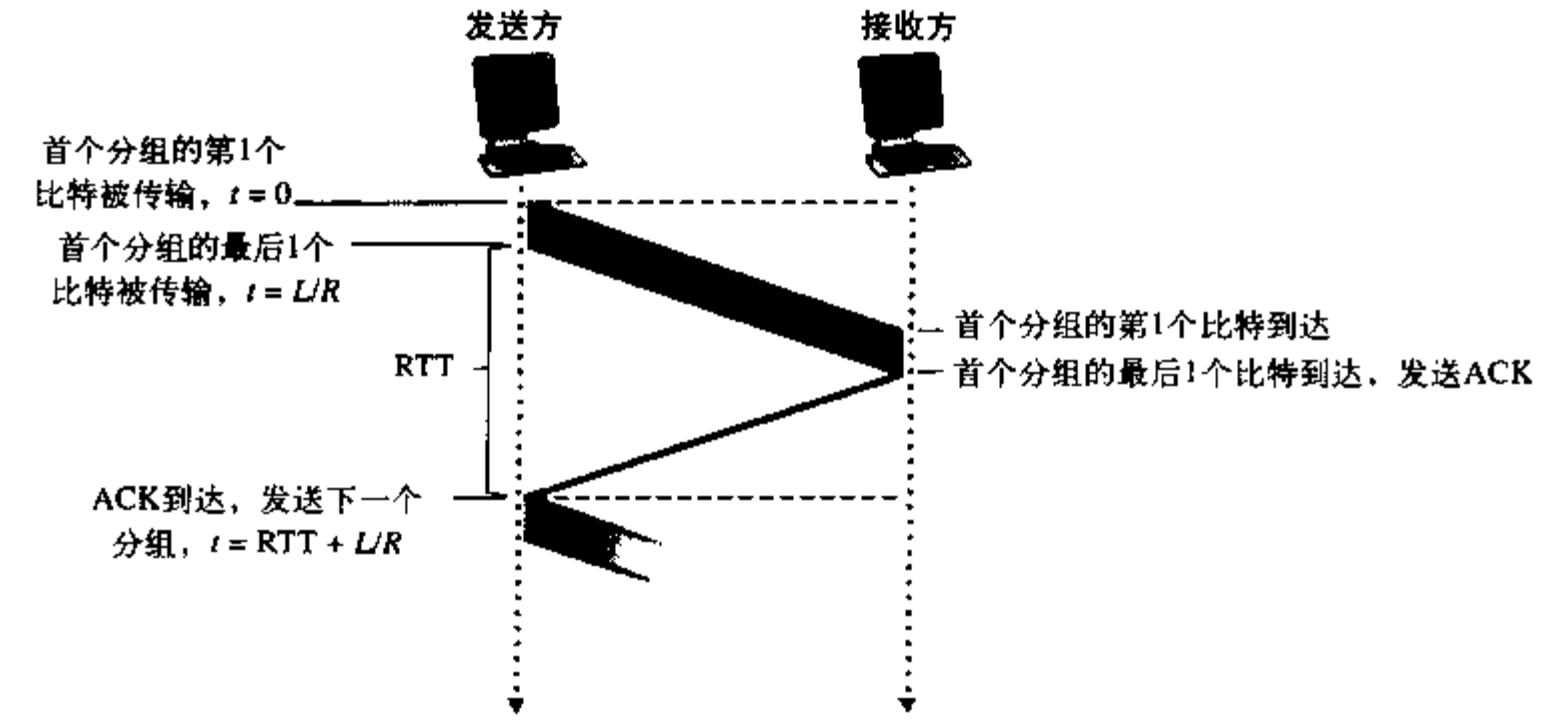
a) 一个运行中的停等协议

b) 一个运行中的流水线协议

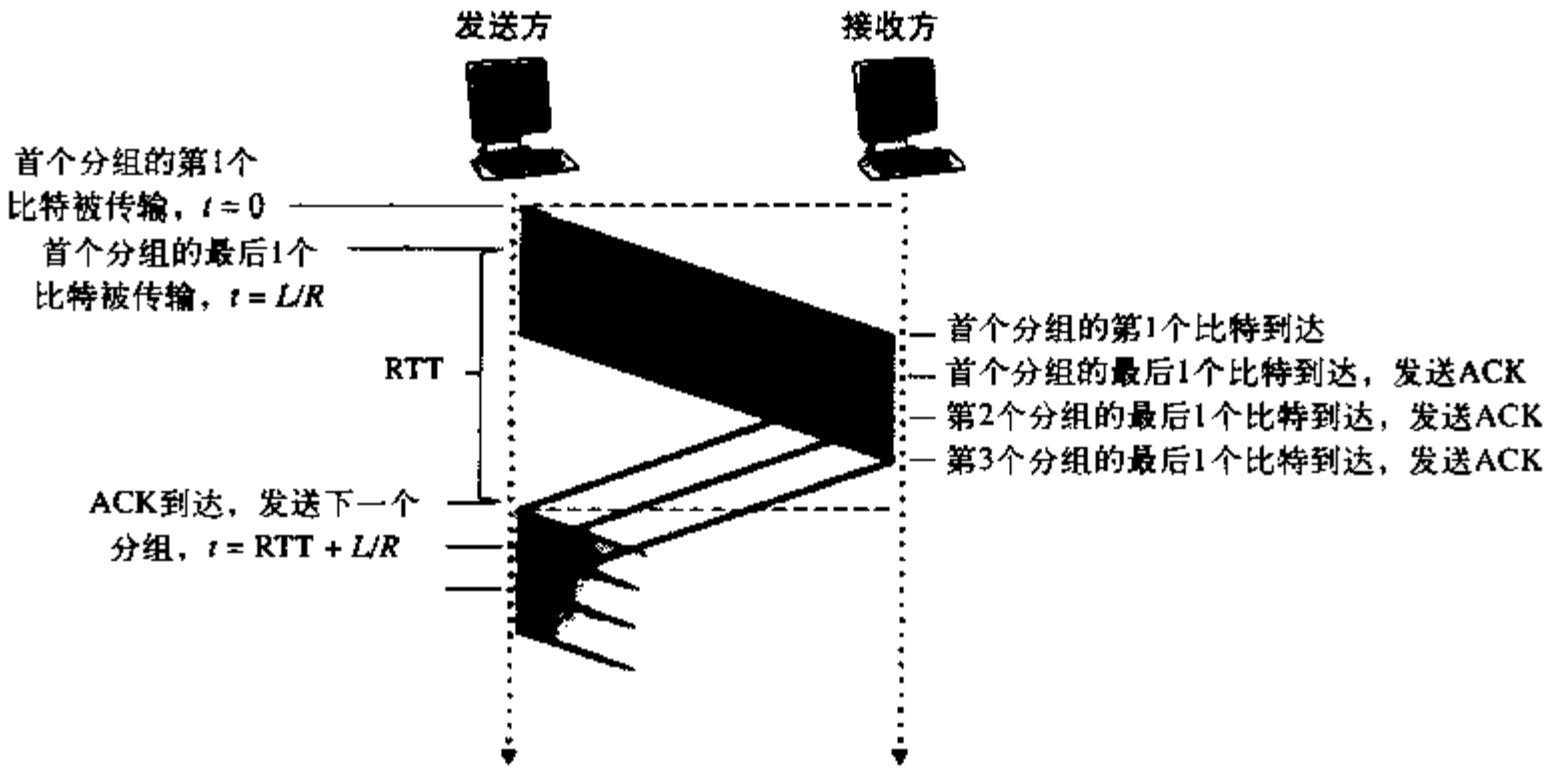
图3-17 停等协议与流水线协议

也就是说，发送方只有万分之2.7的时间是忙的。从其他角度来看，发送方在30.008 ms内只能发送1000字节，有效的吞吐量仅为267 kbps。(即使有1 Gbps的链路可用！）想象一个不幸的网络经理购买了一条千兆比容量的链路，但他仅能得到267 kbps吞吐量的情况！这是一个网络协议限制底层网络硬件所提供的功能的形象示例。而且，我们还忽略了在发送方和接收

方的底层协议处理时间，以及可能出现在发送方与接收方之间的任何中间路由器上的处理与排队时延。再考虑到这些因素，将进一步增加时延，使其性能更糟糕。



a) 停等操作



b) 流水线操作

图3-18 停等操作和流水线操作

解决这种特殊的性能问题的一个简单方法是：不使用停等方式运行，允许发送方发送多个分组而无需等待确认，图3-17b举例说明了这种情况。图3-18b显示了如果发送方可以在等待确认之前发送3个分组，其利用率也将是原来的3倍。因为从发送方向接收方传输的众多分组可以被看成是填充到一条流水线中，故这种技术被称为**流水线**（pipelining）。流水线技术可对可靠数据传输协议带来如下影响：

- 必须增加序号范围，因为每个传输的分组（不计算重传的）必须有一个唯一的序号，而且也许有多个在传输中的未确认的分组。
- 协议的发送方和接收方也许必须缓存多个分组。发送方最低限度应当能缓冲那些已发送

但没有确认的分组。如下面讨论的那样，接收方或许也需要缓存那些已正确接收的分组。

- 所需序号范围和对缓冲的要求取决于数据传输协议处理丢失、损坏及过度延时分组的方式。解决流水线的差错恢复有两种基本方法：回退N步（Go-Back-N）和选择重传（selective repeat）。

3.4.3 回退N步

在**回退N步**（Go-Back-N，GBN）协议中，允许发送方发送多个分组（当有多个分组可用时）而不需等待确认，但它也受限于在流水线中未确认的分组数不能超过某个最大允许数N。在本节中我们较为详细地描述GBN。但在继续阅读之前，建议你操作本书配套Web网站上的GBN Java小程序（这是一个非常好的Java程序）。

图3-19显示了发送方看到的GBN协议的序号范围。如果我们将**基序号**（base）定义为最早的未确认分组的序号，将**下一个序号**（nextseqnum）定义为最小的未使用序号（即下一个待发送分组的序号），则可将序号范围分割成4部分。[0,base-1]内的序号对应于已经发送并确认过的分组。[base,nextseqnum-1]内的序号对应已经发送但未被确认的分组。[nextseqnum,base+N-1]内的序号可用于那些要被立即发送的分组，其数据来自上层。最后，大于或等于base+N的序号是不能使用的，直到当前流水线中未被确认的分组（特别是序号为base的分组）已得到确认为止。

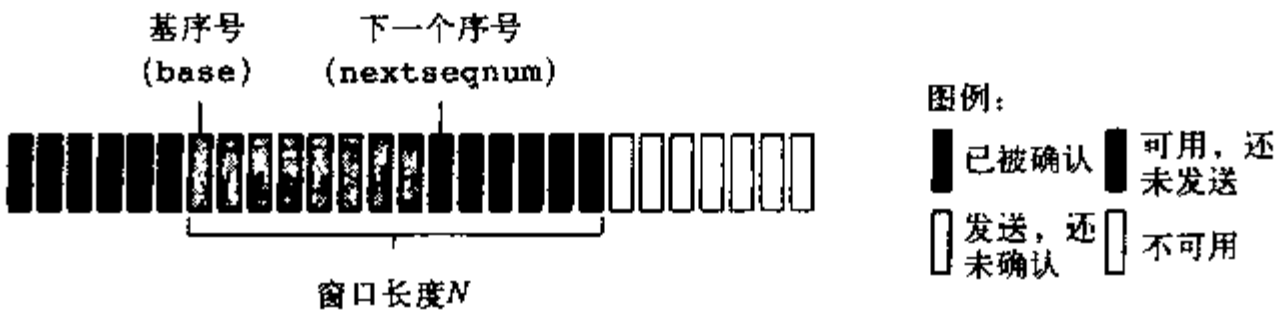


图3-19 在GBN中发送方看到的序号

如图3-19所提示的那样，那些已被发送但还未被确认的分组的许可序号范围可以被看成是一个在序号范围内长度为N的窗口。随着协议的运行，该窗口在序号空间内向前滑动。因此，N常被称为**窗口长度**（window size），GBN协议常被称为**滑动窗口协议**（sliding-window protocol）。你也许会想，为什么先要限制这些被发送的、未被确认的分组的数目为N呢？为什么不允许这些分组的数目为无限制呢？我们将在3.5节看到，流量控制是对发送方施加限制的原因之一。我们将在3.7节学习TCP拥塞控制时分析另一个原因。

在实际中，一个分组的序号承载在分组首部的一个固定长度的字段中。如果分组序号字段的比特数是k，则该序号范围是[0，2^k-1]。在一个有限的序号范围内，所有涉及序号的运算必须使用模2^k运算（即序号空间可被认为是一个长度为2^k的环，其中序号2^k-1紧接序号0）。前面讲过，rdt3.0有一个1 比特的序号，序号范围是[0, 1]。本章后面的几道习题探讨了一个有限序号范围所产生的结果。我们将在3.5节看到，TCP有一个32 比特的序号字段，其中的TCP序号是按字节流中的字节计数的而不是按分组计数的。

图3-20和图3-21给出了一个基于ACK、无NAK的GBN协议的发送方和接收方的扩展FSM描述。之所以称该FSM描述为扩展FSM描述，是因为增加了变量（类似于编程语言中的变量）base和nextseqnum，还增加了对这些变量的操作以及与这些变量有关的条件动作。注意到扩展FSM描述现在变得有点像编程语言描述。[Bochman 1984]对FSM扩展技术提供了一个很

好的综述，也提供了用于定义协议的其他基于编程语言的技术。

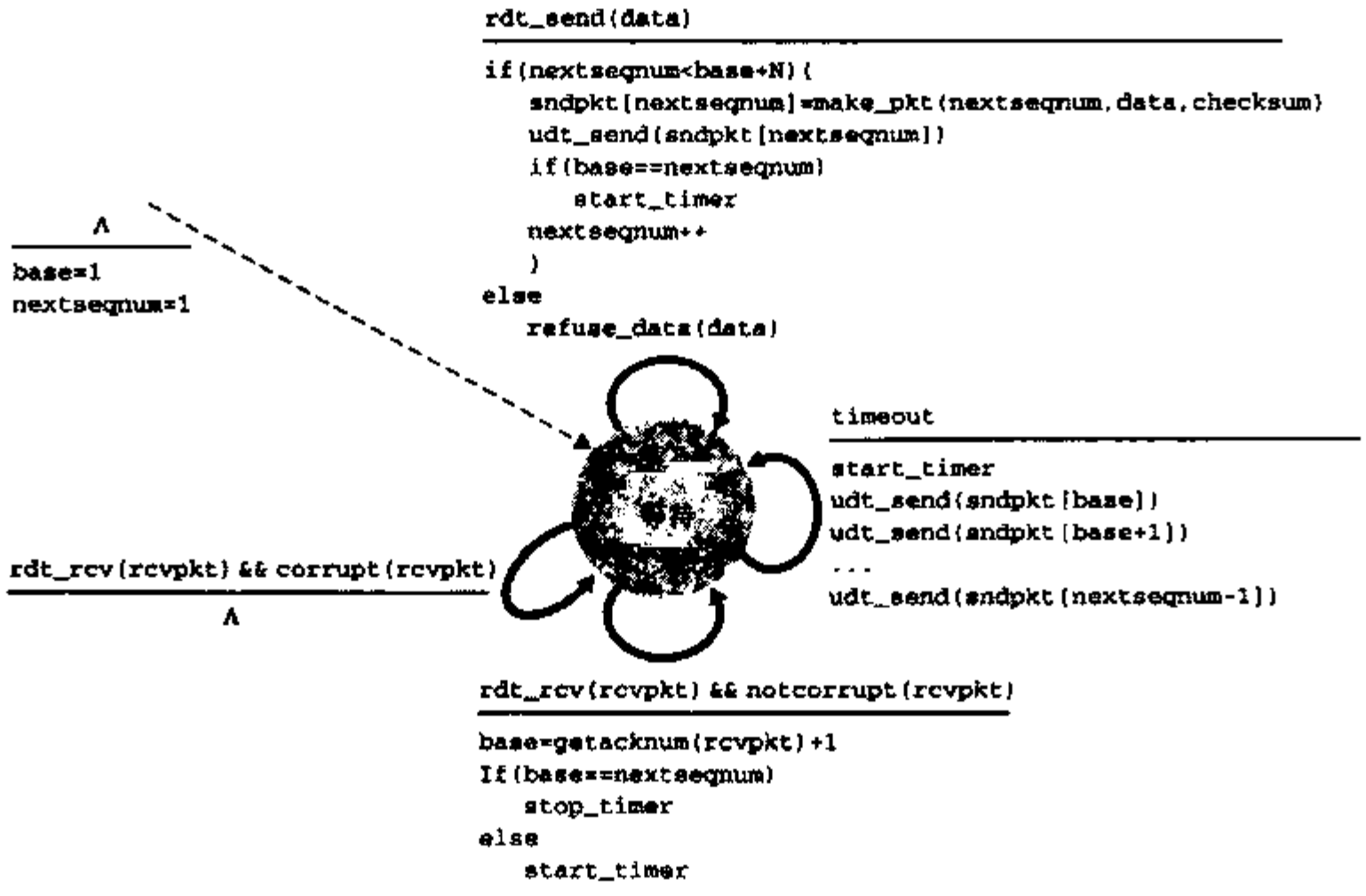


图3-20 GBN发送方的扩展FSM描述

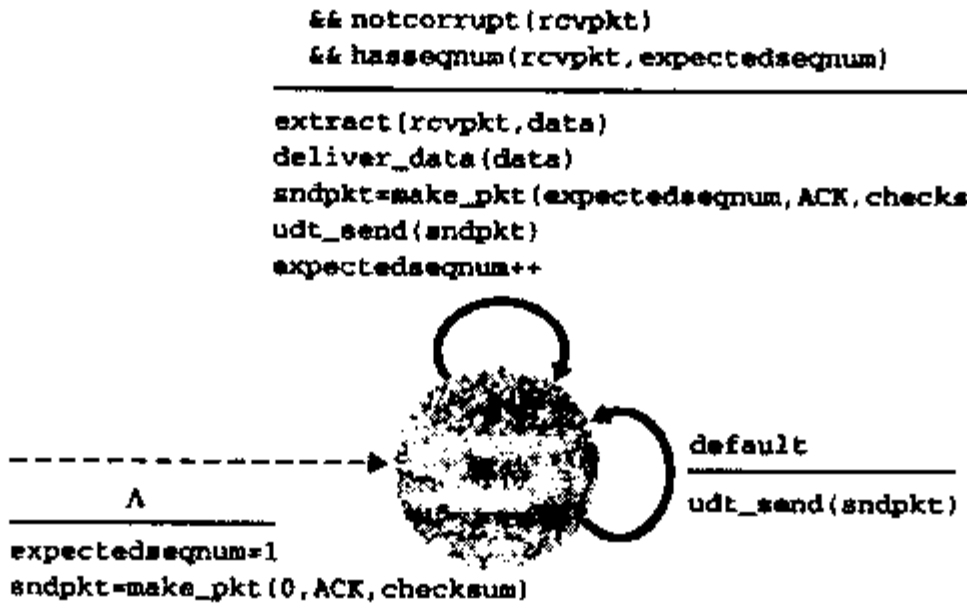


图3-21 GBN接收方的扩展FSM描述

GBN发送方必须响应以下三种类型的事件：

- 上层的调用。当上层调用rdt_send()时，发送方首先检查发送窗口是否已满，即是否有N个已发送但未被确认的分组。如果窗口未满，则创建一个分组并将其发送，变量也相应地更新。如果窗口已满，发送方只需将数据返回给上层，隐式地通知上层该窗口已满。然后，上层可能会过一会儿再试。在实际实现时，发送方更可能缓存（并不立刻发送）这些数据，或者使用同步机制（如一个信号量或标志）允许上层在仅当窗口不满时才调用rdt_send()。
- 收到ACK。在GBN协议中，对序号为n的分组的确认采取**累积确认**（cumulative acknowledgment）的方式，表明接收方已正确接收到序号n以前（包括n在内）的所有分

组。稍后讨论GBN接收方一端时，我们将再次研究这个主题。

- 超时事件。协议的名字“Go-Back-N”是根据出现丢失和过度时延分组时发送方的行为而得出的。就像在停等协议中那样，定时器将再次用于恢复丢失的数据或确认分组。如果出现超时，发送方将重传所有已发送但还未被确认的分组。图3-20中的发送方仅使用一个定时器，它可被当作是最早的已发送但未被确认的分组所使用的定时器。如果收到一个ACK，但仍有已发送但未被确认的分组，则定时器被重新启动。如果没有已发送但未被确认的分组，则该定时器被终止。

在GBN中，接收方的动作也很简单。如果一个序号为n的分组被正确接收到，并且按序（即上次交付给上层的数据是序号为$n-1$的分组），则接收方为分组n发送一个ACK，并将该分组中的数据交付到上层。在所有其他情况下，接收方都丢弃该分组，并为最近按序接收的分组重传ACK。注意到因为一次交付给上层一个分组，如果分组k已接收并交付，则所有比序号k小的分组也已经交付。因此，使用累积确认是GBN的一个很自然的选择。

在GBN协议中，接收方丢弃所有失序分组。尽管丢弃一个正确接收（但失序）的分组有点愚蠢和浪费，但这样做是有道理的。前面讲过，接收方必须按序将数据交付给上层。假定现在期望接收分组n，而分组$n+1$却到了。因为数据必须按序交付，接收方可能缓存（保存）分组$n+1$，然后，在它收到并交付分组n后，再将该分组交付到上层。然而，如果分组n丢失，则该分组及分组$n+1$最终将在发送方根据GBN重传规则而被重传。因此，接收方只需丢弃分组$n+1$即可。这种方法的优点是接收方缓存简单，即接收方不需要缓存任何失序分组。因此，虽然发送方必须维护窗口的上下边界及nextseqnum在该窗口中的位置，但是接收方需要维护的唯一信息就是下一个按序接收的分组的序号。该值保存在变量expectedseqnum中，如图3-21中接收方FSM所示。当然，丢弃一个正确接收分组的缺点是随后对该分组的重传也许会丢失或出错，因此甚至需要更多的重传。

图3-22给出了窗口长度为4个分组的GBN协议的运行情况。因为该窗口长度限制，发送方发送分组0至3，然后在继续发送之前，必须等待直到一个或多个分组被确认。当接收到每一个连续的ACK（例如ACK 0和ACK 1）时，该窗口向前滑动，发送方便可以发送新的分组（分别是分组4和分组5）。在接收方，分组2丢失，因此分组3、4和5被发现是失序分组并被丢弃。

在结束对GBN的讨论之前，需要提请读者注意的是，在协议栈中实现该协议可能有与图3-20中的扩展FSM相似的结构。该实现也可能是以各种过程形式出现，每个过程实现了在响应各种可能出现的事件时要采取的动作。在这种**基于事件的编程**（event-based programming）方式中，这些过程要么被协议栈中的其他过程调用，要么作为一次中断的结果。在发送方，这些事件包括：①上层实体调用rdt_send()，②定时器中断，③报文到达时，底层调用rdt_rcv()。本章后面的编程练习会使你有机会在一个模拟而真实的网络环境中实际实现这些例程。

这里我们注意到，GBN协议中综合了我们将在3.5节中学习TCP可靠数据传输构件时遇到的所有技术。这些技术包括使用序号、累积确认、检验和以及超时/重传操作。

3.4.4 选择重传

在图3-17中，GBN协议允许发送方用多个分组“填充流水线”，因此避免了停等协议中所提到的信道利用率问题。然而，GBN本身也存在着性能问题。尤其是当窗口长度和带宽时延积都很大，在流水线中会有很多分组时更是如此。一个单个分组的差错就可能引起GBN重传大量分组，许多分组根本没有必要重传。随着信道差错率的增加，流水线可能会被这些没必要重传的分组填满。想象一下，在我们口述消息的情况下，如果每次有一个单词含糊不清，其前后1000个单词（例如，窗口长度为1000个单词）不得不被重传的情形。此次口述会由于这些反复述说的单词而变慢。

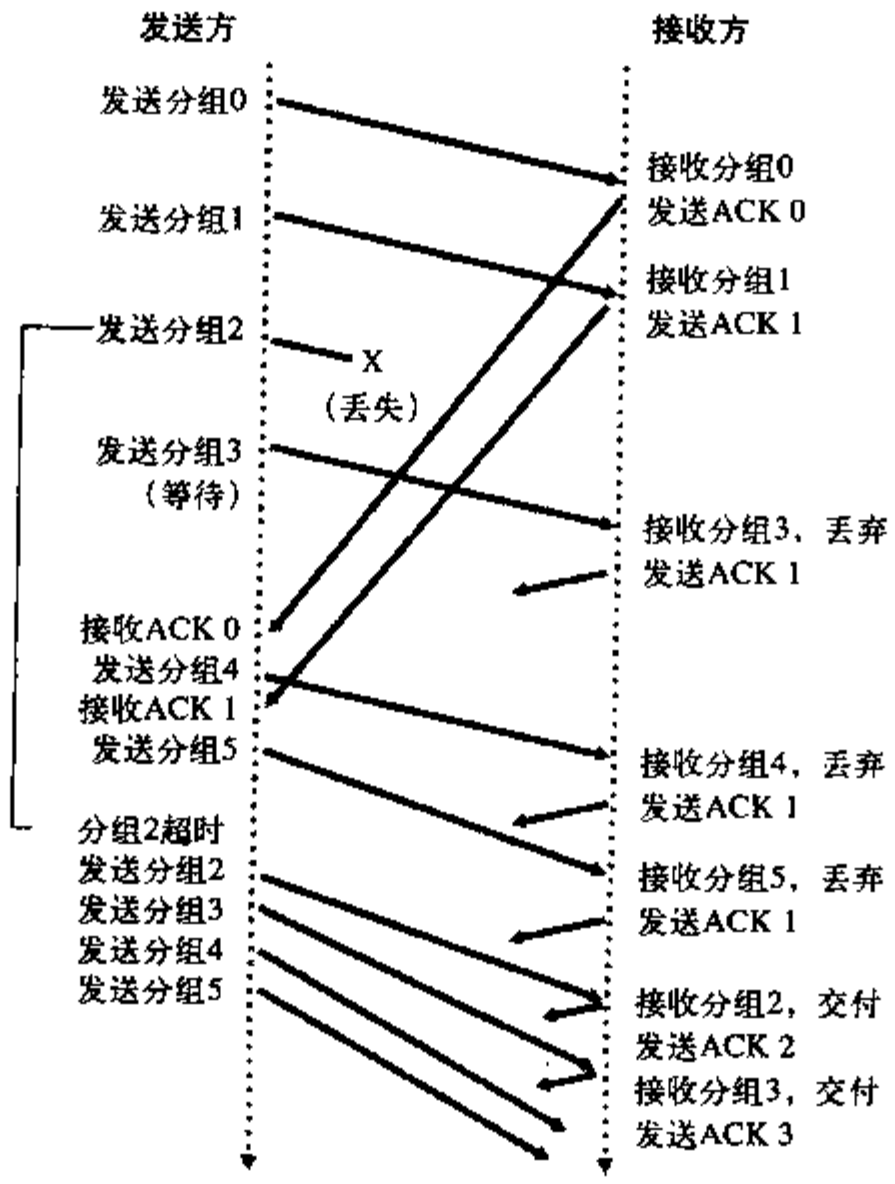


图3-22 运行中的GBN

顾名思义，**选择重传**（SR）协议通过让发送方仅重传那些它怀疑在接收方出错（即丢失或受损）的分组而避免了不必要的重传。这种个别的、按需的重传要求接收方逐个地确认正确接收的分组。再次用窗口长度*N*来限制流水线中未完成、未被确认的分组数。然而，与GBN不同的是，发送方已经收到了对窗口中某些分组的ACK。图3-23显示了SR发送方看到的

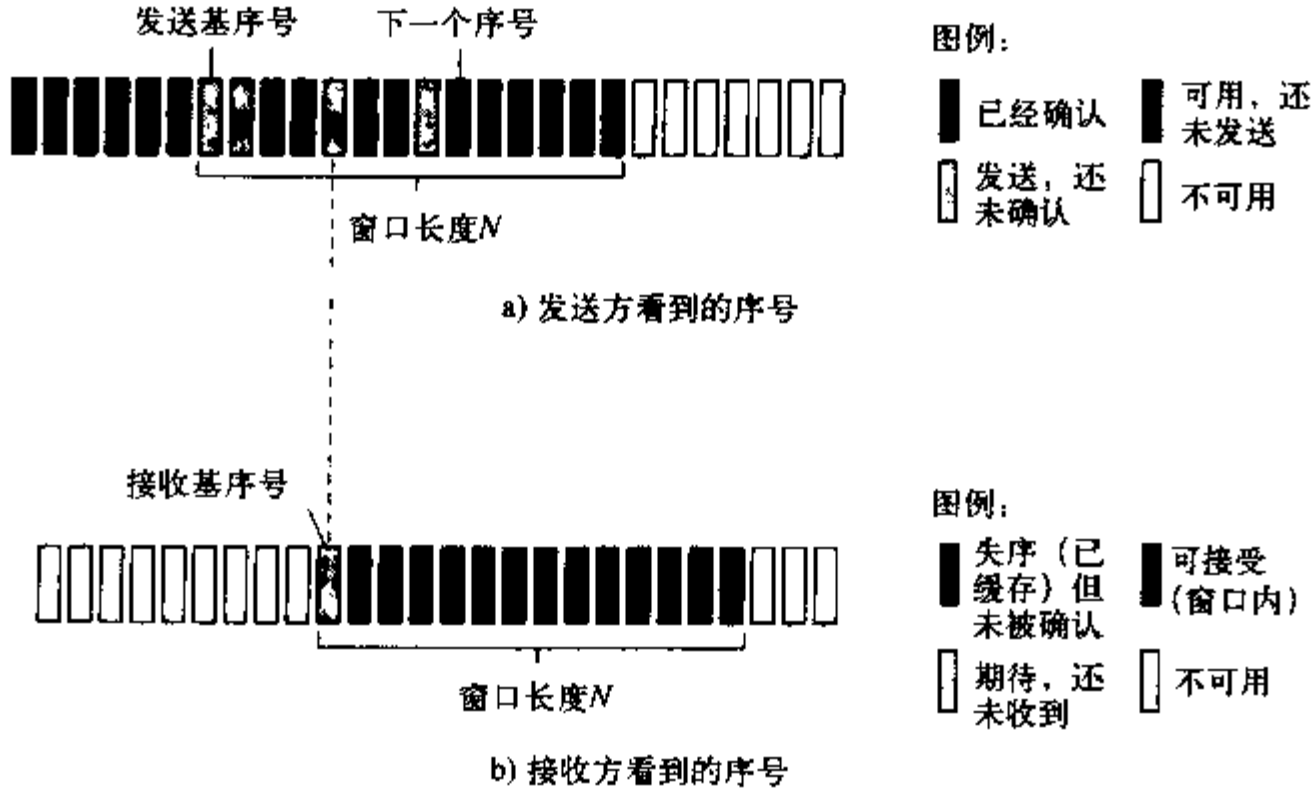


图3-23 选择重传（SR）发送方与接收方的序号空间

序号空间。图3-24详细描述了SR发送方所采取的各种动作。

1. 从上层收到数据。当从上层接收到数据后，SR发送方检查下一个可用于该分组的序号。如果序号在发送方的窗口内，则将数据打包并发送；否则就像在GBN中一样，要么将数据缓存，要么将其返回给上层以便以后传输。

2. 超时。定时器再次被用来防止丢失分组。然而，现在每个分组必须拥有自己的逻辑定时器，因为超时后只能发送一个分组。可以使用单个硬件定时器模拟多个逻辑定时器的操作 [Varghese 1997]。

3. 收到ACK。如果收到ACK，且该分组序号在窗口内，则SR发送方将那个被确认的分组标记为已接收。如果该分组的序号等于send_base，则窗口基序号向前移动到具有最小序号的未确认分组处。如果窗口移动了并且有序号落在窗口内的未发送分组，则发送这些分组。

图3-24 SR发送方的事件与动作

SR接收方将确认一个正确接收的分组而不管其是否按序。失序的分组将被缓存直到所有丢失分组（即序号更小的分组）都被收到，这时才可以将一批分组按序交付给上层。图3-25详细列出了SR接收方所采取的各种动作。图3-26举例说明了存在丢包时SR的操作。注意到图3-26中接收方初始时缓存了分组3、4和5，并在最终收到分组2时，才将它们一并交付给上层。

1. 序号在[rcv_base, rcv_base+N-1]内的分组被正确接收。在此情况下，收到的分组落在接收方的窗口内，一个选择ACK分组被回送给发送方。如果该分组是以前没收到的分组，则被缓存。如果该分组的序号等于接收窗口的基序号（图3-22中的rcv_base），则该分组以及以前缓存的序号连续的分组（起始于rcv_base）交付给上层。然后，接收窗口按向上交付的分组的数量向前移动。举一个例子，考虑一下图3-26。当收到一个序号为rcv_base =2的分组时，该分组与分组3、4、5可被交付给上层。

2. 序号在[rcv_base-N, rcv_base-1]内的分组被正确收到。在此情况下，必须产生一个ACK，即使该分组是接收方以前已确认过的分组。

3. 其他情况。忽略该分组。

图3-25 SR接收方的事件与动作

注意到以下问题很重要，在图3-25中的第2步，接收方重新确认（而不是忽略）已收到过的那些序号小于当前窗口基序号的分组。你应该理解这种重新确认确实是需要的。例如，给出图3-23中所示的发送方和接收方的序号空间，如果没有分组send_base的ACK从接收方传播回发送方，则发送方最终将重传分组send_base，即使显然（对我们而不是对发送方来说）接收方已经收到了该分组。如果接收方不确认该分组，则发送方窗口将永远不能向前滑动！这个例子说明了SR协议（和很多其他协议一样）的一个重要方面。对于哪些分组已经被正确接收、哪些没有，发送方和接收方并不总是能看到相同的结果。对SR协议而言，这就意味着发送方和接收方的窗口并不总是一致。

当我们面对有限序号范围的现实时，发送方和接收方窗口间的不同步会产生严重的后果。考虑下面例子中的情况，对于一个有4个分组序号0、1、2、3的有限范围且窗口长度为3，假定发送了分组0至2，接收方也正确接收且确认了它们。此时，接收方窗口落在第4、5、6个分组上，其序号分别为3、0、1。现在考虑两种情况。第一种情况如图3-27a所示，对前3个分组的ACK丢失，发送方要重传这些分组。因此接收方下一步要接收序号为0的分组，即第一个发送分组的拷贝。

第二种情况如图3-27b所示，对前3个分组的ACK都被正确交付。因此发送方向前移动窗口并发送第4、5、6个分组，其序号分别为3、0、1。序号为3的分组丢失，但序号为0的分组到达（一个包含新数据的分组）。

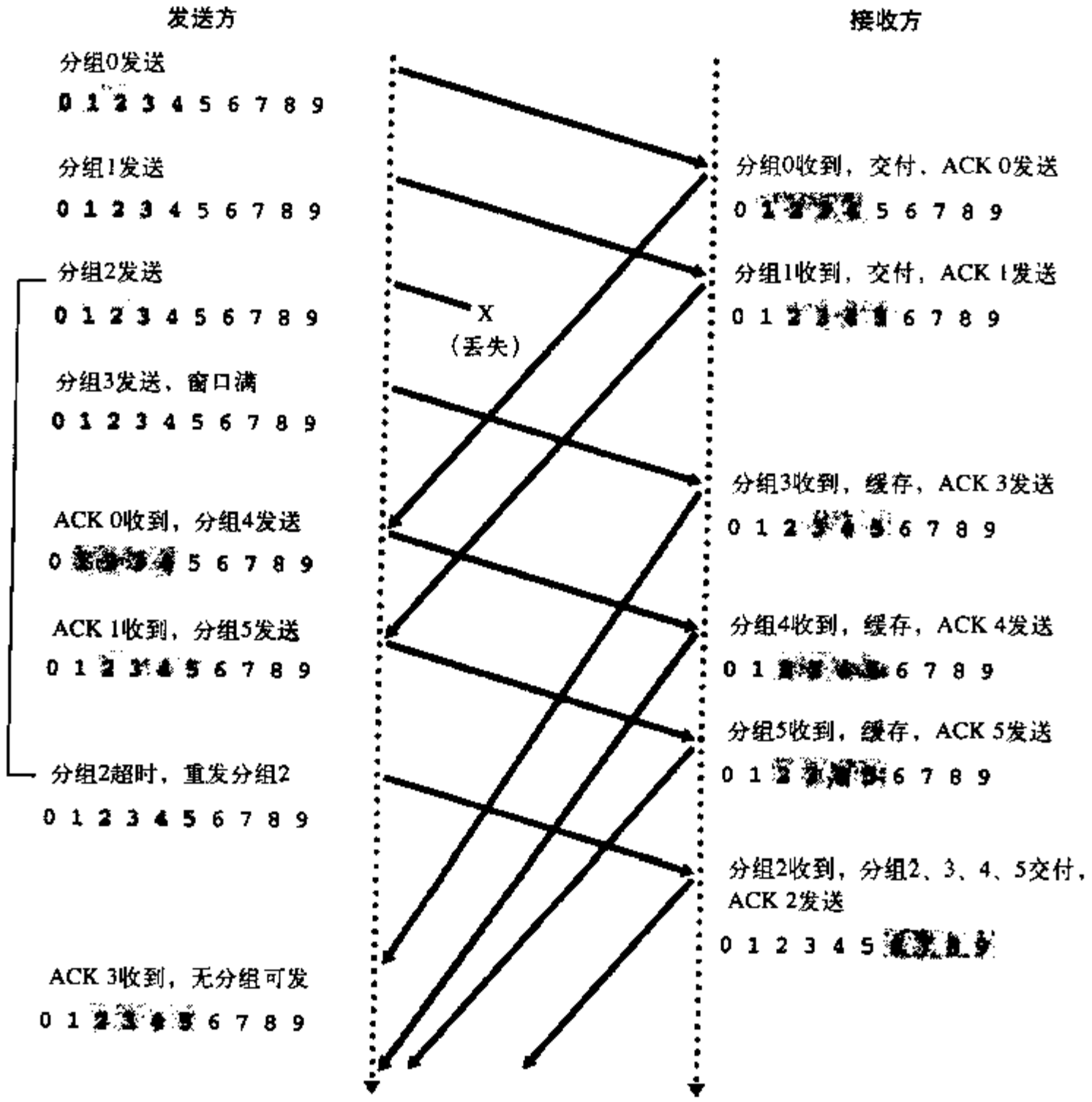
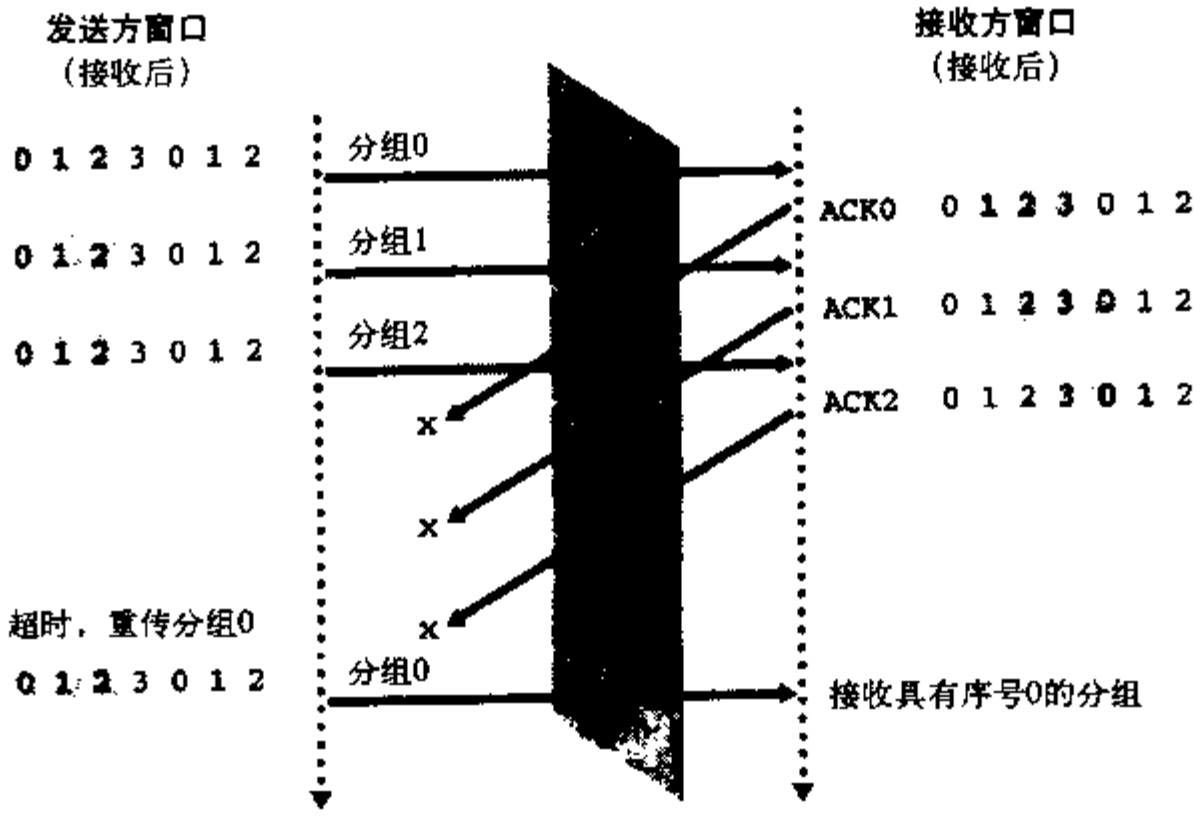


图3-26 SR操作

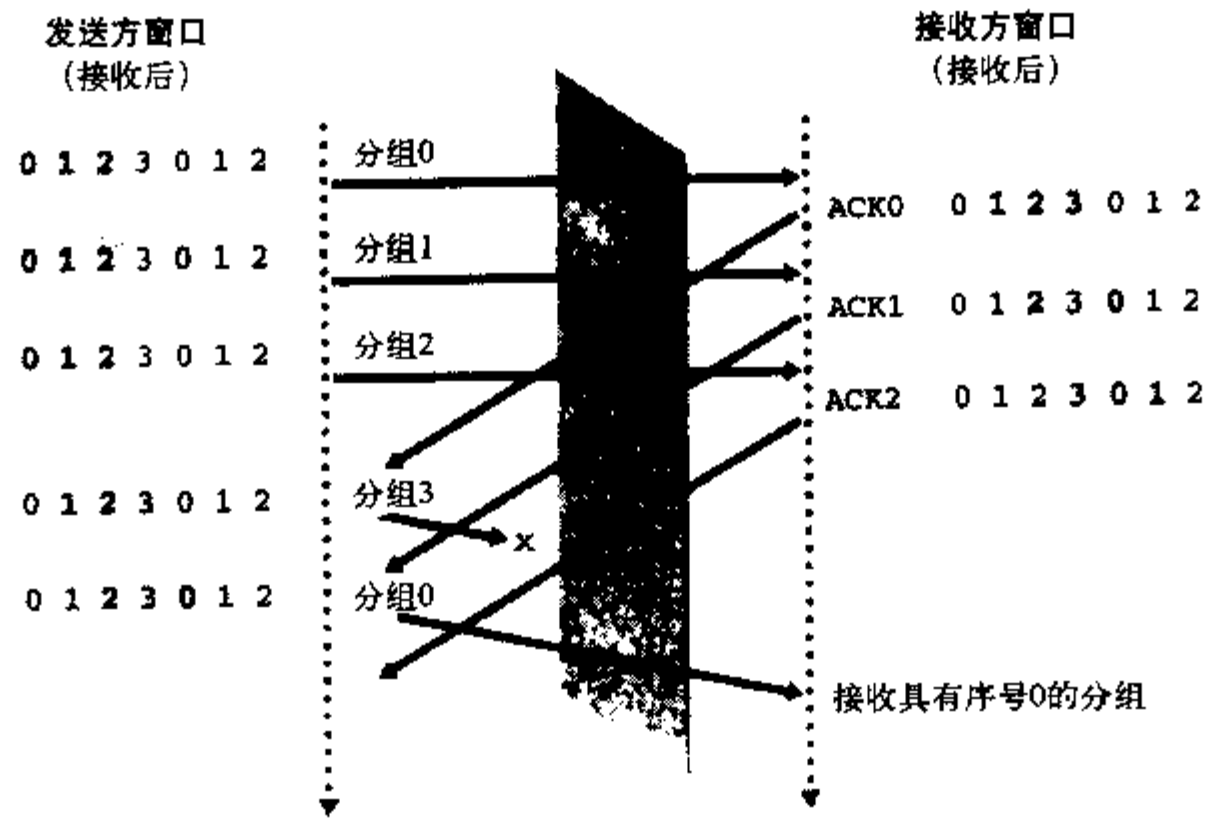
现在考虑一下图3-27中接收方的观点，在发送方和接收方之间有一个假想的帘子，因为接收方不能“看见”发送方采取的动作。接收方所能观察到的是它从信道中收到的以及它向信道中发出的报文序列。就其所关注的而言，图3-27中的两种情况是等同的。没有办法区分是第一个分组的重传还是第五个分组的初次传输。显然，窗口长度比序号空间大小小1时协议无法工作。但窗口必须多小呢？本章后面的一个习题请你说明为何对于SR协议而言，窗口长度必须小于或等于序号空间大小的一半。

在本书配套网站上，可以找到一个模仿SR协议运行的Java小程序。尝试执行与前面的GBN Java小程序相同的实验。结果与你期望的一致吗？

至此，我们结束了对可靠数据传输协议的讨论。我们已涉及了许多主题，并介绍了多种用于提供可靠数据传输的机制。表3-1对这些机制进行了总结。既然我们已经了解了所有这些运行中的机制，并能看得更加全面，我们建议你再复习一遍本节内容，看看这些机制是怎样逐步被添加进来，以涵盖复杂性渐增的（现实的）连接发送方与接收方的各种信道模型的，或者看看这些机制是如何被逐步加入以改善协议性能的。



a)



b)

图3-27 SR接收方在窗口太大时的困境：是一个新分组还是一次重传

表3-1 可靠数据传输机制及其用途的总结

机 制	用途和说明
检验和	用于检测在一个传输分组中的比特错误
定时器	用于检测超时/重传一个分组，可能因为该分组（或其ACK）在信道中丢失了。由于当一个分组延时但未丢失（过早超时），或当一个分组已被接收方收到但从接收方到发送方的ACK丢失时，可能产生超时事件，所以接收方可能会收到一个分组的多个冗余拷贝
序号	用于为从发送方流向接收方的数据分组按顺序编号。所接收分组的序号间的空隙可使该接收方检测出丢失的分组。具有相同序号的分组可使接收方检测出一个分组的冗余拷贝

（续）

机 制	用途和说明
确认	接收方用于告知发送方一个分组或一组分组已被正确地接收到了。确认报文通常携带着被确认的分组或多个分组的序号。确认可以是逐个的或累积的，这取决于协议
否定确认	接收方用于告知发送方某个分组未被正确地接收。否定确认报文通常携带着未被正确接收的分组的序号
窗口、流水线	发送方也许被限制仅发送那些序号落在一个指定范围内的分组。通过允许一次发送多个分组但未被确认，发送方的利用率可在停等操作模式的基础上得到增加。我们很快将会看到，窗口长度可根据接收方接收和缓存报文的能力或网络中的拥塞程度这两种情况之一或全部来进行设置

我们通过考虑在基本信道模型中的一个遗留假设来结束对可靠数据传输协议的讨论。前面曾假定分组在发送方与接收方之间的信道中不能被重新排序。这在发送方与接收方由单段物理线路相连的情况下，通常是一个合理的假设。然而，当连接两端的“信道”是一个网络时，分组重新排序是可能会发生的。分组重新排序的一个表现就是，可能会出现一个具有序号或确认号x的分组的旧拷贝，即使发送方或接收方的窗口中都没有包含x。对于分组重排序，信道可被看成基本上是在缓存分组，并在将来任意时刻自然地释放出这些分组。由于序号可以重新使用，所以必须小心，以免出现这样的冗余分组。实际应用中采用的方法是，确保一个序号不被重新使用，直到发送方“确信”任何先前发送的序号为x的分组都不再在网络中为止。可以通过假定一个分组在网络中“存活”的时间不会超过某个固定最长时间来实现这种方法。在高速网络的TCP扩展中，最长的分组寿命被假定为大约3分钟［RFC 1323］。［Sunshine 1978］描述了一种使用序号的方法，它可使重新排序问题完全避免。

3.5 面向连接的运输：TCP

既然我们已经学习了可靠数据传输的基本原理，我们就可转而学习TCP了。该协议是因特网的运输层、面向连接的可靠运输协议。我们在本节中将看到，为了提供可靠数据传输，TCP依赖于前一节所讨论的许多基本原理，其中包括差错检测、重传、累积确认、定时器以及用于序号和确认号的首部字段。TCP在RFC 793、RFC 1122、RFC 1323、RFC 2018以及RFC 2581中定义。

3.5.1 TCP连接

TCP是**面向连接**（connection-oriented）的，这是因为在一个应用进程可以开始向另一个应用进程发送数据之前，这两个进程必须先相互“握手”，即它们必须相互发送某些预备报文段，以建立确保数据传输所需的参数。作为TCP连接建立的一部分，连接的双方都将初始化与TCP连接相关的许多TCP状态变量（其中的许多状态变量将在本节和3.7节中讨论）。

这种TCP“连接”既不是一条如电路交换网络中的端到端TDM或FDM电路，也不是一条虚电路（参见第1章），因为其连接状态完全保留在两个端系统中。由于TCP协议只在端系统中运行，而不在中间的网络元素（路由器和链路层交换机）中运行，所以中间要素不会维持TCP连接状态。事实上，中间路由器对TCP连接完全不知情，它们看到的是数据报，而不是连接。

TCP连接提供的是**全双工服务**（full-duplex service）：如果一台主机上的进程A与另一台主机上的进程B存在一条TCP连接，那么应用层数据就可在从进程B流向进程A的同时，也从进程A流向进程B。TCP连接也总是**点对点**（point-to-point）的，即在单个发送方与单个接收方之

间的连接。所谓“多播”（参见4.7节）是指在一次发送操作中，从一个发送方将数据传送给多个接收方，这对TCP来说是不可能的。对于TCP而言，两台主机刚好成双，三台主机太多！

我们现在来看看TCP连接是怎样建立的。假设运行在某台主机上的一个进程想与另一台主机上的一个进程建立一条连接。前面讲过，发起连接的这个进程称为*客户机进程*，而另一个进程称为*服务器进程*。客户机应用进程首先要通知客户机运输层，它想与服务器上的一个进程建立一条连接。2.7节讲过，一个Java客户机程序通过发出下面的命令来实现此目的。

```
Socket clientSocket = new Socket ("hostname", portNumber) ;
```

其中hostname是服务器的名字，portNumber标识服务器上的进程。这样，客户机上的运输层便开始与服务器上的运输层建立一条TCP连接。我们将在本节后面详细地讨论连接建立的过程。现在知道下列事实就可以了：客户机首先发送一个特殊的TCP报文段，服务器用另一个特殊的TCP报文段来响应，最后，客户机再用第三个特殊报文段作为响应。前两个报文段不承载“有效载荷”，也就是不包含应用层数据，而第三个报文段可以承载有效载荷。由于在这两台主机之间发送了3个报文段，所以这种连接建立过程常被称为三次握手（three-way handshake）。

历史事件

Vinton Cerf、Robert Kahn与TCP/IP

在20世纪70年代早期，分组交换网开始飞速增长，ARPAnet是因特网的前身，也是当时许多分组交换网中的一个。这些网络都有它们各自的协议。Vinton Cerf 和 Robert Kahn这两个研究人员认识到互联这些网络的重要性，并发明了沟通网络的TCP/IP协议，该协议表示传输控制协议与网际协议（Transmission Control Protocol/ Internet Protocol）。虽然Cerf和 Kahn开始时把该协议看成是单一的实体，但是后来将它分成两部分：TCP和IP，它们独立运行。Cerf和 Kahn在1974年5月的《IEEE Transactions on Communications Technology》杂志上发表了一篇关于TCP/IP的论文[Cerf 1974]。

TCP/IP协议是当今因特网支柱性的协议，但它的发明先于PC机与工作站，先于以太网及其他局域网技术的激增，先于Web、流式音频及网上聊天等。Cerf和 Kahn看到了对于联网的协议的需求，一方面为将来的应用提供了广泛的支持，另一方面允许任何主机与链路层协议互操作。

2004年，Cerf和Kahn由于“联网方面的开创性工作（包括因特网的基本通信协议TCP/IP的设计和实现），以及联网方面富有才能的领导”而获得了ACM的图灵奖，该奖项被认为是“计算机界的诺贝尔奖”。

一旦建立起一条TCP连接，两个应用进程之间就可以相互发送数据了。我们考虑一下从客户机进程向服务器进程发送数据的情况。如2.7节中所述，客户机进程通过套接字（该进程的门户）传递数据流。数据一旦通过该门户，它就由客户机中运行的TCP控制了。如图3-28所示，TCP将这些数据引导到该连接的**发送缓存**（send buffer）里，发送缓存是在三次握手初期设置的缓存之一。接下来TCP就会不时地从发送缓存里取出一块数据。有趣的是，在TCP规约[RFC 793]中却没提及TCP应何时实际发送缓存里的数据，只是描述为“TCP应该在它方便的时候以报文段的形式发送数据”。TCP可从缓存中取出并放入报文段中的数据量受限于**最大报**

文段长（maximum segment size, MSS）。MSS通常根据最初确定的最大链路层帧长度来设置，本地发送主机发送长度是这样的帧，即所谓最大传输单元（maximum transmission unit, MTU），接着设置该MSS以保证一个TCP报文段（当封装在一个IP数据报中时）适合单个链路层帧。MTU的常见值是1460字节、536字节或512字节。有人已经提出了发现路径的MTU的方法，并基于该路径MTU值设置MSS，其中路径MTU是指能在从源到目的的所有链路上发送的最大链路层帧[RFC 1191]。注意到MSS是指报文段里应用层数据的最大长度，而不是指包括TCP首部的TCP报文段的最大长度。（该术语很容易混淆，但是我们不得不采用它，因为它已经根深蒂固。）

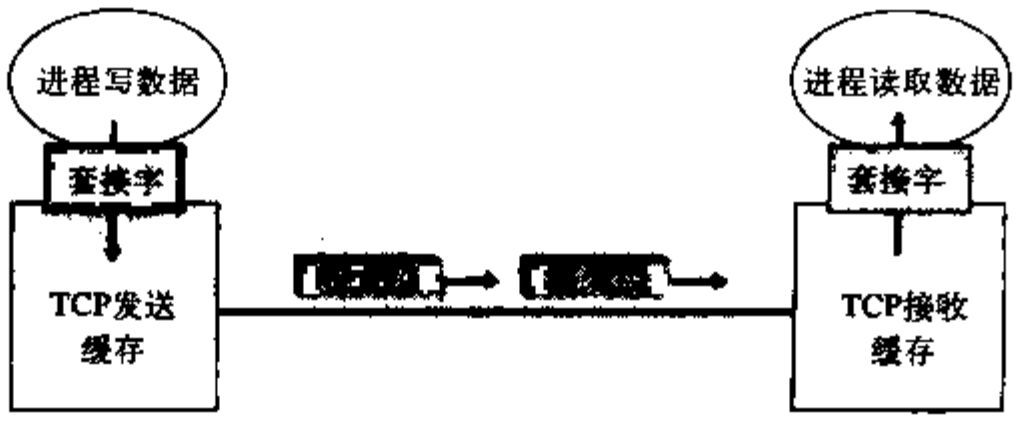


图3-28 TCP发送缓存和接收缓存

TCP将每块客户机数据加一个TCP首部，从而形成多个TCP报文段（TCP segment）。这些报文段被下传给网络层，网络层将其分别封装在网络层IP数据报中。然后这些IP数据报被发送到网络中。当TCP在另一端接收到一个报文段后，该报文段的数据就被放入该TCP连接的接收缓存中，如图3-28所示。应用程序从此缓存中读取数据流。TCP连接的每一端都有各自的发送缓存和接收缓存。（读者可以参阅配套网站http://www.awl.com/kurose-ross上的在线流控制Java小程序，它提供了关于发送缓存和接收缓存的一个动画演示。）

从以上讨论中可以看出，TCP连接的组成包括：一台主机上的缓存、变量和与一个进程连接的套接字，以及另一台主机上的一套缓存、变量和与一个进程连接的套接字。如前面讲过的那样，在这两台主机之间的网络要素（路由器、交换机和中继器）中，没有为该连接分配任何缓存和变量。

3.5.2 TCP报文段结构

简要地了解了TCP连接后，我们研究一下TCP报文段结构。TCP报文段由首部字段和一个数据字段组成。数据字段包含有一块应用数据。如前所述，MSS限制了报文段数据字段的最大长度。当TCP发送一个大文件（例如，某Web页面上的一个图像）时，TCP通常是将文件划分成长度为MSS的若干块（最后一块除外，它通常小于MSS）。然而，交互式应用通常传送长度小于MSS的数据块。例如，对于像Telnet这样的远程登录应用，其TCP报文段的数据字段通常只有一个字节。由于TCP的首部一般是20 字节（比UDP首部多12 字节），所以Telnet发送的报文段也许只有21 字节长。

图3-29给出了TCP报文段的结构。和UDP一样，首部包括源端口号和目的端口号（source and destination port number），它用于多路复用/多路分解来自或送至上层应用的数据。另外，同UDP一样，TCP首部也包括检验和字段（checksum field）。TCP报文段首部还包含下列字段：

- 32比特的序号字段（sequence number field）和32比特的确认号字段（acknowledgment number field）。这些字段被TCP发送方和接收方用来实现可靠数据传输服务，稍后进行

讨论。

- 16比特的**接收窗口**（receive window）字段，该字段用于流量控制。我们很快就会看到，该字段用于指示接收方愿意接受的字节数量。
- 4比特的**首部长度字段**（header length field），该字段指示了以32 比特的字为单位的TCP首部长度。由于TCP选项字段的原因，TCP首部的长度是可变的。（通常，选项字段为空，所以一般TCP首部的长度就是20 字节。）
- 可选与变长的**选项字段**（options field），该字段用于当发送方与接收方协商最大报文段长度（MSS），或在高速网络环境下用作窗口调节因子时使用。首部字段中还定义了一个时间戳选项。可参见RFC 854和RFC 1323以了解其他细节。
- 6比特的**标志字段**（flag field）。ACK比特用于指示确认字段中的值是有效的，即该报文段包括一个对已被成功接收报文段的确认。RST、SYN和FIN比特用于连接建立和拆除，我们将在本节后面讨论该问题。当PSH比特被设置时，就指示接收方应立即将数据交给上层。最后，URG比特用来指示报文段里存在着被发送方的上层实体置为“紧急”的数据。紧急数据的最后一个字节由16比特的**紧急数据指针字段**（urgent data pointer field）指出。当紧急数据存在并给出指向紧急数据尾的指针时，TCP必须通知接收方的上层实体。（在实际中，PSH、URG和紧急数据指针并没有使用。为了完整性起见，我们才提及这些字段。）

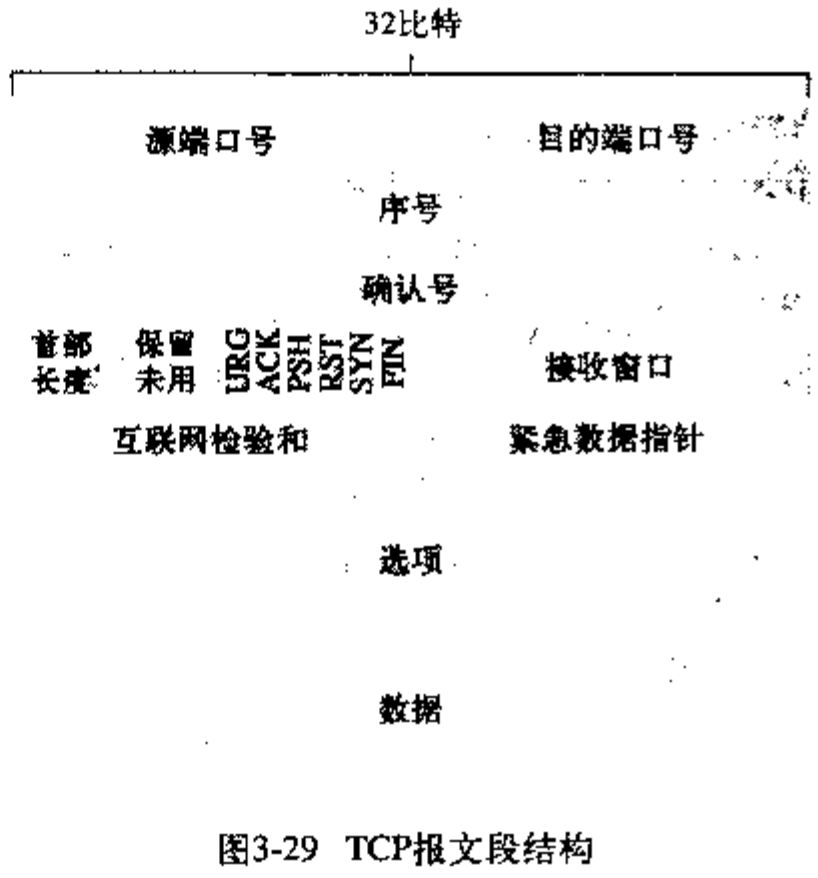


图3-29 TCP报文段结构

1. 序号和确认号

TCP报文段首部两个最重要的字段是序号字段和确认号字段。这两个字段是TCP可靠传输服务的关键部分。但是在讨论这两个字段是如何用于提供可靠的数据传输之前，我们首先来解释一下TCP在这两个字段中究竟放置了什么。

TCP把数据看成一个无结构的但是有序的字节流。我们从TCP对序号的使用上可以看出这一点，这是因为序号是建立在传送的字节流之上，而不是建立在传送的报文段的序列之上。一个**报文段的序号**（sequence number for a segment）因此是该报文段首字节的字节流编号。例如，假设主机A上的一个进程想通过一条TCP连接向主机B上的一个进程发送一个数据流。主机A中的TCP将隐式地对数据流中的每一个字节进行编号。假定数据流由一个包含500 000字节的文件组成，MSS为1000字节，数据流的首字节编号是0。如图3-30所示，TCP将为该数据流构建500个报文段。第一个报文段的序号被赋为0，第二个报文段的序号被赋为1000，第三个报文段的序号被赋为2000，以此类推。每一个序号被填入到相应TCP报文段首部的序号字段中。

现在我们考虑一下确认号字段。确认号要比序号难处理一些。前面讲过，TCP是全双工的，因此主机A也许在向主机B发送数据的同时，也接收来自主机B的数据（都是同一条TCP连接的一部分）。从主机B到达的每个报文段中都有一个序号用于从B流向A的数据。主机A填

充进报文段的确认号是主机A期望从主机B收到的下一字节的序号。举一些例子有助于读者理解实际发生的事情。假设主机A已收到了来自主机B编号为0～535的所有字节，同时假设它要发送一个报文段给主机B。主机A等待主机B的数据流中字节536及后续所有字节。所以，主机A会在它发往主机B的报文段的确认号字段中填上536。

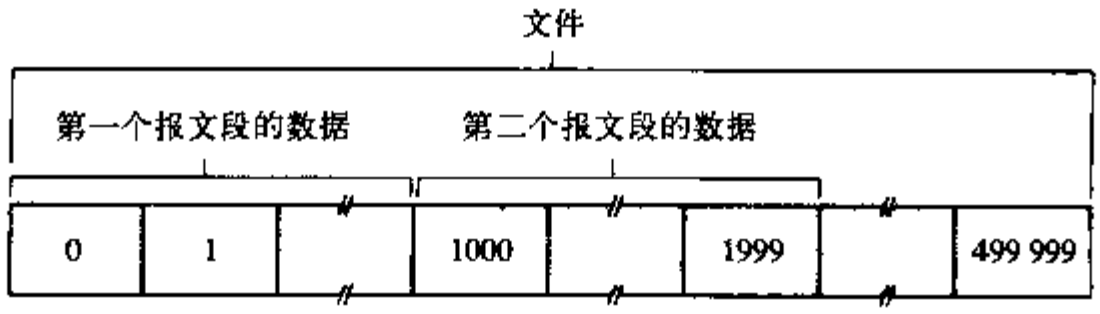


图3-30 把文件数据划分成TCP报文段

再举一个例子，假设主机A已收到一个来自主机B的包含字节0～535的报文段，以及另一个包含字节900～1000的报文段。由于某种原因，主机A还没有收到字节536～899的报文段。在这个例子中，主机A为了重组主机B的数据流，仍在等待字节536（和其后的字节）。因此，A到B的下一个报文段将在确认号字段中包含536。因为TCP只确认数据流中至第一个丢失字节为止的字节，所以TCP被称为是提供**累积确认**（cumulative acknowledgemnt）。

最后一个例子也会引发一个重要而微妙的问题。主机A在收到第二个报文段（字节536～899）之前收到第三个报文段（字节900～1000）。因此，第三个报文段失序到达。问题是：当主机在TCP连接中收到失序报文段时该怎么办？有趣的是，TCP RFC并没有为此明确规定任何规则，而是把这一问题留给实现TCP的编程人员去处理。他们有两个基本的选择：① 接收方立即丢弃失序报文段（如前所述，可以简化对接收方的设计）；② 接收方保留失序的字节，并等待缺少的字节以填补该间隔。显然，后一种选择对于网络带宽更有效，也是实践中采用的方法。

在图3-30中，我们假设初始序号为0。事实上，一条TCP连接的双方均可随机地选择初始序号。这样做可以减少将那些仍在网络中的来自两台主机之间先前已终止的连接的报文段，误认为是后来这两台主机之间新建连接所产生的有效报文段的可能性（它碰巧与旧连接使用了相同的端口号）[Sunshine 1978]。

2. Telnet：*序号和确认号的一个学习案例*

Telnet由RFC 854定义，它现在是一个用于远程登录的流行应用层协议。它运行在TCP之上，被设计成可在任意一对主机之间工作。Telnet与第2章讨论的批量数据传输应用不同，它是一个交互式应用。我们在此讨论一个Telnet例子，该例子很好地阐述了TCP的序号与确认号。我们注意到许多用户现在更愿意采用ssh协议而不是Telnet，因为在Telnet连接中发送的数据（包括口令）是未加密的，这使得Telnet易于受到窃听攻击（如8.7节中讨论的那样）。

假设主机A发起一个与主机B的Telnet会话。因为是主机A发起该会话，因此主机A被标记为客户机，主机B被标记为服务器。用户键入的每个字符（在客户机端）都会被发送至远程主机；远程主机收到后会复制一个相同的字符发回客户机，并显示在Telnet用户的屏幕上。这种“回显”（echo back）用于确保由Telnet用户发送的字符已经被远程主机收到并处理。因此，在从用户击键到字符显示在用户屏幕上之间的这段时间内，每个字符在网络中传输了两次。

现在假设用户输入了一个字符“C”，然后喝一杯咖啡。我们考察一下在客户机与服务器之间发送的TCP报文段。如图3-31所示，假设客户机和服务器的起始序号分别是42和79。前

面讲过，一个报文段的序号就是该报文段数据字段首字节的序号。因此，客户机发送的第一个报文段的序号为42，服务器发送的第一个报文段的序号为79。前面讲过，确认号就是主机期待的数据的下一个字节序号。在TCP连接建立后但没有发送任何数据之前，客户机等待字节79，而服务器等待字节42。

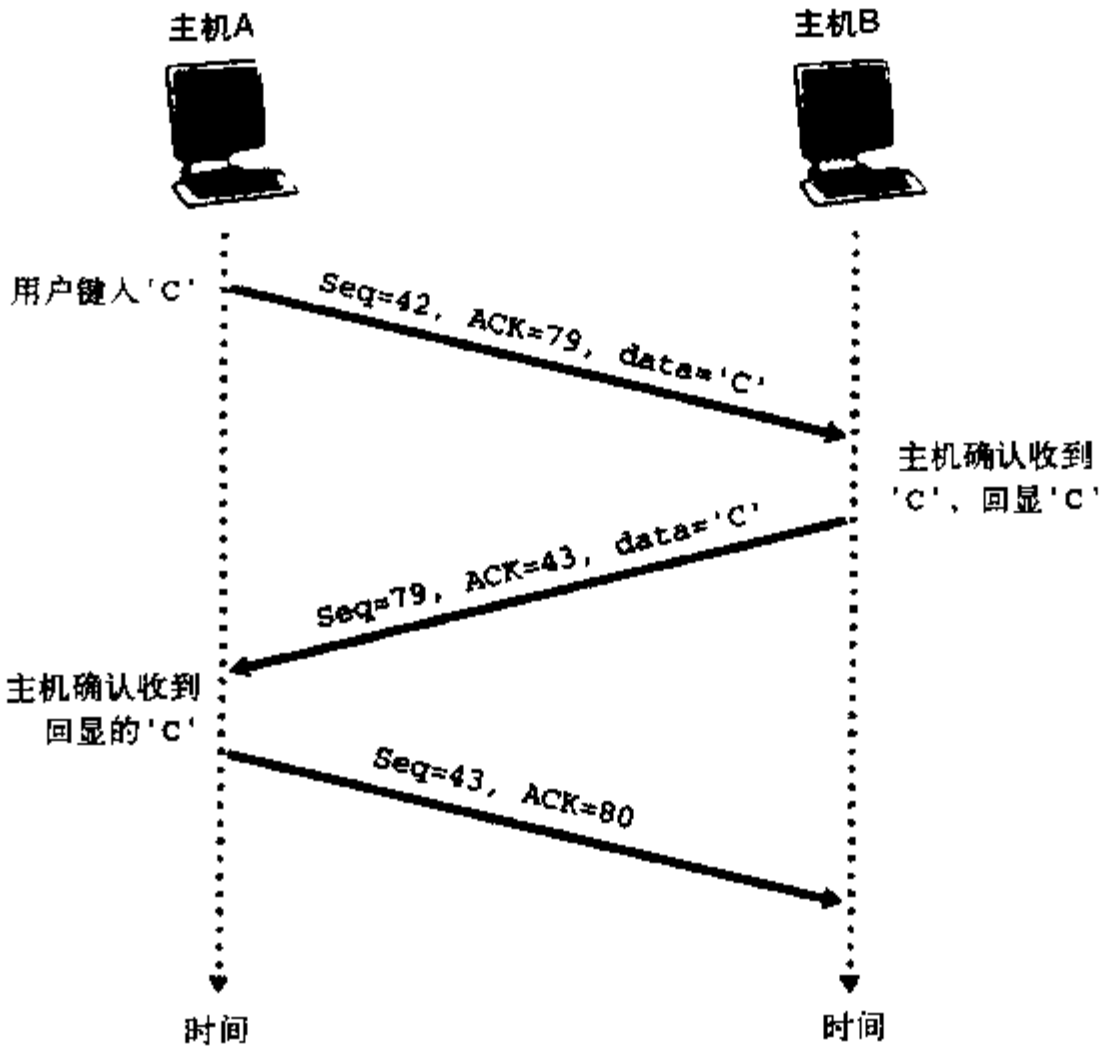


图3-31 一个基于TCP的简单Telnet应用的确认号和序号

如图3-31所示，共发送3个报文段。第一个报文段是由客户机发往服务器，其数据字段里包含一字节的字符“C”的ASCII码，其序号字段里是42，这如我们刚讲到的那样。另外，由于客户机还没有接收到来自服务器的任何数据，因此该报文段中的确认号字段里是79。

第二个报文段是由服务器发往客户机。它有两个目的：第一个目的是为服务器所收到的数据提供确认。服务器通过在确认号字段中填入43，告诉客户机它已经成功地收到字节42及以前的所有字节，现在正等待着字节43的出现。第二个目的是回显字符“C”。因此，在第二个报文段的数据字段里填入的是字符“C”的ASCII码。第二个报文段的序号为79，它是该TCP连接上从服务器到客户机的数据流的起始序号，也是服务器要发送的第一个字节的数据。注意到对客户机到服务器的数据的确认被装载在一个承载服务器到客户机的数据的报文段中，这种确认被称为是**捎带**（piggybacked）在服务器到客户机的数据报文段中的。

第三个报文段是从客户机发往服务器的。它的唯一目的是确认已从服务器收到的数据。（前面讲过，第二个报文段中包含的数据是字符“C”，是从服务器到客户机的。）该报文段的数据字段为空（即确认信息没有被捎带在任何从客户机到服务器的数据中）。该报文段的确认号字段填入的是80，因为客户机已经收到了字节流中序号为79及以前的字节，它现在正等待着字节80的出现。你可能认为这有点奇怪，即该报文段里没有数据但仍有序号。这是因为TCP存在序号字段，报文段需要填入某些序号。

3.5.3 往返时延的估计与超时

TCP如同3.4节所讲的rdt协议一样，它采用超时/重传机制来处理报文段的丢失问题。尽管概念比较简单，但是当在如TCP这样的实际协议中实现超时/重传机制时还是会产生许多微妙的问题。也许最明显的一个问题就是超时间隔长度的设置。显然，超时间隔必须大于TCP连接的往返时延（RTT），即从一个报文段发出到收到其确认时。否则，会造成不必要的重传。但是这个时间间隔到底应该是多大呢？刚开始时应如何估计往返时延呢？是否应该为所有未确认的报文段各设一个定时器？问题是如此之多！本节中的讨论基于[Jacobson 1988]中有关TCP的工作以及IETF关于管理TCP定时器的建议[RFC 2988]。

1. 估计往返时延

我们开始学习TCP定时器的管理问题，考虑一下TCP是如何估计发送方与接收方之间的往返时延的。这是通过如下方法完成的。报文段的样本RTT（表示为SampleRTT）就是从某报文段被发出（即交给IP）到对该报文段的确认被收到之间的时间量。大多数TCP的实现仅在某个时刻做一次SampleRTT测量，而不是为每个发送的报文段测量一个SampleRTT。也就是说，在任意时刻，仅为一个已发送的但目前尚未被确认的报文段估计SampleRTT，从而产生一个接近每个RTT的新SampleRTT值。另外，TCP决不为已重传的报文段计算SampleRTT，而仅为传输一次的报文段测量SampleRTT。（本章后面的一个习题请你考虑一下为什么要这么做。）

显然，由于路由器的拥塞和端系统负载的变化，这些报文段的SampleRTT值会随之波动。由于这种波动，任何给出的SampleRTT值也许都是非典型的。因此，为了估计一个典型的RTT，自然要采取某种对SampleRTT取平均值的办法。TCP维持一个SampleRTT均值（称为EstimatedRTT）。一旦获得一个新SampleRTT，TCP就会根据下列公式来更新EstimatedRTT：

$$\text{EstimatedRTT} = (1-\alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$$

上面的公式是以编程语言的语句方式给出的，即EstimatedRTT的新值是由以前的EstimatedRTT值与SampleRTT新值加权组合而成的。在[RFC 2988]中给出的α参考值是$\alpha = 0.125$（即1/8），这时上面的公式变为：

$$\text{EstimatedRTT} = 0.875 \cdot \text{EstimatedRTT} + 0.125 \cdot \text{SampleRTT}$$

注意到EstimatedRTT是一个由SampleRTT值而得出的加权平均值。如在本章后面课后习题中讨论的那样，这个加权平均对最新样本赋予的权值要大于对老样本赋予的权值。这是很自然的，因为越新的样本越能更好地反映出网络的当前拥塞情况。从统计学观点来讲，这种平均被称为**指数加权移动平均**（Exponential Weighted Moving Average, EWMA）。EWMA中的“指数”一词看起来是指一个给定的SampleRTT的权值在更新的过程中呈指数型衰减。在课后习题中，将要求你推导出EstimatedRTT的指数表达形式。

图3-32显示了当$\alpha = 1/8$时，在gaia.cs.umass.edu（位于美国马萨诸塞州的Amherst）与fantasia.eurecom.fr（位于法国南部）之间的一条TCP连接上的SampleRTT值与EstimatedRTT值。显然，SampleRTT的变化在计算EstimatedRTT的过程中趋于平缓。

除了估算RTT外，测量RTT的变化也是有意义的。[RFC 2988]定义了RTT偏差DevRTT，用于估算SampleRTT一般会偏离EstimatedRTT的程度：

$$\text{DevRTT} = (1-\beta) \cdot \text{DevRTT} + \beta \cdot |\text{SampleRTT} - \text{EstimatedRTT}|$$

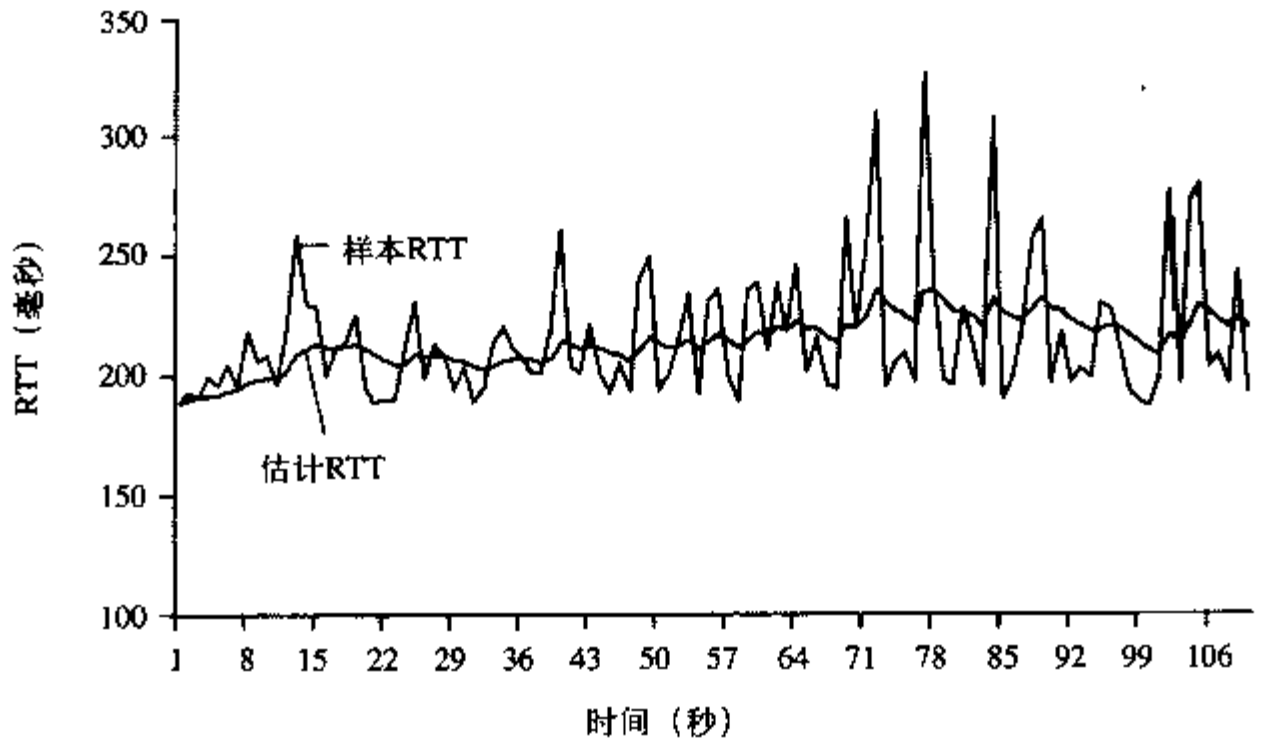


图3-32 RTT样本和RTT估计

注意到DevRTT是一个SampleRTT与EstimatedRTT之间差值的EWMA。如果SampleRTT值波动较小，那么DevRTT的值就会很小；相反，如果波动很大，那么DevRTT的值就会很大。β的推荐值为0.25。

实践原则

与3.4节中所学的方法很像，TCP通过使用肯定确认与定时器来提供可靠数据传输。TCP确认正确接收到的数据，而当认为报文段或其确认报文丢失或受损时，TCP会重传这些报文段。某些版本的TCP还有一个隐式NAK机制（在TCP的快速重传机制下，收到对一个特定报文段的3个冗余ACK就可作为对后面报文段的一个隐式NAK，从而在超时之前触发对该报文段的重传。TCP使用一系列编号以使接收方能识别丢失或重复的报文段。像可靠数据传输协议rdt3.0的情况一样，TCP自己也无法确认一个报文段或其ACK是丢失了、受损了，还是时延过长了。在发送方，TCP的响应是相同的：重传这个有问题的报文段。

TCP也使用流水线，使得发送方在任意时刻都可以有多个已发出但还未被确认的报文段存在。我们在前面已经看到，当报文段长度与往返时延之比很小时，流水线可显著地增加一个会话的吞吐量。一个发送方可以具有的未被确认报文段的确切个数是由TCP的流量控制和拥塞控制机制决定的。TCP流量控制将在本节后面讨论，TCP拥塞控制将在3.7节中讨论。我们暂时只需知道TCP发送方使用了流水线即可。

2. 设置和管理重传超时间隔

假设已经给出了EstimatedRTT值和DevRTT值，那么TCP超时间隔应该用什么值呢？很明显，超时间隔应该大于等于EstimatedRTT，否则将造成不必要的重传。但是超时间隔也不应该比EstimatedRTT大太多，否则当报文段丢失时，TCP不能很快地重传该报文段，从而将给上层应用带来很大的数据传输时延。因此，要求将超时间隔设为EstimatedRTT加上一定余量。当SampleRTT值波动较大时，这个余量应该大些；当波动比较小时，这个余量应该小些。因此DevRTT值应该在这里起作用了。所有这些考虑都被用到决定TCP确定重传超

时间隔的方法中：

```
TimeoutInterval=EstimatedRTT+4 · DevRTT
```

3.5.4 可靠数据传输

前面讲过，因特网的网络层服务（IP服务）是不可靠的。IP不保证数据报的交付，不保证数据报的按序交付，也不保证数据报中数据的完整性。对于IP服务，数据报有可能因在路由器缓存中溢出而永远不能到达目的地，数据报也有可能是乱序到达，而且数据报中的比特可能受损（由0变为1或者相反）。由于运输层报文段是被IP数据报携带着在网络中传输的，所以运输层的报文段也会遇到这些问题。

TCP在IP的不可靠的尽力而为服务的基础上建立了一种**可靠数据传输服务**（reliable data transfer service）。TCP的可靠数据传输服务确保一个进程从其接收缓存中读出非损坏的、无间隔的、非冗余的和按序的数据流，即该字节流与连接的另一方端系统发送出的字节流完全一样。TCP提供可靠数据传输的方法涉及3.4节中所学的许多原理。

在前面阐述可靠数据传输技术时，曾假定每一个已发送但未被确认的报文段都有一个与其相关联的定时器，这在概念上是最简单的。虽然这在理论上很好，但定时器的管理却需要相当大的开销。因此，推荐的定时器管理过程[RFC 2988]仅使用单一的重传定时器，即使有多个已发送但还未被确认的报文段。本小节中描述的TCP协议遵循了这种单一定时器的建议。

我们将分两个步骤来讨论TCP是如何提供可靠数据传输的。我们先给出一个TCP发送方的高度简化的描述，该发送方只用超时来恢复报文段的丢失。然后我们再给出一个更全面的描述，该描述中除了使用超时机制外，还使用冗余确认技术。在下面的讨论中，我们假定数据仅向一个方向发送，即从主机A到主机B，且主机A在发送一个大文件。

图3-33给出了一个TCP发送方的高度简化的描述。我们看到在TCP发送方有3个与发送和重传有关的主要事件：从上层应用程序接收数据；定时器超时；收到ACK报文。一旦第一个主要事件发生，TCP就从应用程序接收数据，将数据封装在一个报文段中，并该报文段交给IP。注意到每一个报文段都包含一个序号，如3.5.2节所讲的那样，这个序号就是该报文段第一个数据字节的字节流编号。还要注意的是，如果定时器还没有为其他报文段而运行，则当报文段被传给IP时，TCP就启动该定时器。（将定时器与最早的未被确认的报文段关联起来，这对我们是有帮助的。）该定时器的过期间隔是`TimeoutInterval`，它是由3.5.3节中所讲`EstimatedRTT`和`DevRTT`计算得出的。

第二个主要事件是超时。TCP通过重传引起超时的报文段来响应超时事件。然后TCP重启定时器。

TCP发送方必须处理的第三个主要事件是一个来自接收方的确认报文段（ACK）的到达（更确切地说是一个包含了有效ACK字段值的报文段）。当该事件发生时，TCP将ACK的值y与变量`SendBase`进行比较。TCP状态变量`SendBase`是最早未被确认的字节的序号。（因此`SendBase-1`就是指接收方已正确按序接收到数据的最后一个字节的序号。）如前所述，TCP采用累积确认，所以y确认了字节编号在y之前的所有字节都已经收到。如果`y > SendBase`，则该ACK是在确认一个或多个先前未被确认的报文段。因此发送方更新其`SendBase`变量；另外，如果当前有未被确认的报文段，TCP还要重新启动定时器。

1. 一些有趣的情况

我们刚刚描述了一个关于TCP如何提供可靠数据传输的高度简化的版本。但是，这种高

```
/*假设发送方不受TCP流量和拥塞控制的限制，来自上层的数据的长度小于MSS，且数据
传输只在一个方向进行。*/

NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber

loop (forever) {
   switch(event)

      event: data received from application above
         create TCP segment with sequence number NextSeqNum
         if (timer currently not running)
            start timer
         pass segment to IP
         NextSeqNum=NextSeqNum+length(data)
         break;

      event: timer timeout
         retransmit not-yet-acknowledged segment with
            smallest sequence number
         start timer
         break;

      event: ACK received, with ACK field value of y
         if (y > SendBase) {
            SendBase=y
            if (there are currently any not-yet-acknowledged segments)
               start timer
            }
            break;

   } /* end of loop forever */
```

图3-33 简化的TCP发送方

度简化的版本仍然存在着许多微妙之处。为了较好地体会该协议的工作过程，我们来研究一些简单情况。图3-34描述了第一种情况，主机A向主机B发送一个报文段。假设该报文段的序号是92，而且包含8 字节数据。在发出该报文段之后，主机A等待一个来自主机B的确认号为100的报文段。虽然A发出的报文段在主机B上被收到，但从主机B发往主机A的确认报文丢失了。在这种情况下，就会发生超时事件，主机A会重传相同的报文段。当然，当主机B收到该重传的报文段时，它将通过序号发现该报文段包含了已经收到的数据。因此，主机B中的TCP将丢弃该重传的报文段中的字节。

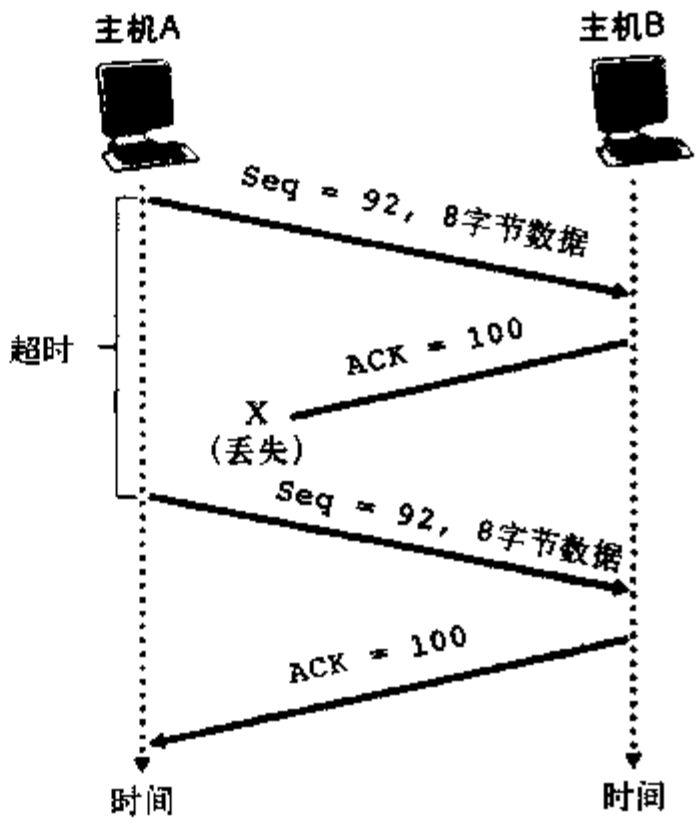


图3-34 由于确认报文丢失而重传

第二种情况如图3-35所示，主机A连续发送了两个报文段。第一个报文段的序号是92，包含8字节数据；第二个报文段的序号是100，包含20 字节数据。假设两个报文段都完好无损地到达主机B，并且主机B为每一个报文段分别发送一个确认。第一个确认报文的确认号是100，第二个确认报文的

确认号是120。现在假设在超时之前这两个报文段中没有一个确认报文到达主机A。当超时事件发生时，主机A重传序号为92的第一个报文段，并重启定时器。只要第二个报文段的ACK在新的超时发生以前到达，则第二个报文段将不会被重传。

在第三种也是最后一种情况中，假设主机A像第二种情况那样发送两个报文段。第一个报文段的确认报文在网络中丢失，但在超时事件发生之前主机A收到一个确认号为120的确认报文。因此主机A知道主机B已经收到了序号为119及之前的所有字节，所以主机A不会重传这两个报文段中的任何一个。这种情况如图3-36所示。

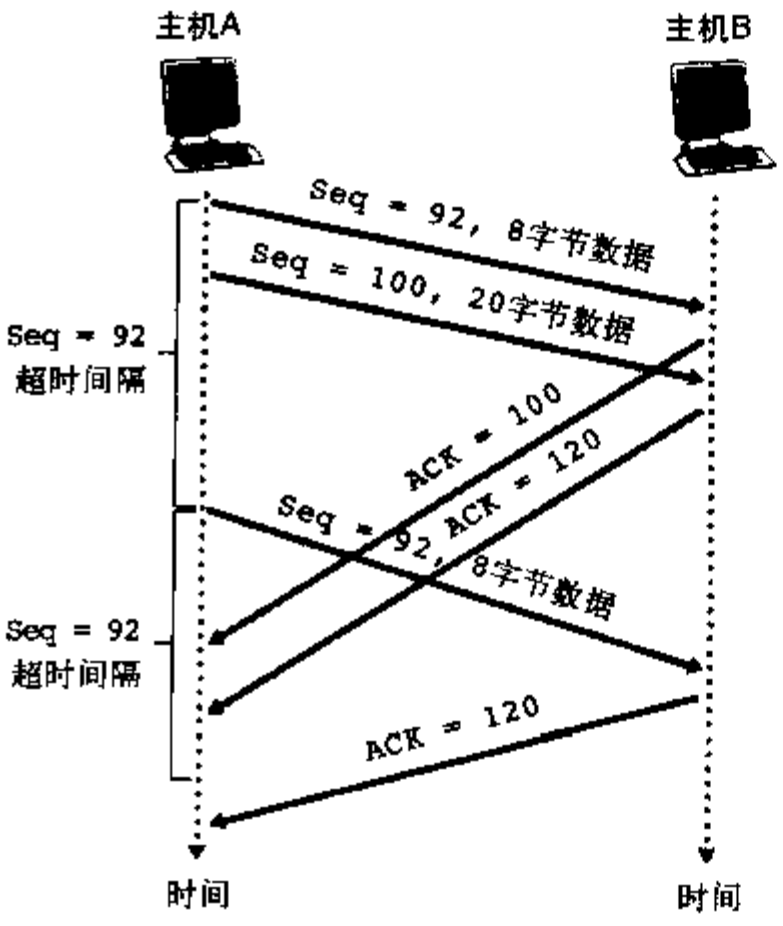


图3-35 报文段100没有重传

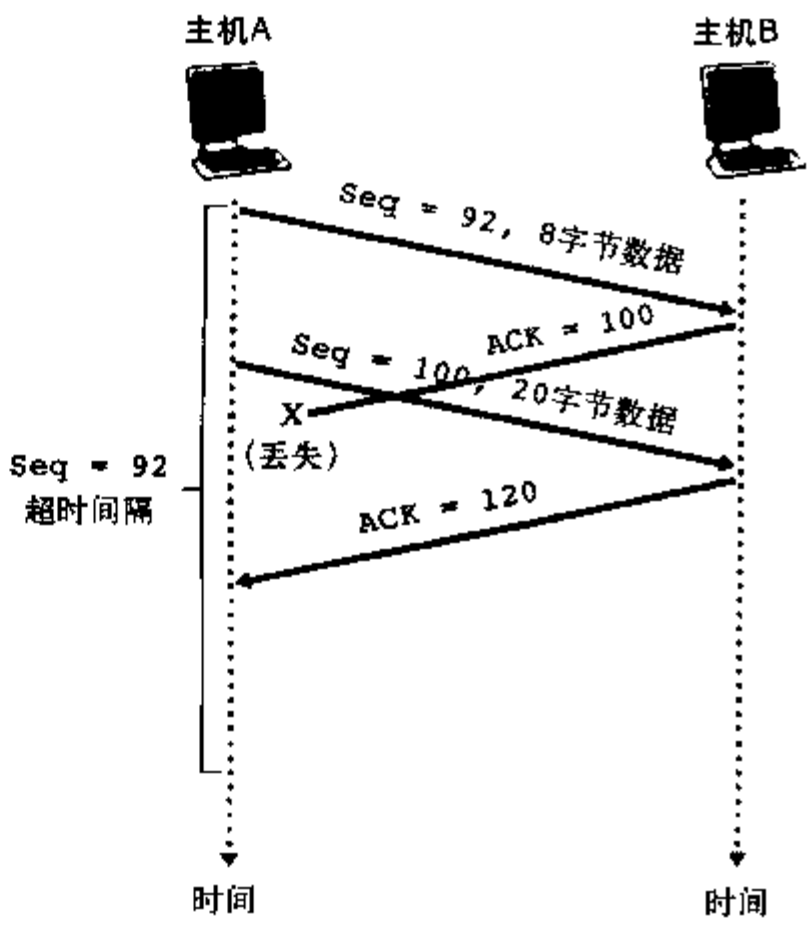


图3-36 累积确认避免了第一个报文段的重传

2. 加倍超时间隔

我们现在讨论一下在大多数TCP实现中所做的一些改动。首先关注的是在定时器超时后的超时间隔的长度。在这种改动中，每当超时事件发生时，如前所述，TCP重传具有最小序号的还未被确认的报文段。但是每次TCP重传都会将下一次的超时间隔设为先前值的两倍，而不是从EstimatedRTT和DevRTT（3.5.3节中描述的）推出。例如，假设当定时器第一次过期时，与最早的未被确认的报文段相关联的TimeoutInterval是0.75秒。TCP重传该报文段时，将把新的过期时间设置为1.5 秒。如果1.5秒后定时器又过期了，则TCP将再次重传该报文段，并把过期时间设置为3.0 秒。因此，超时间隔在每次重传后会呈指数型增长。然而，每当定时器在另两个事件中的任意一个发生（即收到上层应用的数据和收到ACK）时，TimeoutInterval是由最近的EstimatedRTT值与DevRTT值推算得出的。

这种改动提供了一个有限形式的拥塞控制。（更复杂的TCP拥塞控制形式将在3.7节中学习。）定时器过期很可能是由网络拥塞引起的，即太多的分组到达源与目的地之间路径上的一台（或多台）路由器的队列中，造成分组丢失或长时间的排队时延。在拥塞的时候，如果源持续重传分组，会使拥塞更加严重。相反，TCP使用更优雅的方式，每个发送方的重传都是经过越来越长的时间间隔后进行的。在第5章学习CSMA/CD时，将看到以太网采用了类似的思路。

3. 快速重传

超时触发重传存在的另一个问题是超时周期可能相对较长。当一个报文段丢失时，这种长超时周期迫使发送方等待很长时间才重传丢失的分组，因而增加了端到端时延。幸运的是，发送方通常可在超时事件发生之前通过注意所谓冗余ACK来较好地检测丢包情况。冗余ACK (duplicate ACK) 就是再次确认某个报文段的ACK，而发送方先前已经收到对该报文段的确认。要理解发送方对冗余ACK的响应，我们必须首先看一下接收方为什么会发送冗余ACK。表3-2总结了TCP接收方的ACK生成策略[RFC 1122, RFC 2581]。当TCP接收方收到一个具有这样序号（即大于下一个所期望的、按序的序号）的报文段时，它检测到了数据流中的一个间隔，即有报文段丢失。这个间隔可能是由于在网络中报文段丢失或重新排序造成的。因为TCP不使用否定确认，所以接收方不能向发送方发回一个显式的否定确认，而是只需对按序接收到的最后一个字节数据进行重复确认（即产生一个冗余ACK）。(注意到在表3-2中允许接收方不丢弃失序报文段。)

表3-2 产生TCP ACK的建议 [RFC 1122, RFC 2581]

事 件	TCP接收方动作
所期望序号的报文段按序到达。所有在期望序号及其以前的数据都已经被确认	延迟的ACK。对另一个按序报文段的到达最多等待500 ms。如果下一个按序报文段在这个时间间隔内没有到达，则发送一个ACK
有期望序号的报文段按序到达。另一个按序报文段等待ACK传输	立即发送单个累积ACK，以确认两个按序报文段
比期望序号大的失序报文段到达，检测出数据流中的间隔	立即发送冗余ACK，指明下一个期待字节的序号（也就是该间隔的低端字节序号）
能部分或完全填充接收数据间隔的报文段到达	倘若该报文段起始于间隔的低端，则立即发送ACK

因为发送方经常连续发送大量的报文段，所以如果一个报文段丢失，就很可能引起许多一个接一个的冗余ACK。如果TCP发送方接收到对相同数据的3个冗余ACK，它就认为跟在这个已被确认过3次的报文段之后的报文段已经丢失。(在课后习题中，我们将考虑为什么发送方等待3个重复的ACK，而不是仅仅等待一个冗余ACK。) 一旦收到3个冗余ACK，TCP就执行**快速重传** (fast retransmit) [RFC 2581]，即在该报文段的定时器过期之前重传丢失的报文段。如图3-37所示，其中第二个报文段丢失了，在其定时器过期之前重传该报文段。对于具有快速重传的TCP，可用下列代码段代替图3-33中收到ACK事件的代码段：

```
event: ACK received, with ACK field value of y
          if (y > SendBase) {
             SendBase=y
             if (there are currently any not yet
                      acknowledged segments)
                start timer
             }
          else { /* a duplicate ACK for already ACKed
             segment */
             increment number of duplicate ACKs
                received for y
             if (number of duplicate ACKS received
                for y==3) {
                /* TCP fast retransmit */
                resend segment with sequence number y
                }
             break;
```

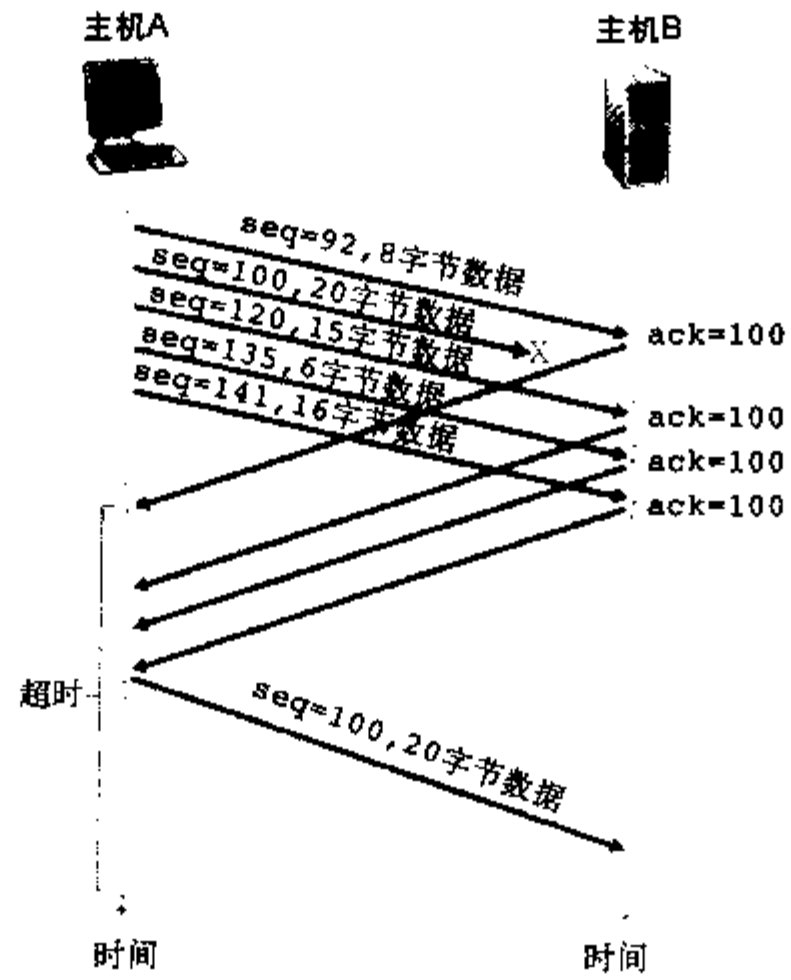


图3-37 快速重传：在某报文段的定时器过期之前重传丢失的报文段

我们在前面提到，当在像TCP这样一个实际协议中实现超时/重传机制时，会产生许多微妙的问题。上面的程序段是在15年以上的TCP定时器使用经验的基础上改进而来的，它应该使读者相信情况确实是这样的。

4. 是回退N步还是选择重传

在结束有关TCP差错恢复机制的学习前，我们考虑下面这个问题：TCP是一个GBN协议还是一个SR协议？前面讲过，TCP确认是累积式的，正确接收但失序的报文段是不会被接收方逐个确认的。因此，如图3-33（也可参见图3-19）所示，TCP发送方仅需维持已发送但未被确认字节的最小序号（SendBase）和下一个要发送字节的序号（NextSeqNum）。在这个意义下，TCP看起来更像一个GBN风格的协议。但是TCP和GBN之间有着一些显著的区别。许多TCP实现中会将正确接收但失序的报文段缓存起来[Stevens 1994]。另外考虑一下，当发送方发送一组报文段1，2，…，N，并且所有的报文段都正确地按序到达接收方时会发生什么呢？进一步假设对分组$n < N$的确认丢失，但是其余$N - 1$个确认报文分别在超时以前到达发送方，这时又会发生什么呢？在该例中，GBN不仅会重传分组n，还会重传所有后继的分组$n+1$，$n+2$，…，N。相反，TCP却至多重传一个报文段，即报文段n。此外，如果对报文段$n+1$的确认报文在报文段n超时之前到达，TCP甚至不会重传报文段n。

对TCP协议提出的一种修改就是所谓的**选择确认**（selective acknowledgment）[RFC 2018]，它允许TCP接收方有选择地确认失序报文段，而不是累积地确认最后一个正确接收的有序的报文段。当将该机制与选择重传机制结合起来使用（即跳过那些已被接收方选择性地确认过的报文段的重传）时，TCP看起来就很像我们通常的SR协议。因此TCP的差错恢复机制也许最好被划分成GBN协议与选择重传协议的混合体。

3.5.5 流量控制

前面讲过，一条TCP连接的双方主机都为该连接设置了接收缓存。当该TCP连接收到正确、

按序的字节后，它就将数据放入接收缓存。相关联的应用进程会从该缓存中读取数据，但没必要数据刚一到达就立即读取。事实上，接收方应用也许正忙于其他任务，甚至要过很长时间后才去读取该数据。如果应用程序读取数据时相当缓慢，而发送方发送数据太多、太快，会很容易使该连接的接收缓存溢出。

TCP为应用程序提供了**流量控制服务**（flow-control service）以消除发送方使接收方缓存溢出的可能性。因此，可以说流量控制是一个速度匹配服务，即发送方的发送速率与接收方应用程序的读速率相匹配。前面提到过，TCP发送方也可能因为IP网络的拥塞而被遏制，这种形式的发送方的控制被称为**拥塞控制**（congestion control），我们将在3.6节和3.7节详细地讨论这个问题。尽管流量控制和拥塞控制采取的动作非常相似（对发送方的遏制），但是它们显然是针对不同原因而采取的措施。不幸的是，许多作者把这两个术语混用，理解力强的读者会明智地区分这两种情况。现在我们来讨论TCP是如何提供流量控制服务的。为了能从整体上看问题，本小节假设TCP实现是这样做的，即TCP接收方丢弃失序报文段。

TCP通过让*发送方*维护一个称为**接收窗口**（receive window）的变量来提供流量控制。非正式地讲，接收窗口用于告诉发送方，该接收方还有多少可用的缓存空间。因为TCP是全双工通信，在连接两端的发送方都各自维护一个接收窗口。我们在一个文件传输情况下研究接收窗口的情况。假设主机A通过一条TCP连接向主机B发送一个大文件。主机B为该连接分配了一个接收缓存，并用`RcvBuffer`来表示其大小。主机B上的应用进程不时地从该缓存中读取数据。定义以下变量：

- `LastByteRead`：主机B上的应用进程从缓存读出的数据流最后一个字节的编号。
- `LastByteRcvd`：从网络中到达的并且已放入主机B接收缓存中的数据流最后一个字节的编号。

由于TCP不允许已分配的缓存溢出，所以下式必须成立：

```
LastByteRcvd-LastByteRead≤RcvBuffer
```

接收窗口用`RcvWindow`表示，根据缓存可用空间的数量来设置：

```
RcvWindow=RcvBuffer-[LastByteRcvd-LastByteRead]
```

由于该空间是随着时间变化的，所以`RcvWindow`是动态的。图3-38对变量`RcvWindow`进行了说明。

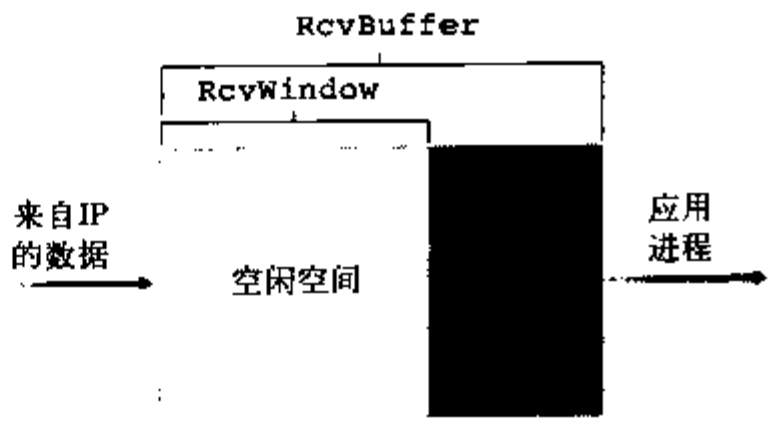


图3-38 接收窗口（RcvWindow）和接收缓存（RcvBuffer）

连接是如何使用变量`RcvWindow`来提供流量控制服务的呢？主机B通过把当前的`RcvWindow`值放入它发给主机A的报文段接收窗口字段中，通知主机A它在该连接的缓存中还有多少可用空间。开始时，主机B设定`RcvWindow = RcvBuffer`。注意到为了实现这一点，主机B必须跟踪几个与连接有关的变量。

主机A轮流跟踪两个变量`LastByteSent`和`LastByteAcked`，这两个变量的意义很明显。注意到这两个变量之间的差`LastByteSent-LastByteAcked`，就是主机A发送到连接中但未被确认的数据量。通过将未确认的数据量控制在值`RcvWindow`以内，就可以保证主机A不会使主机B的接收缓存溢出。因此，主机A在该连接的整个生命周期必须保证：

LastByteSent-LastByteAcked≤RcvWindow

对于这个方案还存在一个小的技术问题。为了理解这一点，假设主机B的接收缓存已经存满，以至于RcvWindow = 0。在将RcvWindow = 0通告给主机A之后，还要假设主机B没有任何数据要发给主机A。此时，考虑一下会发生什么情况。随着主机B上的应用进程将缓存清空，TCP并不向主机A发送带有RcvWindow新值的新报文段；事实上，TCP仅在它有数据或确认要发时才会发送报文段给主机A。这样，主机A不会知道主机B的接收缓存已经有新的空间了，即主机A被阻塞而不能再发送数据！为了解决这个问题，TCP规约中要求：当主机B的接收窗口为0时，主机A继续发送只有一个字节数据的报文段。这些报文段将会被接收方确认。最终缓存将开始清空，并且确认报文里将包含一个非0的RcvWindow值。

网站http://www.awl.com/kurose-ross为本书提供了一个交互式Java小程序，用以说明TCP接收窗口的运行情况。

阐述了TCP的流量控制服务以后，我们在此简要地提一下UDP并不提供流量控制。为了理解这个问题，考虑一下从主机A上的一个进程向主机B上的一个进程发送一系列UDP报文段的情形。作为一个典型的UDP实现，UDP将会把这些报文段添加到相应套接字（进程的门户）“前面”的一个有限大小的缓存中。进程每次从缓存中读取一个完整的报文段。如果进程从缓存中读取报文段的速度不够快，那么缓存将会溢出且报文段将被丢弃。

3.5.6 TCP连接管理

在本小节中，我们较为深入地讨论如何建立和拆除一条TCP连接。尽管这个主题并不特别令人兴奋，但是它很重要，因为TCP连接的建立会显著地增加人们感受到的时延（尤其是在Web上冲浪时）。此外，许多常见的网络攻击（包括难以置信地流行的SYN洪泛攻击）利用了TCP连接管理中的弱点。现在我们观察一下一条TCP连接是如何建立的。假设运行在一台主机（客户机）上的一个进程想与另一台主机（服务器）上的一个进程建立一条连接。客户机应用进程首先通知客户机TCP，它想建立一个与服务器上某个进程之间的连接。客户机中的TCP会用以下方式与服务器中的TCP建立一条TCP连接：

- **第一步**：客户机端的TCP首先向服务器端的TCP发送一个特殊的TCP报文段。该报文段中不包含应用层数据，但是报文段的首部（参见图3-29）中的一个标志位（即SYN比特）被置为1。因此，这个特殊报文段被称为SYN报文段。另外，客户机会选择一个起始序号（client_isn），并将其放置到该起始的TCP SYN报文段的序号字段中。该报文段会被封装在一个IP数据报中，并发送给服务器。为了避免某些安全性攻击，在适当地随机化选择client_isn方面有不少研究成果[CERT 2001-09]。
- **第二步**：一旦包含TCP SYN报文段的IP数据报到达服务器主机（假定它的确到达了），服务器会从该数据报中提取出TCP SYN报文段，为该TCP连接分配TCP缓存和变量，并向客户机TCP发送允许连接的报文段。（我们将在第8章看到，在完成三次握手的第三步之前分配这些缓存和变量，使得TCP易于受到称为SYN洪泛的拒绝服务攻击。）这个允许连接的报文段也不包含应用层数据。但是，在报文段的首部却包含3个重要的信息。首先，SYN比特被置为1。其次，该TCP报文段首部的确认号字段被置为client_isn +1。最后，服务器选择自己的初始序号（server_isn），并将其放置到TCP报文段首部的序号字段中。这个允许连接的报文段实际上表明了：“我收到了你要求建立连接的、带有初始序号client_isn的SYN分组。我同意建立该连接。我自己的初始序号是

server_isn。”该允许连接的报文段有时被称为SYNACK报文段（SYNACK segment）。

- **第三步**：在收到SYNACK报文段后，客户机也要给该连接分配缓存和变量。客户机主机还会向服务器发送另外一个报文段，这个报文段对服务器的允许连接的报文段进行了确认（客户机通过将值server_isn+1放置到TCP报文段首部的确认字段中来完成此项工作）。因为连接已经建立了，所以该SYN比特被置为0。

一旦以上3步完成，客户机和服务器主机就可以相互发送含有数据的报文段了。在以后的每一个报文段中，SYN比特都将被置为0。注意到为了建立连接，在两台主机之间发送了3个分组，如图3-39所示。由于这个原因，这种连接建立过程通常被称为三次握手（three-way handshake）。有关TCP三次握手的几方面问题将在课后习题中讨论。（为什么需要初始序号？为什么需要三次握手，而不是两次握手？）注意到这样一件事是很有趣的，一个攀岩者和一个保护者（他位于攀岩者的下面，其任务是处理好攀岩者的安全绳索）就使用了与TCP相同的三次握手通信协议，以确保攀岩者开始攀爬前双方都已经准备好了。

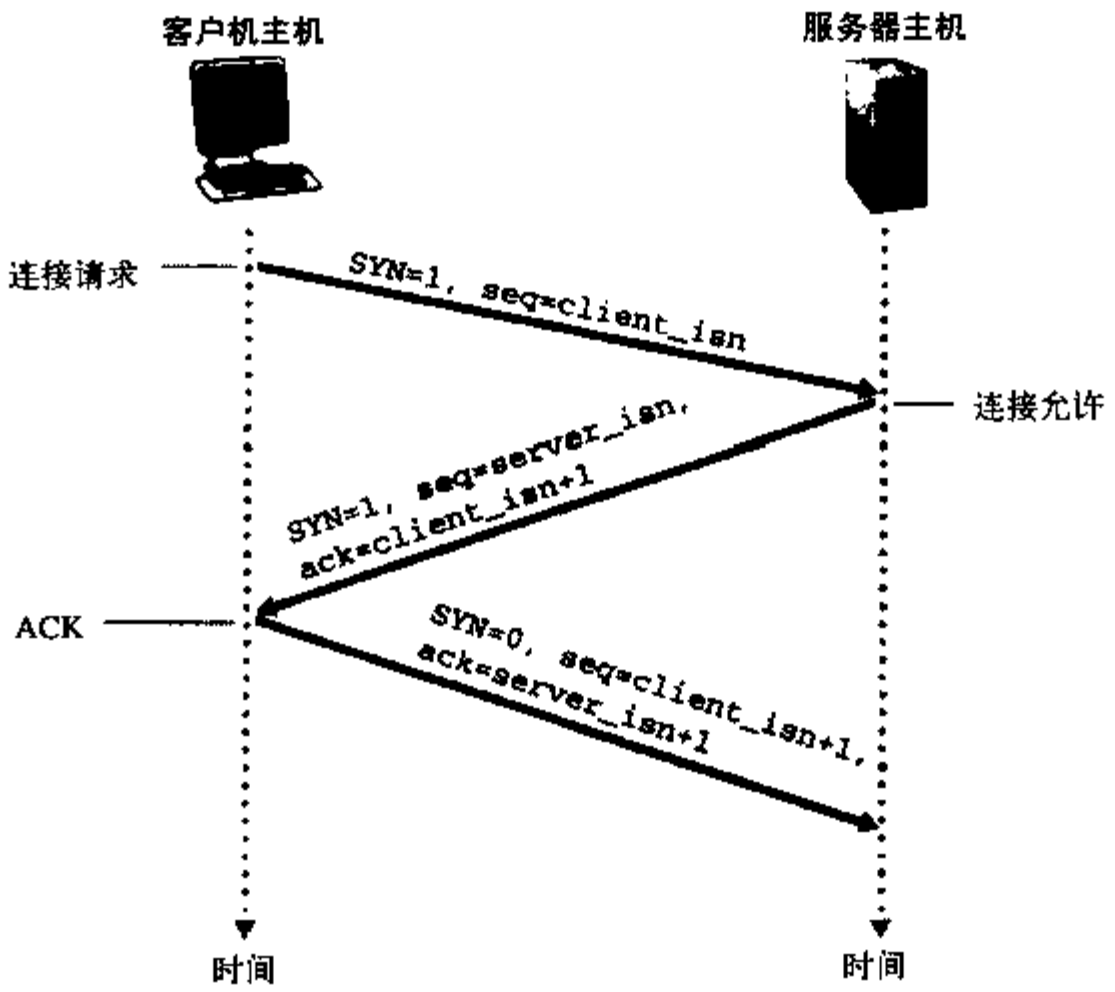


图3-39 TCP三次握手：报文段交换

天下没有不散的宴席，对于TCP连接也是这样。参与TCP连接建立的两个进程中的任何一个都能终止该连接。当连接结束后，主机中使用的“资源”（即缓存和变量）将被释放。举一个例子，假设客户机打算关闭连接，如图3-40所示。客户机应用进程发出一个关闭连接命令。这会引起客户机TCP向服务器进程发送一个特殊的TCP报文段。这个特殊的报文段首部中的一个标志比特，即FIN比特（参见图3-29）将被设置为1。当服务器接收到该报文段后，就向客户机回送一个确认报文段。然后，服务器发送其终止报文段，其FIN比特被置为1。最后，该客户机对这个服务器的终止报文段进行确认。此时，两台主机上用于该连接的所有资源都被释放了。

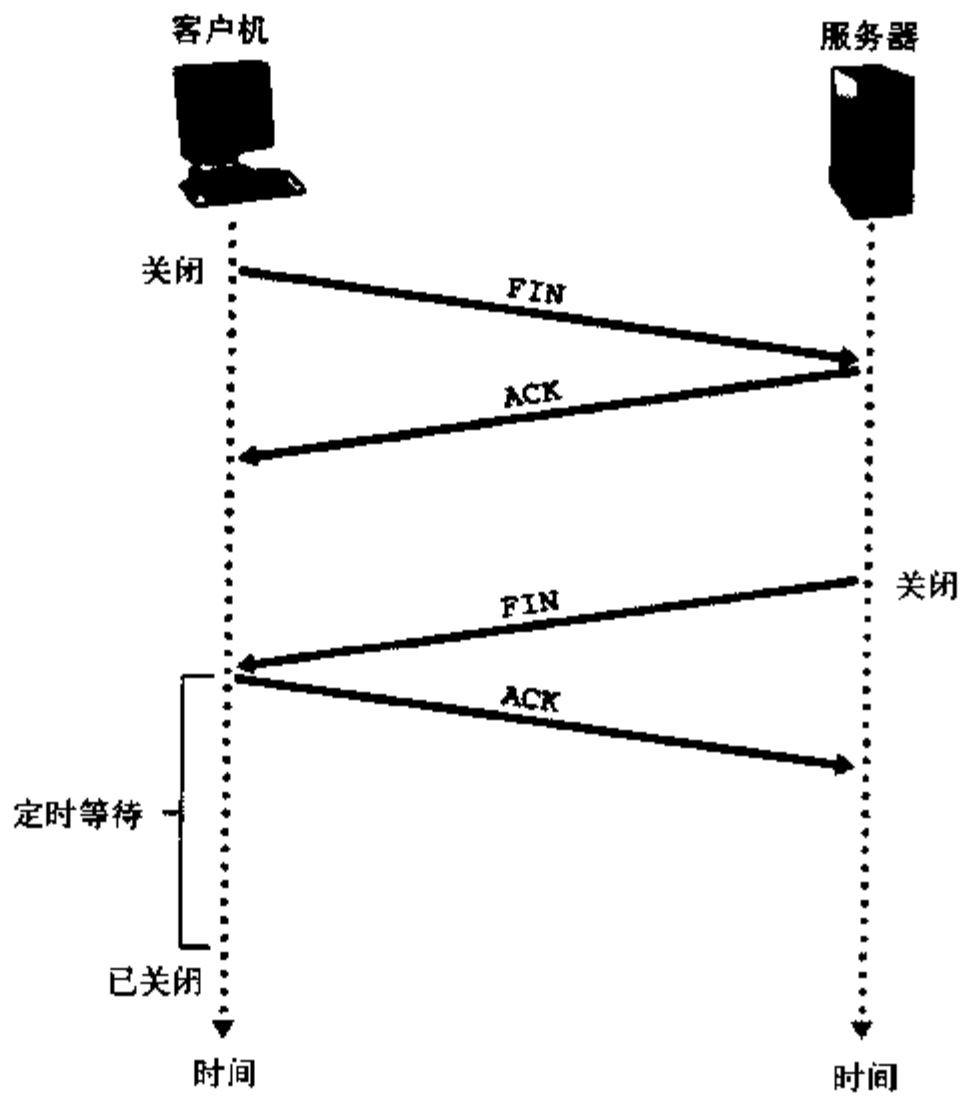


图3-40 关闭一条TCP连接

在一个TCP连接的生命周期内，运行在每台主机中的TCP协议在各种TCP状态（TCP state）之间变迁。图3-41说明了客户机TCP会经历的一系列典型TCP状态。客户机TCP开始时处于CLOSED（关闭）状态。客户机的应用程序发起一个新的TCP连接（可通过在第2章讲过的Java例子中创建一个Socket对象来完成）。这引起客户机中的TCP向服务器中的TCP发送一个

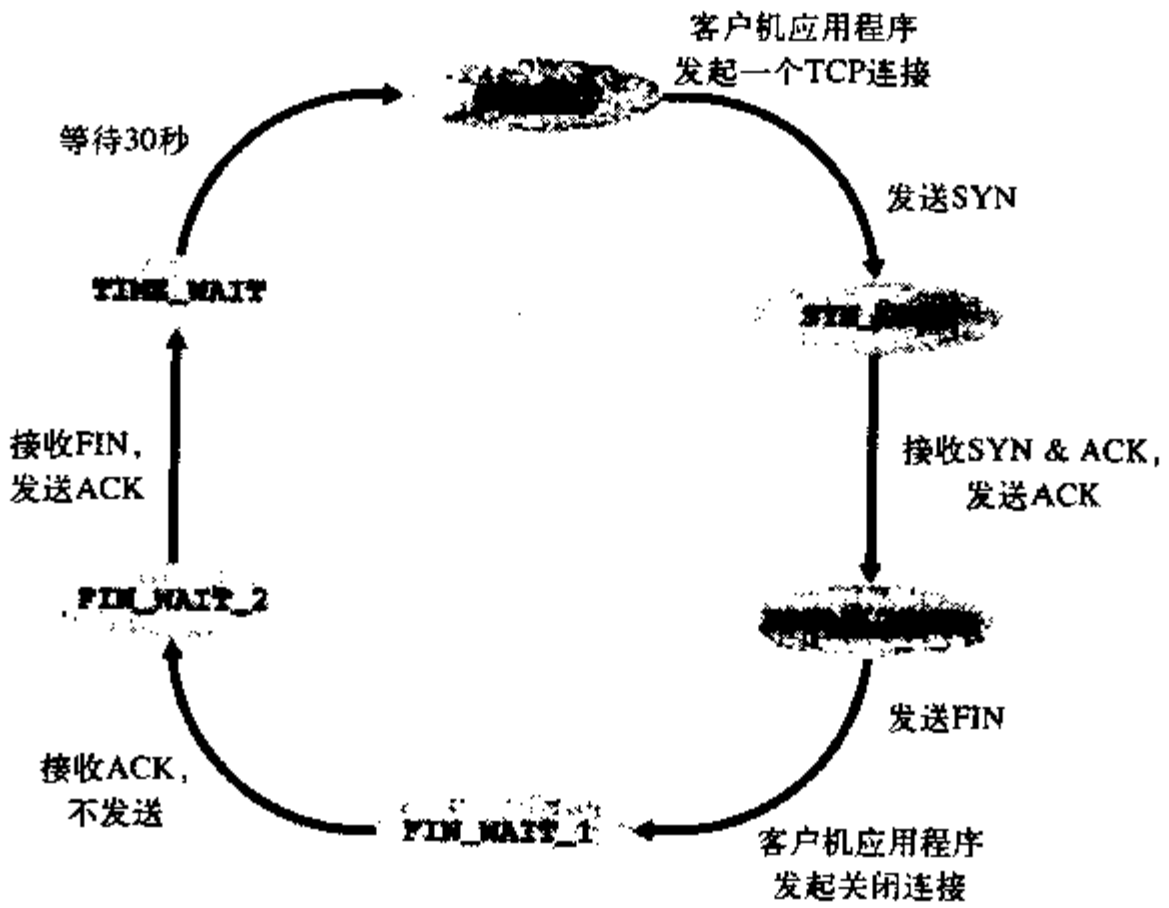


图3-41 客户机TCP经历的典型的TCP状态序列

SYN报文段。客户机在发送过SYN报文段后，进入SYN_SENT（同步发送）状态。当客户机TCP处在SYN_SENT状态时，等待来自服务器TCP对客户机所发报文段进行确认的且SYN比特被置为1的一个报文段。收到这样一个报文段之后，客户机TCP进入ESTABLISHED（已建立）状态。当处在ESTABLISHED状态时，TCP客户机就能发送和接收包含有效载荷数据（即应用层产生的数据）的TCP报文段了。

假设客户机应用程序决定要关闭该连接。（注意，服务器也能选择关闭该连接。）这引起客户机TCP发送一个FIN比特被置为1的TCP报文段，并进入FIN_WAIT_1状态。当处在FIN_WAIT_1状态时，客户机TCP等待一个来自服务器的带有确认信息的TCP报文段。当它收到该报文段时，客户机TCP进入FIN_WAIT_2状态。当处在FIN_WAIT_2状态时，客户机等待来自服务器的FIN比特被置为1的另一个报文段，收到该报文段后，客户机TCP对服务器的报文段进行确认，并进入TIME_WAIT状态。TIME_WAIT状态使得TCP客户机重传最终确认报文，以防该ACK丢失。在TIME_WAIT状态中所消耗的时间是与具体实现有关的，一般是30秒、1分钟或2分钟。经过等待后，连接就正式关闭，客户机端所有与连接有关的资源（包括端口号）将被释放。

关注安全

SYN洪泛攻击

我们在前面TCP三次握手的讨论中已经看到，服务器为了响应一个收到的SYN，分配并初始化连接变量和缓存。然后服务器发送一个SYNACK进行响应，并等待来自客户机的ACK报文段，这是建立一条连接之前的第三步也是最后一步。如果某客户机不发送ACK来完成三次握手的第三步，最终（通常在一分钟之后）服务器将终止该半开连接并回收已分配的资源。

这种TCP连接管理协议为经典的DoS攻击，即**SYN洪泛攻击**（SYN flood attack）搭建了舞台。在这种攻击中，攻击者发送大量的TCP SYN报文段，而不完成三次握手的第三步。通过从多个源发送SYN能够加大攻击力度，产生DDoS（分布式拒绝服务）SYN洪泛攻击。随着这种SYN报文段的纷至沓来，服务器不断地为这些半开连接分配资源，结果导致该服务器的连接资源迅速地消耗殆尽。这种SYN洪泛攻击[CERT SYN 1996]是CERT记载的众多DoS攻击中的第一种[CERT 2007]。

SYN洪泛是一种毁灭性的攻击。幸运的是，现在已有一种有效的防御系统，称为SYN cookies[Skoudis 2006; Cisco SYN 2007; Bernstein 2007]，它被部署在大多数主流操作系统中。SYN cookies的工作原理如下：

- 当服务器接收到一个SYN报文段时，它并不知道该报文段是来自一个合法的用户，还是一个SYN洪泛攻击的一部分。因此服务器不会为该报文段生成一个半开TCP连接。相反，服务器生成一个初始TCP序列号，该序列号是SYN报文段的源和目的IP地址、端口号以及仅被该服务器所知的秘密数的一个复杂函数（散列函数）。（该服务器对大量的连接使用相同的秘密数。）这种精心制作的初始序列号被称为“cookie”。接下来，服务器发送具有这种特殊初始序列号的SYNACK分组。更重要的是，服务器并不记忆该cookie或任何对应于SYN的其他状态信息。
- 如果客户机是合法的，则它将返回一个ACK报文段。服务器一旦收到该ACK，需

要验证与前面发送的某些SYN对应的ACK。如果服务器没有维护有关SYN报文段的信息，这是怎样完成的呢？正如你可能猜测的那样，它是借助于cookie来做到的。特别是，对于一个合法的ACK，确认字段中的值等于SYNACK字段中的值加1（参见图3-39）。服务器将使用在ACK报文段中的相同字段和秘密数运行相同的函数。如果该函数的结果加1与确认号相同的话，服务器就认为该ACK对应于前面发送的SYN报文段，因此它是合法的。这样，服务器就生成一个具有套接字的全开的连接。

- 另一方面，如果客户机没有返回一个ACK报文段，则初始的SYN也没有对该服务器产生危害，因为服务器没有为它分配任何资源。

SYN cookies有效地消除了SYN洪泛攻击的危害。一种SYN洪泛攻击的变种是让怀有恶意的客户机对服务器产生的每个SYNACK报文段返回一个合法的ACK报文段。这将使得服务器创建全开的TCP连接，即使其操作系统应用了SYN cookies。如果数以万计的客户机正被（DDoS攻击）使用，且每个客户机都有不同的源IP地址，则服务器很难区分合法的源和恶意的源。因此，防御这种“完全握手攻击”比防御经典的SYN洪泛攻击更为困难。

图3-42说明了服务器端TCP通常要经历的一系列状态，其中假设该客户机开始连接拆除。这些状态变迁是自解释的。在这两个状态变迁图中，我们只给出了TCP连接是如何正常地被建立和拆除的，而没有描述在某些不正常的情况下发生的事情，例如当连接的双方同时发起或终止一条连接时。如果你对此问题及其他与TCP有关的高级问题感兴趣，可参见Stevens所著的内容更全面的书籍[Stevens 1994]。

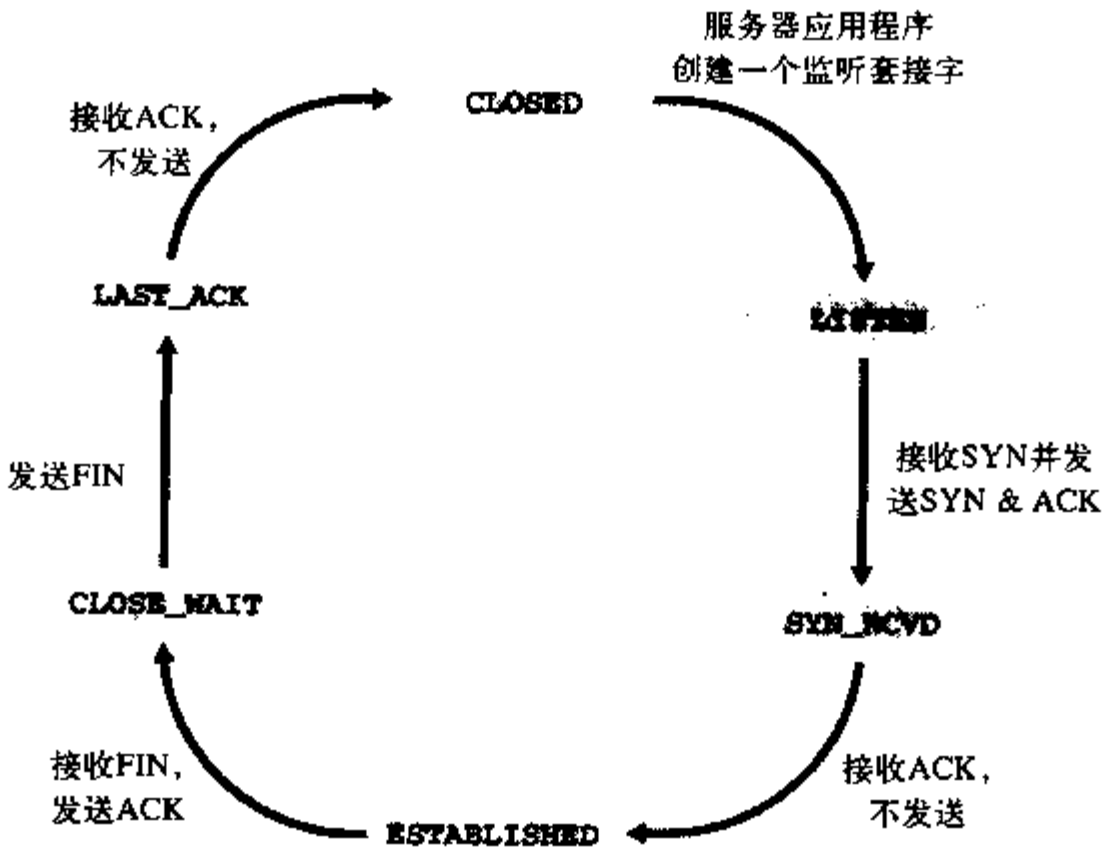


图3-42 服务器TCP经历的典型TCP状态序列

上面的讨论假定了客户机和服务器都准备通信，即服务器正在监听客户机发送它的SYN报文段的端口。我们来考虑当一台主机接收到一个TCP报文段，其端口号或源IP地址与该主机上的进行中的套接字都不匹配的情况。例如，假如一台主机接收了具有目的端口80的一个TCP SYN分组，但该主机在端口80不接收连接（即它不在端口80上运行Web服务器），则该主

机将向源发送一个特殊重置报文段。该TCP报文段将RST标志位（参见3.5.2节）置为1。因此，当主机发送一个重置报文段时，它告诉该源“我没有那个报文段的套接字。请不要再发送该报文段了。”当一台主机接收一个UDP分组，它的目的端口与进行中的UDP套接字不匹配时，该主机发送一个特殊的ICMP数据报，这将在第4章中讨论。

既然我们已经对TCP连接管理有了深入的了解，我们再次回顾nmap端口扫描工具，并更为详细地研究其工作原理。为了考察以太网目标主机上的一个特定的TCP端口，如端口6789，nmap将对那台主机发送一个特殊的TCP SYN报文段。这将会出现以下3种结果：

- 源主机从目标主机收到一个TCP SYNACK报文段。这意味着在目标主机上一个应用程序使用TCP端口6789运行，nmap返回“打开”。
- 源主机从目标主机接收到一个TCP RST报文段。这意味着该SYN报文段到达了目标主机，而目标主机上的TCP端口6789没有应用程序运行。但攻击者至少知道发向主机端口6789的报文段没有被源主机和目标主机之间的任何防火墙所阻挡。（将在第8章中讨论防火墙。）
- 源主机什么也没有收到。这很可能表明该SYN报文段被中间的防火墙所阻挡，不可能到达目标主机。

nmap是一个强大的工具，该工具不仅能“侦察”打开的TCP端口，而且能“侦察”打开的UDP端口，还能“侦察”防火墙及其配置，甚至能“侦察”应用程序和操作系统的版本。其中，大多数任务都能通过操作TCP连接管理报文段完成[Skoudis 2006]。如果你恰好使用的是Linux，就可以在命令行直接键入“nmap”让nmap运行起来。读者可以从http://insecure.org/nmap下载用于其他操作系统的nmap。

至此，我们介绍完了TCP中的差错控制和流量控制。在3.7节中，我们将回到TCP并且更深入地研究TCP拥塞控制问题。然而，在此之前，我们先后退一步，在更广泛情况下讨论拥塞控制问题。

3.6 拥塞控制原理

在前面几节中，我们已经分析了遇到分组丢失时用于提供可靠数据传输服务的基本原理及特定的TCP机制。我们以前讨论过，在实际中，这样的丢失一般是在网络变得拥塞时由于路由器缓存溢出引起的。因此，分组重传作为网络拥塞的征兆（某个特定运输层报文段的丢失），但是却不能解决网络拥塞问题（因为有太多的源主机想以过高的速率发送数据）。要解决网络拥塞原因，需要一些机制在面临网络拥塞时遏制发送方。

在本节中，我们考虑在一般情况下网络拥塞的控制问题，努力去弄懂为什么网络拥塞是一件“坏事情”，网络拥塞是如何在上层应用得到的服务性能中明确地显露出来的，如何用各种方法来避免网络拥塞或对它作出反应。这种对拥塞控制的更一般研究是恰当的，因为就像可靠数据传输一样，它在网络技术中的前10个基础性重要问题清单中位居前列。通过对**异步传送模式**（ATM）网络的**可用比特率**（ABR）服务中的拥塞控制进行讨论来结束本节。下一节详细研究TCP的拥塞控制算法。

3.6.1 拥塞原因与开销

我们通过分析3个复杂性越来越高的发生拥塞的情况，来开始对拥塞控制的一般性研究。在每一种情况下，我们首先将看看出现拥塞的原因以及拥塞的代价（根据资源未被充分利用

以及端系统得到的低劣服务性能）。我们暂不关注如何对拥塞作出反应或避免拥塞，而是关注随着主机增加其传输速率，使得网络变得拥塞时会发生的情况，这是个较简单的问题。

1. 情况1：两个发送方和一个具有无穷大缓存的路由器

我们先考虑也许是最简单的拥塞情况，即两台主机（A和B）都有一个连接，且这两个连接共享源与目的地之间的单跳路由，如图3-43所示。

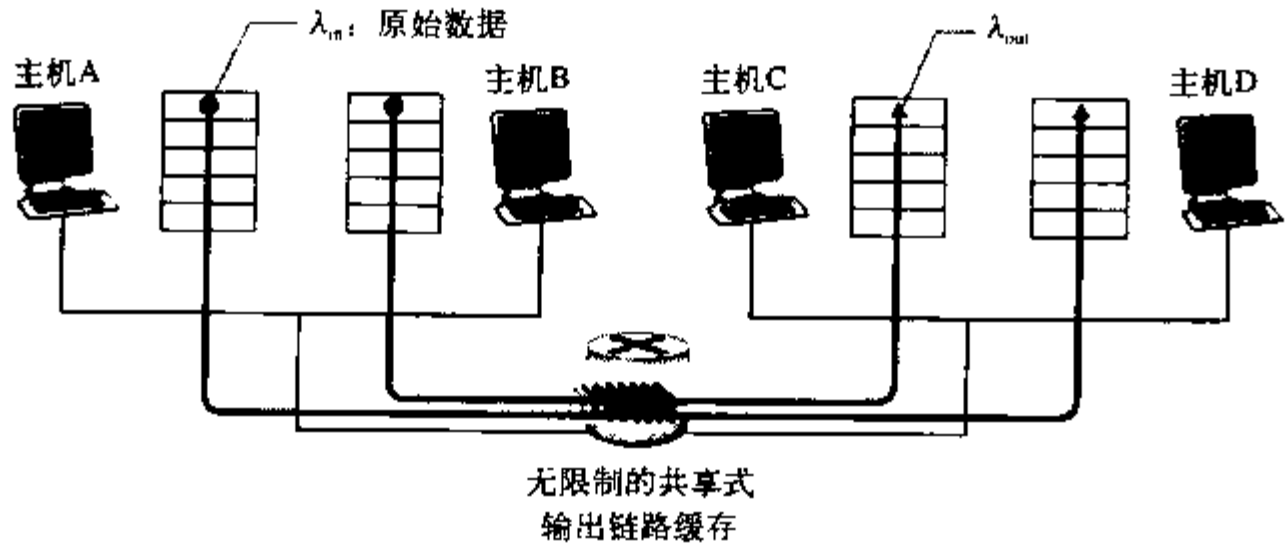


图3-43 拥塞情况1：共享单跳具有无限大缓存的两个连接

假设主机A中的应用程序以平均速率λ_{in}字节/秒将数据发送到连接中（例如，通过一个套接字将数据传递给运输层）。这些数据都是原始数据，意味着每个数据单元仅向套接字中发送一次。下面的运输层协议是一个较简单的协议。数据被封装发送，不执行差错恢复（如重传）、流量控制或拥塞控制。忽略由于添加运输层和较低层首部信息而引起的额外开销，因此在第一种情况下，主机A向路由器提供流量的速率是λ_{in}字节/秒。主机B也以同样的方式运行，为了简化问题，我们假设它也是以速率λ_{in}字节/秒发送数据。来自主机A和主机B的分组通过一台路由器，在一段容量为R的共享式输出链路上传输。该路由器有缓存，可用于当分组到达速率超过该输出链路的容量时存储输入的分组。在此第一种情况下，我们将假设路由器有无限量的缓存空间。

图3-44描绘出了在第一种情况下主机A的连接性能。左边的图形描绘了每连接的吞吐量（per-connection throughput）（接收方每秒接收的字节数）与该连接发送速率之间的函数关系。当发送速率在0～$R/2$之间时，接收方的吞吐量等于发送方的发送速率，即发送方发送的所有数据经一定时延后到达接收方。然而当发送速率超过$R/2$时，它的吞吐量只能达$R/2$。这个吞吐量上限是由两个连接之间对链路容量的共享造成的。链路不能以超过$R/2$的稳定状态速率向接收方交付分组。无论主机A和主机B将其发送速率设置为多高，它们都不会看到超过$R/2$的吞吐量。

实际中，取得每连接$R/2$的吞吐量可能看起来是件好事，因为在将分组交付到目的地的过程中链路被充分利用了。但是，图3-44右侧的图形却显示了以接近链路容量的速率运行时产生的后果。当发送速率接近$R/2$时（从左至右），平均时延会越来越大。当发送速率超过$R/2$时，路由器中的平均排队分组数就会无限增长，源与目的地之间的平均时延也会变成无穷大（假设这些连接以此发送速率运行无限长时间并且有无限量的缓存可用）。因此，虽然从吞吐量角度讲，运行在总吞吐量接近R的状态也许是一个理想状态，但是从时延角度看，却远不是一个理想状态。甚至在这种（极端）理想化的情况中，我们已经发现了拥塞网络的一种开销，即当分组到达速率接近链路容量时，分组经历的巨大排队时延。

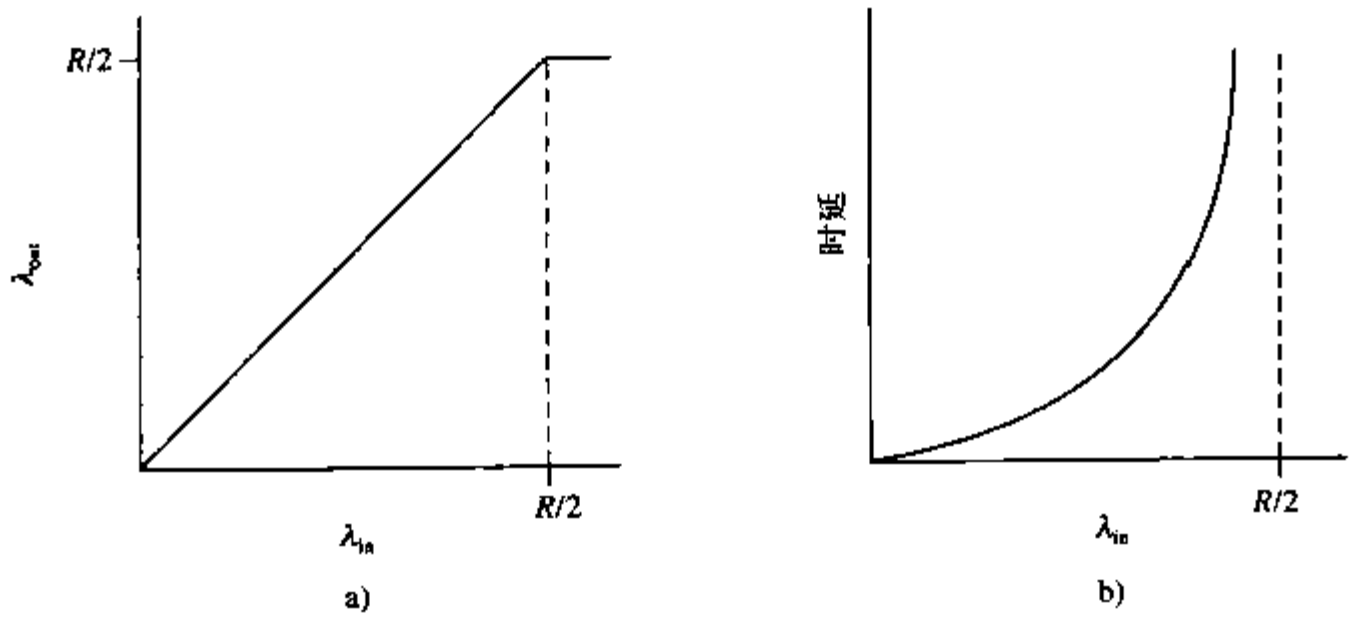


图3-44 拥塞情况1：吞吐量、时延与主机发送速率之间的函数关系

2. 情况2：两个发送方和一个具有有限缓存的路由器

现在我们从下列两个方面对情况1稍微作一些修改（参见图3-45）。首先，假定路由器缓存的容量是有限的。这种假设接近实际的情况，其结果是当分组到达一个已满的缓存时会被丢弃。其次，假定每个连接都是可靠的。如果一个包含有运输层报文段的分组在路由器中被丢弃，那么它终将被发送方重传。由于分组可以被重传，所以我们现在必须更谨慎地使用术语发送速率。特别是，我们再次以λ_{in}字节/秒表示应用程序将初始数据发送到套接字中的速率。运输层向网络中发送报文段（含有初始数据或重传数据）的速率用λ'_{in}字节/秒表示。λ'_{in}有时被称为对网络的**供给载荷**（offered load）。

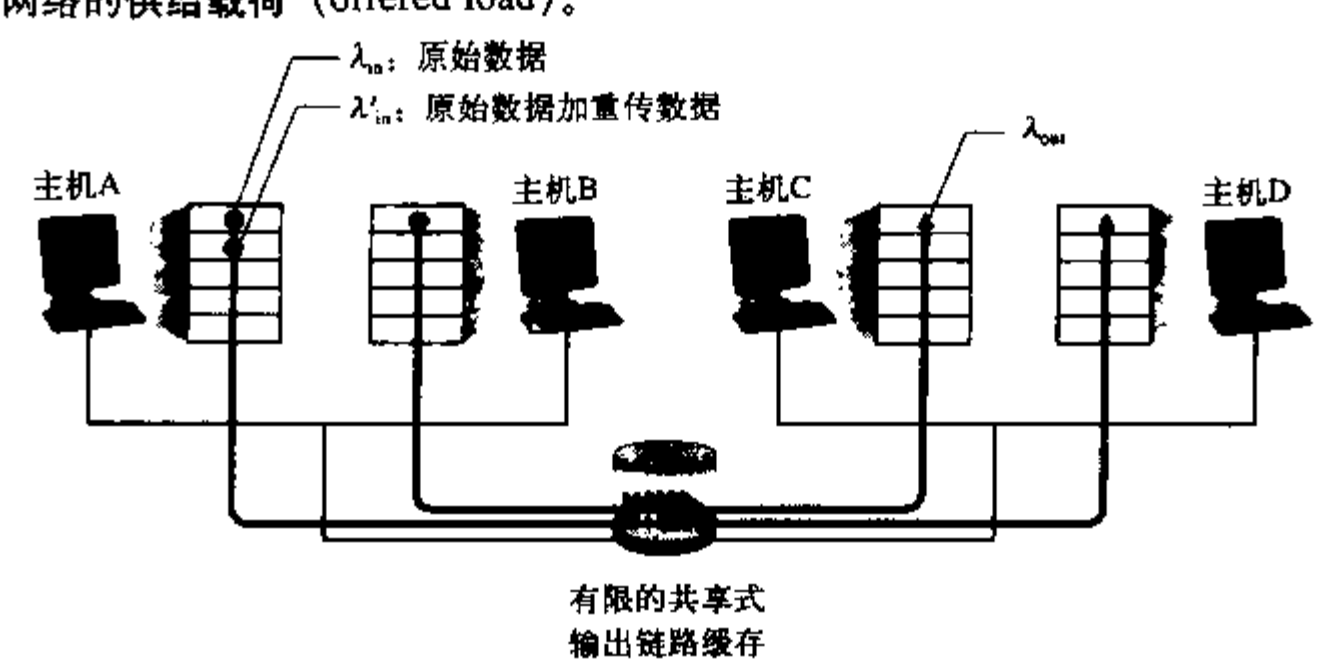


图3-45 情况2：（有重传的）两台主机与一台拥有有限缓存的路由器

在情况2下实现的性能将在很大程度上取决于如何进行重传。首先，我们考虑一种不切实际的情况，即主机A能够以某种方式（不可思议地！）确定路由器中的缓存是否空闲，因而仅当缓存空闲时才发送一个分组。在这种情况下，不会产生丢包，λ_{in}与λ'_{in}相等，连接的吞吐量就等于λ_{in}。图3-46a中描述了这种情况。从吞吐量的角度来看，性能是理想的，即发送的每个分组都接收到。注意到在这种情况下，平均主机发送速率不能超过$R/2$，因为假定不发生分组丢失。

下面我们考虑一种更实际的情况，发送方仅当在确定了一个分组已经丢失时才重传。（我们对所做的假设再做一些扩展。然而，发送主机也有可能将超时时间设置得足够长，以虚拟

地确保一个已经丢失的分组还没有被确认。）在这种情况下，性能就可能与图3-46b所示的情况相似。为了理解这里发生的情况，考虑一下供给载荷λ'_{in}（原始数据传输加上重传的总速率）等于$R/2$的情况。根据图3-46b，在这一载荷值时，数据被交付到接收方应用程序的速率是$R/3$。因此，在所传输的$0.5R$单位数据中，平均来看，$0.333R$字节/秒是原始数据，而$0.166R$字节/秒是重传数据。*我们在此看到了另一种网络拥塞开销，即发送方必须执行重传以补偿因为缓存溢出而丢弃（丢失）的分组。*

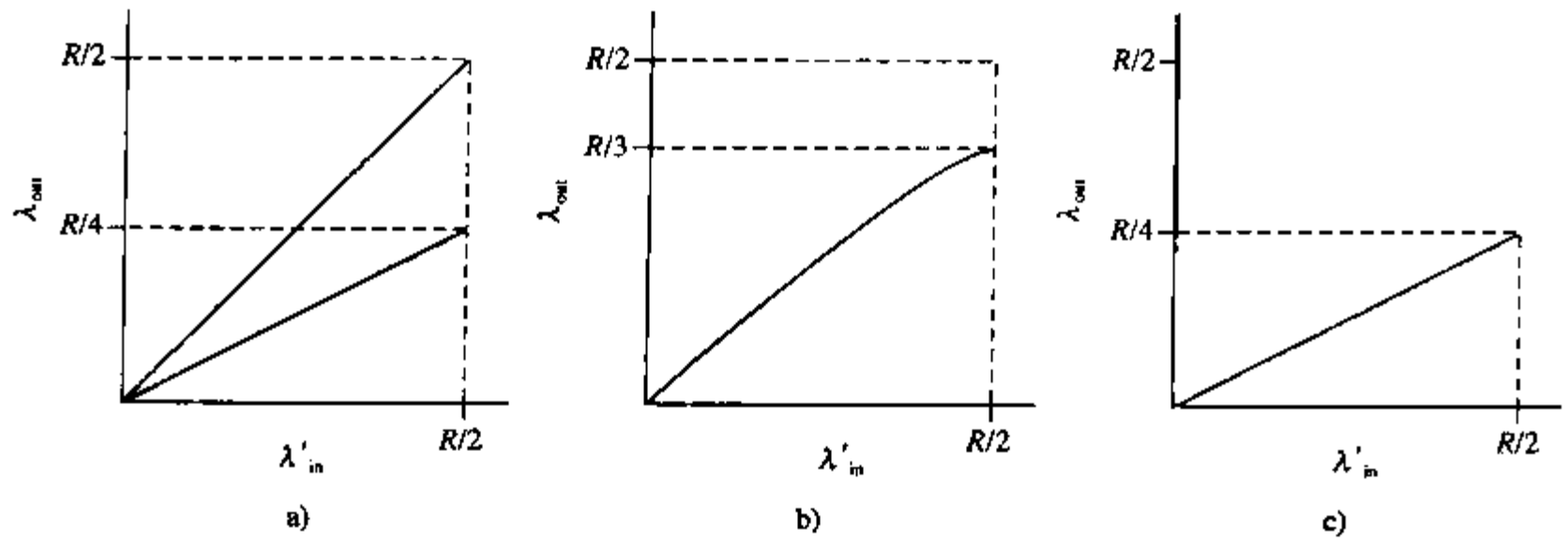


图3-46 情况2中具有有限缓存时的性能

最后，我们考虑一种情况，发送方也许会提前发生超时并重传在队列中已推迟但还未丢失的分组。在这种情况下，原始数据分组和重传分组都可能到达接收方。当然，接收方只需要一份这样的分组拷贝就行了，重传分组将被丢弃。在这种情况下，路由器转发重传的原始分组拷贝是在做无用功，因为接收方已收到了该分组的原始版本。路由器本可以利用链路的传输能力去发送另一个分组。*这里，我们又看到了网络拥塞的另一种开销，即发送方在遇到大时延时所进行的不必要重传会引起路由器利用其链路带宽来转发不必要的分组拷贝。*图3-46c中下方那条曲线显示了当假定每个分组被路由器转发（平均）两次时吞吐量与供给载荷的对比情况。由于每个分组被转发两次，当供给载荷接近$R/2$时，其吞吐量将有$R/4$的渐近值。

3. *情况3：四个发送方、具有有限缓存的多台路由器和多跳路径*

在最后一种拥塞情况中，有4台主机传输分组，每台都是在交叠的两跳路径上传输，如图3-47所示。我们再次假设每台主机都采用超时/重传机制来实现可靠的数据传输服务，所有的主机都有相同的λ_{in}值，所有路由器的链路容量都是R字节/秒。

我们考虑从主机A到主机C的连接，该连接经过路由器R1和R2。A-C连接与D-B连接共享路由器R1，并与B-D连接共享路由器R2。当λ_{in}非常小时，路由器的缓存溢出是很少见的（和拥塞情况1、拥塞情况2中的一样），吞吐量接近供给载荷。当λ_{in}稍微增大时，吞吐量也相应地增大，当更多的原始数据传输到网络中并交付到目的地时，溢出仍然是很少见的。因此，在λ_{in}较小时，λ_{in}的增大会导致λ_{out}的增大。

在考虑了流量很小的情况后，我们下一步分析一下λ_{in}（还有λ'_{in}）很大时的情况。考虑路由器R2。不管λ_{in}的值是多大，到达路由器R2的A-C流量（在经过路由器R1转发后到达路由器R2）的到达速率至多是R，也就是从R1到R2的链路容量。如果λ'_{in}对于所有连接（包括B-D连接）来说是极大的值，那么在R2上B-D流量的到达速率可能会比A-C流量的到达速率大得多。由于A-C流量与B-D流量在路由器R2上必须为有限缓存空间而竞争，所以当来自B-D连接的供

给载荷越来越大时，A-C连接上成功通过R2（即没有由于缓存溢出而丢失）的流量会越来越小。在极限情况下，当供给载荷趋近于无穷大时，R2的空闲缓存会立即被B-D连接的分组占满，因而A-C连接在R2上的吞吐量趋近于0。这就说明在重载情况下，A-C端到端吞吐量将趋近于0。这些考虑引发了供给载荷与吞吐量之间的权衡，如图3-48所示。

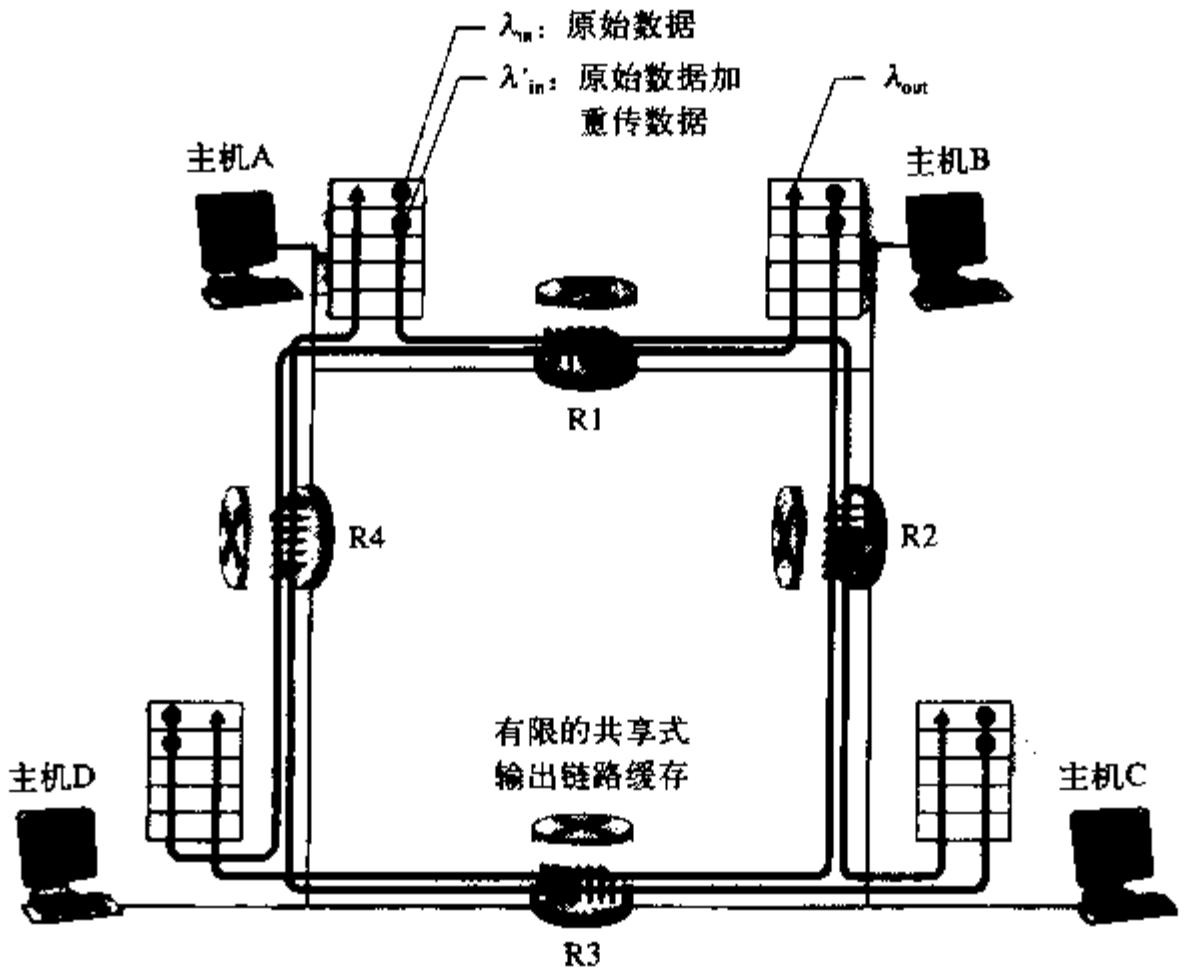


图3-47 四个发送方，具有有限缓存的路由器和多跳路径

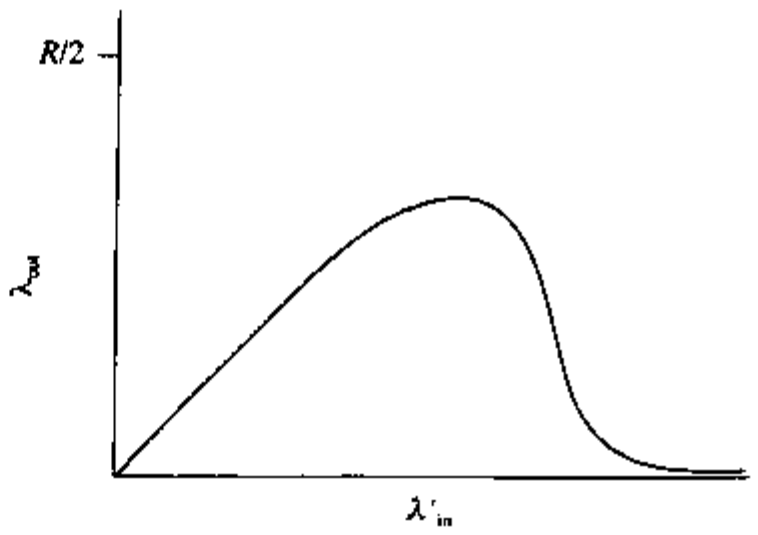


图3-48 具有有限缓存和多跳路径时的情况3性能

如果我们考虑一下网络所做的无用功的数量，就会清楚地知道最终吞吐量会随着供给载荷的增加而减少。在上面描述的大流量情况中，每当有一个分组在第二跳路由器上被丢弃时，第一跳路由器所做的将分组转发到第二跳路由器的工作就是无用的。如果第一跳路由器只是丢弃该分组并保持空闲，则网络中的情况将是同样糟糕。需要指出的是，第一跳路由器所使用的将分组转发到第二跳路由器的传输容量用来传送其他分组可能更有益。（例如，当选择一个分组进行传输时，路由器最好优先考虑那些已历经过一定数量的上游路由器的分组。）所以，我们在此又看到了由于拥塞而丢弃分组的另一种开销，即当一个分组沿一条路径被丢弃时，每个上游路由器用于转发该分组到丢弃该分组而使用的传输容量最终被浪费掉了。

3.6.2 拥塞控制方法

在3.7节中，我们将详细研究TCP用于拥塞控制的特定方法。这里，我们学习实际中所采用的两种主要拥塞控制方法，讨论特定的网络体系结构和使这些方法具体化的拥塞控制协议。

在最宽泛的层次上，我们可根据网络层是否为运输层拥塞控制提供了显式的帮助来区分拥塞控制方法。

- *端到端拥塞控制。*在端到端拥塞控制方法中，网络层没有为运输层拥塞控制提供显式支持。即使在网络中存在拥塞，端系统也必须通过对网络行为的观察（如分组丢失与时延）来推断。我们在3.7节中将看到，TCP必须通过端到端的方法处理拥塞控制，因为IP层不会向端系统提供有关网络拥塞的反馈信息。TCP报文段的丢失（通过超时或3次冗余确认而得知）被认为是网络拥塞的一个迹象，TCP会相应地减小其窗口长度。我们还将看到关于TCP的一些新建议，即TCP通过增加往返时延值作为网络拥塞程度增加的指示。
- *网络辅助的拥塞控制。*在网络辅助的拥塞控制中，网络层组件（即路由器）向发送方提供关于网络中拥塞状态的显式反馈信息。这种反馈可以通过仅用一个比特来指示链路中的拥塞情况。这种方法在早期的IBM SNA [Schwartz 1982]和DEC DECnet [Jain 1989; Ramakrishnan 1990]等体系结构中采用，近来建议将其用于TCP/IP网络 [Floyd TCP 1994; RFC 3168]，而且还用在我们下面要讨论的ATM可用比特率（ABR）拥塞控制中。提供更复杂的网络反馈也是可能的。例如，我们很快将学习的一种ATM ABR拥塞控制，其允许路由器显式地通知发送方，它（路由器）能在输出链路上支持的传输速率。

对于网络辅助的拥塞控制，拥塞信息从网络反馈到发送方通常有两种方式，如图3-49所示。直接反馈信息可以由网络路由器发给发送方。这种方式的通知通常采用一种阻塞分组（choke packet）的形式。（主要是说："我拥塞了！"）另一种形式的通知是，路由器标记或更新从发送方流向接收方的分组中的某个字段来指示拥塞的产生。一旦接收方收到这个有拥塞标记的分组，就会通知发送方网络发生了拥塞。注意，后一种形式的通知至少要经过一个完整的往返时间。

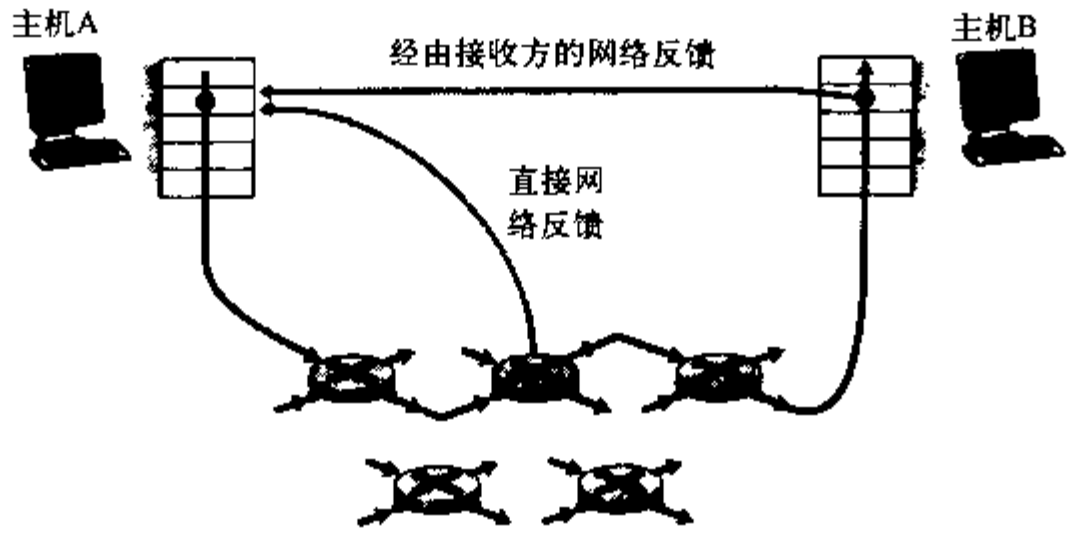


图3-49 网络指示拥塞信息的两条反馈路径

3.6.3 网络辅助的拥塞控制例子：ATM ABR拥塞控制

我们现在用一个对ATM ABR中采用的辅助拥塞控制算法的简要案例研究来概括本节，ATM ABR是一种采用网络辅助方法解决拥塞控制的协议。这里不是详细地描述ATM体系结构的方方面面，而是为了说明拥塞控制所采用的明显不同于因特网TCP协议的协议。事实上，我们下面仅给出为了理解ABR拥塞控制所需要的ATM体系结构的几个方面。

ATM基本上采用一种面向虚电路（VC）方法来处理分组交换问题。回想我们在第1章中的讨论，这意味着从源到目的地路径上的每台交换机将维护有关源到目的地VC的状态。这种

逐个VC的状态允许交换机跟踪各个发送方的行为（例如，跟踪它们的平均传输速率），并采取源特定的拥塞控制动作（例如，当交换机变得拥塞时，向发送方发显式信令以减少它的速率）。在网络交换机上的这种VC的每个状态使得ATM非常适合执行网络辅助拥塞控制。

ABR已被设计成一种弹性数据传输服务，该服务方式使人联想起TCP。当网络轻载时，ABR服务会充分利用空闲的可用带宽；当网络拥塞时，ABR服务会将其传输速率抑制为某些预先确定的最小传输速率。[Jain 1996]中提供了一个关于ATM ABR拥塞控制与流量管理的详细学习指南。

图3-50给出了ATM ABR拥塞控制框架。在下面的讨论中，我们将采用ATM的术语（如使用术语交换机而不使用路由器；使用术语信元（cell）而不使用分组）。对于ATM ABR服务，数据信元从源经过一系列中间交换机传输到目的地。在数据信元中夹杂着所谓的**资源管理信元**（Resource-Management cell, RM cell）；这些RM信元可用来在主机和交换机之间传递与拥塞相关的信息。当一个RM信元到达目的地时，它将被掉转方向回送给发送方（很可能是在目的地修改了该RM信元的内容之后）。交换机也有可能自己产生一个RM信元，并将该RM信元直接发送给源。因此，RM信元可用来提供直接网络反馈和经由接收方的网络反馈，如图3-50所示。

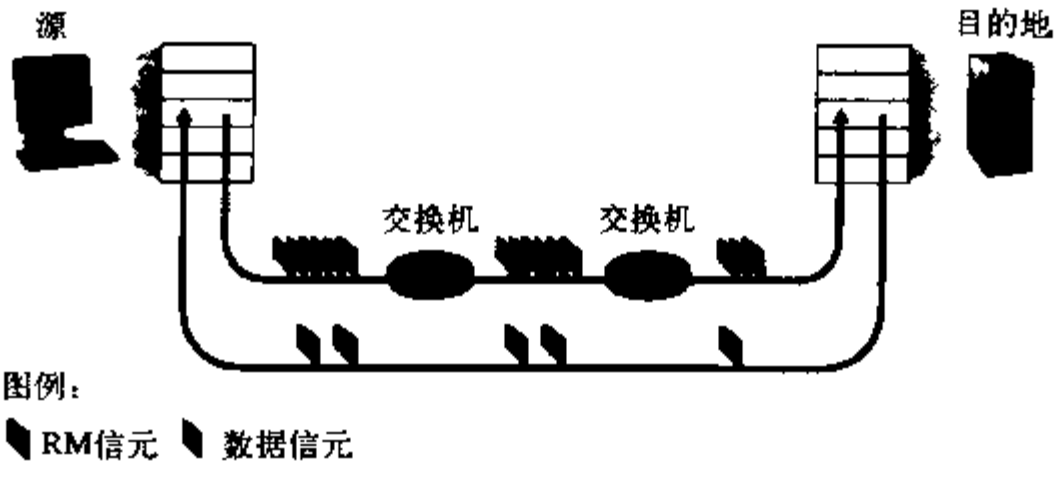


图3-50 ATM ABR服务的拥塞控制框架

ATM ABR拥塞控制是一种基于速率的方法，即发送方明确地计算出它所能发送的最大速率，并据此对自己进行相应的调整。ABR提供以下三种机制用于从交换机向接收方发送与拥塞相关的信令信息：

- EFCI比特。每个数据信元都包含1比特的**显式转发拥塞指示**（Explicit Forward Congestion Indication, EFCI）比特。一个拥塞的网络交换机可把数据信元中的EFCI比特设置为1来向目的主机发送网络已经拥塞的信令。目的地必须检查所有收到的数据信元中的EFCI比特。当一个RM信元到达目的地时，如果多数近来收到的数据信元的EFCI比特都被置为1，则目的地就会将RM信元的拥塞指示比特（CI比特）置为1，并将该RM信元发送回发送方。使用数据信元中的EFCI比特和RM信元中的CI比特，发送方就能在网络交换机拥塞时得到通知。
- CI和NI比特。如上所述，发送方到接收方的RM信元是夹杂在数据信元当中的。RM信元的夹杂比率是一个可调参数，默认值是每32个数据信元中有一个RM信元。这些RM信元中有一个**拥塞指示**（congestion indication, CI）比特和一个**无增长**（no increase, NI）比特，这两个比特可被一台拥塞的交换机设置。特别是，交换机可以在轻微拥塞时将经过的RM信元中的NI比特置为1，在严重拥塞情况下把CI比特置为1。当目的主机收到一

个RM信元时，它将把该RM信元发回给发送方，而保持CI与NI比特不变（除了CI比特也许会因为上面描述的EFCI机制而由目的端置为1之外）。

- ER的设置。每一个RM信元还包含一个两字节的显式速率（Explicit Rate, ER）字段。一个拥塞的交换机也许会降低经过的RM信元中ER字段所包含的值。在这种方式中，ER字段将被设置为在源至目的地的路径上的所有交换机中的最小可支持速率。

ATM ABR源根据返回的RM信元中的CI、NI及ER值，来调整其能够发送信元的速率。进行速率调整的规则非常复杂而且繁琐，感兴趣的读者可以参考[Jain 1996]以得到详细信息。

3.7 TCP拥塞控制

在本节中，我们再次研究TCP。如3.5节所介绍，TCP为运行于不同主机上的两个进程之间提供了可靠传输服务。TCP的另一个关键部分就是其拥塞控制机制。如前一节所述，TCP必须使用端到端拥塞控制而不是网络辅助的拥塞控制，因为IP层不向端系统提供显式的网络拥塞反馈。

TCP采用的方法是让每一个发送方根据所感知到的网络拥塞的程度，来限制其能向连接发送流量的速率。如果一个TCP发送方感知从它到目的地之间的路径上没什么拥塞，则该TCP发送方就会增加其发送速率；如果该发送方感知在该路径上有拥塞，则该发送方就会降低其发送速率。但是这种方法提出了三个问题。首先，一个TCP发送方是如何限制它向其连接发送流量的速率的呢？其次，一个TCP发送方是如何感知从它到目的地之间的路径上存在拥塞的呢？最后，当发送方感知到端到端的拥塞时，采用什么算法来改变其发送速率呢？我们现在来研究TCP Reno拥塞控制算法中的这3个问题，该算法已在大多数现代操作系统[Padhye 2001]中使用。为了使讨论更加具体，我们假设TCP发送方在发送一个大文件。

我们首先分析一下TCP发送方是如何限制向其连接发送流量的。在3.5节中我们看到，TCP连接的每一端都由一个接收缓存、一个发送缓存和几个变量（LastByteRead、RcvWindow等）组成。TCP拥塞控制机制让连接的每一端都记录一个额外的变量，即拥塞窗口（congestion window）。拥塞窗口表示为CongWin，它对一个TCP发送方能向网络中发送流量的速率进行了限制。特别是在一个发送方中未被确认的数据量不会超过CongWin与RcvWindow中的最小值，即

```
LastByteSent-LastByteAcked≤min{CongWin, RcvWindow}
```

为了关注拥塞控制（与流量控制形成对比），我们后面假设TCP接收缓存足够大，以至于可以忽略接收窗口的限制，因此在发送方中未被确认的数据量仅受限于CongWin。

上面的约束限制了发送方中未被确认的数据量，因此间接地限制了发送方的发送速率。为了理解这一点，我们来考虑一个分组丢失和传输时延均可以忽略不计的连接。因此粗略地讲，在每一个往返时延（RTT）的起始点，上面的限制条件允许发送方向该连接发送CongWin个字节的数据，在该RTT结束时发送方接收对数据的确认报文。因此，该发送方的发送速率大概是CongWin/RTT字节/秒。通过调节CongWin的值，发送方就能调整它向连接中发送数据的速率。

我们接下来考虑TCP发送方是如何感知在它与目的地之间的路径上出现了拥塞的。我们定义一个TCP发送方的“丢包事件”为：要么出现超时，要么收到来自接收方的3个冗余ACK。（回想我们在3.5.4节图3-33的超时事件中的讨论和“快速重传”内容中对图3-33的后继修改。）当出现过度的拥塞时，这条路径上的一台（或多台）路由器的缓存会溢出，导致数据报（包

含一个TCP报文段）被丢弃。丢弃的数据报接着会引起发送方的丢失事件（要么超时要么收到3个冗余ACK），发送方就认为在发送方到接收方的路径上出现了拥塞的指示。

考虑了拥塞检测问题后，我们接下来考虑网络没有拥塞的更为乐观的情况，即没有出现丢包事件的情况。在此情况下，TCP的发送方将收到对于以前未确认报文段的确认。如我们将看到的那样，TCP将这些确认的到达作为一切正常的标志（在网络上传输的报文段正被成功地交付给目的地），并使用确认来增加拥塞窗口的长度（及其传输速率）。注意，如果确认以相当慢的速率到达（例如，如果该端到端路径具有高时延或包含一段低带宽链路），则该拥塞窗口将以相当慢的速率增加。另一方面，如果确认以高速率到达，则该拥塞窗口将会更迅速地增大。因为TCP使用确认来触发（或计时）它的拥塞窗口长度的增大，所以TCP被称为是**自计时**（self-clocking）的。

我们现在该考虑一下TCP发送方在感知到拥塞时，为调节其发送速率所采用的算法细节了。这个算法就是广受好评的**TCP拥塞控制算法**（TCP congestion control algorithm）。该算法包括三个主要部分：① 加性增（additive-increase）、乘性减（multiplicative-decrease），② 慢启动（slow start），③ 对超时事件作出反应。

1. 加性增、乘性减

TCP拥塞控制的基本思想是，当出现丢包事件时，让发送方降低其发送速率（通过减小拥塞窗口CongWin的大小）。因为通过该相同拥塞路由器的其他TCP连接也很可能出现丢包事件，所以它们也可能会通过减小其CongWin值来降低其发送速率。因此，该整体作用是让所有通过这一拥塞路由器路径的源来降低它们向网络发送数据的速率，从而减轻了拥塞路由器的拥塞程度。但是当出现丢包事件时，TCP发送方应将其拥塞窗口减少多少呢？TCP采用了一种所谓“乘性减”的方法，即每发生一次丢包事件就将当前的CongWin值减半。因此，如果当前CongWin值是20kB，且检测到一个丢包事件，则CongWin值被减半为10 kB。如果再发生一个丢包事件，则CongWin值被进一步减小到5 kB。CongWin值也许会继续减小，但是不能降到低于1个MSS。（这是对发生一个丢包事件后拥塞窗口如何变化的宏观描述。事实上，这个问题要更复杂些，我们很快就会看到。）

描述了TCP发送方在检测到拥塞时如何减小其发送速率之后，下一步自然要考虑当TCP发送方觉察到网络无拥塞时，即对前面还没有确认的数据有ACK到达时，它应该怎样来增大其发送速率。增大发送速率的基本原理是，如果没有检测到拥塞，则可能有可用（未被使用的）带宽可被该TCP连接使用。在这种情况下，TCP缓慢地增加其拥塞窗口的长度，谨慎地探测端到端路径上的额外的可用带宽。TCP发送方是这样做的，即每次它收到一个确认后就把CongWin增大一点，其目标是在每个往返时延内CongWin增加一个MSS [RFC 2581]。这可以通过几种方式完成，一种通用的方法是对于TCP发送方，无论何时一个新的确认到达，将它的CongWin增加一个MSS（MSS/CongWin）字节。例如，如果MSS是1460字节并且CongWin是14 600字节，则在一个RTT内发送10个报文段。每到达ACK（假定每个报文段一个ACK）增加1/10 MSS的拥塞窗口长度，因此在收到对所有10个报文段的确认后，拥塞窗口的值将增加了MSS，与所期望的一样。

总而言之，当TCP发送方感受到端到端路径无拥塞时就加性地增加其发送速率，当觉察到路径拥塞时（通过丢包事件）就乘性地减小其发送速率。正因如此，TCP拥塞控制算法常常被称为**加性增、乘性减**（Additive-Increase, Multiplicative-Decrease, AIMD）**算法**。TCP拥塞控制协议的线性增长阶段被称为**避免拥塞**（congestion avoidance）。CongWin值重复地经历一种升降循环，即重复地线性增长，然后又突然降至其当前值的一半（当发生丢包事件时），

这种循环使得长寿命TCP连接的CongWin变化呈锯齿形状，如图3-51所示。

2. 慢启动

当一个TCP连接开始时，CongWin的值初始置为1个MSS[RFC 3390]，这就使得初始发送速率大约为MSS/RTT。举个例子来说，如果MSS = 500字节且RTT = 200 ms，则初始发送速率大约只有20 kbps。由于对该连接来说，可用带宽可能比MSS/RTT大得多，因此仅仅线性地增加发送速率，在发送速率跳到某个可观的级别之前将会经历很长的时间，这将是件令人遗憾的事。因此，TCP发送方在初始阶段不是线性地增加其发送速率，而是以指数的速度增加，即每过一个RTT将CongWin值翻倍。TCP发送方继续以指数速度增加其发送速率，直到发生一个丢包事件为止，此时CongWin将被降为一半，然后就会像上面所讲的那样线性地增长。因此，在这个被称为慢启动（Slow Start, SS）的初始化阶段期间，TCP发送方以慢速率（因此叫慢启动）发送，但是以指数的速度快速增加其发送速率。每当一个传输的报文段被确认后，CongWin的值就增加1个MSS，从而使发送方得到指数级增长的发送速率。如图3-52所示，TCP向网络发送第一个报文段并等待一个确认。如果该报文段在丢包事件之前被确认，则TCP发送方将拥塞窗口增加一个MSS，并发送出两个最大长度的报文段。如果这两个报文段在丢包事件之前都被确认，则发送方每收到一个确认报文段就将拥塞窗口增加1个MSS，使得拥塞窗口变为4个MSS，并发送出4个最大长度的报文段。只要确认报文在丢包事件之前到达，这一过程就会继续下去。因此在慢启动阶段，每过一个RTT，CongWin的值将有效地增加一倍。

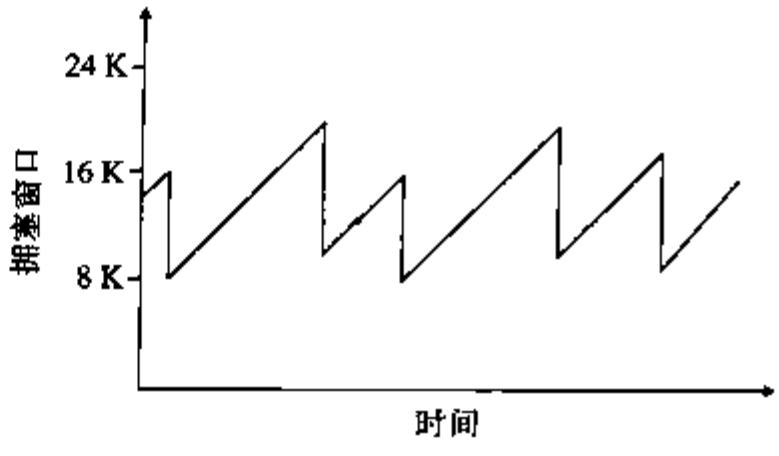


图3-51 加性增、乘性减的拥塞控制

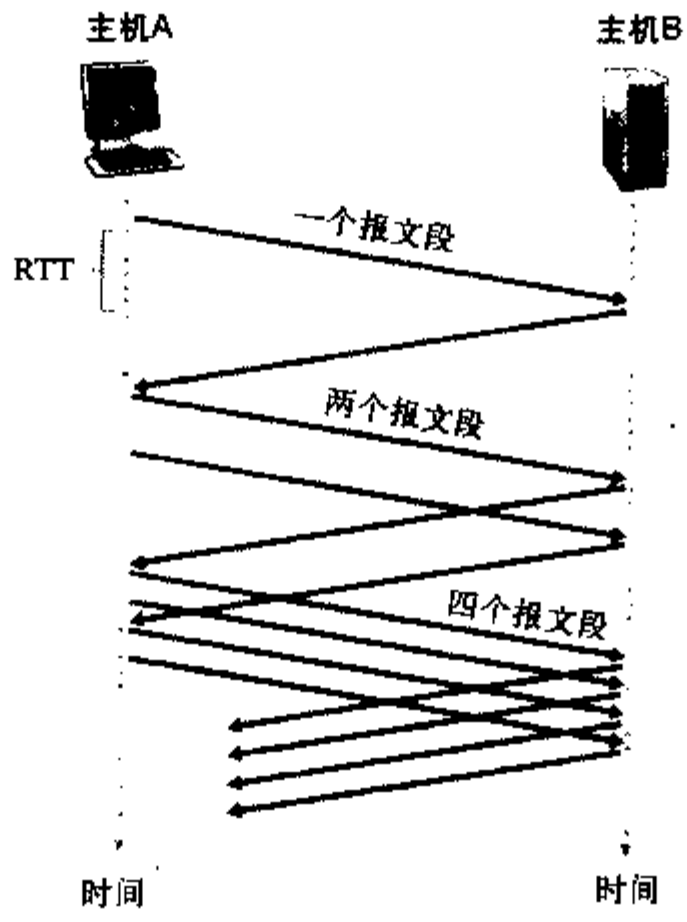


图3-52 TCP慢启动

3. 对超时事件作出反应

到目前为止，我们对TCP拥塞窗口的描述是：从一个MSS（在慢启动阶段）开始到发生丢包事件为止，拥塞窗口以指数速率迅速增加，此时，AIMD“锯齿”形状开始出现。虽然这种描述比较准确，但如果不提及实际中的TCP拥塞控制反应情况将有疏漏之嫌，即对因超时而检测到的丢包事件做出的反应与对因收到3个冗余ACK而检测到的丢包事件做出的反应是不同的。收到3个冗余ACK后，TCP的行为和我们刚描述的一样，将拥塞窗口减小一半，然后线性地增长。但是超时事件发生时，TCP发送方进入一个慢启动阶段，即它将拥塞窗口设置为1 MSS，然后窗口长度以指数速度增长。拥塞窗口持续以指数速率增长，直到CongWin达到超时事件前窗口值的一半为止。此后，CongWin以线性速率增长，就像收到3个冗余ACK一样。

TCP通过维持一个称为**阈值**（Threshold）的变量来管理这些较复杂的动态过程，它是用来确定慢启动将结束并且拥塞避免将开始的窗口长度。变量Threshold初始化时被设置为一个很大的值（实际中为65 kB[Stevens 1994]），以使它没有初始效应。每当发生一个丢包事件时，Threshold值就会被设置为当前CongWin值的一半。例如，如果在丢包事件之前拥塞窗口是20 kB，则Threshold值被设置为10 kB，并将保持该值至发生下一个丢包事件为止。

描述过Threshold变量后，我们现在就能精确地描述在发生一个超时事件后CongWin行为了。如上面所述，TCP发送方在一个超时事件发生后就进入慢启动阶段。在慢启动阶段中，CongWin值以指数速率快速增长，直至CongWin达到Threshold为止。当CongWin达到Threshold时，TCP就进入拥塞避免阶段，在该阶段中，CongWin如前所述线性地增长。

表3-3 TCP发送方拥塞控制[RFC 2581]。假定CongWin的初始值等于MSS，Threshold的初始值大（例如，65 kB[Stevens 1994]），并且TCP发送方开始进入慢启动状态。显示的状态值是正好丢包事件出现前的TCP发送方状态。详情参见[RFC 2581]

状 态	事 件	TCP发送方拥塞控制动作	注 释
慢启动（SS）	收到前面未确认数据的ACK	CongWin = CongWin + MSS，if(CongWin > Threshold)设置状态为“拥塞避免”	使每过RTT Cong-Win翻倍
拥塞避免（CA）	收到前面未确认数据的ACK	CongWin = CongWin + MSS · MSS/CongWin	加性增，每过RTT使CongWin增加1个MSS
SS或CA	由3个冗余ACK检测到的丢包事件	Threshold = CongWin/2，CongWin = Threshold，设置状态为“拥塞避免”	快速恢复，实现乘性减。CongWin将不低于1个MSS
SS或CA	超时	Threshold = CongWin/2，CongWin = Threshold，设置状态为“慢启动”	进入慢启动
SS或CA	冗余ACK	对确认的报文段增加冗余ACK计数	CongWin和Threshold不变

表3-3中总结了我们对TCP拥塞控制算法的讨论。此时，自然要考虑TCP拥塞控制为什么在处理发生的超时事件和收到3个冗余ACK的事件时，要采取不同的动作。特别是，为什么一个TCP发送方在发生超时事件后行为保守，将其拥塞窗口减至1个 MSS，而在收到3个冗余ACK后将其减半？有趣的是，一种早期版本的TCP（如TCP Tahoe），不管发生哪种丢包事件，都无条件地将其拥塞窗口减至1 MSS，并进入慢启动阶段。较新版本的TCP（TCP Reno）在

收到3个冗余ACK事件后会取消慢启动阶段。在这种情况下，取消慢启动阶段的原因是，即使某个分组丢失了，但3个冗余ACK的到来表明了某些报文段（特别是在丢失报文段以外的另外3个报文段）已经被发送方收到了。因此，与发生超时事件的情况不同，网络在显示它自己至少能交付一些报文段，即使其他报文段因拥塞而丢失了。这种在收到3个冗余ACK后取消慢启动阶段的行为称为**快速恢复**（fast recovery）。

图3-53说明了Reno版TCP与Tahoe版TCP的拥塞控制窗口的演变情况。在该图中，阈值初始时等于8个MSS。在前8个传输回合，Tahoe和Reno采取了相同的动作。拥塞窗口在慢启动阶段以指数速率快速爬升，在第4轮传输时达到阈值。然后拥塞窗口以线性速率爬升，直到在第8轮传输后，收到3个冗余ACK时为止。注意，当丢包事件发生时，拥塞窗口值为12 · MSS。于是阈值被设置为0.5 · CongWin = 6 · MSS。在TCP Reno中，拥塞窗口被设置为CongWin = 6 · MSS，然后线性地增长。在TCP Tahoe中，拥塞窗口被设置为1个 MSS，然后呈指数型增长，直至达到阈值为止。这种拥塞控制算法归功于V. Jacobson [Jacobson 1988]，对Jacobson的初始算法所做的若干修改在[Stevens 1994]和[RFC 2581]中进行了描述。

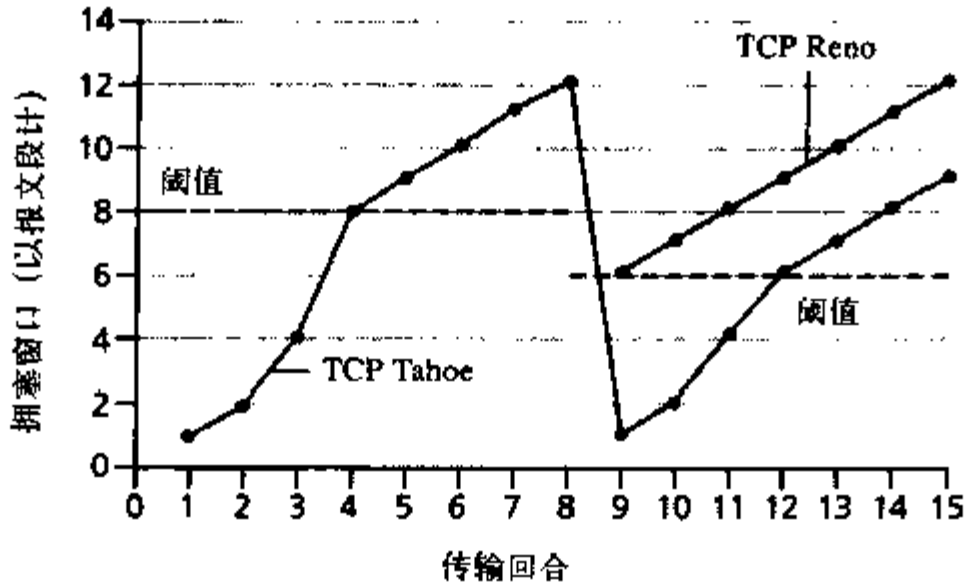


图3-53 TCP拥塞窗口的演变（Tahoe和Reno）

如上所述，当前大多数的TCP实现都是采用Reno算法。人们已提出Reno算法的许多变种[RFC 3782; RFC 2018]。所提出的TCP Vegas算法[Brakmo 1995; Ahn 1995]试图在维持较好的吞吐量时避免拥塞。Vegas的基本思想是：① 在分组丢失发生之前，检测源与目的地之间的路由器中的拥塞情况，② 当检测出快要发生的分组丢失时，线性地降低发送速率。快要发生的分组丢失是通过观察RTT来预测的。分组的RTT越长，路由器中的拥塞越严重。

4. 对TCP吞吐量的宏观描述

给出TCP的锯齿状行为后，自然要考虑一个长存活期的TCP连接的平均吞吐量（即平均速率）可能是多少。在这个分析中，我们将忽略在超时事件后出现的慢启动阶段。（这些阶段通常非常短，因为发送方很快就以指数增长离开该阶段。）在一个特定的往返间隔内，TCP发送数据的速率是拥塞窗口与当前RTT的函数。当窗口长度是w字节，当前往返时延是RTT秒时，TCP的传输速率大约是w/RTT。于是，TCP通过每经过一个RTT将w增加一个MSS来探测额外的带宽，直到一个丢包事件发生为止。当丢包事件发生时，用W表示w的值。假设在连接持续期间RTT和W几乎不变，那么TCP的传输速率在W/(2 · RTT) ~ W/RTT之间变化。

这些假设导出了TCP稳态行为的一个高度简化的宏观模型。当速率增长至W/RTT时，网络丢弃来自连接的分组，发送速率会减半，进而每过一个RTT就增加MSS/RTT，直到再次达

到W/RTT为止。这一过程不断地自我重复。由于TCP吞吐量（即速率）在两个极值之间线性增长，所以我们有

$$\text{一个连接的平均吞吐量} = \frac{0.75 \cdot W}{\text{RTT}}$$

通过这个对于TCP稳态动力学的高度理想模型，我们可以推出一个将连接丢包率与可用带宽联系起来的有趣表达式[Mahdavi 1997]。这个推导将在课后习题中概述。一个根据经验建立以符合测量数据的更复杂模型参见[Padhye 2000]。

5. TCP的未来

认识到下列事实是很重要的：TCP拥塞控制已经演化了多年并仍在继续演化。有关20世纪90年代后期TCP拥塞控制的总结可以在[RFC 2581]中找到；有关TCP拥塞控制发展的讨论参见[Floyd 2001]。以往对因特网（那时大量TCP连接承载的是SMTP、FTP和Telnet流量）有益的东西，对当今HTTP主宰的因特网或对未来的因特网难以想象的服务就不一定有益了。

TCP的继续演化的需求能够通过考虑网格计算应用所需要的高速TCP连接加以说明[Foster 2002]。例如，考虑一条具有1500字节报文段和100 ms RTT的TCP连接，假定我们要通过这条连接以10 Gbps发送数据。根据[RFC 3649]，我们注意到使用上述TCP吞吐量公式，为了取得10 Gbps吞吐量，平均拥塞窗口长度将要求是83 333个报文段。如此大量的报文段使我们相当关注这83 333个传输中的报文段也许会丢失的问题。在丢失的情况下将会出现什么情况呢？换句话说，这些传输的报文段以何种比例丢失将允许在表3-3中列出的TCP拥塞控制算法仍能取得所希望的10 Gbps速率？在本章的课后作业中，要求推导一个关于一条TCP连接的吞吐量的公式，该公式是丢包率（L）、往返时延（RTT）和最大报文段长度（MSS）的函数：

$$\text{一个连接的平均吞吐量} = \frac{1.22 \cdot \text{MSS}}{\text{RTT}/\sqrt{L}}$$

从该公式中可以看到，为了达到10 Gbps的吞吐量，今天的TCP拥塞控制算法能容忍$2 \cdot 10^{-10}$的报文段丢失率（或等价地说，每5 000 000个报文段有一个丢包），这是一个非常低的值。该观察使得一些研究人员为如此高速环境下设计新版TCP，对这些努力的讨论参见[Jin 2004, RFC 3649, Kelly 2003]。

公平性

考虑K条TCP连接，每条连接都有不同的端到端路径，但是都经过一段传输速率为R bps的瓶颈链路。（所谓瓶颈链路，是指对于每条连接，沿着该连接路径上的所有其他段链路都不拥塞，而且与该瓶颈链路的传输容量相比，它们有充足的传输容量。）假设每条连接都在传输一个大文件，而且无UDP流量通过该段瓶颈链路。如果每条连接的平均传输速率接近R/K，即每条连接都得到相同份额的链路带宽，则认为这个拥塞控制机制是公平的。

TCP的AIMD算法是公平的吗？尤其是假定不同的TCP连接可在不同的时间启动，因此在某个给定的时间点可能具有不同的窗口长度情况下，该算法是公平的吗？为什么TCP拥塞控制极力在竞争的TCP连接之间提供对一段瓶颈链路的带宽的平等分享，[Chiu 1989]给出了一个优雅的、直观的解释。

我们考虑有两条TCP连接共享一段传输速率为R的链路的简单例子，如图3-54所示。我们将假设这两条连接有相同的MSS和RTT（这样，如果它们有相同的拥塞窗口长度，就会有相同的吞吐量），它们有大量的数据要发送，且没有其他TCP连接或UDP数据报穿越该段共享链

路。另外，我们将忽略TCP的慢启动阶段，并假设TCP连接一直按CA模式（AIMD）运行。

图3-55描绘出了两条TCP连接实现的吞吐量情况。如果TCP要在这两条TCP连接之间平等地共享链路带宽，那么实现的吞吐量曲线应当是从原点沿45度角的箭头向外发射（相等的带宽份额）。理想情况是，两个吞吐量的和应等于R。（当然，每条连接得到相同但份额为0的链路容量并非我们所期望的！）所以，我们的目标应该是使实现的吞吐量落在图3-55中“相等带宽份额”曲线与“全带宽利用率”曲线的交叉点附近。

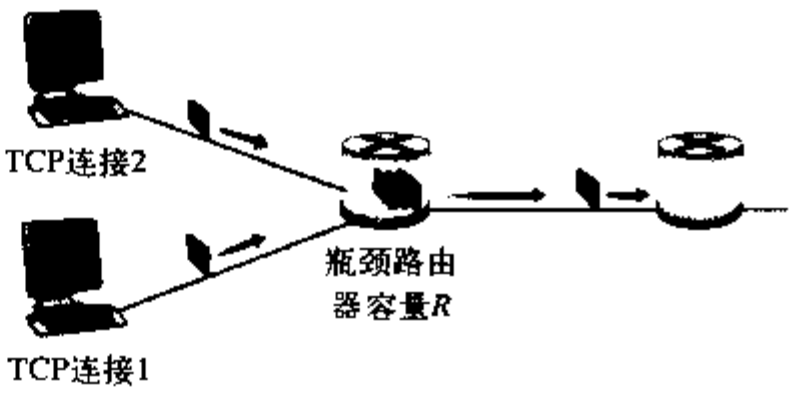


图3-54 两条TCP连接共享一段瓶颈链路

假定TCP窗口长度是这样的情况，即在某给定时刻，连接1和连接2实现了由图3-55中A点所指明的吞吐量。由于这两条连接共同消耗的链路带宽量小于R，所以无丢包事件发生，根据TCP的拥塞避免算法，这两条连接都要每过一个RTT将其窗口增加1个MSS。因此，这两条连接的总吞吐量就会从A点开始沿45度线增长（两条连接都有相同的增长）。最终，这两条连接共同消耗的带宽将超过R，分组丢失将发生。假设连接1和连接2实现B点指明的吞吐量时，它们都遇到分组丢失。连接1和连接2于是就按二分之一减小其窗口。结果实现了C点指明的吞吐量，它正好位于始于B点止于原点的一个向量的中间。因为在C点，共同消耗带宽小于R，所以这两条连接再次沿着始于C点的45度线增加其吞吐量。最后，再次发生丢包事件，如在D点，这两条连接再次将其窗口长度减半，以此类推。应该相信这两条连接实现的带宽终将沿着相同带宽份额的线波动。还应该相信，无论这两条连接位于二维空间的何处，它们最终都会汇聚到该状态！虽然在这种情形下，我们做了许多理想的假设，但是它仍然能对为什么TCP会导致在多个连接之间平等地共享带宽这个问题提供一个直观的感觉。

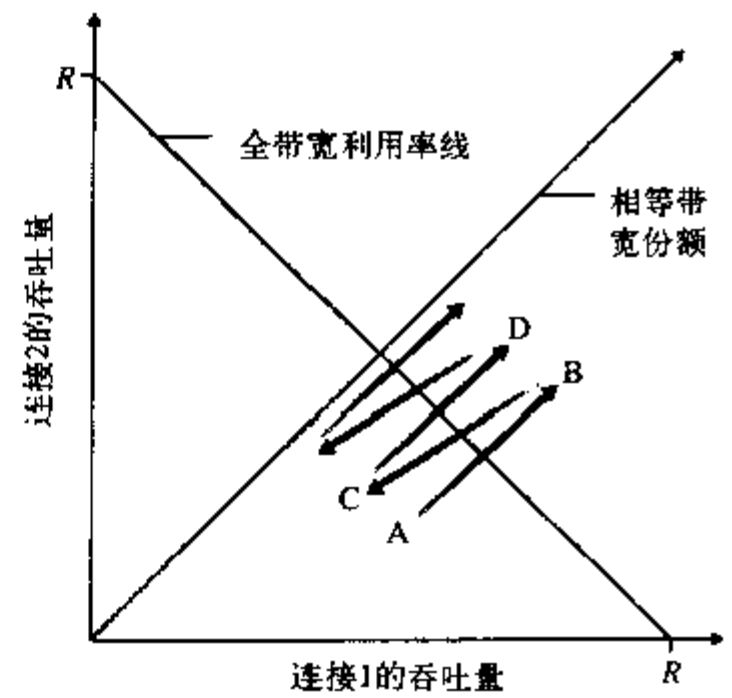


图3-55 TCP连接1和连接2实现的吞吐量

在理想化情形中，我们假设仅有TCP连接穿过瓶颈链路，所有的连接具有相同的RTT值，且对于一个主机目的地对，只有一条TCP连接与之相关联。实际上，这些条件通常是不满足的，客户机/服务器应用因此能获得链路带宽的非常不等同的份额。特别是，已经表明了当多个连接共享一个公共的瓶颈链路时，那些具有较小RTT的连接能够在链路空闲时更快地抢到可用带宽（即较快地打开其拥塞窗口），因而将比那些具有较大RTT的连接享有更高的吞吐量[Lakshman 1997]。

1. 公平性和UDP

我们刚才已经看到，TCP拥塞控制是如何通过拥塞窗口机制来调节一个应用程序的传输速率的。许多多媒体应用（如因特网电话和视频会议）就不是在TCP之上运行，因为它们不想其传输速率被扼制，即使在网络非常拥塞的情况下也是如此。这些应用倾向于在UDP之上

运行，UDP没有内置的拥塞控制。当运行在UDP之上时，这些应用能够以恒定的速率将其音频和视频数据注入网络之中，偶尔会丢失分组。即使这样，它们也不愿在拥塞时将其发送速率降至“公平”级别，而不丢失任何分组。从TCP的观点来看，运行在UDP之上的多媒体应用是不公平的，因为它们既不与其他连接合作，也不适当地调整其传输速率。因为TCP拥塞控制遇到拥塞（丢包）增加的情况将降低其传输速率，而UDP源则不必这样做，UDP源压制TCP流量的现象是可能发生的。当今的一个主要研究领域就是开发一种因特网拥塞控制机制，用于阻止UDP流量不断压制直至中断因特网吞吐量[Floyd 1999; Floyd 2000；Kohler 2006]。

2. 公平性和并行TCP连接

但是，即使我们能够迫使UDP流量行为公平，但公平性问题仍然没有彻底解决。这是因为我们无法阻止基于TCP的应用使用多条并行连接。例如，Web浏览器通常使用多条并行TCP连接来传送在一个Web页中的多个对象。（在多数浏览器中都可以配置连接的确切数目。）当一个应用使用多条并行连接时，它占用了一段拥塞链路较大部分的带宽。例如，考虑一段速率为R且支持9个在线客户机/服务器应用的链路，每个应用使用一条TCP连接。如果一个新的应用加入进来，也使用一条TCP连接，则每个应用得到差不多相同的传输速率$R/10$。但是，如果这个新的应用使用了11条并行TCP连接，则这个新应用就不公平地分到超过$R/2$的带宽。因为Web流量在因特网中非常普遍，所以多条并行连接并不是不常见的。

3.8 小结

我们在本章首先研究了运输层协议能够向网络应用程序提供的服务。一种极端的方案是，运输层协议非常简单，即不向应用程序提供不必要的服务，而仅向通信进程提供多路复用/多路分解的功能。因特网中的UDP协议就是这样一种不提供不必要服务的运输层协议。另一个极端的方案是，运输层协议能够向应用程序提供各种各样的确保，例如数据可靠传输、时延确保和带宽确保。无论如何，运输层协议能够提供的服务经常受下面网络层协议服务模型的限制。如果网络层协议不能向运输层报文段提供时延或带宽确保，那么运输层协议就不能向进程间发送的报文提供时延或带宽确保。

在3.4节中，我们学习到运输层协议能够提供可靠数据传输，即使下面的网络层是不可靠的。我们看到了提供可靠的数据传输会遇到许多微妙的问题，但都可以通过精心地结合确认、定时器、重传以及序号机制来解决该任务。

尽管本章讨论了可靠数据传输，但是我们应该理解在链路层、网络层、运输层或应用层协议中都可以提供可靠数据传输。该协议栈中上面4层的任意一层都可以实现确认、定时器、重传以及序号，也就可以向其上层提供可靠数据传输。事实上，在过去的数年中，工程人员以及计算机科学家就是独立地设计并实现了提供可靠数据传输的链路层、网络层、运输层以及应用层协议（虽然这些协议中的许多已经销声匿迹了）。

在3.5节中，我们详细地研究了TCP协议，它是因特网中面向连接和可靠的运输层协议。我们知道TCP是非常复杂的，它包含了连接管理、流量控制、往返时延估计以及可靠数据传输。事实上，TCP比我们描述的要更为复杂，我们有意不讨论在各种TCP实现版本中广泛实现的各种TCP补丁、调整和改进。然而，所有这些复杂性都对网络层应用隐藏起来。如果一台主机上的客户机希望向另一台主机上的服务器可靠地发送数据，它只需要打开一个对该服务器的TCP套接字，然后将数据注入该套接字。客户机/服务器应用程序则乐于不知晓TCP的复杂性。

在3.6节中我们广泛地研究了拥塞控制，在3.7节中我们阐述了TCP是如何实现拥塞控制的。我们知道了拥塞控制对于网络良好运行是必不可少的。没有拥塞控制，网络很容易出现死锁，使得端到端之间很少或没有数据能够正常传输。在3.7节中我们学习了TCP实现了这样一种端到端拥塞控制机制，即当TCP连接的路径上判断不拥塞时，其传输速率就加性增；当出现丢包时，传输速率就乘性减。这种机制也致力于做到每一个通过拥塞链路的TCP连接能平等地共享该链路带宽。我们也深入地探讨了TCP建立连接和慢启动对时延的影响。我们观察到，在许多重要情形中，建立连接和慢启动会对端到端时延产生显著影响。我们再次强调，尽管TCP在这几年一直在发展，但是它仍然是一个令人感兴趣的研究领域并且将在未来的几年中还可能持续演化。

在本章中我们对特定因特网运输层协议的讨论集中在UDP和TCP上，它们是因特网运输层的两匹“驮马”。然而，使用这两个协议20多年来，人们已经认识到这两个协议都不是完美无缺的。因此，研究人员忙于研制其他的运输层协议，其中几种现在已经成为IETF推荐的标准。

数据报拥塞控制协议（Datagram Congestion Control Protocol, DCCP）[RFC 4340]提供了一种低开销、面向报文、类似于UDP的不可靠服务，但是它具有应用程序选择形式的拥塞控制，这种形式与TCP相兼容。如果某应用程序需要可靠的或半可靠的数据传送，则将在该应用程序本身中执行，也许使用3.4节中讨论过的机制。DCCP被设想用于诸如流媒体（参见第7章）等应用程序中，DCCP能够在数据交付的预定时间和可靠性之间进行折中，但是要对网络拥塞作出响应。

流控制传输协议（Stream Control Transmission Protocol, SCTP）[RFC 2960, RFC 3286]是一种可靠的、面向报文的协议，该协议允许将几个不同的应用层次的“流”多路复用到单个SCTP连接上（一种称为“多流”的方法）。从可靠性方面看，对该连接中的不同流分别进行处理，因此一个流中的分组丢失不会影响其他流中数据的交付。当一台主机与两个或更多个网络连接时，SCTP也允许数据经两条出去的路径传输、失序数据可选地交付等。SCTP的流控制和拥塞控制算法基本上与TCP中的相同。

TCP友好速率控制（TCP-Friendly Rate Control, TFRC）协议[RFC 2448]是一种拥塞控制协议而不是一种功能齐全的运输层协议。它定义了一种拥塞控制机制，该机制可用于诸如DCCP等另一种运输协议（事实上，在DCCP中可供使用的两种应用程序可选的协议之一就是TFRC）。TFRC的目标是使TCP拥塞控制中的“锯齿”行为平滑（参见图3-51），同时维护一种长期的发送速率，该速率是“合理的”接近TCP的速率。使用比TCP更为平滑的发送速率，TFRC非常适合IP电话或流媒体等多媒体应用，这种平滑的速率对于这些应用是非常重要的。TFRC是一种“基于方程”的协议，使用测量到的丢包率作为方程的输入[Padhye 2000]，如果一个TCP会话历经了该丢包率的话，该方程估计TCP的吞吐量将是多大。该速率则被取为TFRC的目标发送速率。

唯有未来才能告诉我们DCCP、SCTP或TFRC是否能得到广泛应用。虽然这些协议明确地提供了超过TCP和UDP的增强能力，但是多年来已经证明了TCP和UDP是“足够好”的。“更好”是否会胜出“足够好”，将取决于技术、社会和商业多方面的考虑。

在第1章中我们讲到计算机网络可以划分成“网络边缘”和“网络核心”。网络边缘包含了在端系统中发生的所有事情。现在我们已经讨论了应用层和运输层，关于网络边界的讨论也就结束了，接下来是探寻网络核心的时候了！从下一章开始，我们将学习网络层，并且在第5章继续学习链路层。

课后习题和问题

复习题

3.1～3.3节

1. 假定网络层提供了下列服务。源主机中的网络层接受最大长度1200字节和来自运输层的目的主机地址的报文段。网络层则保证将报文段交付给位于目的主机的运输层。假定在目的主机上能够运行许多网络应用进程。
   a. 设计最简单的运输层协议，该协议将使应用程序数据到达目的主机上所希望的进程。假设目的主机中的操作系统已经为每个运行的应用进程分配了一个4字节的端口号。
   b. 修改这个协议，使它向目的进程提供一个“返回地址”。
   c. 在你的协议中，该运输层在计算机网络的核心中“什么也不做”吗？
2. 考虑有一个星球，每个人都属于一个六口之家，每个家庭都住在自己的房子里，每个房子都有一个唯一的地址，并且每个家庭中的每个人都有一个唯一的名字。假定该星球有一个从源家庭到目的家庭交付信件的邮政服务。该邮件服务要求：①在一个信封中有一封信；②在信封上清楚地写上目的家庭（并且没有其他东西）的地址。假设每个家庭有一名家庭成员代表为家庭中的其他成员收集和分发信件。这些信没有必要提供接收者。
   a. 使用对上面问题1的解决方案作为启发，描述家庭成员代表能够使用的协议，以便从发送家庭成员向接收家庭成员交付信件。
   b. 在你的协议中，该邮政服务必须打开信封并检查信件内容才能提供其服务吗？
3. 考虑在主机A和主机B之间有一条TCP连接。假设从主机A传送到主机B的TCP报文段使用的源端口号是x，目的端口号是y。那么对于从主机B传送到主机A的TCP报文段而言，源端口号和目的端口号分别是多少？
4. 描述应用程序开发者为什么更倾向于选择在UDP上运行应用程序而不是在TCP上运行。
5. 在今天的因特网中，为什么语音和视频流量常常是经TCP而不是经UDP发送？（提示：答案与TCP的拥塞控制机制没有关系。）
6. 当应用程序运行在UDP上时，该应用程序是否能够得到可靠数据传输？如果能，如何实现？
7. 假定主机C上的一个进程有一个端口号6789的UDP套接字。假定主机A和主机B都用目的端口号6789向主机C发送一个UDP报文段。这两台主机的这些报文段在主机C上都被描述为相同的套接字吗？如果是这样的话，主机C上的进程是怎样区分源于两台不同主机的两个报文段的？
8. 假定在主机C的端口80上运行一个Web服务器。假定这个Web服务器使用持久连接，并且正在接收来自两台不同主机A和B的请求。被发送的所有请求都通过位于主机C的相同套接字吗？如果它们通过不同的套接字传递，这两个套接字都具有端口80吗？讨论和解释之。

3.4节

9. 在rdt协议中，为什么需要引入序号？
10. 在rdt协议中，为什么需要引入定时器？
11. 假定发送方和接收方之间的往返时延是固定的并且为发送方所知。假设分组能够丢失的话，在协议rdt3.0中，仍需要定时器吗？试解释之。
12. 在配套网站上使用回退N步Java小程序。
    a. 让源发送5个分组，在这5个分组任何一个到达目的地之前暂停该动画，然后毁掉第一个分组并继续该动画。描述发生的情况。
    b. 重复该实验，只是现在让第一个分组到达目的地并毁掉第一个确认。再次描述发生的情况。

c. 最后，尝试发送6个分组。发生了什么情况?

13. 重复12题，但是现在使用选择重传Java小程序。选择重传和回退N步有什么不同?

3.5节

14. 是非判断题：

a. 主机A通过一条TCP连接向主机B发送一个大文件。假设主机B没有数据发往主机A。因为主机B不能随数据捎带确认信息，所以主机B将不向主机A发送确认。

b. 在连接的整个过程中，TCP的RcvWindow的长度不会变化。

c. 假设主机A通过一条TCP连接向主机B发送一个大文件。主机A发送的未被确认的字节数不会超过接收缓存的大小。

d. 假设主机A通过一条TCP连接向主机B发送一个大文件。如果对于这次连接的一个报文段序列号为m，则对于后继报文段的序列号将必然是$m+1$。

e. TCP报文段在它的首部中有一个RcvWindow字段。

f. 假定在一条TCP连接中最后的SampleRTT等于1 s，那么对于这一连接的TimeoutInterval的当前值必定≥1 s。

g. 假设主机A通过一条TCP连接向主机B发送一个序号为38的4字节报文段。这个报文段的确认号必定是42。

15. 假设主机A通过一条TCP连接向主机B连续发送两个TCP报文段。第一个报文段的序号为90，第二个报文段的序号为110。

a. 第一个报文段中有多少数据?

b. 假设第一个报文段丢失而第二个报文段到达主机B，那么在主机B发往主机A的确认报文中，确认号应该是多少?

16. 考虑3.5节中讨论的Telnet的例子。在键入字符C数秒之后，用户又键入字符R。那么在用户键入字符R之后，总共发送了多少个报文段？这些报文段中的序号和确认号字段应该填入什么?

3.7节

17. 假设两个TCP连接存在于一个带宽为R bps的瓶颈链路上。它们都要发送一个很大的文件（以相同方向经过瓶颈链路），并且两者是同时开始发送文件。那么TCP将为每个连接分配多大的传输速率?

18. 是非判断题。考虑TCP的拥塞控制，发送方定时器超时时，其阈值将被设置为原来值的一半。

习题

1. 假设客户机A向服务器S发起一个Telnet会话。与此同时，客户机B也向服务器S发起一个Telnet会话。给出下面报文段的源端口号和目的端口号：

a. 从A发往S的报文段。

b. 从B发往S的报文段。

c. 从S发往A的报文段。

d. 从S发往B的报文段。

e. 如果A和B是不同的主机，那么从A发往S的报文段的源端口号是否可能与从B发往S的报文段的源端口号相同?

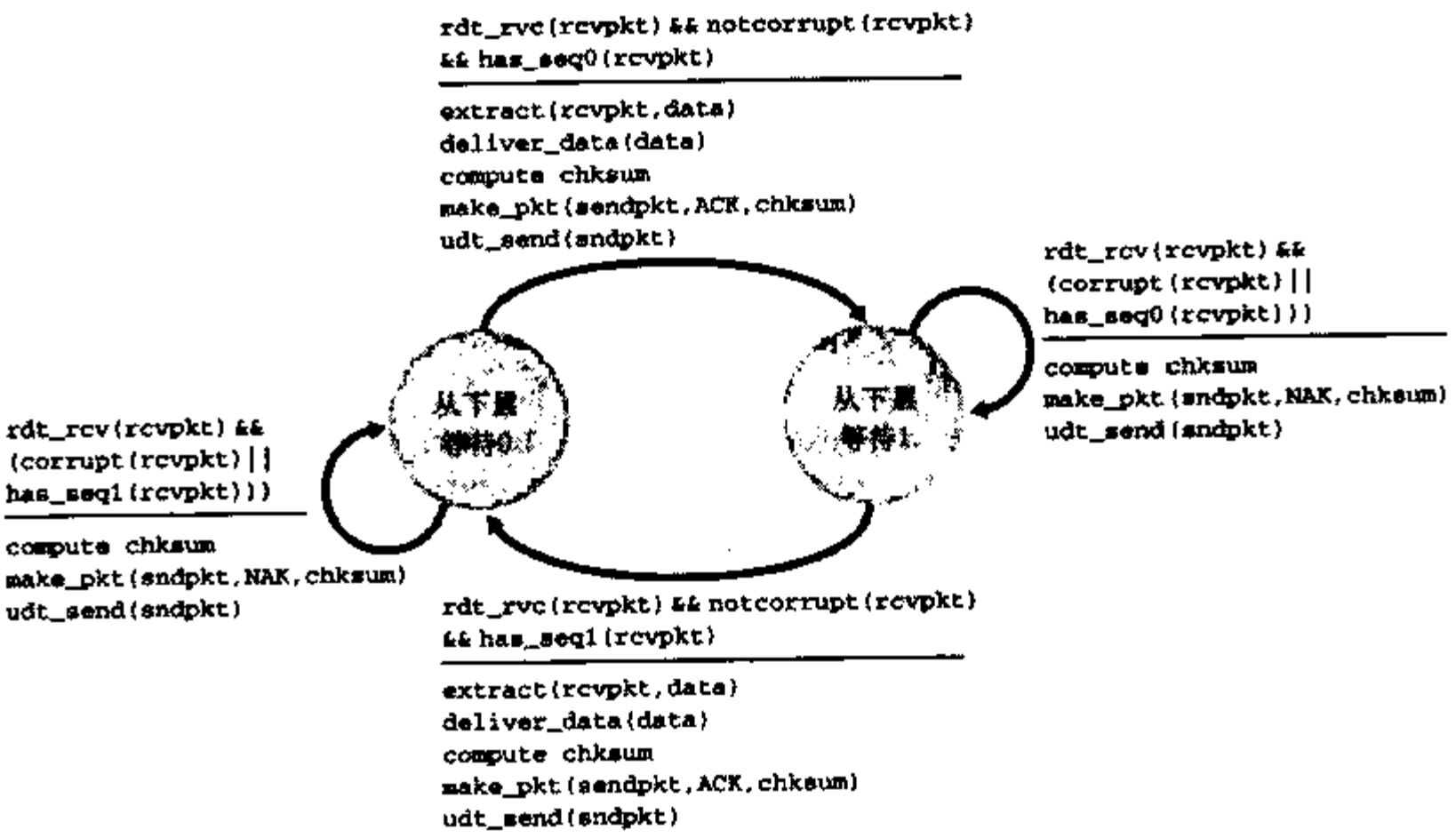
f. 如果它们是同一台主机，情况会怎样?

2. 参见图3-5，从服务器返回客户机进程的报文段的源端口号和目的端口号是多少？传送运输层报文段的网络层数据报中的IP地址是多少?

3. UDP和TCP使用反码来计算检验和。假设有下面3个8 比特字节：01010101, 01110000, 01001100。这

些8 比特字节和的反码是多少？（注意到尽管UDP和TCP使用16 比特的字来计算检验和，但对于这个问题，应该考虑8 比特和。）写出所有工作过程。UDP为什么要用该和的反码，即为什么不直接使用该和呢？使用该反码方案，接收方如何检测出差错？1 比特的差错将可能检测不出来吗？2 比特的差错呢？

4. a. 假定有下列2个字节：00110100和01101001。这2个字节的反码是什么？
   b. 假定有下列2个字节：11110101和00101001。这2个字节的反码是什么？
   c. 对于（a）部分中的字节，给出一个例子，使得这2个字节都在一个比特反转时，其反码不会改变。
5. 假定某UDP接收方对接收到的UDP报文段计算互联网检验和，并发现它与检验和字段中的值相匹配。该接收方能够确信没有出现比特差错吗？解释之。
6. 考虑我们改正rdt2.1协议的动机。考虑下图所示的接收方与如图3-11所示的发送方运行时，该接收方可能会引起发送方和接收方进入死锁状态，即双方都在等待不可能发生的事件。请说明这种情况。



7. 在rdt3.0协议中，从接收方向发送方发送的ACK分组没有序号（尽管它们具有ACK字段，该字段包括了它们正在确认的分组的序号）。为什么这些ACK分组不需要序号呢？
8. 画出rdt3.0协议中接收方的有限状态机（FSM）。
9. 当数据分组和确认分组发生混淆时，画出rdt3.0协议运行的轨迹。你画的轨迹应当类似于图3-16中所使用的图。
10. 考虑一个能够丢失分组但其最大时延已知的信道。修改rdt2.1协议，以包括发送方超时和重传机制。非形式化地论证为什么你的协议能够通过该信道正确通信。
11. rdt3.0协议的发送方直接忽略（即不采取任何措施）所有出现差错和确认号差错的确认分组。假设在这种情况下，rdt3.0只是重传当前的数据分组，该协议是否还能正常运行？（提示：考虑在仅有一个比特差错时会发生什么情况，在分组没有丢失但是定时器超时过早发生时会出现什么情况。考虑到当n趋于无穷时，第n个分组将被发送多少次。）
12. 考虑rdt3.0协议。如果连接发送方和接收方的网络能够对报文重新排序（即在发送方和接收方之间的介质上传播的两个报文能重新排序），那么比特交替协议将不能正确运行（确信你清楚地理解了这时它不能正确运行的原因）。试画图说明。画图时请把发送方放在左边，接收方放在右边，使时间轴朝

下，标出交换的数据报文（D）和确认报文（A）。你要标明与任何与数据和确认报文相关的序号。

13. 考虑一种仅使用否定确认的可靠数据传输协议。假定发送方只是偶尔发送数据。是否只用NAK的协议比使用ACK的协议更好？为什么？现在我们假设发送方要发送大量的数据，并且该端到端连接很少丢包。在第二种情况下，是否只用NAK的协议比使用ACK的协议更好？为什么？

14. 考虑图3-17中跨国网络的例子。窗口长度设置成多少时，才能使该信道的利用率超过90%？

15. 在3.4.4节中学习的一般SR协议中，只要报文可用（如果报文在窗口中），发送方就会不等待确认而传输报文。现在假设我们需要一个SR协议，一次发出一对报文，而且只有在知道第一对报文正确到达后才发送第二对报文。

    假设该信道中可能会丢失报文，但报文不会发生差错和失序。请设计一个差错控制协议实现报文的单向可靠传输。画出发送方和接收方的FSM描述。给出发送方和接收方之间两个方向发送的报文的格式。如果你使用了不同于3.4节（例如`udt_send()`、`start_timer()`、`rdt_rcv()`等）中的过程调用，请详细地描述这些动作。举例说明（用发送方和接收方的时序踪迹图）你所设计的协议是如何恢复报文丢失的。

16. 考虑一种情况，主机A想同时向主机B和主机C发送报文。A与B和C是经过广播信道连接的，即由A发送的分组通过该信道传送到B和C。假设连接A、B和C的这个广播信道具有独立的报文丢失和损坏特性（因此，例如从A发出的报文可能被B正确接收，但没有被C正确接收）。设计一个类似于停等的差错控制协议，以保证分组能从A可靠地传输到B和C。该协议使得A直到得知B和C已经正确接收到当前报文，才获取上层交付的新数据。给出A和C的FSM描述。（提示：B的FSM大体上应当与C的相同。）同时，给出所使用分组的格式。

17. 考虑一种主机A和主机B要向主机C发送报文的情况。A和C通过一条报文能够丢失和损坏（但不重新排序）的信道相连接。B和C由另一条（与连接A和C的信道不同）具有相同性质的信道连接。主机C上的运输层在向上层交付来自主机A和B的报文时应当交替进行（即它应当首先交付来自A的分组中的数据，然后交付来自B的分组中的数据等）。设计一个类似于停等的差错控制协议，以保证分组能从A和B可靠地传输到C，同时以前面描述的方式在C交替地交付。给出A和C的FSM描述。（提示：B的FSM大体上应当与A的相同。）同时，给出所使用分组的格式。

18. 考虑一个GBN协议，其发送方窗口长度为3，序号范围为1024。假设在时刻t，接收方期待的下一个有序分组的序号是k。假设其中的介质不会对报文重新排序。请回答以下问题：

    a. 在t时刻，发送方窗口内的报文序号可能是多少？论证你的回答。

    b. 在t时刻，在当前发送方收到的所有报文中，ACK字段的可能值是多少？论证你的回答。

19. 假定有两个网络实体A和B。B有一些数据报文要通过下列规则传给A：当A从其上层得到一个请求时，就从B接收下一个数据（D）报文。A必须通过A到B信道向B发送一个请求（R）报文。仅当B收到一个R报文后，它才会通过B到A信道向A发送数据（D）报文。A应当将每份D报文的拷贝交付给上层。R报文可能会在A到B的信道中丢失（但不会损坏）；D报文一旦发出总是能够正确交付。两个信道的时延未知且是变化的。

    请设计一个协议（画出FSM），它能够综合适当的机制，以补偿A到B信道中可能出现的丢包，实现在A实体中向上层传递的报文。只采用绝对必要的机制。

20. 考虑GBN协议和SR协议。假设序号空间的长度为k，那么为了避免出现图3-27中的问题，允许的最大发送方窗口是多少？

21. 判断下面的问题，并简要地证实你的回答：

    a. 在SR协议中，发送方可能会收到落在其当前窗口之外的分组的ACK。

    b. 在GBN协议中，发送方可能会收到落在其当前窗口之外的分组的ACK。

c. 当发送方和接收方窗口长度都为1时，比特交替协议与SR协议相同。

d. 当发送方和接收方窗口长度都为1时，比特交替协议与GBN协议相同。

22. 我们曾经说过，应用程序可能选择UDP作为运输层协议，因为UDP（比TCP）提供了更好的应用层控制，以确定在报文段中发送什么数据和发送时机。

a. 应用程序为什么对报文段中发送什么数据有更多的控制?

b. 应用程序为什么对何时发送报文段有更多的控制?

23. 考虑从主机A向主机B传输L字节的大文件，假设MMS为1460字节。

a. 在TCP序号允许范围内，L的值最大是多少？TCP的序号字段为4 字节。

b. 对于你在（a）中得到的L，求出传输此文件要用多长时间。假定运输层、网络层和数据链路层总共加在每个报文段首部上的长度为66 字节，传输分组的链路速率为10 Mbps。不采用流量控制和拥塞控制，因此主机A能够一个接一个、连续不断地发送报文段。

24. 主机A和B通过一个TCP连接通信，并且主机B已经收到了来自A的直到字节248的所有字节。假定主机A随后向主机B发送两个报文段。第一个和第二个报文段分别包含了40和60字节的数据。在第一个报文段中，序号是249，源端口号是503，目的端口号是80。无论何时主机B接收到来自主机A的报文段，它都会发送确认。

a. 在从主机A发往B的第二个报文段中，序号、源端口号和目的端口号各是什么？

b. 如果第一个报文段在第二个报文段之前到达，在第一个到达报文段的确认中，确认号、源端口号和目的端口号各是什么?

c. 如果第二个报文段在第一个报文段之前到达，在第一个到达报文段的确认中，确认号是什么?

d. 假定由A发送的两个报文段按序到达B。第一个确认丢失了，而第二个确认在第一个超时间隔之后到达。画出时序图，显示这些报文段、发送的所有其他报文段和确认。（假设没有其他分组丢失。）对于每个报文段，标出序号和数据的字节编号；对于增加的每个确认，标出确认号。

25. 主机A和B直接通过一条200 Mbps链路连接。在这两台主机之间有一条TCP连接，主机A经这条连接向主机B发送一个大文件。主机A能够向100 Mbps速率的链路发送应用数据，而主机B能够以最大50 Mbps的速率从其TCP接收缓存中读出数据。描述TCP流量控制的作用。

26. 3.5.6节中讨论了SYN cookies。

a. 服务器为什么要在SYNACK中使用一个特殊的初始序号?

b. 假定某攻击者得知一台目标主机使用了SYN cookies。该攻击者能够通过直接向该目标发送一个ACK分组创建半开或全开连接吗？为什么?

27. 考虑TCP估计RTT的过程。假设$\alpha = 0.1$，令SampleRTT_1为最新样本RTT，SampleRTT_2设置为下一个最新样本RTT，等等。

a. 对于一个给定的TCP连接，假定4个确认报文相继到达，带有4个对应的RTT值：SampleRTT_4、SampleRTT_3、SampleRTT_2和SampleRTT_1。请根据这4个样本RTT表示EstimatedRTT。

b. 将你得到的公式扩展到n个RTT样本的情况。

c. 对于（b）中得到的公式，令n趋于无穷。请解释为什么这个平均过程被称为指数移动平均。

28. 在3.5.3节中我们讨论了TCP的往返时延的估计。请解释为什么TCP对重传报文段避免测量SampleRTT。

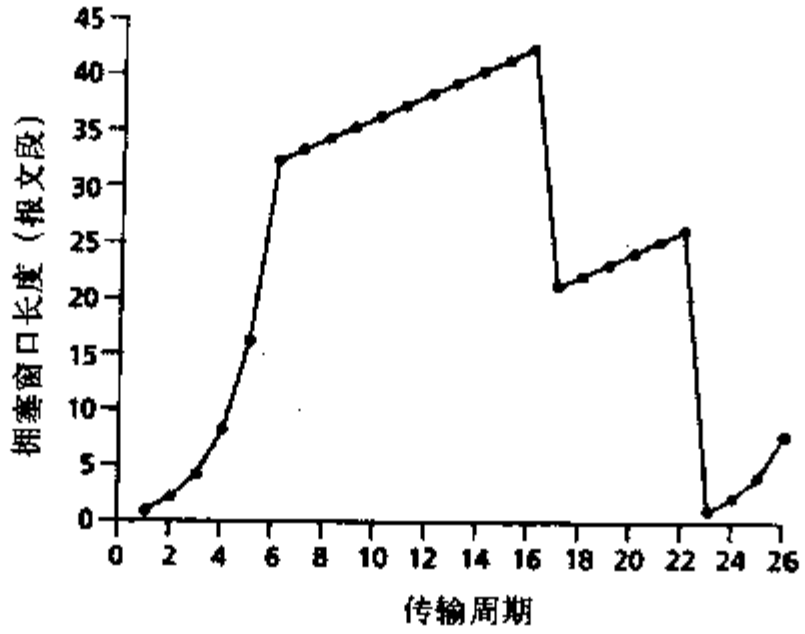
29. 3.5.4节中的变量SendBase和3.5.5节中的变量LastByteRcvd之间有什么关系?

30. 3.5.5节中的变量LastByteRcvd和3.5.4节中的变量y之间有什么关系?

31. 在3.5.4节中，我们看到TCP直到收到3个冗余ACK才执行快速重传。你认为TCP设计者为什么不在收到对报文段的第一个冗余ACK后就快速重传?

32. 考虑图3-46b。如果λ'_{in}增加超过了$R/2$，λ_{out}能增加超过$R/3$吗？试解释之。现在考虑图3-46c。假定一个分组从路由器到接收方平均转发两次的话，如果λ'_{in}增加超过$R/2$，λ_{out}能增加超过$R/4$吗？试解释之。

33. 考虑下图中TCP窗口长度作为时间的函数。



假设TCP Reno是一个经历如上图所示行为的协议，请回答下列问题。在各种情况下，简要地论证你的回答。

a. 指出当运行TCP慢启动时的时间间隔。

b. 指出当运行TCP拥塞避免时的时间间隔。

c. 在第16个传输周期之后，报文段的丢失是根据3个重复确认还是根据超时检测出来的？

d. 在第22个传输周期之后，报文段的丢失是根据3个重复确认还是根据超时检测出来的？

e. 在第一个传输周期里，Threshold的初始值设置为多少？

f. 在第18个传输周期里，Threshold的值设置为多少？

g. 在第24个传输周期里，Threshold的值设置为多少？

h. 第70个报文段在哪一个传输周期内发送？

i. 假定在第26个发送周期后，通过收到3个冗余ACK检测出有分组丢失，那么拥塞的窗口长度和Threshold的值应当是多少？

34. 参考图3-55，该图描述了TCP的AIMD算法的收敛特性。现在假设TCP不采用乘性减，而是采用按某一常量递减。所得的AIAD算法将收敛于一种平均共享算法吗？使用类似于图3-55中的图来证实你的结论。

35. 在3.5.4节中，我们讨论了在发生超时事件后将超时间隔加倍。该机制是拥塞控制的一种形式。为什么TCP除了这种加倍超时间隔机制外，还需要基于窗口的拥塞控制机制（参见3.7节）呢？

36. 主机A通过一条TCP连接向主机B发送一个很大的文件。在这条连接上，不会出现任何分组丢失和定时器超时。主机A与因特网连接的链路的传输速率为R bps。假设主机A上的进程向TCP套接字发送数据的速率为S bps，其中$S = 10 \cdot R$。进一步假设TCP的接收缓存足够大，能够接收整个文件。然而，发送缓存只能容纳这个文件的百分之一。那么，如何防止主机A上的进程连续地向TCP套接字以速率S bps传送数据呢？是用TCP流量控制？是用TCP拥塞控制？还是用其他措施？请阐述之。

37. 考虑从一台主机经一条TCP连接向另一台主机发送一个大文件，这条连接不会丢包。

a. 假定TCP使用不具有慢启动的AIMD进行拥塞控制。假设每当收到一批ACK时，CongWin增加1个MSS，往返时间基本恒定，那么CongWin从1 MSS增加到6 MSS要花费多长时间（假设没有丢包事件）？

b. 对于该连接，直到时间为5 RTT，其平均吞吐量是多少（根据MSS和RTT来计算）？

38. 回想TCP吞吐量的宏观描述。假设在连接速率从W / (2 · RTT) 变化到W / RTT的期间内，只丢失了一个分组（在该期间的结束）。

a. 证明丢包率等于

$$L=\text{丢失率}=\frac{1}{\frac{3}{8}W^2+\frac{3}{4}W}$$

b. 如果一个连接的丢包率为L，那么使用上面的结果，它的平均带宽大约是

$$\frac{1.22*\text{MSS}}{\text{RTT}\sqrt{L}}$$

39. 在3.7节对TCP未来的讨论中，我们注意到为了取得10 Gbps的吞吐量，TCP仅能容忍$2\cdot10^{-10}$的报文段丢失率（或等价为每5 000 000 000个报文段有一个丢包事件）。给出对于3.7节中给定的RTT和MSS值$2\cdot10^{-10}$或5 000 000 000之一的推导。如果TCP需要支持一个100 Gbps的连接，所能容忍的丢包率是多少？

40. 在3.7节对TCP拥塞控制的讨论中，我们隐含地假定TCP发送方总是有数据发送。现在考虑下列情况：某TCP发送方发送大量数据，然后在t_1时刻变得空闲（因为它没有更多的数据要发送）。TCP在相对长时间内保持空闲，然后在t_2时刻要发送更多的数据。让TCP在t_2时刻开始发送数据时，使用它在t_1时刻的CongWin和Threshold值有什么样的优点和缺点？你认为应该使用什么样的方法？为什么？

41. 在这个习题中，我们研究UDP或TCP是否提供了某种程度的端点鉴别功能。

a. 考虑一台服务器接收到一个UDP分组中的请求并对该请求进行响应的情况（例如，像一台DNS服务器所做的那样）。如果一个IP地址为X的客户机用地址Y进行欺骗的话，服务器将向何处发送响应？

b. 假定一台服务器接收到来自IP源地址Y的一个SYN，在用SYNACK响应之后，接收到一个来自IP源地址Y、具有正确确认号的ACK。假设该服务器选择了一个随机初始序号并且没有“中间人”，它能够确定该客户机的确位于Y（并且不在某个其他地址X，用X欺骗为Y）吗？

42. 在这个习题中，我们研究由TCP慢启动阶段引入的时延。一个客户机和一个Web服务器直接通过一条速率为R的链路相连。假定该客户机要取回一个对象，其长度正好等于15S，其中S是最大段长度（MSS）。客户机和服务器之间的往返时延表示为RTT（假设为常数）。不考虑协议首部，在下列情况下，确定取回该对象的时间（包括TCP连接创建过程）。

a. $4S/R > S/R+\text{RTT} > 2S/R$

b. $S/R+\text{RTT} > 4S/R$

c. $S/R > \text{RTT}$

讨论题

1. 什么是TCP连接劫持？怎样做到？
2. 在3.7节中，我们讨论了客户机/服务器能够“不公平”产生许多并行的连接。要使因特网真正公平应该做些什么？
3. 请阅读有关“TCP友好”（TCP friendly）的研究文献，也读一下本章最后有关Sally Floyd的专访。撰写有关TCP友好特性的一页纸的摘要。
4. 在3.7.1节结束时我们讨论了这样的事实：一个应用程序能够打开多个TCP连接并获得较高吞吐量（或等价地一个较快的传输时间）。如果所有应用程序试图通过使用多个连接来改善性能的话，将会发生什么情况？让一个网络元素决定是否一个应用程序正在使用多个TCP连接，会涉及什么样的困难？

5. 除了对TCP和UDP端口扫描外，nmap还有什么功能？用nmap分组交换的Ethereal（或其他分组嗅探器）收集分组轨迹。使用该轨迹来解释某些先进的特色是如何工作的。
6. 阅读有关SCTP文献[RFC 2960, RFC 3286]。SCTP的设计者预想使用SCTP的应用程序是什么？为了满足这些应用程序的需求，SCTP增加了什么特色？

编程作业

实现一个可靠传输协议

在这个实验室编程作业中，你将要编写运输层的发送和接收代码，以实现一个简单的可靠数据传输协议。这个实验有两个版本，即比特交替协议版本和GBN版本。这个实验应当是有趣的，因为你的实现将与实际情况下所要求的东西差异很小。

因为你可能没有你能够修改其操作系统的独立机器，你的代码将不得不在模拟的硬件/软件环境中执行。然而，提供给你的例程的编程接口，即从上层和从下层调用你的实体的代码，非常类似于在实际UNIX环境中做那些事情的代码。（实际上，在本编程作业中描述的软件接口非常真实，使得许多教科书描述的发送方和接收方无限循环）。停止和启动定时器也是模拟的，定时器中断将激活你的定时器处理例程。

Ethereal实验：探究TCP

在这个实验中，你将使用Web浏览器从某Web服务器访问一个文件。如同在前面的Ethereal实验中一样，你将使用Ethereal来俘获到达你的计算机的分组。与前面实验不同的是，你也将能够从该Web服务器下载一个Ethereal可读的分组踪迹，记载你从服务器下载文件的过程。在这个服务器踪迹里，你将发现访问该Web服务器所产生的分组。你将分析客户机端和服务器端来探究TCP方方面面的轨迹。特别是你将评估在你的计算机和Web服务器之间TCP连接的性能。你将跟踪TCP窗口行为，推断分组丢失、重传、流量控制和拥塞控制行为，并估计往返时延。

如同所有的Ethereal实验一样，该实验的全面描述可以从本书的Web站点（http://www.awl.com/kurose-ross）找到。

Ethereal实验：探究UDP

在这个简短实验中，你将进行分组俘获并分析那些你喜爱的使用UDP的应用程序（例如，DNS或Skype等多媒体应用）。如3.3节中所介绍的那样，UDP是一种简单的、不提供不必要服务的运输层协议。在这个实验中，你将研究UDP报文段中的各首部字段并计算检验和。

如同所有的Ethereal实验一样，该实验的全面描述可以从本书的Web站点（http://www.awl.com/kurose-ross）找到。

人物专访

Sally Floyd是因特网研究中心的研究科学家，该研究中心位于ICSI，是一家专门从事因特网和网络问题研究的机构。她在业界以从事因特网协议的设计而著称，特别擅长于可靠多播、拥塞控制（TCP）、分组调度（RED）和协议分析。Sally在加州大学伯克利分校获得社会学学士学位，以及计算机科学的硕士、博士学位。

Sally Floyd

• 您是如何决定学习计算机科学的?

在获得社会学学士学位后，我不得不盘算我的生计问题；于是我从本地社区大学获得了两年的电子工程证书，之后在电子和计算机方面工作了10年。其中的8年我是作为计算机系统工程师，管理运行海湾地区快速列车的计算机。后来，我决定学习一些更加正规的计算机科学知识，申请去加州大学伯克利分校计算机科学系的研究生院读书。

• 您为什么决定专门研究网络呢?

在研究生院中，我开始对计算机科学理论产生了兴趣。我最初的工作是进行算法的概率分析，之后又从事计算学习理论的研究。我每个月还在LBL（Lawrence Berkeley Laboratory，洛仑兹伯克利实验室）工作一天，我的办公室在Van Jacobson机房的对门，他那时一直从事TCP拥塞控制算法的研究。Van问我是否愿意在那个暑期做一些与网络相关问题算法的分析工作，涉及周期性选路报文所不希望的同步问题。我对这个问题非常感兴趣，因此我那个暑期进行了这项研究。

在完成我的学位论文之后，Van为我提供了一个从事网络研究的全职工作。我本没打算在网络方面工作数年，但是对我来说，我更愿意从事网络方面的研究而不是计算机理论方面的研究。我发现我在该应用领域工作更为愉快，工作成果更为切合实际。

• 您在计算机行业的第一份工作是什么？具体做哪方面的工作?

我的第一份计算机工作是1975到1982年期间，在BART（Bay Area Rapid Transit）从事运行BART火车的计算机方面的工作。我开始是一名技术员，负责维护和修理各种运行BART系统的分布式计算机系统。

这些包括用于控制列车运动的一个中央计算机系统和分布式小型计算机系统；一个用于显示广告和用目的地符号表示列车目的站的DEC计算机系统；一个用于在收费口处搜集信息的Modcomp计算机系统。我在BART的后几年主要在一个BART/LBL合作的项目上度过，以设计一个系统来替代过时的BART列车控制的计算机系统。

• 您工作中最具挑战性的部分是什么?

实际的研究是最具挑战性的部分。当前，一个研究主题是探索用于流媒体等应用的拥塞控制问题。第二个主题是解决路由器和端节点之间更为显式通信的网络障碍，包括IP隧道和MPLS路径、丢弃含有IP选项分组的路由器或中间设备、复杂的两层网络和潜在的网络攻击。第三个主题是探讨怎样对所使用的各种模型进行分析、仿真和试验，以用于评价拥塞控制机制的性能。有关这些主题的更多信息位于DCCP、Quick-Start和TMRG的Web页面上，可通过http://www.icir.org/floyd访问到。

• 您认为网络和因特网的未来将怎样发展?

一种可能性是，随着可供使用的带宽增长速率超过用户需求的增长速率，因特网流量遇到的典型拥塞将变得不那么严重。我认为趋势是朝着拥塞减轻的方向发展，尽管由偶尔拥塞崩溃引发的中期的未来拥塞的增加看起来并不是不可能。

因特网自身或者说因特网体系结构的未来，我并不是非常清楚。这其中有许多因素会导致因特网的迅速改变，因此很难预测因特网或因特网体系结构将会如何演化，甚至无法预测这种演化是否能够成功，以避免发展道路上的许多可能的缺陷。

一种众所周知的负面趋势是增加了因特网体系结构改变的难度。因特网体系结构不再是一个条理分明的整体，传输协议、路由器机制、防火墙、负载均衡、安全机制等各种组件，有时看起来是为了错综复杂的目的而工作的。

• 哪些人激发了您的专业灵感?

我在研究生院的论文导师Richard Karp，他从根本上教会了我如何从事科学研究。我在LBL的课题

组负责人Van Jacobson使我对网络产生了兴趣，并使我理解了因特网的基础设施等。Dave Clark对因特网的深刻理解和他通过研究、撰写论文和参加IETF等公共论坛对因特网体系结构发展中所起的作用启发了我。Deborah Estrin的专注和高效以及她能够清楚地知道自己在做什么以及为什么这么做的能力给我以启迪。

我在最近十年中非常喜欢从事网络方面的研究的原因之一就是，在这一领域工作的那么多人是我所喜欢的、所尊敬的和对我有启发的人。他们非常聪明，勤奋工作，对因特网的发展有很强的责任感并做出了卓越的工作。此外，在会议之余，他们也是我能够闲谈和友好争论的好伙伴。

第4章 网 络 层

我们在前一章看到，运输层依赖于由网络层的主机到主机的通信服务，提供了各种形式的进程到进程的通信。我们已经知道，运输层工作时不具备任何有关网络层实际实现这种服务的知识。因此也许你想知道，这种主机到主机通信服务的真实情况是什么？什么使得它工作起来的呢？

在本章中我们将学习网络层实际是怎样实现主机到主机的通信服务的。我们将看到，与运输层不同的是，网络中的每一台主机和路由器中都有一个网络层部分。正因如此，网络层协议是协议栈中最具挑战性（因而也是最有趣）的部分之一。

网络层也是协议栈中最复杂的层次之一，因此我们将涉及大量的基础知识。我们的学习从网络层的概述和它能够提供的服务开始。我们将重温两种用于构造网络层分组交付的方法，即数据报模式和虚电路模式（这是我们在第1章中遇到的知识），并且理解编址在交付分组到其目的主机所起的重要作用。

在本章中，我们将在网络层的转发（forwarding）和选路（routing）功能之间做重要区分。转发涉及分组从一条入链路到一台路由器中的出链路的传送。选路涉及一个网络中的所有路由器，它们经选路协议共同交互，以决定分组从源到目的地节点所采用的路径。当你继续本章学习时，心中要记住这个区别，这将有助于你将所涉及的许多主题置于适当的场合中。

为了加深对分组转发的理解，我们将“进入”路由器内部来观察它的硬件体系结构和组织。接下来我们将观察在因特网中的分组转发，以及著名的网际协议（IP）。我们将研究网络层编址和IPv4的数据报格式。然后我们将探讨网络地址转换（NAT）、数据报分段、互联网控制报文协议（ICMP）和IPv6。

接下来，我们将关注网络层的选路功能。我们将看到选路算法的任务是决定从发送方到接收方的好路径（即路由）。我们将首先学习选路算法的理论，关注两种最为流行的算法类型：链路状态算法和距离向量算法。因为选路算法的复杂性随着网络路由器数量的增加，会有相当大的增加，因此层次选路方法将非常令人感兴趣。当我们涉及因特网的自治系统内部选路协议（RIP、OSPF和IS-IS）和它的自治系统之间选路协议（BGP）时，我们将看到理论是如何付诸实践的。最后，用对广播和多播选路的讨论作为本章的结束。

总之，本章有3个部分。第一部分（4.1节和4.2节）涉及了网络层功能和服务。第二部分(4.3节和4.4节）涉及了转发。第三部分（4.5节到4.7节）涉及了选路。

4.1 概述

图4-1给出了一个具有H1和H2两台主机且在H1与H2之间的路径上有几台路由器的简单网络。假设H1正在向H2发送信息，考虑一下这些主机与中间路由器中的网络层所起的作用。H1中的网络层接收来自H1运输层的每个报文段，将其封装成一个数据报（即网络层分组），然后将数据报向相邻路由器R1发送。在接收方主机H2，其网络层接收来自相邻路由器R2的数据报，提取出运输层报文段，并将其向上交付给H2的运输层。路由器的主要作用便是将数据报从入链路转发到出链路。注意到图4-1中所示的路由器具有删减的协议栈，即没有网络层以上的部分，因为（除了控制目的外）路由器不运行我们已在第2、3章学习过的应用层和运输层协议。

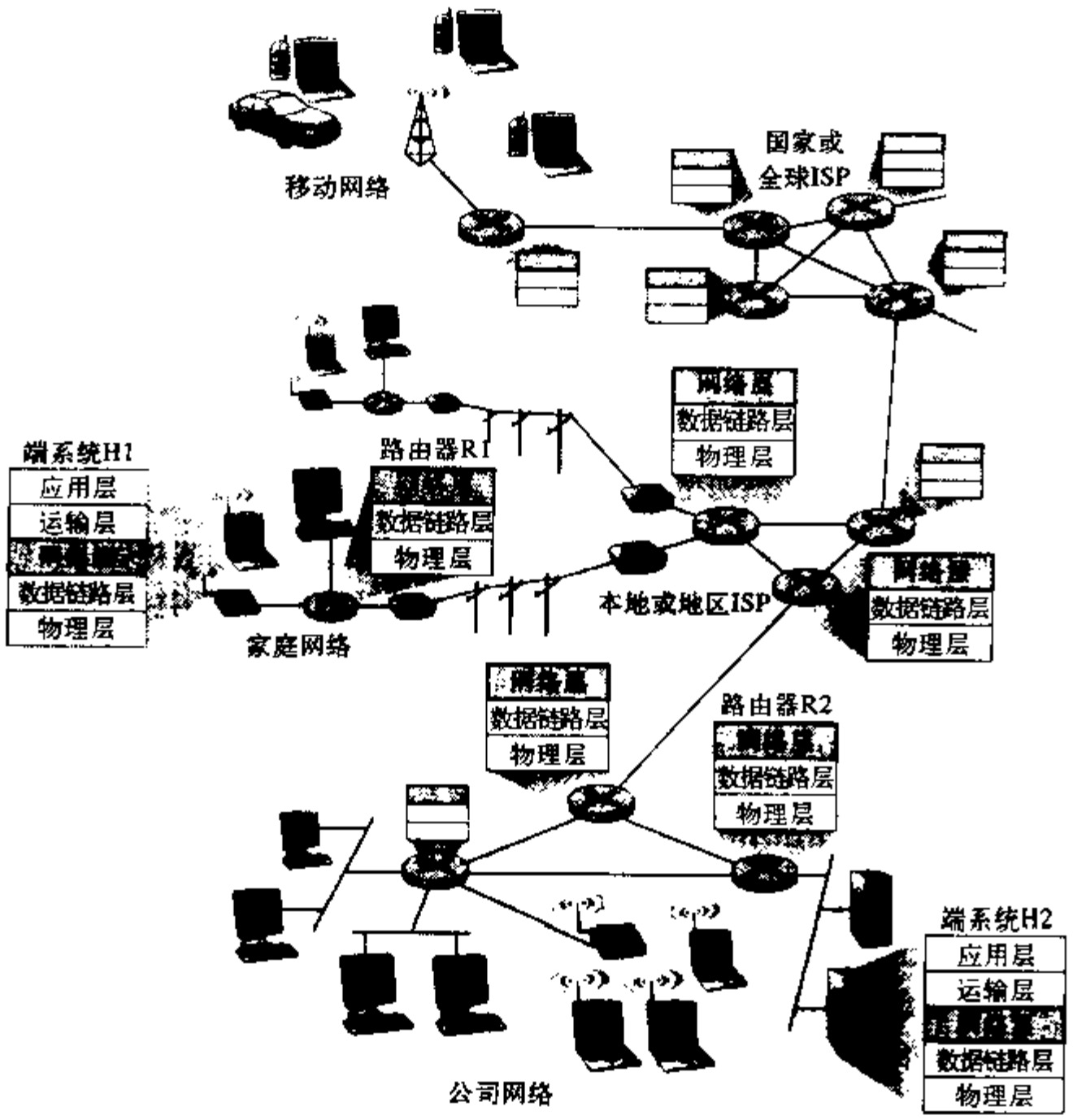


图4-1 网络层

4.1.1 转发和选路

网络层的作用从表面上看极为简单，即将分组从一台发送主机移动到一台接收主机。为此，需要两种重要的网络层功能：

- *转发*。当一个分组到达某路由器的一条输入链路时，该路由器必须将该分组移动到适当的输出链路。例如，来自主机H1到路由器R1的一个分组，必须向H2路径上的下一台路由器转发。在4.3节中，我们将深入路由器内部观察，考察分组在路由器中是如何实际从一条输入链路转发到输出链路的。
- *选路*。当分组从发送方流向接收方时，网络层必须决定这些分组所采用的路由或路径。计算这些路径的算法被称为**选路算法**（routing algorithm）。例如，一个选路算法将决定分组从H1到H2所遵循的路径。

在讨论网络层时，许多作者经常互换地使用*转发*和*选路*这两个术语。我们在本书中将更为精确地使用这些术语。*转发*是指将分组从一个输入链路接口转移到适当的输出链路接口的路由器本地动作。*选路*是指分组从源到目的地时，决定端到端路径的网络范围的进程。用一个驾驶的例子进行类比，考虑1.3.2节中旅行者所历经的从宾夕法尼亚州到佛罗里达州的行程。在这个行程中，驾驶员到佛罗里达州经过了途中的许多立交桥。可以认为转发就像通过单个

立交桥的过程：一辆汽车进入立交桥，决定应当走哪条路来离开该立交桥。而选路是规划从宾夕法尼亚州到佛罗里达州行程的过程：在开始行程之前，驾驶员查阅地图并在许多可能的路径中选择一条，其中每条路径都由一系列经立交桥连接的路段组成。在本章中，我们将首先关注与转发有关的网络层主题，然后再考虑选路问题。

每台路由器具有一张**转发表**（forwarding table）。路由器通过检查到达分组首部中的一个字段的值，然后使用该值在该路由器的转发表中索引查询来转发一个分组。查询转发表的结果是分组将被转发的路由器的链路接口。分组首部中的该值可能是该分组的目的地地址或该分组所属连接的指示，这取决于网络层协议。图4-2给出了一个例子。在图4-2中，一个首部字段值为0111的分组到达路由器。该路由器在它的转发表中索引，决定该分组的输出链路接口是接口2。该路由器则在内部将该分组转发到接口2。在4.3节中我们将深入路由器内部，更为详细地研究转发功能。

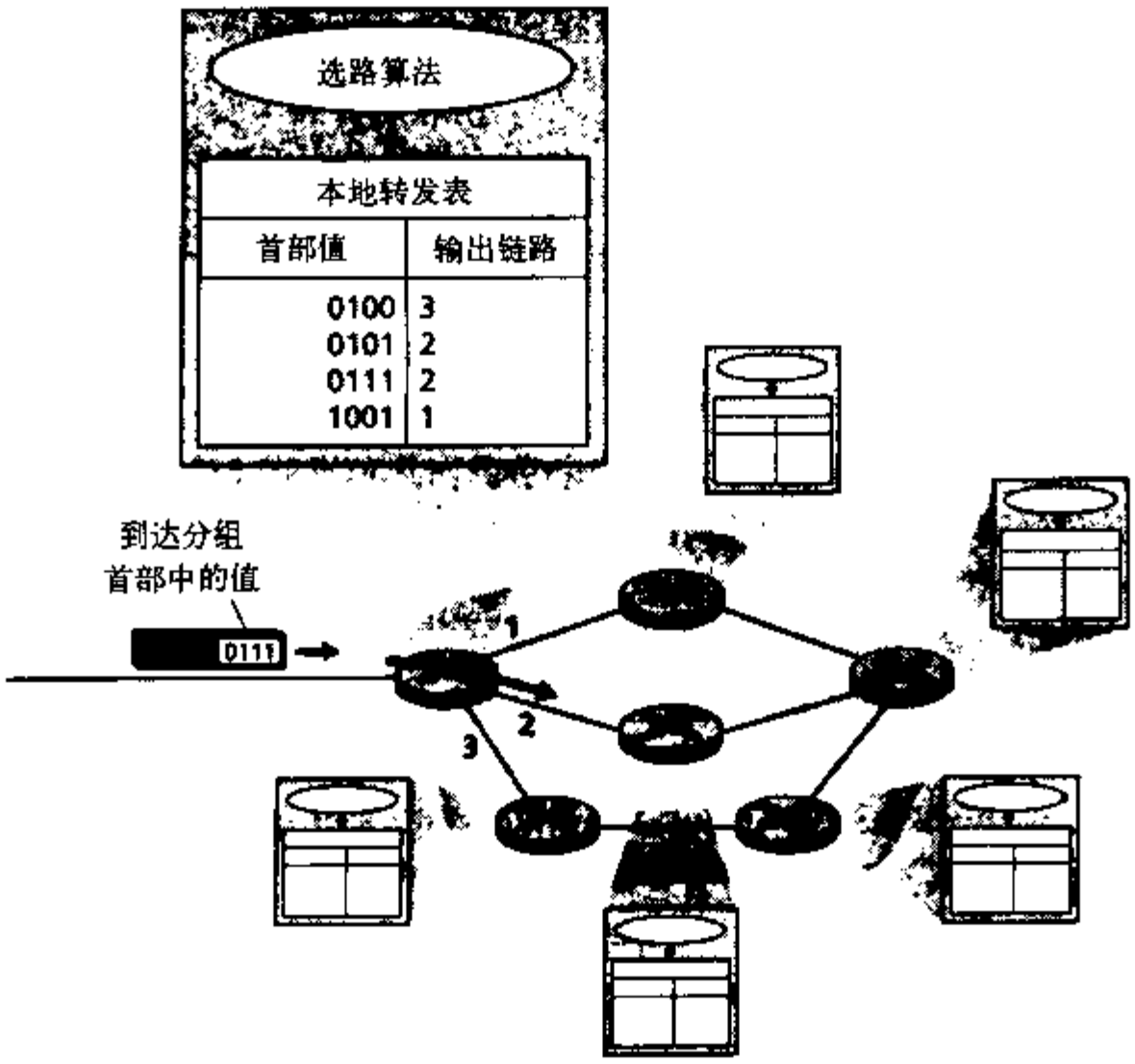


图4-2 选路算法决定转发表中的值

你也许现在想知道路由器中的转发表是如何配置的。这是一个关键问题，它揭示了选路和转发间的重要相互作用关系。如图4-2所示，选路算法决定了插入该路由器的转发表中的值。选路算法可能是集中式的（例如，该算法在某个中心点执行，并向每台路由器下载选路信息），也可能是分布式的（例如，分布式选路算法的各部分运行在每台路由器上）。在任何一种情形下，路由器都接收选路协议报文，并用于配置其转发表。通过考虑网络中的一种假想情况（并不真实，但技术上是可行的），其中转发表的内容由网络操作人员直接配置在路由器中物理上存在的所有转发表中，可进一步说明转发和选路功能的区别和不同的目的。在这种情况下，不需要任何选路协议！当然，操作人员需要彼此交互，以确保该转发表配置得能使分组到达它们想要到达的目的地。很可能出现下列情况：人工配置非常容易出错，并且对于网络

拓扑变化的响应比选路协议更慢。我们因所有网络具有转发和选路功能而感到幸运。

在讨论术语时，需要提及经常互换使用的两个其他术语，而我们将更谨慎地使用它们。我们将约定术语分组交换机是指一台通用分组交换设备，它根据分组首部字段中的值，从输入链路接口到输出链路接口传送分组。某些分组交换机称为**链路层交换机**（link-layer switch）（将在第5章研究），它们基于链路层字段中的值作转发决定。其他分组交换机称为**路由器**（router），它们基于网络层字段中的值作转发决定。（为了全面理解这种重要区别，你可能要回顾1.5.2节，在那里我们讨论了网络层数据报和链路层帧及其关系。）因为本章关注的是网络层，所以我们使用术语路由器代替分组交换机。当讨论虚电路网络中的分组交换机时，我们甚至使用术语路由器（很快将讨论）。

连接建立

前面说过网络层有两个重要的功能——转发和选路。但我们很快将看到在某些计算机网络中，实际上有第三种重要的网络功能，即**连接建立**（connection setup）。回想我们学习TCP时，数据从发送方流向接收方之前需要三次握手。这使得发送方和接收方建立所需的状态信息（例如，序号和初始流控制窗口长度）。以类似的方式，某些网络层体系结构（如ATM、帧中继）而非因特网，要求从源到目的地沿着所选择的路径彼此握手，以便在网络层数据分组能够开始流动之前，给定的源到目的地连接之间建立起状态。在网络层，该过程被称为*连接建立*。我们将在4.2节中研究连接。

4.1.2 网络服务模型

在钻研网络层之前，我们将以开阔的视野来考虑网络层可能提供的不同类型的服务。当位于发送主机的运输层向网络传输分组时（即在发送主机中将分组向下交给网络层），该运输层能指望网络层将该分组交付给目的地吗？当发送多个分组时，它们会按发送顺序按序交付给接收主机的运输层吗？发送两个连续分组的时间间隔与接收到这两个分组的时间间隔相同吗？网络层会提供关于网络中拥塞的反馈信息吗？在发送主机与接收主机中连接运输层通道的抽象视图（特性）是什么？这些问题和其他一些问题的答案由网络层提供的服务模型所确定。**网络服务模型**（network service model）定义网络的一侧边缘到另一侧边缘之间（即发送端系统与接收端系统之间）分组的端到端运输特性。

我们现在考虑网络层能够提供的某些可能的服务。在发送主机中，当运输层向网络层传递一个分组时，能由网络层提供的特定服务包括：

- *确保交付*。该服务确保分组将最终到达其目的地。
- *具有时延上界的确保交付*。该服务不仅确保分组的交付，而且在特定的主机到主机时延上界内（如100 ms内）交付。

此外，下列服务能够为给定的源和目的之间提供分组的流：

- *有序分组交付*。该服务确保分组以它们被发送的顺序到达目的地。
- *确保最小带宽*。这种网络层服务模仿在发送主机和接收主机之间（即使实际的端到端路径可能跨越几条物理链路）一条特定比特率（例如1Mbps）的传输链路的行为。只要发送主机以低于特定比特率的速率传输比特（作为分组的一部分），分组就不会丢失，且每个分组会在预定的主机到主机时延内（如40 ms内）到达。
- *确保最大时延抖动*。该服务确保发送方发送的两个相继分组之间的时间量等于在目的地接收到它们之间的时间量（或这种间隔的变化不超过某些特定的值）。

- 安全性服务。使用仅仅源和目的主机知晓的秘密会话密钥，源主机中的网络层能够加密向目的主机发送的所有数据报负载。目的主机中的网络层则负责解密该负载。这种服务能够为源和目的主机之间的所有运输层报文段（TCP和UDP）提供机密性。除了机密性以外，网络层还能够提供数据完整性和源鉴别服务。

这只是网络层能够提供的服务的部分列表，有无数种可能的服务变种。

因特网的网络层提供了单一的服务，称为**尽力而为服务**（best-effort service）。从表4-1中可以看出，尽力而为服务似乎是根本无服务的一种委婉说法。使用尽力而为服务，分组间的定时是得不到保证的，分组接收的顺序也不能保证与发送的顺序一致，传送的分组也不能保证最终交付。根据这样的定义，一个没有向目的地交付分组的网络也符合尽力而为交付服务的定义。然而，如我们很快要讨论的那样，这种最低要求的网络服务模型是有一些正当原因的。我们将在第7章中学习另外的、仍然在演化的因特网服务模型。

一些其他的网络体系结构已定义和实现了超过因特网尽力而为服务的服务模型。例如，ATM网络体系结构［ATM Forum 2007, Black 1995］提供了多重服务模型，意味着可以在同一网络中为不同的连接提供不同类别的服务。对ATM网络如何提供这样服务的讨论已经超出了本书的范围，我们的目的仅在于说明除了因特网尽力而为模型外，还存在其他模型。两个最重要的ATM服务模型是恒定比特率和可用比特率服务：

- **恒定比特率**（Constant Bit Rate，CBR）ATM网络服务。这是第一个标准化的ATM服务模型，它反映了电话公司对ATM的早期兴趣，以及CBR服务适合用于承载实时、恒定比特率的音频和视频流量。CBR服务的目标从概念上讲是很简单的，就是使网络连接看起来就像在发送主机与接收主机之间存在一条专用的固定带宽的传输链路，以使用性质相同的虚拟管道来提供分组（用ATM术语叫做信元）的流。使用CBR服务，ATM信元流以如下方式被承载跨越网络，即一个信元的端到端时延、一个信元端到端时延中的可变性（即时延抖动）及丢失或推迟交付的信元的比率都可以确保在规定值以下。当第一次建立CBR连接时，发送主机与ATM网络认可这些值。
- **可用比特率**（Available Bit Rate，ABR）ATM网络服务。由于因特网提供了所谓尽力而为服务，所以ABR也许最好被刻画为比尽力而为服务稍好一点的服务。如同因特网服务模型一样，ABR服务也许会丢失信元。然而与因特网服务模型不同的是，信元不能被重排序（虽然它们也可能丢失），对于使用ABR服务的连接来说，最小信元传输速率（MCR）是可以得到保证的。如果在给定时间内网络有足够的空闲资源，发送方也可以用比MCR更高的速率成功地发送信元。另外，如3.6节所述，ATM ABR服务能够为发送方提供反馈信息（利用一个拥塞通知比特，或以明确的速率发送），以便控制发送方在MCR和一个允许的峰值信元速率之间调整其速率。

表4-1 因特网、ATM CBR和ATM ABR服务模型比较

网络体系结构	服务模型	带宽保证	无丢失保证	排 序	定 时	拥塞指示
因特网	尽力而为	无	无	任何可能的顺序	不维持	无
ATM	CBR	保证恒定速率	是	有序	维持	不出现拥塞
ATM	ABR	保证最小速率	无	有序	不维持	提供拥塞指示

4.2 虚电路和数据报网络

回想第3章，运输层能够为应用程序提供无连接服务或面向连接服务。例如，因特网的运

输层为每个应用程序在两种服务间提供了选择：UDP，一种无连接服务；TCP，一种面向连接服务。以类似的方式，网络层也能够提供无连接服务或连接服务。网络层的连接和无连接服务在许多方面等同于运输层的面向连接和无连接服务。例如，网络层连接服务以源和目的主机间的握手开始，网络层无连接服务则没有任何握手预备步骤。

尽管网络层连接和无连接服务与运输层面向连接和无连接服务有类似之处，但也存在重要差异：

- 在网络层中，这些服务是由网络层向运输层提供的主机到主机的服务。在运输层中，这些服务则是运输层向应用层提供的进程到进程的服务。
- 在迄今为止的所有主要的计算机网络体系结构（因特网、ATM、帧中继等）中，网络层或者提供了主机到主机的无连接服务，或者提供了主机到主机的连接服务，而不同时提供两种。仅在网络层提供连接服务的计算机网络被称为**虚电路**（Virtual-Circuit, VC）网络；仅在网络层提供无连接服务的计算机网络被称为**数据报网络**（datagram network）。
- 在运输层实现面向连接的服务与在网络层实现连接服务是根本不同的。我们在前面一章看到，运输层面向连接服务是在位于网络边缘的端系统中实现的；我们很快将看到，网络层连接服务除了在端系统中实现外，也在位于网络核心的路由器中实现。

虚电路和数据报网络是计算机网络的两种基本类型。在作出转发决定时，它们使用了截然不同的信息。我们现在详细探讨它们的实现。

4.2.1 虚电路网络

我们已经知道因特网是数据报网络。然而，许多其他网络体系结构（包括ATM、帧中继）都是虚电路网络，因此在网络层使用连接。这些网络层连接被称为**虚电路**。我们现在考虑在计算机网络中如何实现虚电路服务。

一条虚电路（VC）的组成如下：① 源和目的主机之间的路径（即一系列链路和路由器）；② VC号，沿着该路径的每段链路一个号码；③ 沿着该路径的每台路由器中的转发表表项。属于一条虚电路的分组将在它的首部携带一个VC号。因为一条虚电路在每条链路上可能具有不同的VC号，所以每台中间路由器必须用一个新的VC号替代每个传输分组的VC号。该新的VC号从转发表获得。

为了举例说明这个概念，考虑图4-3中的网络。图4-3中靠近R1链路的号码是链路接口号。现在假定主机A请求该网络在它自己与主机B之间创建一条虚电路。同时假定该网络为该虚电路选择路径A-R1-R2-B，并为这条路径上的这3条链路分配VC号12、22和32。在这种情况下，当这条虚电路中的分组离开主机A时，该分组首部中的VC字段的值是12；当它离开R1时该值是22，而当它离开R2时该值是32。

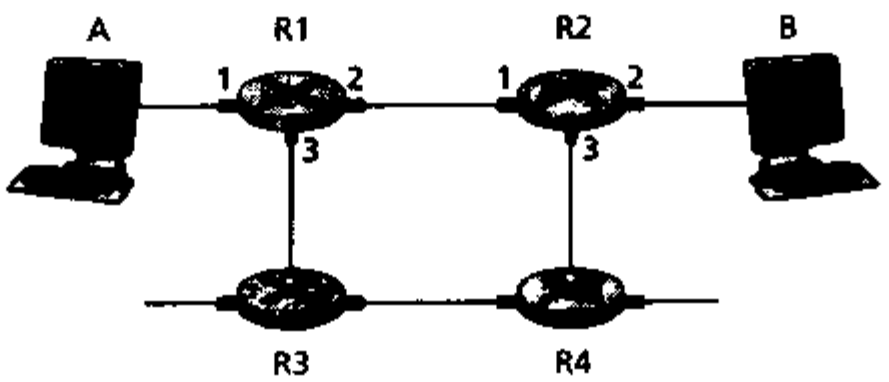


图4-3 一个简单的虚电路网络

对于分组通过某路由器，该路由器怎样决定VC号的更换呢？对于虚电路网络，每台路由器的转发表包括了VC号的转换。例如，R1中的转发表可能有些像下表：

入接口	入VC号	出接口	出VC号
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

当跨越一台路由器创建一条新的虚电路时，转发表就增加一个新项。类似地，无论何时终止一条虚电路，就删除沿着该路径每个表中的相应项。

你也许想知道为什么一个分组沿着其路由在每条链路上不能保持相同的VC号。回答有两个方面。第一，逐链路代替该号码减少了分组首部中VC字段的长度。第二方面更为重要，通过允许沿着该虚电路路径每条链路有一个不同的VC号，大大简化了虚电路的建立。特别是具有多个VC号，该路径上的每条链路能够选择一个VC号，独立于沿着该路径的其他链路所选的VC号。如果沿着某路径的所有链路需要一个共同的VC号的话，路由器将不得不交换并处理相当大量的报文，以约定一个共同的VC号（例如，一个并未由这些路由器任何其他现有虚电路使用的号码）来用于一次连接。

在虚电路网络中，该网络的路由器必须为进行中的连接维持**连接状态信息**（connection state information）。特别是，每当跨越一台路由器创建一个新连接时，就必须将一个新的连接项加到该路由器转发表中；每当释放一个连接时，就必须从该表中删除该项。注意，即使没有VC号转换，也有必要维持连接状态信息，该信息将VC号与输出接口号联系起来。路由器是否对每条进行中的连接维持连接状态信息是一个关键问题，我们在本书中将再次讨论该问题。

在虚电路中有3个明显不同的阶段：

- *虚电路建立*。在建立阶段，发送运输层与网络层联系，指定接收方地址，等待该网络建立虚电路。网络层决定发送方与接收方之间的路径，即该虚电路的所有分组要通过的一系列链路与路由器。网络层也为沿着该路径的每条链路确定一个VC号。最后，网络层在沿着路径的每台路由器的转发表中增加一项。在虚电路建立期间，网络层还可以预留虚电路路径上的资源（如带宽）。
- *数据传送*。如图4-4所示，一旦创建了虚电路，分组就可以开始沿该虚电路流动了。

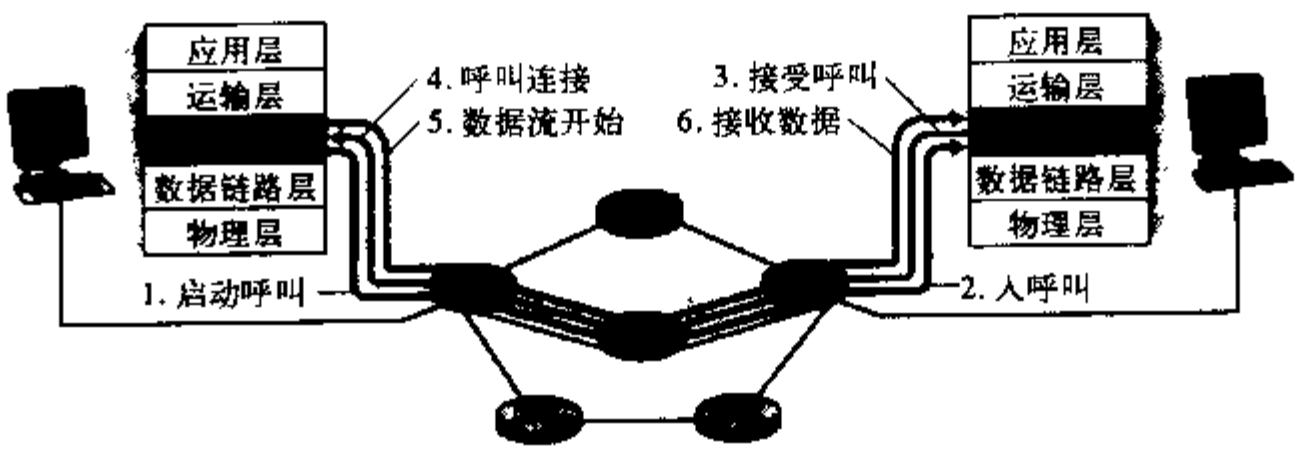


图4-4 虚电路建立

- 虚电路拆除。当发送方（或接收方）通知网络层它想终止该虚电路时，就启动这个阶段。网络层通常将通知网络另一侧的端系统结束呼叫，并更新路径上每台路由器中的转发表以表明该虚电路已不存在了。

在网络层的虚电路建立与运输层的连接建立（例如，第3章学过的TCP三次握手）之间有一个细微但很重要的区别。运输层的连接建立仅涉及两个端系统。在运输层连接建立期间，两个端系统独自决定运输层连接的参数（如初始序号与流量控制窗口大小）。虽然这两个端系统已经知道该运输层连接，但网络中的路由器对这些完全不知情。另一方面，对于一个虚电路网络层，沿两个端系统之间路径上的路由器都要参与虚电路的建立，且每台路由器都完全知道经过它的所有虚电路。

端系统向网络发送指示虚电路启动与终止的报文，以及路由器之间传递的用于建立虚电路（即修改路由器表中的连接状态）的报文被称为**信令报文**（signaling message），用来交换这些报文的协议常称为**信令协议**（signaling protocol）。虚电路建立如图4-4所示。在本书中我们将不涉及虚电路信令协议；有关面向连接网络中信令的一般讨论参见[Black 1997]，有关ATM的Q.2931信令协议的规范参见[ITU-T Q.2931 1994]。

4.2.2 数据报网络

在**数据报网络**（dadgram network）中，每当一个端系统要发送分组时，它就为该分组加上目的地端系统的地址，然后将该分组推进网络中。如图4-5所示，完成这些无需建立任何虚电路。在数据报网络中，路由器不维护任何有关虚电路的状态信息。（因为没有虚电路！）

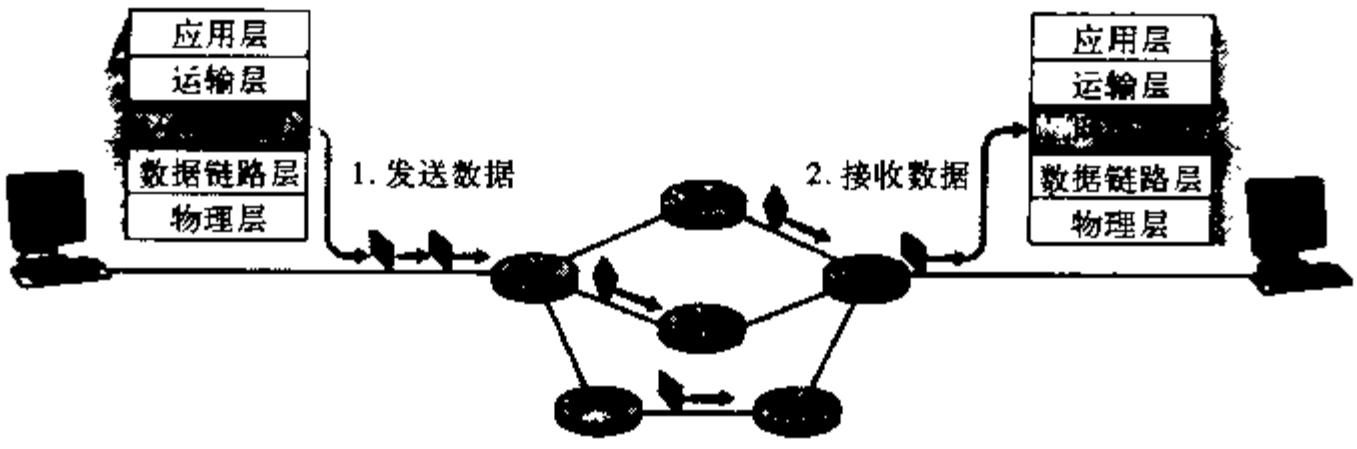


图4-5 数据报网络

随着分组从源向目的地传输，它通过一系列路由器。这些路由器中的每个都使用该分组的目的地址来转发该分组。特别是，每台路由器有一个将目的地址映射到链路接口的转发表；当分组到达路由器时，该路由器使用该分组的目的地址在该转发表中查找适当的输出链路接口。然后，路由器有意识地将该分组向该输出链路接口转发。

为了进一步理解查找操作，我们来看一个特定的例子。假定所有的目的地址都是32比特（这恰好是IP数据报中的目的地址的长度）。蛮力实现转发表，使得对于每个可能的目的地址有一个项。因为有超过40亿个可能的地址，所以这是很难做到的，它将要求一个其大无比的转发表。

现在我们进一步假设路由器具有4条链路，编号从0到3，分组以如下方式转发到链路接口：

目的地址范围	链路接口
11001000 00010111 00010000 00000000	
到	0
11001000 00010111 00010111 11111111	
11001000 00010111 00011000 00000000	
到	1
11001000 00010111 00011000 11111111	
11001000 00010111 00011001 00000000	
到	2
11001000 00010111 00011111 11111111	
其他	3

显然，对于这个例子，在该路由器的转发表中就没有必要有40亿项。例如，只有4个表项的下列转发表就够了：

前缀匹配	链路接口
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
其他	3

使用这种风格的转发表，该路由器用该分组的目的地址的前缀（prefix）与该表中的表项进行匹配；如果存在匹配，该路由器向与该匹配相联系的链路转发该分组。例如，假设该分组的目的地址是11001000 00010111 00010110 10100001。因为该地址的21比特前缀匹配该表中的第一项，所以路由器向链路接口0转发该分组。如果一个前缀不匹配前3项中的任何一项，则该路由器向接口3转发该分组。尽管听起来足够简单，但这里还是有重要技巧的。你可能已经注意到了，一个目的地址可能匹配多项。例如，地址11001000 00010111 00010000 10101010的前24比特与表中的第二项匹配，而前21比特与表中的第三项匹配。当有多个匹配时，该路由器使用**最长前缀匹配规则**（longest prefix matching rule），即在该表中寻找最长的匹配项，并向与最长前缀匹配的链路接口转发该分组。

当然，要使最长匹配有效，每个输出链路接口应当负责转发大块的连续目的地址。我们将在4.4节中看到，因特网地址通常以层次方式来分配，因此这种连续的特性在多数路由器的转发表中是流行的。况且，因特网研究界对越来越多的空洞正在刺穿地址空间表示关注，这使得连续块变得越来越小，且转发表变得越来越大（参见[Meng 2005]、[RFC 3221]和4.4节实践原则中的讨论）。

虽然数据报网络中的路由器不维持连接状态信息，但它们无疑在其转发表中维持了转发状态信息。然而，转发状态信息变化的时间尺度相对要慢。实际上，在数据报网络中，转发表是由选路算法修改的，通常每1～5分钟左右更新一次转发表。在虚电路网络中，只要通过路由器建立一条新的连接，或通过路由器拆除一条现有的连接，就更新路由器中的转发表。这对一台第一层主干路由器而言，很容易以微秒的时间尺度进行更新。

因为数据报网络中的转发表能够在任何时刻修改，所以从一个端系统到另一个端系统发送一系列分组可能在通过网络时走不同的路径，并可能无序到达。[Paxson 1997]和[Jaiswal 2003]对公共因特网上的分组重排和其他现象进行了有趣的测量研究。

4.2.3 虚电路和数据报网络的由来

数据报与虚电路网络的演化反映了它们的由来。作为一条重要的组织原则，虚电路的概念来源于电话界，它采用了真正的电路。由于呼叫建立及每次呼叫的状态要在网络中的路由器上维持，因此面向虚电路的网络显然比数据报网络要复杂得多（对于电路交换与分组交换网络的复杂性的有趣比较可参看[Molinero-Fernandez 2002]）。这也是保持了它的电话传统。电话网络在其网络中必然具有某种复杂性，因为它们要连接哑端系统设备，如转盘电话（转盘电话是一种无按键（仅是一个拨号盘）的模拟电话）。

另一方面，因特网作为一种数据报网络，是由互连计算机的需求发展而来的。由于端系统设备复杂得多，因此因特网设计者们选择使网络层服务模型尽可能简单。如第2、3章中所述，另外的功能（例如，按序传送、可靠数据传输、拥塞控制与DNS名字解析）在端系统中的更高层实现。这正好与电话网模型相反，并产生了一些有趣的结果。

- 提供最少（甚至没有）服务保证（并因此对网络层施加了最小限度的需求）的因特网服务模型，使得互连使用各种不同链路层技术的网络变得更加容易。这些链路层技术包括卫星、以太网、光纤或无线，而且具有不同的传输速率和丢包特性。我们将在4.4节中详细地讨论IP网络的互连。
- 如第2章所述，诸如电子邮件、Web等应用，甚至以网络层为中心的服务（如DNS）都是在位于网络边缘的主机（服务器）上实现的。增加一项新服务能力只需将一台主机连接到网络中，并定义一个新的应用层协议（如HTTP）即可，这种能力可以使Web之类的新服务在相当短的时间内在因特网上得到应用。

然而，如我们将在第7章所见，关于如何演化因特网的网络层体系结构以支持多媒体之类的实时服务，在因特网业界内还存在着相当多的争论。对面向虚电路的ATM网络体系结构与被提议的下一代因特网体系结构的一个有趣的比较在［Crowcroft 1995］中给出。

4.3 路由器工作原理

既然我们已经看到了网络层的功能和服务概况，我们将注意力转向网络层的**转发功能**(forwarding function)，即实际将分组从一台路由器的入链路传送到适当的出链路。我们已经在4.2节中简要地学习了一些转发问题，也就是编址和最长前缀匹配。在本节中我们将观察用于从入链路到出链路传输分组的特定路由器体系结构。我们这里涉及的内容有必要简洁一些，因为深入地讨论路由器设计需要完整的课程才能做到。因此，我们将在本节中提供一些关于更深入研究该主题的材料指南。我们以前提到过，计算机网络研究人员和从业人员经常互换使用转发和交换这两个词，我们也将在本书中使用这两个词。

图4-6显示了一个通用路由器体系结构的总体视图。其中标识了一台路由器的4个组成部分。

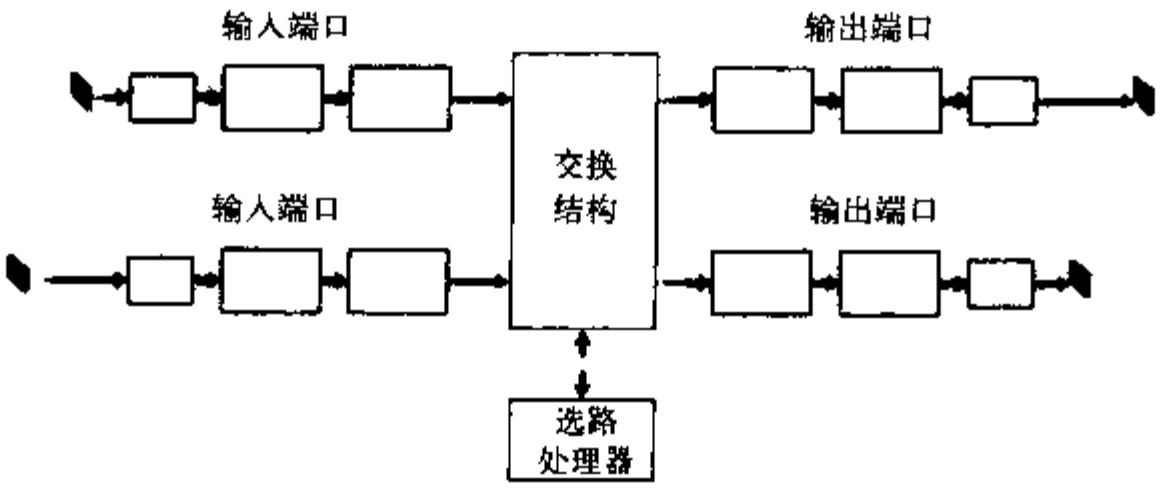


图4-6 路由器体系结构

- 输入端口。输入端口执行几项功能。它要执行将一条输入的物理链路端接到路由器的物理层功能（图4-6中输入端口部分最左侧的方框与输出端口部分最右侧的方框）。它也要执行需要与位于入链路远端的数据链路层功能交互的数据链路层功能（由输入端口与输出端口部分中间方框表示）。它还要完成查找与转发功能（输入端口部分最右侧的方框与输出端口部分最左侧的方框），以便转发到路由器交换结构部分的分组能出现在适当的输出端口。控制分组（如携带选路协议信息的分组）从输入端口转发到选路处理器。实际上，在路由器中，多个端口经常被集中到路由器中的一块线路卡（line card）上。
- 交换结构。交换结构将路由器的输入端口连接到它的输出端口。交换结构完全包容在路由器中，即它是一台网络路由器中的网络！
- 输出端口。输出端口存储经过交换结构转发给它的分组，并将这些分组传输到输出链路。因此，输出端口执行与输入端口顺序相反的数据链路层和物理层功能。当一条链路是双向（承载两个方向的流量）链路时，与链路相连的输出端口通常与输入端口在同一线路卡上成对出现。
- 选路处理器。选路处理器执行选路协议（例如，我们将在4.6节中学习的协议），维护选路信息与转发表，并执行路由器中的网络管理功能（参见第9章）。

在下面各小节中，我们将更为详细地考察输入端口、交换结构和输出端口。[Chuang 2005; Keslassy 2003; Chao 2001; Turner 1988; Giacopelli 1990; McKeown 1997a; Partridge 1998] 提供了对一些特定路由器体系结构的讨论。[McKeown 1997b] 提供了一个可读性很强的现代路由器体系结构综述，其中使用Cisco 12000路由器作为例子。为了具体起见，后继的讨论假定计算机网络是分组交换网络，转发决定基于分组的目的地址（而非虚电路网络中的VC号）。然而，对于虚电路网络来说，概念和技术也是类似的。

历史事件

Cisco系统：主宰网络的核心

在写本书时（2006年10月），Cisco公司雇佣了30 000多人，公司市值1400亿美元。Cisco目前主宰着因特网路由器市场，并在最近几年迅速进入了因特网电话市场。在这个市场中，它与电话设备公司（如Lucent、Alcatel、Nortel和Siemens等）之间开始了直接的竞争。这个网络公司巨人是如何发展起来的呢？它是从1984年在硅谷公寓的一间起居室里开始发展的。

当Len Bosak与他的妻子Sandy Lerner在斯坦福大学工作时，他们就有建造并出售因特网路由器给研究及学术团体的想法。Sandy Lerner起了一个名字Cisco（这是San Francisco的一种略称），还设计了公司的桥形标志图案。公司总部最初是他们的起居室，而且刚开始时他们通过信用卡和兼职咨询工作为该项目筹措资金。到了1986年末，Cisco的月收入达到了250 000美元。到1987年末，Cisco以1/3公司股份为交换条件，终于成功吸引了来自Sequoia Capital的20亿美元的风险投资。在后来的几年中，Cisco持续增长并抢占了越来越多的市场份额。与此同时，Bosak/Lerner与Cisco管理层之间的关系变得紧张起来。Cisco在1990年公开发售股票，但在同年，Lerner与Bosak离开了公司。

多年来，Cisco对非路由器市场进行了成功的扩展，销售安全、无线和IP话音产品。然而，Cisco正面临着日益增长的国际竞争，包括来自华为（一个迅速增长的中国网络设

备公司）的竞争。华为在全世界有38 000多名雇员，最近报道称它拥有超过7%的以太网交换机和路由器市场。在路由器和交换以太网领域与Cisco竞争的公司是Alcatel-Lucent和Juniper。

4.3.1 输入端口

图4-7中给出一个更详细的输入端口功能的视图。如前面讨论的那样，输入端口的线路端接功能与数据链路处理实现了与通向路由器的各个输入链路相关的物理层和数据链路层。输入端口的查找/转发模块对于路由器的转发功能是至关重要的。在许多路由器中，都是在这里来确定一个到达的分组经交换结构转发给哪个输出端口。输出端口的选择是通过使用转发表中包含的信息进行的。虽然转发表是由选路处理器计算的，但通常一份转发表的影子拷贝会被存放在每个输入端口，而且会被更新，这也正是选路处理器所需要的。有了转发表的本地拷贝，就可在每个输入端口本地做出转发决策，而无需调用中央选路处理器。这种分散式的转发可避免在路由器中的某个单点产生转发处理的瓶颈。

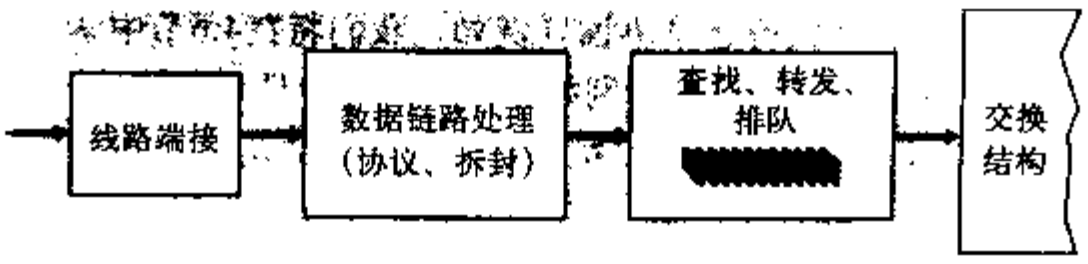


图4-7 输入端口处理

在输入端口处理能力受限制的路由器中，输入端口也许直接将分组转发给中央选路处理器，然后该处理器执行转发表查找并将分组转发到恰当的输出端口。这就是当一个工作站或服务器用作一台路由器时所采用的方法。这里，选路处理器实际就是该工作站的CPU，而且输入端口实际上就是一块网络接口卡（如一块以太网网卡）。

转发表存在后，从概念上讲表查找是很简单的，我们只需搜索转发表，如4.2.2节所描述的那样查找最长前缀匹配。然而，在实际中事情并不是那么简单。也许最重要的复杂化因素是，主干路由器必须高速运行，每秒执行几百万次的查找工作。实际上，人们希望输入端口的处理速度能够达到**线路速度**（line speed），即执行一次查找的时间应少于从输入端口接收一个分组所需的时间。在这种情况下，对收到的分组的输入处理可以在下一个接收操作结束之前完成。为了得到对查找性能需求的概念，可考虑运行速率为2.5Gbps的OC-48链路。对于256 字节长的分组，这意味着查找速度大约为每秒进行一百万次的查找。

给定以目前的高链路速率运行的需求，则在一个庞大的转发表中进行线性搜索是不可能的。一种更合理的技术是将转发表表项存放在一个树形数据结构中。树中的每一级可以认为是与目的地址中的1比特相对应。若要查找一个地址，只需从树的根节点开始。如果地址的第一个比特为0，则左子树包含与该目的地址相对应的转发表表项；反之，表项在右子树中。此后，通过使用剩余的地址比特，能遍历某棵合适的子树（若下一比特为0，则选择初始子树的左子树，否则选择初始子树的右子树）。按照这种方式，可以在N步之内找到相应的转发表表项，其中N是地址中的比特数。（注意，这本质上是一个地址空间为2^N的二分查找。）[Srinivasan 1999] 描述了一种二分查找技术的改进，对分组分类算法的一般性展望可在[Gupta 2001] 中找到。

但即使经过$N = 32$（如一个32比特的IP地址）步，使用二分查找的查找速度对于当今的主

干选路需求来说还是不够快。例如，假设每个步骤访问一次内存，用40ns的内存访问时间可执行每秒少于一百万次的地址查找。因此，为提高查找速度研究出几种技术。**内容可寻址内存**（Content Addressable Momery, CAM）允许一个32比特IP地址提交给CAM，由它再以基本上常数时间返回该地址对应的转发表表项内容。Cisco 8500系列路由器的每个输入端口都有一个64K的CAM。

另一种加快查找速度的技术是将最近访问的转发表表项保存在高速缓存中［Feldmeier 1988］。这里，关注的是高速缓存的潜在大小。最近，甚至提出了更快的数据结构，这些数据结构可以使转发表表项在log(*N*)步内找到［Waldvogel 1997］，或用新奇的方式压缩转发表［Brodnik 1997］。一种优化的、适用于一般情况（待查地址具有24或更少个有效比特）的、基于硬件的查找方法在［Gupta 1998］中进行了讨论。对于高速IP地址查找算法的展望和分类法参见[Ruiz-Sanchez 2001]。

一旦通过查找确定了一个分组的输出端口，则该分组可转发进入交换结构。然而，一个分组可能会在进入交换结构时暂时阻塞（blocked），这是由于来自其他输入端口的分组当前正在使用该交换结构。因此，一个被阻塞的分组必须要在输入端口处排队，并等待稍后被及时调度以通过交换结构。我们将在4.3.4节中进一步研究路由器中分组（包括位于输入端口与输出端口中的分组）的阻塞、排队与调度。

4.3.2 交换结构

交换结构位于一台路由器的核心部位。正是通过交换结构，分组才能实际地从一个输入端口交换（即转发）到一个输出端口中。交换可以通过许多方式完成，如图4-8所示。

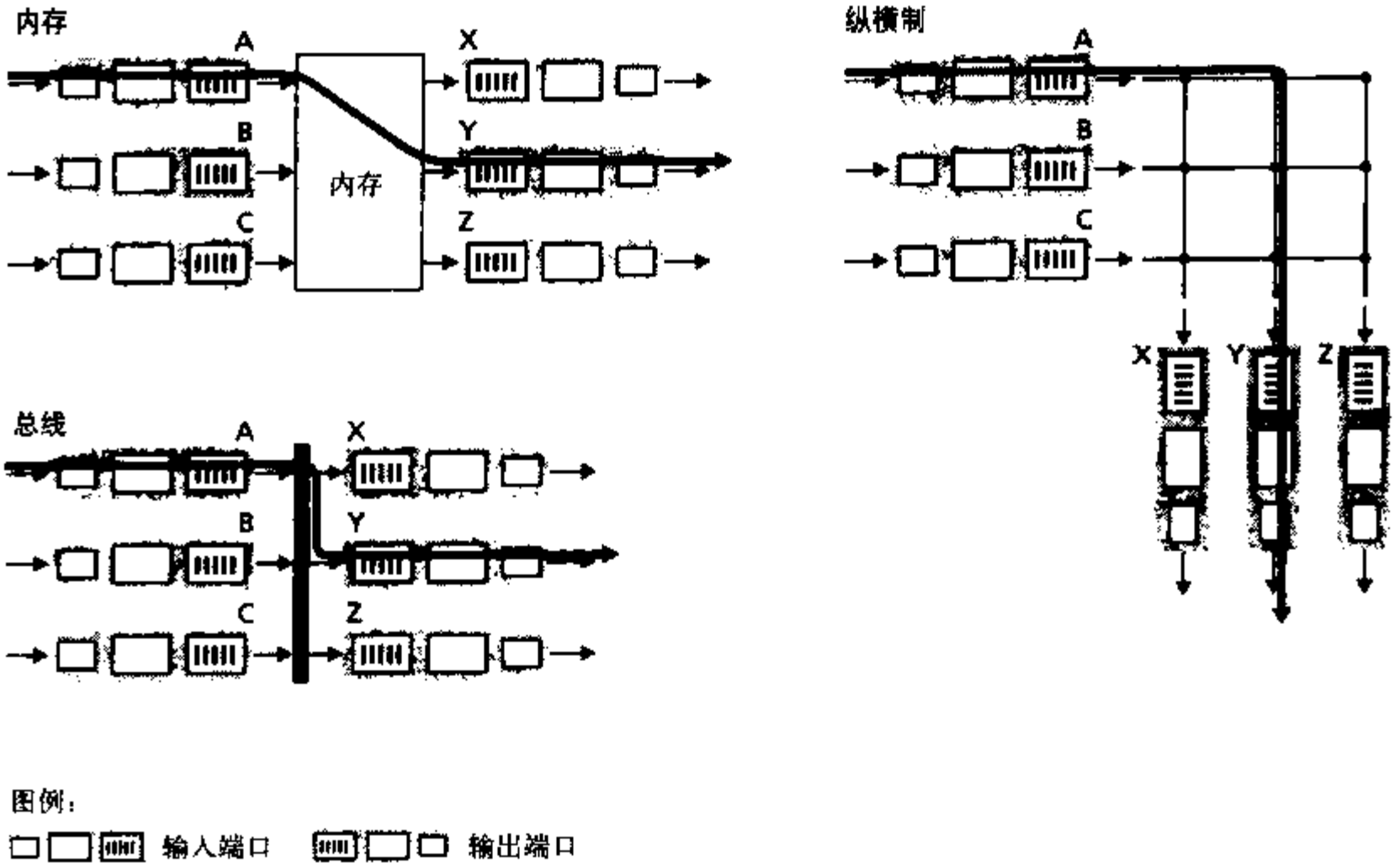


图4-8 三种交换技术

图4-8中的三种交换技术是：

- 经内存交换。最简单、最早的路由器通常是传统的计算机，在输入端口与输出端口之间

的交换是在CPU（选路处理器）的直接控制下完成的。输入端口与输出端口的作用就像传统操作系统中的I/O设备一样。一个分组到达一个输入端口时，该端口会先通过中断方式向选路处理器发出信号。于是，该分组从输入端口处被拷贝到处理器内存中。选路处理器则从分组首部中取出目的地址，在转发表中找出适当的输出端口，并将该分组拷贝到输出端口的缓存中。注意，若内存带宽为每秒可写进或读出B个分组，则总的转发吞吐量（分组从输入端口被传送到输出端口的总速率）必然小于$B/2$。

许多现代路由器也是通过内存交换。然而，与早期路由器的一个主要差别是，目的地址的查找和将分组存储（交换）进适当的存储位置是由输入线路卡上的处理器来执行的。在某些方面，经内存交换的路由器看起来很像共享内存的多处理器，用一个线路卡上的处理器将分组交换进适当输出端口的内存中。Cisco的Catalyst 8500系列交换机[Cisco 8500 2007]是经共享内存转发分组的。研究基于内存交换的性质和与其他形式交换的比较的一种抽象模型可以在[Iyer 2002]中找到。

- *经一根总线交换。*在这种方法中，输入端口经一根共享总线将分组直接传送到输出端口，不需要选路处理器的干预（注意，经内存交换时，分组进出内存也必须跨越系统总线）。虽然选路处理器没有涉及总线传送，但由于总线是共享的，故一次只能有一个分组通过总线传送。某个分组到达一个输入端口时，发现总线正忙于传送另一个分组，则它会被阻塞而不能通过交换结构，并在输入端口排队。因为每个分组必须跨过单一总线，所以路由器的交换带宽受总线速率的限制。

  在当今的技术下，总线带宽可能超过1 Gbps，对于运行在接入网或企业网（如局域网与公司网）中的路由器来说，通过总线交换通常是足够的。基于总线的交换目前已被相当多的路由器产品所采用，其中包括Cisco 5600[Cisco Switches 2007]，它通过一个32 Gbps背板总线来交换分组。

- *经一个互联网络交换。*克服单一、共享式总线带宽限制的一种方法是，使用一个更复杂的互联网络，如过去在多处理器计算体系结构中用来互连多个处理器的网络。一个纵横式交换机就是一个由$2n$条总线组成的互联网络，它将n个输入端口与n个输出端口连接，如图4-8所示。一个到达某个输入端口的分组沿着连到输入端口的水平总线穿行，直至该水平总线与连到所希望的输出端口的垂直总线的交叉点。如果该条连到输出端口的垂直总线是空闲的，则该分组被传送到输出端口。如果该垂直总线正用于传送另一个输入端口的分组到同一个输出端口中，则该到达的分组被阻塞，且必须在输入端口排队。

  Delta与Omega交换结构也已被提议为用于输入端口和输出端口之间的互联网络。对交换机体系结构的概述可参见[Tobagi 1990]。Cisco 12000系列交换机[Cisco 12000 2007]使用了一个互联网络，能提供高达60 Gbps的速率通过交换结构。目前在互联网络设计[Keshav 1998]中的一种趋势是，将长度变化的IP分组分片成固定尺寸的信元，加上标签并通过互联网络进行交换。这些信元在输出端口再被重新装配成初始分组。定长信元与内部标签能够大大简化并加快通过互联网络的分组交换。

4.3.3 输出端口

如图4-9所示，输出端口处理取出存放在输出端口内存中的分组并将其传送到输出链路上。数据链路协议处理与线路端接是发送方的链路层与物理层功能，这些功能实现与输出链路另一端的输入端口之间的交互，如4.3.1节所述。当交换结构将分组交付给输出端口的速率超过

输出链路速率时，就需要排队与缓存管理功能。我们将在下面阐述输出端口排队。

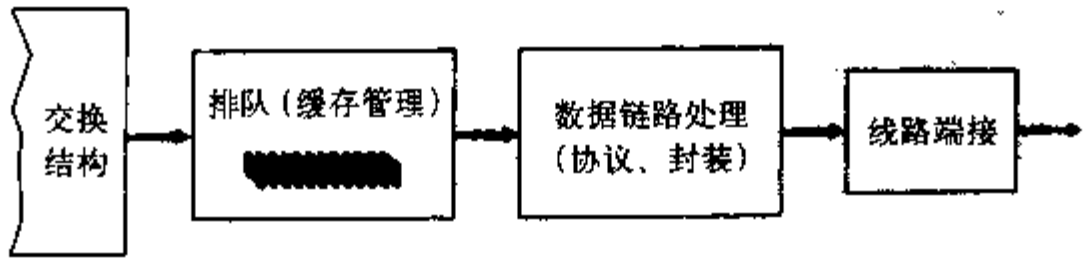


图4-9 输出端口处理

4.3.4 何时出现排队

如果我们观察显示在图4-8中的输入端口和输出端口功能以及配置，就会发现输入端口和输出端口处都能够形成分组队列。更为详细地考虑这些队列是很重要的，因为随着这些队列的增长，路由器的缓存空间将会最终耗尽，且会出现**丢包**（packet loss）。如前所述，分组在网络中丢失或在路由器中丢弃。就是这里，在一台路由器的这些队列中，这些分组丢失或被丢弃了。分组丢失的实际位置（要么在输入端口队列，要么在输出端口队列）将取决于流量负载、交换结构的相对速率和线路速率等几个因素。

假定输入线路速率与输出线路速率都是相同的，有n个输入端口和n个输出端口。定义**交换结构速率**（switching fabric speed）为交换结构能够从输入端口到输出端口移动分组的速率。如果交换结构的速率至少是输入线路速率的n倍，则在输入端口处不会出现排队。这是因为，即使在最坏情况下，所有n条输入线路都在接收分组，交换结构也能在每个输入端口（同时）接收一个分组的时间内，将n个分组从输入端口传送到输出端口。但是在输出端口处会发生什么呢？我们仍然假设交换结构的速率至少是线路速率的n倍。在最坏情况下，到达所有n个输入端口的分组要被发往同一个输出端口。在这种情况下，在它接收（或发送）一个分组的时间内，将有n个分组到达该输出端口。因为输出端口在一个单位时间（分组传输时间）内只能传送一个分组，所以n个到达的分组必须排队（等待）传输到输出链路上。于是，又有n个分组可能在它只能传送队列中一个分组的时间内到达。这种情况不断持续下去，最终排队的分组数会增长得很快，足以耗尽输出端口的存储空间，此时，分组就被丢弃了。

图4-10举例说明了输出端口的排队情况。在时刻t，每个输入端口都到达了一个分组，每个分组都是发往最上侧的输出端口的。假定线路速率相同，交换操作以三倍的线路速率进行，一个时间单位（即接收或发送一个分组所需的时间）以后，所有三个初始分组都被传送到输出端口，并排队等待传输。在下一个时间单位中，这三个分组之一将通过输出链路传送出去。在这个例子中，又有两个新分组已到达交换机的入端，其中的一个分组要发往最上侧的输出端口。

假定需要路由器缓存来吸收流量负载的波动，一个自然的问题就是需要多少缓存。多年以来，计算缓存长度的经验方法[RFC 3439]是缓存量（B）等于平均往返时延（RTT，比如说250 ms）乘以链路的容量（C）。这个结果是分析相对少量的TCP流的排队动态性得到的[Villamizar 1994]。因此，一条250 ms RTT的10 Gbps链路需要的缓存量是$B = \text{RTT} \cdot C = 2.5$ Gbps。然而，最近的理论和试验研究[Appenzeller 2004]提出，当大量的TCP流（N）经过一条链路时，所需要的缓存量是$B = \text{RTT} \cdot C/\sqrt{N}$。对于通常有大量流经过的大型主干路由器链路（参见[Fraleigh 2003]）来说，N的值可能非常大，因而所需的缓存长度将明显减小。[Appenzeller 2004]和[Wischik 2005]从理论、实现和运行的角度提供了可读性很强的有关缓存长度的讨论。

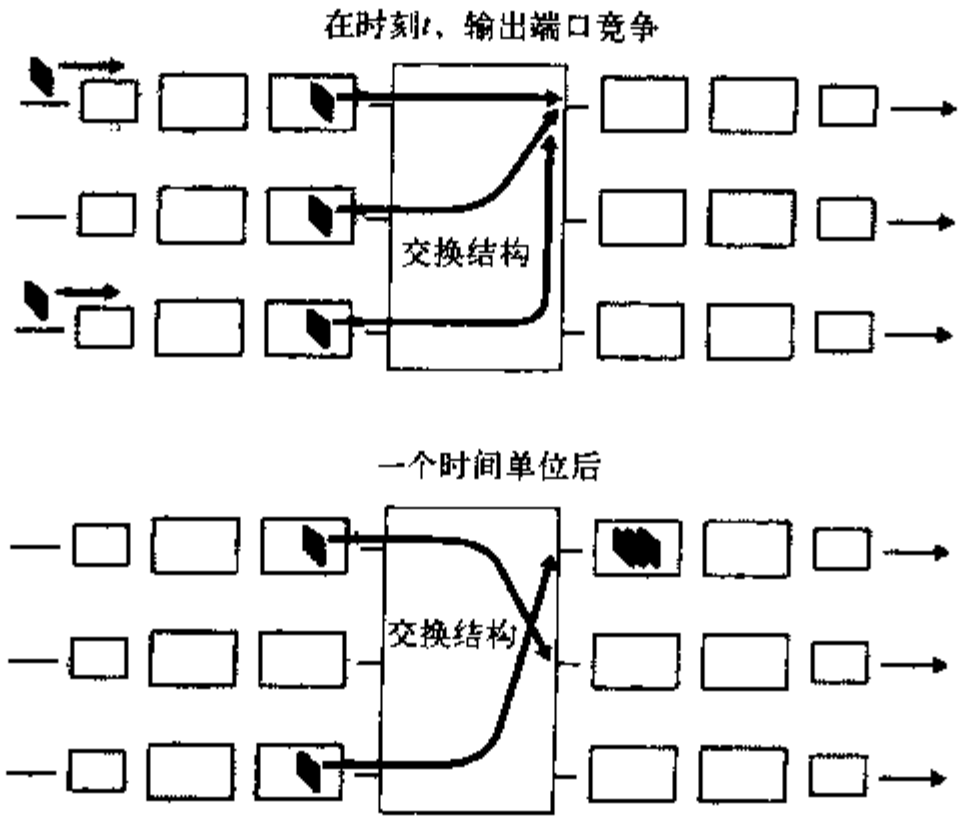


图4-10 输出端口排队

输出端口排队的后果就是，输出端口上的一个**分组调度程序**（packet scheduler）必须在这些排队的分组中选出一个来传送。这种选择可能是根据简单的原则来定，如先来先服务（FCFS）调度；也可能是根据更复杂的调度规则来定，如加权公平排队（WFQ）。WFQ规则是在具有排队等待传输的分组的不同端到端连接之间公平地共享输出链路。分组调度程序在提供**服务质量保证**（quality-of-service guarantee）方面起着关键作用。我们将在第7章更广泛地研究分组调度规则。有关输出端口分组调度规则的讨论参见［Cisco Queue 2007］。

类似地，如果没有足够的内存来缓存一个入分组，那么必须做出决定：要么丢弃到达的分组（一种称为**弃尾**（drop-tail）的策略），要么删除一个或多个已排队的分组以便为新来的分组腾出空间。在某些情况下，在缓存填满前便丢弃（或在首部加标记）一个分组，以便向发送方提供一个拥塞信号的做法是有益的。已经提出和分析了许多分组丢弃与标记策略［Labrador 1999; Hollot 2002］，这些策略统称为**主动队列管理**（Active Queue Management，AQM）算法。**随机早期检测**（Random Early Detection，RED）算法是一种得到最广泛地研究和实现的AQM算法之一。在RED算法中，为输出队列长度维护着一个加权平均值。如果平均队列长度小于最小阈值min_{th}，则当一个分组到达时，该分组被接纳进入队列。相反，如果队列满或平均队列长度大于最大阈值max_{th}，则当一个分组到达时，该分组被标记或丢弃。最后，如果一个分组到达，发现平均队列长度在［min_{th}，max_{th}］之间，则该分组以某种概率值被标记或丢弃，该概率值一般是与平均队列长度、min_{th}和max_{th}有关的函数值。已提出了许多概率标记/丢弃函数，各种版本的RED已被分析建模、模拟和/或实现。［Christiansen 2001］与［Floyd 2007］为提供了与这些问题相关的综述及指南。

如果交换结构不能快得（相对于输入线路速率而言）使所有到达分组无时延地通过它传送，则在输入端口也将出现分组排队，因为到达的分组必须加入输入端口队列中，以等待通过交换结构传送到输出端口。为了举例说明这种排队的重要后果，考虑一下纵横式交换结构并假定：①所有链路速率相同；②一个分组能够以与一条输入链路接收一个分组相同的时间，从任意一个输入端口传送到给定的输出端口；③分组按FCFS方式，从一指定输入队列移动到其要求的输出队列中。只要其输出端口不同，就可以并行传送多个分组。然而，如果位于两

个输入队列前端的两个分组是发往同一输出队列的，则其中的一个分组将被阻塞，且必须在输入队列中等待，因为交换结构一次只能传送一个分组到某指定端口中。

图4-11显示了一个例子，其中输入队列前端的两个分组（带深色阴影）要发往同一个（右上角）输出端口中。假定交换结构决定传送左上角队列前端的分组。在这种情况下，左下角队列中的深色分组必须等待。但不仅该分组要等待，左下角队列中排在该分组后面的浅色分组也要等待，即使右中侧输出端口（浅色分组的目的地）中无竞争。这种现象叫做**输入排队交换机中的线路前部（Head-Of-the-Line, HOL）阻塞**，即在一个输入队列中排队的分组必须等待通过交换结构发送（即使输出端口是空闲的），因为它由位于线路前部的另一个分组阻塞。[Karol 1987] 指出，由于HOL阻塞，只要输入链路上的分组到达率达到其容量的58%，在某些假设前提下，输入队列长度就将无限制增大（非正式地讲，等同于将出现大量的分组丢失）。[McKeown 1997b] 中讨论了许多解决HOL阻塞的方法。

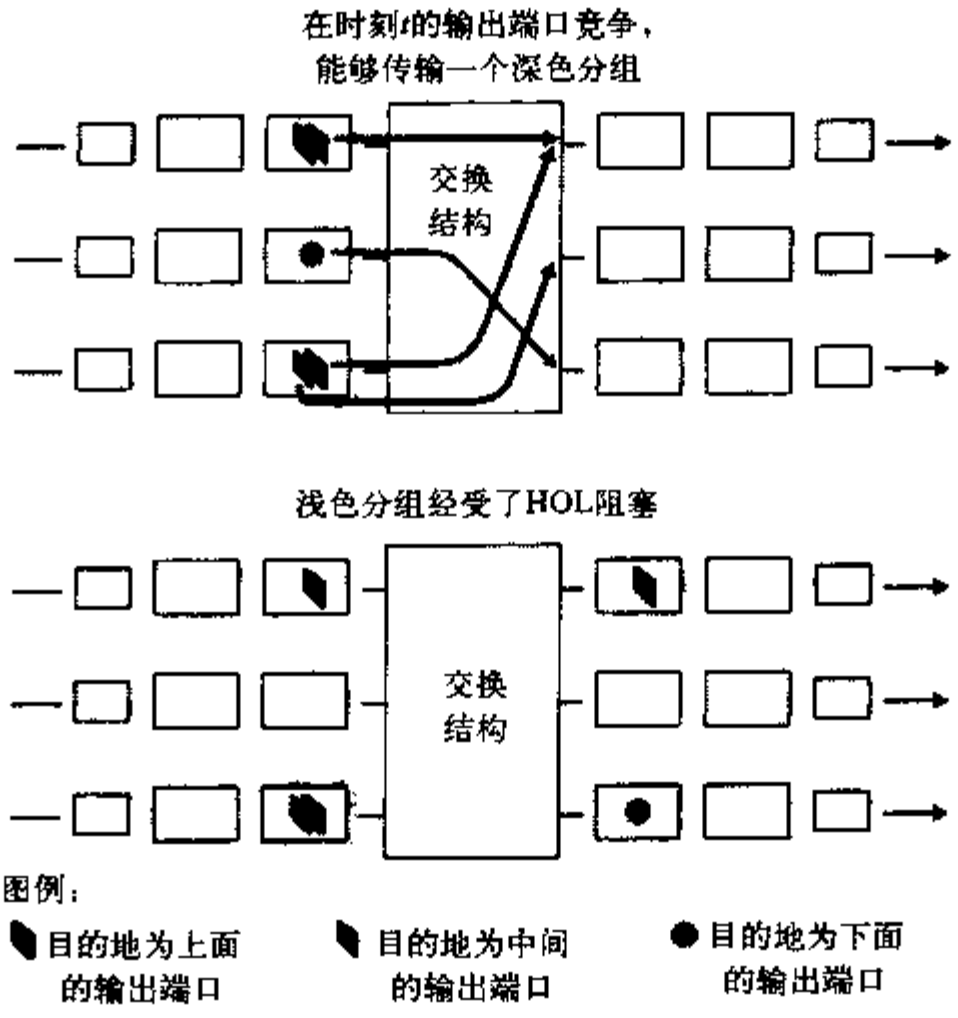


图4-11 在一个输入排队的交换机中的HOL阻塞

4.4 网际协议：因特网中的转发和编址

到目前为止，有关网络层编址和转发并未提及任何特定的计算机网络。在本节中，我们将注意力转向因特网中是如何完成编址和转发的。我们将看到因特网编址和转发是网际协议（IP）的重要组件。目前有两个版本的IP在使用。我们首先研究广泛设置的IP协议版本4，这通常被称为IPv4[RFC 791]。在本节结尾我们研究IP版本6[RFC 2460; RFC 3513]，它已经被提议替代IPv4。

在研究IP之前，我们先来考虑构成因特网的网络层的组件。如图4-12所示，因特网的网络层有三个主要的组件。第一个组件是IP协议，它是本节的主题。第二个主要组件是选路组件，它决定数据报从源到目的地所流经的路径。我们前面讲过选路协议计算在网络中用于转发分

组的转发表。我们将在4.6节中研究因特网的选路协议。网络层的最后一个组件是报告数据报中的差错和对某些网络层信息请求进行响应的设施。我们将在4.4.3节中学习因特网的网络层差错和信息报告协议，即互联网控制报文协议（ICMP）。

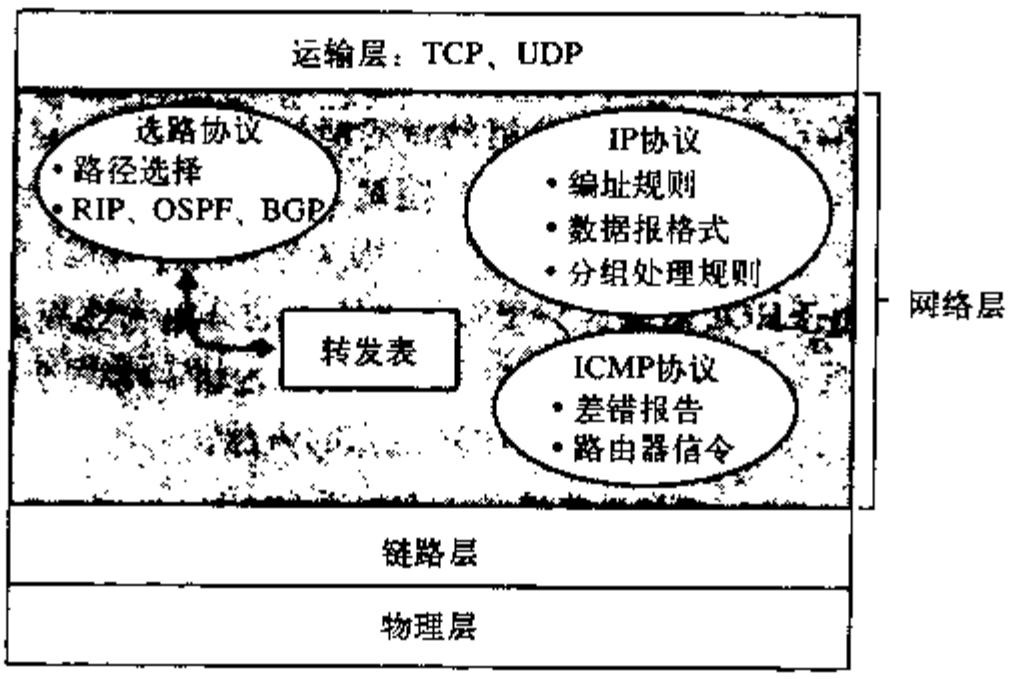


图4-12 因特网网络层的内部

4.4.1 数据报格式

前面讲过网络层分组被称为数据报。我们以概述IPv4数据报的语法和语义开始对IP的学习。你也许认为没有什么比一个分组的比特语法和语义更加枯燥乏味的了。但是，数据报在因特网中起着重要作用，每个网络专业学生和专业人员都需要面对它、学习它并掌握它。IPv4数据报格式如图4-13所示。IPv4数据报中的关键字段如下：

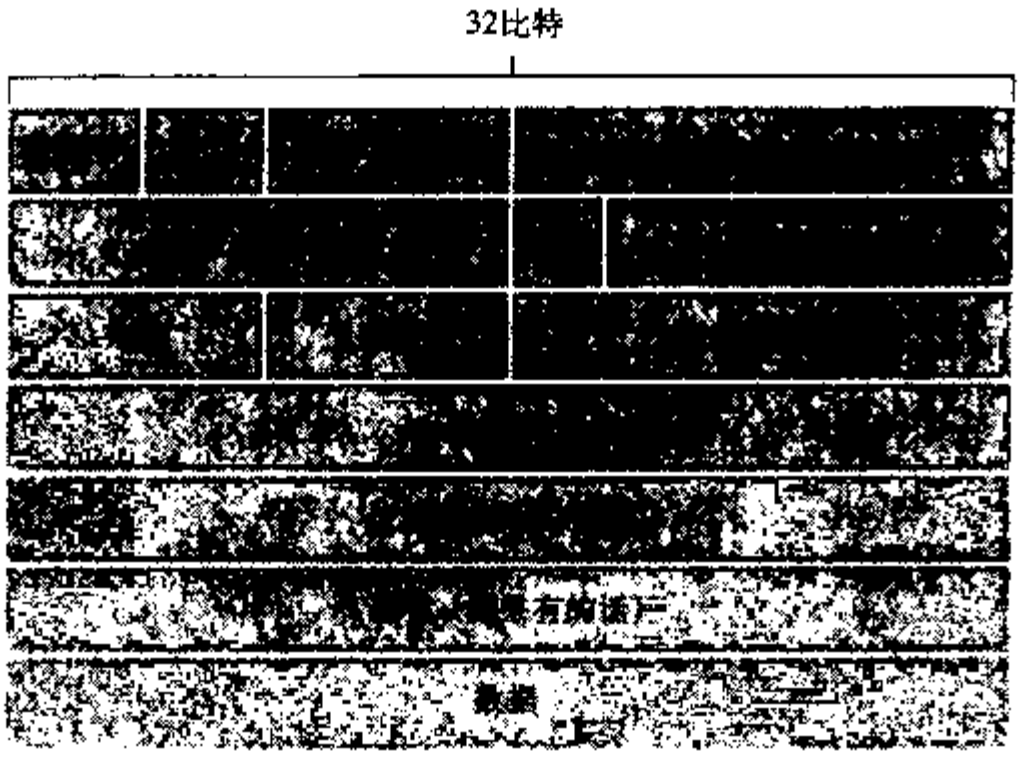


图4-13 IPv4数据报格式

- 版本号。这4比特规定了数据报的IP协议版本。通过查看版本号，路由器可确定如何解释IP数据报的剩余部分。不同的IP版本使用不同的数据报格式。目前使用的IP版本为IPv4，其格式如图4-13所示。新版本的IP（IPv6）数据报格式将在本节后面讨论。
- 首部长度。因为一个IPv4数据报可包含一些可选项（包含在IPv4数据报首部中），故需

要用这4比特来确定IP数据报中的数据部分实际从哪里开始。大多数IP数据报不包含可选项，所以一般的IP数据报都有20 字节的首部。

- 服务类型。服务类型（TOS）比特用来使不同类型的IP数据报（例如，一些特别要求低时延、高吞吐量或可靠性的数据报）能相互区别开来。例如，将实时数据报（如用于IP电话应用）与非实时流量（如FTP）区分开也许是有用的。提供具体的服务等级是一个由路由器管理员确定的策略问题。我们将在第7章详细讨论区分服务主题。
- 数据报长度。这是IP数据报的总长度（首部加上数据），以字节计。因为该字段长为16比特，所以IP数据报的理论最大长度为65 535字节。然而，数据报很少有超过1500字节的。
- 标识、标志、片偏移。该三个字段与所谓IP分片有关，稍后会详细讨论该主题。有趣的是，新版本的IP（即IPv6）不允许在路由器上分片。
- 寿命。寿命（Time-To-Live, TTL）字段用来确保数据报不会永远（如由于长时间的选路循环）在网络中循环。每当数据报经过一台路由器时，该字段的值减1。若TTL字段减为0，则该数据报必须丢弃。
- 协议。该字段仅在一个IP数据报到达其最终目的地时才会用到。该字段值指明了IP数据报的数据部分应交给哪个运输层协议。例如，值为6表明数据部分要交给TCP，而值为17表明数据要交给UDP。对于所有可能值的列表，参见［RFC 1700; RFC 3232］。注意，IP数据报中的协议号所起的作用类似于运输层报文段中端口号字段所起的作用。协议号是将网络层与运输层绑定到一起的黏合剂，而端口号是将运输层和应用层绑定到一起的黏合剂。我们将在第5章看到，链路层帧也有一个特殊字段用于将链路层与网络层绑定到一起。
- 首部检验和。首部检验和用于帮助路由器检测收到的IP数据报中的比特错误。首部检验和是这样计算的：将首部中的每两个字节当作一个数，用反码运算对这些数求和。如3.3节所述，该和的反码（又被称为互联网检验和）存放在检验和字段中。路由器要对每个收到的IP数据报计算其首部检验和，并根据数据报首部中携带的检验和与计算得到的检验和是否一致，来检查是否出错。路由器一般会丢弃检测出的错误数据报。注意，在每台路由器上都必须重新计算检查和并存放在原处，因为TTL字段以及选项字段可能会改变。关于计算互联网检验和的快速算法的有趣讨论参见［RFC 1071］。这里，一个经常问到的问题是，为什么TCP/IP在运输层与网络层都执行差错检测？这种重复有几个原因。首先，在IP层只对IP首部进行了检验，而TCP/UDP检验和是对整个TCP/UDP报文段进行的。其次，TCP/UDP与IP不一定属于同一个协议栈。原则上，TCP能运行在一个不同的协议（如ATM）上，而IP能够携带不一定要传递给TCP/UDP的数据。
- 源和目的IP地址。当源主机产生一个数据报时，它在源IP字段中插入它的IP地址，在目的IP地址字段中插入其最终目的地的地址。通常源主机经DNS查找决定目的地址，如第2章所述。我们将在4.4.2节中详细讨论IP编址。
- 选项。选项字段允许IP首部被扩展。首部选项意味着很少使用，因此在每个数据报首部不包括选项字段中的信息能够节约开销。然而，选项的存在的确是件复杂的事，因为数据报头长度可变，故不能预先确定数据字段从何处开始。而且，还由于有些数据报要求处理选项，而有些数据报则不要求，故导致一台路由器处理一个IP数据报所需的时间变化很大。这些考虑对于高性能路由器和主机上的IP处理来说特别重要。由于这样或那样的原因，IP选项已在IPv6报头中不再采用了，如4.4.4节将讨论的那样。

- 数据（有效载荷）。我们来看看最后的也是最重要的字段，也是数据报存在的首要理由！在大多数情况下，IP数据报中的数据字段含有要交付给目的地的运输层报文段（TCP或UDP）。然而，数据字段也可承载其他类型的数据，如ICMP报文段（将在4.4.3节中讨论）。

注意到一个IP数据报有20 字节首部（假定无选项）。若该数据报承载一个TCP报文段，则每个（无分片的）数据报共承载40字节首部（20字节IP首部加上20字节TCP首部）以及应用层报文。

IP数据报分片

我们将在第5章中看到，并不是所有的链路层协议都能承载相同长度的网络层分组。有的协议能承载大数据报，而有的协议只能承载小分组。例如，以太网帧可承载不超过1500字节的数据，而某些广域网链路的帧可承载不超过576字节的数据。一个链路层帧能承载的最大数据量叫做**最大传输单元**（Maximum Transmission Unit，MTU）。因为每个IP数据报封装在链路层帧中，再从一台路由器运输到下一台路由器，故链路层协议MTU严格地限制着IP数据报的长度。对IP数据报长度的严格限制并不是主要问题，真正的问题是在发送方与目的地路径上的每段链路可能使用不同的链路层协议，且每种协议可能具有不同的MTU。

为了更好地理解这一转发问题，想象你是一个互连几条链路的路由器，且每条链路运行具有不同MTU的链路层协议。假定你从某条链路收到一个IP数据报，通过检查转发表决定出链路，但该出链路的MTU比该IP数据报的长度小。此时，你会感到慌乱，如何将这个过大的IP数据报压缩成链路层帧的有效载荷字段呢？解决该问题的方法是将IP数据报中的数据分片成两个或更多个较小的数据报，用单独的链路层帧封装这些较小的IP数据报，然后向输出链路上发送这些帧。这些较小的数据报叫做片（fragment）。

片在其到达目的地运输层以前需要被重新组装。的确，TCP与UDP都希望从网络层收到完整的未分片的报文。IPv4的设计者认为在路由器中重新组装数据报会给协议带来相当大的复杂性并且影响路由器的性能。（如果你是一台路由器，你愿意将片重新组成报文作为你必须要做的各种工作的首要工作吗？）为坚持使网络内核保持简单的原则，IPv4的设计者决定将数据报的重新组装工作放到端系统中，而不是放到网络路由器中。

当一台目的主机从相同源收到一系列数据报时，它需要确定这些数据报中的某些是否是一些原来较大的数据报的片。如果它确定了某些数据报是片，则它必须进一步确定何时收到了最后一片，并且如何将这些片拼接到一起以形成初始数据报。为了让目的主机执行这些重新组装任务，IPv4的设计者将标识、标志和片偏移字段放在IP数据报中。当创建一个数据报时，发送主机在为该数据报设置源和目的地址的同时加上标识号。发送主机通常将它发送的每个数据报中的标识号加1。当一台路由器需要将一个数据报分片时，形成的每个数据报（即片）附加上初始数据报的源地址、目的地址与标识号。当目的地从同一发送主机收到一系列数据报时，它可以检查数据报的标识号以确定哪些数据报真正是同一较大数据报的片。由于IP是一个不可靠的服务，一个或多个片可能永远到达不了目的地，因此，为了让目的主机绝对地相信它已收到了初始数据报的最后一个片，最后一个片的标志比特被设为0，而所有其他片的标志比特被设为1。另外，为了让目的主机确定一个片是否丢失（且能按正确的顺序重新组装片），用偏移字段指定片应放在初始IP数据报的哪个位置。

图4-14给出了一个例子。一个4000字节的数据报（20字节IP首部加上3980字节IP有效载荷）到达一台路由器，且必须被转发到一条MTU为1500字节的链路上。这就意味着初始数据

报中的3980字节数据必须被分成3个独立的片（每个片也是一个IP数据报）。假定初始数据报附加上的标识号为777。三个片的特点如表4-2所示。表4-2中的值反映了除了最后一片外所有初始有效载荷数据的数量应当是8 字节的倍数，并且偏移值应当被规定以8 字节块为单位。

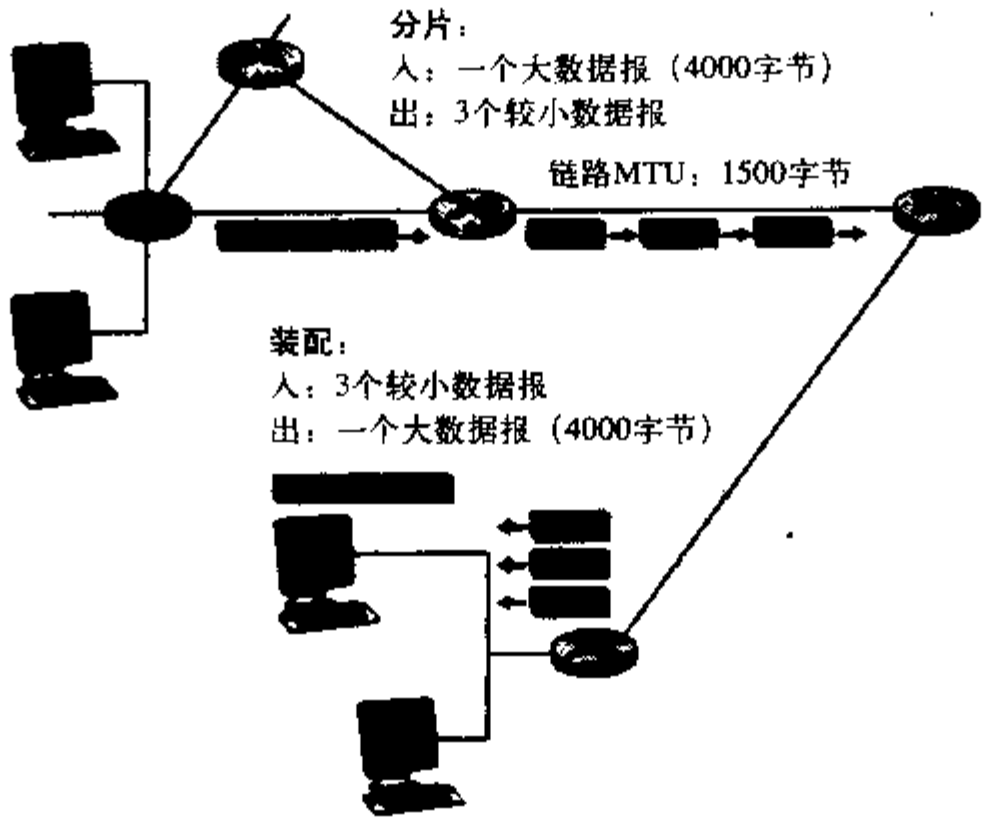


图4-14 IP分片与重新组装

表4-2 IP片

片	字 节 数	标 识	偏 移	标 志
第1片	IP数据报的数据字段中的1480字节	identification = 777	offset = 0（表示插入的数据开始于字节0）	flag = 1（表示后面还有）
第2片	1480字节数据	identification = 777	offset = 185（表示插入的数据开始于1480字节。注意185 · 8 = 1480）	flag = 1（表示后面还有）
第3片	1020字节（=3980 − 1480 − 1480）数据	identification = 777	offset = 370（表示插入的数据开始于2960字节。注意370 · 8 = 2960）	flag = 0（表示这是最后一片）

在目的地，数据报的有效载荷仅当在IP层已完全重构为初始IP数据报时，才被传递给目的地运输层。如果一个或多个片没有到达目的地，则该不完整的数据报被丢弃且不会交给运输层。但是，如前一章所述，若运输层正在使用TCP，则TCP将通过让源重传初始数据报中的数据而恢复丢失的片。

虽然IP分片在将许多不同的链路层技术结合起来的过程中起到了重要作用，但是分片也是有开销的。首先，它使路由器和端系统更为复杂，即必须将数据报分成适当大小的片并且需要重新组装。其次，分片可能引发致命的DoS攻击，即攻击者发送一系列古怪的片。Jolt2攻击是一个典型的例子，其中攻击者向目标主机发送了小片的流，这些片的偏移量都不是0。

当目标试图根据这些不良分组重建数据报时，可能会崩溃。另一类行为是发送交迭的IP分片，即这些片的偏移量值被设置得不能适当地排列起来。易受攻击的操作系统可能由于不知道如何处理交迭的片而崩溃[Skoudis 2006]。如我们将在本节最后看到的那样，IP协议的新版本（IPv6）从根本上废止了分片，从而简化了IP分组的处理，并使得IP不太容易受到攻击。

本书的配套网站上提供了一个Java小程序来产生片。你提供人数据报长度、MTU和人数据报标识，它就会自动为你产生片。

4.4.2 IPv4编址

我们现在将注意力转向IPv4编址。尽管你可能认为编址是相当直接的主题，但通过本章的学习我们希望你能认识到因特网编址不仅是一个丰富多彩的、微妙的和有趣的主题，而且也是一个对因特网极为重要的问题。[3Com Addressing 2007]和 [Stewart 1999] 的第1章都是介绍IPv4编址的优秀读物。

然而，在讨论IP编址之前，我们需要简述一下主机与路由器连入网络的方法。一台主机通常只有一条链路连接到网络；当主机中的IP想发送一个数据报时，它就在该链路上发送。主机与物理链路之间的边界叫做接口（interface）。现在考虑一台路由器及其接口。因为路由器的任务是从链路上接收数据报并将这些数据报从某些其他链路转发出去，所以路由器必须拥有两条或更多条链路与它连接。路由器与它的任意一条链路之间的边界也叫做接口。因此，一台路由器有多个接口，每个接口有一条链路。因为每台主机与路由器都能发送和接收IP数据报，所以IP要求每台主机和路由器接口都拥有自己的IP地址。因此，一个IP地址在技术上是与一个接口相关联的，而不是与包括该接口的主机或路由器相关联的。

每个IP地址长度为32比特（等于4字节），因此总共有2^{32}个可能的IP地址。由于2^{10}可近似地表示成10^3，故容易看出约有40亿个可能的IP地址。这些地址一般按所谓**点分十进制记法**（dotted-decimal notation）的方式书写，即地址中的每个字节用十进制形式书写，各字节间以句号（点）隔开。例如，考虑IP地址193.32.216.9，193是十进制数，等价于该地址中第一个8比特，32是十进制数，等价于该地址中第二个8比特，以此类推。因此，地址193.32.216.9的二进制记法是：

11000001 00100000 11011000 00001001

在全球因特网中，每台主机和路由器上的每个接口都必须有一个全球唯一的IP地址（在NAT后面的接口除外，本节结尾将讨论）。然而，这些地址不能以随意的方式自由选择。一个接口的IP地址的组成部分需要由其连接的子网来决定。

图4-15给出了一个IP编址与接口的例子。在该图中，一台路由器（有3个接口）用于互连7台主机。仔细观察分配给主机和路由器接口的IP地址，有几点需要注意。图4-15中左上侧部分的3台主机以及它们连接的路由器接口都有一个形如223.1.1.xxx的IP地址。这就是说，在它们的IP地址中，最左侧的24比特是相同的。这4个接口也通过一个不包含路由器的网络互连起来。（例如，该网络可以是一个以太LAN，在此情况下，这些接口将通过一台以太网集线器或一台以太网交换机互连，参见第5章。）用IP的术语来说，互连这3台主机的接口与路由器的一个接口的网络形成一个**子网**（subnet）[RFC 950]。（在因特网文献中，子网也称为IP网络或网络）。IP编址为这个子网分配一个地址：223.1.1.0/24，其中的/24记法有时称为**子网掩码**（subnet mask），它表明32比特中的最左侧24比特定义了子网地址。因此子网223.1.1.0/24是由3台主机接口（223.1.1.1、223.1.1.2和223.1.1.3）和1个路由器接口（223.1.1.4）组成的。任何

其他要连到223.1.1.0/24网络的主机都要求其地址形式为223.1.1.xxx。图4-15中显示了另外两个网络：223.1.2.0/24网络与223.1.3.0/24子网。图4-16说明了图4-15中的3个IP子网。

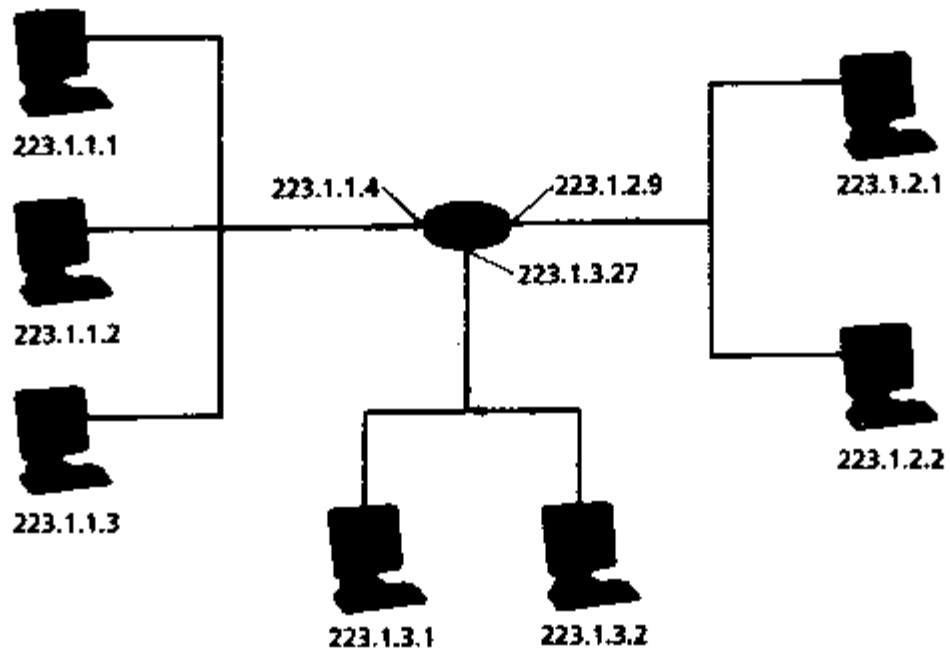


图4-15 接口地址和子网

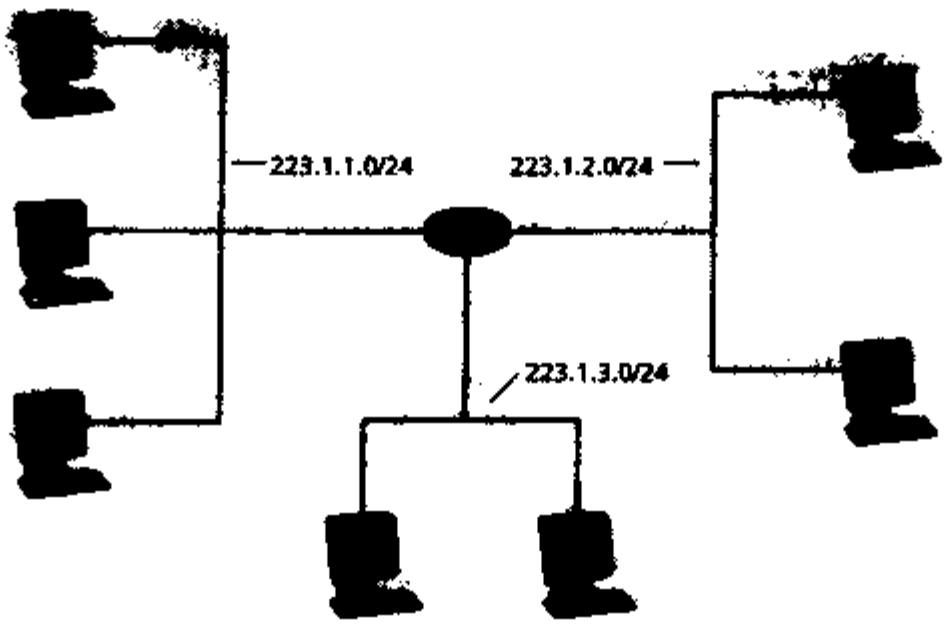


图4-16 子网地址

一个子网的IP定义并不局限于连接多台主机到一台路由器接口的以太网段。为了理解其中的含义，可考虑图4-17，图中显示了3台通过点对点链路互连在一起的路由器。每台路由器有3个接口，一个用于每条点对点链路，一个用于直接将路由器连接到一对主机的广播链路。这里有几个子网呢？有3个子网：223.1.1.0/24、223.1.2.0/24和223.1.3.0/24，它们类似于我们在图4-15中见过的子网。但注意到在该例中还有3个子网：第一个子网是223.1.9.0/24，用于连接路由器R1与R2的接口；第二个子网是223.1.8.0/24，用于连接路由器R2与R3的接口；第三个子网是223.1.7.0/24，用于连接路由器R3与R1的接口。对于一般由路由器和主机组成的互连系统，我们可以使用下列方法定义系统中的子网。

为了确定子网，分开主机和路由器的每个接口，从而产生了几个分离的网络岛，接口端接了这些独立的网络的端点。这些独立的网络中的每个都叫做一个**子网**(subnet)。

如果将该过程用于图4-17中互连的系统上，我们会得到6个岛或子网。

从上述讨论可见，一个具有多个以太网段和点对点链路的组织（如一个公司或学术机构）将具有多个子网，给定子网上的所有设备都具有相同的子网地址。原则上，不同的子网能够具有相当不同的子网地址。然而，在实际中，它们的子网地址经常有许多共同之处。为了理解其中的道理，我们来关注在全球因特网中是如何处理编址的。

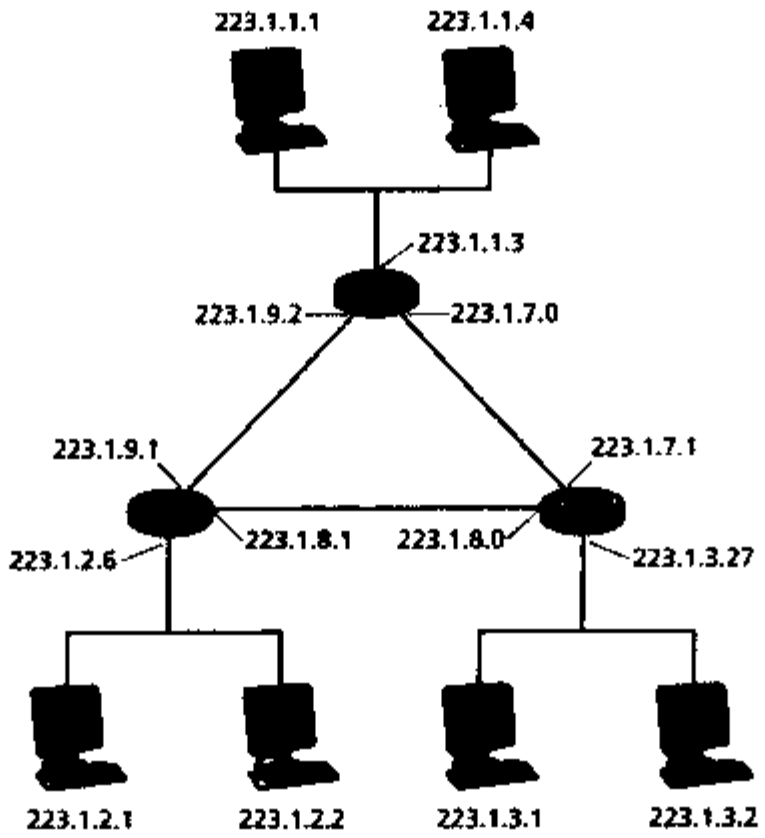


图4-17 3台路由器互连6个子网

因特网的地址分配策略被称为**无类别域间选路**（Classless Interdomain Routing, CIDR，读作cider）[RFC 4632]。CIDR将子网寻址的概念一般化了。因为对于子网寻址，32比特的IP地址被划分为两部分，并且也具有点分十进制形式a.b.c.d/x，其中x指示了在地址的第一部分中的比特数。

实践原则

这是一个ISP将8个组织连接到因特网的例子，它也较好地说明了仔细分配的CIDR化的地址有利于选路的道理。如图4-18所示，假设该ISP（我们称之为Fly-By-Night-ISP）向外界通告，应该发送所有地址的前20比特与200.23.16.0/20相符的数据报。外界其他部分不需要知道在地址块200.23.16.0/20内实际上还存在8个其他组织，每个组织有自己的子网。这种使用单个网络前缀通告多个网络的能力通常称为**地址聚合**（address aggregation），也称为**路由聚合**（route aggregation）或**路由摘要**（route summarization）。

当地址按块分给ISP，然后又由ISP分给客户组织时，地址聚合工作极为有效。但是当地址不是按这样的层次方式分配时，会出现什么情况呢？例如，如果Fly-By-Night-ISP获取了ISPs-R-Us，然后让组织1通过其辅助的ISPs-R-Us与因特网相连，将会发生什么情况呢？如图4-18所示，该辅助的ISPs-R-Us拥有地址块199.31.0.0/16，但很遗憾的是组织1的IP地址在该地址块之外。这里可以采取什么措施呢？组织1无疑可以将其所有的路由器和主机重新编号，使得地址在ISPs-R-Us的地址块内。但这是一种代价很高的方案，而且组织1将来也许还会从ISPs-R-Us更换到另一个ISP。通常采用的做法是，组织1保持其IP地址在200.23.18.0/23内。在这种情况下，如图4-19所示，Fly-By-Night-ISP继续通告地址

块200.23.16.0/20，并且ISPs-R-Us也继续通告地址块199.31.0.0/16。然而，ISPs-R-Us现在还要通告组织1的地址块200.23.18.0/23。当较大因特网上的其他路由器看见地址块200.23.16.0/20（来自Fly-By-Night-ISP）和200.23.18.0/23（来自ISPs-R-Us），并且想选路以到达地址块200.23.18.0/23内的一个地址时，它们将使用最长前缀匹配（参见4.2.2节），并朝着ISPs-R-Us选路，因为它通告了与目的地址相匹配的最长（最具体）的地址前缀。

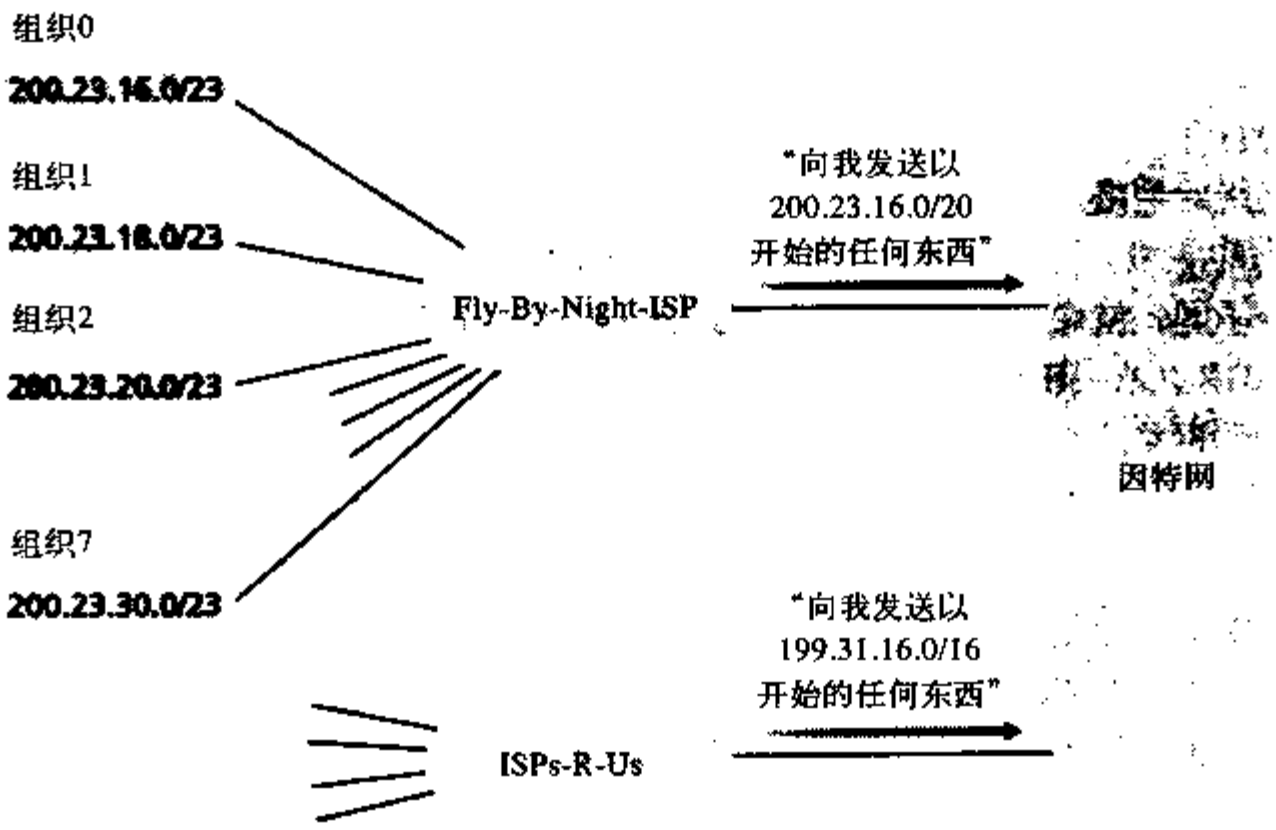


图4-18 层次编址与路由聚合

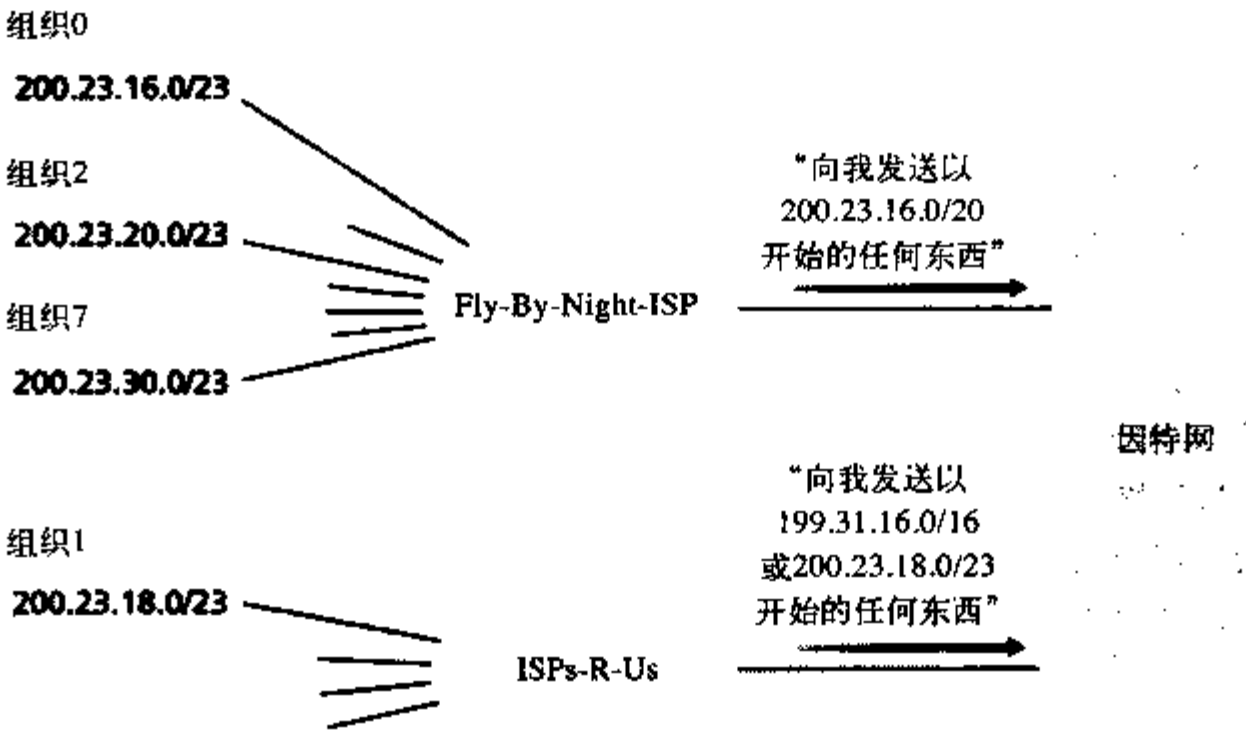


图4-19 ISPs-R-Us到组织1有一条更具体的路由

形式为a.b.c.d/x的地址的x 最高比特构成了IP地址的网络部分，并且经常被称为该地址的前缀（prefix）（或网络前缀）。一个组织通常被分配一块连续的地址，即具有相同前缀的一段

地址（参见实践原则）。在这种情况下，该组织内部的设备的IP地址将共享共同的前缀。当我们在4.6节中论及因特网的BGP选路协议时，将看到组织外部的路由器仅考虑x前面的前缀比特。这就是说，当组织外部的路由器转发一个数据报，且该数据报的目的地址在组织内部时，仅需要考虑该地址前面的x比特。这大大减少了这些路由器中的转发表的长度，因为形式为a.b.c.d/x单一项足以将数据报转发到组织内的任何目的地。

一个地址的剩余32−x比特可认为是用于区分组织内部设备的，其中所有设备具有相同的网络前缀。当组织内部的路由器转发分组时，才会考虑这些比特。如前所述，这些较低阶比特可能（或可能不）具有一个附加的子网结构。例如，假设某CIDR化的地址a.b.c.d/21的前21比特定义了组织的网络前缀，这对组织中的所有主机的IP地址来说是共同的，则其余的11比特标识了组织内的主机。该组织的内部结构可以采用这样的方式，使用这11个最右边的比特在该组织中划分子网，就像前面所讨论的那样。例如，a.b.c.d/24可能表示该组织内的特定子网。

在采用CIDR之前，IP地址的网络部分被限制长度为8、16或24比特，因为具有8、16或24比特子网地址的网络分别被称为A、B和C类网络，所以这种编址方案称为**分类编址**（classful addressing）。要求一个IP地址的网络部分正好为1、2或3字节，已经在支持迅速增加的具有小、中规模子网的组织数量方面出现了问题。一个C类（/24）子网仅能容纳多达$2^8 - 2 = 254$台主机（其中的两个地址预留用于特殊用途），这对于许多组织来说太小了。然而，一个B类（/16）子网可支持多达65 534台主机，又太大了。在分类编址方法下，比方说一个有2000台主机的组织通常被分配一个B类（/16）子网地址。这就导致了B类地址空间的迅速损耗以及所分配地址空间的低利用率。例如，为具有2000台主机的组织分配一个B类地址，就具有足以支持多达65 534个接口的地址空间，剩下的超过63 000个未用的地址却不能被其他组织使用。

但如果不提及另一种其他类型的IP地址，即IP广播地址255.255.255.255，那将是我们的疏漏。当一台主机发出一个目的地址为255.255.255.255的数据报时，该报文会被交付给同一个子网中的所有主机。路由器也会有选择地向邻近的子网转发该报文（虽然它们通常不这样做）。

现在详细学习完IP编址之后，我们需要知道主机或子网最初是如何得到其IP地址的。我们先看一个组织是如何为其设备得到一个地址块的，然后再看一个设备（如一台主机）是如何从本组织地址块中被分配到一个地址的。

1. 获取一块地址

为了获取一块IP地址用于一个组织的子网，网络管理员也许首先会与其ISP联系，ISP会从已分给它的更大地址块中提供一些地址。例如，某ISP也许已被分配了地址块200.23.16.0/20。该ISP可以依次将该地址块分成8个长度相等的较小地址块，为其支持的最多8个组织中的一个分配一小块，如下所示。（为了便于查看，我们已将这些地址的网络部分加了下划线。）

ISP的地址块	200.23.16.0/20	11001000 00010111 00010000 00000000
组织0	200.23.16.0/23	11001000 00010111 00010000 00000000
组织1	200.23.18.0/23	11001000 00010111 00010010 00000000
组织2	200.23.20.0/23	11001000 00010111 00010100 00000000
…	…	…
组织7	200.23.30.0/23	11001000 00010111 00011110 00000000

尽管从一个ISP获取一组地址是得到一块地址的一种方法，但这不是唯一的方法。显然，必须还有一种方法供ISP本身得到一块地址。是否有一个全球性的权威机构最终负责管理IP地址空间并向各ISP和其他组织分配地址块呢？的确有一个！ IP地址由因特网名字与号码分配机

构（Internet Corporation for Assigned Names and Numbers, ICANN）［ICANN 2007］管理，它基于RFC 2050中提出的规则。非营利的ICANN组织［NTIA 1998］的作用不仅是分配IP地址，还管理DNS根服务器。它还有一项有争议的工作，即分配域名与解决域名纷争。ICANN向地区性因特网注册机构（如ARIN、RIPE、APNIC和LACNIC）分配地址，这些机构一起形成了ICANN的地址支持组织[ASO-ICANN 2007]，处理本地域内的地址分配/管理。

2. 获取主机地址：动态主机配置协议

一个组织一旦获得了一块地址，它就可为该组织内的主机与路由器接口分配独立的IP地址。对于路由器接口地址，系统管理员手工配置路由器中的IP地址（在远程通常通过网络管理工具配置）。也可以手动配置主机，但是这项任务目前更常使用**动态主机配置协议**（Dynamic Host Configuration Protocol, DHCP）［RFC 2131］来完成。利用DHCP，主机可以自动获取IP地址。网络管理员可以配置DHCP，以便某给定主机每次与该网络连接时能得到一个相同的IP地址，或者被分配一个**临时的IP地址**（temporary IP address），主机每次与该网络连接时该地址都可能是不同的。除了为主机分配IP地址外，DHCP还允许一台主机获取其他信息，如它的子网掩码、它的第一跳路由器地址（常称为默认网关）与它的本地DNS服务器的地址。

由于DHCP具有能将主机连接进一个网络的自动化网络相关方面的能力，故它又常被称为**即插即用协议**（plug-and-play protocol）。这种能力对于网络管理员来说非常有吸引力，否则他将不得不用手工执行任务！DHCP还广泛地用于住宅区因特网接入网与无线局域网中，其中的主机频繁地加入和离开网络。例如，学生经常带着便携机从宿舍到图书馆再到教室。很有可能在每个位置，这个学生连接到一个新的子网，因此在每个位置都需要一个新的IP地址。在此情形下使用DHCP是最理想的，因为有许多用户来来往往，每个用户需要地址的时间很短。类似地，DHCP在住宅ISP接入网络中也是很有用的。举一个例子，一个住宅区ISP有2000个客户，但不会有超过400个客户同时在线。在这种情况下，动态分配地址的DHCP服务器不需要一个2048个地址的块，而仅需一个含512个地址的地址块（例如，像形式为a.b.c.d/23的块）。当主机加入或离开时，DHCP服务器要更新其可用IP地址表。每当一台主机加入时，DHCP服务器从其当前可用地址池中分配一个任意的地址给它；每当一台主机离开时，其地址便被收回池中。

DHCP是一个客户机/服务器协议。客户机通常是新到达的主机，它要获得网络配置信息，包括用于其自身的IP地址。在最简单的情形下，每个子网（在图4-17的编址意义下）拥有一台DHCP服务器。如果某子网没有DHCP服务器，则需要一个知道用于该网络的一台DHCP服务器地址的DHCP中继代理（通常是一台路由器）。图4-20显示了连接到子网223.1.2/24的一台DHCP服务器，以及一台提供中继代理服务的路由器，它为连接到子网223.1.1/24和223.1.3/24的到达客户机提供DHCP服务。在下面的讨论中，我们将假定DHCP服务器在该子网上是可供使用的。

对于一台新到达的主机而言，按照图4-20所示的网络设置，DHCP协议是一个4个步骤的过程，如图4-21所示。在该图中，`yiaddr`（“你的因特网地址”之意）表示分配给该新到达客户机的地址。

DHCP协议的4个步骤是：

- DHCP服务器发现。新到达的主机的首要任务是发现一个要与其交互的DHCP服务器。这可通过DHCP**发现报文**（DHCP discover message）来完成，客户机在UDP分组中向端

口67发送该发现报文。但是这个数据报应发给谁呢？该主机甚至不知道它所连接网络的IP地址，更不用说用于该网络的DHCP服务器地址了。在这种情况下，DHCP客户机生成包含DHCP发现报文的IP数据报，其中使用广播目的地址255.255.255.255并且使用“本主机”源地址0.0.0.0。DHCP客户机将该IP数据报传递给链路层，链路层然后将该帧广播到所有与该子网连接的子网（我们将在5.4节学习链路层广播的细节）。

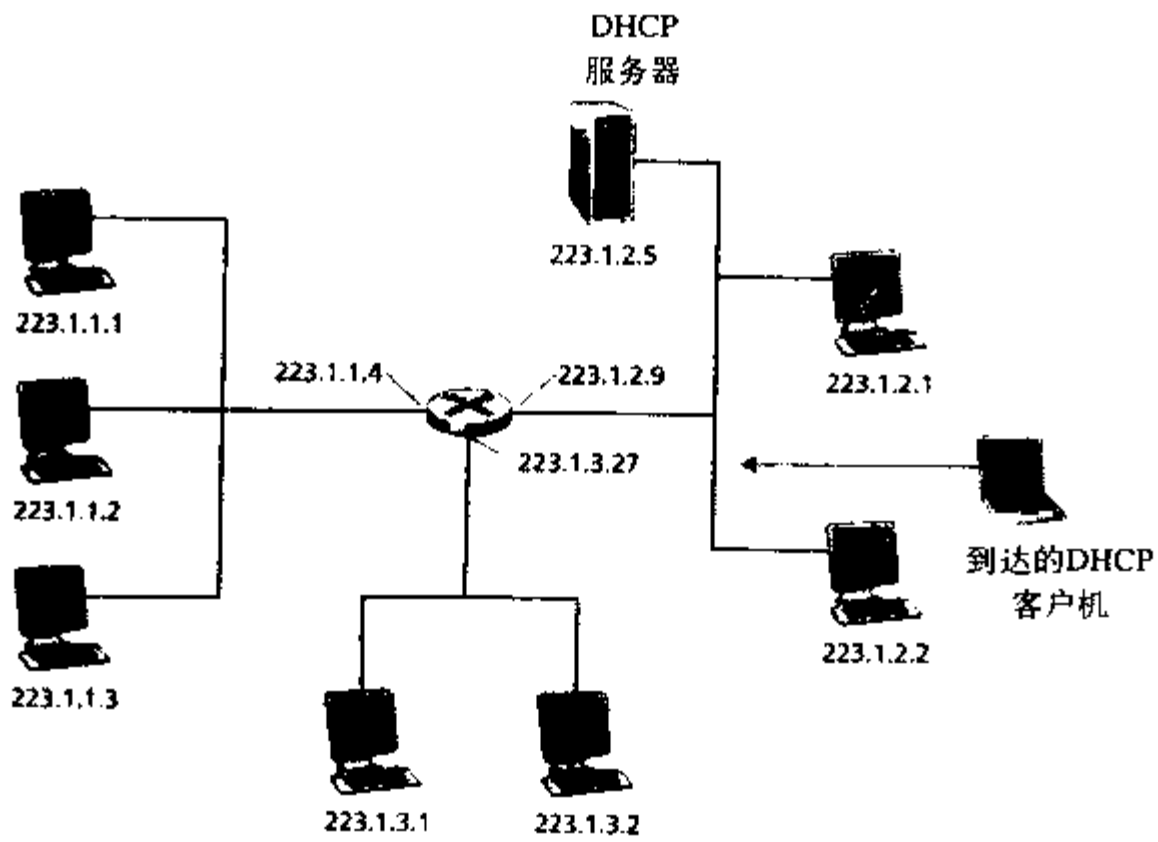


图4-20 DHCP客户机/服务器示例

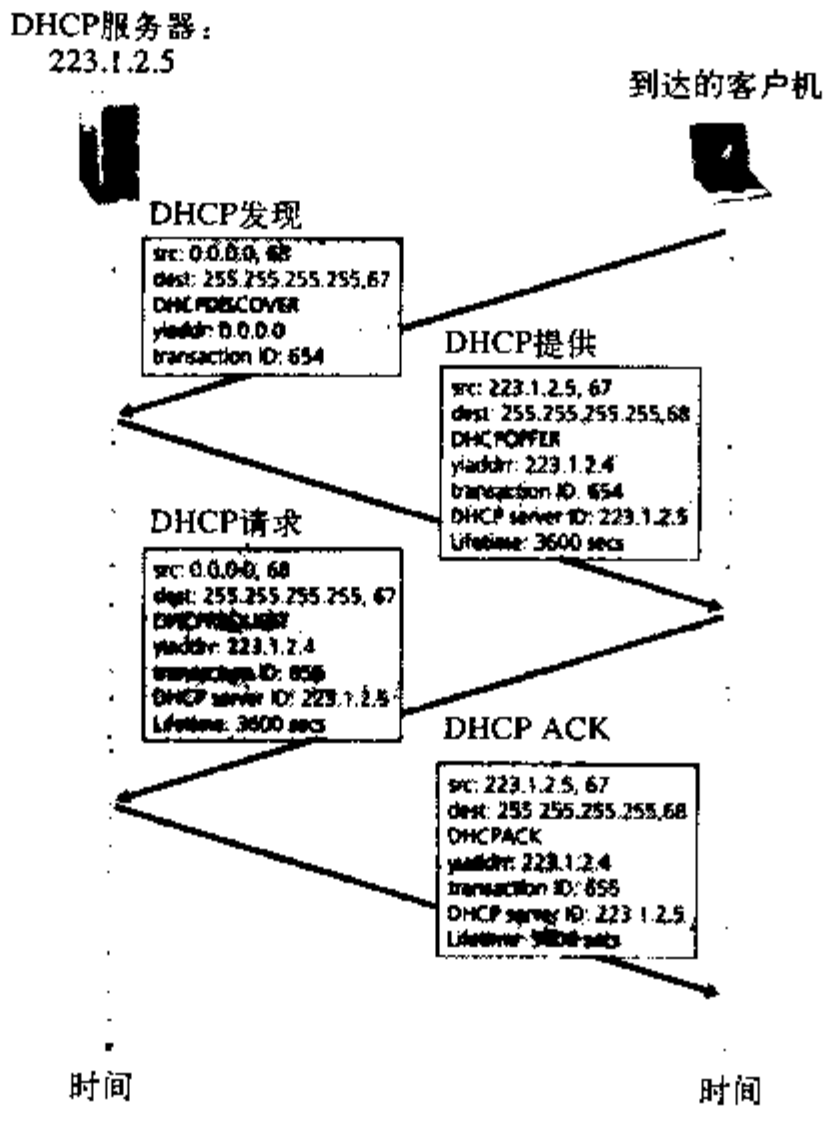


图4-21 DHCP客户机/服务器交互

- DHCP服务器提供。DHCP服务器收到一个DHCP发现报文时，用一个**DHCP提供报文**（DHCP offer message）对客户机做出响应，仍然使用IP广播地址255.255.255.255。（读者可以思考一下为何这个服务器也必须采用广播。）因为在子网中可能有几个DHCP服务器，所以该客户机可以选择其中的一个DHCP服务器。每个服务器提供报文中含有收到的发现报文的事务ID、向客户机推荐的IP地址、网络掩码以及**IP地址租用期**（IP address lease time，即IP地址有效的时间量）。服务器租用期通常设置为几小时或几天［Droms 1999］。
- DHCP请求。新到达的客户机从一个或多个服务器中选择一个，并用一个**DHCP请求报文**（DHCP request message）对选中的服务器进行响应，回显配置参数。
- DHCP ACK。服务器用**DHCP ACK报文**（DHCP ACK message）对DHCP请求报文进行响应，证实所要求的参数。

一旦客户机收到DHCP ACK后，交互便完成了，该客户机就能够在租用期内使用DHCP分配给它的IP地址。为了使客户机能在超出租用期后继续使用该地址，DHCP还提供了一种允许客户机更新它对一个IP地址的租用期的机制。

作为人工配置主机的IP地址的替代方法，DHCP的即插即用能力是非常有价值的。考虑这样的情况：一个学生带着便携机从教室到图书馆再到宿舍，每到一个新的位置就加入一个新的子网并获得一个新的IP地址。如果在每个位置都要系统管理员重新配置便携机（而且除了上过计算机网络课程的学生外，很少有学生有手工配置其计算机的经验），这简直是不可想象的。然而，从移动性角度看，DHCP确实有不足之处。这是因为，每当节点连到一个新子网时，都要从DHCP得到一个新的IP地址，这样当一个移动节点在子网之间移动时，就不能维持与远程应用之间的连接。在第6章中，我们将研究移动IP，它是一种对IP基础设施的扩展，允许移动节点在子网之间移动时能使用其单一永久的地址。其他有关DHCP的细节可参考［Droms 1999］与［dhc 2007］。还可从因特网系统协会［ISC 2007］处得到一个DHCP开放源码参考实现。

3. 网络地址转换（NAT）

进行了有关因特网地址和IPv4数据报格式讨论后，我们现在可清楚地认识到每个IP使能的设备都需要一个IP地址。随着所谓小型办公室、家庭办公室（Small Office, Home Office, SOHO）子网的大量出现，似乎意味着每当一个SOHO想安装一个LAN以互连多台机器时，都需要ISP分配一组地址来供该SOHO的所有机器使用。如果该网络变大了（例如，家里的孩子不仅有自己的计算机，还有手持PDA、IP使能电话及连网游戏机），则需要分配一块较大的地址。但如果ISP已经为SOHO网络的当前地址范围分配了一块连续地址，该怎么办呢？一般的家庭主人为了管理IP地址想（或应该）知道些什么？幸运的是，有一种应用越来越广泛的简单地址分配方法：**网络地址转换**（Network Address Translation，NAT）［RFC 2663; RFC 3022］。

图4-22显示了一台NAT使能路由器运作的情况。位于家中的NAT使能路由器有一个接口，该接口是图4-22中右侧所示家庭网络的一部分。家庭网络内的编址就像我们在上面看到的完全一样，其中的所有4个接口都具有相同的网络地址10.0.0/24。地址空间10.0.0.0/8是在RFC 1918中保留的3部分IP地址空间之一，这些地址用于专用网络或具有专用地址的**地域**（realm），如图4-22中的家庭网络。具有专用地址的地域是指其地址仅对该网络中的设备有意义的网络。为了理解其重要性，考虑有数以百万计家庭网络这样的事实，许多网络使用相同的地址空间10.0.0.0/24。一个给定家庭网络中的设备能够使用10.0.0.0/24编址彼此发送分组。然而，转发

到家庭网络之外进入更大的全球因特网的分组显然不能使用这些地址（要么作为源地址要么作为目的地址），因为有数以百万计的网络使用这块地址。这就是说，10.0.0.0/24地址仅在给定的网络中才有意义。但是如果专用地址仅在给定的网络中才有意义的话，当向因特网发送分组或从因特网接收分组时，如何处理地址？这里地址必须是唯一的。答案在于理解NAT。

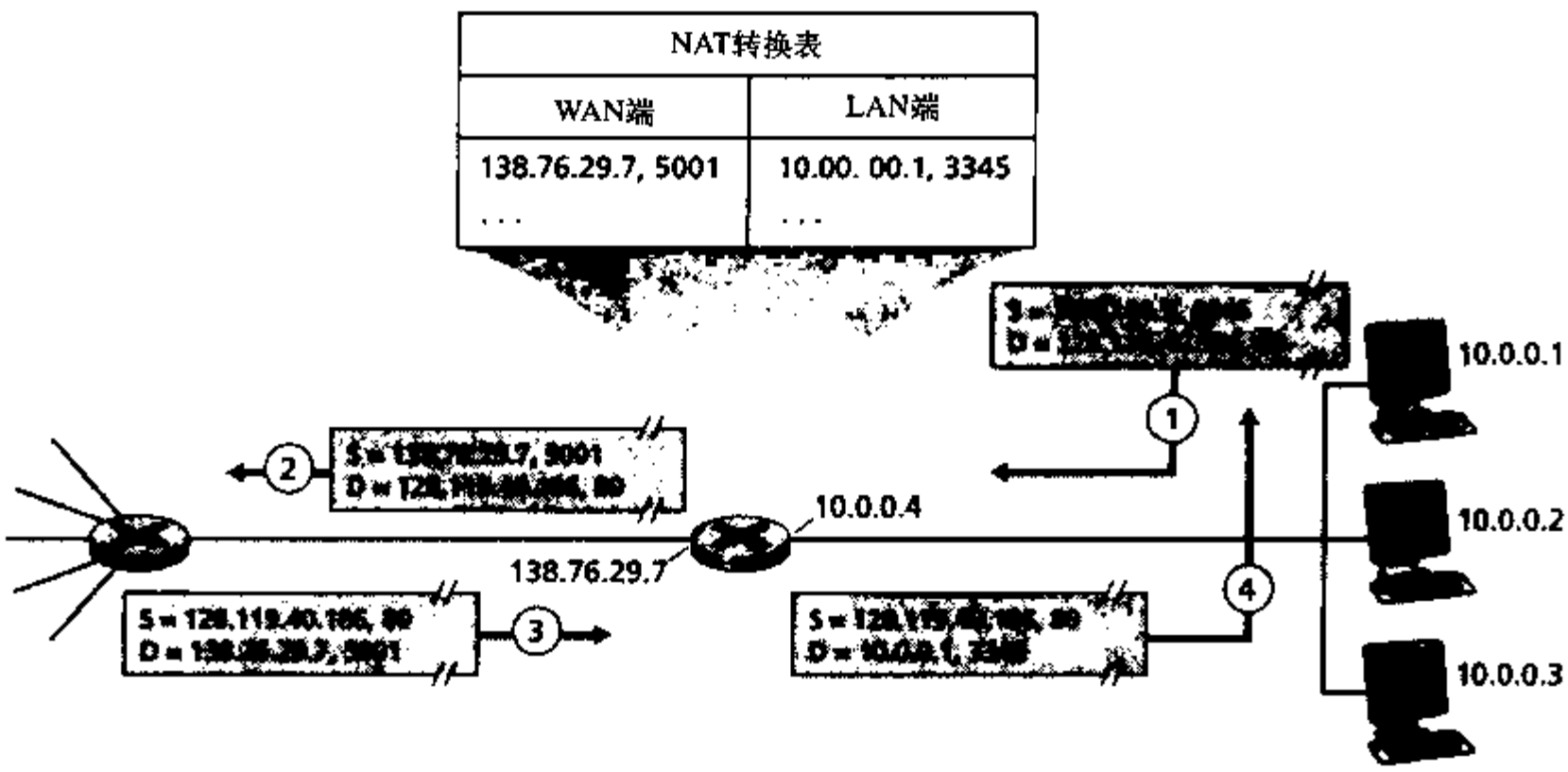


图4-22 网络地址转换

NAT使能路由器对于外界来说甚至不像一台路由器。相反，NAT路由器对外界的行为就如同一个具有单一IP地址的单一设备。在图4-22中，所有离开家庭路由器通向更大的因特网的报文都拥有同一个源IP地址138.76.29.7，且所有进入家庭的报文都拥有同一个目的IP地址138.76.29.7。从本质上讲，NAT使能路由器对外界隐藏了家庭网络的细节。（另外，你也许想知道家庭网络计算机是从哪里得到其地址，路由器又是从哪里得到其单个IP地址的。通常答案是相同的，即DHCP！路由器从ISP的DHCP服务器得到其地址，它再运行一个DHCP服务器，在NAT-DHCP路由器控制的家庭网络的地址空间中为计算机提供地址。）

如果来自广域网到达NAT路由器的所有数据报都有相同的目的IP地址（特别是对NAT路由器广域网一侧的接口而言），那么该路由器是怎样知道它应将一个给定数据报转发给哪个内部主机的呢？技巧就是使用NAT路由器上的一张NAT转换表（NAT translation table），并且在表项中包含了端口号和IP地址。

考虑图4-22中的例子。假设一个用户坐在家庭网络主机10.0.0.1旁，请求IP地址为128.119.40.186的Web服务器（端口80）上的一个Web页面。主机10.0.0.1为其指派了（任意）源端口号3345并将该数据报发送到LAN中。NAT路由器收到该数据报，为该数据报生成一个新的源端口号5001，将源IP地址改为其广域网一侧接口的IP地址138.76.29.7，且将源端口3345更换为新端口号5001。当生成一个新的源端口号时，NAT路由器可选择任意一个当前未在NAT转换表中使用的源端口号。（注意，端口号字段为16比特长，NAT协议可支持60 000多个并行使用路由器广域网一侧IP地址的连接！）路由器中的NAT也在NAT转换表中增加一项。Web服务器并不知道刚到达的包含HTTP请求的数据报已被NAT路由器进行了改装，它会发回一个响应数据报，其目的地址是NAT路由器的IP地址，目的端口是5001。当该数据报到达NAT路由器

时，路由器使用目的IP地址与目的端口号从NAT转换表中检索出家庭网络浏览器使用的正确IP地址（10.0.0.1）和目的端口号（3345）。于是，路由器改写该数据报的目的IP地址与目的端口号，并向家庭网络转发该数据报。

NAT在近几年已得到了广泛的应用。但是，我们应当提及的是，许多IETF团体中的纯化论者（极力主张严格规范的人）大声疾呼反对NAT。第一，他们认为端口号是用于编址进程的方法，而并不是用于编址主机的。（这种违规用法对于在家庭网络中运行的服务器来说确实会引起问题，因为我们已在第2章看到，服务器进程在周知端口号处等待入请求。）第二，他们认为路由器通常应当处理最多达第三层的分组。第三，他们认为NAT协议违反了所谓端到端原则（end-to-end argument），即主机间应相互直接对话，节点不应介入修改IP地址与端口号。第四，他们认为我们应使用IPv6（参见4.4.4节）来解决IP地址短缺问题，而不是不计后果地用一种诸如NAT之类的权宜之计来修补存在的问题。但不管喜欢与否，NAT已成为因特网的一个重要组件。

NAT的另一个重要问题是它妨碍P2P应用程序，包括P2P文件共享应用和P2P的IP之上话音应用。第2章讲过，在一个P2P应用程序中，任何参与对等方A应当能够向任何其他参与向等方B发起一个TCP连接。该问题的实质在于，如果对等方B在一个NAT后面，它就不能充当服务器并接收TCP连接。如我们将在课后习题中所见，如果对等方A不在一个NAT的后面，则该NAT问题就能够解决。具体方法是，对等方A首先通过一个中间对等方C与对等方B联系，其中C不位于NAT之后并与B已经创建了一个进行中的TCP连接。对等方A则能够经对等方C请求对等方B，发起直接返回对等方A的一个TCP连接。一旦对等方A和B之间创建了一个直接的P2P TCP连接，这两个对等方就能够交换报文或文件。这种雇佣关系称为**连接反转**（connection reversal），实际上被许多P2P应用程序用于NAT**穿越**（NAT traversal）。如果对等方A和对等方B都在它们自己的NAT后面，则问题有些棘手，但是可以使用应用程序来中继处理，正如第2章所述的Skype中继那样。

4. UPnP

NAT穿越正越来越多地由通用即插即用（UPnP）提供，UPnP是一种允许主机发现并配置邻近NAT的协议[UPnP Forum 2007]。UPnP要求主机和NAT是UPnP兼容的。使用UPnP，在一台主机上运行的应用程序能够为某些请求的公共端口号请求一个NAT映射，该映射位于（专用IP地址，专用端口号）和（公共IP地址，公共端口号）之间。如果NAT接受该请求并生成该映射，则来自外部的节点能够发起到（公共IP地址，公共端口号）的TCP连接。此外，UPnP让该应用程序知道（公共IP地址，公共端口号），因此该应用程序能够向外部世界通告它。

举一个例子，假定你的主机位于一个UPnP使能的NAT后面，具有专用地址10.0.0.1并且在端口3345上运行BitTorrent。另外，假定该NAT的公共IP地址是138.76.29.7。你的BitTorrent应用程序当然要能够接受来自其他主机的连接，以便它能够同其他主机对换块。此后，你主机上的该BitTorrent应用程序请求NAT产生一个“洞”，将（10.0.1,3345）映射到（138.76.29.7，5001）。（该应用程序选择了公共端口号5501。）你主机中的BitTorrent应用程序也能够向其追踪器通告它在（138.76.29.7，5001）可供使用。以这种方式，运行BitTorrent的外部主机能够联系该追踪器，并知道你的BitTorrent应用程序正运行在（138.76.29.7，5001）。该外部主机能够向（138.76.29.7，5001）发送TCP SYN分组。当NAT接收到该SYN分组时，它将分组中的目的IP地址和端口号改成（10.0.0.1，3345），并通过NAT转发分组。

总之，UPnP允许外部主机使用TCP或UDP向NAT的主机发起通信会话。长期以来NAT一

直不适合P2P应用程序，而UPnP由于提供了有效的、健壮的NAT穿越解决方案，可能成为P2P应用程序的“救世主”。这里我们只是简要地讨论了NAT和UPnP，关于NAT的详细讨论参见[Huston and UPnP 2004, Cisco NAT 2007]。

4.4.3 ICMP：互联网控制报文协议

前面讲过因特网的网络层有3个主要组件：在前面一节中讨论的IP协议，在4.6节中将讨论的因特网选路协议（包括RIP、OSPF和BGP），以及本节的主题——ICMP。

ICMP由RFC 792定义，它用于主机和路由器彼此交互网络层信息。ICMP最典型的用途是差错报告。例如，当运行一次Telnet、FTP或HTTP会话时，你也许会遇到一些诸如“目的网络不可达”之类的错误报文。这种报文就是在ICMP中产生的。在某个位置， IP路由器不能找到一条路径，以通往Telnet、FTP或HTTP应用所指定的主机。该路由器就会创建和发出一个类型3的ICMP报文到你的主机来指示该错误。

ICMP通常被认为是IP的一部分，但从体系结构上讲它是位于IP之上，因为ICMP报文是承载在IP分组中的。这就是说，ICMP报文是作为IP有效载荷承载的，就像TCP与UDP报文段作为IP有效载荷被承载那样。类似地，当一台主机收到一个指明上层协议为ICMP的IP数据报时，它分解该数据报的内容给ICMP，就像分解一个数据报的内容给TCP或UDP一样。

ICMP报文有一个类型字段和一个编码字段，并且包含引起该ICMP报文首次生成的IP数据报（以便发送方能确定引发该差错的数据报）的首部和前8字节内容。在图4-23中显示了部分ICMP报文。注意，ICMP报文并不是仅用于通知差错情况。

众所周知的ping程序发送一个ICMP类型8编码0的报文到指定主机。看到该回显（echo）请求的目的主机发回一个类型0编码0的ICMP回显回答。多数TCP/IP实现直接在操作系统中支持ping服务器；即该服务器不是一个进程。[Stevens 1990]的第11章提供了有关ping客户机程序的源码。注意到客户机程序需要能够指示操作系统产生一个类型8编码0的ICMP报文。

另一个有趣的ICMP报文是源抑制报文。这种报文在实际中很少使用。其最初目的是执行拥塞控制，即允许拥塞的路由器向一台主机发送一个ICMP源抑制报文，以强制该主机减小其传输速率。我们在第3章已看到，TCP有自己在运输层操作的拥塞控制机制，不需要利用网络层中的反馈信息（如ICMP源抑制报文）。

在第1章中我们介绍了Traceroute程序，该程序允许用户跟踪从一台主机到世界上任意一台其他主机之间的路由。有趣的是，Traceroute是用ICMP报文来实现的。为了判断源和目的之间所有路由器的名字和地址，源主机中的Traceroute向目的主机发送一系列普通的IP数据报。这些数据报中的每个都携带了一个具有不可达UDP端口号的UDP报文段。第一个数据报的TTL为1，第二个数据报的TTL为2，第三个数据报的TTL为3，以此类推。该源主机也为每个数据报启动定时器。当第n个数据报到达第n台路由器时，第n台路由器观察到这个数据报的TTL正好终止。根据IP协议规则，路由器将丢弃该数据报并发送一个ICMP告警报文给源主机（类型11编码0）。该告警报文包含有路由器的名字与IP地址。当该ICMP报文到达源主机时，源主机从定时器得到往返时延，从ICMP报文中得到第n台路由器的名字与IP地址。

Traceroute源主机是怎样知道何时停止发送UDP报文段的呢？前面讲过，源主机为它发送的每个报文段的TTL字段加1。因此，这些数据报之一将最终沿着这条路到达目的主机。因为该数据报包含了一个具有不可达端口号的UDP报文段，所以该目的主机将向源主机发送一个端口不可达的ICMP报文。当源主机收到这个特别的ICMP报文时，它便知道了它不需要再发

送另外的探测分组。(标准的Traceroute程序实际上用相同的TTL发送3个一组的分组，因此Traceroute输出对每个TTL提供了3个结果。)

ICMP类型	编 码	描 述
0	0	回显回答（对ping的回答）
3	0	目的网络不可达
3	1	目的主机不可达
3	2	目的协议不可达
3	3	目的端口不可达
3	6	目的网络未知
3	7	目的主机未知
4	0	源抑制（source quench，拥塞控制）
8	0	回显请求
9	0	路由器通告
10	0	路由器发现
11	0	TTL过期
12	0	IP首部错误

图4-23 ICMP报文类型

通过这种方式，源主机知道了位于它与目的主机之间的路由器数量和标识，以及两台主机之间的往返时延。注意，Traceroute客户机程序必须能够指示操作系统产生具有特定TTL值的UDP数据报，也必须能够由它的操作系统通知ICMP报文到达的时间。既然你已明白了Traceroute工作原理，你也许想回去更多地使用它。

关注安全

检查数据报：防火墙和入侵检测系统

假定你负责管理家庭网络、部门网络、大学网络或公司网络。知道你管理的网络IP地址范围的攻击者可以方便地此范围中进行寻址。然后，能够做各种不正当的事情，包括用Ping搜索和端口扫描映射你的网络，用恶意分组损害易受攻击的主机，用纷至沓来的ICMP分组对服务器进行洪泛攻击，通过在分组中带有恶意软件来感染主机。作为网络管理员，应做些什么来将向你的网络发送恶意分组的坏家伙拒之门外呢？针对恶意分组攻击的两种流行防御措施是防火墙和入侵检测系统（IDS）。

作为网络管理员，你可能首先尝试在你的网络和因特网之间安装一台防火墙。(现在，大多数接入路由器具有防火墙能力。）防火墙检查数据报和报文段首部字段，拒绝可疑的数据报进入内部网络。例如，可以将防火墙配置为阻挡所有的ICMP回显请求分组，从而防止攻击者对你的IP地址范围进行传统的ping搜索。防火墙也能基于源IP地址、目的IP地址和端口号阻挡分组。此外，防火墙可以配置为跟踪TCP连接，仅允许属于批准连接的数据报进入。

IDS能够提供另一种保护措施。IDS通常位于网络的边界，执行“纵深分组检查”，不仅检查数据报（包括应用层数据）中的首部字段而且检查其有效载荷。IDS有一个分组特征（特征为攻击的一部分）的数据库。随着新攻击被发现，该数据库自动更新特征。当分组通过IDS时，IDS试图将分组的首部字段和有效载荷与其特征数据库中的特征相匹配。如果发现了这样的一种匹配，就产生一个告警。入侵防护系统（IPS）与IDS类似，只是它除了产生告警外还能实际阻挡分组。在第8章中，我们将更详细地研究防火墙和IDS。

防火墙和IDS能够全面保护你的网络免受所有攻击吗？答案显然是否定的，因为攻击者不停地寻找特征还不能匹配的新攻击方法。但是，防火墙和传统的基于特征的IDS在保护你的网络免受已知攻击方面还是很有用的。

4.4.4 IPv6

20世纪90年代早期，因特网工程任务组就开始致力于开发一种替代IPv4的协议。该努力的最初动机是基于这样的现实：由于新的子网和IP节点以惊人的增长率连到因特网上（并被分配唯一的IP地址），32比特的IP地址空间即将用尽。为了满足这种对大IP地址空间的需求，一种新的IP协议IPv6便被开发出来。IPv6的设计者们还利用这次机会，在IPv4积累的运行经验基础上加入和增强了IPv4的其他方面。

IPv4地址何时会被完全分配完（因此没有新的网络能与因特网相连）是一个存在相当多争论的问题。IETF的地址寿命期望工作组的两位负责人分别估计地址将于2008年和2018年用完［Solensky 1996］。1996年，美国因特网号码注册机构（ARIN）发布报告说，所有的IPv4 A类地址已分配完，62%的B类地址已分配，37%的C类地址已分配［ARIN 1996］。有关IPv4地址空间分配的最新报告，参见[Hain 2005]。虽然这些估计和数字表明离IPv4地址空间用尽的期限还有不少时间，但人们认识到如此大规模地推广应用一项新技术需要相当长的时间，因此人们开始致力于研发下一代IP（Next Generation IP, IPng）［Bradner 1996; RFC 1752］。这种努力的结果就是IP版本6规约（IPv6）［RFC 2460］。（一个经常问到的问题是，IPv5情况怎么样。人们最初预想ST-2协议会成为IPv5，但支持RSVP协议的ST-2后来被舍弃了，我们将在第7章中讨论RSVP协议。）

有关IPv6的优秀信息源是IP下一代主页[Hinden 2007]以及Huitema撰写的有关该主题的一本书【Huitema 1998】。

1. IPv6数据报格式

IPv6数据报的格式如图4-24所示。

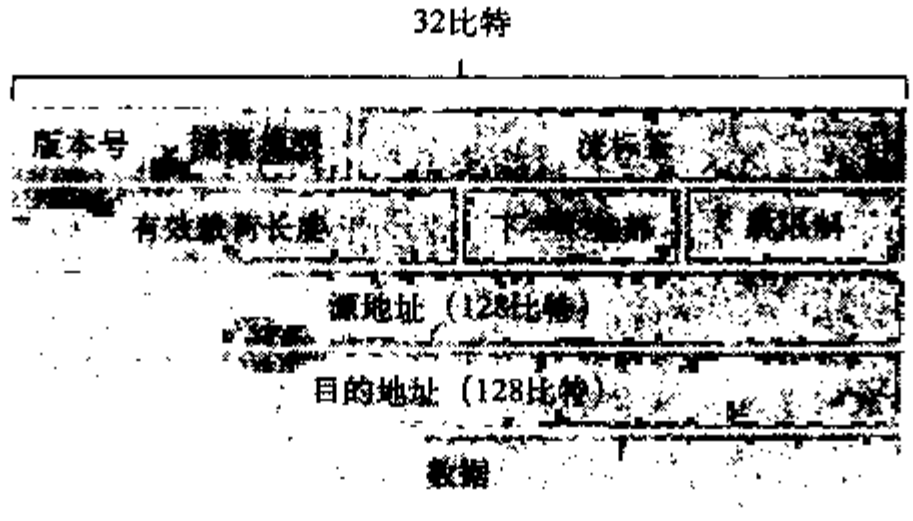


图4-24 IPv6数据报格式

IPv6中引入的最重要的变化表现在其数据报格式中：

- 扩大的地址容量。IPv6将IP地址长度从32比特增加到128比特，这就确保全世界不会用尽IP地址。现在，地球上的每个沙砾都可以用IP地址寻址了。除了单播与多播地址以外，IPv6还引入了一种称为任播地址（anycast address）的新型地址，这种地址可以使一个数据报能交付给一组主机中的任意一个。（例如，这种特性可用于向一组包含给定文档的镜像站点中最近的一个发送一个HTTP GET报文。）
- 简单高效的40字节首部。如下面讨论的那样，许多IPv4字段已被舍弃或作为选项。因而形成的40字节定长首部允许更快地处理IP数据报。一种新的选项编码允许进行更灵活的选项处理。
- 流标签与优先级。IPv6有一个难以捉摸的流（flow）定义。RFC 1752与RFC 2460中描述道，该字段可用于“给属于特殊流的分组加上标签，这些特殊流是发送方要求进行特殊处理的流，如一种非默认服务质量或需要实时服务的流”。例如，音频与视频传输就可能被当作一个流。另一方面，越是传统的应用（如文件传输和电子邮件）越可能不被当作流。为高优先级用户承载的流量（如某些用户为他们的流量得到更好服务而付费）也有可能被当作一个流。然而，IPv6的设计者们显然已预见到最终需要能够区分这些流，即使流的确切含义还未完全确定。IPv6首部中还有一个8比特的流量类型字段，该字段就像IPv4中的TOS字段，可用于给出一个流中某些数据报的优先级，以便指明某些应用的数据报（如ICMP分组）比其他应用的数据报（如网络新闻）有更高的优先权。

如上所述，比较图4-24与图4-13就可看出IPv6数据报的结构更简单、更高效。以下是IPv6中定义的字段：

- 版本号。这4比特字段用于标识IP版本号。毫不奇怪，IPv6将该字段值设为6。注意，将该字段值置为4并不能创建一个合法的IPv4数据报。（如果这样的话，事情就简单多了，参见下面有关“从IPv4向IPv6迁移”的讨论。）
- 流量类型。这8比特字段与IPv4中TOS字段的含义相似。
- 流标签。如上面讨论的那样，该20比特字段用于标识一个数据报的流。
- 有效载荷长度。该16比特值作为一个无符号整数，给出了IPv6数据报中跟在定长的40字节数据报首部后面的字节数量。
- 下一个首部。该字段标识该数据报中的内容（数据字段）需要交付给哪个协议（如TCP或UDP）。该字段使用与IPv4首部中协议字段相同的值。
- 跳限制。转发数据报的每台路由器将对该字段内容减1。如果跳限制计数到达0，则该数据报将被丢弃。
- 源和目的地址。IPv6 128比特地址的各样格式在RFC 4291中进行了描述。
- 数据。这是IPv6数据报的有效载荷部分。当数据报到达目的地时，该有效载荷就从IP数据报中移出，并被交给下一个首部字段中指定的协议处理。

以上说明了IPv6数据报中包括的各字段的用途。若将图4-24中的IPv6数据报格式与图4-13中的IPv4数据报格式进行比较，则会注意到IPv4数据报中出现的几个字段在IPv6数据报中已不复存在：

- 分片/重新组装。IPv6不允许在中间路由器上进行分片与重新组装。这种操作只能在源与目的地上执行。如果一台路由器收到的IPv6数据报因太大而不能转发到出链路上，则该路由器只需丢掉该数据报，并向发送方发回一个“分组太大”的ICMP差错报文即可

（见下文）。于是发送方使用较小长度的IP数据报重发数据。分片与重新组装是一个耗时的操作，将该功能从路由器中删除并放到端系统中，大大加快了网络中的IP转发速度。

- 首部检验和。因为因特网诸层中的运输层（如TCP与UDP）和数据链路层（如以太网）协议执行了检验操作，所以IP设计者可能觉得在网络层中具有该项功能实属多余，可以去除。再次强调的是，快速处理IP分组是关注的重点。在4.4.1节中讨论IPv4时讲过，由于IPv4首部中包含有一个TTL字段（类似于IPv6中的跳限制字段），所以在每台路由器上都需要重新计算IPv4首部检验和。就像分片与重新组装一样，在IPv4中这也是一项耗时的操作。
- 选项。选项字段不再是标准IP首部的一部分了，但它并没有消失，而是可能出现在IPv6首部中由下一个首部指出的位置上。也就是说，就像TCP或UDP协议首部可在IP分组中由下一个首部字段指出一样，选项字段也可在下一个首部字段中指出。删除选项字段导致了一个定长的40 字节的IP首部。

我们在4.4.3节讲过，IP节点使用ICMP协议来报告差错情况，并向端系统提供有限的信息（如对一个ping报文的回显回答）。RFC 4443中定义了一种IPv6使用的新版ICMP。除了能识别现存的ICMP类型和编码定义外，由于IPv6新增功能的需要，ICMPv6还增加了新的类型和编码，其中包括“分组太大”类型与“未识别的IPv6选项”错误编码。另外，ICMPv6还包含了我们将在4.7节中学习的互联网组管理协议（IGMP）。IGMP用于管理主机加入和离开多播组，它在IPv4中是一个与ICMP分开的独立协议。

2. 从IPv4向IPv6迁移

既然我们已了解了IPv6的技术细节，那么我们考虑一个非常现实的问题：基于IPv4的公共因特网如何迁移到IPv6呢？问题就是虽然能处理IPv6的系统可做成向后兼容的，即能发送、选路和接收IPv4数据报，但已设置的IPv4使能的系统不能处理IPv6数据报。要解决这个问题，有几种方法可供选择。

一种方法就是宣布一个标志日，即指定某个日期和时间，届时因特网的所有机器都关机并从IPv4升级到IPv6。上次重大的技术迁移（为得到可靠的运输服务，从使用NCP迁移到使用TCP）大约出现在25年以前。即使回到那时［RFC 801］，因特网很小且仍然由少数“奇才”管理着，人们也会认识到这样一个标志日是不可行的。当今一个牵涉上亿台机器和上百万个网络管理员与用户的标志日更是不可想象的。RFC 4213描述了两种方法（可单独使用，也可一起使用）可逐渐将IPv6主机和路由器整合进IPv4世界中（当然其长远目标是最终将所有IPv4节点向IPv6迁移）。

最直接的引入IPv6使能节点的方法可能是**双栈**（dual-stack），即IPv6节点也具有完整的IPv4实现。这样的节点在RFC 4213中被称为IPv6 /IPv4节点，它有发送和接收IPv4与IPv6两种数据报的能力。当与IPv4节点互操作时，IPv6 /IPv4节点可使用IPv4数据报；当与IPv6节点互操作时，它又能使用IPv6。IPv6 /IPv4节点必须具有IPv6 与IPv4两种地址。此外，它们还必须能确定另一个节点是否是IPv6使能或仅IPv4的。这个问题可使用DNS（参见第2章）来解决，若要解析的节点名字是IPv6使能的，则DNS会返回一个IPv6地址，否则返回一个IPv4地址。当然，如果发出DNS请求的节点是仅IPv4使能的，则DNS只返回一个IPv4地址。

在双栈方法中，如果发送方或接收方中任意一个是仅IPv4使能的，则必须使用IPv4数据报。因此，本质上两个IPv6使能的节点不应相互发送IPv4数据报。图4-25中举例说明了这种情况。假定节点A是IPv6使能的，且要向节点F发一个IP数据报，F也是IPv6使能的。节点A和

B可以交换IPv6数据报。然而，节点B必须创建一个IPv4数据报以便发给C。当然，IPv6数据报的数据字段可被复制到IPv4数据报的数据字段中，并且要做适当的地址映射。然而，在执行IPv6到IPv4的转换时，IPv6数据报中一些IPv6特定的字段（如流标签字段）在IPv4数据报中无对应部分，这些字段的信息将会丢失。因此，即使E和F能交换IPv6数据报，从D到达E的IPv4数据报也不含有从A发出的初始IPv6数据报中的所有字段。

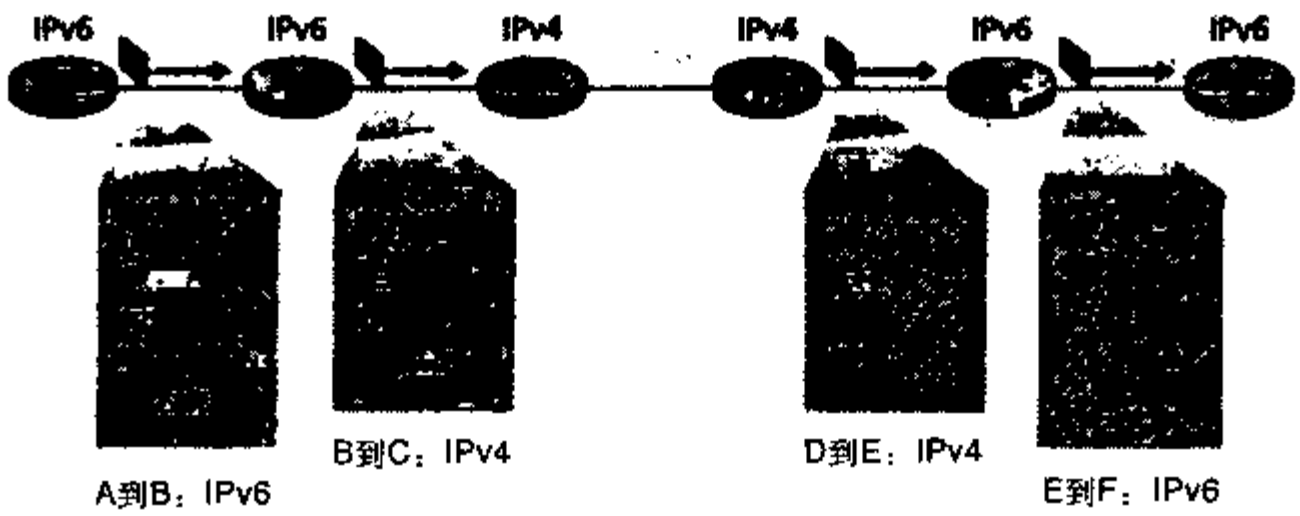


图4-25 一种双栈方法

RFC 4213中也讨论另一种双栈方法，叫做建隧道（tunneling）。该方法能解决上述问题，如可以使E接收由A生成的IPv6数据报。建隧道的基本思想如下：假定两个IPv6节点（如图4-25中的B和E）要使用IPv6数据报进行交互，但它们是经由中间IPv4路由器而互连的。我们将两台IPv6路由器之间中间IPv4路由器的集合称为一个隧道（tunnel），如图4-26所示。借助于隧道，在该隧道发送端的IPv6节点（如B）可将整个IPv6数据报放到一个IPv4数据报的数据（有效载荷）字段中。于是，该IPv4数据报地址设为指向隧道接收端的IPv6节点（如E），再发送给隧道中的第一个节点（如C）。隧道中的中间IPv4路由器在其间为该数据报选路，就像为其他数据报选路一样，完全不知道该IPv4数据报自身就含有一个完整的IPv6数据报。隧道接收端的IPv6节点最终收到该IPv4数据报（它是该IPv4数据报的目的地），并确定该IPv4数据报含有一个IPv6数据报，于是从中取出IPv6数据报，然后再为该IPv6数据报选路，就好像它是从一个直接相连的IPv6邻居那里接收到该IPv6数据报一样。

在结束本节时需要说明的是，尽管IPv6的采用是一个缓慢的过程［Lawton 2001］，但是最近已经有了进展。美国管理和预算局（OMB）已经要求到2008年6月迁移到IPv6上。大量增加的设备（如IP使能电话）与其他便携设备可能会推动IPv6的广泛应用。欧洲的第三代合作计划［3GPP 2007］已规定了IPv6为移动多媒体的标准编址方案。虽然IPv6在其年轻生命的前10年中还未得到广泛应用，但从长远的观点来看其前途一片光明。正如今天的电话号码系统历经几十年的成长期，现在它已盛行将近半个世纪，并无消失的迹象。类似地，IPv6也许会有一段时间的成长期，这段成长期也有可能相当长。因特网体系结构委员会［IAB 2007］前主席、与IPv6相关的几个RFC的作者Brian Carpenter说道："我一直认为这是一个起始于1995年的15年的进程。"［Lawton 2001］。按照Carpenter说的日期，我们仅处于该路程的3/4的位置！

我们能从IPv6经验中学到的重要一点就是，要改变网络层协议是极其困难的。自20世纪90年代早期以来，有许多新的网络层协议被鼓吹为因特网的下一次重大革命，但这些协议中的大多数都未能生存下来。这些协议包括IPv6、多播协议（4.7节）、资源预留协议（第7章）。

的确，在网络层中引入新的协议就如同替换一幢房子的基石（在不能将整幢房子拆掉或临时重新安置房屋住户的情况下是很难完成的）。另一方面，因特网却已见证了应用层中新协议的快速部署。典型的例子当然是Web、即时讯息、P2P文件共享。其他例子包括音频与视频流、分布式游戏。引入新的应用层协议就像给一幢房子重新刷一层漆（这是相对容易做的事），如果你选择了一个好看的颜色，邻居将会照搬你的选择。总之，我们在未来能够看到因特网网络层中的一些变化，但这些变化出现的时间程度要比在应用层中出现的变化慢得多。

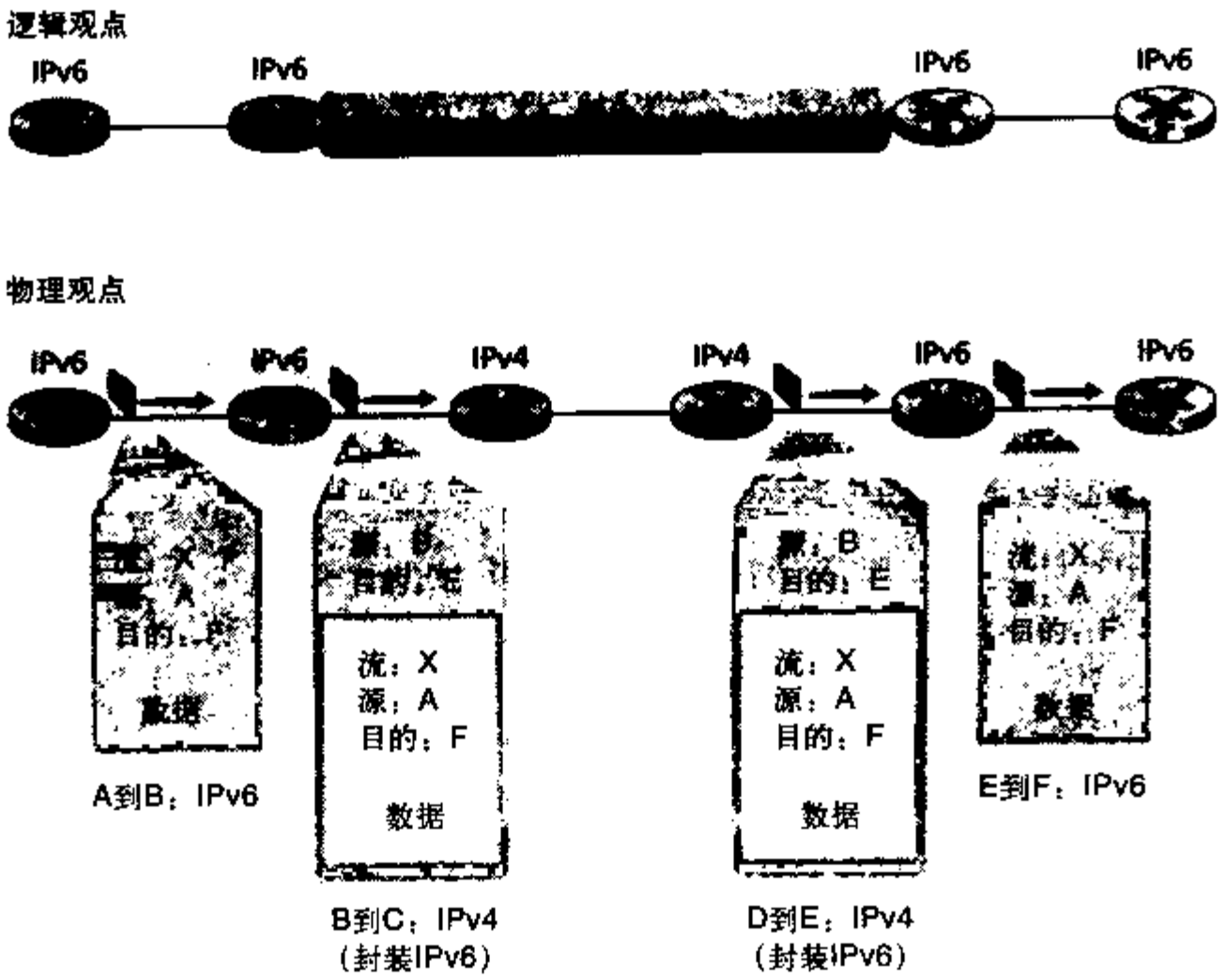


图4-26 隧道技术

4.4.5 IP安全性概述

4.4.3节详细地讨论了IPv4，包括它提供的服务以及如何实现这些服务。在阅读那节内容时，你可能注意到了并没有提到任何安全服务。的确，IPv4是在因特网主要用于相互信任的网络研究人员之间的时代（20世纪70年代）设计的。那时，创建一个能够集成众多链路层技术的计算机网络就已经面临巨大的挑战了，更别提顾及安全性了。

但是，在安全性越来越重要的今天，因特网研究人员开始设计提供各种安全性服务的新型网络层协议。其中一个是IPsec，它是流行的安全网络层协议之一，也是虚拟专用网（VPN）中广泛采用的协议。在本小节中我们对IPsec服务进行简要的、高层次的概述，在第8章中再详细地讨论IPsec及其密码学基础。

IPsec被设计为与IPv4和IPv6向后兼容。特别是，采用IPsec，不需要替代因特网中所有路由器和主机中的协议栈。例如，在运输模式（两种IPsec“模式”之一）下，如果两台主机要安全地通信，IPsec仅需要在这两台主机中可用，而所有的其他路由器和主机能够继续运行普通的IPv4。

为了具体起见，我们这里将关注IPsec的运输模式。在这种模式下，两台主机首先在它们之间创建一个IPsec会话。（因此IPsec是面向连接的！）使用适当的会话，这两台主机之间发

送的所有TCP和UDP报文段都享受IPsec提供的安全性服务。在发送端，运输层向IPsec传递一个报文段。然后IPsec加密该报文段，在该报文段上添加安全性字段，并且在一个普通的IP数据报中封装得到的有效载荷。(实际中比上述过程要复杂一点，如我们将在第8章看到的那样。)接下来发送主机向因特网中发送数据报，通过因特网将数据报传送到目的主机。在那里，IPsec解密报文段并将解密后的报文段传送给运输层。

由IPsec会话提供的服务包括：

- *密码技术协约*。允许两台通信的主机对加密算法和密钥取得一致的机制。
- *IP数据报有效载荷的加密*。当发送主机从运输层接收到一个报文段时，IPsec加密该有效载荷。该有效载荷仅能由接收主机中的IPsec解密。
- *数据完整性*。IPsec允许接收主机验证数据报的首部字段和被加密的有效载荷，以确保在数据报从源到目的地传输时没有被修改过。
- *初始鉴别*。当一台主机从某受信任的源（具有受信任的密钥，参见第8章）接收到一个IPsec数据报时，使该主机确信该数据报中的源IP地址是该数据报的实际源。

当两台主机在它们之间创建一个IPsec会话时，它们之间发送的所有TCP和UDP报文段都将被加密和鉴别。因此IPsec提供了地毯式覆盖，保证两台主机之间的所有网络应用的所有通信都是安全的。

一个公司通过使用IPsec来保证在非安全的公共因特网中能够进行安全通信。我们来看一个简单的例子。考虑一个拥有大批分布在各地的销售人员的公司，公司为每个销售人员配备一台便携机。假定销售人员需要经常查询公司的敏感信息（例如价格和产品信息），这些信息存储在公司总部的一台服务器上。使用IPsec如何做到这一点呢？如你猜想的那样，在这台服务器和所有销售人员的便携机上安装IPsec。借助于在这些主机上安装的IPsec，某销售人员无论何时与服务器通信或与另一名销售人员通信，他们的通信会话都将是安全的。

4.5 选路算法

到目前为止，我们在本章中主要研究了网络层的转发功能。我们知道，当分组到达一台路由器时，该路由器索引其转发表并决定该分组被指向的链路接口。我们也知道选路算法在网络路由器中运行、交换和计算，以配置这些转发表的信息。选路算法和转发表之间的相互影响如图4-2所示。较为深入地研究了转发后，我们将注意力转向本章的其他重要主题，即网络层的至关重要的选路功能。不管网络层提供的是数据报服务（在此情况下，在给定源和目的地址对之间传输不同分组可能采用不同的路由），还是虚电路服务（在此情况下，在给定源和目的地址之间传输的所有分组都将采用相同路径），网络层都必须确定从发送方到接收方的分组所采用的路径。我们将看到选路的工作是从发送方到接收方在通过路由器的网络中确定好路径（即路由）。

一台主机通常直接与一台路由器相连接，该路由器即为该主机的**默认路由器**（default router），又称为该主机的**第一跳路由器**（first-hop router）。每当某主机发送一个分组时，该分组都被传送给它的默认路由器。我们将源主机的默认路由器称为**源路由器**（source router），把目的主机的默认路由器称为**目的路由器**（destination router）。为一个分组从源主机到目的主机选路的问题显然可归结为从源路由器到目的路由器的选路问题。这是本节的重点。

选路算法的目的是很简单的：给定一组路由器以及连接路由器的链路，选路算法要找到一条从源路由器到目的路由器的“好”路径。通常，一条好路径是指具有最低费用的路径。

然而我们将看到，实践中还关心诸如策略之类的问题（例如，有一个规则是“属于组织Y的路由器x不应转发任何来源于组织Z网络的分组”），这也使得概念简单、性能优良的算法变得复杂了。然而这些概念简单、性能优良算法的理论奠定了当今网络选路实践的基础。

图论被用于形式化选路问题。我们知道，图（graph）$G = (N, E)$ 是一个N个节点和E条边的集合，其中每条边是来自N的一对节点。在网络层选路的环境中，图中的节点表示路由器，这是做出分组转发决定的点；连接节点的边表示路由器之间的物理链路。这样一个计算机网络的抽象图如图4-27所示。若要查看某些表示实际的网络图的图，可参阅［Dodge 2007，Cheswick 2000］；对于基于图不同的模型建模与因特网吻合程度的讨论，可参阅［Zegura 1997，Faloutsos 1999，Li 2004］。

如图4-27所示，一条边也有一个值表示它的费用。通常，一条边的费用可反映出对应链路的物理长度（例如，一条越洋链路的费用可能比一条短途陆地链路的费用高）、链路速度或与该链路相关的金融上的费用。为了我们的目的，我们只将这些链路费用看成是给定的，而不必操心这些值是如何确定的。对于E中的任一条边(x, y)，我们用$c(x, y)$表示节点x和y间边的费用。如果节点对(x, y)不属于E，我们置$c(x, y) = \infty$。此外，我们考虑的都是无向图（即图的边没有方向），因此边(x, y)与边(y, x)是相同的并且$c(x, y) = c(y, x)$。节点y也被称为节点x的邻居（neighbor），如果(x, y)属于E。

在抽象图中为各条边指派了费用后，选路算法的目标自然是找出从源到目的间的最低费用路径。为了使问题更为精确，回顾图$G = (N, E)$中的一条路径（path）是一个节点的序列$(x_1, x_2, \cdots, x_p)$，使得每一对$(x_1, x_2), (x_2, x_3), \cdots, (x_{p-1}, x_p)$是$E$中的边。路径$(x_1, x_2, \cdots, x_p)$的费用只是沿着路径所有边费用的总和，即$c(x_1, x_2) + c(x_2, x_3) + \cdots + c(x_{p-1}, x_p)$。给定任何两个节点$x$和$y$，通常在这两个节点之间有多条路径，每条路径都有一个费用。这些路径中的一条或多条是**最低费用路径**（least-cost path）。最低费用路径问题是显而易见的：找出源和目的之间具有最低费用的一条路径。例如，在图4-27中，源节点u和目的节点w之间的最低费用路径是(u, x, y, w)，其路径费用是3。注意，若图中的所有边具有相同的费用，则最低费用路径也就是**最短路径**（shortest path），即源和目的之间链路数量最少的路径。

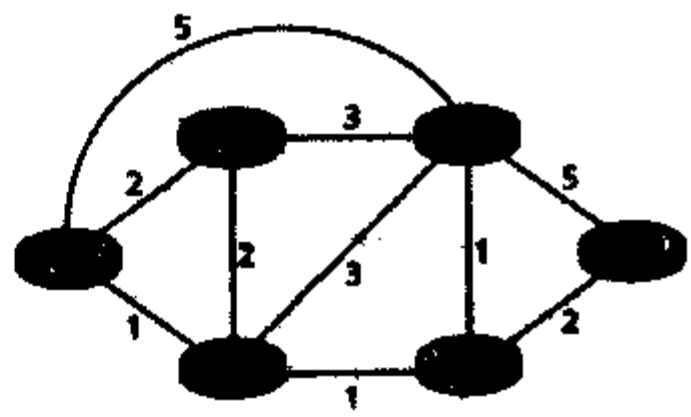


图4-27 一个计算机网络的抽象图模型

作为一个简单练习，试找出图4-27中从节点u到节点z的最低费用路径，并要反映出你是如何算出该路径的。如果你像大多数人一样，那么就是通过考察图4-27，跟踪几条从u到z的路由，找出从u到z的路径，然后以某种方式来确信你所选择的路径就是所有可能的路径中费用最低的路径。（你考察过u到z之间的所有17条可能的路径吗？很可能没有！）这种计算就是集中式选路算法的一个例子，即选路算法在一个位置运行，你的大脑中具有整个网络的信息。从广义上来说，我们对选路算法分类的一种方法就是根据该算法是全局性还是分布式来区分的。

- **全局选路算法**（global routing algorithm）用完整的、全局性的网络知识来计算从源到目的之间的最低费用路径。也就是说，该算法以所有节点之间的连通性及所有链路的费用为输入。这就要求该算法在真正开始计算以前，以某种方式获得这些信息。计算本身可

在某个场点（集中式全局选路算法）运行，也可在多个场点冗余。然而，这里主要的区别在于，全局性的算法具有关于连通性和链路费用方面的完整信息。实际上，具有全局状态信息的算法常被称作**链路状态**（Link-State，LS）算法，因为该算法必须知道网络中每条链路的费用。我们将在4.5.1节中学习LS算法。

- **分布式选路算法**（decentralized routing algorithm）以迭代的、分布式的方式计算出最低费用路径。没有节点拥有关于所有网络链路费用的完整信息，而每个节点仅有与其直接相连链路的费用知识即可开始工作。然后，通过迭代计算过程并与相邻节点（即与该节点相连链路的另一端的节点）交换信息，一个节点逐渐计算出到达某目的节点或一组目的节点的最低费用路径。我们将在4.5.2节学习一个称为**距离向量**（Distance-Vector, DV）算法的分布式选路算法。它之所以叫做DV算法，是因为每个节点维护到网络中的所有其他节点的费用（距离）估计的向量。

选路算法的第二种广义分类方法是根据算法是静态的还是动态的来分类。在**静态选路算法**（static routing algorithm）中，随着时间的推移，路由的变化是非常缓慢的，通常是由于人工干预进行调整（如人手工编辑一台路由器的转发表）。**动态选路算法**（dynamic routing algorithm）能够在网络流量负载或拓扑发生变化时改变选路路径。一个动态算法可周期性地运行或直接地响应拓扑或链路费用的变化而运行。虽然动态算法易于对网络的变化做出反应，但也更容易受诸如选路循环、路由振荡之类问题的影响。

选路算法的第三种分类方法是根据它是负载敏感的还是负载迟钝的进行划分。在**负载敏感算法**（load-sensitive algorithm）中，链路费用会动态地变化以反映出底层链路的当前拥塞水平。如果当前拥塞的一条链路被赋以高费用，则选路算法趋向于绕开该拥塞链路来选择路由。虽然早期的ARPAnet选路算法就是负载敏感的［McQuillan 1980］，但遇到了许多难题［Huitema 1998］。当今的因特网选路算法（如RIP、OSPF和BGP）都是**负载迟钝的**（load-insensitive），因为某条链路的费用不明显地反映其当前（或最近）拥塞水平。

4.5.1 链路状态选路算法

前面讲过，在链路状态（Link-State, LS）算法中，网络拓扑和所有的链路费用都是已知的，也就是说可用作LS算法的输入。在实践中，这是通过让每个节点向网络中的所有其他路由器广播链路状态分组来完成的，其中每个链路状态分组包含它所连接的链路的特征和费用。在实践中（例如使用因特网的OSPF选路协议，讨论见4.6.1节），这经常由**链路状态广播**（link state broadcast）算法［Perlman 1999］来完成。我们将在4.7节中讨论广播算法。节点广播的结果是所有节点具有了该网络的同一个完整的视图。于是每个节点都可像其他节点一样，运行LS算法并计算出相同的最低费用路径集合。

我们下面给出的链路状态选路算法叫做Dijkstra算法，该算法以其发明者命名。一个密切相关的算法是Prim算法，参见［Cormen 2001］中有关图算法的一般性讨论。Dijkstra算法计算从某节点（源节点，我们称之为u）到网络中所有其他节点的最低费用路径。Dijkstra算法是迭代算法，其性质是经算法的第k次迭代后，可知道到k个目的节点的最低费用路径，在到所有目的节点的最低费用路径之中，这k条路径具有k个最低费用。我们定义下列记号：

- $D(v)$：随着算法进行本次迭代，从源节点到目的节点v的最低费用路径的费用。
- $p(v)$：从源节点到目的节点v沿着当前最低费用路径的前一节点（v的邻居）。
- N'：节点子集；如果从源节点到目的节点v的最低费用路径已确知，v在N'中。

该全局选路算法由一个初始化步骤和其后的循环组成。循环执行的次数与网络中的节点个数相同。在结束时，该算法会计算出从源节点u到网络中每个其他节点的最短路径。

源节点u的链路状态（LS）算法

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v is a neighbor of u
5        then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for each neighbor v of w and not in N':
12         D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    least path cost to w plus cost from w to v */
15 until N'= N
```

举一个例子，考虑图4-27中的网络，计算从u到所有可能目的地的最低费用路径。该算法计算过程的总结如表4-3所示，表中的每一行给出了迭代结束时该算法的变量值。我们详细地考虑一下前几个步骤。

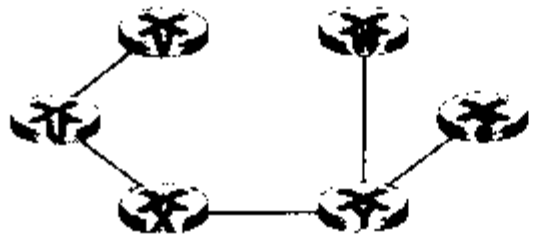
- 在初始化阶段，从u到与其直接相连的邻居v、x、w的当前已知最低费用路径分别初始化为2、1和5。特别注意的是，到w的费用被设为5（尽管我们很快就会看见确实存在一条费用更低的路径），因为这是从u到w的直接（一跳）链路费用。到y与z的费用被设为无穷大，因为它们不直接与u连接。
- 在第一次迭代时，我们观察那些还未加到集合N'中的节点，找出在前一次迭代结束时具有最低费用的节点。那个节点便是x，其费用是1，因此x被加到集合N'中。于是执行LS算法中的第12行程序以更新所有节点v的$D(v)$，产生表4-3中第2行（步骤1）所示的结果。到v的路径费用未变。经过节点x到w（在初始化结束时为5）的路径的费用被确定为4。因此这条具有更低费用的路径被选中，且沿从u开始的最短路径上w的前一个节点被设为x。类似地，到y（经过x）的费用被计算为2，且该表也被相应地更新。
- 在第二次迭代时，节点v与y被发现具有最低费用路径（2），我们任意选择将y加到集合N'中，使得N'中含有u、x和y。到仍不在N'中的其余节点（即节点v、w和z）的费用通过LS算法中的第12行被更新，产生如表4-3中第3行所示的结果。
- 以此类推……

表4-3 在图4-27中的网络上运行的链路状态算法

步 骤	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	$2, u$	$5, u$	$1, u$	∞	∞
1	ux	$2, u$	$4, x$		$2, x$	∞
2	uxy	$2, u$	$3, y$			$4, y$
3	$uxyv$		$3, y$			$4, y$
4	$uxyvw$					$4, y$
5	$uxyvwz$					

当LS算法结束时，对于每个节点，我们都得到从源节点沿着它的最低费用路径的前一节

点。对于每个前一节点，我们又有它的前一节点，按照此方式我们可以构建从源节点到所有目的节点的完整路径。通过为每个目的节点存放从u到它的最低费用路径上的下一跳节点，可以构建一个节点（如节点u）的转发表。图4-28显示了图4-27中网络的最低费用路径和u中的转发表。



目的地	链路
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

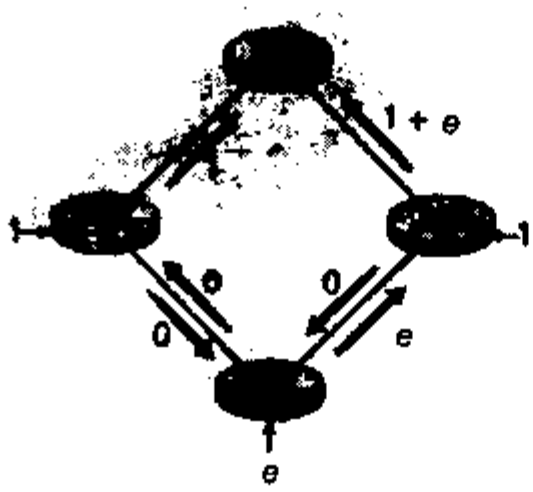
图4-28 最低费用路径和节点u的转发表

该算法的计算复杂性是什么？即给定n个节点（不算源节点），在最坏情况下要经多少次计算，才能找到从源节点到所有目的节点的最低费用路径？在第一次迭代时，我们需要搜索所有的n个节点以确定出节点w，w不在N'中且具有最低费用。在第二次迭代时，我们需要检查$n-1$个节点以确定最低费用。第三次需要迭代$n-2$个节点，以此类推。总之，我们在所有迭代中需要搜寻的节点总数为$n(n+1)/2$，因此我们说前面实现的链路状态算法在最坏情况下的复杂性为n平方阶序：$O(n^2)$。（该算法的一种更复杂的实现是使用一种称为堆的数据结构，能用对数时间而不是线性时间找到第9行中的最小值，因此减少其复杂性。）

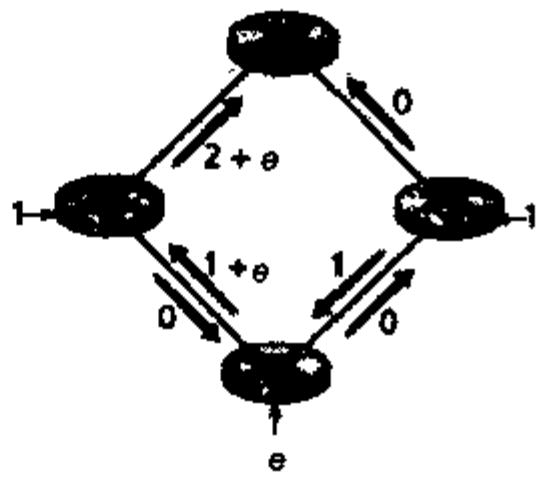
在结束对LS算法的讨论之前，我们考虑一下可能出现的问题。图4-29显示了一个简单的网络拓扑，图中的链路费用等于链路上承载的负载，例如反映要经历的时延。在该例中，链路费用是非对称的，即$c(u, v)$与$c(v, u)$仅当在链路(u, v)两个方向所承载的负载相同时才相等。在该例中，节点z产生发往w的一个单元的流量，节点x也产生发往w的一个单元流量，并且节点y发往w一个数量为e的流量。初始选路情况如图4-29a所示，其链路费用相当于承载的流量。

当LS算法再次运行时，节点y确定（基于图4-29a所示的链路费用）顺时针到w的路径费用为1，而逆时针到w的路径（一直在使用中）费用是$1+e$。因此y到w的最低费用路径现在是顺时针的。类似地，x确定其到w的新的最低费用路径也是顺时针的，产生如图4-29b所示的费用。当LS算法再次运行时，节点x、y和z都检测到一条至w的逆时针方向零费用路径，它们都将其流量引导到逆时针方向的路由上。下次LS算法运行时，x、y和z都将其流量引导到顺时针方向的路由上。

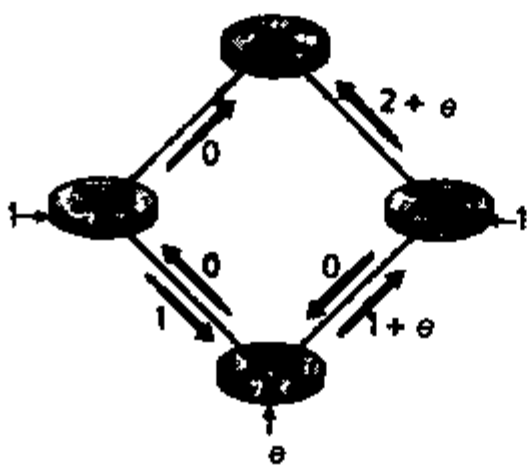
如何才能防止这样的振荡？（它不只是出现在使用拥塞或基于时延的链路测度的链路状态算法中，而可能出现在任何算法中。）一种解决方案可能是强制链路费用不依赖于所承载的流量，但这是一种不可接受的解决方案，因为选路的目标之一就是要避开高拥塞（如高时延）的链路。另一种解决方案就是确保并非所有的路由器都同时运行LS算法。这似乎是一个更合理的方案，因为我们希望即使路由器以相同周期运行LS算法，在每台路由器上算法执行的时机也将是不同的。有趣的是，研究人员近来已注意到了因特网上的路由器能在彼此之间自同步［Floyd Synchronization 1994］。这就是说，即使它们初始时以同一周期但在不同时刻执行算法，算法执行时机最终也会在路由器上变为同步并保持之。避免这种自同步的一种方法是，对于每台路由器随机地发送链路通告的时间。



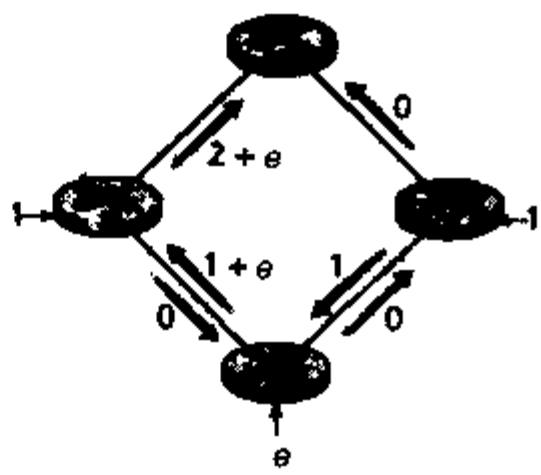
a) 初始选路



b) x和y检测至w的更好路径，顺时针



c) x、y、z检测至w的更好路径，逆时针



d) x、y、z检测至w的更好路径，顺时针

图4-29 使用链路状态选路时的振荡

学习过LS算法之后，我们接下来考虑目前在实践中使用的其他重要选路算法，即距离向量选路算法。

4.5.2 距离向量选路算法

LS算法是一种使用全局信息的算法，而距离向量（Distance-Vector, DV）算法是一种迭代的、异步的和分布式的算法。说它是分布式的，是因为每个节点都要从一个或多个直接相连的邻居接收某些信息，执行计算，然后将计算结果发回给邻居。说它是迭代的，是因为此过程一直要持续到邻居之间没有更多的信息要交换为止。（有趣的是，此算法是自我终结的，即没有计算应该停止的信号，它就停止了。）说该算法是异步的，是因为它不要求所有节点相互之间步伐一致地操作。我们将看到，一个异步的、迭代的、自我终结的、分布式的算法比一个集中式的算法要有趣得多。

在我们给出DV算法之前，有必要先讨论一下存在于最低费用路径的费用之间的一种重要关系。令$d_x(y)$是从节点x到节点y的最低费用路径的费用，则该最低费用与著名的Bellman-Ford方程相关，即

$$d_x(y) = \min_v\{c(x, v) + d_v(y)\} \tag{4-1}$$

方程中的$\min_v$是指取遍x的所有邻居。Bellman-Ford方程是相当直观的。实际上，从x到v遍历之后，如果我们取从v到y的最低费用路径，该路径费用将是$c(x, v) + d_v(y)$。因此我们必须

从遍历某些邻居v开始，从x到y的最低费用是对所有邻居v的$c(x, v) + d_v(y)$的最小值。

但是对于那些怀疑该方程正确性的人，我们核查图4-27中的源节点u和目的节点z。源节点u有3个邻居：节点v、x和w。通过遍历该图中的各条路径，容易看出$d_v(z) = 5$，$d_x(z) = 3$，$d_w(z) = 3$。将这些值带入方程（4-1），连同费用$c(u, v) = 2$，$c(u, x) = 1$，$c(u, w) = 5$，给出$d_u(z) = \min\{2+5, 5+3, 1+3\} = 4$，这显然是正确的，并且这正是Dijskstra算法对同一个网络为我们提供的结果。这种迅速的验证应当有助于消除你可能具有的任何怀疑。

Bellman-Ford方程不仅仅满足了智力上的好奇心，它实际上具有显著的实践重要性。特别是对Bellman-Ford方程的解答为节点x的转发表提供了表项内容。为了理解这一点，令v^*是取得方程（4-1）中最小值的任何相邻节点。接下来，如果节点x要沿着最低费用路径向节点y发送一个分组，它应当首先向节点v^*转发该分组。因此，节点x的转发表将指定节点v^*作为最终目的地y的下一跳路由器。Bellman-Ford方程的另一个实际贡献是它提出了DV算法中发生的邻居到邻居通信的形式。

其基本思想如下。每个节点x以$D_x(y)$开始，对N中的所有节点估计从它自己到节点y的最低费用路径的费用。令$\boldsymbol{D}_x=[D_x(y): y\in N]$是节点$x$的距离向量，该向量是从$x$到$N$中的所有其他节点$y$的费用估计向量。采用DV算法，每个节点$x$维护下列选路数据：

- 对于每个邻居v，从x到直接相连邻居v的费用为$c(x, y)$。
- 节点x的距离向量（即$\boldsymbol{Dx} = [D_x(y): y\in N]$）包含了$x$到$N$中所有目的地的费用的估计值。
- 它的每个邻居的距离向量，即对x的每个邻居v有$\boldsymbol{D}_v=[D_v(y): y\in N]$。

在该分布式的异步算法中，每个节点不时地向它的每个邻居发送它的距离向量拷贝。当节点x从它的任何一个邻居v接收到一个新距离向量时，它保存v的距离向量，然后使用Bellman-Ford方程更新它自己的距离向量：

$$D_x(y) = \min_v\{c(x, v) + D_v(y)\}，对N中的每个节点$$

如果节点x的距离向量因这个更新步骤而改变，则节点x将向它的每个邻居发送它的更新的距离向量。令人惊奇的是，只要所有的节点继续以异步方式交换它们的距离向量，每个费用估计$D_x(y)$就收敛到$d_x(y)$，$d_x(y)$是从节点x到节点y的实际最低费用路径的费用[Bersekas 1991]!

距离向量算法（DV算法）

对每个节点x：

```
1  Initialization:
2     for all destinations y in N:
3        D_x(y) = c(x,y)    /* if y is not a neighbor then c(x,y) = ∞ /*
4     for each neighbor w
5        D_w(y) = ∞ for all destinations y in N
6     for each neighbor w
7        send distance vector D_x =  [D_x(y): y in N] to w
8
9  loop
10    wait (until I see a link cost change to some neighbor w or
11          until I receive a distance vector from some neighbor w)
12
13    for each y in N:
14       D_x(y) = min_v{c(x,v) + D_v(y)}
15
16    if D_x(y) changed for any destination y
17       send distance vector D_x =  [D_x(y): y in N] to all neighbors
18
19 forever
```

在该DV算法中，当节点x看到它的直接相连的链路费用变化或从某个邻居接收到一个距离向量的更新时，就更新其距离向量。但是为了一个给定的目的地y而更新它的转发表，节点x真正需要知道的不是到y的最短路径距离，而是邻居节点$v^*(y)$的最短路径距离，它是沿着最短路径到y的下一跳路由器。如你所期望的那样，下一跳路由器$v^*(y)$是在DV算法第14行中取得最小的邻居v。（如果有多个取得最小的邻居v，则$v^*(y)$可以是其中任何最小的邻居。）因此，对于每个目的地y，在第13～14行中，节点x也决定$v^*(y)$并更新它对目的地y的转发表。

前面讲过，LS算法是一种全局算法，所以它要求每个节点在运行Dijkstra算法之前，首先获得该网络的完整信息。DV算法是分布式的，它不使用这样的全局信息。实际上，节点具有的唯一信息是它到直接相连邻居的链路费用和它从这些邻居接收到的信息。每个节点等待来自任何邻居的更新（第10～11行），当接收到一个更新时计算它的新距离向量（第14行）并向它的邻居分发它的新距离向量（第16～17行）。DV算法在实践中用于许多选路协议中，包括因特网的RIP和BGP、ISO IDRP、Novell IPX和早期的ARPAnet。

图4-30举例说明了DV算法的运行，应用场合是该图顶部有3个节点的简单网络。算法以同步的方式显示出来，其中所有节点同时从其邻居接收报文，计算其新距离向量，在距离向量发生变化时通知其邻居。学习完这个例子后，你应认识到算法以异步方式也能正确运行，在此情况下，可在任意时刻进行节点计算与更新产生/接收。

该图最左边显示了这3个节点各自的初始选路表（routing table）。例如，位于左上角的表是节点x的初始选路表。在一张特定的选路表中，每行是一个距离向量，特别是每个节点的选路表包括了它的距离向量和它的每个邻居的距离向量。因此，在节点x的初始选路表中，第一行是$D_x=[D_x(x), D_x(y), D_x(z)] = [0, 2, 7]$，第二行和第三行分别是最近从节点$y$和$z$收到的距离向量。因为在初始化时，节点$x$还没有从节点$y$和$z$收到任何东西，所以第二行和第三行表项被初始化为无穷大。

初始化后，每个节点向它的两个邻居发送其距离向量。图4-30中显示的从表的第一列到表的第二列的箭头就说明了这一情况。例如，节点x向两个节点y和z发送了它的距离向量$D_x=[0, 2, 7]$。在接收到该更新后，每个节点重新计算它自己的距离向量。例如，节点x计算

$$D_x(x) = 0$$

$$D_x(y) = \min\{c(x, y) + D_y(y), c(x, z) + D_z(y)\} = \min\{2 + 0, 7+1\} = 2$$

$$D_x(z) = \min\{c(x, y) + D_y(z), c(x, z) + D_z(z)\} = \min\{2 + 1, 7+0\} = 3$$

因此，第二列为每个节点显示了该节点的新距离向量连同刚从它的邻居接收到的距离向量。注意，如节点x到节点z的最低费用估计$D_x(z)$已经从7变成了3。还应注意，对于节点x，节点y在该DV算法的第14行中取得了最小值。因此，在该算法的这个阶段，我们在节点x得到了$v^*(y) = y$和$v^*(z) = y$。

在节点重新计算它们的距离向量之后，它们再次向其邻居发送它们的更新距离向量（如果它们已经改变的话）。图4-30中显示的从表的第二列到表的第三列的箭头就说明了这一情况。注意，仅有节点x和节点z发送了更新：节点y的距离向量没有发生变化，因此节点y没有发送更新。在接收到这些更新后，这些节点则重新计算它们的距离向量并更新它们的选路表，这些显示在第三列中。

从邻居接收更新距离向量、重新计算选路表项和通知邻居到目的地的最低费用路径的费用已经变化的过程继续下去，直到无更新报文发送为止。此时，因为无更新报文发送，所以将不出现进一步选路表计算，该算法将进入静止状态，即所有的节点将执行DV算法的第10～

11行中的等待。该算法停留在静止状态直到一条链路费用发生变化，如下面所讨论的那样。

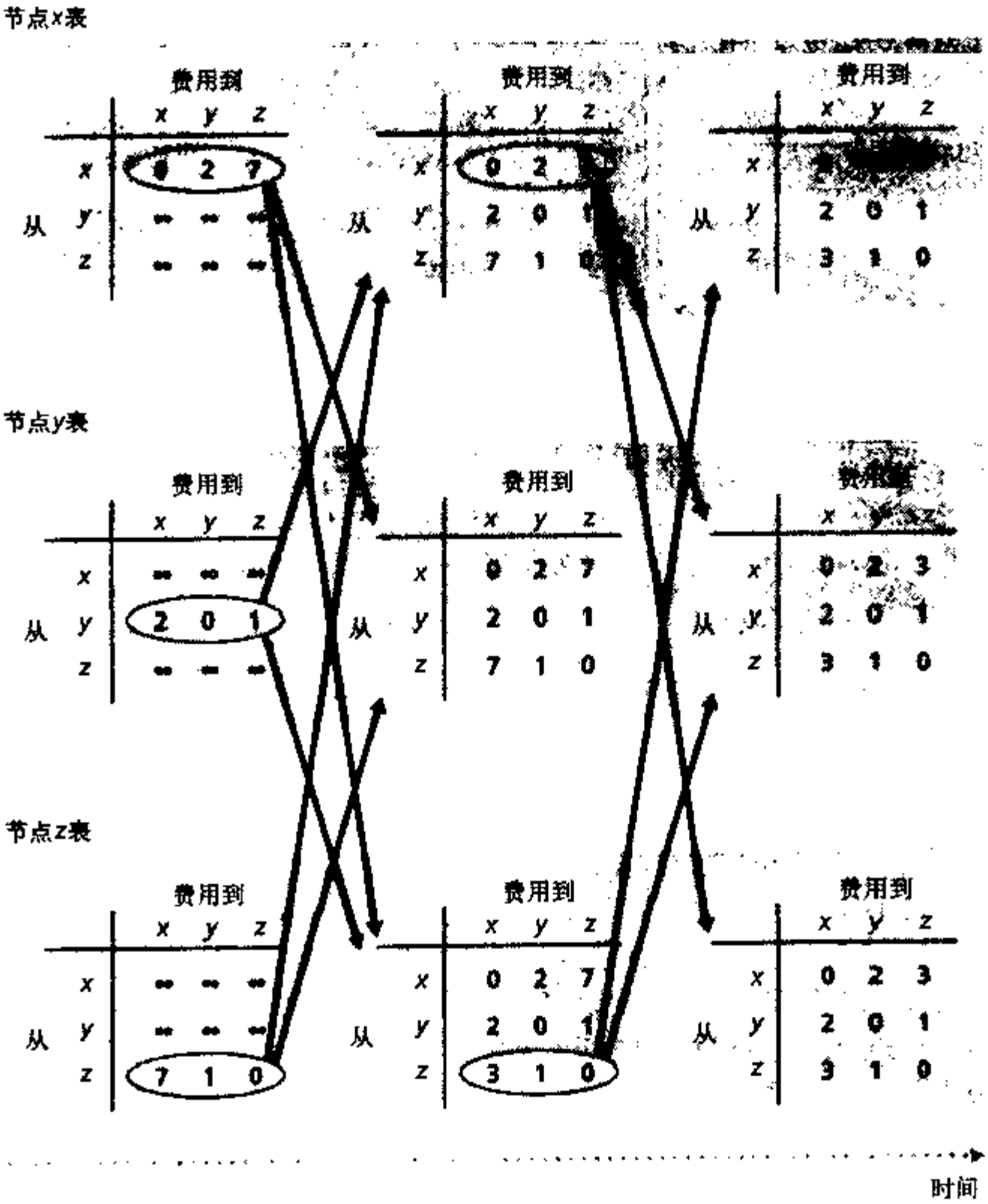


图4-30 距离向量（DV）算法

1. 距离向量算法：链路费用变化与链路故障

当一个运行DV算法的节点检测到从它自己到邻居的链路费用发生变化时（第10～11行），它就更新其距离向量（第13～14行），并且如果最低费用路径的费用发生了变化，则向邻居通知其新的距离向量（第16～17行）。图4-31a举例说明了从y到x的链路费用从4变为1的情况。我们在此只关注y与z到目的地x的距离表中的有关表项。该DV算法引发了下列事件序列：

- 在t_0时刻，y检测到链路费用变化（费用从4变为1），更新其距离向量，并通知其邻居这个变化，因为其距离向量已改变。
- 在t_1时刻，z收到来自y的更新报文并更新了其距离表。它计算出到x的新最低费用（从费用5减为费用2），向其邻居发送了它的新距离向量。
- 在t_2时刻，y收到来自z的更新并更新其距离表。y的最低费用未变，因此y不发送任何报文给z。该算法进入静止状态。

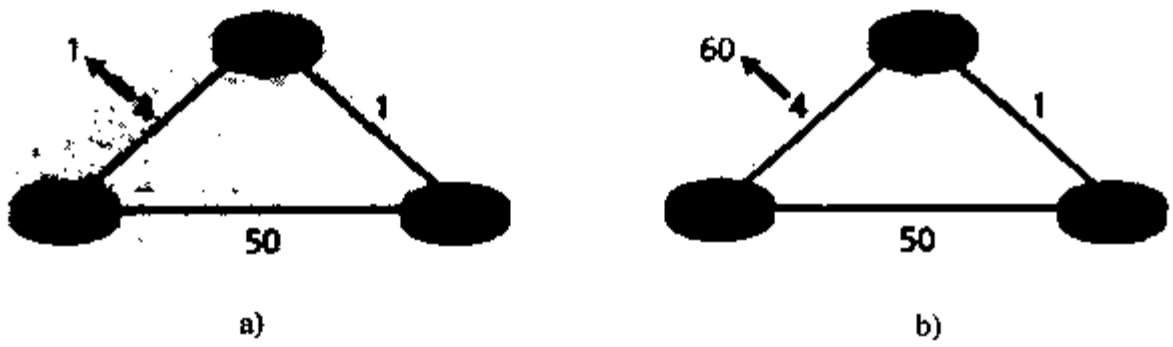


图4-31 链路费用改变

因此，对于该DV算法只需两次迭代就到达了静止状态。x与y之间费用减少的好消息通过网络得到了迅速传播。

我们现在考虑一下当某链路费用增加时发生的情况。假设x与y之间的链路费用从4增加到60，如图4-31b所示。

1）在链路费用变化之前，$D_y(x)=4$，$D_y(z)=1$，$D_z(y)=1$，$D_z(x)=5$。在t_0时刻，y检测到链路费用变化（费用从4变为60）。y计算其到x的新最低费用路径的费用：

$$D_y(x)=\min\{c(y,x)+D_x(x),c(y,z)+D_z(x)\}=\min\{60+0,1+5\}=6$$

当然，从网络全局角度来看，我们可以看出经过z的新费用是错误的。但节点y仅有的信息是：它到x的直接费用是60，且z上次已告诉y，z能以费用5到x。因此，为了到达x，y将通过z选路，完全期望z能以费用5到达x。到了t_1时刻，我们遇到**选路环路**（routing loop），即为到达x，y通过z选路，z又通过y选路。选路环路就像一个黑洞，目的地为x的分组在t_1时到达y或z后，将在这两个节点之间不停地（或直到转发表发生改变为止）来回反复。

2）因为节点y已算出到x的新最低费用，所以它在t_1时刻将该新距离向量通知z。

3）在t_1后的某个时刻，z收到y的新距离向量，它指示了y到x的新最低费用是6。z知道它能以费用1到达y，因此计算出到x的新最低费用$D_z(x)=\min\{50+0,1+6\}=7$。因为$z$到$x$的最低费用已增加了，于是它便在$t_2$时刻通知$y$其新距离向量。

4）以类似方式，在收到z的新距离向量后，y决定$D_y(x)=8$并向z发送其距离向量。接下来z决定$D_z(x)=9$并向y发送其距离向量，等等。

该过程将要持续多久呢？你应认识到循环将持续44次迭代（y与z之间的报文交换），直到z最终算出它经由y的路径费用大于50为止。此时，z将（最终）确定它到x的最低费用路径是经过它到x的直接连接。y将经由z选路到x。关于链路费用增加的坏消息的确传播得很慢！如果链路费用$c(y,x)$从4变为10 000且费用$c(z,x)$为9999时将发生什么情况呢？在这种情况下，我们所见到的问题有时被称为计数到无穷（count-to-infinity）问题。

2. 距离向量算法：增加毒性逆转

刚才描述的特定环路情况可以通过一种称为**毒性逆转**（poisoned reverse）的技术而加以避免。其基本思想较为简单：如果z通过y选路到达目的地x，则z将通告y，它（z）到x的距离

是无穷大，即z将向y通告$D_z(x) = \infty$（即使z实际上知道$D_z(x) = 5$）。只要z经y选路到x，它就持续地向y讲述这个善意的谎言。因为y相信z没有到x的路径，故只要z继续经y选路到x（并谎称这么做），y就永远不会试图经由z选路到x。

我们现在看一下毒性逆转是如何解决图4-31b中遇到的特定环路问题的。作为毒性逆转的结果，y的距离表指示了$D_z(x) = \infty$。当(x, y)链路的费用在t_0时刻从4变为60时，y更新其表，虽然费用高达60仍继续直接选路到x，并将到x的新费用通知z，即$D_y(x) = 60$。z在t_1时刻收到更新后，便立即将其到x的路由为经过费用50的直接(z, x)链路。因为这是一条新的到x的最低费用路径，且因为路径不再经过y，所以z就在t_2时刻通知y现在$D_z(x) = 50$。在收到来自z的更新后，y便用$D_y(x) = 51$更新其距离表。另外，因为z此时位于y到x的最低费用路径上，所以y毒化从z到x的方向路径，在t_3时刻通知z，$D_y(x) = \infty$（即使y实际上知道$D_y(x) = 51$）。

毒性逆转解决了常见的不可计数问题吗？没有。你应认识到涉及3个或更多节点（而不只是两个直接相连的邻居节点）的环路将不能被毒性逆转技术检测到。

3. LS与DV选路算法的比较

DV和LS算法采用互补的方法来解决计算选路问题。在DV算法中，每个节点仅与它的直接相连邻居交谈，但它为它的邻居提供了从其自己到网络中（它所知道的）所有其他节点的最低费用估计。在LS算法中，每个节点（经广播）与所有其他节点交谈，但它仅告诉它们与它直接相连链路的费用。我们使用快速比较它们各自属性的方式来总结我们所学的链路状态与距离向量算法。前面讲过N是节点（路由器）的集合，而E是边（链路）的集合。

- 报文复杂性。我们已经看到LS算法要求每个节点都知道网络中每条链路的费用。这就要求发送$O(|N||E|)$个报文。而且无论何时一条链路的费用改变，都必须向所有节点发送新的链路费用。DV算法要求在每次迭代时，在两个直接相连邻居之间交换报文。我们已经看到算法收敛所需的时间依赖于许多因素。当链路费用改变时，DV算法仅当在新的链路费用导致与该链路相连节点的最低费用路径发生改变时，才传播已改变的链路费用。
- 收敛速度。我们已经看到LS算法的实现是一个要求$O(|N||E|)$个报文的$O(|N|^2)$算法。DV算法收敛较慢，且在收敛时会遇到选路环路。DV算法还会遭受到计数到无穷的问题。
- 健壮性。如果一台路由器发生故障、行为错乱或受到破坏，情况会怎样呢？对于LS算法，路由器能够向其连接的一条链路广播不正确费用（但是没有其他）。作为LS广播的一部分，一个节点也可损坏或丢弃它收到的任何LS广播分组。但是一个LS节点仅计算自己的转发表，其他节点也自行执行类似的计算。这就意味着在LS算法下，路由计算在某种程度上是分离的，这提供了一定程度的健壮性。在DV算法下，一个节点可向任意或所有目的节点通告其不正确的最低费用路径。（实际上，1997年，一个小ISP的一台有故障的路由器向美国国家的主干路由器提供了错误的选路信息，这引起了其他路由器将大量流量引向该故障路由器，并导致因特网的大部分中断连接数小时［Neumann 1997］。）更一般地，我们注意到每次迭代时，DV算法中一个节点的计算会传递给它的邻居，然后在下次迭代时再间接地传递给邻居的邻居。在此情况下，DV算法中一个不正确的节点计算值会扩散到整个网络。

总之，没有一个算法对另一个算法而言是赢家。事实上，这两个算法都在因特网中得到了应用。

4. 其他选路算法

我们已学过的LS算法与DV算法不仅在实践中得到了广泛的应用，而且它们基本上是当前

因特网中使用的仅有的两种选路算法。无论如何，在过去的30年里，研究人员已提出了许多选路算法，从非常简单的到非常复杂的都有。选路算法的广义分类基于将分组流量看作是网络中源和目的之间的流。用这种方法，选路问题可在数学上被形式化为一个称为网络流问题的受限优化问题［Bertsekas 1991］。然而，我们在此要提及的另外一类算法集合，是那些来源于电话界的选路算法。当每条链路资源（如缓冲区、链路带宽的一部分）都要保留给每个要经过该链路的连接时，这些**电路交换选路算法**（circuit-switched routing algorithm）对分组交换数据网是很有意义的。虽然该选路问题的表述可能看起来与我们在本章看到的最低费用选路的表述存在着非常大的差异，但也有许多相似的地方，至少对于路径查找算法（选路算法）是这样的。关于该研究领域的更详细讨论，可参考［Ash 1998; Ross 1995; Girard 1990］。

4.5.3 层次选路

在LS和DV算法研究中，我们将网络只看作是一个互连路由器的集合。从所有路由器执行相同的选路算法以计算穿越整个网络的选路路径这个意义上来说，一台路由器很难同另一台路由器区别开来。在实践中，该模型和关于一组执行同样选路算法的同质路由器集合的观点有一点简单化，至少出于以下两个重要原因：

- *规模*。随着路由器数目变得很大，选路信息的计算、存储及通信（例如LS更新或最低费用路径的变化）的开销逐渐高得惊人。当今的公共因特网由数亿台主机组成。在这些主机中存储的选路信息显然需要巨大容量的内存。公共因特网上的所有路由器中要求的广播LS更新的开销将导致没有剩余的带宽供发送数据分组使用！距离向量算法在如此大量的路由器中的迭代将肯定永远不会收敛！显然，必须采取一些措施来减少像公共因特网这么大的网络中选路计算的复杂性。
- *管理自治*。虽然研究人员倾向于忽略这样的问题，如某公司要求按自己的意愿运行路由器（如运行其选择的某种选路算法），或对外部隐藏其网络的内部组织面貌，但这些都是需要考虑的重要因素。理想情况下，一个组织应当能够按自己的愿望运行和管理其网络，还要能将其网络与其他外部网络连接。

这两个问题都可以通过将路由器组织进**自治系统**（Autonomous System，AS）来解决，每个AS由一组通常在相同管理控制下的路由器组成（例如，由相同的ISP运营或属于相同的公司网络）。在相同的AS内的路由器都全部运行同样的选路算法（如LS或DV算法），且拥有彼此之间的信息（就像在前一节中所讲的理想化模型一样）。在一个自治系统内运行的选路算法叫做**自治系统内部选路协议**（intra-autonomous system routing protocol）。当然，将AS彼此互连将是必需的，因此在一个AS内的一台或多台路由器将有另外的任务，来负责向本AS之外的目的地转发分组。这些路由器被称为**网关路由器**（gateway router）。

图4-32提供了具有3个AS的简单例子：AS1、AS2和AS3。在该图中，粗线表示了路由器对之间的直接链路连接。从路由器连出的细线表示直接与路由器连接的子网。AS1具有4台路由器：1a、1b、1c和1d，它们运行在AS1内部，使用了AS1内部选路协议。因此，这4台路由器都知道如何沿着优化路径将分组转发到AS1内的任何目的地。类似地，自治系统AS2和AS3都有3台路由器。注意到运行在AS1、AS2和AS3中的AS内部选路协议不必是相同的。同时注意到路由器1b、1c、2a和3a都是网关路由器。

现在应当清楚在一个AS中路由器是如何为AS内部的源和目的对确定选路路径的。但是对于端到端选路难题仍然有一大块遗漏的部分，即在某些AS中一台路由器怎样知道将分组选路

到位于该AS外部的目的地呢？如果AS仅有一个网关路由器连接唯一一个其他AS的话，是很容易回答这个问题的。在这种情况下，因为该AS内部的AS选路协议已经决定了从内部路由器到网关路由器的最低费用路径，因此每台内部路由器知道它应当如何转发分组。网关路由器一旦接收到分组，便将分组向通向外部AS的一条链路转发。该链路另一端的AS承担起将该分组向其最终目的地选路的责任。举一个例子，假定图4-32中的路由器2b接收到一个分组，其目的地位于AS2的外部。路由器2b则将该分组转发到路由器2a或2c，这由路由器2b的转发表指定，该转发表由AS2的AS内部选路协议配置。该分组最后到达网关路由器2a，它又将该分组转发到1b。一旦该分组离开2a，AS2对这个分组的任务就完成了。

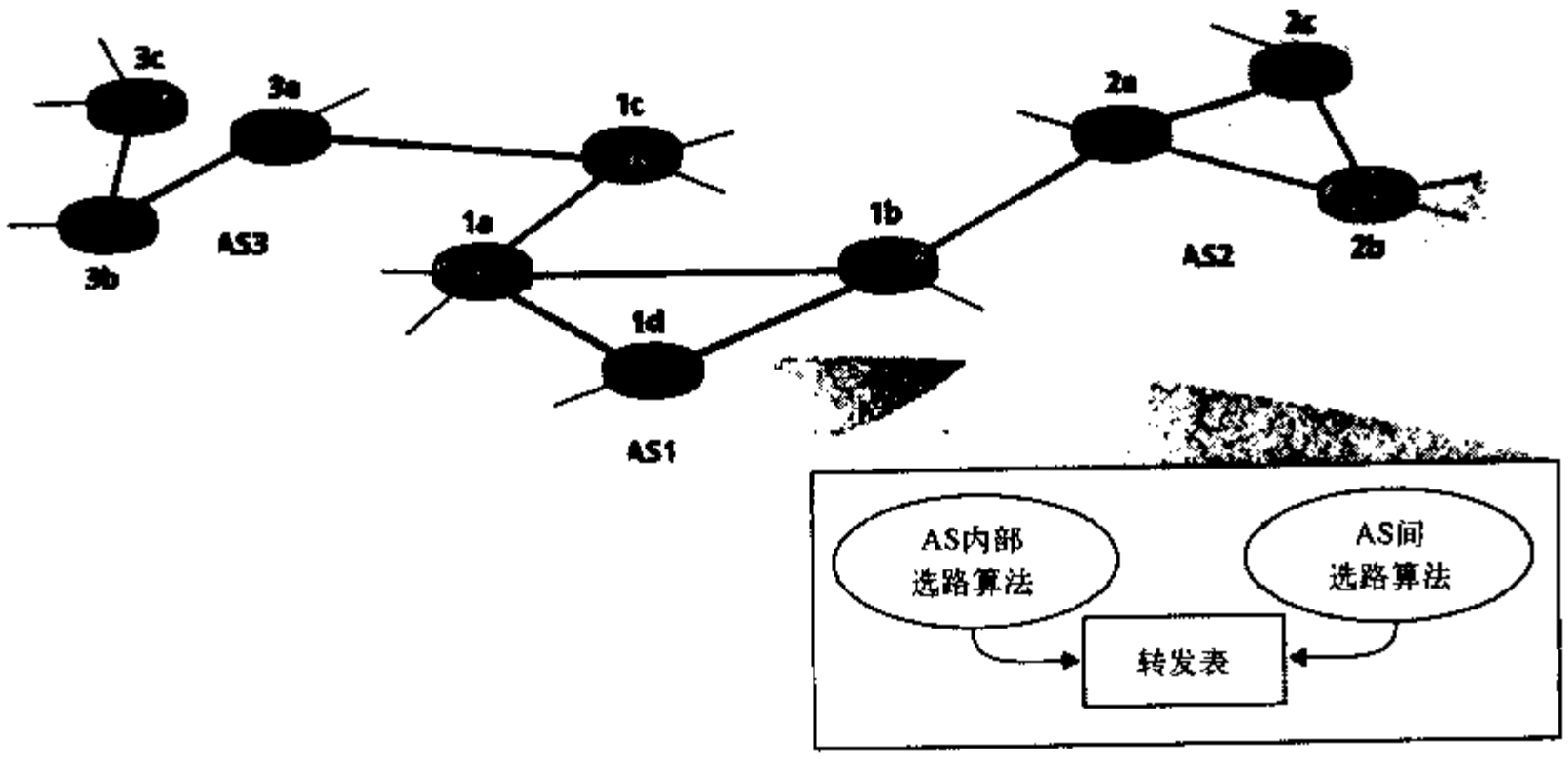


图4-32 一个互连的自治系统的例子

因此当源AS仅有一条通向外部AS的链路时，该问题很容易解决。如果源AS具有两条或更多条链路（通过一台或更多台网关路由器）通向外部AS，情况将会怎样呢？此时，向何处转发该分组的问题具有更大的挑战性。例如，考虑AS1中的一台路由器，假定它接收了一个目的地在该AS外部的分组。该路由器显然应当向它的两个网关路由器之一（1b或1c）转发该分组，但到底是哪个呢？为了解决这个问题，AS1需要：① 知道经AS2可达哪些目的地，经AS3可达哪些目的地；② 向AS1中的所有路由器传播这些可达性信息。因此，每台路由器能够配置它的转发表以处理外部AS目的地。从相邻AS获取可达性信息以及向该AS中的所有路由器传播可达性信息这两项任务由**自治系统间选路协议**(inter-autonomous system routing protocol)负责。因为自治系统间选路协议涉及两个AS之间的通信，所以这两个通信的AS必须运行相同的自治系统间选路协议。事实上，因特网中的所有AS都运行相同的自治系统间选路协议（称为BGP4），我们将在下一节中讨论该协议。如图4-32所示，每台路由器接收来自一个AS内部选路协议和一个AS间选路协议的信息，并使用来自这两个协议的信息配置其转发表。

举一个例子，考虑子网x（由它的CDIR化的地址标识），假定AS1从AS间选路协议知道子网x从AS3可达，而从AS2不可达。AS1则向它的所有路由器传播这个信息。当路由器1d知道从AS3可达子网x时，因此对于网关1c而言，它根据AS内部选路协议提供的信息来决定从路由器1d到网关路由器1c的位于最低费用路径上的路由器接口，比如说接口I。路由器1d则能够将表项(x, I)放入其转发表中。（这个例子和本节中给出的其他例子体现了因特网中实际发生

情况的一般化但被简化了的概念。在下一节中，当我们讨论BGP时，将提供一个更为详细的描述。）

接着上面的例子，现在假设AS2和AS3与其他AS相连，这些AS并没有在该图上显示出来。同时假定AS1从AS间选路协议知道了子网*x*是可达的，或者从AS2经网关1b可达，或者从AS3经网关1c可达。AS1则向它的所有路由器（包括路由器1d）传播该信息。为了配置其转发表，路由器1d将必须决定通过1b或1c中的哪个网关路由器来指引目的地为子网*x*的分组。在实践中经常使用的一种方法是**热土豆选路**（hot potato routing）。在热土豆选路中，AS尽可能快（更准确地讲，是尽可能便宜）地摆脱分组（热土豆）。这通过让路由器向某网关路由器发送分组来完成，同时要求该网关路由器在到目的地路径上的所有网关路由器中具有最低费用。在当前例子中，在1d中运行的热土豆选路将使用来自AS内部选路协议的信息，以决定到1b和1c的路径费用，并选择具有最低费用的路径。一旦选择这条路径，路由器1d就在其转发表中增加用于子网*x*的表项。图4-33总结了路由器1d对转发表增加用于*x*的表项所采取的动作。

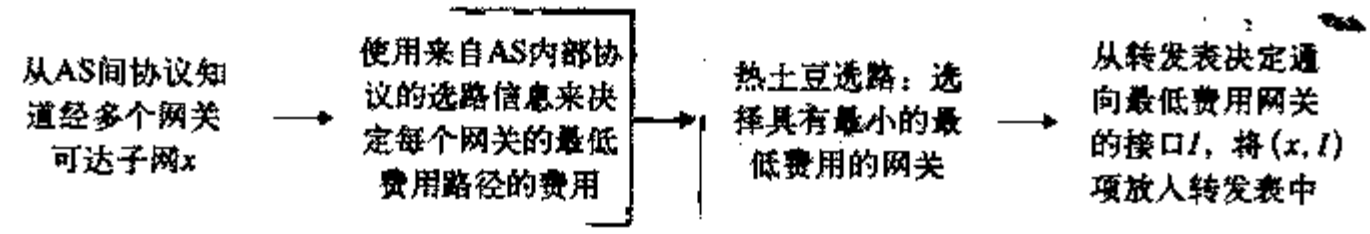


图4-33 在路由器的转发表中增加一个AS之外目的地的步骤

当一个AS从一个相邻AS处知道了一个目的地时，该AS能够向它的某些其他相邻AS通告该选路信息。例如，假定AS1从AS2处知道了经AS2可达子网*x*。AS1则能够告诉AS3经AS1可达*x*。用这种方式，如果AS3需要为一个目的地是*x*的分组进行选路的话，它将向AS1转发该分组，AS1将依次向AS2转发该分组。如我们将在讨论BGP时所见，一个AS在决定向其相邻AS通告哪些目的地时，具有相当大的灵活性。这是一个策略决定，通常更多地取决于经济问题而不是技术问题。

1.5节讲过，因特网是由层次互连的ISP组成的。那么，ISP和AS之间是什么样的关系呢？你也许认为在一个ISP中的路由器和互连它们的链路构成了单个AS。虽然通常是这种情况，但许多ISP将它们的网络划分为多个AS。例如，某些第一层ISP对它们的整个网络使用一个AS，其他ISP则将它们的ISP划分成数十个互连的AS。

总而言之，规模与管理职权的问题可通过定义自治系统来解决。在一个AS内部，所有路由器运行同样的自治系统内部选路协议。在各AS之间，AS运行相同的自治系统间选路协议。因为一个AS内部的路由器仅需要知道本AS内的路由器，所以规模问题可得到解决。因为一个组织可运行它选择的任何AS内部选路协议，所以管理职权问题可得到解决；然而，相连的每对AS需要运行相同的AS间选路协议以交换可达性信息。

在下一节中，我们将学习当今因特网中使用的两种AS内部选路协议（RIP与OSPF）和一种AS间选路协议（BGP）。对这些案例的研究将从不同侧面帮助我们学习层次选路内容。

4.6 因特网中的选路

学习了因特网编址与IP协议以后，我们现在将注意力转到因特网选路协议上来。选路协议的任务就是确定数据报在源与目的地之间采用的路径。我们将看到因特网的选路协议包含了许多我们在本章前面学过的原理。4.5.1节和4.5.2节中学过的链路状态算法与距离向量算法，

以及4.5.3节中讨论的自治系统（AS）概念，都在当今的因特网选路中发挥了重要的作用。

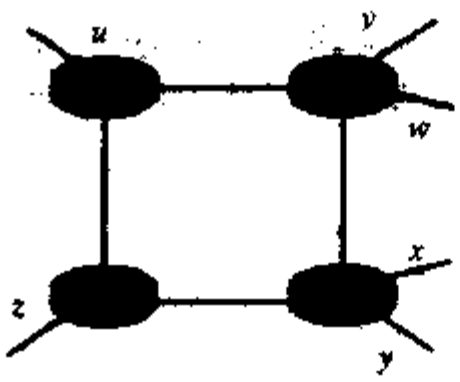
在4.5.3节中讲过，自治系统（AS）是一组处于相同的管理与技术控制下的路由器集合，AS之间都运行相同的选路协议。每个AS通常又包含多个子网（这里使用的术语“子网”具有4.4.2节中精确、编址的含义）。

4.6.1 因特网中自治系统内部选路：RIP

AS内部选路协议用于确定在一个自治系统（AS）内执行选路的方式。AS内部选路协议又称为**内部网关协议**（interior gateway protocol）。历史上有两个选路协议曾被广泛用于因特网上自治系统内的选路：**选路信息协议**（Routing Information Protocol, RIP）与**开放最短路径优先**（Open Shortest Path First, OSPF）。与OSPF密切相关的选路协议是IS-IS协议[RFC 1142, Perlman 1999]。我们首先讨论RIP，然后考虑OSPF。

RIP是最早的AS内部因特网选路协议之一，目前仍在广泛使用。它的产生与命名源于Xerox网络系统（XNS）体系结构。RIP的广泛应用主要是由于它包含在支持TCP/IP的1982年UNIX伯克利软件分布（Berkeley Software Distribution, BSD）版本中。在RFC 1058中定义了RIP版本1，在RFC 2453中定义了其向后兼容的版本2。

RIP是一种距离向量协议，其运行方式很像我们在4.5.2节中学习的理想化DV协议。在RFC 1058中定义的RIP版本使用跳数作为其费用测度，即每条链路的费用为1。在4.5.2节的DV算法中，为了简单起见，费用被定义在路由器对之间。在RIP（也在OSPF）中，费用实际上是从源路由器到目的子网。RIP使用术语跳，跳是沿着从源路由器到目的子网（包括目的子网）的最短路径所经过的子网数量。图4-34举例说明了具有6个叶子子网的一个AS。该图中的表指出了从源A到每个叶子子网的跳数。



目的地	跳数
u	1
v	2
w	2
x	3
y	3
z	2

图4-34 从源路由器A到各个子网的跳数

一条路径的最大费用被限制为15，因此RIP被限制用在网络直径不超过15跳的自治系统内。前面讲过，在DV协议中，相邻路由器之间相互交换距离向量。对任何一台路由器的距离向量是从这台路由器到该AS中子网的最短路径距离的当前估计值。在RIP中，选路更新信息在邻居之间通过使用一种**RIP响应报文**（RIP response message）交换，大约30秒相互交换一次。由一台路由器或主机发出的响应报文包含了一个由多达25个AS内的目的子网列表，还有发送方到其中每个子网的距离。响应报文又被称作**RIP通告**（RIP advertisement）。

我们考察有关RIP通告是如何工作的一个简单例子。考虑图4-35中显示的一个AS的一部分。在该图中，连接路由器的线表示子网。仅有选中的路由器（A、B、C、D）与子网（*w*、*x*、*y*、*z*）被标记了。虚线表示该AS还在继续，因此该自治系统有比图示更多的路由器和链路。

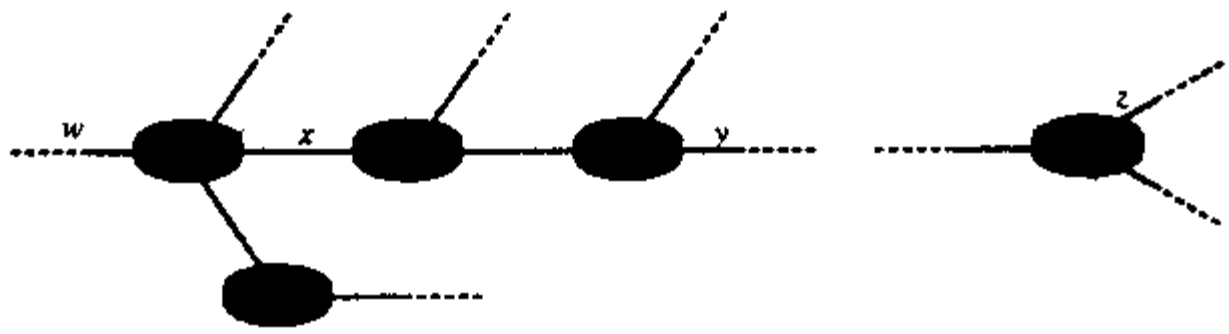


图4-35 一个自治系统的一部分

每台路由器维护一张称为**选路表**（routing table）的RIP表。一台路由器的选路表包括该路由器的距离向量和该路由器的转发表。图4-36显示了路由器D的转发表。注意，该转发表有3列。第一列为目的子网，第二列指出了沿着到目的网络的最短路径上的下一台路由器的标识，第三列指出了沿着最短路径到达目的子网的跳数（即需要穿越的子网数，包括目的子网）。对于该例而言，该表指出了从路由器D到目的子网*w*发送一个数据报，该数据报应首先被转发到相邻路由器A；该表还指出沿着最短路径到目的子网*w*为两跳距离。类似地，该表指出了子网*z*经由路由器B为7跳距离。虽然RIP版本2允许使用类似于我们在4.4节中学习的路由聚合技术，以聚合子网项，但原则上AS内的每个子网应在转发表中占一行。图4-36中的表以及后续的表都只是部分完成了。

目的子网	下一台路由器	到目的地的跳数
w	A	2
y	B	2
z	B	7
x	—	1
…	…	…

图4-36 收到来自路由器A的通告之前路由器D中的转发表

现在假定30秒以后，路由器D收到来自路由器A的如图4-37所示的通告。注意到该通告正是来自路由器A的选路表信息！该信息特别指明了子网*z*离路由器A仅有4跳距离。一旦收到该通告，路由器D将该通告（图4-37）与旧选路表（图4-36）合并。特别是路由器D知道了通过路由器A到子网*z*的路径比通过路由器B的路径短。因此，路由器D更新其转发表，记下该更短的最短路径，如图4-38所示。你也许会问，到子网*z*的最短路径怎么会变得更短呢？很可能是分布式的距离向量算法还处在收敛过程中（参见4.5.2节），或者可能是新的链路和/或路由器加入了该AS，因此改变了AS中的最短路径。

目的子网	下一台路由器	到目的地的跳数
z	C	4
w	—	1
x	—	1
…	…	…

图4-37 来自路由器A的通告

目的子网	下一台路由器	到目的地的跳数
w	A	2
y	B	2
z	A	5
…	…	…

图4-38 收到路由器A的通告后，路由器D中的转发表

我们下面考虑RIP实现方面的几个问题。前面讲过RIP路由器大约每30秒便相互交换通告。如果一台路由器一旦超过180秒没有监听到其邻居，则该邻居不再被认为是可达的，即要么其邻居死机了，要么连接的链路中断了。当发生这种情况时，RIP修改本地选路表，然后通过向

相邻路由器（那些仍然可达的路由器）发送通告来传播该信息。路由器也可通过使用RIP请求报文，请求其邻居到指定目的地的费用。路由器在UDP上使用端口520相互发送RIP请求与响应报文。该UDP报文段在标准IP数据报中承载在路由器之间。RIP使用一个位于网络层协议（IP）之上的运输层协议（UDP）来实现网络层功能（一种选路算法），这个事实看起来似乎相当令人费解。（的确如此！）若再深入一些观察RIP的实现原理将能更明白这一点。

图4-39概略地说明了RIP在一个UNIX系统中通常是如何实现的，例如一台用作路由器的UNIX工作站。一个称为routed（发音为“route dee”）的进程执行RIP，即维护选路信息并与相邻路由器中的routed进程交换报文。因为RIP是被当作一个应用层进程（虽然它是一个能操作UNIX内核中的转发表的特殊进程）来实现的，所以它能在一个标准套接字上发送和接收报文，并且使用一个标准的运输层协议。如显示的那样，RIP是一个运行在UDP上的应用层协议（参见第2章）。

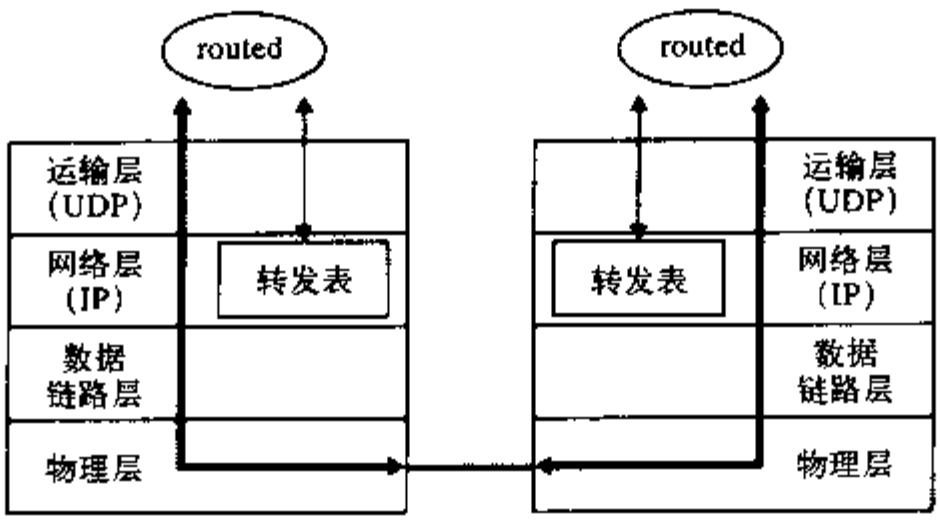


图4-39 作为routed守护程序的RIP实现

4.6.2 因特网中AS内部选路：OSPF

就像RIP一样，OSPF选路也被广泛用于因特网中的AS内部选路。OSPF和它的关系密切的兄弟IS-IS，通常被设置在较顶层的ISP中，而RIP却被设置在较低层ISP和企业网中。OSPF中的开放（Open）是指选路协议规约是公众可用的（与之相反的是，如Cisco的EIGRP协议就不是开放的）。OSPF的最新版本是版本2，由RFC 2328这个公用文档所定义。

OSPF被认为是RIP的后继者，并且有许多先进特性。然而，OSPF的核心就是一个使用洪泛链路状态信息的链路状态协议和一个Dijkstra最低费用路径算法。使用OSPF，一台路由器构建了一幅关于整个自治系统的完整拓扑图（即一幅图）。于是，路由器在本地运行Dijkstra最短路径算法，以确定一个以自身为根节点的到所有子网的最短路径树。各条链路费用是由网络管理员配置的（参见“实践原则：设置OSPF权值”）。管理员也许会选择将所有链路费用设为1，因而实现了最少跳数选路；或者可能会选择将链路权值按与链路容量成反比来设置，从而不鼓励流量使用低带宽链路。OSPF不强制使用如何设置链路权值的策略（那是网络管理员的任务），但提供了一种机制（协议），为给定链路权值集合确定最低费用路径选路。

使用OSPF时，路由器向自治系统内所有其他路由器广播选路信息，而不仅仅是向其相邻路由器广播。每当一条链路的状态发生变化（如费用的变化或连接/中断状态的变化）时，路由器就会广播链路状态信息。即使链路状态未发生变化，它也要周期性（至少每隔30分钟一次）地广播链路状态。RFC 2328中有这样的说明：“这种周期性的链路状态更新通告增加了链路状态算法的健壮性。”OSPF通告包含在OSPF报文中，该OSPF报文直接承载在IP分组中，对

于OSPF其上层协议值为89。因此OSPF协议必须自己实现可靠报文传输、链路状态广播等功能。OSPF协议还要检查链路正在运行（通过向相连的邻居发送HELLO报文），并允许OSPF路由器获得相邻路由器的网络范围链路状态的数据库。

OSPF的优点包括下列几方面：

- 安全。OSPF路由器之间的交换（如链路状态更新）都是经过鉴别的。这意味着仅有受信任的路由器能参与一个AS内的OSPF协议，因此可阻止恶意入侵者（或在利用新学的知识到处试探的网络专业学生）将不正确的信息注入路由器表内。在默认状态下，路由器间的OSPF报文未被鉴别并能被伪造。可以配置两类鉴别——简单的和MD5的（参见第8章有关MD5和鉴别的一般性讨论）。使用简单的鉴别，每台路由器配置相同的口令。当一台路由器发送一个OSPF分组时，它以明文方式包括了口令。显然，简单鉴别并不安全。MD5鉴别基于配置在所有路由器上的共享安全密钥。对每个发送的OSPF分组，路由器对附加秘密密钥的分组内容计算MD5散列值。（参见第7章中关于报文鉴别码的讨论。）然后它将所得的散列值包括在该OSPF分组中。接收路由器使用预配置的密钥，计算出该分组的MD5散列值，并与该分组携带的散列值进行比较，从而验证该分组的真实性。在MD5鉴别中也使用了序号对重放攻击进行保护。
- 多条相同费用的路径。当到达某目的地的多条路径具有相同的费用时，OSPF允许使用多条路径（这就是说，当存在多条相等费用的路径时，无需仅选择单一的路径来承载所有的流量）。
- 对单播选路与多播选路的综合支持。多播OSPF（MOSPF）[RFC 1584]是对OSPF的简单扩展，以便提供多播选路（这是我们将在4.7.2节更深入学习的主题）。MOSPF使用现存的OSPF链路数据库，并为现有的OSPF链路状态广播机制增加了一种新型的链路状态通告。
- 支持在单个选路域内的层次结构。也许OSPF的最重要优点是具有按层次结构构造一个自治系统的能力。在4.5.3节中我们已看到了层次选路结构的许多优点。在本节剩余部分，我们将学习OSPF层次选路的实现。

一个OSPF自治系统可以配置成多个区域。每个区域都运行自己的OSPF链路状态选路算法，一个区域内的每台路由器都向该区域内的所有其他路由器广播其链路状态。因此，一个区域的内部细节对于该区域外的所有路由器来说都是不可见的。区域内选路仅涉及同一区域内的那些路由器。

在一个区域内，一台或多台**区域边界路由器**（area border router）负责为发送到该区域以外的分组选路。准确地说，AS内的一个OSPF区域配置成**主干**（backbone）区域。主干区域的主要作用是为AS内其他区域之间的流量选路。该主干总是包含了AS内的所有区域边界路由器，还可能包含了一些无边界路由器。AS内的区域间选路要求分组首先路由到一个区域边界路由器（区域内选路），再通过主干路由到位于目的区域的区域边界路由器，然后再路由到最终目的地。

图4-40中显示了一幅层次结构的OSPF网络图。我们可以在该图中辨认出4种类型的OSPF路由器。

- 内部路由器。这些路由器位于非主干区域且只执行AS内部选路。
- 区域边界路由器。这些路由器同时属于区域与主干两个区域。
- 主干路由器（非边界路由器）。这些路由器执行主干中的选路，但其自身不是区域边界路由器。在一个非主干区域内，内部路由器通过该区域中的主干路由器，从信息（基本

上是链路状态通告，但通告的是到另一个区域的路由费用，而不是链路费用）广播中知道存在通向其他区域的路由。

- 边界路由器。边界路由器与属于其他自治系统的路由器交换选路信息。例如，这台路由器也许会使用BGP执行AS间选路。其他路由器正是通过这样的边界路由器才知道通向外部网络的路径。

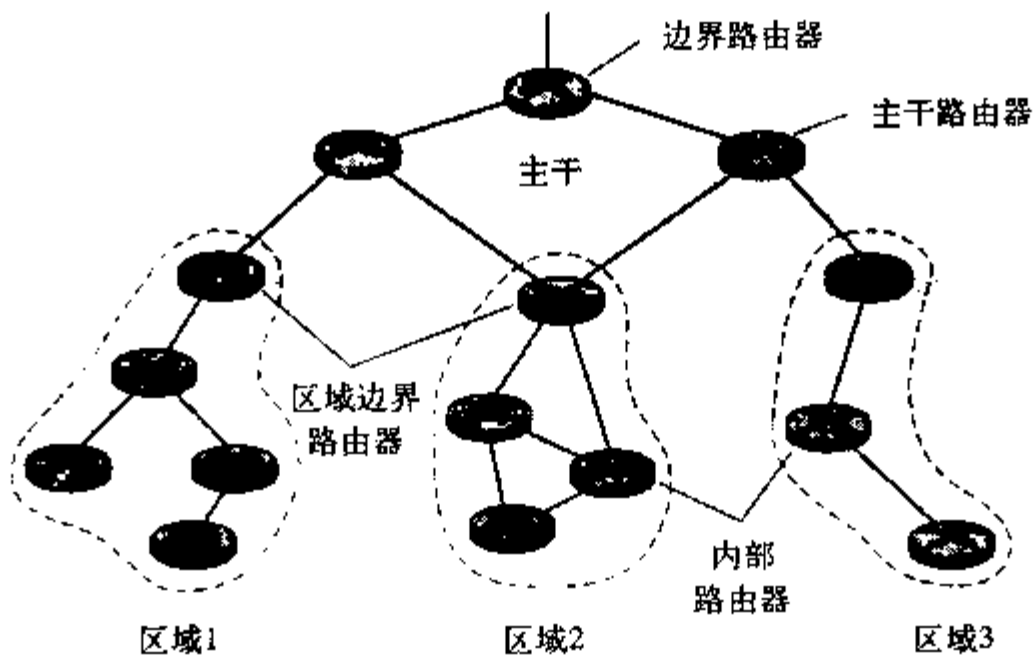


图4-40 具有4个区域的层次结构OSPF自治系统

OSPF是一个相当复杂的协议，而我们这里只是进行了十分简要的学习，[Huitema 1998; Moy 1998; RFC 2328] 提供了更多的细节。

实 践 原 则

设置OSPF链路权值

我们有关链路状态选路的讨论隐含地假设了下列事实：设置了链路权重，运行了诸如OSPF这样的选路算法，根据LS算法计算所得的选路表疏导流量。就原因和结果而言，给定链路权重（即它们先发生），从而得到（经Dijkstra算法）最小化总体费用的选路路径。从这个角度看，链路权重反映了使用一条链路的费用（例如，如果链路权重与容量成反比，则使用高容量链路将具有较小的权重并因此从选路的角度更有吸引力），并且使用Dijkstra算法使得总费用为最小。

在实践中，链路权重和选路路径之间的因果关系也许是相反的，网络操作员配置链路权重，以获取某些流量工程目标的选路路径[Fortz 2000, Fortz 2002]。例如，假设某网络操作员具有在每个入口点进入且发向每个出口点的进入该网络的流量估计。该操作员则可能要设置特定入口到出口的流选路，以最小化经所有网络链路的最大利用率。但使用如OSPF这样的选路算法，该操作员用于调节通过网络的选路流的主要“调节器”是链路的权重。因此，为了取得最小化最大链路利用率这个目标，该操作员必须找出取得该目标的链路权重集合。这是一种相反的因果关系，即已知所希望的流选路，必须找到OSPF链路权重，使该OSPF选路算法导致这种希望的流选路。

4.6.3 自治系统间的选路：BGP

我们刚才学习了ISP如何使用RIP和OSPF来决定位于相同AS内部的源和目的对之间的优化

路径。我们现在研究一下，跨越多个AS的源和目的对之间是如何确定路径的。由RFC 4271（也可参见［RFC 1772; RFC1773］）定义的**边界网关协议**（Broder Gateway Protocol, BGP）版本4是当今因特网中域间选路协议事实上的标准。它通常被称为BGP4或简称为BGP。作为一个自治系统间选路协议（参见4.5.3节），BGP为每个AS提供一种手段，以处理：

1）从相邻AS处获得子网可达性信息。

2）向该AS内部的所有路由器传播这些可达性信息。

3）基于可达性信息和AS策略，决定到达子网的“好”路由。

更为重要的是，BGP允许每个子网向因特网的其余部分通告它的存在。一个子网高声宣布：“我存在，我在这里”，并且BGP确保因特网中的所有AS知道该子网以及如何到达那里。如果没有BGP的话，每个子网将是隔离的，它们孤独并且不为因特网其余部分所知。

1. BGP基础

BGP极其复杂，许多专著致力于研究该主题，并且许多问题仍没有得到很好的理解[Yannuzzi 2005]。此外，作为较高层ISP的设计者或管理员，即使在阅读了这些专著和RFC以后，如果不数月（甚至数年）实际操作BGP的话，你也可能发现难以全面掌握BGP。无论如何，因为BGP是因特网中绝对重要的协议，即从本质上讲，正是这个协议将所有的东西黏合在一起了，因此我们至少要初步了解该协议的工作原理。我们从描述BGP在简单的例子网络环境下如何工作开始，我们曾在图4-32中学习过该例子网络。在下面的描述中，我们将讨论基于4.5.3节中的层次选路上，我们希望你能复习一下该材料。

在BGP中，路由器对使用179端口的半永久TCP连接来交换选路信息。对于图4-32中的网络而言，该半永久TCP显示在图4-41中。对于每条直接连接位于两个不同的AS中的路由器链路而言，通常有一条这样的BGP TCP连接。因此，在图4-41中，在网关路由器3a和1c之间有一条TCP连接，在网关路由器1b和2a之间有另一条TCP连接。在一个AS中，路由器之间还有许多半永久BGP TCP连接。特别是，图4-41显示了一个AS内部的每对路由器之间的一条TCP连接的通常配置，在每个AS内部产生了网状的TCP连接。对于每条TCP连接，位于该连接端点的两台路由器被称为**BGP对等方**（BGP peer），沿着该连接发送所有BGP报文的TCP连接被称为**BGP会话**（BGP session）。此外，跨越两个AS的BGP会话被称为**外部BGP（eBGP）会话**（external BGP session），同一个AS中的两台路由器之间的BGP会话被称为**内部BGP（iBGP）会话**（internal BGP session）。在图4-41中，eBGP会话显示为长虚线，iBGP会话显示为短虚线。注意，图4-41中的BGP会话线并不总是与图4-32中的物理链路对应。

BGP使得每个AS知道经过其相邻AS，哪些目的地是可达的。在BGP中，目的地不是主机而是CDIR化的**前缀**（perfix），每个前缀表示一个子网或一个子网的集合。因此，假定有4个子网与AS2相连：138.16.64/24、138.16.65/24、138.16.66/24和138.16.67/24，则AS2能为这4个子网聚合这些前缀，并使用BGP来向AS1通告单一前缀138.16.64/22。再举一个例子，假定这4个子网中的前3个在AS2中，第四个子网138.16.67/24位于AS3中，则如4.4.2节中的“实践原则”所述，因为路由器使用最长前缀匹配来转发数据报，所以AS3能向AS1通告更特定的前缀138.16.67/24，而AS2仍能向AS1通告聚合的前缀138.16.64/22。

现在我们研究一下BGP是怎样经显示在图4-41中的BGP会话来发布前缀可达性信息的。正如你所预想的那样，在网关路由器3a和1c之间使用eBGP会话，AS3向AS1发送一个自AS3可达的前缀列表；AS1向AS3发送经AS1可达的前缀列表。类似地，AS1和AS2通过它们的网关路由器1b和2a交换它们的可达性信息。同样如你所预想的那样，任何AS中的网关路由器接收到

eBGP学习到的前缀后，该网关路由器使用它的iBGP会话来向该AS中的其他路由器发布这些前缀。因此，不仅AS1中的所有路由器将得知AS3的前缀，而且网关路由器1b也将得知AS3的前缀。AS1中的网关路由器1b因此能向AS2重新通告AS3的前缀。当一台路由器（网关或不是网关）得知一个新前缀时，它为该前缀在其转发表中创建一个项，如4.5.3节所述的那样。

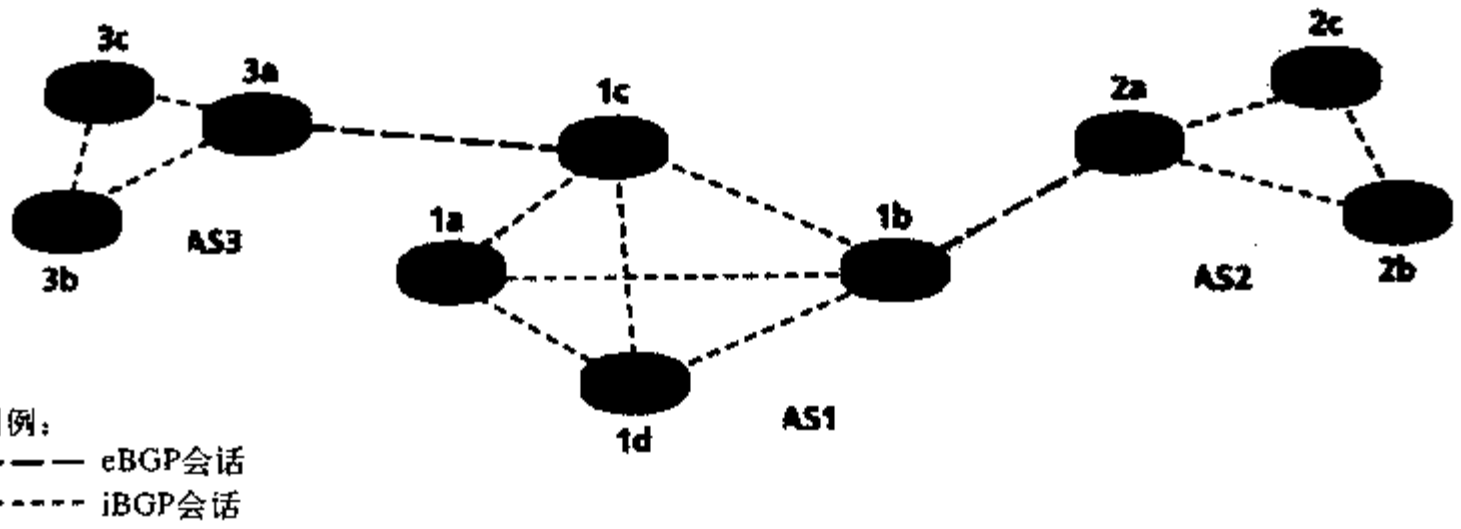


图4-41 eBGP和iBGP会话

2. 路径属性和BGP路由

在对BGP有了一些基本了解后，我们更深入地学习它。（在此过程中将同时增加一些不怎么重要的技术细节！）在BGP中，一个自治系统有其全局唯一的**自治系统号**（Autonomous System Number, ASN）[RFC 1930]。（从技术上讲，并非每个AS都具有一个ASN。特殊情况下，有一种所谓桩（stub）AS通常就没有ASN，这种桩AS仅承载源地址或目的地址为本AS的流量。我们将在下面讨论中忽略这种特殊情况，以便能从全局看问题。）就像IP地址一样，AS号由ICANN地区注册机构分配［ICANN 2007］。

当一台路由器通过BGP会话通告一个前缀时，随着前缀包括一些**BGP属性**（BGP attribute）。用BGP术语来说，带有属性的前缀被称为一条**路由**（route）。因此，BGP对等方彼此通告路由。两个较为重要的属性是AS-PATH和NEXT-HOP。

- AS-PATH。该属性包含了前缀的通告已经通过的那些AS。当一个前缀传送到一个AS时，该AS将它的ASN增加到AS-PATH属性中。例如，考虑图4-41，假定前缀138.16.64/24首先是由AS2向AS1通告的；如果AS1接下来将该前缀向AS3通告，则该AS-PATH将是AS2 AS1。路由器使用该AS-PATH 属性来检测和防止循环通告；特别是，如果一台路由器看到它的AS包含在该路径列表中，它将拒绝该通告。我们将很快讨论到，路由器也使用AS-PATH属性在多条路径中选择相同的前缀。
- 在AS间选路协议和AS内部选路协议之间提供重要链路后，NEXT-HOP属性具有一种微妙而重要的用途。NEXT-HOP是一个开始某AS-PATH的路由器接口。为了深入理解该属性，我们再次观察图4-41。考虑当AS3中的网关路由器3a向AS1中使用eBGP的网关路由器1c通告一条路由时发生的情况。该路由包括通告前缀（我们称该前缀为x）和对该前缀的一个AS-PATH。该通告也包括NEXT-HOP，这是路由器3c通向1c的接口的IP地址。（前面讲过，一台路由器具有多个IP地址，每个接口有一个地址）。现在考虑当路由器1d得知了来自iBGP的这条路由后发生的情况。在得知到x的这条路由后，路由器1d可能要沿着该路由向x转发分组，即路由器1d可能要在其转发表中包括表项(x, l)，其中l是从1d朝着网关路由器1c开始最低费用路径的接口。为了确定l，1d在NEXT-HOP属性中提供

了到它的AS内部选路模块的IP地址。注意，AS内部选路算法已经确定了到AS1中所有与该路由器相连的子网（包括到1c和3a之间的链路的子网）的最低费用路径。从1d到1c-3a子网的这条最低费用路径出发，1d决定它的路由器接口*l*开始于这条路径，进而将表项(*x*, *l*)加进其转发表中。这就实现了这项功能！总而言之，由路由器使用的AS-PATH属性适当地配置了它们的转发表。

图4-42举例说明了另一种情况，其中需要AS-PATH。在该图中，AS1和AS2由两条对等链路连接。AS1中的路由器能够知道到相同前缀*x*的两条不同的路由。这两条路由能够具有相同的到*x*的AS-PATH，但能够具有不同的NEXT-HOP值以对应于不同的对等链路。使用AS-PATH值和AS内部选路算法，该路由器能够确定到每条对等链路的路径的费用，然后应用热土豆选路（参见4.5.3节）来确定适当的接口。

BGP也包括允许路由器对路由分配偏好测度的属性，以及指示前缀如何插入位于起始AS的BGP的属性。对于路由属性的全面讨论，参见[Griffin 2002; Stewart 1999; Halabi 2000; Feamster 2004; RFC 4271]。

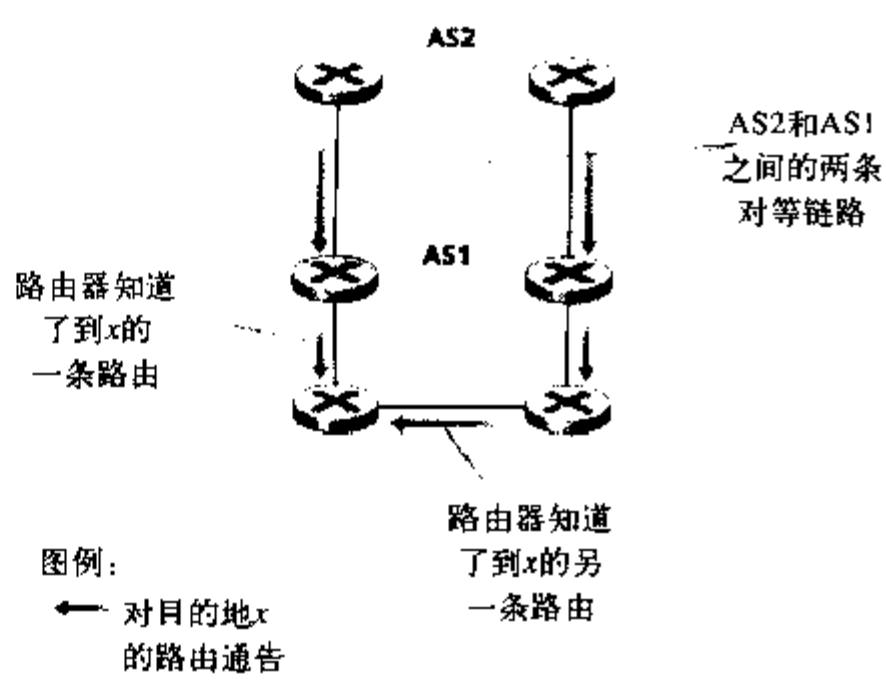


图4-42 通告中的NEXT-HOP属性用于确定使用哪条对等链路

当一台网关路由器接收到一个路由器通告时，它使用其**输入策略**（import policy）来决定是否接收或过滤该路由，是否设置某种属性（如路由器偏好测度）。输入策略可能过滤掉一条路由，因为该AS可能不想通过该路由的AS-PATH中的某个AS发送流量。网关路由器也可能过滤掉一条路由，因为它已经知道了一条到相同前缀的偏好路由。

3. BGP路由选择

如本节前面所述，BGP使用eBGP和iBGP向AS中的所有路由器发布路由。根据这种发布，路由器可能知道到达任何一条前缀的多条路由，在这种情况下，路由器必须在可能的路由中选择一条。输入进该路由选择进程的是被该路由器知道并接受的所有路由的集合。如果到相同前缀存在两条或多条路由，则BGP顺序地调用下列消除规则直到留下一条路由：

- 路由被指派一个本地偏好值作为它们的属性之一。一条路由的本地偏好可能被该路由器设置或可能由相同AS中的另一台路由器学习到。一条策略决定是由该AS的网络管理员设定。（我们随后将更为详细地讨论BGP策略问题。）具有最高本地偏好值的路由将被选择。
- 在余下的路由（所有都具有相同的本地偏好值）中，将选择具有最短AS-PATH的路由。如果该规则是路由选择的唯一规则，则BGP将使用距离向量算法来决定路径，其中距离测度使用AS跳数而不是路由器的跳数。
- 在余下的路由（所有都具有相同的本地偏好和相同的AS-PATH长度）中，将选择最靠近NEXT-HOP路由器的路由。这里，最靠近是指最低费用路径的费用最低的路由器，由AS内部算法来决定。该进程经常被称为热土豆选路，如4.5.3节所述。
- 如果仍余下多条路由，该路由器使用BGP标识符来选择路由，参见[Stewart 1999]。

该消除规则甚至比上面描述的更为复杂。为了避免BGP灾难，最好学习少量的BGP选择

规则方面的知识。

4. 选路策略

我们用一个简单的例子说明BGP选路策略的某些基本概念。图4-43显示了6个互连的自治系统：A、B、C、W、X和Y。重要的是注意到A、B、C、W、X和Y是AS，而不是路由器。假设自治系统W、X和Y是桩网络，A、B、C是主干提供商网络。还假设A、B和C都是彼此对等的，并向它们的客户网络提供全部的BGP信息。所有进入**桩网络**（stub network）的流量必定是以该网络为目的地的，所有离开桩网络的流量必定是源于该网络的。W和Y显然是桩网络。X是一个**多宿桩网络**（multi-homed stub network），因为它是经由两个不同的提供商连到其余网络的（这种方法在实践中用得越来越普遍）。然而，就像W和Y一样，X自身必定是进入/离开X的所有流量的源/目的地。但这种桩网络的行为是如何实现和加强的呢？X如何防止转发B与C之间的流量呢？这很容易通过控制BGP路由的通告方式来实现。特别是，X如果通告（向其邻居B和C）它没有通向任何其他目的地（除自身以外）的路径，那么它将起到一个桩网络的作用。这就是说，即使X可能知道一条路径能到达网络Y，比如说XCY，它也不将该条路径通告给B。由于B不知道X有一条路径到Y，因此B绝不会经由X转发目的为Y（或C）的流量。这个简单的例子说明了如何使用一条选择的路由通告策略来实现客户/提供商关系。

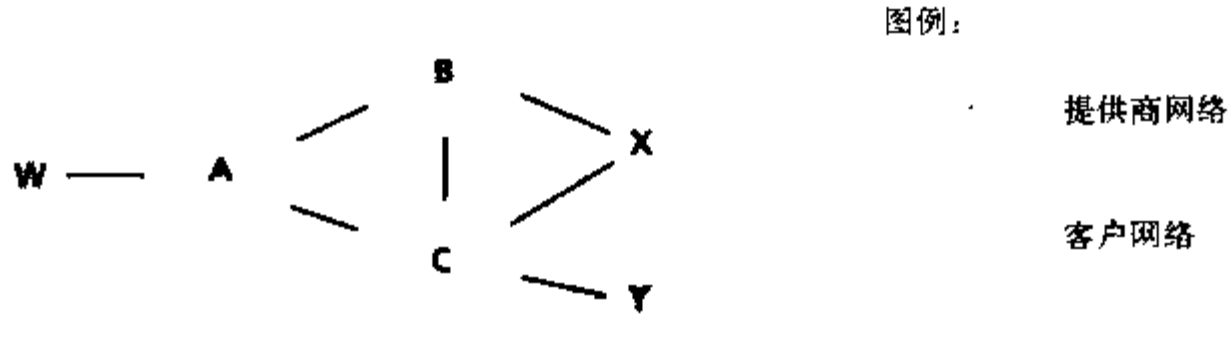


图4-43 一个简单的BGP应用情况

我们接下来关注提供商网络，比如AS B。假定B已经知道了（从A处）A有一条到W的路径AW。B因此能将路由BAW设置到其路由信息库中。显然，B也想向它的客户X通告路径BAW，以便X知道它能够通过B选路到W。但是，B应该将路径BAW通告给C吗？如果它这样做，则C可以经由CBAW将流量引导到W。如果A、B和C都是主干提供商，那么B也许正好觉得它不应该承担A与C之间传送流量的负担（和费用）。B有理由认为，确保C能经过A和C之间的直接连接引导A客户的流量，是A和C的工作（和费用）。目前还没有强制主干ISP之间如何选路的官方标准。然而，商业运行的众多ISP都遵从一个经验法则：任何穿越某ISP主干网的流量必须是或其源或目的（或两者）位于该ISP的某个客户网络中，否则这些流量将会免费搭乘该ISP的网络。独立的对等协定（用于解决前面提到的问题）通常都是在ISP双方之间进行协商，而且经常对外保密；[Huston 1999a]提供了关于对等协定的有趣讨论。选路策略如何反映ISP之间的商业关系的详细描述参见[Gao 2001]。从ISP立场出发，有关BGP选路策略的近期讨论参见[Caesar 2005]。

如上所述，BGP是一个对于公共因特网的AS间选路的事实标准。为了观察从第一层ISP路由器提取出来的各种BGP选路表（大量）的内容，参见http://www.routeviews.org。BGP选路表通常包含数以万计的前缀和对应的属性。有关BGP选路表的规模和特征的统计数据参见[Huston 2001; Meng 2005]。

我们完成了对BGP的简要介绍。理解BGP是很重要的，因为它在因特网中起着核心作用。

我们鼓励你阅读参考文献[Griffin 2002; Stewart 1999; Labovitz 1997; Halabi 2000; Huitema 1998; Gao 2001; Feamster 2004; Caesar 2005]，以学习更多有关BGP的知识。

实践原则

为什么会有不同的AS间选路协议和AS内部选路协议

学习了目前因特网中所采用的特定的AS间和AS内部选路协议后，我们来思考并稍加总结，也许我们首先会问有关这些协议的最根本性问题是（希望你一直在主动思考该问题，并且不致因技术细节而忽略全局）：为什么要使用不同的AS间选路协议和AS内部选路协议？

对该问题的解答涉及AS内部选路目标与AS间选路目标之间的差别本质：

- 策略。在AS之间，策略问题是至关重要的。一个给定AS产生的流量不能穿过另一个特定的AS，这也许很重要。类似地，一个给定AS也许想控制它在其他AS之间传输什么样的流量。我们已看到，BGP承载了路径属性，并提供受控制的选路信息分布，以便能做出这种基于策略的选路决策。在一个AS内部，一切都是以相同的管理控制名义进行的，因此策略问题在AS内部选择路由时起着微不足道的作用。
- 规模。一个选路算法及其数据结构在处理大量网络的选路或大量网络之间的选路时的适应能力，是AS间选路的一个关键问题。在一个AS内部，可缩扩性是第二个要关心的问题。第一个要关心的问题是，如果单个管理域变得太大，总是可以将其分成两个AS，并在这两个新的AS之间执行AS间选路算法。前面讲过，OSPF通过将一个AS分成区域而建立这样的层次结构。
- 性能。由于AS间选路是面向策略的，因此所用路由的质量（如性能）通常是次要关心的问题（即一条更长或费用更高但能满足某些策略条件的路由也许被采用了，而更短但不满足那些条件的路由却不会被采用）。事实上，我们看到了在AS之间甚至没有与路由相关的费用（除了AS跳计数外）概念。然而在一个AS内部，这种对策略的关心就不重要了，可以使选路要考虑的问题更多地集中在一条路由实现的性能级别上。

4.7 广播和多播选路

到目前为止，本章关注的是支持单播（即点对点）通信的选路协议，单个源节点基于这种协议向单个目的节点发送分组。在本节中，我们将注意力转向广播和多播选路协议。在**广播选路**（broadcast routing）中，网络层提供了从一个源节点到网络中的所有其他节点交付分组的服务；**多播选路**（multicast routing）使单个源节点能够向其他网络节点的一个子集发送分组的拷贝。在4.7.1节中，我们将考虑广播选路算法及其在选路协议中的具体实现。在4.7.2节中，我们将研究多播选路。

4.7.1 广播选路算法

也许完成广播通信的最直接方式是由发送节点向每个目的地分别发送分组的拷贝，如图4-44a所示。给定N个目的节点，源节点只是产生该分组的N份拷贝，对不同目的地的每个拷贝进行编址，并用单播选路向N个目的地传输这N份拷贝。这种用N次单播（N-way-unicast）实

现广播的方法是很简单的，无需新的网络层选路协议、分组复制或转发功能。然而，这种方法有几个缺点。第一个缺点是它的低效率。如果源节点经过单一链路与该网络的其余部分相连的话，该（相同）分组的*N*份独立的拷贝都将经过该段链路传输。显然更为有效的方式是，经第一跳仅发送分组的单个拷贝，然后让第一跳后面其他端的节点产生并转发任何附加的拷贝。这就是说，让网络节点本身（而不只是源节点）产生分组的冗余拷贝将更加有效。例如，在图4-44b中，仅有一个分组的拷贝通过R1-R2链路。该分组在R2被复制，经链路R2-R3和R2-R4发送单个拷贝。

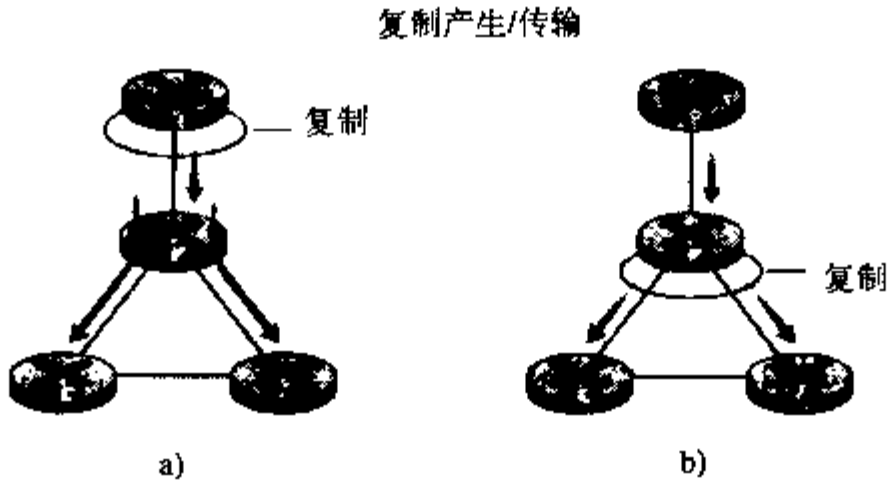


图4-44 源复制与网络中复制的比较

*N*次单播的其他缺点也许更微妙，但不太重要。*N*次单播的一个隐含假设是广播的接收方及其地址均为发送方所知。但是怎样得到这些信息呢？最可能的是，需要有另外的协议机制（如广播成员或目的地注册协议）。这将增加更多的开销，并且重要的是为最初看起来相当简单的协议增加了复杂性。*N*次单播的最后一个缺点与使用广播的目的有关。在4.5节中，我们知道了链路状态选路协议使用广播来分发用于计算单播路由的链路状态信息。显然，在使用广播来产生和更新单播路由的情况下，基于单播选路基础设施来实现广播将（最多）是不明智的。

指出了*N*次单播实现广播的几个缺点后，网络节点本身在分组复制、分组转发和广播路由计算中起积极作用的方法，将显然是令人感兴趣的。下面我们将研究这样的几种方法，并再次采用4.5节中引入的图标记法。我们再次将网络建模为图$G=(N, E)$，其中*N*是节点的集合，*E*是边的集合，而每条边是来自*N*的一对节点。当不致混淆时，使用*N*表示节点的集合以及表示该集合的基数(|*N*|)或长度，虽然这种标记法有点不够严谨。

1. 无控制洪泛

实现广播的最显而易见的技术是使用**洪泛**（flooding）方法，该方法要求源节点向它的所有邻居发送该分组的拷贝。当某节点接收了一个广播分组时，它复制该分组并向它的所有邻居（除了从其接收到该分组的那个邻居）转发之。显然，如果图是相连的，这种方案最终会将广播分组的拷贝交付给该图中的所有节点。虽然这种方案简单而优雅，但它具有致命缺点（在你继续阅读前，看看你是否能猜出这个致命缺点）：如果该图具有圈，则每个广播分组的一个或多个分组拷贝将无休止地循环。例如，在图4-44中，R2将洪泛到R3，R3将洪泛到R4，R4将洪泛到R2，R2将（再次）洪泛到R3，等等。这种简单情况导致两个广播分组无休止地循环，一个朝顺时针方向，一个朝逆时针方向。但还可能存在一种更不幸的致命缺陷：当一个节点与多于两个其他节点连接时，它将产生并转发广播分组的多个拷贝，这些拷贝中的每个又产生多个它们自身的拷贝（在其他节点有多于两个邻居），等等。这种**广播风暴**（broadcast

storm）导致无休止的广播分组的复制，将最终导致在该网络中生成大量的广播分组，使得网络变得毫无用处。（参见本章课后作业中分析这种广播风暴增长速率的问题。）

2. 受控洪泛

避免广播风暴的关键是每个节点明智地选择何时洪泛分组，何时不洪泛分组（例如，如果它已经接收并洪泛了某分组的较早拷贝）。在实践中，这可以通过几种方式来实现。

在序号控制洪泛（sequence-number-controlled flooding）中，源节点将其地址（或其他的唯一标识符）以及**广播序号**（broadcast sequence number）放入广播分组，再向它的所有邻居发送该分组。每个节点维护它已经收到的、复制的和转发的源地址和每个广播分组的序号列表。当一个节点接收到一个广播分组时，它首先检查该分组是否在该列表中。如果在，丢弃该分组；如果不在，复制该分组并向该节点的所有邻居（除了从其接收到该分组的那个节点）转发。第2章中讨论的Gnutella协议使用序号控制洪泛在它的覆盖网络中广播查询。（在Gnutella中，报文复制和转发在应用层执行，而不是在网络层执行。）

受控洪泛的第二种方法称为**反向路径转发**（Reverse Path Forwarding, RPF）[Dalal 1978]，有时也称为反向路径广播（RPB）。RPF的基本思想简单且优雅。当一台路由器接收到具有给定源地址的广播分组时，仅当该分组到达的链路正好是位于它自己到其源的最短单播路径上，它才向其所有出链路（除了它接收分组的那个）传输分组。否则，该路由器只是丢弃入分组而不向任何它的出链路转发分组。因为该路由器知道它在这样一条链路（该链路位于它自到发送方的最短路径上）上或者将接收或者已经接收了该分组的一个拷贝，所以可以将这样的一个分组丢弃。（你也许要弄清楚，事实上，如果发生了这样的情况，环路和广播风暴将不会出现。）注意，RPF不使用单播选路以实际将分组交付给目的地，它也不要求路由器知道从它自己到源的完整最短路径。RPF仅需要知道它到发送方的单播最短路径上的下一个邻居，它仅使用这个邻居的身份来决定是否洪泛一个接收到的广播分组。

图4-45举例说明了RPF。假定用粗线画的链路表示了从接收方到源（A）的最低费用路径。节点A最初广播一个源为A的分组到节点C和B。节点B将向节点C和D转发它从节点A接收到的源为A的分组（因为A位于到A的最低费用路径上）。B将忽略（丢弃而不转发）从任何其他节点（如从路由器C或D）接收的源为A的分组。我们现在考虑节点C，C将直接从A以及从B接收源为A的分组。因为B不在C自己到A的最短路径上，所以C将忽略来自B的任何源为A的分组。另一方面，当C接收到直接来自A的源为A的分组时，它将向节点B、E和F转发该分组。

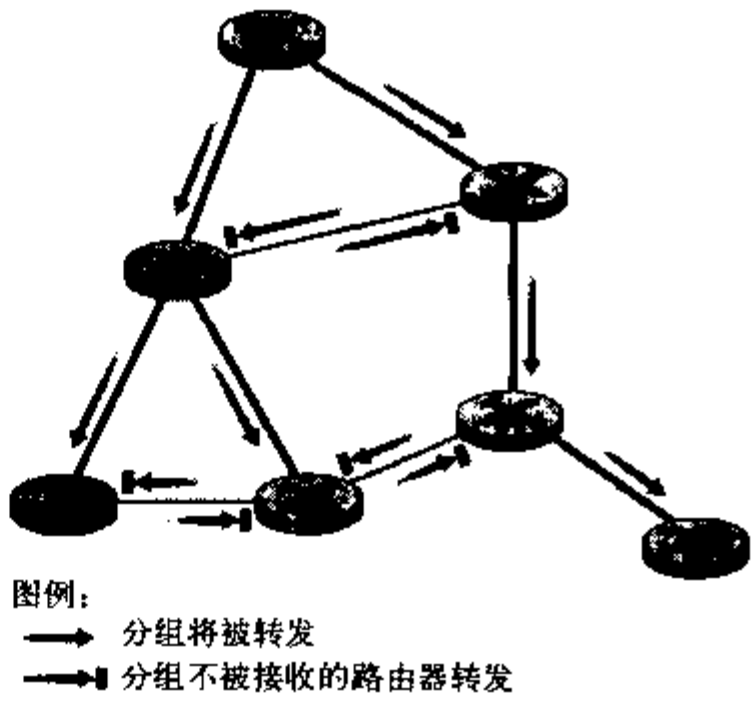
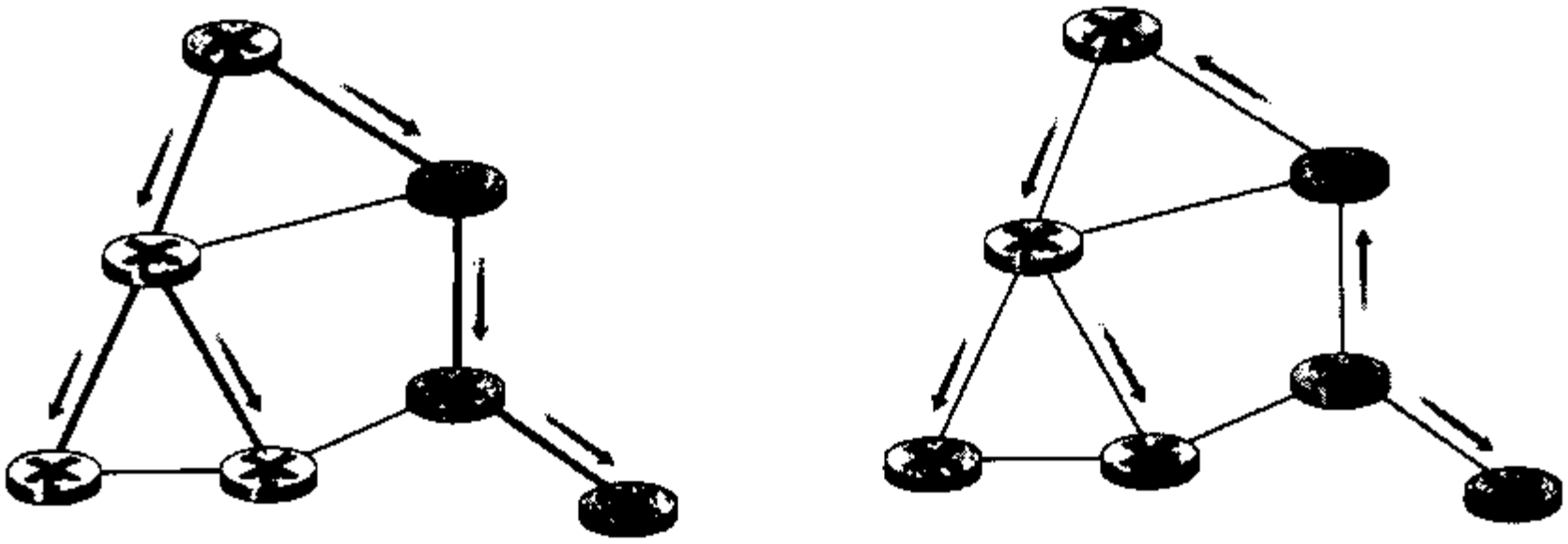


图4-45 反向路径转发

3. 生成树广播

虽然序号控制洪泛和RPF避免了广播风暴，但它们不能完全避免冗余广播分组的传输。例如，在图4-46中，节点B、C、D、E和F接收一个或两个冗余分组。在理想情况下，每个节点应当仅接收广播分组的一个拷贝。查看图4-46a中由粗线相连节点组成的树，可以看到如果广播分组仅沿着该树中的链路转发的话，每个网络节点将恰好接收到广播分组的一个拷贝，这正是我们要寻找的解决方案！该树是一棵生成

树（spanning tree）的例子，它包含了图中的每个节点。更形式化地讲，图$G=(N, E)$的一棵生成树是一个图$G'=(N, E')$，满足E'是E的子集，G'是连通的，G'不包含圈，并且G'包括了G中的所有初始节点。如果每段链路具有相应的费用且一棵树的费用就是其链路费用之和，则该图的所有生成树中费用最小的生成树称为**最小生成树**（minimum spanning tree）。



a) 由A发起的广播　　b) 由D发起的广播

图4-46 沿着一棵生成树的广播

因此，另一种提供广播的方法是首先对网络节点构造出一棵生成树。当一个源节点要发送一个广播分组时，它向所有属于该生成树的特定链路发送分组。接收广播分组的节点则向生成树中的所有邻居转发该分组（它接收该分组的邻居除外）。生成树不仅消除了冗余广播分组，而且一旦合适，该生成树便能够被任何节点用于开始广播分组，如图4-46a和图4-46b所示。注意，一个节点不必知道整棵树，它只需要知道它在G中的哪些邻居是生成树邻居。

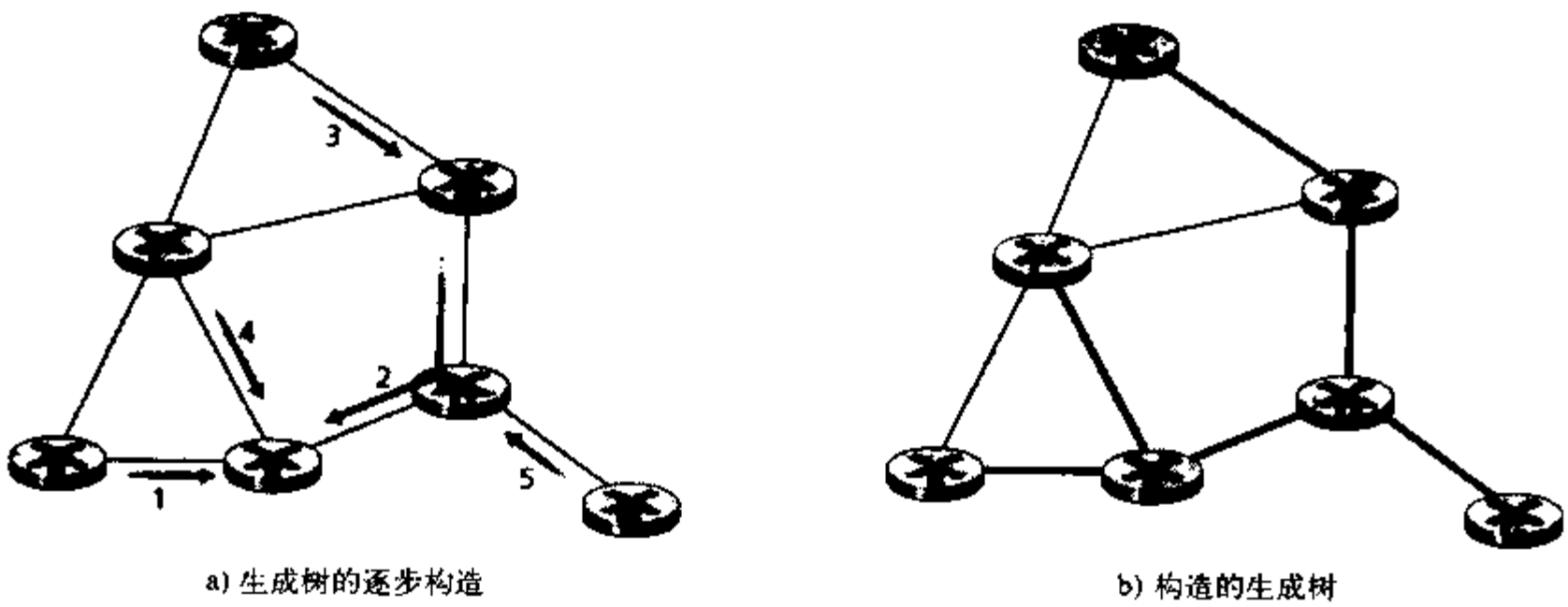
与生成树方法相关的主要复杂性是生成树的生成和维护。已经研制出了若干种分布式生成树算法[Gallager 1983, Gartner 2003]。这里我们仅考虑一种简单的算法。采用基于中心方法（center-based approach）建立一棵生成树时，要定义一个中心节点（也称为汇合点（rendezvous point）或核（core））。节点则向中心节点单播加入树（tree-join）报文。加入树报文使用单播选路朝着中心节点转发，直到它到达一个已经属于生成树的节点或到达该中心。在任一种情况下，加入树报文经过的路径定义了发起加入树报文的边缘节点和中心之间的生成树分支。你可以认为这个新分支已被嫁接到现有的生成树上了。

图4-47举例说明了基于中心生成树的构造过程。假定选择节点E作为该树的中心。假定节点F首先加入树并向E转发加入树报文。单一链路EF成为初始的生成树，然后节点B通过向E发送它的加入树报文来加入生成树。假定单播路径从B路由到E要经过D。在这种情况下，该加入树报文导致路径BDE被嫁接到生成树上。节点A接下来通过向E转发它的加入树报文来加入生成组。如果A到E的单播路径通过B，则因为B已经加入了生成树，所以A的加入树报文到达B将导致该AB链路立即被嫁接到生成树上。节点C接下来通过向E直接转发它的加入树报文，加入了生成树。最后，因为从G到E单播选路必须经过节点D，当G向E发送它的加入树报文时，该GD链路在节点D被嫁接到生成树上。

4. 实践中的广播算法

广播协议在实践中被用于应用层和网络层。如2.6节所述，Gnutella[Gnutella 2007]为了在Gnutella对等方中间广播对内容的查询，使用应用级的广播。其中，在Gnutella网络中，两个

分布式应用级对等进程之间的一条链路实际上是一个TCP连接。Gnutella使用某种形式的序号控制洪泛，其中使用了一个16比特标识符和一个16比特有效载荷描述符（它标识了Gnutella报文类型），来监测一个接收到的广播查询是否以前已经收到、复制和转发。也如2.6节所述，Gnutella也使用一个寿命（TTL）字段来限制转发一个洪泛请求通过的跳数。当一个Gnutella进程接收并复制一个请求时，它在转发请求之前减小TTL字段值。因此，一个洪泛的Gnutella请求将仅到达位于从该请求的发起者到给定数量（TTL的初始值）的应用级跳数范围内的对等方。Gnutella的洪泛机制因此有时被称为范围受限洪泛。



a) 生成树的逐步构造　　b) 构造的生成树

图4-47　基于中心构造一棵生成树

在OSPF[RFC 2328, Perlman 1999]选路算法和中间系统到中间系统（IS-IS）选路算法[RFC 1142, Perlman 1999]中，使用了一种形式的序号控制洪泛来广播链路状态通告（LSA）。OSPF使用一个32比特序列号以及一个16比特年龄（age）字段来标识LSA。前面讲过，当对邻居的一条链路的费用变化时，或当一条链路变为通/断时，OSPF节点周期性地向与它相连的链路广播LSA。LSA序号被用于检测冗余的LSA，但也服务于OSPF中的第二项重要功能。使用洪泛方法，由源在时刻t产生的一个LSA可能晚于由相同源在时刻$t+\delta$产生的较新的LSA到达某节点。由源节点使用的该序号使得一个较旧的LSA区别于较新的LSA。年龄字段起到了类似于一个TTL值的作用。初始年龄字段值被置为0，随着洪泛到每一跳而增加，并且当它位于一台路由器内存中等待洪泛时也会增加。尽管我们这里只是简要地描述了LSA洪泛算法，但我们注意到设计LSA广播协议是件技巧性很强的事情。[RFC 789; Perlman 1999]描述了一次事故，在这次事故中，由于两个故障路由器不正确地传输LSA，引起早期版本的LSA洪泛算法使整个ARPAnet瘫痪了！

4.7.2　多播

我们上一小节中已经看到了使用广播服务，可以将分组交付给网络的所有节点。在本小节中，我们重点学习**多播**（multicast）服务。使用这种服务，可以将多播分组仅交付给网络节点的一个子集。一些新兴的网络应用要求将分组从一个或多个发送方交付给一组接收方，这些应用包括批量数据传送（例如，从软件开发者到需要升级的用户之间的升级软件的传送）、流式连续媒体（例如，将一个演讲实况的音频、视频和文本传送给一组分布在多处的演讲参与者）、数据共享应用（例如，在多个分布的参与者之间共享的白板或电视会议应用）、数据供给（例如，股票报价）、Web缓存更新、交互式游戏（例如，分布式交互虚拟环境或多方游戏）。

在多播通信中，我们立即面临两个问题，即怎样标识多播分组的接收方，以及怎样为发送到这些接收方的分组编址。在单播通信情况下，接收方（目的地）的IP地址承载在每个IP单播数据报中并标识了单个接收方；在广播的情况下，所有节点需要接收广播分组，因此不需要目的地址。但在多播情况下，我们目前面对多个接收方。每个多播分组都携带所有接收方的IP地址，这合理吗？虽然这种方法对于少量的接收方可能是行得通的，但它不能很好地扩展到数以百计或数以千计的接收方场合；在数据报中，编址信息的量将淹没该分组的有效载荷字段中实际可携带的数据量。也需要给出发送方的接收方的明确标识，使得发送方知道所有接收方的标识与地址。我们很快就会看到，在许多场合下是不希望有这种要求的。

由于这些原因，在因特网体系结构（还有其他体系结构，如ATM[Black 1995]）中，多播数据报使用间接地址（address indirection）来编址。这就是说，用一个标识来表示一组接收方，寻址到该组的分组的拷贝被交付给所有与该组相关的多播接收方。在因特网中，表示一组接收方的单一标识就是一个D类多播地址。与一个D类地址相关联的接收方组称为一个**多播组**（multicast group）。多播组抽象图如图4-48所示。图中的4台主机（以深色显示）与多播组地址226.17.30.197关联，而且它们将接收所有寻址到该多播地址的数据报。我们仍然必须应对的困难在于这样一个事实，即每台主机都有一个唯一的IP单播地址，该单播地址完全独立于它所参与的多播组的地址。

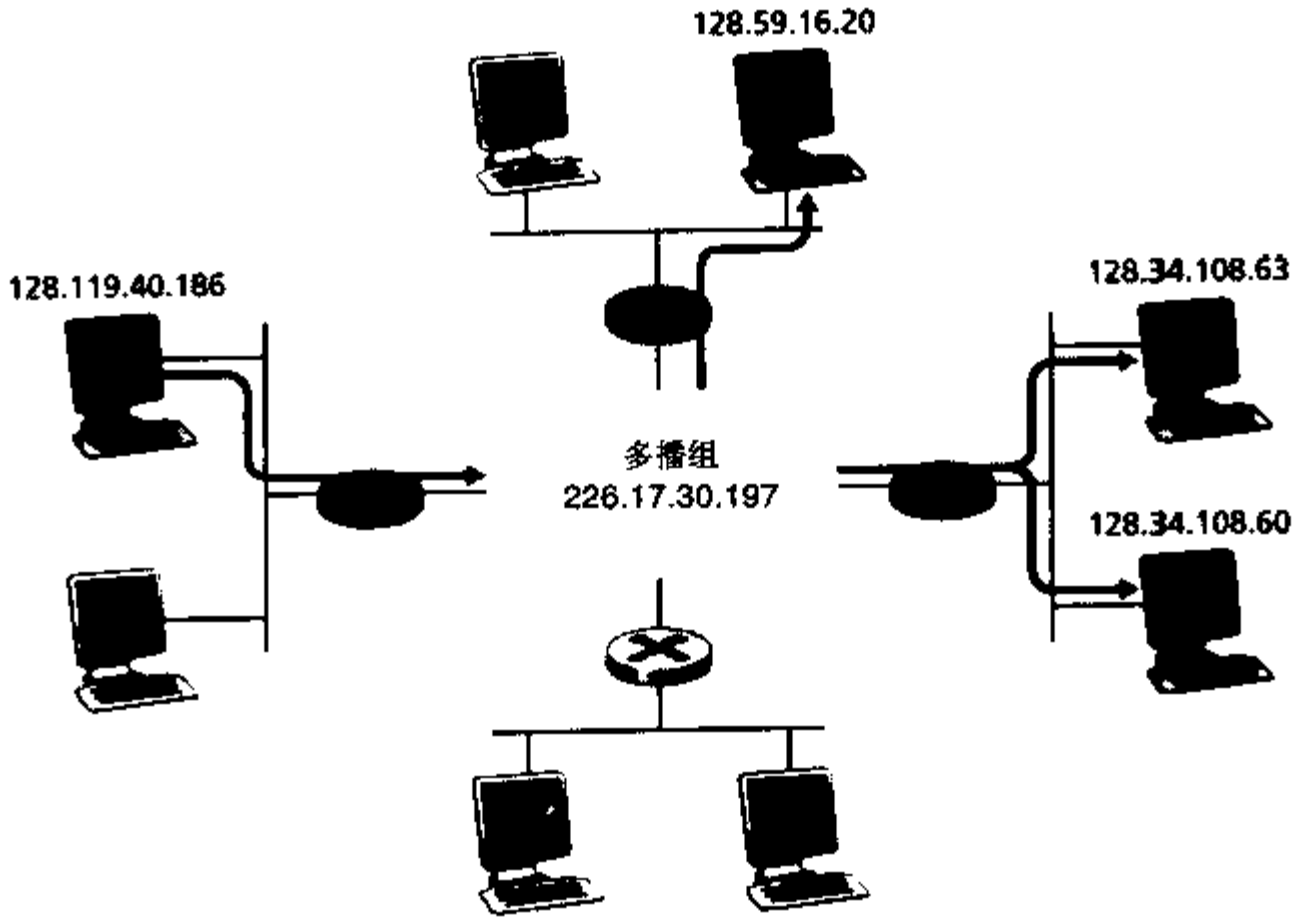


图4-48 多播组：寻址到多播组的数据报被交付给该组的所有成员

虽然多播组抽象是很简单的，但它给主机（故意使用了双关语，英语中host有主机或主人之意）带来了一堆问题。一个组是如何形成又是如何终结的呢？如何选择组地址？新主机如何加入组（要么作为发送方，要么作为接收方）？任何主机都能加入一个组（向该组发送或从该

组接收）或者组成员资格会受到限制吗？如果有限制，由谁限制？作为网络层协议的一部分，一个组成员知道其他组成员的标识吗？网络节点相互之间如何进行交互，以向所有组成员交付一个多播数据报呢？对于因特网，所有这些问题的答案都与因特网组管理协议（IGMP）[RFC 3376]有关。所以，我们接下来讨论IGMP，然后再回到这些更一般的问题上来。

1. 互联网组管理协议

IGMP版本3［RFC 3376］运行在一台主机与其直接相连的路由器之间（不严格地说，我们可将直接相连的路由器看作第一跳路由器，即主机看到的在它自己本地网络外部的到任何其他主机的一条路径，或到该主机的任何路径上的最后一跳路由器），如图4-49所示。图4-49显示了3台第一跳多播路由器，每一台都通过一个向外的本地接口与其相连的主机连接。在该例中，本地接口连到一个LAN上，且每个LAN都有多台相连的主机，在任意给定时间内至多有几台主机属于一个给定的多播组。

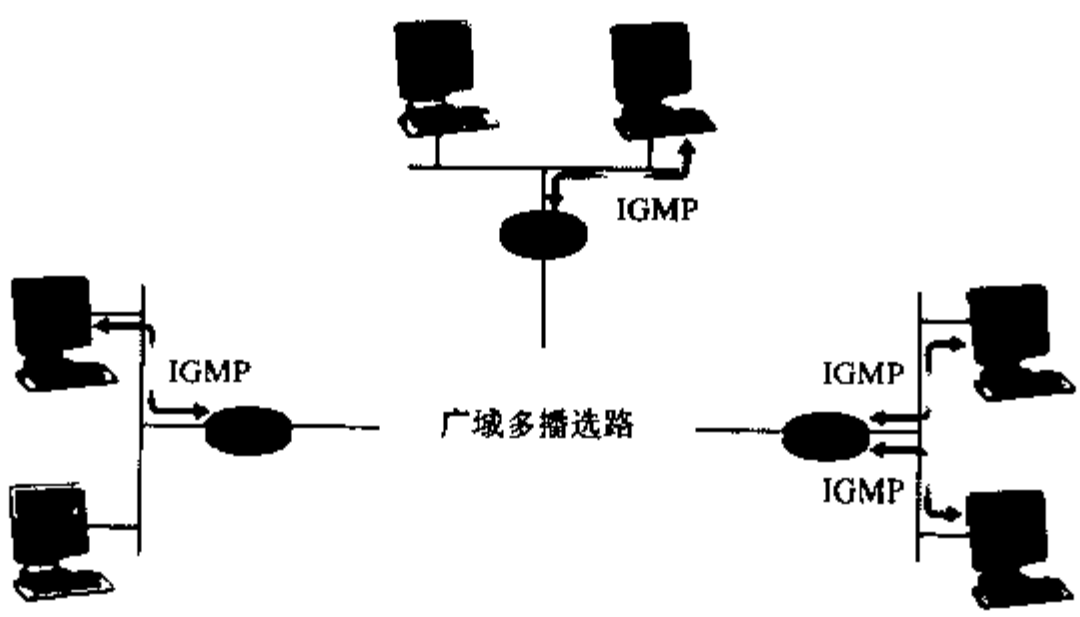


图4-49 因特网中网络层多播的两个组件：IGMP与多播选路协议

IGMP为一台主机提供了手段，可让它通知与其相连的路由器，在本主机上运行的一个应用程序想加入一个特定的多播组。由于IGMP的交互范围被局限于主机与其相连的路由器间，因此显然需要其他协议来协调遍及因特网内的多播路由器（包括相连的路由器），以便多播数据报能路由到其最终目的地。后一个功能是由网络层多播选路算法（如我们马上将讨论的那些算法）完成的。因此，因特网中的网络层多播是由两个互补的组件组成的：IGMP与多播选路协议。

IGMP只有三种报文类型。与ICMP类似，IGMP报文也是承载（封装）在一个IP数据报中，使用的IP协议号为2。由一台路由器向所有与主机（如局域网上的所有主机）相连的接口（例如，在一个局域网上对所有主机）发送一个membership_query报文，以确定该接口上的主机已加入的所有多播组集合。主机用一个membership_report报文来响应membership_query报文。当一个应用程序首次加入一个多播组时，也可由主机产生membership_report报文，而不用等待来自路由器的membership_query报文。最后一种IGMP报文类型是leave_group报文。有趣的是，这种报文是可选的！但如果它是可选的，那么路由器如何检测出一台主机是何时离开该多播组的呢？问题的答案就在于使用了membership_query报文：当无主机响应一个含有给定组地址的membership_query报文时，路由器就推断出已没有主机在这个多播组了。在因特网协议中，这是有时被称为软状态（soft state）的例子。在一个软状态协议中，状态（在IGMP场合中，有主机加入某给定多播

组的事实）如果未被显式地刷新（在本例中，由一相连主机发出的membership_report报文刷新），则通过超时事件（在本例中，通过从路由器周期性地发出membership_query报文）被删除。人们已争论过软状态协议要比硬状态协议的控制简单一些，后者不仅要求状态显式地增加或删除，还要求有一种机制能从负责删除状态的实体提早结束或失败之处进行恢复。有关软状态的有趣讨论可在［Raman 1999；Ji 2003；Lui 2004］中找到。

2. 多播选路算法

图4-50举例说明了**多播选路问题**（multicast routing problem）。加入该多播组的主机着深色，与其直接相连的路由器也着深色。如图4-50所示，只有一部分路由器（那些有相连主机加入该多播组的路由器）实际需要接收多播流量。在图4-50中，只有路由器A、B、E和F需要接收该多播组流量。因为与路由器D相连的主机中没有一个加入了该多播组，且路由器C无相连主机，所以C和D都不需要接收该多播组流量。那么多播选路的目标就是发现一棵链路树，该树连接了所有与属于该多播组的主机相连的路由器。于是多播分组沿着该树从发送方路由到所有属于该多播树的主机。当然，该树也许会包含一些不属于与该多播组主机相连的路由器（例如，在图4-50中，若不涉及路由器C或D，则不可能连接路由器A、B、E和F）。

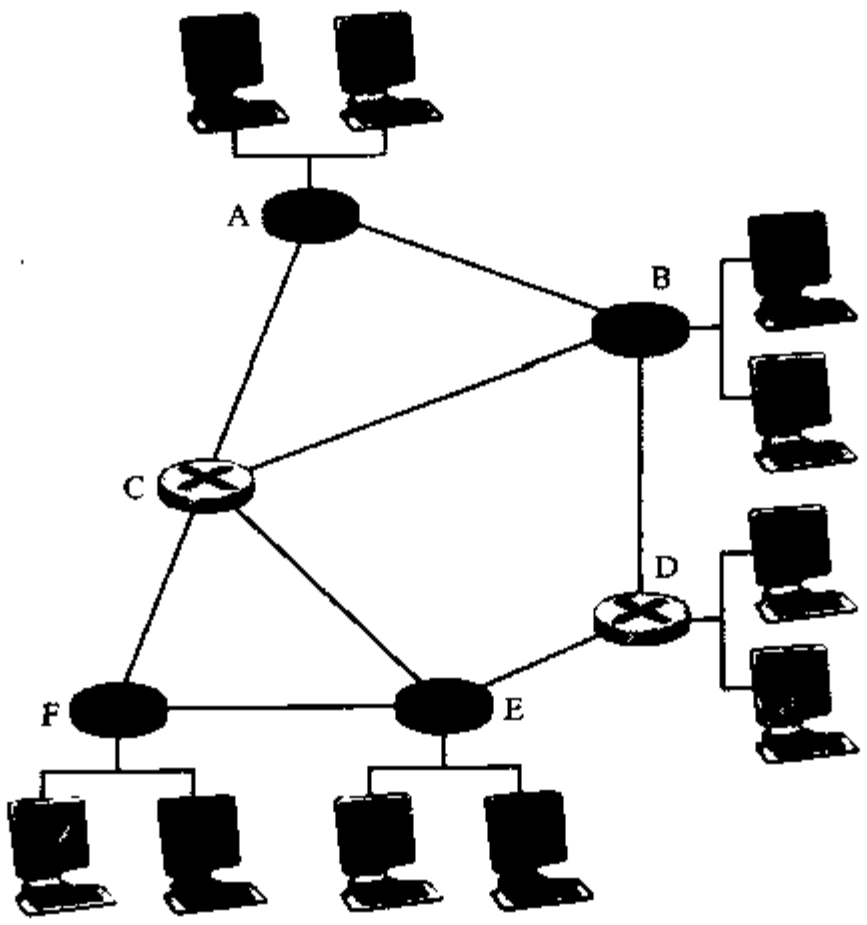


图4-50 多播主机、与其相连的路由器和其他路由器

在实践中，有两种方法用于确定多播选路树，我们已经在讨论广播选路时学习了这两种方法，因此这里只是顺便提一下。这两种方法的区别在于：是用单一的组共享树来为组中的所有发送方分发流量，还是为每个独立的发送方构建一棵特定源的选路树。

- 使用一棵组共享树进行多播选路。像在生成树广播的场合中一样，跨越组共享树的多播选路基于构建一棵树，该树包括所有具有属于该多播组相连主机的边缘路由器。在实践中，使用基于中心方法来构造多播选路树，具有属于多播组相连主机的边缘路由器向中心节点（经单播）发送加入报文。像在广播情况下一样，一个加入报文使用单播选路朝着中心转发，直到它到达已经属于多播树的一台路由器或到达该中心。沿着该加入报文走过路径的所有路由器，将向发起该多播加入的边缘路由器转发接收到的多播分组。基

于中心的树多播选路的一个关键问题就是用于选择树中心的那个过程。中心选择算法在[Wall 1980; Thaler 1997; Estrin 1997]中进行了讨论。

- 使用一棵基于源的树进行多播选路。组共享树多播选路算法构建单一的、共享的选路树来路由所有发送方的分组，而第二种方法为多播组中的每个源构建一棵多播选路树。在实践中，使用RPF算法（具有源节点x）来构造一棵多播转发树，以用于来自源节点x的多播数据报。我们前面学习的RPF算法与其用于多播环境中的要求有些不同。为了理解其中的原因，考虑图4-51中的路由器D。在广播RPF情况下，它将向路由器G转发分组，即使路由器G没有加入该多播组的相连主机。对于D只有一个下游路由器G的情况来说，这还不算太坏。想象一下，若有上千台路由器在D下游时会出现什么情况？这数千台路由器中的每一台都将收到不想要的多播分组。（这种情景并不像看起来那么想当然。最初的MBone——全球第一个多播网络[Casner 1992; Macedonia 1994]就遭遇了这样的问题。）解决在RPF下会收到不想要的多播分组这个问题的方法称为**剪枝**（pruning）。一台接收到多播分组的多播路由器，如它无加入该组的相连主机，则它向其上游路由器发送一个剪枝报文。如果一台路由器从它的每个下游路由器收到剪枝报文，则它就向上游转发一个剪枝报文。

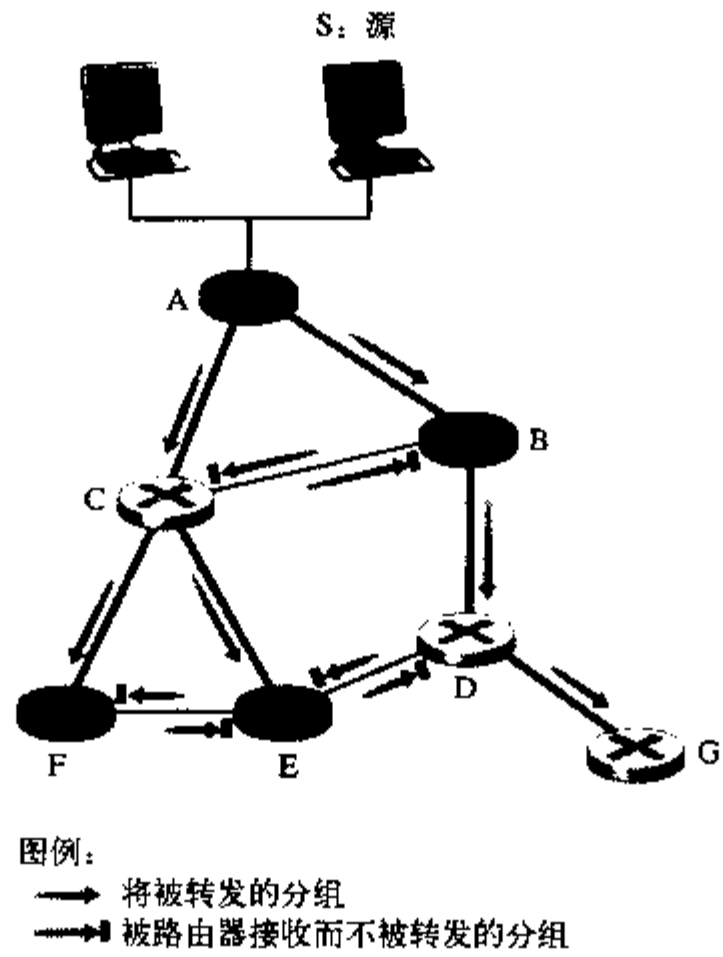


图4-51 多播情况下的反向路径转发

3. 因特网中的多播选路

第一个用于因特网中的多播选路协议是**距离向量多播选路协议**（Distance Vector Multicast Routing Protocol，DVMRP）[RFC 1075]。DVMRP实现了具有反向路径转发与剪枝算法的基于源的树。如前面所讨论的那样，DVMRP使用一种具有剪枝的RFP算法。也许使用最广泛的因特网多播选路协议是**协议无关的多播**（Protocol-Independent Multicast，PIM）**选路协议**，该协议明确辨识两种多播分布情形。在稠密模式（dense mode）[RFC 3973]中，多播组成员的位置分布稠密，即该区域内的许多或大多数路由器都需要参与到多播数据报选路过程之中。

PIM稠密模式是一种洪泛与剪枝反向路径转发技术，类似于DVMRP的思想。

在稀疏模式（sparse mode）[RFC 4601]中，具有相连组成员的路由器数量相对于路由器总数来说很少，组成员极为分散。PIM稀疏模式使用汇合点来建立多播分布树。在源特定多播（Source-Specific Multicast, SSM）[RFC 3569, RFC 4607]中，仅允许单一发送方向多播树中发送流量，从而大大简化了树的构造和维护。

当PIM和DVMP用于一个域中时，网络操作者可以配置该域中的IP多播路由器，配置方法与配置域内单播选路协议（如RIP、IS-IS和OSPF）的方法非常类似。但是，当在不同域间需要多播路由时，将会发生什么情况呢？有与域间BGP等价的多播协议吗？答案是（从文献角度讲）有。RFC 4271定义了对BGP的多协议扩展，使得BGP能够为其他协议承载选路信息，包括多播信息。多播源发现协议（Multicast Source Discovery Protocol, MSDP）[RFC 3618, RFC 4611]可以在不同的PIM稀疏模式域中用于将汇合点连接在一起。

至今为止，IP多播还没有得到大规模的应用。有关当前因特网多播服务模式和设置问题的有趣讨论，参见[Diot 2000, Sharma 2003]。尽管网络层多播没有得到广泛部署，但它远没有“消失殆尽”。多年来，Internet 2上一直在承载多播流量，并且许多网络借助于 Internet 2 进行对等通信[Internet 2 Multicast 2007]。在英国，BBC正在参与经IP多播分发内容的试验[BBC Multicast 2007]。与此同时，正如我们在第2章中看到的PPLive与其他对等系统（如End System Multicast [ESM 2007]）一样，应用层多播使用应用层（而不是网络层）多播协议在对等方之间提供了内容的多播分发。未来的多播服务主要是在网络层（在网络核心中）实现，还是在应用层（在网络边缘）实现呢？虽然当前经对等方到对等方的方法分发内容的狂热暗示着至少在近期天平向应用层多播倾斜，但是IP多播将继续取得进展，总有一天前进的步伐会放慢并稳定下来。

4.8 小结

在本章中，我们开始进入网络的核心部分。我们知道网络层涉及网络中的每台主机与路由器。正因如此，网络层协议在协议栈中是最具挑战性的协议。

我们学习了路由器可能需要在同一时刻处理不同源和目的对之间数以百万计分组的流。为了使得一台路由器能够处理如此大量的流，网络设计者多年前就认识到，路由器的任务应当尽可能简单。为了使路由器的工作更容易，可以采取许多措施，包括使用数据报网络层而不使用虚电路网络层，使用一种流水线和固定长度的首部（如IPv6中那样），取消分片（也如IPv6中那样），提供唯一的尽力而为服务。也许这里最重要的技巧不是试图跟踪各个流，而是使选路决策只依赖于数据报中层次结构化的目的地址。有趣的是，邮政服务已经使用这种方法很多年了。

在本章中，我们还学习了选路算法的基本原理。我们学习了选路算法把计算机网络抽象为一个具有节点和链路的图的方法。有了这种抽象，我们能够利用图论中丰富的最短路径选路理论，在过去40年中该理论在运筹学研究和算法界得到了发展。我们看到了有两大类方法：一种是集中式（全局）方法，在这种方法中，每个节点得到一张完整的网络图并且独立地应用一种最短路径选路算法；另一种是分布式方法，在这种方法中，各节点只有整个网络的部分知识，且各节点一起协调工作以便沿最短路径交付分组。我们也学习了如何使用分层来处理规模问题，即通过将大型网络划分成称为自治系统（AS）的独立管理域来解决规模问题。每个AS独立地为其数据报选路通过本AS，就像各个国家独立地在本国内指定邮件传递路线。

我们学习了集中式、分布式和分层方法是怎样具体应用于因特网的主要选路协议中的：RIP、OSPF和BGP。最后，讨论了广播和多播选路。

完成了对网络层的学习任务之后，我们的旅行将沿协议栈进一步向下，即到达链路层。就像网络层一样，链路层也是网络核心的一部分。但是，我们将在下一章看到，链路层具有更多的在同一链路或LAN的节点之间移动分组的局部任务。虽然从表面上看这种任务与网络层任务比起来似乎是微不足道，但我们将看到链路层涉及许多重要而迷人的问题，这些问题足以使我们忙碌很长的时间。

课后习题和问题

复习题

4.1～4.2节

1. 我们回顾一下本书中使用的某些术语。前面讲过，运输层的分组名字是报文段，数据链路层的分组名字是帧。网络层的分组名字是什么？前面讲过，路由器和链路层交换机都称为分组交换机。路由器与链路层交换机间的根本区别是什么？回想我们对数据报网络和虚电路网络都使用术语路由器。
2. 在数据报网络中，网络层的两个最重要的功能是什么？在虚电路网络中，网络层的3个最重要的功能是什么？
3. 选路和转发的区别是什么？
4. 在数据报网络和虚电路网络中，路由器都使用转发表吗？如果是，描述用于这两类网络的转发表。
5. 描述某些网络层能为单个分组提供的假想服务。对于分组流进行相同的描述。因特网的网络层为你提供了假想服务吗？ATM的CBR服务模型提供了该假想服务吗？ATM的ABR服务模型提供了该假想服务吗？
6. 列出某些得益于ATM的CBR服务模型的应用。

4.3节

7. 讨论为什么在高速路由器的每个输入端口都存储转发表的影子拷贝。
8. 4.3节中讨论了3类交换结构，列出每一类交换结构并简要讨论之。
9. 描述在输入端口会出现分组丢失的原因。描述在输入端口能够消除分组丢失的原因（不使用无限大缓存区）。
10. 描述在输出端口能够出现分组丢失的原因。
11. 什么是HOL阻塞？它出现在输入端口还是输出端口？

4.4节

12. 路由器有IP地址吗？如果有，有多少个？
13. IP地址223.1.3.27的32比特二进制等价形式是什么？
14. 考察使用DHCP获得它的IP地址、网络掩码、默认路由器和其本地DNS服务器的IP地址的主机。列出这些值。
15. 假设在一个源主机和一个目的主机之间有3台路由器。不考虑分片，一个从源主机发送给目的主机的IP报文将通过多少个接口？为了将数据报从源移动到目的地需要检索多少个转发表？
16. 假设某应用每20 ms生成一个40 字节的数据块，每块封装在一个TCP报文中，TCP报文再封装在一个IP数据报中。每个数据报的开销有多大？应用数据所占的百分比是多少？
17. 假定主机A向主机B发送封装在一个IP数据报中的TCP报文段。当主机B接收到该数据报时，主机B中的网络层怎样知道它应当将该报文段（即数据报的有效载荷）交给TCP而不是UDP或某个其他东西呢？

18. 假定你购买了一个无线路由器并将其与电缆调制解调器相连，并且你的ISP动态地为你连接的设备（即你的无线路由器）分配一个IP地址。还假定你家有5台PC，均使用802.11以无线方式与该无线路由器相连。怎样为这5台PC分配IP地址？该无线路由器使用NAT吗？为什么？
19. 比较IPv4和IPv6首部字段。它们有相同的字段吗？
20. 有人说当IPv6通过IPv4路由器建隧道时，IPv6将IPv4隧道作为链路层协议。你同意这种说法吗？为什么？

4.5节

21. 比较链路状态选路算法和距离向量选路算法。
22. 讨论为什么因特网的分层组织使其能够扩展为数以百万计用户。
23. 每个自治系统使用相同的AS内部选路算法是必要的吗？为什么？

4.6节

24. 考虑图4-35。从D中的初始表开始，假设D收到来自A的下面的通告：

目的子网	下一台路由器	到目的地的跳数
z	C	10
w	—	1
x	—	1
....	....	....

D中的表会改变吗？如果是，怎样变化？

25. 比较RIP与OSPF使用的通告。
26. 填空：RIP通告通常是通告到各目的地的跳数。另一方面，BGP则是通告到各目的地的_____。
27. 为什么在因特网中用到了不同类型的AS间与AS内部选路协议？
28. 为什么对于AS内部协议（如OSPF和RIP）与AS间选路协议（如BGP）来说，策略的考虑一样重要呢？
29. 定义和对比下列术语：子网、前缀和BGP路由。
30. BGP是怎样使用NEXT-HOP属性的？又是怎样使用AS-PATH属性的？
31. 描述一个较高层ISP的网络管理员在配置BGP时，能够实现策略的方法。

4.7节

32. 通过多个单播实现广播抽象与通过支持广播的单个网络（路由器）实现广播抽象之间有什么重要区别吗？
33. 对于我们学习的3种一般的广播通信方法（无控制洪泛、受控洪泛和生成树广播），下列说法正确吗？可以假定分组不会因缓存溢出而丢失，所有分组以它们发送的顺序交付给链路。
    a. 一个节点可能接收到同一个分组的多个拷贝。
    b. 一个节点可能跨越相同的出链路转发多个分组的拷贝。
34. 当一台主机加入一个多播组时，它必须将其IP地址改变为它所加入的多播组的地址吗？
35. IGMP和广域多播选路协议所起的作用是什么？
36. 在多播选路场合中，一棵组共享的树与一棵基于源的树之间有什么区别？

习题

1. 考虑关于虚电路和数据报网络的某些优缺点。
    a. 假设在网络层，路由器遇到了通常会引起它不能正常运转的紧急情况。从总体上来看，对于这种路由器故障需要采取什么动作呢？这种争论是有利于虚电路体系结构还有有利于数据报体系结构？

b. 假设为了在沿某条源到目的地的路径上提供性能等级（如时延）方面的保证，网络要求一个发送方宣布其峰值通信速率。如果该发送方宣布的峰值通信速率和现有已宣布的峰值通信速率的现状，无法让从源到目的地的流量满足指定的时延需求，则不允许该源访问网络。这种方法是在虚电路体系结构还是在数据报体系结构中更容易实现呢？

2. 考虑一个虚电路网络。假定其VC号是一个16比特字段。
   a. 链路能够承载的虚电路的最大数量是什么？
   b. 假定某中心节点在连接建立时确定了路径和VC号。假定沿着某虚电路的路径在每段链路使用相同的VC号。描述在连接建立时该中心节点确定VC号的可能方式。下列情况可能出现吗？即进行中虚电路比(a)问题中确定的最大值要少，仍没有相同的未用VC号。
   c. 假定沿着某条虚电路的路径允许不同的VC号。在连接建立期间，在端到端路径确定以后，描述以分散方式且不依赖中心节点，链路怎样选择它们的VC号并配置它们的转发表。
3. 在虚电路网络中，基本转发表具有4列，在这些列中，值的含义是什么？在数据报网络中，基本转发表有两列，在这些列中，值的含义是什么？
4. 考虑一个具有2比特字段用于VC号的虚电路网络。假定该网络要通过4条链路（链路A、链路B、链路C和链路D）建立一条虚电路。假定每条链路当前都承载两条其他的虚电路，这些其他虚电路的VC号如下：

链路A	链路B	链路C	链路D
00	01	10	11
01	10	11	00

在回答下列问题时，记住每个现有的虚电路可能通过这4条链路之一。

   a. 如果每条虚电路要求沿着其路径使用相同的VC号，可以为该新的虚电路分配什么样的VC号？
   b. 如果每条虚电路允许沿着其路径的不同链路使用不同的VC号（因此转发表必须执行VC号转换），可以使用4个VC号的多少种不同的组合（一种组合用于4条链路之一）？
5. 书中我们使用了术语面向连接服务来描述运输层，使用了术语连接服务来描述网络层。为何有这种微妙的差异？
6. 在4.3节中，在假定n个输入线路都具有相同的线路速率条件下，我们注意到如果交换结构比输入线路速率快n倍，将不可能有输入排队。解释（用语言）这是为什么。
7. 考虑一台具有交换结构的路由器，它有2个输入端口（A和B）和2个输出端口（C和D）。假定该交换结构以1.5倍的线路速率运行。
   a. 由于某种原因，如果来自A的所有分组的目的地为D，来自B的所有分组的目的地为C，能够设计出一种交换结构使得在输入端口不存在排队吗？用一句话来解释其中的原因。
   b. 假定此时来自A和B的分组的目的地随机地为C和D。能够设计出一种交换结构使得在输入端口不存在排队吗？用一句话来解释其中的原因。
8. 考虑使用32比特主机地址的数据报网络。假定一台路由器具有4条链路，编号从0到3，分组能被转发到如下的各链路接口：

目的地址范围	链路接口
11100000 00000000 00000000 00000000	
到	0
11100000 11111111 11111111 11111111	

11100001 00000000 00000000 00000000	
到	1
11100001 00000000 11111111 11111111	
11100001 00000001 00000000 00000000	
到	2
11100001 11111111 11111111 11111111	
其他	3

a. 提供一个具有4个表项的转发表，使用最长前缀匹配，将分组转发到正确的链路接口。

b. 描述你的转发表是如何为具有下列目的地址的数据报决定适当链路接口的。

11001000 10010001 01010001 01010101

11100001 00000000 11000011 00111100

11100001 10000000 00010001 01110111

9. 考虑使用8比特主机地址的数据报网络。假定一台路由器使用最长前缀匹配并具有下列转发表：

前缀匹配	接 口
00	0
01	1
10	2
11	3

对这4个接口，给出相关的目的主机地址的范围和该范围中的地址数量。

10. 考虑使用8比特主机地址的数据报网络。假定一台路由器使用最长前缀匹配并具有下列转发表：

前缀匹配	接 口
1	0
11	1
111	2
其他	3

对这4个接口，给出相关的目的主机地址的范围和该范围中的地址数量。

11. 考虑互连3个子网（子网1、子网2和子网3）的路由器。假定这3个子网的所有接口都要求具有前缀223.1.17/24。还假定子网1要求支持多达125个接口，子网2和子网3都要求支持多达60个接口。提供3个满足这些限制的网络地址（形式为a.b.c.d/x）。

12. 4.2.2节中给出了一个转发表（使用最长前缀匹配）的例子。使用a.b.c.d/x记法代替二进制字符串记法重写该转发表。

13. 在习题7中要求给出转发表（使用最长前缀匹配）。使用a.b.c.d/x记法代替二进制字符串记法重写该转发表。

14. 考虑一个具有前缀101.101.101.64/26的子网。给出能被分配给该网络的一个IP地址（具有形式xxx.xxx.xxx.xxx）的例子。假定一个ISP拥有形如101.101.101.128/17的地址块。假定它要从该地址块产生4个子网，每块具有相同数量的IP地址。对这4个子网，其前缀（形式为a.b.c.d/x）是什么？

15. 考虑图4-17中显示的拓扑。在12：00顺时针标记具有主机的3个子网为网络A、B和C，标记没有主机的子网为网络D、E和F。

a. 为这6个子网分配网络地址，要满足下列限制：所有地址必须从214.97.254/23起分配；子网A应当具有足够的地址以支持250个接口；子网B应当具有足够的地址以支持120个接口；子网C应当具有

足够的地址以支持120个接口。当然，子网D、E和F都应当支持两个接口。对于每个子网，分配应当具有的形式是a.b.c.d/x或a.b.c.d/x - e.f.g.h/y。

b. 使用你对(a)部分的答案，为这3台路由器提供转发表（使用最长前缀匹配）。

16. 考虑向具有500字节的MTU的链路发送一个3000字节的数据报。假定初始数据报具有标识号422。将会产生多少个报文段？它们的特征是什么？

17. 假定源主机A和目的主机B之间的数据报被限制为1500字节（包括首部）。假设IP首部为20字节，要发送由400万字节组成的MP3文件需要多少个数据报？

18. 考虑图4-22中建立的网络。假定 ISP此时为路由器分配地址126.13.89.67，家庭网络的网络地址是192.168/16。

a. 在家庭网络中为所有接口分配地址。

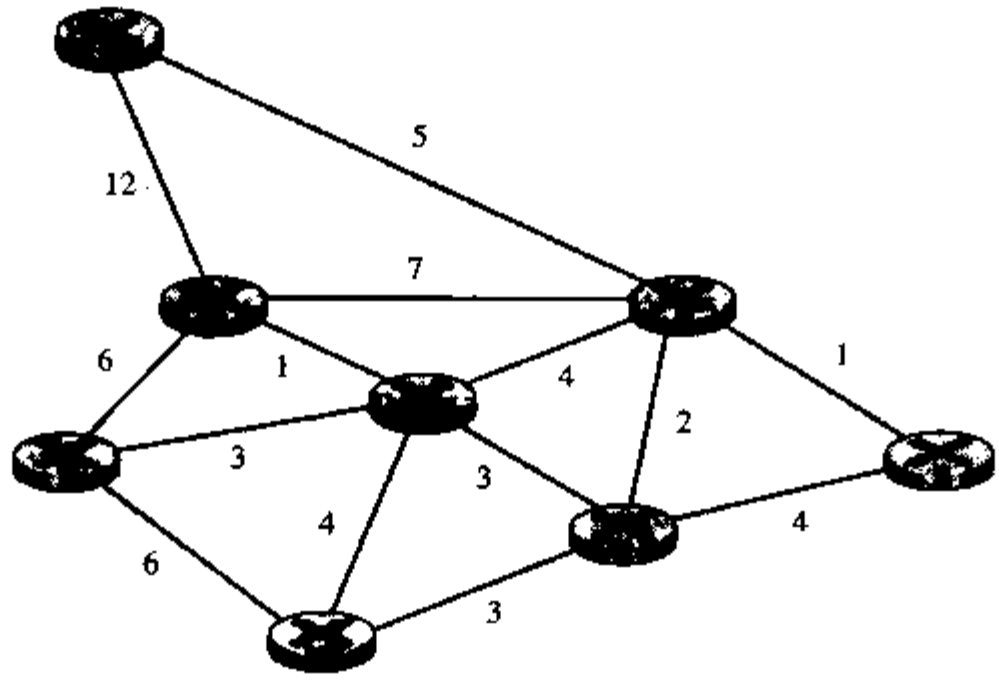
b. 假定每台主机具有两个进行中的TCP连接，且都是针对主机128.119.40.86的80端口的。在NAT转换表中提供6个对应项。

19. 在这个习题中我们将探讨NAT对P2P应用程序的影响。假定用户名为Arnold的对等方通过查询发现用户名为Bernard的对等方有一个他要下载的文件。同时假定Bernard和Arnold都位于NAT后面。尝试设计一种技术使得Arnold与Bernard创建一个TCP连接，而不对NAT进行应用特定的配置。如果难以设计这样的技术，试讨论其原因。

20. 观察图4-27，列举从v到y的不包含任何环路的路径。

21. 重复习题20，列举从x到w、w到u以及z到x的不包含任何环路的路径。

22. 考虑下面的网络。对于标明的链路费用，用Dijkstra最短路算法计算出从x到所有网络节点的最短路径。通过计算一个类似于表4-3的表，说明该算法是如何工作的。



23. 考虑习题22中所示的网络。使用Dijkstra算法和一个类似于表4-3的表来说明你做的工作。

a. 计算出从s到所有网络节点的最短路径。

b. 计算出从t到所有网络节点的最短路径。

c. 计算出从u到所有网络节点的最短路径。

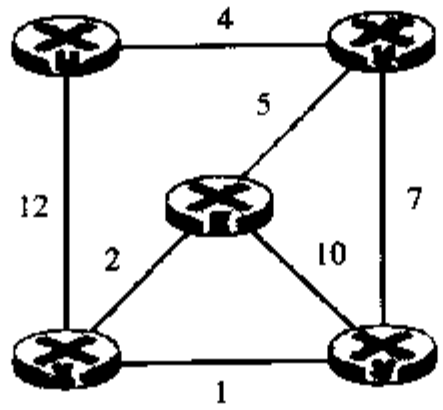
d. 计算出从v到所有网络节点的最短路径。

e. 计算出从w到所有网络节点的最短路径。

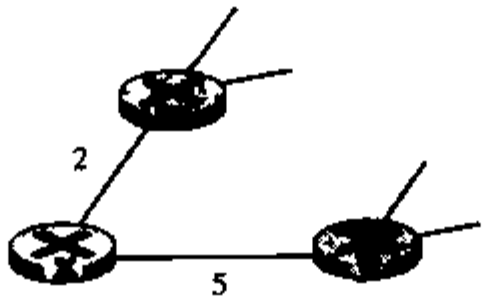
f. 计算出从y到所有网络节点的最短路径。

g. 计算出从z到所有网络节点的最短路径。

24. 考虑下图所示的网络。假设每个节点初始时知道到其每个邻居的费用。考虑距离向量算法，并给出节点z的距离表表项。



25. 考虑一个一般性拓扑（即不是以上所显示的特定网络）和一个同步版本的距离向量算法。假设每次迭代时，一个节点与其邻居交换其距离向量并接收它们的距离向量。假定算法开始时，每个节点只知道到其直接邻居的费用，在该分布式算法收敛前所需的最大迭代次数是多少？验证你的答案。
26. 考虑下图所示的网络段。x只有两个相连邻居w与y。w有一条通向目的地u（没有显示）的最低费用路径，其值为5。y有一条通向目的地u的最低费用路径，其值为6。从w与y到u（以及w与y之间）的完整路径未显示出来。网络中的所有链路费用皆为正整数值。

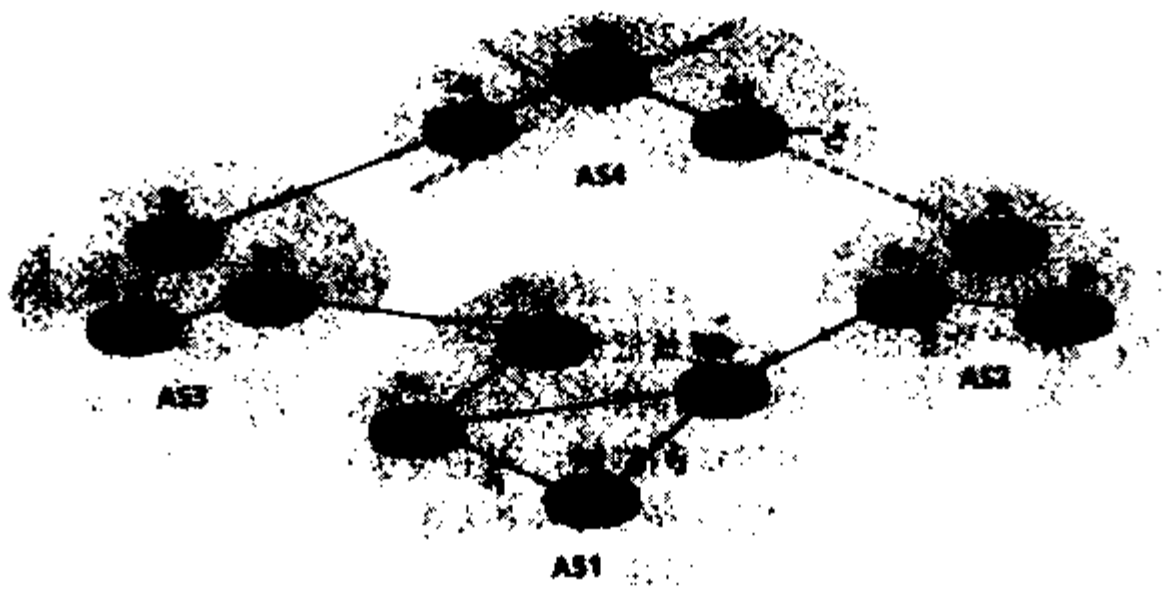


a. 给出x对目的地w、y和u的距离向量。

b. 给出对于$c(x, w)$或$c(x, y)$链路费用的变化，使得执行了距离向量算法后，x将通知其邻居有一条通向u的新最低费用路径。

c. 给出对于$c(x, w)$或$c(x, y)$ 链路费用的变化，使得执行了距离向量算法后，x将不通知其邻居有一条通向u的新的最低费用路径。

27. 考虑图4-30所示的3个节点的拓扑。不使用显示图4-30中的费用值，而采用新链路费用值：$c(x, y) = 5$，$c(y, z) = 6$，$c(z, x) = 2$。在距离向量表初始化后和同步版本的距离向量算法每次迭代后，计算它的距离向量表（如我们以前对图4-30所做的那样）。
28. 描述在BGP中是如何检测路径中的环路的。
29. 考虑下图所示的网络。假定AS3和AS2正在运行其AS内部选路协议OSPF，AS1和AS4正在运行其AS内部选路协议RIP。假定AS间选路协议使用的是eBGP和iBGP。初始时，假定在AS2和AS4之间不存在物理链路。



a. 路由器3c从OSPF、RIP、eBGP或iBGP中的哪个选路协议学习到了前缀x?

b. 路由器3a从哪个选路协议学习到了前缀x?

c. 路由器1c从哪个选路协议学习到了前缀x?

d. 路由器1d从哪个选路协议学习到了前缀x?

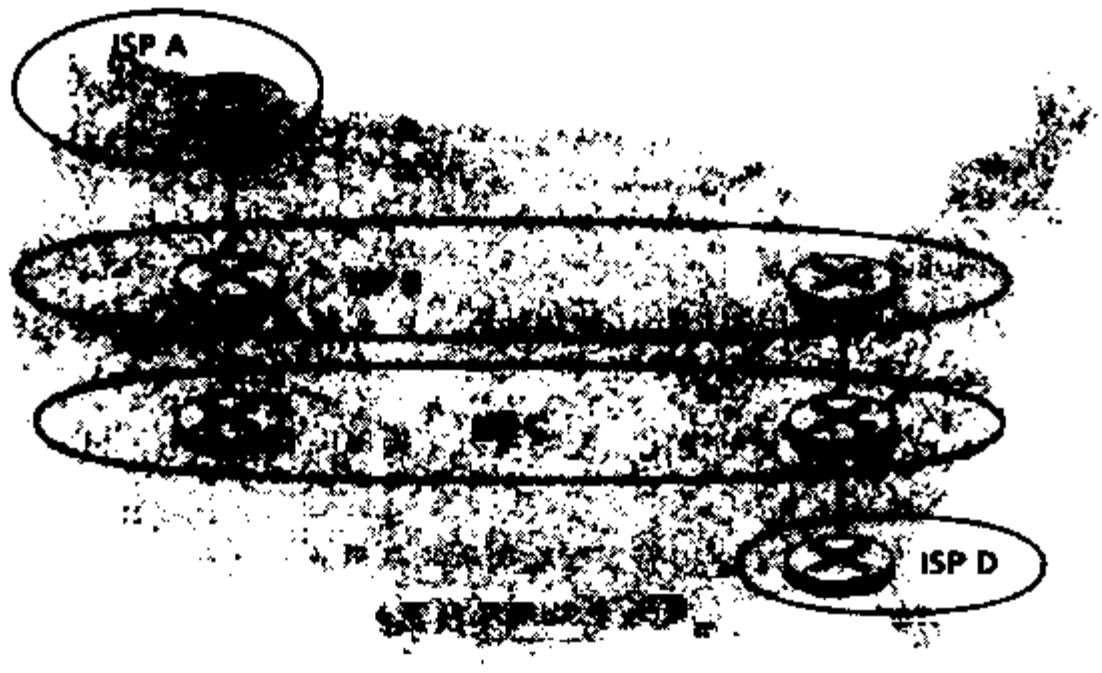
30. 参考习题29，一旦路由器1d知道了x的情况，它就将一个表项(x, l)放入其转发表中。

a. 对这个表项而言，l将设置为l_1还是l_2? 用一句话解释其原因。

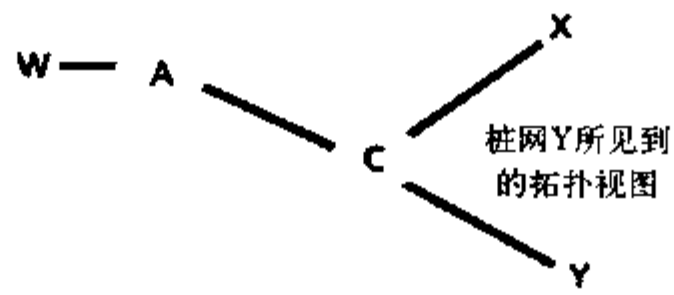
b. 现在假定在AS2和AS4之间有一条物理链路，如图中的虚线所示。假定路由器1d知道经AS2以及经AS3能够访问到x。l将设置为l_1还是l_2? 用一句话解释其原因。

c. 现在假定有另一个AS，称为AS5，它位于路径AS2和AS4之间（没有显示在图中）。假定路由器1d知道经AS2 AS5 AS4以及经AS3 AS4能够访问到x。l将设置为l_1还是l_2? 用一句话解释其原因。

31. 考虑下面的网络。ISP B为地区ISP A提供全国主干服务。ISP C为地区ISP D提供全国主干服务。每个ISP由一个AS组成。通过使用BGP，B和C在两个地方互相对等。考虑从A到D的流量。B宁愿将流量传递给（美国）西海岸C（以致C将必须承受承载流量跨越整个国家的费用），而C宁愿经其（美国）东海岸与B对等的站点得到这些流量（以致B将承载跨越整个国家的流量）。C可能会使用什么样的BGP机制，以便B通过东海岸对等点传递A到D的流量? 要回答这个问题，你需要钻研BGP规约。



32. 在图4-43中，考虑一下到达桩网W、X和Y的路径信息。基于W与X上的可用信息，它们分别看到的网络拓扑是什么? 验证你的答案。Y所见到的拓扑视图如下图所示。



33. 考虑习题22中8个节点的网络（节点标为s到z）。给出根在s包括节点u、v、w和y的最低费用树。非形式化地讨论一下，为什么你给出的树是一棵最低费用树。

34. 考虑实现广播的两种基本方法：单播模拟与网络层（即路由器协助）广播，并假定使用生成树广播来实现网络层广播。考虑有1个发送方与32个接收方。假设发送方通过一棵路由器二叉树与接收方相连。在单播模拟与网络层多播两种情况下，对这个拓扑发送一个广播分组的费用各是多少? 这里每次经单一链路上发送一个分组（或一个分组的拷贝），产生一个单位费用。用什么样的拓扑互连发送

方、接收方和路由器将使得单播模拟与真正的网络层广播产生的费用相差尽可能大？你可按照自己的意愿选择多台路由器。

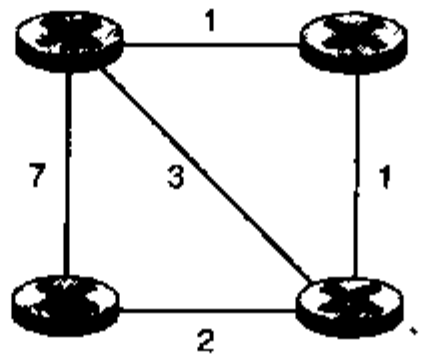
35. 考虑图4-45中反向路径转发（RPF）算法的运行。使用相同的拓扑，找出从所有节点到源节点A的一系列路径（并像图4-45中那样用粗线指出这些路径），使得如果这些路径是最低费用路径，则节点B将接收来自使用RPF的节点A、C和D的A的广播报文的拷贝。
36. 考虑图4-45中所示的拓扑。假定所有链路具有单位费用并且节点E是广播源。给定节点E为源，使用图4-45中所示的箭头指出使用RPF转发分组的链路，以及不转发分组的链路。
37. 考虑图4-47中所示的拓扑，假定每段链路有单位费用。假设节点C在基于中心多播选路算法中被选为中心。假定多播组内每个相连的路由器都用到节点C的最低费用路径来向C发送加入报文，画出所产生的基于中心的多播选路树。产生的树是一棵最低费用树吗？验证你的答案。
38. 在4.5.1节中我们学习了计算单播路径的Dijkstra链路状态选路算法，这些单播路径分别是从源到所有目的地的最低费用路径。这些路径的并集可被认为形成了一棵**最低单播费用路径树**（或一棵最短单播路径树，如果所有链路费用相同的话）。通过构造一个反例，表明最低费用路径树并不总是与最小生成树相同。
39. 考虑所有节点与3个其他节点相连的网络。在单一时间步中，一个节点能够从它的邻居接收到所有传输的广播分组，复制分组，并向它的所有邻居发送之（除了发送给定分组的那个节点）。在下个时间步中，相邻节点能够接收、复制和转发这些分组，等等。假定使用无控制洪泛在这样的网络中提供广播。在时间步t，多少个广播分组的拷贝将被传输？假定在时间步1期间，从源节点向它的3个邻居传输单个广播分组。
40. 我们在4.7节中看到，没有网络层协议能被用于标识参与一个多播组的主机。在这种情况下，多播应用程序怎样知道参与一个多播组的主机的身份？
41. 设计（给出伪代码描述）一个应用级协议，该协议维护参与一个多播组的所有主机的地址。特别要指出你的协议所使用的网络服务（单播或多播），并且指出你的协议发送报文是在带内还是在带外（在多播组参与者之间关于应用数据的流动方式），并说明其理由。
42. 多播地址空间的长度有多大？假设现有两个不同的多播组随机地选择一个多播地址。它们选择同一个地址的概率有多大？假设现有1000个多播组同时在进行，随机选择它们的多播组地址。发生冲突的概率有多大？

讨论题

1. 找出3个当前正在出售高速路由器产品的公司。比较它们的产品。
2. 使用美国因特网号码注册机构提供的whois服务（http://www.arin.net/whois），确定3所大学使用的IP地址块。whois服务能够用于确定一个特定IP地址的确切位置吗？
3. 能用Java语言编写ping客户机程序吗？为什么？
4. 在4.4节中，我们指出部署IPv6是一个缓慢的过程。这个过程为什么是缓慢的？要加快其部署需要些什么？
5. 讨论NAT对IPsec安全性所产生的问题（参见[Phifer 2000]）。
6. 研究UPnP协议。特别描述主机重配置NAT所使用的报文。
7. 假设AS X和Z不是直接相连的，而是通过AS Y相连。再假设X有一个与Y的对等协定，Y有一个与Z的对等协定。最后，假设Z想传送Y的所有流量，但不想传送X的流量。BGP允许Z实现该策略吗？
8. 在4.7节中，我们指出了许多多播应用。哪些应用较适合最低限要求的因特网多播服务模型？为什么？哪些应用不是特别适合该服务模型？

编程作业

在本编程作业中，你要写一段过程的“分布式”集合，为下图所示的网络实现一个分布式异步距离向量选路算法。



你要编写下列例程，这些例程将异步“执行”在为该编程作业提供的模拟环境中。对于节点0，你将写出这样的例程：

- rtinit0()。模拟一旦开始该例程就将被调用。rtinit0()无参数。它应当初始化节点0中的距离表，以反映出到达节点1、2和3的直接费用分别为1、3和7。在上图中，所有链路都是双向的，两个方向的费用皆相同。在初始化距离表和节点0的例程所需的其他数据结构后，它应向其直接连接的邻居（在这里为节点1、2和3）发送它到所有其他网络节点的最低费用路径的费用信息。这种最低费用信息通过调用例程tolayer2()在一个选路更新分组中发送给相邻节点，就像习题中描述的那样。选路更新分组的格式也在习题中有描述。
- rtupdate0(struct rtpkt *rcvpkt)。当节点0收到一个由直接相连邻居发给它的选路分组时，调用该例程。参数*rcvpkt是一个指向接收分组的指针。rtupdate0()是距离向量算法的“核心”。它从其他节点i接收的选路更新分组中包含有节点i到所有其他网络节点的当前最短路费用值。rtupdate0()使用这些收到的值来更新其自身的距离表（这是距离向量算法所规定的）。如果它自己到另外节点的最低费用因此更新而发生改变的话，则节点0通过发送一个选路分组来通知其直接相连邻居这种最低费用的变化。我们在距离向量算法中讲过，仅有直接相连的节点才交换选路分组。因此，节点1和节点2将相互通信，但节点1和节点3将不相互通信。

为节点1、2、3定义类似的例程。因此，你总共要编写8个过程：rtinit0()、rtinit1()、rtinit2()、rtinit3()、rtupdate0()、rtupdate1()、rtupdate2()和rtupdate3()。这些例程将共同实现一个分布式的、与图中所示拓扑和费用相关的距离表的异步计算。

可从本书配套网站http://www.awl.com/kurose-ross处找到该编程作业的全部详细资料，以及你创建模拟硬件/软件环境所需的C程序代码。另外，该网站上还包括一个Java版的编程作业。

Ethereal实验

在本书配套网站http://www.awl.com/kurose-ross上，可以找到两个Ethereal实验作业。第一个实验研究了IP协议的运行，特别是IP数据报的格式。第二个实验探讨了在ping和traceroute命令中ICMP协议的使用。

人物专访

Vinton G. Cerf是Internet Evangelist for Google公司的副总裁兼总管。他在MCI公司工作了16年，从事过各项工作，最后一项工作是担任技术部门的资深副总裁。从1976年到1982年，他在美国国防部高级研究计划署（DARPA）任职期间，在领导因特网以及与因特网相关的数据分组和安全技术的研发方面发挥了重要作用。他于2005年获得了美国总统自由奖章，于1997年获得了美国国家技术奖章。他在斯坦福大学获得数学学士学位，在加州大学洛杉矶分校（UCLA）获得计算机科学的硕士和博士学位。

Vinton G. Cerf

• 是什么使您专注于网络技术的呢?

20世纪60年代末，我在UCLA一直做程序员的工作。我的工作得到了美国国防部高级研究计划署（那时叫ARPA，现在叫DARPA）的支持。我那时在刚创建不久的ARPAnet的网络测量中心，为Leonard Kleinrock教授的实验室工作。ARPAnet的第一个节点于1969年9月1日安装在UCLA。我负责为计算机编程，以获取有关ARPAnet的性能信息，并且报告该信息以便与数学模型作比较，预测网络性能。

我和其他几名研究生负责研制ARPAnet的所谓主机级协议，该协议的过程和格式将使得网络中许多不同类型的计算机彼此交互。这是我进入分布式计算与通信新世界的一次迷人的探索。

• 当您第一次设计该协议时，您曾想象过IP会像今天这样无所不在吗?

当我和Bob Kahn于1973年最初从事该项工作时，我想我们的注意力大多集中在这样一个重要的问题：假定实际上不能改变这些网络本身，我们怎样才能让异构的分组网络能够彼此互操作。我们希望能找到一种方法使任意多的分组交换网以透明的方式互连，以便主机之间不做任何转换就能进行端到端通信。我认为那时我们已经知道了我们正在处理强大的可扩充的技术，但还没清楚地想过有数亿台计算机都连入因特网时的世界会是什么样。

• 您现在能预见网络与因特网的未来吗？您认为在它们的发展中存在的最大挑战或障碍是什么?

我相信因特网本身以及一般的网络都将继续扩大。已有令人信服的证据表明，在因特网上将有数十亿个因特网使能设备，包括手机、冰箱、个人数字助理、家用服务器、电视等家用电器，以及大批通常的便携机、服务器等。重大挑战包括支持移动性、电池寿命、网络接入链路的容量、以不受限的方式扩展网络光学核心的能力。设计因特网的星际扩展是我在喷气推进实验室（Jet Propulsion Laboratory）深入研究的一项计划。我们需要从IPv4（32比特地址）过渡到IPv6（128比特）。要做的事情实在是太多了!

• 是谁激发了您专业上的灵感?

我的同事Bob Kahn；我的论文导师Gerald Estrin；我最好的朋友Steve Crocker（我们在高中就认识了，1960年他将我引入了计算机学科）；以及数千名今天仍在继续推动因特网发展的工程师。

• 您对进入网络/因特网领域的学生有什么忠告吗?

要跳出现有系统的限制来思考问题，想一想什么是可行的；随后再做艰苦工作，以推算出如何从事物的当前状态到达所想的状态。要敢于想象：我和喷气推进实验室的6个同事一直在从事陆地因特网的星际扩展设计。这也许要花几十年才能实现，任务会一个接着一个地出现，可用这句话来总结："一个人总是要不断地超越自我，否则还有什么乐趣可言？"

第5章　链路层和局域网

在上一章中我们学习了网络层提供的两台主机之间的通信服务。如图5-1所示，该通信路径由一系列通信链路组成，从源主机开始，经过一系列路由器，在目的主机结束。当我们沿协议栈继续往下，从网络层到达链路层，我们自然而然地想知道分组是如何通过构成端到端通信路径的每段链路的。为了在单段链路上传输，网络层的数据报是怎样被封装进链路层帧的呢？链路层协议能够提供路由器到路由器的可靠数据传输吗？沿此通信路径，不同的链路能够采用不同的链路层协议吗？我们将在本章回答这些和其他一些重要的问题。

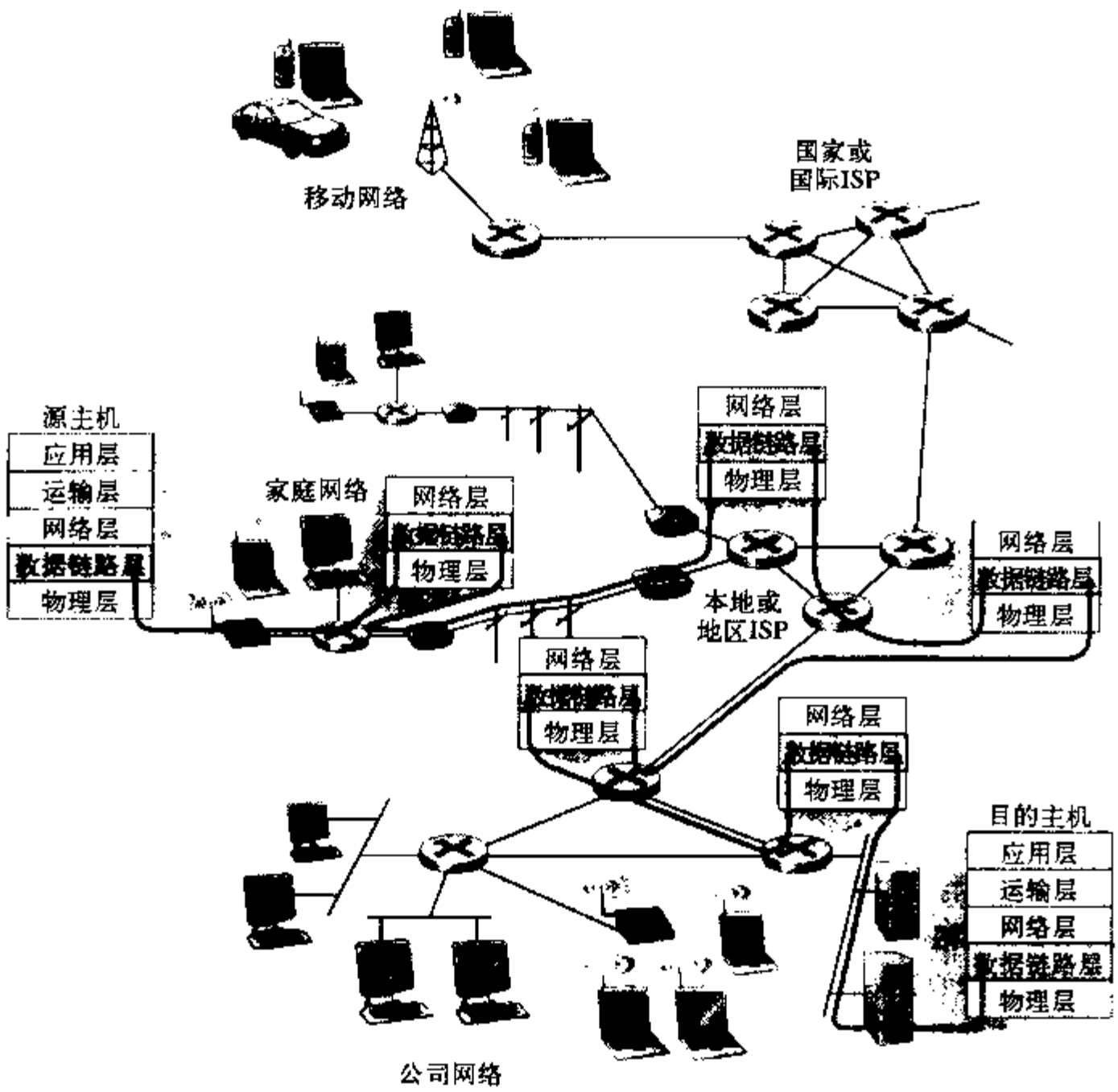


图5-1　链路层

在链路层的讨论中，我们将发现有两种截然不同类型的链路层信道。第一种类型由广播信道组成，这种信道常用在局域网（Local Area Network, LAN）、无线LAN、卫星网和混合光纤电缆（Hybrid Fiber-coaxial Cable, HFC）接入网中。对于广播信道，许多主机被连接到相

同的通信信道，需要所谓的媒体访问协议来协调传输和避免“碰撞”。第二种类型的链路层信道是点对点通信链路，例如两台路由器之间的通信链路或一个住宅的拨号调制解调器与一台ISP路由器之间的通信链路。协调对点对点链路的访问不是一件困难的事，但是仍然有一些重要的问题，如成帧、可靠数据传输、差错检测和流量控制。

我们也将在这一章中探讨几种重要的链路层技术。我们将深入研究以太网，这是目前最流行的有线LAN技术。我们还将学习点对点协议（PPP），这是拨号住宅主机选用的协议。

尽管WiFi及其更一般的无线LAN无疑属于链路层的主题，本章将不涉及它们。这不是因为WiFi是一个不重要的主题，恰恰相反，WiFi革命正在极大地改变着人们接入和使用因特网的方式。第6章将深入地研究WiFi，该章专注于无线计算机网络和移动性。

5.1 链路层：概述和服务

我们以学习一些有用的术语开始。在本章中为方便讨论，将主机和路由器均称为**节点**（node），因为我们很快将看到，我们将不特别关心一个节点是一台路由器还是一台主机。我们也将把沿着通信路径连接相邻节点的通信信道称为**链路**（link）。为了将一个数据报从源主机传输到目的主机，数据报必须通过沿端到端路径上的每段链路传输。在通过特定的链路时，传输节点将此数据报封装在链路层帧中，并将该帧发送到链路上；接收节点然后接收该帧并提取出数据报。

5.1.1 链路层提供的服务

链路层协议用来在独立的链路上移动数据报。**链路层协议**（link-layer protocol）定义了在链路两端的节点之间交互的分组格式，以及当发送和接收分组时这些节点采取的动作。第1章讲过，链路层协议交换的数据单元称为**帧**（frame），每个链路层帧通常封装了一个网络层的数据报。如我们不久将看到的那样，当发送和接收帧时，链路层协议所采取的动作包括差错检测、重传、流量控制和随机接入。链路层协议的例子包括以太网、802.11无线LAN（也称为WiFi）、令牌环和PPP；在很多场合下，ATM也被认为是链路层协议。我们将在本章的后半部分详细涉及这些协议中的许多协议。

虽然网络层的任务是将运输层报文段从源主机端到端地传送到目的主机，而链路层协议的任务是将网络层的数据报通过路径中的单段链路节点到节点地传送。链路层的一个重要特点是数据报在路径的不同链路上可能由不同链路层协议所承载。例如，数据报在第一段链路上可能由以太网承载，在最后一段链路上可能由PPP承载，而在中间的链路上由链路层WAN协议承载。需要着重注意的是，这些链路层协议提供的服务可能是不同的。例如，某个链路层协议可能提供可靠的交付，而另一些协议却可能不提供。因此，网络层在面对各段链路层提供的异构服务集合的情况下，必须能够完成它的端到端任务。

为了透彻理解链路层以及它是如何与网络层关联的，我们考虑一个交通运输的类比例子。假如一个旅行社计划为游客开辟从美国新泽西州的普林斯顿到瑞士洛桑的旅游路线。假定该旅行社认为对于游客而言最为便利的方案是：从普林斯顿乘豪华大轿车到JFK机场，然后乘飞机从JFK机场去日内瓦机场，最后乘火车从日内瓦机场到洛桑火车站。一旦该旅行社作了这3项预定，普林斯顿豪华大轿车公司将负责将游客从普林斯顿带到JFK，航空公司将负责将游客从JFK带到日内瓦，瑞士火车服务将负责将游客从日内瓦带到洛桑。该旅程中3段的每一段都

在两个“相邻”地点之间是“直达的”。注意这3段运输是由不同的公司管理，使用了完全不同的运输方式（豪华大轿车、飞机和火车）。尽管运输方式不同，它们都提供了将旅客从一个地点运输到相邻地点的基本服务。在这个运输类比中，一个游客好比一个数据报，每个运输区段好比一个通信链路，每种运输方式好比一种链路层协议，而该旅行社好比一个选路协议。

尽管任何链路层的基本服务都是将数据报通过单一通信链路从一个节点移动到相邻节点，所提供的服务细节将随链路层协议从一种进入下一种而改变。链路层协议能够提供的可能服务包括：

- *成帧*（framing）。几乎所有的链路层协议都在经链路传送之前，将每个网络层数据报用链路层帧封装起来。一个帧由一个数据字段和若干首部字段组成，其中网络层数据报就插在数据字段中。（一个帧也可能包括尾部字段，然而我们把首部字段和尾部字段合称为首部字段。）帧的结构由链路层协议规定。在本章的后半部分我们研究具体的链路层协议时，将看到几种不同的帧格式。
- *链路接入*（link access）。媒体访问控制（Medium Access Control, MAC）协议规定了帧在链路上传输的规则。对于在链路的一端有一个发送方、链路的另一端有一个接收方的点对点链路，MAC协议比较简单（或者不存在），即只要链路空闲，发送方都能够发送帧。更有趣的情况是多个节点共享单个广播链路，这就是所谓的多路访问问题。这里MAC协议用来协调多个节点的帧传输，我们在5.3节详细讨论MAC协议。
- *可靠交付*（reliable delivery）。当链路层协议提供可靠交付服务时，它保证无差错地经链路层移动每个网络层数据报。前面讲过，某些运输层协议（例如TCP）也提供可靠交付服务。与运输层可靠交付服务类似，链路层的可靠交付服务通常是通过确认和重传取得的（参见3.4节）。链路层可靠交付服务通常用于易产生高差错率的链路，例如无线链路，其目的是本地（也就是在差错发生的链路上）纠正一个差错，而不是通过运输层或应用层协议迫使进行端到端的数据重传。然而，对于低比特差错的链路，包括光纤、铜轴电缆和许多双绞铜线链路，链路层可靠交付可能会被认为是一种不必要的开销。由于这个原因，许多有线的链路层协议不提供可靠交付服务。
- *流量控制*（flow control）。链路每一端的节点都具有有限容量的帧缓存能力。当接收节点以比它能够处理的速率更快的速率接收分组时，这是一个潜在的问题。没有流量控制，接收方的缓存区就会溢出，并使帧丢失。与运输层相似，链路层协议能够提供流量控制，以防止链路一端的发送节点淹没链路另一端的接收节点。
- *差错检测*（error detection）。当帧中的一个比特作为1传输时，接收方节点可能错误地判断为0，反之亦然。这种比特差错是由信号衰减和电磁噪声导致的。因为没有必要转发一个有差错的数据报，所以许多链路层协议提供一种机制以检测是否存在一个或多个差错。通过让发送节点在帧中设置差错检测比特，让接收节点进行差错检测，以此来完成这项工作。差错检测在链路层协议中是一个非常普通的服务。第3章和第4章讲过，在因特网中运输层和网络层也提供了有限形式的差错检测（即互联网检验和）。链路层的差错检测通常更复杂，并且用硬件实现。
- *差错纠正*（error correction）。差错纠正和差错检测类似，区别在于接收方不仅能检测帧中是否引入了差错，而且能够准确地判决帧中的差错出现在哪里（并据此纠正这些差错）。某些协议（例如ATM）只为分组首部而不是整个分组提供链路层差错纠正。我们在5.2

节中讨论差错检测和纠正。

- 半双工和全双工（half-duplex and full-duplex）。采用全双工传输时，链路两端的节点可以同时传输分组。采用半双工传输时，一个节点不能同时进行传输和接收。

如上所述，链路层提供的许多服务与运输层提供的服务是非常相似的。例如，链路层和运输层都能提供可靠交付。尽管在这两层用于提供可靠传输的机制相似（见3.4节），但这两种可靠交付服务是不同的。运输层协议在端到端的基础上为两个进程之间提供可靠交付；可靠链路层协议在由单一链路相连的两个节点之间提供可靠交付服务。类似地，链路层和运输层协议都能提供流量控制和差错检测；此外，运输层协议中的流量控制是在端到端的基础上提供的，而链路层协议是在节点对相邻节点的基础上提供的。

5.1.2 链路层在何处实现

在深入学习链路层的细节之前，我们先来考虑在何处实现链路层的问题。这里我们将关注端系统，因为我们在第4章学习过在路由器中链路层是怎样实现在线路卡中的。主机的链路层是用硬件还是用软件实现的呢？它是实现在一块单独的卡上还是在一个芯片上的呢？它是怎样与主机的硬件和操作系统组件接口的呢？

图5-2显示了一个典型的主机体系结构。链路层的主体部分是在**网络适配器**（network adapter）中实现的，网络适配器也称为**网络接口卡**（Network Interface Card, NIC）。网络适配器的内核是链路层控制器，该控制器通常是实现了许多链路层服务的单个特定目的的芯片，这些服务（成帧、链路接入、流量控制、差错检测等）已在上节中讲过。因此，链路层控制器的许多功能是用硬件实现的。例如，Intel的8254x控制器[Intel 2006]实现了以太网协议，我们将在5.5节学习该协议；Atheros AR5006 [Atheros 2006]控制器实现了802.11 WiFi协议，我们将在6.3节学习该协议。直到20世纪90年代后期，大部分网络适配器还是物理上分离的卡（如一块PCMCIA卡或者插进PC机PCI卡槽中的一块插入卡），但现今越来越多的网络适配器被综合进主机的主板，即所谓“LAN在主板”配置。

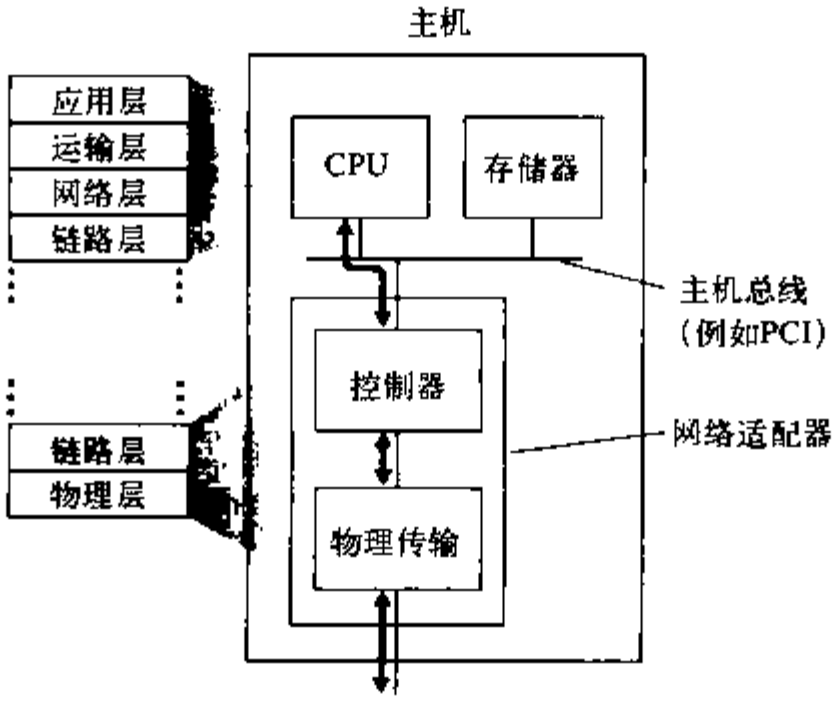


图5-2 网络适配器：它与其他主机组件及其对协议栈功能的关系

在发送方，控制器取得了由协议栈较高层生成并存储在主机内存中的数据报，在链路层帧中封装该数据报（填写该帧的各个字段），然后遵循链路接入协议将该帧传进通信链路中。

在接收端，控制器接收整个帧，提取出网络层数据报。如果链路层执行差错检测，则需要发送适配器在该帧的首部设置差错检测比特，以及接收适配器执行差错检测。如果该链路层执行流量控制，则发送控制器和接收控制器交换流量控制信息，使得发送方以接收方能够处理的速率发送帧。

图5-2显示了与一台主机总线（例如，一根PCI或PCI-X总线）相连的网络适配器，它看起来非常像对其他主机组件的任何其他I/O设备。图5-2也显示了尽管链路层的大部分是在接口卡的硬件上实现的，但该链路层的一部分是由运行在主机CPU上的软件实现的。链路层的软件组件通常实现了较高层次的链路层功能，如从网络层接收数据报，装配链路层寻址信息以及激活控制器硬件。在接收端，链路层软件响应来自控制器的中断（例如，因接收到一个或多个帧），处理差错情况，将数据报向上传递给网络层。因此，链路层是一种硬件和软件的结合体。[Intel 2006]从软件编程的视角提供了有关8254*x*控制器的可读性好的概述（以及详细的描述）。

图5-3显示了发送适配器和接收适配器。借助于由控制器实现的链路层协议的主要功能，这些适配器是半自治的单元，它们的任务是从一个适配器向另一个适配器传送帧。一些研究人员探讨了将（除了链路层处理以外的）更多的功能推进网络适配器的可能性。例如，8254*x*控制器能够用硬件来计算TCP/UDP检验和以及IP检验和，即由链路层控制器实现网络层和运输层功能。虽然这看起来严重地违反了分层的原则，但其优越性在于用硬件比用软件计算检验和要快得多，这种巨大的优势诱使人们可能忽略了这种违反原则的行为。[Mogul 2003]对在适配器上执行TCP处理的优缺点进行了有趣的讨论。[Kim 2005]探讨了在适配器上执行更高层功能（HTTP缓存）。

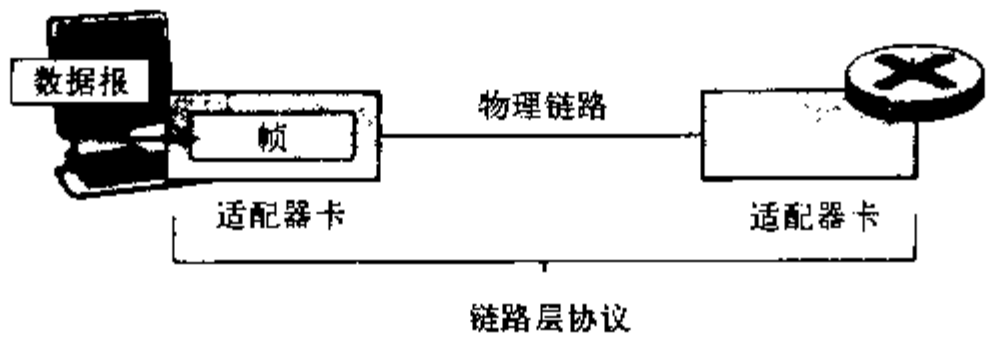


图5-3 通信链路的链路层协议在链路两端的适配器中实现

5.2 差错检测和纠错技术

在上一节中，我们提到了**比特级差错检测和纠错**（bit-level error detection and correction），它们通常是链路层提供的两种服务：对从一个节点发送到另一个物理上连接的邻近节点的链路层帧，检测和纠正其中的比特差错。我们在第3章中看到差错检测和纠错服务通常也由运输层提供。在本节中，我们将研究几种最简单的技术，它们能够用于检测比特差错，而且在某些情况下，能够纠正这种比特差错。对该主题理论和实现的全面描述是许多教科书的主题（例如[Schwartz 1980] 或[Bertsekas 1991]），我们这里的讨论简明扼要。这里我们的目的是对差错检测和纠错技术提供的能力有一个直观感觉，并看一下一些简单技术在链路层中的工作原理以及它是如何实际应用的。

图5-4举例说明了我们研究的环境。在发送节点，为了避免比特差错，使用差错检测和纠错比特（error-detection and-correction, EDC）来增强数据D。通常，要保护的数据不仅包括从

网络层传递下来需要通过链路传输的数据报，而且包括链路级的寻址信息、序列号和在链路帧首部中的其他字段。链路级帧中的*D*和*EDC*都被发送到接收节点。在接收节点，接收到比特序列*D'*和*EDC'*。注意到因传输中比特翻转所致，*D'*和*EDC'*可能与初始的*D*和*EDC*不同。

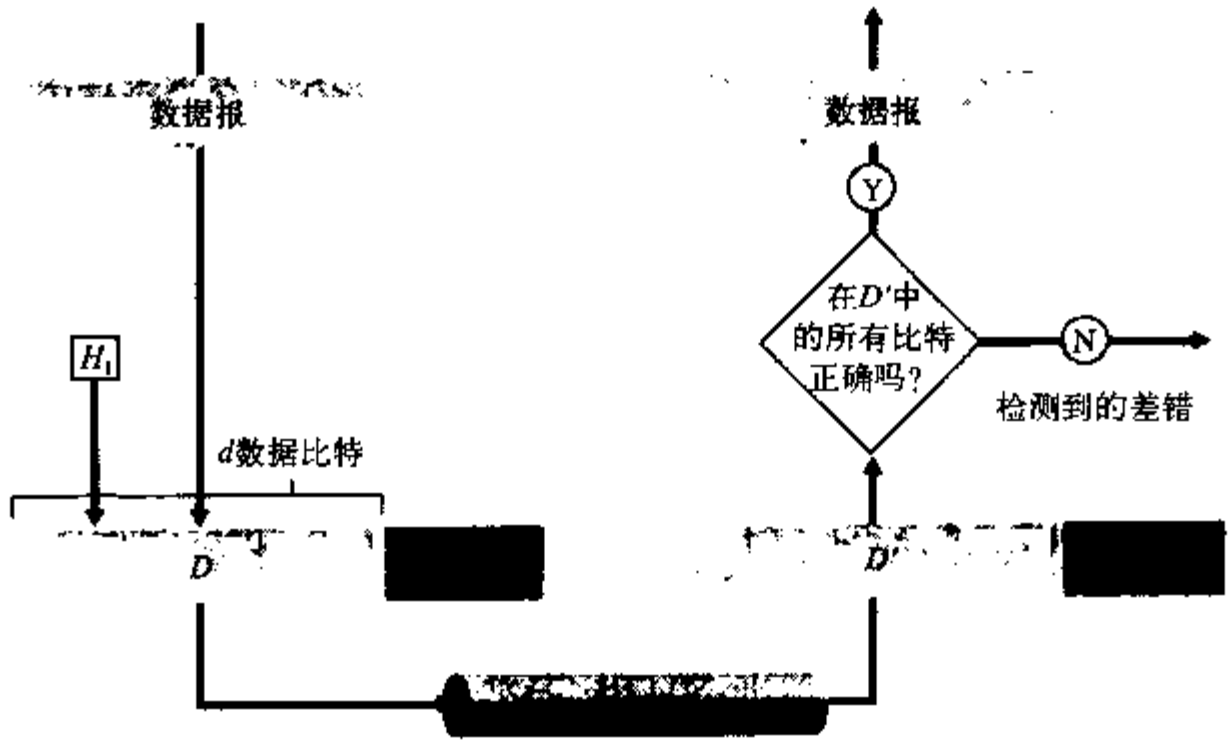


图5-4 差错检测与纠错的场景

接收方面临的挑战是，在它只收到*D'*和*EDC'*的情况下来判决*D'*是否和初始的*D*相同。图5-4中的接收方判决的准确措词（我们问的是是否检测到差错，而不是是否出现了差错！）是很重要的。差错检测和纠正技术有时使接收方检测到已经出现的比特差错，但并总是这样。即使采用差错检测比特，也还是可能有**未检出比特差错**（undetected bit error）的情况。这就是说，接收方可能无法知道接收的信息中包含着比特差错。因此，接收方可能向网络层交付一个损坏的数据报，或者不知道该帧首部的某个其他字段的内容已经损坏。因此我们要选择一个差错检测方案，使得这种事件发生的概率很小。一般而言，差错检测和纠错技术越复杂（即那些出现“未检测出比特差错”概率较小的技术），导致的开销越大，这就是意味着需要更多的计算量及传输更多的差错检测和纠错比特。

我们现在来研究在传输数据中检测差错的3种技术：奇偶校验（它用来描述差错检测和纠错隐含的基本思想）、检验和方法（它通常更多地应用于运输层）和循环冗余检测（它通常更多地应用在适配器中的链路层）。

5.2.1 奇偶校验

也许差错检测最简单的方式就是用单个的**奇偶校验位**（parity bit）。假设在图5-5中要发送的信息*D*有*d*个比特。在偶校验方案中，发送方只需包含一个附加的比特，选择它的值，使得这*d*+1个比特（初始信息加上一个校验比特）中1的总数是偶数。对于奇校验方案，选择校验比特值使得有奇数个1。图5-5描述了一个偶校验的方案，单个校验比特被存放在一个单独的字段中。

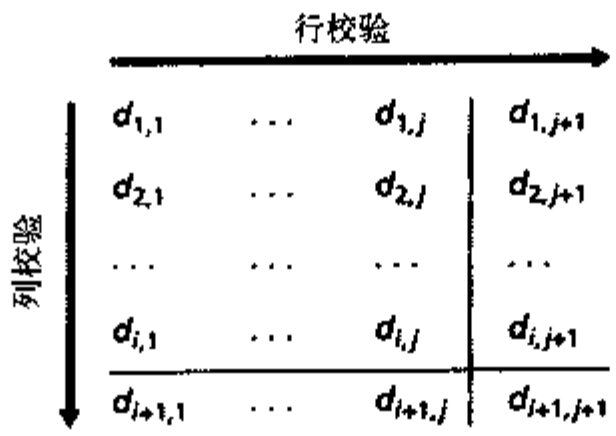
图5-5 1比特偶校验

采用单个校验比特方式，接收方的操作也很简单。接收方只需要数一数接收的d+1比特中

1的数目即可。如果在采用奇校验方案中发现了奇数个值为1的比特，接收方知道了至少出现了一个比特差错。更精确的说法是，出现了奇数个比特差错。

但是如果出现了偶数个比特差错，那会发生什么现象呢？你自己应该意识到这将导致一个未检出的差错。如果比特差错的概率小，而且比特之间的差错可以被看作是独立发生的，在一个分组中多个比特同时出错的概率会非常小。在这种情况下，单个奇偶校验位可能足够了。然而，有数据表明差错经常以“突发”方式聚集在一起，而不是独立地发生。在突发差错的情况下，使用单比特奇偶校验保护的一帧中未检测出差错的概率能够达到50%[Spragins 1991]。显然，需要一个更健壮的差错检测方案。（幸运的是实践中正在使用这样的方式！）但是在研究实际使用的差错检测方案之前，我们考虑对单比特奇偶校验的一种简单一般化方案，这将使我们深入地理解纠错技术。

图5-6显示了单比特奇偶校验方案的二维一般化方案。这里D中的d个比特被划分为i行j列。对每行和每列计算奇偶值。产生的$i + j + 1$奇偶比特构成了链路层帧的差错检测比特。



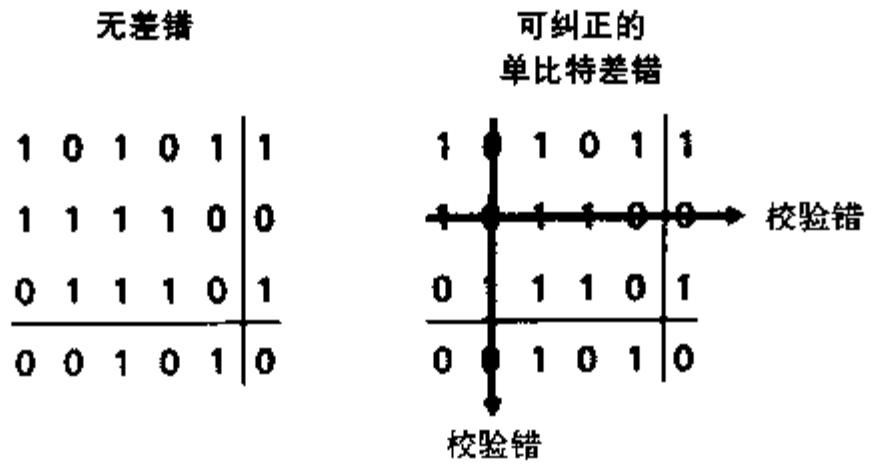


图5-6 二维偶校验

现在假设在初始的d比特信息中出现了单个比特差错。使用这种**二维奇偶校验**（two-dimensional parity）方案，包含比特值改变的列和行的校验值都将会出现差错。因此接收方不仅可以检测到出现了单个比特差错的事实，而且可以利用存在奇偶校验差错的列和行的下标来识别实际发生差错的比特并纠正它！图5-6显示了一个例子，其中位于（2,2）的值为1的比特损坏了，变成了0，该差错就是一个在接收方可检测并可纠正的差错。尽管我们的讨论是针对初始的d比特信息的，但奇偶校验位本身的单个比特差错也是可检测和可纠正的。二维奇偶校验也能够检测（但不能纠正！）一个分组中两个比特差错的任何组合。二维奇偶校验方案的其他特性将在本章后面的习题中进行探讨。

接收方检测和纠正差错的能力被称为**前向纠错**（Forward Error Correction，FEC）。这些

技术通常用于如音频CD这样的音频存储和回放设备中。在网络环境中，FEC技术可以单独应用，或与链路层ARQ技术一起应用，ARQ技术与我们在第3章研究的协议类似。FEC技术很有价值，因为它们可以减少所需的发送方重发的次数。也许更为重要的是，它们允许在接收方立即纠正差错。FEC避免了不得不等待的往返时延，而这些时延是发送方收到NAK分组并向接收方重传分组所需要的，这对于实时网络应用或者具有长传播时延的链路（如深空间链路）可能是一种非常重要的优点[Rubenstein 1998]。研究FEC在差错控制协议中的应用的资料包括[Biersack 1992; Nonnenmacher 1998; Byers 1998; Shacham 1990]。

5.2.2 检验和方法

在检验和技术中，图5-5中的d比特数据被认为是一个k比特整数序列。一个简单的检验和方法就是将这k比特整数加起来，并且用得到的和作为差错检测比特。**互联网检验和**（Internet checksum）就基于这种方法，即数据的两个字节作为16比特的整数对待并求和。这个和的反码形成了携带在报文段首部的互联网检验和。如在3.3节讨论的那样，接收方通过对接收的数据（包括检验和）的和取反码，并且检测其结果是否为全1比特来检测检验和。如果这些比特中有任何比特是0，就可以指示出差错。RFC 1071详细地讨论互联网检验和算法和它的实现。在TCP和UDP协议中，对所有字段（包括首部和数据字段）都计算互联网检验和。在其他协议中，例如XTP[Strayer 1992]，对首部计算一个检验和，对整个分组计算另一个检验和。

检验和方法需要相对小的分组开销。例如，TCP和UDP中的检验和只用了16比特。然而与后面要讨论的并常用于链路层的CRC相比，它们提供相对弱的差错保护。这时，一个很自然的问题是：为什么运输层使用检验和而链路层使用CRC呢？前面讲过运输层通常在主机中作为用户操作系统的一部分并用软件实现。因为运输层差错检测用软件实现，采用简单和快速如检验和这样的差错检测方案是重要的。在另一方面，链路层的差错检测在适配器中用专用的硬件实现，它能够快速地执行更复杂的CRC操作。Feldmeier [Feldmeier 1995]描述了加权检验和编码以及CRC（见下文）和其他编码的快速软件实现技术。

5.2.3 循环冗余检测

现今的计算机网络中广泛应用的差错检测技术是基于**循环冗余检测**（Cyclic Redundancy Check，CRC）**编码**。CRC编码也称为**多项式编码**（polynomial code），因为该编码能够将要发送的比特串看作是系数为0和1的一个多项式，对比特串的操作被解释为多项式算术。

CRC编码操作如下。考虑d比特的数据D，发送节点要将它发送给接收节点。发送方和接收方首先必须协商一个$r+1$比特模式，称为**生成多项式**（generator），我们将其表示为G。我们将要求G的最高有效位（最左边）的比特是1。CRC编码的关键思想如图5-7所示。对于一个给定的数据段D，发送方要选择r个附加比特R，并将它们附加到D上，使得得到的$d+r$比特模式（被解释为一个二进制数）用模2算术恰好能被G整除。用CRC进行差错检测的过程很简单：接收方用G去除接收到的$d+r$比特。如果余数为非零，接收方知道了出现差错；否则认为数据正确而被接受。

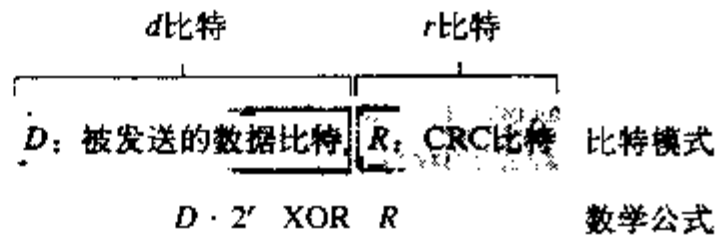


图5-7 CRC编码

所有CRC计算采用模2算术操作，在加法中不进位，在减法中不借位。这意味着加法和减法是相同的，而且这两种操作等价于操作数的按位异或（XOR）。因此，举例来说：

```
1011 XOR 0101 = 1110
1001 XOR 1101 = 0100
```

我们也同样有：

```
1011 - 0101 = 1110
1001 - 1101 = 0100
```

除了所需的加法或减法操作没有进位或借位外，乘法和除法与在二进制算术中是相同的。如在通常的二进制算术中那样，乘以2^k就是以一种比特模式左移k个位置。这样，给定D和R，$D \cdot 2^r$ XOR R产生如图5-7所示的$d+r$比特模式。在下面的讨论中，我们将利用图5-7中这种$d+r$比特模式的代数特性。

现在我们回到发送方怎样计算R这个关键问题上来。前面讲过，我们要求出R使得对于n有：

$$D \cdot 2^r \text{XOR } R = nG$$

也就是说，我们要选择R使得G能够除尽$D \cdot 2^r$ XOR R而没有余数。如果我们对上述等式的两边都用R异或（即用模2加，而没有进位），我们得到

$$D \cdot 2^r = nG \text{ XOR } R$$

这个等式告诉我们，如果我们用G来除$D \cdot 2^r$，余数值刚好是R。换句话说，我们可以这样计算R

$$R = \frac{D \cdot 2^r}{G} \text{的余数}$$

图5-8举例说明了在$D=101110$，$d=6$，$G=1001$和$r=3$的情况下的计算过程。在这种情况下传输的9个比特是101110011。你应该自行检查一下这些计算，并核对一下$D \cdot 2^r = 101011 \cdot G$ XOR R的确成立。

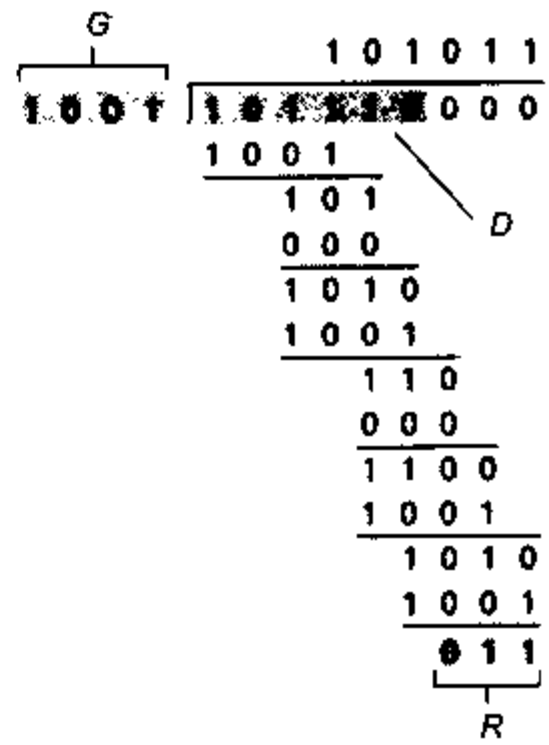


图5-8 一个简单的CRC计算

国际标准已经定义了8、12、16和32比特生成多项式G。一个8比特的CRC用于保护ATM信元中的5 字节的首部。CRC-32 32比特的标准被多种链路级IEEE协议采用，使用的一个生成多项式是：

$$G_{\text{CRC-32}}=100000100110000010001110110110111$$

每个CRC标准都能检测小于$r+1$比特的突发差错（这意味着所有连续的r比特或者更少的差错都可以检测到）。此外，在适当的假设下，长度大于$r+1$比特的突发差错以概率$1-0.5^r$被检测到。每个CRC标准也都能检测任何奇数个比特差错。有关CRC检测实现的讨论可参见[Williams 1993]。CRC编码以及更强的编码依据的理论超出了本文的范围，教科书[Schwarz 1980]对这个主题提供了很好的介绍。

5.3 多路访问协议

在本章的介绍中，我们提到了两种类型的网络链路：点对点链路和广播链路。**点对点链路**（point-to-point link）是由链路一端的单个发送方和链路另一端的单个接收方组成。许多链路层协议都是为点对点链路设计的；点对点协议（Point-to-Point Protocol, PPP）和高级数据链路控制（High-level Data Link Control, HDLC）是两种这样的协议，我们将在本章后面要讨

论它们。第二种类型的链路是**广播链路**（broadcast link），它能够让多个发送和接收节点都连接到相同的、单一的、共享的广播信道上。这里使用术语“广播”是因为当任何一个节点传输一个帧时，该信道广播该帧，每个其他节点都收到一个拷贝。以太网和无线LAN是广播链路层技术的例子。在本节，我们暂缓讨论特定的链路层协议，而先研究一个对链路层很重要的问题：如何协调多个发送和接收节点对一个共享广播信道的访问，这就是**多路访问问题**（multiple access problem）。广播信道通常用于LAN中，LAN是一个在地理上集中于一座建筑物中（或者在一个公司和大学校园）的网络。因此我们也将在本节后面考察一下多路访问信道是如何在LAN中使用的。

我们都很熟悉广播的概念，因为自电视发明以来就使用了这种通信方式。但是传统的电视是一种一个方向的广播（即一个固定的节点向许多接收节点传输），而计算机网络广播信道上的节点能够发送和接收。也许对广播信道的一个更有人情味的类比是鸡尾酒会，在那里许多人聚集在一个大房间里（空气为提供广播的媒体）谈论和倾听。第二个切题的类比是许多读者都很熟悉的地方，即一间教室，在那里老师和同学（们）类似地共享相同的、单一的广播媒体。在这两种场景下一个中心问题是确定谁在什么时候获得说话权力（也就是向信道传输）。作为人类，为了共享这种广播信道，我们已经演化得到了一个精心设计的协议集了：

“给每个人一个讲话的机会。”

“该你讲话时你才说话。”

“不要一个人垄断整个谈话。”

“如果有问题请举手。”

“当有人讲话时不要打断。”

“当其他人讲话时不要睡觉。”

计算机网络有类似的协议，也就是所谓的**多路访问协议**（multiple access protocol），即节点通过这些协议来规范它们在共享的广播信道上的传输行为。如图5-9所示，在各种各样的网络环境下需要多路访问协议，包括有线和无线局域网，以及卫星网络。尽管从技术上讲每个节点通过它的适配器访问广播信道，在本节中我们将把节点作为发送和接收设备。在实际中，成百甚至成千个节点能够通过一个广播信道直接通信。

因为所有的节点都能够传输帧，两个以上的节点可能会同时传输帧。当发生这种情况时，所有节点同时接到多个帧，也就是说，传输的帧在所有的接收方处**碰撞**（collide）了。通常，当碰撞发生时，接收节点没有一个能够获得任何有效的传输帧；在某种意义下，碰撞帧的信号纠缠在一起。因此，涉及此次碰撞的所有帧都丢失了，在碰撞时间间隔中的广播信道被浪费了。显然，如果许多节点要频繁地传输帧，许多传输将导致碰撞，广播信道的大量带宽将被浪费掉。

当多个节点处于活跃状态时，为了确保广播信道执行有用的工作，以某种方式协调活跃节点的传输是必要的。这种协调工作由多路访问协议负责。在过去的30多年中，已经有上千篇文章和上百篇博士论文研究过多路访问协议；有关这部分工作的一个内容丰富的综述是[Rom 1990]。此外，由于新类型链路尤其是新的无线链路不断出现，在多路访问协议方面的积极研究仍在继续。

这些年来，在大量的链路层技术中已经实现了几十种多路访问协议。尽管如此，我们能够将任何多路访问协议划分为3种类型之一：**信道划分协议**（channel partitioning protocol）、

随机接入协议（random access protocol）和**轮流协议**（taking-turns protocol）。我们将在后续的3个小节中讨论这几类多路访问协议。

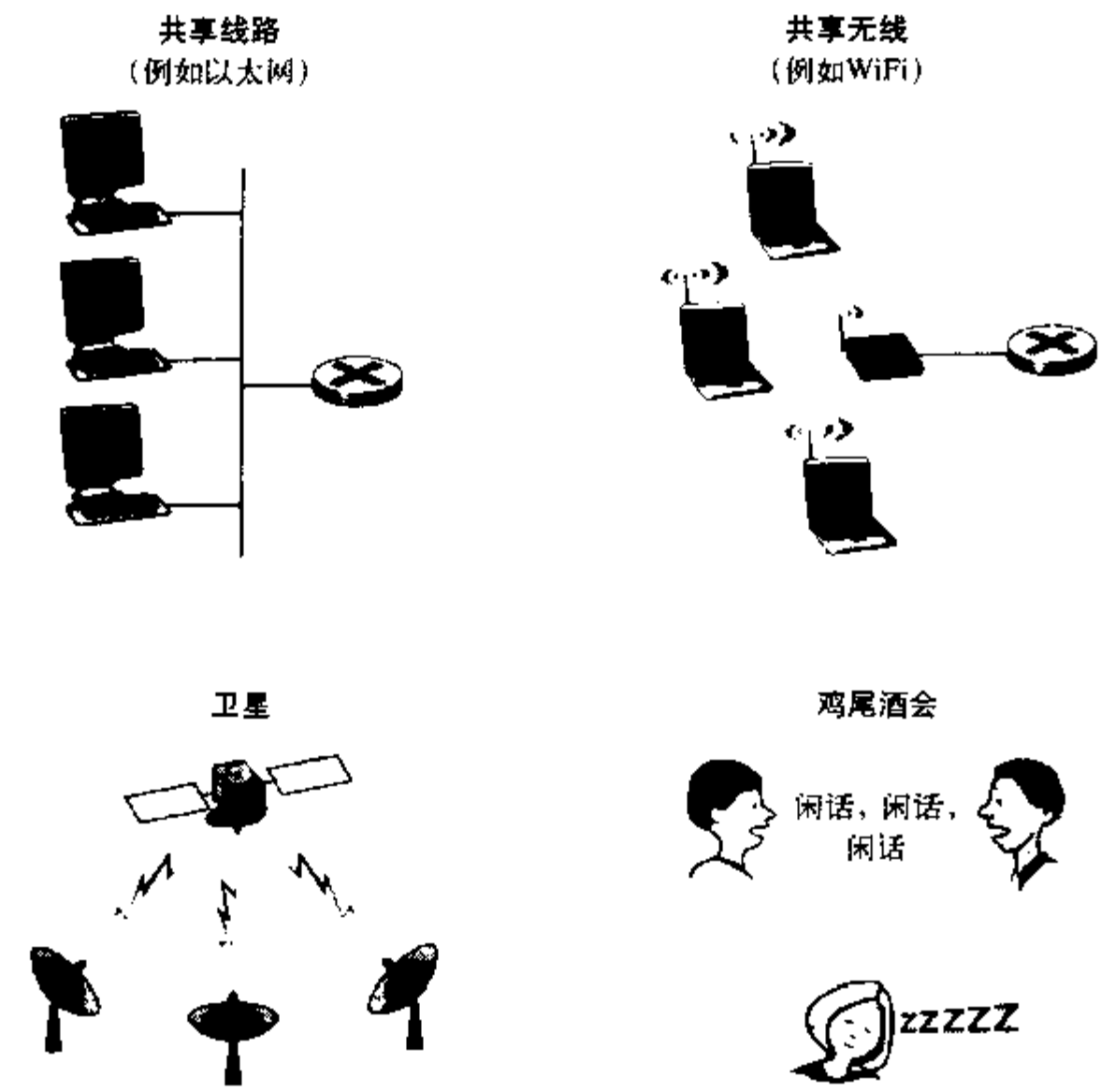


图5-9 各种多路访问信道

我们对这个概述小结如下，在理想情况下，对于速率为每秒R比特的广播信道，多路访问协议应该有以下所希望的特性：

1）当只有一个节点有数据发送时，该节点具有R bps的吞吐量。

2）当有M个节点要发送数据时，每个节点吞吐量为R/M bps。不必要求M个节点中的每一个节点总是有R/M的瞬间速率，而是每个节点在一些适当定义的时间间隔内应该有R/M的平均传输速率。

3）协议是分散的，这就是说不会因某主节点故障而使整个系统崩溃。

4）协议是简单的，使得实现起来代价不是很高。

5.3.1 信道划分协议

我们前面在1.3节讨论过，时分多路复用（TDM）和频分多路复用（FDM）是两种能够在所有共享信道节点之间用于划分广播信道带宽的技术。举例来说，假设一个支持N个节点的信道，信道的传输速率为R bps。TDM将时间划分为**时间帧**（time frame），并进一步划分每个时间帧为N个**时隙**（slot）。（不应当把TDM时间帧与在发送和接收适配器之间交换的链路层数据单元相混淆，后者也被称为帧。为了减小混乱，在本小节中我们将链路层交换的数据单元称为分组。）然后把每个时隙分配给N个节点中的一个。无论何时某个节点在有分组要发送的时候，它在循环的TDM帧中在指定的时隙中传输其分组比特。通常，时隙长度的选择通常使得

在一个时隙内能够传输单个分组。图5-10表示一个简单的4个节点的TDM例子。再回到我们的鸡尾酒会类比中，一个采用TDM规则的鸡尾酒会将允许每个聚会客人在固定的时间段发言，然后再允许另一个聚会客人发言同样时长，如此类推。一旦每个人都有了说话机会，将不断重复着这种模式。

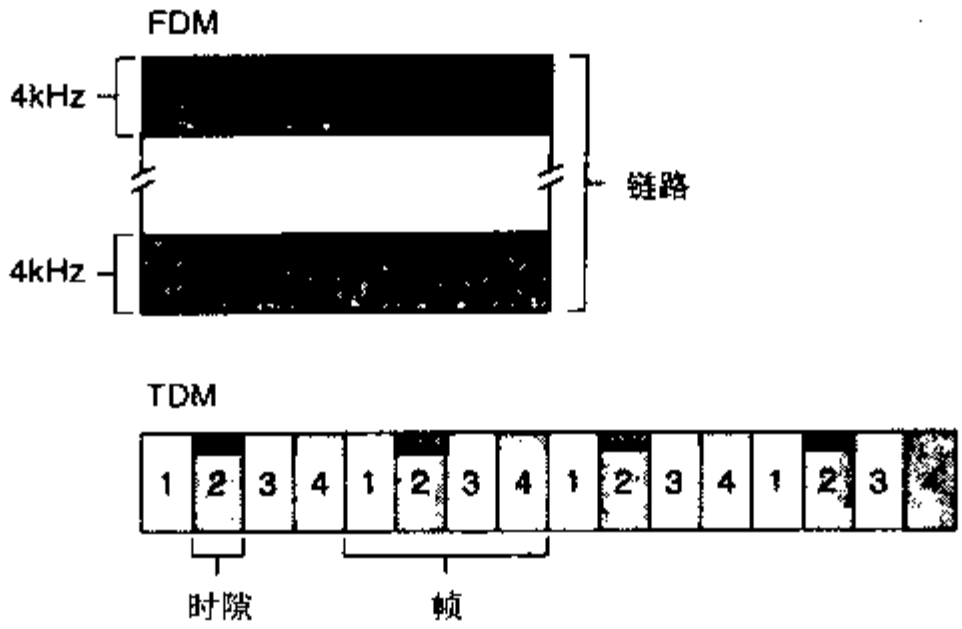


图5-10 一个4节点的TDM与FDM的例子

TDM是很有吸引力的，因为它消除了碰撞，而且非常公平：每个节点在每个帧时间内得到了专用的传输速率R/N bps。然而它有两个主要缺陷。首先，节点被限制于R/N bps的平均速率，即使它是唯一有分组要发送的节点。第二个缺点是节点必须总是等待它在传输序列中的轮次，即使它是唯一一个有帧要发送的节点。想象一下某聚会客人是唯一一个有话要说的人（并且想象一下更少见的情况，酒会上所有的人都想听某个人说些什么）的情形。显然，将一种多路访问协议用于这个特殊聚会，TDM是一种很糟的选择。

TDM在时间上共享广播信道，而FDM将R bps信道划分为不同的频段（每个频段具有R/N带宽），并把每个频率分配给N个节点中的一个。因此FDM在单个较大的R bps信道中创建了N个较小的R/N bps信道。FDM和TDM的优缺点一样。它避免了碰撞，在N个节点之间公平地划分了带宽。然而，FDM也有TDM所具有的主要缺点，也就是限制一个节点只能使用R/N的带宽，即使当它是唯一一个有分组要发送的节点时。

第三种信道划分协议是**码分多址**（Code Division Multiple Access，CDMA）。TDM和FDM分别为节点分配时隙和频率，而CDMA对每个节点分配一种不同的*编码*。然后每个节点用它唯一的编码来对它发送的数据进行编码。如果精心选择这些编码，CDMA网络具有一种奇妙的特性，即不同的节点能够同时传输，并且它们各自相应的接收方仍然正确接收发送方编码后的数据比特（假设接收方知道发送方的编码），而不在乎其他节点的干扰传输。CDMA已经在军用系统中使用了一段时间（由于它的抗干扰特性），目前已经广泛应用于民用，尤其是用于蜂窝电话中。因为CDMA的使用与无线信道紧密相关，我们将把有关CDMA技术细节的讨论留到第6章。现在，我们知道CDMA编码类似于TDM中的时隙和FDM中的频率，能分配给多路访问信道的用户，我们了解这些就足够了。

5.3.2 随机接入协议

第二大类多路访问协议是随机接入协议。在随机接入协议中，一个传输节点总是以信道的全部速率（即R bps）进行发送。当有碰撞时，涉及碰撞的每个节点反复地重发它的帧（也就是分组），直到该帧无碰撞地通过为止。但是当一个节点经受一次碰撞时，它不必立刻重发该帧。相反，它在重发该帧之前等待一个随机时延。涉及碰撞的每个节点独立地选择随机时延。因为该随机时延是独立选择的，下述现象有可能发生：这些节点之一所选择的时延远远小于其他碰撞节点的时延，并因此能够无碰撞地将它的帧在信道中发出。

文献中描述的随机接入协议不说有上百种也有几十种[Rom 1990; Bertsekas 1991]。在本节中，我们将描述一些最常用的随机接入协议，即ALOHA协议[Abramson 1970；Abramson 1985]和载波侦听多路访问（CSMA）协议[Kleinrock 1975b]。然后，在5.5节中，我们将介绍以太网的细节[Metcalfe 1976]，这是一种流行并广泛部署的CSMA协议。

1. 时隙ALOHA

我们以最简单的随机接入协议之一——时隙ALOHA协议，开始我们对随机接入协议的研究。在我们对时隙ALOHA的描述中，我们做下列假设：

- 所有帧由恰好L比特组成。
- 时间被划分为长度为L/R s的时隙（这就是说，一个时隙等于传输一帧的时间）。
- 节点只在时隙起点开始传输帧。
- 节点是同步的，每个节点都知道时隙何时开始。
- 如果在一个时隙中有两个或者更多个帧碰撞，则所有节点在该时隙结束之前检测到该碰撞事件。

令p是一个概率，即一个在0和1之间的数。每个节点的时隙ALOHA操作是简单的：

- 当该节点有一个新帧要发送时，它等到下一个时隙开始并在该时隙传输整个帧。
- 如果没有碰撞，该节点成功地传输它的帧，从而不需要考虑重传该帧。（如果该节点有新帧要传输，它能够为传输准备新帧。）
- 如果有碰撞，该节点在该时隙结束之前检测到这次碰撞。该节点以概率p在后续的每个时隙中重传它的帧，直到该帧被无碰撞地传输出去。

以概率p重传，我们是指某节点有效地投掷一个有偏倚的硬币，硬币正面事件对应着重传，这个事件以概率p出现。硬币反面事件对应着“跳过这个时隙，在下个时隙再掷硬币”，这个事件以概率（$1-p$）出现。所有涉及碰撞的节点独立地投掷它们的硬币。

时隙ALOHA看起来有很多优点。与信道划分不同，当某节点是唯一活跃的节点（一个节点如果有帧要发送就认为它是活跃的）时，时隙ALOHA允许该节点以全速R连续传输。时隙ALOHA也是高度分散的，因为每个节点检测碰撞并独立决定什么时候重传。（然而，时隙ALOHA的确需要在节点中对时隙同步；我们很快将讨论ALOHA协议的一个不分时隙的版本以及CSMA协议，这两种协议都不需要这种同步，因此是完全分散的。）时隙ALOHA也是一个极为简单的协议。

当只有一个活跃节点时，时隙ALOHA工作出色，但是当有多个活跃节点时效率又将如何呢？这里有两个可能要考虑的效率问题。首先，如图5-11所示，当有多个活跃节点时，一部分时隙将有碰撞，因此将被“浪费”掉了。第二个考虑是，时隙的另一部分将是空闲的，因为所有活跃节点由于概率传输策略会节制传输。唯一“未浪费的”时隙是那些刚好有一个节

点传输的时隙。刚好有一个节点传输的时隙称为一个**成功时隙**（successful slot）。时隙多路访问协议的**效率**（efficiency）被定义为当有大量的活跃节点且每个节点总有大量的帧要发送时，长期运行中成功时隙的份额。注意到如果不使用某种形式的访问控制，而且每个节点都在每次碰撞之后立即重传，这个效率将为零。时隙ALOHA显然增加了它的效率，使之大于零，但是效率增加了多少呢？

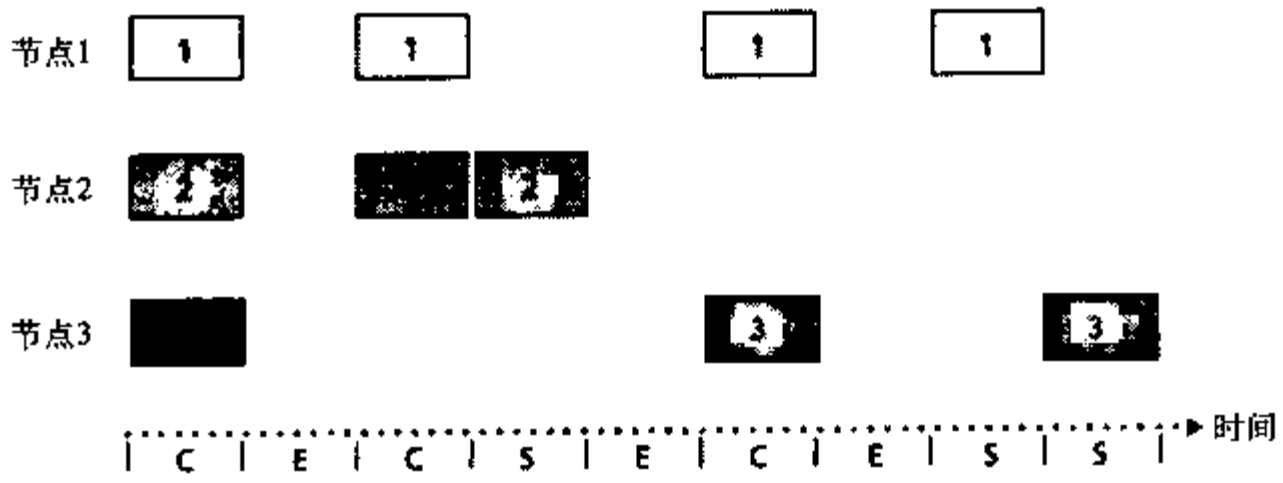


图5-11 节点1、2和3在第一个时隙碰撞。节点2最终在第4个时隙成功，节点1在第8个时隙成功，节点3在第9个时隙成功

现在我们继续概述时隙ALOHA最大效率的推导过程。为了保持推导简单，我们对协议做了一点修改，假设每个节点试图在每个时隙以概率p传输一帧（也就是说，我们假设每个节点总有一帧要发送，而且节点对一个新帧和已经遭受一次碰撞的帧都以概率p传输）。假设有N个节点。则一个给定时隙是成功时隙的概率是节点之一传输而余下的$N-1$个节点不传输的概率。一个给定节点传输的概率是p；剩余节点不传输的概率是$(1-p)^{N-1}$。因此，一个给定的节点成功传送的概率是$p(1-p)^{N-1}$。因为有N个节点，任意一个节点成功传送的概率是$Np(1-p)^{N-1}$。

因此，当有N个活跃节点时，时隙ALOHA的效率是$Np(1-p)^{N-1}$。为了获得N个活跃节点的最大效率，我们必须求出使这个表达式最大化的p^*。（对这个推导的一个大体描述参见课后习题。）而且为了获得大量活跃节点的最大效率，当N趋近于无穷时，我们取$Np^*(1-p^*)^{N-1}$的极限。（同样参见课后习题。）在完成这些计算之后，我们会发现这个协议的最大效率为1/e=0.37。这就是说，当大量节点有很多帧要传输时，则（最多）仅有37%的时隙做有用的工作。因此该信道有效传输速率不是R bps，而仅为0.37R bps！相似的分析还表明37%的时隙是空的，26%的时隙有碰撞。试想一个蹩脚的网络管理员购买了一个100 Mbps的时隙ALOHA系统，希望能够使用网络在大量的用户之间以总计速率如80 Mbps来传输数据。尽管这个信道能够以信道的全速100 Mbps传输一个给定的帧，但从长时间范围看，该信道的成功吞吐量将小于37 Mbps。

2. ALOHA

时隙ALOHA协议要求所有的节点同步它们的传输，以在每个时隙开始处开始传输。第一个ALOHA协议[Abramson 1970]实际上是一个非时隙、完全分散的协议。在纯ALOHA中，当一帧首次到达（即一个网络层数据报在发送节点从网络层传递下来），节点立刻将该帧完整地传输进广播信道。如果传输的一个帧与一个或多个传输经受了碰撞，这个节点将立即（在完

全传输完它的碰撞帧之后）以概率p重传该帧。否则，该节点等待一个帧传输时间。在此等待之后，它则以概率p传输该帧，或者以概率$1-p$等待（保持空闲）另一个帧时间。

为了确定纯ALOHA的最大效率，我们关注某个单独的节点。我们的假设与在时隙ALOHA分析中所做的相同，采用帧传输时间为时间单元。在任何给定时间，某节点传输一个帧的概率是p。假设该帧在时刻t_0开始传输。如图5-12所示，为了使这帧能成功地传输，在时间间隔$[t_0-1, t_0]$中不能有其他节点开始传输。其他节点的传输将与节点i的帧传输起始相重叠。所有其他节点在这个时间间隔不开始传输的概率是$(1-p)^{N-1}$。类似地，当节点i在传输时，其他节点不能开始传输，因为这种传输将与节点i传输的后面部分相重叠。所有其他节点在这个时间间隔不开始传输的概率也是$(1-p)^{N-1}$。因此，一个给定的节点成功传输一次的概率是$p(1-p)^{2(N-1)}$。通过与时隙ALOHA情况一样来取极限，我们求得纯ALOHA协议的最大效率仅为$1/(2e)$，这刚好是时隙ALOHA的一半。这就是完全分散的ALOHA协议所要付出的代价。

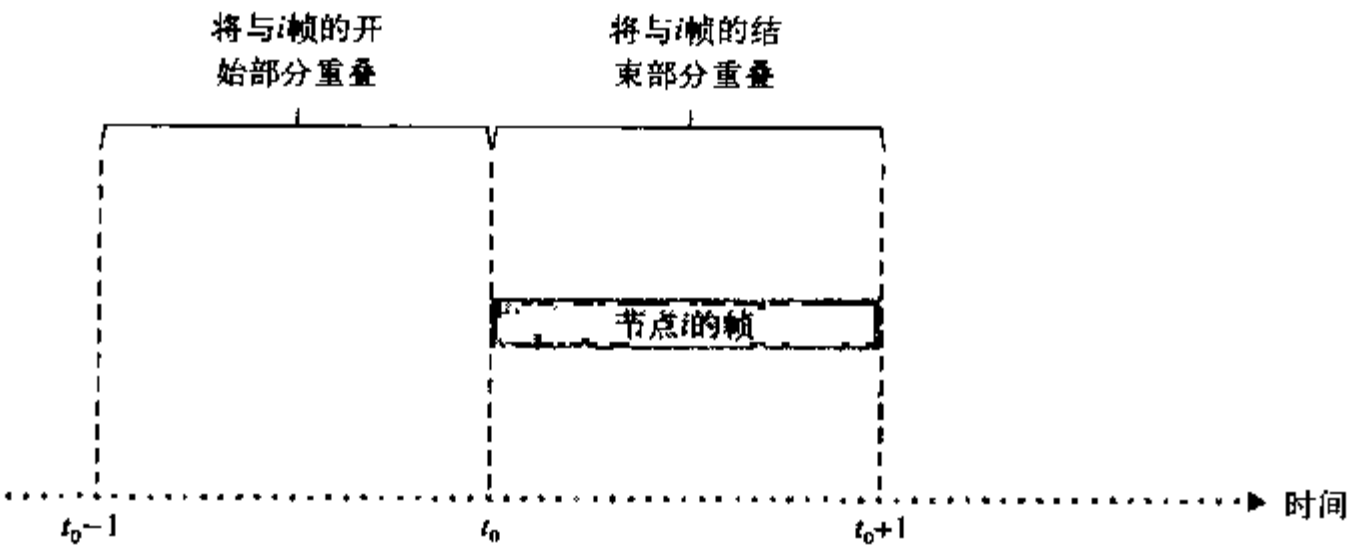


图5-12 纯ALOHA中的干扰传输

历史事件

Norm Abramson和ALOHAnet

Norm Abramson是一名具有博士学位的工程师，对冲浪运动很有激情，而且对分组交换也很感兴趣。这些兴趣的结合于1969年把他带到了夏威夷大学。夏威夷是由许多多山的岛屿组成的，安装和运营基于陆地的网络是很困难的。当不冲浪的时候，Abramson思考设计一种能在无线信道上完成分组交换的网络。他设计的网络有一个中心主机和几个分散在夏威夷群岛上的二级节点。该网络有两个信道，每个信道使用不同的频带。下行链路信道从中心主机向二级主机广播分组；上行信道从二级主机向中心主机发送分组。除了发送信息分组，中心主机还在下行信道上向从二级主机每个成功接收的分组发送确认。

因为二级主机以分散的方式传输分组，在上行信道上出现碰撞是不可避免的。这个观察导致Abramson设计了如本章所描述的纯ALOHA协议。在1970年，通过不断从ARPA获得资助，Abramson将他的ALOHAnet与ARPAnet连接起来。Abramson的工作是很重要的，不仅因为它是无线分组网络的第一个例子，而且因为它激励了Bob Metcalfe。几年之后，Metcalfe修改了ALOHA协议，创造了CSMA/CD协议和以太网LAN。

3. 载波侦听多路访问（CSMA）

在时隙和纯ALOHA中，一个节点传输的决定独立于连接到这个广播信道上的其他节点的活动。特别是，当一个节点既不关心在它开始传输时是否有其他节点碰巧在传输，也不会在有另一个节点开始干扰它的传输时停止传输。在我们的鸡尾酒会类比中，ALOHA协议非常像一个粗野的聚会客人，他喋喋不休地讲话而不顾是否其他人在说话。人类的交流协议，要求我们不仅要更为礼貌，而且要减少在谈话中与他人“碰撞”的时间，从而增加了我们谈话中交流的数据数量。具体而言，有礼貌的人类谈话有两个重要的规则：

- 说话之前先听。如果其他人正在说话，等到他们说完话为止。在网络领域中，这被称为**载波侦听**（carrier sensing），即一个节点在传输前先听信道。如果来自另一个节点的帧正向信道上发送，节点则等待（“后退”）一段随机时间，然后再侦听信道。如果侦听到该信道是空闲的，该节点则开始帧传输。否则，该节点等待另一段随机时间，继续重复这个过程。
- 如果与他人同时开始说话，停止说话。在网络领域中，这被称为**碰撞检测**（collision detection），即一个传输节点在传输时一直在侦听信道。如果它检测到另一个节点正在传输干扰帧，它就停止传输，用某个协议来确定它应该在什么时候再尝试下一次传输。

这两个规则包含在**载波侦听多路访问**（Carrier Sense Multiple Access，CSMA）和具有碰撞检测的CSMA（CSMA with Collision Detection，CSMA/CD）协议族中[Kleinrock 1975b; Metcalfe 1976; Lam 1980; Rom 1990]。人们已经提出了CSMA和CSMA/CD的许多变种。读者能够查阅这些参考文献以得到有关这些协议的具体细节。我们将在5.5节中详细研究在以太网中使用的CSMA/CD方案。这里，我们将考虑一些CSMA和CSMA/CD最重要的、基本的特性。

你可能要问的关于CSMA的第一个问题是，如果所有的节点都进行载波侦听了，为什么首先会发生碰撞？毕竟，某节点无论何时侦听到另一个节点在传输，它都会停止传输。对于这个问题的答案最好能够用时空图来说明[Molle 1987]。图5-13显示了连接到一个线状广播总线的4个节点（A、B、C、D）的时空图。横轴表示每个节点在空间的位置；纵轴表示时间。

在时刻t_0，节点B侦听到信道是空闲的，因为当前没有其他节点在传输。因此节点B开始传输，沿着广播媒体在两个方向上传播它的比特。图5-13中B的比特随着时间的增加向下传播，这表明B的比特沿着广播媒体传播实际需要的时间不是零（虽然以接近光的速度传播）。在时刻t_1（$t_1>t_0$），节点D有一个帧要发送。尽管节点B在时刻t_1正在传输，但B传输的比特还没有到达D，因此D在t_1侦听到信道空闲。根据CSMA协议，从而D开始了传输帧。一个短暂的时间之后，B的传输开始在D干扰D的传输。从图5-13中可以看出，显然广播信道的端到端**信道传播时延**（channel propagation delay）（信号从一个节点传播到另一个节点所花费的时间）在决定其性能上起着关键的作用。该传播时延越长，载波侦听节点不能侦听到网络中另一个节点已经开始传输的机会越大。

在图5-13中，节点没有进行碰撞检测，即使已经出现了碰撞，B和D都将继续完整地传输它们的帧。当某节点执行碰撞检测时，一旦它检测到碰撞将立即停止传输。图5-14表示了和图5-13相同的情况，只是这两个节点在检测到碰撞后很短的时间内都放弃了它们的传输。显然，在多路访问协议中加入碰撞检测，通过不去传输一个无用的、（被来自另一个节点的帧干扰）损坏的帧，将有助于改善协议的性能。我们在5.5节要研究的以太网协议是使用具有碰撞检测的CSMA协议。

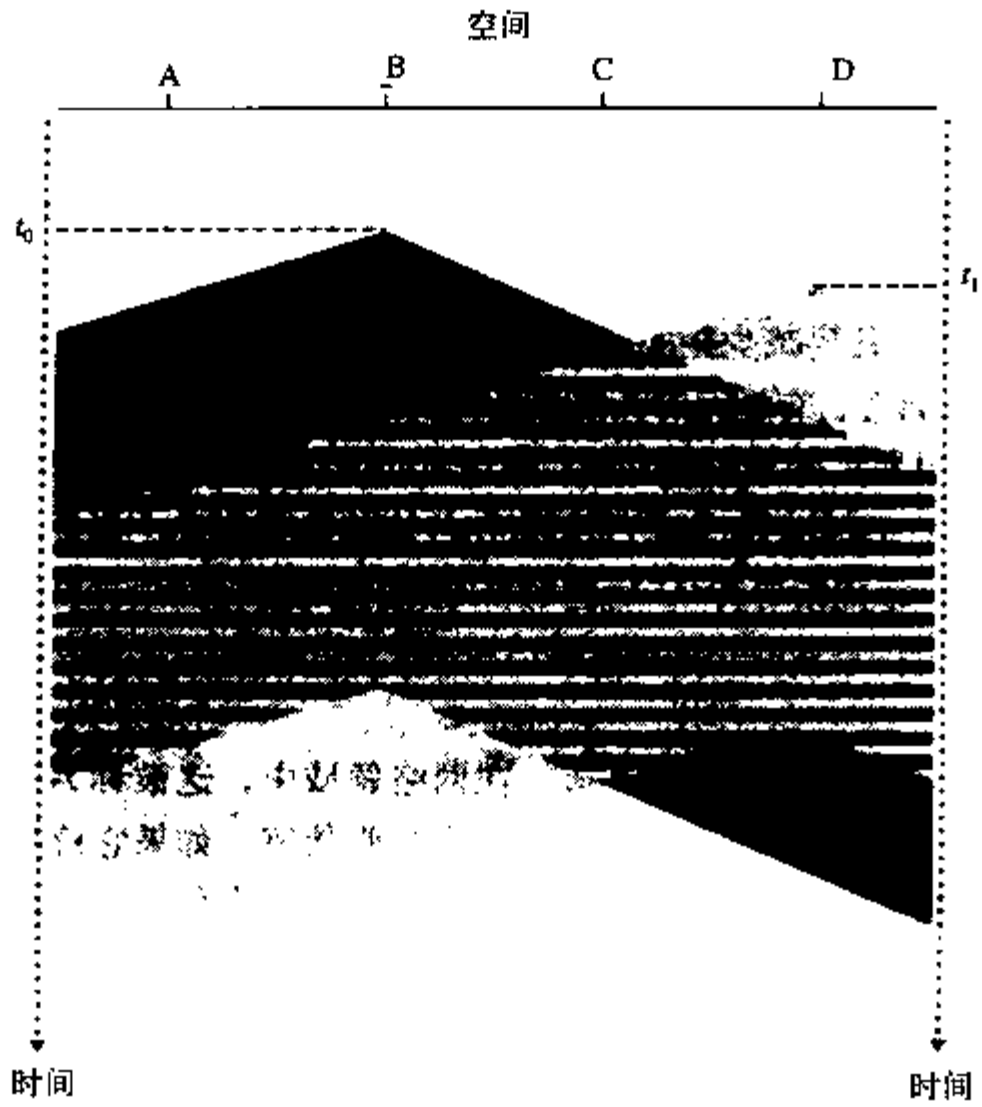


图5-13 发生碰撞传输的两个CSMA节点的时空图

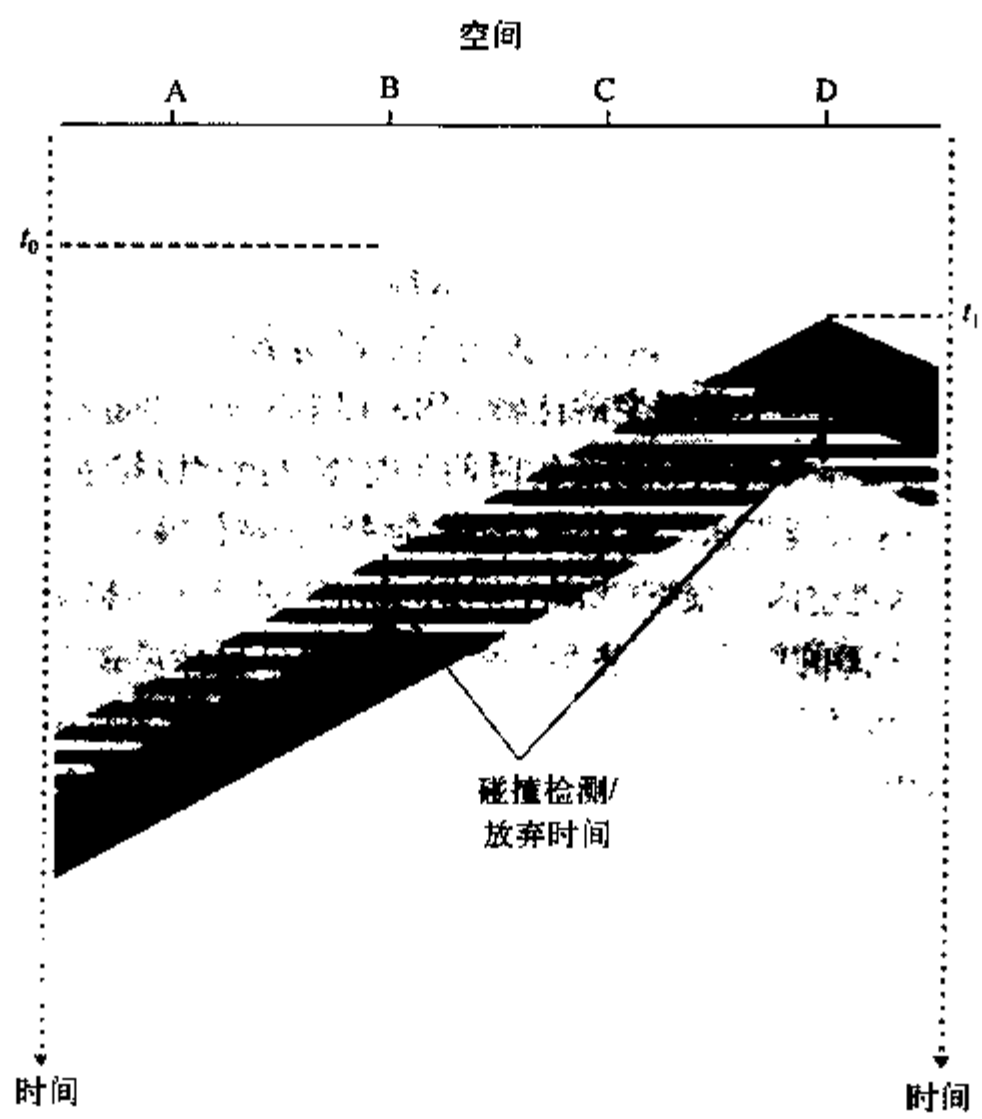


图5-14 具有碰撞检测的CSMA

5.3.3 轮流协议

前面讲过多路访问协议的两个理想特性是：① 当只有一个节点是活跃的，该活跃节点具有R bps的吞吐量；② 当M个节点是活跃的，每个活跃节点的吞吐量接近R/M bps。ALOHA和CSMA协议具有第一个特性，但不具有第二个特性。这激发研究人员创造另一类协议，也就是**轮流协议**（taking-turns protocol）。和随机接入协议一样，有几十种轮流协议，其中每一个协议都有很多变种。这里我们要讨论两种比较重要的协议。第一种是**轮询协议**（polling protocol）。轮询协议要求这些节点之一要被指定为主节点。主节点以循环的方式**轮询**（poll）每个节点。特别是，主节点首先向节点1发送一个报文，告诉它（节点1）能够传输的最大帧数。在节点1传输了某些帧后，主节点告诉节点2能够传输的最大帧数。（主节点能够通过观察在信道上是否缺乏信号，来决定一个节点何时完成了帧的发送。）以这种方式继续上述过程，主节点以循环的方式轮询每个节点。

轮询协议消除了困扰随机接入协议的碰撞和空时隙，这使得轮询取得高得多的效率。但是它也有一些缺点。第一个缺点是该协议引入了轮询时延，即通知一个节点它可以传输所需的时间。例如，如果只有一个节点是活跃的，那么这个节点将以小于R bps的速率传输，因为每次活跃节点发送了它的最大数量帧时，主节点必须依次轮询每一个非活跃的节点。第二个缺点可能更为严重，就是如果主节点有故障，整个信道都变得不可操作。

第二种轮流协议是**令牌传递协议**（token-passing protocol）。在这种协议中没有主节点。一个小的称为令牌（token）的特殊目的帧在节点之间以某种固定的次序进行交换。例如，节点1可能总是把令牌发送给节点2，节点2可能总是把令牌发送给节点3，而节点N可能总是把令牌发送给节点1。当一个节点收到令牌时，仅当它有一些帧要发送时，它才持有这个令牌；否则，它立即向下一个节点转发该令牌。当一个节点收到令牌时，如果它确实有帧要传输，它发送最大数目的帧数，然后把令牌转发给下一个节点。令牌传递是分散的，并有很高的效率。但是它也有其自己的一些问题。例如，一个节点的故障可能会使整个信道崩溃。或者如果一个节点偶然忘记了释放令牌，必须调用某些恢复步骤使令牌返回到循环中来。经过多年，人们已经开发了许多令牌传递协议，每一种都必须解决这些和其他一些棘手的问题。我们将在下列小节中提及这些协议中的两个：FDDI和IEEE 802.5。

5.3.4 局域网

多路访问协议的使用与很多不同类型的广播信道联系在一起。它们已经用于卫星和无线信道中，这些信道的节点在共同的频谱上传输。它们当前用于电缆接入因特网（见1.2节）的上行信道中，并广泛使用于局域网（LAN）中。

前面讲过LAN是一种集中在一个地理区域的计算机网络，例如在一个建筑中或者一个大学校园内。当用户从一所大学或者公司园区接入因特网时，其接入几乎总是通过LAN的方式。特别是这种接入是从主机经LAN经路由器到因特网，如图5-15所示。大多数LAN的传输速率R是非常高的。甚至在20世纪80年代早期，10 Mbps的LAN也是很常见的；现在100 Mbps和1 Gbps的LAN也很常见，并且有10 Gbps的LAN可供使用。

在20世纪80年代和90年代早期，在工作场所有两类LAN技术很流行。第一类由以太网LAN（也称为802.3 LAN [IEEE 802.3 2007]）组成，它是基于随机接入的。第二类LAN技术由令牌传递技术组成，包括**令牌环**（token ring），（也称为IEEE 802.5[IEEE 802.5 2007]）以

及**光纤式分布数据接口**（Fiber Distributed Data Interface, FDDI）[Jain 1994]）。因为我们将在5.5节详细研究以太网技术，这里我们主要讨论令牌传递LAN。我们对令牌传递技术的讨论特意很简洁，因为以太网无情的竞争已经使得这些技术变得过时了。尽管如此，为了提供令牌传递技术例子并给出少许历史展望，简要地介绍一下令牌环是有益的。

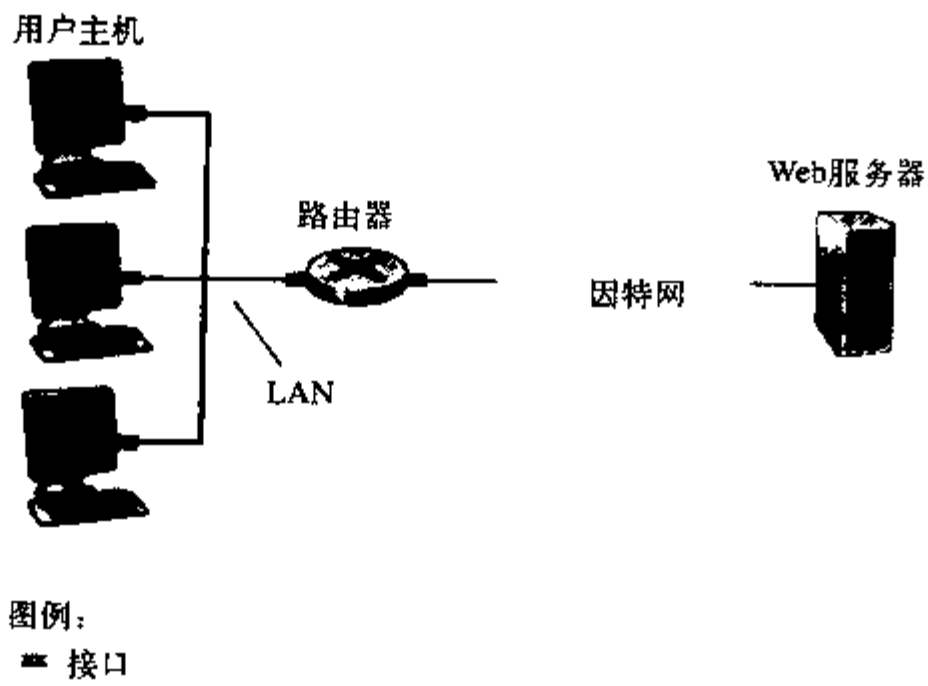


图5-15 用户主机通过LAN访问一台因特网Web服务器。
用户主机和路由器之间的广播信道由一条链路组成

在令牌环LAN中，LAN的N个节点（主机和路由器）通过直接链路连接成一个环。令牌环的拓扑定义了令牌传递的次序。当某节点获得了令牌并发送一个帧时，该帧绕着整个环传播，从而创建了一个虚拟广播信道。当该帧传播过来时，目的节点从链路层媒体中读取该帧。发送该帧的节点有责任从环中去除这个帧。FDDI为地理范围上更大的LAN而设计，包括所谓的**城域网**（Metropolitan Area Network，MAN）。对于地理分布上大的LAN（扩展到几千米），一旦该帧传递给目的节点，让该帧传播返回到发送节点的效率是很低的。FDDI让目的节点从环中去除该帧。（严格来说，FDDI不是一个纯粹的广播信道，因为每个节点并不接收每个传输的帧。）

5.4 链路层编址

节点（即主机和路由器）具有链路层地址。现在你也许会感到惊奇，第4章中不是讲过节点也具有网络层地址吗？你也许会问：究竟为什么我们在网络层和链路层都需要地址呢？除了描述链路层地址的语法和功能，在本节中我们希望明明白白地搞清楚两层地址都有用的原因，事实上这些地址都是必不可少的。我们也将学习地址解析协议（ARP），该协议提供了将IP地址转换为链路层地址的机制。

5.4.1 MAC地址

事实上，并不是节点（即主机或路由器）具有MAC地址，而是节点的适配器具有链路层地址。图5-16中举例说明这种情况。LAN地址有各种不同的称呼：**LAN地址**（LAN address）、**物理地址**（physical address）或**MAC地址**（MAC address）。因为MAC地址看起来是最为流行的术语，我们此后就将链路层地址称为MAC地址。对于大多数LAN（包括以太网和802.11

无线LAN）而言，MAC地址长度为6字节，共有2^{48}个可能的LAN地址。如图5-16所示，该6个字节地址通常用十六进制表示法，地址的每个字节被表示为一对十六进制数。尽管MAC地址被设计为永久的，但现在用软件改变一块适配器的MAC地址是可能的。然而，对于本节的后面部分而言，我们将假设某适配器的MAC地址是固定的。

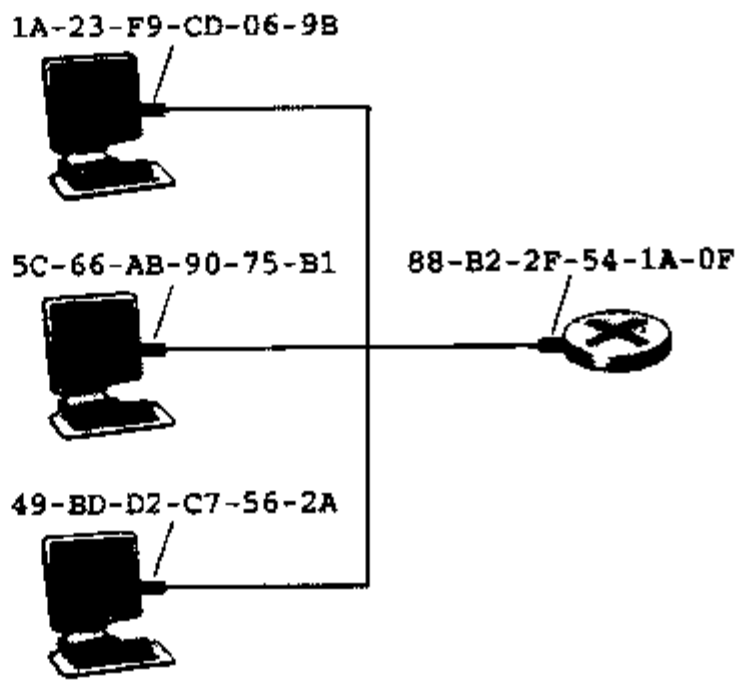


图5-16 与LAN相连的每个适配器都有一个唯一的MAC地址

MAC地址的一个有趣性质是，没有两个适配器具有相同的地址。考虑到适配器是由许多不同国家的不同公司生产的，这看起来似乎是件神奇之事。在台湾的生产适配器的公司如何能够保证与在比利时的生产适配器的公司使用不同的地址？答案是IEEE在管理着该MAC地址空间。特别是，当一个公司要生产适配器时，它支付象征性的费用购买组成上述2^{24}个地址的一块地址空间。IEEE分配这块2^{24}个地址的方式是：固定一个MAC地址的前24比特，让公司自己为每个适配器生成后24比特的唯一组合。

适配器的MAC地址具有扁平结构（这与层次结构相反），而且不论适配器到哪里用都不会变化。带有以太网网卡的便携机总具有同样的MAC地址，无论该计算机位于何方。具有802.11接口的PDA总是具有相同的MAC地址，无论该PDA到哪里。与之形成对照的是，前面说过的IP地址具有层次结构（即一个网络部分和一个主机部分），而且当主机移动时，节点的IP地址需要改变，即改变为它所连接到网络的地址。适配器的MAC地址与人的社会保险号相似，后者也具有扁平寻址结构，而且无论人到哪里该号码都不会变化。IP地址则与一个人的邮政地址相似，它是有层次的，无论何时当人搬家时，该地址都必须改变。就像一个人可能发现邮政地址和社会保险号都有用那样，一个节点具有一个网络层地址和一个MAC地址是有用的。

如我们在本节开始所描述的那样，当某适配器要向某些目的适配器发送一个帧时，发送适配器将目的适配器的MAC地址插入到该帧中，并将该帧发送到LAN上。如果该LAN是一个广播LAN（如802.11和许多以太网LAN），这个帧会被该LAN上的所有其他适配器接收和处理。特别是，接收到该帧的每个适配器将检查，看在该帧中的目的MAC地址是否与它自己的MAC地址匹配。如果匹配，该适配器提取出封装的数据报，并将该数据报沿协议栈向上传递给它的父节点。如果不匹配，该适配器丢弃该帧，则不会沿协议栈向上传递该网络层数据报。所以，当目的节点收到该帧时，仅有它将会中断其父节点。

然而，有时某发送适配器的确要让LAN上所有的其他适配器来接收并处理它打算发送的帧。在这种情况下，发送适配器在该帧的目的地址字段中插入一个特殊的MAC广播地址（broadcast address）。对于使用6 字节地址的LAN（例如以太网和令牌传递LAN）来说，广播地址是48个连续的1组成的字符串（即以十六进制表示法表示的FF-FF-FF-FF-FF-FF）。

实践原则

保持各层独立

节点除了网络层地址之外还有MAC地址，这有几个原因。首先，LAN是为任意网络层协议而设计的，而不只是为IP和因特网。如果适配器被指派IP地址而不是“中立的”MAC地址的话，则适配器将不能够方便地支持其他网络层协议（例如，IPX或者DECnet）。其次，如果适配器使用网络层地址代替MAC地址的话，网络层地址必须存储在适配器的RAM中，并且在每次适配器移动（或加电）时要重新配置。另一种选择是在适配器中不使用任何地址，让每个适配器将它收到的每帧数据（通常是IP数据报）沿协议栈向上传递。然后网络层则能够核对网络地址层是否匹配。这种选择带来的一个问题是，主机将被LAN上发送的每个帧中断，包括被目的地是在相同广播LAN上的其他节点的帧中断。总之，为了使在网络体系结构中各层次大体上成为独立的构建模块，不同的层次需要有它们自己的寻址方案。我们现在已经看到3种类型的地址：应用层的主机名、网络层的IP地址以及链路层的MAC地址。

5.4.2 地址解析协议

因为存在网络层地址（例如，因特网的IP地址）和链路层地址（即MAC地址），所以需要在它们之间进行转换。对于因特网而言，这是**地址解析协议**（Address Resolution Protocol，ARP）[RFC 826]的工作。

为了理解对于诸如ARP这样协议的需求，考虑如图5-17所示的网络。在这个简单的例子中，每个节点有一个单一的IP地址，并且每个节点的适配器都有一个单一的MAC地址。通常，IP地址以点分十进制标记法来表示，MAC地址以十六进制标记法来表示。现在假设IP地址为222.222.222.220的节点要发送IP数据报到节点222.222.222.222。（例如，目的节点222.222.222.222可能是一台Web服务器，发送节点222.222.222.220可能利用DNS确定了该Web服务器的IP地址。）在这个例子中，在4.4.2节的寻址意义下，源和目的节点均位于相同的网络（LAN）中。为了发送数据报，该源节点必须要向它的适配器不仅提供IP数据报，而且要提供目的节点222.222.222.222的MAC地址。提供了IP数据报和MAC地址后，发送节点的适配器将构造一个包含目的节点的MAC地址的链路层帧，并把该帧发进LAN。

图5-17 LAN上的每个节点都有一个IP地址并且每个适配器都有一个MAC地址

在本节中要处理的重要问题是，发送节点如何确定IP地址为222.222.222.222的节点的MAC地址呢？正如你也许已经猜想的那样，它使用ARP。在发送节点中的ARP模块将取在相同LAN上的任何IP地址作为输入，然后返回相应的MAC地址。在手边的这个例子中，发送节点222.222.222.220向它的ARP模块提供了IP地址222.222.222.222，并且其ARP模块返回了相应的MAC地址49-BD-D2-C7-56-2A。

因此我们看到了ARP将一个IP地址解析为一个MAC地址。在很多方面它和DNS（在2.5节研究过）类似，DNS将主机名解析为IP地址。然而，这两种解析器之间的一个重要区别是，DNS为在因特网中任何地方的主机解析主机名，而ARP只为在同一个子网上的节点解析IP地址。如果美国加利福尼亚州的一个节点试图用ARP为美国密西西比州的一个节点解析IP地址，ARP将返回一个错误。

既然已经解释了ARP的用途，我们再来看看它是如何工作的。每个节点（主机或路由器）的ARP模块都在它的RAM中有一个ARP表（ARP table），这张表包含IP地址到MAC地址的映射关系。图5-18显示了在节点222.222.222.220中的ARP表可能的形式。该表还包含一个生存期（TTL）值，它指示了从表中删除每个映射的时间。注意到这张表不必为该子网上的每个节点都包含一个表项；某些节点可能有了已经过期的表项，而其他节点可能从来没有进入到该表中。从一个表项放置到某ARP表中开始，一个表项通常的过期时间是20分钟。

IP地址	MAC地址	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

图5-18 在节点222.222.222.220中的一张可能的ARP表

现在假设节点222.222.222.220要发送一个数据报，该数据报要IP寻址到本子网上另一个节点。在给定目的节点的IP地址的情况下，发送节点需要获得它的MAC地址。如果发送节点的ARP表具有该目的节点的表项，这个任务很容易完成。但如果ARP表中现在没有该目的节点的表项，那又该怎么办呢？特别是假设节点222.222.222.220要向节点222.222.222.222发送数据报。在这种情况下，发送节点用ARP协议来解析这个地址。首先，发送节点构造一个称为ARP分组（ARP packet）的特殊分组。ARP分组有几个字段，包括发送节点和接收节点的IP地址和MAC地址。ARP查询和响应分组都具有相同的格式。ARP查询分组的目的是询问子网上所有其他节点，以确定对应于要解析的IP地址的那个MAC地址。

回到我们的例子上来，节点222.222.222.220向它的适配器传递一个ARP查询分组，并且指示要求适配器应该用MAC广播地址（即FF-FF-FF-FF-FF-FF）来发送这个分组。适配器在链路层帧中封装这个ARP分组，用广播地址作为帧的目的地址，并将该帧传输进子网中。回想我们的社会保险号/邮政地址的类比，一次ARP查询等价于一个人在某个公司（比方说AnyCorp）一个拥挤的房间大喊："邮政地址是加利福尼亚，Palo Alto，AnyCorp，112房间13室的那个人的社会保险号是什么？"包含该ARP查询的帧能被子网上的所有其他适配器接收到，并且（由于广播地址）每个适配器都把在该帧里的ARP分组向上传递给节点中的ARP模块。每个节点检查它的IP地址是否与ARP分组中的目的IP地址相匹配。（至多）一个匹配的节点给查询节点发送回一个带有所希望映射的响应ARP分组。然后查询节点222.222.222.220能

够更新它的ARP表，并发送它的IP数据报，该数据报封装在一个链路层帧中。该帧的目的MAC就是对先前ARP请求进行响应的节点的MAC地址。

关于ARP协议有两件有趣的事情需要注意。首先，查询ARP报文是在广播帧中发送的，而响应ARP报文在一个标准帧中发送。在继续阅读之前，你应该思考一下为什么是这样的。第二，ARP是即插即用的，这就是说，一个节点的ARP表是自动建立的（它不需要系统管理员来配置）。并且如果某节点与子网断开连接，它的表项最终会从留在子网中的节点的表中删除掉。

发送数据报到子网以外的节点

当一个节点要向相同子网（在4.4.2节中已精确定义了子网）的另一个节点发送一个数据报时，ARP的操作过程现在应该很清楚了。但是现在我们来看更复杂的情况，当子网中的某节点要向子网之外（也就是跨越路由器的另一个子网）的节点发送网络层数据报的情况。我们在图5-19的环境中来讨论这个问题，该图显示了一个由一台路由器互联的两个子网组成的简单网络。

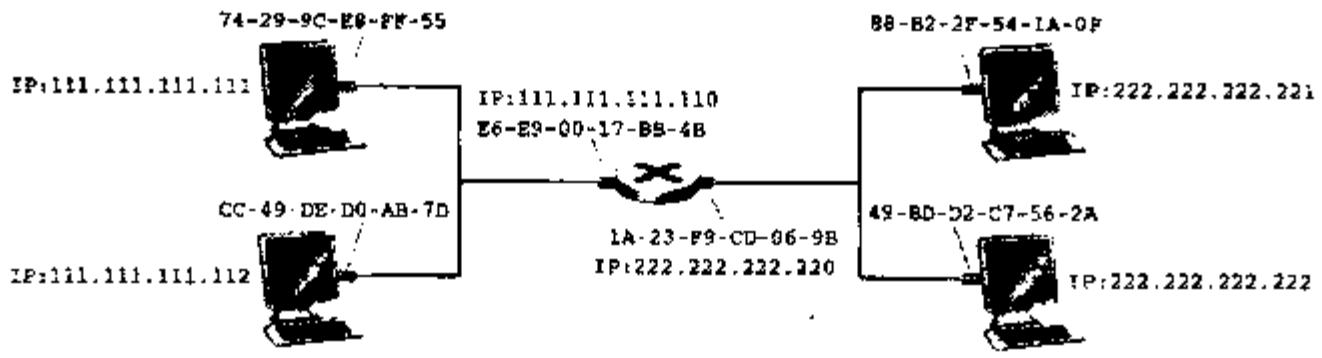


图5-19 由一台路由器互联的两个子网

有关图5-19有几件有趣的事情需要注意。首先，有两种类型的节点：主机和路由器。每台主机仅有一个IP地址和一个适配器。但是，如在第4章讨论的那样，一台路由器对它的每个接口都有一个IP地址。对路由器的每个接口（在路由器里）也有一个ARP模块和一个适配器。因为在图5-19中的路由器有两个接口，它有两个IP地址、两个ARP模块和两个适配器。当然，在网络中的每个适配器有其自己的MAC地址。

还要注意的是，子网1的网络地址为111.111.111/24，子网2的网络地址为222.222.222/24。因此，与子网1相连的所有接口都有格式为111.111.111.xxx的地址，与子网2相连的所有接口都有格式为222.222.222.xxx的地址。

现在我们考察子网1上的某主机将向子网2上的某主机发送数据报。我们特定假设主机111.111.111.111要向主机222.222.222.222发送一个IP数据报。和通常一样，发送主机向它的适配器传递数据报。但是，发送主机还必须向它的适配器指示一个适当的目的MAC地址。该适配器应该使用什么MAC地址呢？有人也许大胆猜测道，这个适当的MAC地址就是主机222.222.222.222的适配器地址，即49-BD-D2-C7-56-2A。然而，这个猜测是错误的。如果发送适配器要用那个MAC地址，那么子网1上所有的适配器都不会费心将该IP数据报传递到它的网络层，因为该帧的目的地址都将不与子网1上所有的适配器的MAC地址匹配。这个数据报将只有死亡，到达数据报天国。

如果我们仔细观察图5-19，我们发现为了使一个数据报从111.111.111.111到子网2上的节点，该数据报必须首先发送给路由器接口111.111.111.110。因此，这个帧适当的MAC地址是路由器接口111.111.111.110的适配器地址，即E6-E9-00-17-BB-4B。但发送主机怎样获得

111.111.111.110的MAC地址呢？当然是通过ARP！一旦发送适配器有了这个MAC地址，它创建一个帧（包含了寻址到222.222.222.222的数据报），并把该帧发送到子网1中。在子网1上的路由器适配器看到该链路层帧是向它寻址的，因此把这个帧传递给路由器的网络层。万岁！该IP数据报终于被成功地从源主机移动到这台路由器了！但是我们的任务还没有结束。我们仍然需要将该数据报从路由器移动到目的地。路由器现在必须决定该数据报要被转发的正确接口。正如在第4章中所讨论的，这是通过查询路由器中的转发表来完成的。转发表告诉这台路由器该数据报要通过路由器接口222.222.222.220转发。然后该接口把这个数据报传递给它的适配器，适配器把该数据报封装到一个新的帧中，并将帧发送进子网2中。这时，该帧的目的MAC地址确实是最终目的地的MAC地址。路由器又是怎样获得这个目的MAC地址的呢？当然是用ARP获得的！

以太网的ARP在RFC 826中定义。在TCP/IP指南RFC 1180中对ARP进行了很好的介绍。我们将在课后习题中更为详细地研究ARP。

5.5 以太网

以太网几乎完全占领着现有的有线局域网市场。在20世纪80年代和90年代早期，以太网面临着来自其他LAN技术包括令牌环、FDDI和ATM的挑战。多年来，这些其他技术成功地抓住了部分LAN市场份额。但是自从20世纪70年代中期发明以太网以来，它就不断演化和发展，保持了它的支配地位。今天，以太网是到目前为止最流行的有线局域网技术，而且到可能预见的将来它仍可能保持这一位置。可以这么说，以太网对本地区域联网的重要性就像因特网对全球联网所具有的地位那样。

以太网的成功有很多原因。首先，以太网是第一个广泛部署的高速LAN。因为它部署得早，网络管理员非常熟悉以太网（它的奇迹和它的奇思妙想），并当其他LAN技术问世时，他们不愿意转而用之。其次，令牌环、FDDI和ATM比以太网更加复杂、更加昂贵，这就进一步阻碍了网络管理员改用其他技术。第三，改用其他LAN技术（例如FDDI和ATM）的最引人注目的原因通常是这些新技术具有更高数据速率；然而以太网总是奋起抗争，产生了工作在相同或更高的数据速率下的版本。在20世纪90年代初期引入了交换式因特网，这就进一步增加了它的有效数据速率。最后，由于以太网已经很流行了，以太网硬件（尤其是适配器和交换机）成为了一个普通商品而且极为便宜。

Bob Metcalfe和David Boggs在20世纪70年代中期发明初始的以太网LAN。图5-20显示了Metcalfe用于该发明的原理图。在这张图上，你能注意到初始的以太网LAN使用同轴电缆总线来互联节点。以太网的总线拓扑实际上从20世纪80年代到90年代中期的大部分时间一直保持不变。使用总线拓扑的以太网是一种广播LAN，即所有传输的帧传送到与该总线连接的所有适配器并被其处理。

到了20世纪90年代后期，大多数公司和大学使用一个基于集线器的星型拓扑，用以太网安装替代了它们的LAN。如图5-21所示，在这种安装中主机（和路由器）直接用双绞铜线与一台集线器相连。**集线器**（hub）是一种物理层设备，它作用于各个比特而不是作用于帧。当表示一个0或一个1的比特到达一个接口时，集线器只是重新生成这个比特，将其能量强度放大，并将该比特向其他所有接口传输出去。因此，具有基于集线器星型拓扑的以太网也是一个广播LAN，即无论何时集线器从它的一个接口接收到一个比特时，它向其所有其他接口发

送该比特的拷贝。特别是，如果某集线器同时从两个不同的接口接收到帧，出现一次碰撞，生成该帧的节点必须重新传输该帧。

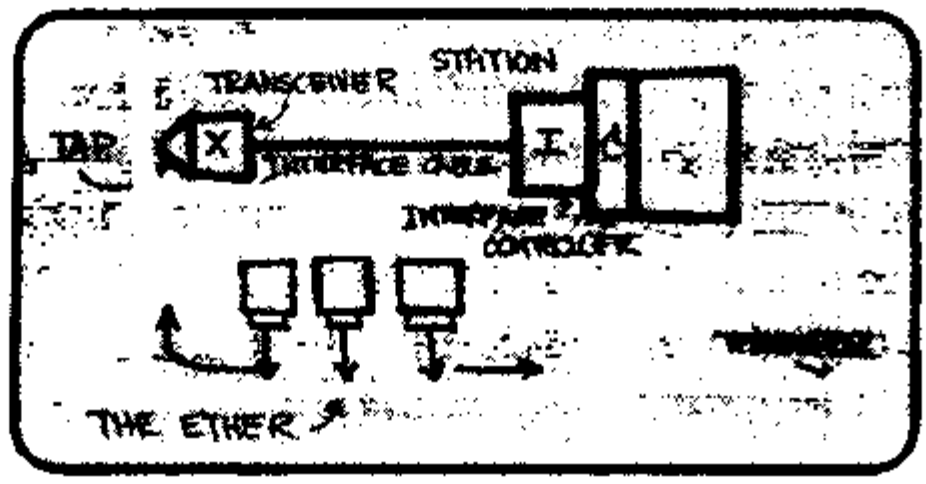


图5-20 Metcalfe最早设计产生了10Base5以太网标准，它包括了连接以太网适配器与外部收发器的接口电缆

在21世纪前期，以太网又经历了一次重要的革命性变革。以太网安装继续使用星型拓扑，但是位于中心的集线器被**交换机**（switch）所替代。在本章后面我们将深入研究交换以太网。就目前而言，我们仅提到交换机不仅是“无碰撞的”，而且也是名副其实的存储转发的分组交换机；但是与运行在下3层的路由器不同，交换机仅运行在下两层。

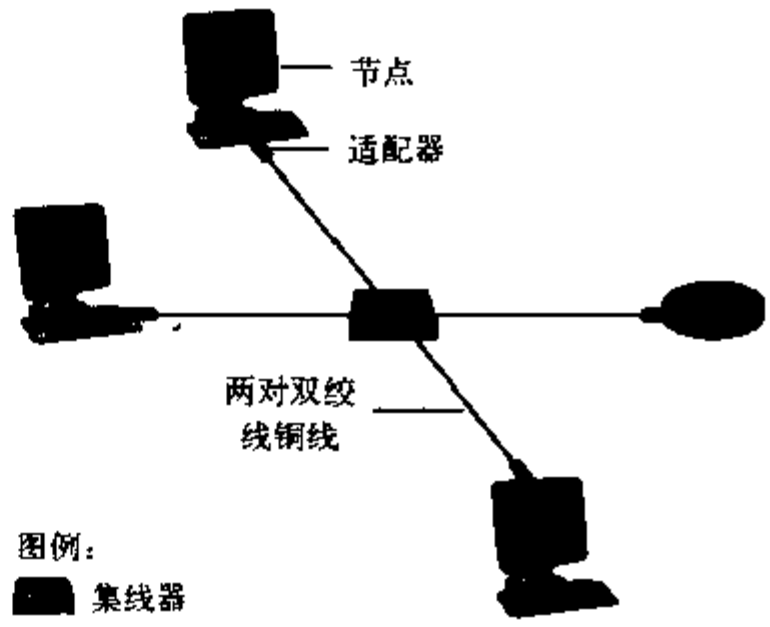


图5-21 用于以太网的星型拓扑。用一个集线器互联节点

5.5.1 以太网帧结构

以太网帧如图5-22所示。通过研究以太网的帧，我们能够学到许多有关以太网的知识。为了将对以太网帧的讨论放到切实的环境中，考虑从一台主机向另一台主机发送一个IP数据报，且这两台主机在相同的以太网LAN上（例如，在图5-21中的以太网LAN）。（尽管以太网帧的负载是一个IP数据报，我们顺便说一句，以太网帧也能够承载其他网络层分组。）设发送适配器（即适配器A）的MAC地址是AA-AA-AA-AA-AA-AA，接收适配器（即适配器B）的MAC地址是BB-BB-BB-BB-BB-BB。发送适配器在一个以太网帧中封装了一个IP数据报，并把该帧传递到物理层。接收适配器从物理层收到这个帧，提取出IP数据报，并将该IP数据报传递给网络层。在这种情况下，我们现在考察如图5-22所示的以太网帧的6个字段：

图5-22 以太网帧结构

- **数据字段**（data field）（46～1500字节）。这个字段承载了IP数据报。以太网的最大传输单元（MTU）是1500字节。这意味着如果IP数据报超过了1500字节，则主机必须将该

数据报分段，如4.4.1节讨论的那样。数据字段的最小长度是46字节。这意味着如果IP数据报小于46字节，数据报必须被填充到46字节。当采用填充时，传递到网络层的数据包括IP数据报和填充部分。网络层使用IP数据报首部中的长度字段来去除该填充。

- 目的地址（destination address）（6 字节）。这个字段包含目的适配器的MAC地址，即BB-BB-BB-BB-BB-BB。当适配器B收到一个以太网帧，它的帧目的地址无论是它自己的MAC地址BB-BB-BB-BB-BB-BB，还是MAC广播地址，都将该帧的数据字段的内容传递给网络层；如果它收到了具有任何其他MAC地址的帧，则丢弃之。
- 源地址（source address）（6 字节）。这个字段包含了传输该帧到LAN上的适配器的MAC地址，在本例中为AA-AA-AA-AA-AA-AA。
- 类型字段（type field）（2字节）。该类型字段允许以太网复用多种网络层协议。为了理解这点，我们需要记住主机能够使用除IP以外的其他网络层协议。事实上，一台给定的主机可以支持多种网络层协议，以对不同的应用采用不同的协议。由于这个原因，当以太网帧到达适配器B，适配器B需要知道它应该将数据字段的内容传递给哪个网络层协议（即多路分解）。IP和其他链路层协议（例如，Novell IPX或AppleTalk）都有它们各自的、标准化的类型编号。此外，ARP协议（在上一节讨论过）有自己的类型编号。注意到该类型字段和网络层数据报中的协议字段、运输层报文段的端口号字段相类似，所有这些字段都是为了把一层中的某协议与上一层的某协议结合起来。
- 循环冗余检测（Cyclic Redundancy Check, CRC）（4字节）。如5.2.3节中讨论的那样，CRC字段的目的是使得接收适配器（适配器B）检测帧中是否引入了差错，也就是说，帧中的比特是否发生了翻转。比特差错的原因包括信号强度的衰减和电磁能量泄漏到以太网电缆和接口卡中。差错检测以如下方式进行。当主机A构造以太网帧时，它计算一个CRC字段，CRC字段是从帧中其他比特（除了前同步码）映射得来的。当主机B接收该帧时，它对该帧运用同样的映射，并核对映射的结果是否与CRC字段的内容相等。接收主机的操作称为**CRC校验**（CRC check）。如果CRC校验失败（即如果映射的结果不等于CRC字段内的内容），则主机B知道该帧中有差错。
- 前同步码（preamble）（8字节）。以太网帧以一个8 字节的前同步码字段开始。该前同步码的前7个字节的值都是10101010；最后一个字节是10101011。前同步码字段的前7个字节用于“唤醒”接收适配器，并且将它们的时钟和发送方的时钟同步。为什么这些时钟会不同步呢？记住适配器A的目的是根据以太网LAN的类型的不同，以10 Mbps、100 Mbps或者1 Gbps的速率传输帧。然而，由于没有任何技术是完美无缺的，适配器A不会以精确的额定速率传输帧；相对于额定速率总有一些漂移，LAN上的其他适配器不会预先知道这种漂移的。接收适配器只需通过锁定前同步码的前7个字节的比特，就能够锁定适配器A的时钟。前同步码的第8个字节的最后两个比特（第一个出现的两个连续的1）警告适配器B，“重要的内容”就要到来了。当主机B看到这两个连续的1，它知道接下来的6个字节是目的地址。适配器只需检测有没有电流，就能够分辨出一个帧何时结束。

以太网使用基带传输：这就是说，适配器直接向广播信道发送数字信号。接口卡不会像ADSL和电缆调制解调器系统那样，将信号搬移到其他频带。许多以太网技术（例如10BASE-T）也使用如图5-23所示的曼彻斯特（Manchester）编码。采用曼彻斯特编码，每个比特包含一次跳变：1从高电平跳变为低电平，而0从低电平跳变为高电平。使用曼彻斯特编码的原因

是发送适配器和接收适配器的时钟没有精确同步。通过在每个比特的中间包含一次跳变，接收主机能够将它的时钟与发送主机的时钟同步。一旦接收适配器的时钟被同步，接收方能够对每个比特定界，并且确定它是1还是0。曼彻斯特编码是一种物理层的操作，而不是链路层的操作。然而，我们在这里简要地描述它，是因为它广泛地应用于以太网中。

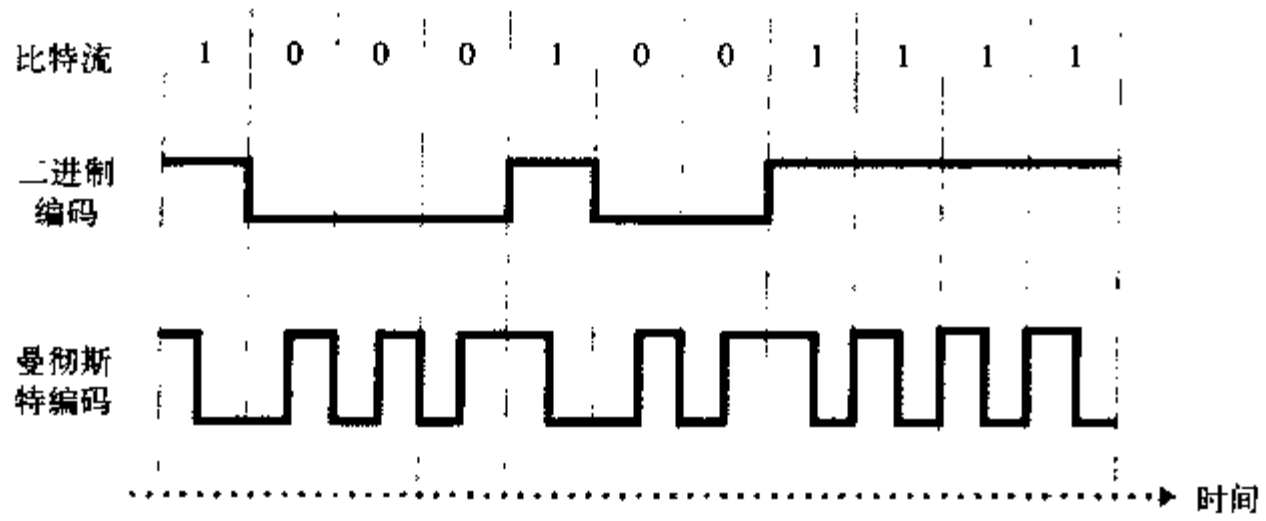


图5-23 曼彻斯特编码

历史事件

Bob Metcalfe和以太网

作为20世纪70年代早期哈佛大学的一名博士生，Bob Metcalfe在MIT从事ARPAnet的研究。在他学习期间，他还受到了Abramson有关ALOHA和随机接入协议工作的影响。在完成了他的博士学位，并在开始Xerox Palo Alto研究中心（Xerox PARC）的工作之前，他用3个月访问了Abramson和他夏威夷大学的同事，获得了ALOHAnet的第一手资料。在Xerox PARC，Metcalfe受到了Alto计算机的影响，这种计算机在很多方面是20世纪80年代个人计算机的先驱。Metcalfe看到了对这些计算机以一种不昂贵的方式组网的需求。因此，基于他对APRAnet、ALOHAnet和随机接入协议知识的了解，Metcalfe和他的同事David Boggs一起发明了以太网。

Metcalfe和Boggs最初的以太网运行速度为2.94 Mbps，连接长达一英里范围的多达256台主机。Metcalfe和Boggs成功地使得Xerox PARC的大多数研究人员通过他们的Alto计算机互相通信。然后Metcalfe推进了Xerox、Digital和Intel联盟，创建了以太网作为一种10 Mbps的以太网标准，该标准后被IEEE认可。Xerox对以太网商业化没有表现出太多的兴趣。在1979年，Metcalfe建立了自己的公司3Com，它发展和商业化了网络技术，包括以太网技术。特别是，3Com在20世纪80年代早期为非常流行的IBM PC机开发了以太网网卡并使之市场化。在1990年，Metcalfe离开了3Com，当时3Com有2000名员工和4亿美元的年收入。

不可靠的无连接服务

所有的以太网技术都向网络层提供**无连接服务**（connectionless service）。这就是说，当适配器A要向适配器B发送一个数据报时，适配器A在一个以太网帧中封装这个数据报，并且把该帧发送到LAN上，没有与适配器B先“握手”。这种第二层的无连接服务类似于IP的第三层

数据报服务和UDP的第4层无连接服务。

所有的以太网技术都向网络层提供了**不可靠服务**（unreliable service）。特别是，当适配器B收到一个来自适配器A的帧，它对该帧执行CRC校验，但是当该帧通过CRC校验时，它不发送确认帧，而当该帧没有通过CRC校验时，它也不发送否定帧。当某帧没有通过CRC校验，适配器B只是丢弃该帧。因此，适配器A根本不知道它传输的帧是否到达了B并通过了CRC校验。（在链路层）不提供可靠的传输有助于使得以太网简单和便宜。但是它也意味着传递到网络层的数据报流可能有间隙。

如果由于丢弃了以太网帧而存在间隙，主机B上的应用也看见这个间隙吗？如我们在第3章中所知，这只取决于该应用是使用UDP还是使用TCP。如果应用使用的是UDP，则主机B中的应用的确会看到数据中的间隙。另一方面，如果应用使用的是TCP，则主机B中的TCP将不会确认包含在丢弃帧中的数据，从而引起主机A的TCP重传。注意到当TCP重传数据时，数据最终将回到曾经丢弃它的以太网适配器。因此，从这种意义上来说，以太网的确重传了数据，尽管以太网并不知道它是正在传输一个具有全新数据的全新数据报，还是一个包含已经被传输过至少一次的数据的数据报。

5.5.2 CSMA/CD：以太网的多路访问协议

当一个节点通过某集线器（与链路层交换机不同）互联时（如图5-21所示），该以太LAN是一个真正的广播LAN，即当某适配器传输一帧时，LAN上的所有适配器都收到该帧。因为以太网能够应用广播，故它需要多路访问协议。以太网使用了广受赞誉的CSMA/CD多路访问协议。5.3节讲过，CSMA/CD使用了以下机制：

1）适配器可以在任何时刻开始传输；这就是说没有时隙的概念。

2）当一个适配器侦听到有某些其他的适配器正在传输，它决不会传输帧；这就是说它使用了载波侦听。

3）一旦传输中的适配器检测到另一个适配器正在传输，就中止它的传输；这就是说它使用了碰撞检测。

4）在尝试重传之前，适配器等待一个随机时间，这个时间通常比传输一帧的时间要短。

这些机制使得在LAN环境下CSMA/CD的性能比时隙ALOHA的性能要好得多。实际上，如果站点之间的最大传播时延很小，CSMA/CD的效率可以接近100%。但是注意到上面所列的第二个和第三个机制要求每个以太网的适配器要能够：①当某些其他适配器在传输时进行侦听，②当它传输时检测碰撞。以太网适配器通过测量传输前和传输中的电压水平来执行这两个任务。

每个适配器运行CSMA/CD协议，无需与以太网上的其他适配器进行显式地协调。在一个特定的适配器中，CSMA/CD协议按下列方式工作：

1）适配器从网络层得到一个数据报，准备一个以太网帧，并把该帧放到适配器缓存区中。

2）如果适配器侦听到信道空闲（即在96比特时间内，没有信号能量从信道进入到适配器），它开始传输该帧。如果适配器侦听到信道忙，它等待到侦听不到信号能量（加上96比特时间），然后开始传输该帧。

3）在传输过程时，适配器监视来自其他适配器的信号能量的出现。如果该适配器传输了整个帧，而没有检测到来自其他适配器的信号能量，它就完成了该帧的传输。

4）如果适配器在传输中检测到来自其他适配器的信号能量，它就停止传输它的帧，而代之以传输一个48比特的阻塞（jam）信号。

5）在中止（即传输阻塞信号）以后，适配器进入一个**指数后退**（exponential backoff）阶段。特别是，当传输一个给定帧时，在该帧经受了一连串的第n次碰撞后，适配器随机地从$\{0,1,2,\cdots,2^m-1\}$为K选择一个值，其中$m=\min(n, 10)$。然后适配器等待$K \cdot 512$比特时间，并返回到第2步。

很有必要对CSMA/CD协议作出一些说明。阻塞信号的目的是确保所有其他的传输中的适配器都意识到此次碰撞。我们看一个例子。假设适配器A开始传输一帧，并且恰好在A的信号到达适配器B之前，适配器B开始传输了。因此当B中止传输时，它将只传输了几个比特。这几个比特确实将传播到A，但是它们可能不足以形成足够的能量来使A检测到此次碰撞。为了确保A检测到这个碰撞（使得它也能够中止传输），B传输该48比特的阻塞信号。

接下来考虑指数后退算法。这里首先要注意的是，1比特时间（即传输单个比特所需的时间）是很短的；对于一个10 Mbps以太网，1比特时间是0.1 μs。现在我们看一个例子。假设一个适配器首次尝试传输一个帧，并在传输中它检测到碰撞。然后该适配器以概率0.5选择$K=0$，以概率0.5选择$K=1$。如果该适配器选择$K=0$，则在传输完阻塞信号之后立刻跳到第2步。如果这个适配器选择$K=1$，它在返回到第2步之前要等待51.2 μs。在第一次碰撞之后，从{0,1,2,3}中等概率地选择K。在3次碰撞之后，从{0,1,2,3,4,5,6,7}中等概率地选择K。在10次或更多次碰撞之后，从$\{0,1,2,\cdots,1023\}$中等概率地选择K。因此从中选择K的集合长度随着碰撞次数的增加呈指数增长（直到$n=10$）；正是由于这个原因，以太网的后退算法被称为**指数后退**。

以太网标准对任意两个节点之间的距离进行了限制。这些限制确保如果适配器A选择了比所有其他涉及碰撞的适配器都要低的一个K值，则适配器A将能够传输它的帧而不会经历新的碰撞。我们将在课后习题中详细地研究这个性质。

为什么要用指数后退呢？例如，为什么不在每次碰撞后从{0,1,2,3,4,5,6,7}中选择K呢？原因是当适配器经历第一次碰撞时，它不知道有多少适配器涉及这次碰撞。如果只有少量的碰撞适配器，从小集合中选择小数值的K是明智的。在另一方面，如果有很多适配器涉及这次碰撞，从一个更大的、数值更分散的集合中选择K是明智的（为什么？）。通过在每次碰撞之后增加集合的长度，适配器可以适当地适应这些不同的情况。

这里我们还要注意到，每次适配器准备传输一个新的帧时，它要运行上面所给出的CSMA/CD算法。特别是，适配器不考虑在最近过去的时间内可能发生的任何碰撞。因此，当几个其他适配器处于指数后退状态时，有可能出现某个具有新帧的适配器立刻插入一次成功的传输的情况。

以太网效率

当只有一个节点有一个帧发送时，该节点能够以该以太网技术的全速进行传输（例如10 Mbps，100 Mbps或者1 Gbps）。然而，如果很多节点都有帧要发送，信道的有效传输速率可能会小得多。我们将**以太网效率**（efficiency of Ethernet）定义为：当有大量的活跃节点，且每个节点有大量的帧要发送时，帧在信道中无碰撞地传输的那部分时间占长期运行时的份额。为了给出一个以太网效率的闭式的近似表示，令d_{prop}表示信号能量在任意两个适配器之间传播所需的最大时间。令d_{trans}表示传输一个最大长度的以太网帧的时间（对于10 Mbps的以太网，该时间近似为1.2 ms）。以太网效率的推导超出了本书的范围（见[Lam 1980]和[Bertsekas

1991])。这里我们只是列出下面的近似式：

$$效率 = \frac{1}{1 + 5d_{prop} / d_{trans}}$$

从这个公式我们看到，当d_{prop}接近0时，效率接近1。这和我们的直觉相符，如果传播时延是0，碰撞的节点将立即中止而不会浪费信道。当d_{trans}变得很大时，效率也接近于1。这也和直觉相符，因为当一个帧取得了信道时，它将占有信道很长时间；这样信道在大多数时间都会有效地工作。

5.5.3 以太网技术

在上面的讨论中，我们已经提及以太网就像是一个单一的协议标准。但事实上，以太网具有许多不同的特色，具有某种令人迷惑的首字母缩写词，如10BASE-T、100BASE-2、100BASE-T、1000BASE-LX和10GBASE-T。这些以及许多其他的以太网技术经过多年发展已经被IEEE 802.3 CSMA/CD（Ethernet）工作组标准化了[IEEE 802.3 2007]。尽管这些首字母缩写词令人眼花缭乱，实际上其中有相当大的规律性。该首字母缩写词的第一个部分指该标准的速率：10、100、1000或10G，分别代表10 Mbps、100 Mbps、1000 Mbps (1 Gbps)和10 Gbps以太网。“BASE”指基带以太网，这意味着该物理媒体仅承载以太网流量；几乎所有的802.3标准都是用于基带以太网。该首字母缩写词的最后一部分指物理媒体本身；以太网既是链路层也是物理层的规格参数，并且能够经各种物理媒体包括同轴电缆、铜线和光缆承载。通常来说，“T”指双绞铜线。

从历史上讲，以太网最初被构想为一段同轴电缆，如图5-20所示。早期的10BASE-2和10Base5标准规定了在两种类型的同轴电缆之上的10 Mbps以太网，每种标准都限制在500m长度之内。通过使用转发器（repeater）能够得到更长的运行距离，而转发器是一种物理层设备，它能在输入端接收信号，并在输出段再生该信号。如图5-20所示的同轴电缆很好地说明我们可以将以太网视为一种广播媒体，即由一个接口传输的所有帧在其他接口收到，并且以太网的CSMA/CD协议很好地解决了多路访问问题。节点直接附着在电缆上，万事大吉啦，我们有了一个局域网！

以太网已经经历了多年来的一系列演化步骤，今天的以太网非常不同于使用同轴电缆的初始总线拓扑的设计。如图5-24所示，在今天的大多数安装中，节点经点对点线段与一台交换机相连，而线段是由双绞铜线或光纤线缆构成。

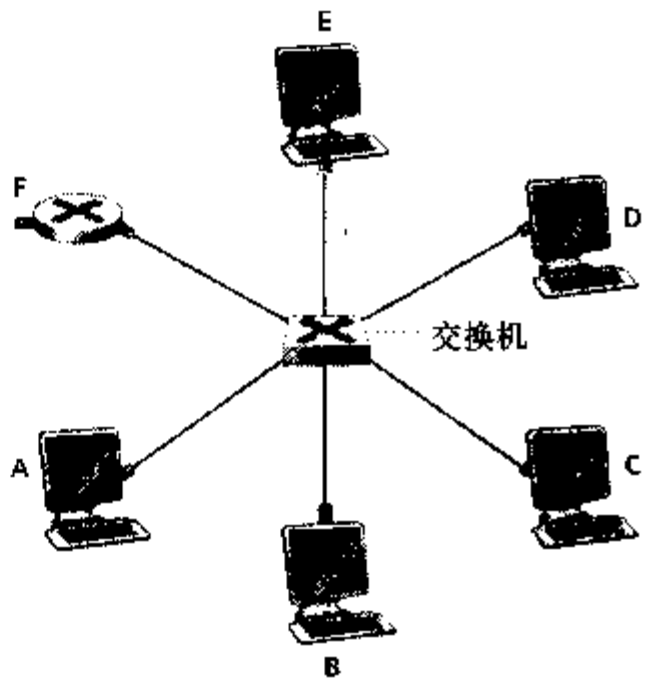


图5-24 一台链路层交换机互联6个节点

在20世纪90年代中期，以太网被标准化为100 Mbps，比10 Mbps以太网快10倍。初始的以太网MAC协议和帧格式保留了下来，但更高速率的物理层被定义用于铜线（100BASE-T）和光纤（100BASE-FX、100BASE-SX、100BASE-BX）。图5-25显示了这些不同的标准和共同的以太网MAC协议和帧格式。位于不同建筑物中的以太网交换机能够相互连接，

100 Mbps以太网用双绞线的距离限制为100m，用光纤的距离限制为几千米。

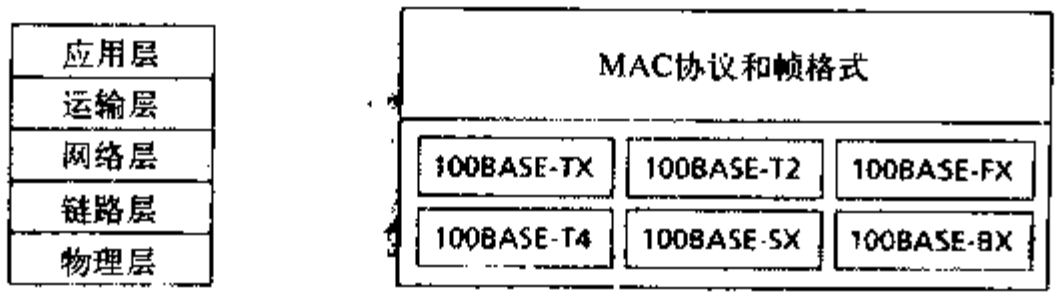


图5-25 100 Mbps以太网标准：共同的链路层，不同的物理层

吉比特以太网是对极为成功的10 Mbps和100 Mbps以太网标准的扩展。吉比特以太网提供1000 Mbps的总数据速率，与大量已经安装的以太网设备的基础完全兼容。吉比特以太网的标准称为IEEE 802.3z，它完成以下工作：

- 使用标准以太网帧格式（参见图5-22），并且后向兼容10BASE-T与100BASE-T技术。这使得吉比特以太网和现有安装的以太网设备基础很容易集成。
- 允许点对点链路以及共享的广播信道。如前所述，点对点链路使用交换机，而广播信道使用集线器。在吉比特术语中，集线器被称为*带缓存的分配器*。
- 使用CSMA/CD来共享广播信道。为了得到可接受的效率，节点之间的最大距离必须严格限制。
- 对于点对点信道，允许在两个方向上都以1000 Mbps全双工操作。

吉比特以太网最初工作于光纤之上，现在能够工作在5类UTP线缆上。在2006年夏季，10 Gbps以太网（10 GBASE-T）已被标准化了，这预示着在不远的将来会有更高的以太网LAN能力。

我们通过提出一个问题来结束有关以太网技术的讨论，这个问题开始可能会难倒你。在总线拓扑和基于集线器星型拓扑技术的时代，以太网很显然是一种广播链路（如在5.3节所定义），当多个节点同时传输时会出现帧碰撞。为了处理这些碰撞，以太网标准包括了CSMA/CD协议，该协议对于跨越一个小的地理半径的有线广播LAN特别有效。但是对于今天广为使用的以太网是基于交换机的星型拓扑，采用的是存储转发分组交换，是否还真正需要一种以太网MAC协议呢？如我们在5.6节所见，交换机协调其传输，在任何时候决不会向相同的接口转发超过一个帧。此外，现代交换机是全双工的，这使得一台交换机和一个节点能够在同时向对方发送帧而没有干扰。换句话说，在基于交换机的以太网LAN中，不会有碰撞，因此没有必要使用MAC协议。

如我们所见，今天的以太网与Metcalfe和Boggs在30多年前构想的初始以太网有非常大的不同，即速度已经增加了3个数量级，以太网帧承载在各种各样的媒体之上，交换以太网已经成为主流，此时甚至连MAC协议也经常是不必要的了！所有这些还真正是以太网吗？答案当然是“是的，完全正确。”然而，注意到下列事实是有趣的：通过所有这些改变，的确还有一个历经30年保持不变的东西，即以太网帧格式。也许这才是以太网标准的一个真正重要的特征。

5.6 链路层交换机

如图5-26所示，现代以太网LAN使用了一种星型拓扑，每个节点与中心交换机相连。到

目前为止，我们对交换机实际要做的工作以及它是怎样工作的含糊其辞。交换机的任务是接收入链路层帧并将它们转发到出链路；我们将很快详细学习这种转发功能。交换机自身对节点是透明的（transparent），这就是说，某节点向另一个节点寻址一个帧（而不是向交换机寻址该帧），顺利地将该帧发送进LAN，而不知道某交换机将会接收该帧并将它转发到另一个节点。这些帧到达该交换机的任何输出接口之一的速率可能暂时会超过该接口的链路容量。为了解决这个问题，交换机输出接口设有缓存，以非常类似于路由器接口为数据报设有缓存的方式工作。现在我们来仔细考察交换机运行的原理。

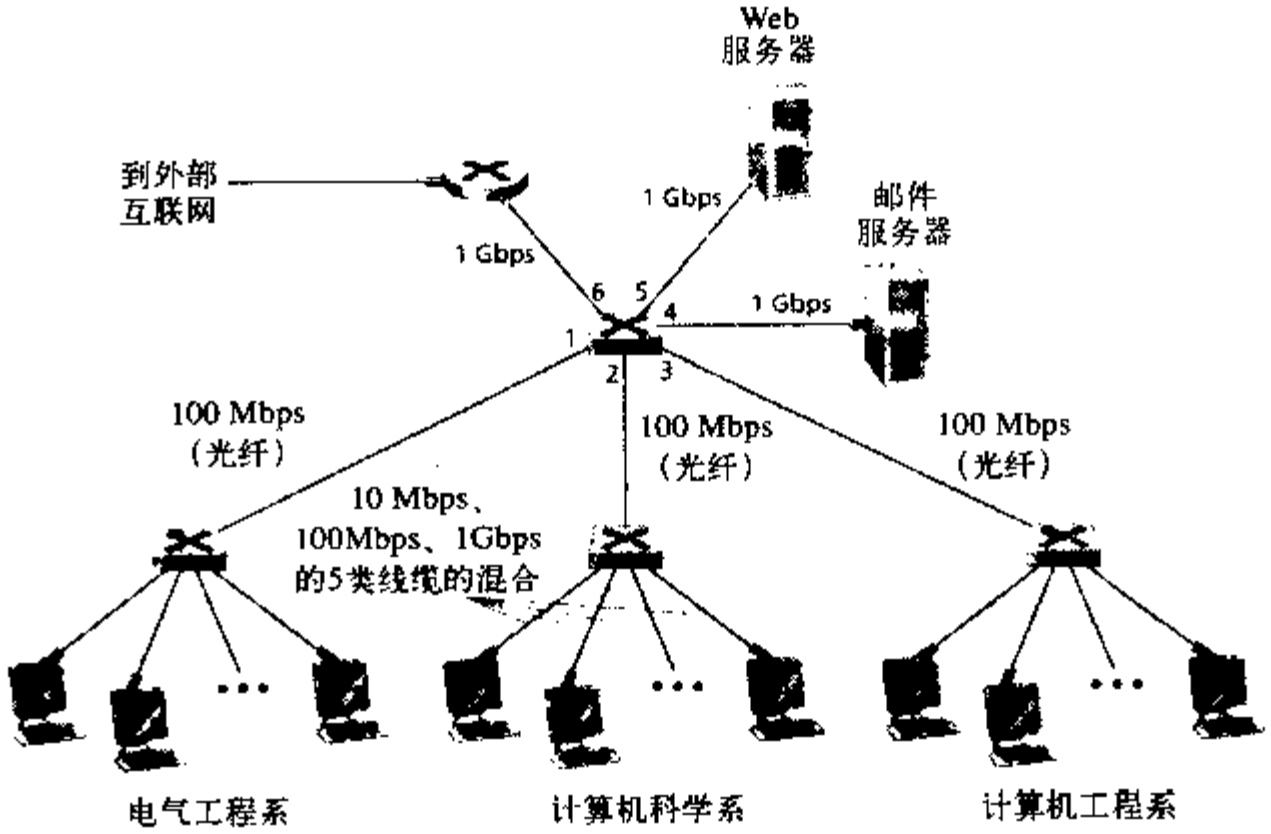


图5-26 使用集线器、以太网交换机和路由器的某机构网络

5.6.1 交换机转发和过滤

过滤（filtering）是交换机决定一个帧是应该转发到某个接口还是应当将其丢弃的功能。**转发**（forwarding）是决定一个帧应该被导向哪个接口，并把该帧接口移动到这些接口的交换机功能。交换机的过滤和转发借助于**交换机表**（switch table）完成。该交换机表包含某LAN上的某些节点但不必是全部节点的表项。交换机表中的一个表项包含：①节点的MAC地址，②到达该节点的交换机接口，③用于节点的表项放置在表中的时间。图5-27中显示了在图5-26中的最上面交换机的一个交换机表的例子。尽管帧转发的描述听起来类似于第4章讨论的数据转发，我们将很快看到它们之间有重要的差异。一个重要差异是交换机转发分组基于MAC地址而不是基于IP地址。我们也将看到交换机表与路由器的转发表的构造方式有很大差别。

为了理解交换机过滤和转发的工作过程，假定具有目的地址DD-DD-DD-DD-DD-DD的帧从交换机接口*x*到达。交换机用MAC地址DD-DD-DD-DD-DD-DD索引它的表。可能存在3种情况：

- 表中没有针对DD-DD-DD-DD-DD-DD的表项。在这种情况下，交换机向除了接口*x*外的所有接口前面的输出缓存转发该帧的拷贝。换言之，如果没有对于目的地址的表项，该交换机广播该帧。

- 表中有一个表项将DD-DD-DD-DD-DD-DD与接口x联系起来。在这种情况下，该帧从包括适配器DD-DD-DD-DD-DD-DD的LAN网段到来。无需将该帧转发到任何其他接口，该交换机通过丢弃该帧执行过滤功能即可。
- 表中有一个表项将DD-DD-DD-DD-DD-DD与接口$y \neq x$联系起来。在这种情况下，该帧需要被转发到与接口y相连的LAN网段。该交换机通过将该帧放到接口y前面的输出缓存区完成转发功能。

地　址	接　口	时　间
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
…	…	…

图5-27　图5-26中最上面交换机交换机表的一部分

我们粗略地看一下用于图5-26中最上面交换机的这些规则以及在图5-27中它的交换机表。假设目的地址62-FE-F7-11-89-A3的一个帧从接口1到达该交换机。交换机检查它的表并且发现其目的地是在与接口1相连的LAN网段上（即电子工程系的LAN）。这意味着该帧已经在包含目的地的LAN网段广播过了。因此该交换机过滤（即丢弃）了该帧。现在假设有同样目的地址的帧从接口2到达。交换机再次检查它的表并且发现其目的地址在接口1的方向上；因此它向接口1前面的输出缓存区转发该帧。这个例子清楚地表明，只要交换机的表是完整的和准确的，该交换机无需任何广播就能向着目的地转发帧。

在这种意义上，交换机比集线器更为“聪明”。但是在一开始这个交换机表是如何配置起来的呢？链路层有与网络层选路协议等价的协议吗？或者必须要管理员人工地配置交换机表？

5.6.2　自学习

交换机具有令人惊奇的特性（特别是对于辛劳的网络管理员），那就是它的表是自动地、动态地、自治地建立的，即没有来自网络管理员或配置协议的任何干预。换句话说，交换机是**自学习**（self-learning）的。这种能力是以如下方式实现的：

1）交换机表初始为空。

2）对于在某接口接收到的每个入帧，该交换机在其表中存储：①在该帧源地址字段中的MAC地址，②该帧到达的接口，③当前的时间。交换机以这种方式在它的表中记录发送节点所在LAN网段。如果在LAN上的每个节点最终都发送了一个帧，则每个节点将在这张表中被记录下来。

3）如果在一段时间（称为**老化期**（aging time））后，交换机没有接收到以该址作为源地址的帧，就在表中删除这个地址。以这种方式，如果一台PC被另一台PC（具有不同的适配器）代替，原来PC的MAC地址将最终从该交换机表中被清除掉。

我们粗略地看一下用于图5-26中最上面交换机的自学习性质以及图5-27中它对应的交换机表。假设在时间9:39，源地址为01-12-23-34-45-56的一个帧从接口2到达。假设这个地址不在交换机表中。于是交换机在其表中增加一个新的表项，如图5-28所示。

地　址	接　口	时　间
01-12-23-34-45-56	2	9:39
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
…	…	…

图5-28　交换机学习到地址为01-12-23-34-45-56的适配器所在位置

继续这个例子，假设该交换机的老化时间是60分钟，在9:32到10:32之间源地址是62-FE-F7-11-89-A3的帧没有到达该交换机。那么在时刻10:32，这台交换机将从它的表中删除该地址。

交换机是**即插即用设备**（plug-and-play device），因为它们不需要网络管理员或用户的干预。要安装交换机的网络管理员除了将LAN网段与交换机的接口相连外，不需要做其他任何事。管理员在安装交换机时或者当某主机从LAN网段之一被去除时，他没有必要配置交换机表。交换机也是双工的，这意味着对于任何将节点与交换机连接的链路，节点和交换机能够同时传输而无碰撞。

5.6.3　链路层交换机的性质

在描述了链路层交换机的基本操作之后，我们现在来考虑交换机的特色和性质。使用图5-24中的LAN，我们能够指出使用交换机的几个优点，这些不同于如总线或基于集线器的星型结构那样的广播链路：

- *消除碰撞*。在使用交换机（不使用集线器）构建的LAN中，没有因碰撞而浪费的带宽！交换机缓存帧并且决不会在网段上同时传输多于一个帧。就像使用路由器一样，交换机的最大聚合带宽是该交换机所有接口速率之和。因此，交换机提供了比使用广播链路的LAN高得多的性能改善。
- *异质的链路*。因为交换机将链路彼此相隔离，LAN中的不同链路能够以不同的速率运行并且能够在不同的媒体上运行。例如，在图5-24中，A可以用10 Mbps 10BASE-T铜缆连接，B可以用100 Mbps 10BASE-FX光缆连接，C可以用1 Gbps 1000BASE-T铜缆连接。因此，对于原有设备与新设备混用的情况，交换机是理想的。
- *管理*。除了提供强化的安全性（参见补充材料“关注安全”），交换机也易于进行网络管理。例如，如果一个适配器工作异常并持续发送以太帧（称为快而含糊的（jabbering）适配器），交换机能够检测到该问题，并在内部断开异常适配器。有了这种特色，网络管理员不用早起开车到工作场所去解决这个问题。类似地，一条割断的缆线仅使得使用该条割断缆线连接到交换机的节点断开连接。在使用同轴电缆的时代，许多网络管理员花费几个小时“沿线循检”（或者更准确地说“在天花板上爬行”），以找到使整个网络瘫痪的电缆断开之处。如在第9章（网络管理）中讨论的那样，交换机也收集带宽使用的统计数据、碰撞率和流量类型，并把这些信息提供给网络管理者使用。这些信息能够用于调试和解决问题，并规划该LAN在未来应当演化的方式。

关注安全

嗅探交换LAN：交换机毒化

当某节点与某交换机相连时，它通常仅接收到明确发送给它的帧。例如，在图5-24中的一个交换LAN。当节点A向节点B发送帧时，在交换机表中有用于节点B的表项，则该交换机将仅向节点B转发该帧。如果节点C恰好在运行嗅探器，节点C将不能嗅探到A到B的帧。因此，在交换LAN的环境中（与如802.11 LAN或基于交换机以太网LAN的广播链路环境形成对比），攻击者嗅探帧更为困难。然而，交换机广播目的地址不在交换机表中的那些帧，C上的嗅探器仍然能嗅探某些不是明确寻址到C的帧。此外，嗅探器将能够嗅探到具有广播地址FF-FF-FF-FF-FF-FF的广播帧。一种众所周知的对抗交换机的攻击称为**交换机毒化**（switch poisoning），它向交换机发送大量的具有不同伪造源MAC地址的分组，因而用伪造表项填满了交换机表，没有为合法节点留下空间。这使得该交换机广播大多数帧，这些帧则能够由嗅探器所俘获到[Skoudis 2006]。由于这种攻击只有技艺高超的攻击者才能做到，因此交换机比集线器和无线LAN更难以受到嗅探。

5.6.4 交换机和路由器的比较

如我们在第4章学习的那样，路由器是使用网络层地址转发分组的存储转发分组交换机。尽管交换机也是一个存储转发分组交换机，但它和路由器是根本不同的，因为它用MAC地址转发分组。交换机是第二层的分组交换机，而路由器是第三层的分组交换机。

尽管交换机和路由器从根本上是不同的，但网络管理员在安装互联设备时经常必须在它们之间进行选择。例如，对于图5-26中的网络，网络管理员本来可以很容易地使用路由器而不是交换机来互联各个系的LAN、服务器和互联网网关路由器。路由器的确使得各系之间通信而不产生碰撞。给定交换机和路由器都是候选的互联设备，这两种方式的优点和缺点各是什么呢？

首先考虑交换机的优点和缺点。如上面提到的那样，交换机是即插即用的，这是所有的网络管理员都喜爱的特性。交换机还能够具有相对高的分组过滤和转发速率，这就像图5-29中所示的那样，交换机必须处理最高仅为第二层的帧，而路由器必须处理最高为第三层的数据报。在另一方面，为了防止广播帧的循环，交换网络的活跃拓扑限制为一棵生成树。还有，大型交换网络要求在该节点中有大的ARP表，这将生成可观的ARP流量和处理量。此外，交换机对于广播风暴并不提供任何保护措施，即如果某主机出了故障并传输出没完没了的以太网广播帧流，该交换机将转发所有这些帧，使得整个以太网崩溃。

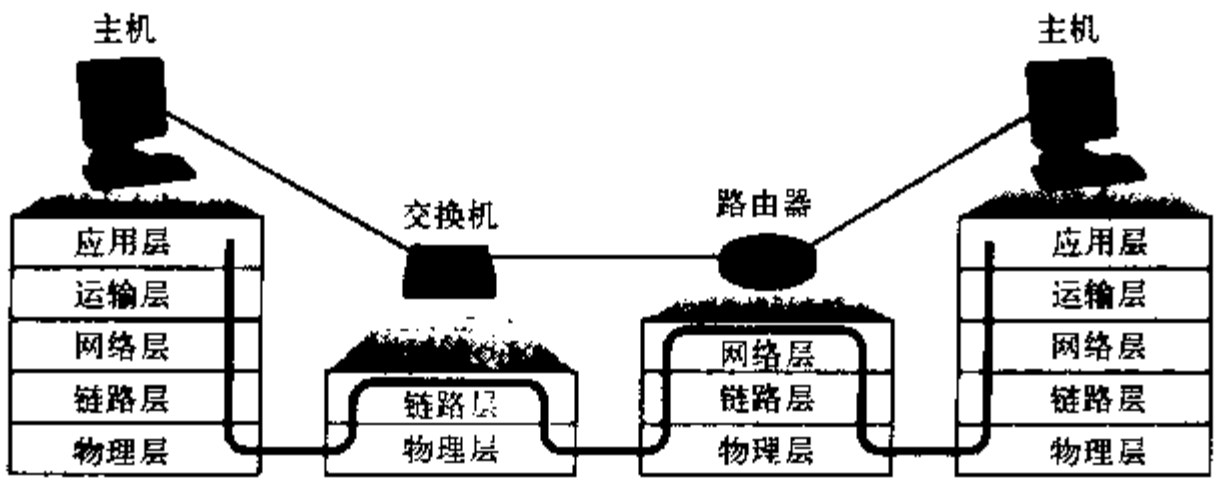


图5-29 在交换机、路由器和主机中分组的处理

现在考虑路由器的优点和缺点。因为网络寻址通常是分层次的（不像MAC寻址那样是扁平的），即使当网络中存在冗余路径，分组通常也不会通过路由器循环。（然而，当路由器表被误配置时，分组能够循环；但是如我们在第4章所学的那样，IP用一个特殊的报文首部字段来限制循环。）所以，分组就不会被限制到一棵生成树上，并可以使用源和目的之间的最佳路径。因为路由器没有生成树限制，所以它们能用各种拓扑结构来构建因特网，例如包括欧洲和北美之间的多条活跃链路。路由器的另一个特色是它们对第二层的广播风暴提供了防火墙保护。尽管也许路由器最重要的缺点就是它们不是即插即用的，即路由器和连接到它们的主机都需要人为地配置IP地址。而且路由器对每个分组的处理时间通常比交换机更长，因为它们必须处理最高达第三层的字段。最后，*路由器*一词有两种不同的发音方法，或者发音为“rootor”，或者发音为“rowter”，人们浪费了许多时间争论正确的发音[Perlman 1999]。

给出了交换机和路由器都具有各自的优点和缺点后，一个机构的网络（例如，大学校园网或者公司园区网）什么时候应该使用交换机，什么时候应该使用路由器呢？通常，由几百台主机组成的小网络通常有几个LAN网段。对于这些小网络，交换机就足够了，因为它们不要求IP地址的任何配置就能使流量局部化并增加总计吞吐量。但是在由几千台主机组成的更大网络中通常在网络中（除了交换机之外）还包括路由器。路由器提供了更健壮的流量隔离方式和对广播风暴的控制，并在网络的主机之间使用更“智能的”路由。

我们在本节学习了集线器、交换机和路由器都能够作为主机和LAN网段的互联设备。表5-1总结了这些互联设备的特点。

表5-1 流行的互联设备的典型特色的比较

	集线器	路由器	交换机
流量隔离	无	有	有
即插即用	有	无	有
优化选路	无	有	无
直通交换	有	无	有

5.7 PPP：点对点协议

我们对链路层协议的讨论大部分到目前为止都集中在用于广播信道的协议上。在本节中我们讨论用于点对点链路的链路层协议，即点对点协议PPP。因为PPP通常是住宅主机拨号链路所选择的协议，所以它无疑是目前部署最广泛的链路层协议之一。目前使用的另一个重要的链路层协议是高级数据链路控制（High-level Data Link Control, HDLC）协议；有关HDLC的讨论见[Spragins 1991]。我们这里对较简单的PPP协议的讨论，将使我们了解到许多点对点链路层协议的重要特色。

顾名思义，点对点协议（PPP）[RFC 1661；RFC 2153]是一个运行于**点对点链路**（point-to-point link）之上的链路层协议，即一条直接连接两个节点的链路，链路的每一端有一个节点。PPP运行的点对点链路可能是一条串行的拨号电话线（例如一条56 K调制解调器连接），一条SONET/SDH链路，一条X.25连接或者一条ISDN电路。如上所述，PPP已经成为家庭用户与ISP通过拨号建立连接所选择的协议。

在深入讨论PPP协议细节之前，分析一下IETF对PPP设计所提出的初始要求[RFC 1547]很

有帮助：

- 分组成帧。PPP协议链路层的发送方必须能够携带网络层的分组，并将它封装在PPP链路层帧中，以便接收方能够确定链路层帧的起始和结束位置和该帧中网络层分组。
- 透明性。PPP协议不能对出现在网络层分组中的数据（首部或者数据）做任何限制。因此，例如PPP协议不能够禁止在网络层分组中使用某种比特模式。我们将很快在讨论字节填充时再研究这个问题。
- 多种网络层协议。PPP协议必须能够支持同时运行在相同的物理链路的多种网络层协议（例如IP和DECnet）。正如需要IP协议在单一端到端连接上多路复用不同的传输层协议（如TCP和UDP）一样，PPP也必须能够在单一点对点连接上多路复用不同的网络层协议。这个要求意味着PPP至少可能需要一个协议类型字段或者某种相似的机制，这样接收方的PPP就能够将接收的帧向上分解到合适的网络层协议。
- 多种类型链路。除了能够承载多种更高层的协议之外，PPP还必须能够运行在各种不同链路类型上，包括串行的（在给定的方向一次传输一个比特）、并行的（并行传输比特）、同步的（和数据比特一起传输时钟信号）或者异步的、高速的或低速的、电的或光的链路类型。
- 差错检测。PPP接收方必须能够检测到接收帧中的比特差错。
- 连接的活性。PPP必须能够检测到链路层次的故障（例如不能把数据从链路发送方传输到链路接收方），并且向网络层通知该差错的情况。
- 网络层地址协商。PPP必须为通信的网络层（例如IP）提供一个机制，来获知或者配置相互的网络层地址。
- 简单性。除了上面列出的要求外，还要求PPP满足一些附加要求。在所有这些要求中，第一和最重要的是简单性。RFC 1547指出，“点对点协议的格言应该是简单性”。对PPP的设计提出所有其他的要求的确很苛刻！现在有50多个RFC文档定义了这个“简单”协议的各个方面。

尽管看起来对PPP设计提出了很多要求，但情况实际上要困难得多！PPP的设计规范也明确提到了不要求PPP实现的协议功能：

- 差错纠正。要求PPP能够检测比特差错，但不要求纠正它们。
- 流量控制。期望一个PPP接收方能够以支撑的物理层的全部速率来接收帧。如果某较高层不能够以这种全速接收分组，那么就该由较高层负责丢弃分组或者遏制位于较高层的发送方。这就是说，不是由PPP发送方遏制自己的传输速度，而是由较高层协议负责遏制分组交付给PPP的发送速率。
- 有序。PPP不要求向链路接收方交付帧的顺序与链路发送方发送帧的顺序相同。注意到下列事实是有趣的：尽管这种灵活性与IP服务模型（它允许IP分组以任何次序在端到端交付）相兼容，但是工作于PPP之上的其他网络层协议的确需要有序地进行端到端分组交付。
- 多点链路。PPP只需要工作于那些具有单个发送方和单个接收方的链路之上。其他链路层协议（例如HDLC）能够在一条链路上容纳多个接收方（例如，类似以太网的情况）。

考虑了PPP的设计目标（和非设计目标）之后，我们来看看PPP的设计是如何满足这些目标的。

PPP数据成帧

图5-30表示了一种使用类似HDLC成帧方法的PPP数据帧[RFC 1662]。PPP帧包含了以下字段：

- 标志字段。每个PPP帧都是用值为01111110的1 字节的标志字段作为开始和结束。
- 地址字段。这个字段唯一可能的值是11111111。
- 控制字段。这个字段唯一可能的值是00000011。地址字段和控制字段都只能取一个固定值，那么为什么要定义这些字段呢。PPP规范[RFC 1662]指出其他值“可能在以后定义”，然而到现在还没有定义。因为这些字段取了固定值，PPP允许发送方不发送地址和控制字节，这样在PPP帧中省下了两个字节的开销。
- 协议。该协议字段告诉PPP接收方所接收的封装数据（即PPP帧信息字段的内容）所属的上层协议。一收到PPP帧，PPP接收方就检测该帧的正确性，然后将封装的数据传递给适当的协议。RFC 1700和RFC 3232定义了PPP使用的16比特协议代码。我们感兴趣的是IP协议（即封装在PPP帧中的数据是一个IP数据报），它的值为十六进制的21；其他网络层协议，如AppleTalk（29）和DECnet（27）。
- 信息。这个字段包含上层协议（例如IP）在PPP链路上发送的被封装分组（数据）。该信息字段最大的默认长度是1500字节，然而当链路首次配置时能够改变该字段的长度，这个下面会讨论。
- 检验和。检验和字段用于检测已传输帧中的比特差错。它使用2或4 字节的HDLC标准的循环冗余码。

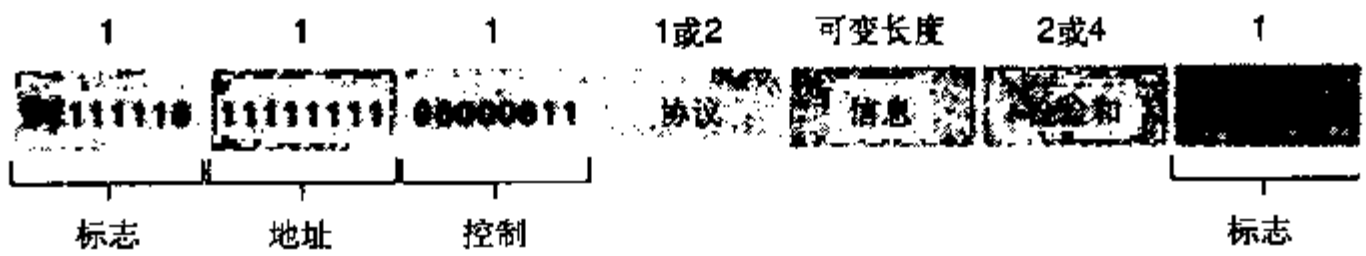


图5-30 PPP数据帧格式

字节填充

在结束PPP成帧的讨论之前，我们考虑一下当任意协议在标志字段使用特殊比特模式定界该帧的起始或结束时会出现的一个问题。如果标志模式本身在分组中其他地方出现会发生什么现象呢？例如，如果标志字段的值01111110出现在信息字段中会发生什么现象呢？接收方会错误地检测为PPP帧结束吗？

解决这个问题的一种办法是让PPP禁止上层协议发送包含标志字段比特模式的数据。上面讨论的对PPP的透明性要求排除了这种可能性。在PPP和其他许多协议中所采用的另一种解决方案，是使用一种称为**字节填充**（byte stuffing）的技术。

PPP定义了一个特殊的控制转义字节01111101。如果标志序列01111110在该帧中出现在除标志字段以外的任何地方，PPP就在该标志模式实例之前插入这样的控制转义字节。这就是说，它在01111110前“填充”（加）一个控制转义字节进入传输的数据流中，以指示随后的01111110不是一个标志值，而事实上是真正的数据。当然，接收方看到01111110之前有一个01111101，会去除填充的控制转义字节来重建初始数据。类似地，如果控制转义字节的比特模式自身作为实际数据出现，它前面也必须填充一个控制转义字节。因此，当接收方在数据

流中看到单个控制转义字节时，它知道该字节是被填充到数据流中的。相继出现的一对控制转义字节意味着在要发送的初始数据中出现了一个控制转义字节的实例。图5-31图示了PPP字节填充过程。（实际上，PPP还用十六进制的20“异或”被转义的数据字节，为了简单起见这里我们省略了细节。）

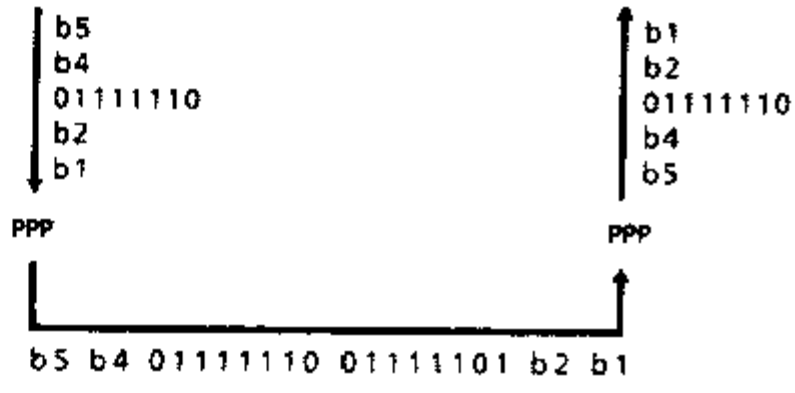


图5-31 字节填充

5.8 链路虚拟化：网络作为链路层

本章关注链路层协议，在我们即将结束本章的讨论时，让我们反思一下我们对已经演化的词汇链路的理解。在本章开始时，我们将链路视为连接两台通信主机的物理线路，如在图5-2中所示那样。在学习多路访问协议时（图5-9），我们看到了多台主机能够通过一条共享的线路连接起来，并且连接主机的这种“线路”能够是无线电频谱或其他媒体。这使我们思考该链路能够更进一步抽象为一条信道，而不是作为一条线路。在我们学习以太网LAN时（图5-26）时，我们看到互联媒体能够是一种相当复杂的交换基础设施。然而，经过这种演化，主机本身维持着这样的观点，即互联媒体只是连接两台或多台主机的链路层信道。例如我们看到，一台以太网主机可以不知道它是通过一个单一短LAN网段（图5-9）还是一个地理上分布的交换LAN（图5-26）与其他LAN主机进行连接。

在5.7节，我们看到在经调制解调器连接的两台主机之间常使用PPP协议。这里，连接这两台主机的链路实际上是电话网，这是一个逻辑分离、全球性的电信网络，它有自己的交换机、链路和协议栈用于数据传输和信令。然而，从因特网链路层的观点看，通过电话网的拨号连接被看成是一根简单的“线路”。从这个意义上，因特网虚拟化了电话网，将电话网视为为两台因特网主机之间提供链路层连接的链路层技术。回想一下第2章中我们对覆盖网络的讨论，类似地，一个覆盖网络将因特网视为为覆盖节点之间提供连接性的一种手段，寻求以因特网覆盖电话网的相同方式来覆盖因特网。

在本节中，我们将考虑异步传递方式（ATM）和多协议标签交换（MPLS）网络。与电路交换的电话网不同，ATM和MPLS就其自身而言都是分组交换的虚电路网络。它们有自己的分组格式和转发行为。因此，从教学法的观点看，有关ATM和MPLS的讨论既适合放在网络层的学习中，也适合放在链路层的学习中。然而，从因特网的观点看，我们认为ATM和MPLS像电话网和交换以太网一样，是为IP设备提供互联服务的链路层技术。因此，我们将在链路层讨论中考虑ATM和MPLS。帧中继网络也能用于互联IP设备，虽然它们看上去有些过时（但仍在部署）的技术，这里将不加讨论；详情请参见参考书[Goralski 1999]。我们对ATM和MPLS的讨论简明扼要，有关这些网络技术的讨论每个都能够写（并且已经写了）一本书。我们在这里主要关注这些网络怎样为互联IP设备提供服务，尽管我们也会更深入地探讨一些支撑基础技术。

5.8.1 异步传输方式

异步传输方式（Asynchronous Transfer Mode, ATM）网络的标准最初在20世纪80年代中期开发起来，其目标是设计一种单一的网络技术，传输实时音频、视频以及文本、电子邮件和图像文件。ATM论坛（现在称为MFA论坛[MFA Forum 2007]）和国际电信联盟[ITU 2007]

参与了ATM标准的开发。它们定义了一套完整的端到端标准，从应用程序到ATM接口的规约，到通过各种光纤、铜线和无线电物理层的ATM数据的比特级成帧。实践中，ATM主要用于电话和IP网络，例如为连接IP路由器提供链路层技术服务，如上所述。

1. ATM的主要特性

如4.1节所述，ATM支持几种服务模式，包括恒定比特率服务、可变比特率服务、可用比特率服务和不指明比特率服务。ATM是一种分组交换、虚电路（VC）网络体系结构。在4.2.1节中我们已经用一定篇幅讨论过虚电路。如图5-32所示，ATM的总体体系结构组织为3个层次。

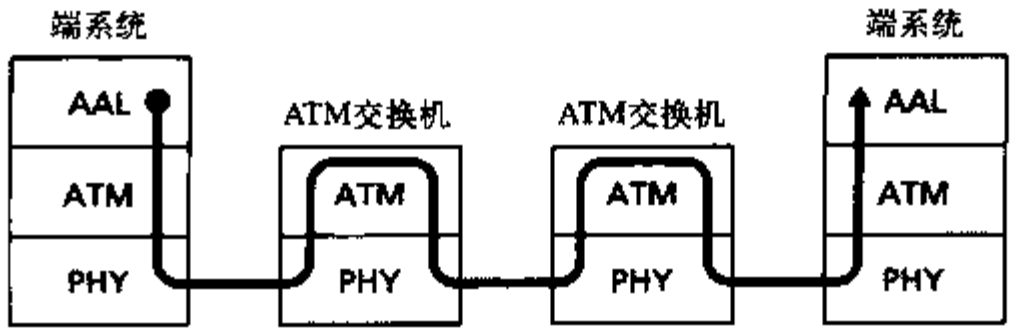


图5-32 3个ATM层次。AAL层仅出现在ATM网络的边缘

ATM适配层（ATM Adaptation Layer, AAL）大致类比于因特网的运输层，并且仅出现在位于ATM网络边缘的ATM设备上。在发送端，AAL从较高层应用程序或协议（如IP，如果ATM被用于连接IP设备的话）传递数据。在接收端，它向上向较高层协议或应用程序传递数据。AAL已经被定义用于恒定比特率服务和电路仿真（AAL1），用于如可变比特率视频这样的可变比特率服务（AAL2），以及用于如IP数据报传输这样的数据服务（AAL5）。在由AAL执行的服务中有差错检测和分段/重装。由AAL处理的数据单元称为AAL协议数据单元（Protocol Data Unit, PDU）——很普通的一个名字，它大致等价于UDP或TCP报文段。

在图5-33中显示了AAL5 PDU。该PDU的字段相对直接明了。PAD确保了该PDU是48 字节的整数倍，因为该PDU将被分段，以适合支撑的ATM分组的48 字节的载荷（称为ATM信元）。长度字段指示了PDU有效载荷的长度，以便在接收方能够去除该PAD。CRC字段使用与以太网相同的循环冗余校验，以提供差错检测。载荷字段能够长达65 535字节。

图5-33 AAL5 PDU

现在我们向下走一层来考虑ATM层（ATM layer），该层位于ATM体系结构的核心。ATM层定义了ATM信元的结构和该信元中各个字段的含义。ATM信元对于ATM网络就像IP数据报对于IP网络一样重要。信元的前5个字节构成了ATM首部；余下的48 字节构成了有效载荷。图5-34显示了ATM信元首部的结构。

ATM信元中的各字段具有下列功能：

- 虚通道标识（VCI）。指示信元属于哪个虚通道。和大多数使用虚电路的网络技术一样，信元的VCI在从一条链路到另一条链路的过程中发生转换（见4.2.1节）。
- 负载类型（PT）。指示包含在信元中的有效载荷的类型。有几种数据有效载荷类型，几种维护有效载荷类型和一种空闲信元有效载荷类型。该PT字段还包括一个比特用于指示

在一个AAL PDU中的最后一个信元。

- 信元丢失优先级（CLP）比特。由源端来设置，用来区分高优先级流量和低优先级流量。如果发生了拥塞并且ATM交换机必须丢弃信元，交换机能够使用该比特来首先丢弃低优先级的流量。
- 首部差错控制（HEC）字节。保护信元首部的差错检测比特。

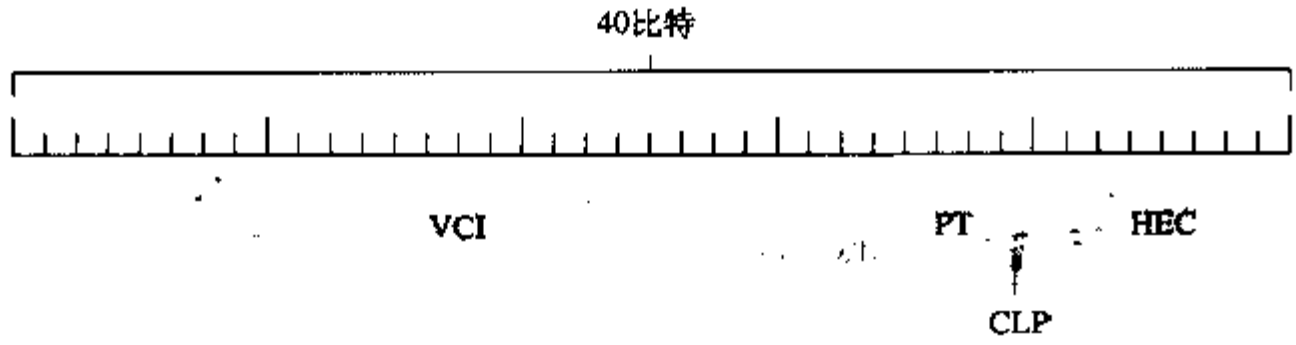


图5-34 ATM信元首部的结构

在源能够向目的地开始发送信元之前，ATM网络必须首先创建一条从源到目的地的虚通道（Virtual Channel, VC）。一条虚通道不过是如在4.2.1节中所描述的一条虚电路。每个VC是由源和目的地之间的链路序列组成的路径。一个**虚通道标识**（Virtual Channel Identifier, VCI）与该VC上的每条链路相联系。无论何时创建或拆除一个VC，必须更新VC转换表（参见4.2.1节）。如果使用永久VC，动态创建和拆除VC就没有必要了。当要求动态创建和拆除VC，Q.2931协议[Black 1977, ITU-T Q.2931 1994]在ATM交换机和端系统之间提供了所需要的信令。

ATM物理层（ATM physical layer）位于ATM协议栈的最底层，处理物理媒体上的电压、比特定时和成帧。许多物理层都依赖于链路的物理特性。有两大类物理层：一类具有传输帧结构（如T1、T3、SONET或SDH），一类则没有这种结构。如果物理层具有帧结构，则它负责产生和定界帧。这里使用术语帧不应当与本章前面所使用的链路层（例如以太网）帧相混淆。这里帧的传输是一种物理层类似于TDM的机制，用于组织比特在链路上发送。

2. 在ATM上传输IP

现在我们考虑ATM网络用于提供IP设备之间的连通性的方法。图5-35显示了用于因特网IP流量的具有4个入口/出口点的ATM主干。注意到每个入口/出口点是一台路由器。一个ATM主干能够跨越整个大陆，可能具有数十台甚至数百台ATM交换机。多数ATM主干在每对入口/出口点具有一个永久的VC。通过使用永久的VC，ATM信元从一个入口点路由到一个出口点，而不必动态地创建和拆除VC。然而，永久的VC仅当入口/出口点数量相对少时才是切实可行的。对于n个入口点，为了直接连接n个入口/出口点需要$n(n-1)$个永久VC。

连接到ATM网络的每台路由器接口将需要两个地址，与一台IP主机具有两个以太网接口地址一样：一个IP地址和一个MAC地址。类似地，一个ATM接口将具有一个IP地址和一个ATM地址。现在考虑一个IP数据报跨越如图5-35所示的ATM网络的情况。在最简单的情况下，ATM网络看起来像一条单一的逻辑链路，即ATM互联这4台路由器就像以太网能被用于互联4台路由器一样。我们称其为路由器，是因为数据报进入ATM网络就像进入“入口路由器”，并且数据报离开该网络就像离开“出口路由器”。入口路由器完成下列工作：

1）检查该数据报的目的地址。

2）索引它的选路表，决定出口路由器IP地址（即在该数据报的路径中的下一台路由器）。

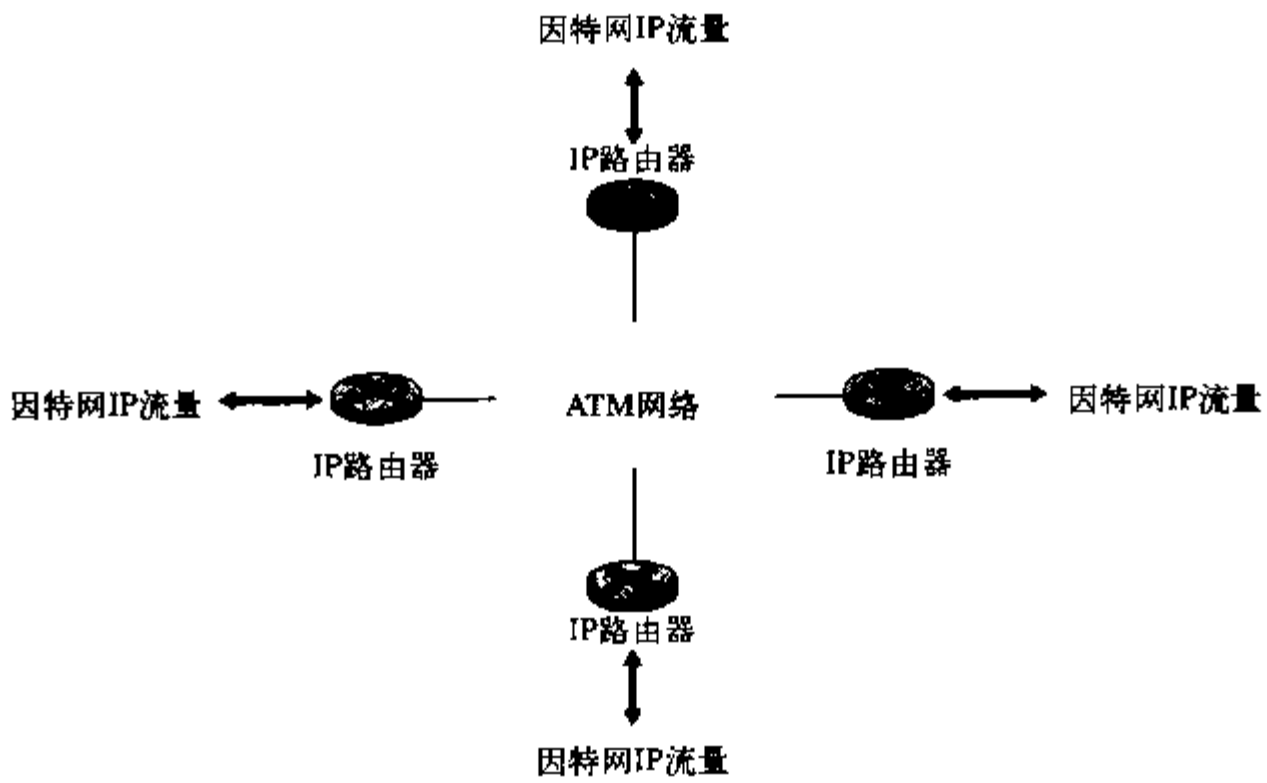


图5-35 ATM网络位于一个因特网主干的核心

3）为使该数据报到达出口路由器，入口路由器将ATM视为另一种链路层协议。为了将数据报移动到下一台路由器，我们必须决定下一跳路由器的物理地址。我们在5.4.2节讲过，这使用ARP来完成。在ATM接口的情况下，入口路由器用出口路由器的IP地址来索引ATM ARP表，并决定出口路由器的ATM地址。[RFC 2225]中描述了ATM ARP协议。

4）入口路由器中的IP将该数据报连同出口路由器的ATM地址，向下传递到链路层（即ATM）。

在完成这4步后，将数据报移动到出口路由器的工作就不由IP负责而由ATM负责了。ATM现在必须将该数据报移动到从步骤3获得的ATM目的地址处。该任务具有两个子任务：

1）确定通向ATM目的地址的虚通道的VCI。

2）在虚通道的发送端（即入口路由器）将数据报分段为信元，在该虚通道的接收端（即出口路由器）将这些信元重装为初始数据报。

第一个子任务是直接的。在发送端的接口维护一张将ATM地址映射为VCI的表。因为我们假定虚通道是永久的，该表是静态的和最新的。（如果虚通道不是永久的，则需要用ATM Q.293信令协议动态地创建和拆除该虚通道。）第二个任务值得更仔细地考虑。一种方法是使用如4.4节讨论的IP分段。使用IP分段，发送路由器将首先将初始数据报分成报文段，每个报文段不超过48 字节，因此该分段能够适合ATM信元的有效载荷。但是这种分段方法存在大问题，即每个IP报文段通常有20字节的首部，因此一个ATM信元携带一个报文段将有25 字节的额外开销，仅有28 字节的有用信息。ATM因此使用AAL5提供更为有效的数据报的分段/重装。

ATM网络接下来跨越网络向ATM目的地址移动每个信元。在ATM源和ATM目的地之间的每台ATM交换机，ATM信元由ATM物理层和ATM层处理，但不被AAL层处理。在每台交换机，通常要转换VCI（参见4.2.1节）并且重计算HEC。当信元到达了ATM目的地址，它们被导向已被分配给特殊虚通道的AAL缓存。然后重新构造AAL5 PDU，提取出IP数据报，并向上传递给IP层的协议栈。

5.8.2 多协议标签交换

通过采用来自虚电路网络领域的一个关键概念——固定长度标签，可以改善IP路由器的转发速度，在一些产业界的努力下，多协议标签交换（Multiprotocol Label Switching, MPLS）在20世纪90年代中后期演化产生。其目标是对于基于固定长度标签和虚电路的技术，在不放弃基于目的地IP数据报转发的基础设施的前提下，当可能时通过选择性地标识数据报并允许路由器基于固定长度的标签转发数据报（而不是目的地IP地址），从而增强其功能。重要的是，这些技术与IP协同工作，使用IP寻址和选路。IETF在MPLS协议中统一了这些努力[RFC 3031, RFC 3032]，有效地将虚通道技术综合进了选路的数据报网络。

下面我们开始研究MPLS，首先考虑由MPLS使能的路由器处理的链路层帧的格式。图5-36显示了在一条PPP链路或LAN（如以太网）上传输的链路层帧具有一个小的MPLS首部，该首部增加到第二层（如PPP或以太网）首部和第三层（如IP）首部之间。RFC 3032定义了用于这样链路的MPLS首部的格式，用于ATM和帧中继网络的首部也定义在其他的RFC文档中。在MPLS首部的其他字段间是标签（它起着虚电路标识符的作用，我们已经在4.2.1节中讨论过该标识符），3比特保留用于实验，单个S比特用于指示一系列的“成栈”的MPLS首部的结束（这是一个高级主题，我们这里不加讨论），以及寿命字段。

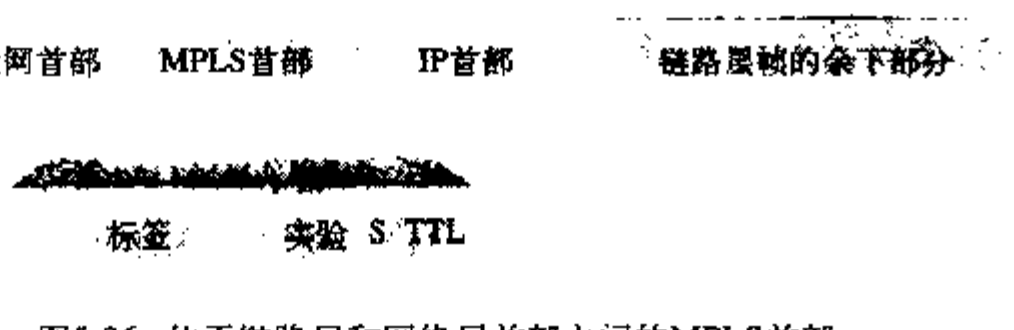


图5-36 位于链路层和网络层首部之间的MPLS首部

从图5-36立即能够看出，一个MPLS加强的帧仅能在两个均为MPLS使能的路由器之间发送。（因为一个非MPLS使能的路由器，当它在期望发现IP首部的地方发现了一个MPLS首部将会相当混淆！）一个MPLS使能的路由器常被称为**标签交换路由器**（label-switched router），因为它通过在其转发表中查找MPLS标签来转发MPLS帧，然后立即将数据报传递给适当的输出接口。所以，MPLS使能的路由器不需要提取目的IP地址和在转发表中执行最长前缀匹配的查找。但是路由器怎样才能知道它的邻居是否的确是MPLS使能的呢？路由器如何知道哪个标签与给定IP目的地相联系的呢？为了回答这些问题，我们将需要看看在一组MPLS使能路由器之间的交互过程。

在图5-37所示的例子中，路由器R1到R4都是MPLS使能的，R5和R6是标准的IP路由器。R1向R2和R3通告了它（R1）能够路由到目的地A，并且具有MPLS标签6的接收帧将要转发到目的地A。路由器R3已经向路由器R4通告了它能够路由到目的地A和D，分别具有MPLS标签10和12的入帧将被朝着这些目的地交换。路由器R2也向路由器R4通告了它（R2）能够到达目的地A，具有MPLS标签8的接收帧将朝着A交换。注意到路由器R4现在处于一个有趣的位置——到达A有两个MPLS路径，经接口0具有出MPLS标签10，经接口1具有出MPLS标签8。在图5-37中画出的外围部分是IP 设备R4、R5、A和D，它们经过一个MPLS基础设施（MPLS使能路由器R1、R2、R3和R4）连接在一起，这与一个交换LAN或ATM网络能够将IP设备连接到一起的方式十分相似。并且与交换LAN或ATM网络相似，MPLS使能路由器R1到R4完成这些工作时甚至没有接触一个分组的IP首部。

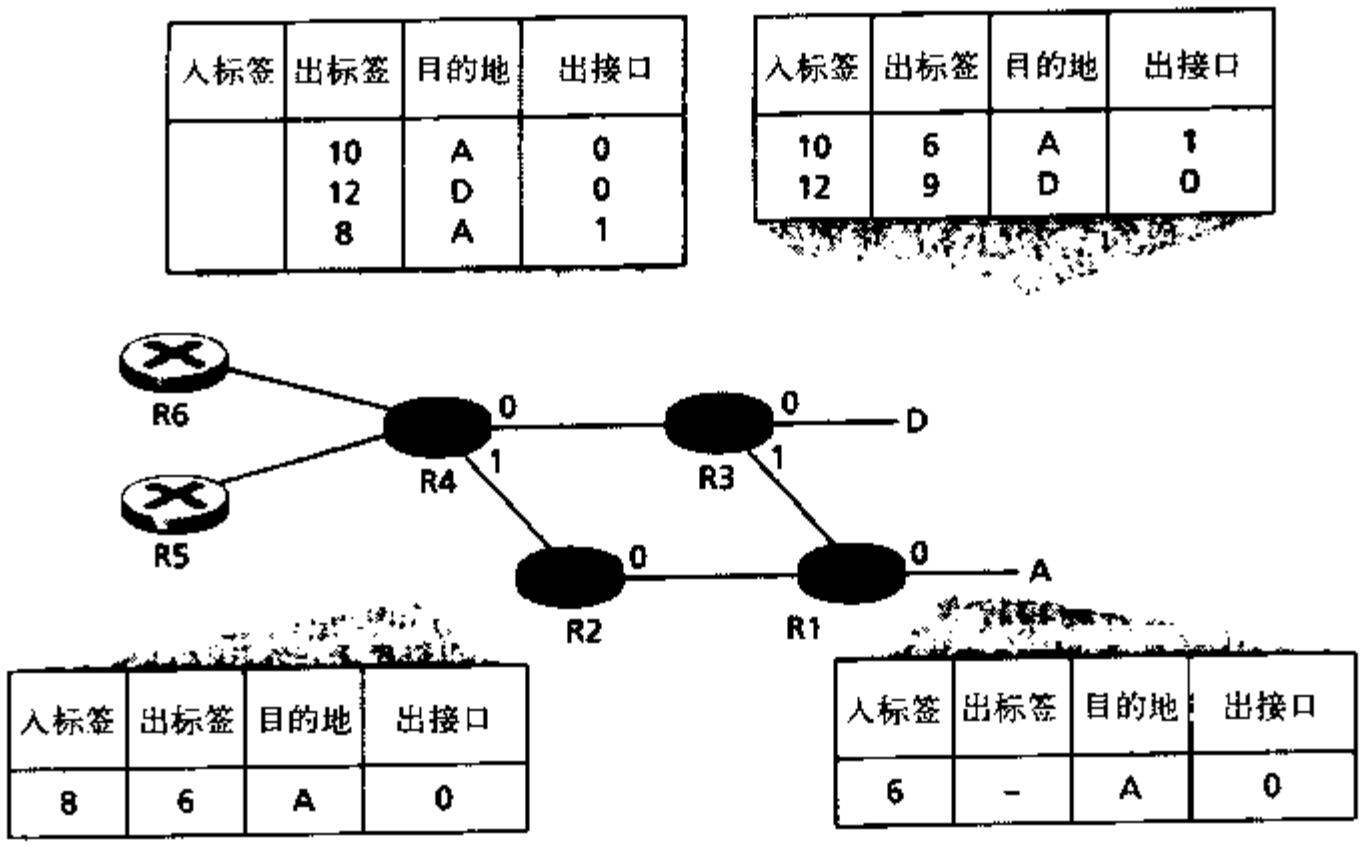


图5-37 MPLS加强的转发

在我们上面的讨论中，我们并没有指定在MPLS使能路由器之间分布标签的特定协议，因为该信令的细节已经超出了本书的范围。然而，我们注意到，IETF的MPLS工作组已经在[RFC 3468]中定义了RSVP协议的一种扩展（我们将在第7章中学习），称之为RSVP-TE[RFC 3209]，它将关注对MPLS信令的努力。所以，我们希望感兴趣的读者查阅RFC 3209。

至今为止，我们对MPLS的讨论重点基于这样的事实：MPLS基于标签执行交换，而不必考虑分组的IP地址。然而，MPLS的真正优点和当前对MPLS感兴趣的原因，并不在于潜在能增加交换的速度，而在于MPLS使能的新的流量管理的能力。如前面所述，R4到A具有两条MPLS路径。如果转发在IP层基于IP地址执行，我们在第4章中学习的IP选路协议将只指定到A的单一的最小费用的路径。所以，MPLS提供了沿着多条路由转发分组的能力，使用标准IP选路协议这些路由将是不可能的。这是使用MPLS的一种简单形式的**流量工程**（traffic engineering）[RFC 3346, RFC 3272, RFC 2702, Xiao 2000]，其中网络运行者能够超越普通的IP选路，迫使某些流量沿着一条路径朝给定的目的地引导，并且朝着相同目的地的其他流量沿着另一条路径流动（无论是由于策略、性能或某些其他原因）。

也能将MPLS用于其他目的。它能用于执行MPLS转发路径的快速恢复，例如对链路故障作出反应，经过一条预计算的无故障路径重新路由流量[Kar 2000, Huang 2002, RFC 3469]。MPLS还能用于实现区分服务框架（diff-serv），我们将在第7章中研究该框架。最后，我们注意到MPLS能够并且已经被用于实现所谓**虚拟专用网**（Virtual Private Network, VPN）。在为用户实现一个VPN的过程中，ISP使用它的MPLS使能网络以将用户的各种网络连接在一起。MPLS能被用于将由用户的VPN使用的资源和寻址方式与其他跨越该ISP网络的用户相隔离，详情参见[DeClercq 2002]。

有关MPLS的讨论是简要的，鼓励读者参阅我们提到的参考文献。我们注意到MPLS的使用有许多可能的方式，看起来它将迅速成为因特网流量工程的瑞士军刀！

5.9 小结

在这一章中，我们研究了链路层，包括它的服务、支撑它的操作的原则和许多重要的特定协议，这些协议使用这些原则实现了链路层服务。

我们看到链路层的基本服务是将网络层的数据报从一个节点（路由器或主机）移动到一个相邻的节点。我们看到，在通过链路向相邻节点传输之前，所有链路层协议都是通过将网络层数据报封装在链路层帧中来进行的。然而，除了这个共同的成帧功能之外，我们知道了不同的链路层协议提供截然不同的链路接入、交付（可靠性、差错检测/纠正）、流量控制和传输（如全双工与半双工）服务。造成这些差异的部分原因是链路层协议必须工作在很多种链路类型上。一个简单的点对点链路有单个发送方和接收方经单一“线路”通信。一个多路访问链路在许多发送方和接收方之间共享；因此，对多路访问信道的链路层协议有一个协调链路接入的协议（它的多路访问协议）。在ATM和MPLS的情况下，连接两个相邻节点（例如，在IP意义上的两台相邻的IP路由器，它们是到某个目的地的下一跳IP路由器）的“链路”，其本身可能实际上就是一个网络。在某种意义来说，将一个网络视为一条“链路”的想法没有什么可奇怪的。例如，连接家庭调制解调器/计算机到远端调制解调器/路由器的一条电话链路，实际上是一条穿过精密复杂的电话网络的路径。

在链路层通信所依据的原理中，我们研究了差错检测和纠错技术、多路访问协议、链路层寻址以及通过集线器和交换机来扩展局域网的构造方法。在差错检测/纠错的情况下，为了对帧通过链路传输时可能发生的比特翻转进行检测并在某些情况下进行纠正，我们研究了在帧的首部增加附加比特的可能方法。我们讨论了简单的奇偶校验和检验和方案，以及更健壮的循环冗余检测。然后我们转向多路访问协议的主题上。我们确定和学习了协调广播信道访问的3大类方法：信道划分方法（TDM、FDM）、随机接入方法（ALOHA协议和CSMA协议）和轮流方法（轮询和令牌传递）。我们看到让多个节点共享单个广播信道的结果，是需要在链路层提供节点地址。我们知道物理地址和网络层地址是非常不同的，而且在因特网的情况下，有个专门的协议（ARP，即地址解析协议）用于在这两种寻址形式之间进行转换。然后我们研究了共享一个广播信道的节点是怎样形成一个LAN的，以及多个LAN怎样互联形成一个更大的LAN，即互联这些本地节点完全不需要网络层选路的干预。

我们也详细讨论了若干特定的链路层协议，即以太网和PPP。我们通过关注当ATM和MPLS网络互联IP路由器时是如何提供链路层服务的，而结束了链路层的学习。在学习了链路层后，我们沿协议栈向下的旅程现在结束了！当然，物理层位于数据链路层之下，但是学习物理层的细节也许最好留给另外一个课程（例如，通信原理课程而不是计算机网络课程）。然而我们在本章（例如，我们在5.5节简要讨论了曼彻斯特编码）和在第1章（我们在1.2节讨论了物理媒体）中已经涉及了物理层的几个方面。当我们在下一章中学习无线链路特性时，将再次考虑物理层。

尽管我们沿协议栈向下的旅程已结束，但我们计算机网络的学习仍然没有结束。在后面的4章中我们将讨论无线网络、多媒体网络、网络安全和网络管理。这4个主题不便放进任何一个层中；实际上每个主题跨越了多个层次。因此理解这些主题（在某些网络教材中被列为高级主题）需要对协议栈所有层次有一个坚实基础，我们对链路层的学习已经完成了这样的基础！

课后习题和问题

复习题

5.1～5.2节

1. 如果因特网中的所有链路都提供可靠的交付服务，TCP可靠传输服务将是多余的吗？为什么？
2. 链路层协议能够向网络层提供哪些可能的服务？在这些链路层服务中，哪些在IP中有对应的服务？哪些在TCP中有对应的服务？

5.3节

3. 假设两个节点同时经一个速率为R的广播信道开始传输一个长度为L的分组。用d_{prop}表示这两个节点之间的传播时延。如果$d_{prop}<L/R$，会出现碰撞吗？为什么？
4. 在5.3节中，我们列出了广播信道的4种理想特性。这些特性中的哪些是时隙ALOHA所具有的？令牌传递具有这些特性中的哪些？
5. 使用人类在鸡尾酒会交互的类比来描述轮询和令牌传递协议。
6. 如果LAN有很大的周长时，为什么令牌环协议将是低效的？

5.4节

7. MAC地址空间有多大？IPv4的地址空间呢？IPv6的地址空间呢？
8. 假设节点A、B和C（通过它们的适配器）都连接到同一个广播LAN上。如果A向B发送上千个IP数据报，每个封装帧都有B的MAC地址，C的适配器会处理这些帧吗？如果会，C的适配器会将把这些帧中的IP数据报传递给C的网络层吗？如果A用MAC广播地址来发送这些帧，你的回答会有怎样的变化？
9. 为什么ARP查询要在广播帧中发送？为什么ARP响应要在具有一个特定目的MAC地址的帧中发送？
10. 对于图5-19中的网络，路由器有两个ARP模块，每个都有自己的ARP表。同样的MAC地址可能在两张表中都出现吗？

5.5节

11. 比较10BASE-T、100BASE-T和吉比特以太网的帧结构。它们有什么不同吗？
12. 假设某10 Mbps的适配器向某信道中发送一个使用曼彻斯特编码的全1的无限长流。适配器发出的信号每秒有多少次跳变？
13. 在CSMA/CD中，在第5次碰撞之后，一个节点选择的K=4的概率是多少？这种K=4的结果对应于10 Mbps以太网上的多少秒时延？

5.6节

14. 考虑图5-26。在4.4节的寻址意义下，有多少个子网呢？

习题

1. 假设分组的信息内容是比特模式1010101010101011，并且使用了偶校验方案。在二维奇偶校验情况下，包含该检验比特的字段的值是什么？你的回答应该使用最小长度检验和字段。
2. 说明（举一个不同于图5-6中那个的例子）两维奇偶校验能够纠正和检测单比特差错。说明（举一个例子）某些双比特差错能够被检测但不能纠正。
3. 假设某分组的信息部分（图5-4中的D）包含10 字节，它是由从整数0到9的8比特无符号二进制表示法的数组成。对该数据计算互联网检验和。
4. 考虑前一个习题，但是不再包括从整数0到9的二进制数。假设这10个字节包含
   a. 数字1到10的二进制表示；

b. 字母A到J（大写）的ASCII表示；

c. 字母a到j（小写）的ASCII表示。

比较这些数据的互联网检验和。

5. 考虑在图5-8中的4比特生成多项式G，假设D的值为10101010。R的值是什么？

6. 考虑上一个习题，这时假设D具有值

a. 10010001

b. 10100011

c. 01010101

7. 在5.3节中，我们提供了时隙ALOHA效率推导的概要。在本习题中，我们将完成这个推导。

a. 前面讲过，当有N个活跃节点时，时隙ALOHA的效率是$Np(1-p)^{N-1}$。求出使这个表达式最大化的p值。

b. 使用在（a）中求出的p值，令N接近于无穷，求出时隙ALOHA的效率。（提示：当N接近于无穷时，$(1-1/N)^N$接近于1/e。）

8. 说明纯ALOHA的最大效率是(1/2e)。注意：如果你完成了上面的习题，本习题很简单。

9. 假定3个活跃节点A、B和C使用时隙ALOHA来竞争访问某信道。假设每个节点有无限个分组要发送。每个节点在每个时隙中以概率p尝试传输。第一个时隙编号为时隙1，第二个时隙编号为时隙2，等等。

a. 节点A在时隙4中首先成功的概率是多少？

b. 某个节点（A、B或C）在时隙2中成功的概率是多少？

c. 在时隙4中出现首先成功的概率是多少？

d. 这个3节点系统的效率是多少？

10. 对N的下列值，画出作为p的函数的时隙ALOHA和纯ALOHA的效率。

a. $N=10$

b. $N=25$

c. $N=50$

11. 考虑有N个节点和传输速率为R bps的一个广播信道。假设该广播信道为多路访问而使用轮询（有一个附加的轮询节点）。假设从某节点完成传输到后续节点允许传输的时间量（即轮询时延）是d_{poll}。假设在一个轮询周期中，一个给定的节点允许传输至多Q比特。该广播信道的最大吞吐量是多少？

12. 如图5-38所示，考虑通过两台路由器互联的3个LAN。

a. 将适配器包含在内，重画这个图。

b. 对所有的接口分配IP地址。对子网1使用形式为111.111.111.xxx的地址，对子网2使用形式为122.222.222.xxx的地址，对子网3使用形式为133.133.133.xxx的地址。

c. 为所有的适配器分配MAC地址。

d. 考虑从主机A向主机F发送一个IP数据报。假设所有的ARP表都是最新的。就像在5.4.2节中对单路由器例子中所做的那样，列举出所有步骤。

e. 重复（d），现在假设在发送主机中的ARP表

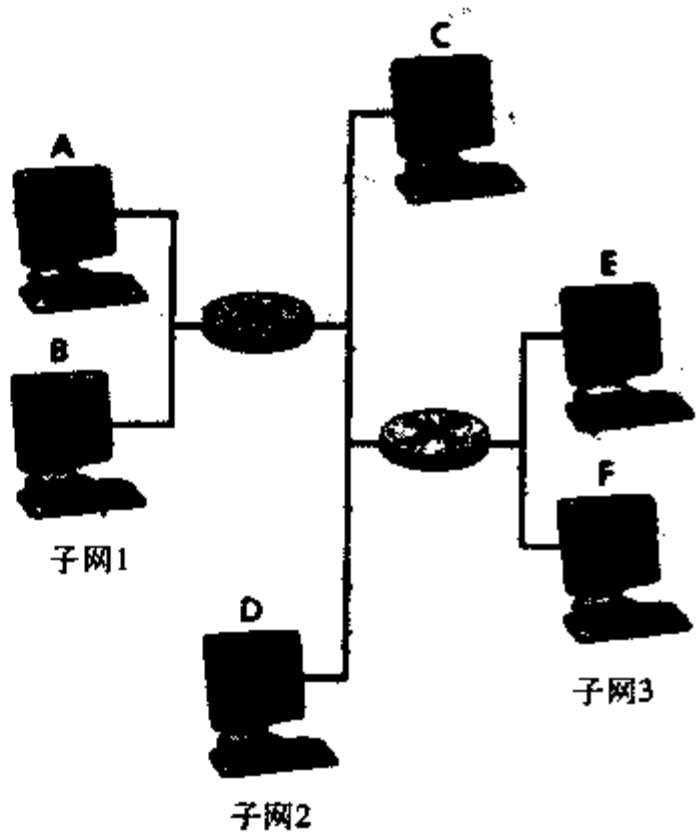


图5-38 由两台路由器互联的3个子网

为空（并且其他表都是最新的）。

13. 考虑前面的习题，但是现在假设子网2和子网3之间的路由器由一台交换机代替。在这种新的场景中回答12题中的问题（a）～（e）。

14. 前面讲过，使用CSMA/CD协议，适配器在碰撞之后等待$K \cdot 512$比特时间，这里K是随机选取的。对于$K=100$，对于一个10 Mbps的以太网来说，适配器返回到第二步要等多长时间？对于100 Mbps的以太网来说呢？

15. 假设节点A和节点B在同一个10 Mbps以太网总线上，这两个节点的传播时延为225比特时间。假设节点A开始传输一帧，并且在它传输结束之前，节点B开始传输一帧。在A检测到B已经传输之前，A能完成传输吗？为什么？如果回答是可以，则A错误地认为它的帧已成功传输而无碰撞。提示：假设在时刻$t=0$比特时间，A开始传输一帧。在最坏的情况下，A传输一个512＋64比特时间的最小长度的帧。因此A将在$t=512+64$比特时间完成帧的传输。如果B的信号在比特时间$t=512+64$比特之前到达A，则答案是否定的。在最坏的情况下，B的信号什么时候到达A？

16. 假设节点A和节点B在同一个10 Mbps以太网总线上，并且这两个节点的传播时延为225比特时间。假设A和B同时发送帧，帧发生了碰撞，然后A和B在CSMA/CD算法中选择不同的K值。假设没有其他节点处于活跃状态，来自A和B的重传会碰撞吗？为了此目的，计算下面的例子就足以说明问题了。假设A和B在$t=0$比特时间开始传输。它们在$t=225$比特时间都检测到了碰撞。它们在$t=225+48=273$比特时间完成了阻塞信号的传输。假设$K_A=0$，$K_B=1$。B会将它的重传调整到什么时间？A在什么时间开始发送？（注意：这些节点在返回第2步之后，必须等待一个空闲信道，参见协议。）A的信号在什么时间到达B呢？B在它预定的时间控制传输吗？

17. 考虑某个让所有节点直接与一个集线器相连的100 Mbps的100BASE-T以太网。为了获得0.50的效率，节点和集线器之间的最大距离是多少？假设帧长为64 字节并且中间没有转发器。这个最大距离也确保正在传输的节点A能够检测出在它传输时是否有其他任何节点在传输吗？为什么？你得到的最大距离和实际的100 Mbps标准比较将有什么结论？

18. 在这个习题中将对一个类似于CSMA/CD多路访问协议的效率进行推导。在这个协议中，时间分为时隙，所有适配器都与时隙同步。然而，和时隙ALOHA不同的是，一个时隙的长度（以秒计）比一帧的时间（即传输一帧的时间）小得多。令S表示一个时隙的长度。假设所有帧都有恒定长度$L=kRS$，其中R是信道的传输速率，k是一个大整数。假定有N个节点，每个节点都有无穷多帧要发送。我们还假设$d_{prop}<S$，以便所有节点在一个时隙时间结束之前能够检测到碰撞。这个协议如下所述：

- 对于某给定的时隙，如果没有节点占有这个信道，所有节点竞争该信道；特别是每个节点以概率p在该时隙传输。如果刚好有一个节点在该时隙中传输，该节点对后续的$k-1$个时隙占有信道，并传输它的整个帧。
- 如果某节点占用了信道，所有其他节点抑制传输，直到占有该信道的这个节点完成了该帧的传输为止。一旦该节点传输完它的帧，所有节点竞争该信道。

注意到信道在两种状态之间交替："生产性状态"（它恰好持续k个时隙）和"非生产性状态"（它持续随机数个时隙）。显然，该信道的效率是$k/(k+x)$，其中x是连续的非生产性时隙的期望值。

a. 对于固定的N和p，确定这个协议的效率。

b. 对于固定的N，确定使该效率最大化的p值。

c. 使用在b小题求出的p（它是N的函数），确定当N趋向无穷时的效率。

d. 说明随着帧长度变大时，该效率趋近于1。

19. 假设两个节点A和B被连接到一个900 m长的电缆的两端，它们都有一个1000比特（包括所有首部和前同步码）的帧要发给对方。两个节点都试图在$t=0$时刻传输。假设在A和B之间有4个转发器，每个

都插入20比特的时延。假设传输速率是10 Mbps，并且使用回退间隔是512比特倍数的CSMA/CD。在第一次碰撞后，在指数后退协议中A取$K=0$，B取$K=1$。忽略阻塞信号和96比特的时延。

a. A和B之间的单向传播时延（包括转发器时延）是多少（以秒计）？假设信号传播速度是2×10^8 m/s。

b. 什么时候（以秒计）A的分组完全交付给B？

c. 现在假设只有A有一个分组要发送，并用交换机代替转发器。假设除存储转发时延外，每台交换机还有20比特的处理时延。A的分组什么时候（以秒计）交付给B？

20. 现在考虑习题12中的图5-38。对主机A、两台路由器和主机F的各个接口提供MAC地址和IP地址。当在下列场合传输该帧时，给出在封装该IP数据报的帧中的源和目的MAC地址：(i)从A到左边的路由器；(ii)从左边的路由器到右边的路由器；(iii)从右边的路由器到F。还要给出到达每个点时封装在该帧中的IP数据报中的源和目的IP地址。

21. 现在假定在图5-38最左边的路由器被一台交换机替换。主机A、B、C和D及右边的路由器以星型方式与这台交换机相连。当在下列场合传输该帧时，给出在封装该IP数据报的帧中的源和目的MAC地址：(i)从A到该交换机；(ii)从该交换机到右边的路由器；(iii)从右边的路由器到F。还要给出到达每个点时封装在该帧中的IP数据报中源和目的IP地址。

22. 考虑图5-26。假设所有链路是100 Mbps。在该网络中的14个端系统能够取得的最大总体聚合吞吐量是多少？为什么？

23. 假定在图5-26中的3台连接各系的交换机用集线器来代替。所有链路是100 Mbps。在该网络中的14个端系统能够取得的最大总体聚合吞吐量是多少？为什么？

24. 假定在图5-26中的所有交换机用集线器来代替。所有链路是100 Mbps。在该网络中14个端系统中的能够取得的最大总体聚合吞吐量是多少？为什么？

25. 我们考虑在图5-24环境中的某学习中的交换机的运行情况。假定(i)A向D发送一个帧；(ii)D向A回答一个帧；(iii)C向D发送一个帧；(iv)D向C回答一个帧。该交换机表初始为空。显示在这些时间的前后该交换机表的状态。对于上述每个事件，确定对传输的帧进行转发的链路，并简要地论证你的答案。

26. 前面讲过ATM使用由5 字节首部和48 字节有效载荷组成的53 字节的分组。53 字节对于定长分组来说非常小；大多数网络协议（IP、以太网、帧中继等）使用平均来说大得多的分组。小分组长度的一个缺点是链路带宽的很大一部分被额外开销字节所消耗，在ATM情况下，几乎10%的带宽被ATM首部“浪费”掉。在本习题中，我们研究为什么要选择这么小的分组长度。最后，假设ATM信元由P字节（可能不是48）和5 字节首部组成。

a. 考虑直接经ATM发送一个数字编码的话音源。假设该源以64 kbps的恒定速率编码。假设在源向网络发送信元之前，每个信元都被完全填充。填充一个信元所需的时间称为**分组化时延**（packetization delay）。用L来表示，以毫秒为单位决定分组化时延。

b. 大于20 ms的分组化时延会导致一个明显的、令人不快的回音。对L=1500字节（大致对应于一个最大长度的以太网分组）和L=48字节（对于一个ATM信元），确定该分组化时延。

c. 对速率R=155 Mbps（ATM流行的链路速率）的链路，计算L=1500字节和L=48字节时，单台ATM交换机的存储转发时延。

d. 对使用小信元长度的优点进行评述。

27. 考虑显示在图5-37中的MPLS网络，假定路由器R5和R6现在是MPLS使能的。假定我们要执行流量工程，使得从R6发往A的分组要经R6-R4-R3-R1交换到A，从R5到A的分组要过R5-R4-R2-R1交换到A。给出R5和R6中的MPLS表以及在R4中修改的表，使得这些成为可能的。

28. 再次考虑如上面习题中相同的场景，而假定从R6发往D的分组经R6-R4-R3交换，从R5发往D的分组经R4-R2-R1-R3交换。给出在所有路由器中的MPLS表，使得这些成为可能的。

讨论题

鼓励读者上网冲浪以回答下列问题。

1. 一个10/100 Mbps适配器当前价格范围大致是多少？一个吉比特以太网适配器呢？这些价格与一个56 kbps拨号调制解调器或者一个ADSL调制解调器相比怎么样？
2. 交换机通常根据接口（在LAN行话中也称为端口）数量来定价。对于一个仅由100 Mbps接口组成的交换机来说，当前每个接口的价格范围大致是多少？
3. 适配器的很多功能都可以在运行于节点CPU上的软件中执行。将这些功能从适配器移到节点有什么优点和缺点？
4. 使用Web搜索在以太网帧中用于IP数据报和ARP分组的协议号。
5. 阅读有关使用MPLS的流量工程的参考文献[Xiao 2000, Huang 2002和RFC 3346]。列出流量工程的一系列目标。这些目标中的哪些仅能用MPLS来满足，哪些能通过使用现有（非MPLS）的协议来满足？在后一种情况下，MPLS提供了哪些优点？

Ethereal实验

在与本教科书配套的Web站点（http://www.awl.com/kurose-rose）上，读者可以找到一个Ethereal实验，该实验研究了IEEE 802.3协议的操作和以太网帧格式。

人物专访

Simon S. Lam是位于奥斯丁的得克萨斯大学计算机科学系的教授和董事会主席。从1971年到1974年，他在UCLA的ARPA网络测量中心工作，从事卫星和无线电分组交换方面的工作。他领导的研究组于1993年发明了安全套接字及其原型，这是第一个安全套接字层（命名为安全网络编程），为此赢得了2004年ACM软件系统奖。他的研究兴趣在于网络协议和安全服务的设计和分析。他从华盛顿州立大学获得了电机工程学士学位，从UCLA获得了硕士和博士学位。

Simon S. Lam

• 为什么您决定专注于网络？

当我于1969年秋季作为一名研究生新生到达UCLA时，我的目的是研究控制论。后来我参加了Leonard Kleinrock的排队论课程，他给我留下了深刻印象。不久后，我以排队系统的自适应控制作为可能的论文题目进行了研究。在1972年初，Larry Roberts启动了ARPAnet卫星系统项目（后来称为分组卫星）。Kleinrock教授请我参加该项目。我们做的第一件事是为时隙ALOHA协议引入一个简单而实用的后退算法。不久后，我发现了许多有趣的研究问题，如ALOHA的不稳定性问题和对自适应回退的需求，这些形成了我学位论文的核心。

• 在20世纪70年代您开始了在UCLA的学生生涯，活跃于因特网的早期阶段，那时的情况怎样？那时人们对因特网将变成什么样有哪些大致想法？

在产业界和学术界，我见证了十分不同于其他系统构建项目的环境。ARPAnet的初始规定的目标是相当谨慎的，即提供从远程位置接入昂贵的计算机的手段，使许多科学家能够使用这些计算机。然而，随着分组卫星于1972年和分组无线电项目于1973年启动，ARPA的目标有了相当大的扩展。到了1973年，ARPA同时建造了3个不同的分组网络，对Vint Cerf和Bob Kahn来说研发互联策略变得有必要了。

回顾过去，所有这些网络的进展被视为（我相信）符合逻辑的而不是魔术性的。没有人能够想象到

今天的因特网规模和个人计算机的能力。在第一台PC出现之前有10年时间，为了正确地处理工作，多数学生以一系列穿孔卡片的形式提交了计算机程序用于批处理；仅有某些学生能够直接接触计算机，而计算机通常位于一个受限的区域中；调制解调器速度缓慢且稀有；作为一名研究生，我在书桌上仅有一部电话机，我使用铅笔和纸从事我的大部分工作。

• 您认为网络和因特网领域在未来将往何处发展？

在过去，因特网IP协议的简单性是它在竞争上取胜的最大法宝，并且成为网络互联事实上的标准。与它的竞争对手如20世纪80年代的X.25和20世纪90年代的ATM不同的是，IP能够在任何链路层网络技术之上运行，因为它仅提供尽力而为的数据报服务。因此任何分组网络能够连到因特网上。

不幸的是，IP的最大长处现在成为了一种缺点。IP目前像一件紧身衣，限制了因特网向特定方向发展。IP层经济上过于重要而不能通过修补来支持新功能，如多播和QoS。近年来，许多研究人员将他们的努力重新定向到支持多播和QoS的应用层和运输层。大多数其他当前因特网研究主题如安全性和P2P系统，仅涉及应用层。在无线自组织网络、传感网络和卫星网络方面也有大量研究工作。这些网络能够看作独立的系统或链路层系统，因为位于IP的紧身衣之外，它们能够繁荣发展。

许多人对于P2P系统可能作为新奇的因特网应用的平台而感到欢欣鼓舞。然而，P2P系统在使用因特网资源方面效率极低。我关注的一个问题是，随着因特网不断互联各种设备和支持未来的P2P使能的应用，是否因特网核心的传输和交换能力将继续增加得比对因特网流量的需求更快。没有大量的容量提前预备，在面临敌意攻击和拥塞情况下，确保网络稳定性将是一项重要的任务。

• 您的工作最具挑战性的部分是什么？

我的工作最具挑战性的部分是，作为一名教授如何传授和激发我课堂上的每个学生以及我所指导的每个博士研究生，而不只是取得重要成就。非常聪明和有学习动力的人可能要求少量的引导而非许多其他东西。我经常从这些学生那里学到很多东西，比他们从我这里学到的多。教育和激励学习落后的学生是一项重要的挑战。

• 您能预见到技术在未来学习方面有哪些影响？

最终几乎所有的人类知识将可以通过因特网得到，因特网将成为最为强有力的学习工具。这种巨大的知识库将具有为全世界的学生打开施展才华的疆场的潜力。例如，在任何国家中受激发的学生们将能够访问最好课程的Web站点、多媒体演讲和教学材料。据说IEEE和ACM数字图书馆已经加速了中国的计算机科学研究的发展。与此同时，因特网将超越所有学习的地理障碍。

第6章　无线网络和移动网络

在电话技术领域，过去15年可认为是蜂窝电话技术的黄金发展期，全球范围的移动蜂窝电话用户数量从1993年的3400万增长到2005年的20多亿，同时蜂窝电话用户数量现在也超过了固定电话用户数量[ITU Statistics 2007]。蜂窝电话的许多优点是显而易见的，通过一个移动性强、重量轻的设备，能在任何地方、任何时间、无缝地接入全球电话网络。随着便携机、掌上计算机和PDA的出现，以及它们随时、随地、无缝地接入全球因特网的承诺的实现，无线因特网设备的使用在不久的将来是否会出现类似的蓬勃发展呢？

不管无线因特网设备未来如何增长，无线网络以及它们提供的移动相关服务显然已经展现在人们面前了。从网络的观点来说，由这些网络引发的挑战，特别是在数据链路层和网络层，与传统的有线网络的差别非常大，需要用单独一章（即本章）的篇幅来专门讨论无线和移动网络。

在本章中，我们首先讨论移动用户、无线链路和网络，以及它们与所连接的更大网络（通常是有线网络）之间的关系。我们将指出以下两方面的差别：一个是在该网络中由通信链路的*无线*特性所带来的挑战，另一个是由这些无线链路使能的*移动性*。在无线和移动性之间进行区分非常重要，它使我们能更好地隔离、标识和掌握在每个领域的重要概念。注意到实际上在许多网络环境中，其中的网络节点是无线的而不是移动的（例如具有固定工作站和大显示器的无线家庭或办公网络），有限的移动性也并不要求用无线链路（例如，一个在家里使用便携计算机的员工，关闭该便携机电源，开车去工作，然后将该便携机连接到公司的有线网络上）。当然，许多最让人激动的网络环境是那些让用户同时具有无线和移动性的网络环境。例如在下列情况下，一个移动用户（不妨假设其正坐在汽车后座上）维持一个IP话音呼叫和多个进行中的TCP连接，同时又以每小时160公里的速度飞驰在高速公路上。正是在这种无线和移动的交集中，我们将会发现最有趣的技术挑战。

首先我们将举例说明用于考虑无线通信和移动性的环境，即无线（并可能是移动的）用户通过位于网络边缘的无线链路连接到更大网络基础设施中的网络。然后在6.2节，我们考虑这种无线链路的特性，还将包括对码分多址（Code Division Multiple Access，CDMA）技术的简要介绍。CDMA是一个在无线网络中经常使用的共享媒体接入协议。在6.3节，我们将较深入地分析IEEE 802.11（WiFi）无线LAN标准的链路层方面；同时我们还将对蓝牙和WiMAX做简要描述。在6.4节，我们概述蜂窝因特网接入，其中包括正在兴起的同时提供话音和高速因特网接入的3G技术。在6.5节，我们将注意力转向移动性，关注于移动用户的定位问题、对移动用户选路以及“切换”（handing off）移动用户（即在网络中从一个接入点动态地移动到另一点）等问题。我们将在6.6节和6.7节分别分析在移动IP标准和GSM中是如何实现这些移动服务的。最后，我们将在6.8节考虑无线链路和移动性对运输层协议和网络应用程序的影响。

6.1 概述

图6-1显示了我们将要讨论无线数据通信和移动性主题的环境。为使讨论具有一般性，我们一开始的讨论会覆盖较多的网络，这些网络包括像IEEE 802.11这样的无线局域网和像3G网络这样的蜂窝网络。然后在后续各节中，我们将对具体的无线体系结构进行更加详细的讨论。在无线网络中我们会看到下列元素：

历史事件

公共WIFI接入：费城的路灯方案很快将实现吗？

WiFi热区（即用户能够找到802.11无线接入的公共位置的地方）在全世界的旅馆、机场和咖啡厅变得日益常见。到2006年底，T-Mobile在美国提供了8000多个位置的热区，包括星巴克咖啡屋和Borders图书音乐店。麦当劳在全世界7000多个快餐店提供了因特网接入服务。

但在美国宾夕法尼亚州费城正在采取一个不同寻常的方法，建立无所不在的无线接入，规划未来将该城市135平方英里的区域对每个邻区引入无线因特网接入。其目标是“将费城变为国家最大的WiFi，有助于改善教育，填平数字鸿沟，增强邻居交往，以及减少政府成本”。该计划要求在大约4000个街道路灯和交通控制设备上安装802.11b无线接入点。

这项雄心勃勃的规划是该城市、无线费城（一个非营利性组织）和因特网服务提供商Earthlink达成的一项协议。它的建设将无需由居民提供资金，在选定的公园和公共场所将能够免费接入。居民和商业用户将付费接入。无线费城将利用所得收入（接入收入的5%）来投资费城的教育和社会项目，包括面向技术的项目。

- 无线主机。如同在有线网络中一样，主机是运行应用程序的端系统设备。**无线主机**（wireless host）可以是便携机、掌上计算机、PDA、电话或者台式计算机。主机本身可能移动，也可能不移动。
- 无线链路。主机通过**无线通信链路**（wireless communication link）连接到一个基站（定义见下文）或者另一个无线主机。不同的无线链路技术有不同的传输速率和传输距离。图6-2显示了几种比较流行的无线链路标准的两种主要特性（覆盖区域和链路速率）。（该图仅表示了提供这些特性的大致概念。例如，这些类型中的某些网络现在正在部署中，某些链路速率取决于距离、信道条件和在无线网络中的用户数量，这些数据可能比显示的值更高或更低。）我们将在本章的前半部分讨论这些标准。在6.2节，我们也考虑其他无线链路特性（如它们的错比特率及其原因）。

在图6-1中，无线链路将位于网络边缘的主机连接到更大的网络基础设施中。我们要补充的是，无线链路有时同样应用在网络内部以连接路由器、交换机和其他网络设备。然而，在本章中我们关注的焦点是无线通信在网络边缘的应用，因为许多最为振奋人心的技术挑战和多数增长就发生在这里。

- 基站。**基站**（base station）是无线网络基础设施的一个关键部分。与无线主机和无线链路不同，基站在有线网络中没有明确的对应设备。它负责向与之关联的无线主机发送数据和从主机那里接收数据（例如分组）。基站通常负责协调与之相关联的多个无线主机

的传输。当我们说一台无线主机与某基站“相关联”时，则是指：①主机位于该基站的无线通信覆盖范围内；②主机使用该基站中继它（该主机）和更大网络之间的数据。**蜂窝网络中的蜂窝塔**（cell tower）和802.11无线LAN中的**接入点**（access point）都是基站的例子。

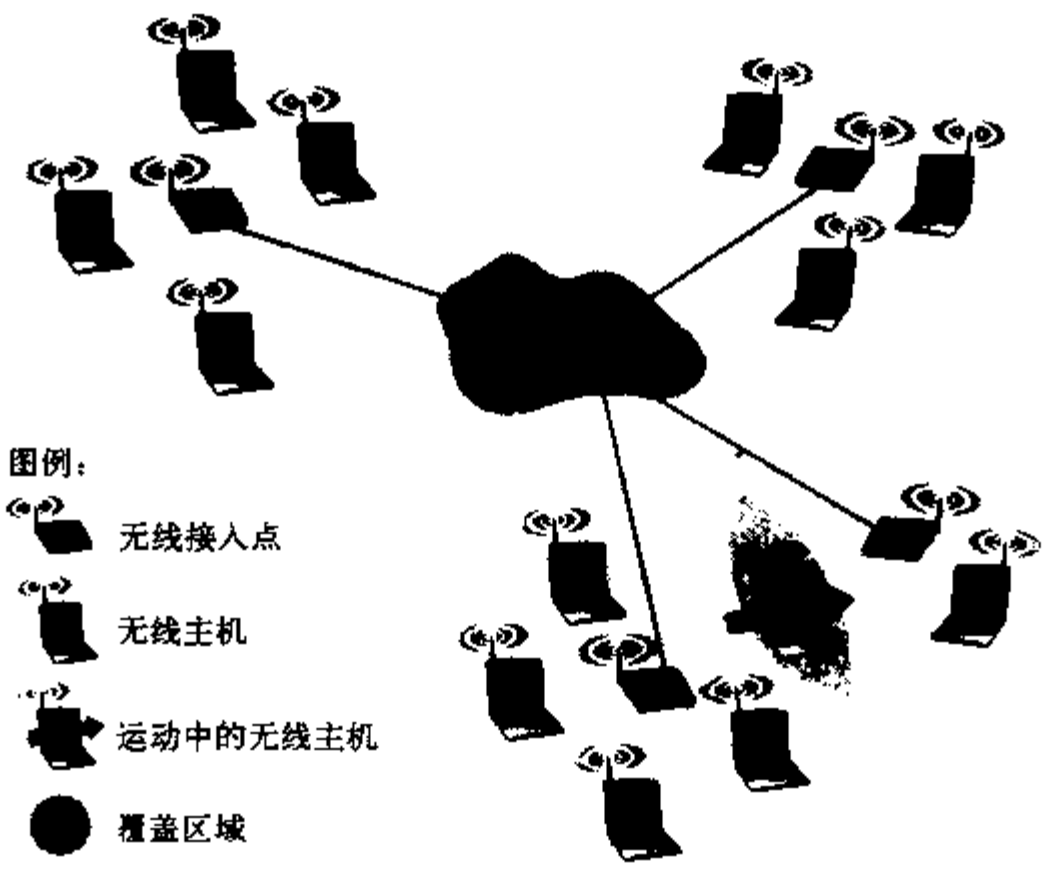


图6-1 无线网络的元素

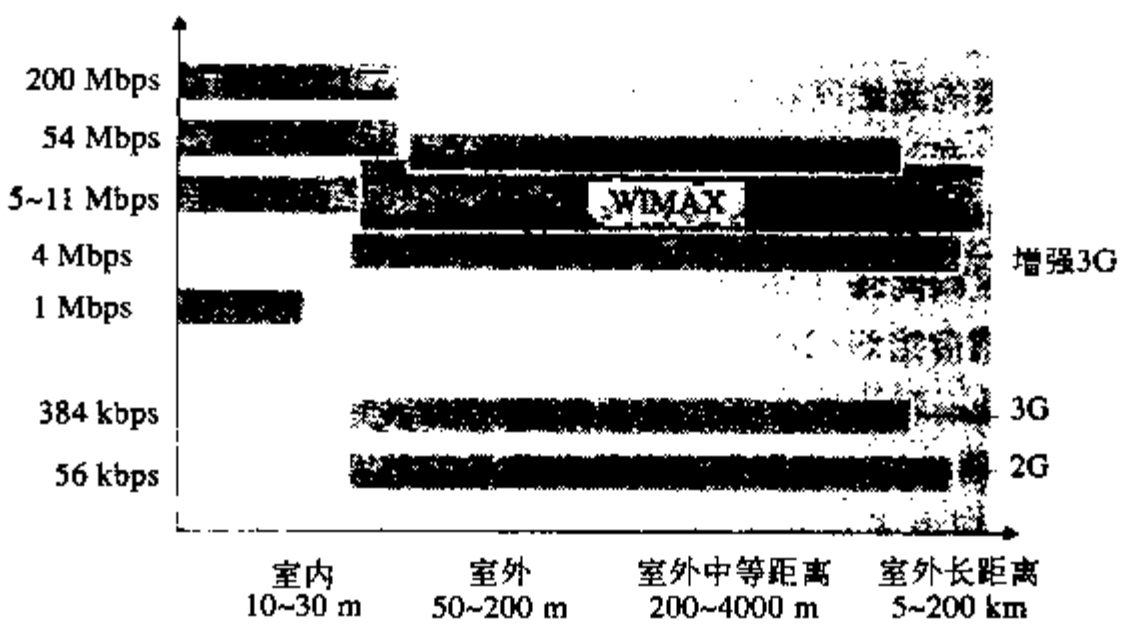


图6-2 几种无线网络标准的链路特性

在图6-1中，基站与更大网络相连（如因特网、公司或家庭网、电话网），因此这种连接在无线主机和与之通信的其他部分之间起着链路层中继的作用。

与基站关联的主机常被称为以**基础设施模式**（infrastructure mode）运行，因为所有传统的网络服务（如地址分配和选路）都由网络向通过基站相连的主机提供。在**自组织网络**（ad hoc network）中，无线主机没有这样的基础设施与之相连。在没有这样的基础设施的情况下，主机本身必须提供诸如选路、地址分配、类似于DNS的名字转换等服务。在本书中，我们主要关注于基础设施模式网络。

当一台移动主机移动范围超出一个基站的覆盖范围而到达另一个基站的覆盖范围后，它将改变其接入更大网络的连接点（例如，改变与之相关联的基站），这一过程称作**切换**（handoff）。这种移动性引发了许多具有挑战性的问题。如果一台主机可以移动，那么如何找到它在网络中的当前位置，从而使得数据可以向该移动主机转发？如果一台主机可以位于许多可能位置中的一个，那么如何进行编址？如果主机在一个TCP连接或者电话呼叫期间移动，数据如何选路而使连接保持不中断？这些以及许多其他问题使得无线网络和移动网络成为一个让人振奋的网络研究领域。

- 网络基础设施。这是无线主机希望与之进行通信的更大网络。

在讨论完无线网络的构件以后，我们注意到这些构件能够以许多不同种方式组合以形成不同类型的无线网络。当阅读本章，或阅读/学习本书之外的更多有关无线网络的内容时，你可能发现这些无线网络类型的分类方法是有用的。在最高层次，我们能够根据两个准则来对无线网络分类：①在该无线网络中的分组是否跨越了一个无线跳或多个无线跳；②网络中是否有诸如基站这样的基础设施：

- 单跳，基于基础设施。这些网络具有与较大的有线网络（如因特网）连接的基站。此外，该基站与无线主机之间的所有通信都经过一个无线跳。你在教室、咖啡厅或图书馆中所使用的802.11网络，以及你将很快学到的802.16 WiMAX网络都属于这种类型。
- 单跳，无基础设施。在这些网络中，存在与无线网络相连的基站。然而，如我们将要见到的那样，在这种单跳网络中，节点之一可以协调其他节点的传输。蓝牙网络（我们将在6.3.6节学习）和具有自组织模式的802.11网络是单跳、无基础设施的网络。
- 多跳，基于基础设施。在这些网络中，一个基站表现为以有线方式与较大网络相连。然而，某种无线节点为了经该基站通信，可能不得不通过其他无线节点中继它们的通信。某些传感网络和所谓的**无线网状网络**（wireless mesh network）属于这种类别。
- 多跳，无基础设施。在这些网络中没有基站，并且节点为了到达目的地可能必须在几个其他无线节点之间中继报文。节点也可能是移动的，在多个节点中改变连接关系，一类网络被称为**移动自组织网络**（Mobile Ad hoc Network, MANET）。如果该移动节点是车载的，该网络是**车载自组织网络**（Vehicular Ad hoc Network, VANET）。如你所想，为这种网络开发协议是一种挑战，它们是许多进行中的研究主题。

在本章中，我们将主要学习内容限制在单跳、基于基础设施的网络中。

现在我们更深一步地研究无线网络和移动网络面对的挑战。我们将首先讨论单独的无线链路，而在本章稍后部分讨论移动性。

6.2 无线链路和网络特征

一开始，我们考虑用一台有线以太网交换机互联主机的一个简单有线网络，如一个家庭网络（参见5.6节）。如果我们用无线802.11网络代替该有线以太网，用无线NIC卡代替主机上的有线以太网卡，用接入点代替以太网交换机，实际上在网络层及其以上层次中不需要有任何变化。这提示我们，当寻找有线和无线网络的重要区别时，应该关注链路层。事实上，我们能够发现有线链路和无线链路间的许多重要区别：

- 递减的信号强度。电磁波在穿过物体（如无线电信号穿过墙壁）时强度将减弱。即使在自由空间中，信号仍将扩散，这使得信号强度随着发送方和接收方距离的增加而减弱

（有时称其为**路径损耗**（path loss））。

- *来自其他源的干扰*。在同一个频段发送信号的电波源将相互干扰。例如，2.4 GHz无线电话和802.11b无线LAN在相同的频段中传输。因此，802.11b无线LAN用户若同时利用2.4 GHz无线电话通信，将会导致网络和电话都不会工作得特别好。除了来自发送源的干扰，环境中的电磁噪声（如附近的电动机、微波）也能形成干扰。
- *多路径传播*。若电磁波的一部分受物体和地面反射，在发送方和接收方之间走了不同长度的路径，则会出现**多径传播**（multipath propagation）。这使得接收方收到的信号变得模糊。位于发送方和接收方之间的移动物体可导致多路径传播随时间而改变。

对于无线信道特征、模型和测量的详细讨论请参见[Anderson 1995]。

上述讨论表明，无线链路中的比特错误将比有线链路中更为常见。因此，无线链路协议（如我们将在下面一节中讨论的802.11协议）不仅采用有效的CRC错误检测码，还采用了链路层ARQ协议来重传错误帧。

考虑了无线信道上可能出现的损伤后，我们将注意力转向接收无线信号的主机。该主机接收到一个电磁信号，而该信号是发送方传输的初始信号的退化形式和环境中的背景噪声的结合，其中的信号退化是由于衰减和我们前面讨论过的多径传播和其他一些因素所引起的。**信噪比**（Signal-to-Noise Ratio, SNR）是所收到的信号（如被传输的信息）和噪声强度的相对测量。SNR的度量单位通常是分贝（dB），有人认为这个主要由电气工程师所使用的度量单位会使计算机科学家迷惑不解。以dB度量的SNR是下列比值的20倍：该比值是接收到的信号的振幅与噪声幅度之比的以10为底的对数。就我们的讨论目的而言，我们仅需要知道较大的SNR使接收方更容易从背景噪声中提取传输的信号。

图6-3（该图选自[Holland 2001]）显示了三种不同调制技术的比特差错率（BER）（大致说来BER是在接收方收到的有错的传输比特的概率）与SNR之比，为在理想信道上传输信息，需要用这些调制技术对信息进行编码。调制和编码理论以及信号提取和BER都超出了本书的范围（对这些主题的讨论参见[Schwartz 1980]）。尽管如此，图6-3显示了几种物理层的特征，这些特征对于理解较高层无线通信协议是重要的。

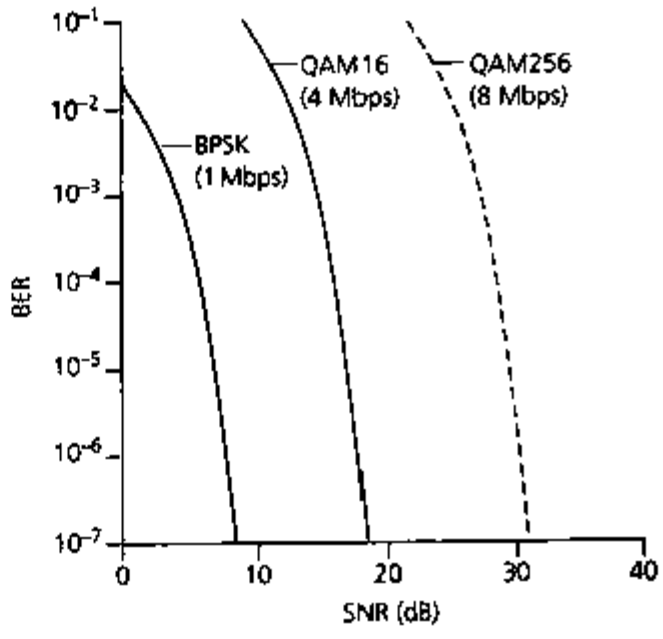


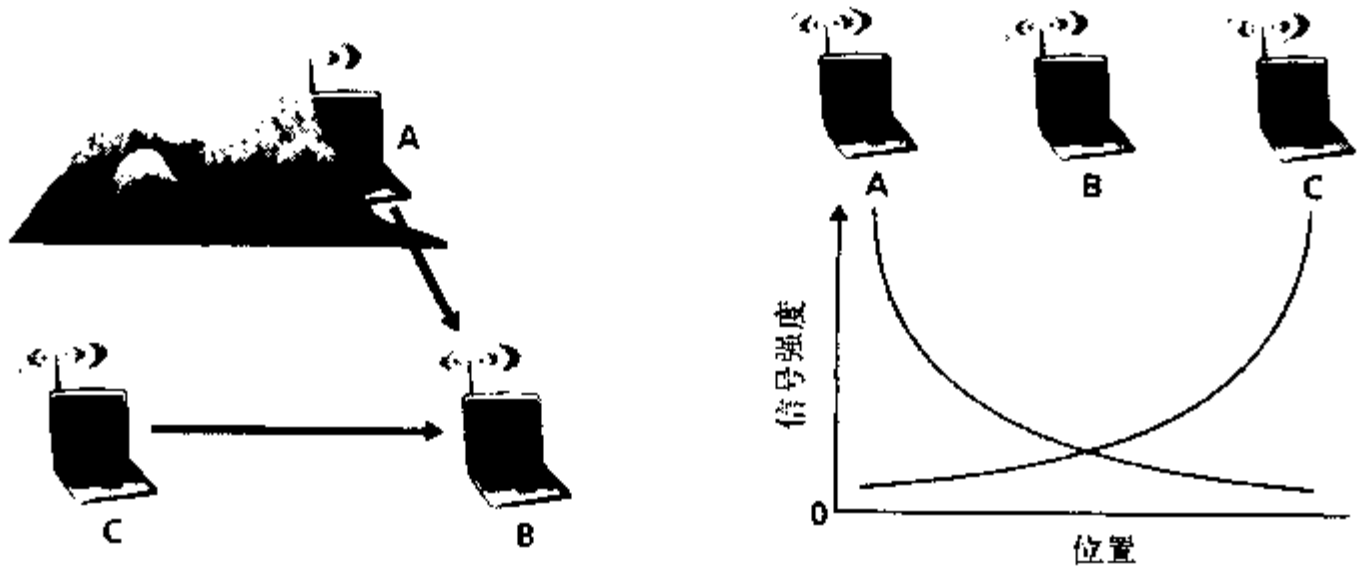
图6-3 比特差错率、传输率和SNR

- *对于给定的调制方案，SNR越高，BER越低*。由于发送方通过增加传输功率就能够增加SNR，因此发送方能够通过增加传输功率来降低接收到差错帧的概率。然而，注意到当该功率超过某个阈值时，如BER从10^{-12}降低到10^{-13}，可证明几乎不会有实际增益。增加传输功率也会有一些缺点：发送方必须消耗更多的能量（对于用电池供电的移动用户这一点非常重要），并且发送方的传输更可能干扰另一个发送方的传输（参见图6-4b）。
- *对于给定的SNR，具有较高传输率的调制技术（无论差错与否）将具有较高的BER*。例如在图6-3中，对于10dB的SNR，具有1 Mbps传输速率的BPSK调制具有小于10^{-7}的BER，而具有4 Mbps传输速率的QAM 16调制，BER是10^{-1}，该值太高而没有实际用处。然而，具有20 dB的SNR，QAM 16调制具有4 Mbps的传输速率和10^{-7}的BER，而BPSK调制具

有仅1 Mbps的传输速率和一个低得“无法在图上表示”的BER。如果人们能够容忍10^{-7}的BER，在这种情况下由QAM 16提供的较高的传输速率将使它成为首选的调制技术。这些考虑引出了我们下面描述的最后一个特征。

- *物理层调制技术的动态选择能用于适配对信道条件的调制技术。*SNR（因此BER）可能因移动性或环境而变化。在蜂窝数据系统和802.16 WiMAX和802.11 WiFi网络中（我们将在6.3节学习）使用了自适应调制和编码。例如，这使得在给定信道特征下，选择一种提供最高可能传输速率的调制技术将受制于BER。

有线和无线链路之间的差异并非仅仅只有较高的、时变的误比特率这一项，前面讲过在有线广播链路中，所有节点能够接收到所有其他节点的传输。而在无线链路中，情况并非如此简单。如图6-4所示，假设站点A正在向站点B发送，站点C也在向站点B传输。由于所谓的**隐藏终端问题**（hidden terminal problem），即使A和C的传输确实在目的地B发生干扰，环境的物理阻挡（例如，一座大山或者一座建筑）也可能会妨碍A和C互相听到对方的传输。这种情况如图6-4a所示。第二种导致在接收方无法检测的碰撞情况是，当通过无线媒体传播时信号强度的**衰减**（fading）。图6-4b描述了这种情况，A和C所处的位置使得它们的信号强度不足以使它们相互检测到对方的传输，然而它们的传输足以强到在站点B相互干扰。正如我们将在6.3节看到的那样，隐藏终端问题和衰减使得多路访问在无线网络中的复杂性远远高于有线网络中的情况。



a) 隐藏终端问题　　b) 衰减

图6-4 隐藏终端问题和衰减

CDMA

在第5章讲过，当不同主机使用一个共享媒体通信时，需要有一个协议来保证多个发送方发送的信号不在接收方互相干扰。在第5章中，我们描述了3类媒体访问协议：信道划分、随机访问和轮流。码分多址（Code Division Multiple Access, CDMA）属于信道划分协议的家族，它在无线LAN和蜂窝技术中应用很广泛。由于CDMA对无线领域十分重要，在后面小节中对具体的无线接入技术进行探讨以前，我们首先对其快速地浏览一下。

在CDMA协议中，要发送的每一个比特都通过乘以一个信号（编码）的比特来进行编码，这个信号的变化速率（通常称为**码片速率**（chipping rate））比初始数据比特序列速率快得多。图6-5表示一个简单的、理想化的CDMA编码/解码情形。假设初始数据比特到达CDMA编码器的速率定义了时间单元，也就是说，每个要发送的初始数据比特需要1比特时隙时间。设d_i为

第i个比特时隙中的数据比特值。为了数学上的便利，我们把数据比特0表示为−1。每个比特时隙又进一步细分为M个微时隙；在图6-5中，$M=8$，不过在实际中M的值要大得多。发送方使用的CDMA编码由M个值的序列c_m组成，$m=1, \cdots, M$，每个值取+1或者−1。在图6-5的例子中，被发送方使用的M比特的CDMA码是（1，1，1，−1，1，−1，−1，−1）。

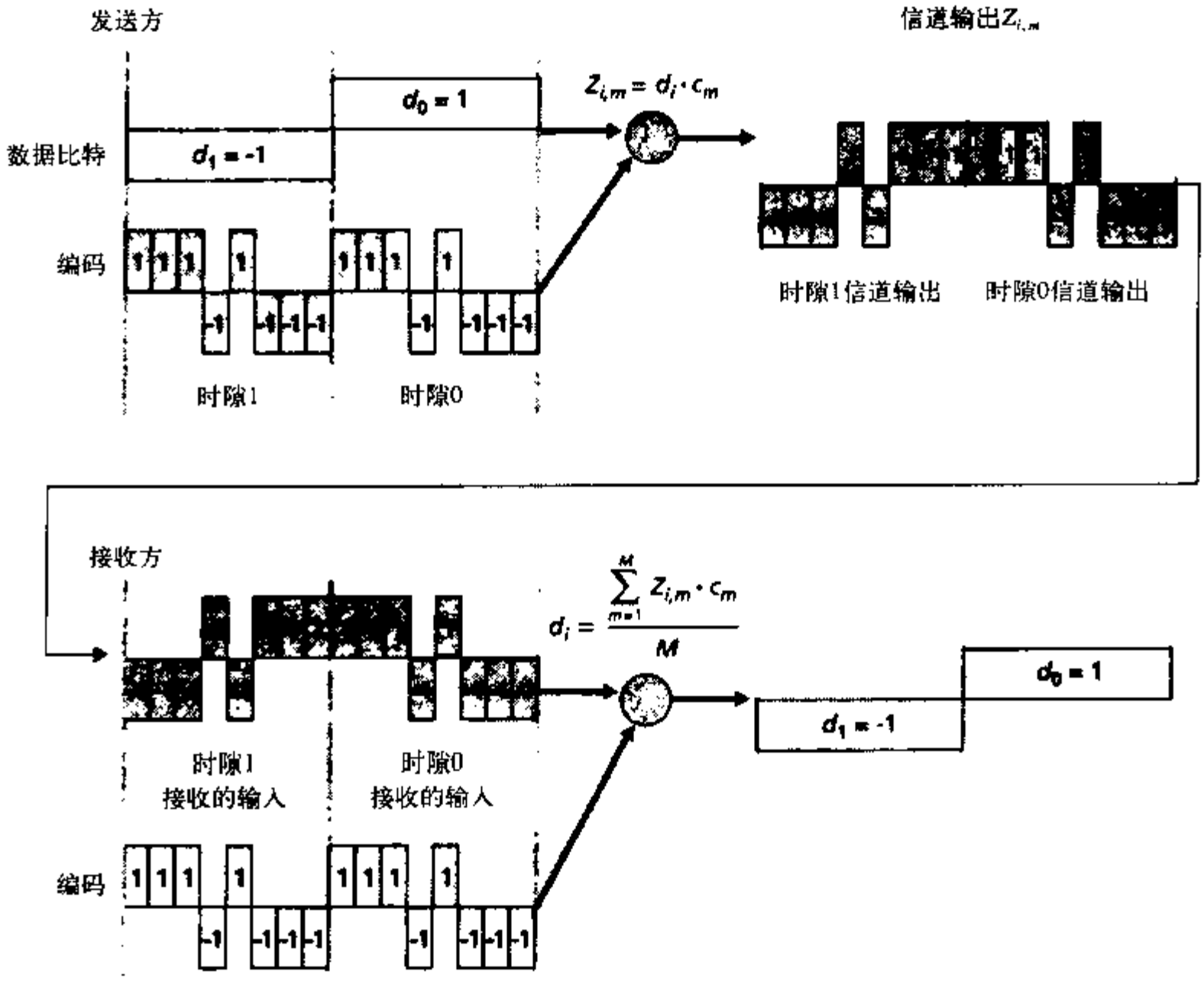


图6-5 一个简单的CDMA例子：发送方编码，接收方解码

为了说明CDMA是如何工作的，我们关注第i个数据比特d_i。对于d_i的比特传输时间的第m个微时隙，CDMA编码器的输出$Z_{i,m}$是d_i乘以分配的CDMA编码的第m比特c_m：

$$Z_{i,m} = d_i \cdot c_m \tag{6-1}$$

在简单的情况下，对没有干扰的发送方，接收方将收到编码的比特$Z_{i,m}$，并且恢复初始的数据比特d_i，计算如下：

$$d_i = \frac{1}{M} \sum_{m=1}^{M} Z_{i,m} \cdot c_m \tag{6-2}$$

读者可能想通过考察图6-5所示例子的细节，来明白使用等式（6-2）在接收方确实正确恢复初始数据比特。

然而，这个世界远不是理想化的，如上所述，CDMA必须在存在干扰发送方的情况下工作，这些发送方用分配的不同编码来编码和传输它们的数据。但是当一个发送方的数据比特和其他发送方发送的比特混在一起时，一个CDMA接收方怎样恢复该发送方的初始数据比特

呢？CDMA的工作有一种假设，即对干扰的传输比特信号是加性的，这意味着，例如在同一个微时隙中，如果3个发送端都发送1，第4个发送端发送－1，那么在那个微时隙中所有的接收方接收的信号都是2（因为1＋1＋1－1=2）。在存在多个发送方时，发送方s计算它编码后的传输$Z^s_{i,m}$，计算方式与用等式（6-1）完全相同。然而在第i个比特时隙的第m个微时隙期间，接收方现在收到的值是在那个微时隙中从所有N个发送方传输比特的总和：

$$Z^*_{i,m} = \sum_{s=1}^{N} Z^s_{i,m}$$

令人吃惊的是，如果仔细地选择发送方的编码，每个接收方只通过等式（6-2）中的同样的方式使用发送方的编码，就能够从聚合的信号中恢复一个给定的发送方发送的数据：

$$d_i = \frac{1}{M}\sum_{m=1}^{M} Z^*_{i,m} \cdot c_m \tag{6-3}$$

图6-6描述了两个发送方的CDMA例子。上部的发送方使用的M比特CDMA编码是（1，1，1，－1，1，－1，－1，－1），而下部的发送方使用的CDMA编码是（1，－1，1，1，1，－1，1，1）。图6-6描述了一个接收方是如何恢复从上部发送方发送的初始数据比特的。注意到这个接收方能够提取来自发送方1的数据，而不管来自发送方2的干扰传输。

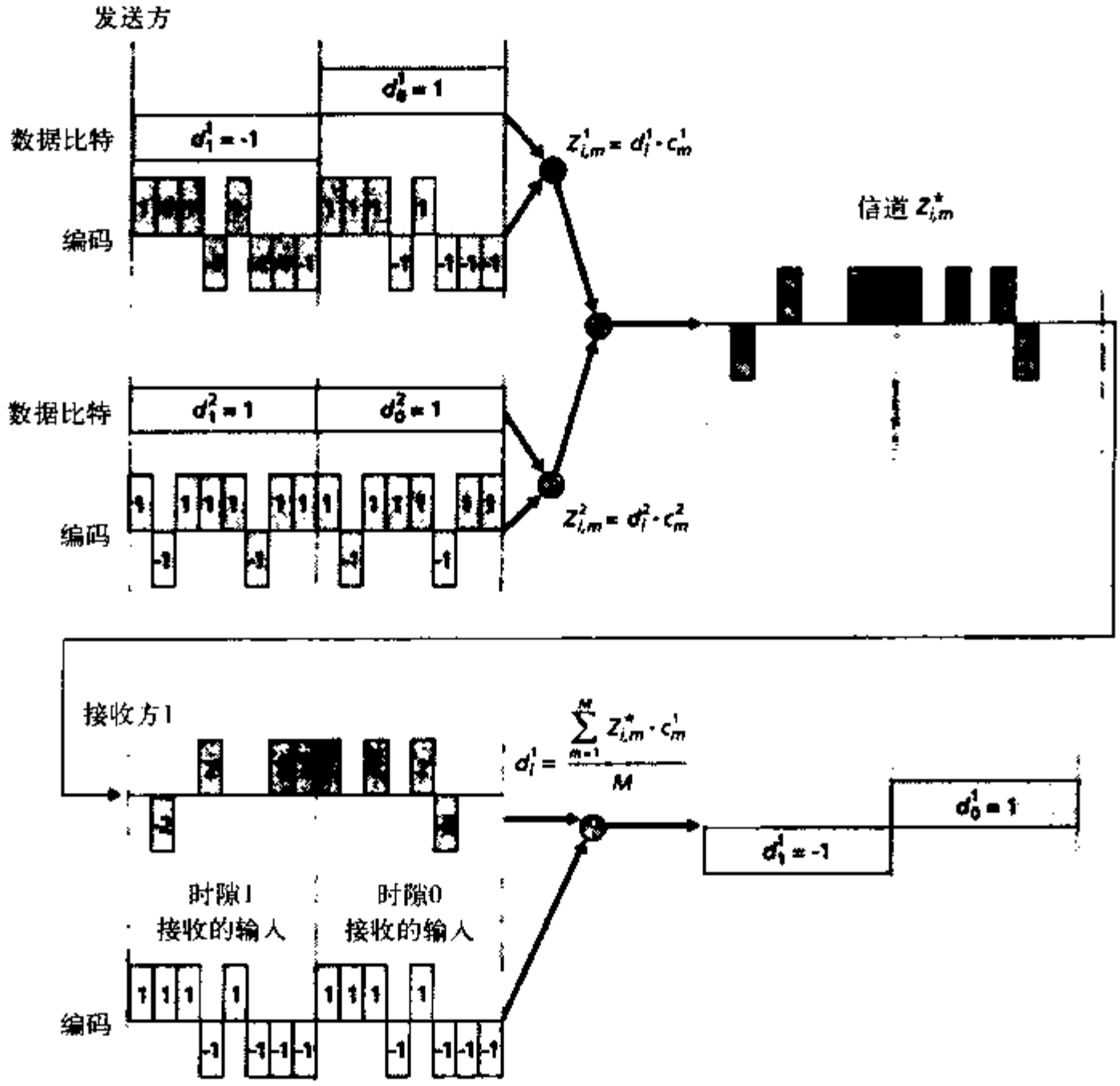


图6-6 有两个发送方的CDMA例子

再回到我们第5章中鸡尾酒会的类比，一个CDMA协议类似于让聚会客人使用多种语言来谈论；在这种情况下，人们实际上非常善于锁定他们能听懂语言的谈话，而过滤了其余的谈话。我们这里看到CDMA是一个分割协议，因为它划分编码空间（和时间或频率相对），并且给每个节点分配一段专用的代码空间。

我们这里对CDMA的讨论是简要的，实践中还必须讨论处理大量的困难问题。首先，为了使CDMA接收方能够提取一个特定的发送方的信号，必须仔细地选择CDMA编码。其次，我们的讨论假设接收方接收到的来自不同发送方的信号强度是相同的；这可能在实际中很难获得。有大量的文章讨论了有关CDMA的这样和那样的问题；详细内容见[Pickholtz 1982; Viterbi 1995]。

6.3 WiFi：802.11无线LAN

当前，无线LAN在工作场所、家庭、教育机构、咖啡馆、机场以及街头得到了广泛的使用，它已成为一种十分重要的因特网接入技术。尽管在20世纪90年代研发了许多有关无线LAN的标准和技术，其中有一类标准已经明显成为了赢家：IEEE 802.11无线LAN，也称为WiFi（Wireless Fidelity，无线保真）。在本节中，我们将详细研究802.11无线LAN，分析802.11帧结构、802.11媒体访问协议以及802.11 LAN与有线以太网LAN的互联。

有几套有关无线LAN的802.11标准，包括802.11b、802.11a和802.11g。表6-1总结了这些标准的主要特征。在本书写作时（2007年春季），接入点和LAN卡厂商提供了大量802.11g设备。一些双模式（802.11a/g）和三模式（802.11a/b/g）设备也可供使用。

表6-1 IEEE 802.11 标准小结

标 准	频率范围	数 据 率
802.11b	2.4～2.485 GHz	最高为11 Mbps
802.11a	5.1～5.8 GHz	最高为54 Mbps
802.11g	2.4～2.485 GHz	最高为54 Mbps

这3个802.11标准具有许多共同特征。它们都使用相同的媒体访问协议CSMA/CA，我们稍后将对其进行讨论。这3个标准都对它们的链路层帧使用相同的帧格式，它们都具有降低传输速率以到达更远距离的能力。并且这3个标准都允许“基础设施模式”和“自组织模式”。然而，如表6-1所示，这3个标准在物理层有一些重要的区别。

802.11b无线LAN具有11 Mbps的数据率，工作在不需要许可证的2.4～2.485 GHz的无线频谱上，与2.4 GHz电话和微波炉争用频谱。802.11a无线LAN具有高得多的比特率上，但它在更高的频率上运行。然而，由于运行的频率更高，802.11a LAN对于一定的功率级别而言传输距离较短，并且它受多路径传播的影响更大。802.11g LAN与802.11b LAN一样工作在较低频段上，并且与802.11b向后兼容（这样你能够逐步地升级802.11b的客户机），并具有与802.11a相同的高传输速率，使得用户能够更好地享受网络服务。

6.3.1 802.11体系结构

图6-7显示了802.11无线LAN体系结构的基本构件。802.11体系结构的基本构件模块是**基本服务集**（Basic Service Set，BSS），一个BBS通常包含一个或多个无线站点和一个在802.11术语中称为**接入点**（Access Point，AP）的中央基站（base station）。图6-7展示了两个BSS中

的AP，它们连接到一个互联设备上（如交换机或者路由器），互联设备又连接到因特网中。在一个典型的家庭网络中，有一个AP和一个将该BSS连接到因特网的路由器（通常与电缆或ADSL调制解调器封装在相同的盒子中）。

与以太网设备类似，每个802.11无线站点都具有一个6 字节的MAC地址，该地址存储在该站适配器（即802.11网络接口卡中）的固件中。每个AP的无线接口也具有一个MAC地址。与以太网类似，这些MAC地址由IEEE所管理，理论上是全球唯一的。

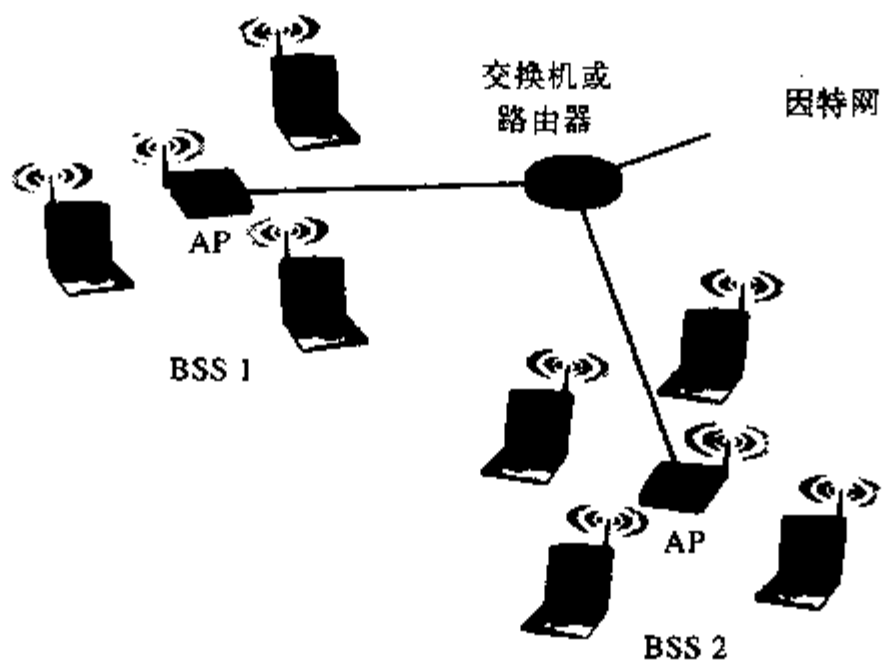


图6-7 IEEE 802.11 LAN体系结构

如6.1节所述，配置AP的无线LAN经常被称作**基础设施无线LAN**（infrastructure wireless LAN），其中的“基础设施”是指AP以及互联AP和路由器的有线以太网。图6-8显示了IEEE 802.11站点也能将它们组合在一起，以形成一个自组织网络，即一个无中心控制和与“外部世界”无连接的网络。这里，该网络由彼此发现已经相互接近且有通信需求的移动设备“动态”形成，并且在它们所处环境中没有预先存在的网络基础设施。当携带便携机的人们聚集在一起时（例如，在一个会议室、一列火车或者一辆汽车中），并且要在没有中央化的AP的情况下交换数据，一个自组织网络就可能形成了。随着通信便携设备的继续激增，人们会越来越关注自组织网络。然而在本节中，我们只关注基础设施无线LAN。

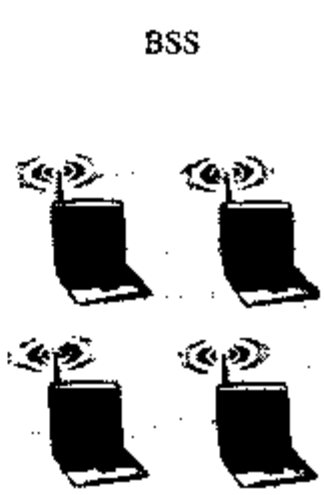


图6-8 IEEE 802.11自组织网络

信道与关联

在802.11中，每个无线站点在能够发送或者接收网络层数据之前，必须与一个AP相关联。尽管所有802.11标准都使用了关联，我们将特别在IEEE 802.11b/g环境中讨论这一主题。

当网络管理员安装一个AP时，他为该接入点分配一个单字或双字的**服务集标识符**（Service Set Identifier，SSID）。例如，当你在Microsoft Windows XP中“查看可用网络时”，将显示某范围内每个AP的SSID。管理员还必须为该AP分配一个信道号。为了理解信道号，回想前面讲过的802.11运行在2.4～2.485 GHz的频段中。在这85 MHz的频段内，802.11定义了11个部分重叠的信道。当且仅当两个信道由4个或更多信道隔开时它们才无重叠。特别是信道1、

6和11的集合是唯一的3个非重叠信道的集合。这意味着管理员可以在同一个物理网络中安装3个802.11b AP，为这些AP分配信道1、6和11，然后将每个AP都连接到一台交换机上，从而创建一个总计最大传输速率为33 Mbps的无线LAN。

既然我们已经对802.11信道有了基本了解，我们则可以描述一个有趣的（且并非完全不寻常的）情况，即有关WiFi丛林。WiFi丛林（WiFi jungle）是任意一个物理位置，在此处无线站点能从两个或多个AP中收到很强的信号。例如，在纽约城的许多咖啡馆中，无线站点可以从附近许多AP中选取一个信号。其中一个AP可能由该咖啡馆管理，而其他AP可能位于咖啡馆附近的住宅区内。这些AP中的每一个都可能位于不同的子网中，并被独立分配一个信道。

现在假定你带着自己的便携机进入这样一个WiFi丛林，寻求无线因特网接入和一个蓝莓松饼。假设在这个丛林中有5个AP。为了获得因特网接入，你的无线站点需要加入其中一个子网并因此需要与其中的一个AP相**关联**（associate）。关联意味着这一无线站点在自身和该AP之间创建一个虚拟线路。特别是，仅有关联的AP才向你的无线站点发送数据帧，并且你的无线站点也仅仅通过该关联AP向因特网发送数据帧。然而，你的无线站点是如何与某个特定的AP相关联的？更为根本的问题是，你的无线站点是如何知道哪个AP位于丛林中的呢？

802.11标准要求每个AP周期性地发送**信标帧**（beacon frame），每个信标帧包括该AP的SSID和MAC地址。你的无线站点为了得知正在发送信标帧的AP，扫描11个信道，找出来自可能位于该区域的AP所发出的信标帧（其中一些AP可能在相同的信道中传输，即这里有一个丛林）。通过信标帧了解到可用AP后，你（或者你的无线主机）选择一个AP用于关联。

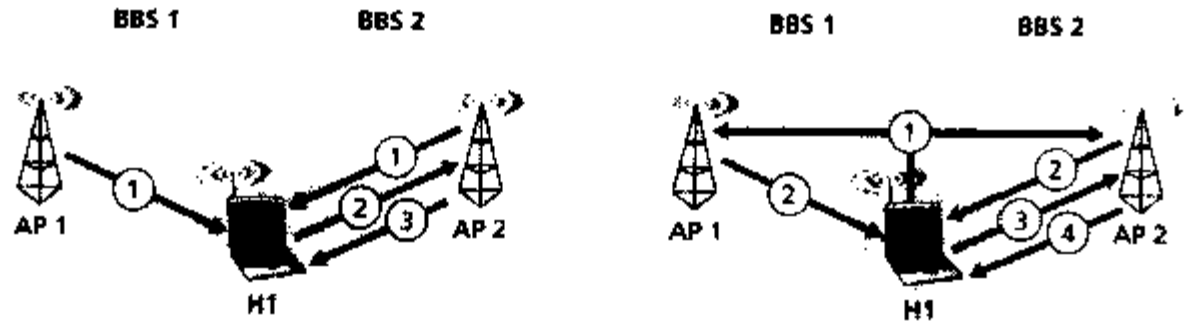
802.11标准没有指定选择哪个可用的AP进行关联的算法，该算法留给802.11固件和无线主机的软件设计者。通常，主机选择接收到具有最高信号强度的信标帧。虽然高信号强度好（例如可参见图6-3），信号强度将不是唯一决定主机收到性能的AP特性。特别是，所选择的AP可能具有强信号，但可能被其他附属主机（将需要共享该AP的无线带宽）所过载，而某未过载的AP由于信号稍弱而未被选择。选择AP的一些可替代的方法近来已被提出[Vasudevan 2005; Nicholson 2006; Sudaresan 2006]。有关信号强度的测量的讨论参见[Bardwell 2007]。

扫描信道和监听信标帧的过程称为**被动扫描**（passive scanning）（参见图6-9a）。无线主机也能够执行**主动扫描**（active scanning），这是通过向位于无线主机范围内的所有AP广播探测帧完成的，如图6-9b所示。AP用一个探测响应帧应答该探测请求帧，无线主机则能够在响应的AP中选择某AP与之相关联。

选定与之关联的AP后，无线主机向该AP发送一个关联请求帧，并且该AP以一个关联响应帧进行响应。注意到对于主动扫描需要这种第二个请求/响应握手，因为一个对初始探测请求的帧进行响应的AP并不知道主机选择哪个（可能多个）响应的AP进行关联，这与DHCP客户机能够从多个DHCP服务器进行选择有诸多相同之处（参见图4-21）。一旦与一个AP关联，主机希望加入该AP所属的子网（以4.4.2节中IP寻址的意义上说）中。这样，主机通常将通过关联的AP向该子网发送一个DHCP发现报文（参见图4-21），以获取在该AP子网中的一个IP地址。一旦获得地址，网络的其他部分将直接视你的主机为该子网中的另一台主机。

为了与特定的AP创建一个关联，无线站点可能要向该AP鉴别它自身。802.11无线LAN提供了几种不同的鉴别和接入方法。一种被许多公司采用的方法是，基于一个站点的MAC地址允许其接入一个无线网络，第二种被许多因特网咖啡屋采用的方法是应用用户名和口令。在两种情况下，AP通常与一个鉴别服务器进行通信，使用一种诸如RADIUS[RFC 2138]和DIAMETER[RFC 3588]的协议，在无线终端站和鉴别服务器之间中继信息。将鉴别服务器和

AP分离开来，使得一个鉴别服务器可以服务于多个AP；同时将（经常是敏感的）鉴别和接入的决定集中到单一服务器中，使得AP费用和复杂性较低。我们将在8.8节看到，定义802.11协议族安全性的新IEEE 802.11i协议就恰好采用了这一方法。



a) 被动扫描
① 自AP发送信标帧
② H1向选择的AP发送关联请求帧
③ 选择的AP向H1发送关联响应帧

b) 主动扫描
① 自H1广播探测请求帧
② 自AP发送探测响应
③ H1项选择的AP发送关联请求帧
④ 选择的AP向H1发送关联的响应帧

图6-9 对接入点的主动扫描和被动扫描

6.3.2 802.11 MAC协议

一旦某无线站点与一个AP相关联后，它就可以经该接入点开始发送和接收数据帧。然而因为许多站点可能希望同时经同样的信道传输数据帧，因此需要一个多路访问协议来协调传输。这里，**站点**（station）可以是一个无线站点，也可以是一个AP。正如在第5章和6.2.1节中讨论的那样，宽泛地讲有三类多路访问协议：信道划分（包括CDMA）、随机访问和轮流。受以太网和随机访问协议巨大成功的鼓舞，802.11的设计者为802.11无线LAN选择了一种随机访问协议，称作**带碰撞避免的CSMA**（CSMA with collision avoidance，CSMA/CA）。与以太网的CSMA/CD相似，CSMA/CA中的“CSMA”代表“载波侦听多路访问”，意味着每个站点在传输之前侦听信道，并且一旦侦听到该信道忙则抑制传输。尽管以太网和802.11都使用载波侦听多址接入，这两种MAC协议有重要的区别。首先，802.11使用碰撞避免而非碰撞检测；其次，由于无线信道相对较高的误比特率，802.11（不同于以太网）使用链路层确认/重传（ARQ）机制。我们将在下面讨论802.11的碰撞避免和链路层确认机制。

在5.3和 5.4节曾讲过，使用以太网的碰撞检测算法，以太网节点在发送过程中监听信道。一旦发送过程中检测到另一节点也在发送，则放弃自己的发送，并且在等待一个小的随机时间后再次发送。与802.3以太网协议不同，802.11 MAC协议并未实现碰撞检测。这主要由两个原因所致：

- 检测碰撞的能力要求站点具有同时发送（站点自己的信号）和接收（检测其他站点是否也在发送）的能力。因为在802.11适配器上，接收信号的强度通常远远小于发送信号的强度，构建具有检测碰撞能力的硬件代价较大。
- 更重要的是，即使适配器可以同时发送和监听信号（并且假设它一旦侦听到信道忙就放弃发送），适配器也会由于隐藏终端问题和衰减问题而无法检测到所有的碰撞，参见6.2节的讨论。

由于802.11无线局域网不使用碰撞检测，一旦站点开始发送一个帧，它就完全地发送该帧；也就是说，一旦站点开始发送，就不会返回。正如人们可能猜想的那样，碰撞存在时仍

发送整个数据帧（尤其是长数据帧）将严重降低多路访问协议的性能。为了降低碰撞的可能性，802.11采用几种碰撞避免技术，我们稍后讨论它们。

然而，在考虑碰撞避免之前，我们首先需要分析802.11的**链路层确认**（link-layer acknowledgment）机制。6.2节讲过当无线LAN中一个站点发送一个帧时，该帧会由于多种原因不能无损地到达目的站点。为了处理这种不可忽视的故障情况，802.11 MAC使用链路层确认。如图6-10所示，目的站点收到一个通过CRC校验的帧后，它等待一个被称作**短帧间间隔**（Short Inter-Frame Spacing，SIFS）的一小段时间，然后发回一个确认帧。如果发送站点在给定的时间内未收到确认帧，它假定出现了错误并重传该帧，使用CSMA/CA协议访问该信道。如果在若干固定次重传后仍未收到确认，发送站将放弃发送并丢弃该帧。

讨论过802.11如何使用链路层确认后，我们开始描述802.11的CSMA/CA协议。假设一个站点（无线站点或者AP）有一个帧要发送。

1）如果初始时某站点监听到信道空闲，它将在一个被称作**分布式帧间间隔**（Distributed Inter-Frame Space，DIFS）的短时间段后发送该帧，如图6-10所示。

2）否则，该站点选取一个随机回退值并且在侦听信道空闲时递减该值。当侦听到信道忙时，计数值保持不变。

3）当计数值减为0时（注意到这只可能发生在信道被侦听为空闲时），该站点发送整个数据帧并等待确认。

4）如果收到确认，发送站知道它的帧已被目的站正确接收了。如果该站点要发送另一帧，它将从第二步开始CSMA/CA协议。如果未收到确认，发送站将重新进入第二步中的回退阶段，并从一个更大的范围内选取随机值。

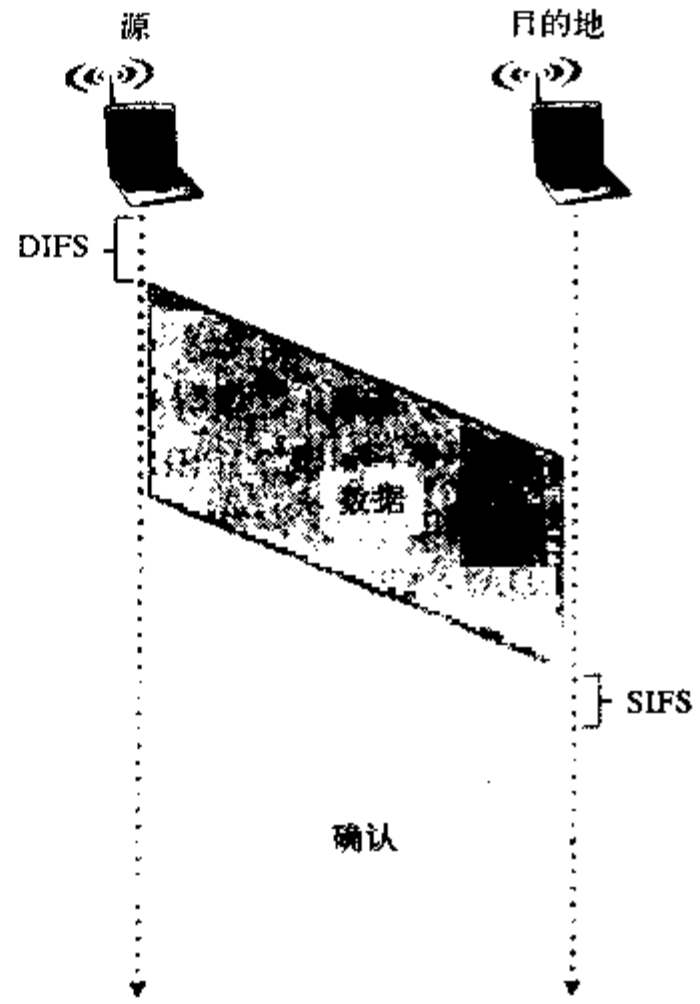


图6-10 802.11使用链路层确认

前面讲过，在以太网的CSMA/CD多路访问协议下（5.5.2节），一旦侦听到信道空闲，站点开始发送。然而，使用CSMA/CA，该站点计数时抑制传输，即使它侦听到该信道空闲。为什么CSMA/CD和CSMA/CA要采用了不同的方法呢？

为了回答这一问题，我们首先考虑这样一种情形：两个站点分别有一个数据帧要发送，但是，由于侦听到第三个站点已经在传输，双方都未立即发送。使用以太网的CSMA/CD协议，两个站点将会在检测到第三方发送完毕后立即开始发送。这将导致一个碰撞，这在CSMA/CD协议中，这种碰撞并非是一个严重的问题，因为两个站点检测到碰撞后都会放弃它们的发送，从而避免了由于碰撞而造成的该帧剩余部分的无用发送。而在802.11中情况却完全不同，因为802.11并不检测碰撞和放弃发送，受碰撞的帧仍将被完全发送。802.11的目标因此是尽可能地避免碰撞。在802.11中，如果两个站点侦听到信道忙，它们都将立即进入随机回退——希望选择不同的回退值。如果这些值的确不同，一旦信道空闲，其中的一个站点将在另一个之前发送，并且（如果两个站点均未对对方隐藏）“失败站点”将会听到“胜利站点”的信号，冻结它的计数器，并在胜利站点完成

传输之前一直抑制传输。通过这种方式，避免了高代价的碰撞。当然，使用802.11在这种情况下仍可能出现碰撞：两个站点可能互相是隐藏的，或者两者可能选择了非常靠近的随机回退值，使得从先开始站点的传输也必须到达第二个站点。回想前面，我们在图5-14的环境中讨论随机访问算法时遇到过这个问题。

1. 处理隐藏终端：RTS和CTS

802.11 MAC协议也包括了一个极好的预约机制（但为可选项），以帮助在出现隐藏终端的情况下避免碰撞。我们在图6-11的环境下研究这种机制，其中显示了两个无线站点和一个接入点。这两个无线站点都在该AP的覆盖范围内（其覆盖范围显示为阴影圆环），并且两者都与该AP相关联。然而，由于衰减，无线节点的信号范围局限在图6-11所示的阴影圆环内部。因此，尽管每个无线站点对AP都不隐藏，两者彼此却是隐藏的。

图6-11 隐藏终端的例子：H1和H2彼此互相隐藏

现在我们考虑为什么隐藏终端会导致问题。假设站点H1正在传输一个帧，并且在H1传输的中途，站点H2要向AP发送一个帧。由于H2未听到来自H1的传输，它将首先等待一个DIFS间隔，然后发送该帧，这就导致了碰撞。从而在H1和H2的整个发送阶段，信道都被浪费了。

为了避免这一问题，IEEE 802.11协议允许站点使用一个短**请求发送**（Request to Send，RTS）控制帧和一个**允许发送**（Clear to Send，CTS）控制帧来预约对信道的访问。当发送方要发送一DATA帧时，它能够首先向AP发送一个RTS帧，指出传输DATA帧和确认帧需要的总时间。当AP收到RTS帧后，它广播一个CTS帧作为响应。该CTS帧有两个目的：给发送方明确的发送允许，也指示其他站点在预约期内不要发送。

因此，在图6-12中，在H1传输DATA帧前，首先广播一个RTS帧，该帧能被在它圈内的包括AP在内的所有站点听到。AP然后用一个CTS帧响应，该帧也被在它范围内的包括H1和H2的所有站点听到。站点H2听到CTS后，在CTS帧中指明的时间内将抑制发送。RTS、CTS、DATA和ACK帧如图6-12所示。

RTS帧和CTS帧的使用可以在两个重要方面提高性能：

- 隐藏终端问题被缓解了，因为长DATA帧只有在信道预约后才被传输。
- 因为RTS帧和CTS帧较短，涉及RTS帧和CTS帧的碰撞将仅持续很短的RTS或CTS帧持续期。一旦RTS帧和CTS帧被正确传输，后续的DATA帧和ACK帧应当能无碰撞地发送。

建议读者去查看本书配套网站上的802.11 Java小程序。这个交互小程序演示了CSMA/CA协议，包括RTS/CTS交换序列。

尽管RTS/CTS交换可以帮助降低碰撞，但它同样引入了时延以及消耗了信道资源。因此，

RTS/CTS交换仅仅用于为长数据帧预约信道。在实际中，每个无线站点可以设置一个RTS门限值，仅当帧长超过门限值时，才使用RTS/CTS序列。对许多无线站点而言，默认的RTS门限值大于最大帧长值，因此对所有发送的DATA帧，RTS/CTS序列都被跳过。

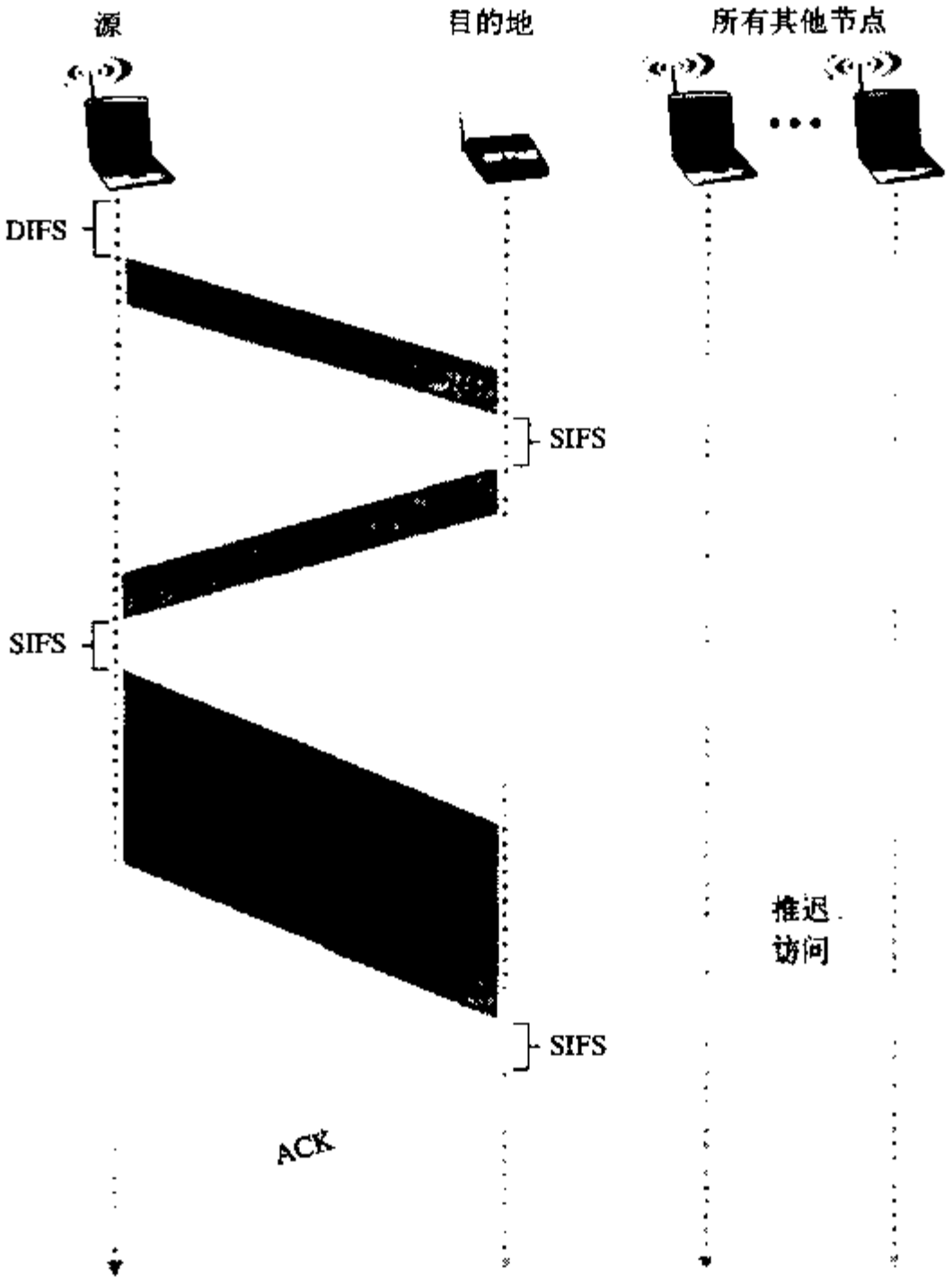


图6-12 使用RTS和CTS帧的碰撞避免

2. 使用802.11作为一个点对点链路

我们的讨论目前关注于多路访问环境中802.11的应用。应该指出，如果两个节点每个都具有一个定向天线，那么它们可以将其定向天线指向对方，并基本上是在一个点对点的链路上运行802.11协议。假如商用802.11硬件产品价格低廉，那么使用定向天线以及增加传输功率将使得802.11成为一个在数十公里距离中提供无线点对点连接的廉价手段。[Raman 2007]描述了这样一个运行于印度恒河郊区平原上的多跳无线网络，其中包含了点对点802.11链路。

6.3.3 IEEE 802.11帧

尽管802.11帧与以太网帧有许多共同特点，但它也包括了许多特定用于无线链路的字段。8021.11帧如图6-13所示，在该帧上的每个字段上面的数字代表该字段的字节长度；在该帧控制字段中，每个子字段上面的数字代表该子字段的比特长度。现在我们研究该帧中各字段以及帧控制字段中一些重要的子字段。

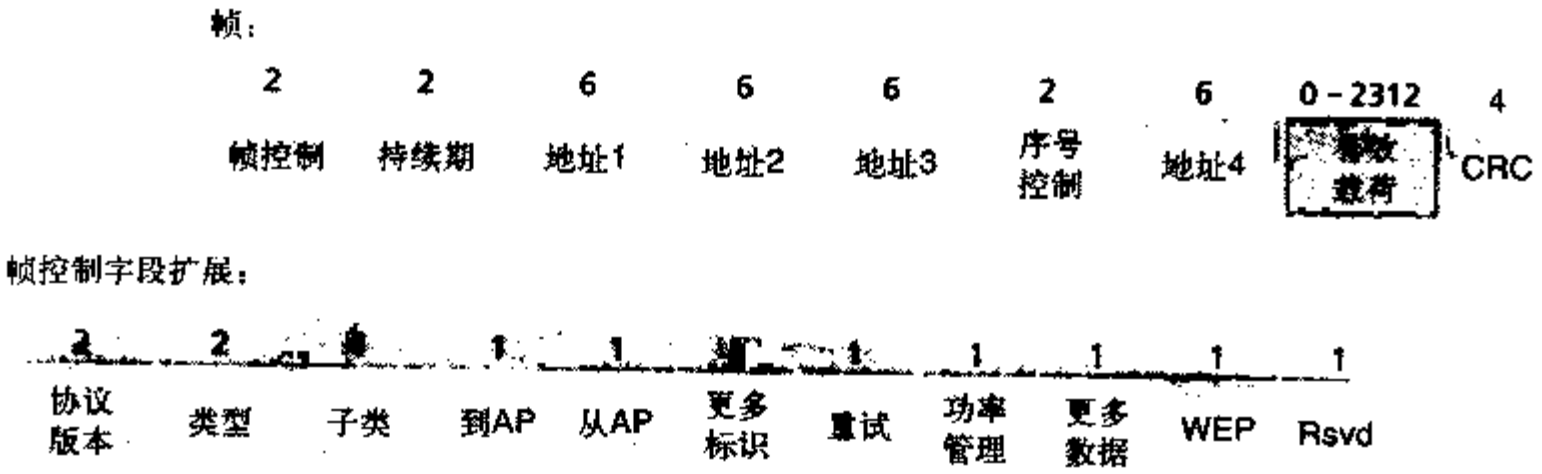


图6-13 802.11帧

1. 有效载荷与CRC字段

帧的核心是有效载荷，它通常是由一个IP数据报或者ARP分组组成。尽管这一字段允许的最大长度为2312字节，但它通常小于1500字节。如同以太网帧一样，802.11帧包括一个循环冗余校验（CRC），从而接收方可以检测所收到帧中的比特错误。如我们所看到的那样，比特错误在无线局域网中比在有线局域网中更加普遍，因此CRC在这里更加有用。

2. 地址字段

802.11帧中最引人注意的不同之处也许是它具有4个地址字段，其中每个都可以包含一个6字节的MAC地址。为什么要4个地址字段？像以太网中那样，一个源MAC地址字段和一个目的MAC地址字段不就足够了？事实证明，出于互联目的需要3个地址字段，特别是将网络层数据报从一个无线站点通过一个AP送到一台路由器接口。当AP在自组织模式中互相转发时使用第4个地址。由于我们这里仅仅考虑基础设施网络，所以只关注前3个地址字段。802.11标准定义这些字段如下：

- 地址2是传输帧的站点的MAC地址。因此，如果一个无线站点传输帧，该站点的MAC地址就被插入在地址2字段中。类似地，如果一个AP传输帧，该AP的MAC地址也被插入在地址2字段中。
- 地址1是要接收帧的无线站点的MAC地址。因此，如果一个移动无线站点传输帧，地址1包含了目的地AP的MAC地址。类似地，如果一个AP传输帧，地址1包含目的无线站点的MAC地址。
- 为了理解地址3，回想BSS（由AP和无线站点组成）是一个子网的一部分，并且这个子网经一些路由器接口与其他子网相连。地址3包含这个路由器接口的MAC地址。

为了对地址3的目的有更深入的理解，我们观察在图6-14环境中的网络互联的例子。在这幅图中，有两个AP，每个AP负责一些无线站点。每个AP到路由器有一个直接连接，路由器依次又连接到全球因特网。我们应当记住AP是链路层设备，它既不能“说”IP又不理解IP地址。现在考虑将一个数据报从路由器接口R1移到无线站点H1。路由器并不清楚在它和H1之间有一个AP；从路由器的观点来说，H1仅仅是路由器所连接的子网中的一台主机。

- 路由器知道H1的IP地址（从数据报的目的地址中得到），使用ARP决定H1的MAC地址，这与在普通的以太网LAN中相同。获取H1的MAC地址后，路由器接口R1将该数据报封装在一个以太网帧中。该帧的源地址字段包含了R1的MAC地址，目的地址字段包含H1的MAC地址。
- 当该以太网帧到达AP后，该AP在将其传输到无线信道前，先将该802.3以太网帧转换为

一个802.11帧。如前所述，AP将地址1和地址2分别填上H1的MAC地址和其自身的MAC地址。对于地址3，AP插入R1的MAC地址。通过这一方式，H1可以确定（从地址3）将数据报发送到子网中的路由器接口的MAC地址。

现在考虑当从无线站点H1移动一个数据报到R1时H1进行响应时发生的情况。

- H1生成一个802.11帧，分别用AP的MAC地址和H1的MAC地址填上地址1和地址2字段，如上所述。对于地址3，H1插入R1的MAC地址。
- 当AP接收该802.11帧后，它将其转换为以太网帧。该帧的源地址字段是H1的MAC地址，目的地址字段是R1的MAC地址。因此，地址3允许AP在构建以太网帧时能够确定目的MAC地址。

总之，地址3在BSS和有线局域网互联中起着重要作用。

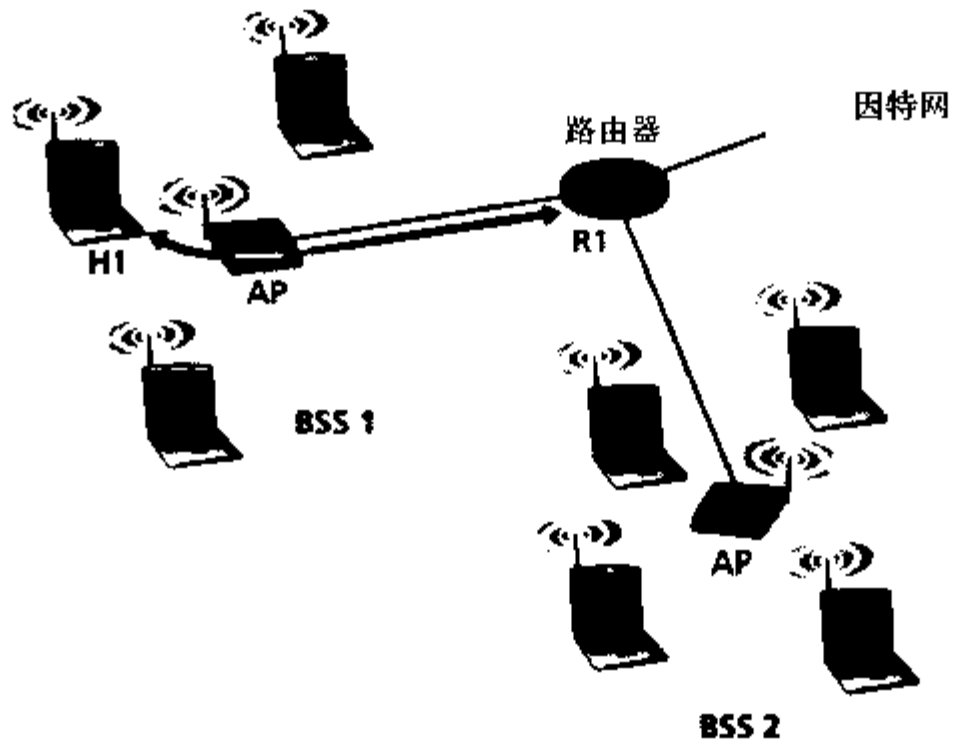


图6-14 在802.11帧中地址字段的使用：在H1和R1之间移动帧

3. 序号、持续期和帧控制字段

前面讲过在802.11网络中，每当一个站点正确收到一个来自于其他站点的帧后，它就回发一个确认。因为确认可能会丢失，发送站点可能会发送一个给定帧的多个拷贝。正如我们在rdt2.1协议讨论中所见（3.4.1节），使用序号可以使接收方区分新传输的帧和以前帧的重传。因此在802.11帧中的序号字段在链路层与在第3章中运输层有着完全相同的作用。

前面讲过802.11协议允许传输节点预约信道一段时间，包括传输其数据帧的时间和传输确认的时间。这个持续期值被包括在该帧的持续期字段中（数据帧和RTS帧及CTS帧中均存在）。

如图6-13所示，帧控制字段包括许多子字段，我们将对其中比较重要的子字段进行介绍。更加完整的讨论请参见802.11规范[Held 2001; Crow 1997; IEEE 802.11 1999]。类型和子类型字段用于区分关联、RTS、CTS、ACK和数据帧。To和From字段用于定义不同地址字段的含义。（这些含义随着使用自组织模式或者基础设施模式而改变，以及在使用基础设施模式时，也随着是无线站点还是AP在发送帧而变化。）最后，WEP字段指示了是否使用加密（WEP将在第8章讨论）。

6.3.4 在相同的IP子网中的移动性

为了增加无线LAN的物理范围，公司或大学经常会在同一个IP子网中部署多个BSS。这自然就引出了在多个BSS之间的移动性问题，即无线站点如何在维持正在进行的TCP会话的情况下，无缝地从一个BSS移动到另一个BSS。正如我们将在本小节中看到的那样，当这些BSS属于同一子网时，移动性可以用一种相对直接的方式解决。当站点在不同子网间移动时，就需要更为复杂的移动性管理协议了，如我们将在6.5节和6.6节学习的那些协议。

我们现在看一个在同一子网中的不同BSS之间的移动性的特定例子。图6-15显示了一台主机H1与两个互联的BSS以及该主机从BSS1移动到BSS2的情况。因为在这个例子中连接两个BSS的互联设备不是一台路由器，故在两个BSS中的所有站点包括AP都属于同一个IP子网。因此，当H1从BSS1移动到BSS2时，它可以保持自己的IP地址和所有正在进行的TCP连接。如果互联设备是一台路由器，则H1必须在它移动进入的子网中获得一个新地址。这种地址的变化将打断（并且最终终止）H1处的任何进行中的TCP连接。在6.6节中，我们将看到有一种网络层移动性协议（如移动IP）能用来避免该问题。

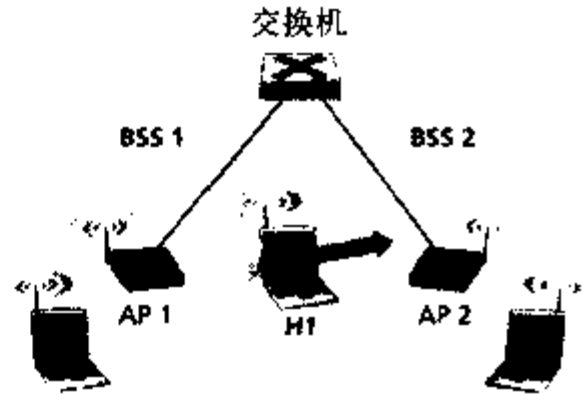


图6-15 同一子网中的移动性

但是H1从BSS1移动到BSS2时具体会发生哪些事呢？随着H1逐步远离AP1，H1检测到来自AP1的信号逐渐减弱并开始扫描一个更强的信号。H1收到来自AP2的信标帧（在许多公司和大学的设置中它与AP1有相同的SSID）。H1然后与AP1解除关联，并与AP2关联起来，同时保持其IP地址和维持正在进行的TCP会话。

从主机和AP的角度说，这就处理了切换问题。但对图6-15中交换机又会发生什么样的情况呢？交换机如何知道主机已经从一个AP移动到另一个AP呢？回想第5章所述，交换机是“自学习”的，并且自动构建它们的转发表。这种自学习的特征很好地处理了偶尔的移动（例如，一个雇员从一个部门调转到另一个部门）。然而，交换机不是设计用来支持用户在不同BSS间高度移动而同时又希望保持TCP连接的。为理解这一问题，回想在移动之前，交换机在其转发表中有一个表项，对应H1的MAC地址与通过它能够到达H1的出交换机端口。如果H1初始在BSS1中，则发往H1的数据报将经AP1导向H1。然而，一旦H1与BSS2关联，它的帧应当被导向AP2。一种解决方法（真有点不规范）是在新的关联形成后，让AP2以H1的源地址向交换机发送一个以太网广播帧。当该交换机收到该帧后，更新其转发表，使得H1可以通过AP2到达。802.11f标准小组正在开发一个AP间的协议来处理这些以及相关的问题。

6.3.5 802.11中的高级特色

我们将简要地讨论802.11网络中具有的两种高级能力，以此来丰富我们关于802.11的讨论。如我们所见，这些能力并不是完全特定于802.11标准的，而是在该标准中可能由特定机制产生的。这使得不同的厂商使用他们自己（专用）的方法来实现这些能力，这也许能让他们增强竞争能力。

1. 802.11速率适应

我们在前面图6-3中看到，不同的调制技术（提供了不同的传输速率）适合于不同的SNR情况。考虑这样一个例子，一个802.11用户最初离基站20米远，这里信噪比高。在此高信噪

比的情况下，该用户能够使用提供高传输速率的物理层调制技术与基站进行通信，同时维持低BER。这个用户多么幸福啊！假定该用户开始移动，向离开基站的方向走去，随着与基站距离的增加，SNR一直在下降。在这种情况下，如果在用户和基站之间运行的802.11协议所使用的调制技术没有改变的话，随着SNR减小，BER将高得不可接受，最终所有传输的帧将不能正确收到。

由于这个原因，某些802.11实现具有一种速率自适应能力，该能力根据当前和近期信道特点自适应地选择下面的物理层调制技术。Lucent的WaveLAN-II 802.11b实现[Kamerman 1997]提供了多个数据传输速率。如果一个节点连续发送两个帧而没有收到确认（信道上一个比特差错的隐式指示），该传输速率降低到前一个较低的速率。如果10个帧连续得到确认，或如果根据自上次降速以来的时间定时器超时，该传输速率提高到上一个较高的速率。这种速率适应机制与TCP的拥塞控制机制具有相同的“探测”原理，即当条件好时（反映为收到ACK），增加传输速率，除非某个“坏事”发生了（ACK没有收到），当某个“坏事”发生了，减小传输速率。因此802.11的速率适应能力和TCP拥塞控制类似于年幼的孩子，他们不断地向父母要求越来越多（如幼儿要糖果，青少年要求推迟睡觉），直到父母亲最后说“够了！”，孩子们不再要求了（仅当以后情况已经变好了才会再次尝试）。已经提出了许多其他方案以改善这个基本的自动速率调整方案[Holland 2001; Lacage 2004]。

2. *功率管理*

功率是移动设备的宝贵资源，因此802.11标准提供了功率管理能力，以使802.11节点最小化它们的侦听、传输和接收功能和其他需要“打开”电路的时间量。802.11功率管理按下列方式运行。一个节点能够明显地在睡眠和唤醒状态之间交替（像在课堂上睡觉的学生！），通过将802.11帧首部中的功率管理比特设置为1，某节点向接入点指示它将打算睡眠。设置节点中的一个定时器，使得正好在AP计划发送它的信标帧前唤醒节点（前面讲过AP通常每100 ms发送一个信标帧）。因为AP从设置功率传输比特知道哪个节点打算睡眠，该AP知道它不应当向这个节点发送任何帧，先缓存目的为睡眠主机的任何帧待以后再传输。

在AP发送信标帧前，恰好唤醒节点，并迅速进入全面活动状态（与睡觉的学生不同，这种唤醒仅需要250μs[Kamerman 1997]）。由AP发送的信标帧包含了一个节点列表——这些节点的帧被缓存在AP中。如果节点没有缓存的帧，它能够返回睡眠状态。否则，该节点能够通过向AP发送一个探询报文明确地请求发送缓存的帧。对于100μs的信标之间的时间，250μs的唤醒时间以及同样小的信标帧接收和检查以确保不存在缓存帧的时间，没有帧要发送和接收的节点能够睡眠99%的时间，从而大大节省了能源。

6.3.6 802.11以外的标准：蓝牙和WiMAX

如图6-2所示，IEEE 802.11 WiFi标准主要用于相距多达100 m的设备间的通信（当使用802.11具有定向天线的点对点配置时除外）。两个其他的IEEE 802协议，即蓝牙（定义在IEEE 802.15.1标准中[IEEE 802.15 2007]）和WiMAX（定义在IEEE 802.16标准中[IEEE 802.16d 2004; IEEE 802.16e 2005]），是分别用于短距离和长距离的标准。

1. *蓝牙*

IEEE 802.15.1网络以小范围、低功率和低成本运行，它本质上是一个低功率、短距离、低速率的“电缆替代”技术，用于互联笔记本、外围设备、蜂窝电话和PDA；而802.11是一个大功率、中等范围、高速率的“接入”技术。为此，802.15.1网络有时被称为无线个人区域

网络（Wireless Personal Area Network，WPAN）标准。802.15.1的链路层和物理层基于早期的用于个人区域网络的蓝牙（Bluetooth）规范[Held 2001，Bisdikian 2001]。802.15.1网络以TDM方式工作于无需许可证的2.4 GHz无线电波段，每个时隙长度为625 μs。在每个时隙内，发送方利用79个信道中的一个进行传输，同时从时隙到时隙以一个已知的伪随机方式变更信道。这种称作**跳频扩展频谱**（Frequency-Hopping Spread Spectrum，FHSS）的信道跳动的形式将传输及时扩展到整个频谱。802.15.1能够提供高达4Mbps的数据率。

802.15.1网络是自组织网络：不需要网络基础设施（如接入点）来互联802.15.1设备。因此，802.15.1设备必须自己进行组织。802.15.1设备首先组织成一个多达8个活动设备的**皮可网**（piconet）。如图6-16所示，其中一个被指定为主设备，其余充当从设备。主节点真正控制皮可网，它的时钟确定了皮可网中的时间，它可以在每个奇数时隙中发送，而从设备仅当主设备在前一时隙与其通信后才可以发送，并且只能发送给主设备。除了从设备，网络中也可以有多达255个寄放（parked）设备。这些设备仅当其状态被主节点从“寄放”转换为“活动”之后才可以进行通信。

希望了解更多有关802.15.1 WPAN信息的读者可以查阅蓝牙参考资料[Held 2001，Bisdikian 2001]，或者IEEE 802.15Web网站[IEEE 802.15 2007]。

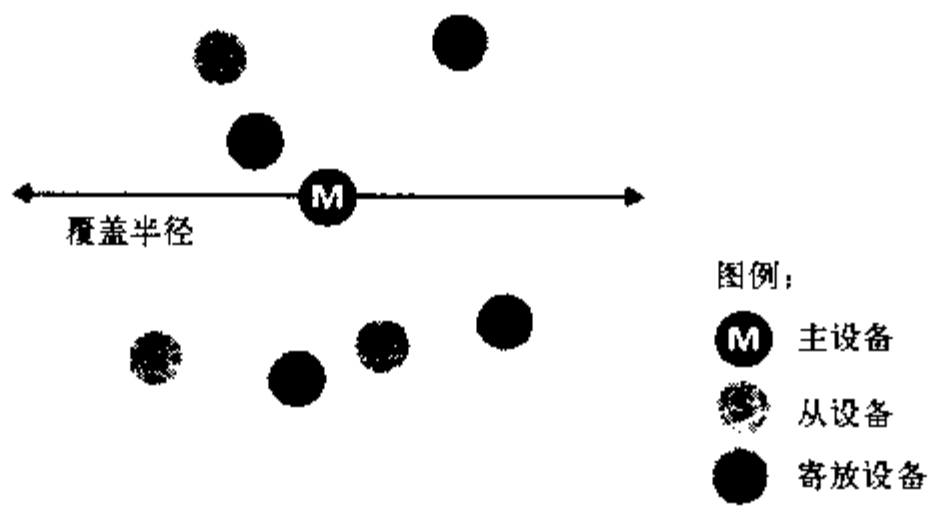


图6-16 蓝牙皮可网

2. WiMAX

WiMAX（全球微波接入互操作）是IEEE 802.16标准家族，该标准的目标是通过广阔的区域以可与电缆调制解调器和ADSL网络相比的速率，向大量的用户交付无线数据。802.16d更新了早期802.16a标准。802.16e标准的目标是支持以每小时70～80英里速度的移动性（相当于欧洲以外的大部分国家的高速公路的速度），具有一种用于小型、资源受限设备（如PDA、电话和膝上电脑）的不同的链路结构。

802.16体系结构基于基站的概念，这些基站在中央为大量潜在的与该基站相关联的客户机（称为客户站点）服务。在此意义下，WiMAX与WiFi的基础设施模式和蜂窝电话网络相像。该基站根据图6-17中所示的TDM帧结构，与下游（从基站到客户站点）和上游（从客户站点到基站）方向协调链路层分组的传输。我们这里将使用术语“分组”而不是术语“帧”，以区分链路层的数据单元与图6-17中所示的TDM帧结构。WiMAX因此以时分多路复用（TDM）方式运行，尽管成帧时间是可变的，如下面所讨论的那样。我们注意到WiMAX也定义了一种

FDM运行模式，但我们将不在这里讨论它。

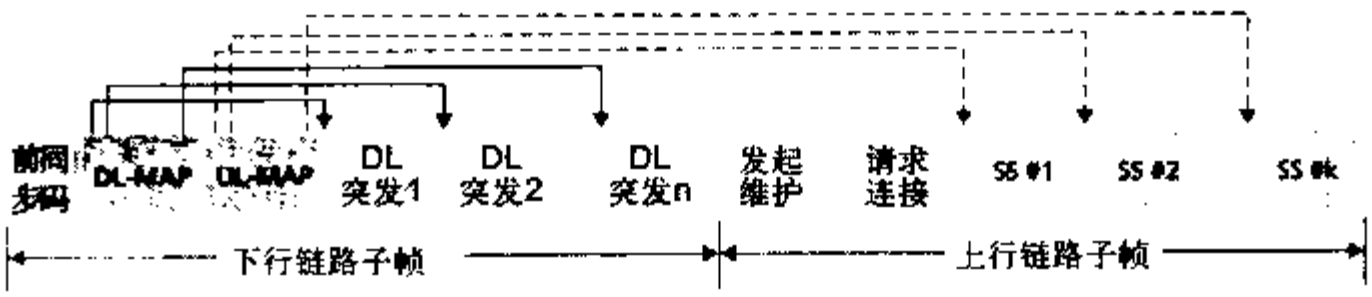


图6-17 802.16 TDM帧结构

在该帧的开始，基站首先发送一个下游MAP（媒体接入协议）列表的报文，该报文通知用户站点的物理层性质（调制方案、编码和纠错参数），这些性质将用于传输帧中的后继分组突发。在帧中可以有多个突发，并在一个突发中有多个分组目的地是给定的用户站点。在该突发中的所有分组都由该基站使用相同的物理层性质进行传输。然而，这些物理层性质可能根据不同的突发而改变，允许基站在最适合接收用户站点的物理层传输方案中选取。该基站根据到每个接收方所估计的当前信道条件，可以选择在该帧期间将发送的接收方集合。这种**机会主义调度法**（opportunistic scheduling）[Bender 2000, Kulkarni 2005]将物理层协议与发送方和接收方之间的信道条件相匹配，基于信道条件选择将分组发送到哪个接收方，这使得基站能最好地利用无线介质。WiMAX标准并不强制要求在给定的情况下必须使用一种特定的物理层参数集合。这留给WiMAX设备厂商和网络运行者来决定。

WiMAX基站也规范了客户站点通过使用UL-MAP报文接入上游信道的方式。这些报文控制了时间量，每个用户站点给定了接入在后继上行链路子帧中的信道。此外，WiMAX标准并不强制规定任何特别的策略来向客户机分配上行信道时间，这是一个留给网络运行者的决定。取而代之的是，WiMAX为实现能够向不同的用户站点给出不同的信道接入时间的策略，提供一种机制（例如UL-MAP控制报文）。上游的子帧的初始部分用于向用户传输下列报文：无线链路控制报文、在WiMAX网络中请求接入和鉴别的报文和高层与管理相关的控制报文（如DHCP和SNMP）。

图6-18显示了WiMAX MAC分组的格式。我们这里谈及的唯一字段是首部中的连接标识符。WiMAX是一个面向连接的体系结构，允许每个连接具有一个相关的服务质量（QoS）、流量参数和其他信息。提供该QoS的方式也与网络运营者有关。WiMAX提供了一种低层次机制（例如，信道估计和连接准入请求字段携带基站与主机之间的信息），这既不是提供QoS的整体方法，也不是提供QoS的策略。即使每个用户站点通常具有一个48比特的MAC地址（如

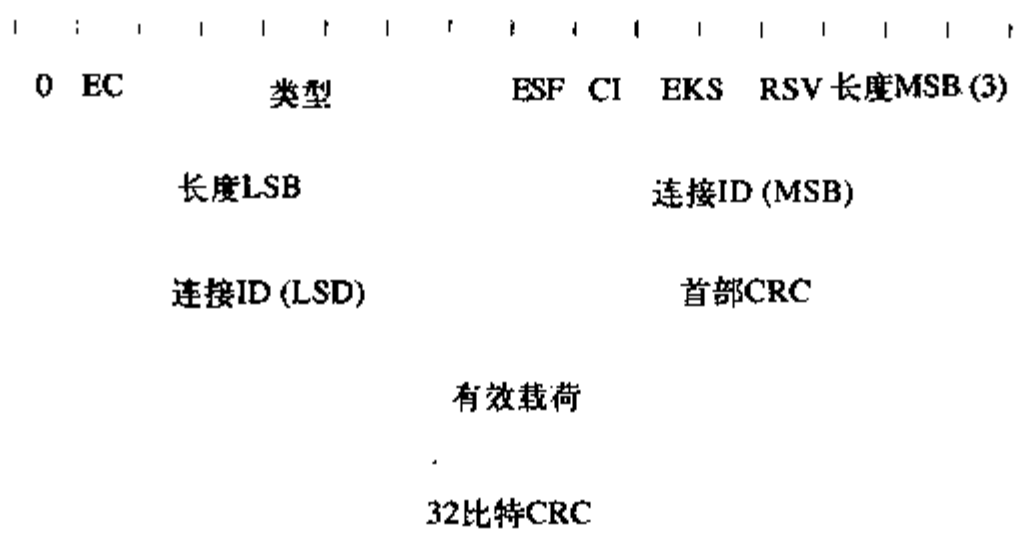


图6-18 802.16分组

同在802.3和802.11网络中一样），这个MAC更适合看作是WiMAX中的一个设备标识符，因为端点之间的通信最终被映射到一个连接标识符上（而不是连接的发送和接收端点的地址）。

我们简要学习了WiMAX的方式，还有许多我们不能在这里考虑的主题，如功率管理（类似于802.11中的睡眠模式）、切换、来自基站的MAC PDU传输的信道状态相关调度、QoS支持和安全性。由于802.16e标准还在研发中，WiMAX系统将在未来几年中继续演化。从上面列出的这些标准阅读这些和其他WiMAX主题可能相当“枯燥”，但[Eklund 2002, Cicconetti 2006]提供了可读性很强的WiMAX概述。

6.4 蜂窝因特网接入

在前一节中，我们研究了一台因特网主机当位于WiFi热区中时，即当它位于一个 802.11 接入点附近时，是如何接入因特网的。然而大多数WiFi热区的覆盖范围直径只有10～100 m，并且更广区域的WiMAX网络还在部署中。当我们十分需要接入因特网但同时又无法访问WiFi热区时，该怎么办呢？

鉴于蜂窝电话目前在全球许多区域已经是无处不在了，一个很自然的策略就是扩展蜂窝网络，使它们不仅支持语音电话，同时也支持无线因特网接入。理想情况下，这种因特网接入将会以一个相当高的速率，并且可以提供无缝的移动性，允许用户在旅行过程中（如在汽车或火车上）保持其TCP会话。使用足够高的上行和下行比特速率，用户甚至可以在移动中维持视频会议。这种情况并非遥不可及。在本书写作时（2007年春季），美国的许多蜂窝电话提供商以低于100美元/月的价格，为用户提供每秒几百千比特的上行和下行比特率的蜂窝因特网接入。随着诸如EVDO和HSDPA等宽带数据服务的实现，将很快可以提供每秒几百兆比特的数据速率。

在本节中，我们对当前和即将出现的蜂窝因特网接入技术进行简要概述。我们这里仍然是关注蜂窝电话和有线电话网络基础设施之间的无线第一跳；在6.7节中，我们将考虑呼叫如何选路到在不同基站间移动的用户。我们的讨论只是对蜂窝技术进行简单和宏观的描述。当然，现代蜂窝通信有更大的宽度和深度，有许多大学提供了关于这一主题的许多课程。希望对此做更深入了解的读者可参阅[Goodman 1997；Scourias 2007；Korhonen 2003；Kaaranen 2001；Lin 2001; Schiller 2003]，以及特别优秀和详尽的参考资料[Mouly 1992]。

历史事件

3G蜂窝移动与无线LAN的比较

许多蜂窝移动电话的运营者正在开发3G蜂窝移动系统，户内的数据速率为2 Mbps，户外的数据速率为384 kbps。3G系统在需要许可证的无线频带中部署，运营者向政府交纳可观的费用来获得许可证。3G系统将允许用户在活动中从遥远的户外接入因特网，使用和现在的蜂窝电话相似的接入方式。例如，3G技术将允许用户在开车的时候访问行车图信息，或者在海滩日光浴时访问电影院的信息。然而，考虑到它的费用和来自无线LAN的竞争，目前很多专家开始质疑3G技术是否会成功。特别是这些专家认为：

- 新兴的无线LAN基础设施将变得几乎无所不在。工作于54 Mbps的IEEE 802.11无线LAN已经得到了广泛部署。几乎所有便携计算机和PDA出厂时都配有802.11 LAN网卡。而且，新兴的因特网设备（例如无线照相机和相框）也将使用体积小、功

率低的无线LAN网卡。

- 我们在6.3.6节中学习的WiMAX期望为移动用户提供每秒几兆比特或更高的广域数据服务。Sprint Nextel近期宣称它将在未来几年中投资35亿美元来部署WiMAX。
- 无线LAN的基站也能处理移动电话装置。未来的电话也许能够连接蜂窝电话网络或使用类Skype语音经IP服务与IP网络相连，因此绕过运营者的蜂窝语音和3G数据服务。

当然，许多其他的专家相信3G不仅将取得巨大的成功，而且将使我们工作和生活的方式发生引人注目的革命。当然，WiFi和3G可能都会成为流行的无线技术，让漫游无线设备自动选择在其所处物理位置能提供最好服务的接入技术（参见本节中对4G无线接入的讨论）。

6.4.1 蜂窝网体系结构概述

术语蜂窝（cellular）指一个地理区域被分成许多称作发射区（cell）的地理覆盖区域，如图6-19左侧所示。每个发射区包括一个基站，负责向/从位于其发射区内的移动站点发送或接收信号。一个发射区的覆盖区域取决于许多因素，包括基站的发射功率、移动站点的传输功率、发射区中的障碍建筑物以及基站天线高度。尽管图6-19显示每个发射区包含一个位于其中心的基站，但当前许多系统将基站放置于3个发射区的交汇处，这样一个具有定向天线的基站就可以服务于3个发射区。

1. 基本的网络体系结构

如图6-19所示，每个基站通过一个有线基础设施连接到一个广域网中，如公共电话交换

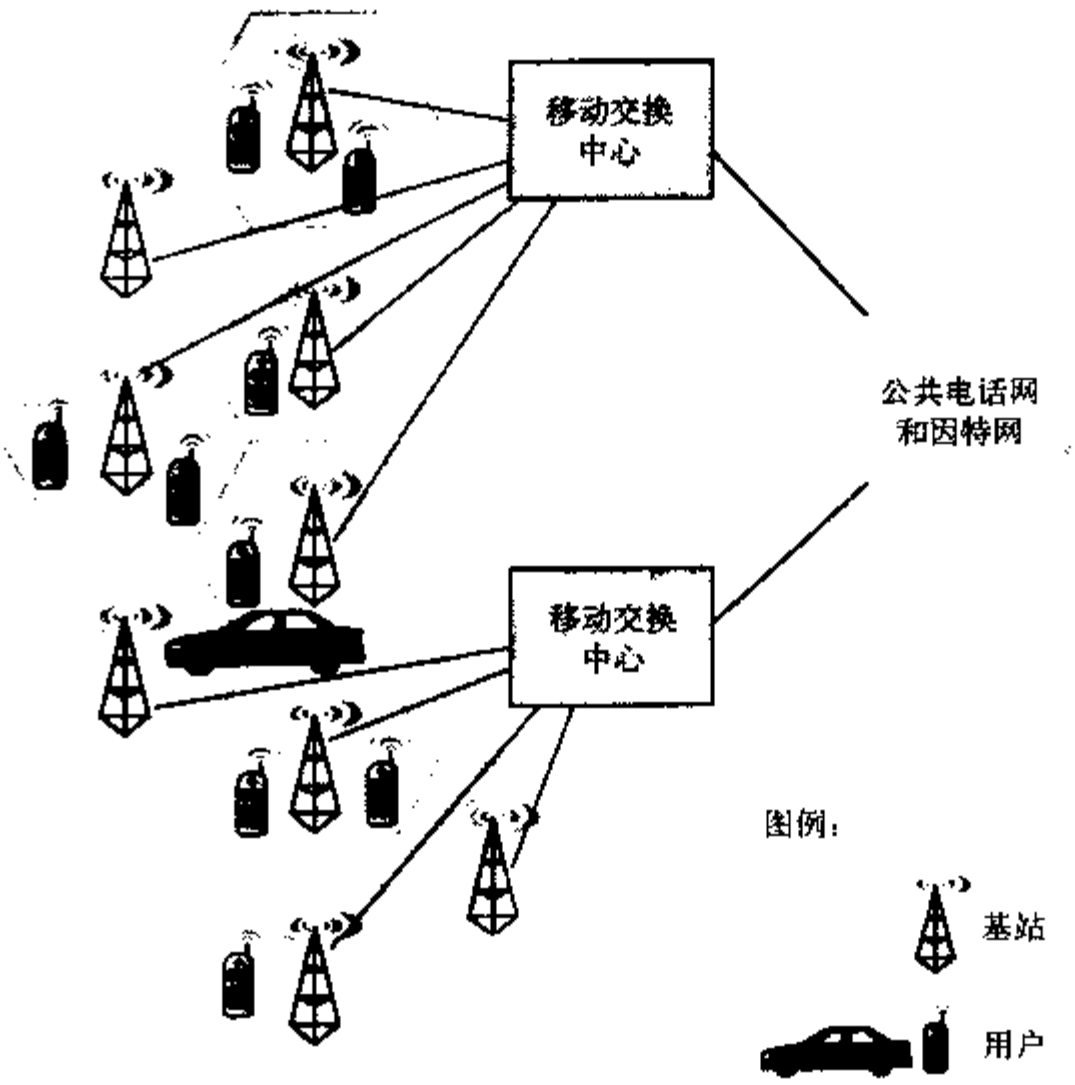


图6-19 蜂窝网络体系结构的构件

网（PSTN）或直接与因特网相连。特别地，图6-19显示每个基站连接到一个**移动交换中心**（Mobile Switching Center，MSC），它负责管理到和来自移动用户的呼叫的建立和拆除。MSC包含普通电话交换中心（如PBX或中心局）所具有的大多数功能，并增加了处理用户移动性所需的附加功能。

2. 空中接口接入技术

在一个给定的发射区中，通常会有许多同时的呼叫。这些呼叫需要共享分配给蜂窝服务提供商的无线电频谱的那个部分。大多数蜂窝系统当前使用如下两种广泛应用方法之一共享无线电频谱：

- *频分多路复用（FDM）和时分多路复用（TDM）的组合。*第1章讲过，在纯FDM中，信道被划分成许多频段，每个呼叫分配一个频段。第1章也讲过，在纯TDM中，时间被划分为帧，每个帧又被进一步划分为时隙，每个呼叫在轮回帧中被分配使用特定的时隙。在FDM/TDM组合的系统中，信道被划分为一组频率子带，在每个子带中，时间又被划分为帧和时隙。因此，对于一个FDM/TDM组合系统，如果某信道被划分为F个子带，并且时间被划分为T个时隙，那么该信道可支持$F \cdot T$个同时呼叫。
- *码分多址（CDMA）。*在6.2.1节中讲过，CDMA并不按时间或频率划分。相反，所有用户同时共享同样的无线电频率。发射区中的每个用户都分配一个独特的比特序列，称作码片序列。如我们在6.2.1节所见，当发送方和接收方使用同样的码片序列时，接收方可以从来自其他发送方的许多并行的发送中将该发送方的传输恢复出来。CDMA一个主要优点是它不必进行频率分配。当使用FDM/TDM系统时，接收方对来自同一频段的其他信号的干扰很敏感。因此，在FDM/TDM系统中，一个频率只有在那些相距足够远，能避免这种干扰的发射区中才能被重用。当设计CDMA时，这种**频率重用**（frequency reuse）不是主要考虑的问题。

6.4.2 蜂窝网标准和技术：简要回顾

当人们讨论蜂窝技术时，经常将其划归为属于某一代。较早的几代技术主要设计用于语音流量，最近的一些蜂窝系统除了语音外也支持因特网接入。由于本书是关于计算机网络而非语音电话的，我们当然对最近几代蜂窝系统更感兴趣。但由于最近几代直接由前几代演化而来，因而通过对第一、第二代无线系统做简要介绍来开始我们的回顾。

在展望这几代技术时，我们首先快速浏览一下多种技术的专业术语。作为读者，你不必强迫自己完全搞清或者记住所有的名词和缩略语。这种展望的目的是将多种无线技术展示出来，以及为许多名词和缩略语提供一个快速参考。

第一代（1G）系统是仅仅为语音通信设计的模拟FDMA系统。这些1G系统现在已基本消失，而被2G数字系统所取代。

1. 第二代（2G）

第二代系统尽管是数字的，但也被设计用于语音通信。由于当前设计用于处理数据通信的2.5G和3G系统源于2G，因此简要介绍2G是重要的。2G蜂窝电话首先将模拟信号转换为数字信号，然后调制该信号并向空中发送出去。已广泛部署了多种2G标准和技术，其中包括：

- 全球移动通信系统（Global System for Mobile Communication，GSM）。在20世纪80年代，欧洲认识到需要一个泛欧洲的数字系统来取代其互相不兼容的1G系统，提供在不同国家之间的无缝移动性以及模拟系统所不具有的其他特征和功能。这一需求催生了有

关蜂窝通信的GSM标准。欧洲在20世纪90年代早期十分成功地部署了GSM技术。然后GSM扩展到了亚洲和北美，现在已成为应用最广泛的蜂窝通信标准。2G蜂窝系统的GSM标准为空中接口使用了FDM/TDM组合。GSM系统由200 kHz频段组成，每个频段支持8个TDM呼叫。GSM以13 kbps和12.2 kbps速率进行语音编码。

- IS-95 CDMA。IS-95 CDMA使用码分多址（参见6.2.1节）。Qualcomm公司在20世纪80年代后期展示了CDMA用于蜂窝电话的生存能力，其后许多IS-95系统开始被部署，尤其是在北美和韩国。

2. 从第二代到第三代的过渡（2.5G）

诸如IS-95和GSM等2G系统是为语音服务而优化的，并不是特别适合数据通信。在20世纪90年代，标准化组织认识到对同时适用于语音和数据通信（包括因特网接入）的3G蜂窝技术的需求。然而，由于广泛部署3G技术需要很长时间，一些公司开发了可以在现有2G基础设施上进行数据传输的中间协议和标准。这些系统被统称为“2.5G蜂窝系统”。它们包括：

- 通用分组无线服务（General Packet Radio Service， GPRS）。GPRS由GSM演化而来。对于数据服务，GSM有效地模拟了在用户设备和目的数据网络之间的调制解调器。如我们在第1章所知，电路交换对突发性的数据而言非常低效。此外，标准GSM仅支持最大9.6 kbps的数据率，这对除简单文字而言的任何数据而言都显得太低了。GPRS是一个提供更加高效、基于分组的数据服务的中间方案，它具有更高的数据率（通常是40～60 kbps），在许多方面与家庭电话能被用于将家庭PC经拨号连接与因特网连接相同。因此，GSM使用电路交换用于标准话音和数据（将用户设备与因特网相连）。GPRS服务由底层的GSM网络提供。然而，与普通的GSM不同，移动GPRS站点可以按需使用一个信道中的多个时隙。在GPRS中，留出许多时隙用于数据通信，并按照移动站点的即时请求进行动态分配。
- 支持全球演化的增强数据速率（Enhanced Data Rates for Global Evolution，EDGE）。EDGE的主要目的是增加GSM/GPRS网络的数据率，即更好地使用具有8个时隙TDMA帧的GSM 200 kHz的信道。这主要通过一个更加有效的机制替代GSM的调制机制。在理论上，EDGE可以为用户数据通信提供384 kbps的数据率。关于EDGE很好的概述是[Ericsson 2007]。
- CDMA 2000，Phase 1。这种2.5G技术是由IS-95演化而来的。它可以以高达144.4 kbps的速率提供分组数据服务，并为CDMA 2000，Phase 2的3G的部署奠定了基础。

3. 第三代（3G）

3G蜂窝系统要求以比其相应的2G系统高得多的速率提供电话服务以及数据通信。特别是3G系统要求提供：

- 驾车行驶时速率达144 kbps。
- 室外静止或行走时速率达384 kbps。
- 室内速率达2 Mbps。

在3G领域有两个主要（并且是相互竞争）的标准：

- CDMA-2000。CDMA-2000由IS-95 2G系统演化而来，并向后兼容IS-95。从其名字可以推测，它同样也使用CDMA作为其空中接口的部分。CDMA-2000正在北美和亚洲部分地区部署。与CDMA-2000关联的数据服务称为1xEVDO（Evolution Data Optimized——演化数据优化，这可能是最糟糕的首字母缩写词之一！），期望数据率达3 Mbps量级。

我们在前面的WiMAX标准中遇到的机会主义调度技术实际上是以前为EVDO和它的CDMA-2000-HDR（高数据率）前任而研发的[Bender 2000]。

- 通用移动电信（Universal Mobile Telecommunications Service，UMTS）。UMTS是GSM为支持3G能力进行的演化。UMTS网络体系结构借鉴了很多已经建立的GSM网络体系结构。然而，UMTS的无线电接入与GSM中使用的FDMA/TDMA方案有很大的不同。具体地说，UMTS使用一种称作直接序列宽带CDMA（DS-WCDMA）的CDMA技术（TDMA时隙的帧在多个频率上可用，这是一种我们前面指出的所有三种专门的信道共享方法的有趣使用）。由于UMTS源于GSM，故欧洲正在广泛部署UMTS也没有什么值得惊奇的。与WCDMA规范相关的数据服务称为HSDPA/HSUPA（高速上行/下行分组接入），期望数据率达到14 Mbps。与EVDO一样，HSDPA/HSUPA使用机会主义调度和其他一些先进的技术。

4. 第四代（4G）

既然我们已经分析了无线LAN技术和全部的蜂窝接入技术，那么我们退一步思考一下，作为用户，我们理想中的无线因特网接入将是怎样的。下面列出了一些愿望：

- 我们希望无处不在的无线因特网接入。不论在家、办公室、车上、咖啡屋，还是在海滩上，我们希望都能够接入因特网。
- 接入因特网速率是所处物理位置和移动速度的函数，我们希望能够以最高可能的速率接入因特网。例如，如果我们位于某个街角，该处同时提供11 Mbps的802.11b、54 Mbps的802.11g和384 kbps的3G接入，我们希望系统能自动选择802.11g，该系统在此时此地能提供最高接入速率。
- 当穿越异构环境漫游时，我们能够自动、透明地从一种接入技术切换到另一种接入技术（如从802.11到3G），这仅仅取决于可用性，不需要任何用户的干预。
- 当然，在移动过程中，我们希望保持正在进行的TCP连接。更进一步，我们希望系统知道我们在何处，从而在移动时新的呼叫能够继续到达我们。
- 我们希望系统在IP上支持语音和实时视频，从而不论在何处，我们都可以戴着Dick Tracy表与朋友和同事开视频会议。

当然，我们希望这一切都是免费的（或者更实际地讲，至少是花费低）。尽管这听起来像是一个无线天堂，但好在是大部分关键技术组件已经可用了，现在真正的问题是用一种综合协议和技术将上述愿望变为现实。这些组件包括像802.11和3G这样的接入技术、移动管理技术（参见6.5节、6.6节、6.7节）、加密和鉴别协议（见第8章），以及多媒体网络技术（如用于基于IP的语音的SIP，见第7章的讨论）。

6.5 移动管理：原理

论述完无线网络中通信链路的*无线特性*后，现在我们转向这些无线链路带来的*移动性*。宽泛地讲，移动节点是随时间改变它与网络连接位置的节点。因为*移动性*这一术语在计算机和电话界有许多含义，所以首先详细讨论一下移动性的各个方面将对我们很有帮助。

- *从网络层的观点来看，用户如何移动？*一个物理上移动的用户将对网络层提出一系列不同寻常的挑战，这取决于他（她）在网络连接点之间如何移动。在图6-20所示的移动程度谱的一端，用户也许带着一台装有无线网络接口卡的便携机在一座建筑物内走动。如我们在6.3.4节中所见，从网络层的观点来看，该用户并没有移动。更重要的是，如果

该用户不论在何处都与同一个接入点相关联，那么从链路层观点来看该用户甚至也没有移动。

在该移动程度谱的另一端，考虑一下该用户在一辆“宝马”轿车内以150 km的时速沿高速公路急速行驶的情况，用户穿过多个无线接入网，并希望在整个旅程中保持一个与远程应用不间断的TCP连接。这个用户无疑是移动的！在这两种极端之间的情况是，一个用户带着一台便携机从一个地方（如办公室或宿舍）到另一个地方（如咖啡店、教室），并且想在新地方连入网络。该用户也是移动的（虽然比“宝马”驾驶员的移动性差一些！），但不需要在网络接入点之间移动时维持一个不间断的连接。图6-20从网络层观点举例说明了用户移动性程度谱。

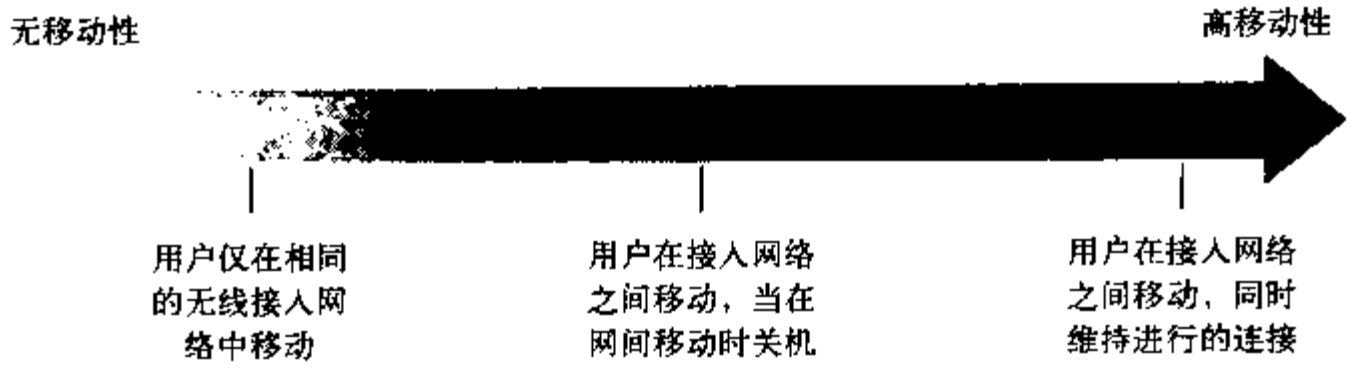


图6-20 从网络层观点来看移动性

- *移动节点的地址始终保持不变有多么重要？*对移动电话而言，当你从一个提供商移动电话网络到另一个的过程中，电话号码（本质上是你电话的网络层地址）始终保持不变。便携机在IP网络之间移动时是否也必须维持相同的IP地址呢？

对这一问题的回答很大程度上取决于所运行的应用程序。对于那个在高速公路上飞驰、同时又希望维持对一个远程应用不间断的TCP连接的宝马司机而言，维持相同的IP地址将会带来便利。回想第3章，一个因特网应用程序需要知道它与之通信的远端实体的IP地址和端口号。如果一个移动实体在移动过程中能够保持其IP地址不变，从应用的角度，移动性就变得不可见。这种透明性有十分重要的价值，应用程序不必关心IP地址潜在的变化，同样的应用程序代码既可用于移动连接，又可用于非移动连接。在下一节我们将会看到的移动IP提供了这种透明性，它允许移动节点在网络间移动的同时维持其永久的IP地址。

另一方面，一个不太喜欢新潮的移动用户也许只想关闭办公室便携机，将其带回家，然后开机，再在家中工作。如果该便携机在家时只是作为一个客户机，使用客户机/服务器模式的应用（如发送/阅读电子邮件、浏览Web、通过Telnet与远程主机相连），则它使用特定IP地址并不那么重要。特别是，用户能够得到一个由服务于家庭的ISP临时分配的IP地址即可。在5.4节我们看到DHCP已经提供这种功能。

- *有哪些可用的有线基础设施的支持？*在所有上述情形中，我们都隐含地假设存在一个固定的基础设施让移动用户连接，例如家庭的ISP网络、办公室的无线接入网络，或者沿高速公路的无线接入网络。如果这样的基础设施不存在会怎么样？如果两个用户位于彼此的通信范围内，他们能否在没有其他网络基础设施存在的情况下建立一个网络连接？自组织网络正好提供了这些功能。这一飞速发展的领域位于移动网络研究的前沿，超出了本书的范围。[Perkins 2000]和IETF移动自组织网络（manet）工作组主页[manet 2007]提供了有关这一主题的详尽讨论。

为了说明允许移动用户在不同网络间移动过程中维持正在进行的连接所涉及的问题，我们考虑一个人类的类比例子。一位20左右的成人从家里搬出，成为流动的人，在宿舍或公寓居住，并经常改换住址。如果一个老朋友想与他联系，这位朋友怎样才能找到这个流动的人呢？一种常用的方法是与他的家庭联系，因为一位流动的青年通常会将其目前的地址告诉家里。其家庭由于有一个永久地址，因此成为其他想与该流动青年联系的人的第一个去处。这些朋友后来与他的通信也许是间接的（如先将邮件发送到其父母家，再转发给该流动青年），也许是直接的（如该朋友用得到的地址直接将邮件发送给流动青年）。

在一个网络环境中，一个移动节点（如一台便携机或一个PDA）的固定“居所”称为**归属网络**（home network），在归属网络中代表移动节点执行移动管理功能的实体叫**归属代理**（home agent）。移动节点当前所在网络叫做**外部（或被访）网络**（foreign or visited network），在外部网络中帮助移动节点完成移动管理功能的实体称为**外部代理**（foreign agent）。对于移动的专业人员而言，他们的归属网络可能就是其公司网，而被访网络也许就是他们正访问的某同行所在的网络。一个**通信者**（correspondent）就是希望与该移动节点通信的实体。图6-21举例说明了这些概念，同时也说明了下面要考虑的编址概念。在图6-21中，我们注意到代理被配置在路由器上（例如，作为在路由器上运行的进程），但它们也能在网络中其他主机或服务器上执行。

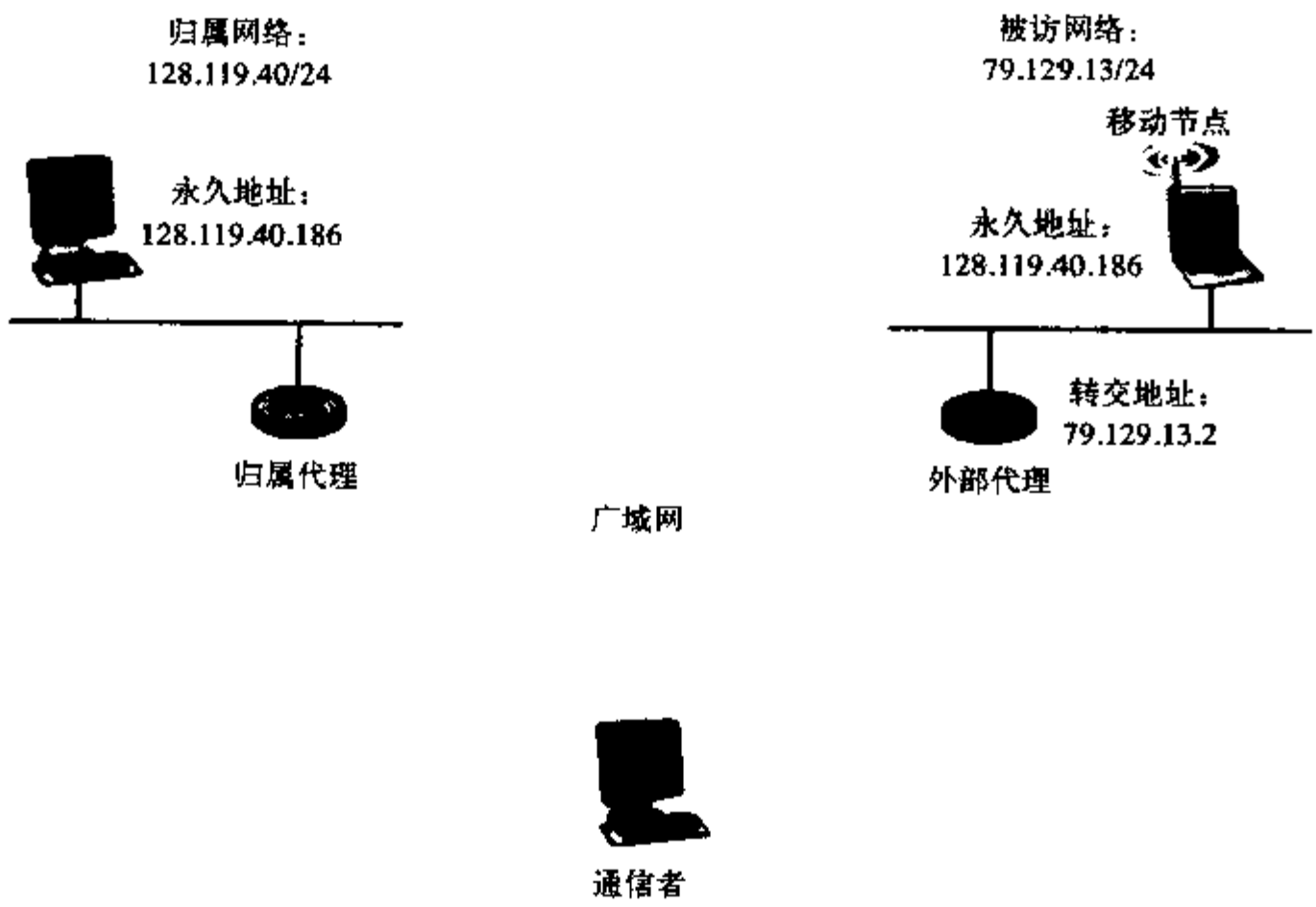


图6-21 移动网络体系结构中的初始元素

6.5.1 寻址

前面讲过，为了使用户移动性对网络应用透明，要求一个移动节点在从一个网络移动到另一个网络时保持其地址不变。当一个移动节点位于一个外部网络时，所有指向此节点固定地址的流量需要导向外部网络。怎样才能做到这一点呢？外部网络可用的一种方法就是向所有其他网络发通告，告诉它们该移动节点正在它的网络中。这通常可通过交换域内和域间选路信息来实现，而且只需对现有选路基础结构做很少的改动即可。外部网络只需通告其邻居

它有一条非常特别的路由能到达该移动节点的固定地址，即告诉其他网络它有一条正确的路径可将数据报导向该移动节点的固定地址（即基本上是通知其他网络，它有一条用于数据报选路到该移动节点的永久地址的正确路径；参见4.4节）。这些邻居将在全网传播该选路信息，而且是当作更新选路信息和转发表的正常过程的一部分工作来做。当移动节点离开一个外部网络后又加入另一个外部网络时，新的外部网络会通告一条新的通向该移动节点的特别路由，旧的外部网络将撤销其与该移动节点有关的选路信息。

这种方法立刻解决了两个问题，且它这样做不需对网络层基础结构做重大改动。其他网络知道该移动节点的位置，很容易将数据报路由到该移动节点，因为转发表将这些数据报导向外部网络。然而它有一个很大的缺陷，即扩展性不好。如果移动性管理是网络路由器的责任，则路由器将必须维护可能多达数百万个移动节点的转发表表项。一些其他的缺陷将在本章后面的习题中进行探讨。

一种替代方法（并在实际中得到了采用）是将移动性功能从网络核心搬到网络边缘，这是我们在研究因特网体系结构时一再重复的主题。一种自然的做法是由该移动节点的归属网络来实现。与那个流动青年的父母跟踪孩子的位置有许多相似之处，在移动节点的归属网络中的归属代理也能跟踪该移动节点所在的外部网络。这当然需要一个在移动节点（或一个代表该移动节点的外部代理）与归属代理之间的协议来更新移动节点的位置。

我们现在来更详细地思考外部代理。如图6-21所示，概念上最简单的方法是将外部代理放置在外部网络的边缘路由器上。外部代理的作用之一就是为移动节点创建一个所谓**转交地址**（care-of address，COA），该COA的网络部分与外部网络相匹配。因此有一个移动节点与两个地址相关联，其**永久地址**（permanent address）（类似于流动青年的家庭地址）与其COA，COA有时又称为**外部地址**（foreign address）（类似于流动青年当前居住的房屋地址）。在图6-21所示的例子中，移动节点的固定地址是128.119.40.186。当访问网络79.129.13/24时，该移动节点的COA为79.129.13.2。外部代理的第二个作用就是告诉归属代理，该移动节点在它的（外部代理的）网络中且有给定的COA。我们很快就会看到，该COA用于将数据报通过外部代理“重新选路”到移动节点。

虽然我们已将移动节点与外部代理的功能分开，但是值得注意的是，移动节点也能承担外部代理的责任。例如，某移动节点可在外部网络中得到一个COA（如使用一个诸如DHCP之类的协议），且由它自己把它的COA通告给归属代理。

6.5.2 选路到移动节点

我们现在已看到一个移动节点是如何得到一个COA的，又是如何告之归属代理该地址的。但让归属代理知道该COA仅能解决部分问题。数据报应怎样寻址并转发给移动节点呢？因为只有归属代理（而不是全网的路由器）知道该移动节点的位置，故如果只是将一个数据报寻址到移动节点的永久地址并将其发送到网络层基础结构中，这样做已不再满足需要了。还有更多的事情要去做。目前有两种不同的方法，我们将称其为间接选路与直接选路。

1. 移动节点的间接选路

我们先考虑一个想给移动节点发送数据报的通信者。在**间接选路**（indirect routing）方法中，通信者只是将数据报寻址到移动节点的固定地址，并将数据报发送到网络中去，完全不知道移动节点是在归属网络中还是正在访问某个外部网络。因此移动性对于通信者来说是完全透明的。这些数据报就像平常一样首先导向移动节点的归属网络。这可以通过图6-22所示

例子中的步骤1加以说明。

我们现在将注意力转向归属代理。除了负责与外部代理交互以跟踪移动节点的COA外，归属代理还有另一项很重要的功能。其第二项工作就是监视寻址到某些节点的到达数据报，这些节点的归属网络就是该归属代理所在的那个网，但这些节点当前却在某个外部网络中。归属代理截获这些数据报，然后按两个步骤将其“重新选路”到某个移动节点。通过使用移动节点的COA，数据报先转发给外部代理（图6-22中的步骤2），然后再从该外部代理转发给移动节点（图6-22中的步骤3）。

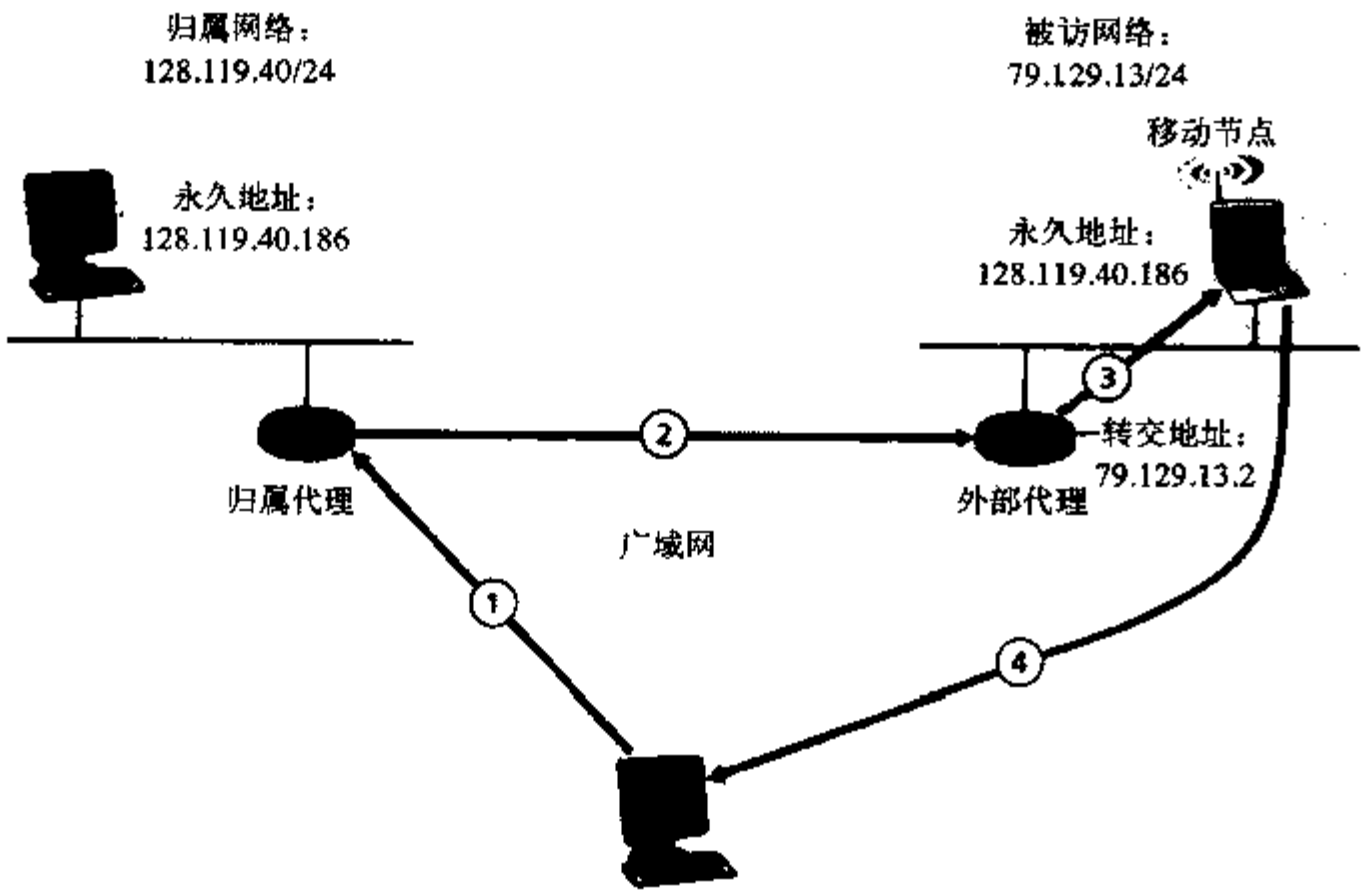


图6-22 间接转发给一个移动节点

仔细思考这种重新选路过程是有益的。归属代理需要用该移动节点的COA来设置数据报地址，以便网络层将该数据报选路到外部网络。而另一方面，需要保持通信者数据报的原样，因为接收该数据报的应用程序应该不知道该数据报是经由归属代理转发而来的。这两个目标都可以得到满足，让归属代理将通信者的原始完整数据报**封装**（encapsulate）在一个新的（较大的）数据报中即可。这个较大的数据报被导向并交付到移动节点的COA。“拥有”该COA的外部代理将接收并拆封该数据报，即从较大的封装数据报中取出通信者的原始数据报，然后再向该移动节点转发原始数据报（图6-22中的步骤3）。图6-23显示了如下过程，一个通信者向归属网络发送原始数据报；向外部代理发送一个封装的数据报；以及向移动节点交付最初的数据报。思维敏锐的读者将会注意到，这里描述的封装/拆封概念等同于在第4章中讲述IP多播与IPv6时所讨论的隧道的概念。

接下来我们考虑一个移动节点如何向一个通信者发送数据报。这很简单，因为移动节点可直接将其数据报寻址到通信者（使用自己的固定地址作为源地址，通信者的地址作为目标地址）。因为移动节点知道通信者的地址，所以不需要通过归属代理迂回传送数据报。这就是显示在图6-22中的步骤4。

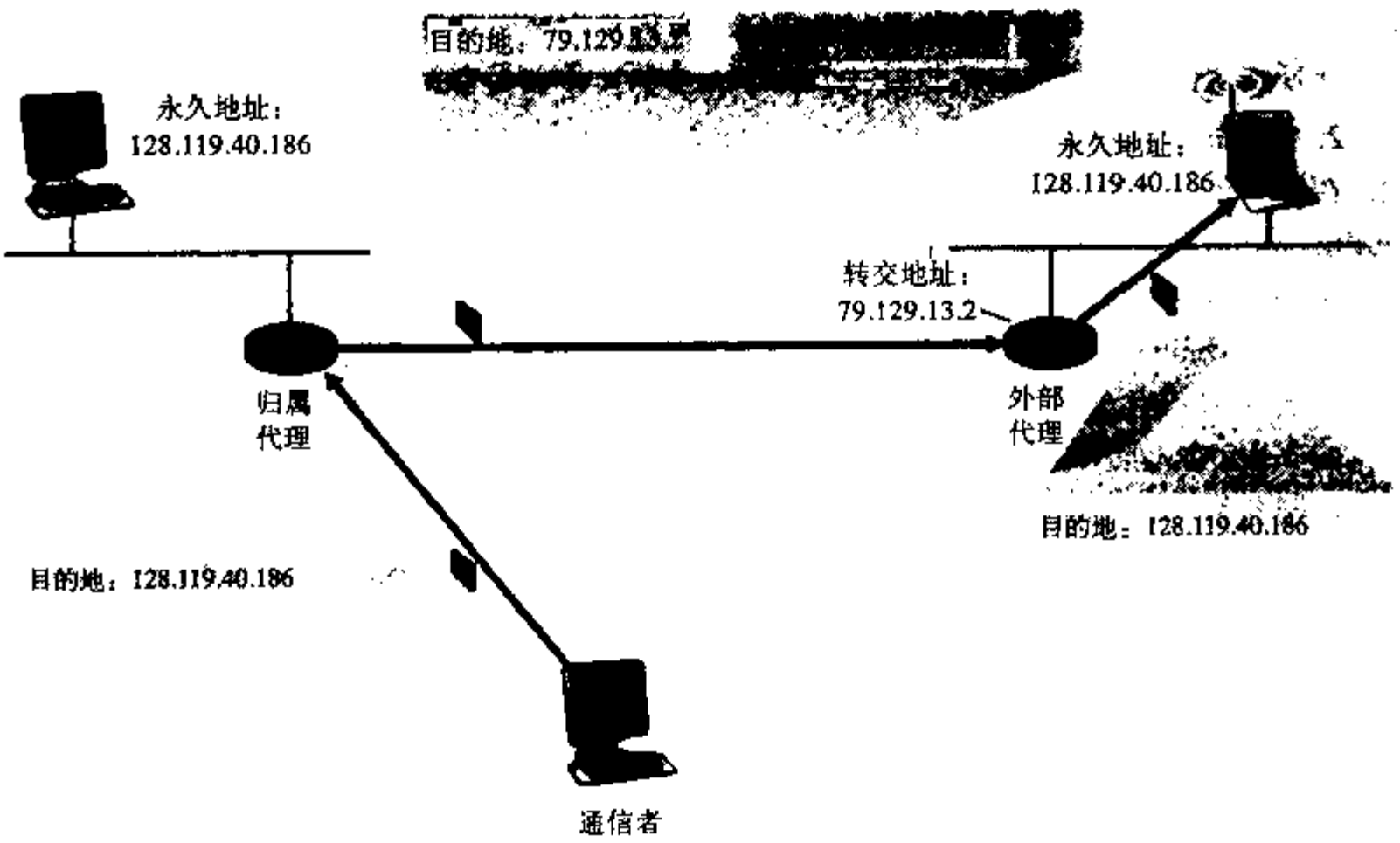


图6-23 封装与拆封

下面我们列出支持移动性所需的网络层新功能，来对有关间接选路的讨论进行小结。

- 移动节点到外部代理的协议。当移动节点与外部网络连接时，它向外部代理注册。类似地，当一个移动节点离开该外部网络时，它将向外部代理取消注册。
- 外部代理到归属代理的注册协议。外部代理将向归属代理注册移动节点的COA。当一个移动节点离开其网络时，外部代理不需要显式地注销COA，因为当移动节点移动到一个新网络时，要注册一个新的COA，而取消以前的COA。
- 归属代理数据报封装协议。将通信者的原始数据报封装在一个目的地址为COA的数据报内，并转发之。
- 外部代理拆封协议。从封装好的数据报中取出通信者的原始数据报，然后再将该原始数据报转发给移动节点。

上述讨论提供了一个移动节点在网络之间移动时要维持一个不间断的连接所需的各部分：外部代理、归属代理和间接转发。举一个例子说明各部分如何协同工作，假设某移动节点连到外部网络A，向其归属代理注册了网络A中的一个COA，且正在接收通过归属代理间接路由而来的数据报。该移动节点现在移动到外部网络B中，并向网络B中的外部代理注册，外部代理将该移动节点的新COA告诉其归属代理。此后，归属代理将重新将数据报导向网络B。就一个通信者关心的东西而言，移动性是透明的，即在移动前后，数据报都是由相同的归属代理选路。就归属代理关心的东西而言，数据报流没有中断，即到达的数据报先是转发到外部网络A，改变COA后，则数据报转发到外部网络B。但是，当移动节点在网络之间移动时，它会看到数据报流中断吗？只要移动节点与网络A断开连接（此时它不能再经A接收数据报）再连接到网络B（此时它将向归属代理注册一个新的COA）用的时间很少，那么几乎没有数据报丢失。第3章讲过，端到端连接可能会由于网络拥塞而丢失数据报。因而当一个节点在网络之间移动时，一条连接中的数据报偶尔丢失算不上什么灾难性问题。如果需要没有丢失的通信，

上层机制将对数据报丢失进行恢复，不管这种丢失是因网络拥塞还是因用户移动而引发的。

在移动IP标准中使用了一种间接选路方法［RFC 3344］，这将在6.6节中讨论。

2. *移动节点的直接选路*

在图6-22中说明了间接选路方法存在一个低效的问题，称为**三角选路问题**（triangle routing problem）。该问题是指即使在通信者与移动节点之间存在一条更有效的路由，发往移动节点的数据报也先要发给归属代理，然后再发送到外部网络。在最坏情况下，设想一个移动用户正在访问一位同行所在的外部网络，两人并排坐在一起且正在通过网络交换数据，从通信者（在该例中为该访问者的同行）处发出的数据报被选路到该移动用户的归属代理，然后再回到该外部网络！

直接选路（direct routing）克服了三角选路的低效问题，但却是以增加复杂性为代价的。在直接选路方法中，通信者所在网络中的一个**通信者代理**（correspondent agent）先知道该移动节点的COA。这可以通过让通信者代理向归属代理询问得知，这里假设与间接选路情况类似，移动节点具有一个在归属代理注册过的最新的COA。与移动节点可以执行外部代理的功能相类似，通信者本身也可能执行通信者代理的功能。参见图6-24的步骤1和2。通信者代理然后将数据报直接通过隧道技术发往移动节点的COA，这与归属代理使用的隧道技术相类似，参见图6-24的步骤3和4。

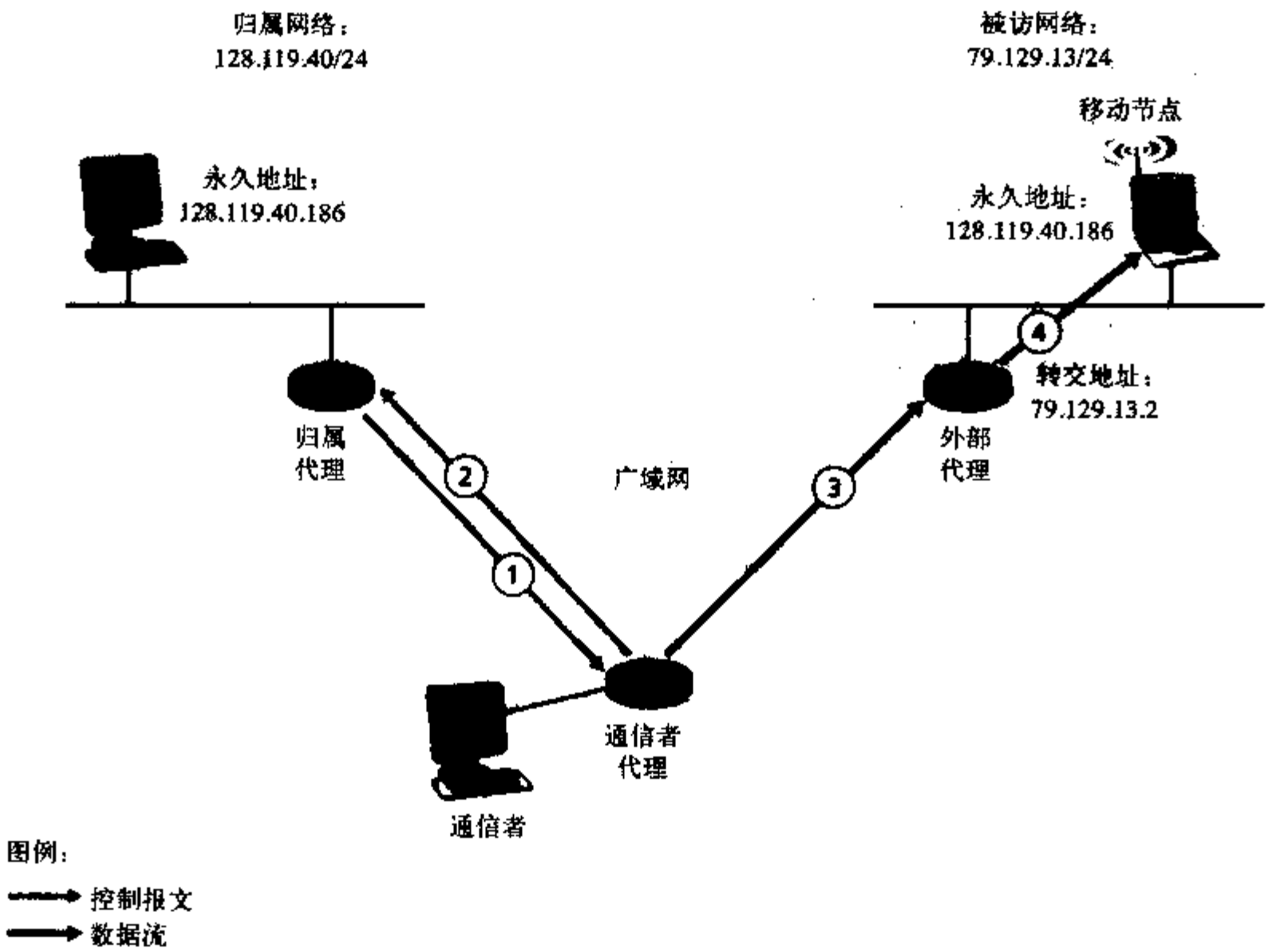


图6-24 到一个移动用户的直接选路

尽管直接选路克服了三角选路问题，但它引入了两个重要的其他挑战：

- **需要一个移动用户定位协议**（mobile-user location protocol），以便通信者代理向归属代

理查询获得移动节点的COA（图6-24中的步骤1和2）。

- 当移动节点从一个外部网络移到另一个外部网络时，如何将数据报转发到新的外部网络？在间接选路情况下，这个问题可以容易地通过更新归属代理维持的COA解决。然而，使用直接选路时，归属代理仅在会话开始时被通信者代理询问一次。因此，当必要时在归属代理中更新COA，这并不能解决将数据选路到移动节点新的外部网络的问题。

一种解决方案是，创建一个新的协议来告知通信者变化后的COA。另一种方案也是在GSM网络实践中所采用的方案，它的工作方式如下：假设数据当前正转发给位于某个外部网络中的移动节点，并且在会话刚开始时该移动节点就位于该网络中（图6-25中步骤1），我们标识移动节点首次被发现的外部网络中的外部代理为锚外部代理（anchor foreign agent）。当移动节点到达一个新外部网络后（图6-25中步骤2），移动节点向新的外部代理注册（步骤3），并且新外部代理向锚外部代理提供移动节点新的COA（步骤4）。当锚外部代理收到一个发往已经离开的移动节点的封装数据报后，它可以使用新的COA重新封装数据报并将其转发给该移动节点（步骤5）。如果移动节点其后又移到另一个外部网络中，在该被访网络中的外部代理随后将与锚外部代理联系，以便建立到该新外部网络的转发。

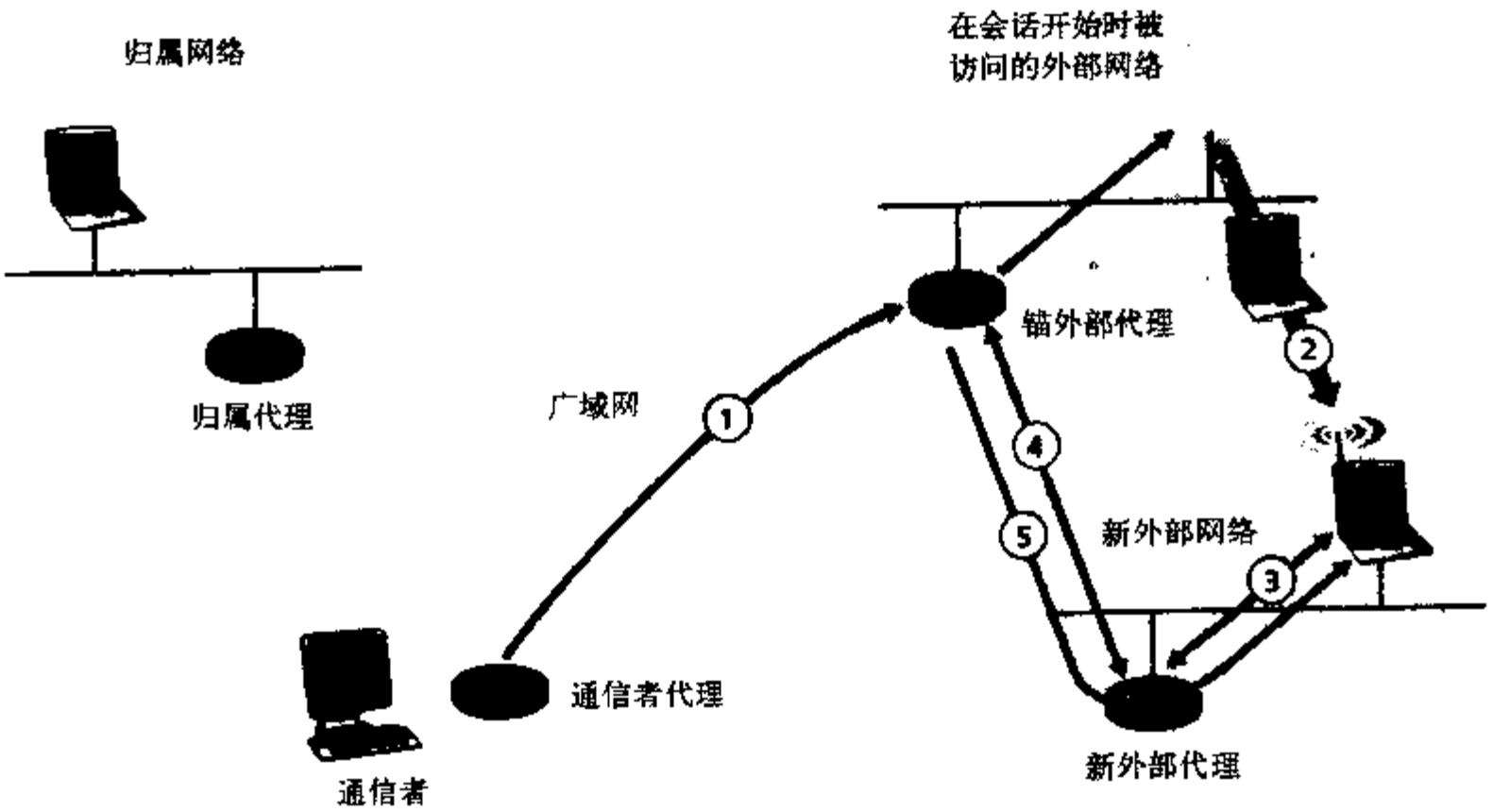


图6-25 用直接选路在网络间进行移动转移

6.6 移动IP

支持移动性的因特网体系结构与协议合起来称为移动IP，它主要由RFC 3344定义。移动IP是一个灵活的标准，支持许多不同的运行模式。例如，具有或不具有外部代理的模式，代理与移动节点相互发现的多种方法，使用单个或多个COA，以及多种形式的封装。同样地，移动IP是一个复杂的标准，需要用整本书才能详细描述；的确有这样一本书，即[Perkins 1998b]。这里，我们最基本的目标是对移动IP最重要的部分进行概述，并说明它在一些常见情形中的使用。

移动IP体系结构包含了我们前面已考虑过的许多元素，包括归属代理、外部代理、转交

地址和封装/拆封等概念。当前的标准［RFC 3344］规定，到移动节点使用间接选路的方法。

移动标准由三部分组成：

- 代理发现。移动IP定义了一个归属代理或外部代理向移动节点通告其服务所使用的协议，以及移动节点请求一个外部代理或归属代理的服务所使用的协议。
- 向归属代理注册。移动IP定义了移动节点和/或外部代理向一个移动节点的归属代理注册或注销COA所使用的协议。
- 数据报间接选路。该标准也定义了数据报被一个归属代理转发给移动节点的方式，包括转发数据报使用的规则、处理差错情况的规则和几种不同的封装形式［RFC 2003; RFC 2004］。

在移动IP标准中，安全性是很重要的。例如，显然需要对一个移动节点进行鉴别以确保恶意用户不能向归属代理注册一个伪造的转交地址，伪造地址将导致所有发给某个IP地址的数据报被重定向到恶意用户。移动IP使用许多机制来实现安全性，我们将在第8章考察这些机制，所以在以下的讨论中我们将不考虑安全性问题。

1. 代理发现

到达一个新网络的某移动IP节点，不管是连到一个外部网络还是返回其归属网络，它都必须知道相应的外部代理或归属代理的身份。实际上，正是由于一个新外部代理的发现，得到一个新的网络地址，才使移动节点中的网络层知道它已进入一个新的外部网络。这个过程又被称为**代理发现**（agent discovery）。代理发现可以通过下列两种方法之一实现：经代理通告或者经代理请求。

借助于**代理通告**（agent advertisement），外部代理或归属代理使用一种现有路由器发现协议的扩展协议［RFC 1256］来通告其服务。代理周期性地在所有连接的链路上广播一个类型字段为9（路由器发现）的ICMP报文。路由器发现报文也包含路由器（即该代理）的IP地址，因此可以使一个移动节点知道该代理的IP地址。路由器发现报文还包括了一个移动性代理通告扩展，其中包含了该移动节点所需的附加信息。在该扩展中的重要字段有：

- 归属代理比特（H）。指出该代理是它所在网络的一个归属代理。
- 外部代理比特（F）。指出该代理是它所在网络的一个外部代理。
- 注册要求比特（R）。指出在该网络中的某个移动用户必须向某个外部代理注册。特别是，一个移动用户不能在外部网络（如使用DHCP）中获得一个转交地址，并假定由它自己承担外部代理的功能，无需向外部代理注册。
- M、G封装比特。指出除了IP-in-IP的封装形式外，是否还要用其他的封装形式。
- 转交地址（COA）字段。由外部代理提供的一个或多个转交地址的列表。在下面的例子中，COA将与外部代理有关，外部代理将接收发给该COA的数据报，然后再转发到适当的移动节点。移动用户在向其归属代理注册时将选择这些地址中的一个作为其COA。

图6-26说明了在代理通告报文中的某些关键字段。

使用**代理请求**（agent solicitation），想知道代理的移动节点不必等待接收代理通告，就能广播一个代理请求报文，该报文只是一个类型值为10的ICMP报文。收到该请求的代理将直接向该移动节点单播一个代理通告，于是该移动节点将继续处理，就好像刚收到一个未经请求的通告一样。

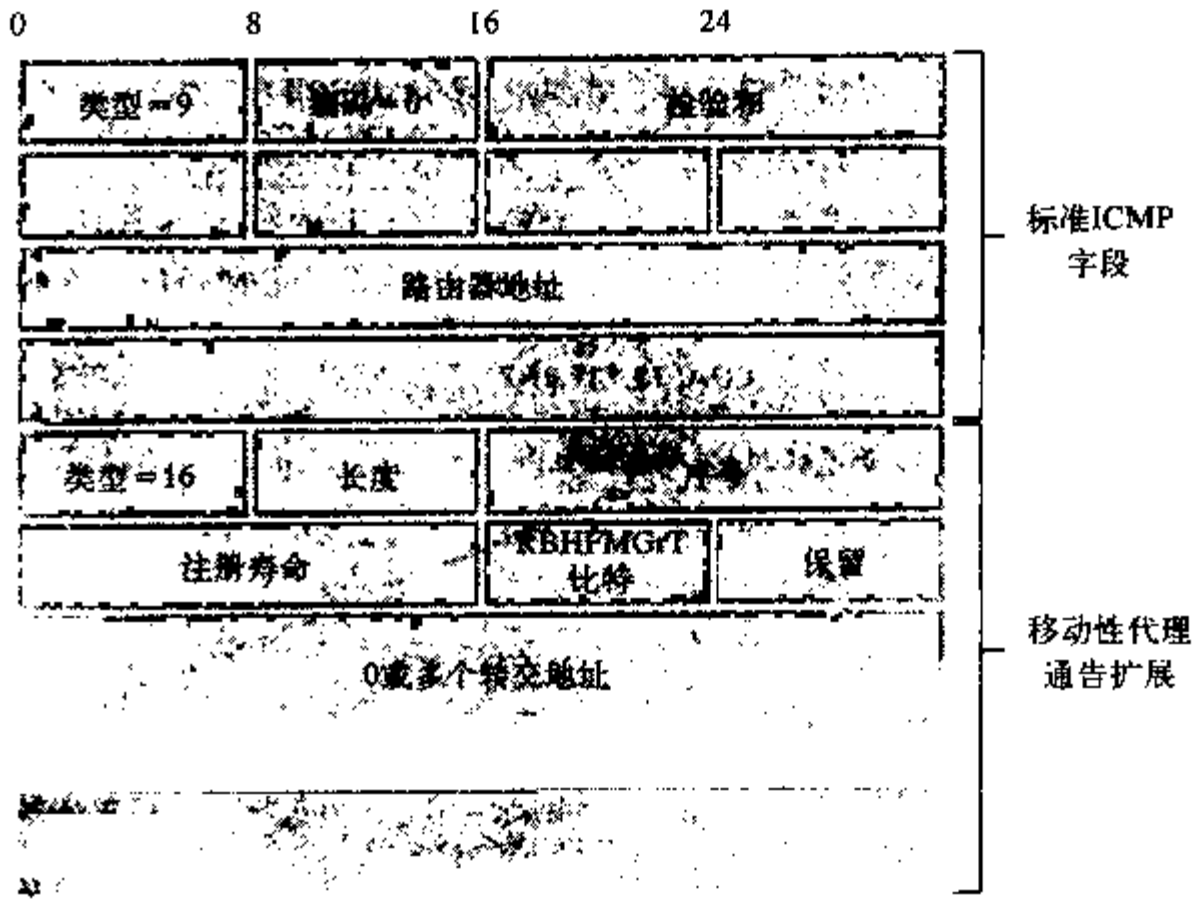


图6-26 具有移动性代理通告扩展的ICMP路由器发现报文

2. 向归属代理注册

一旦某个移动IP节点收到一个COA，则该地址必须要向归属代理注册。这可通过外部代理（由它向归属代理注册该COA）或直接通过移动IP节点自己来完成。我们下面考虑前一种情况，这个过程共涉及4个步骤：

1）收到一个外部代理通告以后，移动节点立即向外部代理发送一个移动IP注册报文。注册报文承载在一个UDP数据报中并通过端口434发送。注册报文携带有一个由外部代理通告的COA、归属代理的地址（HA）、移动节点的永久地址（MA）、请求的注册寿命和一个64比特的注册标识。请求的注册寿命指示了注册有效的秒数。如果注册没有在规定的时间内在归属代理上更新（延长期限），则该注册将变得无效。注册标识就像一个序列号，用于收到的注册回答与注册请求的匹配，这是下面要讨论的内容。

2）外部代理收到注册报文并记录下移动节点的永久IP地址。外部代理知道现在它应该查找这样的数据报，即它封装的数据报的目的地址与该移动节点的固定地址相匹配。外部代理然后向归属代理的434端口发送一个移动IP注册报文（同样封装在UDP数据报中）。这一报文包括COA、HA、MA、封装格式要求、请求的注册寿命以及注册标识符。

3）归属代理接收注册请求并检查真伪和正确性。归属代理把移动节点的永久IP地址与COA绑定在一起。以后，到达该归属代理的数据报与发往移动节点的数据报将被封装并以隧道方式给COA。归属代理发送一个移动IP注册回答，该响应报文中包含有HA、MA、实际注册寿命和被认可的请求报文注册标识。

4）外部代理接收注册响应，然后将其转发给移动节点。

到此，注册便完成了，移动节点就能接收发送到其永久地址的数据报。图6-27说明了这些步骤。注意到归属代理指定的寿命比移动节点请求的寿命要小。

当某个移动节点离开其网络时，外部代理无需显式地取消某个COA的注册。当移动节点移动到一个新网（不管是另一个外部网络还是其归属网络）并注册一个新COA时，上述情况

将自动发生。

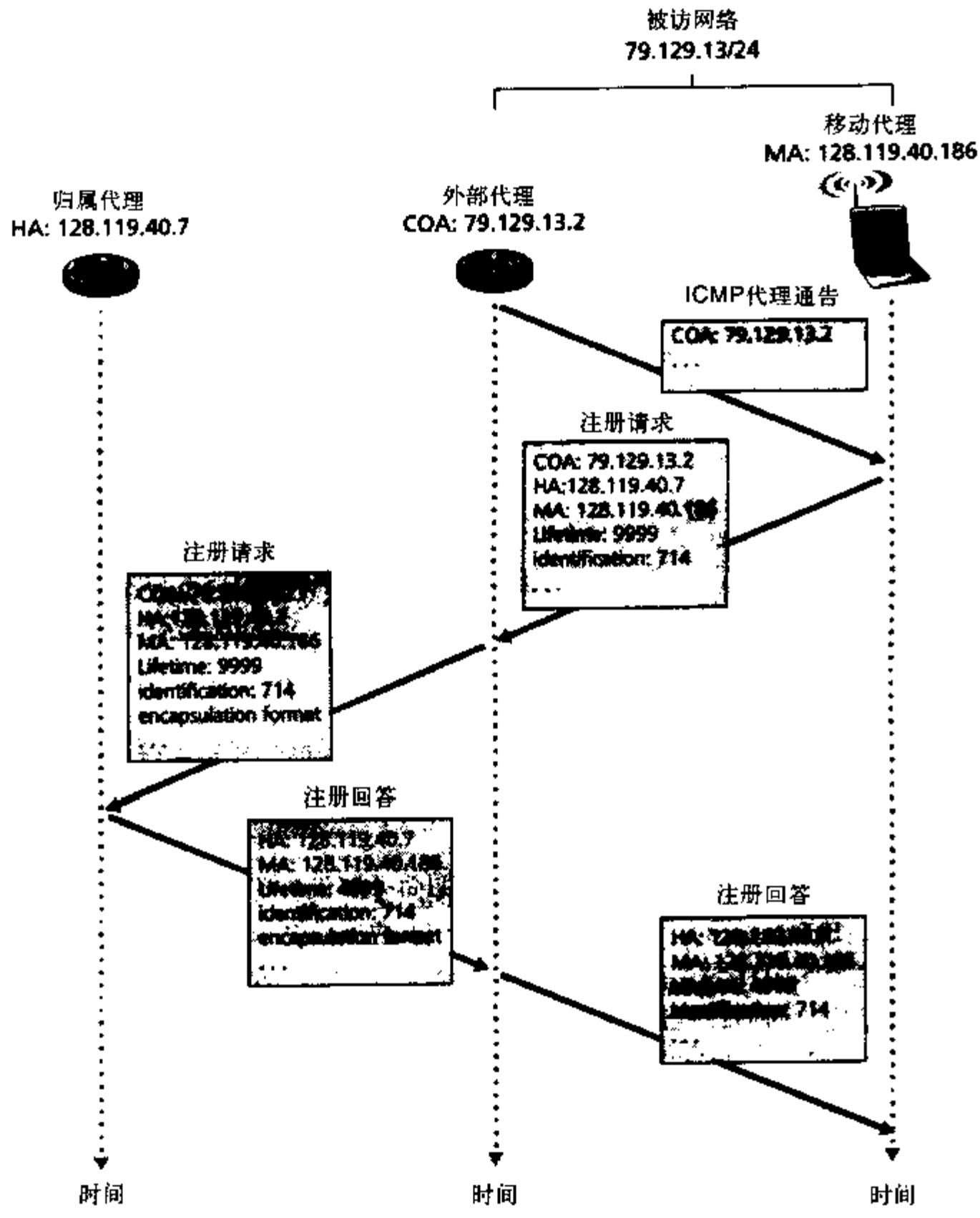


图6-27 代理通告与移动IP注册

除了上面所描述的情况，移动IP标准还允许许多另外的情形和功能，有兴趣的读者可以参阅[Perkins 1998b; RFC 3344]。

6.7 蜂窝网中的移动性管理

分析了在IP网络中如何管理移动性以后，我们现将注意力转向对移动性支持有更长历史的网络，即蜂窝电话网络。尽管在6.4节中我们关注过蜂窝网络中第一跳无线链路，这里我们关注移动性，并以GSM蜂窝网络体系结构[Goodman 1997; Mouly 1992; Scourias 1997; Kaaranen 2001; Korhonen 2003]作为学习的案例，因为它是一个成熟并被广泛部署的技术。与在移动IP中的情况类似，我们将会看到6.5节指出的许多基本原理都被包含在GSM网络体系结

构中。

与移动IP类似，GSM采用了一种间接选路方法（参见6.5.2节），首先将通信者的呼叫选路到移动节点的归属网络，再从那到达被访网络。在GSM术语中，移动主机的归属网络被称作该移动主机的**归属公共地域移动网络**（home Public Land Mobile Network, home PLMN）。由于字首缩略词PLMN有些拗口，考虑到我们避免缩略词字母表的要求，我们直接将GSM归属PLMN称为**归属网络**（home network）。移动用户向某个蜂窝网提供商订购了服务，该蜂窝网就成为了这些用户的归属网络（即该提供商就按月提供的蜂窝服务收取用户的费用）。被访问的PLMN，我们直接称其为**被访网络**（visited network），是移动用户当前所在网络。

与移动IP中情况类似，归属网络和被访网络的职责有很大的差别。

- 归属网络维护一个称作**归属位置注册器**（Home Location Register，HLR）的数据库，其中包括它的每个用户的永久蜂窝电话号码以及用户个人概要信息。重要的是，HLR也包括这些用户当前的位置信息。这就是说，如果一个移动用户当前漫游到另一个提供商的蜂窝网络中，HLR中将包含足够多的信息来获取（通过一个我们即将描述的过程）被访网络中的一个地址，这个地址是对移动用户的呼叫所应该选路到的地址。我们将会看到，当一个呼叫定位到一个移动用户后，通信者将与归属网络中一个被称作**网关移动服务交换中心**（Gateway Mobile services Switching Center，GMSC）的特殊交换机联系。同样，为避免拗口的缩略词，我们这里用一个更具描述性的术语来称呼GMSC，即归属MSC。
- 被访网络维护一个称作**访问者位置注册器**（Visitor Location Register，VLR）的数据库。VLR为每一个当前在其服务网络中的移动用户包含一个表项，VLR表项因此随着移动用户进入和离开网络而出现或消失。VLR通常与移动交换中心（MSC）在一起，该中心协调到达或离开被访网络的呼叫建立。

在实际中，一个服务商的蜂窝网络将为其用户提供归属网络服务，同时为在其他蜂窝服务商订购服务的移动用户提供被访网络服务。

6.7.1 对移动用户呼叫的选路

现在我们描述一个呼叫如何定位到被访网络中的一个移动GSM用户。我们首先考虑下面一个简单的例子，更复杂的例子在[Mouly 1992]中有描述。如图6-28所示，这些步骤如下：

1）通信者拨移动用户的电话号码。该号码本身并不涉及特定的电话线路或位置（毕竟电话号码是固定的，而用户是移动的），号码中的前几位数字足以全局地判别移动用户的归属网络。呼叫从通信者通过公共交换电话网到达移动用户归属网络中的归属MSC。这是呼叫的第一步。

2）归属MSC收到该呼叫并查询HLR来确定移动用户的位置。在最简单的情况下，HLR返回**移动站点漫游号码**（Mobile Station Roaming Number，MSRN），我们简称其为**漫游号码**（roaming number）。注意到这个号码与移动用户的永久电话号码不同，后者是与移动用户的归属网络相关联的，而漫游号码是短暂的：当移动用户进入一个被访网络后，会给移动用户临时分配一个漫游号码。漫游号码的作用就相当于移动IP中转交地址的作用。并且，与COA类似，它也是对通信者和移动用户不可见的。如果HLR不具有该漫游号码，它返回被访网络中VLR的地址。在这种情况下（未在图6-28中显示出来），归属MSC需要查询VLR以便获取移动节点的漫游号码。但是HLR是如何首先得到漫游号码或VLR地址的？移动用户到另一个被访网络后这些值将发生怎样的变化？我们将很快考虑这些重要问题。

3）给定一个漫游号码，归属MSC建立通过网络到达被访网络的MSC呼叫的第二步。至此，该呼叫已经完成，从通信者到达归属MSC，再从归属MSC到达被访MSC，然后到达为移动用户提供服务的基站。

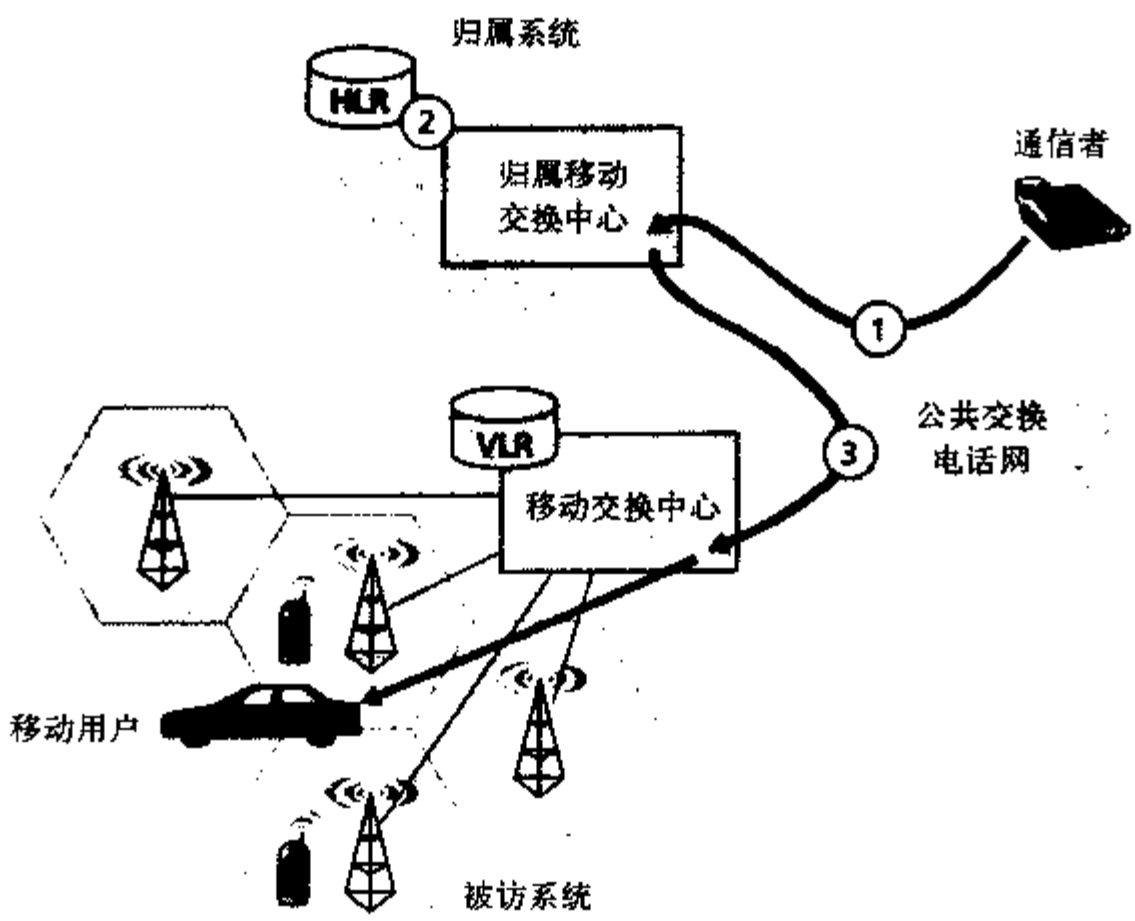


图6-28 将呼叫定位到一个移动用户：间接选路

在第二步中，一个未解决的问题是HLR如何获得有关移动用户位置的信息。当一个移动电话切换或进入一个由新的VLR所覆盖的被访网络以后，移动用户必须向被访网络注册，这是通过在移动用户和VLR之间交换信令报文来实现的。访问VLR随后又向移动用户的HLR发送一个位置更新请求报文。这一报文告知HLR可以用来联系移动用户的漫游号码，或者VLR地址（随后可以查询它以获取移动号码）。作为这个交换的一部分，VLR同样从HLR那里获取移动用户的信息，以及确定被访网络应该给予移动用户什么样的服务。

6.7.2 GSM中的切换

移动站点在一个呼叫过程中，将其关联从一个基站改变到另一个基站，这时就出现了**切换**（handoff）。如图6-29所示，移动用户的呼叫初始时（在切换前）通过一个基站（我们称其为旧基站）选路到该移动用户，而在切换以后它经过另一个基站（我们称其为新基站）选路到移动用户。注意到基站之间的切换不仅导致移动用户向/从一个新的基站传输/接收信号，而且导致正在进行的呼叫从网络中的一个交换点到新基站的重选路。我们首先假设新旧基站共享同一个MSC，并且重选路发生在这个MSC。

有几种原因导致切换的发生，包括：①当前基站和移动用户之间的信号减弱，使得该呼叫有被中断的危险；②一个蜂窝处理的呼叫太多，变得过载。可以通过将一些移动用户切换到邻近不太拥塞的蜂窝中，使这种拥塞得到缓解。

在与一个基站相关联期间，移动用户周期性地测量来自其当前基站和临近它可以“听得到”的基站的信标信号强度。这些测量以每秒1~2次的频率报告给移动用户的当前基站。根据

这些测量值、临近蜂窝的移动用户的当前负载以及其他因素，GSM中的切换由旧的基站发起[Mouly 1992]。GSM标准并未明确规定基站在确定是否进行切换时所采用的具体算法。

图6-30显示了当一个基站决定切换一个移动用户时所包括的步骤：

1）旧基站（BS）通知被访MSC即将要进行一个切换，通知移动用户切换时所涉及的BS（或可能的BS集）。

2）被访MSC开始建立到新BS的路径，分配承载重选路的呼叫所需的资源，以及用信令告知新BS一个切换即将出现。

3）新BS分配并激活一个无线信道供移动用户使用。

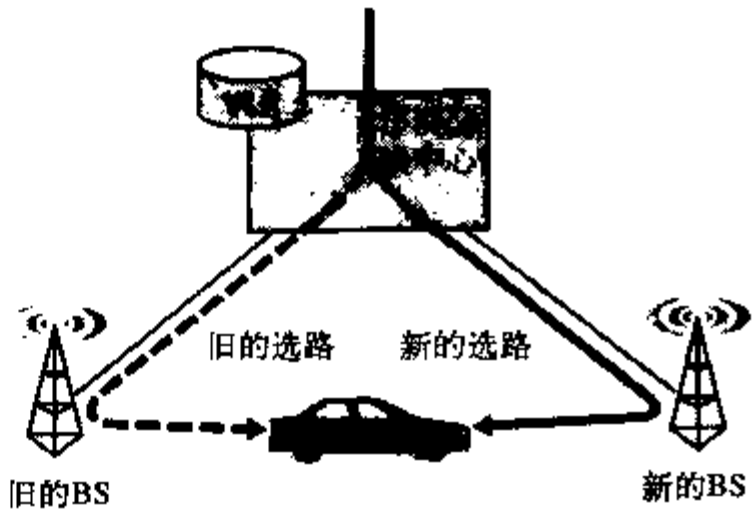


图6-29 在公共MSC中不同基站间的切换情况

4）新BS发出信令返回被访MSC和旧BS，即已经建立了被访MSC到新BS的路径并且告知移动用户即将发生的切换。新BS提供移动用户与新的BS相关联所需要的所有信息。

5）移动用户被告知它应当进行一个切换。注意到此时为止，移动用户完全不知网络已经为切换做好所有底层工作（如在新BS中分配一个信道，分配一条从被访MSC到新BS的路径）。

6）移动用户和新BS交换一个或多个报文，以完全激活新BS中新的信道。

7）移动用户向新BS发送一个切换完成报文，该报文随后向上转发给被访MSC。该被访MSC然后重选路正在进行的到移动用户的呼叫，使其经过新BS。

8）沿着到旧BS的路径分配的资源随后被释放。

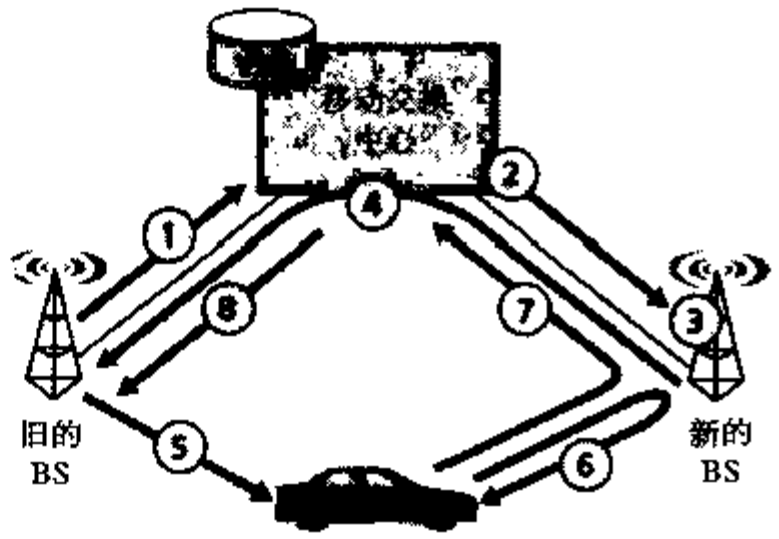
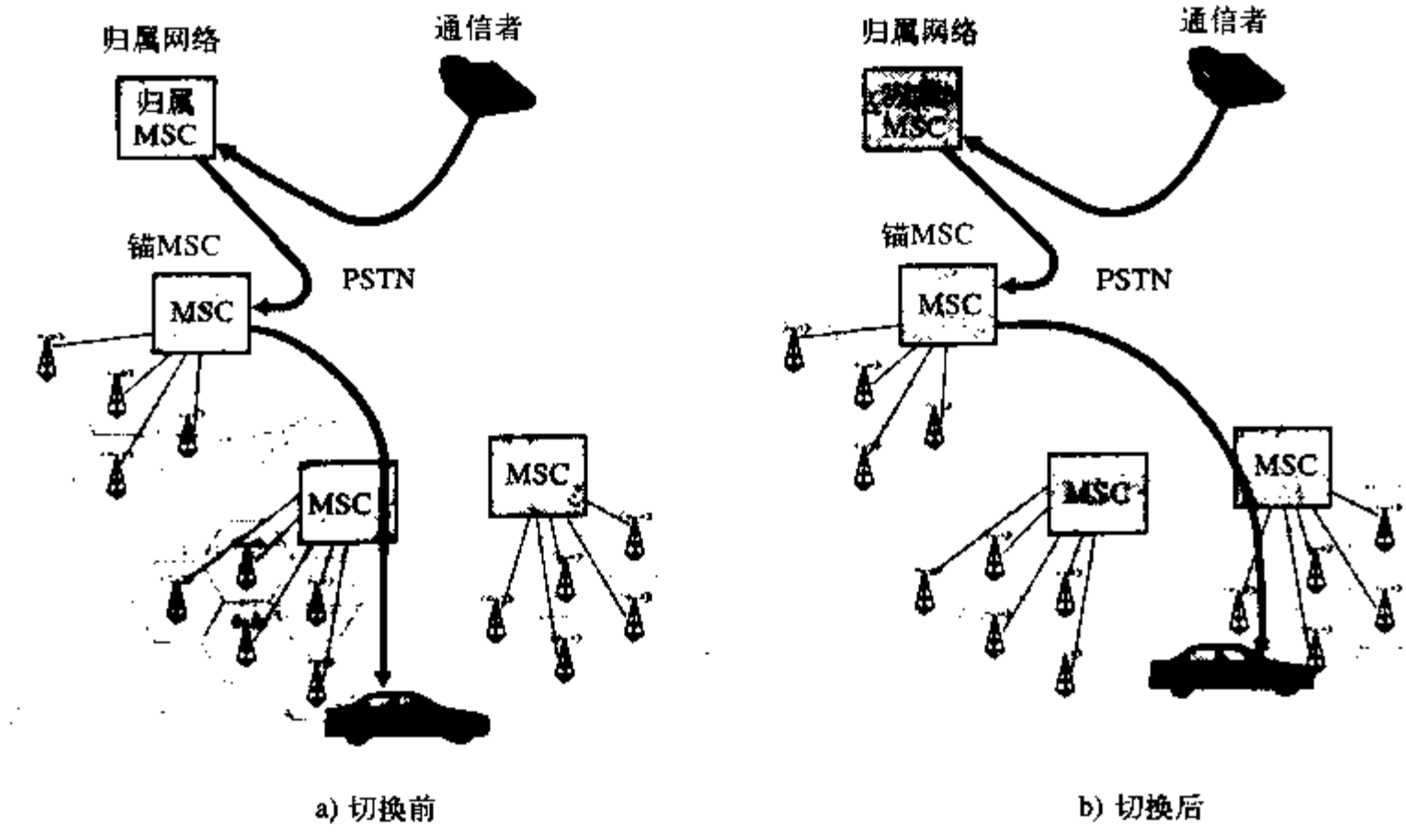


图6-30 在公共MSC中不同基站间完成一个切换的步骤

通过考虑如下情况来总结我们对切换的讨论：当移动用户移动到一个不同于旧BS的、与不同的MSC关联的BS中时，并且当这种MSC之间的切换多次发生时会发生什么情况。如图6-31所示，GSM定义了锚MSC的概念。锚MSC是呼叫首次开始时移动用户所访问的MSC，它因此在整个呼叫持续过程中保持不变。在整个呼叫持续期间，不论移动用户进行多少次MSC间转换，呼叫总是从归属MSC选路到锚 MSC，然后再到移动用户当前所在的被访MSC。当移动用户从一个MSC覆盖区到达另一个MSC覆盖区后，正在进行的呼叫被重选路，从锚MSC到包含新基站的新被访MSC。因此，在任何情况下，通信者和移动用户之间至多有3个MSC（归属MSC，锚 MSC以及被访MSC）。图6-31举例说明了在移动用户所访问的MSC之间的一个呼叫的选路。

另一种方法则不用维持从锚MSC到当前MSC的单一MSC跳，将直接链接移动用户访问的MSC。每当移动用户移到一个新MSC后，让旧MSC将正在进行的呼叫转发给新MSC。这种MSC链接事实上出现在IS-41蜂窝网络中，该网络使用最少步的可选路径来去除在锚MSC和当前被访MSC之间的MSC[Lin 2001]。



a) 切换前　　b) 切换后

图6-31　通过锚MSC重选路

通过对比GSM和移动IP中移动性管理，来完成我们对GSM移动性管理的讨论。表6-2中的对比指出，尽管IP和蜂窝网络在很多方面有很大的区别，但它们共享许多公共功能元素和处理移动性时的总体方法。

表6-2　移动IP和GSM移动性之间的共性

GSM元素	对GSM元素的评论	移动IP元素
归属系统	移动用户永久电话号码所归属的网络	归属网络
网关移动交换中心（或简称归属MSC），归属位置注册器（HLR）	归属MSC：获取移动用户路由地址的联系点。HLR：归属系统中包含移动用户永久电话号码、个人信息、当前位置和定制信息的数据库	归属代理
被访系统	移动用户当前所在的非归属系统网络	被访网络
被访移动服务交换中心，访问者位置注册器（VLR）	被访MSC：负责建立与MSC相关联的发射区中到/从移动节点的呼叫。VLR：被访系统中的临时数据库项，包含每个访问移动用户的订购信息	外部代理
移动站点漫游号码（MSRN），或简称漫游号码	用于归属MSC和被访MSC之间电话呼叫的路由地址，对移动用户和通信者均不可见	转交地址

6.8　无线和移动性：对高层协议的影响

在本章中，我们已经看到了无线网络在链路层（由于无线信道的诸如衰减、多径、隐终端等特性）和网络层（由于移动用户改变他们连接进网络的点）与有线网络的对应物有重大的区别。但在运输层和应用层是否也有重大差别呢？我们很容易认为这些差别是很小的，因为在有线和无线网络中的网络层均为上层提供了同样的尽力而为服务模式。类似地，如果在

有线和无线网络中都是使用诸如TCP和UDP的协议提供运输层服务，那么应用层也应该保持不变。在某方面，我们的直觉是对的，即TCP和UDP可以（也确实）运行在具有无线链路的网络中。在另一方面，运输层协议（特别是TCP）通常在有线和无线网络中有时会有完全不同的性能。这里，在性能方面区别是明显的，我们来研究一下其中的原因。

前面讲过，在发送方和接收方之间的路径上，一个报文段不论是丢失还是出错，TCP都将重传它。在移动用户情况下，丢失可能源于网络拥塞（路由器缓存溢出）或者切换（例如，由于重选路时报文段到移动用户新的网络接入点引入的时延）。在所有情况下，TCP的接收方到发送方的ACK都仅仅表明未能收到一个完整的报文段，发送方并不知道报文段是由于拥塞或切换，还是由于检测到比特错误而被丢弃。在所有情况下，发送方的反应都一样，即重传该报文段。TCP的拥塞控制响应在所有场合也是相同的，即TCP减小其拥塞窗口，如3.7节讨论的那样。由于无条件地降低其拥塞窗口，TCP隐含地假设报文段丢失是由于拥塞而非出错或者切换所致。我们在6.2节看到，在无线网络中比特错误比在有线网络中普遍得多。当这样的比特错误或者切换丢失发生时，没理由让TCP发送方降低其拥塞窗口（并因此降低发送速率）。事实上，此时路由器的缓存可能完全是空的，分组可以在端到端链路中丝毫不受拥塞阻碍地流动。

研究人员在20世纪90年代早期到中期就认识到，由于无线信道的高比特差错率和切换丢失的可能性，TCP的拥塞控制反应在无线情况下可能会出现问题。有三大类方法可能能处理这一问题：

- 本地恢复。本地恢复方法的目标是在比特差错出现的当时和当地（如在无线链路中）将其恢复。如在6.3节学习的802.11 ARQ协议，或者同时使用ARQ和FEC的更加复杂的方法[Ayanoglu 1995]）。
- TCP发送方知晓无线链路。在本地恢复方法中，TCP发送方完全不清楚其报文段跨越一段无线链路。另一种方法是让TCP发送方和接收方知道无线链路的存在，从而将在有线网络中发生的拥塞性丢包和在无线网络中发生的差错/丢包区分开来，并仅对有线网络中的拥塞性丢包采用拥塞控制。[Balakrishnan 1997]在假设端系统能够做出这种区分的情况下，详细研究了多种类型的TCP。[Wei 2004]详细研究了区分端到端路径中有线段丢包和无线段丢包的技术。
- 分离连接方法。在分离连接方法中[Bakre 1995]，移动用户和其他端点之间的端到端连接被断为两个运输层连接：一个从移动主机到无线接入点，一个从无线接入点到其他通信端点（我们假定这是个有线的主机）。该端到端连接因此是由无线部分和有线部分串联形成的。经无线段的运输层能够是一个标准的TCP连接[Bakre 1995]，或是一个特别定制的运行在UDP上的差错恢复协议。[Yavatkar 1994]研究了经无线连接使用运输层选择性重传协议的情况。[Wei 2006]中的测量报告指出了分离TCP连接广泛用于蜂窝数据网络中，通过使用分离TCP连接性能的确能够有很大改进。

我们有关无线链路上的TCP的讨论十分简要，建议读者查阅相关文献，以了解这个正在进行的研究领域的详情。

考虑过运输层协议后，我们接下来考虑无线和移动性对应用层协议的影响。这里一个重要的考虑是：如我们在图6-2中所见，无线链路经常具有相对较低的带宽。因此运行在无线链路尤其是蜂窝无线链路上的应用程序，必须将带宽作为稀有物品对待。例如，一个为在3G电话上运行的Web浏览器提供服务的Web服务器，就不能像为运行在有线连接上的浏览器那样提

供含有大量图片的内容。尽管无线链路的确对应用层提出一些挑战，它们提供的移动性同样使得一大批位置知晓和环境知晓应用成为可能[Chen 2000]。更一般地，无线和移动网络将在实现未来无处不在的计算环境中扮演重要角色[Weiser 1991]。显然，在谈及无线和移动网络对网络应用及其协议的影响时，我们仅看到了冰山一角！

6.9 小结

无线网络和移动网络使电话发生了革命性变化，同时也对计算机网络界产生了深远的影响。伴随着它们对全球网络基础设施的随时、随地、无缝地接入，它们不仅使网络接入变得更加无所不在，而且催生了一组新的、令人兴奋的位置相关服务。考虑到无线网络和移动网络不断增长的重要性，本章关注于原理、通用链路技术以及用于支持无线和移动通信的网络体系结构。

本章以对无线网络和移动网络的介绍开始，描述了由这种网络中通信链路的无线特性所引发的挑战和由这些无线链路带来的移动性所引发的挑战之间的重要区别。这使我们能够更好地隔离、标识和掌握每个领域中的关键概念。我们首先关注于无线通信，在6.2节中考虑了无线链路的特征。在6.3节和6.4节中，我们研究了IEEE 802.11（WiFi）无线LAN标准，802.16 WiMAX标准，802.15.1蓝牙标准和蜂窝因特网接入。然后我们将注意力转向移动性问题。在6.5节中我们确定了多种形式的移动性，不同的移动性面临不同的挑战，提供了不同的解决方案。我们讨论了对移动节点的定位和选路问题，以及对那些动态地从一个网络接入点移到另一个网络接入点的移动用户的切换问题。在6.6节和6.7节中，我们分别分析了这些问题在移动IP和GSM中是如何处理的。最后，我们在6.8节中讨论了无线链路和移动性对运输层协议和网络应用的影响。

尽管我们用了整整一章研究无线网络和移动网络，全面探索这个令人兴奋和快速扩展的领域需要一整本书或更多篇幅。我们鼓励读者通过查阅在本章中提供的许多参考资料，对这一领域进行更深入的了解。

课后习题和问题

复习题

6.1节

1. 一个无线网络运行在“基础设施模式”下是什么含义？如果某网络没有运行在基础设施模式下，那么它运行在什么模式下？这种运行模式与基础设施模式之间有什么不同？
2. 在6.1节中的分类法中，所确定的四种类型的无线网络是什么？你已经使用的无线网络是这些类型中的哪一种？

6.2节

3. 下列类型的无线信道损伤之间有什么区别：路径损耗、多径传播、来自其他源的干扰？
4. 随着移动节点离开基站越来越远，为了保证传送的帧的丢失概率不增加，基站能够采取的两种动作是什么？

6.3节

5. 描述802.11中信标帧的作用。
6. 是非判断：802.11站在传输一个数据帧前，必须首先发送一个RTS帧并收到一个对应的CTS帧。

7. 为什么802.11中用到了确认，而有线以太网中却未使用？
8. 是非判断：以太网和802.11使用相同的帧格式。
9. 描述RTS门限值的工作原理。
10. 假设IEEE 802.11 RTS帧和CTS帧与标准的DATA帧和ACK帧一样长，使用CTS帧和RTS帧还会有好处吗？为什么？
11. 6.3.4节讨论了802.11移动性，其中一个无线站点从一个BSS到同一子网中的另一个BSS。当AP是通过交换机互联时，为了让交换机能正确转发帧，AP需要发送一个带有哄骗的MAC地址的帧，为什么？
12. 蓝牙中的主设备和802.11网络中的基站之间有什么区别？
13. 是非判断：基站必须以相同的信道速率传输到所有的节点。
14. 在WiMAX中“机会主义调度”的含义是什么？
15. 在6.3.2节中我们了解到有两个主要的3G标准：UMTS和CDMA-2000，这两个标准每个分别源于哪个2G和2.5G标准？

6.5～6.6节

16. 如果某种节点与因特网具有无线连接，则该节点必定是移动的吗？试解释之。假设一个使用膝上型电脑的用户带着电脑绕住所散步，并且总是通过相同的接入点接入因特网。从网络的角度看，该用户是移动的吗？试解释之。
17. 永久地址与转交地址之间有什么区别？谁指派转交地址？
18. 考虑经移动IP的一条TCP连接。是非判断：在通信者和移动主机之间的TCP连接阶段经过该移动用户的归属网络，但数据传输阶段直接通过该通信者和移动主机，绕开了归属网络。

6.7节

19. 在GSM网络中，HLR和VLR的目的是什么？移动IP的哪个元素类似于HLR和VLR？
20. 在GSM网络中，锚MSC的作用是什么？

6.8节

21. 为了避免单一无线链路降低一条端到端运输层TCP连接的性能，能够采取的三种方法是什么？

习题

1. 考虑在图6-5中单一发送方的CDMA例子。如果发送方的CDMA码是（1，−1，1，−1，1，−1，1，−1），那么其输出（对于所显示的两个数据比特）是什么？
2. 考虑图6-6中的发送方2，发送方对信道$Z^2_{i,m}$的输出是什么（在它加上来自发送方1的信号前）？
3. 假设在图6-6中的接收方希望接收由发送方2发送的数据。说明通过使用发送方2的代码，（经计算）接收方的确能够将发送方2的数据从聚合信道信号中恢复出来。
4. 对于两个发送方、两个接收方的情形，给出一个包括1和−1值的两个CDMA编码的例子，不允许两个接收方从两个CDMA发送方提取初始传输的比特。
5. 假设有两个ISP在一个特定的咖啡馆内都提供WiFi接入，并且每个ISP有自己的AP和IP地址块。
   a. 进一步假设，两个ISP都意外地配置其AP运行在信道11。在这种情况下，802.11协议是否将完全崩溃？讨论一下当两个各自与不同ISP相关联的站点试图同时传输时，将会发生什么情况。
   b. 现在假设一个AP运行在信道1，而另一个运行在信道11。这时将会发生什么变化？
6. 在CSMA/CA协议的第4步，一个成功地传输了一个帧的站点为传输第2个帧，从第2步而非第1步开始CSMA/CA协议。通过不让这样一个站点立即传输第2个帧（即使侦听到信道空闲），CSMA/CA的设

计者是基于怎样的基本原理来考虑的呢?

7. 假设一个802.11b站点被配置为始终使用RTS/CTS序列预约信道。假设该节点突然希望发送1000字节的数据，并且所有其他站点此时都是空闲的。作为SIFS和DIFS的函数，并忽略传播时延，假设无比特差错，计算发送该帧和收到确认需要的时间。

8. 考虑在图6-32中显示的情形，其中有四个无线节点A、B、C和D。这四个节点的无线电覆盖范围如图中的椭圆型阴影所示；所有节点共享相同的频率。当A传输时，仅有B能听到/接收到；当B传输时，A和C能听到/接收到；当C传输时，B和D能听到/接收到；当D传输时，仅有C能听到/接收到。
假定现在每个节点都有无限多的报文要向每个其他节点发送。如果一个报文的目的地不是近邻，则该报文必须要中继。例如，如果A要向D发送，来自A的报文必须首先发往B，B再将该报文发送给C，C则再将其发向D。时间是分隙的，报文所用的传输时间正好是一个时隙，如在时隙Aloha中的情况一样。在一个时隙中，节点能够做下列工作之一：(i)发送一个报文（如果它有报文向D转发）；(ii)接收一个报文（如果正好一个报文要向它发送）；(iii)保持静默。如通常情况那样，如果一个节点听到了两个或更多的同时发送，出现冲突，并且重传的报文没有一个能成功收到。你这时能够假定没有比特级的差错，因此如果正好只有一个报文在发送，它将被位于发送方传输半径之内的站点正确收到。

   a. 现在假定一个无所不知的控制器（即一个知道在网络中每个节点状态的控制器）能够命令每个节点去做它（无所不知的控制器）希望做的事情，例如发送报文，接收报文，或保持静默。给定这种无所不知的控制器，数据报文能够从C到A传输的最大速率是什么，假定在任何其他源/目的地对之间没有其他报文?

   b. 现在假定A向B发送报文，并且D向C发送报文。数据报文能够从A到B且从D到C流动的组合最大速率是多少?

   c. 现在假定A向B发送报文且C向D发送报文。数据报文能够从A到B且从C到D流动的组合最大速率是多少?

   d. 现在假定无线链路由有线链路代替。在此有线链路情况下，重复问题a~c。

   e. 现在假定我们又在无线状态下，对于从源到目的地的每个数据报文，目的地将向源回送一个ACK报文（例如，如同在TCP中那样）。对这种情况重复上述问题a~c。

图6-32 习题8的情形

9. 描述802.15.1蓝牙帧的格式。需要阅读某些课外读物来获取这方面的信息。在帧格式中有哪些东西本质上限制802.15.1网络中主动节点的数量为8个？解释该问题。

10. 考虑下列理想化的WiMAX情形。下游子帧（参见图6-17）划分为时隙，每子帧为N个下游时隙，且所有时隙具有等长时间。有四个节点A、B、C和D在下游信道上分别以10 Mbps、5 Mbps、2.5 Mbps和1 Mbps速率从基站可达。该基站有无限量的数据要向每个节点发送，并且能在下游子帧中的任何时隙期间发送到节点的任何一个。

    a. 假定基站在每个时隙期间能够向它选择的任何节点发送，它能向节点发送的最大速率是多少?你的

解决方案公平吗？解释并定义你所讲的“公平”的含义。

b. 如果有每个站点在每个下游子帧期间必须收到等量的数据这样的公平要求，在下游子帧期间基站（向所有节点）的平均传输速率是多少？

c. 假定该公平性准则是在子帧期间任何节点能够接收至多是任何其他节点两倍的数据？解释你是如何得到答案的。

11. 在6.5节，一种允许移动用户在外部网络间移动过程中保持其IP地址不变的建议方案是，让外部网络通告一个到该移动用户高度特定的路由，并使用现有的选路基础设施在整个网络中传播这一信息。我们将扩展性作为一种考虑因素。假设移动用户从一个网络移动到另一个网络后，新的外部网络通告一个到移动用户的特定路由，旧的外部网络丢弃其路由。考虑路由信息如何在一个距离向量算法中传播（尤其是对于跨越全球的网络间的域间选路情况）。

a. 一旦外部网络开始通告其路由，其他路由器能否立刻将数据报选路到新的外部网络？

b. 不同的路由器有可能认为移动用户位于不同的外部网络吗？

c. 讨论网络中其他路由器最终知道到达移动用户的路径的时间范围。

12. 假设图6-21中通信者是移动的。概述将数据报从初始移动用户选路到（现在移动的）通信者所需要的额外的网络层基础设施。像图6-22中那样，显示初始移动用户和（现在移动的）通信者之间数据报的结构。

13. 在移动IP中，移动性将对数据报在源和目的间的端到端时延有怎样的影响？

14. 考虑6.7.2节最后讨论的链的例子。假设一个移动用户访问外部网络A、B和C，当通信者在外部网络A中时，它开始一条与移动用户的连接。列出在外部代理之间和外部代理与归属代理之间，当移动用户从网络A到网络B再到网络C的过程中的报文序列。然后，假设未执行链，并且通信者（以及归属代理）必须被显式地告知移动用户的转交地址的改变。列出在第二种情况下需要交换的报文序列。

15. 考虑在一个具有外部代理的外部网络中的两个移动节点。在移动IP中，这两个移动节点是否可能使用相同的转交地址？解释你的答案。

16. 在我们对VLR如何用移动用户当前位置信息更新HLR的讨论中，与VLR地址对HLR相比，提供MSRN所具有的优缺点各是什么？

讨论题

1. 列举当今市场中提供蓝牙或802.15接口的5种产品。

2. 3G无线服务在你所在的地区是否可用？价格如何？目前支持哪些应用？

3. 作为一个IEEE 802.11的用户，你发现了哪些问题？如何改进802.11的设计以便克服这些问题？

4. 利用Web搜索来学习有关WiMAX的部署试验。这些试验的范围有多大呢？在什么样的距离上取得的吞吐量有多大？用户数有多少？

5. 利用Web搜索来学习有关EVDO和HSDPA的部署。至今已经进行广泛部署了吗？在什么地方？

Ethereal实验

在本书的配套Web站点上（http://www.awl.com/kurose-ross），可以找到有关本章的Ethereal实验，这个实验捕获和研究在无线便携机和接入点之间交换的802.11帧。

人物专访

Charles Perkins

Charles E. Perkins 是诺基亚研究中心Palo Alto系统研究中心的一名诺基亚会士，负责研究移动无线网络以及动态配置协议。他在因特网工程任务组（IETF）的移动IP工作组担任文档编辑，并且撰写和与人合写了mip4、mip6、manet、dhc、seamoby（无缝移动性）和自配置工作组的标准轨迹文档，同时也是几种与无线网络相关的ACM和IEEE期刊的编辑。在诺基亚，他一如既往地全身心地投入自组织网络的研究活动，还研究无数可移动无线设备接入因特网的可扩展性和性能问题。Charles撰写和编辑了有关移动IP和自组织网络的书籍，并在移动网络、自组织网络、移动网络中的路由优化、资源发现以及移动计算机自动配置领域发表了多篇论文和获奖文章。他是MobiHoc的创建者之一，并担任它的总负责人和程序委员会主席。Charles服务于IETF的因特网体系结构委员会和国家研究委员会，以及陆军研究实验室和瑞士MICS计划等多个技术评估管理委员会。

• 为什么您决定专门从事无线/移动性研究？

我涉足无线网络和移动性是20世纪80年代后期在IBM研究中心从事一个项目后自然发展的结果。我们具有无线链路，并试图建立一个带有无线连通性和手写识别的“ThinkPad”风格的设备（如一个Palm Pilot）。

我们构建了一个简单的解决方案（后来被称作“移动IP”），并发现它运行正常。运用有关移动IP的经验，我们策划对RIP进行一个快速、有效的修改，使其能实现自组织，这也运行得相当好。我这里所说的“运行”，是指应用程序可以不做任何修改就能很好地运行，并且网络不因为我们的新设计而陷于困境。这些特性可称为“应用透明性”和“可扩展性”。

当然，实验室中的运行与商业成功有很大的不同，两种技术都还有许多未满足的商业潜力。

• 您在计算机行业的第一份工作是什么？

那时我在得克萨斯州休斯敦的TRW Controls工作。那与大学学习生活有很大的不同。

我在TRW Controls知道的一件事情是，甚至对于一些非常重要的实用控制系统，其支持软件也非常糟糕。这些系统被用来控制大电网中的电流流动，但这些应用层软件却以一种十分糟糕的方式构建。另外，进度表经常被压缩，程序员完全不考虑管理的意图和他们的工作环境。

整个系统需要从头重新设计。我没有太多的理由相信在过去30年中情况有所改变，尤其是围绕2003年出现的大断电事故的最近一些事件来看。事实上，就违反常规而言，它几乎是变得更坏了。

我很高兴离开了TRW Controls，加入了Tektronix（Tek Labs）。

• 您工作中最富有挑战性的部分是什么？

我工作中最富有挑战性的部分是理解我应该做什么来帮助我的公司。另外，我把实现我所得到的无线技术，为人们提供更好的服务以及让人们有更愉快的日常经历，作为我工作的一部分。我们公司正从事“连接人们”的业务，我希望我的工作有助于使那些连接变得尽可能和谐、平滑。以一种使我们所开发的技术的潜在利润最大化的方式从事这项工作，使得每天都成为一个新的挑战。特别在无线领域中，我认为安全技术必须发展成为人们所希望的和让人欣赏的（像雨衣一样），而不是不堪重负和令人恐惧的（如同今天大部分情况那样）。

在更加详细的技术层次上，事实上我感到很多安慰。我试图通过如下方式解决网络协议问题：在无线设备（和它们的电池！）上放置最小的负担，以及尽量不让用户感到不便利。以一种新的高速无线技术互联当今的无线电话和因特网，技术上极其引人入胜，并将为那些找到正确前进道路的人提供无限的

商业成功的潜力。

• 您如何看待无线的未来?

整个无线行业正经历着一场巨大的变化，目前无法看到终点。新的高速无线技术不断涌现，并可能具有不可预料的实际影响，这些影响可能会从根本上改变社会。我们当前对隐私权的期望以及我们互相通信过程中的一些局限（如声音、图像和数据）在十年内将不复存在。随着企业越来越多地转向无线通信，很可能会采取新的安全措施，并将极大地改变我们的工作经验。

看起来我们将得到更多的为各种方案而分配的频谱用于无线电通信。这些通信具有非常高的速率。社区将向其居民提供越来越高速的无线通信，仿佛整个城镇是一个局域网。这可以复兴很久以来在我们的社会（至少是在美国）中丢失的社区的概念。当然，社区仍然需要接入因特网。磁盘能力增长得十分迅速，价格变得十分便宜，这使得我们已经能够在口袋中携带整个维基百科（Wikipedia）以及或许全世界的电话号码，更不用说空前的书籍、音乐和电影的个人图书馆。

无线非常可能加速因特网的发展。随着无线设备变得越来越便宜，我们将会看到因特网通信变得无处不在（耳饰，多人游戏，地铁费用读卡机），这将催生新的应用和新的安全解决方案。

第7章　多媒体网络

我们当前正在见证音频和视频应用在因特网上的广泛部署。数以百计的场点（包括CCN、Rhapsody、Napster、MSN、AOL、Yahoo!）产生可用的流式音频和视频内容。YouTube和其他视频共享场点允许用户以按需方式观看已被其他用户上载的视频片段。数以百万计的用户为满足他们的电话和视频会议需求而习惯性地使用Skype。并且某些传统的电视频道现在正通过因特网进行分发，允许因特网用户观看源于世界各个角落的电视频道。多媒体因特网应用的这种爆炸性增长主要是宽带住宅接入和高速无线接入（如WiFi）的日益渗透的结果。如在1.2节所讨论的那样，宽带接入速率将继续增加，因而进一步加速新型、令人兴奋的多媒体应用的部署。

多媒体应用的服务要求与传统的弹性应用——如电子邮件、Web浏览、远程登录和文件下载和共享（这些已在第2章中学习过）——的服务要求有非常大的不同。特别与弹性应用不同的是，多媒体应用对端到端时延和时延变动高度敏感，但是可以容忍偶然的数据丢失。

我们从7.1节的多媒体应用的分类方法开始。我们将看到多媒体应用能够分为*流式存储音频/视频*，*流式实况音频/视频*，或*实时交互音频/视频*等几类。我们将进一步看到这些应用类型中的每一种都对网络有不同的服务需求。在7.2节中，我们较为详细地研究了流式存储音频/视频。在7.3节中，我们将研究一些应用层的技术，能够在今天的尽力而为的因特网上加强多媒体应用的性能。在7.4节中我们涵盖了在当今的因特网上使用的几种多媒体协议。在7.5节中，我们将研究网络中的一些机制，这些机制能用于区分不同类型的流量（如多媒体这样的时延敏感的应用和FTP这样的弹性应用），在几种类型的流量中提供了区分服务。最后，在7.6节中，我们将考虑这样一种情况，即网络必须对某应用确保性能，如基于分组的IP电话呼叫将得到与电路交换电话网络中一样的性能。我们将看到这将要求引入新的网络机制和协议。

7.1　多媒体网络应用

在第2章有关应用服务要求的讨论中，我们确定了一些轴，服务要求可以根据它们分类。其中的两个轴（即定时考虑和数据丢失容忍度）对网络多媒体应用尤其重要。定时考虑是很重要的，因为许多多媒体应用是高度**时延敏感**（delay-sensitive）的。我们不久将看到在很多多媒体应用中，发送方到接收方的时延超过几百毫秒的分组对接收方基本上就没有用处了。另一个方面，大部分网络多媒体应用**容忍丢包**（loss-tolerant）（偶尔的丢失只会在音频/视频回放时偶尔出现干扰信号，而且这些丢失经常可以部分或者全部隐藏）。这些时延敏感但容忍丢失的特性明显不同于那些弹性的服务（例如Web、电子邮件、FTP和Telnet）。对于这些弹性服务，长时延令人恼火，但并不是特别有害，传输数据的完全和完整性是首要的。

7.1.1　多媒体应用的例子

因特网能够支持各种各样激励人心的多媒体应用。在本节，我们考虑三大类多媒体应用：流式存储音频/视频、流式实况音频/视频和实时交互音频/视频。

在本章我们不讨论下载后播放（download-and-then-play）应用，例如在回放MP3之前，通

过一个P2P文件共享应用程序完全下载该MP3。事实上下载后播放应用是弹性的，文件传输应用没有任何特殊时延要求。我们在第2章研究了文件传输（HTTP和FTP）和P2P文件共享系统。

历史事件

IPTV

电视内容传统上通过陆地微波、混合光纤同轴电缆（HFC）和同步卫星信道分发（参见1.2节）。但处于今天的因特网时代，人们对IPTV有着巨大的兴趣，也就是说经过因特网来分发电视内容。

IPTV的挑战之一是处理所要求的巨大带宽，特别是在服务器源头。例如，考虑分发一个重要的体育事件，如世界杯比赛，从一台服务器经因特网到1亿个并行的用户。如果视频速率是不起眼的1Mbps，则所需要的服务器带宽将达到令人震惊的100 Tbps! 因此，通过经典的客户机服务器分发是完全不可能的。如果IP多播通过因特网在世界广泛部署的话，那么将很容易使IPTV成为现实。另一种方法是经过多播覆盖网络分发视频，例如由内容分发网络（CDN）所提供（参见7.3节）。

还有另一种方案是使用对等方到对等方分发，由此每个接收到电视频道的对等方也帮助向其他对等方重分发该频道。也许这种方法最大的诱人之处在于较低的分发成本：如果各个对等方总体上提供了充分的上行带宽，那么就可能需要少量的服务器带宽（也许仅是几倍的视频速率）。以这样的低成本，使用Web照相机的任何人都能够以微不足道的成本向数以百万计的用户分发一个实况节目。

到此为止，一些类似于BitTorrent的P2P IPTV系统已经得到了成功的部署。该领域的先驱CoolStreaming据称2003年的同时用户已超过了4000[CoolStreaming 2005]。最近，一些其他系统包括PPLive和ppstream据称获得了很大的成功，数以万计的同时用户以300 kbps到1 Mbps的速率观看频道。在这些类似于BitTorrent的系统中，对等方形成了一个动态的覆盖网络，并与覆盖邻居交换视频块。在未来的5～10年中，IPTV将如何发展将是十分有趣的事。将使用何种支撑技术：CDN或P2P，或这两种混合的某种技术？世界杯的大部分球迷将能从因特网观看2014年的比赛吗？

1. 流式存储音频和视频

在这类应用中，客户机根据需求请求存储在服务器上的被压缩的音频或视频文件。目前，数以千计的场点提供流式存储音频和视频，包括CNN、微软视频和YouTube。这类应用有三个关键的与众不同的特征：

- *存储媒体*。多媒体的内容已经预先录制，并存储在服务器上。因此一个用户可以暂停、倒退、快进或者检索多媒体内容。从一个客户机提出这种请求到该动作在客户机上表现出来，这期间可接受的响应时间应该为1～10 s的量级。
- *流*。在流式存储音频/视频应用中，客户机通常从服务器接收文件几秒之后，就开始播放音频/视频。这意味着当该客户机在从文件的一个位置开始播放音频/视频的同时，还在从服务器接收文件的后续部分。这种技术被称为流（streaming），它避免了在开始播放之前必须下载整个文件（这会引起了一个潜在的长时延）。有很多流式多媒体客户机程序，包括RealNetworks公司的RealPlayer [RealNetworks 2007]、苹果公司的QuickTime

[QuickTime 2007]和微软的Windows Media [Microsoft Media Player 2007]。

- 连续播放。一旦多媒体内容开始播放，它应该根据初始记录的时序进行。为了在客户机播放，必须从服务器中及时接收数据；否则，用户会经历令人沮丧的缓存时延。尽管存储的媒体应用有连续播放要求，然而它们的端到端时延限制比那些实况的、交互的应用（例如因特网电话和视频会议，见下文）要宽松。

2. 流式实况音频和视频

这类应用类似于传统的电台广播和电视，只是它通过因特网来传输而已。这些应用允许用户接收从世界上任何角落发出的实况无线电广播和电视传输。（例如，本书的一位作者当旅行时经常听他最喜欢的费城无线电台。另一位作者居住在法国的一年间，经常听他钟爱的大学篮球队的实况广播。）这些应用经常是指因特网无线电和IPTV。今天在因特网上有数以千计的无线电台和一些IPTV部署。

因为流式的实况音频/视频不再存储，客户机不能实现媒体的快进。但是通过接收数据的本地存储，能够实现诸如暂停和回退等其他交互操作。像实况广播这样的应用经常有很多接收相同音频/视频节目的客户机。通过使用在4.7节描述的IP多播技术，能够有效地完成向多个接收方分发实况音频/视频。然而，今天的实况音频/视频的分发通常更多的是通过应用层多播（使用P2P或CDN）或多个独立的服务器到客户机的单播流来完成。尽管定时限制没有实时交互应用那么严格，但是它和流式存储多媒体一样要求连续播放。从用户请求传输/播放一个实况传输到播放开始，可以容忍的时延最多为几十秒。

3. 实时交互音频和视频

这类应用允许人们使用音频/视频互相实时通信。在因特网上的实时交互音频通常称为**因特网电话**（Internet telephony），因为从用户的角度上看，它类似于传统的电路交换电话服务。因特网电话能够以非常低的费用提供PBX（专用用户交换机）、本地和长途电话服务。它也能推动传统电路交换网络难以支持的新业务的发展，诸如存在检测、Web电话集成等。现在有很多可用的网络电话产品。例如，Skype用户能够进行PC到电话和PC到PC的语音呼叫。通过实时交互视频，也称为视频会议，个人通信不仅可以听见也可以看见。现在也有许多因特网上可用的实时交互视频产品，包括微软的NetMeeting，Skype视频和各种Polycom产品。注意到在实时交互的音频/视频应用中，用户可以在任何时刻说话或者转换控制权。为了使多个交谈者之间交互谈话，从一个用户说话或者移动，到这个动作在接收主机上呈现，这期间的时延应该小于几百毫秒。对于语音，小于150 ms的时延不会被听者觉察到，150～400 ms的时延能够被接受，当超过400 ms的时延时，即使不会使对话变得完全无法理解，也会使语音交谈变得令人沮丧。

7.1.2 当今因特网上的多媒体障碍

前面讲过，在当今的因特网上使用的IP协议对它携带的所有数据报提供一种**尽力而为服务**（best-effort service）。换句话说，因特网尽力使每个数据报尽可能快地从发送方传送到接收方，但是它对于单个分组的端到端时延不做任何承诺。这种服务对在一个分组流中的分组的时延变化也不做任何承诺。因为TCP和UDP在IP之上运行，这使得这些运输协议中的任何一种对调用的应用都无法保证时延。由于缺乏以同步方式交付分组的任何特定努力，开发成功的因特网多媒体网络应用是一个极富挑战性的问题。无论如何，到现在为止，因特网上的多媒体已经获得了显著的成功。例如，用户交互时延在5～10 s的流式存储音频/视频现在在因

特网上是很普遍的。但是在流量峰值时期，性能可能并不令人满意，特别是当参与的链路拥塞时（例如拥塞的越洋链路）更是如此。

因特网电话和实时交互视频也已经得到了广泛的应用，例如，日常任何给定时间都有700多万个Skype用户在线。实时交互语音和视频对分组时延和时延抖动施加了严格的限制。**分组时延抖动**（packet jitter）是指同一个分组流中分组时延的变化。实时语音和视频在带宽充足时能够工作得很好，因而时延和时延抖动最小。但是只要实时语音或视频分组流遇到了一段有一定程度拥塞的链路，它的质量就变得难以接受了。

如果存在某种第一类和第二类的因特网服务，多媒体应用的设计当然更直接，从而第一类分组在路由器队列中的数量和所接收的服务优先级上都有所限制。这种第一类服务对于时延敏感应用来说可能是令人满意的。但是到现在为止，因特网主要是采用公平的方法在路由器队列中进行分组调度。所有的分组接收相同的服务；任何分组（包括时延敏感的音频和视频分组）在路由器队列中都没有特殊的优先级。不论你有多少钱或者你有多重要，你必须加入队列的最后，等待轮到为你服务的时刻到来。在本章的后半部分，我们将研究意在消除这种限制的体系结构。

因此目前我们必须生活在尽力而为服务之中。但是在此限制之下，我们能够做出几个设计决定和使用一些技巧来改善用户可以觉察到的多媒体网络应用的质量。例如，我们可以通过UDP发送音频和视频，从而避免当TCP进入慢启动阶段时的低吞吐量。我们可以在接收方延迟100 ms或者更长时间播放，来减小网络引入的时延抖动的影响。我们可以在发送方对分组加时间戳，以便接收方知道什么时候应该播放分组。对于存储的音频/视频，当客户机有存储空间和额外带宽可用时，我们可以在播放期间预取数据。我们甚至可以发送冗余信息以减轻网络引入的丢包的影响。我们将在本章前半部分的余下内容中研究许多这方面的技术。

7.1.3 因特网应该如何演化才能更好地支持多媒体

现在有关因特网应该如何发展的争论一直不断，也就是讨论如何更好地适应有严格时间限制的多媒体流量。争论的一个极端是，一些研究者主张应该对因特网进行根本性改造，以便应用可以明确地预留端到端的带宽并因此得到端到端性能的保证。**硬保证**（hard guarantee）是指应用将必定得到它所请求的服务质量（QoS）。**软保证**（soft guarantee）是指应用将以很高的概率得到它所请求的服务质量。这些研究者认为，如果用户要建立一个例如从主机A到主机B的因特网电话呼叫，那么该用户的因特网电话应用应该能够明确地在沿着这两个主机之间路由的每段链路上预留带宽。然而，允许应用进行预留，而且需要网络来支持这些预留，这需要做很大的变化。首先，我们需要一个协议代表应用来预留从发送方到接收方的带宽。第二，我们必须修改路由器的调度策略以便带宽预留能够兑现。通过这些新的调度策略，并非所有的分组得到同样的对待，而是预留（并且付费）的越多则得到的越多。第三，为了兑现这些预留，应用必须向网络提供它们要发送到网络的流量的描述。然后网络则必须监管每个应用的流量来确保它遵守这个描述。最后，网络必须有手段来判断它是否有足够的可用带宽来支持任何新的预留请求。当这些机制结合起来时，则主机和路由器中需要新的、复杂的软件以及新的服务类型。在7.6节中我们将详细地研究这些机制。

争论的另一个极端是，一些研究者主张不必对尽力而为服务和支撑的因特网协议做任何根本的改变，相反，他们主张采取一种自由放任的方法：

- 随着需求的增加，ISP（包括高层和低层的ISP）将扩大它们的网络规模来满足该需求。

特别是ISP将在它们的网络中提供充足的带宽和交换容量，以提供令人满意的时延和丢包性能[Huang 2005]。ISP从而将向客户（用户和客户ISP）提供更好的服务，通过更多的客户和更高的服务费来换得更高的收益。为了保证多媒体应用得到适当的服务，甚至在超载的场合也能得到，ISP可能过度配备带宽和交换能力。借助于适当的流量预测和带宽配备，能够进行软QoS保证。

- 内容分发网络（Content Distribution Network, CDN）复制存储的内容，并将这些复制的内容放到因特网边缘。假定流经因特网的很大部分流量是存储的内容（Web页、MP3和视频），CDN能够明显地减轻ISP的流量负载和ISP之间对等接口的流量负载。此外，CDN给内容提供商提供一种有区别的服务：支付CDN服务费用的内容提供商能够更快、更有效地传输内容。我们将在后面的7.3节中学习CDN。
- 为了处理同时发送给上百万用户的实况流式流量（例如一个体育赛事），可以部署**多播覆盖网络**（multicast overlay network）。多播覆盖网络由散布在因特网上的用户主机和可能专用的服务器组成。这些主机、服务器和它们之间的逻辑链路共同形成一个覆盖网络，这个网络从源用多播（见4.7节）向上百万用户发送流量。和多播IP不同（它的多播功能是由路由器的IP层处理的），覆盖网络则是在应用层进行多播的。例如，源主机可以向3个覆盖服务器发送流；每个覆盖服务器可以将这些流转发到其他覆盖服务器和主机；这个过程继续下去，就在支撑IP网之上用路由器和主机创建了一个分发树。通过覆盖网络来多播受欢迎的实况流量，因特网上的总流量负载较之单播分发会进一步减小。

在“预留阵营”和“自由放任阵营”之间，仍然存在第三个阵营，即区分服务（Diffserv）阵营。这个阵营要在网络层和运输层做一些相对小的变化，在网络的边界（即用户和用户的ISP之间的接口）引入简单的收费和监管计划。这个思想引入了为数不多的流量类别（可能只有两种），每个数据报都属于其中的一种，根据数据报在路由器队列中的类别来提供不同等级的服务，并且用户根据发送到网络中的分组的类别进行收费。我们将在7.5节讨论区分服务。

表7-1对处理多媒体流量的这三种不同的方法（即充分利用尽力而为服务、区分QoS和确保QoS）进行了总结，并分别在7.3节、7.5节和7.6节进行了讨论。

表7-1 支持多媒体应用的三种方法

方法	分配的单元	确保	目前部署	复杂性	机制
充分利用尽力而为服务	无	无或软	到处	最少	应用层支持，CDN，过度装备
区分QoS	流的类型	无或软	一些	中等	监管，调度
确保QoS	各个流	软或硬，一旦某流准入	较少	高	监管，调度，呼叫准入和信令

7.1.4 音频和视频压缩

在音频和视频能够通过计算机网络传输之前，必须数字化和压缩。数字化的需求是很明显的：计算机网络传送比特，因此所有传输的信息必须被表示为比特序列。压缩是重要的，因为没有压缩的音频和视频消耗了大量存储空间和带宽；去除数字化音频和视频信号中内在的冗余，能够以若干数量级的幅度减小需要存储和传输的数据数量。举一个例子来说，一幅由1024 × 1024个像素组成的图片，每个像素用24比特编码（红、绿、蓝每颜色各8个比特），如果不压缩需要3 MB存储空间。在一个64 kbps的链路上发送这个图像需要7分钟。如果这个图像以普通的10：1的压缩率来压缩，存储需求降低到300 KB，传输时间也减少为十分之一。

音频和视频压缩的主题非常广泛。50多年来该领域的研究一直很活跃，现在有上百种关于音频和视频压缩的流行技术和标准。许多大学开设了有关音频和视频压缩的完整课程。因此我们这里仅对这个主题提供一个简要和总体的介绍。

1. 因特网中的音频压缩

一个连续变化的模拟音频信号（可能源自语音或音乐）通常按以下步骤转换为数字信号：

- 模拟音频信号首先以某种固定速率（例如每秒8000个样点）被采样。每个采样值是一个任意的实数。
- 然后每个采样值被“四舍五入”为有限个数值中的一个。这种操作被称为**量化**（quantization）。这些有限数值（称为量化值）通常是2的幂，例如256个量化值。
- 该量化值中的每一个由固定数量的比特来表示。例如，如果有256个量化值，那么每个值（因此每个采样）用一个字节来表示。每个采样都被转换为它的比特表示。所有采样的比特表示连接在一起就形成了该信号的数字表示。

举一个例子，如果一个模拟信号以每秒8000个样值采样，每个采样值被量化并用8比特表示，那么得到的数字信号的速率为64 kbps。这个数字信号能够转换回来（也就是解码），以形成一个模拟信号来播放。然而解码后的模拟信号通常和初始音频信号不同。通过增加采样速率和量化值的数量，解码信号可以近似初始的模拟信号。因此在解码信号的质量和数字信号存储空间及带宽需求之间要有一个明确的折衷。

我们刚才描述的基本编码技术称为**脉冲编码调制**（Pulse Code Modulation, PCM）。语音编码通常采用PCM，采样速率为每秒8000个样点，每个样点用8比特表示，得到64 kbps的速率。音频光盘（CD）也使用PCM，采样速率为每秒44 100个样点，每个样点用16比特表示；这样使得单声道速率为705.6 kbps，立体声速率为1.411 Mbps。

立体声音乐1.411 Mbps的比特速率超过了大多数接入速率，甚至语音的64 kbps也超过了拨号调制解调器用户的接入速率。由于这些原因，PCM编码的语音和音乐很少在因特网中使用。取而代之的是使用压缩技术来减小流的比特速率。流行的语音压缩技术包括GSM（13 kbps）、G.729（8 kbps）和G.723.3（6.4 kbps和5.3 kbps），以及大量的专用技术。接近CD质量的立体声音乐的一种通用压缩技术是MPEG 1第3层，通常称为MP3。MP3编码器通常压缩为96 kbps、128 kbps和160 kbps的速率，而且声音失真非常小。当一个MP3文件被分为几个片时，每个片仍然可以播放。这种没有首部的文件格式允许MP3音乐文件以流方式通过因特网（假设播放比特速率和因特网连接速率相匹配）。MP3压缩标准是复杂的，使用了心理声学遮蔽、冗余减小和比特积蓄缓冲技术。

2. 因特网中的视频压缩

视频是一个图像序列，图像通常以恒定的速率显示，例如每秒24或30幅图像。一个没有压缩的、数字编码的图像由像素阵列组成，每个像素被编码为一定数量的比特来表示亮度和颜色。在视频中有两种类型的冗余，它们都可以用来实现压缩。空间冗余是给定图像内部的冗余。例如，一个主要由空白组成的图像能够有效地压缩。时域冗余反映一幅图像和后续图像的重复。例如，如果一幅图像和后续图像完全一致，没有理由对后续图像再进行编码；在编码时简单地指示后续图像是完全一样的即可，这样效率更高。

MPEG压缩标准是最常用的压缩技术之一。它们包括用于CD-ROM质量视频（1.5 Mbps）的MPEG 1、高质量DVD视频（3~6 Mbps）的MPEG 2和面向对象视频压缩MPEG 4。针对图像压缩，MPEG标准从JPEG标准中吸取了很多有益的东西，它除了借鉴了JPEG开发的空间

冗余之外，还利用了图像间的时域冗余。H.261视频压缩标准在因特网中也是很常用的。此外还有大量专用方案，包括苹果公司的QuickTime和RealNetworks的编码器。

我们鼓励有兴趣学习更多有关音频和视频编码知识的读者去阅读[Rao 1996]和[Solari 1997]。[Crowcroft 1999]是一本关于多媒体网络的好书 。

历史事件

流式存储音频和视频：从RealNetworks到YouTube

RealNetworks是流式音频和视频的先驱，是首家将因特网音频推向主流的公司。它的初始产品RealAudio系统于1995年发行，该系统包括音频编码器、音频服务器和音频播放器。允许用户浏览、选择并按需从因特网播放流式音频，它迅速成为提供娱乐、教育和新闻内容的喜闻乐见的分发系统。

今天的流式音频和视频已经列于因特网中最为流行的服务了。不仅已经有太多的公司在提供流式内容，也有多种多样的不同服务器、播放器和协议技术在应用。一些令人兴奋的例子（到2007年止）包括：

- 来自RealNetworks的Rhapsody（狂想曲）：为用户提供了流式和下载订阅服务。Rhapsody使用它自己的专用客户机，客户机经HTTP从它的专用服务器获取歌曲。当一首歌曲经HTTP到达时，则通过Rhapsody客户机播放。访问下载的内容受到一种数字权限管理（DRM）系统的限制。
- MSN视频：用户以流式播放各种内容，包括国际新闻和音乐视频片段。视频通过流行的Windows媒体播放器（WMP）播放，几乎所有的Windows主机都可使用该播放器。WMP和微软服务器之间的通信使用专用的MMS（微软媒体服务器）协议进行，该协议通常尝试经RTSP/RTP传输流式内容；如果因为防火墙而出现故障，它尝试经HTTP取回内容。
- Muze：向零售商如BestBuy和Yahoo等提供音频样品服务。在这些零售商场选择的音乐样品实际上来自Muze，并通过WMP流式播放。Muze、Rhapsody、YouTube和许多其他流式内容提供商使用内容分发网络（CDN）分发它们的内容，如在7.3讨论的那样。
- YouTube：这个极为流行的共享视频服务使用了一种基于Flash的客户机（嵌入在Web页面中）。客户机和YouTube服务器之间的通信是通过HTTP进行的。

未来存储的是什么？今天大多数流式视频内容是低质量、以500 kbps或更低速率编码的。视频质量将无疑随着宽带和光纤到户的因特网接入变得无所不在而得到改善。并且非常有可能，我们的手持式音乐播放器将不再只是存储音乐，而是能用它按需做所有事，并从无线信道去做！

7.2 流式存储音频和视频

最近几年，音频/视频流已经变成了一种流行的应用并极大地消耗了网络带宽。在流式音频/视频中，客户机请求位于服务器中的压缩音频/视频文件。如我们马上要讨论的那样，这些服务器能够是普通Web服务器或是为音频/视频流式应用而定制的特殊流式服务器。一旦客户机请求，服务器通过向一个套接字发送文件向客户机发送一个音频/视频文件。尽管能够使用

TCP和UDP，今天大多数流式音频/视频流量由TCP传送。(通常将防火墙配置为阻挡UDP流量。此外，通过使用TCP，就具有了它的可靠交付能力，传送给客户机的整个文件没有丢包，允许未来可以从本地高速缓存中重新播放文件。) [Sripanidkulchai 2004]。一旦请求的音频/视频文件开始到达，客户机（通常）在几秒内开始呈现该文件。某些系统也提供用户交互性，例如在音频/视频中的暂停/继续和暂时跳转。在本节最后所讨论的**实时流式协议**（Real-Time Streaming Protocol, RTSP）是一个为用户提供交互能力的公共域协议。

用户经常通过一个Web客户机（也就是浏览器）来请求音频/视频流，而使用**媒体播放器**(media player) 如微软的Windows媒体播放器或Flash播放器来显示和控制音频/视频的播放。媒体播放器执行几项功能，其中包括：

- *解压缩*。为了节省磁盘存储空间和网络带宽，几乎总是压缩音频/视频。一个媒体播放器必须在播放时动态对音频/视频解压缩。
- *消除时延抖动*。分组时延抖动是同一个分组流中分组从源到目的的时延变化。因为音频和视频必须以它被记录时的同样时序来播放，接收方将把接收的分组缓存一小段时间来消除这种时延抖动。我们将在7.3节中详细研究这个主题。

7.2.1 通过Web服务器访问音频和视频

存储的音频/视频能够存储在Web服务器上，通过HTTP向客户机传送音频/视频；也可以存储在专用的音频/视频流式服务器上，通过HTTP协议或某些其他协议传送音频/视频。在本小节中，我们研究从Web服务器传送音频/视频，下一小节我们研究从流式服务器传送的情况。由于防火墙（参见第8章）通常允许HTTP流量通过而阻挡一些专用协议，所以通过HTTP传送流式多媒体成为流行方式。

首先考虑音频流的情况。当音频文件存储在Web服务器上时，该音频文件在服务器文件系统中是一个普通对象，如同HTML和JPEG文件一样。当用户要听该音频文件时，用户的主机和Web服务器之间建立一个TCP连接，并且为该对象发送一个HTTP请求。当收到一个请求，Web服务器在HTTP响应报文中封装这个音频文件，并将这个响应报文通过该TCP连接返回。如果视频的音频和视频部分存储在两个不同的文件中，视频的情况比较麻烦一些。音频和视频也可能交织在同一个文件中，这样只需要给客户机发送一个对象。为了简化我们的讨论，对视频的情况，我们假设音频和视频包含在一个文件中。

在许多经HTTP的流式音频/视频的实现中，客户机方功能被分为两个部分。浏览器的工作是请求**元文件**（metafile），元文件提供有关经HTTP流式发送的多媒体信息（例如一个URL或者编码类型，因此能够确定适当的媒体播放器）。该元文件则从浏览器传送给媒体播放器，后者的工作是联系HTTP服务器。HTTP服务器则经HTTP向媒体播放器发送多媒体文件。这些步骤如图7-1所示：

1）用户点击音频/视频文件的超链接。该超链接并非直接指向某音频/视频文件，而是指向一个元文件。该元文件包括实际音频/视频文件的URL。封装该元文件的HTTP响应报文包括了指示特定的音频/视频应用的内容类型首部行。

2）客户机浏览器分析响应报文的内容类型首部行，调用相关的媒体播放器，并将响应报文的整个体（即元文件）传递给媒体播放器。

3）媒体播放器直接和HTTP服务器建立TCP连接。媒体播放器为该音频/视频文件向TCP连接发送一个HTTP请求报文。该音频/视频文件在HTTP响应报文中发送给媒体播放器。媒体

播放器以流的形式播放该音频/视频文件。

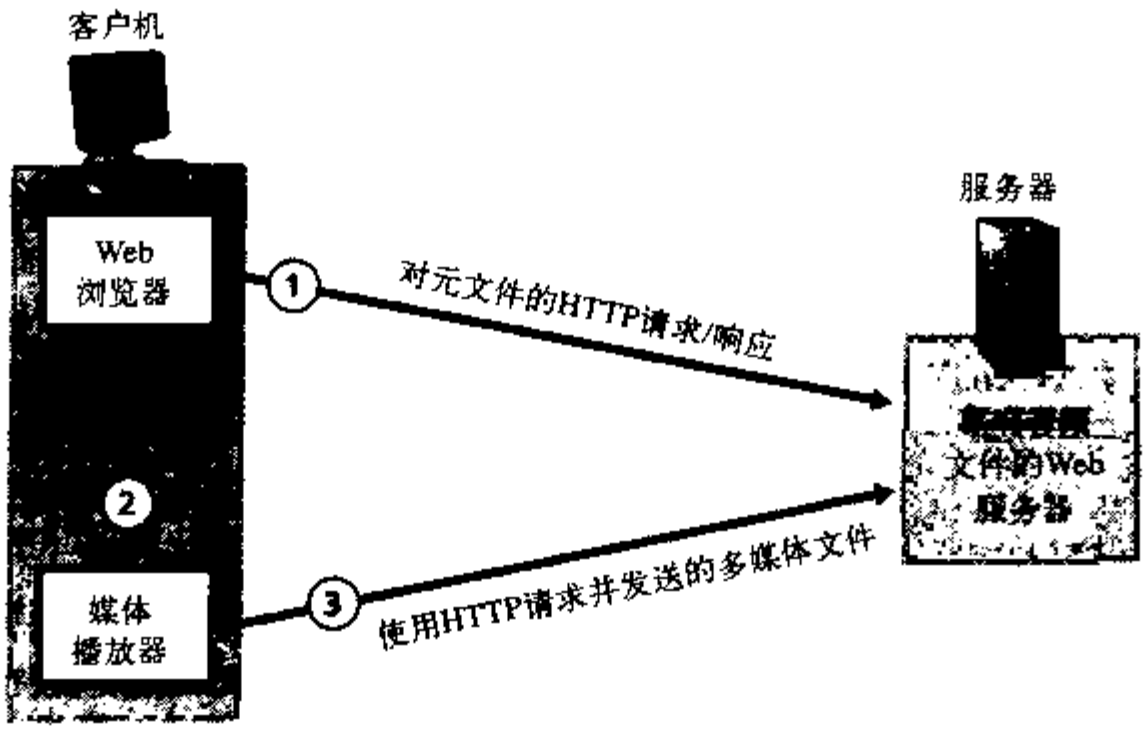


图7-1 Web服务器直接向媒体播放器发送音频/视频

其中的获得元文件的步骤的重要性是显然的。当浏览器看到文件的内容类型时，它可以调用合适的媒体播放器，从而使媒体播放器和服务器直接接触。

我们刚才学习了元文件如何能够使得媒体播放器与存储音频/视频文件的Web服务器直接通信。然而许多销售流式音频/视频产品的公司不推荐我们刚才描述的体系结构。它们推荐来自专用流式服务器的流式存储音频/视频，这些流式服务器为提供流式服务而进行了优化。

7.2.2 从流式服务器向助手应用程序发送多媒体

流式服务器可能是专用的流式服务器，例如RealNetworks和微软在市场上销售的那些类型，或者可能是公共领域流式服务器。通过流式服务器，音频/视频能够经HTTP/TCP发送，或使用可能比HTTP更适合音频/视频的应用层协议来传送音频/视频流，可以在UDP上发送。

这种体系结构需要两台服务器，如图7-2所示。一台服务器是HTTP服务器，用于Web页服务（包括元文件）。第二台服务器是**流式服务器**（streaming server），用于音频/视频文件服务。这两台服务器能够运行在同一个端系统内或者两个不同的端系统中。这种体系结构的工作步骤和上一小节描述的相似。然而，现在媒体播放器从流式服务器而不是从Web服务器中请求该文件，并且现在媒体播放器和流式服务器可以通过它们自己的协议进行交互了。这些协议可以允许用户和音频/视频流进行大量的交互。

在图7-2的体系结构中，从流式服务器向媒体播放器交付音频/视频有很多种选择。下面列出部分选择：

1）音频/视频通过UDP以等于接收方排空（drain）速率（就是音频/视频的编码速率）的恒定速率发送。例如，如果以使用速率为13 kbps的GSM压缩音频，那么服务器以13 kbps速率计时输出该压缩音频文件。客户机一旦从网络收到压缩的音频/视频，就解压缩这个音频/视频，并播放之。

2）这和第一个选项相同，但是为了消除网络引入的时延抖动，媒体播放器延迟2～5 s播放。如图7-3所示，通过将从网络中收到的压缩媒体放置到**客户机缓冲区**（client buffer）中，客户机完成了这个任务。一旦客户机预取了几秒钟的媒体，它开始排空缓冲区。对于这个以

及前一个选项，文件速率$x(t)$等于排空速率d，除非有丢包，这时$x(t)$才会瞬间小于d。

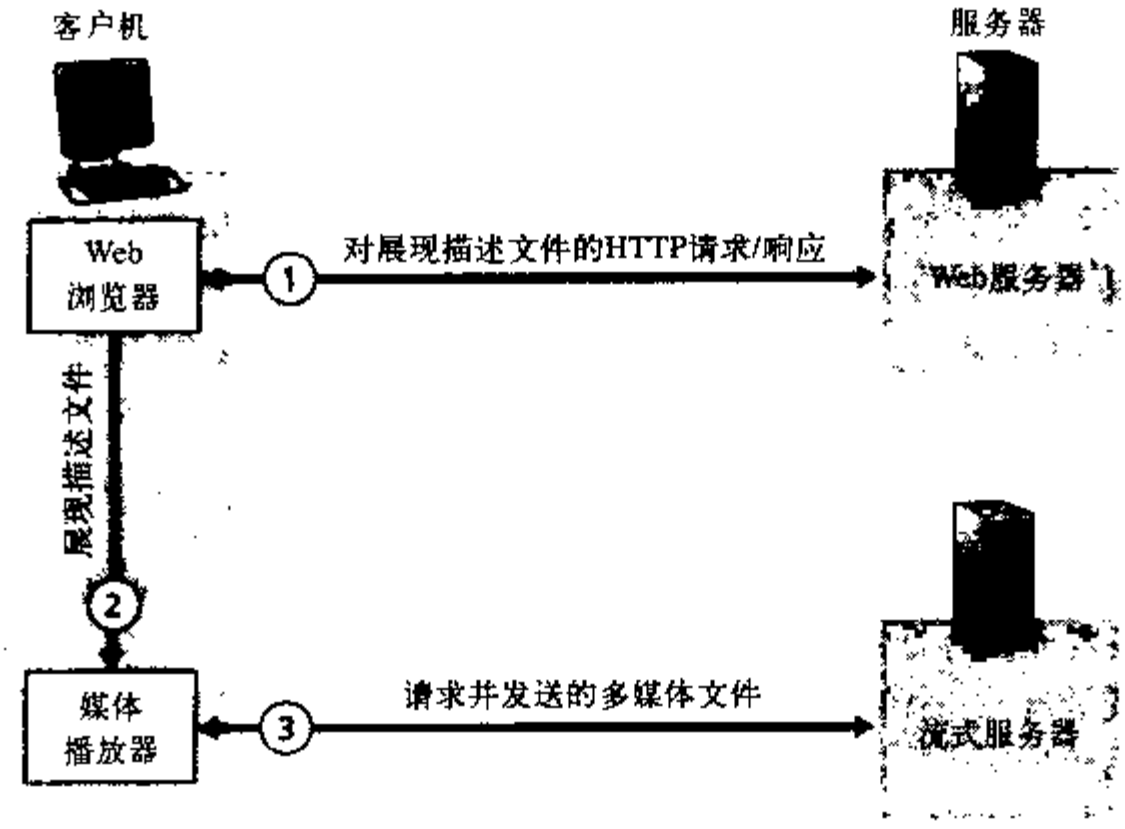


图7-2 从流式服务器向媒体播放器传输流

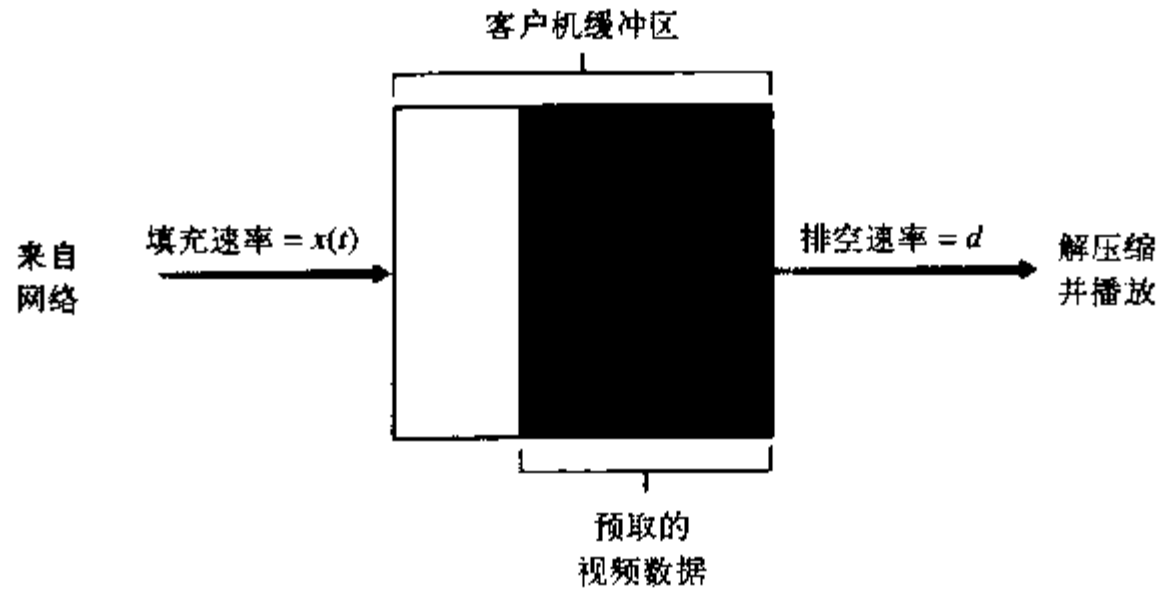


图7-3 客户机缓存区以速率$x(t)$填充，以速率d排空

3）通过TCP发送媒体。服务器尽可能快地将媒体文件压入TCP套接字；客户机（即媒体播放器）尽可能快地从TCP套接字中读取；并将压缩的视频放置到媒体播放器缓冲区中。在初始的2～5 s延时后，媒体播放器以速率d从它的缓冲区中读取，并转发压缩的媒体来解压缩和播放。

因为TCP重传丢失的分组，因此它有潜力提供比UDP更好的声音质量。另一方面，由于TCP拥塞控制和窗口流量控制，填充速率$x(t)$现在随着时间波动。实际上，在丢包后，TCP拥塞控制可能在很长一段时间内将瞬间速率减小到小于d。这样可能清空客户机缓冲区（这种过程称为**饥饿**），并在客户机的音频/视频流输出中引入不理想的暂停。[Wang 2004]显示了当平均TCP吞吐量大致是媒体比特率的两倍时，TCP流式播放导致了最小的饥饿和低起始时延。

对于第三种选择，$x(t)$的行为将非常依赖客户机缓冲区的大小（不要和TCP接收缓冲区搞混淆）。如果这个缓冲区足够大，可以容纳整个媒体文件（可能在磁盘容量以内），那么TCP将利用该连接可用的所有瞬间带宽，使得$x(t)$能够变得远大于d。如果$x(t)$长时间远大于d，那么大部分媒体被预取到客户机，客户机随后就不太可能出现饥饿现象。另一方面，如果客户

机缓冲区很小，那么$x(t)$将在排空速率d附近波动。在这种情况下，客户机出现饥饿现象的危险性就大得多。

7.2.3 实时流协议

许多因特网多媒体用户（特别是手中握有电视遥控器长大的人）希望控制连续媒体的播放，包括暂停播放、将播放重新定位在未来或者过去的时间点、快进可视播放、快退可视播放等等。这些功能类似于用户看DVD录像时使用的DVD播放器，或者听音乐CD时使用的CD播放器所具有的功能。为了允许用户控制播放，媒体播放器和服务器需要一个协议来交换播放控制信息。在RFC 2326中定义的实时流协议（RTSP）就是这样一个协议。

在讨论RTSP的细节之前，我们先指出RTSP不能做什么。

- RTSP没有定义用于音频和视频的压缩方案。
- RTSP没有定义音频和视频在网络传输中是怎样封装在分组中的；流式媒体的封装可以通过RTP或者专用协议来提供（RTP在7.4节中讨论）。例如，RealNetworks的音频/视频服务器和播放器使用RTSP来互相发送控制信息，但是媒体流本身能够封装在RTP分组或者某些专用数据格式中。
- RTSP不限制流式媒体如何传输，它可以在UDP或者TCP上传输。
- RTSP不限制媒体播放器如何缓冲音频/视频。音频/视频可能在它一到达客户机就开始播放，也可能在延迟几秒之后播放，或者完全下载下来再播放。

既然RTSP不做上述任何事情，那它干什么呢？RTSP允许媒体播放器控制媒体流传输。如上所述，控制动作包括暂停/继续、播放重定位、快进和快退。RTSP是一个**带外协议**（out-of-band protocol）。特别是RTSP报文在带外发送，而媒体流的分组结构没有被RTSP定义，它被认为是“带内”的。RTSP报文和媒体流使用不同的端口号，前者使用端口号544。RTSP规范[RFC 2326]允许RTSP报文通过TCP或者UDP发送。

2.3节讲过文件传输协议（FTP）也使用带外的概念。特别是FTP使用两对客户机/服务器套接字，每对有自己的端口号：一个客户机/服务器套接字支持传输控制信息的TCP连接；另一个客户机/服务器套接字支持真正传输文件的TCP连接。RTSP信道在很多方面和FTP的控制信道相似。

我们现在看一个简单的RTSP例子，它在图7-4中描述。Web浏览器首先向Web服务器请求一个展现描述文件。这个展现描述文件有一些指向几个连续媒体文件的引用，也有这些连续媒体文件的同步指示。每个到连续媒体文件的引用都以URL的方法rtsp:// 开始。下面我们提供一个样本展现文件，它是从[Schulzrinne 1997]改编的。在这个展现中，音频和视频流被并行播放，并且按片同步（作为同一组的部分）。对于这个音频流，媒体播放器能够在两个音频记录（一个低保真度的记录和一个高保真度的记录）之间选择（切换）。（这个文件的格式类似于SMIL[SMIL 2004]，它被许多流式产品用来定义同步的多媒体展现。）

Web服务器在一个HTTP响应报文中封装该展现描述文件，并且把这个报文发送给浏览器。当浏览器收到这个HTTP响应报文时，浏览器根据这个报文的内容类型字段调用媒体播放器（也就是助手应用程序）。这个展现描述文件使用URL方法rtsp://来包括对媒体流的引用，如上例所示的那样。如图7-4所示，播放器和服务器然后互相发送一系列RTSP报文。播放器发送一个RTSP SETUP请求，服务器用一个RTSP OK报文来响应。播放器发送一个RTSP PLAY请求，比方说，要求一个低保真度的音频，服务器用一个RTSP OK报文来响应。在这个时候，流式服务器将低保真度音频送入它自己的带内信道。稍后，媒体播放器发送一个RTSP PAUSE请求，服务器用RTSP OK报文响应。当用户结束时，媒体播放器发送RTSP TEARDOWN请求，服务

器用RTSP OK响应来确认。

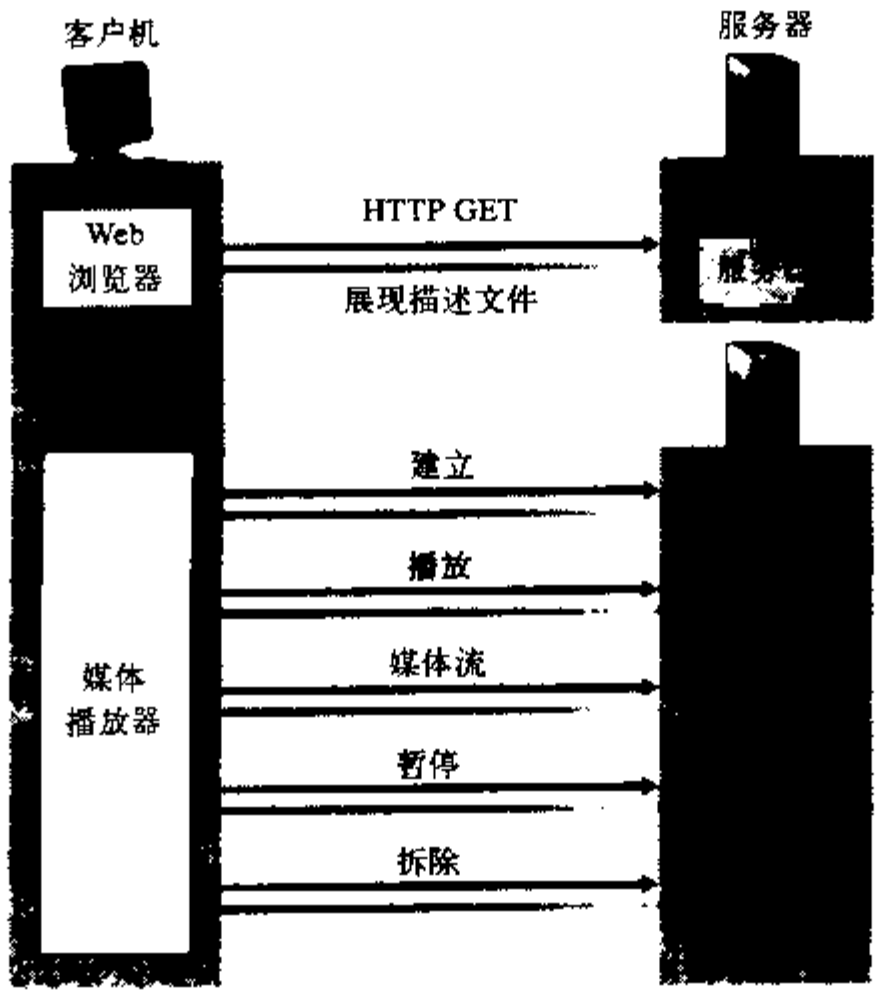


图7-4 客户机与服务器之间使用RTSP交互

```
<title>Twister</title>
<session>
   <group language=en lipsync>
      <switch>
         <track type=audio
                e="PCMU/8000/1"
                src="rtsp://audio.example.com/twister/audio.en/lofi">
         <track type=audio
                e="DVI4/16000/2" pt="90 DVI4/8000/1"
                src="rtsp://audio.example.com/twister/audio.en/hifi">
            </switch>
         <track type="video/jpeg"
                src="rtsp://video.example.com/twister/video">
   </group>
</session>
```

现在我们简要地看一下实际RTSP报文。下面是客户机（C:）和服务器（S:）之间一个简化了的RTSP会话的例子：

```
C: SETUP rtsp://audio.example.com/twister/audio RTSP/1.0
   Cseq: 1
   Transport: rtp/udp; compression; port=3056; mode=PLAY
S: RTSP/1.0 200 OK
   Cseq: 1
   Session: 4231
C: PLAY rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
   Range: npt=0-
   Cseq: 2
   Session: 4231
S: RTSP/1.0 200 OK
   Cseq: 2
   Session: 4231
C: PAUSE rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
```

```
   Range: npt=37
   Cseq: 3
   Session: 4231
S: RTSP/1.0 200 OK
   Cseq: 3
   Session: 4231
C: TEARDOWN rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
   Cseq: 4
   Session: 4231
S: RTSP/1.0 200 OK
   Cseq: 4
   Session: 4231
```

注意HTTP和RTSP之间的相似性很有意义。所有的请求和响应报文都是采用ASCII文本格式，客户机使用标准化的方法（SETUP、PLAY、PAUSE等等），服务器用标准化的应答码来响应。然而一个重要的区别是RTSP服务器一直跟踪每个正在进行的RTSP会话中的客户机状态。例如，服务器记录客户机是位于初始化状态、播放状态还是暂停状态（见本章的编程习题）。作为每个RTSP请求和响应的一部分，会话号和序号帮助服务器跟踪会话状态。会话号在整个会话中不变；客户机每次发送一个新的报文就增加序号；服务器用会话号和现在的序号来回显。

如这个例子中所示，客户机通过SETUP请求发起这个会话，提供要被流式发送的文件的URL和RTSP版本。建立报文中包括媒体所要发送到的客户机端口号。建立报文也指示媒体应该在UDP上使用RTP分组化协议（将在7.4节中讨论）发送。注意在这个例子中，播放器没有选择播放整个展现，但只播放该展现低保真度的部分。

RTSP协议实际上能够做的工作要比这个简单介绍所描述的要多得多。特别是RTSP具有允许客户机向服务器发送流的设施（例如为了记录）。RTSP已经被音频/视频流领域的业界领袖之一RealNetworks所采用。Henning Schulzrinne制作了一个有关RTSP的可用Web页[Schulzrinne-RTSP 2007]。

在本章最后可看到一个编程作业，要求建立一个支持RTSP协议的视频流系统（包括服务器和客户机）。该作业涉及在客户机真正构建和发送RTSP报文的代码。这个作业提供了RTSP服务器代码，该代码解析了RTSP报文并构建合适的响应。我们鼓励有兴趣想更深入理解RTSP的读者完成这个有趣的作业。

7.3 充分利用尽力而为服务

因特网的网络层协议IP提供了尽力而为服务。这就是说该服务尽最大努力将每个数据报尽可能快地从源传送到目的地。然而，对于单个分组的端到端时延范围，或者在这个分组流中的分组时延抖动和分组丢失（或者说丢包）范围，它不做任何承诺。缺乏对时延和分组时延抖动的保证，对设计实时多媒体应用（例如因特网电话和实时视频会议）提出了巨大的挑战，这些应用实际上对分组时延、时延抖动和丢失是很敏感的。

在本节，我们将讨论如何加强经尽力而为网络的多媒体应用的性能，并介绍几种改进性能的方式。我们的重点将放在应用层技术上，即那些不要求在网络核心或者甚至在端系统的运输层中做任何改变的技术。我们首先描述丢包、时延和时延抖动对多媒体应用的影响。然后涉及从这些损伤恢复的技术。随后描述内容分发网络和为避免这样的损伤能够首先使用的过度装备资源。

7.3.1 尽力而为服务的限制

我们前面提到过尽力而为服务可能导致丢包、过大的端到端时延和分组时延抖动。我们

现在更仔细地研究这些问题。为了使讨论具体化，我们在**因特网电话应用**（Internet phone application）的环境下讨论这些机制。该情况与实时视频会议应用[Bolot 1994]类似。

在因特网电话例子中，交谈者产生一个由话音突峰期和静默期交替组成的音频信号。为了节省带宽，我们的因特网电话应用只在话音突峰期间产生分组。发送方在一个话音突峰期以每秒8000字节的速率产生字节，且每20 ms将字节汇聚成块。这样，一个块中的字节数为(20 ms)×(8000字节/s)=160字节。给每个块附加一个特殊的首部（其内容在下面讨论）。在一个UDP报文段中封装这个块和首部，经呼叫到达套接字接口。因此在一个话音突峰期中，每20 ms发送一个UDP报文段。

如果每个分组都到达接收方，并且有一个很小的恒定的端到端时延，那么在话音突峰期中分组每隔20 ms就能周期性地到达接收方。在这种理想的情况下，只要每个块一到达，接收方就能直接播放它。但是不幸的是，某些分组可能丢失，大多数分组没有相同的端到端时延，即使在一个轻度拥塞的因特网中也是如此。由于这个原因，接收方必须更仔细地判断什么时候播放一个块，以及如何处理一个丢失块。

1. 丢包

考虑由因特网电话应用产生的一个UDP报文段。这个UDP报文段封装在IP数据报中。当数据报在网络中徘徊时，为了接入“出链路”它要经过路由器的缓冲区（即队列）。从发送方到接收方的路径上，可能有一个或多个缓冲区是满的，不能接纳该IP数据报。在这种情况下，这个IP数据报就被丢弃了，永远不会到达接收方的应用程序。

通过TCP而不是UDP发送分组可以消除丢失。前面讲过TCP重传那些没有到达目的地的分组。然而，重传机制对于诸如因特网电话这样的交互实时音频应用，通常被认为是不可接受的，因为它们增加了端到端时延[Bolot 1996]。此外，当丢包后，由于TCP拥塞控制，发送方的传输速率可能减小到低于接收方的排空速率。这可能会对接收方的语音可理解度产生严重影响。由于这些原因，几乎所有现有的因特网电话应用运行在UDP上，而不必考虑重传丢失的分组。[Baset 2006]报告称Skype就使用UDP，除非用户位于阻碍UDP报文段的NAT或防火墙之后（这时使用TCP）。

但是分组的丢失并不一定会造成人们想象中的灾难。事实上，根据语音编码和传输方式的不同，以及接收方隐藏丢包的方式，1%～20%的丢包率是可以忍受的。例如，前向纠错（FEC）有助于隐藏丢包。我们在后面可以看到，通过使用FEC，将冗余信息和初始信息一起传输，以便能够从冗余信息中恢复一些丢失的初始数据。无论如何，如果发送方和接收方之间的一段或多段链路严重拥塞，丢包率超过10%～20%（尽管这样的速率在装备良好的网络中很少看到），那么无论采取何种措施都无法获得可以接受的声音质量。显然，尽力而为服务有它的局限性。

2. 端到端时延

端到端时延（end-to-end delay）是以下因素的总和：路由器中的传输、处理和排队时延，链路中的传播时延和端系统的处理时延。对于高度交互的音频应用，例如因特网电话，听电话的人对于小于150 ms的端到端时延是觉察不到的；在150 ms和400 ms之间的时延能够接受，但是不够理想；超过400 ms的时延可能严重妨碍语音谈话的交互性。因特网电话应用程序的接收方通常忽略时延超过特定阈值（例如超过400 ms）的任何分组。因此时延超过该阈值的分组等效于丢弃。

3. 分组时延抖动

端到端时延的一个关键成分是路由器中随机排队时延。由于网络中这些时延是可变的，

分组从源产生到它在接收方被收到的时间对于不同的分组可能会有波动，这个现象称为**时延抖动**（jitter）。

举一个例子，考虑在因特网电话应用中在一个话音突峰期里两个连续的分组。发送方在发送第一个分组20 ms之后发送第二个分组。但是在接收方，这两个分组之间的间隔可能变得大于20 ms。为了说明这一点，假设第一个分组到达路由器的一个几乎为空的队列，但是恰好在第二个分组到达该队列之前，从其他源来的大量分组也到达了这个队列。因为在这个路由器中第一个分组经受了很小的排队时延，而第二个分组经受了较大的排队时延，第一和第二个分组的时间间隔变得超过了20 ms。连续两个分组的间隔可能也会小于20 ms。为了观察这种情况，再次考虑一个话音突峰期里两个连续的分组。假设第一个分组加入一个有大量分组的队列队尾，并且第二个分组到达这个队列时其他源的分组尚未到达这个队列。在这种情况中，两个分组发现它们自己在队列中互相紧挨着。如果在路由器的出链路传输一个分组所需时间小于20 ms，则第一个分组和第二个分组的间隔就变得小于20 ms了。

这种情况可以与在路上开车类比。假设你和你的朋友每人驾驶一辆车从圣地亚哥到凤凰城，而且你们有类似的驾驶风格，并都以交通允许的100 km/h的速度驾驶。最后，假设你的朋友在你之前一个小时出发。那么根据干扰的车流量的不同，你可能在你的朋友之后大于或小于一个小时到达凤凰城。

如果接收方忽略了时延抖动的存在，在该音频块一到达就播放，那么在接收方产生的音频质量很容易变得不可懂。幸运的是，时延抖动通常可以通过使用**序号**（sequence number）、**时间戳**（timestamp）和**播放时延**（playout delay）来消除，如下面所讨论的内容。

7.3.2 在接收方消除音频的时延抖动

对于诸如因特网电话或者音频点播这样的音频应用因特网，接收方应该在随机网络时延抖动存在的情况下尝试提供音频块的同步播放；视频应用通常具有类似的要求。这经常通过结合下面3种机制来完成：

- 为每个块规定一个**序号**（sequence number）。发送方为每个产生的分组的序号加1。
- 为每个块规定一个**时间戳**（timestamp）。发送方用每个块产生的时刻为它加上时间印记。
- 在接收方**延迟播放**（delaying playout）音频块。接收的音频块的播放时延必须足够长，以便大多数分组在它们的预定播放时间之前被接收。这个播放时延可能在整个音频会话期间是固定的，或者在音频会话生命期中可以自适应地变化。在预定播放时间之前还没有到达的分组被认为已经丢失并被遗忘；如上面所说的，接收方可以使用某种形式的语音内插法来试图隐藏丢失。

我们现在讨论当这三种机制结合时，它们如何能够减轻甚至消除时延抖动的影响。我们研究两种播放策略：固定播放时延和自适应播放时延。

1. 固定播放时延

使用固定播放时延策略，接收方试图在块产生q ms之后播放它。因此如果一个块在时刻t打上时间戳，接收方在时刻$t+q$播放这个块（假设这个块在那个时间已经到达）。在预定播放时间之后到达的分组将被丢弃，并被认为已经丢失。

q选择什么值为好呢？因特网电话能够支持高达约400 ms的时延，尽管使用更小的q值可以获得更令人满意的交互体验。另一方面，如果q比400 ms小得多，那么由于网络引入的分组时延抖动会使许多分组可能错过了它们的预定播放时间。概括地说，如果端到端时延经常发生大的变化，用一个大的q更好；另一方面，如果时延很小并且时延变化也很小，用一个较小

的、可能小于150 ms的q更好。

在图7-5中举例说明了播放时延和丢包之间的折衷。该图表示了对于单个话音突峰期分组产生和播放的时间。考虑了两种不同的初始播放时延。如最左边的阶梯所示，发送方以规则的间隔（比方说每20 ms）产生一组分组。在这个话音突峰期的第一个分组在时间r被接收到。如图所示，由于网络时延抖动，后续分组的到达间隔是不均匀的。

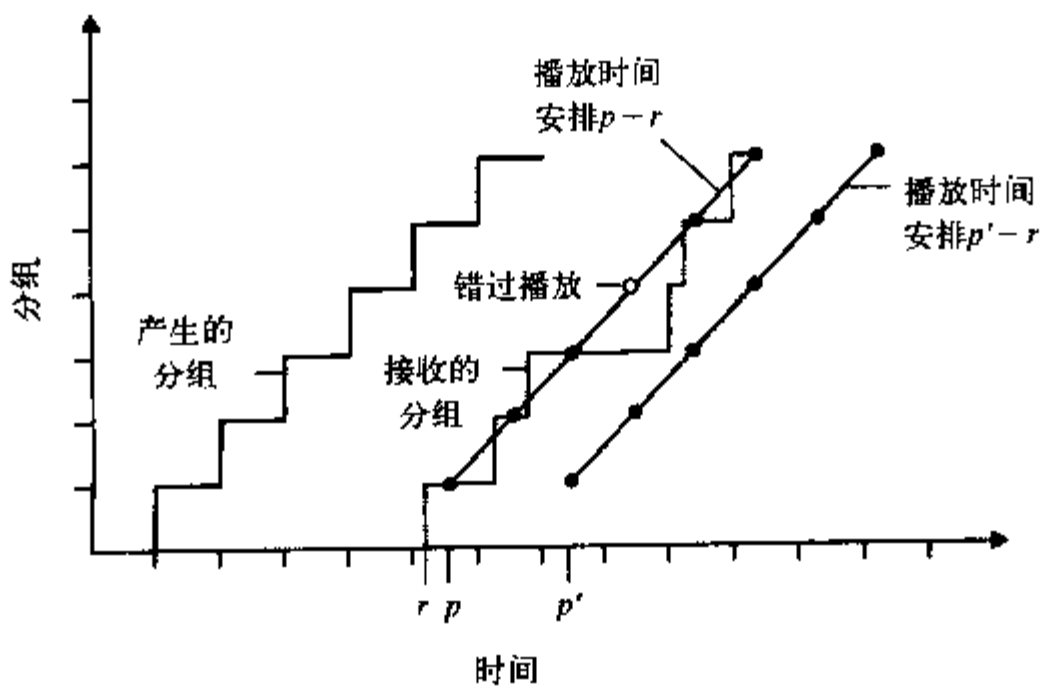


图7-5 不同的固定播放时延情况下的丢包

对于第一个播放时间安排，固定的初始播放时延设置为$p-r$。使用这个方案，第四个分组没有在它的预定播放时间到达，接收方认为它丢了。对于第二个播放时间安排，固定的初始播放时延设置为$p'-r$。对于这个方案，所有的分组都在它们的预定播放时间之前到达，因此没有丢失。

2. 自适应播放时延

上面的例子显示了当使用固定播放时延来设计一个播放策略时所产生的重要时延与丢包的折衷。若初始的播放时延设置得比较大，大多数分组能在它们的截止时间内到达，因此存在的丢失可忽略不计；然而，对于如因特网电话这样的交互服务，长时延即使不是让人无法忍受，也至少令人厌恶。在理想情况下，我们希望在满足丢包率低于一定百分比的限制下，使得播放时延最小化。

处理这种折衷的自然方法是估计网络时延和网络时延变化，并且在每个话音突峰期的开始相应地调整播放时延。在话音突峰期开始时播放时延的自适应调整将导致发送方的静默期的压缩和拉长；然而，静默期的少量压缩和拉长在谈话中是不易觉察的。

根据[Ramjee 1994]，我们现在描述一种接收方可以用于自适应调整它的播放时延的通用算法。为此，令

t_i = 第i个分组的时间戳 = 该分组在发送方产生的时间

r_i = 分组i被接收方接收的时间

p_i = 分组i在接收方播放的时间

第i个分组的端到端网络时延是r_i-t_i。由于网络时延抖动，这个时延在不同的分组之间会发生变化。令d_i表示接收到第i个分组时的平均网络时延的估计值。这个估计值根据下式的时间戳来构造：

$$d_i = (1 - u)d_{i-1} + u(r_i - t_i)$$

式中u是一个固定的常数（例如，$u = 0.01$）。这样，d_i是观察到的网络时延$r_1 - t_1$，…，$r_i - t_i$的一个平滑均值。这个估计值为最近观察到的网络时延设置了比过去一段时间观察到的网络时延有更大的权重。这种估值的形式不应该是完全陌生的；相似的思想在第3章讨论预测TCP往返时间时就使用过。令v_i表示与估计平均时延的估计平均时延绝对偏差。这个估计值也可从时间戳中构建：

$$v_i = (1 - u)v_{i-1} + u|r_i - t_i - d_i|$$

为每个接收的分组计算估计值d_i和v_i，虽然它们仅能用于为任何话音突峰期的第一个分组确定播放点。

一旦计算完了这些估计值，接收方为分组播放应用下列的算法。如果分组i是一个话音突峰期的第一个分组，它的播放时间p_i计算如下：

$$p_i = t_i + d_i + Kv_i$$

这里K是一个正的常数（例如$K=4$）。Kv_i项的目的是给将来设置足够大的播放时间，以便话音突峰期中只有一小部分到达的分组由于迟到而丢失。在一个话音突峰期中任何后续分组的播放点被计算为对于这个话音突峰期的第一个分组播放时间点的偏移。特别是，令

$$q_i = p_i - t_i$$

表示从话音突峰期的第一个分组产生到它播放的时间长度。如果分组j也属于这个话音突峰期，它播放的时刻是

$$p_j = t_j + q_i$$

刚才描述的算法在假设接收方能够辨明一个分组是否是话音突峰期中的第一个分组的情况下是很有意义。如果没有丢包，那么通过比较第i个分组和第（$i-1$）个分组的时间戳，接收方能够判定分组i是否是话音突峰期的第一个分组。事实上，如果$t_i - t_{i-1} > 20$ ms，那么接收方知道第i个分组开始了一个新的话音突峰期。但是现在假设偶尔有丢包出现。在这种情况下，当两个分组属于同一个话音突峰期时，在目的地收到的两个连续的分组可能有相差超过20 ms的时间戳。这时，序号就有用了。接收方可以使用序号来判定时间戳相差大于20 ms是由一个新的话音突峰期引起的还是由于丢包而导致的。

3. 流式存储音频和视频

我们用几句有关流式存储音频和视频的话来总结本节。流式存储音频和视频应用也通常使用序号、时间戳和播放时延来缓解甚至消除网络时延抖动的效应。然而，在实时交互音频/视频和流式存储音频/视频之间有重要的差异。特别是，流式存储音频/视频能够容忍很大的时延。实际上，当一个用户请求一个音频/视频片段时，该用户可能发现在播放开始之前等待5秒或更长是可接受的。大多数用户在交互性动作如在媒体流中瞬时跳转之后，能够容忍类似时延。这种对时延的更大容忍为应用研发者在设计存储媒体应用程序时带来更大的灵活性。

7.3.3 从丢包中恢复

我们已经讨论了一些有关因特网电话应用如何处理分组时延抖动的细节。我们现在简要地描述在存在丢包的情况下几种试图保持可接受音频质量的方案。这些方案称为**丢包恢复方案**（loss recovery scheme）。这里我们定义了广义的丢包：如果某分组不能到达接收方或者如

果在它的预定播放时间之后才到达，该分组则丢失。我们的因特网电话例子将再次用作描述丢包恢复方案的环境。

如在本节开始提到的那样，在诸如因特网电话等交互式实时应用中，重传丢失的分组通常是不适合的。事实上，重传一个已经错过了播放截止时间的分组绝对没有意义。重传一个在路由器队列溢出的分组通常不能完成得足够快。由于这些考虑，因特网电话应用通常使用某种类型的丢包预期方案。两种类型的丢包预期方案是**前向纠错**（Forward Error Correction, FEC）与**交织**（interleaving）。

1. 前向纠错

FEC的基本思想是给初始的分组流增加冗余信息。这或多或少地增加了音频流的传输速率，以此为代价，这些冗余信息可以用来重建一些丢分分组的近似或者准确的版本。根据文献[Bolot 1996]和[Perkins 1998]，我们现在概括了两种简单的FEC机制。第一种机制是在每发送n个块之后发送一个冗余编码的块。这个冗余块通过异或n个初始块来获得[Shacham 1990]。以这种方式，如果这$n+1$个分组的组中任何一个丢包，接收方能够完全重建丢失的分组。但是，如果这一组中有两个或更多丢包，接收方则无法重建丢失的分组。通过保持组的长度$n+1$比较小，当丢失不是很多时，大部分丢失分组都可以恢复。然而组的长度越小，音频流增加的传输速率相对就越多。特别是若传输速率以$1/n$因子增加的话，例如$n=3$，则传输速率增加33%。况且，这个简单的方案增加了播放时延，当接收方在能够开始播放之前，必须等待收到整个组的分组。对于有关FEC在多媒体传输上如何工作的更实际的细节参见[RFC 2733]。

第二个FEC机制是发送一个较低分辨率的音频流作为冗余信息。例如，发送方可能创建一个标称的音频流和一个相应的低分辨率、低比特率的音频流。（这个标称流可能是一个64 kbps的PCM编码，这个较低质量的流可能是一个13 kbps的GSM编码。）这个低比特率流被认为是冗余信息。如图7-6所示，发送方通过从这个标称流中取出第n个块并附加上第（$n-1$）个块的冗余信息，以构建第n个分组。以这种方式，只要没有连续分组的丢失，接收方都可以通过播放和后续分组一起到达的低比特率编码块来隐藏丢失。当然，低比特率块比标称块的质量要低。然而，在一个流主要是由高质量块组成、偶尔出现低质量块并且没有丢失的块的情况下，其整体的音频质量良好。注意到在这种方案中，接收方在播放前只需接收两个分组，因此增加的时延小。况且，如果低比特率编码比标称编码少得多，那么传输速率的额外增加不大。

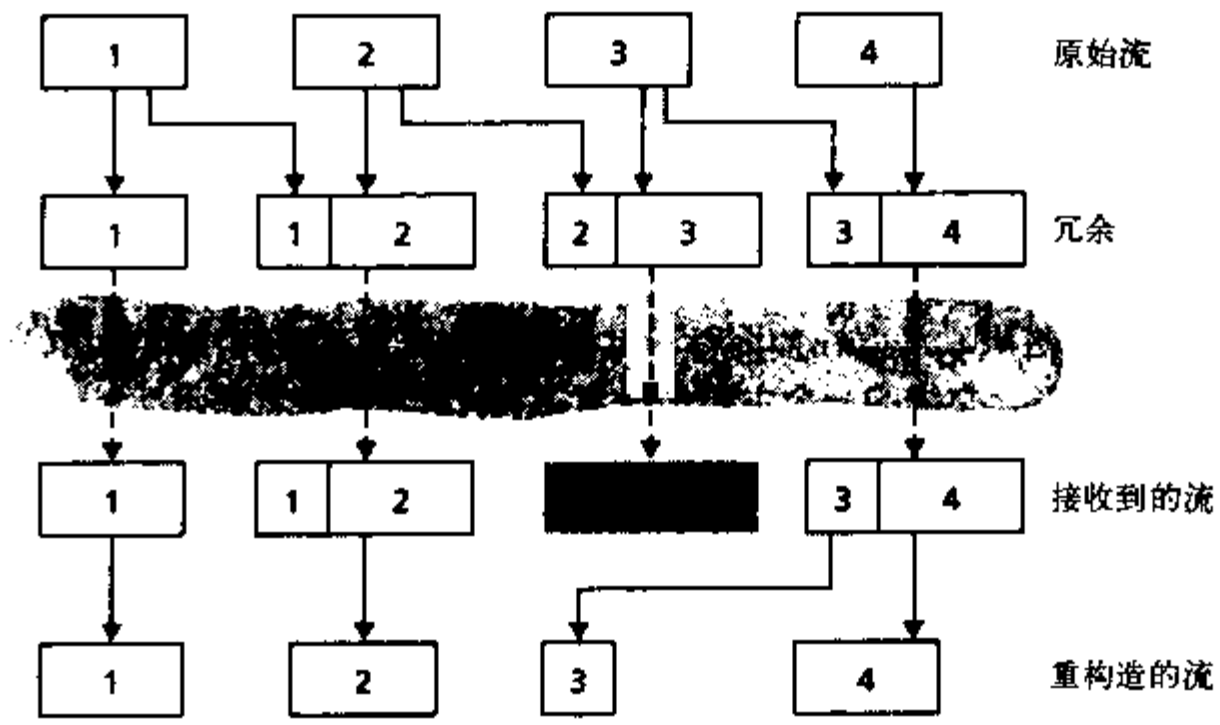


图7-6 捎带低质量的冗余信息

为了对付连续丢失，可以使用一个简单的修正方案。发送方不再仅为第n个标称块附加上第（$n-1$）个低比特块，而是附加上第（$n-1$）个和第（$n-2$）个低比特率块，或者附加第（$n-1$）个和第（$n-3$）个低比特率块等等。通过给每个标称块附加上更多低比特率块，接收方的音频质量在各种恶劣的尽力而为服务环境下变得可以接受。另一方面，附加的块增加了传输带宽和播放时延。

RAT [RAT 2007]是使用FEC的因特网电话应用，其文档很详细。如上面所述，它们能够与标称音频流一起传输低质量音频流。也可以参见[Rosenberg 2000]。

2. 交织

作为冗余传输的另一种替代方案，因特网电话应用可以发送交织的音频。如图7-7所示，发送方在发送之前对音频数据单元重新排序，使得最初相邻的单元在传输流中以一定距离分离开来。交织可以减轻丢包的影响。例如，如果每个单元长5 ms，块是20 ms（也就是每个块4个单元），那么第一个块可能包含1、5、9和13单元；第二个块可能包含2、6、10和14单元等等。图7-7表示了一个交织流的单个丢包导致重建流中的多个小间隙，这与在非交织流中将会导致单个大间隙形成对照。

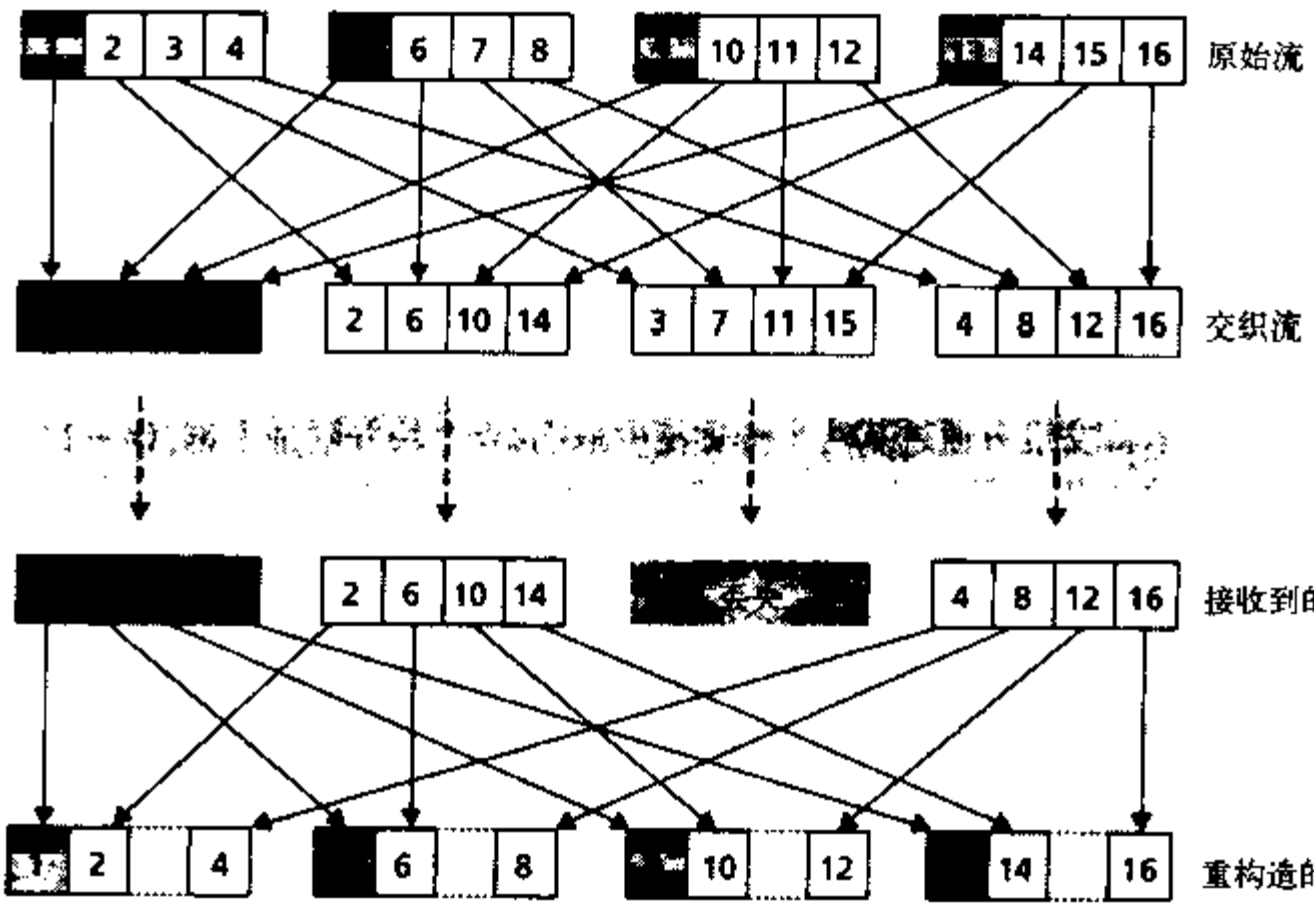


图7-7 发送交织音频

交织能够明显地提高音频流可感觉到的质量[Perkins 1998]。它的开销也较低。交织明显的缺点是增加了时延。这限制了它在如因特网电话这样的交互式应用中的使用，然而它能够很好地处理流式存储音频。交织的一个主要优点是它不增加流的带宽需求。

3. 基于接收方的对受损音频流的修复

基于接收方的恢复方案试图为丢失的分组产生一个和初始值相似的替代物。正如在[Perkins 1998]中讨论的那样，因为音频信号（特别是语音）呈现出的大量的短期自相似性，该方案是可行的。同样，这些技术适合工作于相对小的丢包率（低于15%）和小分组（4～40 ms）情况下。当丢失长度接近音素的长度（5~100 ms）时，这些技术就失效了，因为整个音素可能被听者错过。

基于接收方恢复的最简单的形式也许是分组重复。即用在丢失之前刚到达的分组的拷贝来代替丢失的分组。这种方法的计算复杂度低，并且工作得相当好。基于接收方恢复的另一种形式是内插法，它使用在丢失之前和之后的音频内插形成一个合适分组来隐藏丢失。内插法比分组重复稍微好一些，但是显然需要更高的计算强度[Perkins 1998]。

7.3.4 在今天的因特网中分发多媒体：内容分发网络

随着视频流速率从数百kbps的低分辨率视频到几Mbps的DVD视频，按需传输流式存储视频流到大量地理分布的用户看起来是一项令人畏惧的挑战。对于每个客户机请求，最简单的方法是在单台服务器上存储视频并从一台视频服务器（或服务器场）直接按流传输到客户机，如我们在7.2节讨论的那样。但这种解决方案存在两个明显的问题。首先，因为某客户机可能离服务器很远，服务器到客户机的分组可能通过许多ISP传递，增加较大时延和丢包率的可能性。第二，如果该视频非常流行，该视频可能通过相同的ISP（和通过相同的通信链路）发送许多次，因而消耗了大量带宽。在第2章中我们讨论了使用高速缓存来缓解这些问题。尽管我们根据传统的Web内容讨论高速缓存，显而易见的是高速缓存也适合于如存储音频和视频这样的多媒体内容。在本节中，我们讨论了**内容分发网络**（Content Distribution Network，CDN），它提供了分发存储多媒体内容（以及用于分发传统的Web内容）的另一种方法。

CDN基于这样的原理：如果某客户机不能到达某内容（因为从服务器到客户机的尽力而为路径不能支持流式视频），该内容就应当带给该客户机。CDN因此使用一种不同于Web高速缓存的模式。对于CDN来说，支付费用的用户不再是ISP而是内容提供商。有视频要分发的内容提供商（例如CNN）向CDN公司（如Akamai）付费，使得它的视频可以以更短的可能时延到达请求用户。

一个CDN公司通常以下列方式提供内容分发服务：

1）CDN公司在整个因特网上安装数以百计的CDN服务器（CDN server）。CDN公司通常将CDN服务器放置在**数据中心**（data center），数据中心通常位于较低层ISP中，靠近ISP接入网络和客户机。

2）CDN在其CDN服务器中复制它的用户内容。无论何时用户更新它的内容时，该CDN向CDN服务器重新分发这些刷新的内容。

3）CDN公司提供一种机制，使得当用户请求内容时，该内容能够由以最快速度向该特定用户交付内容的那个CDN服务器来提供。这个服务器可能最靠近用户（可能和该用户同属于某个ISP），也可能CDN与用户之间有着一条不拥塞的路径。

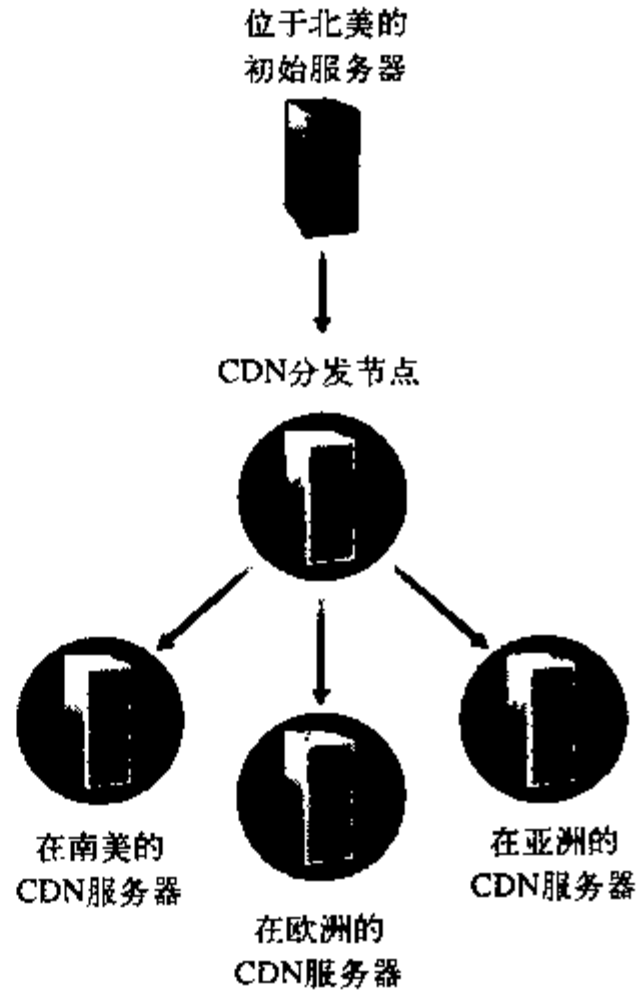


图7-8 CDN把内容提供商标记过的对象推向它的CDN服务器

图7-8显示了内容提供商和CDN公司之间的交互。内容提供商首先决定它的哪些对象（如视频）交给CDN分发。（内容提供商分发余下的内容而无需CDN干预。）内容提供商标记这部分内容，并将它们推向一个CDN节点，该CDN节点依次将这些内容复制并推向所选择的CDN服务器。该CDN公司可

能拥有一个专用网络，以将内容从该CDN节点推向其他CDN服务器。无论何时该内容提供商修改了某个CDN分发的对象，它就立即将刷新版本推向该CDN节点，该节点也会立即向CDN服务器复制并分发这些对象。记住下列问题很重要，每个CDN服务器通常包含来自很多不同内容提供商的对象。

现在有一个有趣的问题。当用户主机上的浏览器被指令获取某个特定的对象（用URL标识）时，浏览器是如何确定它应当从初始服务器还是从某个CDN服务器上获取该对象呢？CND通常利用DNS重定向功能来引导浏览器访问正确的服务器[Kangasharju 2000]。

举一个例子，假设内容提供商的主机名为www.foo.com，CDN公司的名字为cdn.com。再假设该内容提供商只是希望由该CDN分发它的视频MPEG文件，所有其他对象包括基本HTML页面，均直接由该内容提供商分发。为了完成这个方案，该内容提供商在其初始服务器上修改它所有HTML对象，为视频文件的URL加上前缀http://www.cdn.com。因此，如果在内容提供商处的HTML文件最初有对http://www.foo.com/sports/highlights.mpg的引用，该内容提供商通过在该HTML文件中用http://www.cdn.com/www.foo.com/sports/highlights.mpg替代引用来标记该对象。

当浏览器请求包含视频highlights.mpg的Web网页时，出现下列动作：

1）浏览器向初始服务器www.foo.com发送对基本HTML对象的请求，服务器将被请求的HTML对象发送给该浏览器。浏览器解析该HTML文件并发现了对http://www.cdn.com/www.foo.com/sports/highlights.mpg的引用。

2）浏览器然后在www.cdn.com上进行DNS查找，www.cdn.com是所引用URL的主机名。DNS已经进行了配置，使得所有到达根DNS服务器的有关www.cdn.com的查询被转发到它的权威DNS服务器。当这台权威服务器收到查询时，它析取请求浏览器的IP地址。使用为整个因特网构造的内部网络图，CDN的DNS服务器返回CDN服务器的IP地址，该CDN服务器对于请求浏览器而言看起来是最好的（通常是最靠近该浏览器的CDN服务器）。

3）在请求客户机中的DNS接收到带有IP地址的DNS回答。该浏览器则向具有该IP地址的CDN服务器发出HTTP请求。该浏览器从这台CDN服务器获得highlights.mpg视频文件。对于来自www.cdn.com的后继请求，该客户机继续使用相同的CDN服务器，因为www.cdn.com对应的IP地址已经保留在DNS缓存中了（在客户机主机中或本地DNS服务器中）。

如图7-9所示，总结而言，请求主机首先从初始Web服务器那里得到基本HTML对象；然后从该CDN的权威DNS服务器那里得到最好的CDN服务器的IP地址；最终从该CDN服务器获得该视频。注意到实现这种分发方案，并不需要对HTTP、DNS以及浏览器做任何改动。

剩下的事情就是解释CDN公司是如何确定对请求主机来说哪个是“最好的”CDN服务器了。尽管每个CDN公司有自己专门的解决方式，我们不难得到它们解决问题的大致思路。对因特网上的每个接入ISP（包括潜在的请求主机），CDN公司保持着对该接入ISP来说最好的CDN服务器的跟踪。CDN公司确定最好的CDN服务器基于以下知识：因特网选路表（特别是BGP表，我们在第4章中进行了讨论），往返时延估计值，以及它具有的从不同服务器到各个接入网络的其他测量数据；参见[Verma 2001]的讨论。以这种方式，CDN估算哪个CDN服务器为该ISP提供了最好的尽力而为服务。该CDN对因特网上大量的接入ISP进行这项工作，并且使用这些信息来配置权威DNS服务器。对于从大型运行CDN（Akamai）的角度讨论流式多媒体，参见[Sripanidkulchpai 2004]。

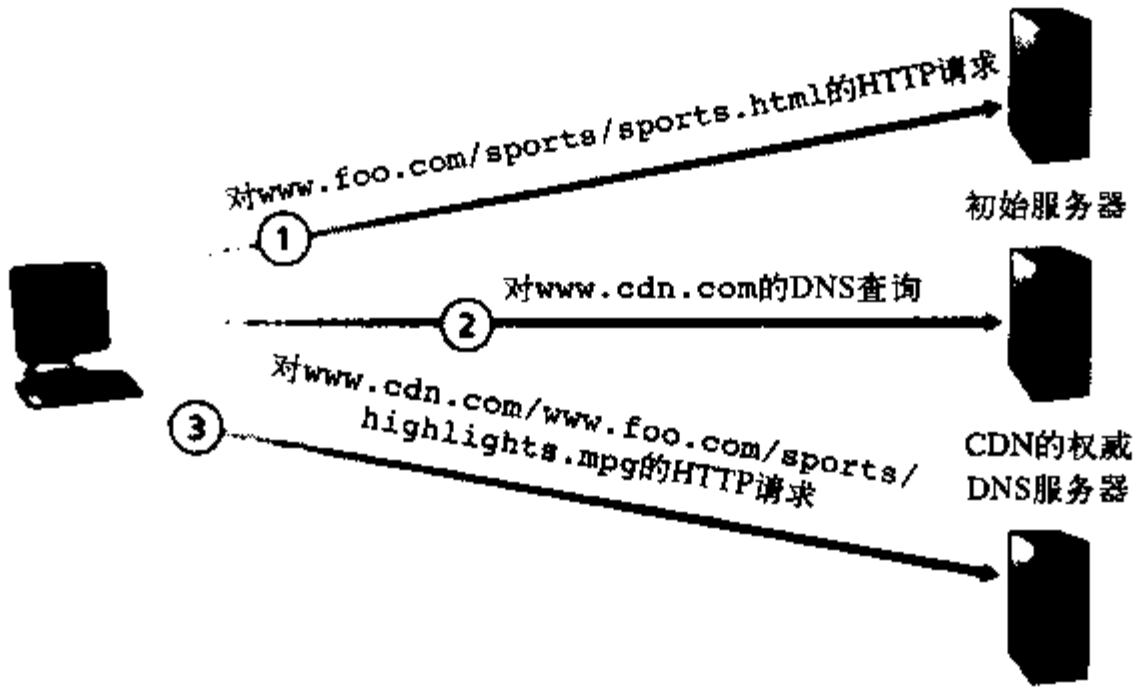


图7-9 CDN使用DNS将请求定向到临近的CDN服务器

7.3.5 规划尽力而为网络以提供服务质量

在本章前面的内容中，我们已经见到了应用层技术能够改善今天的尽力而为因特网中的多媒体应用的质量，这些应用层技术包括分组播放、FEC、主机中的分组交织、遍布网络部署的CDN基础设施。从本质上讲，支持多媒体应用的困难源于它们的性能要求（即低端到端分组时延、时延抖动和丢包），以及无论何时当网络拥塞时都会出现的分组时延、时延抖动和丢包这样一个事实。改善多媒体应用质量的最终方法——一种经常能用来解决只与资源紧张有关的任何问题的方法——是直接“在问题上投钱”，这样直接避免资源紧张。对于网络多媒体来说，这意味着在全网中提供足够的链路容量，使得网络拥塞以及由此产生的分组时延和丢包决不会（或仅非常少地）出现。借助于足够的链路容量，分组能够“飞驰”通过今天的因特网而没有排队时延和丢包。从许多方面看，这是一种理想的情况，即多媒体应用将得到完善执行，用户将为此高兴，并且这一切的取得并没有改变今天因特网的尽力而为的体系结构。当然，问题是要到达这样的完美境界多少容量才是“足够”的，从ISP的商业观点来看，提供“足够”的带宽是否是可实现的。

在给定的拓扑下需要为网络链路提供多少容量才能取得给定等级的端到端性能，这样的问题经常被称为**带宽配备**（bandwidth provisioning）。更为复杂的问题是如何设计一个网络拓扑（何处放置路由器，如何用链路互联路由器，以及为链路分配什么样的容量），以取得给定等级的端到端性能，这样的问题称为**网络规划**（network dimensioning）。带宽配备和网络规划都是复杂的主题，超出了本教科书的范围。然而，我们在这里必须注意到，为了预测两个网络端点之间的应用级性能，必须处理下列问题，满足某应用的性能要求配备足够的容量。

- 网络端点之间的流量要求模型。模型可能需要在呼叫级（例如，用户“到达”网络并启动端到端应用程序）和分组级（例如，进行中的应用程序所产生的分组）是特定的。注意到工作量可以随时间而改变。
- 定义良好的性能要求。例如，支持时延敏感流量（如交互式音频/视频应用）的性能要求也许是某种概率，即该应用的端到端时延大于某最大可容忍时延的概率，其中该最大可容忍时延小于某个小值e[Fraleigh 2003]。
- 对给定的工作量模型预测端到端性能的模型，找到能满足所有用户需求的最小高成本带

宽分配的技术。这里，研究人员正在忙于研发排队模型（参见1.4节）和优化技术，该模型能够对给定的工作量量化性能，该优化技术可以找到满足性能要求的最小成本带宽分配方式。

如果今天的尽力而为因特网通过规划能够（从技术的角度）以适当的性能支持多媒体流量的话，自然的问题是为什么今天的因特网服务提供商做不到。其答案主要是经济上和组织方面的。从经济观点看，用户愿意向ISP支付安装充足带宽的费用以通过尽力而为因特网支持多媒体应用吗？组织方面的问题也许甚至更令人沮丧。注意到两个多媒体端点之间的一条端到端路径通过多个ISP的网络。从组织方面观点看，这些ISP愿意协作以确保端到端路径适当地规划以支持多媒体应用吗？对于这些经济上和组织方面问题的展望，参见[Davies 2005]。对于装备第一层主干网以支持时延敏感流量的展望，参见[Fraleigh 2003]。

7.4 实时交互应用的协议

实时交互应用（包括因特网电话和视频会议）有望大大带动未来因特网的增长。因此标准机构如IETF和ITU多年来一直忙于（而且要继续忙下去！）苦心推敲这类应用的标准就毫不奇怪了。实时交互应用使用合适的标准，各个独立的公司将能够创造新的、引人注目的彼此互操作的产品。在本节中，我们研究用于实时交互应用的RTP、SIP和H.323，这3套标准正广泛地使用于工业产品中。

7.4.1 RTP

在前一节中我们了解到，多媒体应用的发送方在将音频/视频块传递给运输层之前为这些块附加上首部字段。这些首部字段包括了序号和时间戳。因为大多数多媒体网络应用能够利用序号和时间戳，因此有一个包括了音频/视频数据、序号、时间戳以及其他潜在有用字段的标准分组结构将会很方便。定义在RFC 3550中的RTP就是这样一个标准。RTP能够用于传输通用格式，如用于声音的PCM、GSM和MP3，以及用于视频的MPEG和H.263。它也可以用于传输专用的声音和视频格式。目前，RTP在上百个产品和研究原型中得到广泛的实现。它也是其他重要的实时交互协议（包括STP和H.323）的补充。

在本节中，我们介绍RTP及其伙伴协议RTCP。我们也鼓励读者去访问Henning Schulzrinne的RTP站点[Schulzrinne-RTP 2007]，它提供有关这个主题的很多信息。读者也可以访问RAT站点[RAT 2007]，它记载了使用RTP的因特网电话应用。

1. RTP基础

RTP通常运行在UDP之上。发送端在RTP分组中封装媒体块，然后在UDP报文段中封装该分组，并且将该报文段递交给IP。接收端从UDP报文段中提取出这个RTP分组，然后从RTP分组中提取出媒体块，并将这个块传递给媒体播放器来解码和显示。

举一个例子，考虑使用RTP来传输语音。假设语音源采用了64 kbps的PCM编码（也就是采样、量化和数字化）。再假设应用程序在20 ms块中收集这些编码数据，也就是一个块中有160字节。发送端在每个语音数据块的前面加上一个RTP首部（RTP header），这个首部包括音频编码的类型、序号和时间戳。RTP首部通常是12字节。音频块和RTP首部一起形成RTP分组（RTP packet）。然后向UDP套接字接口发送该RTP分组。在接收端，应用程序从它的套接字接口收到该RTP分组，从RTP分组中提取出该音频块，并且使用RTP分组的首部字段来正确解码和播放该音频块。

如果一个应用程序集成了RTP（而不是一个专用的方案来提供负载类型、序号或者时间戳），则该应用程序将更容易和其他网络的多媒体应用程序互操作。例如，如果两个不同的公司都开发因特网电话软件，并且都在它们的产品中集成了RTP，使用一种因特网电话产品的用户将有希望能够和使用另一种因特网电话产品的用户进行通信。在7.4.3节，我们将看到RTP经常和因特网电话标准一起使用。

应该强调的是，RTP本身不提供任何机制来确保数据的及时交付，或者提供其他服务质量保证；它甚至不保证分组的交付，或防止分组的失序交付。事实上，只有在端系统中才看到RTP封装。路由器不对携带RTP分组的IP数据报和不携带RTP分组的IP数据报进行区分。

RTP允许为每个源（例如一架照相机或者一个麦克风）分配一个独立的RTP分组流。例如，对于一个在两个参与者之间的视频会议，可能打开4个RTP流，两个流传输音频（一个方向一个），两个流传输视频（也是一个方向一个）。然而很多流行的编码技术（包括MPEG1和MPEG2）在编码过程中，将音频和视频捆绑在单个流中。当音频和视频与编码器捆绑时，每个方向只产生一个RTP流。

RTP分组并不限于单播应用。它们也可以在一对多和多对多的多播树上发送。对于一个多对多的多播会话，所有的会话发送方和源通常使用同样的多播组来发送它们的RTP流。在一起使用的RTP多播流，例如在视频会议应用中从多个发送方发出的音频和视频流，同属于同一个RTP会话（RTP session）。

2. RTP分组首部字段

如图7-10所示，4个主要的RTP分组首部字段是有效载荷类型、序号、时间戳和源标识符。

图7-10 RTP首部字段

RTP分组中的有效负载类型字段的长度是7比特。对于音频流，有效载荷类型字段用于指示所使用的音频编码类型（例如PCM、自适应增量调制、线性预测编码）。如果发送方在会话过程中决定改变编码，发送方可以通过该有效载荷类型字段把这种变化通知接收方。发送方可能要通过改变该编码来提高语音质量或者减小RTP流比特率。表7-2列出了当前RTP支持的一些音频有效载荷类型。

表7-2 RTP支持的一些音频有效载荷类型

有效载荷类型编号	音频格式	采样速率	速率
0	PCM μ律	8 kHz	64 kbps
1	1016	8 kHz	4.8 kbps
3	GSM	8 kHz	13 kbps
7	LPC	8 kHz	2.4 kbps
9	G.722	16 kHz	48~64 kbps
14	MPEG音频	90 kHz	—
15	G.728	8 kHz	16 kbps

对于一个视频流，有效载荷类型用于指示视频编码类型（例如运动JPEG、MPEG1、MPEG2、H.261）。发送方也可以在会话期间动态改变视频编码。表7-3列出了当前RTP支持的

一些视频有效载荷类型。其他重要的字段如下：

- 序号字段。序号字段长为16比特。每发送一个RTP分组则该序号增加1，而且接收方可以用该序号来检测丢包和恢复分组序列。例如，如果应用的接收端收到的RTP分组流在序号86和89之间存在一个间隙，那么接收方则知道分组87和88丢失了。那么接收方能够设法来隐藏该丢失的数据。
- 时间戳字段。时间戳字段长32比特。它反映了RTP数据分组中的第一个字节的采样时刻。如我们在上一节所见，接收方能够使用时间戳来去除网络中引入的分组时延抖动，提供接收方的同步播放。时间戳是从发送方的采样时钟中获得的。举一个例子，对于音频的每个采样周期（例如对于8 kHz的采样时钟，每125 μs为一个周期）时间戳时钟增加1；如果该音频应用产生由160个编码采样组成的块的话，那么当源激活时，对每个RTP分组则时间戳增加160。即使源未激活，该时间戳时钟也将继续以恒定速率增加。
- 同步源标识符（SSRC）。SSRC字段长为32比特。它标识了RTP流的源。通常在RTP会话中的每个流都有一个不同的SSRC。SSRC不是发送方的IP地址，而是当新的流开始时源随机分配的一个数。两个流被分配相同SSRC的概率是很小的。如果发生了，这两个源应选择一个新的SSRC值。

表7-3 RTP支持的一些视频有效载荷类型

有效载荷类型编号	视频格式
26	运动JPEG
31	H.261
32	MPEG1视频
33	MPEG2视频

3. 开发使用RTP的软件应用

开发基于RTP的网络应用有两种方法。第一种方法是应用开发者手工加入RTP，实际上也就是实际写出在发送方完成RTP封装和在接收方完成RTP解开的代码。第二种办法是应用开发者利用现有的RTP库（对于C程序员）和Java类（对于Java程序员），由它们为应用完成封装和解开。读者可能渴望写自己的第一个使用RTP的多媒体网络程序，那么我们现在就对这两种方法描述得更详细一些。（本章末的编程作业将指导读者创建一个RTP应用。）我们将在单播（而不是多播）通信的情况下做这个工作。

第2章讲过，UDP的API需要发送进程在将分组发送到UDP套接字之前，为它发送的每个UDP报文段设置目的IP地址和目的端口号。然后该UDP报文段将在因特网上传输，并且（如果该报文段没有因路由器缓冲区溢出而丢失）将最终到达这个应用接收进程的门。该门完全通过目的IP地址和目的端口号来寻址。实际上，任何包含该目的IP地址的目的端口号的IP数据报将被导向接收进程的UDP门。（该UDP API也允许应用开发者设置UDP源端口号，然而这个值对该报文段发送给哪个进程没有影响。）重要的是注意到，RTP没有强制指定一个特定的端口号。当应用开发者创建RTP应用程序时，开发者为该应用程序的两端指定该端口号。

作为本章编程作业的一部分，你要写一个RTP服务器，以在RTP分组中封装存储视频帧。你要自己完成这项工作；即你的应用将捕获一个视频帧，给该帧添加RTP首部从而创建一个RTP分组，然后将该RTP帧传递给UDP套接字。为了完成这些工作，你需要为各种RTP首部创建占位符字段，包括一个序号字段和一个时间戳字段。并且对于每个创建的RTP分组，你必须设置合适的序号和时间戳。你要明确地将所有这些RTP操作编码进你的应用的发送方中去。如图7-11所示，你用于网络的API是标准的UDP套接字API。

一种可替代的方法（在编程作业中没有做）是使用Java RTP类（或者对于C程序员来说是C RTP库）来实现RTP操作。如图7-12所示，使用这种方法，应用开发者得到RTP是传输层一

部分的印象，在应用层和传输层之间有一个RTP/UDP API。无需研究本质细节（因为它们与具体类/库有关），当向API发送一个媒体块时，应用发送方需要给接口提供媒体块本身、有效载荷类型号、SSRC和时间戳，以及目的端口号和IP目的地址。这里我们提一下，Jave 媒体框架（Java Media Framework, JMF）包括了完整的RTP实现。

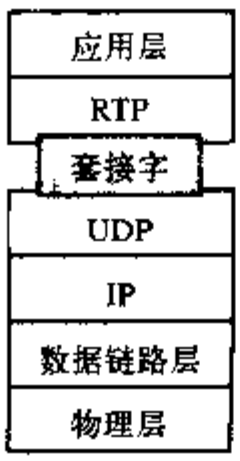


图7-11 RTP作为应用程序的一部分，位于UDP套接字之上

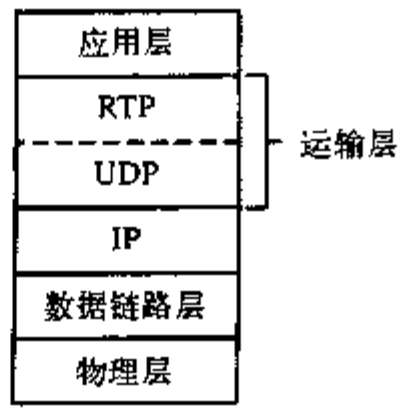


图7-12 RTP可被看成运输层的一个子层

7.4.2 RTP控制协议

RFC 3550也定义了RTCP。RTCP是一个能在网络的多媒体应用中和RTP一起使用的协议。如图7-13中多播情况所示，RTP会话中的每个参与者使用IP多播将RTCP分组传输给会话中的所有其他参与者。对于RTP会话，通常有单个多播地址，属于该会话的所有RTP和RTCP分组使用这个多播地址。RTP和RTCP分组通过使用不同的端口号来相互区分。（RTCP端口号设置为等于RTP端口号加1。）

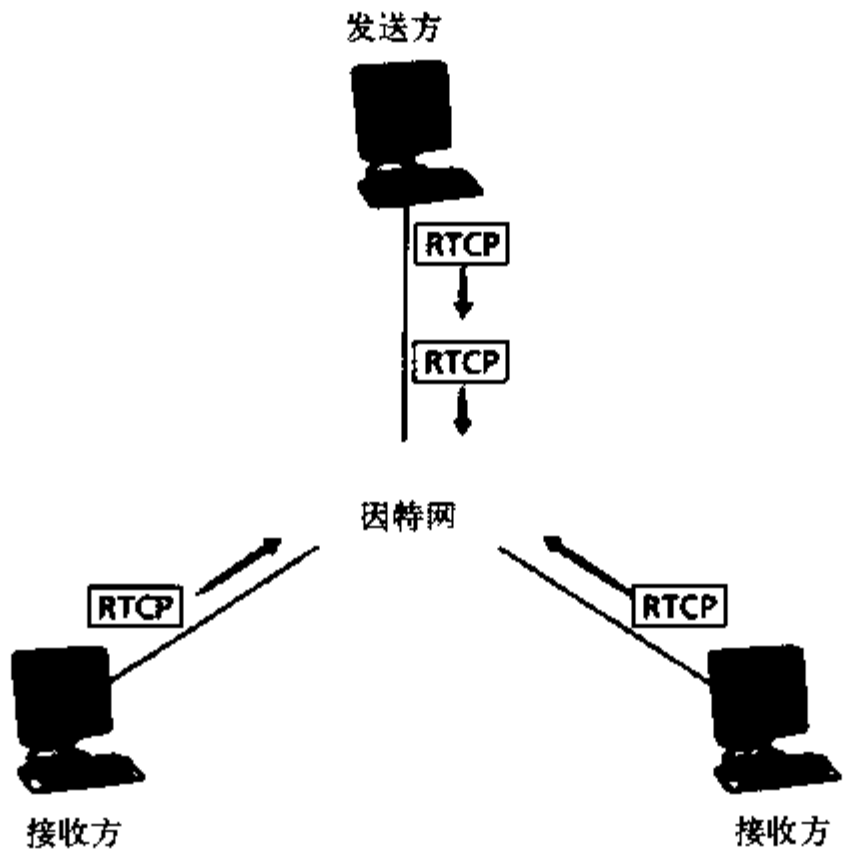


图7-13 发送方与接收方都发送RTCP报文

RTCP分组不封装音频或者视频块。相反，RTCP分组被周期地发送，它包含着发送方和/或接收方发布的对应用可能有用的统计报告。这些统计数据包括发送的分组数量、丢包数量和到达时延抖动。RTP规范[RFC 3550]没有规定应用应该如何处理这个反馈信息，而这些是应

用开发者自己的事。例如，发送方能够使用反馈信息来修改其传输速率。反馈信息也能用于诊断目的；例如，接收方可以判断问题是局部的、区域的或是全局的。

1. RTCP分组类型

对于接收方接收到的作为会话一部分的每个RTP流，接收方都会产生一个接收报告。接收方将它的接收报告汇聚在单个RTCP分组中。然后把该分组发送到连接所有会话参与者的多播树上。该接收报告包括几个字段，下面列出其中最重要的几个。

- RTP流的SSRC，接收报告是为RTP流而产生的。
- 在RTP流中丢失的那部分分组。每个接收方计算丢失的RTP分组数除以流中发送的RTP分组数。如果发送方收到的接收报告指出，接收方只收到发送方传输的一小部分分组，则为了减小网络拥塞和增加接收速率，发送方会转换为一个低的编码速率。
- 在RTP分组流中收到的最后一个序号。
- 到达时延抖动，它是RTP流中连续分组之间的到达时间间隔变化的平滑估计。

对于发送方正在发送的每个RTP流，发送方创建并传输RTCP发送方报告分组。这些分组包括了有关RTP流的信息，其中有：

- RTP流的SSRC。
- 时间戳和该流中最近产生的RTP分组的墙上时钟时间（wall clock time）。
- 该流中发送的分组数。
- 该流中发送的字节数。

发送方报告能够用于同步一次RTP会话中的不同媒体流。例如，考虑一个视频会议应用，其中每个发送方产生两个独立的RTP流，一个用于视频，一个用于音频。在这些RTP分组中的时间戳与视频和音频采样时钟关联起来，而不是与墙上时钟时间（即真实时间）相关联。每个RTCP发送方报告包括了与之相关联的RTP流中最近产生的分组的时间戳和分组创建的墙上时钟时间。因此RTCP发送方报告分组将采样时钟与真实时间关联了起来。接收方能够在RTCP发送方报告中使用这种关联关系来同步音频和视频的播放。

对于每个发送方发送的RTP流，发送方还创建并传输源描述分组。这些分组包含了有关源的信息，例如发送方的电子邮件地址、发送方的名字和产生RTP流的应用。它也包括相关联的RTP流的SSRC。这些分组提供了源标识（也就是SSRC）和用户/主机名字之间的映射。

RTCP分组是可叠加的；即接收方的接收报告、发送方报告和源描述能够连接成单个分组。然后该结果分组被封装到UDP报文段并转发到多播树。

2. RTCP带宽扩展

读者也许已经观察到RTCP有潜在的扩展性问题。例如，考虑由一个发送方和大量接收方组成的一个RTP会话。如果每个接收方周期地产生RTCP分组，则RTCP分组的聚合传输速率可以大大超过发送方发送RTP分组的速率。观察到发送到多播树的RTP流量不会随着接收方数量的增加而改变，而RTCP流量则随着接收方数量的增加而线性增长。为了解决这个扩展性问题，RTCP将参与者向多播树发送RTCP分组的速率修改为会话中参与者数量的函数。因为每个参与者也向其他每个参与者发送控制分组，所以每个参与者能够估计在该会话中参与者的总数[Friedman 1999]。

RTCP试图将它的通信量限制在会话带宽的5%以内。例如，假设有一个发送方，它以2 Mbps的速率发送视频。那么RTCP试图将它的通信量限制为2 Mbps的5%即为100 kbps，具体计算过程如下所述。这个协议将该速率的75%即为75 kbps给接收方；将该速率剩余的25%即

为25 kbps给发送方。对分配给接收方的75 kbps在接收方之间进行等量划分。因此，如果有R个接收方，那么每个接收方以75/R kbps的速率发送RTCP通信量，发送方以25 kbps的速率发送RTCP通信量。参与者（发送方或者接收方）通过动态计算平均RTCP分组长度（贯穿整个会话的）和用它分配的速率来分割该平均RTCP分组长度，确定出RTCP分组的传输周期。总之，发送方传输RTCP分组的周期是：

$$T=\frac{\text{发送方的数量}}{0.25\times 0.05\times \text{会话带宽}}\times \text{（平均RTCP分组长度）}$$

并且接收方传输RTCP分组的周期是：

$$T=\frac{\text{接收方的数量}}{0.75\times 0.05\times \text{会话带宽}}\times \text{（平均RTCP分组长度）}$$

7.4.3 SIP

想象这样一个世界，当你用PC机工作时，你的电话经因特网到达你的PC机。当你起床后开始散步时，新的电话自动地转发到你的PDA上。当你在开车时，新的电话自动地发送到你汽车中的某个因特网装置中。同样在这个世界里，当参与电话会议时，你可以通过一个地址簿来呼叫和邀请其他参与者来参与会议。其他参与者可能正在使用PC，或者带着PDA在散步，或者在开车，不论他们在哪里，你的邀请都透明地传输给他们。同样在这个世界里，当你浏览某个人网页时，可能有一个“呼叫我”的链接；点击该链接，将在你的PC机和该网页所有者（无论那个人在哪里）之间建立起一个因特网电话会话。

在这样一个世界中，不再有电路交换电话网络。相反，所有的呼叫都通过因特网传递(从一端到另一端)。同样在这个世界里，公司不再使用专用电话交换机（PBX）(即处理公司内部电话呼叫的本地电路交换机)。而公司内部的电话流量通过公司的高速LAN来传送。

所有这些听起来像是科学幻想。当然，现在的电路交换网络和PBX在近期不会完全消失[Jiang 2001]。然而协议和产品将要使这个幻想变为事实。在这个方向上最有希望的协议是在[RFC 3261] 中定义的会话发起协议（SIP)。SIP是一个轻型协议，它完成以下工作：

- 它提供在呼叫者和被叫者之间经IP网络建立呼叫的机制。它允许呼叫者通知被叫者它要开始一个呼叫，它允许参与者约定媒体编码，它也允许参与者结束呼叫。
- 它提供让呼叫者判定被叫者现在的IP地址的机制。因为用户可能动态地分配地址（使用DHCP)，而且因为它们可能有多个IP设备，每个都有一个不同的IP地址，所以用户没有单一的、固定的IP地址。
- 它提供进行呼叫管理的机制，例如在呼叫期间增加新媒体流，在呼叫期间改变编码，在呼叫期间邀请新的参与者，呼叫转移和呼叫保持。

1. 到已知IP地址建立一个呼叫

为了理解SIP的本质，最好看一个具体的例子。在这个例子中，Alice在使用PC机，她要呼叫Bob，Bob也在使用PC机工作。Alice和Bob的PC机都配有基于SIP的软件来生成和接收电话呼叫。在这个开始的例子中，我们将假设Alice知道Bob的PC机的IP地址。图7-14描述了这个SIP呼叫建立的过程。

在图7-14中，我们看到当Alice给Bob发送一个INVITE报文（类似于HTTP请求报文）时，SIP会话开始了。该INVITE报文通过UDP发送给SIP的周知端口5060。(SIP报文也可以经TCP发送。）该INVITE报文包括了对Bob的标识（bob@193.64.210.89)、Alice当前IP地址的指示、

Alice希望接收的音频的指示（该音频以格式AVP 0编码——PCM μ律编码，并在RTP中封装），以及指出她要在端口38060接收的RTP分组。在收到了Alice的INVITE报文之后，Bob发送一个SIP响应报文，该报文类似HTTP的响应报文。该响应SIP报文也被发送到SIP端口5060。Bob的响应包括一个200 OK和他的IP地址的指示，他希望用于接收的编码和分组化，以及音频数据应该发送到达的端口号。注意到在这个例子中，Alice和Bob准备使用不同的音频编码机制：要求Alice使用GSM来编码她的音频，而要求Bob使用PCM μ律来编码他的音频。在接收到Bob的响应后，Alice向Bob发送SIP确认报文。在这个SIP事务之后，Bob和Alice可以交谈了。（为了看起来清晰，图7-14显示Alice在Bob之后说话，但是事实上他们通常可以同时进行交谈。）Bob会根据要求对音频进行编码和分组化，并将音频分组发送给IP地址167.180.112.24的端口号38 060。Alice也会根据要求对音频进行编码和分组化，并将音频分组发送给IP地址193.64.210.89的端口号48753。

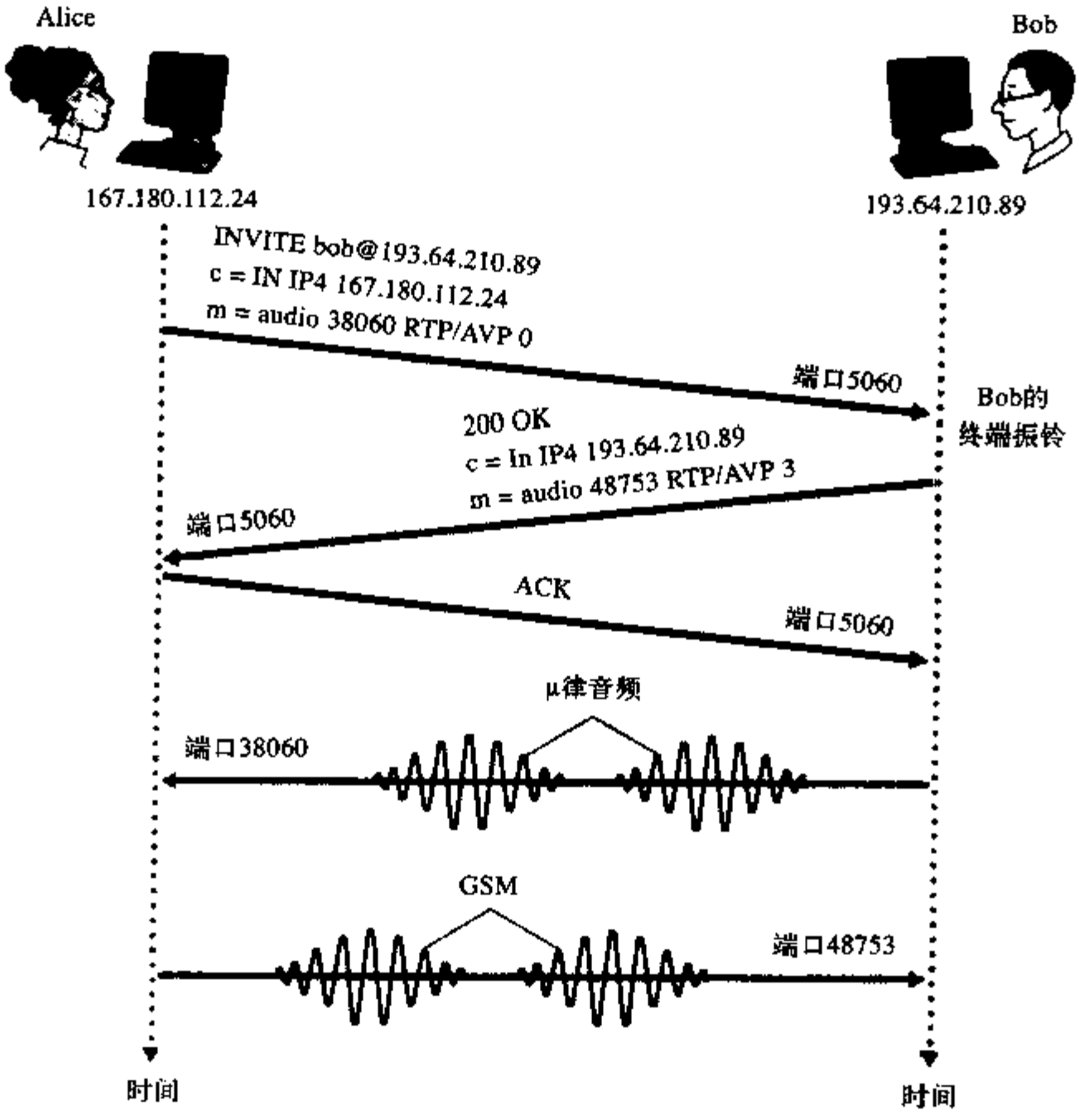


图7-14 当Alice知道Bob的IP地址时的SIP呼叫建立

从这个简单的例子中我们学到了一些SIP的关键特性。首先，SIP是一个带外协议：第一，SIP报文使用了一个不同于发送和接收媒体数据的套接字来发送和接收。第二，SIP报文本身是可读的ASCII，类似HTTP报文。第三，SIP要求所有的报文都要确认，因此它能够在UDP或者TCP上运行。

在这个例子中，我们考虑一下如果Bob没有PCM μ律编解码器对音频进行编码将会发生什

么情况。在这种情况下，Bob不是用200 OK来响应，而是可能用一个600 Not Acceptable（不可接受）来响应，并在报文中列出他能够使用的所有编解码器。然后Alice从中选择一个编解码器，并发送另一个INVITE报文，以此通告了已选择的编解码器。Bob也能够直接通过发送某个拒绝应答代码来直接拒绝该呼叫。(有很多这种代码，包括“busy”(忙)、“gone”(离开)、“payment”(付费)和“forbidden”(禁止)。)

2. SIP地址

在前面的例子中，Bob的SIP地址是sip:bob@193.64.210.89。然而，我们希望许多（即使不是大多数）SIP地址类似于电子邮件地址。例如，Bob的地址可以是sip:bob@domain.com。当Alice的SIP设备发送INVITE报文，该报文包括这个像电子邮件地址的地址；然后SIP的基本设施将该报文转发给Bob正在使用的IP设备（如我们下面要讨论的那样）。其他可能的SIP地址形式可能是Bob固有的电话号码，或者只是Bob的名字/中间名/姓氏（假设它是唯一的）。

SIP地址一个有趣的特点是它们能够被包括在Web页面中，如人们的电子邮件地址用mailto URL形式包含在Web页面中那样。例如，假设Bob有个人主页，并且他要为这个主页的访问者提供一个呼叫他的方法。于是他可能直接在主页中包括URL sip:bob@domain.com。当访问者点击该URL，访问者设备中的SIP应用将启动，向Bob发送INVITE报文。

3. SIP报文

在这个简短的SIP介绍中，我们将无法包括所有SIP报文类型和首部，而将简要地看一下SIP INVITE报文，以及少数通用的首部行。我们再假设Alice要对Bob发起IP电话呼叫，此时Alice只知道Bob的SIP地址bob@domain.com，并不知道Bob正在使用的设备的IP地址。那么她的报文可能看起来有些像下面这样：

```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

这个INVITE行包括SIP的版本，这与HTTP请求报文一样。任何时候SIP报文通过SIP设备(包括产生该报文的设备)，它都附加上一个Via首部，该首部指示该设备的IP地址。(我们不久将看到通常INVITE报文在到达被呼叫者的SIP应用之前会通过很多SIP设备。)与电子邮件报文类似，SIP报文包括一个From首部行和一个To首部行。该报文包括一个Call-ID（呼叫标识符)，它唯一标识该呼叫（类似电子邮件中的报文ID)。它包括一个Content-Type（内容类型）首部行，它定义用于描述包含在SIP报文中的内容的格式。它还包括Content-Length（内容长度）首部行，它提供报文中内容的字节长度。最后，在一个回车和换行之后，该报文包含了内容。在这种情况下，内容提供有关Alice的IP地址和Alice要如何接收该音频的信息。

4. 名字翻译和用户定位

在图7-14的例子中，我们假设Alice的SIP设备知道能够联系到Bob的IP地址。但是这种假设是很不现实的，不仅因为IP地址通常通过DHCP动态分配，而且Bob可能有多个IP设备（例如，他家里、工作地点和车里有不同的设备)。因此现在我们假设Alice只知道Bob的电子邮件

地址bob@domain.com，而且同样的地址用于基于SIP的呼叫。在这种情况下，Alice需要获得用户bob@domain.com正在使用的设备的IP地址。为了获得它，Alice创建一个INVITE报文，它以INVITE bob@domain.com SIP/2.0开始，并将该报文发送给一个**SIP代理**（SIP proxy）。该代理将以一个SIP回答来响应，该回答中也可能包含bob@domain.com正在使用的设备的IP地址。该回答也可以选择包括Bob的语音信箱的IP地址，或者它可能包括一个Web页面的URL（它上面写着“Bob在睡觉，请不要打扰！”）。代理返回的结果也可能取决于呼叫者：如果呼叫是Bob的妻子发出，他可能接受该呼叫，并提供他的IP地址；如果该呼叫是Bob的岳母发出的，他可能用指向“我正在睡觉”的Web页面的URL来响应。

现在你可能想知道这个代理服务器是怎样判定bob@domain.com现在的IP地址的？为了回答该问题，我们需要讲一下另一个SIP设备，即**SIP注册器**（SIP registrar）。每个SIP用户都有一个相关联的注册器。任何时候用户在设备上发起SIP应用时，该应用给注册器发送一个SIP注册报文，向注册器通知它现在的IP地址。例如，当Bob在他的PDA上发起SIP应用时，该应用将发送一个类似于下述行的报文：

```
REGISTER sip:domain.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:bob@domain.com
To: sip:bob@domain.com
Expires: 3600
```

Bob的注册器跟踪Bob现在的IP地址。无论何时Bob切换到一个新的SIP设备时，该新设备将发送一个新的注册报文，指示该新的IP地址。如果Bob长时间使用同样的设备，该设备将发送刷新注册报文，指示最近发送的IP地址仍然有效。（在上面的例子中，需要每隔3600 s发送刷新报文，以维护在注册器中的地址。）值得注意的是注册器和DNS权威名字服务器类似：DNS服务器将固定的用户名翻译为固定的IP地址；SIP注册器把固定的人的标识符（例如bob@domain.com）翻译为一个动态的IP地址。SIP注册器和SIP代理通常运行在同一台主机上。

现在我们研究一下Alice的SIP代理服务器是如何获得Bob当前的IP地址的。从上面的讨论中我们看到，代理服务器只需要转发Alice的INVITE报文给Bob的注册器/代理即可。于是注册器/代理可能将该报文转发给Bob现在的SIP设备。最后，在收到了Alice的INVITE报文之后，Bob可以向Alice发送一个SIP应答。

举一个例子，考虑图7-15，其中正在217.123.56.89上工作的jim@umass.edu要对正在197.87.54.21上工作的keith@upenn.edu发起经IP的语音（VoIP）会话。采用下面的步骤：①Jim向umass的 SIP代理发送INVITE报文。②该代理在SIP注册器upenn.edu上进行DNS查询（在图中没有显示），并将该报文转发给注册服务器。③因为keith@upenn.edu没有在注册器upenn上注册，upenn注册器发送一个重定向应答，指示它应该试一试keith@eurecom.fr。④umass的代理向eurecom的SIP注册器发送INVITE报文。⑤eurecom注册器知道keith@eurecom.fr的IP地址，并将INVITE转发给正运行Keith的SIP客户机的主机197.87.54.21。⑥－⑧通过注册器/代理把SIP响应返回给217.123.56.89上的SIP客户机。⑨媒体直接在两个客户机之间发送。（还有一个SIP确认报文，没有画出来。）

我们有关SIP的讨论集中在语音呼叫的呼叫发起方面。SIP通常是一个发起和结束呼叫的信令协议，它能够用于视频会议呼叫和基于文本的会话。事实上，SIP已经成为许多即时讯息应用的基本组件。我们鼓励想学习更多有关SIP知识的读者访问Henning Schulzrinne的SIP Web站点[Schulzrinne-SIP 2007]。特别是，在这个站点上你会发现SIP客户机和服务器的开放

源软件[SIP Software 2007]。

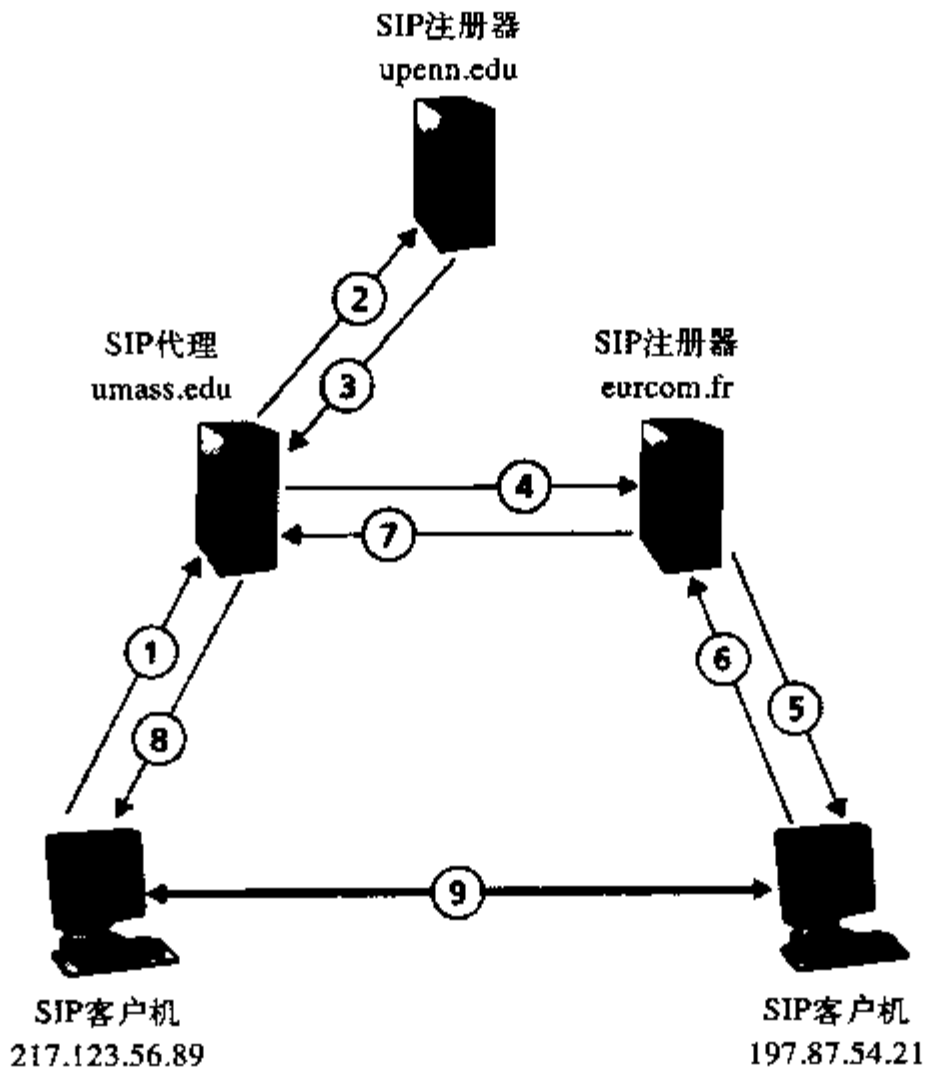


图7-15 发起会话，涉及SIP代理与注册器

7.4.4 H.323

作为SIP的一个替代物，H.323是因特网上端系统之间一个实时音频和视频会议的流行标准。如图7-16所示，该标准也涉及连接到因特网的端系统与连接到普通电路交换电话网的电话之间如何通信。(SIP也完成这项工作，尽管我们没有讨论它。) H.323网守（gatekeeper）是一个类似于SIP注册器的设备。

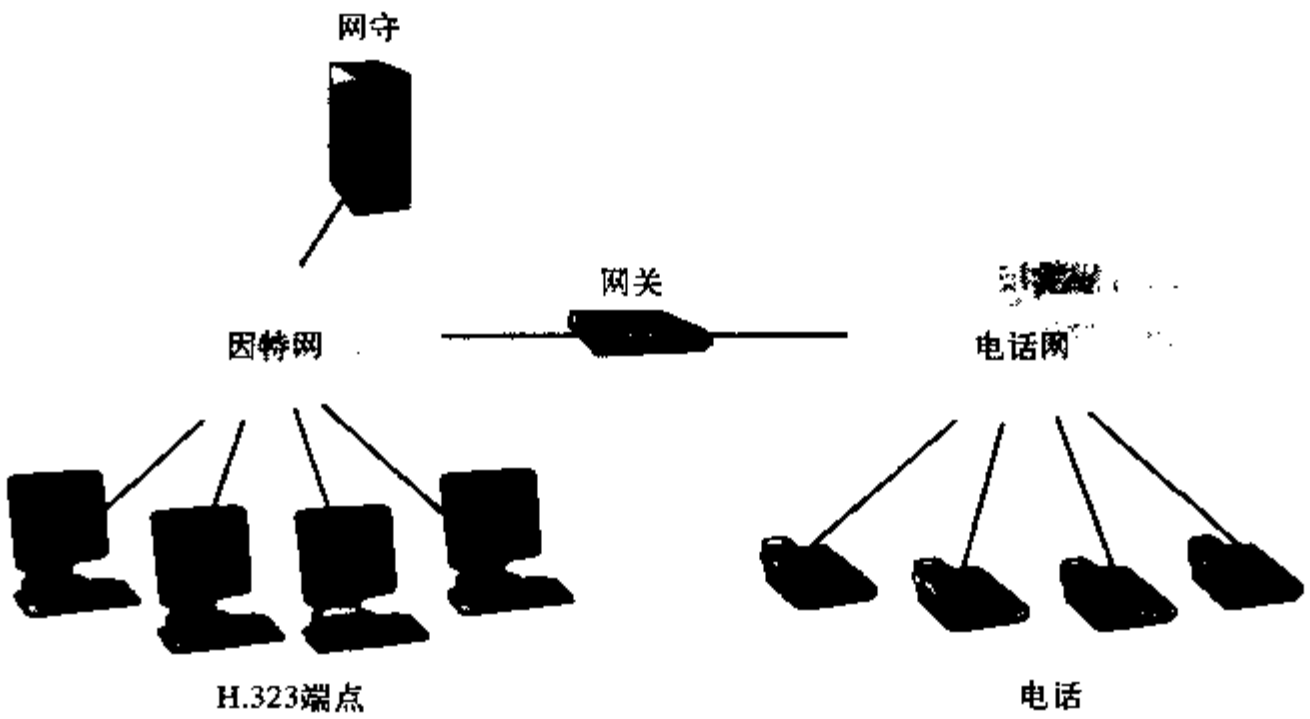


图7-16 连到因特网的H.323端系统能与连到电路交换电话网的电话进行通信

H.323标准是一种伞状的规范，包括下面的规范：

- 有关端点如何协商通用的音频/视频编码的规范。因为H.323支持各种各样的音频和视频编码标准，所以需要一个协议来允许通信的端点约定一个公共的编码。
- 有关音频和视频块如何封装和在网络上发送的规范。特别是为了这个目的，H.323 强制使用RTP。
- 有关端点如何和它们相关的网守之间通信的规范。
- 有关因特网电话如何通过网关与在公共电路交换电话网中的普通电话之间通信的规范。

在最低限度上，每个H.323端点必须支持G.711语音压缩标准。G.711使用PCM来产生56 kbps或者64 kbps的数字化的语音。尽管H.323要求每个端点都具有语音能力（通过G.711），但视频能力是可选的。因为视频支持是可选的，故终端的制造商能够出售较简单的语音终端和较复杂的支持音频和视频的终端。H.323端点的视频能力是可选的。然而，如果一个端点支持视频，那么它必须（以最低限度）支持QCIF H.261（176×144像素）视频标准。

H.323是一个内容丰富的伞状标准，它除了上面描述的协议和标准外，强制使用H.245控制协议、Q.931信令信道和通过网守来注册的RAS协议。

我们通过强调H.323和SIP之间的一些最重要的区别来对本节进行小结。

- H.323是关于多媒体会议的完整的、垂直集成的协议族：信令、注册、准入控制、传输和编解码。
- 另一方面，SIP只处理会话的发起和管理，是单一组件。SIP使用RTP，但是不强制使用它。它使用G.711语音编解码和QCIF H.261视频编解码，但是不强制使用它们。它能够与其他协议和服务结合使用。
- H.323来源于ITU（电话），而SIP来源于IETF，并从Web、DNS和因特网电子邮件中借鉴了很多概念。
- H.323作为伞状标准，大而复杂。SIP使用KISS（Keep It Simple, Stupid）原则：保持简单。

有关H.323、SIP和VoIP的一个很优秀的总体讨论参见[Hersent 2000]。

7.5 提供多个等级的服务

在前面几节中，我们学习了当今因特网中的多媒体应用是怎样使用序号、时间戳、FEC、RTP和H.323的。CDN给出了一种分发多媒体内容的系统范围的解决方案。但是就凭这些技术足以支撑可靠而健壮的多媒体应用，例如与现在电话网服务等价的IP电话服务吗？在回答该问题之前，我们再次回顾一下当今因特网给它的所有应用提供的尽力而为服务，即对于应用将得到的服务质量（QoS）不作任何承诺。应用将收到网络当时能够提供的任何等级的性能（例如，端到端分组时延和丢失）。前面还讲过，当前的公用因特网不允许时延敏感的多媒体应用请求任何特殊的对待。因为路由器等同对待所有的分组，包括时延敏感的音频和视频分组，毁坏正在进行的IP电话呼叫质量，所需的是足够强的干扰流量（即网络拥塞），这将显著地增加一个IP电话呼叫的时延和丢包。

但是如果目标是在当前的因特网上提供一个比尽力而为服务更强的服务模型的话，能提供的到底是什么样的服务呢？一个简单的强化服务模型是将流量划分为多个类别，并为这些不同类别的流量提供不同等级的服务。例如，相比于弹性流量如FTP或HTTP而言，某ISP可能要为时延敏感的VoIP或电信会议流量提供更高的服务类别（并对该服务收取更高的费用）。我们从日常生活中都熟悉了不同类型的服务，飞机上头等舱乘客比公务舱乘客得到更好的服务，公务舱乘客又比经济舱乘客得到更好的服务；VIP在社交场合有立即进入的优待而其他人

必须排队等待；在某些国家中老者被尊重，提供了荣誉座位和桌上最精细的食物。

注意到在聚合流量中，例如在多种流量类型中，提供这种有差别的服务是重要的。例如，所有头等舱乘客被一视同仁（头等舱乘客得到的服务都是一样的），就像所有的VoIP分组在网络中得到了相同的对待，独立于特定的它们所属的端到端连接。正如我们将见到的那样，通过处理少量的流量聚合，而不是大量的个别连接，提供好于尽力而为服务的新型网络机制，要能够保持相对简单。

早期因特网设计者的心中清楚地有这种多种类型服务的概念。在图4-13中讲过，在IPv4首部中的服务类型（ToS）字段。IEN123 [ISI 1979]描述也呈现在早先的IPv4数据报中的ToS字段时说："服务类型[字段]提供了所希望的服务质量的参数的指示。当传输一个数据报通过某特定网络时，这些参数被用于引导实际服务参数的选择。几种网络提供了服务优先权，该优先权以某种方式对待高优先权流量，使之比其他流量更为重要。"甚至在30年前，向不同等级的流量提供不同等级的服务的设想就是清晰的！然而，又花费了我们30年才实现了这种设想。

在7.5.1节中，我们通过考虑几种场景来开始我们的研究，这几种场景将激发我们对支持多种类别服务的特定机制的需求。我们在7.5.2节中将涉及两个重要的主题，即链路级调度和分组分类/监管。在7.5.3节中，我们将讨论区分服务，这是因特网当前提供区分服务的标准。

7.5.1 启发研究的场景

图7-17表示了一种简单的网络情况。假设两个应用分组流产生于一个LAN中的主机H1和H2，其目的地是另一个LAN中的主机H3和H4。在这两个LAN中的两台路由器通过一条1.5 Mbps的链路连接起来。我们假设LAN的速度远远高于1.5 Mbps，我们主要集中于路由器R1的输出队列；如果H1和H2的总计发送速率超过了1.5 Mbps，分组延时和丢包就要在这里发生。我们现在考虑几种情况，每种情况将为我们深入理解支持多种服务类型的特定机制。

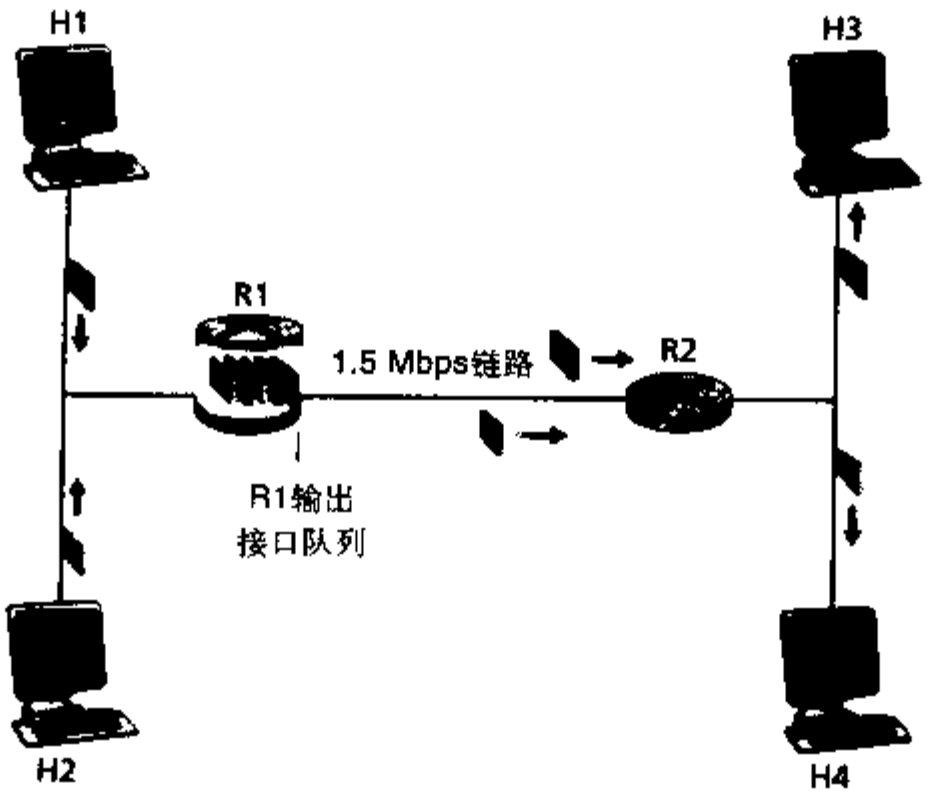


图7-17 有两个应用的简单网络

情况1：一个1 Mbps的音频应用和一个FTP传输

情况1图示在图7-18中。这里一个1 Mbps的音频应用（例如一个CD质量的音频呼叫）和一个从H2到H4传输文件的FTP应用共享R1和R2之间1.5 Mbps的链路。在尽力而为服务的因特

网中，该音频和FTP分组在R1的输出队列混合，并且（通常）以先进先出（FIFO）的次序传输。在这种情况下，从FTP源产生的突发分组可能潜在地填充满这个队列，引起IP音频分组过度延迟或者由于R1的缓冲区溢出而丢失。我们应该如何解决这个潜在的问题呢？假定该FTP应用没有时间限制，我们的直觉也许是在R1为音频分组分配严格的优先级。在一个严格的优先级调度规则下，在R1输出缓冲区的音频分组总是在R1输出缓冲区中的任何FTP分组之前传输。对音频流量而言，从R1到R2的链路看起来像一条1.5 Mbps专用链路，而对于FTP流量，只有当没有音频流量排队时才使用R1到R2的链路。

为了让R1在它的队列中区分音频和FTP分组，每个分组必须被标记为属于这两类流量中的哪一类。这是IPv4中服务类型（ToS）字段的最初目的。显而易见，这是我们对需要提供多种类型流量的机制的第一个直觉法则。

直觉法则1：标记分组使得路由器可以区分属于不同类型流量的分组。

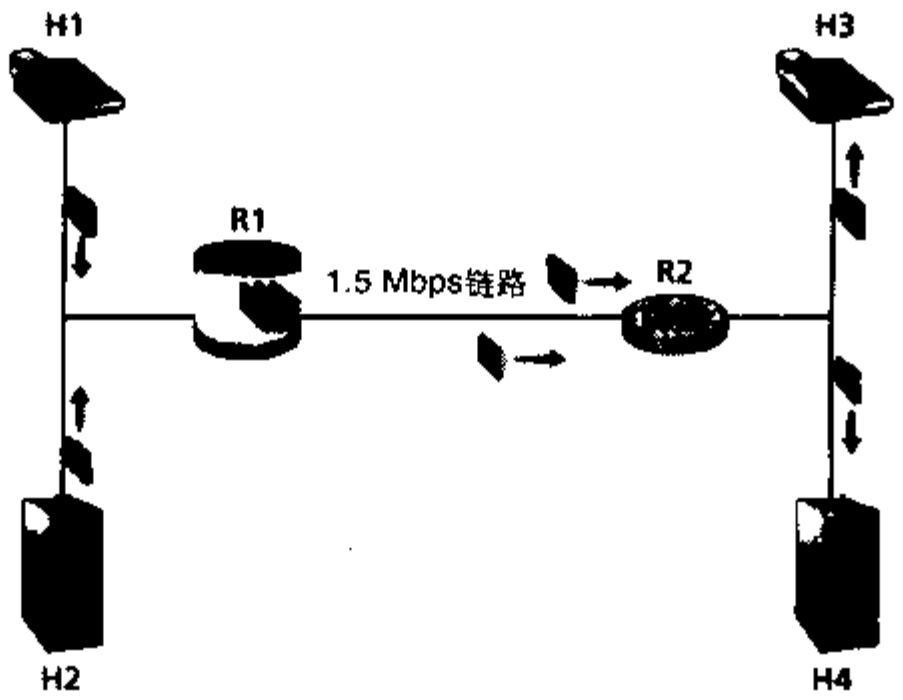


图7-18 竞争的音频应用与FTP应用

情况2：一个1 Mbps的音频应用和一个高优先级FTP传输

第二种情况只与第一种情况稍微有些不同。现在假设FTP用户已经从它的ISP购买了“白金服务”（也就是高价的）因特网接入，而音频用户买了便宜的、低价的因特网服务，该服务只值白金服务极小部分。在这种情况下，这个廉价的用户音频分组应该获得比FTP分组更高的优先级吗？当然不。在这种情况下，根据发送方的IP地址来区分分组看起来更合理一些。更一般地，我们看到对于路由器来说，根据某种准则来对分组进行分类是必要的。那么要求对“直觉法则1”进行一些小的修改：

直觉法则1（修正）：分组分类使得路由器可以区分属于不同类型流量的分组。

明显的分组标记是一种可以区分分组的方法。然而，分组携带的标记本身并不能强制该分组将得到给定的服务质量。标记只是区分分组的一种机制。路由器区分分组并对它们进行不同的处理，其方式是一个策略问题。

情况3：一个异常的音频应用和一个FTP传输

现在假设不知何故（通过使用我们在后继部分要学习的机制），路由器知道它应该给来自1 Mbps音频应用的分组高优先级。因为输出链路速度是1.5 Mbps，即使FTP分组得到低优先级，它们仍然可以收到平均0.5 Mbps的传输服务。但是如果音频应用开始以1.5 Mbps或者更高的

速率（恶意的或者由于应用的差错）发送分组，那会发生什么样的情况呢？在这种情况下，FTP分组将挨饿，也就是在R1到R2的链路上得不到任何服务。如果有同等优先级的多个应用（例如，多个音频呼叫）共享一段链路带宽，类似的问题会发生；一个不合时宜的流可能会降低和毁坏其他流的性能。理想情况下，为了保护一个流不受另一个异常流的影响，希望流的类型之间有一定程度的隔离，并且在同类流之间也有有一定程度的隔离。在一个给定流量类型中保护各个流使之免受其他流干扰的概念与我们前面的观察相矛盾，该观察是指对来自同一类别的所有流的分组应当有相同对待。在实践中，位于一个类别的分组的确被位于网络核心的路由器同等对待。然而，在网络边缘，可以监测一个给定流内的分组以确保各条流的聚合速率不能超过一个给定的值。

这些考虑让我们有了第2个直觉法则。

直觉法则2：希望在流量类型之间和流之间提供一定程度的隔离，以便一个类或一条流不会受到另一类或一条异常流的负面影响。

在下节中，我们将研究流量类型或流之间提供这种隔离的特定机制。这里我们注意到，有两种广义的方法可以使用。首先，如图7-19所示那样监管（police）流量是可能的。如果流量类型或流必须满足一定的标准（例如，音频流不超过1 Mbps的峰值速率），那么可以设置一个监管机制来确保这些准则的确被遵守。如果被监管的应用行为异常，这个监管机制将采取某种行动（例如，将那些和标准冲突的分组丢弃或者延时），以便实际进入网络的流量符合该标准。我们在下一节中研究的漏桶机制可能是使用最广泛的监管机制。在图7-19中，分组分类和标记机制（观察1）以及监管机制（观察2）都协同作用于网络的边缘，或在端系统或在边界路由器中。

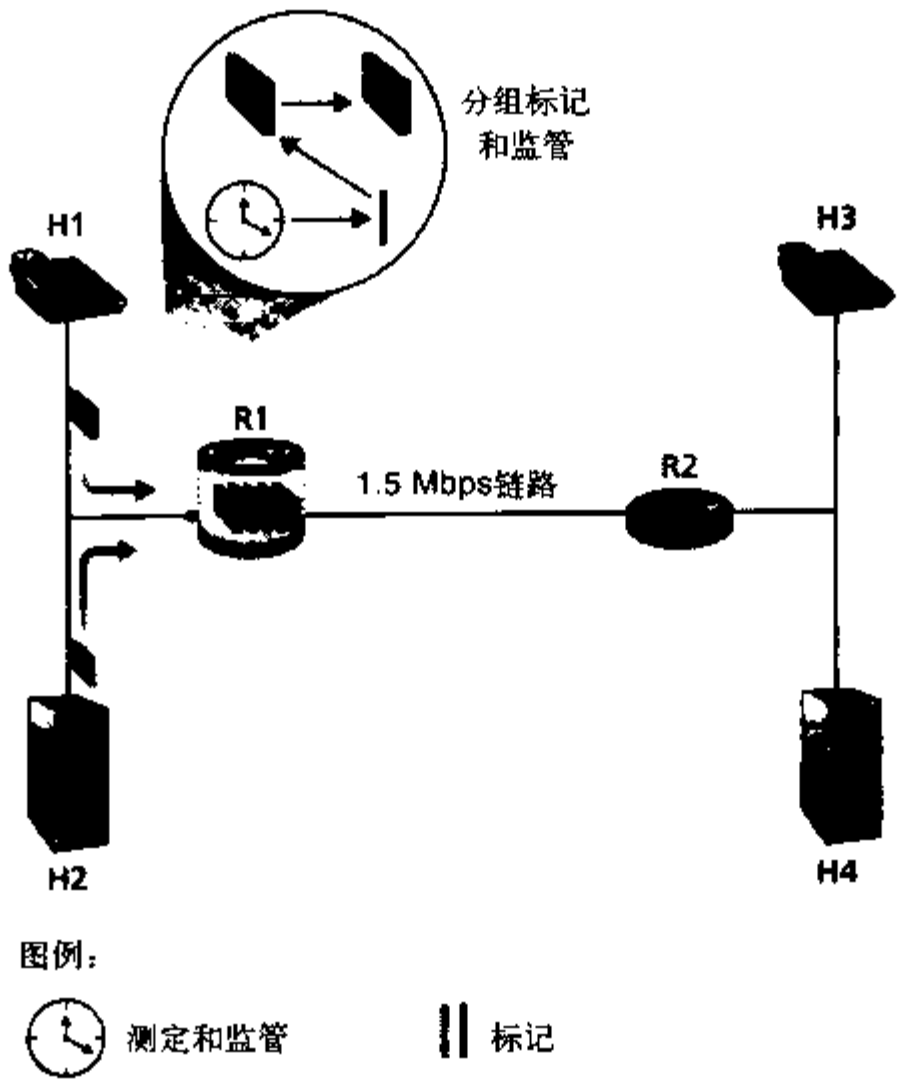


图7-19 监管（与标记）音频与FTP通信流

提供流量类型或流之间隔离的另一种可选择的方法是使用链路级的分组调度机制来明确地给每个应用流分配固定量的链路带宽。例如，R1可以分配1 Mbps给音频流，可以分配0.5 Mbps给FTP流。在这种情况下，音频和FTP流分别看到了容量为1.0和0.5 Mbps的逻辑链路，如图7-20所示。

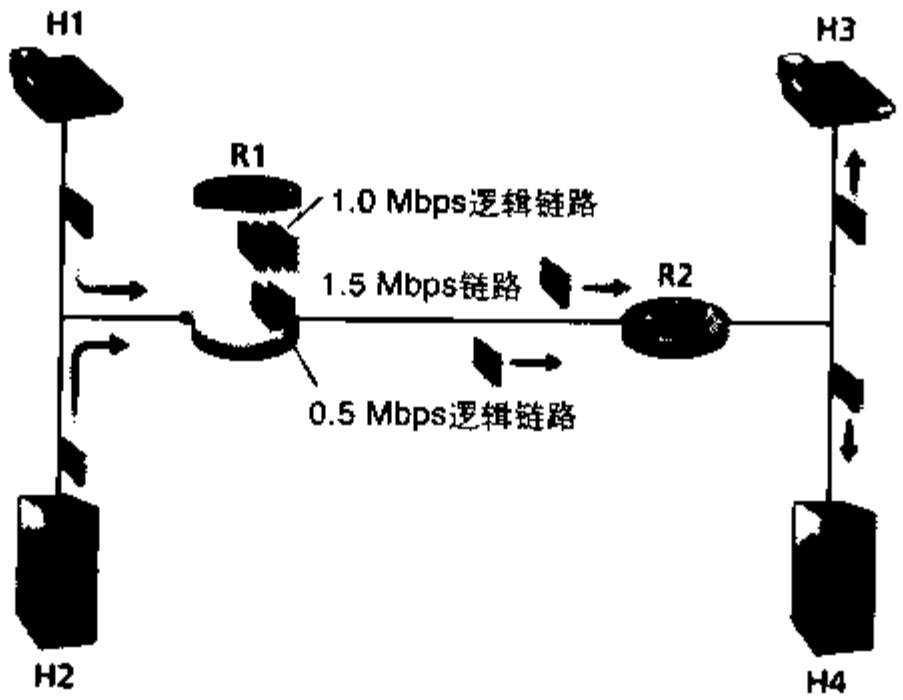


图7-20 音频与FTP应用流的逻辑隔离

通过严格执行链路级的带宽分配，一个流量类型或流只能够使用已经分配的带宽数量；特别是它不能利用其他应用现在不使用的带宽。例如，如果音频流静默了（假如谈话者暂停了，没有产生音频分组），FTP流仍然不能够在R1到R2的链路上传输超过0.5 Mbps，即使音频流分配的1 Mbps带宽在那时没有使用。因此希望能够尽可能有效地使用带宽，允许一个流在任何给定的时刻及时地使用另一个流没有使用的带宽。这是我们对提供服务质量的第3个直觉法则。

直觉法则3：当为流量类型或流之间提供隔离时，希望尽可能有效地使用资源（例如链路带宽和缓冲区）。

7.5.2 调度和监管机制

既然我们已经深入地分析了提供不同类型服务所需的机制，现在我们就来详细地考虑两种最重要的机制，即调度和监管。

1. 调度机制

我们在1.3节和4.3节曾经讨论过，属于各种网络流的分组被复用在一起，并且在与链路关联的输出缓冲区排队等待传输。选择排队的分组在链路上传输的方式称为**链路调度规则**（link-scheduling discipline）。我们现在更详细地考虑几个最重要的链路调度规则。

（1）先进先出（FIFO）

图7-21表示了**先进先出**（First-In-First-Out, FIFO）链路调度规则的排队模型的抽象。如果链路正忙于传输另一个分组，到达链路输出队列的分组要排队等待传输。如果没有足够的缓冲区空间来容纳到达的分组，那么队列的**分组丢弃策略**（packet-discarding policy）判定该分组是要丢弃（丢失），还是从队列中去除其他分组来给该到达的分组提供空间。在下面的讨论中，我们将忽视分组丢弃。当一个分组通过输出链路完全传输（也就是接受服务）时，它从队列中去除。

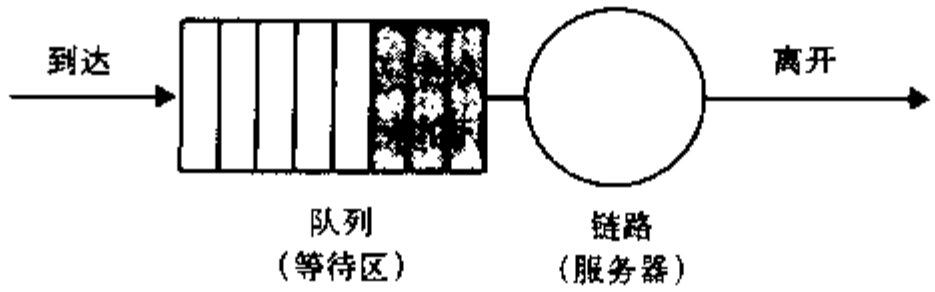


图7-21 FIFO排队抽象

FIFO调度规则（也称为先来先服务，即FCFS）按照分组到达输出链路队列的相同次序来选择分组在链路上传输。我们都很熟悉公共汽车站（尤其在英格兰，那里的排队看起来已经很完善了），或者其他服务中心的FIFO排队，在那里到达的顾客加入单一等待队列的最后，保持次序，然后当他们到达队伍的前面时就接受服务。

图7-22表示了工作中的FIFO队列。分组的到达由上部时间线上带编号的箭头来指示，该编号指示了分组到达的次序。各个分组的离开表示在下部时间线的下面。分组在服务中（被传输）花费的时间是通过这两个时间线之间的阴影矩形来指示的。由于FIFO规则，分组按照到达的同样次序离开。注意在分组4离开之后，链路在分组5到达之前保持空闲（因为分组1～4已经被传输并从队列中去除）。

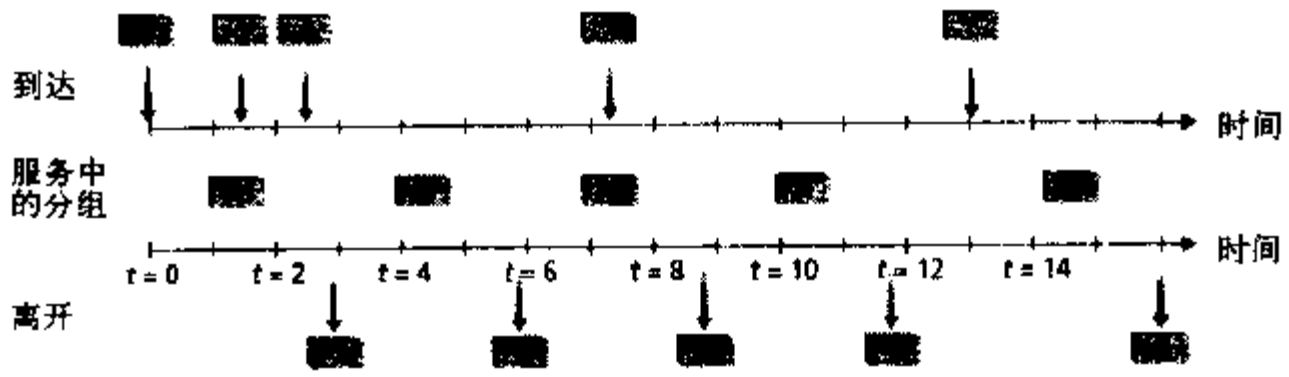


图7-22 操作中的FIFO队列

（2）优先级排队

使用了**优先级排队**（priority queuing）规则后，到达输出链路的分组被分类成输出队列中的优先级类，如图7-23中所示。正如上一节讨论的，一个分组的优先级类可能取决于在它的分组首部携带的一个明显的标记（例如，在一个IPv4分组中ToS比特的值）、它的源或者目的地址、它的目的端口号或者其他标准。每个优先级类通常都有自己的队列。当选择一个分组

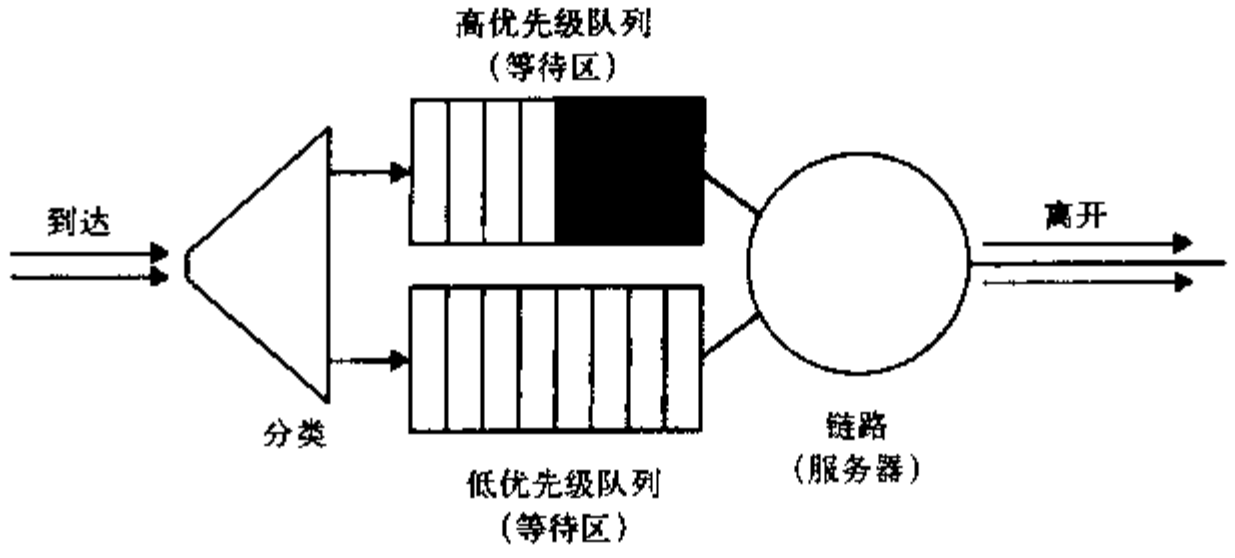


图7-23 优先级排队模型

传输时，优先级排队规则将从队列为非空（也就是有分组等待传输）的最高优先级类中传输一个分组。在同一优先级类的分组之间，选择方式通常以FIFO方式完成。

图7-24描述了有两个优先级类的优先级队列的操作。分组1、3和4属于高优先级类，分组2和5属于低优先级类。分组1到达并发现链路是空闲的，就开始传输。在分组1的传输过程中，分组2和3到达，并分别在低优先级和高优先级队列中排队。在传输完分组1，分组3（一个高优先级的分组）被选择在分组2（即使它到达的较早，但它是一个低优先级分组）之前传输。在分组3的传输结束后，分组2然后开始传输。分组4（一个高优先级分组）在分组2（一个低优先级分组）的传输过程中到达。在所谓的非抢占优先级排队规则下，一旦分组开始传输，传输就不能打断。在这种情况下，分组4排队等待传输，并在分组2的传输完成之后开始传输。

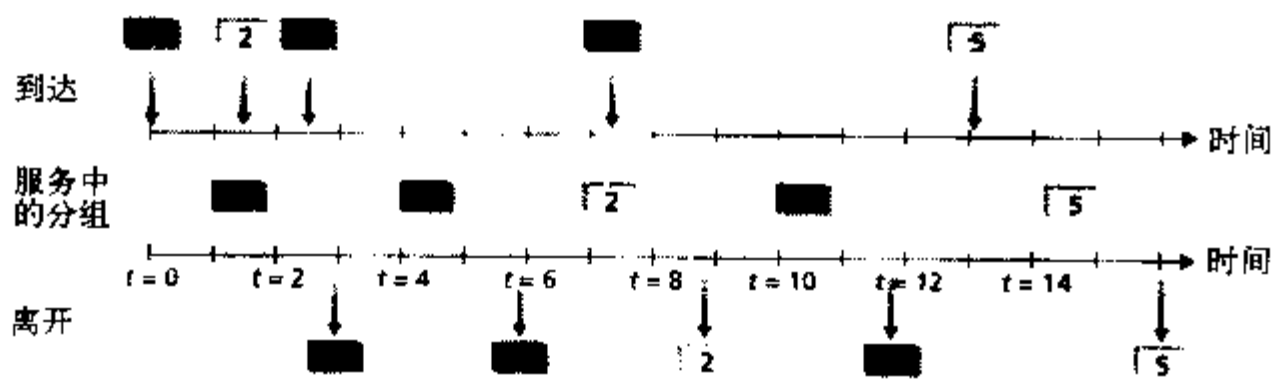


图7-24 优先级队列的操作

（3）循环和加权公平排队

使用**循环排队规则**（round robin queuing discipline）后，分组和优先级排队一样也被分成类。然而，在类之间不存在严格的服务优先级，循环调度器在这些类之间轮流提供服务。在最简单的循环调度形式中，类1的分组被传输，接着是类2的分组，接着又是类1的分组，再接着又是类2的分组等等。一个所谓保持工作的排队规则是说在有（任何类的）分组排队等待传输时，不允许链路保持空闲。当寻找给定类的分组但是没有找到时，**保持工作的循环规则**（work-conserving round robin discipline）将立即检查循环序列中下一个类。

图7-25描述了有两个类的循环队列的操作。在这个例子中，分组1、2和4属于第一类，分组3和5属于第二类。分组1一到达输出队列就立即开始传输。分组2和3在分组1的传输过程中到达，因此排队等待传输。在分组1传输后，链路调度器寻找类2的分组，因此传输分组3。在分组3传输完成后，调度器寻找类1的分组，因此传输分组2。在分组2传输完成后，分组4是唯一的排队的分组；因此在分组2后立刻传输分组4。

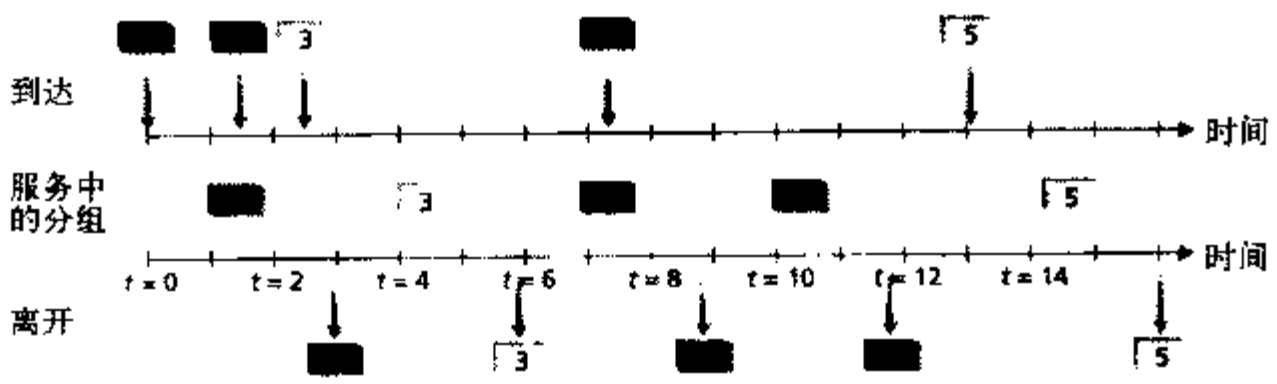


图7-25 有两个类的循环队列的操作

在QoS体系结构中已经得到大量使用的循环排队的一个通用抽象就是所谓的**加权公平排队**（Weighted Fair Queuing, WFQ）规则[Demers 1990; Parekh 1993]。图7-26对WFQ进行了描述。

到达的分组也被分类并在合适的每个类的等待区域排队。和循环调度一样，WFQ调度器也以循环的方式为各个类提供服务，即首先服务第1类，然后服务第2类，接着再服务第3类，然后（假设有3个类别）重复这种服务模式。WFQ也是一个保持工作的排队规则，因此在发现一个空的类队列时，它立即移向服务序列中的下一个类。

WFQ和循环排队不同之处在于，每个类在任何时间间隔内可能收到不同数量的服务。具体而言，每个类i被分配一个权w_i。使用WFQ方式，在类i有分组要发送的任何时间间隔中，第i类将确保接收到服务等于$w_i/(\Sigma w_j)$的那部分，式中分母中的和是计算所有有分组排队等待传输的类别得到的。在最坏的情况下，即使所有的类都有分组排队，第i类仍然保证分配到的带宽的$w_i/(\Sigma w_j)$部分。这样，对于一条传输速率为R的链路，第i类总能获得至少为$R \cdot w_i/(\Sigma w_j)$的吞吐量。我们对WFQ的描述理想化了，因为我们没有考虑这样的事实：分组是离散的数据单元并且不能将一个分组的传输打断来开始传输另一个分组；[Demers 1990]和[Parekh 1993]讨论了这个分组化问题。正如我们将在下面的几节中看到的，WFQ在QoS体系结构中起了重要的作用。现在的路由器产品中也用到了WFQ[Cisco QoS 2007]。

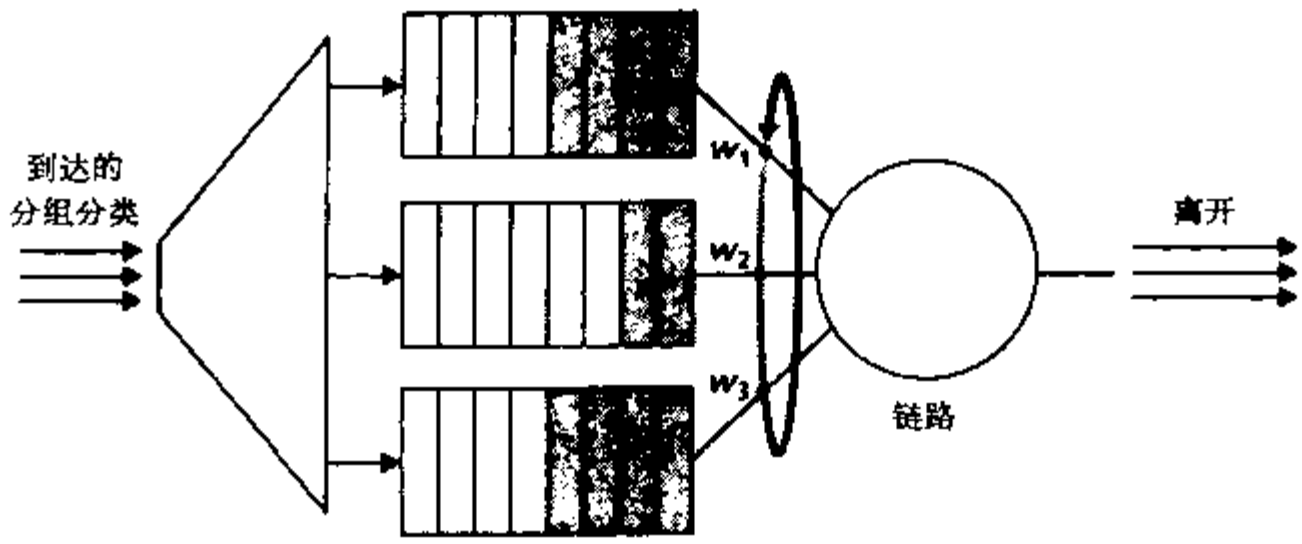


图7-26 加权公平排队

2. 监管：漏桶

7.5.1节中我们得到的直觉法则之一是监管，调节一类或流向网络注入分组的速率是一种重要的QoS机制，在下面的讨论中我们将假定监管的单位是流。但是应该对一个流的分组速率的哪些方面进行监管呢？我们能够确定3个重要的监管准则，每个准则根据被监管分组流的时间范围而互不相同：

- 平均速率。网络可能希望限制一个流的分组能够发送到网络中的长期平均速率（每个时间间隔的分组数）。这里一个关键的问题是监管平均速率的时间间隔。一个平均速率被限制为每秒100个分组的流要比一个每分钟6000个分组的源受到的约束更严格，即使在一个足够长的时间间隔上它们有相同的平均速率。例如，后者的限制允许一个流在给定1 s长的时间间隔内发送1000个分组，而前者的限制不允许这种发送行为。
- 峰值速率。平均速率约束限制了在一个相对长的时间能够发送到网络中的流量，而峰值约束限制了在一个较短时间能够发送的最大分组数。使用我们上面的例子，网络可以以每分钟6000个分组的平均速率来监管一个流，但是限制该流的峰值速率为每秒1500个分组。
- 突发长度。网络也许还希望对在极短的时间间隔能够发送到网络中的最大分组数（分组的“突发”）进行限制。在这个限制中，因为时间间隔长度趋近于0，该突发长度限制了

能够瞬间发送到网络中的分组数量。即使瞬间发送多个分组到网络中在物理上是不可能的（毕竟每条链路都有一个无法超越的物理传输速率！），对最大突发长度的抽象也是有用的。

漏桶机制是一个能够用来表征这些监管限制的抽象。如图7-27所示，漏桶由一个能够容纳b个令牌的桶组成。令牌加进该桶的过程如下。可能潜在地加入桶中的新令牌总是以每秒r个令牌的速率产生。（为了简单起见，我们这里假设时间单元是秒。）当一个令牌产生时，如果桶中少于b个令牌，新产生的令牌加入到该桶中；否则忽略该新产生的令牌，令牌桶保持具有b个令牌的满状态。

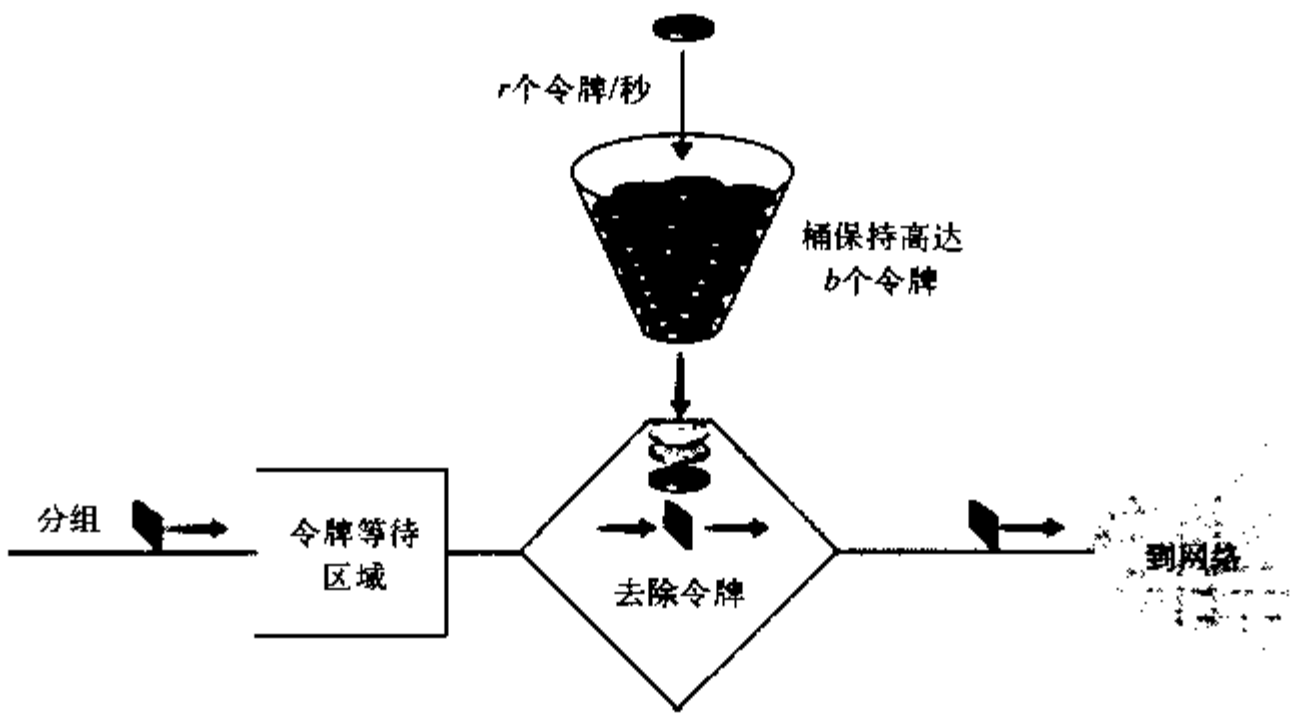


图7-27 漏桶监管器

现在我们考虑如何用漏桶来监管分组流。假设在一个分组向网络传输之前，必须首先从令牌桶中去除一个令牌。如果令牌桶是空的，分组必须等待一个令牌。（另一种方法是丢弃该分组，尽管我们这里不讨论这种选择。）现在我们考虑这种行为是如何监管一个通信流的。因为在桶中最多能有b个令牌，漏桶监管的流的最大突发长度是b个分组。而且因为令牌产生速率是r，在任何长度t的时间间隔内能够进入到网络中的最大分组数目为$rt+b$。因此令牌产生速率r用于限制分组能够进入网络的长期平均速率。除了监管长期平均速率之外，使用漏桶（具体来说，串联的两个漏桶）来监管流的峰值速率也是有可能的；见本章末的习题。

漏桶＋加权公平排队=在队列中的可证明的最大时延

我们很快将研究在因特网中提供服务质量的所谓的综合服务（Intserv）和区分服务（Diffserv）方法。我们看到漏桶监管和WFQ调度都能够起重要的作用。因此在本节结束之前，我们使用WFQ调度考虑对n个流进行多路复用的路由器的输出，其中每个流被一个参数为b_i和r_i的漏桶监管，$i=1,\ldots,n$。这里我们使用术语“流”大致是指不能被调度器相互区别的分组集合。实践中，一个流可能是由单个端到端连接上的流量或者许多这种连接的集合的流量组成的，参见图7-28。

我们前面讨论WFQ时讲过，每个流i保证收到至少等于$R \cdot w_i/(\sum w_j)$的共享链路带宽，这里R是以分组/秒为单位的链路传输速率。那么当以WFQ方式等待服务时（也就是通过漏桶传递之后），分组经受的最大时延是什么？我们关注于流1。假设流1的令牌桶最初是满的。然后b_1个分组的突发到达流1的漏桶监管器。这些分组去除了漏桶中所有的令牌（没有等待），然后

加入了流1的WFQ等待区域。因为这些b_1个分组以至少$R \cdot w_i/(\Sigma w_j)$分组/秒的速度被服务，那么直到这些分组的最后一个传输完成，将有最大时延d_{max}，这里

$$d_{max} = \frac{b_1}{R \cdot w_1 / \Sigma w_j}$$

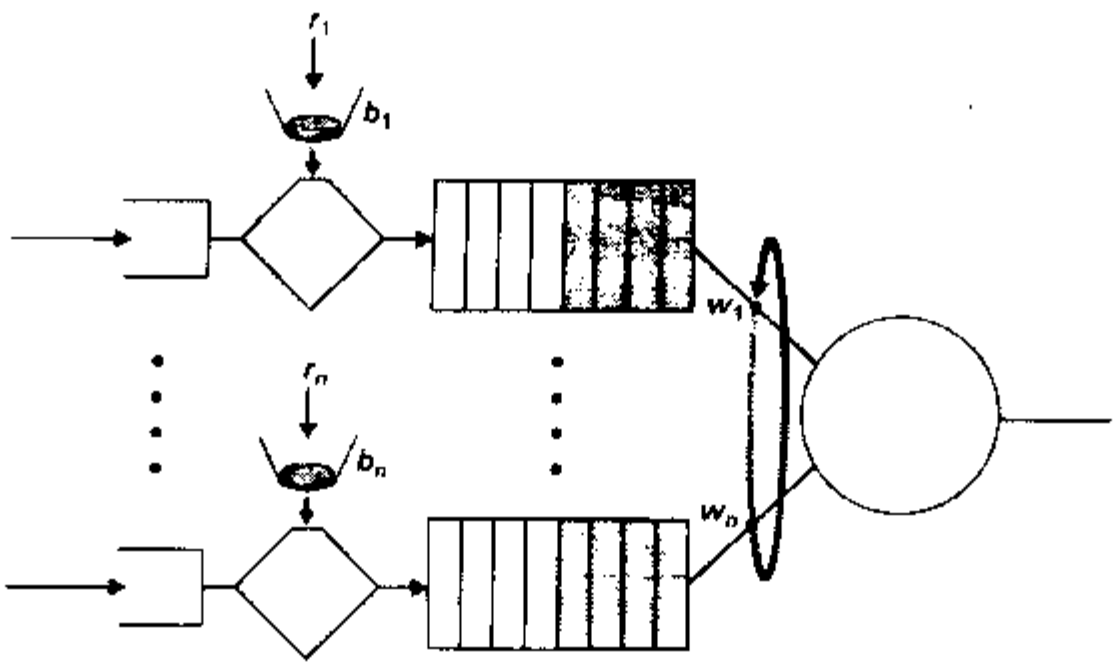


图7-28 采用WFQ调度机制的n路复用的漏桶流

这个公式的基本原理是，如果在队列中有b_1个分组并且分组以至少每秒$R \cdot w_i/(\Sigma w_j)$个分组的速率从队列中接受服务（被去除），那么直到最后一个分组的最后一个比特被传送，时间量不会超过$b_1/(R \cdot w_i/(\Sigma w_j))$。有一道习题要求读者证明，只要$r_1<R \cdot w_i/(\Sigma w_j)$，那么$d_{max}$确实是流1中任何分组在WFQ队列中要经受的最大时延。

7.5.3 区分服务

因特网区分服务（Diffserv）体系结构[RFC 2475; Kilkki 1999]的目标是提供服务区分，也就是在因特网中用不同的方式处理不同“类别”流量的能力，并以一种可缩扩和灵活的方式进行。*可缩扩性*的需求来源于这样的事实：在因特网的一个主干路由器上同时存在几十万个源到目的的并行流。我们不久将看到，仅仅通过在网络的核心放置简单的功能，而在网络的“边缘”实现更复杂的控制操作，就可以满足该需要。对*灵活性*的要求来源于这样的事实：可能出现的新的服务类别和旧的服务类别可能变得过时。区分服务体系结构从它没有定义具体的服务或者服务类别的意义上来说是很灵活的。相反，区分服务体系结构提供了功能组件，即网络体系结构中的各部分，这些服务可以通过这些组件来构造。我们现在仔细研究一下这些组件。

1. *区分服务：一种简单的情况*

为了设置用于定义区分服务模型的体系结构组件框架，我们从图7-29表示的简单网络来开始讨论。在本节中，我们描述Diffserv组件的一种可能用法。如RFC 2475中描述的那样，可能有许多其他的变化。我们这里的目标是介绍区分服务的重要方面，而不是非常详细地描述这个体系结构的模型。想学习更多Diffserv知识的读者可以阅读一本内容全面的书[Kilkki 1999]。

区分服务体系结构由两个功能元素集合组成：

- *边界功能：分组分类和流量调节*。在网络的入口边缘（也就是在产生流量的Diffserv使

能的主机或者在流量经过的第一个Diffserv使能的路由器上），到达的分组被标记。更具体地说，分组首部的区分服务（DS）字段被设置为某个值。例如，在图7-29中，从H1发送到H3的分组可能在R1被标记，而从H2发送到H4的分组可能在R2被标记。分组得到的标记标识了该分组所属的流量类别。于是不同类别的流量在核心网络接受到不同的服务。

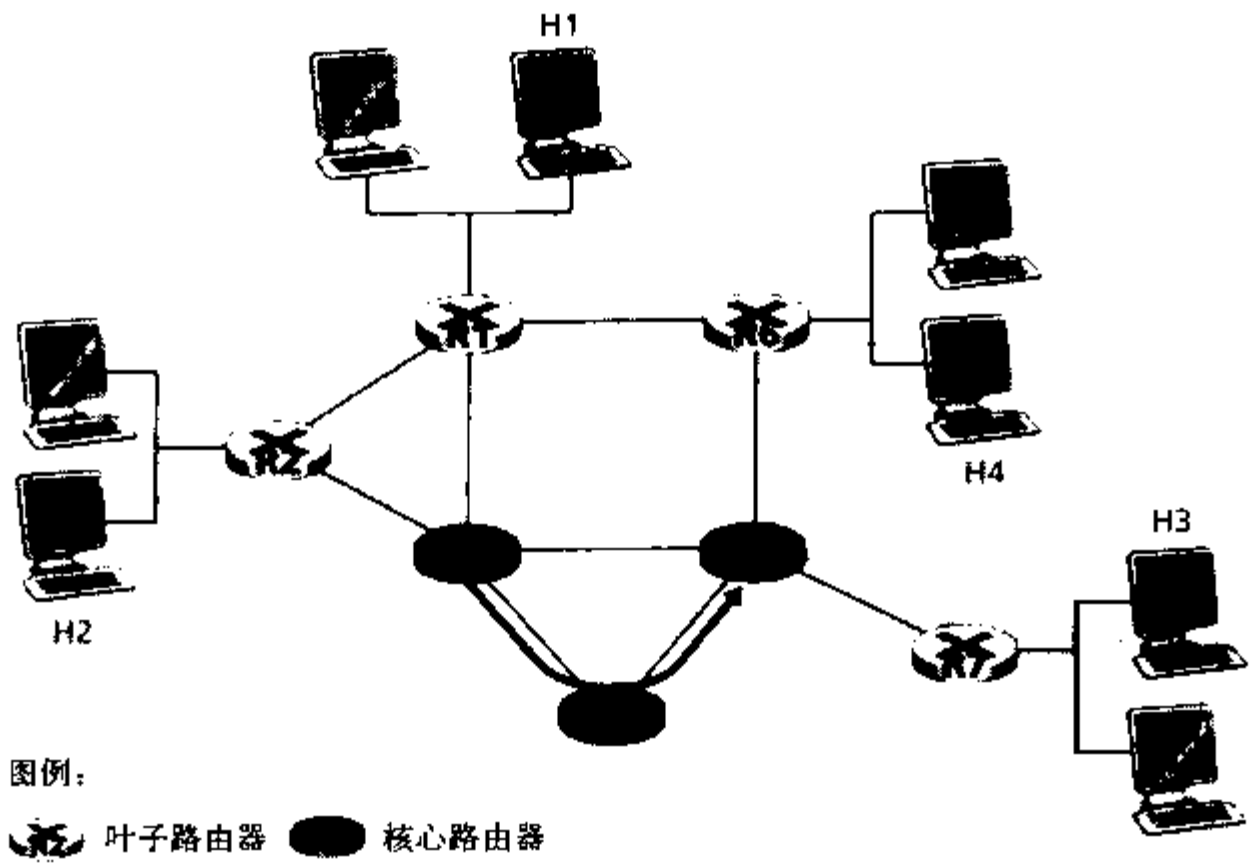


图7-29 一个简单的Diffserv网络的例子

- *核心功能：转发。*当一个DS标记的分组到达一个有Diffserv能力的路由器时，根据与分组类别相关的所谓的每跳行为（per-hop behavoir），该分组被转发到下一跳。每跳行为影响在竞争的流量类别之间共享路由器的缓冲区和链路带宽的方式。Diffserv体系结构的一个关键原则是路由器的每跳行为只基于分组标记，即分组所属的流量类别。因此，如果图7-29中从H1发送到H3的分组和从H2发送到H4的分组收到同样的标记，那么网络路由器将这些分组作为聚合体处理，而不区别这些分组是H1产生的还是H2产生的。例如，当这些分组向R4转发时，R3不区分是从H1还是从H2产生的分组。因此，区分服务体系结构消除了为各个源到目的地对而保留路由器状态的要求，这是满足本节一开始讨论的可缩扩性需求的一个重要考虑。

这里有一个类比可能有助于理解相关概念。在很多大规模的社会活动（例如，一个大型的公众招待会、一个大型跳舞俱乐部或者迪斯科舞厅、一场音乐会或者一场足球赛）中，参加活动的人们收到某种类型的入场券。对于非常重要的人物有VIP入场券；对21岁或者年龄更大的人有21岁以上的入场券（例如是否可以享受酒类饮料）；在音乐会上有后台入场券；对于记者有新闻入场券；对于普通人有普通的入场券。这些入场券通常在该活动的入口分发，也就是在该活动的边缘进行分发。正是在边缘，进行着计算密集型操作，例如交入场费、检查适合的邀请类型以及对照检查邀请与证件。况且，对于允许进入某个活动的特定类型的人数可能有限制。如果有这种限制，人们可能在进入活动之前必须等待。一旦进入了活动，一个人的入场券使他在活动中的很多场所接受有区别的服务，如给一个VIP提供免费的饮料、较好的桌子、免费食物、进入单独的房间和殷勤的服务。与之相对照的是，一个普通人禁止进

入某些区域，要为饮料支付费用，并且只享受到基本服务。在这两种情况下，活动中得到的服务只依赖于入场券的类型，而且同一类别的所有人得到相同的对待。

2. Diffserv流量分类和调节

图7-30提供了在边缘路由器中分类和标记功能的逻辑视图。到达边缘路由器的分组首先被分类。分类器根据一个或多个分组首部字段的值（例如源地址、目的地址、源端口、目的端口和协议ID）来选择分组，并引导该分组去做合适的标记功能。分组的标记携带在IPv4或IPv6分组首部的DS字段[RFC 3260]中。定义DS字段的意图是取代早期的IPv4服务类型字段的定义和我们在第4章中讨论的IPv6流量类型字段。

在某些情况下，端用户可能认可了限制其分组发送速率以符合某个**流量配置文件**（traffic profile）的声明。该流量配置文件可能包含对峰值速率和分组流的突发度的限制，如我们前面在漏桶机制中见到的那样。只要用户以符合协商的流量配置文件的方式发送分组到网络中，这些分组就会得到它们的优先级标记，并沿着到目的地的路径转发。另一方面，如果违反了该流量配置文件，那些超出流量配置文件的分组就可能被打上不同的标记，或被整形（例如为了能够遵守最大速率限制而延时），或可能在网络边缘被丢弃。图7-30中所示的**测定功能**（metering function）的作用是比较进入的分组流和协商的流量配置文件，并判断某分组是否在协商的流量配置文件之内。是否立即重新标记、转发、延时或者丢弃一个分组，这是由网络管理员决定的策略问题，而不是由Diffserv体系结构规定的。

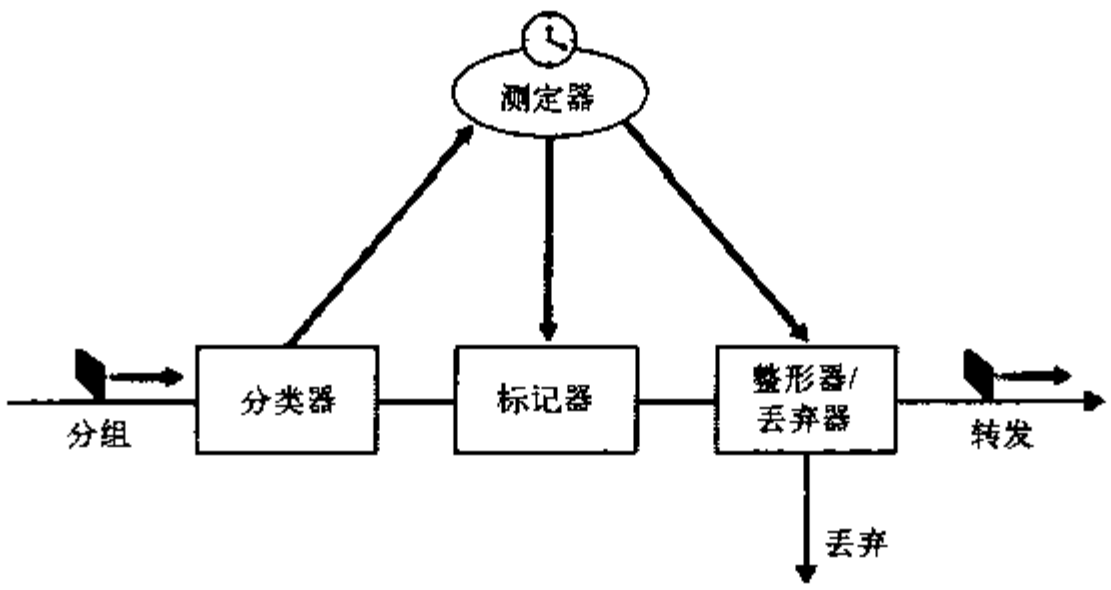


图7-30 在端路由器上的分组分类与流量调节的逻辑视图

3. 每跳行为

到现在为止，我们一直关注着区分服务体系结构中的边缘功能。Diffserv体系结构的第二个关键组件包含了由Diffserv使能路由器所执行的**每跳行为**（Per-Hop Behavior, PHB）。该每跳行为有点神秘，但被仔细地定义为“Diffserv 节点的外部可观察的转发行为的描述，该节点应用了一个特定的Diffserv行为聚合”[RFC 2475]。稍微深入地钻研一下这个定义，我们可以看到它里面包含的几个重要的考虑：

- PHB能够导致不同服务类别流量收到不同性能（即不同的外部可观察的转发行为）。
- 虽然PHB在类别之间定义性能（行为）差别，它不强求任何特定机制以获得这些性能。只要外部可观察的性能准则得到满足，任何实现机制和任何缓冲区/带宽分配策略都可以使用。例如，一个PHB不要求为了获得一个特定的行为而使用特定的分组排队规则（例如优先级队列、加权公平排队队列或先来先服务队列）以取得特定的行为。PHB是

目标，资源分配和实现机制是达到它的手段。

- 性能的差别必须是可以观察到的，并且因此是可以测量的。

当前，已经定义了两种PHB：一种是加速转发（Expedited Forwarding, EF）PHB [RFC 3246]，另一种是确保转发（Assured Forwarding, AF）PHB [RFC 2597]。

- **加速转发**PHB规定了一类流量离开路由器的速率必须等于或者大于一个已配置的速率。即在任何时间间隔内，能够保证该类别流量收到足够的带宽以便该流量的输出速率等于或者大于这个最小配置速率。注意到EF每跳行为隐含着流量类别之间的某种形式的隔离，因为这种保证是独立于到达路由器的其他类别的流量强度而做出的。因此，即使其他类别的流量抢占了绝大多数路由器和链路带宽资源，仍然必须使得该类别得到足够的资源来确保它得到其最小速率保证。这样EF为一个类别提供了具有最小保证链路带宽的链路的简单抽象。
- **确保转发**PHB更为复杂。AF将流量分为4类，这里每个AF类都确保提供某种最小数量带宽和缓冲区。在每个类别中，分组进一步划分为3个丢弃优先等级种类中的一个。当在一个AF类中发生拥塞时，路由器能够根据分组的丢弃优先等级值来丢弃分组。详细的内容参见[RFC 2597]。通过改变分配给每类的资源数量，ISP能够向不同的AF流量类别提供不同的性能等级。

4. Diffserv回顾

近20年来，人们进行了许多向分组交换网络中引入QoS的尝试（绝大部分不成功）。到目前为止，各种尝试之所以失败的原因主要是因为经济和已有系统的原因而不是技术原因所致。这些尝试包括端到端ATM网络和TCP/IP网络。现在我们看一下在Diffserv环境（我们将在下节中简要研究）中所涉及的一些问题。

到目前为止我们隐含地假设Diffserv部署在单个管理域中。更典型的情况是必须由跨越通信端系统之间的多个ISP来形成端到端的服务。为了提供端到端的Diffserv服务，端系统之间的所有ISP不仅必须提供这种服务，而且为了向端用户提供真正的端到端服务，多数ISP之间还要进行协作并作出安排。如果没有这种协作，直接向客户机出售Diffserv的ISP将不得不重复地说“是的，我们知道你支付了额外费用，但是我们和更高层的ISP之间没有服务约定。我很抱歉在你的VoIP电话中有很多间隙！”

即使在一个单一的管理域中，仅有Diffserv是不足以提供对一种特定服务类型的服务质量保证的。Diffserv仅使得不同类型的流量得到不同等级的性能。如果一个网络严重地低于规划水平，即使高优先权类型的流量也不可能得到可接受的良好性能。因此，为了使这种机制有效，Diffserv必须与适当的网络规划相配合（参见7.3.5节）。然而，Diffserv能够使得ISP在网络容量中的投资发挥更大作用。通过使得资源对高优先权（且高付费）的流量在它任何需要的时候可用，ISP能够向这些高优先权类型交付更高等级的性能。当这些资源不为高优先权类型所需要时，它们能够由较低优先权流量类型所用（假定为这种较低服务类型支付较少的费用）。

对这些先进的服务的另一个关注是它们需要监管和可能的流量整形，这可能导致复杂和较大花费。人们还需要为服务支付不同的费用，可能主要根据数量而不是像大多数ISP现在所用的每月支付固定费用的方式，这是对ISP的另一个开销大的要求。最后，如果Diffserv实际存在并且该网络运行的负载不大，大多数时间尽力而为服务和Intserv/Diffserv服务之间将没有明显的差异。事实上，现在端到端的时延通常是由接入速率和路由器跳数造成的，而不是由

路由器中的排队时延造成的。想象一下下列情景：一个支付了增值服务的不幸Intserv/Diffserv客户，发现为其他人提供的尽力而为服务几乎总是具有与增值服务相同的性能！

7.6 提供服务质量保证

在前一节中我们已经看到了分组标记和监管、流量隔离和链路级调度能够为一类服务提供比另一类更好的性能。在某些调度规则下，如优先权调度，较低优先权类型的流量对最高优先权类型的流量而言基本“不可见”。借助于适当的网络规划，最高类型的服务的确能够取得极低的丢包和时延，基本上是类似于电路的性能。但是，这种网络能够确保在一种高优先权流量类型中进行中的流，仅使用我们已经描述的这些机制就能在整个流期间持续得到这样的服务吗？这是不行的。在这节中，我们将看到需要另外的网络机制和协议提供服务质量保证的原因。

7.6.1 一个有启发的例子

我们返回7.5.1节的场景中，考虑两个1 Mbps的音频经1.5 Mbps的链路传输它们的分组，如图7-31所示。这两个流总数据率（2 Mbps）超过了这条链路容量。即使使用分类和标记、流量的隔离和未使用带宽的共享（这里并没有未使用的带宽）等机制，这很明显也是一个与丢包有关的命题。因为没有足够的带宽同时满足应用的需求。如果这两个应用平等地共享该带宽，每个只会收到0.75 Mbps。从另一个角度来看，每个应用都会丢失25%的传输分组。这是一个无法接受的低QoS，以至于两个音频应用完全不可用；甚至首先没有必要传输任何音频分组。

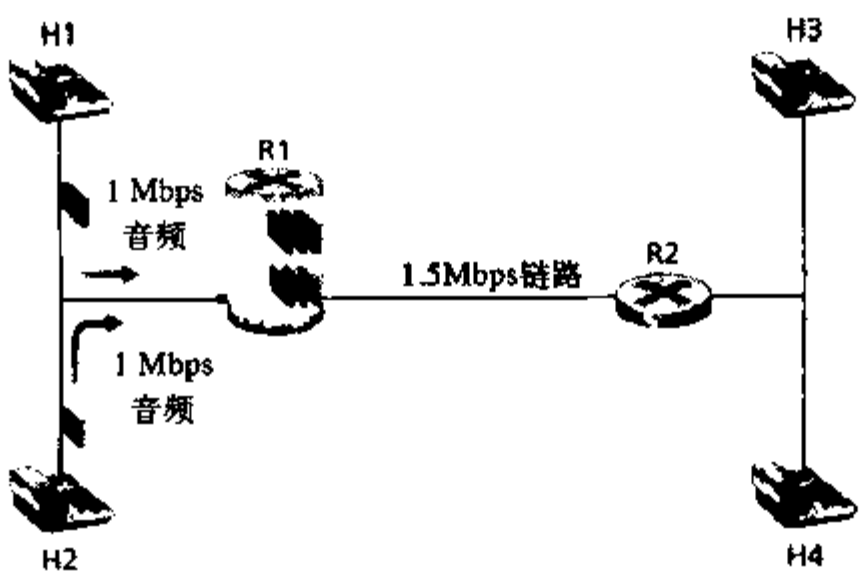


图7-31 两个竞争的音频应用过载R1到R2的链路

在图7-31中给出的两个应用不能同时被满足，这个网络应当做些什么呢？允许两者以一种不可用的QoS继续浪费应用流上的网络资源，最终没有向端用户提供任何效用。答案十分清楚，应当阻塞应用流中的一个（即拒绝接入网络），而另一个应当允许继续进行，使该应用使用所需的全部1Mbps。电话网就是一个执行这种呼叫阻塞的网络例子，如果要求的资源（在电话网的情况下，是一个端到端的电路）不能分配给该呼叫，该呼叫就被阻塞了（阻止进入该网络），并且返回给用户一个忙信号。在我们的例子中，如果一个流没有分配到足够的QoS来使得自己成为可用的，允许它进入网络没有任何好处。事实上，接纳一个不能得到它需要的QoS的流是要付出代价的，因为网络资源被用于支持一个对端用户没有用的流。

通过基于流的资源要求来明确地准入或阻塞流，同时考虑已准入流的资源要求，网络能

够确保准入的流将得到它们所请求的QoS。需要为流提供有保证的QoS，隐含的是需要该流来声明它的QoS需求。让流声明它的QoS需求，然后让网络接受该流（以所要求的QoS）或者阻塞该流的过程称为呼叫准入（call admission）过程。这是我们对提供QoS的机制的第4个直觉法则（另外3个直觉法则见7.5.1节）。

直觉法则4：如果充分的资源并不总是可用，并且QoS要被保证，则需要一个呼叫准入的过程，其中的流声明它们的QoS需求，然后要么被网络准入（以所要求的QoS），要么被网络阻塞（如果网络不能提供所要求的QoS）。

7.6.2 资源预约、呼叫准入、呼叫建立

我们有启发的例子强调了需要几种新的网络机制和协议，如果一个呼叫（一个端到端流）确保能得到给定的服务质量，一旦它开始则需要：

- *预留的资源。*为确保一个呼叫具有所需的资源（链路带宽、缓存）以满足它所需的QoS，其唯一方法是显式地为该呼叫分配这些资源，用网络的行话来讲是**资源预约**（resource reservation）的过程。一旦资源被预约，该呼叫在其整个过程中按需访问这些资源，而不管所有其他呼叫的需求。如果一个呼叫预约并得到链路带宽的x Mbps的保证，并且该呼叫的传输速率从不大于x，那么该呼叫将得到无丢包和无时延的性能。
- *呼叫建立。*如果预约了资源，则该网络必须具有一种用于呼叫请求和预约的机制，即一种称为呼叫准入的过程。由于资源不是无限的，如果请求的资源不可用，则进行呼叫准入请求的呼叫将被拒绝准入，即被阻塞。电话网执行的就是这种呼叫准入，当我们拨一个号码时，就请求了资源。如果完成该呼叫所需的电路（TDMA时隙）是可用的，分配电路并且完成呼叫。如果电路不可用，则该呼叫被阻塞，我们得到了忙音。为了得到网络的准入，被阻塞的呼叫能够一再尝试，但是直到它成功地完成呼叫准入过程，才允许向网络发送流量。

当然，就像1.3.1节中的餐馆经理不应当接受比本餐馆具有的餐桌更多的预约那样，分配链路带宽的路由器不应当分配超过链路可用带宽的资源。通常，一个呼叫仅能预约一条链路带宽的一部分，因此一台路由器可以为多于一条呼叫分配链路带宽。然而，为所有呼叫分配的带宽总和应当小于该链路的容量。

*呼叫建立信令。*上面描述的呼叫准入过程要求一条呼叫能够沿着它的源到目的地路径上的每台网络路由器预约充足的资源，以确保它的端到端QoS请求得到满足。每台路由器必须决定会话所请求的本地资源，考虑已经承诺的其他进行中的会话的资源量，决定它在这台路由器上是否有足够的资源来满足该会话的每跳QoS要求，而不违反对所有已经准入的会话作出的本地QoS保证。需要一个信令协议来协调上述各种活动，即本地资源的每跳分配，以及该呼叫是否能够在沿着端到端路径上的每台路由器预约充分的资源。这是**呼叫建立协议**（call setup protocol）的任务。

图7-32描绘了呼叫建立过程。我们现在更详细地考虑呼叫准入涉及的步骤：

1）*所希望的QoS的流量特征和规范。*要让一台路由器来确定它的资源是否足以满足一个会话的QoS需求，那个会话必须首先声明它的QoS需求，并对它将要发送到网络中的流量的特征进行描述，以及为此它需要一个QoS保证。在Intserv体系结构中，所谓的Rspec（R为预留之意）[RFC 2215]定义了一个呼叫要求的具体QoS；所谓的Tspec（T为流量之意）[RFC 2210]分别刻画发送方要发送到网络中的流量或者接收方要从网络中接收的流量。Rspec和Tspec的

具体形式根据需要的服务的不同而变化，如下面讨论的那样。在ATM网络中，用户流量描述和QoS参数信息元素分别携带了与Tspec和Rspec类似目的的信息；详情参见[Black 1997]。

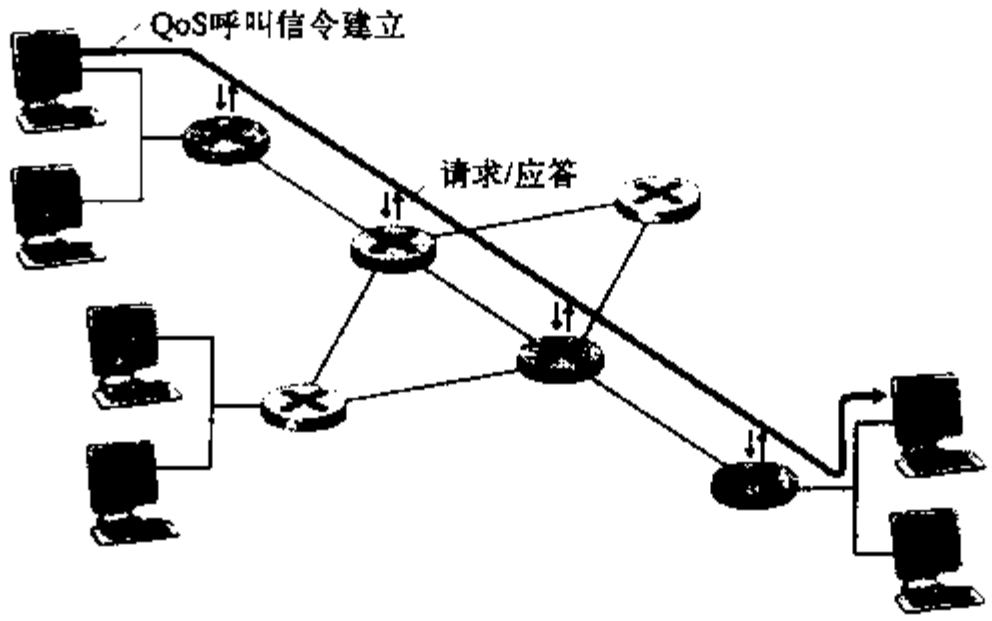


图7-32 呼叫建立过程

2）呼叫建立的信令。一个会话的流量描述和QoS请求必须被携带给路由器，在路由器中为该呼叫预约资源。在因特网中，在Intserv体系结构中RSVP协议[RFC 2210]用于这个目的。在ATM网络中，Q2931b[Black 1997]协议在ATM的交换机和端点之间携带这种信息。

3）每个元素呼叫准入。一旦路由器接收到流量规格参数和QoS，它必须决定是否准入该呼叫。这个呼叫准入决定将取决于流量规格参数、请求的服务类型和已经由路由器对正在进行的呼叫作出承诺的现有资源。例如，7.5.3节讲过，我们看到过一个由漏桶控制的资源和WFQ结合能够用于确定对某源的最大排队时延。每个元素的呼叫准入情况如图7-33所示。

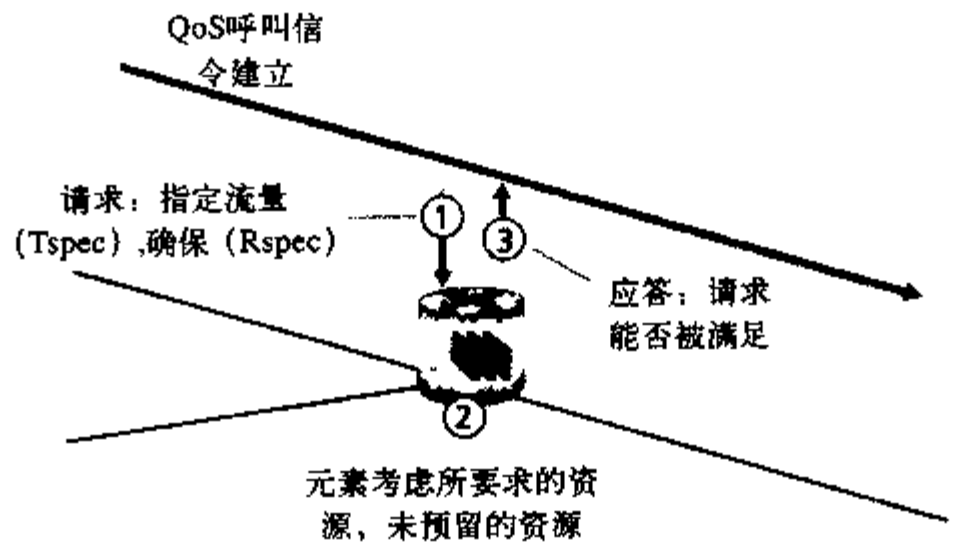


图7-33 每个元素的呼叫行为

7.6.3 在因特网中确保QoS：Intserv和RSVP

综合服务（Intserv）体系结构是一个由IETF研发的框架，用于在因特网中为各个应用会话提供个性化的QoS保证。Intserv的确保服务规范定义在[RFC 2212]中，为分组提供了在路由器中经受的排队时延的严格的（数学上可证明的）界限。尽管有保证服务背后的细节相当复杂，但是基本思想非常简单。作为第一个近似，一个源的流量特征通过一个参数为（r，b）的漏桶（见7.5.2节）来给出，并且需要的服务通过分组传输的传输速率R来刻画。本质上说，请求有保证服务的呼叫要求保证它分组的比特转发速率为R bit/s。给定流量定义为具有漏桶特

性，并且要求确保速率R，限制该路由器的最大排队时延还是可能的。前面讲过，在具有漏桶流量特性的情况下，在任何长度t的时间间隔内产生的流量的数量（以比特为单位）被限制在$rt+b$以内。在7.5.2节讲过，当一个漏桶的源被注入一个队列时，且该队列能保证排队的流量至少以每秒R比特速率服务，只要R大于r，任何分组经受的最大排队时延将限定为b/R。Intserv服务确保的第二种形式也已经被定义了，称之为受控的负载服务，它定义为：一个呼叫将接受“与相同流在无负载网络单元中获得的QoS非常接近的服务质量”[RFC 2211]。

实践原则

软状态原则

RSVP被用于在路由器中建立状态（带宽预留），并被称为软状态协议。广义而言，我们将术语软状态与信令方法相联系，除非周期性地接收到信令报文（通常来自最初建立该状态的实体）进行刷新，否则所建立的状态超时（并被去除）。信令报文指示了该状态应当继续保持。因为未刷新的状态将最终超时，软状态信令既不显式地要求去除状态，也不要求有一个过程在状态建立者崩溃时来去除孤立状态。类似地，状态建立和刷新报文将跟随着后继的周期性刷新报文，不需要可靠的信令。术语软状态由Clark [Clark 1988]提出，他描述了通过端系统周期性地发送状态刷新信息的概念，并提出使用这种刷新报文，在崩溃中状态能够丢失并能自动地由后继刷新报文恢复，这些都对端系统透明并且不调用任何明确的崩溃恢复过程：

“……在维护所希望的与流相联系的服务类型时，状态信息将不是至关重要的。相反，服务的类型将由端点所强制，即周期性地发送报文以确保适当的服务类型与该流相联系。以这种方法，与该流相联系的状态信息在崩溃中可能丢失，而不会永久地破坏所使用的服务特性。我们称这个概念为“软状态”，它也许能让我们很好地取得生存能力和灵活性的基本目标……”

粗略地讲，软状态方法的本质是使用了尽力而为的周期性状态建立/刷新，这是由状态建立者和状态保持者因超时而去除状态来提供的。软状态方法已经用于若干种协议中，包括RSVP、PIM（4.7节）、SIP（7.4.3节）和IGMP（4.7节），以及在透明网桥的转发表中（5.6节）。

硬状态信令采取与软状态相反的方法，即建立的状态保持不变，除非从状态建立者收到状态拆除报文后而明确地去除。因为其状态保持建立时不变，除非明确去除，硬状态信令要求有一种机制，在状态建立者崩溃或离开而没有去除状态后，能去除孤立状态。类似地，状态建立和去除仅执行一次（并且无状态刷新或状态超时），状态建立者知道何时建立或去除状态是重要的。可靠的（而不是尽力而为的）信令协议因此通常与硬状态协议相联系。粗略地讲，硬状态的本质是可靠的和明确的状态信息的建立和去除。硬状态方法已经用于诸如ST-II[Partridge 1992, RFC 1190]和Q.2931[ITU-T Q.2931 1994]等协议中。

RSVP自提出以来，已经提供了明确的（尽管可选）去除预留的方法。

基于ACK的可靠信令作为RSVP的扩展在[RFC 2961]中引入，并在[Pan 1997]中被建议。RSVP因此被某些硬状态信令方法有选择地采纳为某些元素。对于软状态与硬状态协议的讨论和比较参见[Ji 2003]。

资源预留协议（resource ReSerVation Protocol, RSVP） [RFC 2205; Zhang 1993]是一种因特网信令协议，能被用于执行由Intserv所需的执行呼叫建立的信令。RSVP也能与DiffServ结合使用以协调跨越多个网络DiffServ功能，并且已经进行了扩展，以及在其他场合下用于信令协议，也许最为引人注目的是用于MPLS信令的RSVP-TE[RFC 3209]，如在5.8.2节中所讨论的那样。

在Intserv环境下，RSVP协议允许应用为它们的数据流预留带宽。它由主机使用，代表应用数据流向网络要求特定量的带宽。路由器也使用RSVP来转发带宽预留请求。为了实现RSVP，在沿着图7-32中所示的端到端路径中的接收方、发送方和路由器中必须具有RSVP软件。RSVP两个主要的特征是：

- 它提供**多播树中的带宽预留**（reservations for bandwidth in multicast tree），单播被作为多播的退化情况来处理。这对于诸如流式广播经IP的TV等多媒体应用而言特别重要，其中许多接收方可能要接收来自单一源的相同多媒体流量。
- 它是**面向接收方**（receiver-oriented）的，也就是说数据流的接收方发起并维护用于该流的资源预留。RSVP所采用的这种创新性的、以接收方为中心视角的方法，使接收方有力地控制着它们接收到的流量，例如，允许不同的接收方接收并观察不同分辨率下的多媒体多播。这与ATM的Q2931b中所采用的以发送方为中心视角的信令形成对比。

RSVP标准[RFC 2205]没有定义网络向数据流提供预约带宽的方法，它只是一个允许应用预约必要链路带宽的协议。一旦某预约付诸实施，因特网中的路由器就实际向数据流提供预约的带宽。这种配备很可能使用7.5节讨论的监管和调度机制（漏桶，优先权调度，加权公平排队）来完成。有关RSVP更多的信息，参见[RFC 2205; Zhang 1993]和与其他本书相关联的在线电子材料。

7.7 小结

多媒体网络可能是当今因特网中最为激动人心（目前仍被公认）的发展方向之一。人们在收音机和电视机前花费的时间越来越少，而转向通过因特网来接收音频和视频传输，包含实况转播和预先录制的节目。随着高速接入渗透到更多的住宅，该趋势将继续下去，全世界终日懒散在家的人将通过因特网而不是通过传统的广播分发信道来访问他们最喜欢的视频节目。除了音频和视频分发，因特网也用于传输电话。实际上，未来的10年后，因特网可能致使传统的电路交换电话系统在很多国家中接近废弃。因特网不仅将以更少的钱来提供电话服务，而且将提供大量的增值服务，例如视频会议、在线目录服务、语音消息传递服务和Web集成。

在7.1节中，我们将多媒体应用分为3种类别：流式存储音频和视频，实时音频和视频的一对多传输，以及实时交互式音频和视频。我们强调的是多媒体应用是时延敏感和丢包容忍的（这与容忍时延而不容忍丢包的静态内容应用是截然不同的特征）。我们也讨论了在现在的尽力而为服务因特网中使用多媒体应用所面临的一些障碍。我们审视了克服这些障碍的几个建议，包括直接改善现有的网络基础设施（通过增加更多的带宽、更多的网络高速缓冲器、更多的CDN节点以及使用多播），为因特网增加功能以便应用能够预留端到端的资源（以便网络能够兑现这些预留），最后介绍了服务类别以提供区分服务。

在7.2节到7.4节中，我们研究了在尽力而为服务网络中的多媒体网络的体系结构和机制。

在7.2节中，我们考虑了流式存储音频和视频的几种体系结构。我们讨论了用户交互（例如暂停/继续、重定位和可视的快进），并介绍了RTSP（一种对流应用提供客户机服务器交互的协议）。我们在7.3节中研究了如何设计交互的实时应用使得它在尽力而为服务网络上运行。我们理解了结合使用客户机缓冲区、分组序号和时间戳能够大大减小网络引入的时延抖动的影响。在7.4节中，我们也讨论了CDN是如何通过提前将存储多媒体推进位于“邻近”用户端点的CDN服务器而加速流式存储多媒体的。

在7.5节中，我们介绍了几种网络机制（链路级调度规则和流量监管）如何用于在几类流量之间提供区分服务。最后，在7.6节中，我们研究了网络是如何为准入网络的呼叫提供服务质量保证的。这里，还需要另外的新型网络机制和协议，包括资源预约、呼叫准入和呼叫信令。这些新的网络元素共同使得明天的保证QoS的网络与今天的尽力而为的因特网十分不同（并且更为复杂）。

课后习题和问题

复习题

7.1～7.2节

1. 流式存储音频/视频交互性意味着什么？实时交互音频/视频交互性意味着什么？
2. 为了改进因特网以便它能够更好地支持多媒体应用，我们讨论了3个“阵营”。简要总结一下每个阵营的观点。你属于哪个阵营？
3. 在7.1节中所讨论的对图像和音频应用某些典型的压缩率（未压缩对象的比特数量与该对象的压缩版本的比特数之比）及其压缩技术是什么？
4. 图7-1和图7-2提出了流式存储媒体的两种方案。每一种方案的优点和缺点是什么？

7.3～7.4节

5. 端到端时延和分组时延抖动的区别是什么？分组时延抖动的原因是什么？
6. 为什么在预定的播放时间之后收到的分组被认为是丢失了？
7. 7.3节描述了两个FEC方案，简要地总结它们。这两种方案通过增加开销而增加了流的传输速率。交织技术也会增加传输速率吗？
8. 在CDN中的DNS的作用是什么？必须修改DNS来支持CDN吗？CDN必须向DNS提供什么样的信息（如果有的话）？
9. 要规划一个网络，需要什么样的信息以取得给定的服务质量？
10. 接收方如何识别在不同会话中的不同RTP流？同一个会话中的不同流是怎样识别的？RTP和RTCP的分组（作为同一个会话的部分）是怎样区别的？
11. 在7.4节描述了3种RTCP分组类型。简要地总结包含在每一种分组类型中的信息。
12. SIP注册器的作用是什么？一个SIP注册器的作用与移动IP中的归属代理的作用有怎样的差异？

7.5～7.6节

13. 在7.5节，我们讨论了非抢占优先级排队。抢占优先级排队是什么？抢占优先级排队对于计算机网络有意义吗？
14. 举一个非保持工作的调度规则的例子。
15. 给出在你的日常生活中FIFO、优先权、RR和WFQ排队的例子。

16. 与Intserv模型和每流资源预留相关的某些困难是什么?

习题

1.在Web上冲浪，寻找两个流式存储音频和/或视频的站点。对于每个站点，使用Ethereal决定：

a. 是否使用了元文件

b. 音频/视频是在UDP还是TCP上发送

c. 是否使用了RTP

d. 是否使用了RTSP

2. 考虑在图7-3中所示的客户机缓冲区。假设流式系统使用第三个选项，即服务器尽可能快地将媒体发送到套接字。假设在大多数时间内，可用的TCP带宽远大于d。再假设客户机缓冲区只能够容纳大约这个媒体的三分之一。描述$x(t)$和客户机缓冲区的内容是如何随着时间变化的。

3. TCP接收缓冲区和媒体播放器的客户机缓冲区是同一个东西吗？如果不是，它们是如何交互的?

4. 在7.3节的因特网电话的例子中，设h是添加到每个块首部的字节总数，包括UDP和IP首部。

a. 假设每20 s发出一个IP数据报，求出该应用一端产生的数据报的传输速率，以每秒比特为单位。

b. 当使用RTP时，h的典型值是什么?

5. 考虑在7.3节中描述预测平均时延d_i的过程。假设u=0.1。令r_1-t_1是最近的采样时延，令r_2-t_2是下一个最近的采样时延，等等。

a. 对于一个特定的音频应用，假设四个分组到达接收方的采样时延为r_4-t_4，r_3-t_3，r_2-t_2和r_1-t_1。根据这4个采样来表示时延d的估计值。

b. 对于n个采样时延归纳出公式。

c. 对于b部分归纳出的公式，令n趋于无穷，给出最后的公式。评论一下为什么这个平均过程被称为一个指数移动平均数。

6. 重复习题5中的a部分和b部分，求出平均时延偏差的估计值。

7. 对于7.3节中的因特网电话例子，我们引入了一个预测时延的在线过程（指数移动平均数）。在本习题中，我们将研究另一种过程。设t_i是接收到的第i个分组的时间戳；设r_i是收到第i个分组的时间。令d_n表示在收到第n个分组后我们对平均时延的预测。在收到第一个分组后，我们设置时延预测等于$d_1 = r_1-t_1$。

a. 假设我们希望对于所有的n，有$d_n = (r_1-t_1+r_2-t_2+\cdots+r_n-t_n)/n$。根据$d_{n-1}$、$r_n$和$t_n$，给出$d_n$的递归公式。

b. 描述一下对于因特网电话，为什么在7.3节中描述的时延预测比a部分叙述的时延预测更合适。

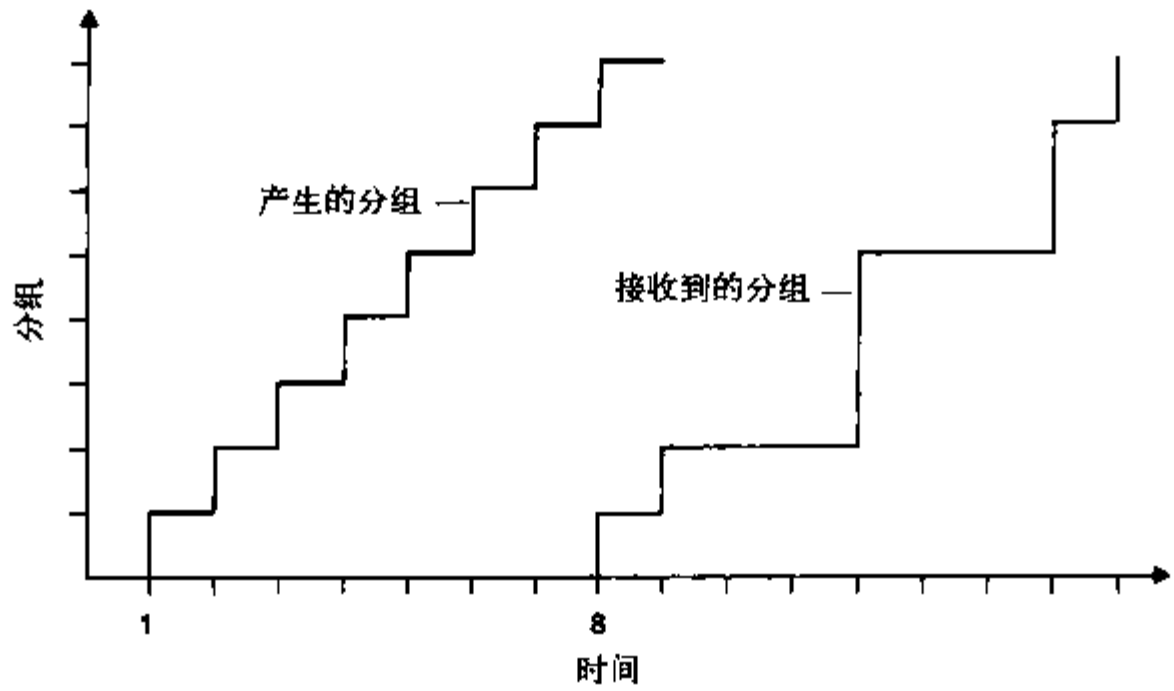
8. 比较7.3节描述的预测平均时延的过程和3.5节预测往返时间的过程。这些过程有什么异同?

9. 考虑7.3节中描述的自适应的播放策略。

a. 当两个连续的分组属于同一个话音突峰期时，在目的地接收的这两个分组如何能够具有时延差异超过20 ms的时间戳?

b. 接收方如何能够使用序号来判断一个分组是否是话音突峰期中的第一个分组？具体说明。

10. 考虑下图（它类似于图7-5)。某发送方在t=1时开始周期性地发送分组化的音频。在t=8，第一个分组到达接收方。



a. 分组2到分组8的时延（从发送方到接收方，忽略任何播放时延）是什么？注意到在该图上每个垂直和水平线段具有1、2或3个时间单元的长度。

b. 如果一旦第一个分组在t=8时到达接收方，音频开始播放，发送的前8个分组中的哪些将不能按时到达进行播放？

c. 如果在t=9音频开始播放，发送的前8个分组中的哪些将不能按时到达进行播放？

d. 在接收方，使得所有前8个分组按时到达进行播放的最小播放时延是什么？

11. 再次考虑在习题10中的图，显示分组音频传输和接收时间。

a. 计算分组2到分组8的估计时延，使用7.3.2节中的对于d_i的公式。使用u=0.1这个值。

b. 使用7.3.2节中对于v_i的公式，从对分组2到分组8的估计平均时延，计算其估计的偏差。

12. 在7.3节中讲过描述因特网电话的两个FEC方案。假设第一个方案为每4个初始块产生一个冗余块。假设第二种方案使用传输速率为标称流传输速率的25%的低比特率编码。

a. 每个方案需要多少额外带宽？每个方案增加多少播放时延？

b. 如果在每组的5个分组中第一个分组丢失了，这两种方案如何进行？哪一种方案有更好的音频质量？

c. 如果在每组的2个分组中第一个分组丢失了，这两种方案如何进行？哪一种方案有更好的音频质量？

13. 假定某CDN不增加在网络中的链路容量的量（假设该CDN使用现有的链路在CDN节点之间分发其内容），主机怎样看到CDN改善了性能？给出一个例子。

14. 一台主机请求一个多媒体对象时，CDN提供了比该主机直接从远程初始主机请求该对象更糟糕的性能，这种情况可能吗？试解析之。

15. 在RTCP接收报告中到达时延抖动是如何计算的？（提示：阅读RTP RFC。）

16. a. 假设我们向因特网发送两个IP数据报，每个数据报携带不同的UDP段。第一个数据报的源IP地址为A1，目的IP地址为B，源端口为P1，目的端口为T。第二个数据报的源IP地址为A2，目的IP地址为B，源端口为P2，目的端口为T。假设A1和A2不同，P1和P2不同。假设这两个数据报都到达它们的目的地址，这两个UDP数据报会被同一个套接字接收吗？为什么？

b. 假设Alice，Bob和Claire要使用SIP和RTP来进行音频会议呼叫。Alice与Bob和Claire之间发送和接收RTP分组，只有一个UDP套接字足够吗（SIP报文所需的套接字除外）？如果是，那么Alice的SIP客户机如何区分RTP分组是来自Bob还是来自Claire？

17. 考虑包含4个用户的一个RTP会话，所有的用户都在同一个多播地址发送和接收RTP分组。每个用户以100 kbps发送视频。

a. RTCP将它的流量速率限制为多少？

b. 一个特定的接收方要分配多少RTCP带宽?

c. 一个特定的发送方要分配多少RTCP带宽?

18. a. RTSP和HTTP相似之处是什么? RTSP有方法吗? HTTP能够用于请求一个流吗?

b. RTSP和HTTP不同之处是什么? 例如, HTTP是带内的还是带外的? RTSP要维护有关客户机的状态信息吗? (考虑暂停/继续功能。)

19. 是非判断题:

a. 如果存储视频直接从Web服务器流向媒体播放器, 那么这个应用使用TCP作为底层的传输协议。

b. 当使用RTP时, 发送方有可能在会话的中间改变编码。

c. 所有使用RTP的应用必须使用端口87。

d. 假设一个RTP会话对每个发送方有独立的音频和视频流, 则这些音频和视频流使用同样的SSRC。

e. 在区分服务中, 尽管每跳行为定义了类别之间的性能差别, 它没有要求为了获得这些性能而使用任何特定机制。

f. 假设Alice要和Bob建立一个SIP会话。在她的INVITE报文中包括了这样的行: m=audio 48753 RTP/AVP 3 (AVP 3指示GSM音频)。因此Alice在该报文中指示她要发送GSM音频。

g. 参考前一句说法, Alice在她的INVITE报文中指示了将把音频发送到端口48753。

h. SIP报文在SIP实体之间通常使用一个默认的SIP端口号发送。

i. 为了维护注册, SIP客户机必须周期地发送REGISTER报文。

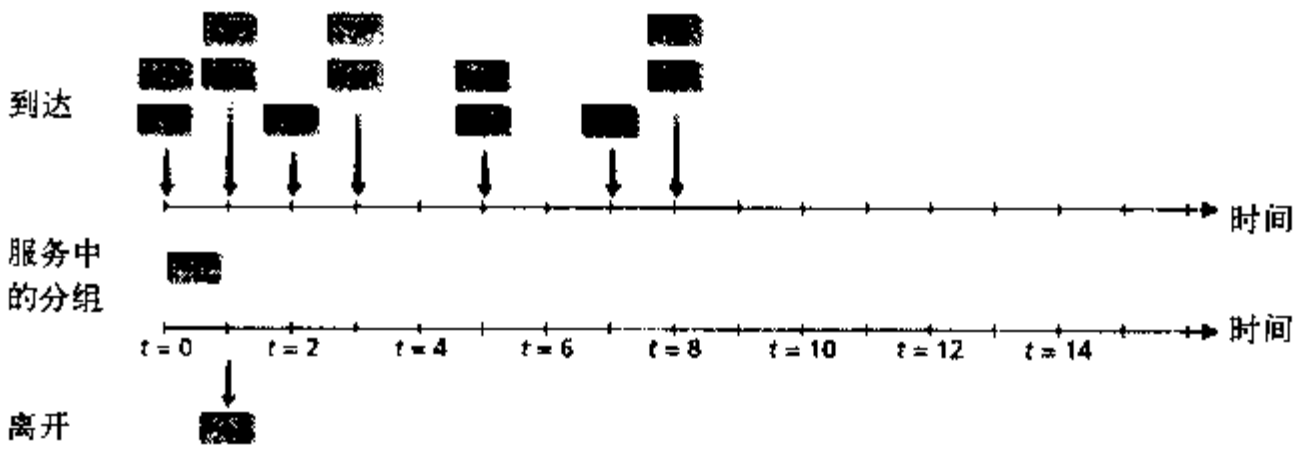
j. SIP强制所有的SIP客户机支持G.711音频编码。

20. 假设对一个支持3种类别的缓冲区运用WFQ调度策略, 并假设这3种类别的权重分别为0.5、0.25和0.25。

a. 假设每类在缓冲区里都有大量的分组。为了获得这些WFQ权重, 这3个类别可能以什么顺序接受服务? (对于循环调度, 一种自然的序列为123123123...)。

b. 假设类1和类2在缓冲区中有大量的分组, 缓冲区中没有类3的分组。为了获得这些WFQ权重, 这3个类别可能以什么顺序接受服务?

21. 考虑下图, 它类似于图7-22至图7-25。回答下列问题:



a. 假设FIFO服务, 指出分组2～12每个离开队列的时间。对每个分组, 它到达和传输开始的时隙之间的时延是什么? 所有12个分组的平均时延是什么?

b. 现在假设优先权服务, 并假设奇数编号的分组是高优先权, 偶数编号的分组是低优先权。指出分组2～12每个离开队列的时间。对每个分组, 它到达和传输开始的时隙之间的时延是什么? 所有12个分组的平均时延是什么?

c. 现在假设循环服务。假设分组1、2、3、6、11和12属于类别1, 分组4、5、7、8、9和10 属于类别2。指出分组2～12每个离开队列的时间。对每个分组, 它到达和传输开始的时隙之间的时延是什

么？所有12个分组的平均时延是什么？

d. 现在假定加权公平排队（WFQ）服务。假设奇数编号的分组假设属于类别1，偶数编号的分组属于类别2。注意到也许不可能取得像课文中描述的那种理想的WFQ调度，故指出在每个时隙为什么你选择了特定分组进入服务。对每个分组，它到达和传输开始的时隙之间的时延是什么？所有12个分组的平均时延是什么？

e. 在所有四类（FCFS、RR、优先权和WFQ）的平均时延中你注意到什么？

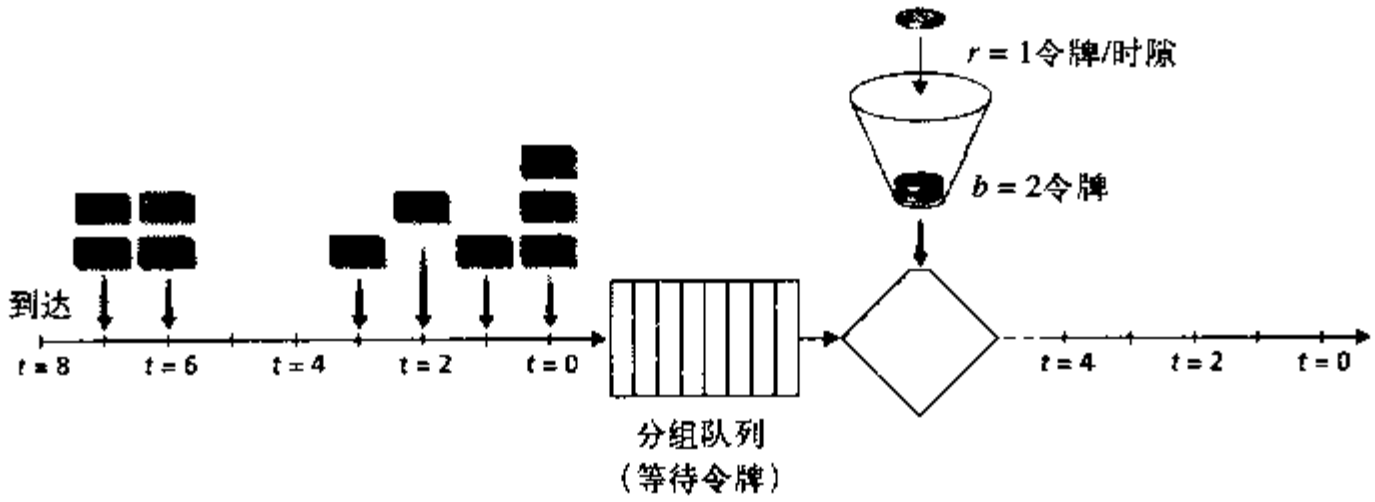
22. 再次考虑习题21中的图。

a. 假设优先权服务，分组1、4、5、6和11为高优先权分组。余下的分组是低优先权。指出分组2～12每个离开队列的时隙。

b. 现在假设使用循环服务，分组1、4、5、6和11属于一类流量，余下的分组属于第二类流量。指出分组2～12每个离开队列的时隙。

c. 现在假设使用WFQ服务，分组1、4、5、6和11属于一类流量，余下的分组属于第二类流量。类1具有的WFQ权重为1，而类2具有的WFQ权重为2（注意到这些权重与前面问题中的不同）。指出分组2～12每个离开队列的时隙。还要注意本问题中有关WFQ服务的提醒。

23. 考虑下图，图中显示了一个漏桶监管器流入分组流。令牌桶能够保持至多两个令牌，并最初在t=0时是满的。新的令牌以每时隙1个令牌的速率到达。输出链路速率采用下列规则：如果两个分组在一个时隙的开始获得令牌，它们能够在相同的时隙中进入输出链路。该系统的定时细节如下：



1. 分组在时隙的开始时到达（如果有的话）。因此在该图中，分组1、2和3在时隙0到达。如果在队列中已经有分组，则到达的分组加入该队列的尾部。分组以FIFO方式朝着该队列的前方行进。

2. 当到达分组增加进队列之后，如果有任何排队分组，那些分组中的一个或两个（取决于可用令牌的数量）每个将从令牌桶中去除一个令牌，并在那个时隙去往输出链路。因此，分组1和2从缓存中去除一个令牌（因为最初有两个令牌）并在时隙0期间去往输出链路。

3. 如果令牌缓存未满则新令牌加入，因为令牌产生速率是r=1令牌/时隙。

4. 然后时间前进到下一个时隙，这些步骤重复。

回答下列问题：

a. 对每个时隙，到达分组处理后（上述步骤1）但在任何分组通过队列传输并去除一个令牌之前的那个时刻，指出位于队列中的分组和位于缓存中的令牌数量。因此，对于在上述例子中的t=0时刻的时隙，分组1、2和3位于队列中，在缓存中有两个令牌。

b. 对每个时隙，在令牌从队列中去除后哪个分组出现在输出链路。所以，对于上述例子中的t=0时隙，在时隙0期间从令牌缓存，分组1和2出现在输出链路上。

24. 重复23题，但假设$r=2$。再次假设开始时桶是满的。

25. 考虑24题并假设现在r=3并且与以前一样b=2。你对上述问题的回答有变化吗?
26. 考虑一下监管分组流的平均速率和突发长度的漏桶监管器（在7.5中讨论的）。我们现在也要监管峰值速率p。说明这个漏桶监管器的输出如何能够流入第二个漏桶监管器中，使得这两个串行的漏桶能够监管平均速率、峰值速率和突发长度。要给出第二个监管器的桶长度和令牌产生速率。
27. 如果对于任何t来说，在每个时间间隔长度t内到达漏桶的分组数小于$rt+b$个，那么就说分组流符合一个突发长度b和平均速率r的漏桶规范（r, b）。一个符合漏桶规范（r, b）的分组流必须在参数r和b的漏桶监管器那里等待吗？证实你的答案。
28. 说明只要$r_1<Rw_1/(\Sigma w_j)$，那么d_{max}实际上是流1中任何分组在WFQ队列中要经受的最大时延。

讨论题

1. 寻找一家使用P2P分发做实况视频流的公司。撰写有关支撑技术的报告。
2. 你认为流式存储音频/视频最好是运行在TCP上还是UDP上?
3. 写一个有关Cisco SIP产品的报告。
4. 只通过提供足够的带宽，也就是给所有的链路容量升级，使得带宽限制不再是一个问题，这样能够解决提供QoS保证的问题吗?
5. 一个有趣的新兴市场正在使用因特网电话和公司的高速LAN代替本公司的PBX（用户交换机）。就这个问题写一份一页的报告。在你的报告中包括下面的问题：
   a. 传统的PBX是什么？谁使用它们?
   b. 考虑公司内的一个用户和公司外连接到传统电话网的另一个用户之间的一个呼叫。在LAN和传统的电话网之间的接口需要什么类型的技术?
   c. 除了因特网电话软件和b部分的接口之外，替换PBX还需要其他什么?
6. 思考高速公路网络的规划方式（例如决定进入和开出某个大城市的高速公路的车道数量）。列出规划该公路时你认为运输工程师应当采取的四个步骤。在规划一个计算机网络时，类似的步骤是什么?

编程作业

在这个实验中，你将实现流式视频服务器和客户机。该客户机使用实时流协议（RTSP）来控制服务器的动作。服务器使用实时协议（RTP）来分组化视频以便在UDP上传输。

将为你提供在客户机和服务器中部分实现了RTSP和RTP的Java代码。你的工作是完成客户机和服务器代码。当你完成时，你已经创建了一个进行下面工作的客户机服务器应用：

- 客户机发送SETUP、PLAY、PAUSE和TEARDOWN等RTSP命令，并且服务器应答这些命令。
- 当服务器处于播放状态，它周期地抓取存储的JPEG帧，用RTP对该帧分组化，并将该RTP分组发送到一个UDP套接字中。
- 客户机接收该RTP分组，去除JPEG帧，解压缩该帧，并在客户机的监管器上再现该帧。

为你提供的代码在服务器实现RTSP协议，在客户机实现对RTP解分组化。这个代码也考虑传输视频的显示。你需要在客户机实现RTSP和RTP服务器。

该编程作业将极大地增强学生对RTP、RTSP和流式视频的理解。我们极力推荐它。该作业也建议进行大量可选的练习，包括在客户机和服务器实现RTSP的DESCRIBE命令。你能够在Web站点http://www.awl.com/kurose-ross找到该作业的详细内容和Java代码的重要片段。

人物专访

Henning Schulzrinne是哥伦比亚大学的教授，计算机科学系的主任，因特网实时实验室的负责人。他是RTP、RTSP、SIP和GIST这些因特网音频和视频通信的关键协议的作者之一。Henning在德国TU Darmstadt获得了电子和工业工程学士学位，在辛辛那提大学获得了电子和计算机工程硕士学位，以及在马萨诸塞大学阿默斯特学院获得了电子工程博士学位。

Henning Schulzrinne

• 是什么使得您决定致力于多媒体网络？

这几乎是巧合。作为一名博士生，我从事DAPTnet方面的工作，DAPTnet是一个用T1线路跨越美国的实验网络，它用于为多播和因特网实时工具提供场所。这促使我写了我的第一个音频工具NeVoT。通过一些DAPTnet的参与者，我开始参与IETF和那时新成立的音频视频传输工作组的工作。这个组后来完成了RTP的标准化。

• 您在计算机行业中的第一份工作是什么？它给您带来什么教益？

我在计算机行业的第一份“工作”是我在加利福尼亚的Livermore读高中时焊接一个牵牛星（Altair）计算机工具包。回到德国，我开了一个小咨询公司来给旅游社设计地址管理程序，为我们的TRS-80将数据存储在磁带上，并通过一个自己制作的硬件接口把IBM的电动打字机作为打印机使用。

我第一份真正的工作是在AT&T的贝尔实验室，为在实验室环境下构建实验网络而研发的一个网络仿真器。

• 因特网实时实验室的目的是什么？

我们的目的是为因特网作为单一的未来通信平台基础设施提供组件和构件块。这包括开发协议，如GIST（用于网络层信令）和LoST（用于由特定区域寻找资源），或我们以前从事的加强协议，如SIP，继续在丰富呈现、对等系统、下一代紧急情况呼叫和服务产生工具方面工作。最近，随着802.11b和802.11n网络以及WiMAX网络很可能马上成为重要的用于电话的技术，我们也广泛地研究用于VoIP的无线系统。

我们试图做实践性相关的工作，通过构建原型和开放源码系统，通过测量实际系统的性能，并对IETF标准作出贡献。

• 您认为多媒体网络将来的发展方向是怎样的？

我们现在正处于过渡阶段，距离IP成为多媒体服务的通用平台只有几年之遥了，从IPTV到VoIP。我们期望收音机、电话和电视即使在暴风雨和地震中都能工作，所以当因特网接替了这些专用网络的职责时，用户将期待有同样级别的可靠性。

我们将必须学会为这样一个生态系统设计网络技术，该生态系统包括竞争的电信公司、服务和内容提供商，服务于大量的技术上未受训练的用户并保护他们免受少量但具破坏性的一批恶意和犯罪用户的侵害。改变协议变得日益艰难。协议也变得更为复杂，因为它们需要考虑竞争的商业利益、安全性、隐私以及由防火墙和网络地址转换引起的缺乏网络透明性。

因为多媒体网络正在成为几乎所有消费者娱乐的基础，因此将强调以低成本管理非常大的网络。用户将期待易于使用，如在他们所有的设备上找到相同的内容。

• 为什么SIP的未来很有前途？

随着现在的无线网络继续向3G网络升级，希望单个多媒体信令机制能够跨越所有类型的网络，包括从电缆调制解调器到企业电话网和公用无线网络。连同软件无线电一道，这在将来使下列东西成为可能：单一设备能被用于家庭网络，能被作为无绳蓝牙电话，在经802.11的企业网中和经3G网络的广域网中应用。即使在我们有这样一个通用的无线设备之前，个人移动机制使得隐藏网络之间差别成为可能。一个

标识成为找到一个人的通用方法，而不必记住或者分发几种特定技术（或特定位置）的电话号码。

SIP还将提供语音（比特）传输和语音服务分离。现在打破本地电话垄断在技术上已成为可能，即一个公司提供中间比特传输，其他的公司提供IP“拨号音”和常用的电话服务，例如网关、呼叫转移和主叫者ID。

除了多媒体信令，SIP提供一种因特网中缺少的新服务：事件通知。我们已经有了这种具有HTTP不完善系统和电子邮件功能的近似服务，但是这决不令人非常满意。因为事件是分布式系统的通用抽象，这样可能简化新服务的构建。

• 您对进入到网络领域的学生有什么建议吗?

网络几乎是一个经典的交叉学科。它涵盖了电子工程、计算机科学、运筹学和其他学科。因此，网络研究者必须熟悉除了协议和选路算法以外的主题。

既然网络已经成为日常生活中如此重要的一部分，要在该领域标新立异的学生们应当思考在网络中新的资源限制：人的时间及努力，而不只是带宽或存储。

从事网络研究工作能够给人以极大的满足，因为它使得人们能够相互通信和交换思想，这是人类所必需的。因特网服务提供商已经成为第三大全球性基础设施，接近于运输系统和能源配给。经济发展几乎离不开高性能网络，因此对可预测的将来将有大量机会。

第8章　计算机网络中的安全

我们首先介绍一下Alice和Bob，这俩人要进行通信，并希望该通信过程是“安全”的。由于本书是一本网络教科书，Alice和Bob可以是两台需要安全地交换选路表的路由器；也可以是希望建立一个安全传输连接的客户机和服务器；或者是两个交换安全电子邮件的电子邮件应用程序；在本章后面我们将要考虑这些案例研究。总之，Alice和Bob是安全性领域中的两个众所周知的固定设备，也许因为使用Alice和Bob更为有趣，这与命名为“A”的实体需要安全地与命名为“B”实体通信作用是一样的。需要安全通信的例子通常包括不正当的情人关系、战时通信和商业事务往来；我们宁愿用第一个例子而不用后两个例子，并乐于使用Alice和Bob作为我们的发送方和接收方，并以第一种情况为背景来讨论问题。

我们说过Alice和Bob要进行通信并希望做到“安全”，那么此处的安全实际上意味着什么呢？如我们将看到的那样，安全性（像爱一样）是多姿多彩的东西；也就是说，安全性有许多方面。毫无疑问，Alice和Bob希望他们之间的通信内容对于窃听者（比如，一个嫉妒的配偶）是保密的。他们可能也想要确保当他们需要进行通信时，确实是在和对方通信，还希望如果他们之间的通信被窃听者篡改时，他们能够检测到该通信已被这种篡改破坏。在本章的第一部分，我们将讨论能够加密通信的密码技术，鉴别正在与他通信的对方并确保报文完整性。

在本章的第二部分，我们将研究怎样应用基本的密码学原则生成安全的网络协议。我们再次采用自顶向下方法，从应用层开始，将逐层（上面四层）研究安全协议。我们将研究如何加密电子邮件，如何加密一条TCP连接，如何在网络层提供覆盖式安全性，以及如何使无线LAN安全。

你可能记得在1.6节中我们概览了几种坏家伙能够在机构网络（如大学和公司网络）中造成一些攻击，包括几种不同形式的DoS攻击。在本章的第三部分，我们将考虑运行的安全性，这与保护机构网络免受攻击有关。特别是，我们将仔细观察防火墙和入侵检测系统是怎样加强机构网络的安全性的。

8.1　什么是网络安全

我们还是以要进行“安全”通信的情人Alice和Bob为例，开始我们的网络安全的研究。这实际上意味着什么？显然，Alice希望即使他们在一个不安全的媒体上进行通信，只有Bob能够理解她所发送的报文；入侵者（入侵者名叫Trudy）能够在该媒体上截获从Alice向Bob传输的报文。Bob也需要确保从Alice接收到的报文确实是由Alice所发送的，Alice同样想要确保和她进行通信的人的确就是Bob。Alice和Bob还要确保他们报文的内容在传输过程中没有被篡改。他们也要确信他们首先能够通信，即无人能够拒绝他们接入通信所需的资源。给定这些问题，我们可以指明**安全通信**（secure communication）具有下列所需要的特性。

- 机密性（confidentiality）。仅有发送方和希望的接收方能够理解传输的报文内容。因为窃听者可以截获报文，这必须要求报文在一定程度上进行加密（encrypted）（即进行数据伪装），以便截获者不能解密（decrypted）（即理解）截取到的报文。机密性的这个方面大概就是通常意义上对于术语安全通信的理解。但需要注意的是，这不是对安全通信

的严格定义（下面我们列出了安全通信的其他方面），也不是机密性的严格定义。例如，Alice可能要求她和Bob通信（或者通信的时间和频度）这个事实是保密的。在8.2节中，我们将学习数据加密和解密的密码学技术。

- 报文完整性（message intergrity）。即使发送方和接收方可以互相鉴别对方，他们还需要确保其通信的内容在传输过程中未被改变，恶意篡改或者意外改动。我们在可靠传输和数据链路协议中遇到的检验和技术在扩展后能够用于提供这种报文完整性，我们将在8.3节中研究该主题。
- 端点鉴别（end-point authentication）。发送方和接收方都应该能证实通信过程所涉及的另一方，以确信通信的另一方确实具有他们所声称的身份。人类的面对面通信可以通过视觉轻易地解决这个问题。当通信实体在不能看到对方的媒体上交换报文时，鉴别就不是那么简单了。举例来说，如果你收到一封电子邮件，其中所包含的文本信息称这封电子邮件来自你的一个朋友，你如何相信这封邮件确实是你的那个朋友所发？如果有人打电话给你，说他是你存款银行的职员，向你索取你的账号、秘密的个人身份号码（PIN）和账户结存额以便他进行核对，你会在电话上告诉他这些信息吗？你最好别那样做！在8.4节中我们将分析端点鉴别技术。
- 运行安全性（operational security）。几乎所有的机构（公司、大学等等）今天都有了与公共因特网相连接的网络。这些网络都潜在地能被由公共因特网接入网络的攻击者危及其安全。攻击者能够试图在网络主机中安放蠕虫，获取公司秘密，勘察内部网络配置并发起DoS攻击。我们将在8.9节中看到防火墙和入侵检测系统等运行设备正被用于反制对机构网络的攻击。防火墙位于机构网络和公共网络之间，控制来自网络的分组接入。入侵检测系统执行“深度分组检查”任务，向网络管理员发出有关可疑活动的告警。

明确了我们所指的网络安全的具体含义后，我们接下来考虑入侵者可能要访问的到底是哪些信息，以及入侵者可能采取哪些行动。图8-1描述了一种情况。Alice（发送方）想要发送数据给Bob（接收方）。为了安全地交换数据，即在满足机密性、端点鉴别和报文完整性等要求的情况下，Alice和Bob将交换控制报文和数据报文（以非常类似于TCP发送方和接收方交换控制报文和数据报文的方式进行）。通常将这些报文全部或部分加密。如在1.6节所讨论的那样，入侵者能够潜在地执行下列行为：

- 窃听——监听并记录信道上传输的控制报文和数据报文。
- 修改、插入或删除报文或报文内容。

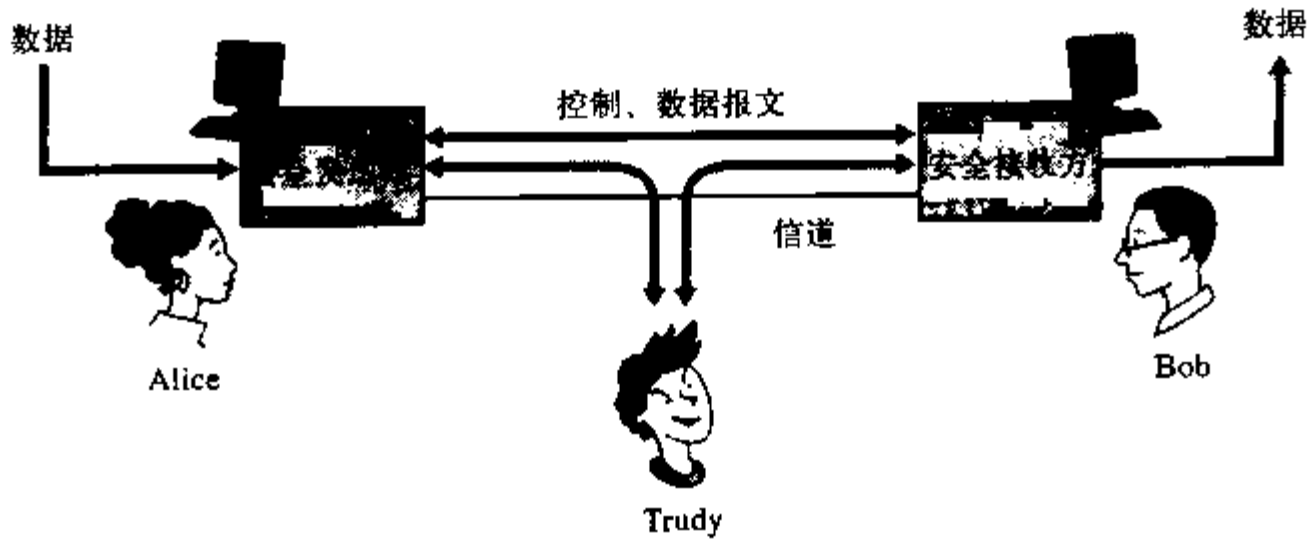


图8-1 发送方、接收方和入侵者（Alice、Bob和Trudy）

如我们所见，除非采取适当的措施，以下这些能力使得入侵者可以用多种方式发动多种安全攻击：窃听通信内容（可能窃取口令和数据），假冒另一个实体，“劫持”一个正在进行的会话，通过使系统资源过载拒绝合法网络用户的服务请求等等。这些和其他安全性威胁也在论文集[Denning 1997]和可读性很强的Rubin的著作[Rubin 2001]中进行了讨论。已报道的攻击一览表见CERT Coordination Center[CERT 2007]。也可参见[Cisco Security 2007; Voydock 1983; Bhimani 1996]。

已经知道在因特网中的确存在真实的威胁，那么Alice和Bob（我们的两个需要安全通信的朋友）在因特网上的对应实体是什么呢？当然，Alice和Bob可以是位于两个端系统的人类用户，例如，真实的Alice和真实的Bob需要交换安全的电子邮件。他们也可以参与电子商务事务。例如，一个真实的Bob希望安全地向一台Web服务器传输他的信用卡号码，以在线购买商品。类似地，真实的Alice要与银行在线交互。但是，正如[RFC 1636]所述，需要安全通信的各方自身也可能是网络基础设施的一部分。前面讲过，域名系统（DNS，参见2.5节）或交换选路信息的选路守护程序（参见4.6节）需要在两方之间安全通信。对于第9章将要讨论的网络管理应用程序也是一样。能主动干扰控制或破坏DNS查找和更新（如在2.5节中讨论的那样）、选路计算[Murphy 2003]或网络管理功能[RFC 2574]的入侵者能够在因特网上造成不可估量的破坏。

建立了上述框架，明确了一些重要定义以及网络安全需求之后，我们将深入学习密码学。密码学的使用在提供机密性方面是不言而喻的，我们很快看到它对于提供端点鉴别、报文完整性也起到了核心作用，这使得密码学成为网络安全的基石。

8.2 密码学的原则

尽管密码学的漫长历史可以追溯到朱利叶斯·凯撒（Julius Caesar）时代，但现代密码技术（包括许多正在今天的因特网中应用的）基于过去30年取得的进展。Kahn的著作《The Codebreakers》（[Kahn 1967]和Singh的著作《The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography》[Singh 1999] 回顾了引人入胜的密码学的悠久历史。对于密码学详细的技术性描述可参见[Kaufman 1995]，这本书特别从网络的观点描述了密码学，趣味性和可读性很强。[Diffie 1998]提供了与密码学有千丝万缕联系的政治和社会问题（例如隐私权）的引人瞩目和最新的分析。对密码学进行全面讨论本身就需要一本完整的书[Kaufman 1995; Schneier 1995]，所以我们只能初步了解密码学的基本方面，特别是因为这些东西正在今天的因特网上发挥作用。一个相当好的在线网站是RSA实验室的FAQ（常见问题解答）网页[RSA FAQ 2007]。我们也要注意到，尽管本节的重点是密码学在机密性方面的应用，但我们将很快看到密码学技术和鉴别、报文完整性和不可否认性等是紧密相关的。

密码技术使得发送方可以伪装数据，使得入侵者不能从截取到的数据中获得任何信息。当然，接收方必须能够从伪装的数据中恢复出初始数据。图8-2说明了一些重要的术语。

假设现在Alice要向Bob发送一个报文。Alice报文的最初形式（例如，“`Bob, I love you.Alice`”）被称为**明文**（plaintext, cleartext）。Alice使用**加密算法**（encryption algorithm）加密其明文报文，生成的加密报文称为**密文**（ciphertext），则该密文对于入侵者是不可懂的。有趣的是在许多现代密码系统中，包括因特网上所使用的那些，加密技术本身是已知的，即公开发行的，标准化的，任何人都可使用的（例如 [RFC 1321; RFC 2437; RFC 2420; NIST 2001]），即使对于潜在的入侵者也是如此！显然，如果任何人都知道数据编码的方法，则一

定有一些秘密信息可以阻止入侵者解密被传输的数据。这些秘密信息就是密钥。

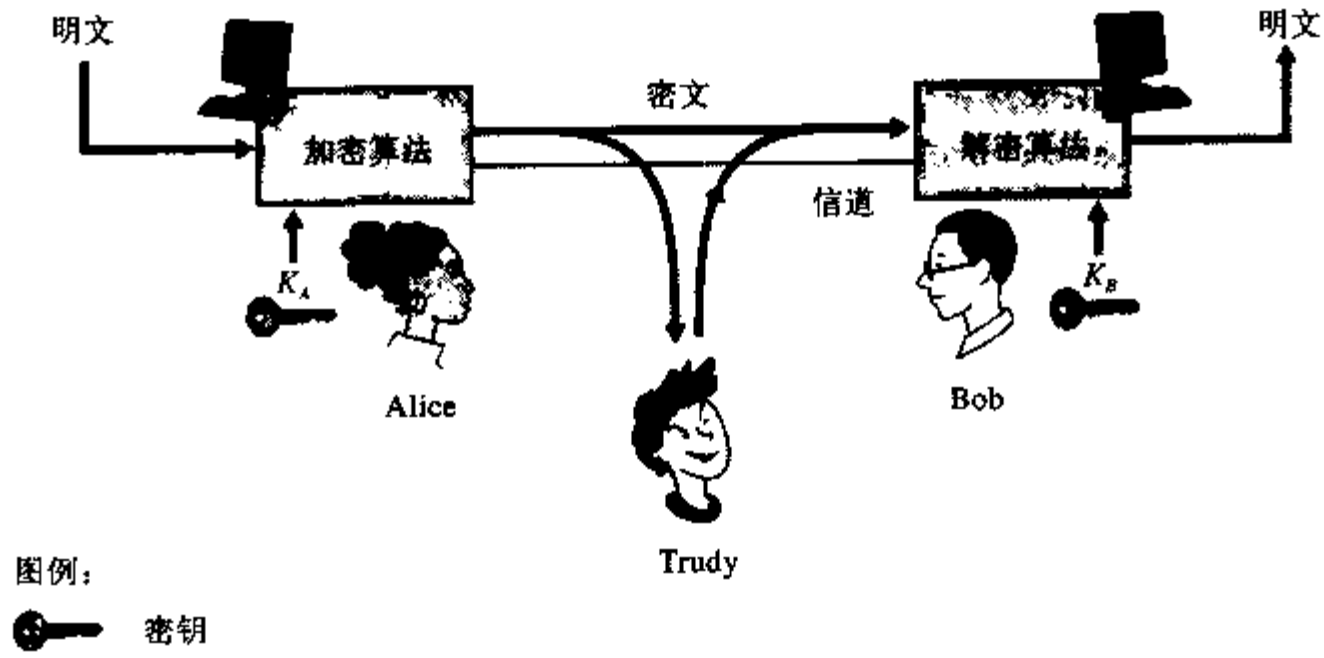


图8-2 密码学组成部分

在图8-2中，Alice提供了一个**密钥**（key）K_A，它是一串数字或字符，作为加密算法的输入。加密算法以密钥和明文报文m为输入，生成的密文作为输出。用符号$K_A(m)$表示（使用密钥K_A加密的）明文报文m的密文形式。使用密钥K_A的实际加密算法显然与上下文有关。类似地，Bob将为**解密算法**（decryption algorithm）提供密钥K_B，将密文和Bob的密钥作为输入，输出初始明文。也就是说，如果Bob接收到一个加密的报文$K_A(m)$，他可通过计算$K_B(K_A(m)) = m$进行解密。在**对称密钥系统**（symmetric key system）中，Alice和Bob的密钥是相同的并且秘密的。在**公钥系统**（public key system）中，使用一对密钥：一个密钥为Bob和Alice俩人所知（实际上，它为全世界所知），另一个密钥只有Bob或Alice知道（而不是双方都知道）。在下面两小节中，我们将更为详细地考虑对称密钥和公钥系统。

历史事件

密码破译竞赛

1977年被美国政府首先采用56比特算法的数据加密标准（DES），仍广泛地在世界范围的金融服务和其他产业中使用，以保护敏感信息。RSA安全公司发起了一系列破解DES密文的竞赛，以强调使用比当前56比特标准更强的密码的必要性。每一次竞赛都是要求在特定时间内破解使用56比特DES加密的一个报文。破解者通过穷举搜索可能的所有密钥完成了这一挑战。RSA公司为获胜者提供了10 000美元的奖金。

1997年，第一次DES挑战赛由来自科罗拉多州的一个团队赢得，他们在不到4个月的时间里找出了密钥。从那时起，不断改进的技术使得更快的穷举搜索成为可能。在1998年2月，Distributed.Net赢得了RSA的DES挑战赛II-1，这次他们只用了41天，到了7月，Electronic Frontier Foundation（EFF）用了56个小时破解了DES报文，赢得了RSA的DES Challenge II-2。

1999年1月，Distributed.Net（一个世界范围的计算机爱好者联盟）使用EFF的“Deep Crack”（一台专门设计的超级计算机）和因特网上全世界范围内的近100 000台PC，仅用了22小时15分钟就破解了密文，赢得了RSA数据安全公司的DES挑战赛III。当该密钥被找到时，EFF的“Deep Crack”和Distributed.Net以每秒245 000 000 000个密钥的速度测试密钥。

8.2.1 对称密钥密码学

所有密码算法都包含用一种东西替换另一种东西的思想，例如，取明文的一部分进行计算，替换适当的密文以生成加密的报文。在分析现代基于密钥的密码系统之前，我们首先学习一下**凯撒密码**（Caesar cipher）找找感觉，该密码是一种加密数据的方法。这种非常古老而简单的对称密钥算法是由Julius Caesar发明的。

凯撒密码用于英语文本时，将明文报文中的每个字母用字母表中该字母k个字母后的那个字母替换（允许绕回，即把字母“a”排在字母“z”之后）。例如，如果$k = 3$，则明文中的字母“a”变成密文中的字母“d”；明文中的字母“b”变成密文中的字母“e”，依此类推。因此，k的值可以看作密钥。举一个例子，明文报文“bob,i love you.alice”在密文中变成“ere,l oryh brx.dolfh”。尽管密文看起来像乱码，但如果你知道是用凯撒密码加密的，因为密钥值只有25个，所以用不了多久就可以破解它。

凯撒密码的一种改进方法是**单码代替密码**（monoalphabetic cipher），它也是使用字母表中的一个字母替换该字母表中的另一个字母。但是，并不按照规则的模式进行替换（例如，明文中的所有字母都用偏移量为k的字母进行替换），只要每个字母都有一个唯一的替换字母，任一字母都可用另一字母替换，反之亦然。图8-3为加密明文的一种可行替换规则。

```
明文字母：  a b c d e f g h i j k l m n o p q r s t u v w x y z
密文字母：  m n b v c x z a s d f g h j k l p o i u y t r e w q
```

图8-3 单码代替密码

明文报文“bob,i love you.alice”变成“nkn,sgktc wky.mgsbc”。因此与用凯撒密码情况一样，这看起来像乱码。单码代替密码的性能看来要比凯撒密码的好得多，可能的字母配对为26！（10^{26}的数量级），而不是25种可能的配对。使用蛮力法要尝试所有的10^{26}个可能的配对工作量，比破解加密算法和解密报文要多得多。但是，通过对明文语言的统计分析，例如，在典型的英语文本中，由于已知字母“e”和字母“t”出现的频率最高（这些字母出现分别为13%和9%），还可知常见的二三个字母的组合通常在一起出现（例如，“in”，“it”，“in”，“the”，“ion”，“ing”等等），这就使得破解该密文变得相对容易。如果入侵者具有某些该报文内容的知识，则破解该密码就会更为容易。例如，如果入侵者Trudy是Bob的妻子，怀疑Bob和Alice有暧昧关系，则她可能猜想“bob”和“alice”这些名字可能会出现在密文中。如果Trudy确信这两个名字出现在密文中，并有了上述报文的密文拷贝，她则能够立即决定这26字母配对中的7个，比蛮力（brute-force）法少检查10^9种可能性。毫无疑问，如果Trudy怀疑Bob有不正当的男女关系，她的确可以期待从该报文中找到某些其他选择的词汇。

当考虑Trudy破解Bob和Alice之间加密方案的难易程度时，可以根据入侵者所拥有的信息分为三种不同的情况。

- *唯密文攻击*。有些情况下，入侵者只能得到截取的密文，也不了解明文报文的内容。我们已经看到，统计分析有助于对加密方案的**唯密文攻击**（ciphertext-only attack）。
- *已知明文攻击*。前面已经看到，如果Trudy以某种方式确信在密文报文中会出现“bob”和“alice”，她就可以确定字母a、l、i、c、e、b和o的（明文，密文）匹配关系。Trudy也可能会幸运地记录到传输的所有密文，然后在一张纸上找到Bob写下的已解密的明文。当入侵者知道（明文，密文）的一些匹配时，我们将其称之为对加密方案的**已知明文攻**

击（known-plaintext attack）。

- 选择明文攻击。在**选择明文攻击**（chosen-plaintext attack）中，入侵者能够选择某一明文报文并得到该明文报文对应的密文形式。对于我们前面所说的简单加密算法来说，如果Trudy能让Alice发送报文“The quick brown fox jumps over the lazy dog”，则Trudy就能够完全破解Alice和Bob所使用的加密方案。但是随后我们将看到，对于更为复杂的加密技术来说，使用选择明文攻击不一定意味着能够攻破该加密机制。

500年前，发明了**多码代替密码**（polyalphabetic encryption），这种技术是对单码代替密码的改进。多码代替密码的基本思想是使用多个单码代替加密，一个单码代替密码用于加密明文报文中一个特定位置的一个字母。因此，在明文报文中不同位置出现的相同字母可能以不同的方式编码。图8-4中显示了多码代替密码机制的一个例子。它使用两个凯撒密码（其中$k = 5$和$k = 19$），如图不同的行所示。加密时可以选择使用这两种凯撒密码方案C_1和C_2，以C_1, C_2, C_2, C_1, C_2的次序循环。即明文的第1个字母用C_1加密，第2和第3个字母用C_2编码，第4个字母使用C_1，第5个字母用C_2，然后循环重复该模式，即第6个字母用C_1加密，第7个字母用C_2加密，依此类推。这样一来，明文报文“bob,i love you”加密后成为“ghu,n etox dhz”。注意到明文报文中的第一个“b”用C_1加密为“g”，第二个“b”用C_2加密为“u”。在这个例子中，加密和解密“密钥”是两个凯撒密码密钥（$k = 5$和$k = 19$）的知识和C_1, C_2, C_2, C_1, C_2的次序模式。

图8-4 使用两个凯撒密码的多码代替密码

1. 块密码

我们现在跳回到现代社会中来，学习今天的对称密钥密码的工作方式。对称加密技术有两种宽泛的类型：**流密码**和**块密码**。当我们研究无线LAN的安全性时，将在8.8节中简要地研究流密码。在本节中，我们关注块密码，该密码用于多种因特网协议的加密，包括PGP（用于安全电子邮件）、SSL（用于使TCP连接更安全）和IPsec（用于使网络层传输更安全）。

在块密码中，将被加密的报文以k比特的块进行处理。例如，如果k=64，则报文被划分为64比特的块，每块被独立加密。为了加密一个块，该密码采用了一对一映射，将明文的k比特块映射为密文的k比特块。我们用一个简单的例子更详细地来探讨块密码。假设k=3，因此块密码将3比特输入（明文）映射为3比特输出（密文）。下表给出了一种可能的映射：

输入	输出	输入	输出
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

注意到这是一种一对一映射，即对每种输入有不同的输出。这种块密码将报文划分成3比特的块并根据上述映射关系进行加密。读者可以验证报文010110001111被加密为101000111001。

继续这个3比特块的例子，注意到上述映射只是许多可能映射中的一种。有多少种可能的映射呢？要回答这个问题，观察发现一个映射只不过是所有可能输入的排列。共有2^3（=8）可能的输入（排列在输入栏中）。这8种输入能够排列为8！=40 320种不同方式。因为这些排列的每一种都定义了一种映射，共有40 320种可能的映射。我们能够将这些映射的每种视为一个密钥，如果Alice和Bob都知道该映射（密钥），他们能够加密和解密在他们之间发送的报文。

对这种密码的蛮力攻击是试图使用所有映射解密密文。仅使用40 320种映射（当k=3），这能够在一台桌面PC上迅速完成。为了挫败蛮力攻击，块密码通常使用大得多的块，由64比特或甚至更多比特组成。注意到对于通常k块密码的可能映射数量是2^k！，对于即使不大的k值（如k=64），这是一个天文数字。

尽管刚才所述全表块密码对于不大的k值能够产生健壮的对称密钥加密方案，然而不幸的是它们难以实现。对于k=64和给定的映射，Alice和Bob将要求维护一张具有2^{64}个输入值的表，这是一个难以实现的任务。况且，如果Alice和Bob要改变密钥，他们将不得不每人重新生成该表。因此，全表块密码在所有输入和输出之间提供了预先决定的映射（如在上述例子中那样），这简直是不可能实现的。

取而代之的是，块密码通常使用模拟随机改变次序表的功能。在图8-5中显示了对k=64的这样的一个例子（摘自[Kaufman 1995]）。该函数首先将64比特块划分为8个块，每个块由8比特组成。每个8比特块由一个8比特到8比特表处理，这是个可管理的长度。例如，第一个块由标志为T_1的表来处理。接下来，这些8个输出块被重新装配成一个64比特的块。该输出被回馈到64比特的输入，开始了第二个循环。经n个这样的循环后，该功能提供了一个64比特的密文块。这种循环的目的是使得每个输入比特影响最后输出比特的大部分（如果不是全部的话）。（如果仅使用一个循环，一个给定的输入比特将仅影响64输出比特中的8比特。）这种块密码算法的密钥将是8张转置表（假定置乱功能是固定的）。

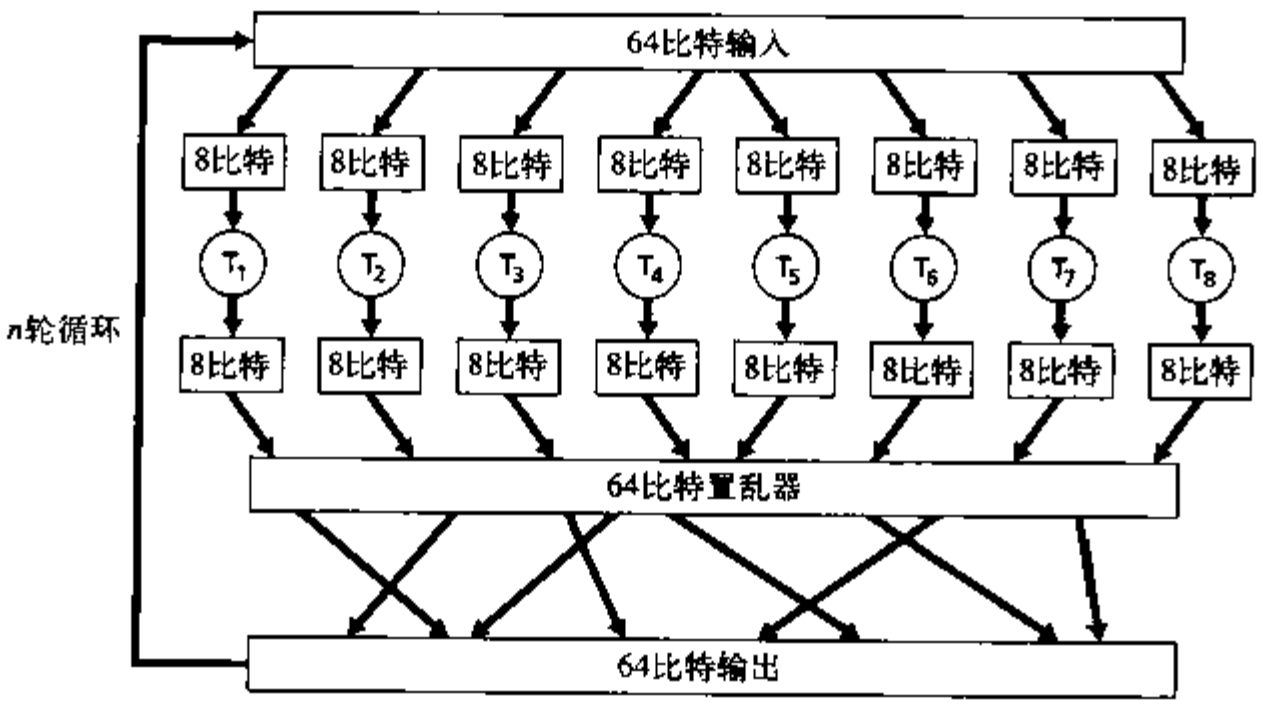


图8-5 一个块密码的例子

目前有一些流行的块密码，包括DES（Data Encryption Standard，数据加密标准），3DES和AES（Advanced Encryption Standard，高级加密标准）。这些标准中的每种都使用了函数，而不是预先决定的表，以及图8-5的线（虽然对每种密码更为复杂和特定）。这些算法中的每种也使用了比特串作为密钥。例如，DES使用了具有56比特密钥的64比特块。AES使用128比特块，能够使用128、192和256比特长的密钥进行操作。一个算法的密钥决定了特定“小型表”

映射和该算法内部的转置。对这些密码中的每种进行蛮力攻击要循环通过所有密钥，用每个密钥应用解密算法。可以发现，对于长度为n的密钥，有2^n个可能的密钥。NIST [NIST 2001]估计，如果用1秒钟破解56比特 DES的计算机（就是说，每秒搜索2^{56}个密钥）来破解一个128比特的AES密钥的话，要用大约149万亿年的时间才有可能。

2. 密码块链接

在计算机网络应用中，我们通常需要加密长报文（或长数据流）。我们应用前面描述的块密码，通过将报文切割成k比特块并独立地加密每块，将出现一个微妙而重要的问题。要理解这个问题，注意到两个或更多个明文块可能是相同的。例如， 两个或更多块中的明文可能是“HTTP/1.1”。对于这些相同的块，块密码当然将产生密文。一个攻击者也许根据Alice和Bob之间使用的网络协议，当它看到相同的密文块时，能够潜在地猜出其明文。攻击者甚至能够通过识别相同的密文块以及通过使用有关支撑协议结构[Kaufman 1995]来解密整个报文。

为了解决这个问题，块密码使用了一种称为**密码块链接**（Cipher Block Chaining，CBC）的技术。为了解释CBC，令$m(i)$表示第i个明文块，$c(i)$表示第i个密文块，$a \oplus b$表示两个比特串的异或（XOR）。同时假设块长度是64比特。将具有密钥S的块密码加密算法表示为K_S。CBC运行过程如下：

1）在加密报文（或数据流）之前，发送方生成一个随机的64比特串，称为**初始向量**(Initialization Vector, IV)。将该初始向量表示为$c(0)$。发送方以明文方式将IV发送给接收方。

2）对第一个块，发送方计算$m(1) \oplus c(0)$，即用IV计算明文第一块的异或值。然后通过块密码算法运行该结果，以得到对应的密文块，即$c(1)=K_S(m(1) \oplus c(0))$。发送方向接收方发送加密块$c(1)$。

3）对于第i个块，发送方根据$c(i)=K_S(m(i) \oplus c(i-1))$生成第$i$个密文块。

我们现在来研究这种方法的某些后果。首先，接收方将仍能够恢复初始报文。毫无疑问，当接收方接收到$c(i)$，它用K_S解密之以获得$s(i) = m(i) \oplus c(i-1)$；因为接收方已经知道$c(i-1)$，它则从$m(i)=s(i) \oplus c(i-1)$获得明文块。第二，即使两个明文块是相同的，相应的密文块将（几乎总是）不同。最后，虽然发送方以明文发送IV，入侵者将仍不能解密密文块，因为该入侵者不知道秘密密钥S。

当设计安全网络协议时，CBC具有重要的后果：需要在从发送方向接收方分发IV的协议中提供一种机制。我们将在本章稍后看到几个协议做这项工作。

8.2.2 公开密钥加密

从凯撒密码时代直到20世纪70年代的两千多年以来，加密通信都需要通信双方共享一个共同秘密，即用于加密和解密的对称密钥。这种方法的一个困难是两方必须就共享密钥达成一致；但是这样做的前提是需要通信（可假定是安全的）！可能是双方首先会面，人为协商确定密钥（例如，凯撒的两个百夫长在罗马浴室碰头），此后才能进行加密通信。但是，在网络世界中，通信各方之间可能从未见过面，也不会在网络以外的任何地方交谈。此时通信双方能够在没有预先商定的共享密钥的条件下进行加密通信吗？在1976年，Diffie和Hellman [Diffie 1976]论证了一个解决这个问题的算法（现在称为Diffie-Hellman密钥交换），这个令人惊奇的优雅地处理安全通信的算法采用完全不同于以往的方式，从而开创了如今的公开密钥密码系统的发展之路。我们很快就会看到公开密钥密码系统也有许多很好的特性，使得它不仅可以用于加密，还可以用于鉴别和数字签名。有趣的是，最近发现20世纪70年代早期由英

国通信电子安全团体的研究人员独立研制的一系列秘密报告的思想，与[Diffie 1976]和[RSA 1978]中的思想类似[Ellis 1987]。事实常常是这样，伟大的想法通常会在许多地方独立闪现；幸运的是，公钥的进展不仅秘密地发生，而且也在公开地发生。

公开密钥密码的使用在概念上相当简单。假设Alice要和Bob通信。如图8-6所示，这时Alice和Bob没有共享一个密钥（如同在对称密钥系统情况下），而Bob（Alice报文的接收方）则有两个密钥，一个是世界上的任何人（包括入侵者Trudy）都可得到的**公钥**（public key），另一个是只有Bob知道的**私钥**（private key）。我们使用符号K_B^+和K_B^-来分别表示Bob的公钥和私钥。为了与Bob通信时，Alice首先取得Bob的公钥，然后用这个公钥和一个众所周知（例如，已标准化）的加密算法，加密她要传递给Bob的报文m；即Alice计算$K_B^+(m)$。Bob接收到Alice的加密报文后，用其私钥和一个众所周知（例如，已标准化的）的解密算法解密Alice的加密报文，即Bob计算$K_B^-(K_B^+(m))$。下面我们可看到，存在着可以选择公钥和私钥的加密/解密算法和技术，使得$K_B^-(K_B^+(m)) = m$；也就是说，用Bob的公钥K_B^+加密报文m（得到$K_B^+(m)$），然后再用Bob的私钥K_B^-解密报文的密文形式（就是计算$K_B^-(K_B^+(m))$），就能得到最初的明文m。这是个不寻常的结果！照此办理，Alice可以使用Bob公开可用的密钥给Bob发送机密信息，而他们任一方都无需分发任何密钥！我们很快能够看到，交换公钥和私钥加密同样能够得到这个不寻常的结果，即$K_B^-(K_B^+(m)) = K_B^+(K_B^-(m)) = m$。

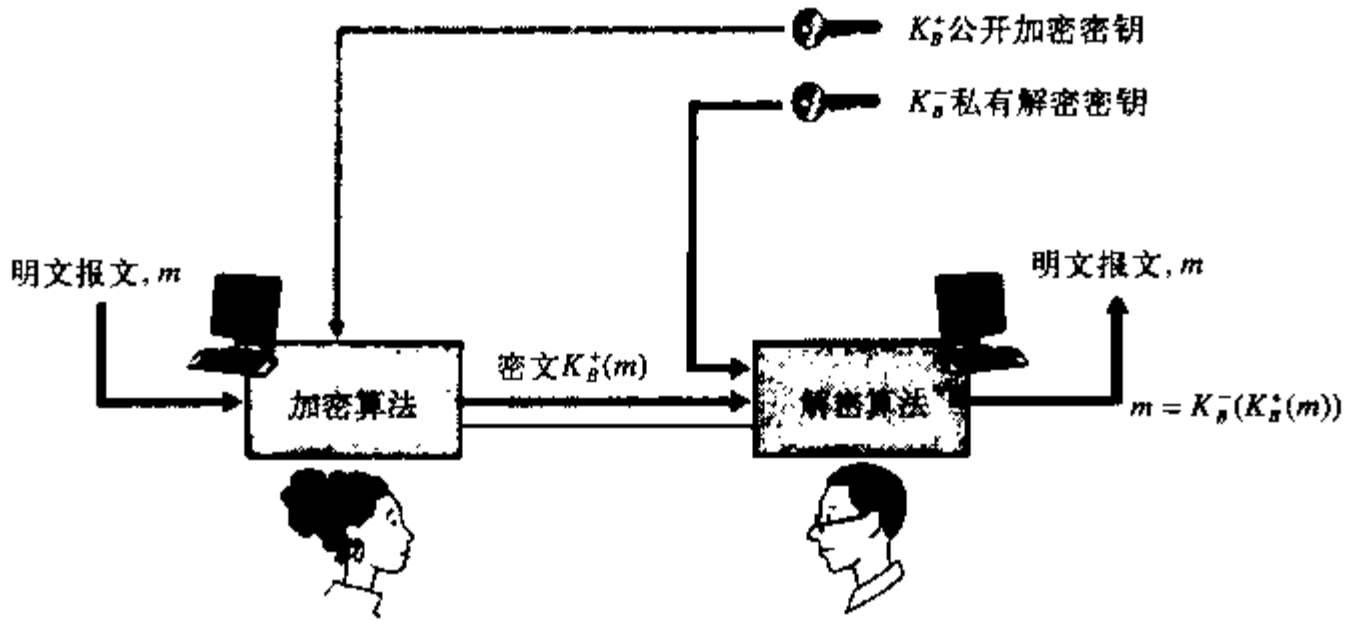


图8-6 公开密钥密码

因此公开密钥密码体制的使用在概念上是简单的。但是有两点必须注意。首先应关注的是，尽管入侵者截取到Alice的加密报文看到的只是乱码，但是入侵者知道该公钥（Bob的公钥是全世界都可以使用的）和Alice加密所用的算法。Trudy可以据此发起选择明文攻击，使用已知的标准加密算法和Bob的公开可用的加密密钥对她所选择的任意报文加密！例如，Trudy可以尝试对她怀疑的可能是Alice发送的报文的全部或部分编码。很明显，如果公开密钥密码要起作用的话，密钥选择和加密/解密算法必须保证任一个入侵者都不能（至少要困难得几乎不可能）确定出Bob的私钥或以某种方式解密或猜出Alice发给Bob的报文。第二个值得关注问题是，由于Bob的加密密钥是公开的，任何人（包括Alice和其他声称自己是Alice的人）都可能向Bob发送一个已加密的报文。在单一共享密钥情况下，发送方知道共享秘密密钥的事实就已经向接收方隐含地证实了发送方的身份。然而在公钥体制中，这点就行不通了，因为任何一个人都可向Bob发送使用Bob的公开可用密钥加密的报文。这就需要在数字签名，我们在8.3节将要讨论，它把发送方和报文绑定起来。

尽管可能有许多算法和密钥能处理这些问题，但RSA算法（RSA algorithm，取创立人Ron Rivest、Adi Shamir和Leonard Adleman的首字母命名）几乎已经成了公开密钥密码的代名词。我们首先来观察RSA的工作原理，然后再考察RSA所起的作用。假设Bob要接收已加密的报文，如图8-6所示。RSA有两个互相关联的部分：

- 公钥和私钥的选择。
- 加密和解密算法。

为了选择公钥和私钥，Bob必须执行如下步骤：

1）选择两个大素数p和q。那么p和q应该多大呢？该值越大，RSA越难于破解，但是执行加密和解密所用的时间也越长。RSA实验室推荐，公司使用时，p和q的乘积为1024比特的数量级，对于“价值不太高的信息”，p和q的乘积应为768比特的数量级[RSA Key 2007]（这导致人们想知道为什么在公司中的使用被认为比在其他场合下的使用重要得多！）。对于选择大素数的方法的讨论参见[Caldwell 2007]。

2）计算$n = pq$和$z = (p-1)(q-1)$。

3）选择小于n的一个数e，且使e和z没有（非1的）公因数（这时称e与z互素）。使用字母e表示这个数是因为这个值将用于加密。

4）找到一个数d，使得$ed-1$可以被z整除（就是说，没有余数）。使用字母d表示这个数是因为这个值将用于解密。换句话说，给定e，然后选择d使得ed被z除的整数余数为1。（当整数x被整数n除时，整数余数表示为$x \bmod n$。）

5）Bob对外界公布的公钥K_B^+就是二元组(n, e)；其私钥K_B^-就是二元组(n, d)。

Alice执行的加密和Bob进行的解密过程如下：

- 假设Alice要给Bob发送一个比特组合或数m，且$m < n$。为了进行编码，Alice先做指数运算m^e，然后计算m^e被n除的整数余数。因此Alice所发送的明文报文m的加密结果c就是：

$$c = m^e \bmod n$$

- 为了对收到的密文报文c解密，Bob计算：

$$m = c^d \bmod n$$

这要求使用他的私钥(n, d)。

举一个RSA的简单例子，假设Bob选择p =5和q = 7。（坦率地讲，这样小的值是不能够保证安全的。）则n = 35，z = 24。因为5和24没有公因数，Bob选择e = 5；最后，因为$5 \times 29 - 1$（即$ed - 1$）可以被24整除，Bob选择d = 29。Bob公开了n = 35和e = 5这两个值，并秘密保存了d = 29。观察公开的这两个值，假定Alice要发送字母“l”、“o”、“v”和“e”给Bob。用1到26之间的每个数表示一个字母，其中1表示“a”，…26表示“z”，Alice和Bob分别执行如表8-1和表8-2所示加密和解密运算。

假定表8-1和表8-2中这样极为简单的示例已经生成了某些极大的数，并且我们前面说过p和q每个都是具有数百比特长的数，这些都是实际使用RSA时必须要牢记的。关于如何选择大素数？如何选择e和d，以及如何对大数进行指数运算？对这些重要问题的详细讨论超出了本书的范围，详情请参见[Kaufman 1995]以及其中的参考文献。

表8-1 Alice的RSA加密，$e = 5, n = 35$

明文字母	m: 数字表示	m^e	密文 $c = m^e \bmod n$
l	12	248 832	17
o	15	759 375	15
v	22	5 153 632	22
e	5	3125	10

表8-2 Bob的RSA解密，$d = 29, n = 35$

密文c	c^d	$m = c^d \bmod n$	明文字母
17	481968572106750915091411825223071697	12	l
15	12783403948858939111232757568359375	15	o
22	851643319086537701956194499721106030592	22	v
10	10000000000000000000000000000000	5	e

1. 会话密钥

需要指出的是，RSA所要求的指数运算过程是相当耗费时间的。形成对比的是，DES用软件实现要比RSA快100倍，用硬件实现则要快1000－10 000倍[RSA Fast 2007]。所以，在实际应用中RSA通常和对称密钥密码结合使用。例如，如果Alice要发送大量已加密数据给Bob，她可以用下面方式做。首先，Alice选择一个用于加密数据本身的密钥；这个密钥有时称为会话密钥（session key）K_S。Alice必须把这个会话密钥告知Bob，因为这是他们用于对称密钥密码（如DES或AES）的共享对称密钥。然后，Alice可以使用Bob的RSA公钥加密该会话密钥，即计算$c = (K_S)^e \bmod n$。Bob收到RSA加密的会话密钥c后，解密得到会话密钥K_S。此时Bob已经知道了Alice将要用做加密传输的会话密钥了。

2. RSA的工作原理

RSA加密/解密看起来相当神奇。为什么那样应用加密算法然后再运行解密算法，就能恢复出初始报文呢？要理解RSA的工作原理，我们需要执行以n为模的算术操作。在模运算中，也可以执行通常的加、乘和指数操作。但是，每一操作的结果都由整除n以后的整数余数替代。我们将取$n = pq$，此处的p和q是RSA算法中的大素数。

前面讲过在RSA加密过程中，加密时首先把一个报文m（用整数表示）做e次的幂运算，然后做模n的算术运算。解密则先把密文值做d次幂，再做模n运算。因此先加密再解密的结果是$(m^e)^d$。下面我们来看，就这个量我们能够得到什么。我们有

$$(m^e)^d \bmod n = m^{ed} \bmod n$$

尽管我们试着揭开RSA工作原理的神秘面纱，但这将要用到数论中一个相当神奇的结论：具体而言，如果是p和q素数，且有$n = pq$，则$x^y \bmod n$与$x^{(y \bmod (p-1)(q-1))} \bmod n$得到的结果相同[Kaufman 1995]。应用这个结论，可得

$$(m^e)^d \bmod n = m^{(ed \bmod (p-1)(q-1))} \bmod n$$

但是要记住，我们是这样选择e和d的，即$ed-1$能被$(p-1)(q-1)$整除（也就是说没有余数），或者等价地说ed被$(p-1)(q-1)$除的整数余数为1，即$ed \bmod (p-1)(q-1) = 1$，由此得到

$$(m^e)^d \bmod n = m^1 \bmod n = m$$

亦即

$$(m^e)^d \bmod n = m$$

这样就得到了我们希望得到的结果！先对m做e次幂（加密）再做d次幂（解密），然后做模n的算术运算，就可得到原来的m。甚至更为奇妙的是，如果颠倒加密和解密的次序，先执行解密过程再执行加密操作，一样能得到原来的m（对这一结论的证明同样可按照上面的推理过程进行）。即

$$(m^e)^d \bmod n = m = (m^d)^e \bmod n$$

很快我们就会认识到RSA的这一特性的重要用途。

RSA的安全性依赖于这样的事实：目前没有已知的算法可以快速进行一个数的因数分解，在此情况下是把公开值n分解成素数p和q。如果已知p和q，并给出公开值e，则就很容易计算出秘密密钥d。在另一方面，也不确定是否存在因数分解一个数的快速算法，所以从这种意义上来说，RSA的安全性也不是确保的。

8.3 报文完整性

在前面一节中我们看到了能够使用密码术为两个通信实体提供机密性。在这节中我们转向提供**报文完整性**（message integrity）这个同等重要的主题。报文完整性也称为报文鉴别。在本节中我们也讨论数字签名。我们将看到报文完整性和数字签名的共有部分是密码散列函数。

我们再次使用Alice和Bob来定义报文完整性问题。假定Bob接收到一个报文（这可能已经加密也可能是明文），并且他认为这个报文是由Alice发送的。为了确认这个报文，Bob需要证实：

1）该报文的确源自Alice；

2）该报文在到达Bob的途中没有被篡改。

我们将在8.5～8.8节看到，报文完整性在所有安全网络协议中都是至关重要的问题。

举一个特定的例子，考虑一个使用链路状态选路算法（例如OSPF）的计算机网络，在该网络中每对路由器之间决定路由（参见第4章）。在一个链路状态算法中，每台路由器需要向该网络中的所有其他路由器广播一条链路状态报文。路由器的链路状态报文包括直接相连邻居的列表以及到这些邻居的直接费用。一旦一台路由器从其他所有路由器收到了链路状态报文，它能够生成该网络的全图，运行它的最小费用选路算法并配置它的转发表。对选路算法的一个相对容易的攻击是，Trudy分发具有不正确状态信息的虚假链路状态报文。因此需要报文完整性：当路由器B收到了来自路由器A的链路状态报文，路由器B应当证实路由器A实际生成了该报文，并且进一步证实在传输过程中该报文没有被篡改。

在本节中我们描述了一种由许多安全网络协议所使用的流行的报文完整性技术。但在做此事之前，我们需要涉及密码学中的另一个重要主题，即密码散列函数。

8.3.1 密码散列函数

如图8-7所示，散列函数以m为输入，经过计算得到一个称为散列的固定长度的字符串。互联网检验和（第3章）、CRC（第4章）满足这个定义。一个**密码散列函数**（crytographic hash function）要求具有下列附加性质：

- 找到任意两个不同的报文x和y使得$H(x) = H(y)$，在计算上是不可能的。

不严格地说，这种性质就意味着入侵者在计算上不可能找到其他报文来替换由散列函数

保护的报文。即如果$(m, H(m))$是发送方所生成的报文和报文散列的话，则入侵者不能伪造另一个报文y，使得该报文与原报文有相同的散列值。

我们来论证一个简单的检验和（如互联网检验和）只能算作劣质的密码散列函数。不像在互联网检验和中执行1的补运算那样，我们把每个字符看作一个字节并把这些字节加到一起，一次使用4字节的块来计算检验和。假定Bob欠Alice 100.99美元并发送一个借条给Alice，这个借条包含以下文本字符串“IOU100.99BOB”，这些字符的ASCII表示（以十六进制形式表示）为49，4F，55，31，30，30，2E，39，39，42，4F，42。

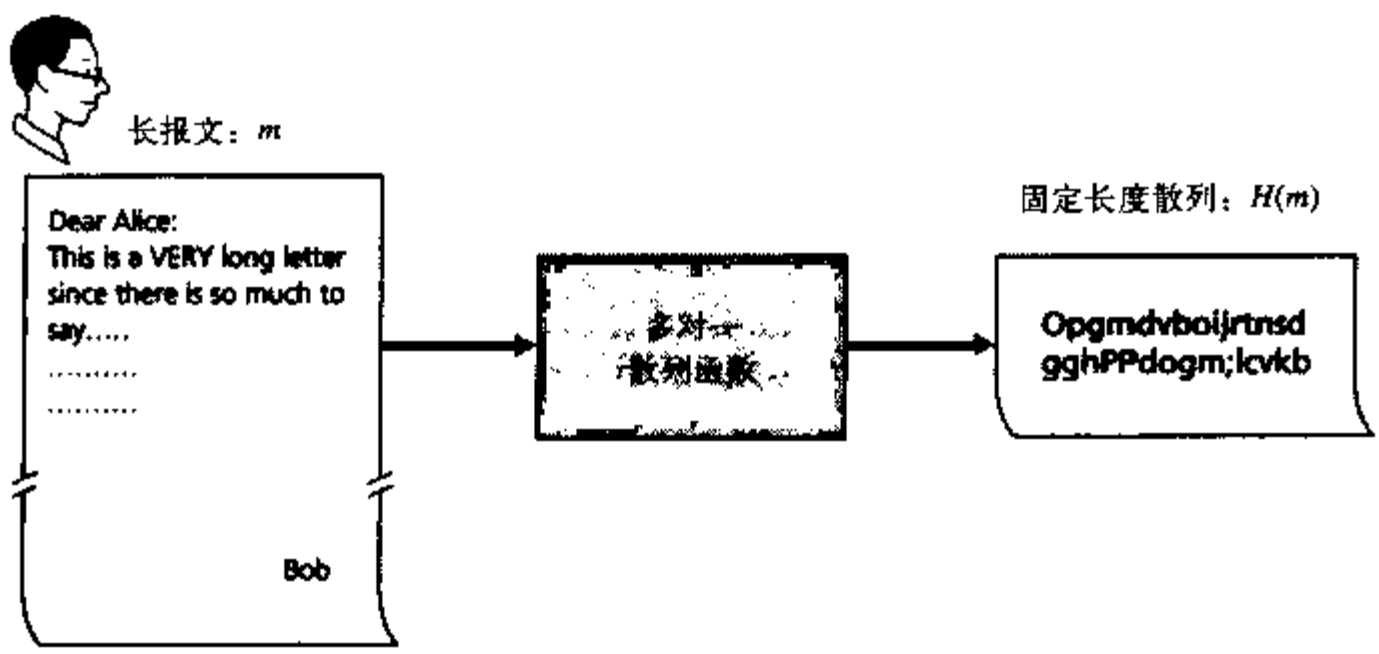


图8-7 散列函数

图8-8上半部分显示了这个报文的4字节检验和是B2 C1 D2 AC。图8-8下半部分显示的是一条稍微不同的报文（但是Bob要支付的钱却多了许多）。报文“IOU100.99BOB”和“IOU900.19BOB”有相同的检验和。因此，这种简单的检验和算法违反了上述的要求。给定初始数据，很容易找到有相同检验和的另一组数据。很明显，为了安全起见，我们需要比检验和更为强劲的散列函数。

报文	ASCII表示			
I O U 1	49	4F	55	31
0 0 . 9	30	30	2E	39
9 B O B	39	42	4F	42
	B2	C1	D2	AC 检验和

报文	ASCII表示			
I O U 9	49	4F	55	39
0 0 . 1	30	30	2E	31
9 B O B	39	42	4F	42
	B2	C1	D2	AC 检验和

图8-8 初始报文和欺诈报文具有相同的检验和

Ron Rivest [RFC 1321]的MD5散列算法如今得到广泛使用。这个算法通过4步过程计算得到128比特的散列。这4步过程由下列步骤组成：①填充，先填1，然后填足够多的0，直到报

文长度满足一定的条件；②添加，在填充前添加一个64比特的报文长度表示；③初始化累加器；④在最后循环步骤中，对报文的16字的块进行4轮处理。MD5的描述（包括一个C源代码实现）可参见[RFC 1321]。

目前正使用的第二个主要散列算法是安全散列算法SHA-1（Security Hash Algorithm）[FIPS 1995]。这个算法的原理类似于MD4[RFC 1320]，而MD4是MD5的前身。SHA-1是美国联邦政府的标准，联邦政府的应用程序如果需要使用密码散列算法的话，都要求使用SHA-1。SHA-1生成一个160比特的报文摘要。较长的输出长度可使SHA-1更安全。

8.3.2 报文鉴别码

我们现在再回到报文完整性的问题。既然我们理解了散列函数，就先来看看将如何执行报文完整性：

1）Alice生成报文m并计算散列$H(m)$（例如用SHA-1）。

2）然后Alice将$H(m)$附加到报文m上，生成一个扩展报文$(m, H(m))$，并将该扩展报文发给Bob。

3）Bob接收到一个扩展报文(m, h)并计算$H(m)$。如果$H(m)=h$，Bob得出结论：一切正常。

这种方法存在明显缺陷。Trudy能够生成虚假报文m'，在其中声称她就是Alice，计算$H(m')$并发送给Bob$(m', H(m'))$。当Bob接收到该报文，一切将在步骤3中核对通过，并且Bob无法猜出这种不道德的行为。

为了执行报文完整性，除了使用密码散列函数，Alice和Bob还需要共享秘密s。这个共享的秘密只不过是一个比特串，称为**鉴别密钥**（authentication key）。使用这个共享秘密，报文完整性执行如下：

1）Alice生成报文m，用s级连m生成$m+s$，并计算散列$H(m+s)$（例如用SHA-1）。$H(m+s)$被称为报文鉴别码（Message Authentication Code, MAC）。

2）然后Alice将MAC附加到报文m上，生成扩展报文$(m, H(m+s))$，并将该扩展报文发送给Bob。

3）Bob接收到一个扩展报文(m, h)，并且（已经收到m并已经知道s）计算MAC $H(m+s)$。如果$H(m+s)=h$，Bob得到结论：一切正常。

图8-9中总结了上述过程。读者应当注意到，这里的MAC（表示报文鉴别码）与数据链路层中所用于的MAC（表示媒体访问控制）是不一样的！

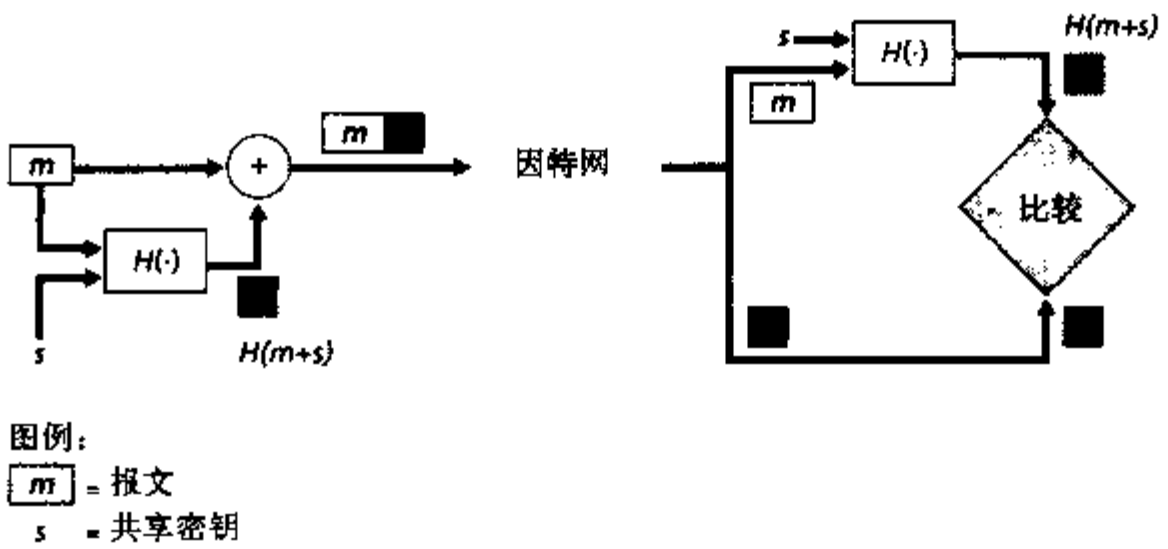


图8-9 报文鉴别码

MAC的一个优良特点是它不需要加密算法。毫无疑问，在许多应用中，包括前面讨论的

链路状态选路算法，通信实体仅关心报文完整性，并不关心报文机密性。使用一个MAC，实体能够鉴别它们相互发送的报文，而不必在完整性过程中综合进复杂的加密过程。

如你所期待的那样，多年来已经提出了若干种对MAC的不同标准。目前最为流行的标准是HMAC，这能够用于MD5或SHA-1。HMAC实际上通过散列函数运行数据和鉴别密钥两次[Kaufman 1995; RFC 2104]。

这里还遗留一个重要问题。我们怎样分发这个共享的鉴别密钥给通信实体呢？例如，在链路状态选路算法中，在某种程度上我们需要向计算机网络中的每个信任的路由器分发该秘密鉴别密钥（注意路由器能够使用相同的鉴别密钥）。网络管理员可以通过物理地访问每台路由器来实际地完成这项工作。或者，如果这名网络管理员不够勤快，且如果每台路由器有它自己的公钥，则网络管理员能够用该路由器的公钥加密鉴别密钥并向任何一台路由器分发，从而通过网络向路由器发送加密的密钥。

8.3.3 数字签名

回想一下在过去的一周中，你在纸上已经签过多少次你的名字？你可能经常会在支票、信用卡收据、法律文件和信件上签名。你的签名证明你（而不是其他人）承认或同意这些文件的内容。在数字领域，人们通常需要指出一个文件的所有者或创作者，或者表明某人认可一个文件的内容。**数字签名**（digital signature）就是在数字领域实现这些目标的一种密码技术。

和人手签字一样，数字签名也应当以可鉴别的、不可伪造的方式进行。这就是说，必须能够证明某个人在一个文档上的签名确实是由该人签署的（签名必须是可证实的），且只有那个人能够签署那个文件（签名不能被伪造）。

我们现在来考虑怎样设计一个数字签名方案。当Bob签署一个报文时，可以观察到Bob必须将某些对他来说是独特的东西放置在该报文上。Bob能够附加一个MAC用作签名，其中MAC通过散列函数运行报文而生成，以及一个对他来说是独一无二的秘密。但是对于Alice来说，为了验证该签名，她必须也具有该密钥的拷贝，在这种情况下该密钥对Bob将不是唯一的了。因此，MAC在这里是无法胜任这项工作的。

回想到使用公钥密码，Bob具有公钥和私钥，这两种密钥对Bob均为独特的。因此，公钥密钥是一种用于提供数字签名的优秀候选者。我们现在来研究一下这是怎样完成的。

假设Bob要以数字方式签署一个文档m。我们可以认为这个文档是Bob打算签署并发送的一个文件或一个报文。如图8-10所示，要签署这个文档，Bob直接使用他的私钥K_B^-计算$K_B^-(m)$。乍一看，会感觉很奇怪，Bob怎么会用他的私钥（在8.2节中，我们用私钥解密用其公钥加密的报文）签署文档！但是回想加密和解密都只是数学运算而已（RSA中所做的e或d幂运算；参见8.2节），而此处Bob的目的不是弄乱或掩盖文档的内容，而只是要以可鉴别的、不可伪造的方式签署这个文档。Bob对文档m签名之后所得的文档就是$K_B^-(m)$。

数字签名$K_B^-(m)$是否满足可鉴别的、不可伪造的？假设Alice有m和$K_B^-(m)$。她要在法庭上证明（进行诉讼）Bob确实签署过这个文档，他就是唯一能够签署该文档的人。Alice持有Bob的公钥K_B^+，并把它用于Bob的数字签名$K_B^-(m)$上，从而得到了文档m。也就是说，Alice计算$K_B^+(K_B^-(m))$，瞧！Alice在经历了令人注目的慌乱后得到了m，它完全与初始文档一致。然后，Alice就可以论证仅有Bob能够签署过这个文档，基于如下理由：

- 任何人签署这个报文都必定使用了K_B^-这个私钥，计算得到了签名$K_B^-(m)$，使得$K_B^+(K_B^-(m)) = m$。

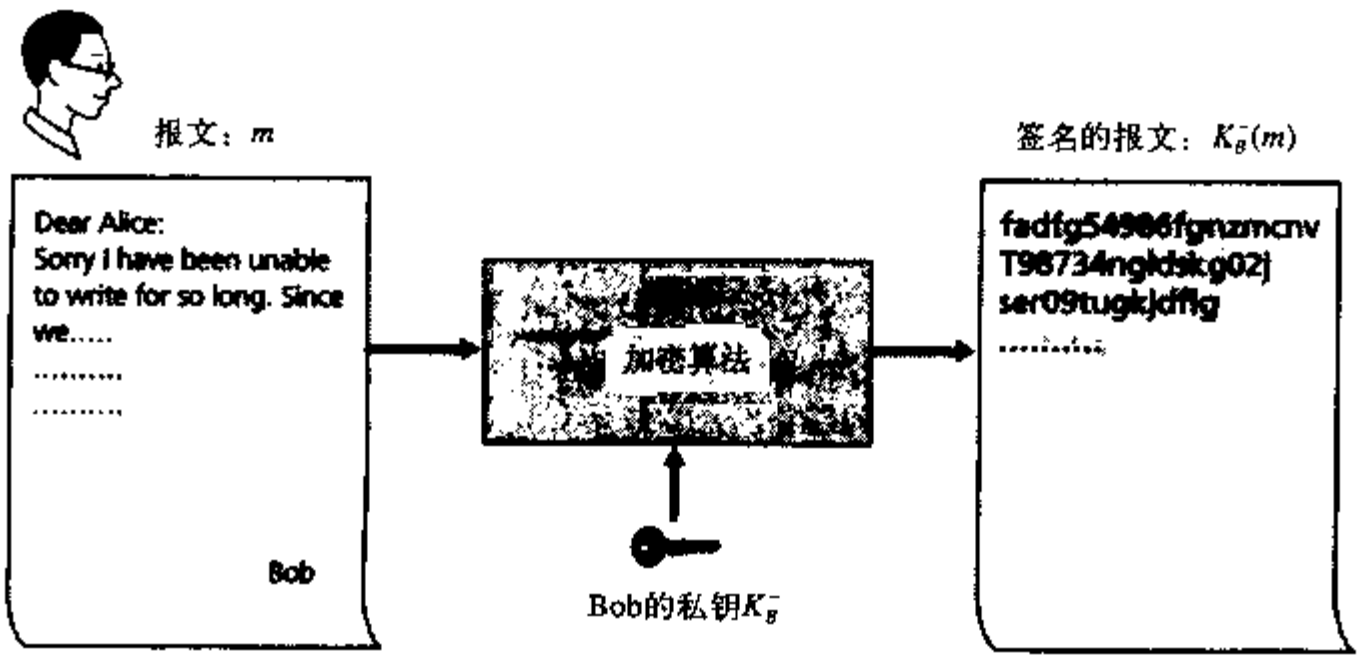


图8-10 对文档生成数据签名

- 知道K_B^-这个私钥的唯一人只有Bob。从8.2节我们对RSA的讨论中可知，知道公钥K_B^+无助于得知私钥K_B^-的信息。因此，知道私钥K_B^-的人才是生成密钥对（K_B^+，K_B^-）的人，而这个人就是Bob。（注意到此处假设Bob没有把K_B^-泄漏给任何人，也没有人从Bob处窃取到K_B^-。）

注意到下列问题是重要的，如果源文档m被修改过，比如改成了另一个文档m'，则Bob对m生成的签名对m'无效，因为$K_B^+(K_B^-(m))$不等于m'。因此我们看到，数字签名也提供完整性，使得接收方验证该报文未被篡改，同时也验证了该报文的源。

用加密对数据进行签名所担心的是，加密和解密的计算代价昂贵。给定加解密的开销，通过完全加密/解密对数据签名是杀鸡用牛刀。更有效的方法是将散列函数引入数字签名。8.3.2节中讲过，一种散列算法是取一个任意长的报文m，计算生成该报文的一个固定长度的数据“指纹”，表示为$H(m)$。散列对于数据保护的意义在于：如果m改变为m'，则由初始数据计算而得（并随该数据一起传输）的m的散列$H(m)$不会与数据已改变为m'的散列$H(m')$相匹配。

使用散列函数，Bob对报文的散列签名而不是对报文本身进行签名，即Bob计算$K_B^-(H(m))$。因为$H(m)$通常比报文m小得多，生成数字签名所需要的计算量大为降低。

在Bob向Alice发送报文的情况下，图8-11提供了生成一个数字签名的操作过程的概览。Bob通过一个散列函数运行他的初始长报文。然后他用自己的私钥对得到的散列进行数字签名。明文形式的初始报文连同已经数字签名的报文摘要（从此以后可称为数字签名）一道被发送给Alice。图8-12提供了鉴别报文完整性的操作过程的概览。Alice先用发送方的公钥应用于报文获得一个散列结果。然后她再用该散列函数应用于明文报文以得到第二个散列结果。如果这两个散列匹配，则Alice可以确信其完整性报文的发送方及其完整性。

在继续学习之前，我们简要地将数字签名与MAC进行比较，尽管它们有类似之处，但也有重要的微妙差异。数字签名和MAC都以一个报文（或一个文档）开始。为了从该报文中生成一个MAC，我们为该报文附加一个鉴别密钥，然后取得该结果的散列。为了生成一个数字签名，我们首先取得该报文的散列，然后用我们私钥加密该报文（使用公钥密码体制）。因此，数字签名是一种“较重的”技术，因为它需要一个如后面描述的、具有认证中心支撑的公钥基础设施（PKI）。我们将在8.5节中看到，PGP是一种流行的安全电子邮件系统，它使用数字签名验证报文完整性。我们已经看到了OSPF使用MAC验证报文完整性。我们将在8.6节和8.7节中看到MAC也能用于流行的运输层和网络安全协议。

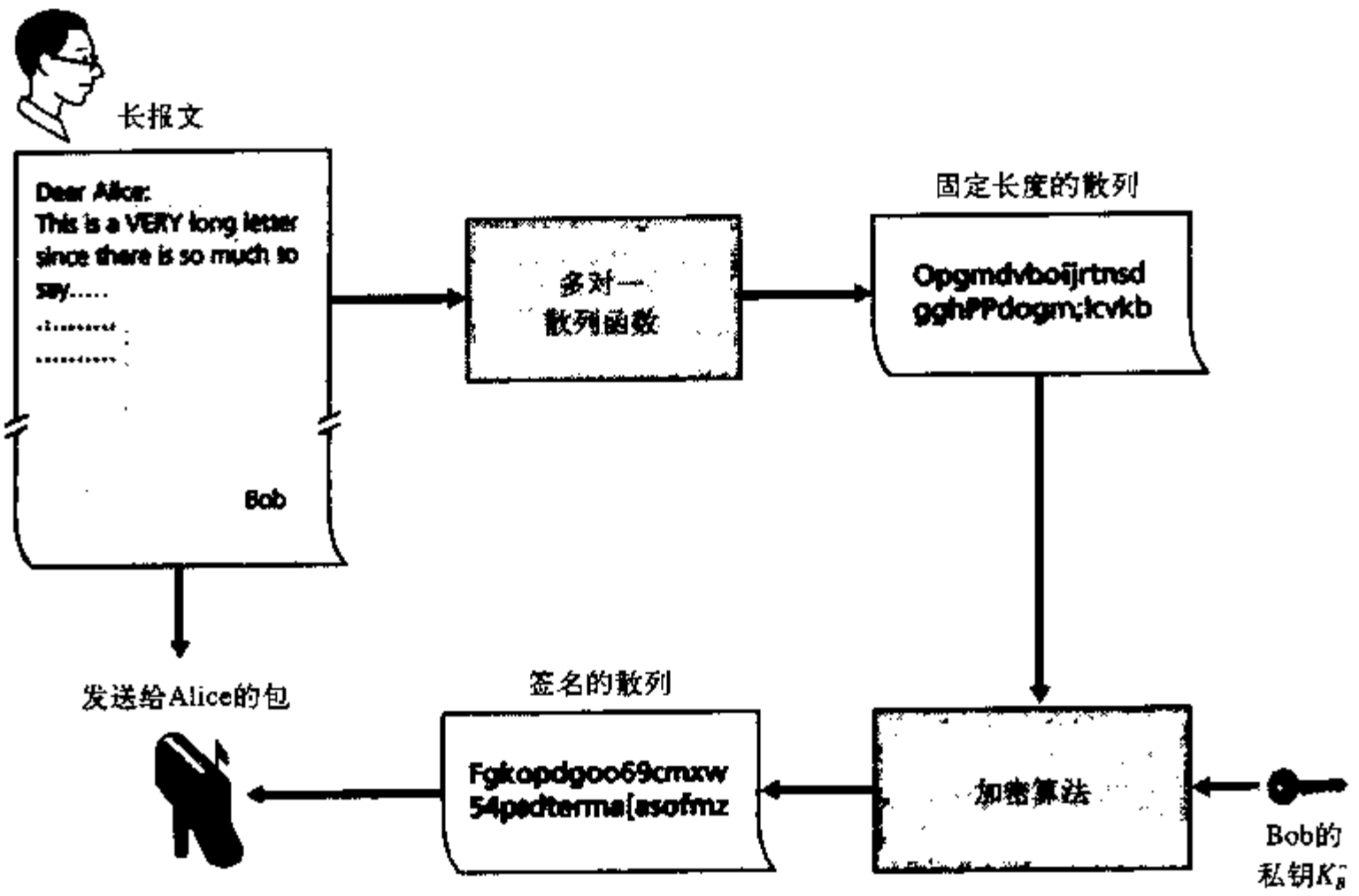


图8-11 发送数字签名的报文

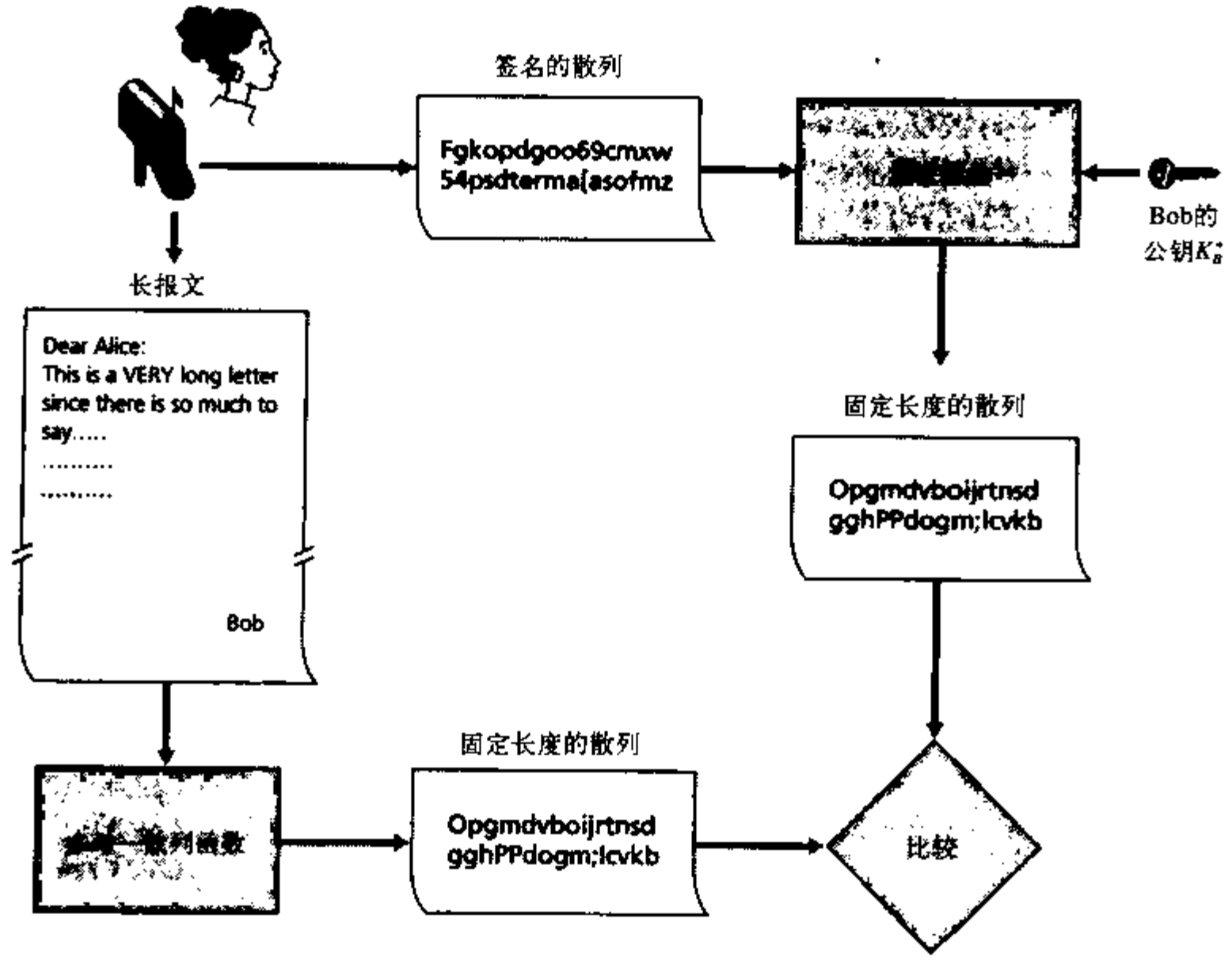


图8-12 验证签名报文

公钥认证

数字签名的一个重要应用是**公钥认证**（public key certification），即证实一个公钥属于某个特定的实体。公钥认证用于许多流行的安全网络协议中，包括IPsec和SSL。

为了深入理解这个问题，我们考虑一个经典的因特网商务版本的“比萨恶作剧”。假定Alice正在从事比萨派送业务，从因特网上接收订单。Bob是一个爱吃比萨的人，他向Alice发送了一份包含其家庭地址和他所要的比萨种类的明文报文。Bob的这个报文中也包含一个数字签名（对原明文报文的签名的散列），以向Alice证实他是该报文的真正来源。为了验证这个数字签名，Alice获得了Bob的公钥（也许从公钥服务器或通过电子邮件报文）并核对该数字签名。通过这种方式，Alice确信是Bob而不是某些恶作剧者下的比萨订单。

在聪明的Trudy出现之前，这一切看起来进行得相当好。如图8-13中所示，Trudy沉溺于一场恶作剧之中。Trudy向Alice发送一个报文，在这个报文中她说她是Bob，给出了Bob家的地址并订购了一份比萨。在这个报文中，她也包括了她（Trudy）的公钥，虽然Alice自然假定它是Bob的公钥。Trudy也附加了一个签名，但是这是用她自己（Trudy）的私钥生成的。在收到该报文后，Alice就会用Trudy的公钥（Alice认为它是Bob的公钥）来解密该数字签名，并得到结论：这个明文报文确实是由Bob生成的。而当外送人员带着具有意大利辣香肠和凤尾鱼的比萨到达Bob家时，他会感到非常惊讶！

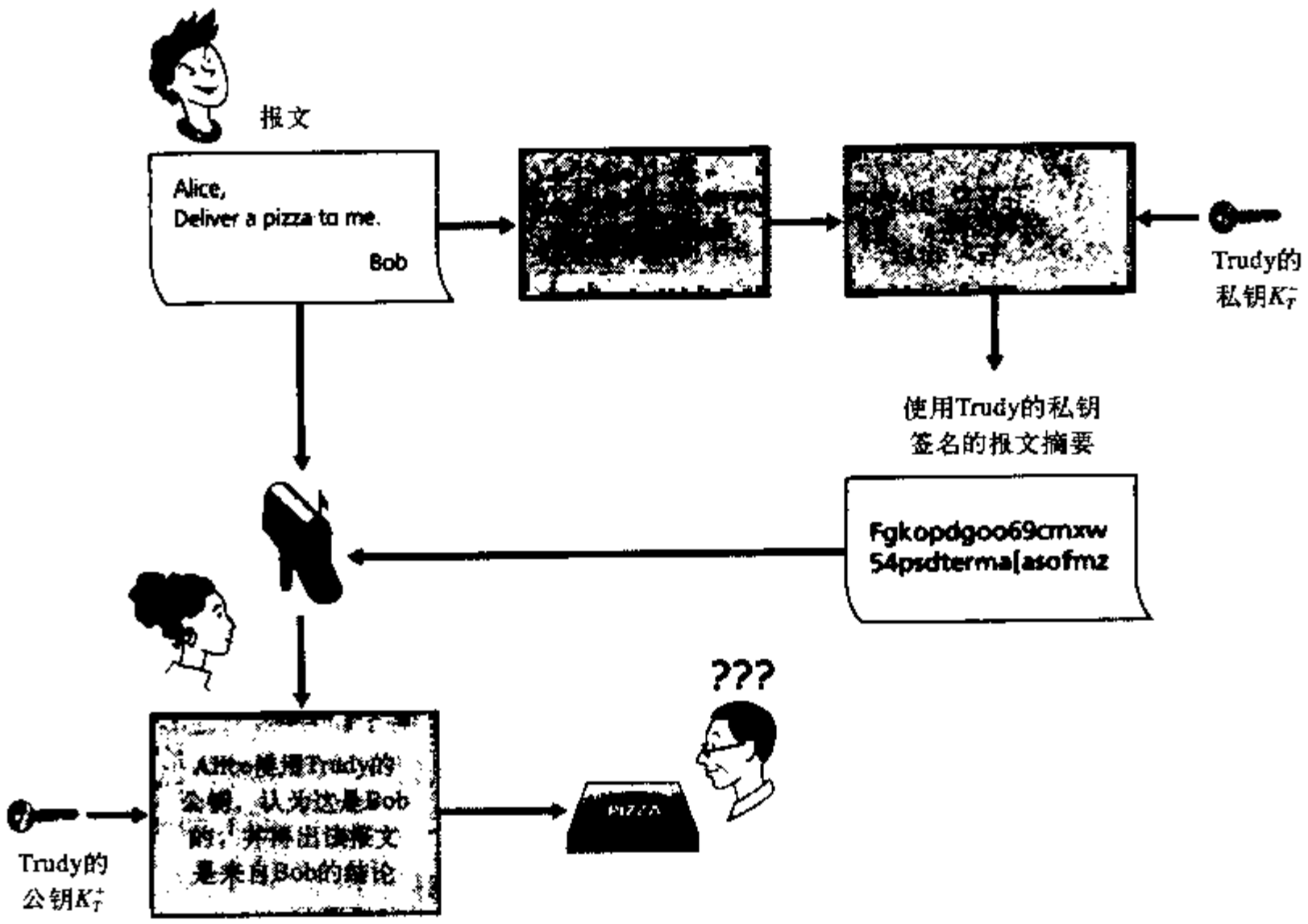


图8-13 Trudy用公钥密码体制冒充Bob

从这个例子我们看到，要使公钥密码有用，需要能够证实你具有的公钥实际上就是与你要进行通信的实体（人员、路由器、浏览器等）的公钥。例如，当Alice与Bob使用公钥密码通信时，她需要证实她假定是Bob的那个公钥确实是Bob的公钥。

将公钥与特定实体绑定通常是由认证中心（Certification Authority, CA）完成的，CA的职责就是验证身份和发行证书。CA具有下列作用：

1）CA证实一个实体（一个人、一台路由器等）的真实身份。如何进行认证并没有强制的过程。当与一个CA打交道时，一方必须信任这个CA能够执行适当的严格身份验证。例如，如果Trudy可以走进名为Fly-by-Night的证书权威机构并只是宣称“我是Alice”，就可以得到该机构颁发的与Alice的身份相关联的证书的话，则人们不会对Fly-by-Night证书权威机构所签发的公钥证书有太多的信任。在另一方面，人们可能愿意（或不愿意）信任某个CA，如果这个CA是联邦/州计划的一部分的话。你对与公钥相联系的身份的信任程度，取决于你信任CA及其身份验证技术的程度。我们编织了多么混乱的Web可信性！

2）一旦CA验证了某个实体的身份，这个CA生成了一个把其身份和实体的公钥绑定起来的证书（certificate）。这个证书包含这个公钥和公钥所有者全局唯一的身份标识信息（例如，一个人的名字或一个IP地址）。由CA对这个证书进行数字签名。这些步骤显示在图8-14中。

我们现在来看如何通过认证来对抗“比萨订购”中的恶作剧者（如Trudy）和其他意外情况。当Bob下了他的订单时，他也发送了他的CA签署的证书。Alice使用CA的公钥来核对Bob证书的合法性并提取Bob的公钥。

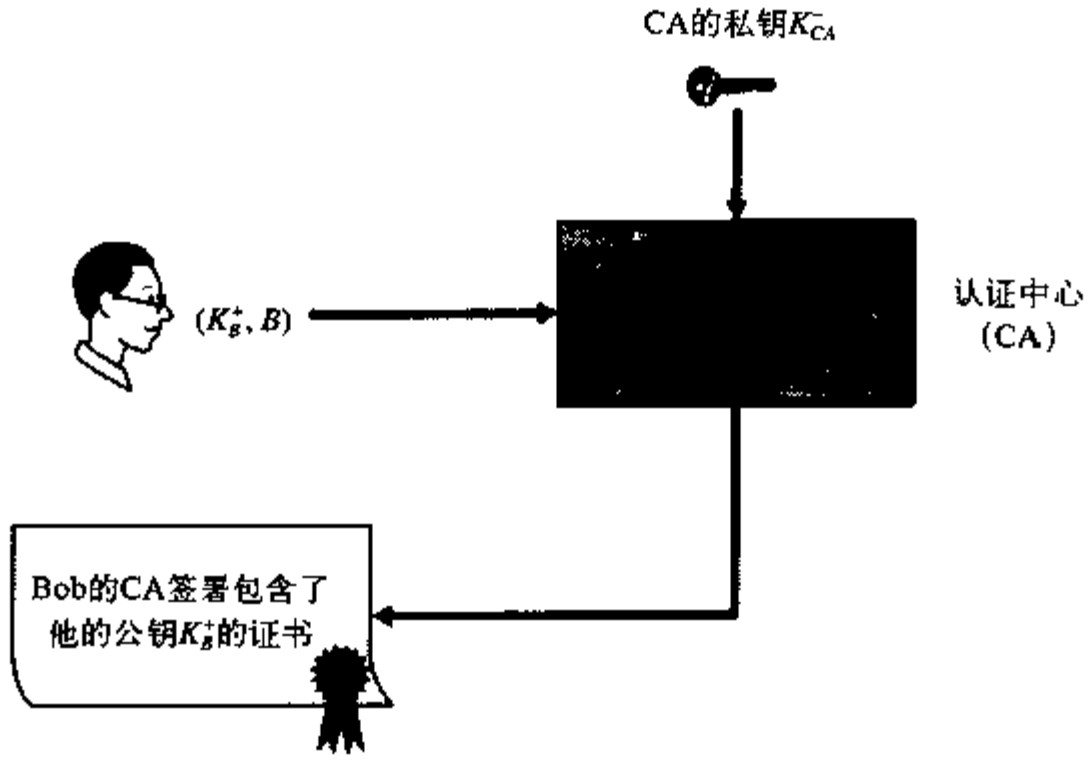


图8-14 Bob获得一份来自CA的证书

国际电信联盟（International Telecommunication Union, ITU）和IETF都研发了用于CA的系列标准。ITU X.509 [ITU 1993]规定了证书的鉴别服务以及用于证书的特定语法。[RFC 1422]描述了安全因特网电子邮件所用的基于CA的密钥管理。它和X.509兼容，但比X.509增加了密钥管理体系结构的创建过程和规则的内容。表8-3显示了一份证书中的某些重要字段。

表8-3 在X.509和RFC 1422公钥证书中的部分字段

字段名	描述
版本（Version）	X.509规范的版本号
序列号（Serial number）	由CA发布的证书的独特标识符
签名（Signature）	规定了CA所用的对该证书签名的算法
颁发者名称（Issuer name）	CA发行该证书的标识符，用的是区别名（DN）格式[RFC 2253]

（续）

字段名	描　述
有效期（Validity period）	证书合法性开始和结束的时间范围
主题名（Subject name）	其公钥与该证书相联系的实体标识符，用DN格式
主题公钥（Subject public key）	该主题的公钥以及该公钥使用的公钥算法（及其参数）的指示

随着电子商务的飞速发展以及对随之而来的安全交易的广泛需求，对认证中心的关注日益增加。提供CA服务的公司包括VeriSign [VeriSign 2007]。

8.4 鉴别

端点鉴别（end-point authentication）就是向其他人证明一个人身份的过程。人类可以通过多种方式互相鉴别：见面时互相识别对方的面容，打电话时分辨对方的声音，海关的检查官员通过护照上的照片对我们进行鉴别。

在本节中，我们讨论经网络通信的双方如何能够鉴别对方。此处我们重点关注当通信实际发生时，鉴别“活动的”实体。鉴别过程与证明过去的某点接收到的报文确实来自声称的发送方稍有不同，后者是8.3节研究的内容。

当经网络进行鉴别时，通信各方不能依靠生物信息比如外表、声波纹等进行身份鉴别。实际上，我们会在后面的实例研究看到，诸如路由器、客户机/服务器进程等网络元素通常必须相互鉴别。此处，鉴别应当在报文和数据交换的基础上，作为某**鉴别协议**（authentication protocol）的一部分独立完成。鉴别协议通常在两个通信实体运行其他协议（例如，可靠数据传输协议、选路表信息交换协议或电子邮件协议）之前运行。鉴别协议首先建立相互满意的各方的标识；仅当鉴别完成之后，各方才继续下面的工作。

和第3章中我们研制可靠数据传输协议（rdt）的情况类似，我们发现研制各种版本的鉴别协议是有启发的。此处我们将称为ap（authentication protocol），然后随着我们学习的深入指出各个版本的漏洞。（如果你喜欢这种逐步式的设计演化，你也许喜欢看 [Bryant 1988]，这本书虚构了开放网络鉴别系统的设计者间的故事，其中还涉及许多奇妙的问题。）

我们假设Alice要向Bob鉴别她自己的身份。

8.4.1 鉴别协议ap1.0

也许我们能够想象出的最简单的鉴别协议就是，Alice直接发送一个报文给Bob，说她就是Alice。这个协议如图8-15所示。这个协议的缺陷是明显的，即Bob无法判断发送报文“我

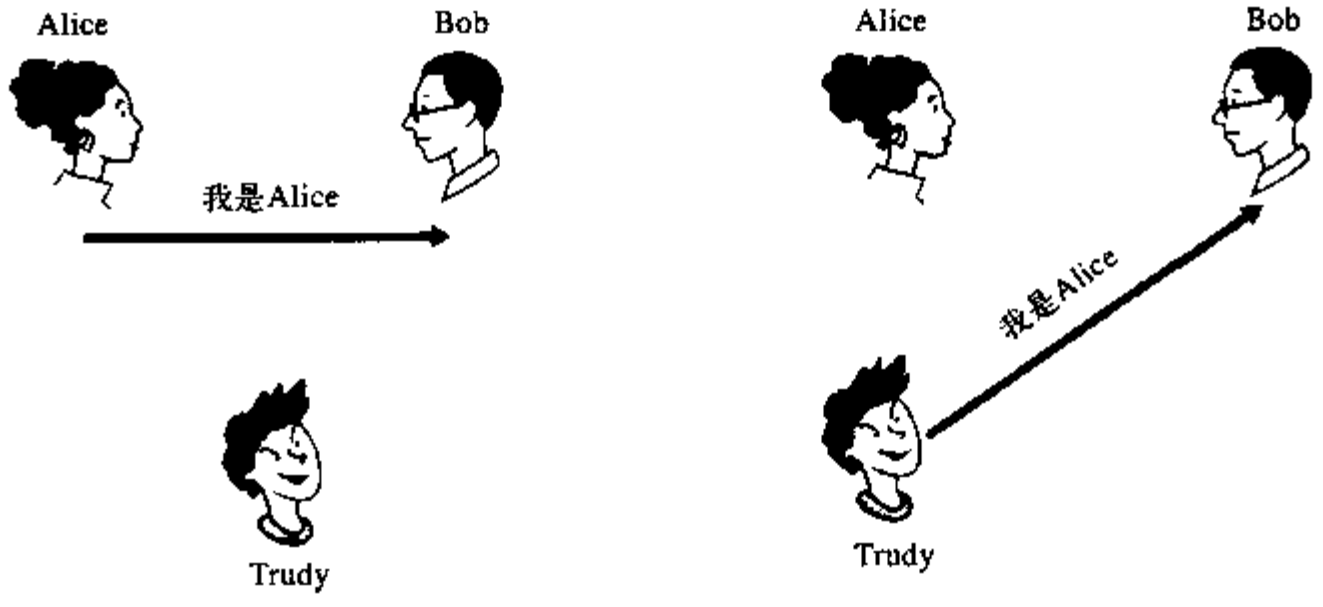


图8-15　协议ap1.0和一种失败的情况

是Alice”的人确实就是Alice。例如，Trudy（入侵者）也可以发送这样的报文。

8.4.2 鉴别协议ap2.0

如果Alice有一个用于通信的周知网络地址（如一个IP地址），则Bob能够试图通过验证携带鉴别报文的IP数据报的源地址是否与Alice的周知IP地址相匹配来进行鉴别。在这种场合，则Alice就可被鉴别了。这可能阻止对网络一无所知的人假冒Alice，但是它却不能阻止决心学习本书的学生或许多其他人！

根据我们学习的网络层和数据链路层的知识，我们就会知道做下列事情并不困难：生成一个IP数据报，并在IP数据报中填入我们希望的任意源地址（比如Alice的周知IP地址），再通过链路层协议把生成的数据报发送到第一跳路由器（例如，如果一个人能够访问操作系统代码并能构建自己的操作系统内核，比如Linux和许多其他免费可用的操作系统）。此后，具有不正确源地址的数据报就会忠实地向Bob转发。这种方法显示在图8-16中，它是IP哄骗的一种形式。如果Trudy的第一跳路由器被设置为只转发包含Trudy的IP源地址的数据报，就可以避免IP哄骗[RFC 2827]。然而，这一措施并未得到广泛采用或强制实施。Bob可能因为假定Trudy的网络管理员（这个管理员可能就是Trudy自己）已经配置Trudy的第一跳路由器，使之只能转发适当地址的数据报而被欺骗。

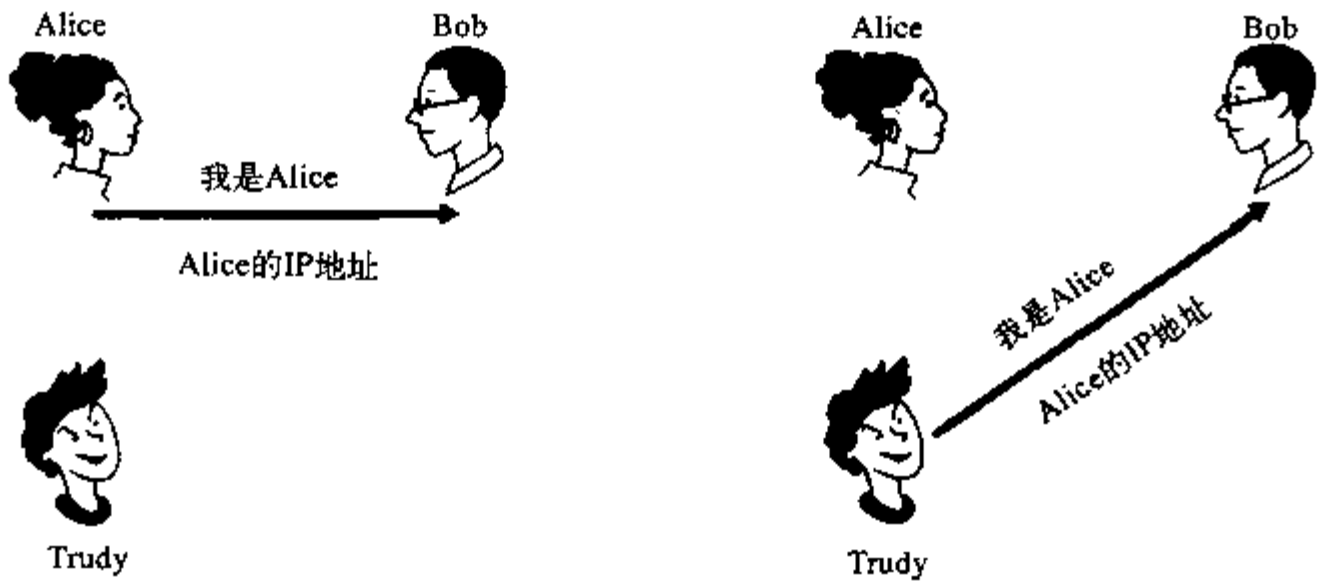


图8-16 协议ap2.0和一种失败的情况

8.4.3 鉴别协议ap3.0

进行鉴别的一种经典方法是使用秘密口令。在使用银行自动柜员机时，我们使用PIN来标识自己的身份，并使用注册口令进入操作系统。口令是鉴别器和被鉴别者之间的一个共享秘密。Telnet和FTP使用口令鉴别。在协议ap3.0中，Alice向Bob发送其秘密口令，如图8-17所示。

由于口令的广泛使用，我们会猜想协议ap3.0相当安全。如果这样想，就错了！这里的安全性缺陷相当明显：如果Trudy窃听了Alice的通信，则可得到Alice的口令。为了使你认识到这种可能性，考虑这样的事实，当你Telnet到另一个机器上并登录时，登录口令未加密就发送到了Telnet服务器。连接到Telnet客户机或服务器LAN的某个人可能嗅探（sniff）（读并存储）在局域网上传输的所有数据分组，并因此窃取到该注册口令。实际上，这是一种窃取口令的周知方法（例如参见[Jimenez 1997]）。这样的威胁显然是真实存在的，所以协议ap3.0明显也不可行。

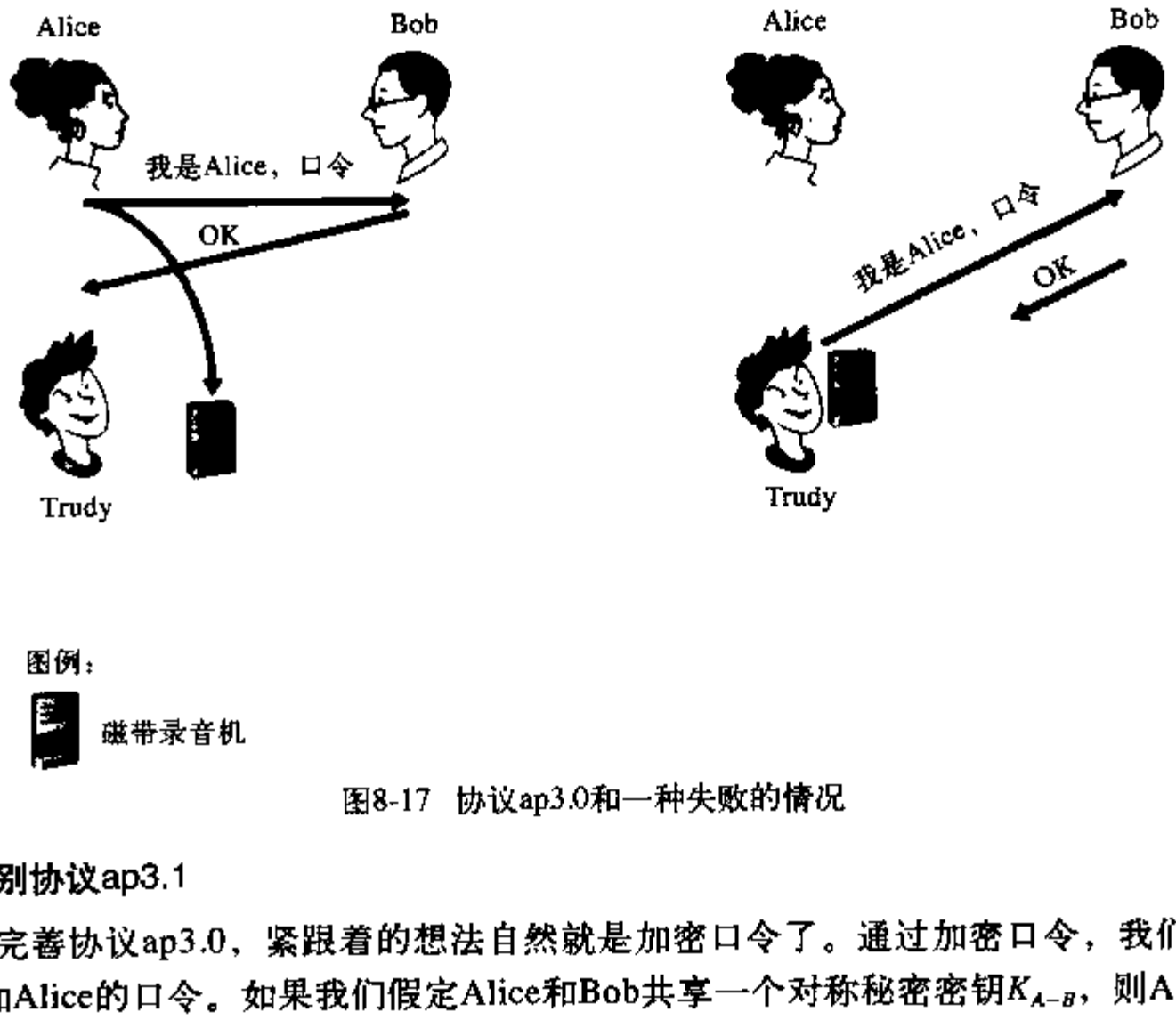


图8-17 协议ap3.0和一种失败的情况

8.4.4 鉴别协议ap3.1

我们完善协议ap3.0，紧跟着的想法自然就是加密口令了。通过加密口令，我们能够防止Trudy得知Alice的口令。如果我们假定Alice和Bob共享一个对称秘密密钥K_{A-B}，则Alice可以加密口令，并发送其标识报文“我是Alice”，从而加密的口令给了Bob。Bob则解密口令，如果口令正确则鉴别了Alice。因为Alice不仅知道口令而且知道用于加密口令的共享秘密密钥值，Bob才可以轻松地鉴别Alice的身份。我们称这个协议为ap3.1。

尽管协议ap3.1确实防止了Trudy得知Alice的口令，但此处使用密码学并不能解决鉴别问题。Bob受制于**回放攻击**（playback attack）：Trudy只需窃听Alice通信，并记录下该口令的加密版本，并向Bob回放该该口令的加密版本，以假装她就是Alice。协议ap3.1中加密口令的使用，并未使它比图8-17中的协议ap3.0的有明显改观。

8.4.5 鉴别协议ap4.0

协议ap3.1的问题在于反复重复使用相同的密码。解决这个问题的一种方式是每次都使用不同的口令。Alice和Bob可以协商一个口令序列（或一个产生口令的算法），并依照此顺序每个口令只使用一次。在S/KEY系统[RFC 1760]中使用了这种思想，采用Lamport[Lamport 1981]提出的一种产生口令序列的方法。

然而，与其仅采用这样的方法，我们不如考虑一种更通用的方法来对抗回放攻击。图8-17中的失败的情况，是因为Bob不能区分Alice的初始鉴别报文和后来被入侵者回放的Alice的初始鉴别报文所致。也就是说，Bob无法判断Alice是否还“活着”，即当前是否还在连接的另一端，或他接收到的报文是否就是前面鉴别Alice时录制的回放。观察力极强的读者会记起TCP的三次握手协议需要处理相同的问题，如果接收的SYN报文段来自较早连接的一个SYN报文段的旧拷贝（重新传输）的话，TCP连接的服务器端不会接受该连接。TCP服务器端如何解决“判断客户机是否真正还活着”的问题呢？它选择一个很长时间内都不会再次使用的初始序号，然后把这个序号发给客户机，然后等待客户机以包含这个序号的ACK报文段进行响应。此处我们可以采用同样的思路来解决鉴别问题。

不重数（nonce）是在一个协议的生存期中只使用一次的一个数。也就是说，一旦某协议使用了一个不重数，就永远不会再使用那个数字了。协议ap4.0以如下方式使用一个不重数：

1）Alice向Bob发送报文“我是Alice”。

2）Bob选择一个不重数 R，然后把这个值发送给Alice。

3）Alice使用她与Bob共享的对称秘密密钥K_{A-B}来加密这个不重数，然后把加密的不重数$K_{A-B}(R)$发回给Bob。和在协议ap3.1中一样，由于Alice知道K_{A-B}并用它加密一个值，就使得Bob知道收到的报文是由Alice产生的。这个不重数用于确定Alice“活着”。

4）Bob解密接收到的报文。如果解密得到的不重数等于他发送给Alice的那个不重数，则可鉴别Alice的身份。

协议ap4.0如图8-18所示。通过使用这个在生存期中只出现一次的值R，然后核对返回的值$K_{A-B}(R)$，Bob就可以确定Alice是她所声称的那个人（因为她知道加密R所需的秘密密钥），也可以确定Alice“活着”（因为她已经加密了Bob刚刚产生的不重数R）。

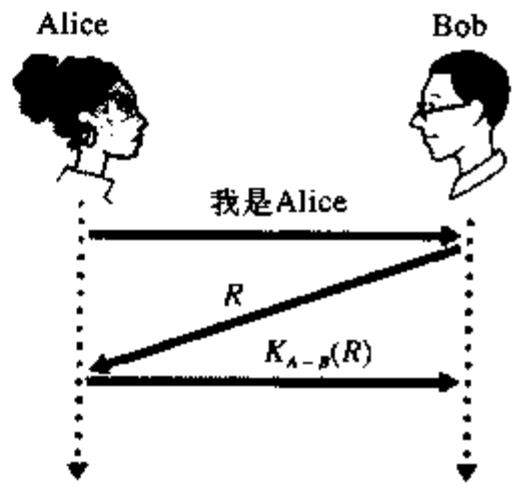


图8-18 协议ap4.0：无失败的情况

8.4.6 鉴别协议ap5.0

对称密钥密码系统和不重数的使用构成了我们成功的鉴别协议ap4.0的基础。一个自然而然的问题就是，我们是否能够可以使用不重数和公钥密码（而不是对称密码）来解决鉴别问题。使用公钥密码可以绕过共享密钥系统的一个难题，即如何使双方一开始知道这个共享的秘密密钥。协议ap5.0以类似于协议ap4.0使用对称密钥密码的方式来使用公钥密码。

1）Alice向Bob发送报文“我是Alice”。

2）Bob选择一个不重数R，然后把这个值发送给Alice。这个不重数再次被用于确定Alice“活着”。

3）Alice使用她的私钥K_A^-加密这个不重数，然后把加密的结果值$K_A^-(R)$发回给Bob。因为只有Alice知道她自己的私钥K_A^-，除了Alice没有别人能产生$K_A^-(R)$。

4）Bob用Alice的公钥K_A^+解密接收到的报文；即Bob计算$K_A^+(K_A^-(R))$。回忆我们在8.2节中对RSA公钥密钥的讨论：$K_A^+(K_A^-(R)) = R$。因此，Bob计算R并鉴别Alice的身份。

协议ap5.0的运行如图8-19所示。协议ap5.0是否和协议ap4.0同样安全呢？它们都使用了不重数。但是协议ap5.0使用公钥技术需要Bob获取Alice的公开密钥。这就出现了图8-20所示的有趣情况，这时Trudy可以对于Bob假冒Alice。

1）Trudy向Bob发送报文“我是Alice”。

2）Bob选择一个不重数R，然后把它发送给Alice，但是这个报文被Trudy中途截取。

3）Trudy使用她的私钥K_T^-加密这个不重数，然后把加密的结果值$K_T^-(R)$发回给Bob。对Bob来说$K_T^-(R)$只是一个比特块，他并不知道这个比特块代表$K_T^-(R)$还是$K_A^-(R)$。

4）此时Bob要得到Alice的公钥K_A^+以解密他刚收到的值。他向Alice发送一个报文索要K_A^+（Bob也可从Alice的网站得到Alice的公开密钥）。Trudy同样可以中途截取这个报文，并把她自己的公钥K_T^+发给Bob。Bob计算$K_T^+(K_T^-(R)) = R$，则Bob就把Trudy鉴别为Alice！

从上面的情况可以看出，协议ap5.0仅仅因为公钥分发而不够安全。幸运的是，我们能够

使用证书来安全地分发公钥，如我们在8.3节中所见的那样。

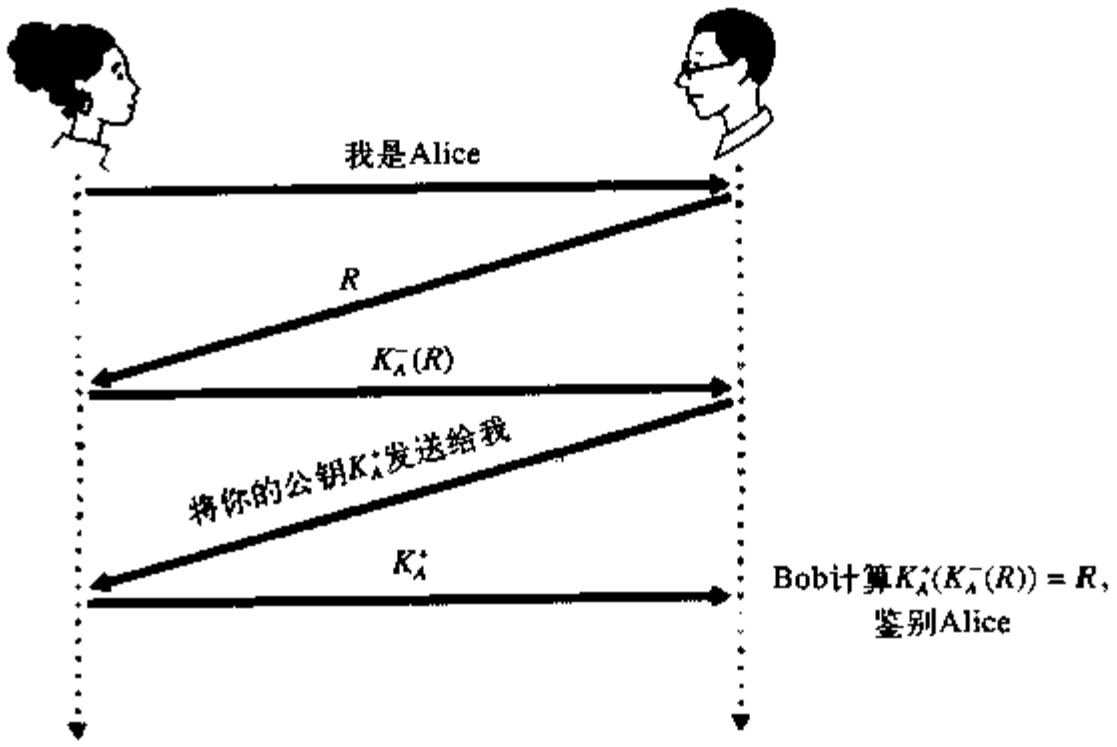


图8-19 协议ap5.0正确工作

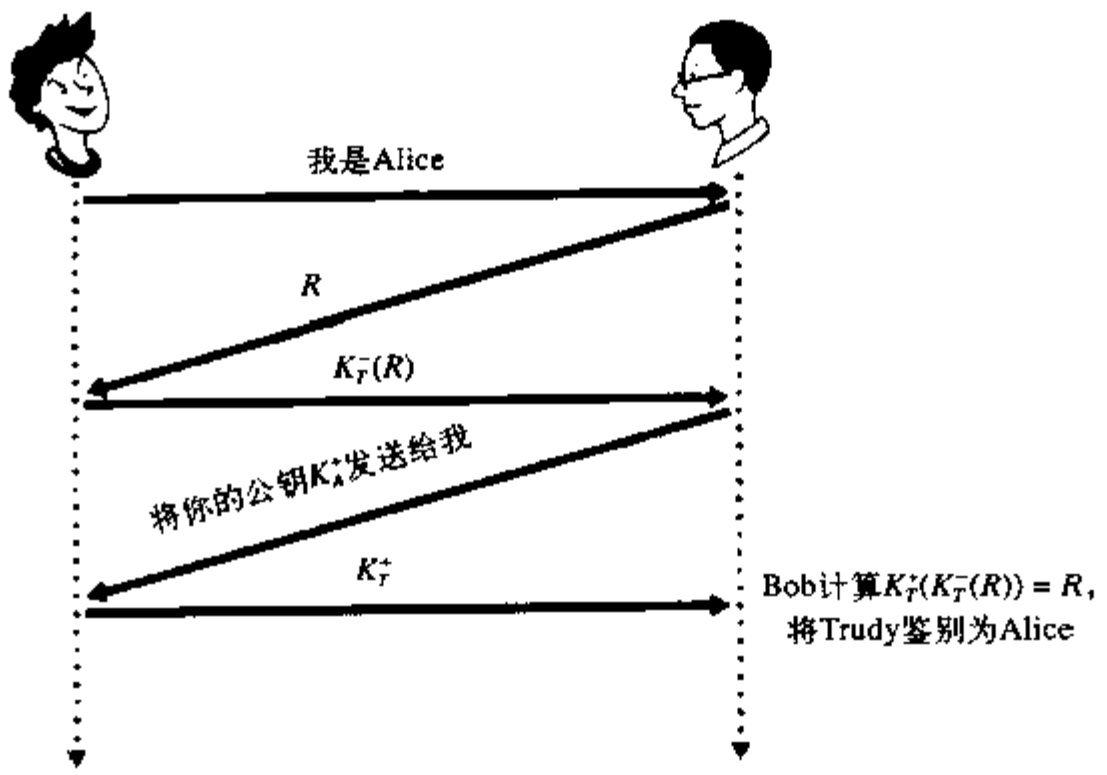


图8-20 在协议ap5.0中的安全漏洞

在图8-20中，Alice和Bob最后可能同时发现有些不对劲，因为Bob将宣称已经与Alice交互过，而Alice知道自己从未与Bob交互过。还有一种可以躲过这种检测的甚至更为隐蔽的攻击。在如图8-21所示的情况下，Alice和Bob相互交谈，但Trudy利用鉴别协议中的相同漏洞，可以透明地插在Alice和Bob之间。特别是，如果Bob开始向Alice发送加密数据（这些数据用Trudy发给他的密钥加密），Trudy可以在Bob与Alice的通信过程中恢复出明文。与此同时，Trudy能够重新使用Alice的真实公钥加密后向Alice转发Bob的数据。

就这样，Bob愉快地发送了加密的数据，Alice也愉快地接收到了使用她自己的公钥加密的数据，双方都不会觉察到Trudy的存在。即使Alice和Bob以后见面，谈起这次通信，因为Alice确实正确地收到Bob发给她的数据，所以根本检测不到任何异常。这是所谓**中间人攻击**（man-in-the-middle attack）的一个例子。有时这也称为**救火水桶接力攻击**（bucket-brigade attack），

因为Trudy在Alice和Bob之间传递数据，就像一队人从远处的水源传递水桶灭火一样。

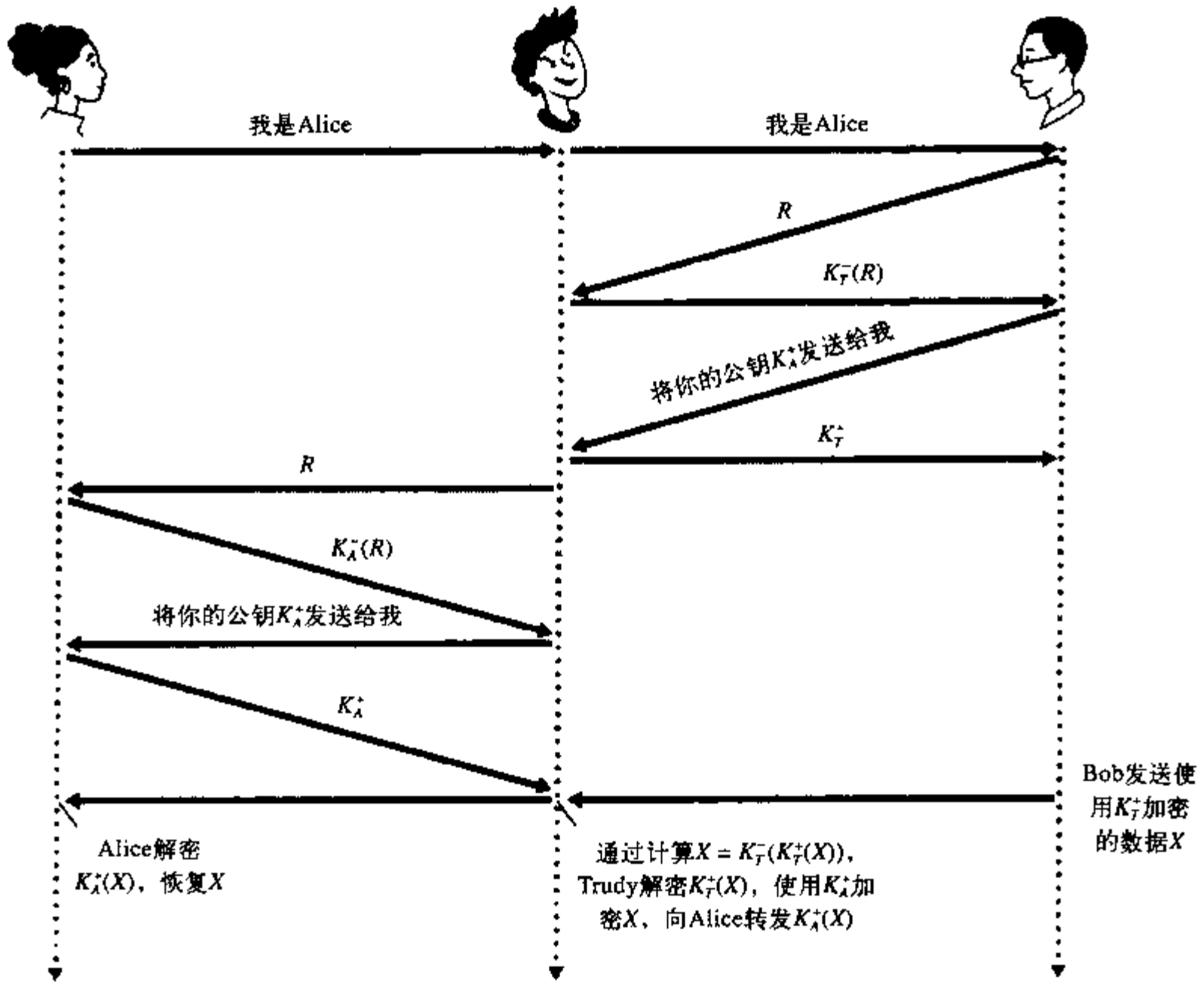


图8-21 “中间人”攻击

8.5 电子邮件安全

在前面各节中我们分析了网络安全中的基本问题，包括对称密钥和公开密钥密码学、端点鉴别、密钥分发、报文完整性和数字签名。我们现在着手研究如何使用这些工具在因特网中提供安全性。有趣的是，可以为因特网协议栈上面4层的任一层提供安全性服务。当为某一特定应用层协议提供安全性时，则使用这一协议的应用程序将能使用一种或多种安全服务，比如机密性、鉴别或完整性。当为某一运输层协议提供安全性时，则所有使用这一协议的应用程序都可以使用到该运输层协议所提供的安全性服务。在基于主机到主机的网络层提供安全性时，则所有运输层报文段（当然也包括所有应用层数据）都可以享用该网络层所提供的安全服务。当基于一条链路提供安全性时，则所有经过这个链路传输的帧中的数据都得到了为该链路提供的安全性服务。

在8.5～8.8节中，我们研究如何在应用层、运输层、网络层和数据链路层中使用这些安全性工具。为了与本书的整体框架保持一致，我们从协议栈的顶层开始，讨论应用层的安全性。此处我们的方法是使用特定的应用程序即电子邮件，作为应用层安全性的一个案例来研究。然后我们沿协议栈向下，分析SSL协议（该协议在运输层提供安全性），IPsec协议（它在网络层提供了安全性），以及IEEE 802.11无线局域网协议的安全性。

你可能会感到奇怪：为什么要在因特网的多个层次上提供安全性功能呢？仅在网络层提供安全性功能并加以实施还不足够吗？对这个问题有两个答案。首先，尽管网络层安全性可以通过加密数据报中的所有数据（即所有的运输层报文段），以及通过鉴别所有数据报的源IP地址，提供“地毯式覆盖”，但是却并不能提供用户级的安全性。例如，一个商业站点不能依赖IP层安全性来鉴别一个在该站点购买商品的顾客。因此，此处除了较低层的地毯式覆盖外，还需要更高层的安全性功能。其次，在协议栈的较高层上设置新的因特网服务（包括安全服务）通常较为容易。而等待在网络层上广泛地设置安全性，则可能要在未来的若干年才能解决，许多应用程序开发者“正在这样做”，并在他们钟意的应用程序中引入安全性功能。一个典型的例子就是PGP（Pretty Good Privacy），它提供了安全的电子邮件（将在本节稍后讨论）。由于只需要客户机和服务器应用程序代码，PGP是第一个在因特网上得到广泛应用的安全性技术。

8.5.1 安全的电子邮件

我们现在使用8.2～8.3节的密码学原则来生成一个安全电子邮件系统。我们以递进的方式来进行这个高层设计，每一步引入一种新的安全服务。当设计安全电子邮件系统时，我们需要记住最初在8.1节中所介绍的那个有趣的例子，即Alice和Bob之间的风流韵事。设想一下Alice发送一个电子邮件报文给Bob，而Trudy试图入侵的情况。

在做出为Alice和Bob设计一个安全电子邮件系统的努力之前，我们应当首先考虑他们最为希望的安全特性是什么。重中之重是*机密性*。正如8.1节讨论的那样，Alice和Bob都不希望Trudy阅读到Alice所发送的电子邮件报文。Alice和Bob最希望在该电子邮件系统中看到的第二个特性是具备*发送方鉴别*。特别是当Bob收到来自Alice的这样的报文：“I don't love you anymore. I never want to see you again. Formerly yours, Alice（我不再爱你了。我再也不想看到你了。Alice）”时，Bob自然而然地要确定这个报文确实来自Alice，而非Trudy发送的。另外，这两个情人欣赏的另一种特性是*报文完整性*，也就是说，确保Alice所发的报文在发送给Bob的过程中没有被改变。最后，电子邮件系统应当提供*接收方鉴别*，即Alice希望确定她的确正在向Bob发信，而不是向假冒Bob的其他人（如Trudy）发信。

因此我们从处理最为关注的机密性开始。提供机密性的最直接方式是Alice使用对称密钥技术（如DES或AES）加密所要传输的报文，而Bob则在接收时对报文解密。如在8.2节中讨论的那样，如果对称密钥足够长，且仅有Alice和Bob拥有该密钥，则其他人（包括Trudy）要想读懂这条报文极为困难。尽管这种方法直截了当，但是在分发对称密钥时使得仅有Alice和Bob具有该密钥是件非常困难的事（我们在8.2节中讨论过）。因此我们自然就考虑用其他方法——公钥密码系统（例如使用RSA）。在公钥方法中，Bob让他的公钥公开可用（例如，从一台公钥服务器或其个人网站上得到），Alice用Bob的公钥加密她的报文，然后将该加密的报文向Bob的电子邮件地址发送。（该加密的报文使用MIME首部封装，并通过普通的SMTP发送，如在2.4节中讨论的那样。）当Bob接收到这个报文时，只需用他的私钥即可解密之。假定Alice确定得到的公钥是Bob的公钥（并且此公钥足够长），则这种方法是提供所希望的机密性的极好方法。但是，存在的一个问题是公钥加密的效率相对低下，尤其对于长报文更是如此。（因为电子邮件越来越多地使用了附件、图像、音频和视频，在因特网上长电子邮件报文现在是司空见惯的。）

为了克服效率问题，我们利用了会话密钥（在8.2.2节中讨论过）。特别是，①Alice选择一

个随机对称会话密钥K_S，②用这个对称密钥加密她的报文m，③用Bob的公钥K_B^+加密这个对称密钥，④级连该加密的报文和加密的对称密钥形成一个“包”，⑤向Bob的电子邮件地址发送这个包。这些过程显示在图8-22中（在这张图和下一张图中，带圈的“+”表示级连，带圈的“−”表示级连的分解）。当Bob接收到这个包时，①他使用他的私钥K_B^-得到对称密钥K_S，②使用这个对称密钥K_S解密报文m。

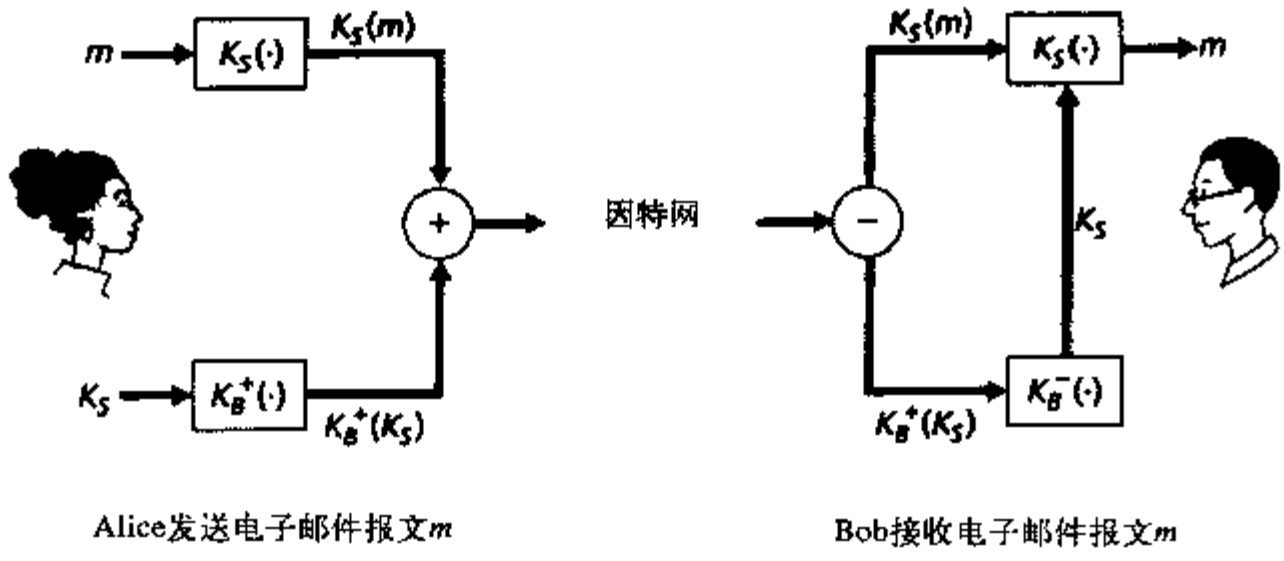


图8-22 Alice使用对称会话密钥K_S向Bob发送一个安全电子邮件

设计完提供机密性的安全电子邮件系统后，现在我们设计另一个可以提供发送方鉴别和完整性的系统。我们暂且假设Alice和Bob目前不关心机密性（他们要和其他人分享他们的爱情！），只关心发送方鉴别和报文完整性。为了实现这个任务，我们使用如8.3节所描述的数字签名和报文摘要。具体说来，①Alice对她要发送的报文m应用一个散列函数H（例如MD5），以得到一个报文摘要，②用她的私钥K_A^-对得到的散列签名，从而得到一个数字签名，③把初始报文（未加密）和该数字签名级连起来生成一个包，④向Bob的电子邮件地址发送这个包。当Bob接收到这个包时，①他用Alice的公钥K_A^+解密被签名的报文摘要，②将解密得到的报文摘要与他自己对该报文的散列H进行比较。在图8-23中图示了这些步骤。如8.3节中所讨论的，如果这两个结果相同，则Bob完全可以确信这个报文来自Alice且未被篡改。

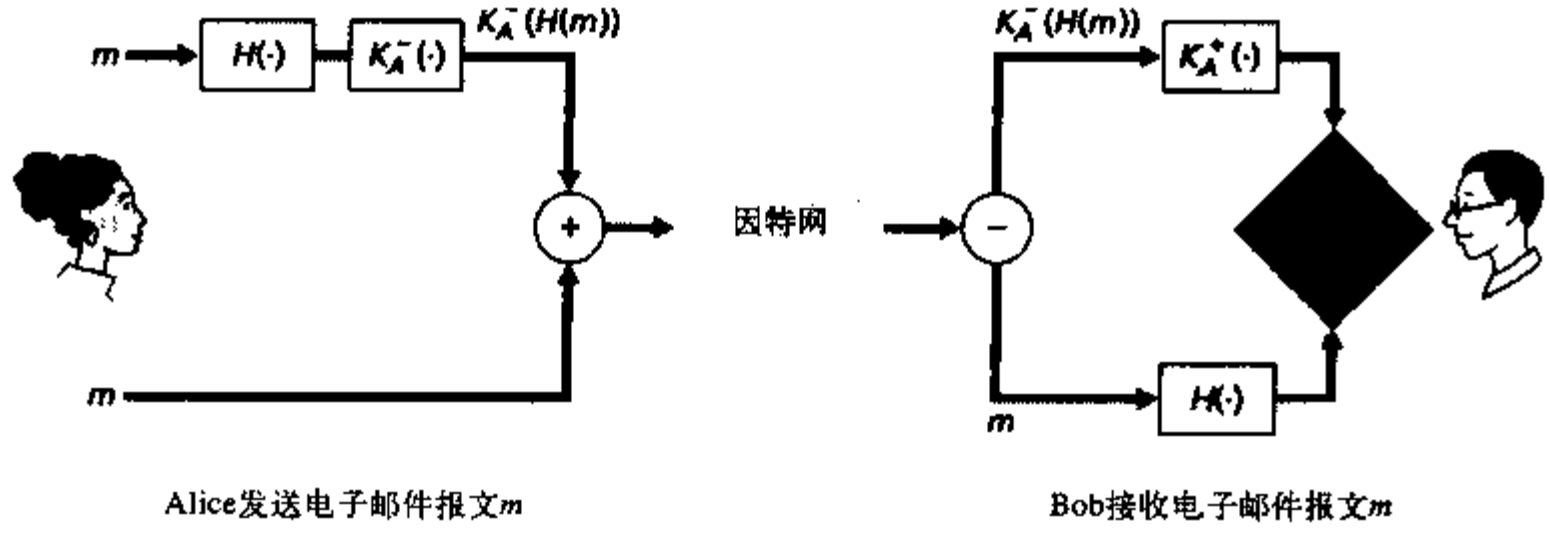


图8-23 使用散列函数和数字签名来提供发送方鉴别和消息完整性

现在我们考虑设计一个提供机密性、发送方鉴别和报文完整性的电子邮件系统。这可以把图8-22和图8-23中的过程结合来实现。Alice首先生成一个预备包，它与图8-23中的包完全相同，其中包含她的初始报文和该报文的数字签名散列。然后Alice把这个预备包看作一个报文，再用图8-22中的发送方的步骤发送这个新报文，即生成一个新包发给Bob。Alice所做的

这些步骤如图8-24所示。当Bob接收到这个包后，他首先应用图8-22中他这一侧的步骤，然后再应用图8-23中他这一侧的步骤。应当明确这一设计的目标是，提供机密性、发送方鉴别和报文完整性。注意到在这一方案中，Alice两次使用了公钥密码：一次用了她的私钥，另一次用了Bob的公钥。同样，Bob也两次使用了公钥密码：一次用了他的私钥，一次用了Alice的公钥。

图8-24所示的安全电子邮件系统可能在大多数情况下都能为大多数电子邮件用户提供满意的安全性。但是仍有一个重要的问题没有解决。图8-24中的设计要求Alice获得Bob的公钥，也要求Bob获得Alice的公钥。但这些公钥的分发并不是一个小问题。例如，Trudy可能假冒Bob，发给Alice她自己的公钥，并告诉Alice这个公钥是Bob的公钥，使得Trudy就能接收到Alice发给Bob的报文。我们在8.3节中学过，安全地分发公钥的一种常用方法是通过CA验证该公钥。

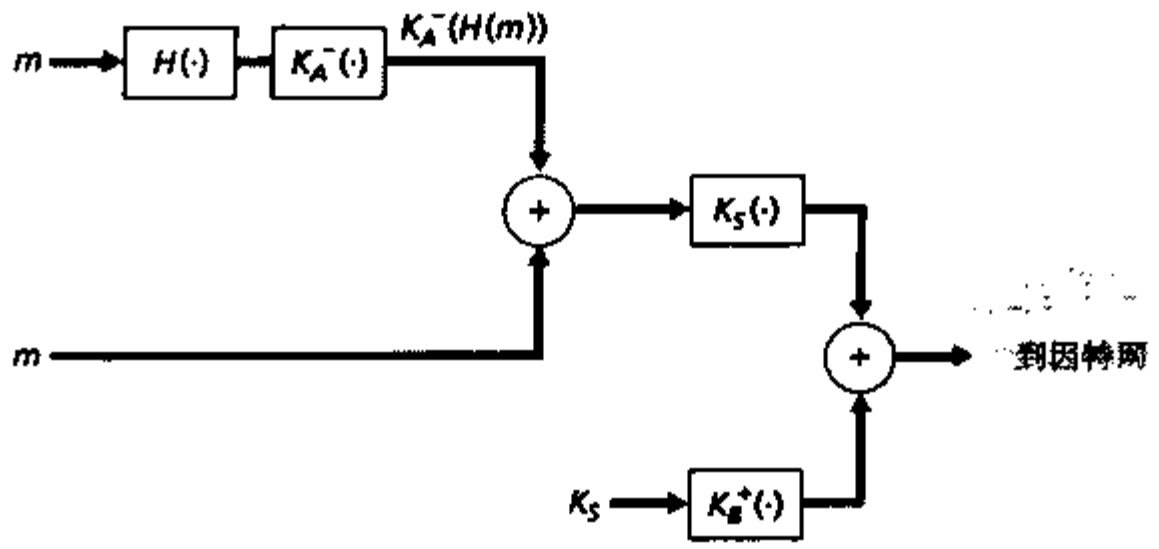


图8-24 Alice使用对称密钥密码、公开密钥密码、散列函数和数字签名以提供安全性、发送方鉴别和报文完整性

历史事件

Phil Zimmermann和PGP

Philip R. Zimmermann是PGP（Pretty Good Privacy）的创造者。他因此在长达三年的时间里成为犯罪调查的目标，因为政府认为在1991年以后的时间里，PGP在世界范围内以免费软件形式发布，违反了美国对加密软件出口的限制。在PGP作为共享软件发布后，有人把PGP放到了因特网上，美国以外的外国人也可以下载它。在美国，加密程序被划为军用品，依据联邦法律不得出口到国外。

尽管缺乏资金，没有收入，也没有大公司的支持，还要受到政府的干预，PGP还是成为了世界上广泛使用的电子邮件加密软件。奇怪的是，美国政府无意中促进了PGP的传播，因为Zimmermann案件使得它更为流行。

美国政府在1996年初放弃了这个案子。这一声明得到了许多因特网团体的喝彩。Zimmermann的案子已经成为无辜的个人为了自己的权益反抗强大的政府滥用职权的典故。政府的让步是值得庆幸的事，部分原因是由于在国会中对因特网审查制度的游说以及FBI的推动，目的是适应越来越多的政府侦听。

在政府撤案后，Zimmermann建立了PGP公司，该公司于1997年12月并入网络联盟(Network Associates)。Zimmermann现在是密码学领域中一位独立顾问。

8.5.2 PGP

Phil Zimmermann于1991年所写的PGP（Pretty Good Privacy）是一个电子邮件加密方案，如今已经成为一个事实上的标准。其Web站点以每个月百万页的规模，为在166个国家的用户提供服务[PGPI 2007]。公共领域中有PGP的各种版本可供使用，例如，能够在国际PGP的主页上为你喜爱的平台找到PGP软件以及许多有趣的读物[PGPI 2007]（PGP作者所撰写的一篇特别有趣的文章是[Zimmermann 2007]）。PGP在商业上也得到了应用，它作为许多电子邮件用户代理的插件，应用在包括Microsoft的Exchange和Outlook在内的许多软件中。PGP的设计在本质上和图8-24中所示的设计相同。PGP软件的不同的版本，使用MD5或使用SHA来计算报文摘要；使用CAST、三重DES或IDEA进行对称密钥加密；使用RSA进行公钥加密。另外，PGP还提供了数据压缩。

安装PGP时，软件为用户产生一个公开密钥对。该公钥能被张贴到其网站上或放置在一台公钥服务器上。私钥则使用口令进行保护。用户每次访问私钥时都要输入这个口令。PGP允许用户选择是否对报文进行数字签名、加密报文，或数字签名并加密。图8-25显示了一个PGP签名的报文。这个报文在MIME首部之后出现。报文中的加密数据为$K_A^-(H(m))$，即数字签名的报文摘要。如我们在上面所讨论的，Bob为了验证报文的完整性，需要具有Alice的公钥。

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash:  SHA1
Bob:
Can I see you tonight?
Passionately yours, Alice
-----BEGIN PGP SIGNATURE-----
Version: PGP for Personal Privacy 5.0
Charset:  noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZP9t6n7G6m5Gw2
-----END PGP SIGNATURE-----
```

图8-25 PGP签名报文

图8-26显示了一个秘密PGP报文。这个报文也出现在MIME首部之后。当然，明文报文并不包括在这个秘密电子邮件报文中。当一个发送方（例如Alice）要确保机密性和完整性时，PGP在如图8-25中的报文中包含一个类似于图8-26中的报文。

```
-----BEGIN PGP MESSAGE-----
Version: PGP for Personal Privacy 5.0
u2R4d+/jKmn8Bc5+hgDsqAewsDfrGdszX68liKm5F6Gc4sDfcXyt
RfdS10juHgbcfDssWe7/K=lKhnMikLo0+1/BvcX4t==Ujk9PbcD4
Thdf2awQfgHbnmKlok8iy6gThlp
-----END PGP MESSAGE
```

图8-26 一个秘密PGP报文

PGP也提供了公钥认证的机制，但是这种机制不同于更为传统的CA。PGP公钥由一个可信Web验证。当Alice相信一个密钥/用户名对确实匹配时，她自己就可以验证这一密钥/用户名对。此外，PGP允许Alice为其所信任的用户鉴别更多密钥提供担保。一些PGP用户通过保持密钥签署方（key-signing party）互相签署对方的密钥。用户实际走到一起，交换公钥，并用自己的私钥对对方的公钥签名来互相验证密钥。PGP公钥也可由因特网上的PGP公钥服务器（PGP public key servers）分发。当用户向一个这样的服务器提交公钥时，该服务器存储该公

钥的一个拷贝，并向所有其他公钥服务器发送该拷贝，并为请求这个公钥的用户发放这一公钥。尽管密钥签署方和PGP公钥服务器实际存在，但是到目前为止，用户分发其公钥的最常见方式还是把公钥发布在他们的个人Web网页上或通过电子邮件分发。

8.6 使TCP连接安全：SSL

在前一节中，我们看到对于一个特定的应用（即电子邮件），密码技术是怎样提供机密性、数据完整性和端点鉴别的。在这一节中，我们在协议栈中向下一层，研究如何使用密码技术来加强TCP的安全服务，包括机密性、数据完整性和端点鉴别。TCP的这种强化版本通常被称为**安全套接字层**（Secure Socket Layer，SSL）。SSL版本3的稍加修改的版本称为**运输层安全性**（Transport Layer Security，TLS），已经被IETF标准化了[RFC 2246]。

SSL最初由Netscape设计，但使TCP安全的基本思想是先于Netscape的工作的（例如，参见[Woo 1994]）。由于它的崭露头角，SSL已经得到了广泛部署。SSL得到了所有流行Web浏览器和Web服务器的支持，并基本上被用于所有因特网商业站点（包括Amazon, eBey, Yahoo!, MSN等等）。每年经SSL花费了数百亿美元。事实上，如果你使用信用卡经因特网购买任何东西的话，在你的浏览器和服务器之间的通信几乎一定使用了SSL。（当你使用浏览器时，若URL以https:开始而不是以http开始，就能认定正在使用SSL。）

为了理解SSL的需求，我们看一个典型的因特网商业的情况。Bob在Web上冲浪，到达了Alice公司的站点，这个站点正在出售香水。Alice公司站点显示了一个表格，假定Bob可以在该表格中输入香水的类型和所希望的数量、他的地址和他的支付卡号等信息。Bob输入这些信息，点击“提交”，就期待收到（通过普通邮政邮件）所购买的商品；他也期待着在他的下一次支付卡报表中收到对所购物品的支付信息。所有这一切听起来不错，但是如果不采取安全措施，Bob也许会有一些意外。

- 如果没有使用机密性（加密），一个入侵者可能截取Bob的订单并得到他的支付卡信息，然后这个入侵者可以用Bob的费用来购买商品。
- 如果没有使用完整性，入侵者可能修改Bob的订单，让他购买比希望多十倍瓶数的香水。
- 最后，如果没有使用服务器鉴别，这个显示Alice公司著名徽标的服务器可能实际上是由Trudy维护的一个站点，Trudy正在假冒Alice公司。当Trudy收到Bob的订单后，可能拿了Bob的钱一走了之。或者Trudy可能充当一名身份窃贼，收集Bob的名字、地址和信用卡号码。

SSL通过采用机密性、数据完整性、服务器鉴别和客户机鉴别来强化TCP，从而处理这些问题。

SSL经常用于为发生在HTTP之上的事务提供安全性。然而，因为SSL使TCP安全了，因此它能被应用于运行在TCP之上的任何应用程序。SSL提供了一个简单的具有套接字的应用编程接口（API），该接口类似于TCP的API。当一个应用程序要使用SSL时，该应用程序包括为应用程序研发者提供SSL套接字接口的SSL类/库。如图8-27所示，尽管SSL技术上位于应用层中，但从研发者的角度看，提供加强

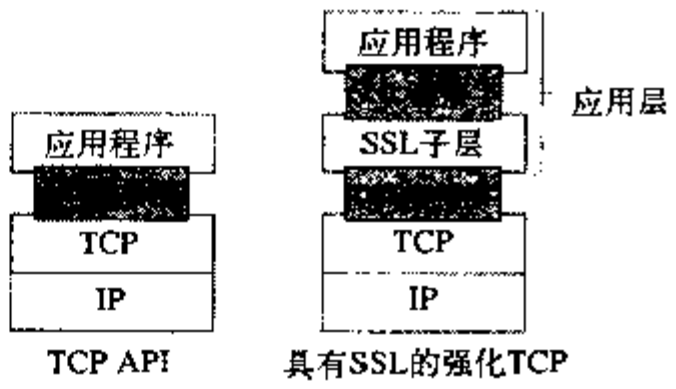


图8-27 尽管SSL技术上位于应用层中，但从研发者的角度看它是运输协议

安全性服务的TCP服务是运输协议。

8.6.1 宏观描述

我们从描述一个简化的SSL版本开始，这将使我们从宏观上理解SSL的工作原理和工作过程。我们将这个SSL的简化版本称之为“类SSL”。描述过类SSL之后，在下一小节中我们将详细描述真实的SSL。类SSL（和SSL）具有三个阶段：握手，密钥导出和数据传输。我们现在针对一台客户机（Bob）和一台服务器（Alice）之间的通信会话描述这三个阶段，其中Alice具有私钥/公钥对和将她的身份与其公钥绑定的证书。

1. 握手

在握手阶段，Bob需要：①与Alice创建一条TCP连接，②验证Alice是真实的Alice，③发送给Alice一个主密钥，该主密钥是Bob和Alice用来生成SSL会话所需的所有对称密钥。这三个步骤显示在图8-28中。注意到一旦创建了TCP连接，Bob就向Alice发送一个Hello报文。Alice则用她的证书进行响应，证书中包含了她的公钥。如在8.3节讨论的那样，因为该证书已被某CA证实过，Bob明白无误地知道该公钥属于Alice。然后，Bob产生一个主密钥（MS）（该MS将仅用于这个SSL会话），用Alice的公钥加密该MS以生成加密的主密钥（EMS），并将该EMS发送给Alice。Alice用她的私钥解密该EMS从而得到该MS。在这个阶段后，Bob已经鉴别了Alice，并且Bob和Alice（而无别的人）知道了用于这次SSL会话的主密钥。

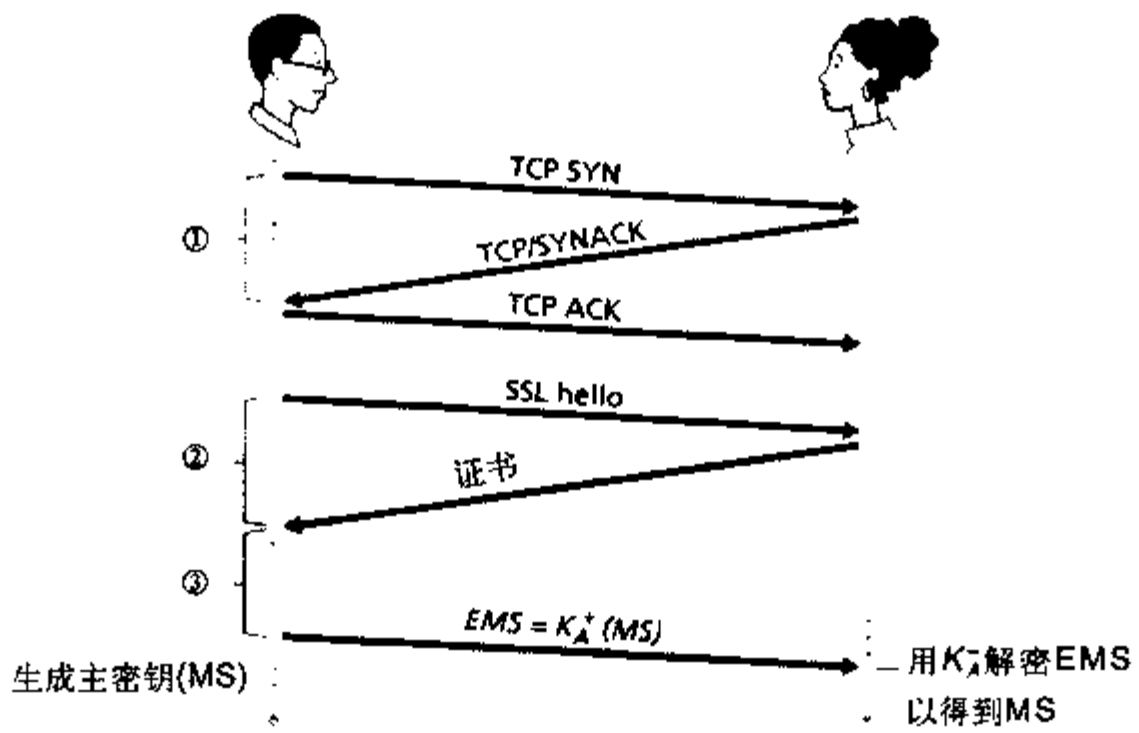


图8-28 类SSL握手，首先建立一个TCP连接

2. 密钥导出

从原则上讲，MS此时已由Bob和Alice共享，它能够用于所有后继加密和数据完整性检查的对称会话密钥。然而，对于Alice和Bob而言，每使用不同的密码密钥，并且对于加密和完整性检查也使用不同的密钥，通常认为更为安全。因此，Alice和Bob使用MS生成了四个密钥：

- E_B=用于从Bob发送到Alice数据的会话加密密钥。
- M_B=用于从Bob发送到Alice数据的会话MAC密钥。
- E_A=用于从Alice发送到Bob数据的会话加密密钥。
- M_A=用于从Alice发送到Bob数据的会话MAC密钥。

Alice和Bob每人都从MS生成4个密钥。这能够通过直接将该MS分为4个密钥来实现（但在真实的SSL中情况更为复杂一些，我们后面将会看到）。在密钥导出阶段最后，Alice和Bob都有了4个密钥。其中的两个加密密钥将用于加密数据；两个MAC密钥将用于验证数据的完整性。

3. 数据传输

既然Bob确信他正在与Alice交谈，并且Alice和Bob共享相同的4个会话密钥（E_B，M_B，E_A和M_A），他们就能够经TCP连接开始发送安全的数据。因为TCP是一种字节流协议，一种自然的方法是对SSL在传输中加密应用数据，然后将加密的数据在传输中传给TCP。但是如果这样做的话，我们将用于完整性检查的MAC置于何处？我们无疑不希望等到TCP会话的结束才验证所有Bob数据的完整性，Bob数据的发送要经历整个会话!为了解决这个问题，SSL将数据流分割成记录，为了完整性检查对每个记录附加一个MAC，然后加密该“记录＋MAC”。为了产生这个MAC，Bob将数据连同密钥M_B放入一个散列函数中，如在8.3节讨论的那样。为了加密“记录＋MAC”这个包，Bob使用他的会话加密密钥E_B。这个加密的包被传递给TCP经因特网传输。

虽然这种方法几经周折，然而它为整个报文流提供数据完整性时仍未达到无懈可击。特别是，假定Trudy是一名“中间妇女”，并且有在Alice和Bob之间发送的TCP报文段的流中插入、删除和代替报文段的能力。例如，Trudy能够俘获由Bob发送的两个报文段，颠倒这两个报文段的次序，调整TCP报文段的序号（这些未被加密），然后将这两个次序翻转的报文段发送给Alice。假定每个TCP报文段封装了正好一个记录，我们现在看看Alice是如何处理这些报文段的。

1）在Alice端运行的TCP将认为一切正常，将这两个记录传递给SSL子层。

2）在Alice端的SSL将解密这两个记录。

3）在Alice端的SSL将使用每个记录中的MAC来验证这两个记录的数据完整性。

4）然后SSL将解密的两条记录的字节流传递给应用层，但Alice收到的完整字节流由于记录的颠倒而次序不正确。

鼓励读者考虑类似的情况，如Trudy删除报文段或Trudy重放报文段。

对该问题的解决方案正如所猜想的那样，就是使用序号。SSL采用如下的方式。Bob维护一个序号计数器，它开始为0，对他发送的每个SS L记录都增加1。Bob并不实际在记录中包括一个序号，但当他计算MAC时，他在MAC的计算中包括该序号。所以，该MAC现在是数据加MAC密钥M_B加当前序号的散列。Alice跟踪Bob的序号，通过在MAC的计算中包括适当的序号，使她验证一条记录的数据完整性。使用SSL序号阻止了Trudy执行诸如重新安排顺序或重放报文段等中间人攻击。（为什么？）

4. SSL记录

SSL记录（以及类SSL记录）显示在图8-29中。该记录是由类型字段、版本字段、长度字段、数据字段和MAC字段组成。注意到前三个字段是不加密的。类型字段指出了该字段是握手报文还是包含应用数据的报文。它也用于关闭SSL连接，如下面所讨论。在接收端的SSL使用长度字段以从到达的TCP字节流中提取SSL记录。版本字段是自解释的。

图8-29 SSL记录格式

8.6.2 更完整的描述

前一小节讨论了类SSL协议，其目的是让我们对SSL的工作原理和工作过程有一个基本理解。既然我们已经对SSL有了基本了解，下面能够更深入研究实际SSL协议的要点了。为了便于理解对SSL协议的描述，鼓励读者完成Ethereal SSL实验，这在本教科书配套的Web网站上提供。

1. SSL握手

SSL并不强制Alice和Bob使用某种特定的对称密钥算法、特定的公钥算法或特定的MAC。相反，SSL允许Alice和Bob在握手阶段的SSL会话开始时，就密码算法取得一致。此外，在握手阶段，Alice和Bob彼此发送不重数，该数被用于会话密钥（E_B，M_B，E_A和M_A）的生成中。真实的SSL握手的步骤如下：

1）客户机发送它支持的密码算法的列表，连同一个客户机的不重数。

2）从该列表中，服务器选择一种对称算法（例如AES）、一种公钥算法（例如具有特定密钥长度的RSA）和一种MAC算法。它向客户机发送回它的选择，以及证书和一个服务器不重数。

3）客户机验证该证书，提取服务器的公钥，生成一个主密钥MS，用服务器的公钥加密该MS，并将其发送给服务器。

4）使用相同的密钥导出函数（就像SSL标准定义的那样），客户机和服务器独立地从MS和不重数中计算加密和MAC密钥。自此以后，客户机和服务器之间发送的所有报文均被加密和鉴别（使用MAC）。

5）客户机发送所有握手报文的一个MAC。

6）服务器发送所有握手报文的一个MAC。

最后两个步骤保护了握手免受篡改。为了理解这一点，观察在第一步中，客户机通常提供一个算法列表，有些算法强，有些算法弱。因为这些加密算法和密钥还没有被协商好，算法的这张列表以明文形式发送。Trudy作为中间人，能够从列表中删除较强的算法，迫使客户机选择一种较弱的算法。为了防止这种篡改攻击，在步骤5中客户机发送一个级连它发送和接收的所有握手报文的MAC。如果有不一致，服务器能够终止该连接。类似地，服务器发送一个它已经看到的握手报文的MAC，使得客户机核对不一致性。

2. 连接关闭

在某个时刻，Bob或者Alice将要终止SSL会话。一个方法是让Bob通过直接终止支撑的TCP连接来结束该SSL会话，即通过Bob向Alice发送一个TCP FIN报文段。但是这种幼稚的设计为*截断攻击*（truncation attack）创造了条件，Trudy再一次介入一个进行中的SSL会话中，并用TCP FIN过早地结束了该会话。如果Trudy这样做的话，Alice将会认为她收到了Bob的所有数据，而实际上她仅收到了其中的一部分。对这个问题的解决方法是，在类型字段中指出该记录是否是用于终止该SSL会话的。（尽管SSL类型是以明文形式发送的，但在接收方可以

使用记录的MAC对此进行鉴别。）通过包括了这样一个字段，如果Alice在收到了一个关闭SSL记录之前突然收到了一个TCP FIN，她可能知道正在进行着某些耍花招的事情。

我们完成了对SSL的介绍。我们已经看到了它使用了8.2节和8.3节中讨论的许多密码学原则。希望更深入地研究SSL的读者可以阅读Rescorla的有关SSL的可读性很强的书籍[Rescorla 2001]。

8.7 网络层安全性：IPsec

IP安全（IP Security）协议更常被称为IPsec，它是为网络层提供安全性的一组协议。如在第4章所述，IPsec处于大多数虚拟专用网（VPN）的核心位置。IPsec相当复杂，10多个RFC文档对它的不同部分分别进行了描述。在本节中，我们在特定的环境下讨论IPsec，即其中的两台通信的主机都装备了IPsec。RFC 4301和RFC 2411是IPsec的两个关键性文档，前者描述了总体IP安全体系结构，后者提供了一个IPsec协议集的概述。

在深入分析IPsec技术细节之前，我们先后退一步考虑在网络层提供安全性意味着什么。首先考虑提供**网络层机密性**（network-layer confidentiality）是什么含义。如果由IP数据报携带的所有有效载荷都被加密了，则该网络层将提供机密性。这意味着主机无论何时要发送数据报，在它向网络发送之前要加密数据报的有效载荷。原则上，这种加密可用对称密钥密码、公钥密码或通过公钥密码协商的会话密钥加密完成。该有效载荷可能是一个TCP报文段、一个UDP报文段或一个ICMP报文等。如果这样的网络服务可用的话，则所有由主机发送的数据包括电子邮件、Web页、控制报文和管理报文（如ICMP和SNMP），对可能嗅探网络的任何第三方都是隐藏的。因此，这种服务可以为所有因特网流量提供某种全面覆盖，进而为我们提供了某种意义上的安全性。

除了机密性，我们也需要网络层提供**源鉴别**（source authentication）。当目的主机接收一个有特定IP源地址的IP数据报时，它通过判断这个IP数据报是否确实由具有这个特定IP源地址的主机生成而进行源鉴别。这种服务可防止IP地址哄骗。

在IPsec协议族中有两个主要协议：**鉴别首部协议**（Authentication Header (AH) protocol）和**封装安全性载荷协议**（Encapsulation Security Payload (ESP) protocol）。当源主机向目的主机发送安全数据报时，可以使用AH协议或ESP协议。AH协议提供源鉴别和数据完整性服务，但是不提供机密性服务。ESP协议提供鉴别、数据完整性和机密性服务。因为ESP协议提供了更多的服务，所以它比AH协议更复杂，也需要进行更多的处理。

在AH和ESP这两个协议中，在从源主机向目的主机发送安全数据报之前，源主机和网络主机握手并创建了一个网络层的逻辑连接。这个逻辑通道称为**安全关联**（Security Association，SA）。因此，IPsec把因特网传统的无连接网络层转化成了一个具有逻辑连接的层。这个由一个SA定义的逻辑连接是一个单工连接；也就是说，它是单向的。如果两台主机间要互相发送安全数据报，则需创建两个SA，每个方向一个。SA由一个称为安全参数索引（Security Parameter Index, SPI）的32比特的连接标识符标识。特别是，每一个IPsec数据报都有一个用于SPI的首部字段。在相同SA中（即在从源到目的地主机相同的逻辑连接中）的所有数据报具有相同的SPI。

8.7.1 鉴别首部协议

前面讲过，AH协议[RFC 4302; RFC 4305]提供了源鉴别和数据完整性服务，但是不提供

机密性服务。当一个特定的源主机要向一个特定目的地发送一个或多个数据报时，它首先和该目的地创建一个SA。在创建了SA以后，源和目的主机共享一个秘密鉴别密钥。我们假定源主机具有共享密钥，能够被信任，特别是它的IP地址不会被哄骗。

创建了一个SA并有了一个共享的密钥，源主机就可向目的主机发送安全数据报了。如图8-30所示，安全数据报包括AH首部，该首部插在初始IP数据报载荷（例如一个TCP或UDP报文段）和IP首部之间。所以，AH首部增大了初始有效载荷，这个增大的数据字段当作一个标准的IP数据报封装起来。对于在IP首部中的协议字段，使用值51指示该数据报包含一个AH首部。当目的主机接收到该IP数据报时，检查到协议字段值为51时，就会用AH协议来处理该数据报。（前面讲过IP数据报的协议字段被用于确定上层协议，如UDP、TCP或ICMP，这些协议的报文段放在IP数据报的数据部分传输。）中间路由器对这些数据报的处理与往常一样，即检查目的IP地址并相应地转发它们。

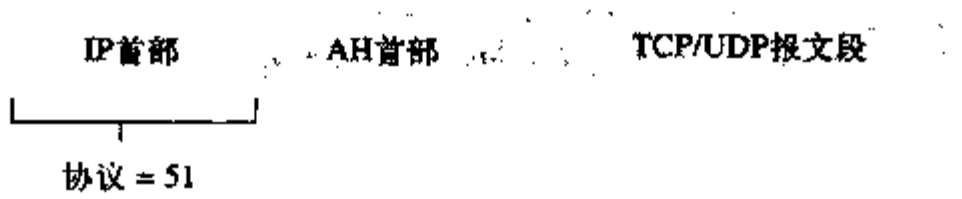


图8-30 在IP数据报中的AH首部的位置

AH首部包括几个字段：

- 下一个首部（Next Header）字段。该字段的作用相当于普通数据报的协议字段。它表明了在AH首部之后的数据是TCP报文段、UDP报文段、ICMP报文段还是其他类型。（前面讲过该IP数据报的协议字段现在被用于表示AH协议了，所以它不能再用于指示运输层协议。）
- 安全性参数索引（Security Parameter Index，SPI）字段。一个任意32比特的值，与目的IP地址和安全协议结合使用，唯一地标识该数据报的SA。
- 序号（Sequence Number）字段。是一个32比特的字段，包含针对每个数据报的序号。在初始创建一个SA时，该字段置为0。AH协议用这个序号防止重放攻击和中间人攻击（类似于SSL，参见8.6节）。
- 鉴别数据（Authentication Data）字段。是一个可变长字段，包含对该数据报的MAC。使用这个共享的密钥，该MAC是对初始IP数据报以及AH首部（除了IP TTL字段和AH鉴别数据字段）计算而得的。对于该MAC，借助于由SA规定的散列函数（如MD5或SHA），IPsec使用HMAC（参见8.3节）。

当目的主机接收到一个具有AH首部的IP数据报时，它通过处理鉴别数据字段来决定该数据报的SA，进而鉴别该数据报的完整性和源。

8.7.2 ESP协议

ESP协议[RFC 4303; RFC 4305]除了提供源主机鉴别和数据完整性外，还提供网络层机密性。同样，它也要以源主机与目的主机创建SA开始。在创建SA以后，源和目的主机共享一个秘密加密密钥和一个秘密鉴别密钥。然后源主机才能够向目的主机发送安全数据报。如图8-31所示，通过在初始IP数据报有效载荷前后分别添加首部和尾部字段，并把这个封装后的数据插入进一个IP数据报的有效载荷字段，生成了一个安全数据报。对于在该IP数据报的首部中的协议字段来说，值50用于指示这个数据报包括了ESP首部和尾部。当目的主机接收到这个IP

数据报时，检查到协议字段的值为50时，就会用ESP协议来处理这个数据报。如图8-31所示，初始IP数据报有效载荷连同ESP尾部都被加密了。ESP首部由一个32比特的SPI字段和一个32比特的序号字段组成，这两个字段的作用与它们在AH中的相同。尾部包括了下一个首部字段，该字段的作用也与其在AH中的相同。注意到因为下一个首部字段连同源数据都被加密，入侵者就不能确定正在使用的运输层协议。紧跟尾部有一个鉴别数据字段，该字段仍与其在AH协议中的作用相同。有关ESP协议的进一步细节能够在文献[RFC 4303; RFC 4305]中找到。

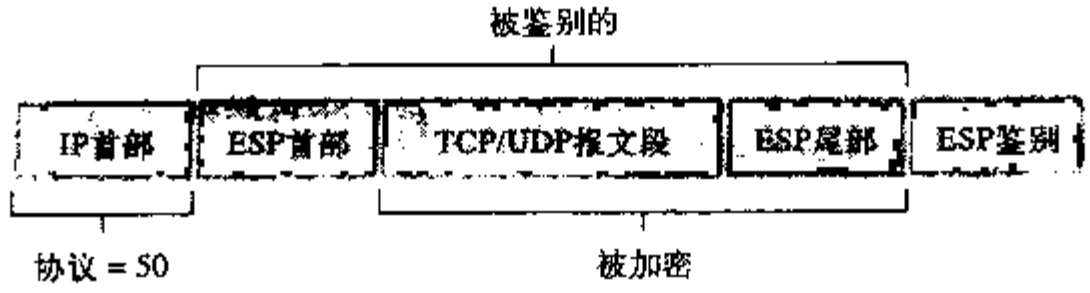


图8-31 在IP数据报中的ESP字段

8.7.3 SA和密钥管理

在8.7.1节中我们看到AH协议使用了一个秘密鉴别密钥，该密钥由源和目的主机所共享。在8.7.2节中我们看到ESP协议附加使用一个秘密加密密钥，该密钥用于对称加密。但是源和目的主机是如何获得秘密密钥的呢？此外，源和目的主机如何就密码算法（用于对称密钥加密和用于HMAC）取得一致的呢？

大体说来，有两类方法：

- 人工的：系统管理员手工为主机配置密码算法和秘密密钥。
- 自动的：对每个SA而言，密码算法和针对每个SA的密钥自动地按需获取。这能使用因特网密钥交换（Internet Key Exchange，IKE）算法[RFC 2409]完成。如你可能猜测的那样，IKE使用公钥密码来分发这些密钥。IKE能够以几种方式之一完成，参见[Kaufman 1995]的细致讨论。

这些内容构成了我们对IPsec的小结。我们已经在IPv4 和传输模式的环境下讨论了IPsec。IPsec还定义了隧道模式，在隧道模式下是路由器而不是主机引入了安全功能。最后，IPsec描述了除了用于IPv4也用于IPv6的加密过程。

8.8 使无线LAN安全

在无线网络中，安全性是特别重要的考虑因素，因为这时携带数据帧的无线电波可以传播到远离包含无线基站和主机的建筑物以外的地方。在本节中，我们对无线安全性做了简要介绍。对于更为深入的探讨，参见由Edney和Arbaugh撰写的可读性很强的书[Edney 2003]。

在802.11中的安全性问题受到了技术界和传播媒体的极大关注。在进行大量讨论的同时，一个几乎没有争论的事实是，看起来被广泛认同的初始802.11规范有一些严重的安全性缺陷。事实上，现在能够下载的利用这些漏洞的公共域软件，使得那些使用这种802.11安全性机制的用户面对安全性攻击，就像根本没有使用安全性措施的网络用户一样，门户洞开。

在下面一节中，我们讨论在最初的802.11规范中标准化的安全性机制，统称为**有线等效保密**（Wired Equivalent Privacy，WEP）。顾名思义，WEP意欲提供类似于有线网络中的安全性水平。接下来我们将讨论WEP中的安全性漏洞并讨论802.11i标准，这是在2004年采纳的802.11的更为安全的基础版本。

8.8.1 有线等效保密

IEEE 802.11的WEP协议[IEEE 802.11 1999]使用对称共享密钥方法，在主机和无线接入点（即基站）之间提供鉴别和数据加密。WEP并没有指定密钥管理算法，因此假定该主机和无线接入点之间通过带外方式就密钥达成某种一致。鉴别以我们在8.4节中开发的ap4.0协议的方式进行。其中包括了4步：

1）无线主机通过接入点请求鉴别。

2）该接入点以一个128字节的不重数值响应该鉴别请求。

3）该无线主机用它与这个接入点共享的密钥加密这个不重数值。

4）该接入点解密主机加密的不重数值。

如果解密所得的不重数值与最初发给主机的值相同，则接入点鉴别了该主机。

图8-32显示了WEP数据加密算法。假定主机和接入点都知道一个秘密的40比特的对称密钥K_S。此外，有一个24比特的初始向量（IV）添加在这个40比特的密钥后，以产生用于加密单个帧的64比特密钥。每一个帧所使用的IV都不同，所以每一帧都由不同的64比特密钥加密。加密以如下方式进行：首先为每个数据有效载荷计算一个4字节的CRC值（见5.2节），然后用RC4的流密码加密该有效载荷和该4字节CRC值。我们这里不涉及RC4的细节（细节请参见[Schneier 1995]和[Edney2003]）。就我们的目的而言，知道下列事实即可：对于密钥值（此时为64比特（K_S, IV）密钥值），RC4算法产生的密钥值的流为k_1^{IV}, k_2^{IV}, k_3^{IV}, ...，这些密码值用于加密一帧中的数据和CRC值。出于实用的目的，我们可以认为每次对一个字节执行这些操作。加密通过把数据的第i字节d_i和由（K_S, IV）对生成的密钥值流中的第i个密钥k_i^{IV}执行异或操作进行，以产生密文的第i字节c_i：

$$c_i = d_i \oplus k_i^{IV}$$

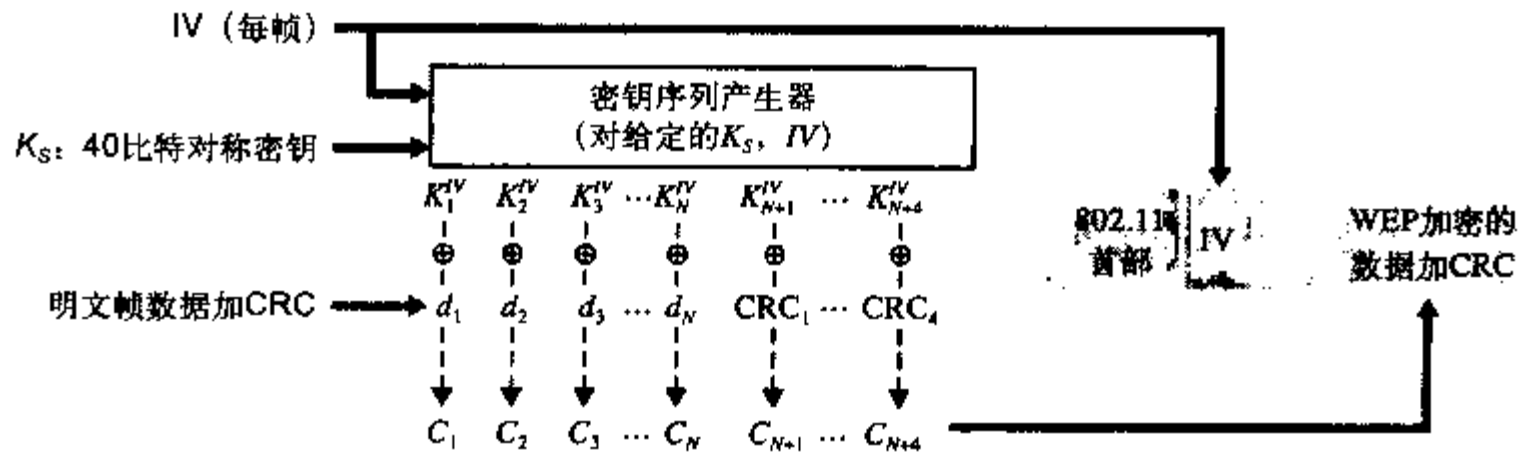


图8-32 802.11 WEP协议

如图8-32中所示，该IV值逐帧而变化，以明文形式出现在每一个WEP加密的802.11帧的首部中。接收方用它与发送方共享的秘密40比特对称密钥，添加这个IV后，并使用形成的64比特的密钥（它与发送方执行加密所用的密钥相同）来解密这个帧。

$$d_i = c_i \oplus k_i^{IV}$$

妥善使用RC4算法要求同一个64 比特密钥决不能使用超过1次。前面讲过WEP密钥是每一帧变换一次。对于某给定的K_S（如果它有变化的话也是很少的），这意味着只有2^{24}个不同的密钥可用。如果随机选择这些密钥的话，我们能够看到[Walker 2000；Edney 2003]，则仅在处理12 000帧之后选中相同IV值（从而使用相同64比特密钥）的概率超过99%。在1K帧长和11Mbps 的数据传输率的情况下，传输12 000帧仅需几秒的时间。此外，由于IV值在该帧中以

明文形式传输，窃听者就会发现何时使用了一个重复的IV值。

为了理解重复使用一个密钥可能出现的几个问题之一，考虑下面的选择明文攻击的情况，仍以Trudy对Alice进行攻击为例。假定Trudy（可能使用IP哄骗）向Alice发出一个请求（例如，一个HTTP或FTP请求），要求Alice传输内容已知的文件$d_1, d_2, d_3, d_4, \cdots$，Trduy也观察到Alice发送的已加密数据$c_1, c_2, c_3, c_4, \cdots$，由于$d_i = c_i \oplus k_i^{IV}$，如果在这个等式两边同时异或$c_i$，可得到：

$$d_i \oplus c_i = k_i^{IV}$$

根据这个关系，Trudy就可以使用已知的d_i和c_i值计算出k_i^{IV}。下一次Trudy看到使用同一IV值时，就知道密钥流为$k_1^{IV}, k_2^{IV}, k_3^{IV}, \cdots$，并可使用这些密钥解密报文。

对于WEP还有几个其他安全性关注点。[Fluhrer 2001]描述了一种攻击方法，即当选择某些弱密钥时在RC4中暴露出的一种已知弱点。[Stubblefield 2002]讨论了实现和开发这种攻击的有效方法。对WEP的另一种关注与在图8-32中显示并在802.11帧中传输的CRC比特有关，用以检测在有效载荷中改变的比特。然而，攻击者在改变被加密内容（例如用乱七八糟的东西替代初始的被加密数据）后，对这些被替换的东西计算出一个CRC，并将该CRC放置在WEP帧中能够产生一个将被接收方接受的802.11帧。此时所需要的是诸如我们在8.3节中学习的报文完整性技术，以检测内容篡改或替换。有关WEP安全性更多的细节，请参见[Edney2003；Walker 2000；Weatherspoon 2000；802.11 Security 2007]及其中的参考文献。

8.8.2 IEEE 802.11i

IEEE 802.11于1999年发布后不久，就开始了研发具有更强安全性机制的新型的、改进的802.11版本。这个新标准被称为802.11i，在2004年最终得到批准。如我们将看到的那样，虽然WEP提供了相对弱的加密，执行鉴别仅有单一机制，并且没有密钥分发机制，但IEEE 802.11i却提供了强得多的加密形式，一种可扩展的鉴别机制的集合，以及一种密钥分发机制。下面我们概述一下802.11i，[TechOnline 2004]是关于802.11i的优秀（流音频）技术概述。

图8-33概述了802.11i的框架。除了无线客户机和接入点外，802.11i定义了一台鉴别服务

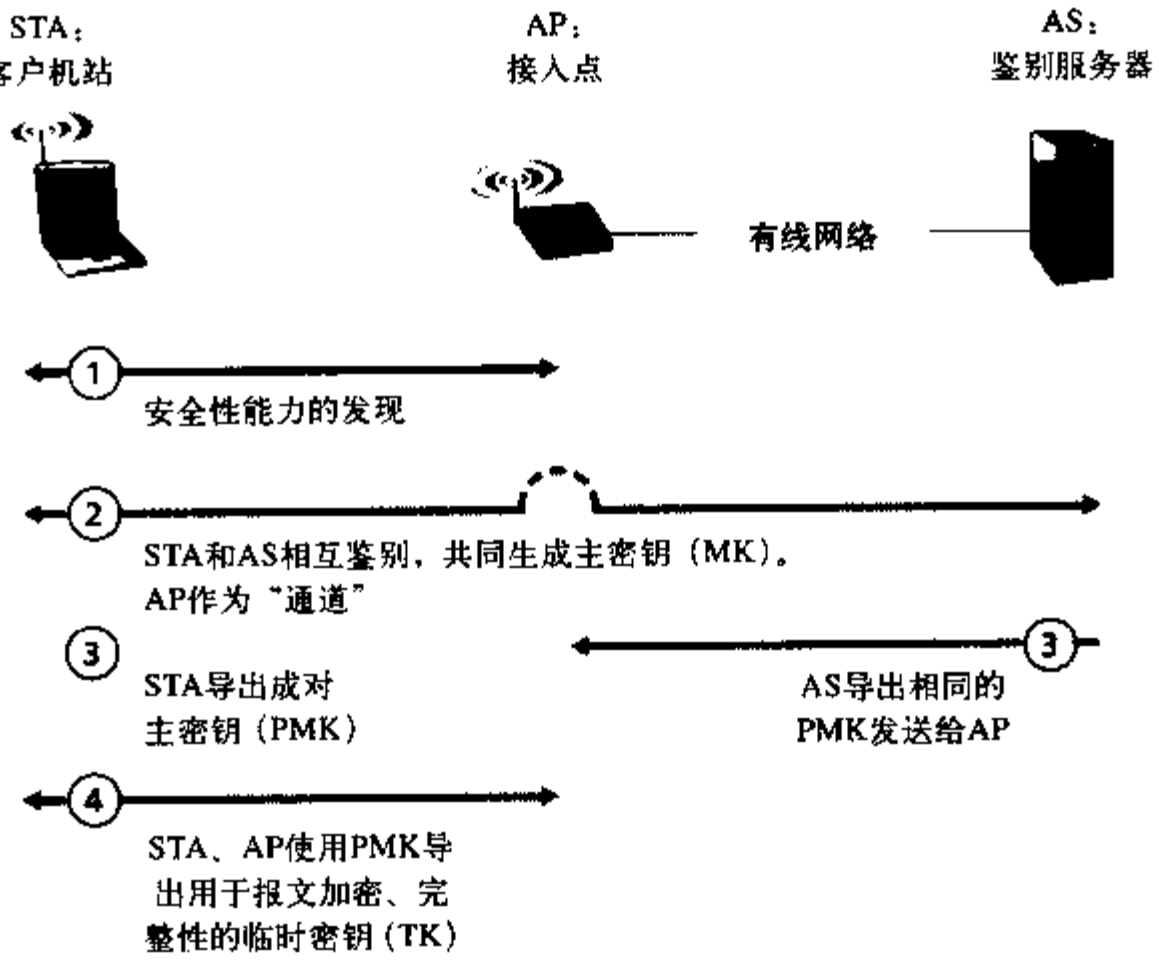


图8-33 802.11i：运行的4个阶段

器，AP能够与它通信。将鉴别服务器与AP分离，使得一台鉴别服务器服务于许多AP，集中在一台服务器中作出有关鉴别和接入（通常是敏感）的决定，降低了AP的成本和复杂性。802.11i运行分为4个阶段：

1）发现。在发现阶段，AP通告它的存在并能够向无线客户机节点提供鉴别和加密的格式。该客户机则请求它希望的特定鉴别和加密格式。尽管客户机和AP已经交换了报文，但该客户机还没有被鉴别，也没有加密密钥，因此在该客户机通过无线信道能够与任何远程主机通信之前，还需要进行几个其他步骤。

2）相互鉴别和主密钥（MK）生成。鉴别发生在无线客户机和鉴别服务器之间。在这个阶段，接入点基本是起中继的作用，在客户机和鉴别服务器之间转发报文。**可扩展鉴别协议**（Extensible Authentication Protocol, EAP）[RFC 2284]定义了一种端到端报文格式，用于客户机和鉴别服务器之间交互的简单的请求/响应模式。如图8-34所示，EAP报文使用EAPoL（EAP over LAN, [IEEE 802.1X]）封装报文并通过802.11无线链路发送。这些EAP报文在接入点被拆封，然后再使用RADIUS协议重新封装，经UDP/IP传输到鉴别服务器。虽然RADIUS服务器和协议[RFC 2865]并不为802.11i所要求，但它们是802.11i的事实上的标准组件。最近标准化的DIAMETER协议[RFC 3588]很可能在不久的将来会替代RADIUS。

使用EAP，鉴别服务器能够选择若干方式中的一种来执行鉴别。802.11i虽未强制一种特殊的鉴别方法，常常使用EAP-TLS鉴别方案[RFC 2716]。EAP-TLS使用类似于我们在8.3节和8.4节中研究的公钥技术（包括不重数加密和报文摘要），允许客户机和鉴别服务器彼此相互鉴别，并导出为双方所知的一个主密钥（MK）。

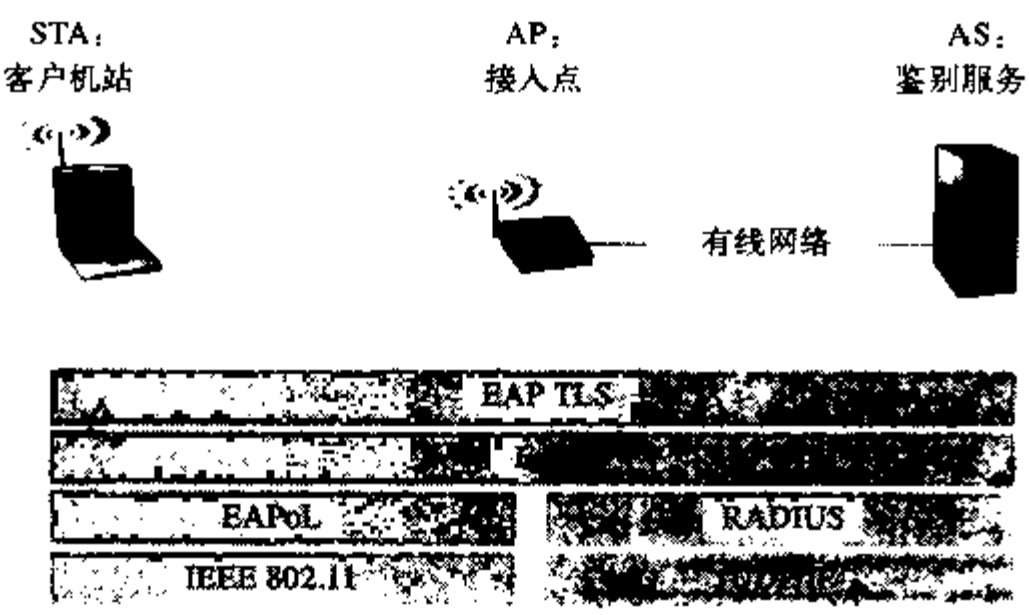


图8-34 EAP是一个端到端协议。EAP报文使用EAPoL封装，运行在客户机和接入点之间的无线链路之上，并使用RADIUS运行在接入点和鉴别服务器之间的UDP/IP之上

3）成对主密钥（Pairwise Master Key, PMK）生成。MK是一个仅为客户机和鉴别服务器所知的共享密钥，它们彼此使用MK生成一个次密钥，即成对主密钥（PMK）。鉴别服务器向AP发送该PMK。这正是我们所希望达到的目的！客户机和AP现在具有一个共享的密钥（前面讲过在WEP中根本不涉及密钥分发问题），并彼此相互鉴别。它们此时已经快要能发挥效用了。

4）临时密钥（Temporal Key, TK）生成。使用PMK，无线客户机和AP现在能够生成附加的用于通信的密钥。其中关键是临时密钥，TK将被用于执行经无线链路向任意远程主机发送

数据的链路级的加密。

802.11i提供了几种加密形式，包括一种基于AES的加密方案和WEP加密的强化版本。

8.9 运行安全性：防火墙和入侵检测系统

从本章的内容我们可以看到，因特网并不是非常安全的地方，有“坏家伙”出没，从事着各种各样的破坏活动。我们考虑一个机构网络和管理它的网络管理员。从网络管理员的角度看，世界可以很清楚地分为两个阵营。一部分是好人，他们属于机构网络，可以以相对不受限制的方式访问该机构网络中的资源；另一部分是恶意攻击者，必须经过仔细审查才能确定是否允许他们访问网络资源。在许多机构中，从中世纪的城堡到现代化公司的建筑物，都有单一的出口/入口点，无论好人坏人出入该机构，都需要进行安全检查。在一个城堡中，可以在吊桥的一端的门口执行安全检查；在公司大厦中，这些工作可由安全台来完成。在计算机网络中，当通信流量进入/离开网络时要执行安全检查，被记录、丢弃和/或转发，这些工作都由称为防火墙、入侵检测系统（IDS）和入侵防止系统（IPS）的运行设备完成。

8.9.1 防火墙

防火墙（firewall）是一个硬件和软件的结合体，它将一个机构的内部网络与整个因特网隔离开，允许一些数据分组通过而阻止另一些通过。防火墙允许网络管理员控制外部世界和被管理网络内部资源之间的访问，这种控制是通过管理流入和流出这些资源的流量实现的。防火墙具有3个目标：

- 从外部到内部和从内部到外部的所有流量都通过防火墙。图8-35显示了一个防火墙，位于被管理网络和因特网其余部分之间的边界处。虽然许多大机构可使用多级防火墙或分布式防火墙[Skoudis 2006]，但在对该网络的单一接入点处设置一个防火墙，如图8-35中所示，这使得管理和施加安全访问策略更为容易。
- 仅被批准的流量（由本地安全策略定义）允许通过。随着进入和离开机构网络的所有流量流经防火墙，该防火墙能够限制对授权流量的访问。
- 防火墙自身免于渗透。防火墙自身是一种与网络连接的设备，如果设计或安装得不适当，将能够危及安全，这样它仅提供了一种安全的假象。（这比根本没有防火墙更糟糕！）

Cisco和Check Point是当今两个领先的防火墙厂商。读者也可以很容易地从Linux套件使用iptables（通常与Linux装载在一起的公共域软件）产生一个防火墙（分组过滤器）。

防火墙能够分为3类：**传统的分组过滤器**（traditional packet filter）、**状态过滤器**（stateful filter）和**应用程序级网关**（application-level gateway）。在下面的两小节中，我们将依次学习它们。

1. 传统的分组过滤器

如图8-35所示，一个机构通常都有一个将其内部网络与其ISP相连的网关路由器，并因此与更大的公共因特网相连。所有离开和进入内部网络的流量都要经过这个路由器，而这个路由器正是出现**分组过滤**（packet filtering）的地方。分组过滤器逐个地检查每个数据报，然后基于管理员规定的规则，以决定是丢弃该数据报还是允许该数据报通过。过滤决定通常基于下列因素：

- IP源或目的地址。
- IP数据报协议字段中的类型：TCP、UDP、ICMP、OSPF，等等。

- TCP或UDP的源和目的端口。
- TCP标志比特：SYN、ACK，等等。
- ICMP报文类型。
- 数据报离开和进入网络的不同规则。
- 对不同路由器接口的不同规则。

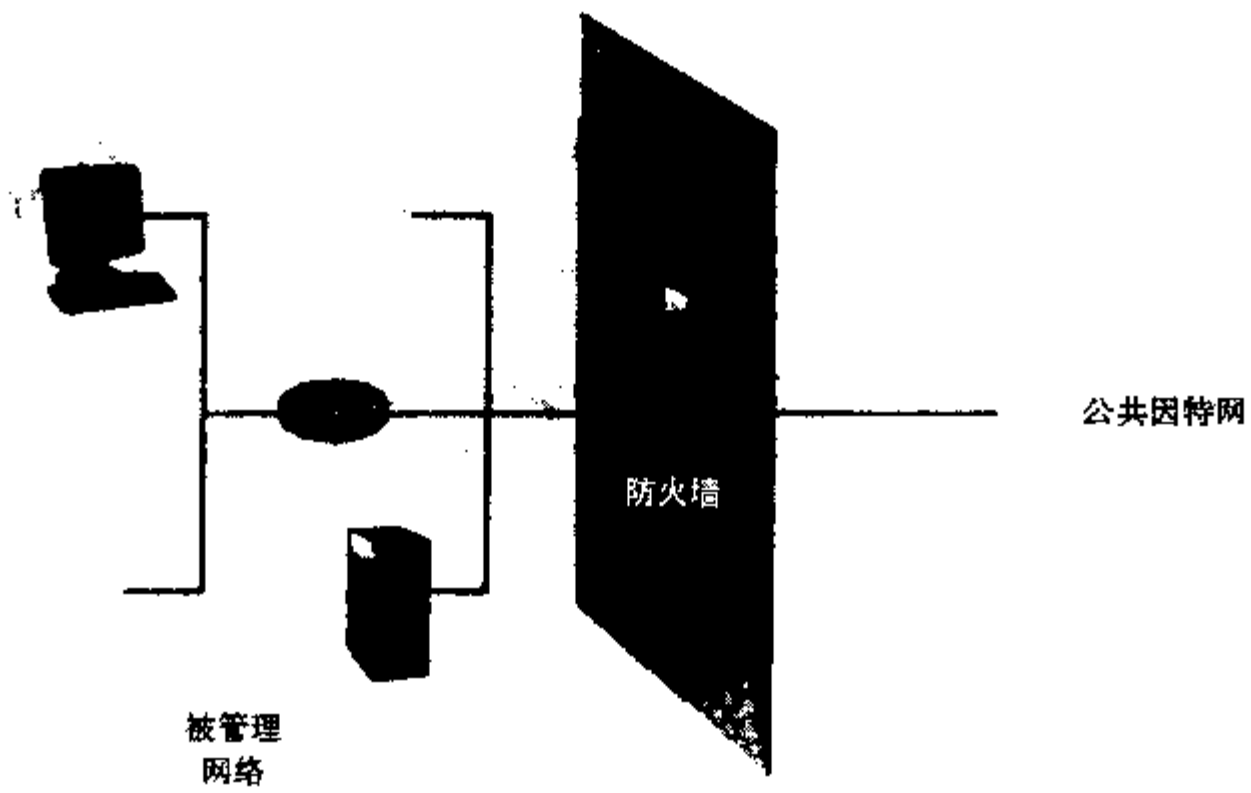


图8-35 在被管理网络和外部世界之间放置防火墙

网络管理员基于机构的策略配置防火墙。该策略可以考虑用户具有的产生分组的性质和带宽使用以及一个机构关注的安全性。表8-4列出了一个机构可能具有的若干可能的策略，以及它们将用一个分组过滤器来处理分组的方法。例如，如果该机构除了那些访问它的公共Web服务器外不希望任何人TCP连接的话，它能够阻挡所有的人TCP SYN报文段，但具有目的地端口80的TCP SYN报文段除外，并且该目的地IP地址对应于该Web服务器。如果该机构不希望它的用户独占了具有因特网无线电应用的访问带宽，它能够阻挡所有非关键性UDP流量（因为因特网无线电经常是通过UDP发送的）。如果该机构不希望它的内部网络被外部绘制结构图（被跟踪路由），它能够阻挡所有ICMP TTL过期的报文离开该机构的网络。

表8-4 对于某机构网络130.207/16，且Web服务器在130.207.244.203，对应的过滤规则和策略

策 略	防火墙设置
无外部Web访问	丢弃所有到任何IP地址端口80的出分组
无入TCP连接，但只访问机构公共Web服务器的那些分组除外	丢弃所有到任何IP的入TCP SYN分组，但到130.207.244.203端口80的除外
防止Web无线电占据可用带宽	丢弃所有入UDP分组，但DNS分组除外
防止你的网络被用于一个smurf DoS攻击	丢弃所有去往某“广播”地址（如130.207.255.255）的ICMP ping分组
防止你的网络被跟踪路由	丢弃所有ICMP TTL过期的出流量

过滤策略能够基于地址和端口号的结合。例如，一台过滤路由器能够转发所有Telnet数据

报（那些具有端口号23的数据报），但那些去往和来自一个包括某些特定IP地址的除外。这些策略允许Telnet连接到（和来自）允许列表中的主机。不幸的是，基于外部地址的策略没有对它们的源地址被哄骗的数据报提供保护。

也可根据TCP ACK比特是否设置来进行过滤。如果一个机构要使内部客户机连接到外部服务器，却要防止外部客户机连接到内部服务器，这个方法很有效。3.5节讲过在每个TCP连接中的第一个报文段的ACK比特都设为0，而连接中的所有其他报文段的ACK比特都设为1。因此，如果一个机构要阻止外部客户机发起到内部服务器的连接，就只需直接过滤进入的所有ACK比特设为0的报文段。这个策略终止了所有从外部发起的所有TCP连接，但是允许内部发起TCP连接。

路由器中使用访问控制列表实现防火墙规则，每个路由器接口具有它自己的列表。表8-5中显示了对于某机构222.22/16的访问控制列表的例子。该访问控制列表用于将路由器与机构外部ISP连接的接口。这些规则被应用到通过该接口自上而下传递的每个数据报。前两条规则一起允许内部用户在Web上冲浪：第一条规则允许任何具有目的端口80的TCP分组离开本机构的网络，第二条规则允许任何具有源端口80且ACK比特设为0进入该机构的网络。注意到如果一个外部源尝试试图与一台内部主机创建一条TCP连接，该连接将被阻挡，即使该源或目的端口为80。第二个两条规则一起允许DNS分组进入和离开该机构的网络。总而言之，这种限制性相当强的访问控制列表阻挡所有流量，但由该机构内发起的Web流量和DNS流量除外。[CERT Filtering 2007]提供了一张推荐的端口/协议分组过滤的列表，以避免在现有网络应用中的一些周知的安全性漏洞。

表8-5 对于路由器接口的一张访问控制列表

动作	源地址	目的地址	协议	源端口	目的端口	标志比特
允许	222.22/16	222.22/16的外部	TCP	>1023	80	任意
允许	222.22/16的外部	222.22/16	TCP	80	>1023	ACK
允许	222.22/16	222.22/16的外部	UDP	>1023	53	—
允许	222.22/16的外部	222.22/16	UDP	53	>1023	—
拒绝	全部	全部	全部	全部	全部	全部

2. 状态分组过滤器

在传统的分组过滤器中，根据每个分组独立作出过滤决定。状态过滤器实际跟踪TCP连接，并使用这种知识作出过滤决定。

为了理解状态过滤器，我们来重新审视在表8-5中的访问控制列表。尽管限制性相当强，表8-5中的访问控制列表仍然允许来自外部的ACK=1且源端口为80的任何分组到达，通过该过滤器。这样的分组能够由试图用异常分组来崩溃内部系统、执行拒绝服务攻击或绘制内部网络的攻击者使用。幼稚的解决方案也是阻挡TCP ACK分组，但是这样的方法将妨碍机构内部的用户在Web上冲浪。

表8-6 状态过滤器的连接表

源地址	目的地址	源端口	目的端口
222.22.1.7	37.96.87.123	12699	80
222.22.93.2	199.1.205.23	37654	80
222.22.65.143	203.77.240.43	48712	80

状态过滤器通过一张连接表跟踪所有进行中的TCP连接。这是可行的，因为防火墙能够通过观察三次握手（SYN、SYNACK和ACK）来观察一条新连接的开始；当它看到该连接的一个FIN分组时，它能够观察该连接的结束。当防火墙对某连接经数秒（如60秒）还没有看到任何活动性，它也能够（保守地）假设该连接结束了。某防火墙的一张连接表例子显示在表8-6中。这张连接表指出了有3条进行中的TCP连接，所有的连接都是从该机构内部发起的。此外，该状态过滤器在它的访问控制列表中包括了“核对连接”栏，如表8-7中所示。注意到表8-7与表8-5中的访问控制列表相同，只是现在它指示连接应当用两条规则来核对。

表8-7 用于状态过滤器的访问控制列表

动作	源地址	目的地址	协议	源端口	目的端口	标志比特	核对连接
允许	222.22/16	222.22/16的外部	TCP	>1023	80	任意	
允许	222.22/16的外部	222.22/16	TCP	80	>1023	ACK	X
允许	222.22/16	222.22/16的外部	UDP	>1023	53	—	
允许	222.22/16的外部	222.22/16	UDP	53	>1023	—	X
拒绝	全部	全部	全部	全部	全部	全部	

我们看一些例子来了解连接表和扩展访问控制表是如何联手工作的。假设攻击者通过发送具有TCP源端口80和ACK标志置0的一个数据报，试图向机构网络中发送一个异常分组。进一步假设该分组具有源端口号12543和源IP地址150.23.23.155。当这个分组到防火墙时，该防火墙核对表8-7中的访问控制列表，该表指出了在允许该分组进入机构网络之前，该连接表也必须被核对。该防火墙适时地核对了连接表，发现这个分组不是某进行中的TCP连接的一部分，从而拒绝了该分组。举第二个例子，假设一个内部的用户要在外部Web站点冲浪。因为该用户首先发送了一个TCP SYN报文段，该用户的TCP连接在该连接表中有了记录。当Web服务器发送返回分组（ACK比特进行了必要的设置），该防火墙核对了该表并了解一条对应的连接在进行中。该防火墙因此将让这些分组通过，从而不会干扰内部用户的Web冲浪活动。

3. 应用程序网关

在上面的例子中，我们已经看到了分组级过滤使得一个机构可以根据IP的内容和TCP/UDP首部（包括IP地址、端口号和确认比特）执行粗粒度过滤。但是如果一个机构仅为内部用户的一个受限集合提供Telnet服务（与IP地址情况正相反）该怎样做呢？如果该机构要这些特权用户在允许创建向外部的Telnet会话之前首先鉴别他们自己该怎样做呢？这些任务都超出了传统过滤器和状态过滤器的能力。实际上，有关内部用户的身份信息是应用层数据，并不包括在IP/TCP/UDP首部中。

为了得到更高水平的安全性，防火墙必须把分组过滤器和应用程序网关结合起来。应用程序网关基于应用数据来看待IP/TCP/UDP报头，并作策略决定。一个应用程序网关（application gateway）是一个应用程序特定的服务器，所有应用程序数据（入和出的）都必须通过应用程序网关。多个应用程序网关可以在同一主机上运行，但是每一个网关都是具有其单独进程的单独服务器。

为了更深入地了解应用程序网关，我们来设计一个防火墙，它只允许内部客户机的受限集合向外Telnet，不允许任何外部客户机向内Telnet。这一策略可将分组过滤（在一台路由器上）和一个Telnet应用程序网关结合来实现，如图8-36所示。该路由器的过滤器配置为阻塞所有Telnet连接，但从该应用程序网关IP地址发起的连接除外。这样的过滤器配置迫使所有向外

的Telnet连接都通过应用程序网关。现在考虑一个要向外部Telnet的内部用户。这个用户必须首先和应用程序网关建立一个Telnet会话。该网关上一直运行应用程序，网关监听进入的Telnet会话，提示用户输入用户ID和口令。当用户提供这些信息时，应用程序网关检查这个用户是否有权向外Telnet。如果没有，网关则中断这个内部用户向该网关发起的Telnet连接。如果该用户已被允许，则这个网关会：①提示用户输入它所要连接的外部主机的主机名，②在这个网关和某外部主机之间建立一个Telnet会话，③中继从这个用户到达的所有数据到外部主机，并且把来自外部主机的所有数据都中继给这个用户。所以该Telnet应用程序网关不仅执行用户授权而且同时充当一台Telnet服务器和一台Telnet客户机，在这个用户和这个远程Telnet服务器之间中继信息。注意到过滤器因为这个网关发起向外部的Telnet连接，将允许执行步骤2。

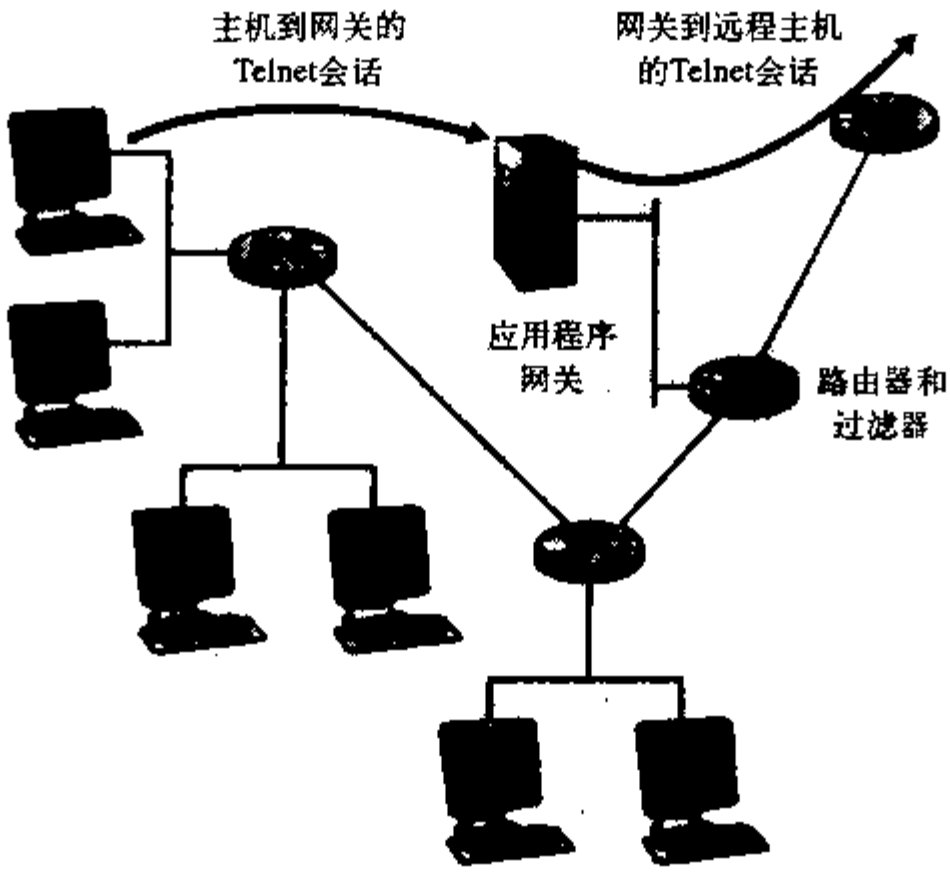


图8-36 由应用程序网关和过滤器组成的防火墙

内部网通常有多个应用程序网关，例如Telnet、HTTP、FTP和电子邮件网关。实际上，一个机构的邮件服务器（见2.4节）和Web高速缓存就是应用程序网关。

应用程序网关也有其缺陷。首先，每一个应用程序都需要一个不同的应用程序网关。第二，还要以性能的损失为代价，因为所有数据都由网关转发。当多个用户或应用程序使用同一个网关计算机时这个问题尤其严重。最后，当用户发起一个请求时，客户机软件必须知道如何联系这个网关，并且必须告诉应用程序网关连接到哪个外部服务器。

8.9.2 入侵检测系统

我们刚刚看到了当决定让哪个分组通过防火墙时，分组过滤器（传统的和状态的）检查IP、TCP、UDP和ICMP首部字段。然而，为了检测多种攻击类型，我们需要执行**深度分组检查**（deep packet inspection），即查看首部字段以外部分，查看分组携带的实际应用数据。如我们在8.9.1节所见，应用程序网关经常做深度分组检查。而一个应用程序网关仅对一种特定的应用程序执行这种检查。

显然，存在适用于另一种设备的环境，即一种不仅能够检查所有通过它传递的分组的首

部（类似于分组过滤器），而且能执行深度分组检查（与分组过滤器不同）。当这样的设备观察到一个可疑的分组时，或一系列可疑的分组时，它能够防止这些分组进入该机构的网络。或者因为仅仅是觉得该活动可疑，该设备能够让这些分组通过，而向网络管理员发出告警，网络管理员然后密切关注该流量并采取适当的行动。能够观察到潜在恶意流量并产生告警的设备称为**入侵检测系统**（Intrusion Detection System, IDS）。滤除可疑流量的设备称为**入侵防止系统**（Intrusion Prevention System, IPS）。在本节中我们一起研究IDS和IPS这两种系统，因为这些系统的最有意思的技术方面是它们如何检测可疑流量（而不是它们是发送告警还是丢弃分组）。我们因此将IDS系统和IPS系统统称为IDS系统。

IDS能够用于检测范围广泛的攻击，包括网络映射（例如使用nmap进行分析）、端口扫描、TCP栈扫描、DoS带宽洪泛攻击、蠕虫和病毒、操作系统脆弱性攻击和应用程序脆弱性攻击（参见1.6节有关网络攻击的概述内容）。目前，数以千计的机构应用了IDS系统。在这些部署的系统中有许多是专用的，如Cisco、Check Point和其他安全装备厂商在市场上销售的系统。但是许多部署的IDS系统是公共域系统，如极为流行的Snort IDS系统（我们将简要讨论）。

一个机构可能在它的机构网络中部署一个或多个IDS传感器。图8-37显示了一个具有3个IDS传感器的机构。当部署了多个传感器时，它们通常一起工作，向一个中心IDS处理器发送有关可疑流量活动的信息，中心处理器收集并综合这些信息，当认为适合时向网络管理员发送告警。在图8-37中，该机构将其网络划分为两个区域：一个高度安全区域，由分组过滤器和应用程序网关保护，并且由IDS系统监视；一个较低安全区域（称之为**非军事区**(demilitarized zone, DMZ)），该区域仅由分组过滤器保护，但也由IDS系统监视。注意到DMZ包括了该机构的需要与外部通信的服务器，如它的公共Web服务器和它的权威DNS服务器。

此时你也许想知道，为什么使用多个IDS传感器？为什么在图8-37中不只是在分组过滤器后面放置一个IDS传感器（或者甚至与分组过滤器综合）？我们将很快看到，IDS不仅需要做深度分组检查，而且必须要将每个过往的分组与数以万计的“特征（signature）”进行比较；这可能导致极大的处理量，特别是如果机构从因特网接收每秒数十亿比特的流量时更是如此。将IDS传感器进一步向下游放置，每个传感器仅看到该机构流量的一部分，这样更容易进行维护。无论如何，目前有高性能IDS和IPS系统可供使用，许多机构实际上能够在靠近其接入点附近只使用一个传感器。

IDS系统大致可分类为**基于特征的系统**（signature-based system）或**基于异常的系统**(anomaly-based system)。一个基于特征的IDS维护了一个存有广泛攻击特征的数据库。每个特征是一个与入侵活动相关联的规则集。一个特征可能只是单个分组的特性列表（例如源和目的端口号、协议类型和在分组有效载荷中的特定比特串），或者可能与一系列分组有关。这些特征通常是由研究了已知攻击、技艺熟练的网络安全工程师生成。一个机构的网络管理员能够定制这些特征或者将其加进数据库中。

运行时，基于特征的IDS嗅探通过的每个分组，并与数据库中的特征进行比较。如果分组（或一系列分组）与数据库中的特征匹配，IDS产生告警。告警可以用邮件的形式发给网络管理员，也可以发给网络管理系统，或者只是记录下来以备将来检查。

虽然广泛地部署了基于特征的IDS系统，但仍有一些限制。更重要的是，它们要求根据攻击的以前知识来产生一个准确的特征。换言之，基于特征的IDS对将被记录的新攻击完全缺乏判断力。另一个缺点是即使与一个特征匹配，它也可能不是一个攻击，因此产生了一个虚假告警。最后，因为每个分组必须与范围广泛的特征集合相比较，IDS可能处于处理过载状态并

因此难以检测出许多恶意分组。

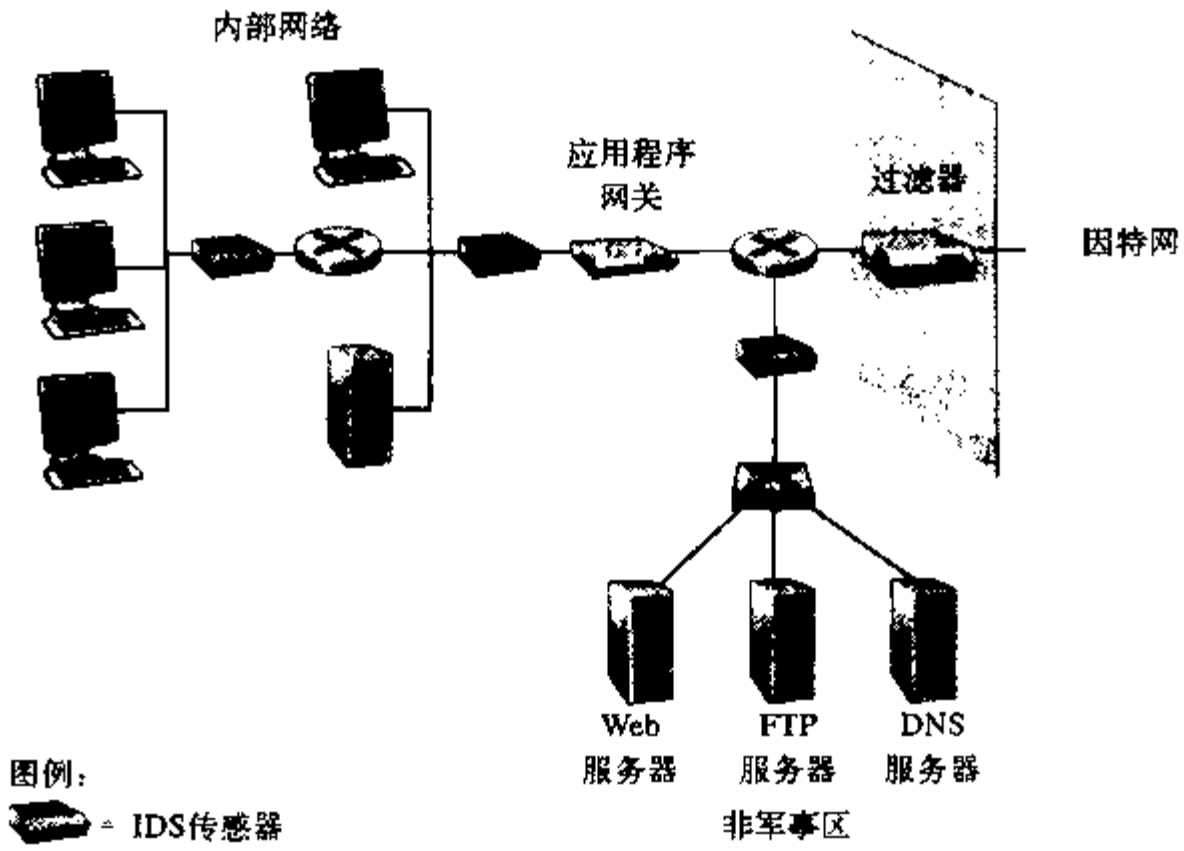


图8-37 部署过滤器、应用程序网关和IDS传感器的机构

当基于异常的IDS观察正常运行的流量时，它会生成一个流量概况文件。然后，它寻找统计上不寻常的分组流，例如，ICMP分组过度的百分比，或端口扫描和ping扫描导致的突然的指数增长。有关基于异常的IDS系统的最大特点是它们不依赖有关现有攻击的以前知识。另一方面，区分正常流量和统计异常流量是一个极具挑战性的问题。至今为止，大多数部署的IDS主要是基于特征的，尽管某些IDS包括了某些基于异常的特性。

Snort

Snort是一种公共域开放源码的IDS，现在部署了几十万份[Snort 2007；Koziol 2003]。它能够运行在Linux、UNIX和Windows平台上。它使用了通用的嗅探接口libpcap，Ethereal和许多其他分组嗅探器也使用libpcap。它能够轻松地处理100 Mbps的流量；对于安装在千兆比特/秒流量速率下工作，需要多个Snort传感器。

为了对Snort有一些认识，我们来看一个Snort特征的例子：

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any

(msg:"ICMP PING NMAP"; dsize: 0; itype: 8;)
```

这个特征与从外部（$EXTERNAL_NET）进入机构网络（$HOME_NET）的任何ICMP分组相匹配，其类型是8（ICMP ping）并且具有空负载（dsize=0）。因为nmap（参见1.6节）用这些特定的特征产生这些ping分组，设计出该特征，以检测nmap的ping扫描。当某分组匹配该特征时，Snort产生一个包括“ICMP PING NAMP”报文的告警。

也许关于Snort印象最为深刻的是，有人数众多的用户和安全专家在维护它的特征数据库。通常在一个新攻击出现的几个小时内，Snort团体编写并发布一个攻击特征，然后它就能被分布在全世界的数十万Snort部署者下载。况且，使用Snort特征的语法，网络管理员能够根据他们自己的机构需求，或通过修改现有的特征或通过创建全新的特征来裁剪某个特征。

8.10 小结

在本章中，我们研究了能够用于安全通信的各种机制（这些机制能被用于秘密情人Bob和Alice之间）。我们看到Bob和Alice对下列因素感兴趣：机密性（只有他们才能理解传输报文的内容）、端点鉴别（他们确信他们正在与对方交谈）和报文完整性（他们确信在传输过程中他们的报文未被篡改）。当然，安全通信的需求并不限于秘密情人。实际上，我们在8.5～8.8节中看到，在网络体系结构中的各个层次中都能够使用安全性，使之免受采用各种各样攻击手段的“坏家伙”的侵扰。

本章第一部分给出了安全通信依据的各种原理。8.2节讲述了对数据进行编码和解码的密码技术，包括对称密钥密码和公钥密码。我们分析了DES和RSA作为具体案例研究，它们是今天网络中使用的两种重要的密码技术。

在8.3节中，我们研究了提供报文完整性的两种方法：报文鉴别码（MAC）和数字签名。这两种方法有一些共同之处。它们都使用了密码散列函数，这两种技术都使我们能够验证报文的源以及报文自身的完整性。一个重要的差异是MAC不依赖于加密，而数字签名要求公钥基础设施。如我们在8.5～8.8节所见，这两种技术在实际中都得到了广泛应用。此外，数字签名用于生成数字证书，数字证书对于证实公钥的合法性是重要的。

在8.4节中，我们转而考虑端点鉴别的问题，研制了一系列越来越复杂的鉴别协议，以确保通信的对方确实就是所宣称的那个人，并且“活着”。我们发现对称密钥密码和公钥密码不仅在伪装数据（加密/解密）方面，而且在执行鉴别方面都起到了重要作用。

在8.5～8.8节中，我们研究了几种在实践中得到广泛使用的安全网络协议。我们看到了对称密钥密码在PGP、SSL、IPsec和无线安全性中的核心地位。我们看到了公钥密码对PGP和SSL是至关重要的。我们看到PGP使用数字签名而SSL和IPsec使用MAC来保证报文完整性。在理解了密码学的基本原理并学习了这些原理如何实际应用之后，你现在已经具有了设计自己的安全网络协议的能力了！

利用8.2～8.4节所包含的技术，Bob和Alice就能够安全通信了。而机密性仅是整个网络安全的一小部分。现在网络安全的焦点越来越多的是关注网络基础设施的安全性，以防止“坏家伙”潜在的猛烈攻击。在本章的后面部分，我们因此学习了防火墙和IDS系统，它们检查进入和离开机构网络的分组。

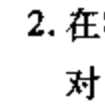
本章涉及许多基础性问题，同时关注现代网络安全中最为重要的主题。希望深入钻研的读者最好研究本章中引用的文献。特别是，我们推荐对于攻击和运行安全性有兴趣的读者阅读[Skoudis 2006]，对于密码学及其如何应用于网络安全有兴趣的读者阅读[Kaufman 1995]，[Rescorla 2001]是一本有深度且可读性强的有关SSL处理的书，[Edney 2001]透彻地讨论802.11安全性，且对WEP及其缺陷做了深入研究。读者们也可以参考[Ross 2007]，以得到关于网络安全性的内容广泛的PowerPoint幻灯片集。

课后习题和问题

复习题

1. 报文机密性和报文完整性之间的区别是什么？能否考虑其中一个问题而忽略另一个问题？论证你的答案。

2. 主动入侵者和被动入侵者之间的区别是什么?
3. 对称密钥系统和公钥系统的重要区别是什么?
4. 假定一个入侵者拥有某报文的加密版本和解密版本。这个入侵者能够发起已知密文攻击、已知明文攻击和选择明文攻击中的哪种攻击呢?
5. 假定N个人中的每个人都和其他$N-1$个人使用对称密钥密码通信。任两人（i和j）之间的所有通信对该N个人的组中的所有其他人都是可见的，且该组中的其他人都不应当能解密这个通信。则这个系统总共需要多少个密钥?如果假定使用公钥密码，此时需要多少个密钥?
6. 在某端点鉴别协议中，使用不重数的目的是什么?
7. 一个不重数被描述为在生存期中只使用一次的值，这意味着什么?指谁的生存期?
8. 简述中间人攻击。使用对称密钥时会发生中间人攻击吗?
9. 一个已签名的文档是可鉴别的和不可伪造的，其含义是什么?
10. 散列以何种方式提供比检验和（如互联网检验和）更好的报文完整性检验?
11. 公钥加密的报文散列以何种方式提供比使用公钥加密报文更好的数字签名?
12. 你能够“解密”某报文的散列，以得到初始报文吗?解释你的答案。
13. 何谓认证中心?
14. 概述在IPsec中由鉴别首部（AH）协议和封装安全性载荷（ESP）协议所提供的服务的主要区别。



习题

1. 使用图8-3中的单码替换密码，加密报文“This is an easy problem”；并解密报文“rmij 'u uamu xyj”。
2. 在8.2.1节的例子中，显示Trudy使用了已知明文攻击，其中她知道了7个字母的（密文，明文）转换对，减少了将被检查的大约10^9数量级个可能替换的数量。请说明之。
3. 考虑图8-4所示的多码替换密码系统。利用报文“The quick brown fox jumps over the lazy dog”得到的明文编码，选择明文攻击足以破解所有报文吗?为什么?
4. 使用RSA，选择$p = 3$和$q = 11$，加密短语“hello”。对已加密报文应用解密算法恢复出原报文。
5. 考虑我们的鉴别协议ap4.0，其中Alice向Bob鉴别她自己，并且看起来工作正常（即我们没有发现其中有缺陷）。现在假定在Alice向Bob鉴别她自己的同时，Bob必须向Alice鉴别他自己。给出一个情况，此时Trudy假装是Alice，向Bob鉴别证明她自己是Alice。（提示：考虑协议ap4.0的运行顺序，鉴别过程可由Trudy或Bob发起，能够任意地交织在一起。特别注意Bob和Alice将使用一个不重数这样一个事实，如果不小心的话，能够恶意地使用了相同的不重数。）
6. 在图8-21所示的中间人攻击中，Alice没有鉴别Bob。如果Alice要求Bob使用协议ap5.0鉴别他自己的话，是否可避免中间人攻击?说明你的理由。
7. 因特网BGP路由协议使用MAC而不是公钥密码对BGP报文签名。你认为此处为何选择MAC而不选择公钥密钥?
8. 计算一个不同于图8-8中的两个报文的第三个报文，使该报文具有与图8-8中的报文相同的检验和。
9. 假定Alice要与采用对称密钥密码体制的Bob使用一个会话密钥K_S通信。在8.2节中，我们知道了如何使用公钥密码从Alice向Bob分发该会话密钥。在本习题中，我们研究不使用公钥密码而使用一个密钥分发中心（KDC）分发会话密钥的方法。KDC是一个与每个注册用户共享唯一秘密对称密钥的服务器。对于Alice和Bob而言，$K_{A\text{-}KDC}$和$K_{B\text{-}KDC}$表示了这些密钥。设计一个使用KDC向Alice和Bob分发K_S的方案。你的方案应当使用三种报文来分发会话密钥：一种从Alice到KDC的报文；一种从KDC到Alice的报文；最后是一种从Alice到Bob的报文。

10. 图8-24显示了Alice必须执行的操作，以提供机密性、鉴别和完整性。图示Bob接收来自Alice的分组时必须执行的对应操作。
11. 是非判别题：
    a. 考虑从主机A到主机B使用IPsec发送分组流。通常，要为该流中的每个分组创建一个新的SA。
    b. 假定TCP使用AH协议在IPsec之上运行。如果TCP重传相同的分组，则这两个分组在该AH首部将具有相同的序号。
    c. 假定Alice和Bob正在通过SSL会话通信。假定某个没有任何共享密钥的攻击者，在分组流中插入一个伪造的TCP报文段（该报文段具有正确的TCP检验和、序号，以及正确的IP地址和端口号）。在接收侧的SSL将接受该伪造分组，并向接收应用程序传递有效载荷。
    d. 假定certifier.com为foo.com生成了一份证书。通常该整个证书将用certifier.com的公钥加密。
    e. 前面讲过密码散列用于分发OSPF报文，如在课堂上讨论的那样。一台路由器为了验证报文的完整性，它必须与生成该报文的路由器共享一个秘密密钥。
    f. 考虑采用密码块链接加密一个大文件的情况。使用这种机制，源向接收方以明文发送初始向量（IV）和秘密密钥。
12. 考虑下列的伪WEP协议。密钥是4比特，IV是2比特。当产生密钥流时，IV被附加到密钥的后面。假定共享的密钥是1010。四个可能的输入的密钥流如下：

    101000: 0010101101010101001011010100100...

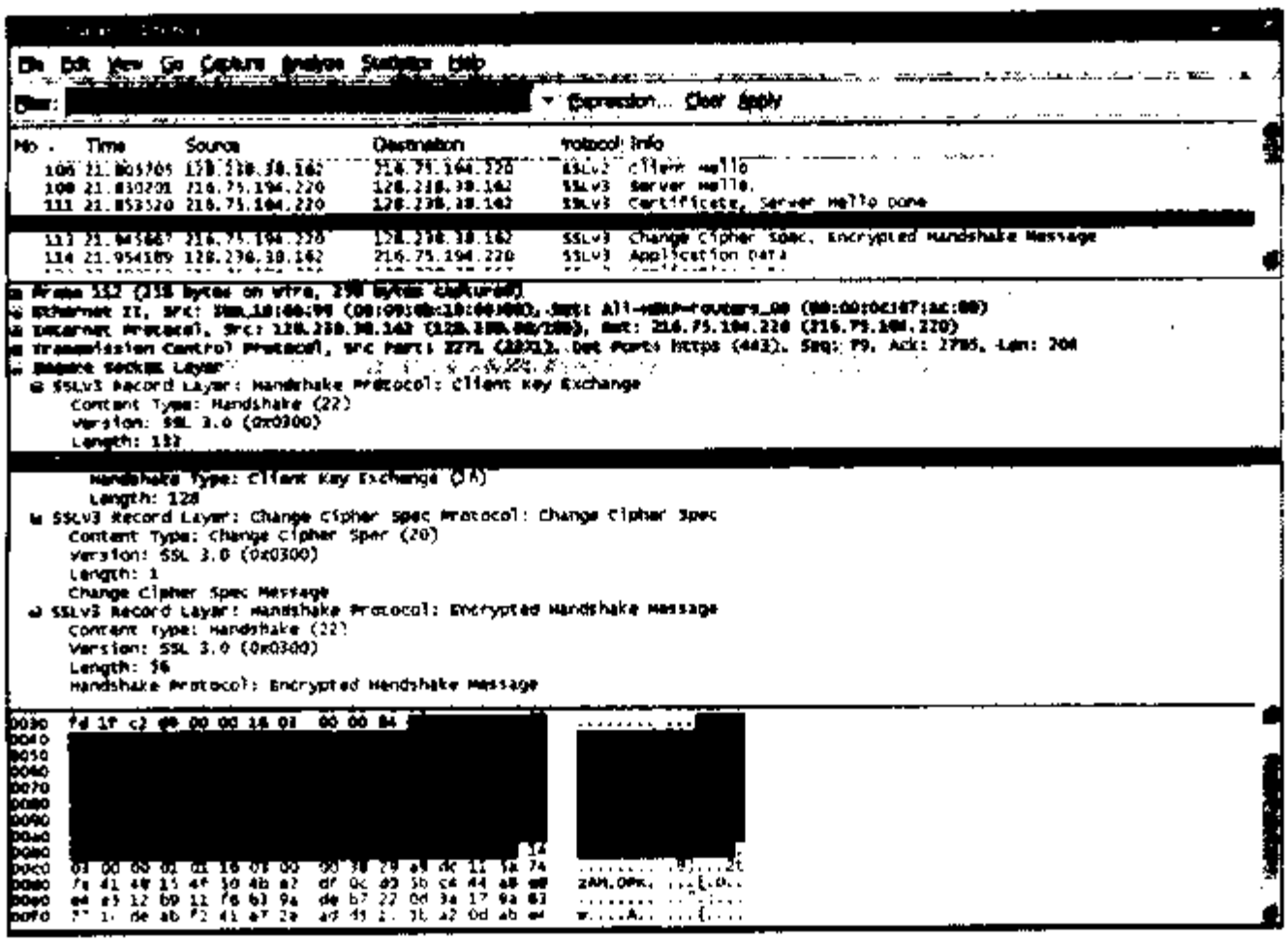
    101001: 1010011011001010110100100101101...

    101010: 0001101000111100010100101001111...

    101011: 1111101010000000101010100010111...

    假定所有报文都是8比特长。假定ICV（完整性检查）是4比特长，并且通过用数据的后4比特异或数据的前4比特来计算。假定该伪WEP分组由3个字段组成：首先是IV字段，然后是报文字段，最后是ICV字段，这些字段中的某些被加密。
    a. 我们希望使用IV=11和WEP发送报文m=10100000。在这3个WEP字段中将有什么样的值？
    b. 说明当接收方解密该WEP分组时，它恢复的报文和ICV。
    c. 假定Trudy截获了一个WEP分组（并不必要使用IV=11）并在向接收方转发前修改该分组。假定Trudy翻转了第一个ICV比特。假定Trudy并不知道用于任何IV的密钥流，则Trudy也必须翻转哪些其他比特，以使得接收到的分组通过ICV检查？
    d. 评价你的答案：修改部分a中WEP分组中的比特，解密所生成的分组，并验证完整性检查。
13. 对于尽可能进行限制但能实现下列功能的一台有状态防火墙，提供一张过滤器表和一张连接表：
    a. 允许所有的内部用户与外部用户创建Telnet会话。
    b. 允许外部用户冲浪公司位于222.22.0.12的Web站点。
    c. 否则阻挡所有入流量和出流量。
14. 考虑下面图中所示对于某SSL会话的一部分的Ethereal的输出。
    a. 以太网分组112是由客户机还是由服务器发送？
    b. 服务器的IP地址和端口号是什么？
    c. 假定没有丢包和重传，由客户机发送的下一个TCP报文段的序号将是什么？
    d. 以太网分组112包含了多少个SSL记录？
    e. 分组112包含了一个主密钥或者一个加密的主密钥，或者两者都不是？
    f. 假定握手类型字段是1字节并且每个长度字段是3字节，主密钥（或加密的主密钥）的第一个和最后一个字节的值是什么？
    g. 客户机加密的握手报文考虑了多少SSL记录？

h. 服务器加密的握手报文考虑了多少SSL记录?



讨论题

1. 假定一个入侵者可以把DNS报文插入网络，也可以从网络删除DNS报文。描述入侵者可能引起麻烦的三种情况。
2. 尚无人正式证明3DES和RSA的安全性。在这种情况下，我们有什么证据认为它们确实是安全的?
3. 如果IPsec在网络层提供安全性，为什么在IP以上的层次还需要安全机制?
4. 浏览国际PGP主页（http://www.pgpi.org/）。允许你合法地下载何种版本的PGP（给定你所在的国家）?

Ethereal实验

在这个实验中（参见与本书配套的Web站点），我们研究安全套接字层（SSL）协议。8.6节讲过，使用SSL使得TCP连接更为安全，在因特网事务安全方面，SSL已经在实践中得到了广泛应用。在本实验中我们关注经TCP连接发送的SSL记录。我们将试图对每个记录定界和分类，目标是理解每个记录的工作原理和工作过程。我们研究各种SSL记录类型以及在SSL报文中的字段。我们通过分析你的主机与一台电子商务服务器之间发送的SSL记录的踪迹来实现这一切。

人物专访

Steven M. Bellovin在位于新泽西州Florham Park的AT&T实验研究所的网络服务研究实验室工作多年后，成为了哥伦比亚大学的教师。他的研究重点是网络和安全，以及将两者有机结合起来。1995年，因创立了Usenet，即第一个连接两个或多个计算机并允许用户共享信息和参与讨论的新闻组交换网络，而被授予Usenix终生成就奖。Steven也是国家工程学会的当选成员。他在哥伦比亚大学获得学士学位，在Chapel Hill的北卡罗来纳大学获得博士学位。

Steven M. Bellovin

• 什么原因使您决定专注于网络安全领域的研究?

听起来可能很奇怪，但是答案却很简单：只是因为感兴趣而已。我以前的背景是从事系统编程和系统管理，这很自然就发展到安全领域了。而且我一直对通信很感兴趣，这可以追溯到我在上大学时，那时我就兼职做系统编程方面的工作。

我在安全领域的工作继续下去主要受两个因素的激励：一个希望使计算机有用，这意味着它们的功能不会被攻击者破坏；另一个希望保护隐私。

• 当初您在研发Usenet时，您对它的愿望是什么？现在呢?

我们最初将它看作是一种能够在全国范围内讨论计算机科学和计算机编程的手段，考虑了许多事务管理和销售广告等许多本地使用的情况。事实上，我最初的预测是，每天从至多50到100个站点有一到两个报文。但是实际增长在于与人相关的主题方面，包括（但不限于）人与计算机的相互作用。这么多年来，我喜欢的新闻组有rec.woodworking以及sci.crypt。

在某种程度上，网络新闻已经被Web取代。如果现在要我再设计它的话，就会和那时的设计大不相同了。但是它仍然是大量对于某一主题感兴趣的读者进行沟通的一种极好手段，而不必依赖特定的Web站点。

• 是否有人给过您专业上的启示和灵感？以什么样的方式呢?

Fred Brooks教授对我的专业生涯影响重大。他是位于Chapel Hill的北卡罗来纳大学计算机科学系的创立者和原系主任，也是研发IBM S/360和OS/360团队的管理者，也是“The Mythical Man Mouth”（《人月神话》）的作者。最重要的是，他教给我们观察和折中，即如何在现实世界环境中观察问题（不论这个现实世界比理论上的要复杂多少倍），以及在设计一种解决方案时如何平衡竞争各方的利益。大部分计算机工作都是工程性的，正确折中的艺术能够满足许多相互矛盾的目标。

• 您对未来的网络和安全性的展望是什么?

到目前为止，我们所具有的安全性大多来自隔离。例如，防火墙的工作是通过切断某些机器和服务实现的。但是我们正处在增加连通性的时代，这使得隔离变得更为困难。更糟糕的是，我们的生产系统要求的远不止是分离的部件，而需要通过网络将它们互联起来。我们面临的最大挑战之一是使所有都安全。

• 您认为在安全性方面已经取得的最大进展是什么？未来我们还能有多大作为?

至少从科学上讲，我们知道了密码学的原理。这是非常有帮助的。但是多数安全问题的起因是其代码错误成堆，从而成为非常困难的问题。事实上，它是计算机科学中花费时间最长而没有解决的问题，并且我认为该问题仍会持续。挑战在于我们不得不使用不安全的组件来设计出安全工作的系统。我们面对硬件故障已经能够解决可靠性问题了；面对安全性问题，我们是否能够做到这一点呢?

• 对于从事因特网和网络安全的学生们，您有何忠告?

学习各种安全机制是件容易的事。学习如何“思维偏执”是困难的。你必须记住概率分布在下列场合并不适用，即攻击者能够和将能够发现不可能的情况。并且其细节情况不胜枚举。

第9章 网络管理

通过对本书前8章内容的学习，我们现在已经知道网络是由许多复杂的、交互的硬件和软件实体组成的，包括构成网络的物理部件的链路、交换机、路由器、主机和其他设备，也包括控制和协调这些设备的许多协议（以硬件和软件形式出现）。当一个机构将数以百计或数以千计的这些部件拼装在一起成为一个网络时，出现如下一些现象是不足为奇的：网络部件偶尔出现故障，网络元素配置错误，网络资源过度使用，网络部件完全“崩溃”（例如，电缆被切断，一听可乐翻倒在一台路由器上）。网络管理员的工作是保持该网络“启动和运行”，他们必须能够对这些问题做出反应，最好是避免它们的出现。由于成千上万的网络部件可能散布在广阔的区域中，网络运营中心（NOC）的网络管理员显然需要工具来监视、管理和控制网络。在本章中，我们将探讨网络管理员在网络管理任务中所使用的体系结构、协议和信息库。

9.1 什么是网络管理

在深入研究网络管理之前，我们首先来考虑“真实世界”的非网络环境中的几个说明性例子，在这个复杂系统中，管理人员要对该系统中交互的部件进行监测、管理和控制。发电厂（至少如电影《The China Syndrome》这样的流行媒体描述的那样）有控制室，其中有标度盘、测量仪和灯光用来监视远程的阀门、管子、容器和其他工厂设备的状态（温度、压力、流量）。这些设备允许操作员监测该工厂的许多部件，并当故障到来时可能向操作员告警（用周知的红色闪光警示灯）。工厂操作员采取措施来控制这些部件。类似地，飞机座舱的安排使飞行员能够监视和控制飞机的许多部件。在这两个例子中，“管理人员”监测远程设备并分析相关数据，以确保这些设备能够使用，并在所规定的限制中运行（例如，核电厂的核反应堆不会很快熔化，或者飞机的燃料不会用完）；根据系统中的变化或其环境进行调整，反应式地进行控制；以及主动式地管理系统（例如，通过检测趋势或异常行为，在出现严重问题之前采取行动）。与此类似，网络管理人员要主动监测、管理和控制他们所负责的系统。

在网络发展的初期，计算机网络还处于用于研究的初级阶段，而不是数千万人每日必用的重要基础设施，“网络管理”还是一个闻所未闻的超前概念。如果一个人遇到了网络问题，他可能多次运行ping程序对该问题的源进行定位，接着修改系统设置，重新启动硬件或软件，或者打电话让远程的同事来做这些事情。（一个非常值得阅读的材料是，在1980年10月27日ARPAnet第一次严重“崩溃”后的很长一段时间里，才有可供使用的网络管理工具，网络管理员努力工作以恢复崩溃的系统并试图理解这次崩溃的原因[RFC 789]）。随着公共因特网和专用内联网从小型网络成长为巨大的全球基础设施，更为系统地管理这些网络中的大量硬件和软件组件也就变得更为重要了。

为了激发读者学习网络管理的热情，我们将从一个简单的例子开始讨论。图9-1显示了一个小型网络，该网络由3台路由器和若干台主机及服务器组成。即使在这样一个简单的网络中，网络管理员要使用适当的网络管理工具的情况也有很多：

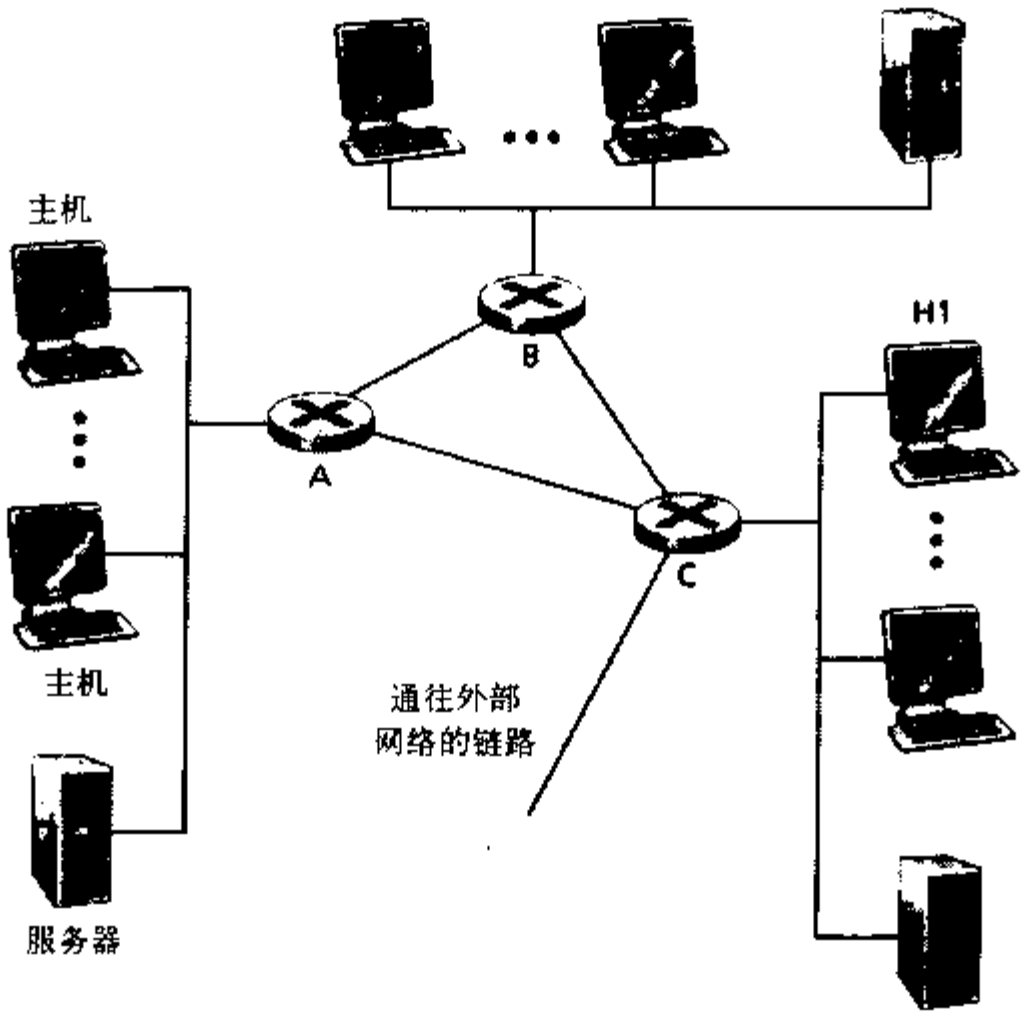


图9-1 一个简单的说明网络管理用途的例子

- 检测主机或路由器接口卡的故障。利用适当的网络管理工具，一个网络实体（如路由器A）可以向网络管理员报告它的一个接口卡已经失效。（这无疑比一个发怒的用户打电话向网络运营中心（NOC）抱怨网络连接已经中断要好得多！）主动监测和分析网络流量的网络管理员可以事先检测到该接口存在的问题，并在它发生故障之前换掉它，这样会给原来要发怒的用户留下深刻的印象。例如，如果管理员注意到由这块行将失效的接口所发送帧中的检验和差错增加，就可能发现接口卡存在的问题。
- 主机监测。在这种场合下，网络管理员可以进行周期性的检查，看是否所有的网络主机正在开机和运行。同样，网络管理员在用户报告之前提前对问题（如主机停机）做出响应，也会给网络用户留下深刻的印象。
- 监测流量以帮助部署资源。网络管理员可以通过监测源到目的地流量的模式和通告，在LAN网段间切换服务器，使得跨越多个LAN的流量大大减少。若不增加新设备成本而取得更好的网络性能，是件皆大欢喜的事情（对较高管理层而言尤其如此）。类似地，通过监测链路利用率，网络管理员可以确定某条LAN网段或通往外部的链路将要过载，因此应当提供更高带宽的链路（当然要增加成本）。网络管理员也可能希望在出现链路拥塞等级超过给定阈值的情况下，自动得到通知，使得在拥塞变得严重之前，配备更大带宽的链路。
- 检测选路表中的快速变化。路由颤动（route flapping），即选路表内容的经常变化，可能表明选路的不稳定或路由器配置不正确。那些不正确地配置了路由器的网络管理员，无疑希望自己在网络瘫痪之前发现这个错误。
- 服务等级协定的监测。随着**服务等级协定**（Service Level Agreement, SLA）的出现，对流量监测的关注在过去几年中大大增加了[Larsen 1997, Huston 1999a]。SLA定义了特定

的性能测度和网络提供商提供的可接受的相对于这些性能测度的性能等级。Verizon和Sprint是两个网络提供商，它们与其他ISP一样向其用户提供确保的SLA[Verizon 2007; Sprint 2007]。这些SLA包括服务可用性（断线率）、时延、吞吐量和断线通知要求。很显然，如果性能标准是网络提供商和其用户间服务协定的一部分的话，那么测量和管理性能对于网络管理员是十分重要的。

- 入侵检测。当网络流量来自或预定要流向一个可疑的源（例如，某特定的主机地址或端口号）时，网络管理员可能希望得到通知。类似地，网络管理员可能希望检测（并在许多情况下过滤）某些类型流量的存在（例如，源选路分组或指向给定主机的大量SYN分组），这被认为具有安全性攻击类型的特征，我们已在第8章中讨论过了。

国际标准化组织（ISO）已经建立了一个网络管理模型，该模型对于将上述各种情况放置于更为结构化的框架中是很有用的。该模型定义了网络管理的5个领域：

- 性能管理（performance management）。性能管理的目标是量化、测量、报告、分析和控制不同网络部件的性能（如利用率和吞吐量）。这些部件包括各个设备（如链路、路由器和主机）以及诸如通过某网络的路径那样的端到端抽象。我们很快将会看到，像简单网络管理协议（Simple Network Management Protocol, SNMP）[RFC 3410]这样的协议标准在因特网性能管理中起着中心作用。
- 故障管理（fault management）。故障管理的目标是记录、检测和响应网络中的故障情况。故障管理和性能管理之间的区别是相当模糊的。可以认为故障管理是对暂时性网络故障（例如，链路、主机或路由器硬件或软件的停转）的及时处理，而性能管理采取更为长期的观点，面对变化的流量要求和偶尔的网络设备故障，提供可接受的性能等级。与性能管理一样，SNMP协议在故障管理中起着中心作用。
- 配置管理（configuration management）。配置管理允许网络管理员跟踪被管理网络中的那些设备，跟踪这些设备的硬件和软件配置。有关基于IP网络的配置管理和要求的概述可参见[RFC 3139]。
- 账户管理（accounting management）。账户管理允许网络管理员定义、记录、控制用户和设备访问网络资源。限额使用、基于使用的收费和分配资源访问权限都属于账户管理的范围。
- 安全管理（security management）。安全管理的目标是根据某些定义明确的策略，控制对网络资源的访问。我们在8.3节中讨论的密钥分发中心是安全管理的一部分。使用防火墙监视和控制网络的外部接入点既是8.9节中讨论的主题，也是本章的另一个重要部分。

在本章中，我们将仅涉及网络管理的入门知识。我们有意识地专注有限的范围，讨论网络管理的基础设施，即总体体系结构、网络管理协议和信息库——网络管理员将通过信息库保持网络正常运行。我们将不涉及网络管理员的决策过程，网络管理员必须规划、分析和对传送到NOC的网络信息做出响应。在这个领域中，诸如故障标识和管理[Katzela 1995; Medhi 1997; Steinder 2002]、主动异常检测[Thottan 1998]、告警关联[Jakobson 1993]及更多的主题正在研究过程中。我们将不涉及更宽泛的服务管理主题[Saydam 1996; RFC 3052; AT&T SLM 2006]，如带宽、服务器容量和其他为满足企业特定任务的服务要求所需要的计算/通信资源。在服务管理这个领域中，如TMN[Glitho 1995; Sidor 1998]和TINA [Hamada 1997]这样的标准是更大、包容更多（因而将是庞大不灵活）的标准，用于处理大型问题。例如，TINA被描述为“涉及服务、资源和分布式处理环境部分管理的一系列公共目标、原则和概念”[Hamada

1997]。显然，这些主题中的每一个都足以单独写一本书，并使我们偏离计算机网络技术方面的讨论。因此，如前面所述，我们更合适的目标将是涉及重要的体系结构的“细节”。通过这些，网络管理员能使比特流动得更顺畅。

一个经常被问到的问题是：什么是网络管理？我们上面的讨论是从网络管理的需求和展示几个网络管理应用的角度进行的。在这一节中，我们将用一句话（虽然它相当冗长）概括网络管理的定义[Saydam 1996]：

> “网络管理包括了硬件、软件和人类元素的设置、综合和协调，以监视、测试、轮询、配置、分析、评价和控制网络和网元资源，用合理的成本满足实时性、运营性能和服务质量的要求。”

这句话虽绕口，但它是一个好而易于使用的定义。在后面的内容中，我们将为这个相当精干的网络管理定义增加一些内容。

9.2 网络管理的基础设施

我们在前面一节中看到，网络管理要求具有“监视、测试、轮询、配置……和控制”网络中的硬件和软件组件的能力。这些网络设备是分布式的，这将在最低程度上要求网络管理员能够从远程的一个实体收集数据（例如，为了监视目的），以及能够引起远程实体的改变（例如，控制它）。一个类比人类的例子将有助于我们理解网络管理所需要的基础设施。

假定你是一个在全世界具有分支机构的大组织的领导。你的工作是确保该组织的各部分运转正常。你将如何做到这一点？至少，你将周期性地从这些分支机构以报告的形式收集数据，并且定量地测算各种活动、生产率和预算。你偶尔会（不会是经常）被明确地告知这些分支机构中的某个机构存在一个问题；该分支机构经理是一个希望向公司高层爬的人（也许希望得到你的位子），他也许主动地向你提供报告，指出在其领导的分支机构中工作进行得十分顺利。你会详细地审查收到的报告，希望看到各地运行正常，但无疑发现了需引起你重视的问题。你可能与存在问题的分支机构发起一对一对话，为了理解该问题收集更多的数据，然后向该分支机构的经理传达一个行政命令（“纠正这件事”）。

隐含在这个非常普通的人类活动背后的是一个控制组织的基础设施，即老板（你）、被控制的远程站点（分支机构）、你的远程代理（分支机构的经理）、通信协议（用于传输标准的报告和数据或一对一对话）和数据（报告内容以及各种活动、生产率和预算的定量估算）。人类组织管理中的这些部件，每个都在网络管理中有对应的东西。

网络管理系统的体系结构在概念上等同于这个简单的人类组织的类比例子。网络管理领域对于网络管理体系结构的各种组件都有其自己的特定术语，因此这里采用这些术语。如图9-2所示，网络管理体系结构有三个重要组件：一个管理实体（上述类比中的老板，即你）、多个被管设备（分支机构）和一个网络管理协议。

管理实体（managing entity）是一个应用程序，运行在网络运营中心（NOC）的集中式网络管理工作站上，通常人们要对它不断地重复发布指令。管理实体是执行网络管理活动的所在地；它控制网络管理信息的收集、处理、分析和/或显示。正是在这里，人类网络管理员与网络设备打交道，发起控制网络行为的动作。

被管设备（managed device）是网络装备的一部分（包括它的软件），它位于被管理的网络中，相当于我们上述类比中的分支机构。一个被管设备可能是主机、路由器、交换机、集线器、打印机或调制解调器。在一个被管设备中，有几个**被管对象**（managed object）。这些

被管对象是被管设备中硬件的某些有效部分（如一块网络接口卡）以及某些硬件和软件的配置参数集合（如RIP这样的域内选路协议）。在上述类比中，被管对象可能是分支中的一个部门。这些被管对象具有与它们相联系的部分信息，这些信息收集在**管理信息库**（Management Information Base, MIB）中，我们将看到，这部分信息的值可供管理实体利用（并且在许多情况下能够被它设置）。在前面的类比中，MIB对应于分支机构和中心办事处之间交换的定量数据（活动、生产率和预算的度量，并且MIB能够被管理实体设置）。我们将在9.3节中详细学习MIB。最后，在每个被管设备中还驻留着**网络管理代理**（network management agent），它是运行在被管设备中与管理实体通信的进程，在管理实体的命令和控制之下，在被管设备上采取本地动作。网络管理代理是上述类比中的分支机构经理。

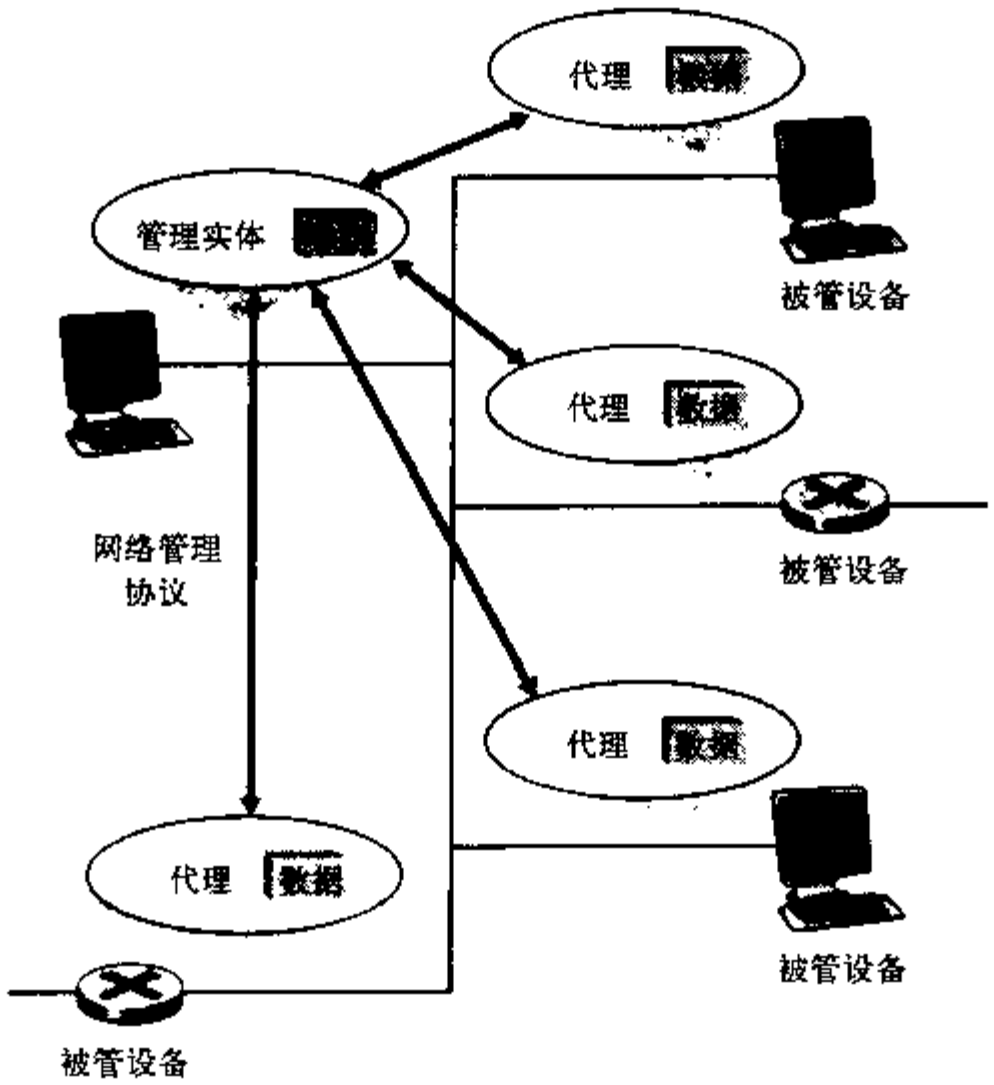


图9-2 网络管理体系结构的主要组件

网络管理体系结构的第三部分是**网络管理协议**（network management protocol）。该协议运行在管理实体和被管设备之间，允许管理实体查询被管设备的状态，并经过其代理间接地在这些设备上采取行动。代理能够使用网络管理协议向管理实体通知异常事件（如组件故障或超过了性能阈值）。需要注意的是，网络管理协议自己不能管理网络。恰恰相反，它为网络管理员提供了一种工具，网络管理员用它来管理（“监视、测试、轮询、配置、分析和控制”）网络。这是一种细微但却重要的区别。

实践原则

SprintLink的网络运营中心

各种规模的网络都在大行其道，从最小的家庭网到最大的第一层ISP，网络运营商的工作是确保网络平稳运营。而在网络运营中心（NOC）中是什么样的情况呢？网络操作

员实际干些什么呢?

Sprint具有跨越全球的网络，是全球最大的第一层ISP之一。该网络被称为SprintLink（更多的信息可查阅www.sprint.com），它具有超过70个汇集点（Points of Presence，PoP）（PoP是一个包括SprintLink IP路由器的场所，是客户与网络互连的地方）和超过800台路由器。该公司提供了多么巨大的带宽呀！SprintLink的主NOC位于美国维吉尼亚州的Reston，备份NOC位于佛罗里达州、乔治亚州和堪萨斯城。Sprint也针对它的ATM网络、帧中继网络和底层光纤设备维护NOC。在任何时候，由4个网络运营中心组成的团队管理着SprintLink IP网络核心的设备。另一个团队备份处理所有客户故障报告并对他们的请求做出响应。在监视（告警关联、故障识别和服务恢复）、配置管理和客户故障报告方面的自动化管理，使得这一小组操作员能够管理这样一个大型的复杂网络。

上图显示了Sprint技术人员在运营中心监视着网络的正常运行

当出现问题时，SprintLink操作员首先关注的是为客户迅速地恢复服务。NOC操作员为对一些已知的问题做出响应，遵循一个定义的过程集合来执行问题分类、诊断和恢复。操作员在严格规定的时间等级框架内（如15分钟）不能够立即诊断或不能修复的问题，就提交给下一级支持部门。Sprint国家技术协助中心（NTAC）提供这样的支持。NTAC成员负责深入分析引发问题的原因并解决问题；根据需要，他们还为NOC写处理过程，并与设备供应商（如路由器厂商）一起诊断和找出设备相关的问题。大约90%的问题直接由NOC技术人员和工程师处理。NOC和NTAC人员也与其他部门相互协作，其中包括合作伙伴的NOC（内部的和外部的）以及在Sprint汇集点提供现场“眼睛、手和耳朵”的Sprint人员。

如同本章前面讨论的那样，SprintLink（以及其他ISP）的“网络管理”已经从故障管理演化为性能管理，进而到服务管理，越来越强调客户需要。在关注用户需求的同时，Sprint操作员们也在优良地运营、维护和保护世界上最大的ISP之一的全球网络基础设施方面取得了很大进展。

尽管网络管理基础设施从概念上讲是很简单的，但人们经常被网络管理行话词汇（如“管理实体”、“被管设备”、“管理代理”和“管理信息库”等）搞得一头雾水。例如，用网络管理行话来说，在简单的监视主机情况中，位于“被管设备”中的“管理代理”被“管理实

体”周期性地查询，这样一个简单的想法说起来却十分绕口。幸运的是，当我们继续研究本章内容的时候，心中想着与人类组织的类比——这个类比与网络管理具有的明显对应，将有助于我们的学习。

上述有关网络管理体系结构的讨论具有一般性，能够概括性地应用于一些网络管理标准和过去提出的方法中。网络管理标准成熟于20世纪80年代后期，当时推出了OSI CMISE/CMIP（公共管理信息服务元素/公共管理信息协议）[Piscatello 1993; Stallings 1993; Glitho 1998]和因特网SNMP（Simple Network Management Protocol，简单网络管理协议）[RFC 3410; Stallings 1999; Rose 1996]两个重要的标准[Miller 1997; Subramanian 2000]。这两个标准的设计与特定厂商的产品或网络无关。在网络管理需求十分强烈的时候，因为SNMP设计和实施得十分迅速，所以得到最为广泛的应用和接受。今天，SNMP已经成为应用和实施最为广泛的网络管理框架。我们将在下面内容中详细学习SNMP。

9.3 因特网标准管理框架

与SNMP名字所暗示的相反，在因特网中进行网络管理比在管理实体和它的代理之间用于传递管理数据的协议所涉及的内容要多得多，它的不断发展已经比“简单”一词所暗示的要复杂得多。当前的因特网标准管理框架可追溯到简单网关监视协议——SGMP [RFC 1028]。SGMP是由一批大学网络研究者、用户和管理者所设计的，他们具有丰富的SGMP经验，所以仅在数月中就设计、实现和部署了SNMP [Lynch 1993]，这对于今天的延续时间相当漫长的标准化过程是一种触动。从那时起，SNMP从SNMPv1到SNMPv2再演化到最近的版本SNMPv3 [RFC 3410]，该版本于1999年4月公布，并于2002年12月更新。

当描述任何一种网络管理的框架时，不可避免地要涉及以下问题：

- 要监视什么（从语义的角度）？网络管理员能够执行什么形式的控制？
- 报告和/或交换的信息采用什么样的格式？
- 交换这些信息采用什么样的通信协议？

回想前一节中人类组织机构的类比。老板和分支机构经理需要认可报告分支机构状态的活动、生产率和预算的度量方法。类似地，他们需要认可老板所采取的措施（例如，削减预算、命令分支经理改变该办事处运作的某些方面，或解雇职员和关闭分支机构）。在较低层次的细节方面，他们需要认可数据报告的格式。例如，报告预算时采用何种货币（美元或欧元）？生产率将以什么单位来度量？虽然这些是微不足道的细节，但无论如何也要达成一致。最后，必须规定信息在中心办事处和分支机构之间传递的方式（即它们的通信协议）。

因特网标准管理框架处理上面提出的问题。该框架由4个部分组成：

- *网络管理对象的定义*。这些对象被称为MIB对象。在因特网标准管理框架中，管理信息表现为管理对象的集合，这些对象形成了一个虚拟信息存储，称为管理信息库（MIB）。一个MIB对象可以是一个计数器，例如由于IP数据报首部差错而导致的被丢弃的IP数据报的数量，或在以太网接口卡中载波侦听到的差错数量；如运行在一台DNS服务器上软件版本的描述性信息；如一个特定设备功能是否正确的状态信息；如到一个目的地的路径这种协议相关的信息。MIB对象因此定义了由被管设备维护的管理信息。相关的MIB对象聚集在MIB模块中。在人类组织类比中，MIB定义了分支机构和中心办事处之间传递的信息。
- *数据定义语言*。该语言被称为管理信息结构（SMI）。SMI定义了数据类型、对象模型，以及写入和修改管理信息的规则。MIB对象用这种数据定义语言进行定义。在人类组织

类比中，SMI用于定义被交换的信息格式的细节。

- 协议SNMP。它用于在管理实体和代理之间传递信息和命令，代理代表被管网络设备中的实体而执行操作。
- 安全性和管理能力。这些能力的增加代表了SNMPv3在SNMPv2基础上的加强。

因特网管理体系结构采用模块化的设计，采用了一种协议无关的数据定义语言和MIB，以及一种MIB无关的协议。有意思的是，这种模块化的体系结构最先提出的原因是，使基于SNMP的网络管理易于向国际标准化组织研制的网络管理框架迁移。后者是在最初构想SNMP时所提出的一种有竞争性的网络管理体系结构。然而，这种迁移并没有出现。随着时间的推移，SNMP的设计方法使它进行了3个主要版本的演化，并且允许上面讨论的SNMP的4个主要部分独立演化。显然，过去的模块化决定是正确的，即使其原因是错误的！

在下面的内容中，我们将更为详细地学习因特网标准管理框架的4个主要组件。

9.3.1 管理信息结构：SMI

管理信息结构（Structure of Management Information, SMI）是一个名字相当奇特的网络管理框架的组件，从它的名字看不出它的功能。SMI是用于定义驻留在被管网络实体中的管理信息的语言。为确保网络管理数据的语法和语义定义明确和无二义性，需要这样一种定义语言。注意SMI不是用于定义被管网络实体中特定数据的实例的，而是定义这种信息的语言。用于描述SNMPv3的SMI文档（它被称为SMIv2，这特别容易让人混淆）是 [RFC 2578; RFC 2579; RFC 2580]。我们以自底向上的方式来研究SMI，并从SMI中的基本数据类型开始。然后我们观察如何用SMI来描述被管对象，以及相关的被管对象是如何分组而成为模块的。

1. SMI基本数据类型

RFC 2578规定了用于SMI MIB模块定义语言中的基本数据类型。尽管SMI基于ASN.1（抽象语法记法1）[ISO 1987; ISO X.680 1998]对象定义语言（参见9.4节），但已经增加了足够多的SMI特定数据类型，因而可以认为SMI是具有专门用途的数据定义语言。在RFC 2578中定义的11种基本数据类型显示在表9-1中。除了这些标量对象外，还可以使用SEQUENCE OF将一些有序的MIB对象集合组合成列结构，细节请参见RFC 2578。对多数读者而言，表9-1中的多数数据类型将是熟悉的（或自解释的）。我们很快将更详细地讨论一种数据类型——OBJECT IDENTIFIER，它被用于命名对象。

表9-1 SMI的基本数据类型

数据类型	描述
INTEGER	32比特整数，如ASN.1所定义，其值在-2^{31}和$2^{31}-1$之间，或一个来自可能被命名的常数值列表的值
Integer32	32比特整数，其值在-2^{31}和$2^{31}-1$之间
Unsigned32	无符号的32比特整数，其值在0和$2^{32}-1$之间
OCTET STRING	ASN.1格式字节串，表示任意二进制或文本数据，长度最多为65 535字节
OBJECT IDENTIFIER	由管理指派的ANS.1格式（结构化名称），参见9.3.2节
IPaddress	32比特因特网地址，以网络字节顺序
Counter32	32比特计数器，能从0增加到$2^{32}-1$，然后回归到0
Counter64	64比特计数器
Gauge32	32比特整数，当增加或减少时，它不能在$2^{32}-1$以上计数，也不能降到0以下
TimeTicks	时间，自某事件起计算，精度为0.01秒
Opaque	未解释的ASN.1字符串，用于向后兼容

2. SMI较高层结构

除了基本数据类型外，SMI数据定义语言还提供了较高层的语言结构。

OBJECT-TYPE结构用于定义被管对象的数据类型、状态和语义。概括来说，这些数据对象包含了位于网络管理中心的管理数据。在各种因特网RFC中，有大约10 000个已定义的对象[RFC 3410]。OBJECT-TYPE结构具有4条子句。OBJICT-TYPE定义的SYNTAX子句规定了与该对象相关的基本数据类型。MAX-ACCESS子句规定了该被管对象是否能读、能写、能创建，或在一个通知中是否包括它的值。STATUS子句指出了该对象定义是否是当前的、合法的、过时的（在这种情况下，它不应当被实现，因为该定义仅为了历史原因才出现）或不赞成的（过时的，但为了与过去的实现互操作而实现）。DESCRIPTION子句包含有关该对象的人可读的文本定义；它是该被管对象用途的文档，提供了实现该被管对象所需的所有语义信息。

作为OBJECT-TYPE结构的一个例子，考虑RFC 4293定义的ipSystemStatsInDelivers对象类型。该对象定义了一个32比特的计数器，用以跟踪被管设备接收到并成功地传递给较高层协议的IP数据报的数量。该定义的最后一行与该对象名字有关，这是我们将在下一节中考虑的主题。

```
ipSystemStatsInDelivers OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS read-only
    STATUS      current
    DESCRIPTION
          "The total number of datagrams successfully
          delivered to IPuser-protocols (including ICMP).

          When tracking interface statistics, the counter
          of the interface to which these datagrams were
          addressed is incremented. This interface might
          not be the same as theinput interface for
          some of the datagrams.

          Discontinuities in the value of this counter can
          occur at re-initialization of the management
          system, and at other times as indicated by the
          value of ipSystemStatsDiscontinuityTime."
    ::= { ipSystemStatsEntry 18 }
```

MODULE-IDENTITY结构允许相关的对象在一个“模块”中分为一组。例如，RFC 4293规定了用于定义被管对象（包括ipSystemStatsInDelivers）的MIB模块，这些被管对象用于网际协议（IP）和与之相关的互联网控制报文协议（ICMP）的管理实现。RFC 4022规定了用于TCP的MIB模块，RFC 4133规定了用于UDP的MIB模块。RFC 4502定义了用于RMON远程监视的MIB模块。除了包括在该模块中的被管对象的OBJECT-TYPE定义外，MODULE-IDENTITY结构还包含了该模块作者的联系信息、最后更新的日期、修订历史和该模块的文本性描述。举一个例子，考虑下面用于IP协议管理的模块定义：

```
ipMIB MODULE-IDENTITY
    LAST-UPDATED "200602020000Z"
    ORGANIZATION "IETF IPv6 MIB Revision Team"
    CONTACT-INFO
          "Editor:
          Shawn A. Routhier
          Interworking Labs
          108 Whispering Pines Dr. Suite 235
          Scotts Valley, CA 95066
```

```
        USA
        EMail: <sar@iwl.com>"
    DESCRIPTION
        "The MIB module for managing IP and ICMP
        implementations, but excluding their
        management of IP routes.

        Copyright (C) The Internet Society (2006). This
        version of this MIB module is part of RFC 4293;
        see the RFC itself for full legal notices."

    REVISION      "200602020000Z"
    DESCRIPTION
        "The IP version neutral revision with added
        IPv6 objects for ND, default routers, and
        router advertisements.  As well as being the
        successor to RFC 2011, this MIB is also the
        successor to RFCs 2465 and 2466. Published
        as RFC 4293."

    REVISION      "1994110100002"
    DESCRIPTION
        "A separate MIB module (IP-MIB) for IP and
        ICMP management objects. Published as RFC
        2011."

    REVISION      "9109331000002"
    DESCRIPTION
        "The initial revision of this MIB module was
        part of MIB-II, which was published as RFC
        1213."
    ::= { mib-2 48}
```

NOTIFICATION-TYPE结构用来定义由代理或管理实体产生的有关SNMPv2-Trap和InformationRequest报文的信息（参见9.3.3节）。该信息包括了文本性DESCRIPTION：何时发送这样的报文，以及包括在产生的报文中的值列表，详情参见[RFC 2578]。MODULE-COMPLIANCE结构定义了一个代理必须实现的模块中的被管对象集合。AGENT-CAPABILITIES结构定义了代理关于对象和事件通知定义的能力。

9.3.2 管理信息库：MIB

如上所述，**管理信息库**（Management Information Base, MIB）可以被认为是一个虚拟信息库，库中被管对象的值总体上反映了该网络的当前“状态”。查询和/或设置这些值，可以由管理实体通过向代理发送SNMP报文来进行，而代理代表管理实体在被管设备上执行操作。被管对象使用前面讨论的OBJECT-TYPE SMI结构来定义，并使用MODULE-IDENTITY结构汇集在**MIB模块**（MIB module）中。

IETF已经对与路由器、主机和其他网络设备相关的MIB模块进行了标准化。这包括了有关硬件的特定部分的基本标识数据以及设备的网络接口和协议的管理信息。到2006年，有200多个标准的MIB模块，以及更多的厂商特定（专用）的MIB模块。对于所有这些标准，IETF需要一种方法来标识和命名标准化的模块以及模块中的特定被管对象。IETF没有白手起家，而是采用国际标准化组织（ISO）已经提出的一种标准化对象标识（命名）框架。由于具有许多标准化组织，所以ISO对于它们的标准化对象标识框架有“庞大计划”，即标识任何网络中每个可能的标准化对象（如数据格式、协议或部分信息），而不管该对象是由哪个网络标准组织（例如，因特网IETF、ISO、IEEE或ANSI）、设备制造商或网络拥有者所定义的。这的确

是一个很崇高的目标！由ISO采纳的对象标识框架是ASN.1（抽象语法记法1）对象定义语言的一部分[ISO 1987; ISO X.680 1998]，我们将在9.4节中讨论。标准的MIB模块在这个无所不包的命名框架中具有它们自己的合适的位置，如下面所讨论的那样。

如图9-3所示，对象在ISO命名框架中以等级结构方式进行命名。注意该树上的每个分支点具有一个名字和一个编码（显示在圆括号中），该树中的任何点因此可由名字或编码的序列所标识，它规定了从根到标识树的那个点的路径。一个有趣的（但不完整且是非正式的）基于Web的实用程序可用于浏览部分该对象标识树（使用志愿者提供的分支信息），有关信息可在[Alvestrand 1997]和[France Telecom 2006]中找到。

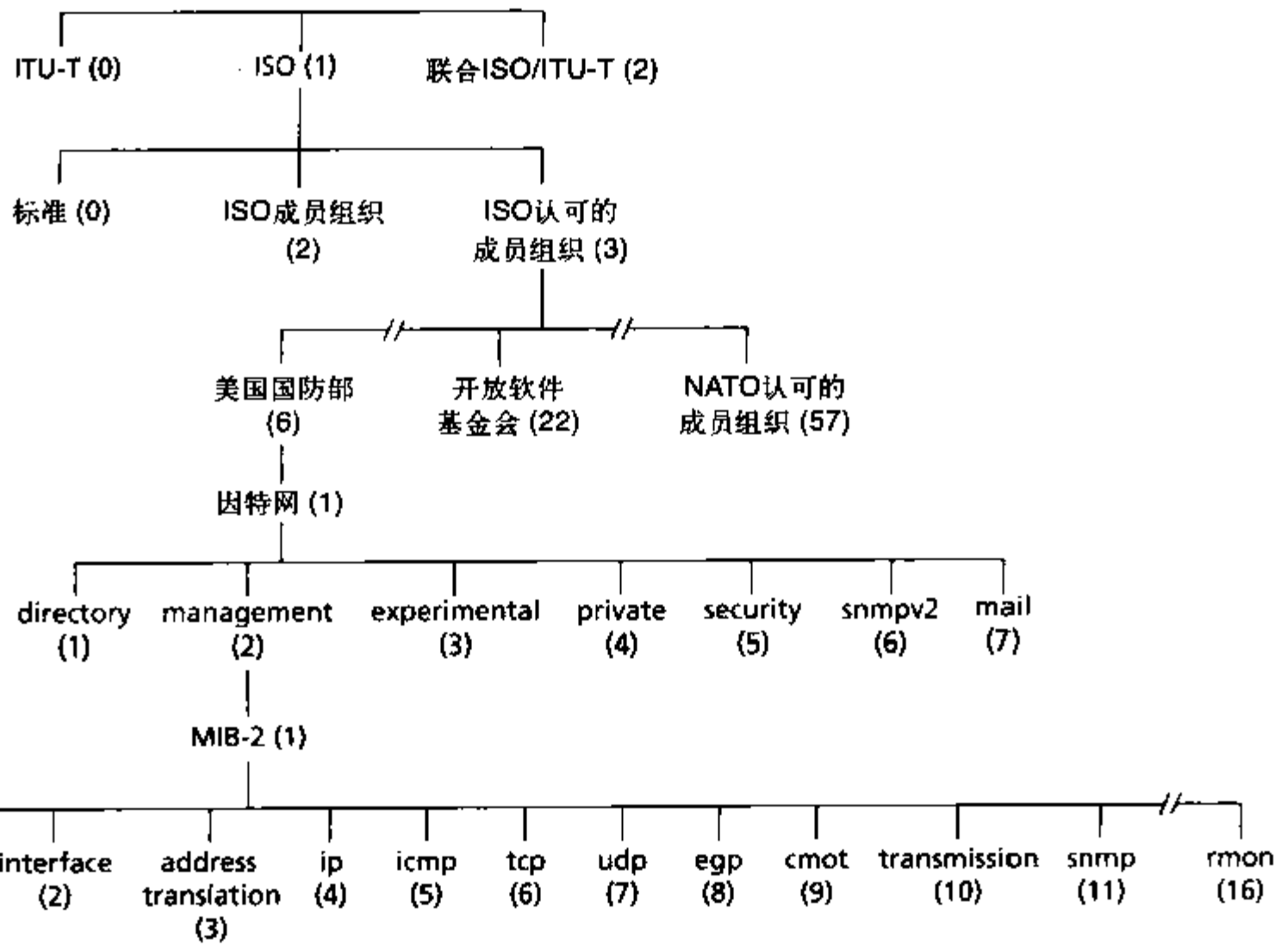


图9-3 ANS.1对象标识树

在该等级结构的顶部是ISO和国际电信联盟（ITU-T）的电信标准化组织，以及这两个组织联合工作的一个分支机构。这两个主要的标准化组织共同研究ANS.1。在这棵树的ISO分支下，我们还发现一些条目，它们分别对应所有ISO标准（1.0）和各个ISO成员国的标准组织所发布的标准（1.2）。尽管没有在图9-3中显示出来，但在ISO成员组织（又可表示为1.2）下，我们能发现USA（1.2.840），在它下面我们将能发现IEEE、ANSI和公司特定标准的编码，其中包括RSA（1.2.840.11359）和微软（1.2.840.113556）。在微软下面，我们能发现微软文件格式（1.2.840.113556.4）用于各种微软产品，如Word（1.2.840.113556.4.2）。但是我们现在感兴趣的是网络（并非微软的Word文件），因此我们将注意力转向标号为1.3的分支，这些标准由ISO认可的组织发布。这些组织包括美国国防部（6）（在它下面我们能发现因特网标准）、开放软件基金会（Open Software Foundation）（22）、航空协会SITA（69）和北大西洋公约组织（NATO）认可的成员组织（57），以及许多其他组织。

在该树的因特网分支（1.3.6.1）下面，有7个类别。在private（1.3.6.1.4）分支下，列出

了许多名字和专用企业编码[IANA 2007b]，该列表有几千个已经在因特网编号分配机构(Internet Assigned Number Authority, IANA）注册的专门公司[IANA 2007]。在对象标识树的management（1.3.6.1.2）和MIB-2（1.3.6.1.2.1）分支下，我们发现了标准MIB模块的定义。要到达ISO名字空间的各个角落需要经历多么漫长的旅行啊！

标准化的MIB模块

在图9-3中，树的最底层显示了某些重要的面向硬件的MIB模块（system和interface）以及与某些最重要的因特网协议相关的模块。RFC 3700列出了所有的标准MIB模块。虽然阅读MIB相关的RFC相当乏味和枯燥，但讨论一些MIB模块定义，对认识模块中的信息类型是有指导意义的（就像吃蔬菜对你身体有好处一样）。

位于system之下的被管对象包含了有关被管设备的一般性信息；所有被管设备必须支持system MIB对象。表9-2定义了system组中的对象，这些对象由RFC 1213所定义。表9-3定义了在一个被管实体中用于UDP协议的MIB模块中的被管对象。

表9-2 在MIB-2 system组中的被管对象

对象标识符	名 字	类 型	描述（引自RFC 1213）
1.3.6.1.2.1.1.1	sysDescr	OCTET STRING	"系统的硬件类型、软件操作系统和网络软件的全名和版本标识"
1.3.6.1.2.1.1.2	sysObjectID	OBJECT IDENTIFIER	分配给厂商的对象ID，"它提供了一种易于操作和无二义性的方法来决定被管理的'单元的类型'"
1.3.6.1.2.1.1.3	sysUpTime	TimeTicks	"自系统的网络管理部分最后被重新初始化以来的时间（精度为0.01秒）"
1.3.6.1.2.1.1.4	sysContact	OCTET STRING	"该被管节点的联系人，以及与该人的联系方式"
1.3.6.1.2.1.1.5	sysName	OCTET STRING	"为该节点正式分配的名字。按惯例，这是该节点的全称域名"
1.3.6.1.2.1.1.6	sysLocation	OCTET STRING	"该节点的物理位置"
1.3.6.1.2.1.1.7	sysServices	Integer32	指出在该节点可用的服务集合的编码值：物理的（如一个转发器），数据链路/子网（如网桥），因特网（如IP网关），端到端（如主机），应用程序

表9-3 在MIB-2 UDP模块中的被管对象

对象标识符	名 字	类 型	描述（引自RFC 4113）
1.3.6.1.2.1.7.1	udpInDatagrams	Counter32	"交付给UDP用户的UDP数据报总数"
1.3.6.1.2.1.7.2	udpNoPorts	Counter32	"在那些没有应用程序的目的端口所接收到的UDP数据报总数"
1.3.6.1.2.1.7.3	udpInErrors	Counter32	"接收的不能传递的UDP数据报数量，其原因并非是目的端口没有应用程序"
1.3.6.1.2.1.7.4	udpOutDatagrams	Counter32	"从该实体发送的UDP数据报总数"

9.3.3 SNMP协议运行和传输映射

简单网络管理协议版本2（SNMPv2）[RFC 3416]用于管理实体和代表管理实体执行的代理之间传递MIB信息。SNMP的最常用方式是**请求响应**（request-response）模式，其中SNMPv2管理实体向SNMPv2代理发送一个请求，代理接收到该请求后，执行某些动作，然后向该请求发送一个回答。请求通常用于查询（检索）或修改（置位）与某被管设备相关的

MIB对象值。第二个常用的SNMP报文是代理主动向管理实体发送的报文，该报文被称为陷阱报文（trap message）。该报文用于通知管理实体，一个异常情况已经导致了MIB对象值的改变。我们在9.1节中已看到，网络管理员在出现如下情况时可能希望接收到陷阱报文：一个接口不工作；某链路上的拥塞达到一个预定的级别；发生了某些其他值得注意的事件。注意，在轮询（请求响应交互）和陷阱之间有许多折中，参见课后习题。

表9-4中显示了SNMPv2定义的7种类型的报文，这些报文一般称为协议数据单元（PDU）。图9-4显示了这些PDU的格式。

表9-4 SNMPv2 PDU类型

SNMPv2 PDU类型	发送方到接收方	描 述
GetRequest	管理者到代理	取得一个或多个MIB对象实例值
GetNextRequest	管理者到代理	取得列表中的下一个MIB对象实例值
GetBulkRequest	管理者到代理	以大数据块方式取得值，例如大表中的值
InformRequest	管理者到管理者	向不能访问的远程管理实体通知MIB值
SetRequest	管理者到代理	设置一个或多个MIB对象实例的值
Response	代理到管理者或管理者到管理者	对GetRequest、GetNextRequest、GetBulkRequest、SetRequest PDU或InformRequest产生的响应
SNMPv2-Trap	代理到管理者	向管理者通知一个异常事件

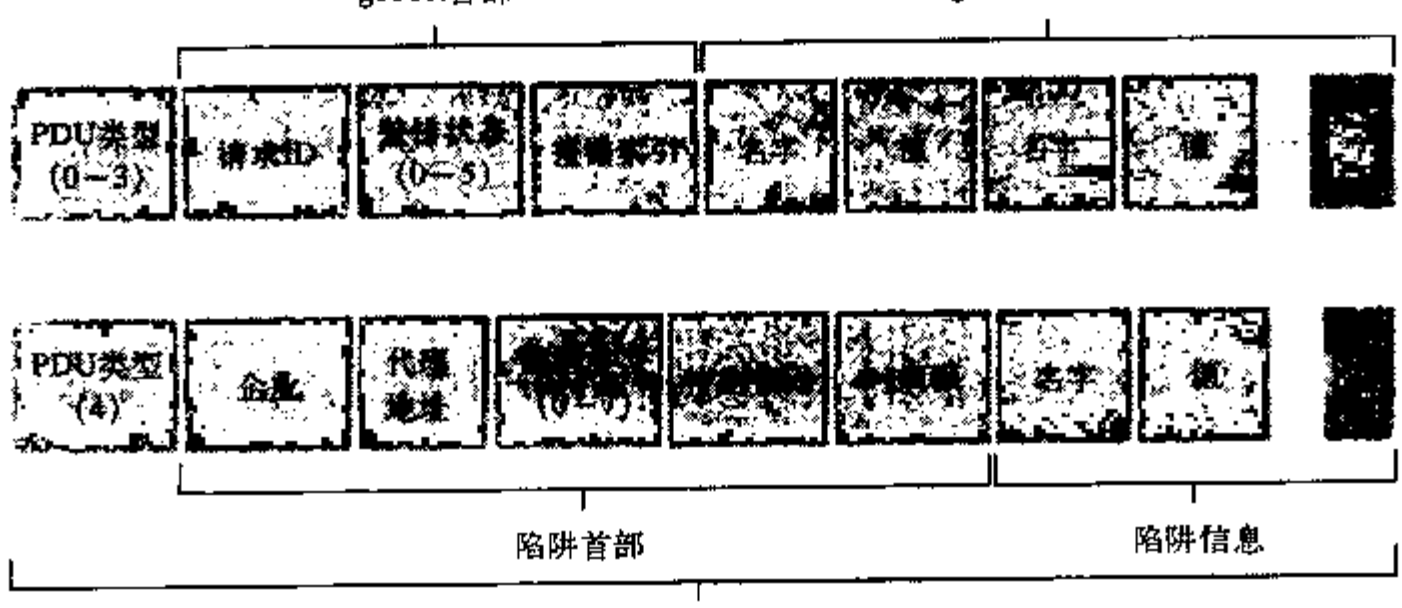


图9-4 SNMP PDU格式

- GetRequest、GetNextRequest和GetBulkRequest PDU都是管理实体向代理发送的，以请求位于该代理所在的被管设备中的一个或多个MIB对象值。其值被请求的MIB对象的对象标识符定义在该PDU的变量绑定部分。GetRequest、GetNextRequest和GetBulkRequest的差异在于它们的数据请求粒度。GetRequest能够请求MIB值的任意集合；多个GetNextRequest能用于顺序地读取MIB对象的序列或表；GetBulkRequest允许读取大块数据，能够避免因发送多个GetRequest或GetNextRequest报文可能导致的额外开销。在这三种情况下，代理用包括该对象标识符和它们相关值的Response PDU进行响应。
- SetRequest PDU是管理实体用来设置被管设备中的一个或多个MIB对象值的。代理用

带有“无错”（noError）差错状态的Response PDU来回答，证实该值的确已经被设置。

- InformRequest PDU是一个管理实体用来通知另一个管理实体的，其中MIB信息是接收实体不能访问的。该接收实体用带有“无错”（noError）差错状态的Response PDU来确认收到了InformRequest PDU。
- SNMPv2 PDU的最后一种类型是陷阱报文。陷阱报文是异步产生的，即它们不是为了响应接收到的请求而产生的，而是为了响应管理实体所需要的通知事件而产生的。RFC 3418定义了周知的陷阱类型，其中包括设备进行的冷启动或热启动，链路从故障状态变为工作状态或相反，相邻设备找不到，或鉴别故障事件。接收到的陷阱请求不要求从管理实体得到响应。

给出了SNMPv2请求响应性质后，值得注意的是，尽管SNMP PDU能够通过许多不同的运输协议传输，但SNMP PDU通常还是在UDP数据报的负载中传输的。事实上，RFC 3417认为UDP是“首选的运输映射”。由于UDP是一种不可靠的运输协议，因而不能确保一个请求或它的响应能够被它的目的地接收到。管理实体用该PDU的请求ID字段作为它向代理发送的请求编号；代理的响应从接收到的请求中获取该请求ID。因此，请求ID字段能被管理实体用来监测丢失的请求或回答。如果在一定时间后还没有收到对应的响应，由管理实体来决定是否重传一个请求。特别是，SNMP标准没有强制任何特殊的重传过程，即便是首先进行了重传。它只是要求管理实体“需要根据重传的频率和周期可靠地动作”。这当然引起人们想知道一个“可靠的”协议应当如何去做！

9.3.4 安全性和管理

SNMPv3的设计者们说过，“可以将SNMPv3看成是具有附加的安全性和管理能力的SNMPv2”[RFC 3410]。当然，SNMPv3比起SNMPv2有一些变化，但是其他方面的变化都没有管理和安全性领域的变化那么明显。在SNMPv2中，安全性的中心地位是特别重要的，因为缺乏安全性会导致SNMP只能用于监视而不能用于控制（例如，在SNMPv1中很少使用SetRequest）。

SNMP经历了3个版本后已经成熟，它的功能有了增加，但不幸的是，与SNMP相关的标准文档的数量也同时增加了。这可以通过下述事实来证实：现在甚至有了“描述用于描述SNMP管理框架的体系结构”的RFC [RFC 3411]！虽然描述一个框架的体系结构这个概念也许不太好理解，但RFC 3411的目的是值得肯定的，它引入了一种共同语言，用于描述功能和由SNMPv3代理或管理实体所采取的动作。SNMPv3实体的体系结构是简单明了的，浏览该体系结构将有助于我们理解SNMP。

SNMP应用程序（SNMP application）由命令产生器、通知接收器和代理转发器（这些通常位于管理实体中）、命令响应器和通知源发器（这两者通常位于代理中）以及其他可能的应用程序组成。命令产生器产生GetRequest、GetNextRequest、GetBulkRequest和SetRequest PDU，处理对这些PDU接收到的响应。这些PDU都是我们在9.3.3节讨论过的命令。命令响应器在代理中执行，接收、处理和（用Response报文）回答接收到的GetRequest、GetNextRequest、GetBulkRequest和SetRequest PDU。代理中的通知源发器应用程序产生Trap PDU，这些PDU最终由管理实体中的通知接收器应用程序接收和处理。代理转发器应用程序转发请求PDU、通知PDU和响应PDU。

由SNMP应用程序发送的PDU在经过适当的运输协议发送之前，通过SNMP“引擎”传递。

图9-5显示了由命令产生器应用程序产生的一个PDU先进入调度模块，在那里决定SNMP的版本。然后该PDU在报文处理系统中进行处理，在这里被封装在包括SNMP版本号、报文ID和报文长度信息的报文首部中。如果需要加密或鉴别，还要包括针对这些信息的适当首部字段信息，详见[RFC 3411]。最后，SNMP报文（应用程序产生的PDU加上首部信息的报文）被传递到适当的运输协议。携带SNMP报文的首选运输协议是UDP（即SNMP报文被作为UDP数据报的负载传送），用于SNMP的首选端口号是端口161。端口162用于陷阱报文。

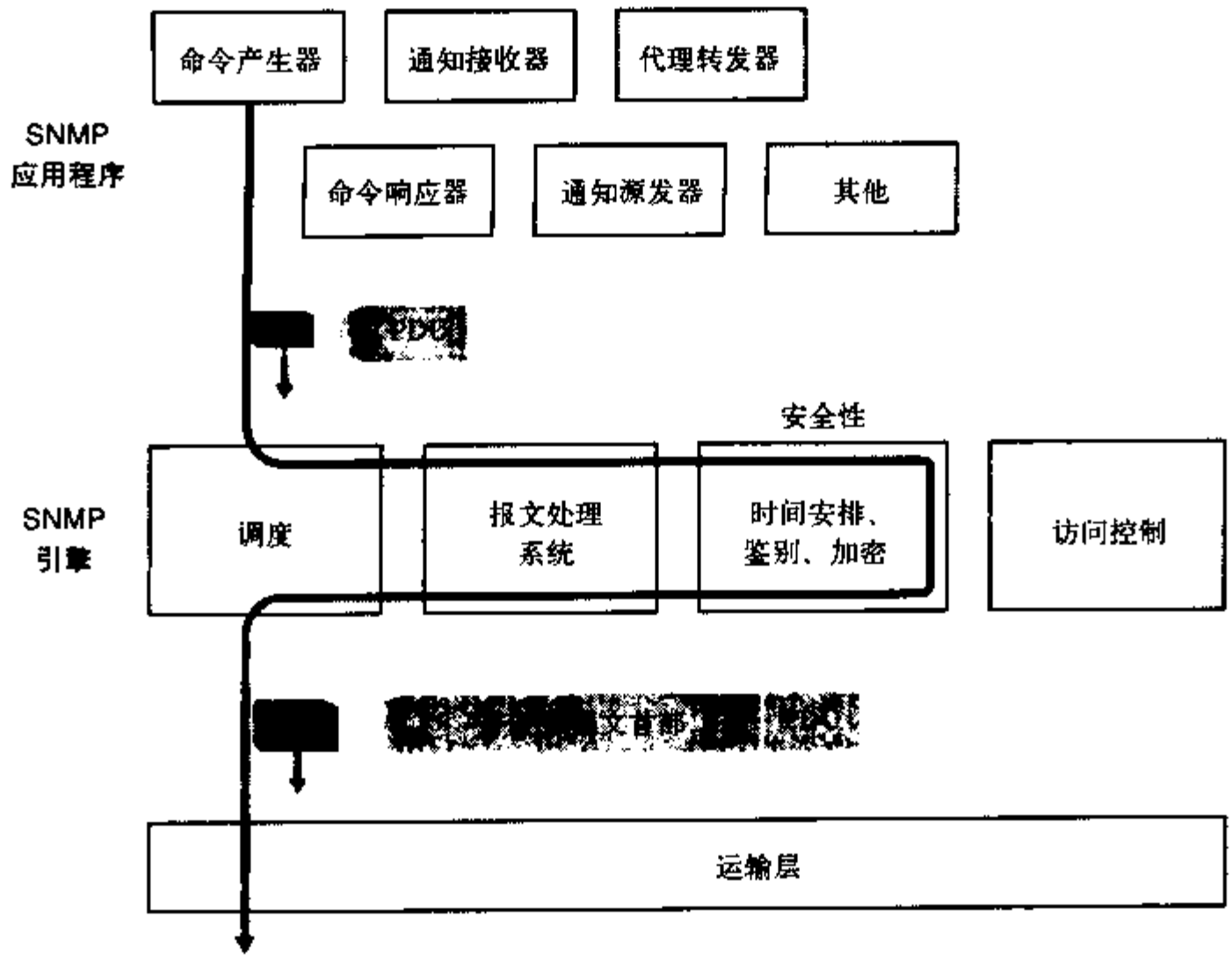


图9-5 SNMPv3引擎和应用程序

前面我们已经看到SNMP报文不仅能用于监视，也能用于控制（例如，通过SetRequest命令）网络元素。显然，若一个入侵者能够截获SNMP消息和/或产生它自己的SNMP报文并向管理基础设施发送，则他就可能会对网络造成损害。因此，安全地传输SNMP报文是至关重要的。令人惊奇的是，仅在SNMP最近的版本中，安全性才得到应有的注意。SNMPv3的安全性被称为**基于用户的安全性**（user-based security）[RFC 3414]，这是因为它应用了传统的用户概念。用户采用用户名来标识，还有相关的口令、密码值或访问权限等安全信息。SNMPv3提供了加密、鉴别、对重放攻击的防护（参见8.4节）和访问控制功能。

实践原则

今天有数以百计（就算不是数以千计的话）的网络管理产品可供使用，这些产品都在某种程度上包括了我们在本节中研究过的网络管理框架和SNMP基础。对这些产品的综述已经超出了本书的范围，（无疑）也会分散读者的注意力。因此，我们在这里给出一些非常杰出的产品的要点。对于众多网络管理工具的概括性了解可从阅读文献[Subramanian 2000]中的第12章开始。

网络管理工具大体上分为两类：一类来自网络设备制造商，它们专用于管理该厂商的设备；另一类针对具有异构设备的网络的管理。前一类产品包括用于Cisco设备的网络管理工具——Cisco网络应用程序性能分析（NAPA）套件[Cisco NAPA 2007]，用于网络配备和SLA/QoS支持的Lucent和Juniper提供的运行支持系统（OSS）[Lucent 2006]。

用于管理异构网络的流行工具包括惠普公司的OpenView[OpenView 2007]、Aprisma公司的Spectrum [Aprisma 2007]和Sun公司的Solstice网络管理系统[Sun 2007]。这三个系统都采用了分布式系统体系结构，用多个服务器从它们的管理域中收集网络管理信息。网络管理工作站然后能够从这些服务器收集结果，显示信息和采取控制动作。这三个产品都支持SNMP和CMIP协议，并对事件/告警关联提供自动帮助。

- 加密。SNMP PDU能够使用数据加密标准（DES）的密码分组链接模式加密。注意DES是一个共享密钥的系统，用户用于加密数据的密钥必须要为对该数据解密的接收实体所知。
- 鉴别。SNMP结合使用了散列函数和密钥值，以提供鉴别并保护数据不受篡改，其中的散列函数如我们在8.3节学习的MD5算法。称为HMAC（Hashed Message Authentication Codes，散列报文鉴别码）[RFC 2104]的方法在概念上是简单的。假定发送方具有一个SNMP PDU m，希望将其发送到接收方。该PDU可能已经被加密。又假定发送方和接收方都知道一个共享的密钥K，K不必与用于加密的密钥相同。该发送方将向接收方发送m。然而，发送时携带的并不是简单的报文摘要或报文完整性码（MIC）——MIC(m)，该码是对m计算而得，以保护其不被篡改。发送方将该共享密钥附加到m并计算得到MIC，即MIC(m, K)是由对PDU和密钥都进行计算而得。该MIC(m, K)的值（而不是密钥）连同m一起被传输。当接收方接收到m时，它附加上该密钥K，并计算MIC(m, K)。如果该计算的值与传输的值MIC(m, K)相匹配，则接收方不仅知道该报文没有被篡改，而且知道该报文是由知道K值的人所发送的，即该报文是由值得信任和已被鉴别的人所发送的。在上述过程中，HMAC实际上执行了两次附加和散列操作，每次都使用了一个稍加修改的密钥值，详见[RFC 2104]。
- 对重放攻击的防护。在第8章中我们讨论了不重数（nonce）能用于防护重放攻击。SNMPv3采用了相关方法。为了确保一个接收的报文不是某个先前报文的重放，接收方要求发送方在每个报文中包括一个基于接收方的计数器的值。该计数器的功能如同不重数一样，反映了自接收方的网络管理软件最后一次启动以来的时间和自接收方的网络管理软件最后一次配置以来启动的总数。只要接收报文中的计数器位于接收方的实际值的某种错误容限内，该报文就被认为是无重放攻击的报文，此时它已被鉴别和/或加密。详见[RFC 3414]。
- 访问控制。SNMPv3提供了一种基于视图的访问控制[RFC 3415]，该方法控制了哪些网络管理信息能被哪些用户查询和/或设置。一个SNMP实体在本地配置库（Local Configuration Datastore, LCD）保留了有关访问权限和策略的信息。LCD的某些部分自身可作为被管对象访问，在基于视图的访问控制模型配置MIB中定义[RFC 3415]，并因此能够经SNMP从远程进行管理和操作。

9.4 ASN.1

在本书中，我们已经涉及了计算机网络中的许多有趣主题。本节有关ASN.1的内容或许算

不上前10个有趣主题。然而，像蔬菜有益于健康一样，有关ASN.1的知识和表示服务的更广泛问题是对你有益的东西。ASN.1是一种ISO发明的标准，用于一些因特网相关的协议中，特别是用于网络管理领域中。例如，我们在9.3节中看到，SNMP中的MIB变量与ASN.1有不解之缘。尽管ASN.1的有关资料可能相当枯燥，但我们希望读者能够认识到这些相关材料的重要性。

为了便于我们下面的讨论，考虑下列可能的实验。假定一个人能够可靠地将数据从一台计算机的内存直接拷贝到远程另一台计算机的内存中。如果他能够这样做，其中的通信问题是如何解决的？该问题的答案取决于他对“该通信问题”的定义。毫无疑问，一个完善的内存到内存拷贝将是从一台机器到另一台机器的比特和字节的实际通信。当接收计算机上运行的软件存取这些数据时，这些比特和字节的实际拷贝意味着什么？我们能够看到与发送计算机内存中所存储的一样的值吗？对该问题的答案是“不一定”！该问题的症结在于，不同的计算机体系结构、不同的编译程序具有不同的存储和表示数据的常规方法。如果数据要通信或在多台计算机中存储（因为数据在各通信网络中），显然要解决数据表示的问题。

作为该问题的一个例子，考虑下面简单的C代码段。这个结构怎样在内存中安排呢？

```
struct {
  char code;
  int x;
  } test;
test.x = 259;
test.code = 'a';
```

图9-6的左侧显示了这些数据在一种假想的体系结构中的可能的内存安排：在一个单字节内存中包含了字符“a”，其后的16比特的字包含了整数值259，以高阶字节在前的顺序存储。在另一台计算机中的内存安排显示在图9-6的右侧：字符“a”后面是16比特的整数值，以低阶字节在前的顺序存储，它们以16比特的字为边界对齐。当然，如果某人在这两台机器的内存之间执行逐字拷贝，使用相同的结构定义来存取存储的值，他将在这两台计算机上看到极为不同的结果！

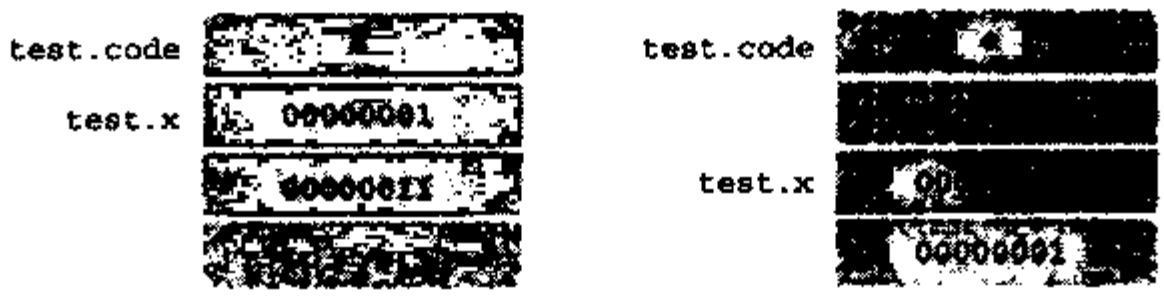


图9-6 两种不同的体系结构上的数据排列

不同的体系结构具有不同的内部数据格式，这是一个真实而普遍的问题。以不同的形式存储整数的特定问题非常常见，并且有了专有名称。用于存储整数的“大端法”（big-endian）次序要求先存储最高阶字节（在最低的存储地址），“小端法”（little-endian）次序要求先存储最低阶字节。Sun SPARC和Motorola处理器是“大端在先”方式，而Intel和DEC/Compaq Alpha处理器是“小端在先”方式。这里插入一个小插曲，术语“big-endian”和“little-endian”来自Jonathan Swift的书《格列佛游记》，书中的两群人固执己见地坚持以两种不同的方式做一件简单的事情（希望对计算机体系结构方面的这个类比有助于理解）。来自小人国的一群人坚持在较大一端打开鸡蛋（“大端法”），而其他人坚持在较小一端打开鸡蛋。这个分歧导致了大规模民事冲突和叛乱。

既然存在不同方式的计算机存储和数据表示，网络协议该怎样处理这些问题呢？例如，

如果一个SNMP代理打算发送一个响应报文，其中包含接收到的UDP数据报数量的整数计数值，它怎样表示将向管理实体发送的该整数值呢？是以大端法还是以小端法次序呢？一种选择是代理以管理实体中存储的相同顺序来发送这些字节。另一种选择是代理以自己的存储顺序发送，并让接收实体根据需要重新排序。任何一种选择都要求发送方或接收方知道其他方的整数表示格式。

第三种选择是用与机器无关、OS无关、语言无关的方式描述整数和其他数据类型（即一种数据定义语言）以及规则，该规则定义了每种数据类型跨越网络传输的方式。当接收到给定类型的数据时，它以一种已知格式来接收，并能以任意机器特定的格式进行存储。我们在9.3节中学习的SMI和ASN.1都采用了第三种选择。用ISO术语来说，这两个标准描述了一种**表示服务**（presentation service），该服务用于从一种机器特定的格式到另一种机器特定的格式来传输和转换信息。图9-7显示了一个真实世界中的表示问题：两个接收方都不理解交流的基本思想——谈话者喜欢的东西。如图9-8所示，一个表示服务能够解决该问题，即通过将思想转换成共同理解的（通过表示服务）、与个人无关的语言，向接收方发送该信息，然后转换成接收方能理解的语言。

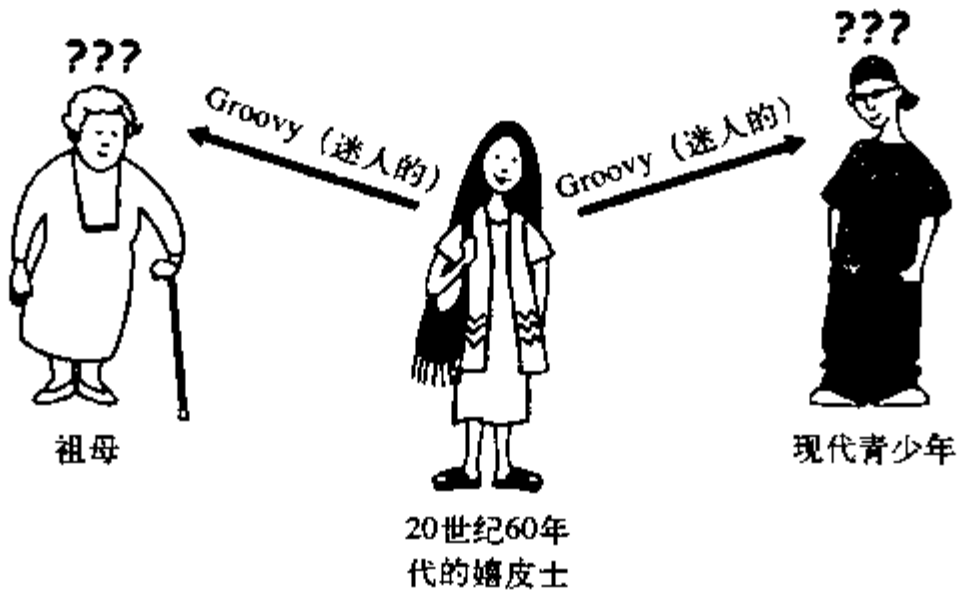


图9-7 表示问题

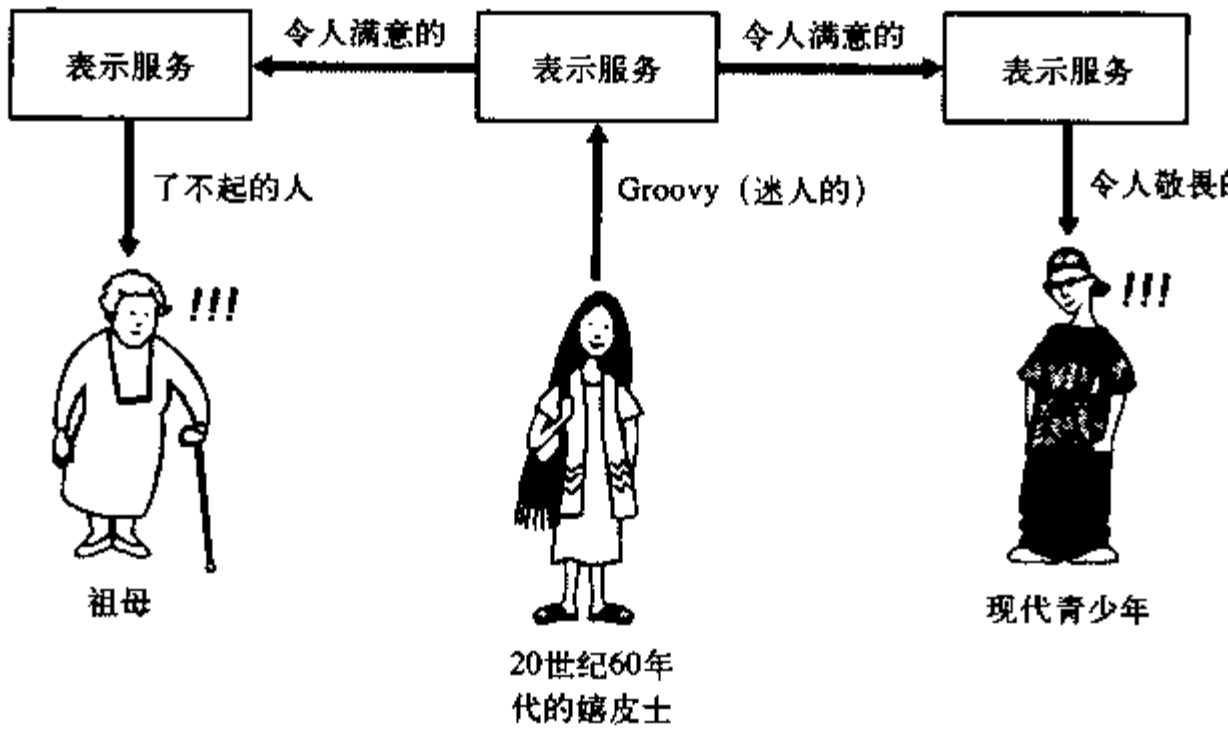


图9-8 表示问题的解决

表9-5显示了一些ASN.1定义的数据类型。我们在前面学习SMI时遇到过INTEGER、

OCTET STRING和OBJECT IDENTIFIER数据类型。这里我们的目标不是提供对ASN.1的完整概述，推荐读者阅读描述ASN.1类型和结构的标准或已出版和在线的书籍[Larmouth 1996]。这些结构如SEQUENCE和SET，用它们可定义结构化的数据类型。

表9-5 部分ASN.1数据类型

标 志	类 型	描 述
1	BOOLEAN	值为“真”或“假”
2	INTEGER	可以为任意大
3	BITSTRING	一个或多个比特的列表
4	OCTET STRING	一个或多个字节的列表
5	NULL	无值
6	OBJECT IDENTIFIER	名字，位于ASN.1标准命名树上，参见9.2.2节
9	REAL	浮点

除了提供数据定义语言外，ASN.1也提供**基本编码规则**（Basic Encoding Rule, BER）。BER定义了采用ASN.1数据定义语言所定义的对象实例经由网络发送的方式。BER采用TLV（Type, Length, Value，类型、长度、值）方法对发送的数据进行编码。对于被发送的每个数据项，将以该数据项的数据类型、长度和该数据项实际值的顺序进行发送。对于这个简单的规则，接收到的数据基本上是自标识的。

图9-9给出了如何发送两个数据项的一个简单例子。在这个例子中，假定采用大端法顺序，

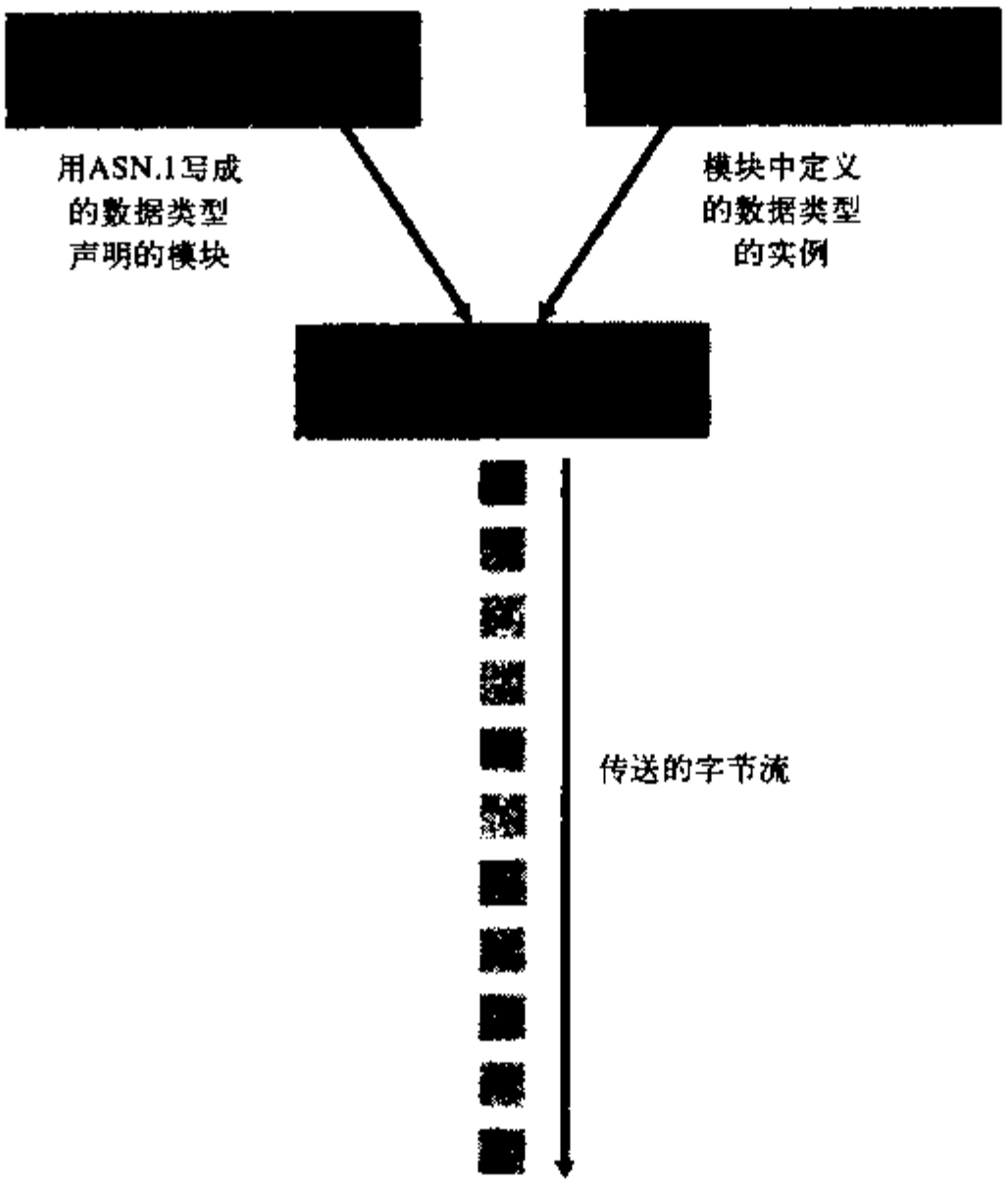


图9-9 BER编码例子

发送方要发送字符串“smith”，后面带有值为259的十进制数（这等于二进制数00000001 00000011，或值为1的一个字节后跟值为3的一个字节）。这个传输流中的第一个字节具有值4，指出后面的数据项的类型是OCTET STRING；这是TLV编码中的“T”。该流中的第二个字节包含了OCTET STRING的长度，本例中是5。该传输流中的第三个字节开始了长度为5的OCTET STRING，它包含了字母“s”的ASCII表示。下个数据项的T、L和V的值分别是2（INTEGER类型标志值）、2（即长度为2字节的一个整数）和十进制数259的2字节的大端法表示。

在以上讨论中，我们仅仅接触了ASN.1的少而简单的内容。如要学习更多有关ASN.1的内容，可以参考ASN.1标准文档[ISO 1987; ISO X.680 1998]、OSI相关的在线书籍[Larmouth 1996]和ASN.1相关的Web站点[OSS 2007]和[France Telecom 2006]。

9.5 小结

我们对网络管理以及所有与网络有关的内容的学习现在已经结束！

在有关网络管理的最后一章中，我们以需求为引导，为网络管理员提供适当的工具，以监视、测试、轮询、配置、分析、评估和控制网络的运营。网络管理员的责任是保持网络正常运营。书中给出的与复杂系统（如发电厂、飞机和人类组织）的类比，有助于启发这种需求。我们看到网络管理系统的体系结构围绕5个关键组件：①一个网络管理者；②一组被管的（距网络管理者）远程设备；③在这些设备中的管理信息库（MIB），其中包含有关这些设备的状态和操作数据；④报告MIB信息并在网络管理者控制下采取措施的远程代理；⑤网络管理者和远程设备之间的通信协议。

接着我们深入研究了因特网网络管理框架，特别是SNMP协议的细节。我们看到SNMP如何对网络管理体系结构的5个关键组件实例化，花了不少时间来研究MIB对象、SMI（特定于定义MIB的数据定义语言）和SNMP协议本身。注意SMI和ASN.1是密不可分的，ASN.1在ISO/OSI 7层参考模型的表示层中起着关键作用，所以我们简要地学习了ASN.1。也许比ASN.1自身细节更为重要的是，在网络中特定于机器的数据格式之间的转换。尽管某些网络体系结构由于具有表示层而明确声明了这种服务的重要性，但在因特网协议栈中缺少这个层次。

还值得一提的是，本书还有许多重要的网络管理主题我们有意没有涉及，例如，故障识别和管理、前摄性异常检测、告警关联以及服务管理的更大的问题（例如，与网络管理形成对照）。虽然这些主题很重要，但对它们的讨论应当由专门的文献来进行，请读者参考9.1节中提到的文献。

课后习题和问题

复习题

9.1节

1. 为什么网络管理员得益于使用网络管理工具？描述5种相关情况。
2. ISO定义的网络管理的5个领域是什么？
3. 网络管理和服务管理的主要区别是什么？

9.2节

4. 定义下列术语：管理实体、被管设备、管理代理、MIB和网络管理协议。

9.3节

5. 网络管理中SMI的作用是什么？

6. 在SNMP中，请求响应报文和陷阱报文之间的重要区别是什么？
7. SNMP中使用的7种报文类型是什么？
8. “SNMP引擎”的含义是什么？

9.4节

9. ASN.1对象标识树的目的是什么？
10. 在ISO/OSI参考模型的表示层中，ASN.1的作用是什么？
11. 因特网具有表示层吗？如果没有，如何关注所涉及的机器体系结构的差异（例如，在不同机器上整数的不同标识）？
12. TLV编码意味着什么？

习题

1. 考虑在管理实体和被管设备之间发生通信的两种方式：请求响应方式和陷阱方式。从以下方面考虑这两种方式的优缺点：①开销，②当异常事件出现时通知的时间，③有关管理实体和设备之间丢失报文的健壮性。
2. 在9.3节中我们看到，用不可靠的UDP数据报传输SNMP报文是更可取的方式。请你阐述SNMP设计者选择UDP而不是TCP作为SNMP运输协议的理由。
3. ICMP协议的ASN.1对象标识符是什么（参见图9-3）？
4. 假设你在为一家美国公司工作，希望开发它自己的MIB来管理生产线。它将在对象标识树（图9-3）的什么地方注册？（提示：为了回答这个问题，你需要钻研RFC或其他文档。）
5. 9.3.2节讲过，一家私营公司（企业）可以在专用分支1.3.6.1.4.1下面创建自己的MIB变量。假定Netscape公司要为它的Web服务器软件创建一个MIB。在1.3.6.1.4.1后面的下一个OID 限定词是什么？（为了回答这个问题，你需要参考[IANA 2007b]。）搜索Web，看是否能找到这样一个用于Netscape服务器的MIB。
6. 考虑图9-9。{weight, 271}{lastname, "Julia"}的BER编码是什么？
7. 考虑图9-9。{weight, 160}{lastname, "child"}的BER编码是什么？

讨论题

1. 除了发电厂或飞机座舱外，举出一个与需要控制的复杂分布式系统类比的例子。
2. 考虑在9.1节激励我们学习网络管理的场景。你认为网络管理员可能需要监视什么样的其他活动？为什么？
3. 阅读RFC 789。如果ARPAnet的管理员具有今天的网络管理工具的话，1980年的ARPAnet崩溃将可能避免（或简单地进行恢复）吗？

人物专访

Jeff Case是SNMP研究公司（SNMP Research, Inc.）的创立者和首席技术官。SNMP（简单网络管理协议）公司是网络管理的因特网标准和基于标准的产品的主流生产商。Jeff在美国普度大学获得双学士学位（产业教育和电气工程技术）和两个硕士学位（产业教育和电气工程）。他在伊利诺伊大学厄巴纳—尚佩恩分校获得了技术教育博士学位。

Jeff Case

• 您为什么决定专门研究网络？

当我是个蹒跚学步的孩子时，就对将东西连到一起着迷，甚至将发针插入电源输出口。在我的青少年时代，发展到对音频装置感兴趣，制作的摇滚乐队音响系统似乎效果强得能将混凝土变成粉末。当我在大学作为一名电视和无线电修理工（大多数是音响设备）时，我对计算机故障略知一二，对数字化的一切感兴趣，包括计算机硬件和计算机软件。我开始对古怪材料与其他古怪材料的接口感兴趣。首先，是与处理器接口的外部设备，后来是系统与系统的接口。网络是最终的接口。对今天而言，因特网是最终的网络。

• 您在计算机行业的第一份工作是什么？它对您有何启发？

我早期的大部分时间是在普度大学度过的。在不同的时期，我教了电气和计算机工程技术专业的大学生课表上的几乎每门课，包括创建有关微处理器硬件和软件的正在出现的主题的新课程。我们一个学期的课程是用芯片设计计算机，在该课程的实验阶段来构建它，一个组做CPU，一个组做内存子系统，一个组做I/O子系统等。在接下来的一个学期中，我们为建造的硬件编写系统软件。

与此同时，我开始在校园范围的计算技术中担任领导角色，最后报告给大学校长，成为学术计算机用户服务中心的主任。

• 您工作中最有挑战性的东西是什么？

面对到目前为止的所有变化，有技术和商务两方面的。我是一个技术性的管理者，在我们的行业中保持技术的先进性越来越困难。我的任务也要求我跟踪商务领域的变化，例如合并和收购。

• 对于网络和因特网的未来，您的预见是什么？

更……更……更：更快的速率；应用范围更普及；更丰富的内容；在无政府和控制管理方式之间将会有更为紧张的气氛；更多的垃圾，更多的清除垃圾的方法；更多的安全性问题，更多的安全性解决方案。最后，我们将期待更多的无法预料的事情发生。

• 哪些人激励了您的专业生涯？

我的继父Dilbert，他是个成功的商人；Vint Cerf博士、Jon Postel博士、Marshall Rose博士和Chuck Davin，他们是因特网业界著名的人物；Bill Seifer，现在是VC合伙人；Rupert Evans博士，我的学位论文的教授；我的妻子，她和我一同打点生意；最后但不是最不重要的Jesus。

• 我阅读过您的一个引人注目的“格言”集。当您成为一个计算机科学教授时，您会有一些格言提供给学生们吗？

“一个实例胜过两本书。”（我想这是Gauss讲过的话。）

“有时理论和实践之间存在差距。理论和理论中的实践之间的差距不如理论和实践中的实践之间的差距那样大。”（我不知道这句话来自何处。）

• 在建立因特网标准时的最大障碍是什么？

金钱，政治，利己主义，缺乏领导。

• 使用SNMP技术最为令人惊奇的是什么？

所有的一切。我实际上涉足因特网管理是为了满足我自己的生存需求。我需要有某些合适的工具来管理我公司的网络基础设施。许多其他人解决类似问题的广泛成功，源于偶然发现、幸运和大量艰苦的工作。更重要的是，我们先前设计的体系结构极为成功。

参考文献

有关URL的注解。在下面的参考文献中，我们提供了Web网页的URL，但仅限于Web的文档以及没有被会议或杂志出版的其他材料（如果我们能够确定这些材料的URL的话）。与本书前几个版本不同的是，我们没有提供会议和杂志文档的URL，因为这些文档通常能够通过如下方式找到：使用某个搜索引擎，从该会议的Web站点上（例如，所有ACM Sigcomm会议和讨论会中的文章可以通过http://www.acm.org/sigcomm找到），或订阅数字图书馆。尽管到2006年12月，下面提供的所有URL都是有效的，但这些URL会因过期而不可用。对于过期的文献，请参考本书的在线版本。

有关请求评论（RFC）的注释。因特网RFC的拷贝在多个网站上都可使用。因特网学会（管理RFC文档的组织）的RFC编辑们维护着网站http://www.rfc-editor.org。该网站允许你通过题目、编号或作者来搜索某个特定的RFC文档，并将显示出对所列RFC的更新。因特网RFC可以被后继的RFC更新或废弃。我们喜欢的获取RFC文档的网站是初始的RFC源，即http://www.rfc-editor.org。

[3Com Addressing 2007] 3Com Corp., "White paper: Understanding IP addressing: Everything you ever wanted to know," http://www.3com.com/other/pdfs/infra/corpinfo/en_US/501302.pdf

[3GPP 2007] Third Generation Partnership Project homepage, http://www.3gpp.org/

[802.11 Security 2007] The Unofficial 802.11 Security Web Page, http://www.drizzle.com/~aboba/IEEE/

[Abitz 1993] P. Albitz and C. Liu, *DNS and BIND*, O'Reilly & Associates, Petaluma, CA, 1993.

[Abramson 1970] N. Abramson, "The Aloha System—Another Alternative for Computer Communications," *Proc. 1970 Fall Joint Computer Conference, AFIPS Conference*, p. 37, 1970.

[Abramson 1985] N. Abramson, "Development of the Alohanet," *IEEE Transactions on Information Theory*, Vol. IT-31, No. 3 (Mar. 1985), pp. 119–123.

[Ahn 1995] J. S. Ahn, P. B. Danzig, Z. Liu, and Y. Yan, "Experience with TCP Vegas: Emulation and Experiment," *Proc. 1995 ACM SIGCOMM* (Boston, MA, Aug. 1995), pp. 185–195.

[Akamai 2007] Akamai homepage, http://www.akamai.com.

[Akella 2003] A. Akella, S. Seshan, A. Shaikh; "An empirical Evaluation of Wide-area Internet Bottlenecks," *Proc. 2003 ACM Internet Measurement Conf.* (Miami FL, Nov. 2003).

[Alvestrand 1997] H. Alvestrand, "Object Identifier Registry," http://www.alvestrand.no/harald/objectid/top.html

[Anderson 1995] J. B. Andersen, T. S. Rappaport, S. Yoshida, "Propagation Measurements and Models for Wireless Communications Channels," *IEEE Communications Magazine*, (Jan. 1995), pp. 42–49.

[Appenzeller 2004] G. Appenzeller, I. Keslassy, N. McKeown, "Sizing Router Buffers,"

Proc. 2004 ACM SIGCOMM (Portland, OR, Aug. 2004).

[Aprisma 2007] Aprisma homepage, http://www.aprisma.com/

[ARIN 1996] ARIN, "IP allocation report," ftp://rs.arin.net/netinfo/ip_network_allocations

[Ash 1998] G. R. Ash, *Dynamic Routing in Telecommunications Networks*, McGraw Hill, NY, NY, 1998.

[ASO-ICANN 2007] The Address Supporting Organization home page, http://www.aso.icann.org

[AT&T SLM 2006] AT&T Business, "AT&T Enterprise Hosting Services Service Guide," http://www.att.com/abs/serviceguide/docs/eh_sg.pdf

[Atheros 2006] Atheros Communications Inc. "Atheros AR5006 WLAN Chipset Product Bulletins," http://www.atheros.com/pt/AR5006Bulletins.htm

[Ayanoglu 1995] E. Ayanoglu, S. Paul, T. F. La Porta, K. K. Sabnani, R. D. Gitlin, "AIRMAIL: A Link-Layer Protocol for Wireless Networks," *ACM ACM/Baltzer Wireless Networks Journal*, 1: 47–60, Feb. 1995.

[Bakre 1995] A. Bakre, B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *Proc. 1995 Int. Conf. on Distributed Computing Systems (ICDCS)*, May 1995, pp. 136–143.

[Balakrishnan 1997] H. Balakrishnan, V. Padmanabhan, S. Seshan, R. Katz, "A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links," *IEEE/ACM Transactions on Networking* Vol. 5, No. 6 (Dec. 1997),

[Baran 1964] P. Baran, "On Distributed Communication Networks," *IEEE Transactions on Communication Systems*, Mar. 1964. Rand Corporation Technical report with the same title (Memorandum RM-3420-PR, 1964). http://www.rand.org/publications/RM/RM3420/

[Bardwell 2007] J. Bardwell, "You Believe You Understand What You Think I Said… The Truth About 802.11 Signal And Noise Metrics: A Discussion Clarifying Often-Misused 802.11 WLAN Terminologies," http://madwifi.org/attachment/wiki/UserDocs/RSSI/you_believe_D100201.pdf?format=raw

[Baset 2006] S. A. Basset and H. Schulzrinne, "An analysis of the Skype peer-to-peer Internet Telephony Protocol," *Proc. 2006 IEEE Infocom* (Barcelona, Spain, Apr. 2006).

[BBC 2001] BBC news online "A Small Slice of Design," Apr. 2001, http://news.bbc.co.uk/2/hi/science/nature/1264205.stm

[BBC Multicast 2007] BB, "BBC Multicast Trial" http://support.bbc.co.uk/multicast/

[Bender 2000] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, A. Viterbi, "CDMA/HDR: A bandwidth-efficient high-speed wireless data service for nomadic users," *IEEE Commun. Mag.*, Vol.38, No. 7 (July 2000) pp.70–77.

[Berners-Lee 1989] T. Berners-Lee, CERN, "Information Management: A Proposal," Mar. 1989, May 1990. http://www.w3.org/History/1989/proposal.html

[Berners-Lee 1994] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyk Nielsen, A. Secret, "The World-Wide Web," *Communications of the ACM*, Vol. 37, No. 8 (Aug. 1994), pp. 76–82.

[Bernstein 2007] D. Bernstein, "SYN Cookies," http://cr.yp.to/syncookies.html

[Bertsekas 1991] D. Bertsekas, R. Gallagher, *Data Networks, 2nd Ed.*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[Bhimani 1996] A. Bhimani: "Securing the Commercial Internet," *Communications of the ACM*, Vol. 39 No. 6 (Mar. 1996), pp. 29–35.

[Biddle 2003] P. Biddle, P. England, M. Peinado, B. Willman, "The Darknet and the Future of Content Distribution," *2002 ACM Workshop on Digital Rights Management*, (Nov. 2002, Washington, D.C.) http://crypto.stanford.edu/DRM2002/darknet5.doc

[Biersack 1992] E. W. Biersack, "Performance evaluation of forward error correction in ATM networks," *Proc. 1999 ACM SIGCOMM* (Baltimore, MD, Aug. 1992), pp. 248–257.

[BIND 2007] Internet Software Consortium page on BIND, http://www.isc.org/bind.html

[Bisdikian 2001] C. Bisdikian, "An Overview of the Bluetooth Wireless Technology," *IEEE Communications Magazine*, No. 12 (Dec. 2001), pp. 86–94.

[Bishop 2003] M. Bishop, *Computer Security: Art and Science*, Boston: Addison Wesley, Boston MA, 2003

[BitTorrent 2007] BitTorrent.org homepage, http://www.bittorrent.org

[Black 1995] U. Black, *ATM Volume I: Foundation for Broadband Networks*, Prentice Hall, 1995.

[Black 1997] U. Black, ATM, Volume II: Signaling in Broadband Networks, Prentice Hall, 1997.

[Blumenthal 2001] M. Blumenthal, D. Clark, "Rethinking the Design of the Internet: the End-to-end Arguments vs. the Brave New World," *ACM Transactions on Internet Technology*, Vol. 1, No. 1 (August 2001) pp. 70–109.

[Bochman 1984] G. V. Bochmann, C. A. Sunshine, "Formal methods in communication protocol design," *IEEE Transactions on Communications*, Vol. 28, No. 4 (Apr. 1980), pp. 624–631.

[Bolot 1994] J-C. Bolot, T. Turletti, "A rate control scheme for packet video in the Internet," *Proc. 1994 IEEE Infocom*, pp. 1216–1223.

[Bolot 1996] J-C. Bolot, A. Vega-Garcia, "Control Mechanisms for Packet Audio in the Internet," *Proc. 1996 IEEE Infocom*, pp. 232–239. ftp://ftp-sop.inria.fr/rodeo/bolot/96.Audio_ctl.ps.gz

[Bradner 1996] S. Bradner, A. Mankin, *IPng: Internet Protocol Next Generation*, Addison-Wesley, Reading, MA, 1996.

[Brakmo 1995] L. Brakmo, L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal of Selected Areas in Communications*, Vol. 13, No. 8, pp. 1465–1480, Oct. 1995.

[Breslau 2000] L. Breslau, E. Knightly, S. Shenker, I. Stoica, H. Zhang, "Endpoint Admission Control: Architectural Issues and Performance," *Proc. 2000 ACM SIGCOMM* (Stockholm, Sweden, Aug. 2000).

[Brodnik 1997] A. Brodnik, S. Carlsson, M. Degemark, S. Pink, "Small Forwarding Tables for Fast Routing Lookups," *Proc. 1997 ACM SIGCOMM* (Cannes, France, Oct. 1997), pp. 3–15.

[Bush 1945] V. Bush, "As We May Think," *The Atlantic Monthly*, July 1945. http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm

[Byers 1998] J. Byers, M. Luby, M. Mitzenmacher, A Rege, "A digital fountain approach to reliable distribution of bulk data," *Proc. 1998 ACM SIGCOMM* (Vancouver, Canada, Aug. 1998), pp. 56–67.

[Cablelabs 2007] CableLabs homepage, http://www.cablelabs.com

[CacheLogic 2007] CacheLogic homepage, http://www.cachelogic.com

[Caesar 2005] M. Caesar, J Rexford, "BGP Routing Policies in ISP Networks," *IEEE Network Magazine*, vol. 19, no. 6 (Nov. 2005).

[Caldwell 2007] C. Caldwell, "The Prime Pages," http://www.utm.edu/research/primes/prove

[Cardwell 2000] N. Cardwell, S. Savage, T. Anderson, "Modeling TCP Latency," *Proc. 2000 IEEE Infocom*, (Tel-Aviv, Israel, Mar. 2000).

[CASA 2007] Center for Collaborative Adaptive Sensing of the Atmosphere, http://www.casa.umass.edu

[**Casner 1992**] **S.** Casner, S. Deering, "First IETF Internet Audiocast," *ACM SIGCOMM Computer Communications Review,* Vol. 22, No. 3 (July 1992), pp. 92–97.

[**Ceiva 2007**] Ceiva homepage, http://www.ceiva.com/

[**CENS 2007**] Center for Embedded Network Sensing, http://www.cens.ucla.edu/

[**Cerf 1974**] V. Cerf and R. Kahn, "A Protocol for Packet Network Interconnection," *IEEE Transactions on Communications Technology,* Vol. COM-22, No. 5, pp. 627–641.

[**CERT 2001–09**] CERT, "Advisory 2001–09: Statistical Weaknesses in TCP/IP Initial Sequence Numbers," http://www.cert.org/advisories/CA-2001-09.html

[**CERT 2003–04**] CERT, "CERT Advisory CA-2003-04 MS-SQL Server Worm," http://www.cert.org/advisories/CA-2003-04.html

[**CERT 2007**] CERT Coordination Center, http://www.cert.org/advisories

[**CERT Filtering 2007**] CERT, "Packet Filtering for Firewall Systems," http://www.cert.org/tech_tips/packet_filtering.html

[**Cert SYN 1996**] CERT, "Advisory CA-96.21: TCP SYN Flooding and IP Spoofing Attacks," http://www.cert.org/advisories/CA-1998-01.html

[**Chao 2001**] H. J. Chao, C. Lam, E. Oki, Broadband Packet Switching Technologies—A Practical Guide to ATM Switches and IP Routers, John Wiley & Sons, 2001.

[**Chen 2000**] G. Chen, D. Kotz, "A Survey of Context-Aware Mobile Computing Research," *Technical Report TR2000-381*, Dept. of Computer Science, Dartmouth College, Nov. 2000. http://www.cs.dartmouth.edu/~dfk/papers/chen:survey-tr.pdf

[**Chen 2006**] K.-T. Chen, C.-Y. Huang, P. Huang, C.-L. Lei, "Quantifying Skype User Satisfaction," *Proc. 2006 ACM SIGCOMM* (Pisa, Italy, Sept. 2006).

[**Cheswick 2000**] B. Cheswick, H. Burch, S. Branigan, "Mapping and Visualizing the Internet," *Proc. 2000 Usenix Conference* (June 2000, San Diego)

[**Chiu 1989**] D. Chiu, R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," *Computer Networks and ISDN Systems,* Vol. 17, No. 1, pp. 1–14. http://www.cis.ohio-state.edu/~jain/papers/cong_av.htm

[**Christiansen 2001**] M. Christiansen, K. Jeffay, D. Ott, F. D. Smith, "Tuning Red for Web Traffic," *IEEE/ACM Transactions on Networking*, Vol. 9, No. 3 (June 2001), pp. 249–264.

[**Chuang 2005**] S. Chuang, S. Iyer, N. McKeown, "Practical Algorithms for Performance Guarantees in Buffered Crossbars," *Proc. 2005 IEEE Infocom.*

[**Cicconetti**] C. Cicconetti, L. Lenzini, A. Mingozi, K. Eklund, "Quality of Service Support in 802.16 Networks," *IEEE Network Magazine,* Mar./Apr. 2006, pp. 50–55.

[**Cisco 12000 2007**] Cisco Systems Inc., "Cisco XR 12000 Series and Cisco 12000 Series Routers," http://www.cisco.com/en/US/products/hw/routers/ps167/products_data_sheet0900aecd8027c8dd.html

[**Cisco 8500 2007**] Cisco Systems Inc., "Catalyst 8500 Campus Switch Router Architecture," http://www.cisco.com/univercd/cc/td/doc/product/l3sw/8540/rel_12_0/w5_6f/softcnfg/1cfg8500.pdf

[**Cisco NAT 2007**] Cisco Systems Inc., "How NAT Works," http://www.cisco.com/warp/public/556/nat-cisco.shtml

[**Cisco NAPA 2007**] Cisco Systems Inc., "Cisco Network Application Performance Analysis (NAPA) Solution," http://www.cisco.com/en/US/products/sw/netmgtsw/index.html

[**Cisco QoS 2007**] Cisco Systems Inc., "Advanced QoS Services for the Intelligent Internet," http://www.cisco.com/warp/public/cc/pd/iosw/ioft/ioqo/tech/qos_wp.htm

[Cisco Queue 2007] Cisco Systems Inc., "Interface Queue Management," http://www.cisco.com/warp/public/614/16.html

[Cisco Security 2007] Cisco Systems Inc., "Why You Need a Firewall," http://www.cisco.com/en/US/products/sw/secursw/ps743/products_user_guide_ chapter09186a008007f303.html

[Cisco Switches 2007] Cisco Systems Inc, "Cisco Catalyst 1900/2820—Affordable Switching Solutions" http://www.cisco.com/warp/public/cc/pd/si/index.shtml

[Cisco SYN 2007] Cisco Systems Inc., "Defining Strategies to Protect Against TCP SYN Denial of Service Attacks," http://cio.cisco.com/warp/public/707/4.html#tcpsyn

[Clark 1988] D. Clark, "The Design Philosophy of the DARPA Internet Protocols," *Proc. 1988 ACM SIGCOMM* (Stanford, CA, Aug. 1988), http://www.acm.org/sigcomm/ccr/archive/1995/jan95/ccr-9501-clark.html

[Clarke 2002] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, B. Wiley, "Protecting Free Expression Online with Freenet," *IEEE Internet Computing*, Jan.–Feb. 2002, pp. 40–49.

[Cohen 1977] D. Cohen, "Issues in Transnet Packetized Voice Communication," *Proc. Fifth Data Communications Symposium*, (Snowbird, Utah, Sept. 1977) pp. 6–13.

[Cohen 2003] B. Cohen, "Incentives to Build Robustness in BitTorrent," *First Workshop on the Economics of Peer-to-Peer Systems,* Berkeley, CA, June 2003.

[Cookie Central 2007] Cookie Central homepage, http://www.cookiecentral.com

[CoolStreaming 2005] X. Zhang, J. Liu, J., B. Li, and T.-S. P. Yum, "CoolStreamingDONet/: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming," *Proc. IEEE INFOCOM*, (March 2005, Miami FL).

[Cormen 2001] T. H. Cormen, *Introduction to Algorithms, 2nd Ed.*, MIT Press, Cambridge, MA, 2001.

[Crow 1997] B. Crow, I. Widjaja, J. Kim, P. Sakai, "IEEE 802.11 Wireless Local Area Networks," *IEEE Communications Magazine*, Sept. 1997, pp. 116–126.

[Crowcroft 1995] J. Crowcroft, Z. Wang, A. Smith, J. Adams, "A Comparison of the IETF and ATM Service Models," *IEEE Communications Magazine*, Nov./ Dec. 1995, pp. 12–16.

[Crowcroft 1999] J. Crowcroft, M. Handley, I. Wakeman, *Internetworking Multimedia*, Morgan-Kaufman, San Francisco, 1999.

[Culler 2004] D. Culler, D. Estrin, M. Srivastava, "Overview of Sensor Networks," *IEEE Computer*, Vol. 37 No. 8, pp. 41–49, Aug. 2004.

[Cusumano 1998] M. A. Cusumano, D. B. Yoffie, *Competing on Internet Time: Lessons from Netscape and its Battle with Microsoft*, Free Press, NY, NY, 1998.

[Daigle 1991] J. N. Daigle, *Queuing Theory for Telecommunications*, Addison-Wesley, Reading, MA, 1991.

[Dalal 1978] Y. Dalal, R. Metcalfe, "Reverse Path Forwarding of Broadcast Packets," *Communications of the ACM*, Vol. 21, No. 12, (Dec. 1978), pp. 1040–1048.

[Davie 2000] B. Davie and Y. Rekhter, *MPLS: Technology and Applications*, Morgan Kaufmann Series in Networking, 2000.

[Davies 2004] G. Davies, F. Kelly, "Network Dimensioning, Service Costing, and Pricing in a Packet-switched Environment," *Telecommunications Policy*, Vol. 28 (no. 4), pp. 391–412.

[DEC 1990] Digital Equipment Corporation, "In Memoriam: J. C. R. Licklider 1915–1990," SRC Research Report 61, Aug. 1990. http://www.memex.org/licklider.pdf

[DeClercq 2002] J. DeClercq, O. Paridaens, "Scalability Implications of Virtual Private Networks," *IEEE Communications Magazine*, Vol. 40 No. 5 (May 2002), pp. 151–157.

[Demers 1990] A. Demers, S. Keshav, S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," *Internetworking: Research and Experience*, Vol. 1, No. 1, 1990, pp. 3–26.

[Denning 1997] D. Denning (Editor), P. Denning (Preface), *Internet Besieged: Countering Cyberspace Scofflaws*, Addison-Wesley, Reading, MA, 1997.

[dhc 2007] IETF Dynamic Host Configuration working group homepage, http://www.ietf.org/html.charters/dhc-charter.html

[Diffie 1998]W. Diffie, S. Landau, *Privacy on the Line, the Politics of Wiretapping and Encryption,* MIT Press, Cambridge MA, 1998.

[Diggavi 2004] S. N. Diggavi, N. Al-Dhahir, A. Stamoulis, R. Calderbank, "Great Expectations: The Value of Spatial Diversity in Wireless Networks," *Proceedings of the IEEE,* vol. 92, no. 2, Feb 2004.

[Diot 2000] C. Diot, B. N. Levine, B. Lyles, H. Kassem, D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network,* Vol. 14, No. 1 (Jan./Feb. 2000), pp. 78–88.

[Dodge 2007] M. Dodge, "An Atlas of Cyberspaces," http://www.cybergeography.org/atlas/isp_maps.html

[Donahoo 2001] M. Donahoo, K. Calvert, *TCP/IP Sockets in C: Practical Guide for Programmers*, Morgan Kaufman, 2001.

[Doucer 2002] J. R. Douceur, "The Sybil Attack," *First International Workshop on Peer-to-Peer Systems (IPTPS '02)* (Cambridge, MA, Mar. 2002).

[Droms 1999] R. Droms, T. Lemon, *The DHCP Handbook*, Macmillan Technical Publishing, Indianapolis, IN, 1999.

[DSL 2007] DSL Forum homepage, http://www.dslforum.org/

[Edney 2003] J. Edney and W. A. Arbaugh, *Real 802.11 Security: Wi-Fi Protected Access and 802.11i,* Addison-Wesley Professional, 2003.

[Eklund 2002] K. Eklund, R. Marks, K. Stanswood, S. Wang, "IEEE Standard 802.16: A Technical Overview of the Wireless MAN Air Interface for Broadband Wireless Access," *IEEE Communications Magazine,* June 2002, pp. 98–107.

[Ellis 1987] H. Ellis, "The Story of Non-Secret Encryption," http://www.cesg.gov.uk/site/publications/media/ellis.pdf

[Ericsson 2007] Ericsson, "EDGE: Introduction of High-Speed Data in GSM/GPRS Networks." http://www.ericsson.com/products/white_papers_pdf/edge_wp_technical.pdf

[ESM 2007] End System Multicast homepage, http://esm.cs.cmu.edu/

[Estrin 1997] D. Estrin, M. Handley, A. Helmy, P. Huang, D. Thaler, "A Dynamic Bootstrap Mechanism for Rendezvous-based Multicast Routing," *Proceedings of IEEE Infocom '98,* (New York, NY, April 1998).

[Ethereal 2007] Ethereal homepage, http://www.ethereal.com

[Faloutsos 1999] C. Faloutsos, M. Faloutsos, P. Faloutsos, "What Does the Internet Look Like? Empirical Laws of the Internet Topology," *Proc. 1999 ACM SIGCOMM* (Boston, MA, Aug. 1999).

[Feamster 2004] N. Feamster, J. Winick, J. Rexford, "A Model for BGP Routing for Network Engineering," *Proc. 2004 ACM SIGMETRICS,* (NY, NY, June 2004).

[Feldmeier 1988] D. Feldmeier, "Improving Gateway Performance with a Routing Table Cache," *Proc. 1988 IEEE Infocom* (New Orleans LA, Mar. 1988).

[Feldmeier 1995] D. Feldmeier, "Fast Software Implementation of Error Detection Codes," *IEEE/ACM Transactions on Networking,* Vol. 3, No. 6 (Dec. 1995), pp. 640–652.

[FIPS 1995] Federal Information Processing Standard, "Secure Hash Standard," FIPS Publication 180-1. http://www.itl.nist.gov/fipspubs/fip180-1.htm

[Floyd 1999] S. Floyd, K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking,* Vol. 6, No. 5 (Oct. 1998), pp. 458–472.

[Floyd 2000] S. Floyd, M. Handley, J. Padhye, J. Widmer, "Equation-Based Congestion Control for Unicast Applications," *Proc. 2000 ACM SIGCOMM* (Stockholm, Sweden, Aug. 2000).

[Floyd 2001] S. Floyd, "A Report on Some Recent Developments in TCP Congestion Control," *IEEE Communications Magazine* (Apr. 2001),

[Floyd 2007] S. Floyd, "References on RED (Random Early Detection) Queue Management," http://www.icir.org/floyd/red.html

[Floyd Synchronization 1994] S. Floyd, V. Jacobson, "Synchronization of Periodic Routing Messages," *IEEE/ACM Transactions on Networking*, Vol. 2, No. 2 (Apr. 1997), pp. 122–136.

[Floyd TCP 1994] S. Floyd, "TCP and Explicit Congestion Notification," *ACM SIGCOMM Computer Communications Review*, Vol. 24, No. 5, pp. 10–23, Oct. 1994.

[Fluhrer 2001] S. Fluhrer, I. Mantin, A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," *Eighth Annual Workshop on Selected Areas in Cryptography*, (Toronto, Canada, Aug. 2002).

[Fortz 2000] B. Fortz, M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," *Proc. 2000 IEEE Infocom.* (Tel Aviv, Israel, Apr. 2000).

[Fortz 2002] B. Fortz, J. Rexford, M. Thorup, "Traffic Engineering with Traditional IP Routing Protocols," *IEEE Communication Magazine,* Oct. 2002.

[Foster 2002] I. Foster, "The Grid: A New Infrastructure for 21st Century Science," *Physics Today*, 55(2):42–47, 2002.

[Fraleigh 2003] C. Fraleigh, T. Tobagi, C. Diot, "Provisioning IP backbone Networks to Support Latency Sensitive Traffic," *Proc. IEEE Infocom Conference*, (San Francisco, March 2003).

[France Telecom 2006] Object Identifier (OID) repository, http://asn1.elibel.tm.fr/oid/

[Fraleigh 2003] C. Fraleigh, F. Tobagi, C. Diot, "Provisioning IP Backbone Networks to Support Latency Sensitive Traffic," *Proc. 2003 IEEE Infocom* (San Francisco, CA, Mar. 2003).

[Friedman 1999] T. Friedman, D. Towsley "Multicast Session Membership Size Estimation," *Proc. 1999 IEEE Infocom* (New York, USA, Mar. 1999)

[Frost 1994] J. Frost, "BSD Sockets: A Quick and Dirty Primer," http://world.std.com/~jimf/papers/sockets/sockets.html

[Gallagher 1983] R. G. Gallagher, P. A. Humblet, P. M. Spira, "A Distributed Algorithm for Minimum Weight-Spanning Trees," *ACM Trans. on Programming Languages and Systems*, 1(5), (Jan. 1983), pp. 66–77.

[Gao 2001] L. Gao, J. Rexford, "Stable Internet Routing Without Global Coordination," *IEEE/ACM Trans. Networking*, Vol. 9, No. 6 (Dec. 2001), pp. 681–692.

[Garces-Erce 2003] L. Garces-Erce, K. W. Ross, E. Biersack, P. Felber, G. Urvoy-Keller, "TOPLUS: Topology Centric Lookup Service," *Fifth Int. Workshop on Networked Group Communications (NGC 2003)*, (Munich, Sept. 2003) http://cis.poly.edu/~ross/papers/TOPLUS.pdf

[Gartner 2003] F. C. Gartner, "A Survey of Self-Stabilizing Spanning-Tree Construction Algorithms," *Technical Report IC/2003/38*, Swiss Federal Institute of Technology (EPFL), School of Computer and Communication Sciences, June 10, 2003. http://ic2.epfl.ch/publications/documents/IC_TECH_REPORT_200338.pdf.

[Gauthier 1999] L. Gauthier, C. Diot, and J. Kurose, "End-to-end Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet," *Proc. 1999 IEEE Infocom* (New York, NY, Apr. 1999).

[Giacopelli 1990] J. Giacopelli, M. Littlewood, W. D. Sincoskie "Sunshine: A high performance self-routing broadband packet switch architecture," *1990 International Switching Symposium.* An extended version of this paper appeared in *IEEE J. Selected. Areas in Communications*, Vol. 9, No. 8 (Oct. 1991), pp. 1289–1298.

[Gill 2005] V. Gill, "Abstract: Design Decisions and Architecture Analysis of a Global 10G Backbone," *NANOG 34*, http://www.nanog.org/mtg-0505/gill.html

[Girard 1990] A. Girard, *Routing and Dimensioning in Circuit-Switched Networks,* Addison-Wesley, Reading, MA, 1990.

[Glitho 1995] R. Glitho, S. Hayes (eds.), special issue on Telecommunications Management Network, *IEEE Communications Magazine*, Vol. 33, No. 3 (Mar. 1995).

[Glitho 1998] R. Glitho, "Contrasting OSI Systems Management to SNMP and TMN," *Journal of Network and Systems Management,* Vol. 6, No. 2 (June 1998), pp. 113–131.

[Gnutella 2007] "The Gnutella Protocol Specification, v0.4" http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

[Goodman 1997] David J. Goodman, *Wireless Personal Communications Systems*, Prentice-Hall, 1997.

[Goralski 1999] W. Goralski, *Frame Relay for High-Speed Networks,* John Wiley, New York, 1999.

[Goralski 2001] W. Goralski, *Optical Networking and WDM,* Osborne/McGraw-Hill, Berkeley, CA, 2001.

[Griffin 2002] T. Griffin, "Interdomain Routing Links," http://www.research.att.com/~griffin/interdomain.html

[Guha 2006] S. Guha, N. Daswani, R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System," *Proc. Fifth Int. Workshop on P2P Systems,* (Santa Barbara, CA, 2006).

[Gupta 1998] P. Gupta, S. Lin, N. McKeown. "Routing lookups in hardware at memory access speeds," *Proc. 1998 IEEE Infocom* (San Francisco, CA, Apr. 1998), pp. 1241–1248.

[Gupta 2001] P. Gupta, N. McKeown, "Algorithms for Packet Classification," *IEEE Network Magazine,* Vol. 15, No. 2 (Mar./Apr. 2001), pp. 24–32.

[Hain 2005] T. Hain, "A Pragmatic Report on IPv4 Address Space Consumption," *Internet Protocol Journal,* Vol. 8, No. 3.

[Halabi 2000] S. Halabi, *Internet Routing Architectures, 2nd Ed.*, Cisco Press, 2000.

[Hamada 1997] T. Hamada, H. Kamata, S. Hogg, "An Overview of the TINA Management Architecture," *Journal of Network and Systems Management,* Vol. 5. No. 4 (Dec. 1997). pp. 411–435.

[Heidemann 1997] J. Heidemann, K. Obraczka, J. Touch, "Modeling the Performance of HTTP over Several Transport Protocols," *IEEE/ACM Transactions on Networking,* Vol. 5, No. 5 (Oct. 1997), pp. 616–630.

[Held 2001] G. Held, *Data Over Wireless Networks: Bluetooth, WAP, and Wireless LANs,* McGraw-Hill, 2001.

[Hersent 2000] O. Hersent, D. Gurle, J-P. Petit, *IP Telephony: Packet-Based Multimedia Communication Systems*, Pearson Education Limited, Edinburgh, 2000.

[Hinden 2007] R. Hinden, "IP Next Generation (IP ng)," http://playground.sun.com/pub/ipng/html/ipng-main.html

[Holland 2001] G. Holland, N. Vaidya, V. Bahl, "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks," *Proc. 2001 ACM Int. Conference of Mobile Computing and Networking (Mobicom01),* (Rome, Italy, July 2001).

[Hollot 2002] C.V. Hollot, V. Misra, D. Towsley, W. Gong, "Analysis and design of controllers for AQM routers supporting TCP flows," *IEEE Transactions on Automatic Control,* Vol. 47, No. 6 (June 2002), pp. 945–959.

[Huang 2002] C. Haung, V. Sharma, K. Owens, V. Makam, "Building Reliable MPLS Networks Using a Path Protection Mechanism," *IEEE Communications Magazine,* Vol. 40, No. 3 (Mar. 2002), pp. 156–162.

[Huang 2005] Y. Huang, R. Guerin, "Does Over-Provisioning Become More or Less Efficient as Networks Grow Larger?," *Proc. IEEE Int. Conf. Network Protocols (ICNP),* (Boston MA, November 2005).

[Huitema 1998] C. Huitema, *IPv6: The New Internet Protocol, 2nd Ed.*, Prentice Hall, Englewood Cliffs, NJ, 1998.

[Huston 1999a] G. Huston, "Interconnection, Peering, and Settlements—Part I," *The Internet Protocol Journal,* Vol. 2, No. 1, (Mar. 1999).

[Huston 2001] G. Huston, "Analyzing the Internet BGP Routing Table," *The Internet Protocol Journal*, Vol. 4, No. 1 (Mar. 2001).

[Huston 2004] G. Huston, "NAT Anatomy: A Look Inside Network Address Translators," *The Internet Protocol Journal,* Volume 7, Number 3 (Sept. 2004).

[IAB 2007] Internet Architecture Board homepage, http://www.iab.org/iab/

[IANA 2007] Internet Assigned Number Authority homepage, http://www.iana.org/

[IANA 2007b] Internet Assigned Number Authority, "Private Enterprise Numbers," http://www.iana.org/assignments/enterprise-numbers

[ICANN 2007] The Internet Corporation for Assigned Names and Numbers homepage, http://www.icann.org

[IEC Optical 2007] IEC Online Education, "Optical Access," http://www.iec.org/online/tutorials/opt_acc/

[IEEE 802 2007] IEEE 802 LAN/MAN Standards Committee homepage, http://www.ieee802.org/

[IEEE 802.11 1999] IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Network—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, http://standards.ieee.org/getieee802/download/802.11-1999.pdf

[IEEE 802.11n] IEEE, "IEEE P802.11—Task Group N—Meeting Update: Status of 802.11n" http://grouper.ieee.org/groups/802/11/Reports/tgn_update.htm

[IEEE 802.15 2007] IEEE 802.15 Working Group for WPAN homepage, http://grouper.ieee.org/groups/802/15/.

[IEEE 802.16d 2004] IEEE, "IEEE Standard for Local and metropolitan area networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems," http://standards.ieee.org/getieee802/download/802.16-2004.pdf

[IEEE 802.16e 2005] IEEE, "IEEE Standard for Local and metropolitan area networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1," http://standards.ieee.org/getieee802/download/ 802.16e-2005.pdf

[IEEE 802.1X] IEEE Std 802.1X-2001 Port-Based Network Access Control, http://standards.ieee.org/reading/ieee/std_public/description/lanman/802.1x-2001_desc.html

[IEEE 802.3 2007] IEEE, "IEEE 802.3 CSMA/CD (Ethernet)," http://grouper.ieee.org/groups/802/3/

[IEEE 802.5 2007] IEEE, IEEE 802.5 homepage, http://www.ieee802.org/5/www8025org/

[IETF 2007] Internet Engineering Task Force homepage, http://www.ietf.org

[IMAP 2007] The IMAP Connection, http://www.imap.org/

[Intel 2006] Intel Corp, "PCI/PCI-X Family of Gigabit Ethernet Controllers Software Developer's Manual," http://download.intel.com/design/network/manuals/ 8254x_GBe_SDM.pdf

[Intel WiMax 2007] Intel Corp., "WiMax Broadband Wireless Technology Access," http://www.intel.com/netcomms/technologies/wimax/

[Interlinknetworks 2004] Interlinknetworks, "Introduction to 802.1x for Wireless Local Area Networks," http://www.interlinknetworks.com/resource/wp5-1-1.htm

[Internet Home Alliance 2007] Internet Home Alliance Research Council homepage, http://www.caba.org/iha/

[Internet2 Multicast 2007] Internet2 Multicast Working Group homepage, http://multicast.internet2.edu/

[ISC 2007] Internet Systems Consortium homepage, http://www.isc.org

[ISI 1979] Information Sciences Institute, "DoD Standard Internet Protocol," Internet Engineering Note 123, Dec. 1979. http://www.isi.edu/in-notes/ien/ien123.txt

[ISO 1987] International Organization for Standardization, "Information processing systems—Open Systems Interconnection—," International Standard 8824 (Dec. 1987). http://asn1.elibel.tm.fr/en/standards/index.htm

[ISO 2007] International Organization for Standardization homepage, International Organization for Standardization, http:// www.iso.org/

[ISO X.680 1998] International Organization for Standardization, "X.680: ITU-T Recommendation X.680 (1997) | ISO/IEC 8824-1:1998, Information Technology—Abstract Syntax Notation One (ASN.1): Specification of Basic Notation." http://asn1.elibel.tm.fr/en/standards/index.htm

[ITU 2005] International Telecommunication Union, *The Internet of Things*, 2005, http://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf

[ITU 2007] The ITU homepage, http://www.itu.int/

[ITU Statistics 2007] International Telecommunications Union, "ICT Statistics," http://www.itu.int/ITU-D/icteye/Reports.aspx

[ITU-T Q.2931 1994] ITU-T, "Broadband Integrated Service Digital Network (B-ISDN) Digital Subscriber Signaling System no.2 (DSS2) User Network Interface Layer 3 Specification for Basic Call/Connection Control," *ITU-T Recommendation Q.2931*, Geneva: International Telecommunication Union, 1994.

[Iyer 2002] S. Iyer, R. Zhang, N. McKeown, "Routers with a Single Stage of Buffering," *Proc. 2002 ACM SIGCOMM* (Pittsburgh, PA, Aug. 2002).

[Jacobson 1988] V. Jacobson, "Congestion Avoidance and Control," *Proc. 1988 ACM SIGCOMM* (Stanford, CA, Aug. 1988), pp. 314–329.

[Jain 1989] R. Jain, "A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks," *ACM SIGCOMM Computer Communications Review*, Vol. 19, No. 5 (1989), pp. 56–71.

[Jain 1994] R. Jain, *FDDI Handbook: High-Speed Networking Using Fiber and Other Media*, Addison-Wesley, Reading, MA, 1994.

[Jain 1996] R. Jain. S. Kalyanaraman, S. Fahmy, R. Goyal, S. Kim, "Tutorial Paper on ABR Source Behavior," *ATM Forum*/96-1270, Oct. 1996. http://www.cis.ohio-state.edu/~jain/atmf/a96-1270.htm

[Jaiswal 2003] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, "Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP backbone," *Proc. 2003 IEEE Infocom.*

[Jakobson 1993] G. Jacobson, M. Weissman, "Alarm Correlation," *IEEE Network Magazine*, 1993, pp. 52–59.

[Ji 2003] P. Ji, Z. Ge, J. Kurose, D. Towsley, "A Comparison of Hard-state and Soft-state Signaling Protocols," *Proc. 2003 ACM SIGCOMM* (Karlsruhe, Germany, Aug. 2003).

[Jiang 2001]W. Jiang, J. Lennox, H. Schulzrinne, K. Singh, "Towards Junking the PBX: Deploying IP Telephony," *NOSSDAV'01* (Port Jefferson, NY, June 2001).

[Jimenez 1997] D. Jimenez, "Outside Hackers Infiltrate MIT Network, Compromise Security," *The Tech*, Vol. 117, No. 49 (Oct. 1997), p. 1. http://www-tech.mit.edu/V117/N49/hackers.49n.html

[Jin 2004] C. Jin, D. X. We, S. Low, "FAST TCP: Motivation, architecture, algorithms, performance," *Proc. 2004 IEEE Infocom*, (Hong Kong, March 2004).

[Kaaranen 2001] H. Kaaranen, S. Naghian, L. Laitinen, A. Ahtiainen, V. Niemi, *Networks: Architecture, Mobility and Services*, New York: John Wiley & Sons, 2001.

[Kahn 1967] D. Kahn, *The Codebreakers: Tthe Story of Secret Writing*, The Macmillan Company, 1967.

[Kahn 1978] R. E. Kahn, S. Gronemeyer, J. Burchfiel, R. Kunzelman, "Advances in Packet Radio Technology," *Proc. of the IEEE*, 66, 11 (Nov. 1978).

[**Kamerman 1997**] A Kamerman, L. Monteban, "WaveLAN-II: A High –Performance Wireless LAN for the Unlicensed Band," *Bell Labs Technical Journal,* Summer 1997, pp. 118–133.

[**Kangasharju 2000**] J. Kangasharju, K. W. Ross, J. W. Roberts, "Performance Evaluation of Redirection Schemes in Content Distribution Networks," *Proc. 5th Web Caching and Content Distribution Workshop*, (Lisbon, Portugal, May 2000).

[**Kar 2000**] K. Kar, M. Kodialam, T. V. Lakshman, "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications," *IEEE J. Selected Areas in Communications*, Dec. 2000.

[**Karol 1987**] M. Karol, M. Hluchyj, A. Morgan, "Input Versus Output Queuing on a Space- Division Packet Switch," *IEEE Transactions on Communications*, Vol. 35, No. 12, (Dec. 1987), pp. 1347–1356.

[**Katabi 2002**] D. Katabi, M. Handley, C. Rohrs, "Internet Congestion Control for Future High Bandwidth-Delay Product Environments," *Proc. 2002 ACM SIGCOMM* (Pittsburgh, PA, Aug. 2002).

[**Katzela 1995**] I. Katzela, M. Schwartz. "Schemes for Fault Identification in Communication Networks," *IEEE/ACM Transactions on Networking*, Vol. 3, No. 6 (Dec. 1995), pp. 753–764.

[**Kaufman 1995**] C. Kaufman, R. Perlman, M. Speciner, *Network Security, Private Communication in a Public World,* Prentice Hall, Englewood Cliffs, NJ, 1995.

[**Kelly 2003**] T. Kelly, "Scalable TCP: improving performance in high speed wide area networks," *ACM SIGCOMM Computer Communications Review*, Volume 33, No. 2 (Apr. 2003), pp 83–91.

[**Keshav 1998**] S. Keshav, R. Sharma, "Issues and Trends in Router Design," *IEEE Communications Magazine,* Vol. 36, No. 5 (May 1998), pp. 144–151.

[**Keslassy 2003**] I. Keslassy, S. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, McKeown, "Scaling Internet Routers Using Optics," *Proc. 2003 ACM SIGCOMM* (Karlsruhe, Germany, Aug. 2003).

[**Kilkki 1999**] K. Kilkki, *Differentiated Services for the Internet*, Macmillan Technical Publishing, Indianapolis, IN, 1999.

[**Kim 2005**] H. Kim, S. Rixner, V. Pai, "Network Interface Data Caching," *IEEE Transactions on Computers*, Volume 54, No. 11, (Nov. 2005), pp. 1394–1408.

[**Kleinrock 1961**] L. Kleinrock, "Information Flow in Large Communication Networks," RLE Quarterly Progress Report, July 1961.

[**Kleinrock 1964**] L. Kleinrock, *1964 Communication Nets: Stochastic Message Flow and Delay,* McGraw-Hill, NY, NY, 1964.

[**Kleinrock 1975**] L. Kleinrock, *Queuing Systems, Vol. 1*, John Wiley, New York, 1975.

[**Kleinrock 1975b**] L. Kleinrock, F. A. Tobagi, "Packet Switching in Radio Channels: Part I—Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics," *IEEE Transactions on Communications*, Vol. 23, No. 12 (Dec. 1975), pp. 1400–1416.

[**Kleinrock 1976**] L. Kleinrock, *Queuing Systems, Vol. 2*, John Wiley, New York, 1976.

[**Kleinrock 2004**] L. Kleinrock, "The Birth of the Internet," http://www.lk.cs.ucla.edu/LK/Inet/birth.html

[**Kohler 2006**] E. Kohler, M. Handley, S. Floyd, "DDCP: Designing DCCP: Congestion Control Without Reliability," *Proc. 2006 ACM SIGCOMM* (Pisa, Italy, Sept. 2006).

[**Korhonen 2003**] J. Korhonen, *Introduction to 3G Mobile Communications*, 2nd ed., Artech House, 2003.

[**Koziol 2003**] J. Koziol, *Intrusion Detection with Snort*, Sams Publishing, 2003.

[**Krishnamurthy 2001**] B. Krishnamurthy, and J. Rexford, *Web Protocols and Practice: HTTP/1.1, Networking Protocols, and Traffic Measurement*, Addison-Wesley, Boston, MA, 2001.

[**Kulkarni 2005**] S. Kulkarni, C. Rosenberg, "Opportunistic Scheduling: Generalizations to Include Multiple Constraints, Multiple Interfaces, and Short Term Fairness," *Wireless*

Networks, 11, 557–569, 2005.

[Kumar 2006] R. Kumar, K.W. Ross, "Optimal Peer-Assisted File Distribution: Single and Multi-Class Problems," *IEEE Workshop on Hot Topics in Web Systems and Technologies*, Boston, 2006.

[Kurose 1996] J. F. Kurose, Unix Network Programming, http://manic.cs.umass.edu/~amldemo/courseware/intro.html

[Labovitz 1997] C. Labovitz, G. R. Malan, F. Jahanian, "Internet Routing Instability," *Proc. 1997 ACM SIGCOMM* (Cannes, France, Sept. 1997), pp. 115–126.

[Labrador 1999] M. Labrador, S. Banerjee, "Packet Dropping Policies for ATM and IP Networks," *IEEE Communications Surveys*, Vol. 2, No. 3 (Third Quarter 1999), pp. 2–14.

[Lacage 2004] M. Lacage, M.H. Manshaei, T. Turletti, "IEEE 802.11 Rate Adaptation: A Practical Approach," *ACM Int. Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM)* (Oct. 2004, Venice, Italy).

[Lakshman 1997] T. V. Lakshman, U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Transactions on Networking*, Vol. 5 No. 3 (1997). pp. 336–350.

[Lam 1980] S. Lam, "A Carrier Sense Multiple Access Protocol for Local Networks," *Computer Networks*, Vol. 4 (1980), pp. 21–32, 1980.

[Lamport 1981] L. Lamport, "Password Authentication with Insecure Communication," *Communications of the ACM*, Vol. 24, No. 11 (Nov. 1981), pp. 770–772.

[Larmouth 1996] J. Larmouth, *Understanding OSI*, International Thomson Computer Press 1996. Chapter 8 of this book deals with ASN.1 and is available on-line at http://www.salford.ac.uk/iti/books/osi/all.html#head8

[Larsen 1997] A. Larsen, "Guaranteed Service: Monitoring Tools," *Data Communications*, June 1997, pp. 85–94.

[Lawton 2001] G. Lawton, "Is IPv6 Finally Gaining Ground?" *IEEE Computer Magazine* (Aug. 2001), pp. 11–15.

[Leiner 1998] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, S. Woolf, "A Brief History of the Internet," http://www.isoc.org/internet/history/brief.html

[Li 2004] L. Li, D. Alderson, W. Willinger, J. Doyle, "A First-Principles Approach to Understanding the Internet's Router-Level Topology," *Proc. 2004 ACM SIGCOMM* (Portland, Oregon, Aug. 2004).

[Liang 2005] J. Liang, R. Kumar, "The Kazaa Overlay: A Measurement Study," *Computer Networks (Special Issue on Overlays)*, 2005.

[Liang 2006] J. Liang, N. Naoumov, K.W. Ross, "The Index Poisoning Attack in P2P File-Sharing Systems," *Proc. 2006 IEEE Infocom 2006* (Barcelona, Spain, April 2006).

[Lin 2001]Y. Lin, I. Chlamtac, *Wireless and Mobile Network Architectures*, John Wiley and Sons, New York, NY, 2001.

[Liu 2002] B. Liu, D. Goeckel, D. Towsley, "TCP-Cognizant Adaptive Forward Error Correction in Wireless Networks," *Proc. 2002 Global Internet.*

[Loh 2006] V. Loh, "Real-World Interoperability Tests of Five 802.11n Routers," http://www.extremetech.com/article2/0,1697,2013303,00.asp

[Lucent 2006] Lucent Technologies, "OSS Software for Lucent Technologies and Juniper Networks Unified Solutions," http://www.lucent.com/solutions/juniper_oss.html

[Lui 2004] J. Lui, V. Misra, D. Rubenstein, "On the Robustness of Soft State Protocols," *Proc. IEEE Int. Conference on Network Protocols (ICNP '04)*, pp. 50–60.

[Luotonen 1998] A. Luotonen, *Web Proxy Servers*, Prentice Hall, Englewood Cliffs, New Jersey, 1998.

[Lynch 1993] D. Lynch, M. Rose, *Internet System Handbook*, Addison-Wesley, Reading, MA, 1993.

[Macedonia 1994] M. Macedonia, D. Brutzman, "MBone Provides Audio and Video Across the Internet," *IEEE Computer Magazine,* Vol. 27, No. 4 (Apr. 1994), pp. 30–36.

[Mahdavi 1997] J. Mahdavi, S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control," unpublished note, Jan. 1997.

[Malware 2006] Computer Economics, "2005 Malware Report: The Impact of Malicious Code Attacks,"http://www.computereconomics.com

[manet 2007] IETF Mobile Ad-hoc Networks (manet) Working Group, http://www.ietf.org/html.charters/manet-charter.html

[Maymounkov 2002] P. Maymounkov, D. Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric." *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, pp. 53–65, Mar. 2002.

[McKeown 1997a] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, M. Horowitz, "The Tiny Tera: A Packet Switch Core," *IEEE Micro Magazine,* Jan.–Feb. 1997.

[McKeown 1997b] N. McKeown, "A Fast Switched Backplane for a Gigabit Switched Router," *Business Communications Review,* Vol. 27, No. 12. http://www.bcr.com/bcrmag/12/mckeown.htm

[McQuillan 1980] J. McQuillan, I. Richer, E. Rosen, "The New Routing Algorithm for the Arpanet," *IEEE Transactions on Communications,* Vol. 28, No. 5 (May 1980), pp. 711–719.

[Medhi 1997] D. Medhi, D. Tipper (eds.), Special Issue: Fault Management in Communication Networks, *Journal of Network and Systems Management,* Vol. 5. No. 2 (June 1997).

[Meng 2005] X. Meng, "IPv4 Address Allocation and the BGP Routing Table Evolution," *Computer Communication Reviews,* Vol. 35, No 1 (2005), pp. 71–80.

[Metcalfe 1976] R. M. Metcalfe, D. R. Boggs. "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the Association for Computing Machinery,* Vol. 19, No. 7, (July 1976), pp. 395–404.

[MFA Forum 2007] MFA Forum homepage, http://www.mfaforum.org/

[Microsoft Player Media 2007] Microsoft Windows Media homepage, http://www.microsoft.com/windows/windowsmedia/

[Miller 1997] M.A. Miller, *Managing Internetworks with SNMP,* 2nd ed., M & T Books, New York, 1997.

[Mirkovic 2005] J. Mirkovic, S. Dietrich, D. Dittrich. P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms*, Prentice Hall, 2005.

[Mockapetris 1988] P. V. Mockapetris, K. J. Dunlap, "Development of the Domain Name System," *Proc. 1988 ACM AIGCOMM* (Stanford, CA, Aug. 1988).

[Mockapetris 2005] P. Mockapetris, Sigcomm Award Lecture, video available at http://www.postel.org/sigcomm

[Mogul 2003] J. Mogul, "TCP offload is a dumb idea whose time has come". *Proc. HotOS IX: The 9th Workshop on Hot Topics in Operating Systems,* (2003) USENIX Association.

[Molinero-Fernandez 2002] P. Molinaro-Fernandez, N. McKeown, H. Zhang, "Is IP Going to Take Over the World (of Communications)?" *Proc. 2002 ACM Hotnets.*

[Molle 1987] M. L. Molle, K. Sohraby, A. N. Venetsanopoulos, "Space-Time Models of Asynchronous CSMA Protocols for Local Area Networks," *IEEE Journal on Selected Areas in Communications,* Vol. 5, No. 6, (1987) pp. 956–968.

[Moore 2001] D. Moore, G. Voelker, S. Savage, "Inferring Internet Denial of Service Activity," *Proc. 2001 USENIX Security Symposium,* (Washington DC, Aug. 2001).

[Moore 2003] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, N. Weaver, "Inside the Slammer Worm," *2003 IEEE Security and Privacy Conference.*

[Moshchuck 2006] A. Moshchuk, T. Bragin, S. Gribble, H. Levy, "A Crawler-based Study of Spyware on the Web," *Proc. 13th Annual Network and Distributed Systems Security Symposium (NDSS 2006),* (San Diego, CA, Feb. 2006).

[Mouly 1992] M. Mouly, M. Pautet, *The GSM System for Mobile Communications*, Cell and Sys, Palaiseau, France, 1992.

[Moy 1998] J. Moy, *OSPF: Anatomy of An Internet Routing Protocol*, Addison-Wesley, Reading, MA, 1998.

[Mukherjee 1997] B. Mukherjee, *Optical Communication Networks*, McGraw-Hill, 1997.

[Murphy 2003] S. Murphy, "BGP Security Vulnerabilities Analysis," draft-ietf-idr-bgp-vuln-00.txt, June 2003, ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-ietf-idr-bgpvuln-00.txt

[Nahum 2002] E. Nahum, T. Barzilai, D. Kandlur, "Performance Issues in WWW Servers," *IEEE/ACM Transactions on Networking*, Vol 10, No. 1 (Feb. 2002).

[Naoumov 2006] N. Naoumov, K.W. Ross, "Exploiting P2P Systems for DDoS Attacks," *Intl Workshop on Peer-to-Peer Information Management*, (Hong Kong, May 2006),

[Neumann 1997] R. Neumann, "Internet Routing Black Hole," *The Risks Digest: Forum on Risks to the Public in Computers and Related Systems*, Vol. 19, No. 12 (May 1997). http://catless.ncl.ac.uk/Risks/19.12.html#subj1.1

[Nicholson 2006] A Nicholson, Y. Chawathe, M. Chen, B. Noble, D. Wetherall, "Improved Access Point Selection," *Proc. 2006 ACM Mobisys Conference*, (Uppsala Sweden, 2006).

[Nielsen 1997] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie, C. Lilley, "Network Performance Effects of HTTP/1.1, CSS1, and PNG," *W3C Document*, 1997 (also appears in *Proc. 1997 ACM SIGCOM*, (Cannes, France, Sept 1997), pp. 155–166.

[NIST 2001] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," Federal Information Processing Standards 197, Nov. 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[Nmap 2004] Nmap homepage, http://www.insecure.com/nmap

[Nonnenmacher 1998] J. Nonnenmacher, E. Biersak, D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 4 (Aug. 1998), pp. 349–361.

[NTIA 1998] National Telecommunications and Information Administration (NTIA), US Department of Commerce, "Management of Internet names and addresses," Docket Number: 980212036-8146-02. http://www.ntia.doc.gov/ntiahome/domainname/6_5_98dns.htm

[Odlyzko 2003] A. Odlyzko, "Internet Traffic Growth: Sources and Implications," A. M. Optical Transmission Systems and Equipment for WDM Networking II, *Proc. SPIE*, 5247, 2003, pp. 1–15. http://www.dtc.umn.edu/~odlyzko/doc/itcom.internet.growth.pdf.

[OpenView2007] HP OpenView homepage, http://www.openview.hp.com/

[OSI 2007] International Organization for Standardization homepage, http://www.iso.org/iso/en/ISOOnline.frontpage

[OSS 2007] OSS Nokalva, "ASN.1 Resources," http://www.oss.com/asn1/

[Padhye 2000] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, "Modeling TCP Reno Performance: A Simple Model and its Empirical Validation," *IEEE/ACM Transactions on Networking*, Vol. 8 No. 2 (Apr. 2000), pp. 133–145.

[Padhye 2001] J. Padhye, S. Floyd, "On Inferring TCP Behavior," *Proc. 2001 ACM SIGCOMM*, (San Diego, CA, Aug. 2001).

[Pan 1997] P. Pan, H. Schulzrinne, "Staged Refresh Timers for RSVP," *Proc. 2nd Global Internet Conference*, (Phoenix, AZ, Dec. 1997).

[Parekh 1993] A. Parekh, R. Gallagher, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 3 (June 1993), pp. 344–357.

[Partridge 1992] C. Partridge, S. Pink, "An Implementation of the Revised Internet Stream Protocol (ST-2)," *Journal of Internetworking: Research and Experience*, Vol. 3, No. 1 (Mar. 1992).

[Partridge 1998] C. Partridge, et al. "A Fifty Gigabit per second IP Router," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 3 (Jun. 1998), pp. 237–248.

[Paxson 1997] V. Paxson, "End-to-end Internet packet dynamics," *Proc. 1997 ACM SIGCOMM* (Cannes, France, Sept 1997).

[Perkins 1994] A. Perkins, "Networking with Bob Metcalfe," *The Red Herring Magazine*, Nov. 1994.

[Perkins 1998] C. Perkins, O. Hodson, V. Hardman, "A Survey of Packet Loss Recovery Techniques for Streaming Audio," *IEEE Network Magazine*, Sept./Oct. 1998, pp. 40–47.

[Perkins 1998b] C. Perkins, *Mobile IP: Design Principles and Practice*, Addison-Wesley, Reading, MA, 1998.

[Perkins 2000] C. Perkins, *Ad Hoc Networking*, Addison-Wesley, Reading, MA, 2000.

[Perlman 1999] R. Perlman, *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*, 2nd ed., Addison-Wesley Professional Computing Series, Reading, MA, 1999.

[PGPI 2007] The International PGP Home Page, http://www.pgpi.org

[Phifer 2000] L. Phifer, "The Trouble with NAT," *The Internet Protocol Journal*, Vol. 3, No. 4 (Dec. 2000), http://www.cisco.com/warp/public/759/ipj_3-4/ipj_3-4_nat.html

[Pickholtz 1982] R. Pickholtz, D. Schilling, L. Milstein, "Theory of Spread Spectrum Communication—a Tutorial," *IEEE Transactions on Communications*, Vol. 30, No. 5 (May 1982), pp. 855–884.

[pingplotter 2007] pingplotter homepage, http://www.pingplotter.com

[Piscatello 1993] D. Piscatello, A. Lyman Chapin, *Open Systems Networking*, Addison-Wesley, Reading, MA, 1993.

[Point Topic 2006] Point Topic Ltd., *World Broadband Statistics Q1 2006*, http://www.point-topic.com

[PPLive 2007] PPLive homepage, http://www.pplive.com

[Primetrica 2007] PriMetrica Inc, "Global Internet Geography 2006," http://www.telegeography.com

[QuickTime 2007] QuickTime homepage, http://www.apple.com/quicktime

[Quittner 1998] J. Quittner, M. Slatalla, *Speeding the Net: The Inside Story of Netscape and How it Challenged Microsoft*, Atlantic Monthly Press, 1998.

[Ramakrishnan 1990] K. K. Ramakrishnan, R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks," *ACM Transactions on Computer Systems*, Vol. 8, No. 2 (May 1990), pp. 158–181.

[Raman 1999] S. Raman, S. McCanne, "A Model, Analysis, and Protocol Framework for Soft State-based Communication," *Proc. 1999 ACM SIGCOMM* (Boston, MA, Aug. 1999).

[Raman 2007] B. Raman, K. Chebrolu, "Experiences in using WiFi for Rural Internet in India," *IEEE Communications Magazine*, Special Issue on New Directions in Networking Technologies in Emerging Economies, Jan 2007.

[Ramaswami 1998] R. Ramaswami, K. Sivarajan, *Optical Networks: A Practical Perspective*, Morgan Kaufman Publishers, 1998.

[Ramjee 1994] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks," *Proc. 1994 IEEE Infocom*.

[Rao 1996] K. R. Rao and J. J. Hwang, *Techniques and Standards for Image, Video and Audio Coding*, Prentice Hall, Englewood Cliffs, NJ, 1996.

[RAT 2007] Robust Audio Tool, http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/

[Ratnasamy 2001] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A Scalable Content-Addressable Network," *Proc. 2001 ACM SIGCOMM*, (San Diego, CA, Aug. 2001).

[RealNetworks 2007] RealNetworks homepage, http://www.realnetworks.com

[Ren 2006] S. Ren, L. Guo, and X. Zhang, "ASAP: an AS-aware peer-relay protocol for high quality VoIP," *Proc. 2006 IEEE ICDCS* (Lisboa, Portugal, July 2006).

[Rescorla 2001] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*, Addison-Wesley, Boston, 2001.

[RFC 001] S. Crocker, "Host Software," RFC 001 (the *very first* RFC!).

[RFC 768] J. Postel, "User Datagram Protocol," RFC 768, Aug. 1980.

[RFC 789] E. Rosen, "Vulnerabilities of Network Control Protocols," RFC 789.

[RFC 791] J. Postel, "Internet Protocol: DARPA Internet Program Protocol Specification," RFC 791, Sept. 1981.

[RFC 792] J. Postel, "Internet Control Message Protocol," RFC 792, Sept. 1981.

[RFC 793] J. Postel, "Transmission Control Protocol," RFC 793, Sept. 1981.

[RFC 801] J. Postel, "NCP/TCP Transition Plan," RFC 801 Nov. 1981.

[RFC 826] D. C. Plummer, "An Ethernet Address Resolution Protocol—or—Converting Network Protocol Addresses to 48 bit Ethernet Address for Transmission on Ethernet Hardware," RFC 826, Nov. 1982.

[RFC 829] V. Cerf, "Packet Satellite Technology Reference Sources," RFC 829, Nov. 1982.

[RFC 854] J. Postel, J. Reynolds, "TELNET Protocol Specification," RFC 854. May 1993.

[RFC 950] J. Mogul, J. Postel, "Internet Standard Subnetting Procedure," RFC 950, Aug. 1985.

[RFC 959] J. Postel and J. Reynolds, "File Transfer Protocol (FTP)," RFC 959, Oct. 1985.

[RFC 977] B. Kantor, P. Lapsley, "Network News Transfer Protocol," RFC 977, Feb. 1986.

[RFC 1028] J. Davin, J.D. Case, M. Fedor, M. Schoffstall, "A Simple Gateway Monitoring Protocol," RFC 1028, Nov. 1987.

[RFC 1034] P. V. Mockapetris, "Domain Names—Concepts and Facilities," RFC 1034, Nov. 1987.

[RFC 1035] P. Mockapetris, "Domain Names—Implementation and Specification," RFC 1035, Nov. 1987.

[RFC 1058] C. L. Hendrick, "Routing Information Protocol," RFC 1058, June 1988.

[RFC 1071] R. Braden, D. Borman, and C. Partridge, "Computing The Internet Checksum," RFC 1071, Sept. 1988.

[RFC 1075] D. Waitzman, C. Partridge, S. Deering, "Distance Vector Multicast Routing Protocol," RFC 1075, Nov. 1988.

[RFC 1112] S. Deering, "Host Extension for IP Multicasting," RFC 1112, Aug. 1989.

[RFC 1122] R. Braden, "Requirements for Internet Hosts—Communication Layers," RFC 1122, Oct. 1989.

[RFC 1123] R. Braden, ed., "Requirements for Internet Hosts—Application and Support," *RFC-1123*, Oct. 1989.

[RFC 1142] D. Oran, "OSI IS-IS Intra-domain Routing Protocol," RFC 1142, Feb. 1990.

[RFC 1190] C. Topolcic, "Experimental Internet Stream Protocol: Version 2 (ST-II)," RFC 1190, Oct. 1990.

[RFC 1191] J. Mogul, S. Deering, "Path MTU Discovery," RFC 1191, Nov. 1990.

[RFC 1213] K. McCloghrie, M. T. Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II," RFC 1213, Mar. 1991.

[RFC 1256] S. Deering, "ICMP Router Discovery Messages," RFC 1256, Sept. 1991.

[RFC 1320] R. Rivest, "The MD4 Message-Digest Algorithm," RFC 1320, Apr. 1992.

[RFC 1321] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321, Apr. 1992.

[RFC 1323] V. Jacobson, S. Braden, D. Borman, "TCP Extensions for High Performance," RFC 1323, May 1992.

[RFC 1422] S. Kent, "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-

Based Key Management," RFC 1422.

[RFC 1547] D. Perkins, "Requirements for an Internet Standard Point-to-Point Protocol," RFC 1547, Dec. 1993.

[RFC 1584] J. Moy, "Multicast Extensions to OSPF," RFC 1584, Mar. 1994.

[RFC 1633] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994.

[RFC 1636] R. Braden, D. Clark, S. Crocker, C. Huitema, "Report of IAB Workshop on Security in the Internet Architecture," RFC 1636, Nov. 1994.

[RFC 1661]W. Simpson (ed.), "The Point-to-Point Protocol (PPP)," RFC 1661, July 1994.

[RFC 1662]W. Simpson (ed.), "PPP in HDLC-like framing," RFC 1662, July 1994.

[RFC 1700] J. Reynolds and J. Postel, "Assigned Numbers," RFC 1700, Oct. 1994.

[RFC 1752] S. Bradner, A. Mankin, "The Recommendations for the IP Next Generation Protocol," RFC 1752, Jan. 1995.

[RFC 1760] N. Haller, "The S/KEY One-Time Password System," RFC 1760, Feb. 1995.

[RFC 1772]Y. Rekhter, P. Gross, "Application of the Border Gateway Protocol in the Internet," RFC 1772, Mar. 1995.

[RFC 1773] P. Traina, "Experience with the BGP-4 protocol," RFC 1773, Mar. 1995.

[RFC 1918] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, "Address Allocation for Private Internets," RFC 1918, Feb. 1996.

[RFC 1930] J. Hawkinson, T. Bates, "Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)," RFC 1930, Mar. 1996.

[RFC 1938] N. Haller, C. Metz, "A One-Time Password System," RFC 1938, May 1996.

[RFC 1939] J. Myers and M. Rose, "Post Office Protocol—Version 3," RFC 1939, May 1996.

[RFC 1945] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol—HTTP/1.0," RFC 1945, May 1996.

[RFC 2003] C. Perkins, "IP Encapsulation within IP," RFC 2003, Oct. 1996.

[RFC 2004] C. Perkins, "Minimal Encapsulation within IP," RFC 2004, Oct. 1996.

[RFC 2018] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgment Options," RFC 2018, Oct. 1996.

[RFC 2045] N. Freed, N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies," RFC 2045, Nov. 1996.

[RFC 2050] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, J. Postel, "Internet Registry IP Allocation Guidelines," RFC 2050, Nov. 1996.

[RFC 2104] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, Feb. 1997.

[RFC 2131] R. Droms, "Dynamic Host Configuration Protocol," RFC 2131, Mar. 1997.

[RFC 2136] P. Vixie, S. Thomson, Y. Rekhter, J. Bound, "Dynamic Updates in the Domain Name System," RFC 2136, Apr. 1997.

[RFC 2153]W. Simpson, "PPP Vendor Extensions," RFC 2153, May 1997.

[RFC 2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification," RFC 2205, Sept. 1997.

[RFC 2210] J. Wroclawski, "The Use of RSVP with IETF Integrated Services," RFC 2210, Sept. 1997.

[RFC 2211] J. Wroclawski, "Specification of the Controlled-Load Network Element Service," RFC 2211, Sept. 1997.

[RFC 2215] S. Shenker, J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements," RFC 2215, Sept. 1997.

[RFC 2225] M Laubach, J. Halpern, "Classical UP and ARP over ATM," RFC 2225,

Apr. 1998.

[RFC 2246] T. Dierks and C. Allen, "The TLS Protocol," RFC 2246, Jan. 1998.

[RFC 2253] M. Wahl, S. Kille, T. Howes, "Lightweight Directory Access Protocol (v3)," RFC 2253, Dec. 1997.

[RFC 2284] L. Blunk, J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)," RFC 2284, Mar. 1998.

[RFC 2326] H. Schulzrinne, A. Rao, R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326, Apr. 1998.

[RFC 2328] J. Moy, "OSPF Version 2," RFC 2328, Apr. 1998.

[RFC 2409] D. Harkins, D. Carrel, "The Internet Key Exchange (IKE)," RFC 2409, Nov. 1998. http://www.rfc-editor.org/rfc/rfc2409.txt

[RFC 2420] H. Kummert, "The PPP Triple-DES Encryption Protocol (3DESE)," RFC 2420, Sept. 1998.

[RFC 2437] B. Kaliski, J. Staddon, "PKCS #1: RSA Cryptography Specifications, Version 2," RFC 2437, Oct. 1998.

[RFC 2448] M. Civanlar, G. Cash, B. Haskell, "AT&T's Error Resilient Video Transmission Technique,"Nov. 1998.

[RFC 2453] G. Malkin, "RIP Version 2," RFC 2453, Nov. 1998.

[RFC 2460] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, Dec. 1998.

[RFC 2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services," RFC 2475, Dec. 1998.

[RFC 2578] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)," RFC 2578, Apr. 1999.

[RFC 2579] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Textual Conventions for SMIv2," RFC 2579, Apr. 1999.

[RFC 2580] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Conformance Statements for SMIv2," RFC 2580, Apr. 1999.

[RFC 2581] M. Allman, V. Paxson, W. Stevens, " TCP Congestion Control," RFC 2581, Apr. 1999.

[RFC 2597] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB Group," RFC 2597, June 1999.

[RFC 2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, R. Fielding, "Hypertext Transfer Protocol—HTTP/1.1," RFC 2616, June 1999.

[RFC 2663] P. Srisuresh, M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations," RFC 2663.

[RFC 2702] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, J. McManus, "Requirements for Traffic Engineering Over MPLS," Sept. 1999.

[RFC 2716] B. Aboba, D. Simon, "PPP EAP TLS Authentication Protocol," RFC 2716, Oct. 1999.

[RFC 2733] J. Rosenberg, H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction," RFC 2733, Dec. 1999.

[RFC 2821] J. Klensin, ed., "Simple Mail Transfer Protocol," RFC 2821, Apr. 2001.

[RFC 2827] P. Ferguson, D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing," RFC 2827, May 2000.

[RFC 2865] C. Rigney, S. Willens, A. Rubens, W. Simpson, "Remote Authentication Dial In User Service (RADIUS)," RFC 2865, June 2000.

RFC 2960] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, "Stream Control Transmission Protocol," RFC 2960, Oct. 2000.

[RFC 2961] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, S. Molendini, "RSVP Refresh Overhead Reduction Extensions," RFC 2961, Apr. 2001.

[RFC 2988] V. Paxson, M. Allman, "Computing TCP's Retransmission Timer," RFC 2988, Nov. 2000.

[RFC 3007] B. Wellington, "Secure Domain Name System (DNS) Dynamic Update," RFC 3007, Nov. 2000.

[RFC 3022] P. Srisuresh, K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, Jan. 2001.

[RFC 3031] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, Jan. 2001.

[RFC 3032] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, A. Conta, "MPLS Label Stack Encoding," RFC 3032, Jan. 2001.

[RFC 3052] M. Eder, S. Nag, "Service Management Architectures Issues and Review," RFC 3052, Jan. 2001.

[RFC 3139] L. Sanchez, K. McCloghrie, J. Saperia, "Requirements for Configuration Management of IP-Based Networks," RFC 3139, June 2001.

[RFC 3168] K. Ramakrishnan, S. Floyd, D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, Sept. 2001.

[RFC 3209] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, Dec. 2001.

[RFC 3221] G. Huston, "Commentary on Inter-Domain Routing in the Internet," RFC 3221, Dec. 2001.

[RFC 3232] J. Reynolds, "Assigned Numbers: RFC 1700 is Replaced by an On-line Database," RFC 3232, Jan. 2002.

[RFC 3246] B. Davie, A. Charny, J.C.R. Bennet, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)," RFC 3246, Mar. 2002.

[RFC 3260] D. Grossman, "New Terminology and Clarifications for Diffserv," RFC 3260, Apr. 2002.

[RFC 3261] J. Rosenberg, H. Schulzrinne, G. Carmarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, July 2002.

[RFC 3272] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awduche, B. Christian, W.S. Lai, "Overview and Principles of Internet Traffic Engineering," RFC 3272, May 2002.

[RFC 3286] L. Ong, J. Yoakum, "An Introduction to the Stream Control Transmission Protocol (SCTP)," RFC 3286, May 2002.

[RFC 3344] C. Perkins, ed., "IP Mobility Support for IPv4," RFC 3344, Oct. 2002.

[RFC 3346] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awduche, B. Christian, W. S. Lai, "Applicability Statement for Traffic Engineering with MPLS," RFC 3346, Aug. 2002.

[RFC 3376] B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, "Internet Group Management Protocol, Version 3," RFC 3376, Oct. 2002.

[RFC 3390] M. Allman, S. Floyd, C. Partridge, "Increasing TCP's Initial Window," RFC 3390, Oct. 2002.

[RFC 3410] J. Case, R. Mundy, D. Partain, "Introduction and Applicability Statements for Internet Standard Management Framework," RFC 3410, Dec. 2002.

[RFC 3411] D. Harrington, R. Presuhn, B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," RFC 3411, Dec. 2002.

[RFC 3414] U. Blumenthal, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)," RFC 3414, Dec. 2002.

[RFC 3415] B. Wijnen, R. Presuhn, K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)," RFC 3415, Dec. 2002.

[RFC 3416] R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, "Version 2 of the

Protocol Operations for the Simple Network Management Protocol (SNMP)," Dec. 2002.

[RFC 3417] R. Presuhn, "Transport Mappings for the Simple Network Management Protocol," (SNMP), RFC 3417, Dec. 2002.

[RFC 3439] R. Bush and D. Meyer, "Some internet architectural guidelines and philosophy," RFC 3439, Dec. 2003.

[RFC 3468] L. Andersson, G. Swallow, "The Multiprotocol Label Switching (MPLS) Working Group Decision on MPLS Signaling Protocols," RFC 3468, Feb. 2003.

[RFC 3469] V. Sharma, Ed., F. Hellstrand, Ed, "Framework for Multi-Protocol Label Switching (MPLS)-based Recovery," RFC 3469, Feb. 2003. ftp://ftp.rfc-editor.org/in-notes/rfc3469.txt

[RFC 3501] M. Crispin, "Internet Message Access Protocol—Version 4rev1," RFC 3501, Mar. 2003.

[RFC 3513] R. Hinden, S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture," RFC 3513, Apr. 2003.

[RFC 3550] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003.

[RFC 3569] S. Bhattacharyya (ed.), "An Overview of Source-Specific Multicast (SSM)," RFC 3569, July 2003.

[RFC 3588] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, "Diameter Base Protocol," RFC 3588, Sept. 2003.

[RFC 3618] B. Fenner, D. Meyer, Ed., "Multicast Source Discovery Protocol (MSDP)," RFC 3618, Oct. 2003.

[RFC 3649] S. Floyd, "High Speed TCP for Large Congestion Windows," RFC 3649, Dec. 2003.

[RFC 3700] J. Reynolds, S. Ginoza, Ed, "Internet Official Protocol Standards," RFC 3700, July 2004.

[RFC 3782] S. Floyd, T. Henderson, A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 3782, Apr. 2004.

[RFC 3973] A. Adams, J. Nicholas, W. Siadak, "Protocol Independent Multicast—Dense Mode (PIM-DM): Protocol Specification (Revised)," RFC 3973, Jan. 2005.

[RFC 4022] R. Raghunarayan, Ed., "Management Information Base for the Transmission Control Protocol (TCP)," RFC 4022, Mar. 2005.

[RFC 4113] B. Fenner, J. Flick, "Management Information Base for the User Datagram Protocol (UDP)," RFC 4113, June 2005.

[RFC 4213] E. Nordmark, R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers," RFC 4213, Oct. 2005.

[RFC 4271] Y. Rekhter, T. Li, S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Jan. 2006.

[RFC 4291] R. Hinden, S. Deering, "IP Version 6 Addressing Architecture, " RFC 4291, February 2006.

[RFC 4293] S. Routhier, Ed. "Management Information Base for the Internet Protocol (IP)," RFC 4293, Apr. 2006.

[RFC 4301] S. Kent, K. Seo, "Security Architecture for the Internet Protocol," RFC 4301, December 2005.

[RFC 4302] S. Kent, "IP Authentication Header," RFC 4302, December 2005.

[RFC 4303] S. Kent, "IP Encapsulating Security Payload (ESP)," RFC 4303, December 2005.

[RFC 4305] D. Eastlake, "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)," RFC 4305, December 2005.

[RFC 4340] E. Kohler, M. Handley, S. Floyd, "Datagram Congestion Control Protocol

(DCCP)," RFC 4340, Mar. 2006.

[RFC 4443] A. Conta, S. Deering, M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," RFC 4443, Mar. 2006.

[RFC 4346] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," RFC 4346, April 2006.

[RFC 4502] S. Waldbusser, "Remote Network Monitoring Management Information Base Version 2," RFC 4502, May 2006.

[RFC 4601] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, "Protocol Independent Multicast—Sparse Mode (PIM-SM): Protocol Specification (Revised)," RFC 4601, Aug. 2006.

[RFC 4607] H. Holbrook, B. Cain, "Source-Specific Multicast for IP," RFC 4607, Aug. 2006.

[RFC 4611] M. McBride, J. Meylor, D. Meyer, "Multicast Source Discovery Protocol (MSDP) Deployment Scenarios," RFC 4611, Aug. 2006.

[RFC 4632] V. Fuller, T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan," RFC 4632, Aug. 2006.

[Rhee 1998] I. Rhee, "Error Control Techniques for Interactive Low-bit Rate Video Transmission over the Internet," *Proc. 1998 ACM SIGCOMM,* (Vancouver BC, Aug. 1998).

[Roberts 1967] L. Roberts, T. Merril, "Toward a Cooperative Network of Time-Shared Computers," *AFIPS Fall Conference,* Oct. 1966.

[Roberts 2004] J. Roberts, "Internet Traffic, QoS and Pricing," Proceedings of the IEEE, Volume 92, No. 9, (Sept. 2004), pp. 1389–1399.

[Rom 1990] R. Rom, M. Sidi, *Multiple Access Protocols: Performance and Analysis,* Springer-Verlag, New York, 1990.

[Root Servers 2007] http://www.root-servers.org/

[Rose 1996] M. Rose, *The Simple Book: An Introduction to Internet Management, Revised Second Edition,* Prentice Hall, Englewood Cliffs, NJ, 1996.

[Rosenberg 2000] J. Rosenberg, L. Qiu, H. Schulzrinne, "Integrating Packet FEC into Adaptive Playout Buffer Algorithms on the Internet," *Proc. 2000 IEEE Infocom* (Tel Aviv, Israel, Apr. 2000).

[Ross 1995] K. W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks,* Springer, Berlin, 1995.

[Ross 2007] K. W. Ross, PowerPoint slides on network Security, http://cis.poly.edu/~ross

[Rowston 2001] A. Rowston, P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," in *Proc. 2001 IFIP/ACM Middleware,* Heidelberg, Germany, 2001.

[RSA 1978] R. Rivest, A. Shamir, L. Adelman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Communications of the ACM,* Vol. 21, No. 2, pp. 120–126, Feb. 1978.

[RSA FAQ 2004] RSA Inc., "RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1," http://www.rsasecurity.com/rsalabs/faq

[RSA Fast 2007] RSA Laboratories, "How fast is RSA?" http://www.rsasecurity.com/rsalabs/faq/3-1-2.html

[RSA Key 2007] RSA Laboratories, "How large a key should be used in the RSA Crypto system?" http://www.rsasecurity.com/rsalabs/faq/3-1-5.html

[Rubenstein 1998] D. Rubenstein, J. Kurose, D. Towsley, "Real-Time Reliable Multicast Using Proactive Forward Error Correction," *Proceedings of NOSSDAV '98* (Cambridge, UK, July 1998).

[Rubin 2001] A. Rubin, *White-Hat Security Arsenal: Tackling the Threats,* Addison-Wesley, 2001.

[Ruiz-Sanchez 2001] M. Ruiz-Sánchez, E. Biersack, W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms," *IEEE Network Magazine,* Vol. 15, No. 2, pp. 8–23, Mar./Apr. 2001

[Saltzer 1984] J. Saltzer, D. Reed, D. Clark, "End-to-End Arguments in System Design," *ACM Transactions on Computer Systems (TOCS),* Vol. 2, No. 4 (Nov. 1984).

[Saydam 1996] T. Saydam, T. Magedanz, "From Networks and Network Management into Service and Service Management," *Journal of Networks and System Management,* Vol. 4, No. 4 (Dec. 1996), pp. 345–348.

[Schiller 2003] J. Schiller, *Mobile Communications 2nd edition*, Addison Wesley, 2003.

[Schneier 1995] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C,* John Wiley and Sons, 1995.

[Schulzrinne 1997] H. Schulzrinne, "A Comprehensive Multimedia Control Architecture for the Internet," *NOSSDAV'97 (Network and Operating System Support for Digital Audio and Video),* (St. Louis, MO, May 1997).

[Schulzrinne-RTP 2007] Henning Schulzrinne's RTP site, http://www.cs.columbia.edu/~hgs/rtp

[Schulzrinne-RTSP 2007] Henning Schulzrinne's RTSP site, http://www.cs.columbia.edu/~hgs/rtsp

[Schulzrinne-SIP 2007] Henning Schulzrinne's SIP site, http://www.cs.columbia.edu/~hgs/sip

[Schwartz 1977] M. Schwartz, *Computer-Communication Network Design and Analysis,* Prentice-Hall, Englewood Cliffs, N.J., 1997.

[Schwartz 1980] M. Schwartz, *Information, Transmission, Modulation, and Noise,* McGraw Hill, NY, NY 1980.

[Schwartz 1982] M. Schwartz, "Performance Analysis of the SNA Virtual Route Pacing Control," *IEEE Transactions on Communications,* Vol. 30, No. 1, (Jan. 1982), pp. 172–184.

[Schwiebert 2001] L. Schwiebert, S. Gupta, J. Weinmann, "Research Challenges in Wireless Networks of Biomedical Sensors," *ACM Mobicom 2001,* 2001, pp. 151–165.

[Scourias 2007] J. Scourias, "Overview of the Global System for Mobile Communications: GSM." http://www.privateline.com/PCS/GSM0.html

[Segaller 1998] S. Segaller, *Nerds 2.0.1, A Brief History of the Internet,* TV Books, New York, 1998.

[Shacham 1990] N. Shacham, P. McKenney, "Packet Recovery in High-Speed Networks Using Coding and Buffer Management," *Proc. 1990 IEEE Infocom* (San Francisco, CA, Apr. 1990), pp. 124–131.

[Sharma 2003] P. Sharma, E, Perry, R. Malpani, "IP Multicast Operational Network management: Design, Challenges, and Experiences," *IEEE Network Magazine,* Mar. 2003, pp. 49–55.

[Sidor 1998] D. Sidor, "TMN Standards: Satisfying Today's Needs While Preparing for Tomorrow," *IEEE Communications Magazine,* Vol. 36, No. 3 (Mar. 1998), pp. 54–64.

[Singh 1999] S. Singh, *The Code Book: The Evolution of Secrecy from Mary, Queen of Scotsto Quantum Cryptography,* Doubleday Press, 1999.

[SIP Software 2007] H. Schulzrinne Software Package site, http://www.cs.columbia.edu/IRT/software

[Skoudis 2004] E. Skoudis, L. Zeltser, *Malware: Fighting Malicious Code*, Prentice Hall, 2004.

[Skoudis 2006] E. Skoudis, T. Liston, *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses (2nd Edition),* Prentice Hall, 2006.

[Skype 2007] Skype homepage, www.skype.com

[SMIL 2007] W3C Synchronized Multimedia homepage, http://www.w3.org/AudioVideo

[Snort 2007] Sourcefire Inc., Snort homepage, http:// http://www.snort.org/

[Solari 1997] S. J. Solari, *Digital Video and Audio Compression,* McGraw Hill, NY, NY, 1997.

[Solensky 1996] F. Solensky, "IPv4 Address Lifetime Expectations," in *IPng: Internet Protocol Next Generation* (S. Bradner, A. Mankin, ed.), Addison-Wesley, Reading, MA, 1996.

[Spragins 1991] J. D. Spragins, *Telecommunications Protocols and Design,* Addison-Wesley, Reading, MA, 1991.

[Sprint 2007] Sprint Corp., "Dedicated Internet Access Service Level Agreements," http://www.sprint.com/business/resources/dedicated_internet_access.pdf

[Srinivasan 1999] V. Srinivasan and G. Varghese, "Fast Address Lookup Using Controlled Prefix Expansion," *ACM Transactions Computer Sys.*, Vol. 17, No. 1 (Feb 1999), pp. 1–40.

[Sripanidkulchai 2004] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the Internet," *Proc. 4th ACM SIGCOMM Internet Measurement Conference,* (Taormina, Sicily, Italy), pp. 41–54, 2004.

[Stallings 1993] W. Stallings, *SNMP, SNMP v2, and CMIP The Practical Guide to Network Management Standards,* Addison-Wesley, Reading, MA, 1993.

[Stallings 1999] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2,* Addison-Wesley, Reading, MA, 1999.

[Steinder 2002] M. Steinder, A. Sethi, "Increasing robustness of fault localization through analysis of lost, spurious, and positive symptoms," *Proc. 2002 IEEE Infocom.*

[Stevens 1990]W. R. Stevens, *Unix Network Programming,* Prentice-Hall, Englewood Cliffs, NJ.

[Stevens 1994]W. R. Stevens, *TCP/IP Illustrated, Vol. 1: The Protocols,* Addison-Wesley, Reading, MA, 1994.

[Stevens 1997]W.R. Stevens, *Unix Network Programming, Volume 1: Networking APIs-Sockets and XTI,* 2nd edition, Prentice-Hall, Englewood Cliffs, NJ, 1997.

[Stewart 1999] J. Stewart, *BGP4: Interdomain Routing in the Internet,* Addison-Wesley, 1999.

[Stoica 2001] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. 2001 ACM SIGCOMM* (San Diego, CA, Aug. 2001).

[Stone 1998] J. Stone, M. Greenwald, C. Partridge, J. Hughes, "Performance of Checksums and CRC's Over Real Data," *IEEE/ACM Transactions on Networking,* Vol. 6, No. 5 (Oct. 1998), pp. 529–543.

[Stone 2000] J. Stone, C. Partridge, "When Reality and the Checksum Disagree," *Proc. 2000 ACM SIGCOMM* (Stockholm, Sweden, Aug. 2000).

[Strayer 1992] W. T. Strayer, B. Dempsey, A. Weaver, *XTP: The Xpress Transfer Protocol,* Addison-Wesley, Reading, MA, 1992.

[Stubblefield 2002] A. Stubblefield, J. Ioannidis, A. Rubin, "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP," *Proceedings of 2002 Network and Distributed Systems Security Symposium* (2002), 17–22.

[Subramanian 2000] M. Subramanian, *Network Management: Principles and Practice,* Addison-Wesley, Reading, MA, 2000.

[Subramanian 2002] L. Subramanian, S. Agarwal, J. Rexford, R. Katz, "Characterizing the Internet Hierarchy from Multiple Vantage Points," *Proc. 2002 IEEE Infocom.*

[Sundaresan 2006] K.Sundaresan, K. Papagiannaki, "The Need for Cross-layer Information in Access Point Selection," *Proc. 2006 ACM Internet Measurement Conference*, (Rio De Janeiro, Oct. 2006).

[Suh 2006] K. Suh, D.R. Figueiredo, J. Kurose and D. Towsley, "Characterizing and detecting relayed traffic: A case study using Skype," *Proc. 2006 IEEE Infocom* (Barcelona, Spain, Apr. 2006).

[**Sun 2007**] Sun Microsystems, "Solstice Enterprise Manager," http://www.sun.com/software/solstice/sem/index.

[**Sunshine 1978**] C. Sunshine, Y. Dalal, "Connection Management in Transport Protocols," *Computer Networks,* North-Holland, Amsterdam, 1978.

[**TechnOnLine 2004**] TechOnLine, "Protected Wireless Networks," online webcast tutorial, http://www.techonline.com/community/tech_topic/internet/21752

[**Thaler 1997**] D. Thaler and C. Ravishankar, "Distributed Center-Location Algorithms," *IEEE Journal on Selected Areas in Communications,* Vol. 15, No. 3, (Apr. 1997), pp. 291–303.

[**Think 2007**] Technical History of Network Protocols, "Cyclades," http://www.cs.utexas.edu/users/chris/think/Cyclades/index.shtml

[**Thottan 1998**] M. Thottan, C. Ji, "Proactive Anomaly Detection Using Distributed Intelligent Agents," *IEEE Network Magazine,* Vol. 12, No. 5 (Sept./Oct. 1998), pp. 21–28.

[**Tobagi 1990**] F. Tobagi, "Fast Packet Switch Architectures for Broadband Integrated Networks," *Proc. of the IEEE,* Vol. 78, No. 1 (Jan. 1990), pp. 133–167.

[**Turner 1988**] J. S. Turner "Design of a Broadcast packet switching network," *IEEE Transactions on Communications,* Vol. 36, No. 6 (June 1988), pp. 734–743.

[**UPnP Forum 2007**] UPnP Forum homepage, http://www.upnp.org/

[**Varghese 1997**] G. Varghese, A. Lauck, "Hashed and Hierarchical Timing Wheels: Efficient Data Structures for Implementing a Timer Facility," *IEEE/ACM Transactions on Networking,* Vol. 5, No. 6, (Dec. 1997), pp. 824–834.

[**Vasudevan 2006**] S. Vasudevan, C. Diot, J. Kurose, D. Towsley, "Facilitating Access Point Selection in IEEE 802.11 Wireless Networks," *Proc. 2005 ACM Internet Measurement Conference*, (San Francisco CA, Oct. 2005).

[**Verisign 2007**] http://www.verisign.com

[**Verizon 2007**] Verizon, "US Products and Services," http://www.verizonbusiness.com/terms/us/products/

[**Verma 2001**] D.C. Verma, *Content Distribution Networks: An Engineering Approach,* John Wiley, 2001.

[**Villamizar 1994**] C. Villamizar, C. Song. "High performance tcp in ansnet," *ACM SIGCOMM Computer Communications Review*, Vol. 24, No. 5 (1994), pp. 45–60.

[**Viterbi 1995**] A. Viterbi, *CDMA: Principles of Spread Spectrum Communication,* Addison-Wesley, Reading, MA, 1995.

[**von Lohmann 2003**] F. von Lohmann, "Peer-to-Peer File Sharing and Copyright Law: A Primer for Developers," *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03),* (Berkeley, CA, 2003).

[**Voydock 1983**] V.L. Voydock, S.T. Kent, "Security Mechanisms in High-Level Network Protocols," *ACM Computing Surveys,* Vol. 15, No. 2 (June 1983), pp. 135–171.

[**W3C 1995**] The World Wide Web Consortium, "A Little History of the World Wide Web," 1995. http://www.w3.org/History.html

[**Wakeman 1992**] I. Wakeman, J. Crowcroft, Z. Wang, D. Sirovica, "Layering Considered Harmful," *IEEE Network,* Jan. 1992, pp. 20–24.

[**Waldvogel 1997**] M. Waldvogel et al., "Scalable High Speed IP Routing Lookup," *Proc. 1997ACM SIGCOMM* (Cannes, France, Sept. 1997).

[**Walker 2000**] J. Walker, "IEEE P802.11 Wireless LANs, Unsafe at Any Key Size; An Analysis of the WEP Encapsulation," Oct. 2000, http://www.drizzle.com/~aboba/IEEE/0-362.zip

[**Wall 1980**] D. Wall, *Mechanisms for Broadcast and Selective Broadcast,* Ph.D. thesis, Stanford University, June 1980.

[**Wang 2004**] B. Wang, J. Kurose, P. Shenoy, D. Towsley, "Multimedia Streaming via TCP: An Analytic Performance Study," *Proc. ACM Multimedia Conf.*, (NY, NY, Oct. 2004).

[**Weatherspoon 2000**] S. Weatherspoon, "Overview of IEEE 802.11b Security," *Intel Technology Journal,* (2nd Quarter 2000), http://developer.intel.com/technology/itj/q22000/articles/art_5.htm

[**Wei 2005**] W. Wei, B. Wang, C. Zhang, J. Kurose, D. Towsley, "Classification of Access Network Types: Ethernet, Wireless LAN, ADSL, Cable Modem or Dialup?," *Proc. 2005 IEEE Infocom* (Apr. 2005).

[**Wei 2006**] W. Wei, C. Zhang, H. Zang, J. Kurose, D. Towsley, "Inference and Evaluation of Split-Connection Approaches in Cellular Data Networks," *Proc. Active and Passive Measurement Workshop,* (Adelaide, Australia, Mar. 2006).

[**Wei 2007**] D. X. Wei, C. Jin, S. H. Low, S. Hegde, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *IEEE/ACM Transactions on Networking, to* appear 2007.

[**Weiser 1991**] M. Weiser, "The Computer for the Twenty-First Century," *Scientific American* (Sept. 1991): 94–10. http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[**Wigle.net 2007**] Wireless Geographic Logging Engine, http://www.wigle.net

[**Williams 1993**] R. Williams, "A Painless Guide to CRC Error Detection Algorithms," http://www.ross.net/crc/crcpaper.html

[**WiMax Forum 2007**] WiMax Forum, http://www.wimaxforum.org/tech

[**Wischik 2005**] D. Wischik, N. McKeown, "Part I: Buffer Sizes for Core Routers," *ACM SIGCOMM Computer Communications Review*, Vol. 35, No. 3, July 2005.

[**Woo 1994**] T. Woo, R. Bindignavle, S. Su, S. Lam, "SNP: an interface for secure network programming," *Proc. 1994 Summer USENIX,* Boston, MA, June 1994, pp. 45–58.

[**Wood 2007**] L. Wood, "Lloyds Satellites Constellations," http://www.ee.surrey.ac.uk/Personal/L.Wood/constellations/iridium.html

[**Xanadu 2007**] Xanadu Project homepage, http://www.xanadu.com/

[**Xiao 2000**] X. Xiao, A. Hannan, B. Bailey, L. Ni, "Traffic Engineering with MPLS in the Internet," *IEEE Network,* Mar./Apr. 2000.

[**Yannuzzi 2005**] M. Yannuzzi, X. Masip-Bruin, O. Bonaventure, "Open Issues in Interdomain Routing: A Survey," *IEEE Network Magazine,* Nov./Dec. 2005.

[**Yavatkar 1994**] R. Yavatkar, N. Bhagwat, "Improving End-to-End Performance of TCP over Mobile Internetworks," *Proc. Mobile 94 Workshop on Mobile Computing Systems and Applications,* Dec. 1994.

[**Youtube 2007**] Youtube homage, www.youtube.com

[**Yu 2006**] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, SybilGuard: Defending Against Sybil Attacks via Social Networks, *Proc. 2006 ACM* (Pisa, Italy, Sept. 2006).

[**Zegura 1997**] E. Zegura, K. Calvert, M. Donahoo, "A Quantitative Comparison of Graph-based Models for Internet Topology," *IEEE/ACM Transactions on Networking,* Vol. 5, No. 6, (Dec. 1997). See also http://www.cc.gatech.edu/projects/gtim for a software package that generates networks with a transit-stub structure.

[**Zhang 1993**] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, "RSVP: A New Resource Reservation Protocol," *IEEE Network Magazine,* Vol. 7, No. 9 (Sept. 1993), pp. 8–18.

[**Zhao 2004**] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, J. Kubiatowicz, "Tapestry: A Resilient Global-scale Overlay for Service Deployment," *IEEE Journal on Selected Areas in Communications,* Vol. 22, No. 1 (Jan. 2004).

[**Zimmerman 1980**] H. Zimmerman, "OS1 Reference Model-The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications,* Vol. 28, No. 4 (Apr. 1980), pp. 425–432.

[**Zimmermann 2007**] P. Zimmermann, "Why do you need PGP?" http://www.pgpi.org/doc/whypgp/en/