更多资料免费下载

http://pc10201.cech.com.cn/#show=file

我的原创文章——《三个月零基础奇迹通过软考网工经验》

http://pc10201.blog.hexun.com/22835463_d.html

网工考试，思科认证，MCSE（微软认证），RHCE（红帽认证）等视频资料

http://pc10201.blog.hexun.com/25188617_d.html

网工考试必读书电子版

http://pc10201.blog.hexun.com/27541752_d.html

更多资料更新请关注

http://pc10201.blog.hexun.com/

联系我

QQ :276148575，阿里旺旺 :pc10201　QQ 群 :17892693

下一页是正文，敬请阅读

高等学校
电子信息类 规划教材

计算机网络

（第4版）

谢希仁 编著

43

電子工業出版社
Publishing House of Electronics Industry
www.phei.com.cn

计算机网络

（第4版）

ISBN 7-5053-8786-3

9 787505 387867 >

责任编辑：杜振民

封面设计：王海军

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书

ISBN 7-5053-8786-3/TP·5098　定价：35.00元（赠光盘一张）

高等学校电子信息类规划教材

计算机网络

（第4版）

谢希仁　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书为1989年出版的、获第二届全国优秀教材奖的《计算机网络》的第4版，在内容和结构方面都做了很大的修改。

全书为10章，比较全面系统地介绍了计算机网络的发展和原理体系结构、物理层、数据链路层、局域网、广域网、网络互连、运输层、应用层、计算机网络的安全和因特网的演进等内容。各章均附有练习题。此外，附录F给出了部分习题的答案和提示。随书配套的光盘中，收录了作者教学中经常遇到的150多个问题，并予以解答；还收录了在2003年1月前发表的全部RFC文档，供读者参阅。

本书的特点是概念准确、论述严谨、内容新颖、图文并茂，突出基本原理和基本概念的阐述，同时力图反映出计算机网络的一些最新发展。第4版更加突出了以TCP/IP协议族为核心的一些常用网络协议以及一些网络新技术。本书可供通信和计算机专业的大学本科生和研究生使用，对从事计算机网络工作的工程技术人员也有学习参考价值。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，翻版必究。

图书在版编目（CIP）数据

计算机网络／谢希仁编著. —4版. —北京：电子工业出版社，2003.6
高等学校电子信息类规划教材
ISBN 7-5053-8786-3

Ⅰ.计… Ⅱ.谢… Ⅲ.计算机网络－高等学校－教材 Ⅳ.TP393

中国版本图书馆CIP数据核字（2003）第044070号

责任编辑：杜振民
印　　刷：北京东光印刷厂
出版发行：电子工业出版社　www.phei.com.cn
　　北京市海淀区万寿路173信箱　邮编：100036
经　　销：各地新华书店
开　　本：787 × 1092　1/16　印张：29.5　字数：730千字
版　　次：2003年6月第4版　2003年7月第2次印刷
定　　价：35.00元（赠光盘一张）

凡购买电子工业出版社的图书，如有缺页、倒页、脱页者，请向购买书店调换。若书店售缺，请与本社发行部联系调换。联系电话：88211980　68279077

出 版 说 明

为做好全国电子信息类专业“九五”教材的规划和出版工作，根据国家教委《关于“九五”期间普通高等教育教材建设与改革的意见》和《普通高等教育“九五”国家级重点教材立项、管理办法》，我们组织各有关高等学校、中等专业、出版社，各专业教学指导委员会，在总结前四轮规划教材编审、出版工作的基础上，根据当代电子信息科学技术的发展和面向21世纪教学内容和课程体系改革的要求，编制了《1996～2000年全国电子信息类专业教材编审出版规划》。

本轮规划教材是由个人申报，经各学校、出版社推荐，由各专业教学指导委员会评选，并由我部教材办商各专指委、出版社后，审核确定的。本轮规划教材的编制，注意了将教学改力度较大，有创新精神、特色风格的教材和质量较高、教学适用性较好、需要修订的教材以及教学急需，尚无正式教材的选题优先列入规划。在重点规划本科、专科和中专教材的同时，选择了一批对学科发展具有重要意义，反映学科前沿的选修课、研究生课教材列入规划，以适应高层次专门人才培养的需要。

限于我们的水平和经验，这批教材的编审、出版工作还可能存在不少缺点和不足，希望使用教材的学校、教师、同学和广大读者积极提出批评和建议，以不断提高教材的编写、出版质量，共同为电子信息类专业教材建设服务。

原电子工业部教材办公室

再 版 前 言

本教材的前一版是根据原电子工业部的《1996-2000 年全国电子信息类专业教材编审出版规划》，由计算机教学指导委员会编审、推荐出版，是普通高等教育“九五”国家级重点教材。

虽然本教材中的新概念和协议都是他人提出的，但对教材的编者来说，还需要精心选择内容。在非常有限的篇幅中，应当把基本原理讲清楚。即使这样，学生学完了本书后也只能做到对计算机网络的基本内容有一定深度的了解（不是精通）。例如，若想精通以太网，就至少应认真学习 IEEE 的 802.3 标准，但此标准的文档超过 1500 页。

本教材的参考学时数为 70～90 学时。全书以 TCP/IP 协议族为线索，突出因特网的应用。与上一版相比，删除或压缩了一些离 TCP/IP 主线较远的次要内容。根据读者们反映出的问题增加了一些用来加深理解的例子或图。新版在重点问题上的阐述更加清楚了，如路由器的结构和路由选择机制，子网掩码和 CIDR，客户服务器工作方式，TCP 的拥塞控制等。新增加的内容有：计算机网络的主要性能指标，波分复用和码分复用，循环冗余检验举例，采用随机早期丢弃 RED 进行拥塞控制，应用编程接口 API，传输实时数据的协议 RTP 和 RTCP 等。在改进因特网的“尽最大努力交付”服务方面，如 IntServ、RSVP、DiffServ 和 MPLS 等，均在第 10 章中讨论。

新版共有 8 个附录，从附录 A 到附录 H 分别是：最基本的排队系统、随机接入技术 ALOHA、综合业务数字网 ISDN、关于 ATM 的通信量、最短路径算法（Dijkstra 算法）、部分习题解答（而不是详细解题步骤）、英文缩写词、参考文献与网址。

本书的 CD-ROM 内是作者在多年教学中收集整理的常见问题的解答和在 2003 年 1 月以前能收集到的 RFC 文档，供读者参阅。

对于最基本的内容，在目录中的相应章节前面加上一个星号“*”。

陈鸣教授和杨心强、高素青、张兴元、齐望东、吴礼发、胥光辉副教授、张涛博士，以及在北京工作的赵刚副研究员等，对本书的修改都提出了很多宝贵的意见。吴自珠副教授一直对本教材的出版给予全力支持。对这些，作者均表示诚挚的谢意。由于作者水平所限，书中难免还存在一些缺点和错误，殷切希望广大读者批评指正。

谢希仁

2003 年 1 月

于解放军理工大学通信工程学院，南京

作者的电子邮件地址：xiexr@public1.ptt.js.cn

（欢迎指出书中的各种错误，但无法满足索取解题详细步骤的要求，请谅解。）

前　言

本教材系按原电子工业部的《1996-2000年全国电子信息类专业教材编审出版规划》，由计算机教学指导委员会编审、推荐出版。本教材为普通高等教育“九五”国家级重点教材，由南京通信工程学院谢希仁教授编著，陈鸣教授主审，邵军力教授为责任编委。

本教材的第一版（1989年大连理工大学出版社）曾获第二届全国优秀教材奖。1994年和1996年曾分别在电子工业出版社和大连理工大学出版社各再修订出版一次。

本教材的参考学时数为70～90学时。全书共分12章。与过去的版本相比，这次的改动较多。全书以TCP/IP协议族为线索，突出Internet上的应用，增加了对许多新技术的介绍。

例如：56 kb/s调制解调器，高速局域网（包括100BASE T, 100VG AnyLAN, HIPPI和千兆比以太网等），无线局域网，帧中继技术，IPv6，国际数据加密算法IDEA，MD5报文摘要算法，Internet的安全体系，防火墙，万维网WWW（包括URL，HTTP，HTML，CGI等），网络管理SNMP，ATM技术（包括各种AAL协议和ATM通信量管理等），ATM与IP相结合（包括IPOA，LANE，MPOA和MPLS等），IP电话，居民接入网RAN（包括xDSL, HFC以及FTTx等），等等，均在书中占有一定的篇幅。

为了不使全书篇幅过大，对一些比较陈旧的内容进行了适当的压缩。

编写教材最难处理的就是内容的取舍。网络技术的飞速发展使得新的网络技术和标准不断问世。在非常有限的篇幅中，应当将哪些最为重要的内容交给学生呢？经验证明，最重要的就是要在教材中把基本原理讲清楚。理论联系实际是十分必要的，但显然不应将教材写成为工程实践中的使用手册。新的但很不成熟的内容也不宜写入教材。教科书不应写成为标准文档的缩写。

每章的最后都附有若干练习题。附录A是为未学过排队论的读者参考的。附录B是英文缩写词。附录C是参考文献和一些有参考价值的网点，以使读者能够更快地从Internet上找到所需的资料。为了使本教材能够适用于不同要求的读者，对于最重要的内容，在目录中的相应章节前面加上一个星号“*”。

陈鸣、胡谷雨教授和张兴元、高素青副教授以及齐望东博士、徐哲博士对本书的初稿提出了很多宝贵的意见。已毕业并工作多年的研究生赵刚也从北京通过电子邮件提出了许多很好的建议。

赵小凡高工、刘南杰教授以及林波、黄振荧副教授曾对本书的前一个版本提出了许多建设性意见。吴自珠副教授一直对本教材的出版给予全力支持。谢冲提供了最新的一些文献的光盘和多本美国原版新书。对这些，作者均表示诚挚的谢意。由于作者水平所限，书中难免还存在一些缺点和错误，殷切希望广大读者批评指正。

作者的电子邮件地址：xiexr@public1.ptt.js.cn

谢希仁

1999年1月于南京通信工程学院

目　录

第 1 章　概述 …… (1)
1.1　计算机网络在信息时代中的作用 …… (1)
1.2　计算机网络的发展过程 …… (2)
*1.2.1　分组交换的产生 …… (2)
*1.2.2　因特网时代 …… (8)
*1.2.3　关于因特网的标准化工作 …… (10)
1.2.4　计算机网络在我国的发展 …… (12)
1.3　计算机网络的分类 …… (13)
1.3.1　计算机网络的不同定义 …… (13)
1.3.2　几种不同的分类方法 …… (14)
*1.4　计算机网络的主要性能指标 …… (16)
1.4.1　带宽 …… (16)
1.4.2　时延 …… (17)
1.4.3　时延带宽积和往返时延 …… (18)
*1.5　计算机网络的体系结构 …… (19)
1.5.1　计算机网络体系结构的形成 …… (19)
1.5.2　划分层次的必要性 …… (21)
1.5.3　具有五层协议的体系结构 …… (23)
1.5.4　实体、协议、服务和服务访问点 …… (25)
1.5.5　面向连接服务与无连接服务 …… (27)
1.5.6　OSI 与 TCP/IP 体系结构的比较 …… (27)
*1.6　应用层的客户-服务器方式 …… (29)
习题 …… (31)

第 2 章　物理层 …… (33)
*2.1　物理层的基本概念 …… (33)
*2.2　数据通信的基础知识 …… (33)
2.2.1　数据通信系统的模型 …… (33)
2.2.2　有关信道的几个基本概念 …… (35)
2.2.3　信道的最高码元传输速率 …… (36)
2.2.4　信道的极限信息传输速率 …… (38)
2.3　物理层下面的传输媒体 …… (39)
2.3.1　导向传输媒体 …… (39)
2.3.2　非导向传输媒体 …… (45)
2.4　模拟传输与数字传输 …… (47)

2.4.1 模拟传输系统 …… (48)
*2.4.2 调制解调器 …… (49)
*2.4.3 数字传输系统 …… (53)
*2.5 信道复用技术 …… (56)
2.5.1 频分复用、时分复用和统计时分复用 …… (56)
2.5.2 波分复用 …… (58)
2.5.3 码分复用 …… (59)
*2.6 同步光纤网 SONET 和同步数字系列 SDH …… (61)
2.7 物理层标准举例 …… (63)
*2.7.1 EIA-232-E 接口标准 …… (63)
2.7.2 RS-449 接口标准 …… (65)
习题 …… (66)

第 3 章 数据链路层 …… (68)
*3.1 数据链路层的基本概念 …… (68)
3.2 停止等待协议 …… (69)
*3.2.1 完全理想化的数据传输 …… (69)
*3.2.2 具有最简单流量控制的数据链路层协议 …… (70)
*3.2.3 实用的停止等待协议 …… (71)
3.2.4 循环冗余检验的原理 …… (73)
*3.2.5 停止等待协议的算法 …… (74)
3.2.6 停止等待协议的定量分析 …… (75)
3.3 连续 ARQ 协议 …… (77)
*3.3.1 连续 ARQ 协议的工作原理 …… (77)
3.3.2 连续 ARQ 协议的吞吐量 …… (78)
*3.3.3 滑动窗口的概念 …… (79)
*3.3.4 信道利用率 …… (82)
3.4 选择重传 ARQ 协议 …… (83)
3.5 面向比特的链路控制规程 HDLC …… (83)
*3.5.1 HDLC 协议概述 …… (83)
3.5.2 HDLC 的帧结构 …… (84)
*3.6 因特网的点对点协议 PPP …… (87)
3.6.1 PPP 协议的工作原理 …… (87)
3.6.2 PPP 协议的帧格式 …… (88)
3.6.3 PPP 协议的工作状态 …… (90)
习题 …… (91)

第 4 章 局域网 …… (93)
*4.1 局域网概述 …… (93)
4.2 传统以太网 …… (94)
*4.2.1 以太网的工作原理 …… (94)

*4.2.2 传统以太网的连接方法 …… (100)
4.2.3 以太网的信道利用率 …… (104)
*4.3 以太网的 MAC 层 …… (107)
4.3.1 MAC 层的硬件地址 …… (107)
4.3.2 两种不同的 MAC 帧格式 …… (110)
*4.4 扩展的局域网 …… (112)
4.4.1 在物理层扩展局域网 …… (112)
4.4.2 在数据链路层扩展局域网 …… (113)
4.5 虚拟局域网 …… (118)
4.5.1 虚拟局域网的概念 …… (118)
4.5.2 虚拟局域网使用的以太网帧格式 …… (119)
*4.6 高速以太网 …… (120)
4.6.1 100BASE-T 以太网 …… (120)
4.6.2 吉比特以太网 …… (121)
4.6.3 10 吉比特以太网 …… (123)
4.7 其他种类的高速局域网 …… (125)
4.7.1 100VG-AnyLAN 局域网 …… (125)
4.7.2 光纤分布式数据接口 FDDI …… (125)
4.7.3 高性能并行接口 HIPPI …… (126)
4.7.4 光纤通道 …… (126)
4.8 无线局域网 …… (126)
*4.8.1 无线局域网的组成 …… (126)
4.8.2 802.11 标准中的物理层 …… (129)
4.8.3 802.11 标准中的 MAC 层 …… (130)
习题 …… (135)

第 5 章 广域网 …… (138)
*5.1 广域网的基本概念 …… (138)
5.1.1 广域网的构成 …… (138)
5.1.2 数据报和虚电路 …… (139)
*5.2 广域网中的分组转发机制 …… (141)
5.2.1 在结点交换机中查找转发表 …… (142)
5.2.2 在路由表中使用默认路由 …… (143)
*5.3 拥塞控制 …… (145)
5.3.1 拥塞控制的意义 …… (145)
5.3.2 拥塞控制的一般原理 …… (147)
5.4 X.25 网 …… (148)
5.5 帧中继 FR …… (150)
*5.5.1 帧中继的工作原理 …… (150)
5.5.2 帧中继的帧格式 …… (153)
5.5.3 帧中继的拥塞控制 …… (154)

5.6 异步传递方式 ATM …… (155)
*5.6.1 ATM 的基本概念 …… (155)
5.6.2 ATM 的协议参考模型和信元结构 …… (157)
5.6.3 ATM 的逻辑连接机制 …… (162)
5.6.4 AAL 层举例：AAL5 …… (166)
习题 …… (167)

第 6 章 网络互连 …… (170)
*6.1 路由器在网际互连中的作用 …… (170)
6.1.1 路由器的构成 …… (170)
6.1.2 交换结构 …… (173)
6.1.3 互联网与因特网 …… (174)
*6.2 因特网的网际协议 IP …… (176)
6.2.1 分类的 IP 地址 …… (176)
6.2.2 IP 地址与硬件地址 …… (180)
6.2.3 地址解析协议 ARP 和逆地址解析协议 RARP …… (183)
6.2.4 IP 数据报的格式 …… (185)
6.2.5 IP 层转发分组的流程 …… (189)
*6.3 划分子网和构造超网 …… (192)
6.3.1 划分子网 …… (192)
6.3.2 使用子网掩码的分组转发过程 …… (196)
6.3.3 无分类编址 CIDR …… (198)
*6.4 因特网控制报文协议 ICMP …… (203)
*6.5 因特网的路由选择协议 …… (206)
6.5.1 有关路由选择协议的几个基本概念 …… (206)
6.5.2 内部网关协议 RIP …… (209)
6.5.3 内部网关协议 OSPF …… (214)
6.5.4 外部网关协议 BGP …… (219)
6.6 IP 多播和因特网组管理协议 IGMP …… (223)
6.6.1 IP 多播的基本概念 …… (223)
6.6.2 因特网组管理协议 IGMP …… (225)
6.6.3 多播路由选择 …… (228)
6.7 虚拟专用网 VPN 和网络地址转换 NAT …… (229)
6.7.1 虚拟专用网 VPN …… (229)
6.7.2 网络地址转换 NAT …… (231)
6.8 下一代的网际协议 IPv6(IPng) …… (232)
*6.8.1 解决 IP 地址耗尽的措施 …… (232)
6.8.2 IPv6 的基本首部 …… (232)
6.8.3 IPv6 的扩展首部 …… (234)
6.8.4 IPv6 的地址空间 …… (236)
6.8.5 从 IPv4 向 IPv6 过渡 …… (240)

6.8.6 ICMPv6 …… (243)
习题 …… (243)

第 7 章 运输层 …… (248)
*7.1 运输层协议概述 …… (248)
*7.2 TCP/IP 体系中的运输层 …… (250)
7.2.1 运输层中的两个协议 …… (250)
7.2.2 端口的概念 …… (251)
*7.3 用户数据报协议 UDP …… (253)
7.3.1 UDP 概述 …… (253)
7.3.2 UDP 用户数据报的首部格式 …… (255)
7.4 传输控制协议 TCP …… (256)
*7.4.1 TCP 概述 …… (256)
*7.4.2 TCP 报文段的首部 …… (257)
*7.4.3 TCP 的数据编号与确认 …… (260)
*7.4.4 TCP 的流量控制与拥塞控制 …… (262)
*7.4.5 TCP 的重传机制 …… (267)
7.4.6 采用随机早期丢弃 RED 进行拥塞控制 …… (269)
*7.4.7 TCP 的运输连接管理 …… (271)
7.4.8 TCP 的有限状态机 …… (274)
习题 …… (276)

第 8 章 应用层 …… (279)
*8.1 域名系统 DNS …… (279)
8.1.1 域名系统概述 …… (279)
8.1.2 因特网的域名结构 …… (280)
8.1.3 用域名服务器进行域名解析 …… (282)
8.2 文件传送协议 …… (284)
8.2.1 概述 …… (284)
*8.2.2 FTP 的基本工作原理 …… (285)
8.2.3 简单文件传送协议 TFTP …… (287)
8.3 远程终端协议 TELNET …… (289)
*8.4 电子邮件 …… (290)
8.4.1 概述 …… (290)
8.4.2 简单邮件传送协议 SMTP …… (293)
8.4.3 电子邮件的信息格式 …… (295)
8.4.4 邮件读取协议 POP3 和 IMAP …… (295)
8.4.5 通用因特网邮件扩充 MIME …… (296)
8.5 万维网 WWW …… (300)
*8.5.1 概述 …… (300)
*8.5.2 统一资源定位符 URL …… (302)

*8.5.3 超文本传送协议 HTTP …… (304)
*8.5.4 超文本标记语言 HTML …… (309)
*8.5.5 万维网页面中的超链 …… (312)
8.5.6 动态万维网文档与 CGI 技术 …… (316)
8.5.7 活动万维网文档 …… (322)
8.5.8 万维网上的信息检索系统 …… (324)
*8.6 引导程序协议 BOOTP 与动态主机配置协议 DHCP …… (327)
8.6.1 引导程序协议 BOOTP …… (327)
8.6.2 动态主机配置协议 DHCP …… (328)
8.7 简单网络管理协议 SNMP …… (330)
*8.7.1 网络管理的基本概念 …… (330)
8.7.2 简单网络管理协议 SNMP 概述 …… (331)
8.7.3 管理信息库 MIB …… (332)
8.7.4 SNMPv1 的五种协议数据单元 …… (334)
8.7.5 管理信息结构 SMI …… (337)
8.7.6 SNMPv2 和 SNMPv3 …… (342)
8.8 应用进程跨越网络的通信 …… (343)
*8.8.1 系统调用和应用编程接口 …… (343)
8.8.2 服务器的两种工作方式 …… (345)
8.8.3 进程通过系统调用接口进行通信的过程 …… (346)
习题 …… (348)

第 9 章 计算机网络的安全 …… (351)
*9.1 网络安全问题概述 …… (351)
9.1.1 计算机网络面临的安全性威胁 …… (351)
9.1.2 计算机网络安全的内容 …… (352)
9.1.3 一般的数据加密模型 …… (353)
*9.2 常规密钥密码体制 …… (354)
9.2.1 替代密码与置换密码 …… (354)
9.2.2 数据加密标准 DES …… (356)
*9.3 公开密钥密码体制 …… (359)
9.3.1 公开密钥密码体制的特点 …… (359)
9.3.2 RSA 公开密钥密码体制 …… (360)
9.3.3 数字签名 …… (361)
*9.4 报文鉴别 …… (362)
*9.5 密钥分配 …… (364)
9.6 电子邮件的加密 …… (365)
9.6.1 PGP …… (365)
9.6.2 PEM …… (366)
*9.7 链路加密与端到端加密 …… (367)
9.7.1 链路加密 …… (367)

9.7.2 端到端加密 …… (368)
9.8 因特网商务中的加密 …… (368)
9.8.1 安全插口层 SSL …… (369)
9.8.2 安全电子交易 SET …… (370)
9.9 因特网的网络层安全协议族 IPsec …… (371)
*9.10 防火墙 …… (372)
习题 …… (374)

第 10 章 因特网的演进 …… (375)
*10.1 概述 …… (375)
10.2 因特网的多媒体体系结构 …… (377)
10.2.1 实时运输协议 RTP …… (377)
10.2.2 实时运输控制协议 RTCP …… (380)
10.2.3 实时流式协议 RTSP …… (381)
10.3 IP 电话 …… (381)
10.3.1 IP 电话概述 …… (381)
10.3.2 H.323 …… (383)
10.3.3 会话发起协议 SIP …… (384)
10.3.4 IP 电话的通话质量 …… (385)
10.4 改进“尽最大努力交付”的服务 …… (387)
10.4.1 使因特网提供服务质量 …… (387)
10.4.2 调度和管制机制 …… (389)
10.4.3 综合服务 IntServ 与资源预留协议 RSVP …… (392)
10.4.4 区分服务 DiffServ …… (395)
10.5 多协议标记交换 MPLS …… (397)
10.5.1 MPLS 的产生背景 …… (397)
10.5.2 MPLS 的工作原理 …… (399)
10.6 居民接入网 RAN …… (403)
10.6.1 xDSL 技术 …… (403)
10.6.2 光纤同轴混合网(HFC 网) …… (405)
10.6.3 FTTx 技术 …… (407)
10.6.4 以太网接入 …… (408)
10.7 关于“三网融合” …… (409)
习题 …… (410)

附录 A 最基本的排队系统 …… (412)
A.1 稳定状态下的数据流 …… (412)
A.1.1 李特尔(Little)定律 …… (412)
A.1.2 通信量强度 …… (413)
A.2 几种排队模型 …… (415)
A.2.1 M/G/1 模型 …… (415)

A.2.2 M/M/1 模型 …… (418)
A.2.3 M/D/1 模型 …… (419)

附录 B 随机接入技术：ALOHA …… (420)
B.1 纯 ALOHA …… (420)
B.2 时隙 ALOHA(S-ALOHA) …… (423)
习题 …… (426)

附录 C 综合业务数字网 ISDN …… (427)
C.1 窄带综合业务数字网 N-ISDN …… (427)
C.2 宽带综合业务数字网 B-ISDN …… (428)

附录 D 关于 ATM 的通信量 …… (429)
D.1 ATM 通信量的特点 …… (429)
D.2 ATM 通信量管理中的一些重要参数 …… (430)
D.3 ATM 服务的五个种类 …… (432)

附录 E 最短路径算法——Dijkstra 算法 …… (434)

附录 F 部分习题的解答 …… (436)

附录 G 英文缩写词 …… (447)

附录 H 参考文献与网址 …… (455)

第1章 概 述

1.1 计算机网络在信息时代中的作用

我们已经进入了 21 世纪。21 世纪的一些重要特征就是**数字化**、**网络化**和**信息化**，它是一个**以网络为核心的信息时代**。

当前世界经济正在从工业经济向**知识经济**(knowledge-based economy)转变。知识经济是相对于农业经济、工业经济而出现的一种正在形成中的崭新的经济形态。知识经济就是指以知识为基础的经济，并且经济的发展在很大程度上取决于对知识的发掘和积累。知识经济的诞生不仅对人们的工作、学习、交往等各个方面起着非常大的作用，而且也影响了整个社会的发展。知识经济已成为推动生产力发展的巨大动力。

知识经济中的两个重要特点就是**信息化**和**全球化**。要实现信息化和全球化，就必须依靠完善的网络。因此网络现在已经成为信息社会的命脉和发展知识经济的重要基础。网络对社会生活的很多方面以及对社会经济的发展已经产生了不可逆转的影响。

这里所说的网络是指“三网”，即**电信网络**（主要的业务是电话，但也有其他业务，如传真、数据等）、**有线电视网络**和**计算机网络**。虽然这三种网络在信息化过程中都起到十分重要的作用，但其中发展最快的并起到核心作用的是计算机网络，而这正是本书所要讨论的内容。

进入 20 世纪 90 年代以后，以**因特网**(Internet)为代表的计算机网络得到了飞速的发展，已从最初的教育科研网络逐步发展成为商业网络，并已成为仅次于全球电话网的世界第二大网络。不少人认为现在已经是因特网的时代，这是因为因特网正在改变着我们工作和生活的各个方面，它已经给很多国家（尤其是因特网的发源地美国）带来了巨大的好处，并加速了全球信息革命的进程。可以毫不夸大地说，因特网是自印刷术以来人类通信方面最大的变革。现在人们的生活、工作、学习和交往都已离不开因特网。

1993 年 9 月 15 日，美国政府发布一个在全世界引起很大反响的文件，其标题是“**国家信息基础结构(NII)行动计划**”。NII 即 National Information Infrastructure 的缩写，也可译为**国家信息基础设施**。这个文件提出，高速信息网是国家信息基础结构的一个重要组成部分。为了更加生动而形象地说明这个“NII 行动计划”，人们常用“**信息高速公路**”这个名词作为“国家信息基础结构”的通俗名称。

1994 年 9 月美国又提出建立全**球信息基础结构** GII，建议将各国的 NII 互连起来组成世界范围的信息基础结构。当前的因特网就是这种全球性的信息基础结构的雏形。

现在全世界所有的工业发达国家和很多的发展中国家都纷纷研究和制订本国建设信息基础结构的计划。这就使得计算机网络的发展进入了一个新的历史阶段，并变成了几乎人人都知道而且都十分关心的热门学科。

下面我们将从计算机网络的发展开始讨论。

1.2 计算机网络的发展过程

1.2.1 分组交换的产生

计算机网络涉及到通信与计算机两个领域。计算机与通信日益紧密的结合，已对人类社会的进步做出了极大的贡献。

计算机与通信的相互结合主要有两个方面。一方面，通信网络为计算机之间的数据传递和交换提供了必要的手段；另一方面，数字计算技术的发展渗透到通信技术中，又提高了通信网络的各种性能。当然，这两个方面的进展都离不开人们在半导体技术（主要是超大规模集成电路 VLSI 技术）上取得的辉煌成就。

现代计算机网络实际上是 20 世纪 60 年代美苏冷战时期的产物。在 60 年代初，美国国防部领导的远景研究规划局 ARPA (Advanced Research Project Agency)[①] 提出要研制一种崭新的、能够适应现代战争的、生存性(survivability)很强的网络，其目的是对付来自前苏联的核进攻威胁。我们知道，传统的**电路交换**(circuit switching)的电信网虽然已经四通八达，但在战争期间，一旦正在通信的电路中有一个交换机或有一条链路被炸毁，则整个通信电路就要中断。如要立即改用其他迂回电路通信，还必须重新拨号建立连接。这将要延误一些时间（例如十几秒钟），但这也可能造成不可挽回的重大损失。

根据当时美国军方提出的需求，这种新型的网络必须满足以下的一些基本要求：

（1）和传统的电信网不同，这种新型的网络不是为了打电话，而是**用于计算机之间的数据传送**。

（2）新型的网络**能够连接不同类型的计算机**，即不局限于单一类型的计算机。

（3）**所有的网络结点都同等重要**。因为网络必须经受得起敌人的核打击，所以在网络中不能有某些特别重要的结点[②]，否则敌人将首先瞄准和摧毁这些重要的结点。将所有的结点设计成同等重要的，就可以大大提高网络的生存性。

（4）计算机在进行通信时，**必须有冗余的路由**。当网络中的某一个结点或链路被破坏时，冗余的路由能够使正在进行的通信自动地找到合适的路由，使通信维持畅通。

（5）网络的结构应当**尽可能地简单，但能够非常可靠地传送数据**。

根据以上的这些要求，一批专家终于设计出了使用分组**交换**（packet switching）的新型计算机网络。为了掌握分组交换的概念，我们先简单地回顾一下电路交换的特点。

在电话问世后不久，人们就发现，要让所有的电话机都两两相连接是不现实的。图 1-1(a)表示两部电话只需要用一对电线就能够互相连接起来。但若有 5 部电话要两两相连，则需要 10 对电线，见图 1-1(b)所示。显然，若 N 部电话要两两相连，就需要 $N(N-1)/2$ 对电线。当电话机的数量很大时，这种连接方法需要的电线数量就太大了（与电话机的数量的平方成正比）。于是人们认识到，要使得每一部电话能够很方便地和另一部电话进行通信，就应当使用

① 注：也有人将 ARPA 译为“高级研究计划署”。本书采用的译名取自清华大学编写的《英汉技术词典》，国防工业出版社，1978 年。ARPA 后来称为 DARPA，而 D 代表国防部(Defense)。

② 注：请读者注意：“结点”的英文名词是 node。虽然 node 有时也可译为“节点”，但这是指像天线上的驻波的节点，即像竹竿的“节”。在网络中的 node 的标准译名是“结点”而不是“节点”[MINGCI94]。这里的带方括弧的字符串表示可参考的文献，在本书的附录 H 中可查到。

电话交换机将这些电话连接起来，如图 1-1(c)所示。每一部电话都连接到交换机上，而交换机使用交换的方法，让电话用户彼此之间可以很方便地通信。一百多年来，电话交换机虽然经过多次更新换代，但交换的方式一直都是电路交换[①]。

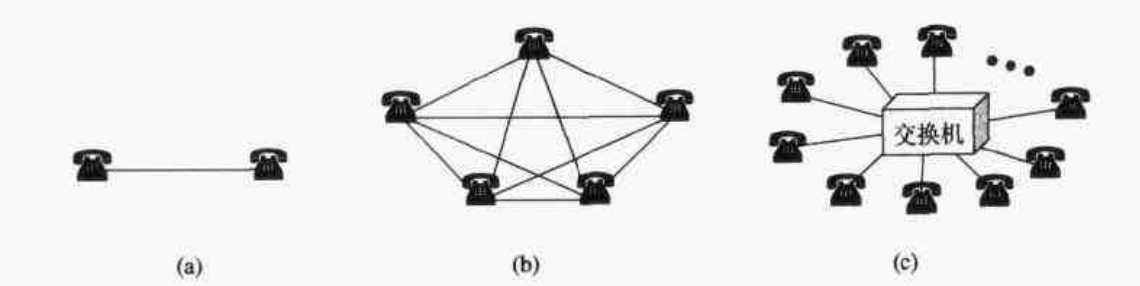


图 1-1　电话机的不同连接方法

当电话机的数量增多时，就要使用很多彼此连接起来的交换机来完成全网的交换任务。用这样的方法，就构成了覆盖全世界的电信网。

从通信资源的分配角度来看，“**交换**”就是按照某种方式动态地分配传输线路的资源。在使用电路交换打电话之前，必须先拨号建立连接。当拨号的信令通过许多交换机到达被叫用户所连接的交换机时，该交换机就向用户的电话机振铃。在被叫用户摘机且摘机信令传送回到主叫用户所连接的交换机后，呼叫即完成。这时，从主叫端到被叫端就建立了一条连接（物理通路）。此后主叫和被叫双方才能互相通电话。通话完毕挂机后，挂机信令告诉这些交换机，使交换机释放刚才使用的这条物理通路。这种必须经过“建立连接→通信→释放连接”三个步骤的连网方式称为**面向连接的**(connection-oriented)。**电路交换必定是面向连接的**。

图 1-2 为电路交换的示意图。为简单起见，图中没有区分市话交换机和长途电话交换机。应当注意的是，用户线归电话用户专用，而对交换机之间拥有大量话路的中继线则是许多用户共享的，正在通话的用户只占用了其中的一个话路，而**在通话的全部时间内，通话的两个用户始终占用端到端的固定传输带宽**。图中电话机 A 和 B 之间的通路共经过了四个交换机，而电话机 C 和 D 是属于同一个交换机的地理覆盖范围中的用户，因此这两个电话机之间建立的连接就不需要再经过其他的交换机。

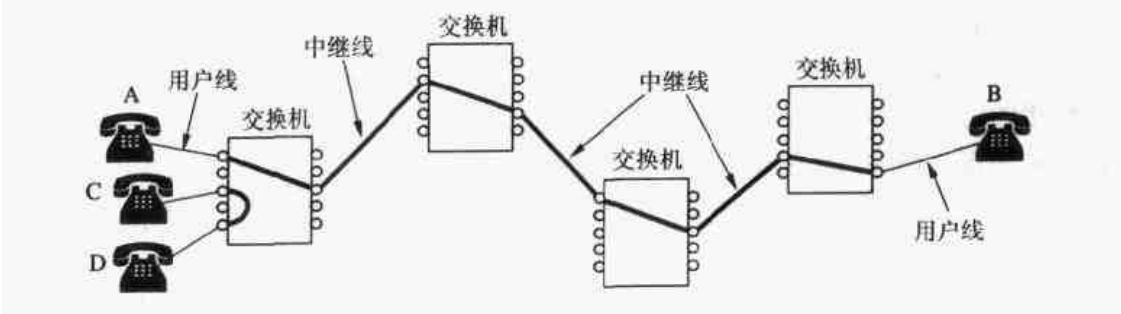


图 1-2　电路交换的示意图

当使用电路交换来传送计算机数据时，其**线路的传输效率往往很低**。这是因为计算机数

① 注：电路交换的特征是**基于位置**(position-based)。对于已数字化的网络，电路交换就是在某一位置的比特经交换后变更到另一个位置上。电路交换有多种形式。**空分交换**是交换比特流所经过的交换机的**端口号**，**时分交换**是交换比特所在的**时隙**，而**波分交换**则是交换荷载比特的光的**波长**。

据是突发式地出现在传输线路上，因此线路上真正用来传送数据的时间往往不到 10%甚至 1%。在绝大部分时间里，已被用户占用的通信线路实际上是空闲的。例如，当用户阅读终端屏幕上的信息或用键盘输入和编辑一份文件时，或计算机正在进行处理而结果尚未返回时，宝贵的通信线路资源实际上并未被利用而是白白被浪费了。

分组交换则采用**存储转发**技术[①]。图 1-3 画的是分组的概念。通常我们将欲发送的整块数据称为一个**报文**(message)。在发送报文之前，先将较长的报文划分成为一个个更小的等长数据段，例如，每个数据段为 1024 bit[②]。在每一个数据段前面，加上一些必要的控制信息组成的**首部**(header)后，就构成了一个**分组**(packet)。分组又称为“**包**”，而分组的首部也可称为“**包头**”。分组是在计算机网络中传送的数据单元。在一个分组中，“首部”是非常重要的，正是由于分组的首部包含了诸如目的地址和源地址等重要控制信息，每一个分组才能在分组交换网中独立地选择路由。因此，分组交换的特征是基于**标记**(label-based)。上述的分组首部就是一种标记。使用分组交换时，在传送数据之前可以不必先建立一条连接。这样就减少了建立连接和释放连接所需的开销，使得数据的传输效率更高。这种不先建立连接而随时可发送数据的连网方式，称为**无连接**(connectionless)式。分组交换还可以使用面向连接方式（如将在第 5 章讨论的几种广域网）。因此面向连接的网络不一定是电路交换的网络。

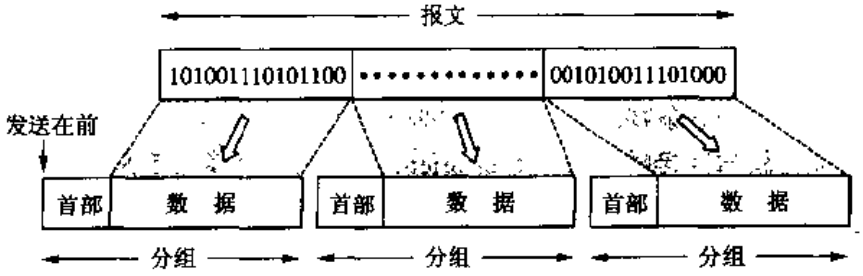


图 1-3 分组的概念

分组交换网由若干个**结点交换机**(node switch)和连接这些交换机的链路组成。图 1-4(a)是其示意图。用圆圈表示的结点交换机是网络的核心部件。从概念上讲，一个结点交换机就是一个小型计算机。图 1-4(b)和图 1-4(a)的表示方法是一样的，但强调了结点交换机具有多个端口的概念。端口就是结点交换机和外部线路相连接的地方。图 1-4 用一个方框表示结点交换机。我们应注意到，每一个结点交换机都有两组端口。一些小半圆表示的一组端口用来和计算机相连，所连接的链路速率较低。而一些小方框表示的一组端口则用高速链路和网络中其他的结点交换机的相连。图中 H_1～H_6 都是一些可进行通信的计算机，但在计算机网络中常称它们为**主机**(host)。在 ARPANET 建网初期，分组交换网中的结点交换机曾被称为**接口报文处理**

① 注：存储转发的概念最初是在 1964 年 8 月由巴兰(Baran)在美国兰德(Rand)公司的“论分布式通信”的研究报告中提出的。在 1962～1965 年，美国国防部远景研究规划局 DARPA 和英国的国家物理实验室 NPL 都在对新型的计算机通信网进行研究。1966 年 6 月，NPL 的戴维斯(Davies)首次提出“分组”(packet)这一名词[DAVI86]。1969 年 12 月，美国的分组交换网 ARPANET（当时仅 4 个结点）投入运行。从此，计算机网络的发展就进入了一个崭新的纪元。1973 年英国的 NPL 也开通了分组交换试验网。现在大家都公认 ARPANET 为分组交换网之父，并将分组交换网的出现作为现代电信时代的开始。除英美两国外，法国也在 1973 年开通其分组交换网 CYCLADES。

② 注：在本书中，bit 和 b 都表示“比特”。在计算机领域中，bit 又常常译为“位”。在许多情况下，“比特”和“位”是可以通用的。

机 IMP (Interface Message Processor)。但 IMP 这一名词现已不再使用。在图 1-4 中的主机和结点交换机都是计算机，但它们的作用明显不同。**主机是为用户进行信息处理的**，并且可以通过网络和其他的主机交换信息。**结点交换机则是进行分组交换的**，是用来转发分组的。各结点交换机之间也要经常交换路由信息，但这是为了进行路由选择，即为转发分组找出一条最好的路径。

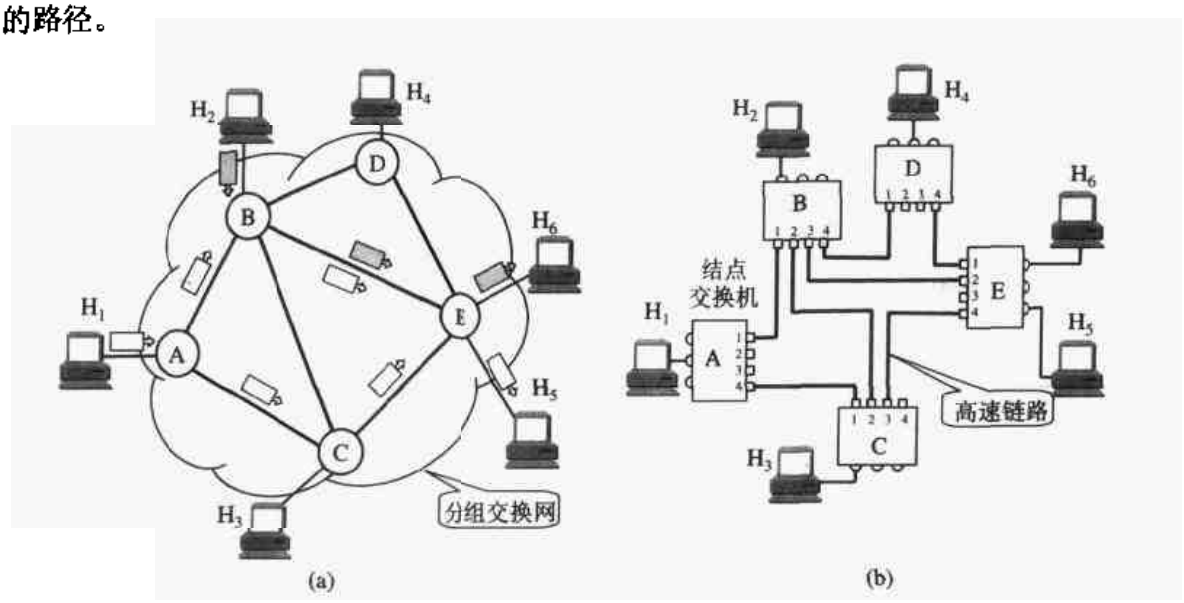


图 1-4　分组交换网的示意图。(a)分组交换网和主机；(b)具有两组端口的结点交换机

这里特别要强调的是，在结点交换机中的输入和输出端口之间是没有直接连线的。结点交换机处理分组的过程是：将收到的分组先放入缓存，再查找转发表，找出到某个目的地址应从哪个端口转发，然后由交换机构将该分组传递给适当的端口转发出去。

现在假定图 1-4(b)的主机 H_1 向主机 H_5 发送数据。主机 H_1 先将分组逐个地发往与它直接相连的结点交换机 A。此时，除链路 H_1–A 外，网内其他通信链路并不被目前通信的双方所占用。需要注意的是，即使是链路 H_1–A，也只是当分组正在此链路上传送时才被占用。在各分组传送之间的空闲时间，链路 H_1–A 仍可为其他主机发送的分组使用。

结点交换机 A 将主机 H_1 发来的分组放入缓存。假定从结点交换机 A 的转发表中查出应将该分组送到该结点交换机的端口 4。于是分组就经链路 A–C 到达结点交换机 C。当分组正在链路 A–C 传送时，该分组并不占用网络其他部分的资源。

结点交换机 C 继续按上述方式查找转发表，假定查出应从其端口 3 进行转发。于是分组又经结点交换机 C 的端口 3 向结点交换机 E 转发。当分组到达结点交换机 E 时，交换机 E 就将分组直接交给主机 H_5。

假定在某一个分组的传送过程中，链路 A-C 的通信量太大，那么结点交换机 A 可以将分组转发端口改为端口 1。于是分组就沿另一个路由到达结点交换机 B。交换机 B 再通过其端口 3 将分组转发到结点交换机 E，最后将分组送到主机 H_5。图 1-4(a)还画出了在网络中可同时有其他主机也在进行通信，如主机 H_2 经过结点交换机 B 和 E 与主机 H_6 通信。

这里要注意，结点交换机暂时存储的是一个个短分组，而不是整个的长报文。短分组是暂存在交换机的存储器（即内存）中而不是存储在磁盘中。这就保证了较高的交换速率。

在图中只画了两对主机(H_1 和 H_5，H_2 和 H_6)在进行通信。实际上，一个分组交换网可以容许很多主机同时进行通信，而一个主机中的多个进程（即正在运行中的多道程序）也可以

各自和不同主机中的不同进程进行通信。

在传送分组的过程中，由于采取了专门的措施，因而保证了数据的传送具有非常高的可靠性。当分组交换网中的某些结点或链路突然被破坏时，在各结点交换机中运行的路由选择协议(protocol)能够自动找到其他路径转发分组。这些将在下面有关章节中详细讨论。

从以上所述可知，采用存储转发的分组交换，实质上是采用了在数据通信的过程中断续（或动态）分配传输带宽的策略。这对传送突发式的计算机数据非常合适，使得通信线路的利用率大大提高了。

为了提高分组交换网的可靠性，常采用网状拓扑结构，使得当发生网络拥塞或少数结点、链路出现故障时，可灵活地改变路由而不致引起通信的中断或全网的瘫痪。此外，通信网络的主干线路往往由一些高速链路构成，这样就能以较高的数据率迅速地传送计算机数据。

综上所述，分组交换网的主要优点可归纳如表 1-1 所示。

表 1-1　分组交换的优点

优　点	所采用的手段
高效	在分组传输的过程中动态分配传输带宽，对通信链路是逐段占用
灵活	为每一个分组独立地选择转发路由
迅速	以分组作为传送单位，可以不先建立连接就能向其他主机发送分组；网络使用高速链路
可靠	完善的网络协议；分布式多路由的分组交换网，使网络有很好的生存性

分组交换也带来一些新的问题。例如，分组在各结点存储转发时需要排队，这就会造成一定的时延。当网络通信量过大时，这种时延也可能会很大。在表 1-1 中提到分组交换的优点之一是“迅速”，是指和电路交换相比时，分组交换可以省去建立连接所花费的时间，而且还可以在高速链路上以较高的数据率来传送数据。但分组交换网中的每一个结点又因存储转发产生了时延。因此，整个分组交换网是否能够比电路交换更快地传送数据，还取决于网络中的结点是否能够迅速地转发分组。这点在学完第 6 章后就会更加清楚。

分组交换网带来的另一个问题是各分组必须携带的控制信息也造成了一定的开销(overhead)。整个分组交换网还需要专门的管理和控制机制。

应当指出，从本质上讲，这种断续分配传输带宽的存储转发原理并非完全新的概念。自古代就有的邮政通信，就其本质来说也是属于存储转发方式。而在 20 世纪 40 年代，电报通信也采用了基于存储转发原理的报文交换(message switching)。在报文交换中心，一份份电报被接收下来，并穿成纸带。操作员以每份报文为单位，撕下纸带，根据报文的目的站地址，拿到相应的发报机转发出去。这种报文交换的时延较长，从几分钟到几小时不等。现在报文交换已经很少有人使用了。分组交换虽然也采用存储转发原理，但由于在交换结点上使用了电子计算机，且分组长度不大，完全可放在交换结点计算机的存储器中进行处理，这就使分组的转发非常迅速。例如 ARPANET 建网初期的经验表明，在正常的网络负荷下，当时横跨美国东西海岸的端到端平均时延小于 0.1 秒。这样，分组交换虽然采用了某些古老的交换原理，但实际上已变成了一种崭新的交换技术。

图 1-5 表示电路交换、报文交换和分组交换的主要区别。图中的 A 和 D 分别是源点和终点，而 B 和 C 是在 A 和 B 之间的中间结点。

从图 1-5 不难看出，若要连续传送大量的数据，且其传送时间远大于连接建立时间，则电路交换具有传输速率较快的优点。报文交换和分组交换不需要预先分配传输带宽，在传送

突发数据时可提高整个网络的信道利用率。分组交换比报文交换的时延小，但其结点交换机必须具有更强的处理能力。

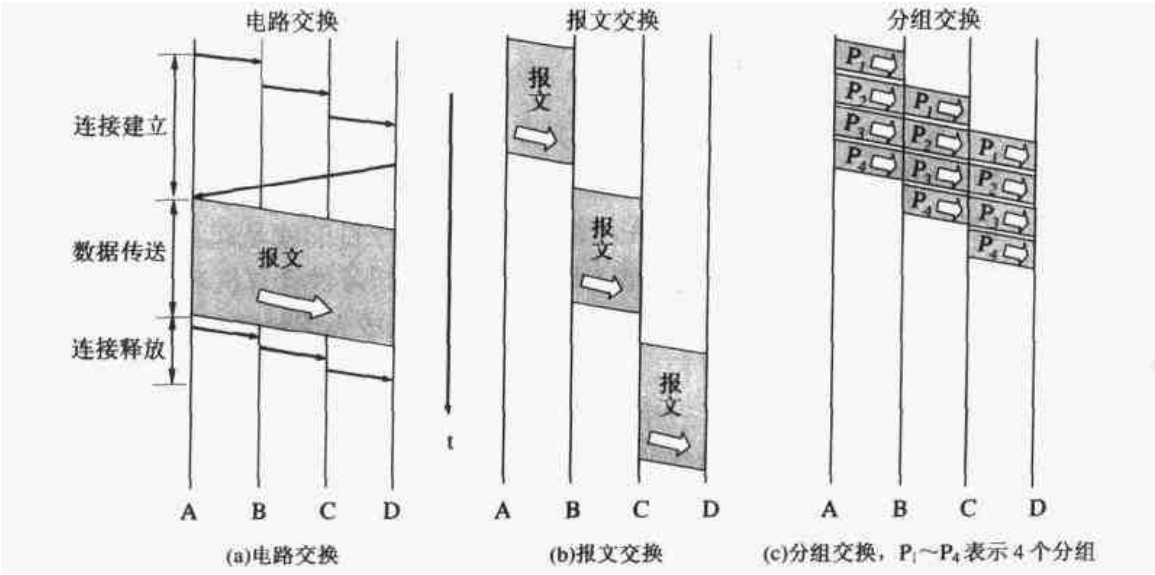


图 1-5　三种交换的比较

ARPANET 的试验成功使计算机网络的概念发生了根本的变化。图 1-6(a)画的是早期的面向终端的计算机网络，它是**以单个主机为中心**的星形网，各终端通过通信线路共享昂贵的中心主机的硬件和软件资源。但分组交换网则是**以网络为中心**，主机都处在网络的外围，见图 1-6(b)。用户通过分组交换网可共享连接在网络上的许多硬件和各种丰富的软件资源。在有些文献中，将分组交换网称为通信子网，而将用户的主机的集合称为资源子网。但是，在后面重点要讲述的传输控制协议/网际协议，即 TCP/IP 协议族中需要使用“子网划分”技术，而这里的“子网”和上述“通信子网”、“资源子网”中的子网的意思完全不同。为了不引起混乱，本书一般不使用“通信子网”和“资源子网”这两个名词。

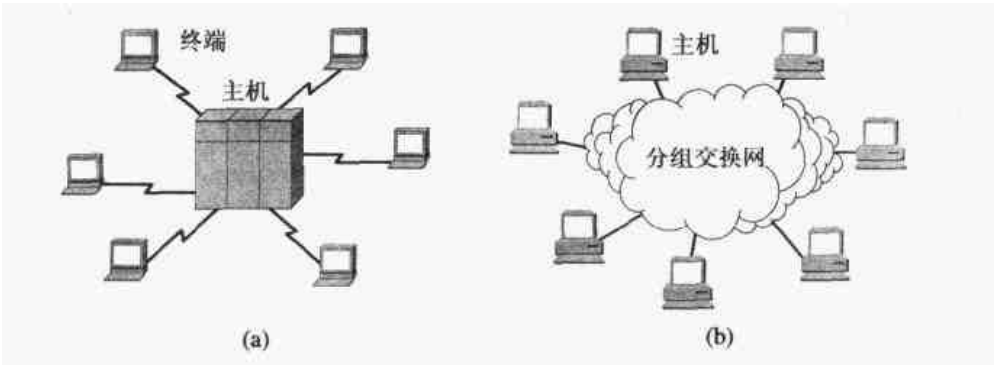


图 1-6　从以单个主机为中心(a)演变到以分组交换网为中心(b)

这种以分组交换网为中心的计算机网络比最初的面向终端的计算机网络的功能扩大了很多，成为 20 世纪 70 年代计算机网络的主要形式。

必须指出，分组交换网之所以能得到迅速的发展，很重要的一个原因就是：分组交换技术给用户带来了经济上的好处，其费用比使用电路交换更为低廉[ROBE82]。在美国，分组交换的计算机网络能如此迅速发展的其他因素是：

（1）已经建成了一个相当发达的电信网络作为物质基础；

（2）社会生产力的发展使整个社会对信息的处理、传递与交换有迫切的要求；

（3）及时开展了有关计算机网络的理论研究，特别是将排队论成功地用于计算机网络，并在试验网络上进行了现场实验。

有关排队论的要点见本书的附录 A。

1.2.2 因特网时代

下面我们要介绍因特网的发展概况

进入 20 世纪 80 年代末期以来，在网络领域最引人注目的就是起源于美国的**因特网**(Internet)的飞速发展。现在因特网已发展成为世界上最大的国际性计算机互联网[①]。由于因特网已影响到人们生活的各个方面，这就使得 20 世纪 90 年代成为**因特网时代**，或简称为**网络时代**。下面简单介绍因特网的发展过程。

因特网的基础结构大体上经历了三个阶段的演进。但这三个阶段在时间划分上并非截然分开而是有部分重叠的，这是因为网络的演进是逐渐的而不是突然的。

第一阶段是从单个网络 ARPANET 向互连网发展的过程。1969 年美国国防部创建的第一个分组交换网 ARPANET 最初只是一个单个的分组交换网（并不是一个互连网）。所有要连接在 ARPANET 上的主机都直接与就近的交换结点机相连。在 ARPANET 问世后，其规模一直增长很快。1984 年 ARPANET 上的主机已超过 1000 台。但到了 70 年代中期，人们已认识到不可能仅使用一个单独的网络来满足所有的通信问题。于是 ARPA 开始研究多种网络（如分组无线电网络）互连的技术，这就导致后来互连网的出现。1983 年 TCP/IP 协议成为 ARPANET 上的标准协议。同年，ARPANET 分解成两个网络。一个仍称为 ARPANET，是进行实验研究用的科研网。另一个是军用的计算机网络 MILNET (MILNET 拥有 ARPANET 当时的 113 个结点中的 68 个)。这样，在 1983～1984 年间因特网 Internet 就形成了。1990 年 ARPANET 正式宣布关闭，因为它的实验任务已经完成。

第二阶段的特点是建成了三级结构的因特网。ARPANET 的发展使美国国家科学基金会 NSF (National Science Foundation)认识到计算机网络对科学研究的重要性，因此从 1985 年起，美国国家科学基金会就围绕六个大型计算机中心建设计算机网络。1986 年，NSF 建立了国家科学基金网 NSFNET。它是一个三级计算机网络，分为**主干网**、**地区网**和**校园网**（见图 1-7）。这种三级计算机网络覆盖了全美国主要的大学和研究所。1987 年因特网上的主机超过

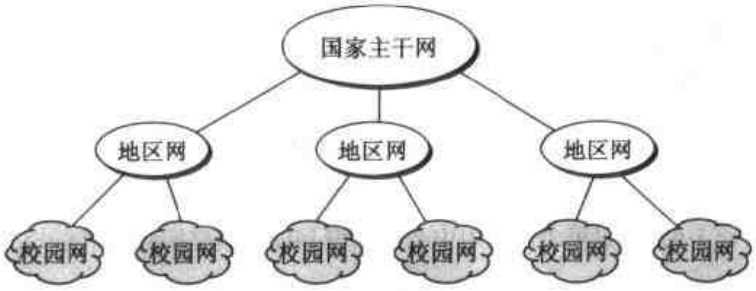


图 1-7 三级结构因特网

① 注：1994 年全国自然科学名词审定委员会公布的名词中，interconnection 是“互连”，interconnection network 是“互连网络”，internetworking 是“网际互连”[MINGCI94]。但 1997 年 8 月全国科学技术名词审定委员会在其推荐名（一）中，将 internet, internetwork, interconnection network 均推荐译名为“互联网”，而在注释中说“又称互连网”，即“**互联网**”与“**互连网**”这两个名词均可使用。请读者注意。

1 万台。最初，NSFNET 的主干网的速率不高，仅为 56kb/s。1989 年 NSFNET 主干网的速率提高到 1.544 Mb/s，即 T1 的速率，并且成为因特网中的主要部分。

1991 年，NSF 和美国的其他政府机构开始认识到，因特网必将扩大其使用范围，不会仅限于大学和研究机构。世界上的许多公司纷纷接入到因特网，使网络上的通信量急剧增大，每日传送的分组数达 10 亿个之多。因特网的容量已满足不了需要。于是美国政府决定将因特网的主干网转交给私人公司来经营，并开始对接入因特网的单位收费。1992 年因特网上的主机超过 1 百万台。1993 年因特网主干网的速率提高到 45 Mb/s (T3 速率)。不久，三级结构因特网（由美国政府资助的）又演进到现在第三阶段的多级结构因特网（由许多公司经营的）。因此，现在的因特网并不是某个单个组织所拥有的。

第三阶段的多级结构因特网是这样形成的。从 1993 年开始，由美国政府资助的 NSFNET 逐渐被若干个商用的**因特网主干网**替代。这种主干网也叫做**服务提供者网络**(service provider network)。任何人只要向**因特网服务提供者** ISP (Internet Service Provider)交纳规定的费用，就可通过该 ISP 接入到因特网。考虑到因特网商用化后可能会出现很多的 ISP，为了使不同 ISP 经营的网络都能够互通，在 1994 年开始创建了 4 个**网络接入点** NAP (Network Access Point)，分别由 4 个电信公司经营。所谓网络接入点 NAP 就是用来交换因特网上流量的结点。在 NAP 中安装有性能很好的交换设施（例如，使用 ATM 交换技术）。到本世纪初，美国的 NAP 的数量已达到十几个。

这样，从 1994 年到现在，因特网逐渐演变成多级结构网络。NAP 是最高级的接入点。它主要是向不同的 ISP 提供交换设施，使它们能够互相通信。NAP 又称为对**等点**(peering point)(图 1-8)。

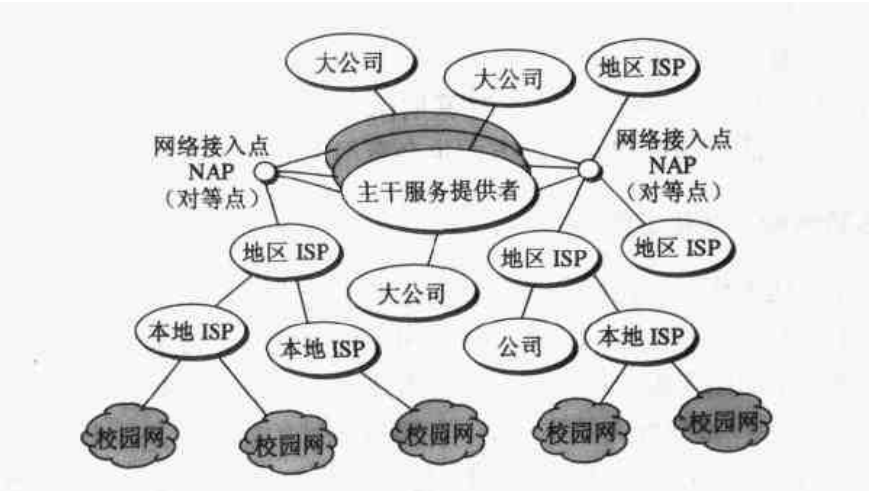


图 1-8　多级结构的因特网

从图 1-8 可看出，今日的因特网已经很难对其网络结构给出很细致的描述，但大致上可将因特网分为以下五个接入级。第一级是网络接入点 NAP，第二级是由多个公司经营的国家主干网，第三级是地区 ISP（商用的、国家的），第四级是本地 ISP，第五级是校园网、企业或家庭 PC 机上网用户。

到 1996 年速率为 155 Mb/s 的主干网 vBNS (very high-speed Backbone Service)建成。1998 年又开始建造更快的主干网 Abilene，数据率最高达 2.5 Gb/s。1999 年 MCI 和 Worldcom 公司开始将美国的因特网主干网速率提高到 2.5Gb/s。到 1999 年底，因特网上注册的主机已超过 1 千万台。

因特网已经成为世界上规模最大和增长速率最快的计算机网络，没有人能够准确说出因特网

究竟有多大。因特网的迅猛发展始于20世纪90年代。由欧洲原子核研究组织CERN开发的**万维网WWW** (World Wide Web)被广泛使用在因特网上，大大方便了广大非网络专业人员对网络的使用，成为因特网的这种指数级增长的主要驱动力。万维网的站点数目也急剧增长。1993年底只有627个，1994年底就超过1万个，1996年底超过60万个，1997年底超过160万个，而1999年底则超过了950万个，上网用户数则超过2亿。在因特网上的数据通信量每月约增加10%。

表1-2是因特网上的网络数、主机数、用户数和管理机构数的简单概括[COME00]。

表1-2 因特网的发展情况概况

	网络数	主机数	用户数	管理机构数
1980	10	10^2	10^2	10^0
1990	10^2	10^5	10^6	10^1
2000	10^5	10^7	10^8	10^2

由于因特网存在着技术上和功能上的不足，加上用户数量猛增，使得现有的因特网不堪重负。因此1996年美国的一些研究机构和34所大学提出研制和建造新一代因特网的设想，并宣布在今后5年内用5亿美元的联邦资金实施**“下一代因特网计划”**，即**“NGI计划”** (Next Generation Internet Initiative)。

NGI计划要实现的一个目标是：开发下一代网络结构，以比现有的因特网高100倍的速率连接至少100个研究机构，以比现有的因特网高1000倍的速率连接10个类似的网点。其端到端的传输速率要超过100Mb/s至10Gb/s。另一个目标是使用更加先进的网络服务技术和开发许多带有革命性的应用，如远程医疗、远程教育、有关能源和地球系统的研究、高性能的全球通信、环境监测和预报、紧急情况处理等。NGI计划将使用超高速全光网络，能实现更快速的交换和路由选择，同时具有为一些**实时**(real time)应用保留带宽的能力。在整个因特网的管理和保证信息的可靠性和安全性方面也会有很大的改进。

1.2.3 关于因特网的标准化工作

因特网的标准化工作对因特网的发展起到了非常重要的作用。我们知道，标准化工作的好坏对一种技术的发展有着很大的影响。缺乏国际标准将会使技术的发展处于比较混乱的状态，而盲目自由竞争的结果很可能形成多种技术体制并存且互不兼容的状态（如过去形成的彩电三大制式），给用户带来较大的不方便。但国际标准的制定又是一个非常复杂的问题，这里既有很多技术问题，也有很多属于非技术问题，如不同厂商之间经济利益的争夺问题等。标准制定的时机也很重要。标准制定得过早，由于技术还没有发展到成熟水平，会使技术比较陈旧的标准限制了产品的技术水平。其结果是以后不得不再次修订标准，造成浪费。反之，若标准制定得太迟，也会使技术的发展无章可循，造成产品的互不兼容，因而也会影响技术的发展。因特网在制定其标准上很有特色。它的一个很大的特点是面向公众。因特网所有的技术文档都可从因特网上免费下载（具体的网址见附录H），而且任何人都可以用电子邮件随时发表对某个文档的意见或建议。这种方式对因特网的迅速发展影响很大。

1992年由于因特网不再归美国政府管辖，因此成立了一个国际性组织叫做**因特网协会** (Internet Society，简称为ISOC)，以便对因特网进行全面管理以及在世界范围内促进其发展和使用。ISOC下面有一个技术组织叫做**因特网体系结构委员会 IAB** (Internet Architecture

Board)[①]，负责管理因特网有关协议的开发。IAB 下面又设有两个工程部：

（1）**因特网工程部 IETF** (Internet Engineering Task Force)

IETF 是由许多**工作组** WG (Working Group)组成的论坛(forum)，具体工作由**因特网工程指导小组** IESG (Internet Engineering Steering Group)管理。这些工作组划分为若干个领域(area)，每个领域集中研究某一特定的短期和中期的工程问题，主要是针对协议的开发和标准化。

（2）**因特网研究部 IRTF** (Internet Research Task Force)

IRTF 是由一些**研究组** RG (Research Group)组成的论坛，具体工作由**因特网研究指导小组** IRSG(Internet Reseach Steering Group)管理。IRTF 的任务是进行理论方面研究和开发一些需要长期考虑的问题。

上述的这些机构之间的关系见图 1-9 所示。

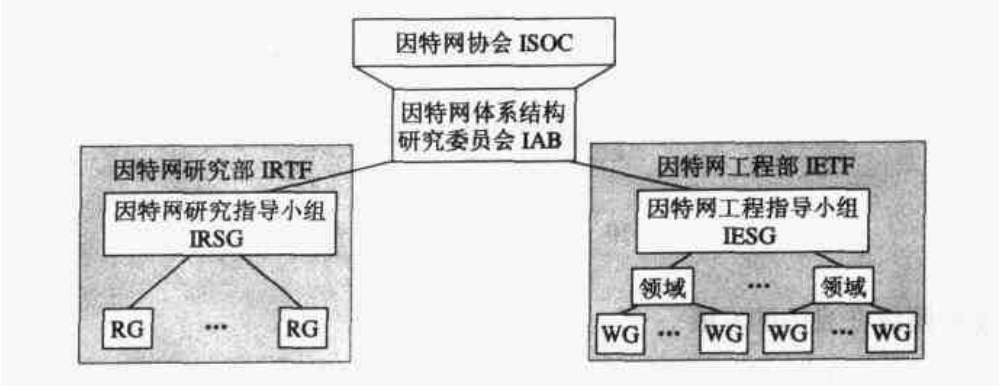


图 1-9　因特网的管理机构

所有的因特网标准都是以 RFC 的形式在因特网上发表。RFC (Request For Comments)的意思就是“请求评论”。所有的 RFC 文档都可从因特网上免费下载[W-RFC][②]。但应注意，并非所有的 RFC 文档都是因特网标准，只有一小部分 RFC 文档最后才能变成因特网标准。RFC 按收到时间的先后从小到大编上序号（即 RFC xxxx，这里的 xxxx 是阿拉伯数字）。一个 RFC 文档更新后就使用一个新的编号，并在文档中指出原来老编号的 RFC 文档已成为陈旧的。例如，2001 年 8 月公布了因特网正式协议标准[RFC 2900]，此文档注明了：[RFC 2800]已变成为陈旧的。但到了 11 月，[RFC 2900]文档又更新了，新文档的编号是[RFC 3000]，文档又注明：[RFC 2900]已变成为陈旧的。现有的 RFC 文档中有不少已变为陈旧的，在参考时应当注意。

制订因特网的正式标准要经过以下的四个阶段：

（1）**因特网草案**(Internet Draft)——在这个阶段还不是 RFC 文档。

（2）**建议标准**(Proposed Standard)——从这个阶段开始就成为 RFC 文档。

（3）**草案标准**(Draft Standard)

（4）**因特网标准**(Internet Standard)。

因特网草案的有效期只有六个月。只有到了建议标准阶段才以 RFC 文档形式发表。本书的许多内容都注明其相关的 RFC 文档号以便进一步学习。

---

① 注：最初的 IAB 中的 A 曾经代表 Activities(活动)。在一些旧的 RFC 中使用的是这个旧名词。

② 注：作者已将 2003 年 1 月以前发表的全部 RFC 文档放在本书的光盘内，供读者查阅。

除了以上几种 RFC 外，还有三种 RFC，即历史的、实验的和提供信息的。历史的 RFC 或者是被后来的规约所取代，或者是从未到达必要的成熟等级因而未变成为因特网标准。实验的 RFC 表示其工作属于正在实验的情况。实验的 RFC 不能够在任何实用的因特网服务中进行实现。提供信息的 RFC 包括与因特网有关的一般的、历史的或指导的信息。这种 RFC 通常是由非因特网的组织中的某个人（例如，设备的卖主）写出的。图 1-10 表示各种 RFC 之间的关系。

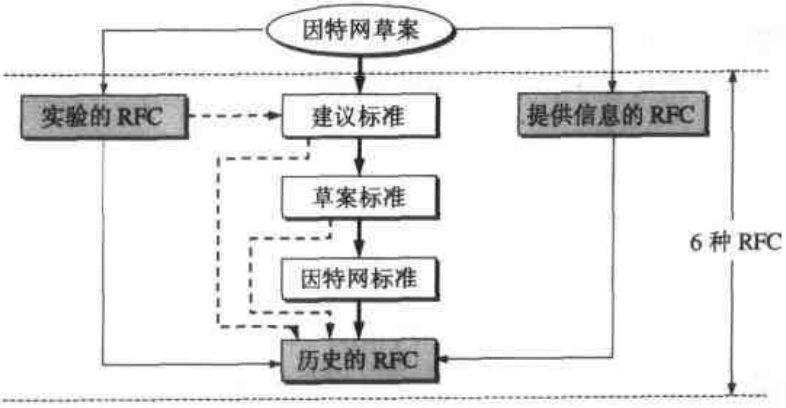


图 1-10　各种 RFC 之间的关系

1.2.4　计算机网络在我国的发展

下面简单介绍一下计算机网络在我国的发展情况。

最早着手建设专用计算机广域网的是铁道部。铁道部在 1980 年即开始进行计算机联网实验。1989 年 11 月我国第一个公用分组交换网 CNPAC 建成运行。CNPAC 分组交换网由 3 个分组结点交换机、8 个集中器和一个双机组成的网络管理中心所组成。1993 年 9 月建成新的中国公用分组交换网，并改称为 CHINAPAC，由国家主干网和各省、区、市的省内网组成。在北京、上海设有国际出入口。

在 20 世纪 80 年代后期，公安、银行、军队以及其他一些部门也相继建立了各自的专用计算机广域网。这对迅速传递重要的数据信息起着重要的作用。

除了上述的广域网外，从 20 世纪 80 年代起，国内的许多单位都陆续安装了大量的局域网。局域网的价格便宜，其所有权和使用权都属于本单位，因此便于开发、管理和维护。局域网的发展很快，对各行各业的管理现代化和办公自动化已起了积极的作用。

这里应当特别提到的是 1994 年 4 月 20 日我国用 64 kb/s 专线正式连入因特网。从此，我国被国际上正式承认为接入因特网的国家。同年 5 月中国科学院高能物理研究所设立了我国的第一个万维网服务器。在 9 月中国公用计算机互联网 CHINANET 正式启动。到目前为止，我国陆续建造了基于因特网技术的并可以和因特网互连的 9 个全国范围的公用计算机网络。这就是：

（1）中国公用计算机互联网 CHINANET

（2）中国教育和科研计算机网 CERNET

（3）中国科学技术网 CSTNET

（4）中国联通互联网 UNINET

（5）中国网通公用互联网 CNCNET

（6）中国国际经济贸易互联网 CIETNET

（7）中国移动互联网 CMNET

（8）中国长城互联网 CGWNET（建设中）

（9）中国卫星集团互联网 CSNET（建设中）

此外，还有一个中国高速互连研究试验网 NSFnet，是中国科学院、北京大学、清华大学等单位在北京中关村地区建造的为研究因特网新技术的高速网络。

上述这些基于因特网技术的计算机网络发展得非常快，几乎每个月都有新的发展，请读者经常在有关网站上查找这些计算机网络的有关数据（如用户数、网站数、主干网带宽等）。

表 1-3 是中国互联网络信息中心公布的我国最近几年来因特网的发展情况。

表 1-3 我国因特网的发展情况

统计时间	上网计算机数（万）	上网用户数（万）	cn 下注册的域名数	WWW 站点数	国际线路总容量 (Mb/s)
1997.10	29.9	62	4066	1500	25.408
1999.1	74.7	210	18396	5300	143.256
2000.1	350	890	48695	15153	351
2001.1	892	2250	122099	265405	2799
2002.1	1254	3370	127319	277100	7597.5
2003.1	2083	5910	179544	371600	9380

1.3 计算机网络的分类

1.3.1 计算机网络的不同定义

计算机网络的精确定义并未统一。

关于计算机网络的最简单的定义是：**一些互相连接的、自治的计算机的集合**[TANE96]。若按此定义，则早期的面向终端的网络都不能算是计算机网络，而只能称为联机系统（因为那时的许多终端不能算是自治的计算机）。但随着硬件价格的下降，许多终端都具有一定的智能，因而“终端”和“自治的计算机”逐渐失去了严格的界限。因此，若用微型计算机作为终端使用，按上述定义，则早期的那种面向终端的网络也可称为计算机网络。

最简单的计算机网络就只有两台计算机和连接它们的一条链路，即两个结点和一条链路。因为没有第三台计算机，因此不存在交换的问题。

最庞大的计算机网络就是因特网。它由非常多的计算机网络通过许多路由器互连而成。因此因特网也称为“**网络的网络**”(network of networks)。

有时我们也可能见到“计算机通信网”这一名词。但这个名词容易使人误认为这是一种专门为了通信而设计的计算机网络。计算机网络当然应具有通信的功能，但这种通信功能并非计算机网络最主要的功能。因此本书不使用“计算机通信网”这一名词。

“计算机通信”与“数据通信”这两个名词也常混用。早期的数据通信与现代的计算机通信显然是有区别的。但随着技术的进步，数据通信的含义也在发生变化。因此，可以认为计算机通信与数据通信是可混用的名词。美国的著名期刊 *Data Communications* 所刊登的文

章现在也都是计算机网络领域的文章。在许多情况下，数据通信网往往指的是计算机网络中的分组交换网。

最后还要指出，计算机网络与分布式计算机系统虽然有相同之处，但二者并不等同。**分布式计算机系统**的最主要特点是**整个系统中的各计算机对用户都是透明的**。也就是说，对用户来说，这种分布式计算机系统就好像只有一个计算机一样。用户通过键入命令就可以运行程序，但用户并不知道是哪一个计算机在为他运行程序。是操作系统为用户选择一个最合适的计算机来运行其程序，并将运行的结果传送到合适的地方。这些都不需要用户的干预。而计算机网络则不同。用户必须先在欲运行程序的计算机进行登录，然后按照该计算机的地址，将程序通过计算机网络传送到该计算机去运行。最后，根据用户的命令将结果传送到指定的计算机。由此可见，计算机网络并不等同于分布式计算机系统。二者的区别主要是软件的不同。一般说来，分布式计算机系统是计算机网络的一个特例。当然，也有一些分布式计算机系统根本就不是计算机网络（例如，分布式计算机）。

1.3.2 几种不同的分类方法

可以从不同的角度对计算机网络进行分类。

1．从网络的交换功能进行分类

网络的设计者常从交换的功能来将网络分类。常用的交换方法有：

（1）**电路交换**；

（2）**报文交换**；

（3）**分组交换**；

（4）**混合交换**。

前三种交换方式已简单介绍过了。混合交换是在一个数据网中同时采用电路交换和分组交换。

2．从网络的作用范围进行分类

有时需要从网络的作用范围进行如下的分类：

（1）**广域网 WAN** (Wide Area Network)　广域网的作用范围通常为几十到几千公里，因而有时也称为**远程网**(long haul network)。广域网是因特网的核心部分，其任务是通过长距离（例如，跨越不同的国家）运送主机所发送的数据。连接广域网各结点交换机的链路一般都是高速链路，具有较大的通信容量。广域网的主要内容将在第 5 章中讨论。

（2）**局域网 LAN** (Local Area Network)　局域网一般用微型计算机或工作站通过高速通信线路相连（速率通常在 10 Mb/s 以上），但地理上则局限在较小的范围（如 1 km 左右）。在局域网发展的初期，一个学校或工厂往往只拥有一个局域网，但现在局域网已被广泛使用，一个学校或企业大都拥有许多个局域网。因此，又出现了**校园网**或**企业网**的名词。我们将在第 4 章详细讨论局域网。

（3）**城域网 MAN** (Metropolitan Area Network)　城域网的作用范围在广域网和局域网之间，例如作用范围是一个城市，可跨越几个街区或甚至整个的城市。城域网可以为一个或几个单位所拥有，但也可以是一种公用设施，用来将多个局域网进行互连。城域网的传送速率比局域网的更高，但作用距离约为 5～50 km。从网络的层次上看，城域网是广域网和局域网

（或校园网）之间的桥接区（见图 1-11）。城域网因为要和很多种的局域网（或校园网）连接，因此必须适应多种业务、多种网络协议以及多种的数据传输速率，并要保证能够很方便地将各种局域网（或校园网）连接到广域网。城域网内部的结点之间或城域网之间也需要有高速链路相连接，并且城域网的范围也逐渐在扩大，因此现在城域网在某些地方有点像范围较小的广域网。城域网在最近一段时期发展较快。从技术上看，目前很多城域网采用的是以太网技术。由于城域网与局域网使用相同的体系结构，有时也常并入局域网的范围进行讨论。

（4）**接入网** AN (Access Network)　接入网又称为**本地接入网**或**居民接入网**，它也是近年来由于用户对高速上网需求的增加而出现的一种网络技术。从图 1-11 可看出，接入网是局域网（或校园网）和城域网之间的桥接区。接入网提供多种高速接入技术，使用户接入到因特网的瓶颈得到某种程度上的解决。我们将在第 10 章讨论接入网。

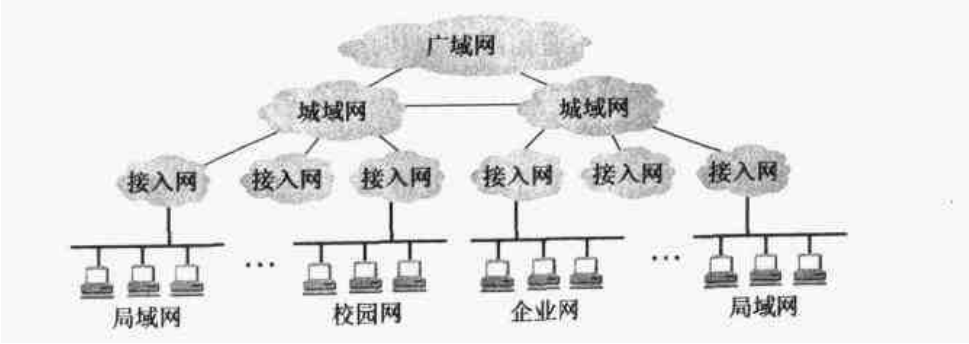


图 1-11　广域网、城域网、接入网和局域网的关系

若中央处理机之间的距离非常近（如仅 1 m 的数量级或甚至更小些），则一般就称之为**多处理机系统**而不称它为计算机网络。关于多个处理机互连的系统按其大小的大致分类见表 1-4 所示。

表 1-4　多个处理机互连的系统按其大小的分类

处理机之间的典型距离	处理机所在的范围	实　例
0.1 m	印制板	数据流计算机
1 m	系统	多处理机
10 m	房间	局域网、校园网、企业网
100 m	建筑物	
1 km	校园	
10 km	城市	城域网
100 km	国家	广域网
1000 km	国家、洲	广域网，互连的广域网

3. 从网络的使用者进行分类

这可以划分为公用网和专用网。

（1）**公用网**(public network)　这是指国家的电信公司（国有或私有）出资建造的大型网络。“公用”的意思就是所有愿意按电信公司的规定交纳费用的人都可以使用。因此公用网也可称为**公众网**。

（2）**专用网**(private network)　这是某个部门为本单位的特殊业务工作的需要而建造的网络。这种网络不向本单位以外的人提供服务。例如，军队、铁路、电力等系统均有本系统的

专用网。

公用网和专用网都可以传送多种业务。如传送的是计算机数据，则分别是公用计算机网络和专用计算机网络。

1.4 计算机网络的主要性能指标

计算机网络的最主要的两个性能指标就是**带宽**与**时延**。下面分别介绍这两个指标的含义。

1.4.1 带宽

“**带宽**”(bandwidth)本来是指某个**信号具有的频带宽度**。我们知道，一个特定的信号往往是由许多不同的频率成份组成的。因此。一个信号的带宽是指该信号的各种不同频率成份所占据的频率范围。例如，在传统的通信线路上传送的电话信号的标准带宽是 3.1 kHz（从 300 Hz 到 3.4 kHz，即话音的主要成分的频率范围）。**带宽的单位**是**赫**（或**千赫**、**兆赫**等）。

在过去很长的一段时间，通信的主干线路都是用来传送**模拟信号**（即连续变化的信号）。因此，表示通信线路允许通过的信号频带范围就称为线路的**带宽**（或**通频带**）。

当通信线路用来传送数字信号时，**数据率**就应当成为数字信道最重要的指标。但习惯上，**人们愿意将“带宽”作为数字信道的“数据率”的同义语**，尽管这种叫法不太严格。但应注意，数字信道的数据率的单位并不采用频率的单位。

数字信道传**送数字信号的速率**称为**数据率**或**比特率**。比特(bit，可简写为 b)是计算机中的数据的最小单元，它也是**信息量的度量单位**。英文字 bit 来源于 binary digit，意思是一个“**二进制数字**”，因此一个比特就是二进制数字中的一个 1 或 0。这样，网络或链路的**带宽的单位就是**“**比特每秒**”，或 b/s (bit/s)[①]，而更常用的带宽单位是千比每秒 kb/s、兆比每秒 Mb/s (10^6 b/s)、吉比每秒 Gb/s (10^9b/s)或太比每秒 Tb/s (10^{12}b/s)。现在人们常用更简单的并且是很不严格的记法来描述网络或链路的带宽，如“线路的带宽是 10M 或 10G”，而省略了后面的 b/s，它的意思就表示数据率（即带宽）为 10 Mb/s 或 10Gb/s。

正是因为带宽代表数字信号的发送速率，因此带宽有时也称为**吞吐量**(throughput)。在实际应用中，吞吐量常用每秒发送的比特数（或字节数、帧数）来表示。

顺便指出，在通信领域和计算机领域，对数量单位“千”、“兆”和“吉”等的英文缩写在有些情况下意思略有不同。如计算机中的数据量往往用字节作为度量的单位。一个字节(byte)代表 8 个比特，它的缩写是大写 B。“千字节”的“千”用大写 K 表示，它等于 2^{10}，即 1024，而不是 1000。同样，在计算机中，1 MB 或 1 GB 也并非表示 10^6 或 10^9 个字节，而是表示 2^{20}(1 048 576)或 2^{30}(1 073 741 824)个字节。在通信领域小写的 k 表示 10^3，即准确的 1000 而不是 1024。但有的书也不这样严格区分，而是用大写 K 既表示 1000 又表示 1024。

如果我们在网络中某一个点上观察数字信号流随时间的变化，那么信号在时间轴上的宽度就随着带宽的增大而变窄。例如，当信号为 1 和 0 相间的脉冲时，在带宽为 1 Mb/s 链路上，每一个比特在时间轴上的宽度为 1 μs，但在带宽为 4 Mb/s 的链路上，每一个比特在时间轴上的宽度就减小到原来的四分之一，即只有 0.25 μs（见图 1-12）。

① 注：另一种常用的写法是将 b/s 写为 bps (即 bit per second)。

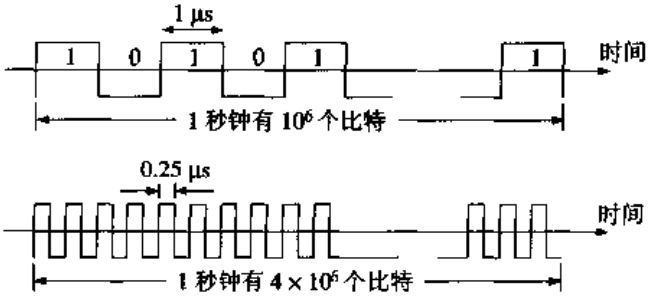


图 1-12　随着带宽的增大，数字信号在时间轴上的宽度就越窄

1.4.2　时延

时延(delay 或 latency)是指一个报文或分组从一个网络（或一条链路）的一端传送到另一端所需的时间。

需要注意的是，时延是由以下几个不同的部分组成的：

（1）**发送时延**　**发送时延是结点在发送数据时使数据块从结点进入到传输媒体所需要的时间**，也就是从数据块的第一个比特开始发送算起，到最后一个比特发送完毕所需的时间。发送时延又称为**传输时延**，它的计算公式是：

$$\text{发送时延} = \frac{\text{数据块长度}}{\text{信道带宽}} \tag{1-1}$$

信道带宽就是数据在信道上的发送速率，它也常称为数据在信道上的**传输速率**。

（2）**传播时延**　**传播时延是电磁波在信道中需要传播一定的距离而花费的时间**。传播时延的计算公式是：

$$\text{传播时延} = \frac{\text{信道长度}}{\text{电磁波在信道上的传播速率}} \tag{1-2}$$

电磁波在自由空间的传播速率是光速，即 3.0×10^5 km/s。电磁波在网络传输媒体中的传播速率比在自由空间要略低一些：在铜线电缆中的传播速率约为 2.3×10^5 km/s，在光纤中的传播速率约为 2.0×10^5 km/s。例如，1000 km 长的光纤线路产生的传播时延大约为 5 ms。

从以上讨论可以看出，信号**传输**速率（即**发送**速率）和电磁波在信道上的**传播**速率是两个完全不同的概念，因此不能将发送时延和传播时延弄混。

（3）**处理时延**　这是数据在交换结点为存储转发而进行一些必要的处理所花费的时间。在结点缓存队列中分组排队所经历的时延是处理时延中的重要组成部分。因此，处理时延的长短往往取决于网络中当时的通信量。当网络的通信量很大时，还会发生队列溢出，使分组丢失，这相当于处理时延为无穷大。有时可用**排队时延**作为处理时延。

这样，数据经历的总时延就是以上三种时延之和：

$$\text{总时延} = \text{传播时延} + \text{发送时延} + \text{处理时延} \tag{1-3}$$

图 1-13 画出了三种时延所产生的地方，希望读者能够更好地分清这三种时延。

我们在这里要指出，在总时延中，究竟是哪一种时延占主导地位，必须具体分析。现在我们暂时忽略处理时延。假定有一个长度为 100MB 的数据块（这里的 M 显然不是指 10^6 而是指 2^{20}，即 1 048 576。B 是字节，1 字节有 8 个比特）。在带宽为 1Mb/s 的信道上（这里的 M 是 10^6）



图 1-13 三种时延产生的地方不一样

的发送时延是 $100 \times 1\,048\,576 \times 8 \div 10^6 = 838.9$ s，即将近要用 14 分钟才能把这样大的数据块发送完毕。然而若将这样的数据用光纤传送到 1000 km 远的计算机，那么每一个比特在 1000 km 的光纤上只需用 5 ms 就能传送到目的地。因此对于这种情况，发送时延占主导地位。如果我们将传播距离减小到 1 km，那么传播时延也会相应地减小到原来数值的千分之一。然而由于传播时延在总时延中的比重是微不足道的，因此总时延的数值基本上还是原来由发送时延来决定。

再看一个例子。要传送的数据仅有 1 个字节（即在键盘上键入一个字符，共 8 bit）。在 1 Mb/s 的信道上的发送时延是 $8 \div 10^6 = 8 \times 10^{-6}$ s $= 8$ μs。当传播时延为 5 ms 时，总时延为 5.008 ms。显然，在这种情况下，传播时延决定了总时延。这时，即使将信道的带宽提高到 1000 倍（即将数据的发送速率提高到 1 Gb/s），总时延也不能减小多少。这个例子告诉我们，不能笼统地认为："数据的发送速率越高传送得就越快"。这是因为数据传送的总时延是由公式(1-3)右端的三项时延组成的，不能仅考虑发送时延一项。

必须强调指出，初学网络的人容易产生这样错误的概念，就是"在高速链路（或高带宽链路）上，比特应当跑得更快些"。但这是不对的。我们知道，汽车在路面质量很好的高速公路上可明显地提高行驶速率。然而**对于高速网络链路，我们提高的仅仅是数据的发送速率而不是比特在链路上的传播速率**。荷载信息的电磁波在通信线路上的传播速率（这是光速的数量级）与数据的发送速率并无关系。或者说，比特的传播时延与链路的带宽无关。**提高链路带宽只是减小了数据的发送时延**。还有一点也应当注意，就是数据的发送速率的单位是每秒发送多少个比特，是指某个点或某个端口上的发送速率。而传播速率的单位是每秒传播多少公里，是指传输线路上比特的传播速率。因此，通常所说的"光纤信道的传输速率高"是指向光纤信道的发送数据的速率可以很高，而光纤信道的传播速率实际上还要比铜线的传播速率还略低一点。这是因为经过测量得知，光在光纤中的传播速率和电磁波在铜线（如 5 类线）中的传播速率分别为每秒 20.5 万公里和每秒 23.1 万公里。上述这个概念请读者务必弄清。

1.4.3 时延带宽积和往返时延

将以上讨论的网络性能的两个度量——传播时延和带宽——相乘，就得到另一个很有用的度量：**传播时延带宽积**，即

$$\text{时延带宽积} = \text{传播时延} \times \text{带宽}$$

我们可以用图 1-14 的示意图来表示时延带宽积。这是一个代表链路的圆柱形管道，管道的长度是链路的传播时延（请注意，现在以**时间**作为单位来表示链路长度），而管道的截面积是链路的带宽。因此时延带宽积就表示这个管道的体积，表示这样的链路可容纳多少个比特。例如，设某段链路的传播时延为 20 ms，带宽为 10 Mb/s。算出时延带宽积 $= 20 \times 10^{-3} \times 10 \times 10^6 = 2 \times 10^5$ bit。这就表示，若发送端连续发送数据，则在发送的第一个比特即将达到终点时，发送端就已经发送了 20 万个比特，而这 20 万个比特都正在链路上传输。

因此，链路的时延带宽积又称为**以比特为单位的链路长度**。



图 1-14　链路像一条空心管道

有时在发送端和接收端之间相隔有好几个网络。发送端发送出去的数据要经过多次转发才能达到接收端。这时我们仍然可以使用上面这样的从发送端到接收端的传输管道，以及使用时延带宽积这个度量。但这时管道的时延就不再仅仅是网络的传播时延，而是从发送端到接收端的所有时延的总和，包括在所有各中间结点引起的处理时延和发送时延。这时的管道只是一种抽象的概念，而管道中的比特数表示从发送端发出的但尚未达到接收端的比特。

在计算机网络中，**往返时延** RTT (Round-Trip Time)也是一个重要的性能指标，它表示从发送端发送数据开始，到发送端收到来自接收端的确认（接收端收到数据后便立即发送确认），总共经历的时延。对于上述例子，往返时延 RTT 是 40 ms，而往返时延和带宽的乘积是 4×10^5(bit)。对于复杂的互联网，往返时延要包括各中间结点的处理时延和转发数据时的发送时延。

往返时延带宽积的意义就是当发送端连续发送数据时，在收到对方的确认之前，就已经将这样多的比特发送到链路上了。对于上述例子，如果数据的传送终点及时发现了差错，那么发送端得知这一信息时，即使立即停止发送，也已经发送了 40 万个比特了。

对于一条正在传送数据的链路，只有在代表链路的管道都充满比特时，链路才得到充分的利用。

1.5　计算机网络的体系结构

在计算机网络的基本概念中，分层次的体系结构是最基本的。因此我们在这里对计算机网络的体系结构进行简单的阐述。计算机网络体系结构的抽象概念较多，在学习时要多思考。这些概念对后面的学习很有帮助。

1.5.1　计算机网络体系结构的形成

计算机网络是个非常复杂的系统。为了说明这一点，可以设想一个最简单的情况：连接在网络上的两台计算机要互相传送文件。

显然，在这两台计算机之间必须有一条传送数据的通路。但这还远远不够。至少还有以下几件工作需要去完成：

（1）发起通信的计算机必须将数据通信的通路进行**激活**(activate)。所谓“激活”就是要发出一些信令，保证要传送的计算机数据能在这条通路上正确发送和接收。

（2）要告诉网络如何识别接收数据的计算机。

（3）发起通信的计算机必须查明对方计算机是否已准备好接收数据。

（4）发起通信的计算机必须弄清楚，在对方计算机中的文件管理程序是否已做好文件接收和存储文件的准备工作。

（5）若计算机的文件格式不兼容，则至少其中的一个计算机应完成格式转换功能。

（6）对出现的各种差错和意外事故，如数据传送错误、重复或丢失，网络中某个结点交

换机出故障等，应当有可靠的措施保证对方计算机最终能够收到正确的文件。

还可以举出一些要做的其他工作。由此可见。相互通信的两个计算机系统必须高度协调工作才行，而这种“协调”是相当复杂的。为了设计这样复杂的计算机网络，早在最初的ARPANET 设计时即提出了分层的方法。**“分层”**可将庞大而复杂的问题，转化为若干较小的局部问题，而这些较小的局部问题就比较易于研究和处理。

1974 年，美国的 IBM 公司宣布了它研制的**系统网络体系结构** SNA (System Network Architecture)。这个著名的网络标准就是按照分层的方法制订的。以后 SNA 又不断得到改进，更新了几个版本。现在它是世界上使用得较为广泛的一种网络体系结构。不久后，其他一些公司也相继推出本公司的一套体系结构，并都采用不同的名称。

网络体系结构出现后，使得一个公司所生产的各种设备都能够很容易地互连成网。这种情况显然有利于一个公司垄断自己的产品。用户一旦购买了某个公司的网络，当需要扩大容量时，就只能再购买该公司的产品。如果同时又再购买了其他公司的产品，那么由于网络体系结构的不同，就很难互相连通。

然而全球经济的发展使得不同网络体系结构的用户迫切要求能够互相交换信息。为了使不同体系结构的计算机网络都能互连，国际标准化组织 ISO 于 1977 年成立了专门机构研究这个问题。不久，他们就提出一个试图使各种计算机在世界范围内互连成网的标准框架，即著名的**开放系统互连基本参考模型** OSI/RM (Open Systems Interconnection Reference Model)，简称为 OSI。“开放”是指：只要遵循 OSI 标准，一个系统就可以和位于世界上任何地方的、也遵循这同一标准的其他任何系统进行通信。这一点很像世界范围的电话和邮政系统，这两个系统都是开放系统。“系统”是指在现实的系统中与互连有关的各部分。所以开放系统互连参考模型 OSI/RM 是个抽象的概念。在 1983 年形成了开放系统互连基本参考模型的正式文件，即著名的 ISO 7498 国际标准，也就是所谓的七层协议的体系结构。

OSI 试图达到一种理想境界，即全世界的计算机网络都遵循这统一的标准，因而全世界的计算机都将能够很方便地进行互连和交换数据。在 20 世纪 80 年代，许多大公司甚至一些国家的政府机构都纷纷表示支持 OSI。当时看来似乎在不久的将来全世界一定会都按照 OSI 制订的标准来构造自己的计算机网络。然而到了 20 世纪 90 年代初期，虽然整套的 OSI 国际标准都已经制订出来了，但由于因特网已抢先在全世界覆盖了相当大的范围，而与此同时却几乎找不到有什么厂家生产出符合 OSI 标准的商用产品。因此人们得出这样的结论：OSI 只获得了一些理论研究的成果，但在市场化方面 OSI 则事与愿违地失败了。现今规模最大的、覆盖全世界的计算机网络因特网并未使用 OSI 标准。OSI 失败的原因可归纳为：

（1）OSI 的专家们缺乏实际经验，他们在完成 OSI 标准时没有商业驱动力；

（2）OSI 的协议实现起来过分复杂，而且运行效率很低；

（3）OSI 标准的制定周期太长，因而使得按 OSI 标准生产的设备无法及时进入市场；

（4）OSI 的层次划分也不太合理，有些功能在多个层次中重复出现。

按照一般的概念，网络技术和设备只有符合有关的国际标准才能在大范围获得工程上的应用。但现在情况却反过来了。得到最广泛应用的不是**法律上的国际标准** OSI，而是非国际标准 TCP/IP。这样，TCP/IP 就常被称为是**事实上的国际标准**。从这种意义上说，能够占领市场的就是标准。在过去制订标准的组织中往往以专家、学者为主。但现在许多公司都纷纷挤进各种各样的标准化组织，使得技术标准具有浓厚的商业气息。一个新标准的出现，有时

不一定反映出其技术水平是最先进的，而是往往有着一定的市场背景。

顺便说一下，虽然 OSI 在一开始是由 ISO 来制订，但后来的许多标准都是 ISO 与原来的国际电报电话咨询委员会 CCITT[①]联合制订的。从历史上来看，CCITT 原来是从通信的角度考虑一些标准的制定，而 ISO 则关心信息的处理。但随着科学技术的发展，通信与信息处理的界限变得比较模糊了。于是，通信与信息处理就都成为 CCITT 与 ISO 所共同关心的领域。CCITT 的建议书 X.200 就是关于开放系统互连参考模型，它和上面提到的 ISO 7498 基本上是相同的。

1.5.2 划分层次的必要性

在计算机网络中要做到有条不紊地交换数据，就必须遵守一些事先约定好的规则。**这些规则明确规定了所交换的数据的格式以及有关的同步问题**。这里所说的**同步**不是狭义的（即同频或同频同相）而是广义的，即在一定的条件下应当发生什么事件（如发送一个应答信息），因而**同步含有时序的意思**。这些为**进行网络中的数据交换而建立的规则、标准或约定**即称为**网络协议**(network protocol)。网络协议也可简称为**协议**。更进一步讲，网络协议主要由以下三个要素组成：

（1）**语法**，即数据与控制信息的结构或格式；

（2）**语义**，即需要发出何种控制信息，完成何种动作以及做出何种响应；

（3）**同步**，即事件实现顺序的详细说明。

由此可见，网络协议是计算机网络的不可缺少的组成部分。实际上，只要我们想让连接在网络上的另一台计算机做点什么事情（例如，从网络上的某个主机下载文件），我们都需要有协议。但是当我们经常在自己的 PC 机上进行文件存盘操作时，**就不需要任何协议**，除非这个用来存储文件的磁盘是网络上的某个文件服务器的磁盘。

协议通常有两种不同的形式。一种是使用便于人来阅读和理解的文字描述。另一种是使用让计算机能够理解的程序代码。这两种不同形式的协议都必须能够对网络上交换的信息做出精确的解释。

ARPANET 的研制经验表明，对于非常复杂的计算机网络协议，其结构应采用层次式的。我们可以举一个简单的例子来说明划分层次的概念。

现在假定我们在计算机 1 和计算机 2 之间通过一个通信网络传送文件。这是一件比较复杂的工作，因为还需要做不少的工作。

我们可以将要做的工作划分为三类。第一类工作与传送文件直接有关。例如，发送方的文件传送应用程序应当确信接收方的文件管理程序已做好接收和存储文件的准备。若两个计算机所用的文件格式不一样，则至少其中的一个计算机应完成文件格式的转换。这两件工作可用一个文件传送模块来完成。这样，两个计算机可将文件传送模块作为最高的一层（图 1-15）。在这两个模块之间的虚线表示两个计算机系统交换文件和一些有关文件交换的命令。

---

① 鉴于“有线电”和“无线电”的关系日益密切，国际电信联盟 ITU (International Telecommunication Union)已将国际电报电话咨询委员会 CCITT 和国际无线电咨询委员会 CCIR 合并为电信标准化部门 TSS (Telecommunication Standardization Sector)。从 1993 年 3 月 1 日起，CCITT 和 CCIR 就不复存在。今后有关电信的标准就由国际电联(ITU)的电信标准化部门颁布，并在每个建议书的前面加上 ITU-T 这几个字，例如，原来的 CCITT X.25 现在就称为 ITU-T X.25。为了节约经费，以后不再是每隔四年就出版全套的建议书，而是只出版新通过的建议书或旧建议书中有变化的部分。CCITT 虽然不存在了，但过去 CCITT 所制定的标准并未作废，凡未过时的标准我们在需要时都可继续引用。

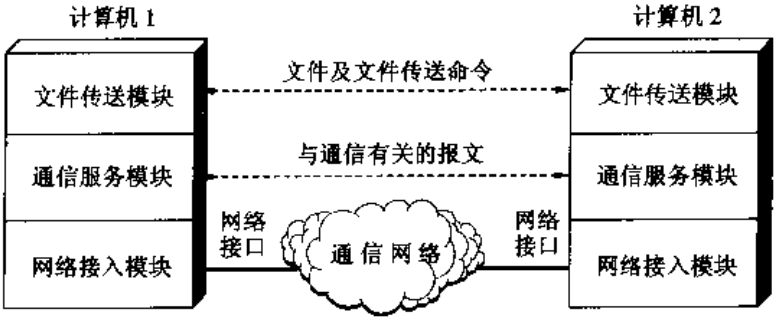


图 1-15　划分层次的举例

但是，我们并不想让文件传送模块完成全部工作的细节，这样会使文件传送模块过于复杂。可以再设立一个通信服务模块，用来保证文件和文件传送命令可靠地在两个系统之间交换。也就是说，让位于上面的文件传送模块利用下面的通信服务模块所提供的服务。我们还可以看出，如果将位于上面的文件传送模块换成电子邮件模块，那么电子邮件模块同样可以利用在它下面的通信服务模块所提供的可靠通信的服务。

同样道理，我们再构造一个网络接入模块，让这个模块负责做与网络接口细节有关的工作，并向上层提供服务，使上面的通信服务模块能够完成可靠通信的任务。

从上述简单例子可以更好地理解分层可以带来很多好处。如：

（1）**各层之间是独立的**。某一层并不需要知道它的下一层是如何实现的，而仅仅需要知道该层通过层间的接口（即界面）所提供的服务。由于每一层只实现一种相对独立的功能，因而可将一个难以处理的复杂问题分解为若干个较容易处理的更小一些的问题。这样，整个问题的复杂程度就下降了。

（2）**灵活性好**。当任何一层发生变化时（例如由于技术的变化），只要层间接口关系保持不变，则在这层以上或以下各层均不受影响。此外，对某一层提供的服务还可进行修改。当某层提供的服务不再需要时，甚至可以将这层取消。

（3）**结构上可分割开**。各层都可以采用最合适的技术来实现。

（4）**易于实现和维护**。这种结构使得实现和调试一个庞大而又复杂的系统变得易于处理，因为整个的系统已被分解为若干个相对独立的子系统。

（5）**能促进标准化工作**。因为每一层的功能及其所提供的服务都已有了精确的说明。

分层时应注意使每一层的功能非常明确。若层数太少，就会使每一层的协议太复杂。但层数太多又会在描述和综合各层功能的系统工程任务时遇到较多的困难。通常每一层所要实现的一般功能往往是下面的一种功能或多种功能：

① **差错控制**　使得和网络对端的相应层次的通信更加可靠。

② **流量控制**　使得发送端的发送速率不要太快，要使接收端来得及接收。

③ **分段和重装**　发送端将要发送的数据块划分为更小的单位，在接收端将其还原。

④ **复用和分用**　发送端几个高层会话复用一条低层的连接，在接收端再进行分用。

⑤ **连接建立和释放**　交换数据前先建立一条逻辑连接。数据传送结束后释放连接。

分层当然也有一些缺点，例如，有些功能会在不同的层次中重复出现，因而产生了额外开销。

我们将**计算机网络的各层及其协议的集合**，称为网络的**体系结构**(architecture)。换种说法，**计算机网络的体系结构就是这个计算机网络及其部件所应完成的功能的精确定义**。需要强调

的是：这些功能究竟是用何种硬件或软件完成的，则是一个遵循这种体系结构的**实现**(implementation)的问题。体系结构的英文名词 architecture 的原意是建筑学或建筑的设计和风格。它和一个具体的建筑物的概念很不相同。例如，我们可以走进一个明代的建筑物中，但却不能走进一个明代的建筑风格之中。同理，我们也不能把一个具体的计算机网络说成是一个抽象的网络体系结构。总之，**体系结构是抽象的，而实现则是具体的，是真正在运行的计算机硬件和软件。**

1.5.3 具有五层协议的体系结构

OSI 的七层协议体系结构既复杂又不实用，但其概念清楚，体系结构理论较完整。TCP/IP 的协议现在得到了广泛的应用，但它原先并没有一个明确的体系结构。TCP/IP 是一个四层的体系结构，它包含应用层、运输层、网际层和网络接口层。不过从实质上讲，TCP/IP 只有三层，即应用层、运输层和网际层，因为最下面的网络接口层并没有什么具体内容。因此在学习计算机网络的原理时往往采取折中的办法，即综合 OSI 和 TCP/IP 的优点，采用一种只有五层协议的体系结构（图 1-16），这样既简洁又能将概念阐述清楚[TANE96]。

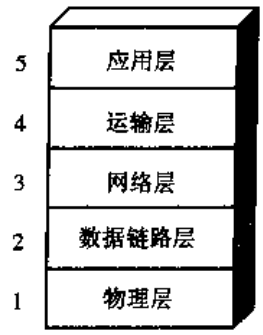


图 1-16 五层协议的体系结构

现在结合因特网的情况，自上而下地简单介绍一下各层的主要功能。实际上，只有认真学习完本书各章的协议后才能真正弄清各层的作用。

（1）**应用层**(application layer) 应用层是体系结构中的最高层。应用层确定进程之间通信的性质以满足用户的需要（这反映在用户所产生的服务请求）。这里的**进程**就是指正在运行的**程序**。应用层不仅要提供应用进程所需要的信息交换和远地操作，而且还要作为互相作用的应用进程的**用户代理**(user agent)，来完成一些为进行语义上有意义的信息交换所必须的功能。应用层直接为用户的应用进程提供服务。在因特网中的应用层协议很多，如支持万维网应用的 HTTP 协议，支持电子邮件的 SMTP 协议，支持文件传送的 FTP 协议等等。

（2）**运输层**(transport layer) 运输层的任务就是负责主机中**两个进程之间的通信**。

因特网的运输层可使用两种不同协议。即面向连接的**传输控制协议** TCP (Transmission Control Protocol)，和无连接的用**户数据报协议** UDP (User Datagram Protocol)。运输层的数据传输的单位是**报文段**(segment)（当使用 TCP 时）或**用户数据报**（当使用 UDP 时）。面向连接的服务能够提供可靠的交付，但无连接服务则不保证提供可靠的交付，它只是“**尽最大努力交付**(best-effort delivery)”。这两种服务方式都很有用，各有其优缺点。

在分组交换网内的各个交换结点机都没有运输层。运输层只能存在于分组交换网外面的主机之中。运输层以上的各层就不再关心信息传输的问题了。正因为如此，运输层就成为计算机网络体系结构中非常重要的一层。

（3）**网络层**(network layer) 网络层负责为分组交换网上的不同**主机**提供通信。在发送数据时，网络层将运输层产生的报文段或用户数据报封装成分组或包进行传送。在 TCP/IP 体系中，分组也叫作 **IP 数据报**，或简称为**数据报**。**本书以后将“分组”和“数据报”作为同义词使用**。但请读者注意：不要将运输层的“用户数据报”和网络层的“IP 数据报”弄混。网络层的另一个任务就是要选择合适的路由，使源主机运输层所传下来的分组能够交付到目

的主机。

这里要强调指出，网络层中的“网络”二字，已不是我们通常谈到的具体的网络，而是在计算机网络体系结构模型中的专用名词。

对于由广播信道构成的分组交换网，路由选择的问题很简单，因此这种网络的网络层非常简单，甚至可以没有。

因特网是一个很大的互联网，它由大量的**异构**(heterogeneous)网络通过**路由器**(router)相互连接起来。因特网主要的网络层协议是无连接的**网际协议** IP (Internet Protocol)和许多种路由选择协议，因此因特网的网络层也叫做**网际层**或 **IP 层**。在本书中，网络层、网际层和 IP 层都是同义语。

（4）**数据链路层**(data link layer)　在发送数据时，数据链路层的任务是将在网络层交下来的 IP 数据报组装成**帧**(framing)，在两个相邻结点间的链路上传送以**帧**(frame)为单位的数据。每一帧包括数据和必要的控制信息（如同步信息、地址信息、差错控制，以及流量控制信息等）。控制信息使接收端能够知道一个帧从哪个比特开始和到哪个比特结束。控制信息还使接收端能够检测到所收到的帧中有无差错。如发现有差错，数据链路层就丢弃这个出了差错的帧，然后采取下面两种方法之一：或者不作任何其他的处理；或者由数据链路层通知对方重传这一帧，直到正确无误地收到此帧为止。数据链路层有时也常简称为**链路层**。

（5）**物理层**(physical layer)　物理层的任务就是**透明地传送比特流**。在物理层上所传**数据的单位是比特**。传递信息所利用的一些物理媒体，如双绞线、同轴电缆、光缆等，并不在物理层之内而是在物理层的下面。因此也有人把物理媒体当做第 0 层。

“**透明**”是一个很重要的术语。它表示：**某一个实际存在的事物看起来却好像不存在一样**。“透明地传送比特流”表示经实际电路传送后的比特流没有发生变化，因此，对传送比特流来说，由于这个电路并没有对其产生什么影响，因而比特流就“看不见”这个电路。或者说，这个电路对该比特流来说是透明的。这样，任意组合的比特流都可以在这个电路上传送。当然，哪几个比特代表什么意思，则不是物理层所要管的。

物理层要考虑用多大的电压代表“1”或“0”，以及当发送端发出比特“1”时，在接收端如何识别出这是比特“1”而不是比特“0”。物理层还要确定连接电缆的插头应当有多少根腿以及各个腿应如何连接。

在因特网所使用的各种协议中，最重要的和最著名的就是 TCP 和 IP 两个协议。现在人们经常提到的 TCP/IP 并不一定是指 TCP 和 IP 这两个具体的协议，而往往是表示因特网所使用的体系结构或是指整个的 **TCP/IP 协议族**(protocol suite)[①]。

图 1-17 说明的是应用进程的数据在各层之间的传递过程中所经历的变化。这里为简单起见，假定两个主机是直接相连的。

假定计算机 1 的应用进程 AP_1 向计算机 2 的应用进程 AP_2 传送数据。AP_1 先将其数据交给第 5 层（应用层）。第 5 层加上必要的控制信息 H_5 就变成了下一层的数据单元。第 4 层（运输层）收到这个数据单元后，加上本层的控制信息 H_4，再交给第 3 层（网络层），成为第 3 层的数据单元。依次类推。不过到了第 2 层（数据链路层）后，控制信息分成两部分，分别

① 注：请注意 suite 这个字的特殊读音/swi:t/，不要读错。

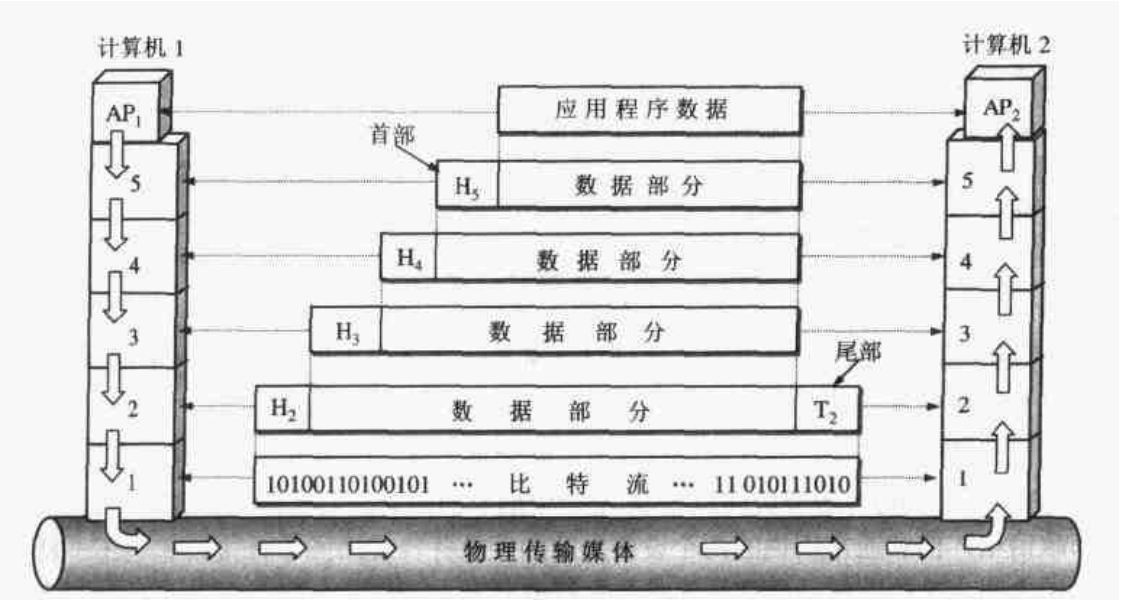


图 1-17　数据在各层之间的传递过程

加到本层数据单元的首部(H_2)和尾部(T_2)，而第 1 层（物理层）由于是比特流的传送，所以不再加上控制信息。

在 OSI 参考模型中，在对等层次上传送的数据，其单位都称为该层的**协议数据单元** PDU (Protocol Data Unit)。这个名词现已被许多非 OSI 标准采用。

当这一串的比特流经网络的物理媒体传送到目的站时，就从第 1 层依次上升到第 5 层。每一层根据控制信息进行必要的操作，然后将控制信息剥去，将该层剩下的数据单元上交给更高的一层。最后，把应用进程 AP_1 发送的数据交给目的站的应用进程 AP_2。

可以用一个简单的例子来比喻上述过程。有一封信从最高层向下传。每经过一层就包上一个新的信封。包有多个信封的信传送到目的站后，从第 1 层起，每层拆开一个信封后就交给它的上一层。传到最高层后，取出发信人所发的信交给收信用户。

虽然应用进程数据要经过如图 1-17 所示的复杂过程才能送到对方的应用进程，但这些复杂过程对用户来说，却都被屏蔽掉了，以致应用进程 AP_1 觉得好像是直接把数据交给了应用进程 AP_2。同理，任何两个同样的层次（例如在两个系统的第 4 层）之间，也好像如同图中的水平虚线所示的那样，将数据（即数据单元加上控制信息）通过水平虚线直接传递给对方。这就是所谓的“**对等层**”(peer layers)之间的通信。我们以前经常提到的各层协议，实际上就是在各个对等层之间传递数据时的各项规定。

在文献中也还可以见到术语“**协议栈**”(protocol stack)。这是因为几个层次画在一起很像一个**栈**(stack)的结构。

1.5.4　实体、协议、服务和服务访问点

当研究在开放系统中进行交换信息时，发送或接收信息的究竟是一个进程、是一个文件还是一个终端，都没有实质上的影响。为此，可以用**实体**(entity)这一较为抽象的名词表示**任何可发送或接收信息的硬件或软件进程**。在许多情况下，实体就是一个特定的软件模块。

协议是控制两个对等实体进行通信的规则的集合。协议的语法方面的规则定义了所交换的信息的格式，而协议的语义方面的规则就定义了发送者或接收者所要完成的操作，例如，在何种条件下数据必须重发或丢弃。

在协议的控制下，两个对等实体间的通信使得本层能够向上一层提供服务。要实现本层协议，还需要使用下面一层所提供的服务。

一定要弄清楚，协议和服务在概念上是很不一样的。

首先，协议的实现保证了能够向上一层提供服务。**本层的服务用户只能看见服务而无法看见下面的协议。下面的协议对上面的服务用户是透明的。**

其次，**协议是“水平的”**，即协议是控制对等实体之间通信的规则。但**服务是“垂直的”**，即服务是由下层向上层通过层间接口提供的。另外，并非在一个层内完成的全部功能都称为服务。只有那些能够被高一层看得见的功能才能称之为“服务”。上层使用下层所提供的服务必须通过与下层交换一些命令，这些命令在 OSI 中称为**服务原语**。

在同一系统中相邻两层的实体进行交互（即交换信息）的地方，通常称为**服务访问点** SAP (Service Access Point)。服务访问点 SAP 是一个抽象的概念，它实际上就是一个逻辑接口，有些像邮政信箱，但和通常所说的两个设备之间的硬件并行接口或串行接口是很不一样的。OSI 将层与层之间交换的数据的单位称为**服务数据单元** SDU (Service Data Unit)，它可以与 PDU 不一样。例如，可以是多个 SDU 合成为一个 PDU，也可以是一个 SDU 划分为几个 PDU。

这样，在任何相邻两层之间的关系可概括为图 1-18 所示的那样。这里要注意的是，某一层向上一层所提供的服务实际上已包括了在它以下各层所提供的服务。所有这些对上一层来说就相当于一个服务提供者。在服务提供者的上一层的实体，也就是**“服务用户”**，它使用服务提供者所提供的服务。图中两个对等实体（服务用户）通过协议进行通信，为的是可以向上提供服务。

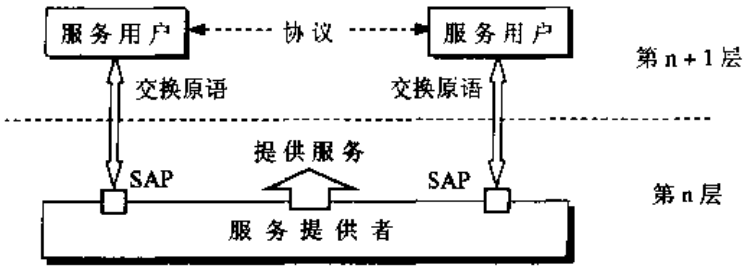


图 1-18　相邻两层之间的关系

计算机网络的协议还有一个很重要的特点，就是协议必须将所有不利的条件事先都估计到，而**不能假定将在很顺利的条件下进行通信**。例如，两个朋友在电话中约会好，下午 3 时在某公园门口碰头，并且约定“不见不散”。这就是一个很坏的协议，因为任何一方临时有急事来不了而又无法通知对方时（如对方的电话或手机都无法接通），则按照协议另一方就必须永远等待下去。因此，看一个计算机网络协议是否正确，不能光看在正常情况下协议是否正确，而且还必须**非常仔细地检查这个协议能否应付绝大部分不利情况**。

下面是一个有关网络协议的非常著名的例子。

占据东边和西边两个山顶的蓝军与驻扎在这两个山之间的山谷的白军作战。其力量对比是：一个山顶上的蓝军打不过白军，但两个山顶的蓝军协同作战则可战胜白军。东边蓝军拟于次日正午向白军发起攻击。于是用计算机发送电文给西边的友军。但通信线路很不好，电文出错或丢失的可能性较大（没有电话可使用）。因此要求收到电文的友军必须送回一个确认电文。但此确认电文也可能出错或丢失。试问能否设计出一种协议使得两个山顶的蓝军能够实现协同作战因而一定（即 100 %而不是 99.999...%）取得胜利？

东边蓝军先发送："拟于明日正午向白军发起攻击。请协同作战，并确认。"

西边蓝军收到电文后加以确认，回答："同意。"

然而现在两边的蓝军都不敢冒然下决心进攻。因为，西边蓝军不知道此确认电文对方是否正确地收到了。如未正确收到，东边蓝军必定不敢冒然进攻。在此情况下，如果自己发起进攻就肯定要失败。因此，必须等待东边蓝军发送"对确认的确认"。

假定西边蓝军收到了东边蓝军发来的确认。但东边蓝军同样关心自己发出的确认是否已被对方正确地收到。因此还要等待西边蓝军的"对确认的确认的确认"。

这样无限循环下去，两边的蓝军都始终无法确定自己最后发出的电文对方是否已经收到（图 1-19）。因此，没有一种协议能够使两边的蓝军能够 100%地确定双方将于次日正午发起的协同进攻。

从这个例子可以看出，不可能设计出100%可靠的协议。

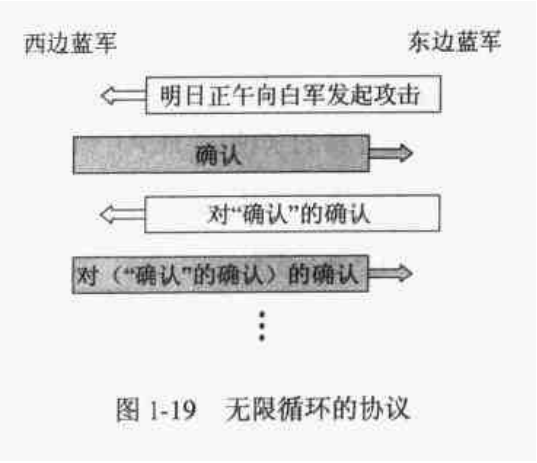


图 1-19　无限循环的协议

1.5.5　面向连接服务与无连接服务

从通信的角度看，各层所提供的服务可分为两大类，即**面向连接的**(connection-oriented)与**无连接的**(connectionless)。现分别介绍如下。

1．面向连接服务

所谓**连接**，就是两个对等实体为进行数据通信而进行的一种结合。

面向连接服务具有连接建立、数据传输和连接释放这三个阶段。面向连接服务是在数据交换之前，必须先建立连接。当数据交换结束后，则必须终止这个连接。在传送数据时是按序传送的。面向连接服务比较适合于在一定期间内要向同一目的地发送许多报文的情况。对于发送很短的零星报文，面向连接服务的开销就显得过大了。

2．无连接服务

在无连接服务的情况下，两个实体之间的通信不需要先建立好一个连接，因此其下层的有关资源不需要事先进行预定保留。这些资源将在数据传输时动态地进行分配。

无连接服务的另一特征就是它不需要通信的两个实体同时是活跃的(active)。当发送端的实体正在进行发送时，它才必须是活跃的。这时接收端的实体并不一定必须是活跃的。只有当接收端的实体正在进行接收时，它才必须是活跃的。

无连接服务的优点是灵活方便和比较迅速。但无连接服务不能防止报文的丢失、重复或失序。

无连接服务的特点不需要接收端做任何响应，因而是一种不可靠的服务。这种服务常被描述为"**尽最大努力交付**"(best effort delivery)或"尽力而为"。

1.5.6　OSI 与 TCP/IP 体系结构的比较

OSI 参考模型中采用了七个层次的体系结构，也就是将前面所讲的原理体系结构中的应用层再划分为三个层次。这三个层次从上到下的名称是：**应用层、表示层和会话层**。

由于 OSI 参考模型过于复杂，现在流行的因特网体系结构中已经不使用 OSI 的表示层和会话层了。愿意了解 OSI 参考模型历史资料的读者可参考[TANA96]。

图 1-20 画出了 TCP/IP 与 OSI 这两种体系结构的对比。图 1-20(a)是已成为历史的 OSI 体系结构。图 1-20(b)是目前因特网使用的 TCP/IP 体系结构（但也有些人将下面的网络接口层划分为两层，即网络接口层和物理层，因而成为五层的体系结构）。图 1-20(c)概括了 TCP/IP 的三个服务层次，即应用层向应用进程提供应用服务，运输层在使用 TCP 协议时向应用层提供面向连接的可靠的（使用 TCP）或不可靠的（使用 UDP）运输服务，而网际层使用 IP 向运输层提供无连接分组交付服务（即尽最大努力交付）。

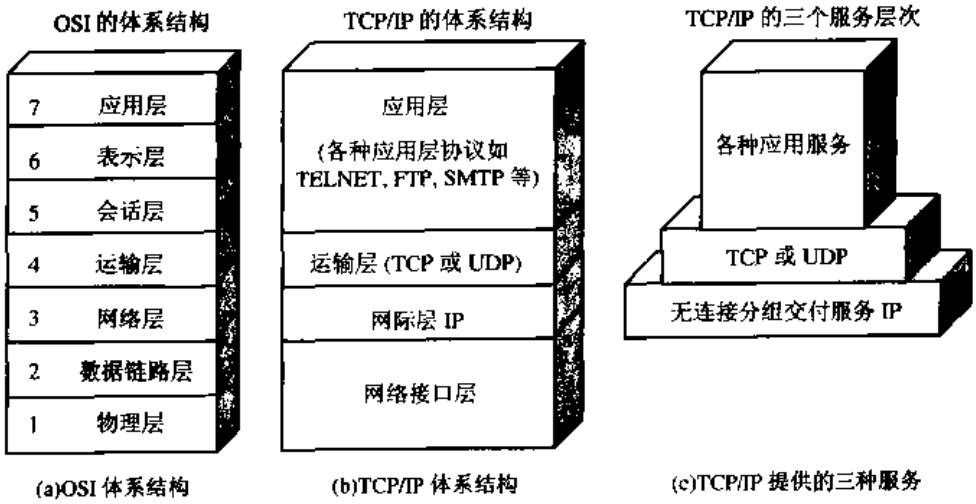


图 1-20 TCP/IP 与 OSI 体系结构的对比

值得注意的是，在以下的一些问题的处理上，TCP/IP 与 OSI 是很不相同的。例如：

（1）TCP/IP 一开始就考虑到多种异构网的互连问题，并将网际协议 IP 作为 TCP/IP 的重要组成部分。但 ISO 和 CCITT 最初只考虑到全世界都使用一种统一的标准公用数据网将各种不同的系统互连在一起。后来，ISO 认识到了网际协议 IP 的重要性，然而已经来不及了，只好在网络层中划分出一个子层来完成类似 TCP/IP 中 IP 的作用。

（2）TCP/IP 一开始就对面向连接服务和无连接服务并重，而 OSI 在开始时只强调面向连接这一种服务。一直到很晚 OSI 才开始制定另一种无连接服务的有关标准。

（3）TCP/IP 较早就有较好的网络管理功能，而 OSI 到后来才开始考虑这个问题。

当然，TCP/IP 也有不足之处。例如，TCP/IP 的模型对“服务”、“协议”和“接口”等概念并没有很清楚地区分开。因此在使用一些新的技术来设计新的网络时，采用这种模型就可能会遇到一些麻烦。另外，TCP/IP 模型的通用性较差，很难用它来描述其他种类的协议栈。还有，TCP/IP 的网络接口层严格来说并不是一个层次而仅仅是一个接口。

在许多文献中，也常见到 TCP/IP 的四层协议的表示方法。例如，在讨论两个主机通过两个网络用路由器互连在一起时，可以使用如图 1-21 所示的层次关系。图中的逻辑链路控制和物理层都简化为网络接口层。实际上，现在插在主机中的网络接口板上的硬件和软件就实现了数据链路层和物理层这两层的功能。应当注意的是，在网络互连中起重要作用的路由器则没有应用层和运输层，其数据链路层和物理层也是在网络接口板上实现的。

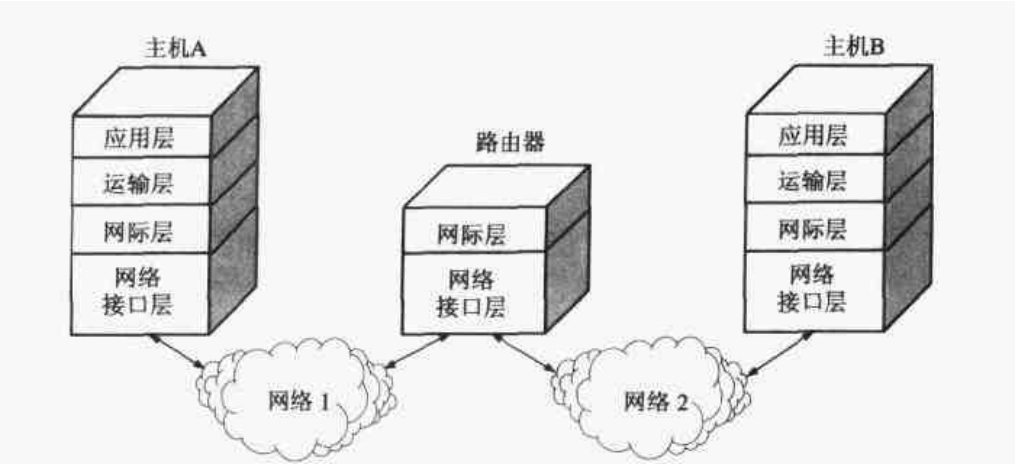


图 1-21　TCP/IP 四层协议的表示方法举例

还有一种方法，就是分层次画出具体的协议来表示 TCP/IP 协议族（图 1-22），它的特点是上下两头大而中间小：应用层和网络接口层都有多种协议，而中间的 IP 层很小，上层的各种协议都向下汇聚到一个 IP 协议中。这种很像沙漏计时器形状的 TCP/IP 协议族表明：TCP/IP **可以为各式各样的应用提供服务**（所谓的 everything over IP），同时也**可以连接到各式各样的网络上**（所谓的 IP over everything）。正因为如此，因特网才会发展到今天的这种全球规模。从图 1-22 不难看出 IP 协议在因特网中的核心作用。

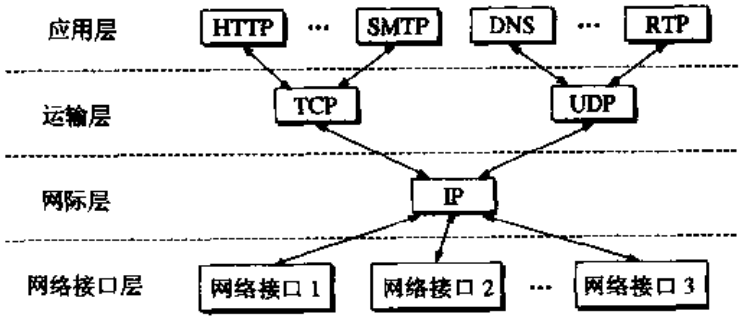


图 1-22　沙漏计时器形状的 TCP/IP 协议族

1.6　应用层的客户-服务器方式

在 TCP/IP 的应用层协议使用的是**客户-服务器方式**。由于这个概念特别重要，因此在这里我们先进行简单的介绍。

前面已经讲过，应用层直接为用户的应用进程提供服务。**计算机的进程**(process)就是运行着的计算机程序。在网络环境下，许多问题的解决往往是通过位于不同主机中的多个进程之间的通过网络的通信和协同工作来完成的。**这些为了解决具体的应用问题而彼此通信的进程就称为“应用进程”**。而应用层的具体内容就是**规定应用进程在通信时所遵循的协议**。

当进行电话通信时，电话机的振铃声使被叫用户知道现在有一个电话呼叫。计算机通信的对象是应用层中的应用进程，这显然不能用响铃的办法来通知所要找的应用进程。为此，TCP/IP 体系采用客户-服务器方式使两个应用进程能够进行通信。

客户(client)和**服务器**(server)都是指通信中所涉及的两个应用进程。客户-服务器方式所描述的是进程之间服务和被服务的关系。当 A 进程需要 B 进程的服务时就主动呼叫 B 进程，

在这种情况下，A 是客户而 B 是服务器。可能在下一次通信中，B 需要 A 的服务，此时，B 就是客户而 A 就是服务器。这里最主要的特征就是：**客户是服务请求方，服务器是服务提供方**。在实际应用中，客户软件和服务器软件通常还具有以下一些主要特点。

客户软件：

（1）在进行通信时临时成为客户，但它也可在本地进行其他的计算。

（2）被用户调用并在用户的计算机上运行，在打算通信时主动向远地服务器发起通信。

（3）可与多个服务器进行通信。

（4）不需要特殊的硬件和很复杂的操作系统。

服务器软件：

（1）是一种专门用来提供某种服务的程序，可同时处理多个远地或本地客户的请求。

（2）在共享计算机上运行。当系统启动时即自动调用并一直不断地运行着。

（3）被动地等待并接受来自多个客户的通信请求。

（4）一般需要强大的硬件和高级的操作系统支持。

客户与服务器的通信关系一旦建立，通信就可是双向的，客户和服务器都可发送和接收信息。大多数的应用进程都是使用 TCP/IP 协议进行通信。图 1-23 画出了这种情况。

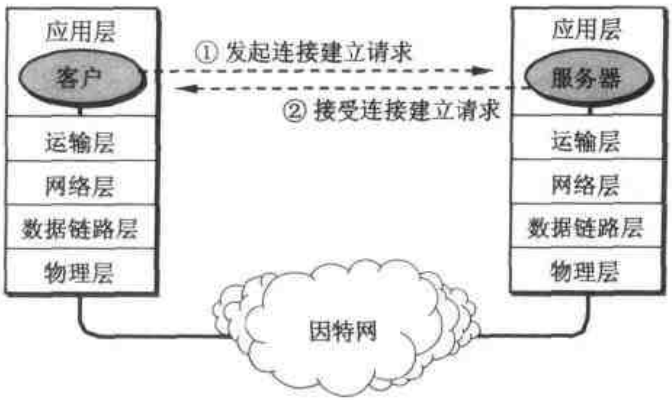


图 1-23　客户进程和服务器进程使用 TCP/IP 协议进行通信

图 1-23 强调了客户-服务器建立通信关系的两个主要步骤，即客户首先发起连接建立请求，而服务器接受连接建立请求。以后就逐级使用下一层提供的服务。如应用进程使用下面的 TCP 连接，而 TCP 又使用下面的 IP 数据报，等等。因此整个协议栈都要用到。

功能较强的计算机可同时运行多个服务器进程（如图 1-24 中的计算机 3）。这时，计算机 1 和计算机 2 中的客户进程分别和计算机 3 中的两个服务器进程进行通信。应注意：计算机 3 到因特网的物理连接只有一条（即多个 TCP 连接复用到一条物理链路上）。

顺便要说一下，使用计算机的人是“用户”(user)而不是“客户”(client)。这里的**客户和服务器都指的是进程，即计算机软件**。过去曾将 client 译为“客户机”，但这不是标准译名，而且使用“客户机”这种译名容易使人误认为 client 是硬件。需要指出的是，由于运行服务器进程的机器往往有许多特殊的要求（不同于普通的 PC 机），因此人们经常将主要运行服务器进程的机器（硬件）**不严格地**称为服务器。例如，说：“这台机器是服务器。”其实这句话的意思是：“这台机器（硬件）主要是用来运行服务器进程（软件）。”于是，服务器一词有时指的是软件，但也有时指的是硬件。在本书中，在涉及到运行客户（或服务器）进程的机器时，我们就常说“客户端”或“服务器端”。

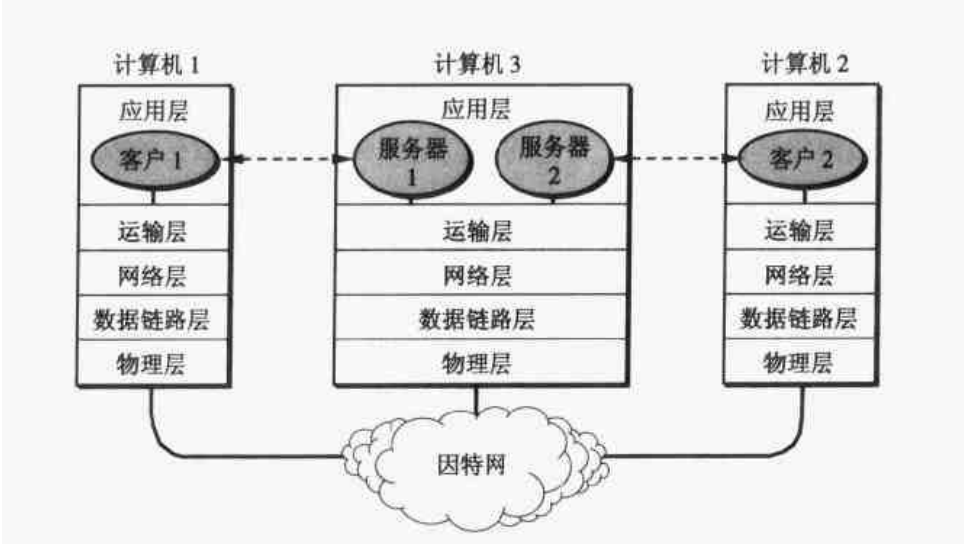


图 1-24　一台计算机中的多个服务器可被多个计算机的客户访问

在讲述了计算机网络体系结构的基本概念后，从下一章起，就逐层讨论更加具体的内容。

习题

1-01　计算机网络的发展可划分为几个阶段？每个阶段各有何特点？

1-02　试简述分组交换的要点。

1-03　试从多个方面比较电路交换、报文交换和分组交换的主要优缺点。

1-04　为什么说因特网是自印刷术以来人类通信方面最大的变革？

1-05　试讨论在广播式网络中对网络层的处理方法。讨论是否需要这一层？

1-06　试将 TCP/IP 和 OSI 的体系结构进行比较。讨论其异同之处。

1-07　计算机网络可从哪几个方面进行分类？

1-08　计算机网络中的主干网和本地接入网各有何特点？

1-09　计算机网络由哪几部分组成？

1-10　试在下列条件下比较电路交换和分组交换。要传送的报文共 x(bit)。从源站到目的站共经过 k 段链路，每段链路的传播时延为 d(s)，数据率为 b(b/s)。在电路交换时电路的建立时间为 s(s)。在分组交换时分组长度为 p(bit)，且各结点的排队等待时间可忽略不计。问在怎样的条件下，分组交换的时延比电路交换的要小？

1-11　在上题的分组交换网中，设报文长度和分组长度分别为 x 和 $(p + h)$(bit)，其中 p 为分组的数据部分的长度，而 h 为每个分组所带的控制信息固定长度，与 p 的大小无关。通信的两端共经过 k 段链路。链路的数据率为 b(b/s)，但传播时延和结点的排队时间均可忽略不计。若打算使总的时延为最小，问分组的数据部分长度 p 应取为多大？

1-12　网络体系结构为什么要采用分层次的结构？试举出一些与分层体系结构的思想相似的日常生活。

1-13　面向连接服务与无连接服务各自的特点是什么？

1-14　协议与服务有何区别？有何关系？

1-15　网络协议的三个要素是什么？各有什么含义？

1-16　试述具有五层协议的网络体系结构的要点，包括各层的主要功能。

1-17　试举出日常生活中有关“透明”这种名词的例子。

1-18 解释以下名词：协议栈、实体、对等层、协议数据单元、服务访问点、客户、服务器、客户-服务器方式。

1-19 什么是计算机网络链路的带宽？带宽的单位是什么？什么是数据的发送时延、传播时延、排队时延和往返时延 RTT？

1-20 收发两端之间的传输距离为 1000 km，信号在媒体上的传播速率为 2×10^8 m/s。试计算以下两种情况的发送时延和传播时延：

（1）数据长度为 10^7 bit，数据发送速率为 100 kb/s

（2）数据长度为 10^3 bit，数据发送速率为 1 Gb/s。

从以上计算结果可得出什么结论？

1-21 假设信号在媒体上的传播速率为 2×10^8 m/s。媒体长度 l 分别为：

（1）10 cm（网卡）

（2）100 m（局域网）

（3）100 km（城域网）

（4）5000 km（广域网）

试计算当数据率为 1 Mb/s 和 10 Gb/s 时在以上媒体中正在传播的比特数。

1-22 长度为 100 字节的应用层数据交给运输层传送，需加上 20 字节的 TCP 首部。再交给网络层传送，需加上 20 字节的 IP 首部。最后交给数据链路层的以太网传送，加上首部和尾部共 18 字节。试求数据的传输效率。

若应用层数据长度为 1000 字节，数据的传输效率是多少？

第2章 物 理 层

本章首先讨论物理层的基本概念。然后介绍有关信道极限容量的重要概念，我们将给出与数据传输速率有关的两个著名公式，但不进行证明。接着讨论各种传输媒体的主要特点（当然传输媒体本身并不在物理层的范围之内），以及模拟传输和数字传输的一些常用技术。最后简单介绍常用的物理层标准。

2.1 物理层的基本概念

首先要强调指出，物理层考虑的是怎样才能在连接各种计算机的传输媒体上传输数据比特流，而不是指连接计算机的具体的物理设备或具体的传输媒体。大家知道，现有的计算机网络中的物理设备和传输媒体的种类非常繁多，而通信手段也有许多不同方式。物理层的作用正是要尽可能地屏蔽掉这些差异，使物理层上面的数据链路层感觉不到这些差异，这样就可使数据链路层只需要考虑如何完成本层的协议和服务，而不必考虑网络具体的传输媒体是什么。用于物理层的协议也常称为物理层**规程**(procedure)。其实物理层规程就是物理层协议。只是在“协议”这个名词出现之前人们就先使用了“规程”这一名词。

可以将物理层的主要任务描述为确定与传输媒体的接口的一些特性，即：

（1）**机械特性** 指明接口所用接线器的形状和尺寸、引线数目和排列、固定和锁定装置等等。这很像平时常见的各种规格的电源插头的尺寸都有严格的规定。

（2）**电气特性** 指明在接口电缆的各条线上出现的电压的范围。

（3）**功能特性** 指明某条线上出现的某一电平的电压表示何种意义。

（4）**规程特性** 指明对于不同功能的各种可能事件的出现顺序。

在物理连接上的传输方式一般都是**串行传输**，即一个一个比特按照时间顺序传输。但是，有时也可以采用多个比特的**并行传输**方式。出于经济上的考虑，远距离的传输通常都是串行传输。

具体的物理层协议是相当复杂的。这是因为物理连接的方式很多（例如，可以是点对点的，也可以采用多点连接或广播连接），而传输媒体的种类也非常之多（如架空明线、双绞线、对称电缆、同轴电缆、光缆，以及各种波段的无线信道等）。因此在学习物理层时，应将重点放在掌握基本概念上。

考虑到使用本教材的一部分读者可能没有学过“接口与通信”或有关数据通信的课程，因此我们利用下面的 2.2 节简单地介绍一下有关现代通信的一些最基本的知识和最重要的结论（但不给出证明）。对于已具有这部分知识的读者可略过这部分内容。

2.2 数据通信的基础知识

2.2.1 数据通信系统的模型

下面我们通过一个最简单的例子来说明数据通信系统的模型。这个例子就是两个 PC 机

经过普通电话机的连线，再经过公用电话网进行通信。

如图 2-1 所示，一个数据通信系统可划分为三大部分，即**源系统**（或**发送端**）、**传输系统**（或**传输网络**）和**目的系统**（或**接收端**）。

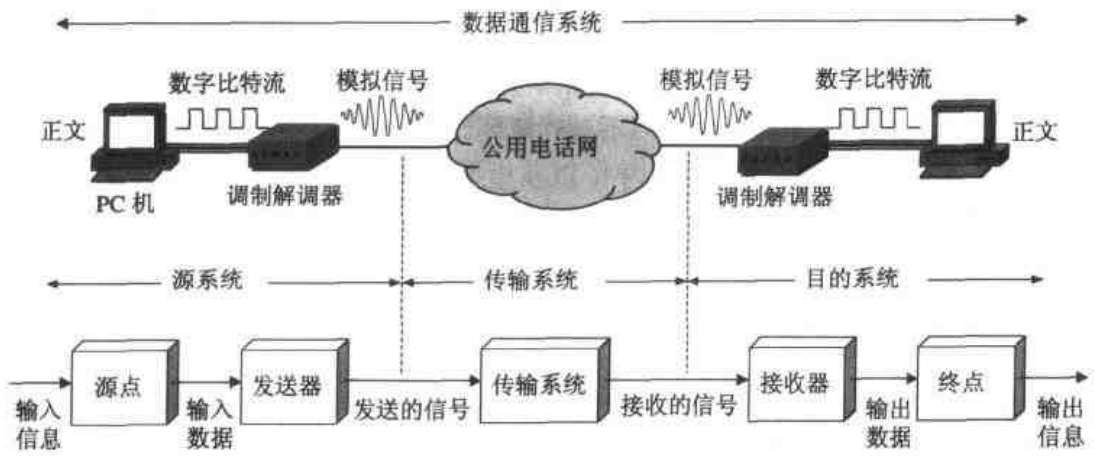


图 2-1　数据通信系统的模型

源系统一般包括以下两个部分：

- **源点**：源点设备产生要传输的数据，例如正文输入到 PC 机，产生输出的数字比特流。源点又称为源站。
- **发送器**：通常源点生成的数据要通过发送器编码后才能够在传输系统中进行传输。例如，调制解调器将 PC 机输出的数字比特流转换成能够在用户的电话线上传输的模拟信号。现在很多 PC 机使用内置的调制解调器，用户在 PC 机外面看不见调制解调器。

目的系统一般也包括以下两个部分：

- **接收器**：接收传输系统传送过来的信号，并将其转换为能够被目的设备处理的信息。例如，调制解调器接收来自传输线路上的模拟信号，并将其转换成数字比特流。
- **终点**：终点设备从接收器获取传送来的信息。终点又称为目的站。

在源系统和目的系统之间的传输系统可能是简单的传输线，也可以是连接在源系统和目的系统之间的复杂网络系统。

图 2-1 所示的数据通信系统，说它是计算机网络也可以。这里我们使用数据通信系统这个名词，主要是为了从通信的角度来介绍一个数据通信系统中的一些要素，而有些数据通信的要素在计算机网络中可能就不去讨论它们了。

在进一步讨论图 2-1 中的一些细节之前，我们先要明确几个术语。

数据(data)是运送信息的实体。而**信号**(signal)则是数据的电气的或电磁的表现。无论数据或信号，都既可是模拟的也可是数字的。所谓“**模拟的**”就是连续变化的，而“**数字的**”就表示取值仅允许为有限的几个离散数值。例如，**数字数据**(digital data)就是用取值为不连续数值的数据。

虽然数字化已成为当今的趋势，但这并不等于说：使用数字数据和数字信号就一定是“先进的”，而使用模拟数据和模拟信号就一定是“落后的”。数据究竟是应当数字的还是模拟的，是由所产生的数据的性质决定的。例如，当我们说话时，声音大小是连续变化的，因此运送话音信息的声波就是模拟数据。但数据必须转换成为信号才能在网络媒体上传输。可是有的传输媒体只适合于传送模拟信号。因此，即使数据是数字形式的，有时我们仍要将数字数据

转换为模拟信号方能在这种媒体上传输。将数字数据转换为模拟信号的过程叫作**调制**。

有了上述的一些基本概念，就可以理解图 2-1 所示的数据通信系统的模型了。这里要指出的是，如果网络的传输信道都是合适于传送数字信号，那么 PC 机输出的数字比特流就没有必要再转换为模拟信号了。现在因为要使用一段电话用户线（这是当初为模拟电话建造的），所以必须使用调制解调器（使用其中的调制器）将 PC 机输出的数字信号转换为模拟信号。在公用电话网中，在交换机之间的中继线路大都已经数字化了。因此模拟信号还必须转换为数字信号才能在数字信道上传输。为简单起见，这部分信号的变化在图中没有画出。等到信号要进入接收端的用户线时，数字信号再转换为模拟信号。最后再经过调制解调器（这时是使用其中的解调器）转换为数字信号进入接收端的计算机（这叫作**解调**），经计算机的处理，再恢复成正文。在学习计算机网络时，我们一定要搞清在某处的信号是数字的还是模拟的。

一般说来，模拟数据和数字数据都可以转换为模拟信号或数字信号。因此我们有以下四种情况：

（1）**模拟数据、模拟信号**　最早的电话系统就是这样的。

（2）**模拟数据、数字信号**　将模拟数据转化成数字形式后，就可以使用先进的数字传输和交换设备。

（3）**数字数据、模拟信号**　为什么数字数据要转换成模拟信号来传输呢？这是因为有些传输媒体只适合于传播模拟信号，譬如光纤和无线信道。使用这样的信道时，必须将数字数据经调制变换为模拟信号后才能传输。

（4）**数字数据、数字信号**　一般说来，把数字数据编码成数字信号的设备比起从数字到模拟的调制设备更简单，更廉价。

图 2-2 给出了模拟的和数字的数据、信号的示意图。

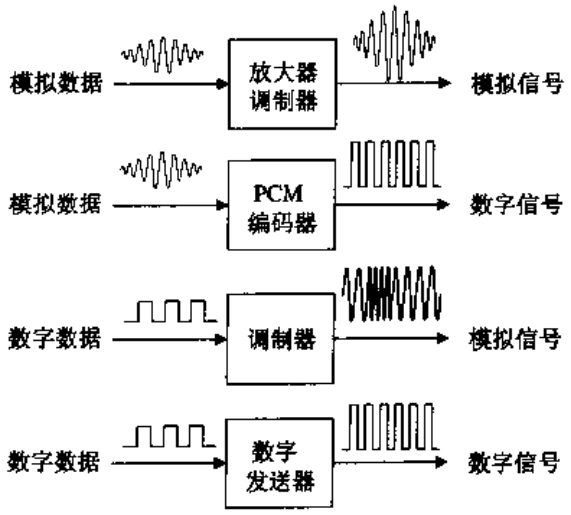


图 2-2　模拟数据、模拟信号、数字数据和数字信号

2.2.2　有关信道的几个基本概念

在许多情况下，我们要使用“**信道**(channel)”这一名词。信道和电路并不等同。信道一般都是用来表示向某一个方向传送信息的媒体。因此，一条通信电路往往包含一条发送信道

和一条接收信道。

从通信的双方信息交互的方式来看，可以有以下三个基本方式：

（1）**单向通信**　又称为**单工通信**，即只能有一个方向的通信而没有反方向的交互。无线电广播或有线电广播以及电视广播就属于这种类型。

（2）**双向交替通信**　又称为**半双工通信**，即通信的双方都可以发送信息，但不能双方同时发送（当然也就不能同时接收）。这种通信方式是一方发送另一方接收，过一段时间后再反过来。

（3）**双向同时通信**　又称为**全双工通信**，即通信的双方可以同时发送和接收信息。

单向通信只需要一条信道，而双向交替通信或双向同时通信则都需要两条信道（每个方向各一条）。显然，双向同时通信的传输效率最高。

这里要提醒读者注意，有时人们也常用“**单工**”这个名词表示“双向交替通信”。如常说的“单工电台”并不是只能进行单向通信。正因为如此，ITU-T 才不采用“单工”、“半双工”和“全双工”这些容易弄混术语作为正式的名词。

信道可以分成传送模拟信号的**模拟信道**和传送数字信号的**数字信道**两大类。但应注意，数字信号在经过数模变换后就可以在模拟信道上传送，而模拟信号在经过模数变换后也可在数字信道上传送。

信道上传送的信号还有**基带**(baseband)信号和**宽带**(broadband)信号之分。简单说来，所谓基带信号就是将数字信号 1 或 0 直接用两种不同的电压来表示，然后送到线路上去传输。而宽带信号则是将基带信号进行调制后形成的频分复用模拟信号。基带信号进行调制后，其频谱搬移到较高的频率处。由于每一路基带信号的频谱被搬移到不同的频段，因此合在一起后并不会互相干扰。这样做就可以在一条电缆中同时传送许多路的数字信号，因而提高了线路的利用率。

在通信网的发展初期，所有的通信信道都是模拟信道。但由于数字信道可提供更高的通信服务质量，因此过去建造的模拟信道正在被新的数字信道所代替。现在计算机通信所使用的通信信道，在主干线路上已基本是数字信道，但目前大量的用户线则基本上还是传统的模拟信道。模拟信道与数字信道并存的局面也使得物理层的内容较为复杂。

有了上述的一些基本概念之后，我们再讨论信道的极限容量。这就是信道上的最高码元传输速率和信道上的最高信息传输速率。

2.2.3　信道的最高码元传输速率

为了提高信号的传输效率，我们总是希望在一定的时间内能够传输尽可能多的码元。然而任何实际的信道都不是理想的，在传输信号时会产生各种失真以及带来多种干扰。图 2-3 给出了数字信号通过实际的信道会引起输出波形失真的示意图。我们可以看出，当失真不严重时（图 2-3(a)），在输出端还可根据已失真的输出波形还原出发送的码元来。但当失真严重时（图 2-3(b)），在输出端就很难判断这个信号在什么时候是 1 和在什么时候是 0。码元传输的速率越高，或信号传输的距离越远，在信道的输出端的波形的失真就越严重。

然而即使信道比较理想，码元的传输速率也不是不受限制的。早在 1924 年，奈奎斯特(Nyquist)就推导出**在理想低通信道下的最高码元传输速率的公式**：

$$\text{理想低通信道的最高码元传输速率} = 2W \text{ Baud} \qquad (2\text{-}1)$$

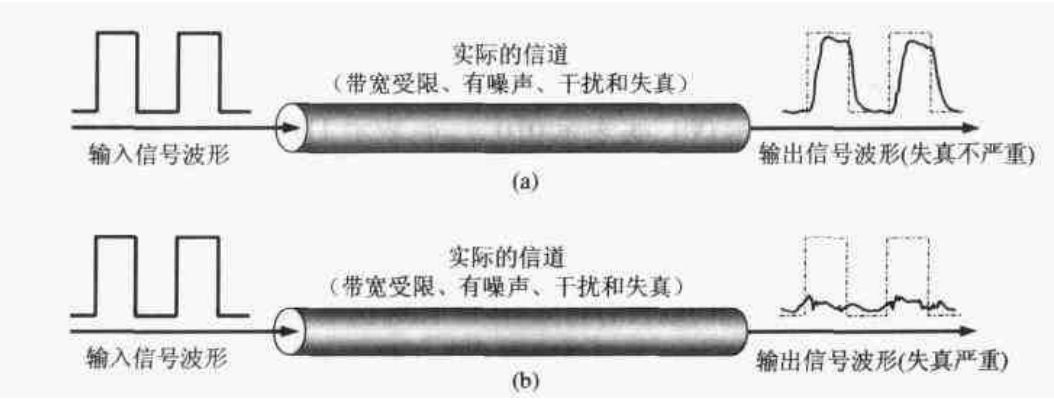


图 2-3　数字信号通过实际的信道：(a)失真不严重；(b)失真严重。

这里 W 是理想低通信道[①]的带宽，单位为赫(Hz)；

Baud 是波特，是码元传输速率的单位，1 波特为每秒传送 1 个码元。

(2-1)式就是著名的**奈氏准则**。奈氏准则的另一种表达方法是：**每赫带宽的理想低通信道的最高码元传输速率是每秒 2 个码元**。若码元的传输速率超过了奈氏准则所给出的数值，则将出现码元之间的相互干扰，以致在接收端无法正确判定在发送方所发送的码元是 1 还是 0。

对于**具有理想带通矩形特性的信道**[②]（带宽为 W），奈氏准则就变为：

$$\text{理想带通信道的最高码元传输速率} = W\ \text{Baud} \qquad (2\text{-}2)$$

即**每赫带宽的带通信道的最高码元传输速率为每秒 1 个码元**。

这里我们要强调以下两点：

（1）上面所说的具有理想低通特性的信道是**理想化的信道**，它和实际上所使用的信道当然有相当大的差别。所以一个实际的信道所能传输的最高码元速率，要明显地低于奈氏准则给出的这个上限数值。

（2）**波特和比特是两个不同的概念**。

波特是**码元传输的速率单位**，它说明每秒传多少个码元。码元传输速率也称为**调制速率、波形速率或符号速率**。

比特是**信息量的单位**，与码元的传输速率“波特”是两个完全不同的概念。

但是，信息的传输速率“比特/秒”与码元的传输速率“波特”在数量上却有一定的关系。若 1 个码元只携带 1 bit 的信息量，则“比特/秒”和“波特”在数值上是相等的。但若使 1 个码元携带 n bit 的信息量，则 M Baud 的码元传输速率所对应的信息传输速率为 $M \times n$ b/s。关于这个问题，最好用一个例子来说明。

【例】　有一个带宽为 3 kHz 的理想低通信道，其最高码元传输速率为 6000 Baud。若 1 个码元能携带 3 bit 的信息量，则最高信息传输速率为 18000 b/s。

那么，怎样才能使一个码元携带 3 bit 的信息量呢？假定我们的基带信号是：

101011000110111010…

我们将这个信号中的每 3 个比特编为一个组，即 101, 011, 000, 110, 111, 010, …。3 个比特共

① 注：“理想低通信道”就是信号的所有低频分量，只要其频率不超过某个上限值，都能够不失真地通过此信道。而频率超过该上限值的所有高频分量都不能通过该信道。

② 注：“理想带通信道”就是频率在频带下限频率 f_1 到频带上限频率 f_2 之间的频率分量能够不失真地通过此信道，而低于 f_1 和高于 f_2 的所有频率分量都不能通过该信道。

有 8 种不同的排列。我们可以不同的调制方法（见后面图 2-16）来表示这样的信号。可以用 8 种不同的振幅，或 8 种不同的频率，或 8 种不同的相位进行调制。现在假定我们采用相位调制，用相位φ_0表示 000, φ_1 表示 001, φ_2 表示 010,..., φ_7 表示 111。这样，原来的信号就转换为：

$$\varphi_5\ \varphi_3\ \varphi_0\ \varphi_6\ \varphi_7\ \varphi_2...$$

也就是说，原来要发送 18 个码元，每个码元只携带 1 bit 的信息量。但经过变化后，只需要发送 6 个码元，而每个码元（它们的载波的相位不同）能够携带 3 bit 的信息量。若以同样的速率发送码元，则同样时间所传送的信息量就提高到了 3 倍。关于这个问题后面还要讨论。这里主要是说明，“比特/秒”和“波特”在概念上是完全不同的。

2.2.4 信道的极限信息传输速率

1948 年，香农(Shannon)用信息论的理论推导出了带宽受限且有高斯白噪声干扰的信道的极限信息传输速率。当用此速率进行传输时，可以做到不产生差错。如用公式表示，则**信道的极限信息传输速率** C 可表达为

$$C = W \log_2(1+S/N)\ \text{b/s} \qquad (2\text{-}3)$$

其中 W 为信道的带宽（以 Hz 为单位）；

S 为信道内所传信号的平均功率；

N 为信道内部的高斯噪声功率。

公式(2-3)就是著名的**香农公式**。香农公式表明，**信道的带宽或信道中的信噪比越大，则信息的极限传输速率就越高**。但更重要的是，香农公式指出了：只要信息传输速率低于信道的极限信息传输速率，就一定可以找到某种办法来实现无差错的传输。不过，香农没有告诉我们具体的实现方法。这要由研究通信的专家去寻找。[①]

从香农公式可看出，若信道带宽 W 或信噪比 S/N 没有上限（实际的信道当然不可能是这样的），那么信道的极限信息传输速率 C 也就没有上限。

自从香农公式发表后，各种新的信号处理和调制方法不断出现，其目的都是为了尽可能地接近香农公式给出的传输速率极限。在实际信道上能够达到的信息传输速率要比香农的极限传输速率低不少。这是因为在实际信道中，信号还要受到其他一些损伤，如各种脉冲干扰和在传输中产生的失真等等。这些因素在香农公式的推导过程中并未考虑。

由于码元的传输速率受奈氏准则的制约，所以要提高信息的传输速率，**就必须设法使每一个码元能携带更多个比特的信息量**。这就需要采用**多元制**（又称为多进制）的调制方法。例如，当采用 16 元制时，一个码元可携带 4 bit 的信息。一个标准电话话路的频带为 300～3400 Hz，即带宽为 3100 Hz。在这频带中接近于理想信道的也就是靠中间的一段，其带宽约为 2400 Hz 左右。如使码元的传输速率为 2400 Baud(这相当于每赫带宽的码元传输速率为 1 Baud)，则信息的传输速率即可达到 9600 b/s。读者从(2-3)式可以很容易地计算出所需信噪比的最低值。但实际信道所需的信噪比要比这个最低值还要高不少。

对于 3.1 kHz 带宽的标准电话信道，如果信噪比 $S/N = 2500$，那么由香农公式可以知道，无论采用何种先进的编码技术，信息的传输速率一定不可能超过由(2-3)式算出的极限数值，

---

① 注：信噪比 S/N 是信号功率和噪声功率之比，常用分贝(dB)表示。即信噪比(dB)= 10 log10 (S/N) (dB)。例如，当 $S/N = 10$ 时，信噪比为 10 dB，而当 $S/N = 1000$ 时，信噪比为 30 dB。

即 35 kb/s 左右。若想超过这个数值，只能设法提高信道中的信噪比，或者提高信道的传输带宽。

图 2-4 说明了奈氏准则和香农公式在数据通信系统中的作用范围。

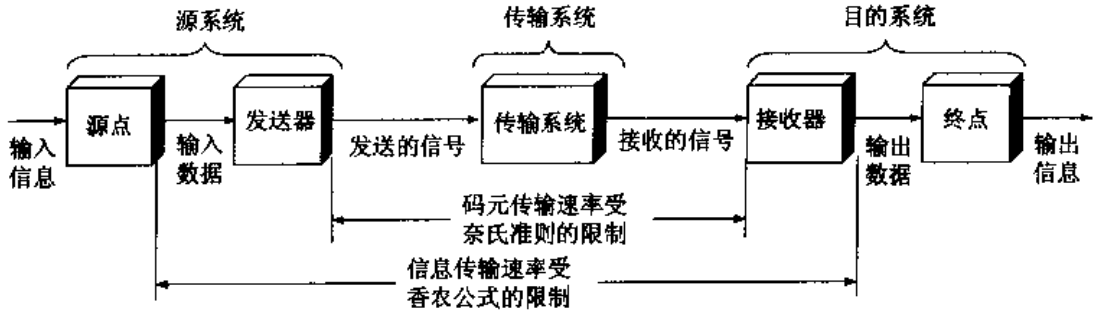


图 2-4　奈氏准则和香农公式在数据通信系统中的作用范围

这里我们要再强调一下，传输媒体处于物理层之下。在通信媒体上传输的是只信号。**传输媒体本身没有物理层**，因此传输媒体并不知道它传输了多少比特，当然也更不知道传输了多少帧或分组。不过通常为了方便，我们还是习惯于不太严格地说："分组或数据帧在某一条通信线路上传输"。但实际上应当是："分组或数据帧的信号在某一条通信线路上传输"。

2.3　物理层下面的传输媒体

传输媒体也称为传输介质或传输媒介，它就是数据传输系统中在发送器和接收器之间的物理通路。传输媒体可分为两大类，即**导向传输媒体**和**非导向传输媒体**。在导向传输媒体中，电磁波被导向沿着固体媒体（铜线或光纤）传播，而非导向传输媒体就是指自由空间，在非导向传输媒体中电磁波的传输常称为无线传输。图 2-5 是电信领域使用的电磁波的频谱。

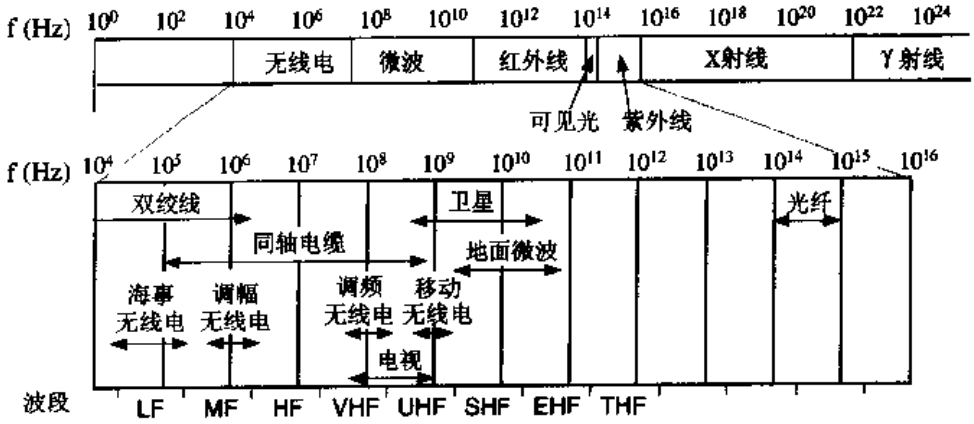


图 2-5　电信领域使用的电磁波的频谱

2.3.1　导向传输媒体

1．双绞线

双绞线也称为双扭线，它是最古老但又是最常用的传输媒体。把两根互相绝缘的铜导线

并排放在一起，然后用规则的方法**绞合**(twist)起来就构成了双绞线。绞合可减少对相邻导线的电磁干扰。使用双绞线最多的地方就是到处都有的电话系统。几乎所有的电话都用双绞线连接到电话交换机。这段从用户电话机到交换机的双绞线称为**用户线**或**用户环路**(subscriber loop)。通常将一定数量的这种双绞线捆成电缆，在其外面包上硬的护套。

模拟传输和数字传输都可以使用双绞线，其通信距离一般为几到十几公里。距离太长时就要加放大器以便将衰减了的信号放大到合适的数值（对于模拟传输），或者加上中继器以便将失真了的数字信号进行整形（对于数字传输）。导线越粗，其通信距离就越远，但导线的价格也越高。在数字传输时，若传输速率为每秒几个兆比特，则传输距离可达几公里。由于双绞线的价格便宜且性能也不错，因此使用十分广泛。

为了提高双绞线的抗电磁干扰的能力，可以在双绞线的外面再加上一个用金属丝编织成的屏蔽层。这就是**屏蔽双绞线**，简称为 STP (Shielded Twisted Pair)。它的价格当然比**无屏蔽双绞线** UTP (Unshielded Twisted Pair)要贵一些。图 2-6 是无屏蔽双绞线和屏蔽双绞线的示意图。

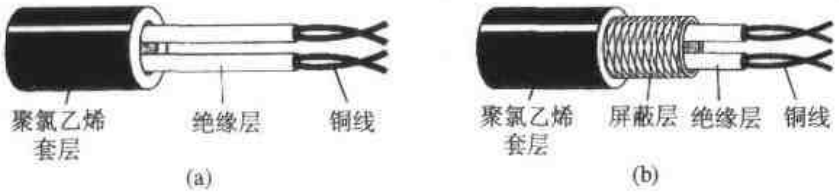


图 2-6　无屏蔽双绞线 (a)和屏蔽双绞线 (b)的示意图

1991 年，美国电子工业协会 EIA (Electronic Industries Association)和电信工业协会 TIA 联合发布了一个标准 EIA/TIA-568，它的名称是“商用建筑物电信布线标准”(Commercial Building Telecommunications Cabling Standard)。这个标准规定了用于室内传送数据的无屏蔽双绞线和屏蔽双绞线的标准。随着局域网上数据传送速率的不断提高，EIA/TIA 在 1995 年将布线标准更新为 EIA/TIA-568-A。此标准规定了 5 个种类的 UTP 标准（从 1 类线到 5 类线）。对传送数据来说，现在最常用的 UTP 是 5 类线(Category 5 或 CAT5)。

5 类线与 3 类线(CAT3)的最主要的区别就是一方面大大增加了每单位长度的绞合次数。3 类线的绞合长度是 7.5 至 10 cm，而 5 类线的绞合长度是 0.6 至 0.85 cm。另一方面，5 类线在线对间的绞合度和线对内两根导线的绞合度都经过了精心的设计，并在生产中加以严格的控制，使干扰在一定程度上得以抵消，从而提高了线路的传输特性。表 2-1 是特性阻抗为 100 Ω 的 UTP 中的 3 类线和 5 类线以及 150 Ω的 STP 的衰减和近端串扰的比较。

表 2-1　无屏蔽双绞线和屏蔽双绞线的衰减和近端串扰的比较

	每 100 m 长的衰减(dB)			近端串扰(dB)		
频率(MHz)	UTP 3 类线	UTP 5 类线	150 Ω的 STP	UTP 3 类线	UTP 5 类线	150 Ω的 STP
1	2.6	2.0	1.1	41	62	58
4	5.6	4.1	2.2	32	53	58
16	13.1	8.2	4.4	23	44	50.4
25		10.4	6.2	—	41	47.5
100		22.0	12.3	—	32	38.5
300	—	—	21.4	—	—	31.3

近端串扰(Near-End crossTalk，通常记为 NEXT)是双绞线性能中的一个重要指标，它是指来自一对导体的信号在另一对导体上发生的耦合现象。这些导体可能是连接头中的金属针或是电缆中的线对。近端指的是耦合发生在被传输的信号进入链路时又耦合到同一端的接收导体，即近端发送的信号被近端接收线收到。

我们注意到，无论是哪一种线，衰减都随频率的升高而增大。在设计布线时，要考虑到受到衰减的信号还应当有足够大的振幅，以便在有噪声干扰的条件下能够在接收端正确地被检测出来。双绞线究竟能够传送多高速率(Mb/s)的数据还与数字信号的编码方法有很大的关系。随着技术的发展，EIA/TIA-568 标准还会不断修订。例如，超 5 类线和 6 类线都已开始试用，我们应注意这些新的布线技术的进展。

2. 同轴电缆

同轴电缆由内导体铜质芯线（单股实心线或多股绞合线）、绝缘层、网状编织的外导体屏蔽层（也可以是单股的）以及保护塑料外层所组成（图 2-7）。由于外导体屏蔽层的作用，同轴电缆具有很好的抗干扰特性，现被广泛用于传输较高速率的数据。

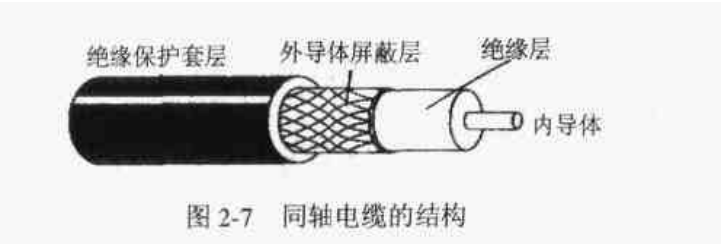


图 2-7　同轴电缆的结构

当需要将计算机连接到同轴电缆上的某一处时，比用双绞线要麻烦得多。通常都是利用**T 型分接头**（或称为 T 型连接器，即 T junction）。T 型接头主要有两种。一种必须先把电缆剪断，然后再进行连接。另一种则不必剪断电缆，但要用另一种较昂贵的、特制的**插入式分接头**(vampire tap)，利用螺丝分别将两根电缆的内外导线连接好。保持电缆接头处的接触良好，是使用电缆作为传输媒体时必须特别加以注意的事项。

通常按特性阻抗数值的不同，将同轴电缆分为两类：

（1）50Ω 同轴电缆

主要用于在数据通信中传送基带数字信号。因此，50 Ω同轴电缆又称为**基带同轴电缆**。用这种同轴电缆以 10 Mb/s 的速率将基带数字信号传送 1 km 是完全可行的。一般说来，传输速率越高，所能传送的距离就越短。在局域网中广泛使用这种同轴电缆作为物理媒体。

在传输基带数字信号时，可以有多种不同的编码方法。图 2-8 画的是未经编码的原基带数字信号和在计算机网络中常用的两种编码方法，即：**曼彻斯特**(Manchester)**编码**和**差分曼彻斯特编码**。未经编码的二进制基带数字信号就是高电平和低电平不断交替的信号。至于用高电平还是用低电平代表 1 或 0 都是可以的。使用这种最简单的基带信号的最大问题就是当出现一长串的连 1 或连 0 时，在接收端无法从收到的比特流中提取位同步信号。曼彻斯特编码则可解决这一问题。它的编码方法是将每一个码元再分成两个相等的间隔。码元 1 是在前一个间隔为高电平而后一个间隔为低电平。码元 0 则正好相反，从低电平变到高电平。这种编码的好处就是可以保证在每一个码元的正中间时间出现一次电平的转换，这对接收端的提取位同步信号是非常有利的。但是从曼彻斯特编码的波形图不难看出其缺点，这就是它所占的频带宽度比原始的基带信号增加了一倍。

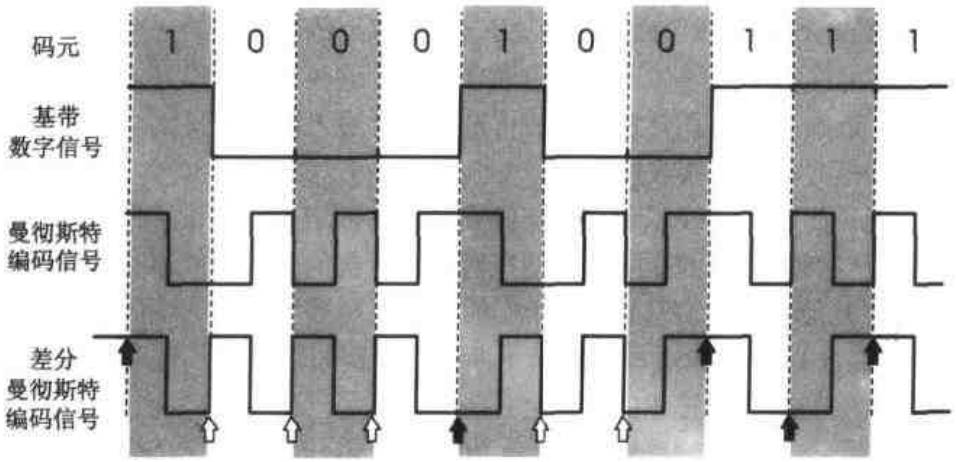


图 2-8　三种常用的编码方法

另一种曼彻斯特编码的变种叫做**差分曼彻斯特编码**，它的编码规则是：若码元为 1，则其前半个码元的电平与上一个码元的后半个码元的电平一样（见图中的实心箭头）；但若码元为 0，则其前半个码元的电平与上一个码元的后半个码元的电平相反（见图中的空心箭头）。不论码元是 1 或 0，在每个码元的正中间的时刻，一定要有一次电平的转换。差分曼彻斯特编码需要较复杂的技术，但可以获得较好的抗干扰性能。

（2）75 Ω 同轴电缆

这种同轴电缆用于模拟传输系统，它是有线电视系统 CATV 中的标准传输电缆。在这种电缆上传送的信号采用了频分复用的宽带信号。这样，75 Ω同轴电缆又称为**宽带同轴电缆**。顺便指出，过去在电话通信系统中，带宽超过一个标准话路(4 kHz)的频分复用系统都可称为是“宽带”的，但在计算机通信中，“宽带系统”常常是指采用了频分复用和模拟传输技术的**同轴电缆网络**。

宽带同轴电缆用于传送模拟信号时，其频率可高达 500 MHz 以上，而传输距离可达 100 km。宽带电缆通常都划分为若干个独立信道，例如，每一个 6 MHz 的信道可以传送一路模拟电视信号。当每一个 6 MHz 信道用来传送数字信号时，数据率一般可达 3 Mb/s。

由于在宽带系统中要用到放大器来放大模拟信号，而这种放大器只能单向工作，因此在宽带电缆的双工传输中，一定要有数据发送和数据接收两条分开的数据通路。采用双电缆系统和单电缆系统都可以达到这个目的。

双电缆系统的示意图如图 2-9(a)所示。电缆的拓扑一般为树形。两套电缆是一样的，分别供计算机发送和接收信号之用。由于发送和接收采用的是不同的电缆，因此可以采用同样的频率。**头端**(headend）的作用是将各计算机从发送电缆发过来的信息转换到接收电缆，使得各计算机能从接收电缆上收到发送给它们的信息。在简单的情况下，头端是无源的。但当电缆较长时，也可在头端和电缆线路中增加一个放大器，使接收电缆上的信号有足够的强度。图中小圆圈为计算机，小四方形为电缆上可能需要加上的放大器。

图 2-9(b)为**单电缆系统**。这种系统是在同一条电缆上进行双向通信。方法是把电缆频带分成两部分。各计算机使用低频段发送信息。头端收到后进行变频，将信息在高频段转发出去，然后各计算机再接收这些信息。虽然单电缆系统只需一条电缆，但现在可用的频带却只有双电缆系统的一半。

从图 2-9 可以看出，头端的可靠性是非常重要的。头端一旦出故障，整个宽带网络就会瘫痪。

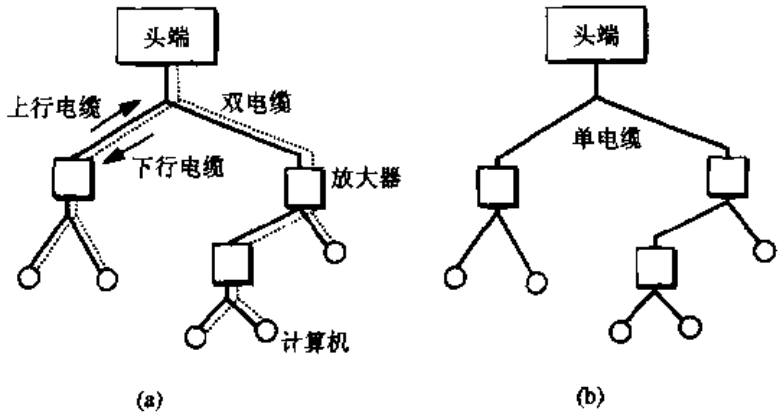


图 2-9　宽带网络：(a)双电缆系统；(b)单电缆系统

3．光缆

从 20 世纪 70 年代到现在，通信和计算机都发展得非常快。二十多年来，计算机的运行速度大约每 10 年提高 10 倍。但在通信领域里，信息的传输速率则提高得更快，从 70 年代的 56 kb/s 提高到现在的几个到几十个 Gb/s(使用光纤通信技术)。相当于每 10 年提高 100 倍。因此光纤通信就成为现代通信技术中的一个十分重要的领域。

光纤通信就是利用光导纤维（以下简称为光纤）传递光脉冲来进行通信。有光脉冲相当于 1，而没有光脉冲相当于 0。由于可见光的频率非常高，约为 10^8 MHz 的量级，因此一个光纤通信系统的传输带宽远远大于目前其他各种传输媒体的带宽。

光纤是光纤通信的传输媒体。在发送端有光源，可以采用发光二极管或半导体激光器，它们在电脉冲的作用下能产生出光脉冲。在接收端利用光电二极管做成光检测器，在检测到光脉冲时可还原出电脉冲。

光纤通常由非常透明的石英玻璃拉成细丝，主要由纤芯和包层构成双层通信圆柱体。纤芯很细，其直径只有 8 至 100μm($1\ \mu m = 10^{-6}$ m)。光波正是通过纤芯进行传导。包层较纤芯有较低的折射率。当光线从高折射率的媒体射向低折射率的媒体时，其折射角将大于入射角（图 2-10）。因此，如果入射角足够大，就会出现全反射，即光线碰到包层时就会折射回纤芯。这个过程不断重复，光也就沿着光纤传输下去。

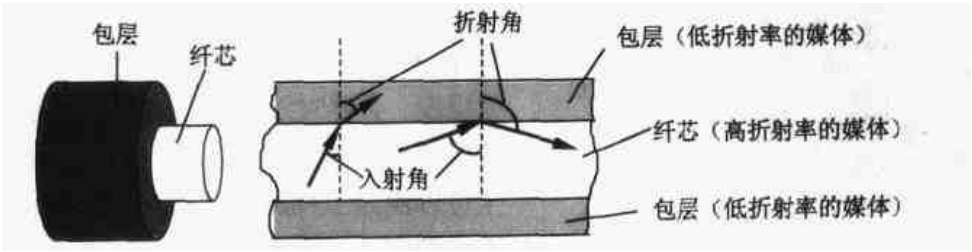


图 2-10　光线在光纤中的折射

图 2-11 画出了光波在纤芯中传播的示意图。现代的生产工艺可以制造出超低损耗的光纤，即做到光线在纤芯中传输数公里而基本上没有什么衰耗。这一点乃是光纤通信得到飞速发展的最关键因素。

图 2-11 中只画了一条光线。实际上，只要从纤芯中射到纤芯表面的光线的入射角大于某一个临界角度，就可产生全反射。因此，可以存在许多条不同角度入射的光线在一条光纤中



图 2-11　光波在纤芯中的传播

传输。这种光纤就称为**多模光纤**（图 2-12(a)）。光脉冲在多模光纤中传输时会逐渐展宽，造成失真。因此多模光纤只适合于近距离传输。若光纤的直径减小到只有一个光的波长，则光纤就像一根波导那样，它可使光线一直向前传播，而不会产生多次反射。这样的光纤就称为**单模光纤**（图 2-12(b)）。单模光纤的纤芯很细，其直径只有几个微米，制造起来成本较高。同时单模光纤的光源要使用昂贵的半导体激光器，而不能使用较便宜的发光二极管。但单模光纤的衰耗较小，在 2.5 Gb/s 的高速率下可传输数十公里而不必采用中继器。

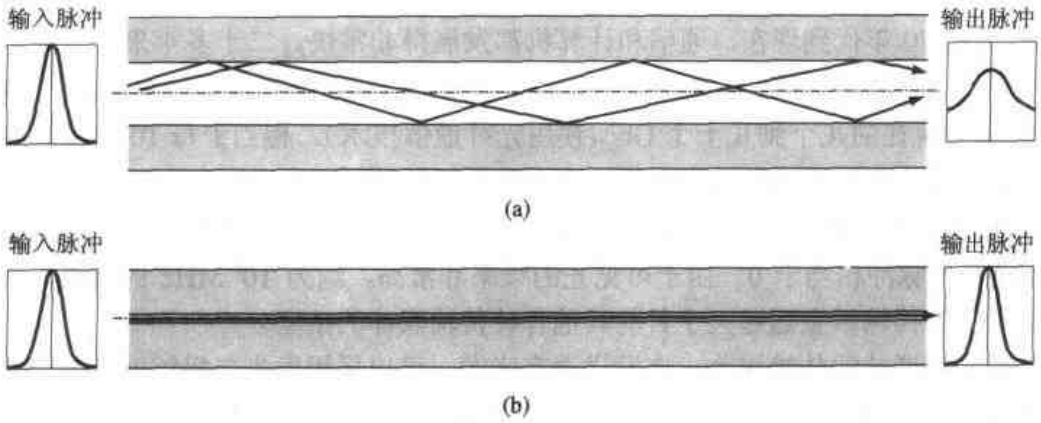


图 2-12　多模光纤(a)和单模光纤(b)的比较

在光纤通信中常用的三个波段的中心分别位于 0.85 μm, 1.30μm 和 1.55 μm。对于后两种情况的衰减都较小。0.85μm 波段的衰减较大，但在此波段的其他特性均较好。所有这三个波段都具有 25 000～30000 GHz 的带宽，可见光纤的通信容量非常大。

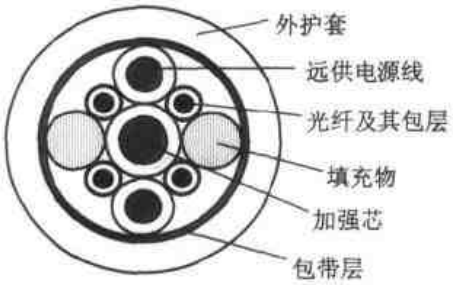


图 2-13　四芯光缆剖面的示意图

由于光纤非常细，连包层一起的直径也不到 0.2 mm。因此必须将光纤做成很结实的光缆。一根光缆少则只有一根光纤，多则可包括数十至数百根光纤，再加上加强芯和填充物就可以大大提高其机械强度。必要时还可放入远供电源线。最后加上包带层和外护套，就可以使抗拉强度达到几公斤，完全可以满足工程施工的强度要求。图 2-13 为四芯光缆剖面的示意图。

光纤不仅具有通信容量非常大的优点，而且还具有其他的一些特点：

（1）传输损耗小，中继距离长，对远距离传输特别经济。

（2）抗雷电和电磁干扰性能好。这在有大电流脉冲干扰的环境下尤为重要。

（3）无串音干扰，保密性好，也不易被窃听或截取数据。

（4）体积小，重量轻。这在现有电缆管道已拥塞不堪的情况下特别有利。例如，1 公里

长的 1000 对双绞线约重 8000 kg，而同样长度但容量大得多的一对光纤仅重 100 kg。

但光纤也有一定的缺点。这就是要将两根光纤精确地连接需要专用设备。目前光电接口还较贵，但价格是在逐年下降的。

当采用光纤连网时，常常将一段段点到点的链路串接起来构成一个环路，通过 T 形接头连接到计算机。

T 形接头有两种：**无源的**和**有源的**。无源的 T 形接头由于完全是无源的，因此非常可靠。它里面有一个光电二极管（供接收用）和一个发光二极管 LED（供发送用），都熔接在主光纤上。即使光电二极管或发光二极管出了故障，也只会使连接的计算机处于脱机状态，而整个光纤网还是连通的。由于在每一个接头处光线会有些损失，因此整个光纤环路的长度受到了限制。

有源的 T 形接头实际上就是一个**有源转发器**（如图 2-14 所示）。进入的光信号通过光电二极管变成电信号，再生放大后，再经过发光二极管 LED 变成光信号继续向前传送。利用有源转发器使得每两个计算机之间的距离可长达数公里。有源转发器的缺点是：一旦 T 形接头出了故障，整个光纤环路即断开不能工作。现在纯光的信号再生器也已开始使用。由于不需要进行光电和电光转换，因此其工作带宽大大增加。

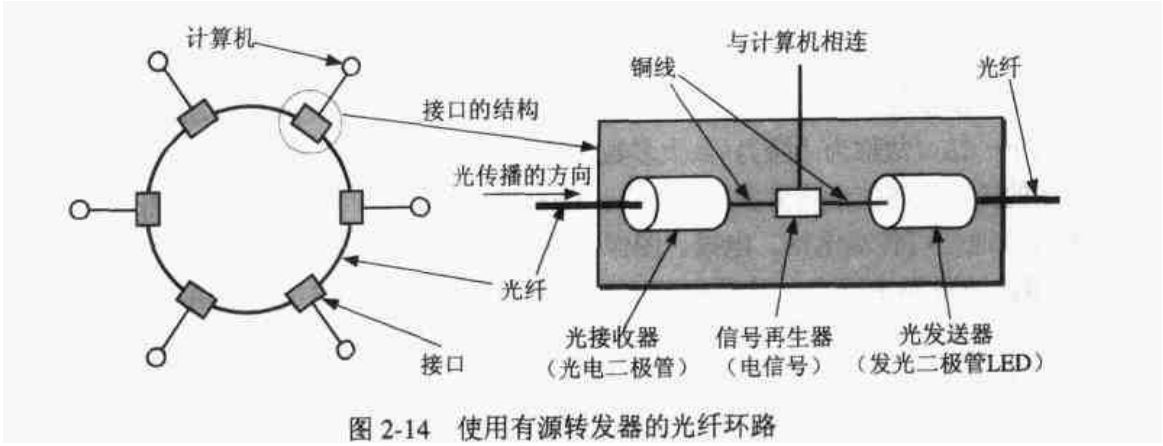


图 2-14　使用有源转发器的光纤环路

最后要提一下，在导向传输媒体中，还有一种是**架空明线**（铜线或铁线）。这是在本世纪初就已大量使用的方法——在电线杆上架设的互相绝缘的明线。架空明线安装简单，但通信质量差，受气候环境等影响较大。在许多国家现在都已停止了铺设架空明线。目前在我国的一些农村和边远地区的通信仍使用架空明线。

2.3.2　非导向传输媒体

前面介绍了三种导向传输媒体。但是，若通信线路要通过一些高山或岛屿，有时就很难施工。即使是在城市中，挖开马路敷设电缆也不是一件很容易的事。当通信距离很远时，敷设电缆既昂贵又费时。但利用无线电波在自由空间的传播就可较快地实现多种的通信。由于这种通信方式不使用上一节所介绍的各种导向传输媒体，因此就将自由空间称为“非导向传输媒体”。

特别要指出的是，由于信息技术的发展，社会各方面的节奏变快了。人们不仅要求能够在运动中进行电话通信（即移动电话通信），而且还要求能够在运动中进行计算机数据通信。因此在最近十几年无线电通信发展得特别快，因为利用无线信道进行信息的传输，是在运动中通信的惟一手段。

无线传输所使用的频段很广。从图 2-5 可以看出，人们现在已经利用了好几个波段进行通信。紫外线和更高的波段目前还不能用于通信。图 2-5 的最下面还给出了 ITU 对波段取的

正式名称。例如，LF 波段的波长是从 10 km 到 1 km(对应于 30 kHz 到 300 kHz)。LF, MF 和 HF 的中文名字分别是低频、中频和高频。更高的频段中的 V, U, S 和 E 分别对应于 Very, Ultra, Super 和 Extremely, 相应的频段的中文名字分别是**甚高频、特高频、超高频和极高频**，最高的一个频段中的 T 是 Tremendously，目前尚无标准译名。在低频 LF 的下面其实还有几个更低的频段，如，甚低频 VLF，特低频 ULF，超低频 SLF 和极低频 ELF 等，因都不用于一般的通信，故未画在图中。

短波（高频, HF）通信主要是靠电离层的反射。但电离层的不稳定所产生的衰落现象和电离层反射所产生的多径效应[①]，使得短波信道的通信质量较差。因此，当必须使用短波无线电台传送数据时，一般都是低速传输，即速率为一个标准模拟话路传几十至几百比特/秒。只有在采用复杂的调制解调技术后，才能使数据的传输速率达到几千比特/秒。

无线电微波通信在数据通信中占有重要地位。微波的频率范围为 300MHz～300GHz，但主要是使用 2～40 GHz 的频率范围。微波在空间主要是直线传播。由于微波会穿透电离层而进入宇宙空间，因此它不像短波那样可以经电离层反射传播到地面上很远的地方。这样，微波通信就有两种主要的方式：**即地面微波接力通信和卫星通信。**

由于微波在空间是直线传播，而地球表面是个曲面，因此其传播距离受到限制，一般只有 50 km 左右。但若采用 100 m 高的天线塔，则传播距离可增大到 100 km。为实现远距离通信必须在一条无线电通信信道的两个终端之间建立若干个中继站。中继站把前一站送来的信号经过放大后再发送到下一站，故称为“接力”。大多数长途电话业务使用 4～6 GHz 的频率范围。目前各国大量使用的微波设备信道容量多为 960 路、1200 路、1800 路和 2700 路。我国多为 960 路。

微波接力通信可传输电话、电报、图像、数据等信息。其主要特点是：

（1）微波波段频率很高，其频段范围也很宽，因此其通信信道的容量很大。

（2）因为工业干扰和天电干扰的主要频谱成分比微波频率低得多，对微波通信的危害比对短波和米波通信小得多，因而微波传输质量较高。

（3）与相同容量和长度的电缆载波通信比较，微波接力通信建设投资少，见效快。

当然，微波接力通信也存在如下的一些缺点：

（1）相邻站之间必须直视，不能有障碍物。有时一个天线发射出的信号也会分成几条略有差别的路径到达接收天线，因而造成失真。

（2）微波的传播有时也会受到恶劣气候的影响。

（3）与电缆通信系统比较，微波通信的隐蔽性和保密性较差。

（4）对大量中继站的使用和维护要耗费一定的人力和物力。

常用的卫星通信方法是在地球站之间利用位于约 36000 公里高空的人造同步地球卫星作为中继器的一种微波接力通信。通信卫星就是在太空的无人值守的微波通信的中继站。可见卫星通信的主要优缺点应当大体上和地面微波通信的差不多。

卫星通信的最大特点是通信距离远，且通信费用与通信距离无关。同步卫星发射出的电磁波能辐射到地球上的通信覆盖区的跨度达 18000 多公里。只要在地球赤道上空的同步轨道上，等距离地放置 3 颗相隔 120 度的卫星，就能基本上实现全球的通信。

---

① 注：多径效应就是同一个信号经过不同的反射路径到达同一个接收点，但各反射路径的衰减和时延都不相同，使得最后得到的合成信号失真很大。

和微波接力通信相似，卫星通信的频带很宽，通信容量很大，信号所受到的干扰也较小，通信比较稳定。为了避免产生干扰，卫星之间相隔如果不小于 2 度，那么整个赤道上空只能放置 180 个同步卫星。好在人们想出来可以在卫星上使用不同的频段来进行通信。因此总的通信容量还是很大的。目前常用的三个频段如表 2-2 所示。

表 2-2 卫星通信常用的三个频段

波段	频率(GHz)	下行(GHz)	上行(GHz)	问　题
C	4/6	3.7～4.2	5.925～6.425	地面上的干扰
Ku	11/14	11.7～12.2	14.0～14.5	降雨
Ka	20/30	17.7～21.7	27.5～30.5	降雨；设备价格贵

一个典型的卫星通常拥有 12～20 个转发器。每个转发器的频带宽度为 36～50 MHz。一个 50 Mb/s 的转发器可用来传输 50 Mb/s 速率的数据，或 800 路 64 kb/s 的数字化话音信道。如果两个转发器使用不同的极化方式，那么即使使用同样的频率也不会产生干扰。

在卫星通信领域中，甚小孔径地球站 VSAT (Very Small Aperture Terminal)已被大量使用。这种小站的天线直径往往不超过 1 米，因而每一个小站的价格就较便宜。在 VSAT 卫星通信网中，需要有一个比较大的中心站用来管理整个卫星通信网。对于某些 VSAT 系统，所有小站之间的数据通信都要经过中心站进行存储转发。对于能够进行电话通信的 VSAT 系统，小站之间的通信在呼叫建立阶段要通过中心站。但在连接建立之后，两个小站之间的通信就可以直接通过卫星进行，而不必再经过中心站。

卫星通信的另一特点就是具有较大的传播时延。由于各地球站的天线仰角并不相同，因此不管两个地球站之间的地面距离是多少（相隔一条街或相隔上万公里），从一个地球站经卫星到另一地球站的传播时延在 250～300 ms 之间。一般可取为 270 ms。这和其他的通信有较大差别（请注意：这和两个地球站之间的距离没有什么关系。对比之下，地面微波接力通信链路的传播时延一般取为 3.3 μs/km。

这里我们要注意的是："卫星信道的传播时延较大"并不等于"用卫星信道传送数据的时延较大"。这是因为传送数据的总时延除了传播时延外，还有发送时延和处理时延这两部分。传播时延在总时延中所占的比例有多大，取决于具体情况。

卫星通信非常适合于广播通信，因为它的覆盖面很广。但从安全方面考虑，卫星通信系统的保密性是较差的。

通信卫星本身和发射卫星的火箭造价都较高。受电源和元器件寿命的限制，同步卫星的使用寿命一般只有 7～8 年。卫星地球站的技术较复杂，价格还比较贵。这些都是选择传输媒体时应全面考虑的。

除上述的同步卫星外，低轨道卫星通信系统已开始使用。低轨道卫星相对于地球不是静止的，而是不停地围绕地球旋转，这些卫星在天空上构成了高速的链路。由于低轨道卫星离地球很近，因此轻便的手持通信设备都能够和卫星进行通信。

红外通信也是使用非导向媒体。可用于近距离的笔记本电脑的相互传送数据。

2.4 模拟传输与数字传输

从概念上讲，对传送计算机数据最合适的应当是数字信道。但早在计算机网络出现之前，

采用模拟传输技术的电话网就已经工作了近一个世纪，并且已遍布在世界上的各个角落。由于数字传输的性能优于模拟传输，因此各国都纷纷将传统的模拟传输干线更换成先进的数字传输干线，并且大量地采用光纤技术。但是从用户的电话机到市话局的用户线，现在还是使用老式的双绞线（铜线）。因此目前的情况是模拟传输与数字传输并存。这样，在学习计算机网络时，我们还需要对传统的模拟传输系统有一定的了解。

下面将讨论有关模拟传输和数字传输的一些最基本的概念。严格说来，“传输”和“交换”是两个不同的概念。但为方便起见，我们在讨论传输的问题时，也要涉及到一些有关交换的概念。

2.4.1 模拟传输系统

传统的电话通信系统都是分级交换。我国的电话网络原先分为 5 级，上面 4 级是长途电话网络，最低一级是市话电话网。现在这 4 级长途交换已改为更加先进的**动态无级选路** DNHR (Dynamic NonHierarchical Routing)体制，即只分两级。在下面的一级是本地网，其交换中心有 320 个左右。在本地网上面就是省的交换中心（30 个），而各省的交换中心组成全连通网络。这样可大大减少转接次数和提高转接速率，也提高了电话的接通率。

从市话局到用户的电话机的用户线是采用最廉价的双绞线电缆。通信距离约为 1～10 km。在电话机较稠密的城市，用户到市话局的距离就比较短。用户环的投资占整个电话网投资的一个相当大的比重。

长途干线最初采用频分复用 FDM (Frequency Division Multiplexing)的传输方式，也就是许多用户可在同样的时间占用大家共享的线路资源，但从频率域来看，它们占用的频率范围是各自分开因而互不干扰。所谓的载波电话就是使用频分复用的电话通信系统。一个标准话路的频率范围是 300～3400 Hz。但由于话路之间应有一些频率间隔，因此国际标准取 4 kHz 为一个标准话路所占用的频带宽度。一般说来，级别越高的交换局之间的长途干线就需要更多的话路容量才能满足通信业务的需求。我们平时常说的 60 路、300 路或 1800 路等，就是指长途干线频分复用的话路数目。

在长途干线中，由于使用了只能单向传输的放大器，因此不能像市话线路那样使用二线制而是要使用四线制，即要用两对线来分别进行发送和接收。也就是说，发送和接收各需要占用一条信道。这样，当市话线路和长途线路相连接时，就需要加入一个二线与四线之间的转换器。我们经常遇到的情况就是在电话用户的两端都是采用二线制的市话线路，而中间的一段则是采用四线制的长途线路。由于二、四线之间的转换不可能是理想的，这就产生了所谓的回波(echo, 又称为回声)的问题。当电话通信的一方说话的话音信号传到对方的二、四线转换器时，不可避免地会有一部分话音信号又反射回来进入讲话人的耳机，因而产生了回波。当通信的距离很长时(例如超过 2000 km)，回波会使讲话人感到很不舒服，严重时会使讲话人无法正常进行电话交谈。为此，在长途电话线路中要装上回波抑制器。回波抑制器在检测到某一方在讲话时，就自动将其接收线路切断，因而抑制了回波。实际上，回波抑制器就是把全双工的电路变为半双工的。由于正常的电话通信是按半双工的方式进行的，所以回波抑制器的加入不会影响正常的电话交谈。但是当装有回波抑制器的电话线路用来传送计算机的数据时，全双工的通信就无法进行。

目前我国长途线路已基本实现数字化，因而现在的模拟电路就基本上只剩下从用户电话机到市话交换机之间的这一段几公里长的用户线上。

2.4.2 调制解调器

1．调制解调器的作用

下面观察一下计算机数据经过模拟传输系统后会出现什么结果。图 2-15 表示了出现一个误码的示意图。

在图 2-15 中接收到的基带信号与发送端发送的信号有很大的不同。这是因为：

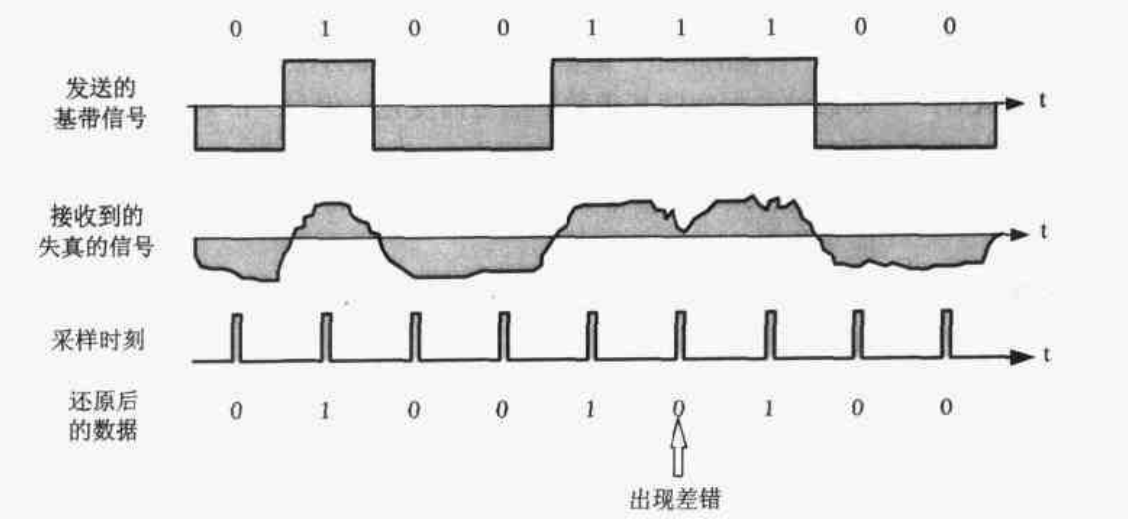


图 2-15　基带信号经电话线路传输后产生误码

（1）发送的基带信号包含有各种的频率成分，其中的一部分已经落到模拟电话通信系统所能通过的频率范围(300～3400 Hz)之外，因而通不过去。由于收到的信号中缺少了这部分频率成分，因此使数字信号产生了失真。

（2）在能够通过电话线路的这部分频率成分中，各频率成分经受的衰减和时延可能会有些不同。这也要产生失真。

（3）电话线路中存在噪声和各种干扰信号，使信号失真。

上述这些因素对传送话音信号同样要产生失真。由于话音信号中的信息冗余度很大，只要电话线路的各项技术指标都满足电话通信的各项标准，即使存在这些失真，电话信号中的主要成分还是能够通过去的。因此人们对这样的电话通信质量仍然是满意的。

但数据通信是靠机器来判定收到的码元是什么。接收端一般是在每个码元的中间时刻产生一个采样时刻，并在此采样时刻对收到的信号进行判决。当失真或干扰严重时就会出现差错，即产生了误码。**若所传送的码元速率越高，信号的失真就越严重**。

为解决上述(1)和(2)两个因素产生的失真，必须将计算机输出的数字信号转换为频率范围在(300～3400 Hz)之间的模拟信号来传输。具体的做法就是在模拟信道两端各加上一个调制解调器。至于要解决上述的因素(3)，则要利用差错检测和纠正技术。

由于计算机之间的通信经常都是双向通信，因此一个调制解调器包括了为发送信号用的调制器和为接收信号用的解调器。**调制解调器**(modem)就是由**调制器**(MOdulator)和**解调器**(DEModulator)这两个字各取其字头合并而成的。如果没有特殊的说明，本书中的调制解调器就是在一条标准的二线模拟话路(3.1 kHz 的标准话路带宽)上提供全双工的异步数字通信的调制解调器。

调制器的主要作用就是个**波形变换器**，它将基带数字信号的波形变换成适合于模拟信道传输的波形（注意：这并不改变数据的内容，即转换后的模拟信号仍然携带原来的数字信号所携带的数字信息）。解调器的作用就是个**波形识别器**，它将经过调制器变换过的模拟信号恢复成原来的数字信号。若识别不正确，则产生误码。在调制解调器中还要有差错检测和纠正的设施，以防止线路上的噪声和干扰引起在传送的信息中产生误码。

2．几种最基本的调制方法

所谓调制就是进行**波形变换**。或者更严格些说，是进行**频谱变换**，将基带数字信号的频谱变换成为适合于在模拟信道中传输的频谱。最基本的二元制调制方法有以下几种（图 2-16）：

（1）**调幅(AM)**　即载波的振幅随基带数字信号而变化。例如，0 对应于无载波输出，而 1 对应于有载波输出。

（2）**调频(FM)**　即载波的频率随基带数字信号而变化。例如，0 对应于频率 f_1，而 1 对应于频率 f_2。

（3）**调相(PM)**　即载波的初始相位随基带数字信号而变化。例如，0 对应于相位 0 度，而 1 对应于相位 180 度。

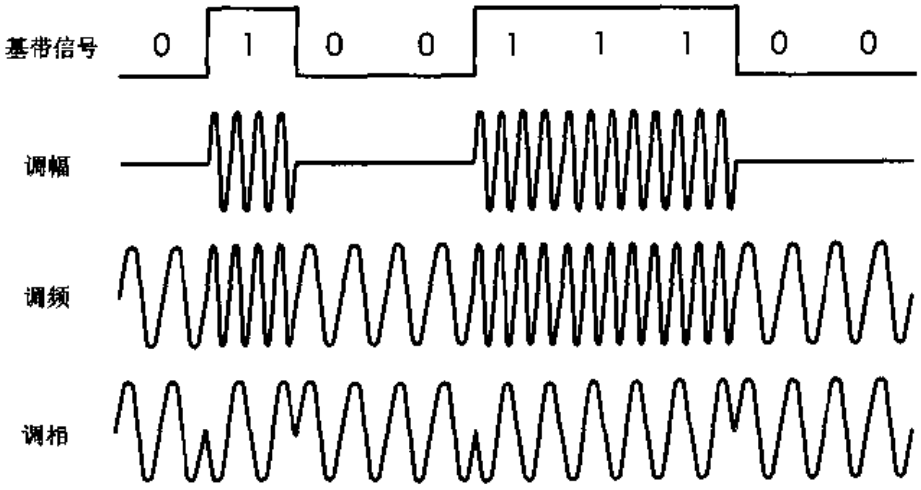


图 2-16　对基带数字信号的几种调制方法

上述的对数字信号的调频和调相，分别称为**移频键控** FSK (Frequency Shift Keying)和**移相键控** PSK (Phase Shift Keying)。而对移相键控还可再分为**绝对移相键控和相对移相键控** (DPSK)，即 0 对应于相位发生变化，而 1 对应于相位不变化。由于检测相位的变化要比检测相位本身的数值更加容易，因此 DPSK 具有更好的抗干扰性。

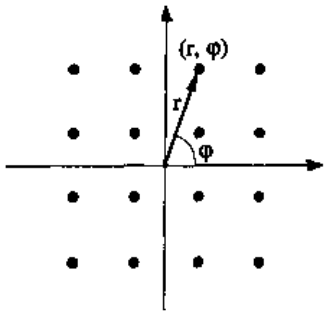


图 2-17　可得到 1 码元表示 4 bit 的正交幅度调制的星座图

为了达到更高的信息传输速率，必须采用技术上更为复杂的多元制的振幅相位混合调制方法。图 2-17 画的是一种**正交调制** QAM (Quadrature Amplitude Modulation)的星座图。可以看出，可供选择的相位有 8 种，而对于每一种相位又有两种振幅可供选择。星座图中的 16 个点的坐标(r, φ)都是

不相同的。这里 r 代表振幅，而φ代表相位。这样我们就可以使用与这 16 个点相对应的 16 种不同的码元来传送数据。由于 4bit 编码共有 16 种不同的组合，因此这 16 个点中的每一个点可对应于一种 4bit 的编码。可见采用这种编码方法，每一码元可表示 4 bit 的信息，因此传送 1 个码元就相当于传送了 4 bit，因而用 2400Baud 的码元速率就可得到 9600 b/s 的信息传送速率。但是，图 2-17 也告诉我们，**若每一个码元可表示的比特数越多**（即在星座图中的点数越多），**则在接收端进行解调时要正确识别每一种状态就越困难**。这是因为线路上的各种干扰和噪声使得在接收端收到的码元的振幅和相位都可能会在一定的范围内变化。因此实际上每一种状态在接收端星座图上对应的并不是一个几何上的点，而是一块面积。若失真太大，这些面积会互相重叠，这就可能无法正确识别状态。

3．关于调制解调器的速率

在近二十年里，调制解调器的速率得到了非常大的提高。在 20 世纪 80 年代中期，在一个标准话路上使用的调制解调器的速率一般也就是 300 b/s。因此，过去将信息传输速率小于 600 b/s 的称为低速调制解调器，而信息传输速率高于 9600 b/s 的称为高速调制解调器。但技术的进步，使得调制解调器的传输速率从 300 b/s 发展到几年前的 28.8～33.6 kb/s(即符合 ITU-T 的 V.34 标准)和 56 kb/s (ITU-T V.90)的水平。这里使用了大量的数字信号处理技术和专用超大规模集成电路 VLSI 芯片。调制解调器信息传输的速率已很接近于香农的信道容量极限了。

在前面的 2.2.4 节已经讲过，信息传输的最高速率取决于信道带宽和信噪比。对于电话信道来说，电话交换机给每一个话路的频带限制为 3.1 kHz 的标准早已确定，这是不能改变的（因为数量极大的电话交换机早已遍布全世界，设备变动耗资太大）。要提高信息传输速率，只能设法提高信噪比。那么信噪比究竟能够提高到什么程度呢？

原先的调制解调器是为两个计算机用户通过网络进行通信用的。图 2-18(a)表示两个计算机用户 A 和 B 通过 V.34 调制解调器(33.6kb/s)进行通信。用户 A 的计算机输出到了调制解调器就变成了模拟信号。模拟信号进入交换机 1，经过 2 线至 4 线的变换后，通过 A/D 变换，变成标准的 64 kb/s 的 PCM 数字信号（关于 PCM 见 2.4.3 节）。以后可能还要经过多个程控交换机的转接（但都保持是这种 PCM 数字信号，如图中的粗线箭头所示），一直到用户 B 接入的交换机 2，通过 D/A 变换才变成模拟信号，经过 4 线至 2 线的变换后，传到用户 B 的 V.34 调制解调器，再经过一次 A/D 变换，还原成计算机可接受的数字信号。统计数据表明，从 A 到 B 的整个传输过程中，最大的噪声来自模拟到数字的模数转换所带来的**量化噪声**①。图 2-18(a)指出了 A 和 B 通信时量化噪声出现的地方（每个方向各有两处）。这种量化噪声已是无法再减小了。因此，按照香农公式得出的信道容量的极限值约为 35 kb/s。也就是说，V.34 的 33.6 kb/s 的速率基本上已达到极限数值了。

然而后来人们发现，提高调制解调器速率的主要动机并不是要在两个用户之间进行通信，而是要通过**因特网服务提供者 ISP** (Internet Service Provider)从因特网上以更高的速率下载有用的信息。由于大部分的 ISP 都使用租用数字专线接入到电话交换机，因此用户与 ISP

① 注：模拟信号是连续变化的，但数字信号只能取离散的数值。因此模拟到数字的模数转换会带来一些误差。这就是说，相接近的许多模拟值都将转换为同样的离散值，而这就使信号还原后和原来的数值有些不同，即产生了失真。这种模数转换产生的误差对信号传输带来的失真相当于一种噪声，称作量化噪声。请注意，数字到模拟的数模转换并不会带来量化噪声。

之间的通信信道是图 2-18(b)所示的那样。我们看出，网络中的下行信道不包含模数转换。从因特网一直到交换机 1，都是数字传输。因此，下行信道的信噪比可以明显地提高。根据这一情况，就研制出了 56 kb/s 的调制解调器。例如，K56flex 和 x2 调制解调器。1998 年 ITU-T 已通过了 56 kb/s 的标准 V.90。我们还可看出，从用户到 ISP 也只需经过一段模拟的用户环路，因而上行信道的信噪比也有所提高。

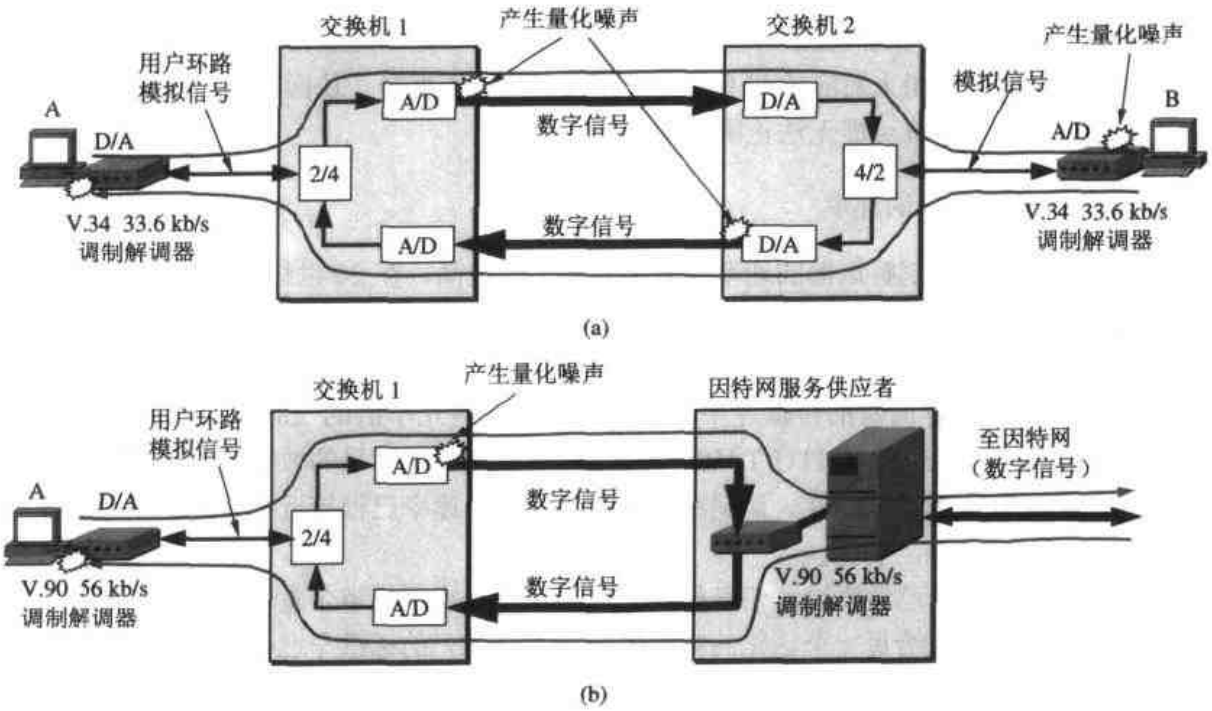


图 2-18　量化噪声产生的地方：(a)用户到用户的通信；(b)用户到 ISP 的通信。

应当指出，56 kb/s 调制解调器是指下行信道的速率为 56 kb/s，即从 ISP 到上网的用户。上行信道的信噪比虽然有所提高，但 V.90 的标准是将上行信道的速率设定为 33.6 kb/s，即仍为 V.34 的标准。显然，上行信道是还有潜力的。因此，在 2000 年 ITU-T 发布了 V.92 的调制解调器标准。虽然下行信道的速率仍为 56 kb/s，但上行信道的速率却提高到了 48 kb/s。此外，V.92 调制解调器的连接建立时间只有 8 至 10 秒（而 V.90 调制解调器的连接建立时间则需 30 秒左右）。顺便指出，若两个用户各使用一个 56 kb/s 的调制解调器，是无法以 56 kb/s 的速率通过网络相互进行通信的（因为 A/D 变换引起的量化噪声太大）。这时，双方的传输速率都将是 33.6 kb/s。56 kb/s 的调制解调器的使用条件是 ISP 使用特殊的数字连接端口（这里是为了进行数字信号不同编码之间的转换，而不是数模转换），并且在 ISP 与电话交换机之间都是数字信道。若 ISP 使用的只是 V.34 调制解调器，则用户端的 56 kb/s 调制解调器会自动降低到与 V.34 调制解调器相同的速率进行通信。若用户携带装有 56 kb/s 调制解调器的笔记本电脑使用某宾馆客房中的电话机插口上网，那么由于这时又经过了宾馆的小交换机的一次模数转换，使调制解调器的速率不能达到 56 kb/s。总之，在使用 56 kb/s 的调制解调器时，若线路的条件达不到图 2-18(b)所示的那样，则这种调制解调器的性能就会下降。

4．调制解调器使用异步通信方式

现在用户在家里上网用的调制解调器都是使用**异步通信方式**（只有某些 UNIX 服务器和

大型机在专用线路环境下才使用同步调制解调器)。“异步(asynchronous)”这个名词需要进一步解释一下。我们先看一下什么是同步通信。

在进行数据通信时，一个很重要的问题是：数字信号传输到接收端时，接收端必须设法判断所收到的码元是 1 还是 0。但是，接收端应当用什么手段才能保证对收到的比特流进行判决的时间是准确的呢？如果这个判决时间取得不准确，就可能导致判决错误，因而无法保证正确接收。从这点出发，数据通信可分为**同步通信**和**异步通信**两大类。

同步通信就是要求接收端的时钟频率和发送端的时钟频率相等（这常称为**收发双方的时钟是同步的**），以便使接收端对收到的比特流的采样判决的时间是准确的。当收发双方的时钟不是精确同步时，在接收端对收到的码元进行判决的时间就会逐渐向前或向后移动。当接收端的判决点移动的时间超过码元宽度的一半时（本来判决点应当处于每一个码元的中间），就要产生差错（比特重读或漏读），这就是所谓的**滑动**(slip)。例如，数据传输的速率是 1 Mb/s，即每 1 μs 发送一个比特。在接收端，采样的时刻应当在每一个比特的中心位置。如果接收端的时钟速率有 1/100 的误差，那么每接收一个比特，采样点就偏离比特的中心位置 0.01 μs。当接收了 50 个比特后，采样点就偏离比特中心位置 0.5 μs(半个比特的宽度)，这时就要产生判决错误。所以像这样的不精确的接收端时钟是不能用于同步通信的。

严格的同步通信是用一个非常精确的主时钟负责全网的同步，全网的其他所有的时钟频率都来自这个主时钟频率（长期精度优于$\pm 1.0 \times 10^{-11}$）。但是这种同步方式需要使用十分复杂的技术，而且价格昂贵。因此在过去相当长的时间，各国的数字网主要是采用准同步(plesiochronous)方式。准同步方式是各有关信号使用一些**独立的、具有相同的频率标称值的时钟源**，但这些频率的实际数值允许有**微小的误差**（在容许范围之内）。

异步通信则采用另一种方法。这就是在发送端将欲发送的数据以字节（8 个比特）为单位进行逐个字节的封装，即对每一个字节增加一个起始比特和一个停止比特，共 10 个比特。然后将这种 10 比特的数据单元一一发送出去。接收端的时钟并没有和发送端的时钟同步，但接收端每收到一个起始比特，就知道有一个 10 比特的数据单元到了，于是开始进行判决，但只判决这个数据单元的 10 个比特。因此，即使接收端的时钟不太准确，只要它能够保证正确接收 10 个比特就行（如果在判决第 10 个比特时采样点的移动已超过半个比特的宽度，那么这种低精度的时钟就不能使用，见习题 2-18）。

异步通信的另一个特点就是发送端在发送完一个字节后（即在停止比特结束后），可以经过任意长的时间间隔再发送下一个字节。当然，每一个字节中的所有比特（包括增加的起始比特和停止比特）的发送时间间隔都必须是恒定的。现在的调制解调器都有对通信线路质量的自适应功能。当线路质量状况不好时，调制解调器发送数据的速率会自动降低，而接收端的调制解调器在进行接收时，也会自动将自己的采样频率降低到合适的数值。从这个意义上讲，异步通信中也包含了某种意义上的同步。

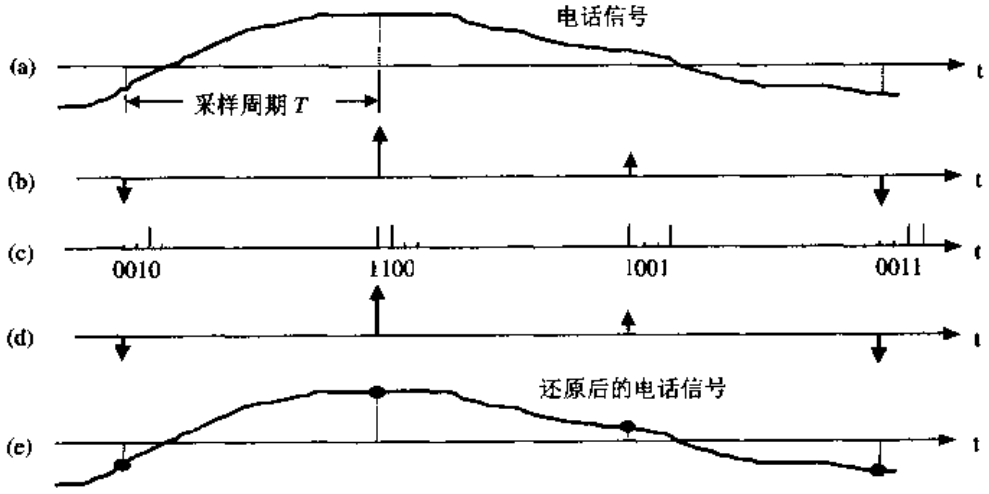
总之，**异步通信是通过增加通信开销**（每发送 10 个比特就有两个比特的额外开销，因而数据的有效传输速率就降低了）**使接收端能够使用廉价的、具有一般精度的时钟来进行数据通信**。用户的调制解调器正好适应异步通信的特点，因为一般用户的通信量并不大，远远不是每天 24 小时连续工作，而且用户也负担不起购买同步通信所需的昂贵设备。

2.4.3 数字传输系统

现在的数字传输系统都是采用**脉码调制** PCM (Pulse Code Modulation)体制。PCM 最初并

不是为传送计算机数据用的。它是为了使电话局之间一条中继线不是只传送一路电话而是可以传送几十路的电话。由于历史上的原因，PCM 有两个互不兼容的国际标准，即北美的 24 路 PCM(简称为 T1)和欧洲的 30 路 PCM(简称为 E1)。我国采用的是欧洲的 E1 标准。T1 的速率是 1.544 Mb/s，E1 的速率是 2.048 Mb/s。下面简单讲一下这些速率是怎样得出的。

为了将模拟电话信号转变为数字信号，必须先对电话信号进行采样。根据采样定理，只要采样频率不低于电话信号最高频率的 2 倍，就可以从采样脉冲信号无失真地恢复出原来的电话信号。标准的电话信号的最高频率为 3.4 kHz，为方便起见，采样频率就定为 8 kHz，相当于采样周期为 125 μs。图 2-19 表示了上述的概念。图 2-19(a)画的是一个模拟电话信号的一段。T 为采样周期。连续的电话信号经采样后成为图 2-19(b)所示的离散脉冲信号，其振幅对应于采样时刻电话信号的数值。下一步就是进行编码。为简单起见，图 2-19(c)将不同振幅的脉冲编为 4 位二进制码元。在我国使用的 PCM 体制中，电话信号是采用 8 bit 编码，也就是说，将采样后的模拟的电话信号量化为 256 个不同等级中的一个。模拟信号转换为数字信号后就进行传输（为提高传输质量，还可再进行一些编码，这里从略）。在接收端进行解码的过程与编码过程相反。只要数字信号在传输过程中不发生差错，解码后就可得出和发送端一样的脉冲信号（图 2-19(d)）。经滤波后最后得出还原后的模拟电话信号（图 2-19(e)）。



(a)模拟电话信号；(b)采样后的脉冲信号；
(c)编码后的数字信号；(d)解码后的脉冲信号；(e)还原后的模拟电话信号。

图 2-19　PCM 的基本原理

这样，一个话路的模拟电话信号，经模数变换后，就变成为每秒 8000 个脉冲信号，每个脉冲信号再编为 8 位二进制码元。因此一个话路的 PCM 信号速率为 64 kb/s。这里要指出，64 kb/s 的速率是最早制订出的话音编码的标准速率。随着话音编码技术的不断发展，人们可以用更低的数据率来传送同样质量的话音信号。现在已经能够用 32 kb/s, 16 kb/s 或甚至低到 8 kb/s 以下的数据率来传送一路话音信号。但是，使用 64 kb/s 标准的电话交换机已经遍及全世界，现在很难再更新换代了。

为了有效地利用传输线路，通常总是将许多个话路的 PCM 信号用**时分复用** TDM (Time Division Multiplexing)的方法装成帧（即时分复用帧），然后再送往线路上一帧接一帧地传输。

图 2-20 说明了 E1 的时分复用帧的构成。不难看出，时分复用是所有的用户在不同的时间，即在分配给自己的专用时隙（当然用完后要归还）占用大家共享的公共信道（因而不会发生干扰）。但从频率域来看，大家所占用的频率范围却都是一样的。

E1 的一个时分复用帧（其长度 T=125 μs）共划分为 32 相等的时隙，时隙的编号为 CH0～CH31。时隙 CH0 用作帧同步用，时隙 CH16 用来传送信令（如用户的拨号信令）。可供用户使用的话路是时隙 CH1～CH15 和 CH17～CH31，共 30 个时隙用作 30 个话路。每个时隙传送 8 bit。因此整个的 32 个时隙共用 256 bit。每秒传送 8000 个帧，因此 PCM 一次群 E1 的数据率就是 2.048 Mb/s。图 2-20 在 2.048 Mb/s 的传输线路两端同步旋转的开关（这只是为阐述原理用的示意图），表示 32 个时隙中的比特的发送和接收必须和时隙的编号相对应，不能弄乱。

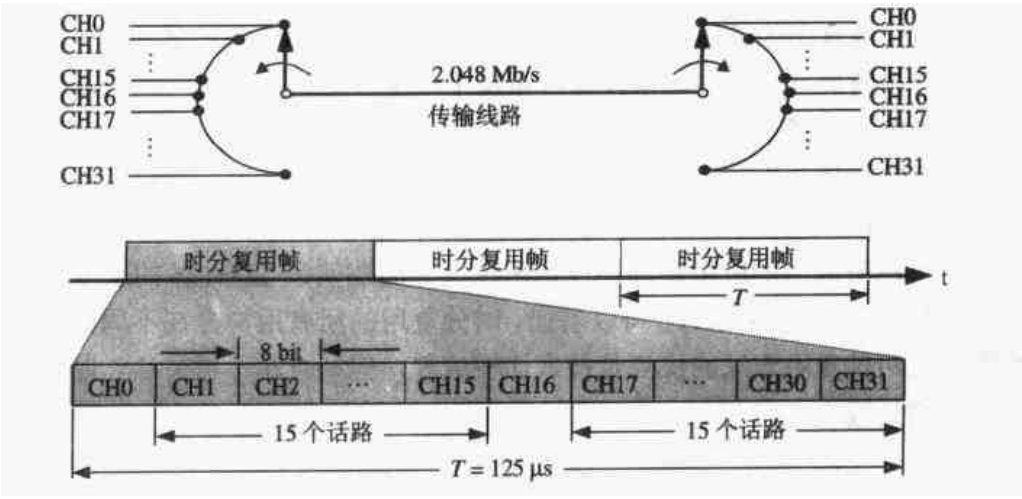


图 2-20　E1 的时分复用帧

北美使用的 T1 系统共有 24 个话路。每个话路的采样脉冲用 7 bit 编码，然后再加上 1 位信令码元，因此一个话路也是占用 8 bit。帧同步码是在 24 路的编码之后加上 1 bit，这样每帧共有 193 bit。因此 T1 一次群的数据率为 1.544 Mb/s。

当需要有更高的数据率时，可以采用复用的方法。例如，4 个一次群就可以构成一个二次群。当然，一个二次群的数据率要比 4 个一次群的数据率的总和还要多一些，因为复用后还需要有一些同步的码元。表 2-3 给出了欧洲和北美系统的高次群的话路数和数据率。日本的一次群用 T1，但自己另有一套高次群的标准。

表 2-3　数字传输系统的高次群的话路数和数据率

系统类型		一次群	二次群	三次群	四次群	五次群
欧洲体制	符号	E1	E2	E3	E4	E5
	话路数	30	120	480	1920	7680
	数据率(Mb/s)	2.048	8.448	34.368	139.264	565.148
北美体制	符号	T1	T2	T3	T4	
	话路数	24	96	672	4032	
	数据率(Mb/s)	1.544	6.312	44.736	274.176	

应当指出，如果在两个计算机之间的通信电路中，传输电路是模拟信道与数字信道交替组成的，那么由于要进行多次模数和数模转换，数字传输的优越性就不能充分发挥。只有整个端到端通信电路都是数字传输，数字传输的优越性才能得到充分的发挥。现在通信网正是

朝着这样的方向去发展的。

2.5 信道复用技术

2.5.1 频分复用、时分复用和统计时分复用

前面已初步介绍了**复用**(multiplexing)的基本概念。这就是频分复用 FDM（按频率划分不同的信道）和时分复用 TDM（按时间划分不同的信道）。这是最基本的信道复用技术。在计算机网络的信道中还广泛地使用其他种复用技术，如**统计时分复用** STDM，**密集波分复用** DWDM 和**码分多址** CDMA。后两种复用将在下面的两小节中介绍。

频分复用和时分复用的特点分别如图 2-21(a)和(b)所示。频分复用最简单，用户在分配到一定的频带后，在通信过程中自始至终都占用这个频带。可见**频分复用的所有用户在同样的时间占用不同的带宽资源**（请注意，这里的“带宽”是频率带宽而不是数据的发送速率）。而时分复用则是将时间划分为一段段等长的时分复用帧(TDM 帧)。每一个时分复用的用户在每一个 TDM 帧中占用固定序号的时隙。为简单起见，在图 2-21(b)中只画出了 4 个用户 A, B, C 和 D。每一个用户所占用的时隙是周期性地出现（其周期就是 TDM 帧的长度）。因此 TDM 信号也称为**等时**(isochronous)信号。可以看出，**时分复用的所有用户是在不同的时间占用同样的频带宽度**。这两种复用方法的优点是技术比较成熟，但缺点是不够灵活。时分复用则更有利于数字信号的传输。

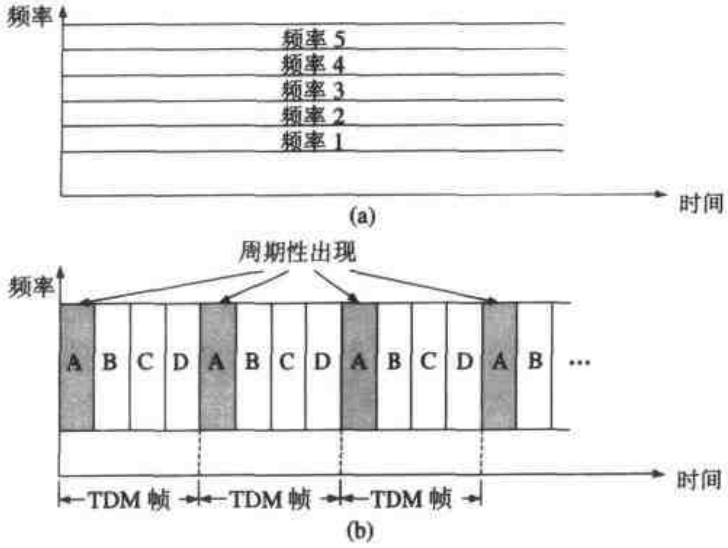


图 2-21 频分复用和时分复用

在使用**频分复用**时，若每一个用户占用的带宽不变，则当复用的用户数增加时，复用后的信道的总带宽就跟着变宽。例如，传统的电话通信每一个标准话路的带宽是 4 kHz (即通信用的 3.1 kHz 加上两边的保护频带)，那么若有 1000 个用户进行频分复用，则复用后的总带宽就是 4 MHz。但在使用时分复用时，每一个时分复用帧的长度是不变的，始终是 125 μs。若有 1 千个用户进行时分复用，则每一个用户分配到的时隙宽度就是 125 μs 的千分之一，即 0.125 μs，时隙宽度变得非常窄。我们应注意到，时隙宽度非常窄的脉冲信号所占的频谱范围也是非常宽的。

在进行通信时，**复用器**(multiplexer)总是和**分用器**(demultiplexer)成对地使用。在复用器

和分用器之间是用户共享的高速信道。分用器的作用正好和复用器的相反，它将高速线路传送过来的数据进行分用，分别送到相应的用户处。前面给出的图 2-20 中的传输线路左边的旋转开关实际上就是一个复用器，而右边的旋转开关实际上就是一个分用器。

当使用时分复用系统传送计算机数据时，由于计算机数据的突发性质，一个用户对已经分配到的的子信道的利用率一般是不高的。当用户在某一段时间暂时无数据传输时（例如用户正在键盘上输入数据或正在浏览屏幕上的信息），那就只能让已经分配到手的子信道空闲着，而其他用户也无法使用这个暂时空闲的线路资源。图 2-22 说明了这一概念。这里假定有 4 个用户 A、B、C 和 D 进行时分复用。复用器按①→②→③→④的顺序依次扫描用户 A、B、C 和 D 的各时隙，然后构成一个个时分复用帧。图中共画出了 4 个时分复用帧，每个时分复用帧有 4 个时隙。可以看出，当某用户暂时无数据发送时，在时分复用帧中分配给该用户的时隙只能是处于空闲状态，其他用户即使一直有数据要发送，也不能使用这些空闲的时隙。这就导致复用后的信道利用率不高。

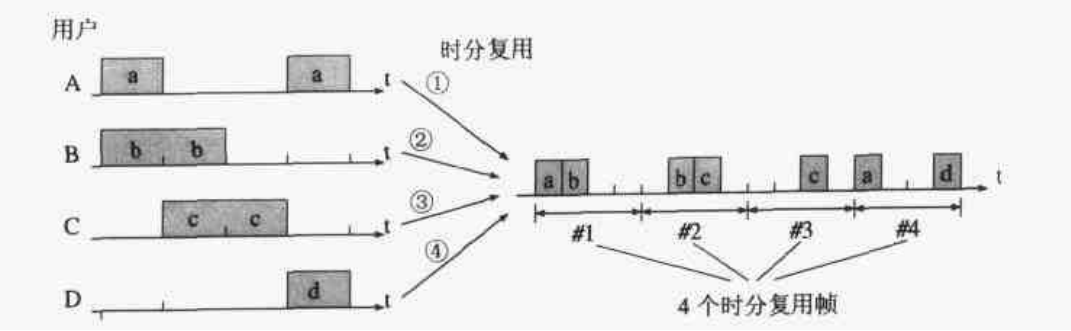


图 2-22　时分复用可能回造成线路资源的浪费

统计时分复用 STDM (Statistic TDM)是一种改进的时分复用，它能明显地提高信道的利用率。**集中器**(concentrator）常使用这种统计时分复用。图 2-23 是统计时分复用的原理图。一个使用统计时分复用的集中器连接 4 个低速用户，然后将它们的数据集中起来通过高速线路发送到一个远地计算机。

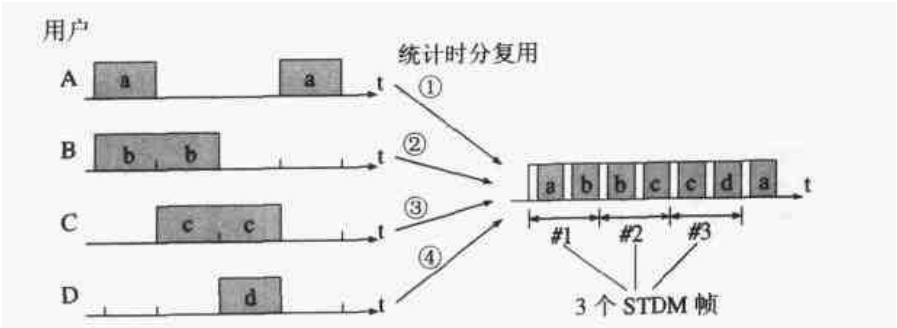


图 2-23　统计时分复用的工作原理

统计时分复用使用 STDM 帧来传送复用的数据。但每一个 STDM 帧中的时隙数小于连接在集中器上的用户数。各用户有了数据就随时发往集中器的输入缓存，然后集中器按顺序依次扫描输入缓存，将缓存中的输入数据放入 STDM 帧中。对没有数据的缓存就跳过去。当一个帧的数据放满了，就发送出去。因此，STDM 帧不是固定分配时隙，而是按需动态地分配时隙。因此统计时分复用可以提高线路的利用率。我们还可看出，在输出线路上，某一个

用户所占用的时隙并不是周期性地出现。因此统计复用又称为**异步时分复用**，而普通的时分复用称为**同步时分复用**。这里应注意的是，虽然统计时分复用的输出线路上的数据率小于各输入线路数据率的总和，但**从平均的角度来看，这二者是平衡的**。假定所有的用户都不间断地向集中器发送数据，那么集中器肯定无法应付，它内部设置的缓存都将溢出。所以集中器能够正常工作的前提是假定各用户都是间歇地工作。

由于 STDM 帧中的时隙并不是固定地分配给某个用户，因此在每个时隙中还必须有用户的地址信息，这是统计时分复用必须要有的和不可避免的一些开销。在图 2-23 输出线路上每个时隙之前的白色小时隙就是放入这样的地址信息。使用统计时分复用的集中器也叫做**智能复用器**，它能提供对整个报文的存储转发能力（但大多数复用器一次只能存储一个字符或一个比特），通过排队方式使各用户更合理地共享信道。此外，许多集中器还可能具有路由选择、数据压缩、前向纠错等功能。

最后要强调一下，TDM 帧和 STDM 帧都是在物理层传送的比特流中所划分的帧。这种“帧”和我们以后要讨论的数据链路层的“帧”是完全不同的概念，不可弄混。

2.5.2 波分复用

波分复用就是**光的频分复用**。光纤技术的应用使得数据的传输速率空前提高。目前一根单模光纤的传输速率可达到 2.5 Gb/s。再提高传输速率就比较困难了。如果设法对光纤传输中的色散(dispersion)问题[①]加以解决，如采用色散补偿技术，则一根单模光纤的传输速率可达到 10 Gb/s。这几乎已到了单个光载波信号传输的极限值。

但是，人们借用传统的载波电话的频分复用的概念，就能做到使用一根光纤来同时传输多个频率很接近的光载波信号。这样就使光纤的传输能力成倍地提高了。由于光载波的频率很高，因此习惯上用波长而不用频率来表示所使用的光载波。这样就得出了波分复用这一名词。最初，人们只能在一根光纤上复用两路光载波信号。这种复用方式称为**波分复用 WDM**。随着技术的发展，在一根光纤上复用的光载波信息路数越来越多。现在已能做到在一根光纤上复用 80 路或更多路数的光载波信号。于是就使用了**密集波分复用** DWDM (Dense Wavelength Division Multiplexing)这一名词。图 2-24 说明了波分复用的概念。

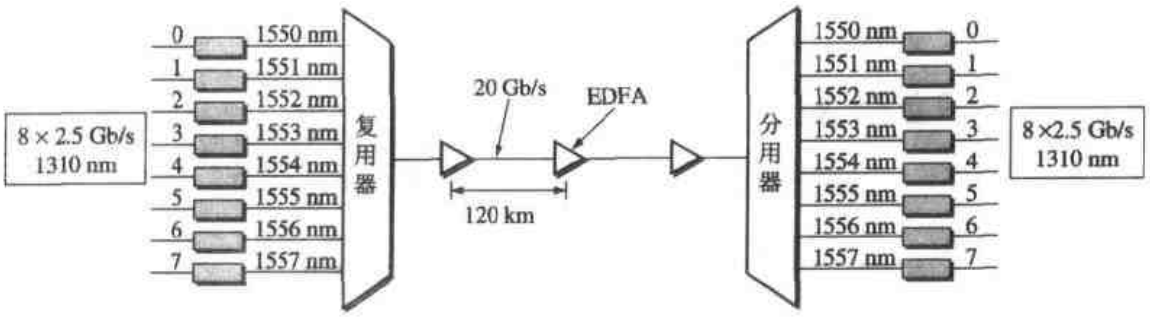


图 2-24 波分复用的概念

图 2-24 表示 8 路传输速率均为 2.5 Gb/s 的光载波（其波长均为 1310 nm[②]）。经光的调制

① 注：色散即光脉冲中不同频率的分量的传输速率不同，这导致信号的失真因而产生误码。当传输速率增高时，色散问题就越来越严重。例如，2.5 Gb/s 时的色散是 622 Mb/s 时的 16 倍。

② 注：单位 nm 是“纳米”，即 10^{-9} 米。1310 nm = 1.31 μm。

后，分别将波长变换到 1550～1557 nm，每个光载波相隔 1 nm (这里只是为了说明问题的方便。实际上，对于密集波分复用，光载波的间隔一般是 0.8 或 1.6 nm)。这 8 个波长很接近的光载波经过**光复用器**（波分复用的复用器又称为**合波器**）后，就在一根光纤中传输。因此，在一根光纤上数据传输的总速率就达到了 8×2.5 Gb/s = 20 Gb/s。但光信号传输了一段距离后就会衰减，因此对衰减了的光信号必须进行放大才能继续传输。现在已经有了很好的**掺铒光纤放大器** EDFA(Erbium Doped Fiber Amplifier)。它是一种光放大器，不需要像以前那样复杂，先把光信号转换成电信号，经过电放大器放大后，再转换成为光信号。**掺铒**光纤放大器 EDFA 不需要进行光电转换而直接对光信号进行放大，并且在 1550 nm 波长附近有 35 nm (即 4.2 THz) 频带范围提供较均匀的、最高可达 40～50 dB 的增益。两个光纤放大器之间的光缆线路长度可达 120 km，而光**复用器**和**光分用器**（波分复用的分用器又称为分**波器**）之间的无光电转换的距离可达 600 km（只需放入 4 个光纤放大器）。而在使用波分复用技术和光纤放大器之前，要在 600 km 的距离传输 20 Gb/s，需要铺设 8 根速率为 2.5 Gb/s 的光纤，而且每隔 35 km 要用一个再生中继器进行光电转换后的放大，并再转换为光信号（这样的中继器总共需要有 128 个之多）。

在地下铺设光缆是耗资很大的工程。因此现在人们总是一根光缆中放入尽可能多的光纤（例如，放入 100 根以上的光纤），然后对每一根光纤再使用密集波分复用技术。因此，对于具有 100 根速率为 2.5 Gb/s 的光纤，采用 16 倍的密集波分复用，得到的总数据率即达 4 Tb/s。这里的 T 为 10^{12}，中文名词是“太”，即“兆兆”。

2.5.3 码分复用

码分复用 CDM (Code Division Multiplexing)是另一种共享信道的方法。实际上，人们更常用的名词是**码分多址 CDMA** (Code Division Multiple Access)。每一个用户可以在同样的时间使用同样的频带进行通信。由于**各用户使用经过特殊挑选的不同码型，因此各用户之间不会造成干扰**。码分复用最初是用于军事通信，因为这种系统发送的信号**有很强的抗干扰能力，其频谱类似于白噪声，不易被敌人发现**。随着技术的进步，CDMA 设备的价格和体积都大幅度下降，因而现在已广泛使用在民用的移动通信中，特别是在无线局域网中。采用 CDMA 可提高通信的话音质量和数据传输的可靠性，减少干扰对通信的影响，增大通信系统的容量（是使用 GSM 的 4～5 倍[①]），降低手机的平均发射功率等等。下面简述其工作原理。

在 CDMA 中，每一个比特时间再划分为 m 个短的间隔，称为**码片**(chip)。通常 m 的值是 64 或 128。在下面的原理性说明中，为了画图简单起见，我们设 m 为 8。

使用 CDMA 的每一个站被指派一个惟一的 m bit **码片序列**(chip sequence)。一个站如果要发送比特 1，则发送它自己的 m bit 码片序列。如果要发送比特 0，则发送该码片序列的二进制反码。例如，指派给 S 站的 8 bit 码片序列是 00011011。当 S 发送比特 1 时，它就发送序列 00011011，而当 S 发送比特 0 时，就发送 11100100。为了方便，我们按惯例将码片中的 0 写为–1，将 1 写为+1。因此 S 站的码片序列是(–1 –1 –1 +1 +1 –1 +1 +1)。

现假定 S 站要发送信息的数据率为 b b/s。由于每一个比特要转换成 m 个比特的码片，因此 S 站实际上发送的数据率提高到 mb b/s，同时 S 站所占用的频带宽度也提高到原来数值的

① 注：GSM (Group Special Mobile)即群组专用移动通信体制，是欧洲和我国现在广泛使用的移动通信体制。

m 倍。这种通信方式是**扩频**(spread spectrum)通信中的一种。扩频通信通常有两大类。一种是**直接序列**(direct sequence)，如上面讲的使用码片序列就是这一类，记为 DS-CDMA。另一种是**跳频**(frequency hopping)，记为 FH-CDMA。

CDMA 系统的一个重要特点就是这种体制给每一个站分配的码片序列不仅必须各不相同，并且还必须互相正交(orthogonal)。在实用的系统中是使用**伪随机码序列**。

用数学公式可以很清楚地表示码片序列的这种正交关系。令向量 **S** 表示站 S 的码片向量，再令 **T** 表示其他任何站的码片向量。两个不同站的码片序列正交，就是向量 **S** 和 **T** 的规格化内积(inner product)都是 0：

$$\mathbf{S} \cdot \mathbf{T} \equiv \frac{1}{m}\sum_{i=1}^{m} S_i T_i = 0 \qquad (2\text{-}4)$$

例如，向量 **S** 为(–1 –1 –1 +1 +1 –1 +1 +1)，同时设向量 **T** 为(–1 –1 +1 –1 +1 +1 +1 –1)，这相当于 T 站的码片序列为 00101110。将向量 **S** 和 **T** 的各分量值代入（2-4）式就可看出这两个码片序列是正交的。不仅如此，向量 **S** 和各站码片反码的向量的内积也是 0。另外一点也很重要，即任何一个码片向量和该码片向量自己的规格化内积都是 1：

$$\mathbf{S} \cdot \mathbf{S} = \frac{1}{m}\sum_{i=1}^{m} S_i S_i = \frac{1}{m}\sum_{i=1}^{m} S_i^2 = \frac{1}{m}\sum_{i=1}^{m} (\pm 1)^2 = 1 \qquad (2\text{-}5)$$

而一个码片向量和该码片反码的向量的规格化内积值是 -1。这从（2-5）式可以很清楚地看出，因为求和的各项都变成了 -1

现在假定在一个 CDMA 系统中有很多站都在相互通信，每一个站所发送的是数据比特和本站的码片序列的乘积，因而是本站的码片序列（相当于发送比特 1）和该码片序列的二进制反码（相当于发送比特 0）的组合序列，或什么也不发送（相当于没有数据发送）。我们还假定所有的站所发送的码片序列都是同步的，即所有的码片序列都在同一个时刻开始。利用**全球定位系统** GPS 就不难做到这点。

现假定有一个 X 站要接收 S 站发送的数据。X 站就必须知道 S 站所特有的码片序列。X 站使用它得到的码片向量 **S** 与接收到的未知信号进行求内积的运算。X 站接收到的信号是各个站发送的码片序列之和。根据上面的公式(2-4)和(2-5)，再根据叠加原理（假定各种信号经过信道到达接收端是叠加的关系），那么求内积得到的结果是：所有其他站的信号都被过滤掉（其内积的相关项都是 0），而只剩下 S 站发送的信号。当 S 站发送比特 1 时，在 X 站计算内积的结果是 +1，当 S 站发送比特 0 时，内积的结果是 –1。

图 2-25 是 CDMA 的工作原理。设 S 站要发送的数据是 1 1 0 三个码元。再设 CDMA 将每一个码元扩展为 8 个码片，而 S 站选择的码片序列为(–1 –1 –1 +1 +1 –1 +1 +1)。S 站发送的扩频信号为 $\mathbf{S}_x$。我们应当注意到，S 站发送的扩频信号 $\mathbf{S}_x$ 中，只包含互为反码的两种码片序列。T 站选择的码片序列为(–1 –1 +1 –1 +1 +1 +1 –1)，T 站也发送 1 1 0 三个码元，而 T 站的扩频信号为 $\mathbf{T}_x$。因所有的站都使用相同的频率，因此每一个站都能够收到所有的站发送的扩频信号。对于我们的例子，所有的站收到的都是叠加的信号 $\mathbf{S}_x + \mathbf{T}_x$。

当接收站打算收 S 站发送的信号时，就用 S 站的码片序列与收到的信号求规格化内积。这相当于分别计算 $\mathbf{S} \cdot \mathbf{S}_x$ 和 $\mathbf{S} \cdot \mathbf{T}_x$，然后再求它们的和。显然，后者是零，而前者就是 S 站发送的数据比特。

关于 CDMA 更进一步的内容可参阅[VITE95]。

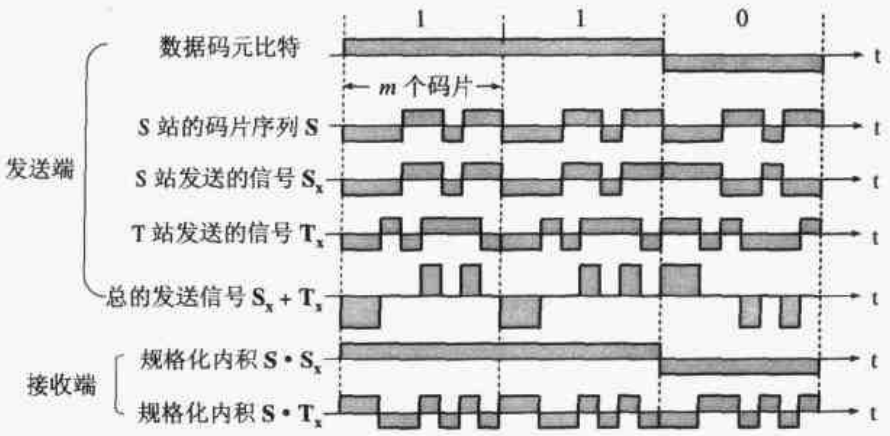


图 2-25　CDMA 的工作原理

2.6　同步光纤网 SONET 和同步数字系列 SDH

在前面 2.4.3 节所介绍的数字传输系统存在着许多缺点。其中最主要的是以下两个方面：

（1）**速率标准不统一**。PCM 的一次群数字传输速率有两个国际标准，一个是北美和日本的 T1 速率，而另一个是欧洲的 E1 速率。但是到了高次群日本又搞了第三种不兼容的标准。如果不对高次群的数字传输速率进行标准化，国际范围的高速数据传输就很难实现，因为高次群的数字传输速率的转换十分困难。然而高次群的数字传输速率各国都已使用了不少时间，谁都不愿意抛弃正在使用的大量设备并改用别人的数字传输速率标准。

（2）**不是同步传输**。前面已经讲过，在过去相当长的时间，为了节约经费，各国的数字网主要是采用**准同步**方式。这时，必须采用复杂的脉冲填充方法才能补偿由于频率不准确而造成的定时误差。这就给数字信号的复用和分用带来许多麻烦。当数据传输的速率较低时，收发双方时钟频率的微小差异并不会带来严重的不良影响。但是当数据传输的速率不断提高时，这个收发双方时钟同步的问题就成为迫切需要加以解决的问题。

为了解决上述问题，美国在 1988 年首先推出了一个数字传输标准，叫做**同步光纤网** SONET (Synchronous Optical Network)。整个的同步网络的各级时钟都来自一个非常精确的主时钟（通常采用昂贵的铯原子钟，其精度优于$\pm 1 \times 10^{-11}$）。SONET 为光纤传输系统定义了同步传输的线路速率等级结构，其传输速率以 51.84 Mb/s 为基础[①]，大约对应于 T3/E3 的传输速率，此速率对电信号称为第 1 级**同步传送信号**(Synchronous Transport Signal)，即 STS-1；对光信号则称为第 1 级**光载波**(Optical Carrier)，即 OC-1。现已定义了从 51.84 Mb/s(即 OC-1)一直到 9953.280 Mb/s(即 OC-192/STS-192)的标准。

ITU-T 以美国标准 SONET 为基础，制订出国际标准**同步数字系列** SDH (Synchronous Digital Hierarchy)，即 1988 年通过的 G.707～G.709 等三个建议书。到 1992 年又增加了十几个建议书。一般可认为 SDH 与 SONET 是同义词，但其主要不同点是：SDH 的基本速率

① 注：SONET 规定，SONET 每秒传送 8000 帧（和 PCM 的采样速率一样）。每个 STS-1 帧长为 810 字节，因此 STS-1 的数据率为 $8000 \times 810 \times 8 = 51\,840\,000$ b/s。为了便于表示，通常将一个 STS-1 帧画成 9 行 90 列的字节排列。在这种排列中的每一个字节对应的数据率是 64 kb/s。一个 STS-n 的帧长就是 STS-1 的帧长的 n 倍，也同样是每秒传送 8000 帧，因此 STS-n 的数据率就是 STS-1 的数据率的 n 倍。

为 155.52 Mb/s，称为**第 1 级同步传递模块**(Synchronous Transfer Module)，即 STM-1，相当于 SONET 体系中的 OC-3 速率。表 2-4 为 SONET 和 SDH 的比较。表中带有星号*的 3 种速率是现在最常用的。为方便起见，在谈到 SONET/SDH 的常用速率时，往往不使用速率的精确数值而是使用表中最后一列给出的近似值。

表 2-4　SONET 的 OC 级/STS 级与 SDH 的 STM 级的对应关系

线路速率 (Mb/s)	SONET 符号	ITU-T 符号	表示线路速率的常用近似值
51.840	OC-1/STS-1	—	
155.520*	OC-3/STS-3	STM-1	155 Mb/s(按四舍五入则应为 156 Mb/s)
466.560	OC-9/STS-9	STM-3	
622.080*	OC-12/STS-12	STM-4	622 Mb/s
933.120	OC-18/STS-18	STM-6	
1 244.160	OC-24/STS-24	STM-8	
1 866.240	OC-36/STS-36	STM-12	
2 488.320*	OC-48/STS-48	STM-16	2.5 Gb/s
4 876.640	OC-96/STS-96	STM-32	
9 953.280	OC-192/STS-192	STM-64	10 Gb/s

SDH/SONET 定义了标准光信号，规定了波长为 1310 nm 和 1550 nm 的激光源；在物理层为宽带接口使用了帧技术以传递信息；为数字信号的复用和操作过程定义了帧结构。

SONET 标准定义了四个光接口层。这虽然在概念上有点像 OSI 参考模型，但 SONET 自身只对应于 OSI 的物理层。SONET 的层次自下而上为（图 2-26）：

- **光子层**(Photonic Layer)　处理跨越光缆的比特传送，并负责进行同步传送信号 STS 的电信号和光载波 OC 的光信号之间的转换。在此层由电光转换器进行通信。
- **段层**(Section Layer)　在光缆上传送 STS-N 帧，有成帧和差错检测功能。

上述两层是必须要有的，但下面两层是可供选择的。

- **线路层**(Line Layer)　负责路径层的同步和复用，以及交换的自动保护。
- **路径层**(Path Layer)　处理**路径端接设备** PTE (Path Terminating Element)之间的业务的传输，这里 PTE 是具有 SONET 能力的交换机。路径层还具有与非 SONET 网络的接口。

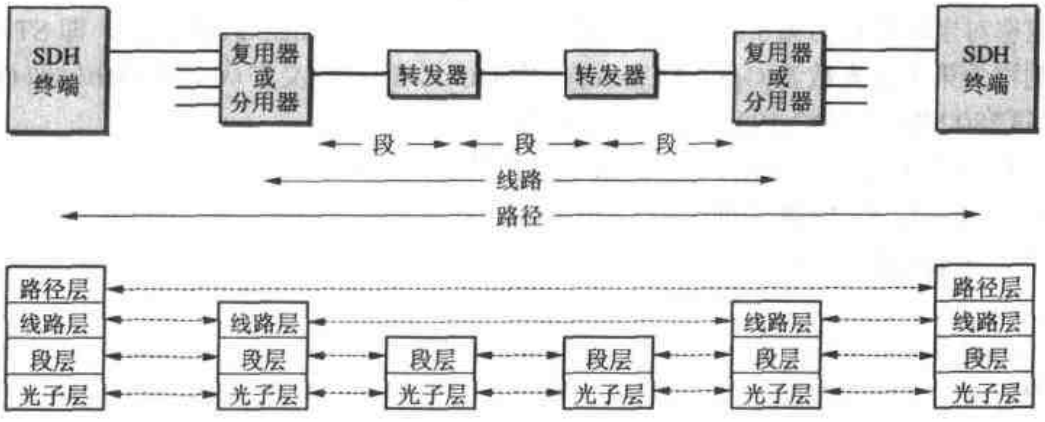


图 2-26　SONET 的体系结构

SDH 的帧结构是一种块状帧，其基本信号是 STM-1，更高的等级是用 N 个 STM-1 复用组成 STM-N。如 4 个 STM-1 构成 STM-4，16 个 STM-1 构成 STM-16。SDH 简化了复用和分用技术，需要时可直接接入到低速支路，而不经过高速到低速的逐级分用，上下电路方便。SDH 采用自愈混合环形网结构，并与**数字交接系统** DACS (Digital Access and Cross-connect System)结合使用，可使网络按预定方式重新组配，避免了耗资的人工操作，因而大大提高了通信网的灵活性和可靠性。光纤信道的带宽充裕，因此 SDH 可在其帧结构中使用较多的比特用于管理，这就大大增强了通信网的运行、维护、监控和管理功能。

SDH/SONET 标准的制订，使北美、日本和欧洲这三个地区三种不同的数字传输体制在 STM-1 等级上获得了统一。各国都同意将这一速率以及在此基础上的更高的数字传输速率作为国际标准。这是第一次真正实现了数字传输体制上的世界性标准。现在 SDH/SONET 标准已成为公认的新一代理想的传输网体制，因而对世界电信网络的发展具有重大的意义。SDH 标准也适合于微波和卫星传输的技术体制[COMM90]。

2.7 物理层标准举例

本节我们简单介绍两种最常用的物理层标准。ITU-T 的 X.21 虽也是物理层的著名标准，但由于使用得较少，这里从略。

2.7.1 EIA-232-E 接口标准

本节主要介绍 EIA-232-E 标准，同时也说明 RS-449 的特点。

EIA-232-E 是美国电子工业协会 EIA 制订的著名物理层异步通信接口标准。它最早是 1962 年制订的标准 RS-232。这里 RS 表示 EIA 的一种“推荐标准”，232 是个编号。在 1969 年修订为 RS-232-C，C 是标准 RS-232 以后的第三个修订版本。1987 年 1 月，修订为 EIA-232-D。1991 年又修订为 EIA-232-E。由于标准修改得并不多，因此现在很多厂商仍用旧的名称。有时简称为 EIA-232，甚至说得更简单些：“提供 232 接口”。

EIA-232 是 DTE 与 DCE 之间的接口标准。因此下面先介绍什么是 DTE 和 DCE。

DTE (Data Terminal Equipment)是**数据终端设备**，也就是具有一定的数据处理能力以及发送和接收数据能力的设备。大家知道，大多数的数字数据处理设备的数据传输能力是很有限的。直接将相隔很远的两个数据处理设备连接起来，是不能进行通信的。必须在数据处理设备和传输线路之间，加上一个中间设备。这个中间设备就是**数据电路端接设备** DCE (Data Circuit-terminating Equipment)。DCE 的作用就是在 DTE 和传输线路之间提供信号变换和编码的功能，并且负责建立、保持和释放数据链路的连接。图 2-27 画出了 DTE 通过 DCE 才连接到通信传输线路上。

DTE 可以是一台计算机或一个终端，也可以是各种的 I/O 设备。典型的 DCE 则是一个与模拟电话线路相连接的调制解调器。我们从图 2-27 可以看到，DCE 虽然处于通信环境内，但它和 DTE 均属于用户设施。用户环境只包括 DTE。

DTE 与 DCE 之间的接口一般都有许多条并行线，包括多种信号线和控制线。DCE 将 DTE 传过来的数据，按比特顺序逐个发往传输线路，或者反过来，从传输线路收下来串行的比特流，然后再交给 DTE。很明显，这里需要高度协调地工作。为了减轻数据处理设备用户的负担，就必须对 DTE 和 DCE 的接口进行标准化。这种接口标准也就是所谓的物理层协议。

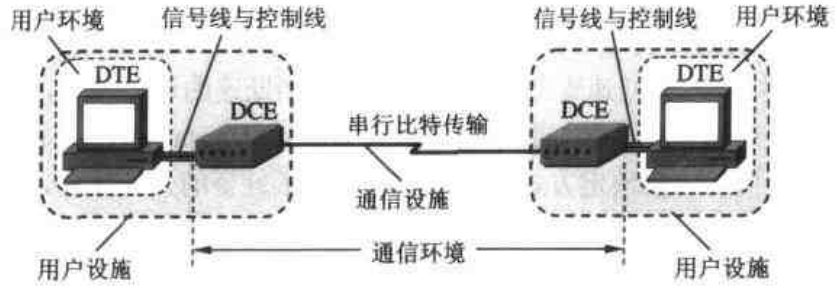


图 2-27　DTE 通过 DCE 与通信传输线路相连

多数的物理层协议使用如图 2-27 所示的模型。但也有一些不是这样。例如，在局域网中，物理层协议所定义的是一个数据终端设备和链路的传输媒体的接口，而并没有使用这种 DTE/DCE 模型。有一种常用的 DCE 类型叫做 CSU/DSU，意思是**信道服务单元/数据服务单元**(Channel Service Unit/Data Service Unit)，它将 DTE/DCE 接口转换为通常的电话接口。

下面扼要介绍一下物理层标准 EIA-232 的一些主要特点。

在机械特性方面，EIA-232 使用 ISO 2110 关于插头座的标准。这就是使用 25 根引脚的 DB-25 插头座。引脚分为上、下两排，分别有 13 和 12 根引脚，其编号分别规定为 1 至 13 和 14 至 25，都是从左到右（当引脚指向人时）。

在电气性能方面，EIA-232 与 CCITT 的 V.28 建议书一致。这里要提醒读者注意的是：EIA-232 采用负逻辑。也就是说，逻辑 0 相当于对信号地线有+3 V 或更高的电压，而逻辑 1 相当于对信号地线有−3 V 或更负的电压。逻辑 0 相当于数据的“0”（空号）或控制线的“接通”状态，而逻辑 1 则相当于数据的“1”（传号）或控制线的“断开”状态。当连接电缆线的长度不超过 15 m 时，允许数据传输速率不超过 20 kb/s。但是当连接电缆长度较短时，数据传输速率就可以大大提高。

EIA-232 的功能特性与 CCITT 的 V.24 建议书一致。它规定了什么电路应当连接到 25 根引脚中的哪一根以及该引脚的作用。图 2-28 画的是最常用的 10 根引脚的作用，括弧中的数目为引脚的编号。其余的一些引脚可以空着不用。图中引脚 7 是信号地，即公共回线。引脚 1 是保护地（即屏蔽地），有时可不用。引脚 2 和引脚 3 都是传送数据的数据线。“发送”和“接收”都是对 DTE 而言。有时只用图中的 9 个引脚（将“保护地”除外）制成专用的 9 芯插头，供计算机与调制解调器的连接使用。

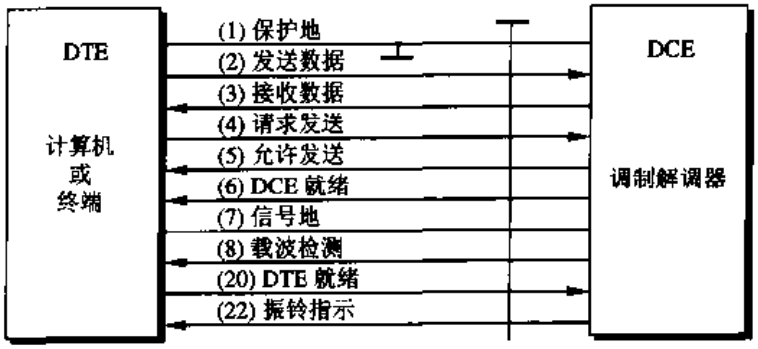


图 2-28　EIA-232/V.24 的信号定义

EIA-232 的规程特性规定了在 DTE 与 DCE 之间所发生的事件的合法序列。这部分内容与 CCITT 的 V.24 建议书一致。

下面通过图 2-29 的例子，说明 DTE-A 要向 DTE-B 发送数据所要经过的几个主要步骤。

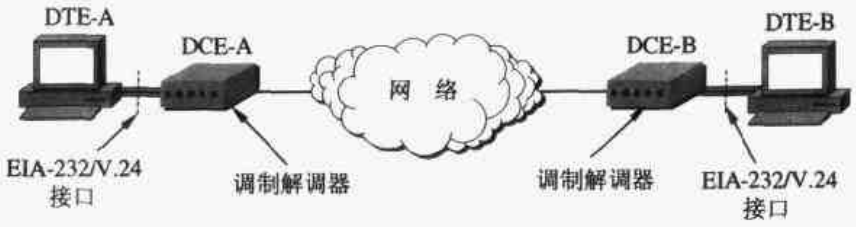


图 2-29 两个 DTE 通过 DCE 进行通信的例子

（1）当 DTE-A 要和 DTE-B 进行通信时，就将引脚 20“DTE 就绪”置为 ON，同时通过引脚 2“发送数据”向 DCE-A 传送电话号码信号。

（2）DCE-B 将引脚 22“振铃指示”置为 ON，表示通知 DTE-B 有入呼叫信号到达（在振铃的间隙以及其他时间，振铃指示均为 OFF 状态）。DTE-B 就将其引脚 20“DTE 就绪”置为 ON。DCE-B 接着产生载波信号，并将引脚 6“DCE 就绪”置为 ON，表示已准备好接收数据。

（3）当 DCE-A 检测到载波信号时，将引脚 8“载波检测”和引脚 6“DCE 就绪”都置为 ON，以便使 DTE-A 知道通信电路已经建立。DCE-A 还可通过引脚 3“接收数据”向 DTE-A 发送在其屏幕上显示的信息。

（4）DCE-A 接着向 DCE-B 发送其载波信号，DCE-B 将其引脚 8“载波检测”置为 ON。

（5）当 DTE-A 要发送数据时，将其引脚 4“请求发送”置为 ON。DCE-A 作为响应将引脚 5“允许发送”置为 ON。然后 DTE-A 通过引脚 2“发送数据”来发送其数据。DCE-A 将数字信号转换为模拟信号向 DCE-B 发送过去。

（6）DCE-B 将收到的模拟信号转换为数字信号经过引脚 3“接收数据”向 DTE-B 发送。

其他的一些引脚的作用是：选择数据的发送速率，测试调制解调器，传送数据的码元定时信号，以及从另一个辅助信道反向发送数据等。但是这些引脚在实际中却很少使用。

许多产品都声称自己的串行接口是“与 EIA-232 标准兼容”。读者应当注意：这只是说，该接口的电气特性和机械特性与 EIA-232 接口标准没有矛盾。但我们仍无法得知该接口是否能够支持 EIA-232 的全部功能。这是因为，很多厂商出售的调制解调器只使用了接口的 25 根引脚中的 4～12 根。因此他们所实现的很可能只是整个 EIA-232 标准的一个子集。因此应弄清你所需要的性能是否已包括在这个子集之中。

EIA 还规定了插头应装在 DTE 上，插座应装在 DCE。因此当终端或计算机与调制解调器相连时就非常方便。然而有时却需要将两台计算机通过 EIA-232 串行接口直接相连。这显然有点麻烦。例如，这台计算机通过引脚 2 发送数据，但仍然传送到另一台计算机的引脚 2，这就使对方无法接收。为了不改动计算机内标准的串行接口线路，可以采用虚调制解调器的方法。所谓虚调制解调器就是一段电缆，具体的连接方法如图 2-30 所示。这样对每一台计算机来说，都好像是与一个调制解调器相连，但实际上并没有真正的调制解调器存在。

2.7.2 RS-449 接口标准

EIA-232 接口标准有两个较大的弱点，即：

（1）数据的传输速率最高为 20 kb/s；

（2）连接电缆的最大长度不超过 15 m。

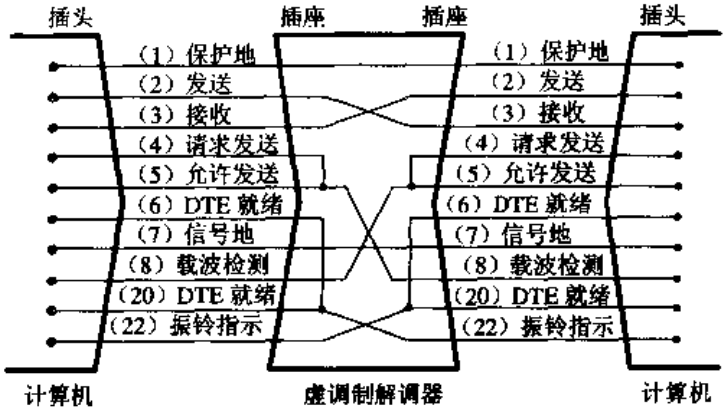


图 2-30　利用调制解调器与两台计算机相连

这就促使人们制订性能更好的接口标准。出于这种考虑，EIA 于 1977 年又制定了一个新的标准 RS-449，以便逐渐取代旧的 RS-232。

实际上，RS-449 由 3 个标准组成。即：

（1）RS-449　规定接口的机械特性、功能特性和过程特性。RS-449 采用 37 根引脚的插头座。在 CCITT 的建议书中，RS-449 相当于 V.35。

（2）RS-423-A　规定在采用非平衡传输时（即所有的电路共用一个公共地）的电气特性。当连接电缆长度为 10 m 时，数据的传输速率可达 300 kb/s。

（3）RS-422-A　规定在采用平衡传输时（即所有的电路没有公共地）的电气特性。它可将传输速率提高到 2 Mb/s，而连接电缆长度可超过 60 m。当连接电缆长度更短时（如 10 m），则传输速率还可以更高些（如达到 10 Mb/s）。

通常 EIA-232/V.24 用于标准电话线路（一个话路）的物理层接口，而 RS-449/V.35 则用于宽带电路（一般都是租用电路），其典型的传输速率为 48～168 kb/s，都是用于点到点的同步传输。

关于 RS-449 的详细说明可参阅有关标准，这里从略。

以上所讲的 EIA-232 和 RS-449 标准只是 ITU-T 为在模拟电话网上传送数据的接口标准系列中的一部分。全面详细的标准都由 ITU-T 的 V 系列建议书给出。

习题

2-01　物理层要解决哪些问题？物理层的主要特点是什么？

2-02　试给出数据通信系统的模型并说明其主要组成构件的作用。

2-03　试解释以下名词：数据，信号，模拟数据，模拟信号，数字数据，数字信号，单工通信，半双工通信，全双工通信。

2-04　物理层的接口有哪几个方面的特性？各包含些什么内容？

2-05　奈氏准则与香农公式在数据通信中的意义是什么？比特和波特有何区别？

2-06　用香农公式计算一下，假定信道带宽为 3100 Hz，最大信息传输速率为 35 kb/s，那么若想使最大信息传输速率增加 60%，问信噪比 *S/N* 应增大到多少倍？如果在刚才计算出的基础上将信噪比 *S/N* 再增大到 10 倍，问最大信息速率能否再增加 20%？

2-07　常用的传输媒体有哪几种？各有何特点？

2-08 什么是曼彻斯特编码和差分曼彻斯特编码？其特点如何？

2-09 模拟传输系统与数字传输系统的主要特点是什么？

2-10 EIA-232 和 RS-449 接口标准各用在什么场合？

2-11 基带信号与宽带信号的传输各有什么特点？

2-12 有 600 MB(兆字节)的数据，需要从南京传送到北京。一种方法是将数据写到磁盘上，然后托人乘火车将这些磁盘捎去。另一种方法是用计算机通过长途电话线路（设信息传送的速率是 2.4 kb/s）传送此数据。试比较这两种方法的优劣。

若信息传送速率为 33.6 kb/s，其结果又如何？

2-13 56 kb/s 的调制解调器是否已突破了香农的信道极限传输速率？这种调制解调器的使用条件是怎样的？

2-14 在 2.3.1 节介绍双绞线时，我们说："在数字传输时，若传输速率为每秒几个兆比特，则传输距离可达几公里。"但目前我们使用调制解调器与 ISP 相连时，数据的传输速率最高只能达到 56 kb/s，与每秒几个兆比特相距甚远。这是为什么？

2-15 试写出下列英文缩写的全文，并进行简单的解释。

FDM，TDM，STDM，WDM，DWDM，CDMA，SONET，SDH，STM-1，OC-48，DTE，DCE，EIA，ITU-T，CCITT，ISO。

2-16 码分多址 CDMA 为什么可以使所有用户在同样的时间使用同样的频带进行通信而不会互相干扰？这种复用方法有何优缺点？

2-17 共有 4 个站进行码分多址 CDMA 通信。4 个站的码片序列为：

A: (–1 –1 –1 +1 +1 –1 +1 +1)　　B: (–1 –1 +1 –1 +1 +1 +1 –1)

C: (–1 +1 –1 +1 +1 +1 –1 –1)　　D: (–1 +1 –1 –1 –1 –1 +1 –1)

现收到这样的码片序列：(–1 +1 –3 +1 –1 –3 +1 +1)。问哪个站发送数据了？发送数据的站发送的 1 还是 0？

2-18 假定在进行异步通信时，发送端每发送一个字符就要发送 10 个等宽的比特（一个起始比特，8 个比特的 ASCII 码字符，最后一个结束比特）。试问当接收端的时钟频率和发送端的时钟频率相差 5%时，双方能否正常通信？

第 3 章　数据链路层

数据链路层的许多概念都是计算机网络的重要概念。本章在介绍数据链路层的基本概念后，就详细地讨论两个重要的协议：停止等待协议和连续 ARQ 协议，包括滑动窗口的概念。接着阐明面向比特的链路控制规程 HDLC 的要点。最后介绍因特网中的数据链路层协议 PPP。

3.1　数据链路层的基本概念

我们已多次使用过“链路”和“数据链路”这两个术语。这里要强调一下，“链路”和“数据链路”并不是一回事。所谓**链路**(link)就是一条无源的点到点的物理线路段，中间没有任何其他的交换结点。在进行数据通信时，两个计算机之间的通路往往是由许多的链路串接而成的。可见一条链路只是一条通路的一个组成部分。**数据链路**(data link)则是另一个概念。这是因为当需要在一条线路上传送数据时，除了必须有一条物理线路外，还必须有一些必要的通信协议来控制这些数据的传输（这将在后面几节讨论）。若把实现这些协议的硬件和软件加到链路上，就构成了数据链路。现在最常用的方法是使用**适配器**（在许多情况下适配器就是**网卡**）来实现这些协议的硬件和软件。一般的适配器都包括了数据链路层和物理层这两层的功能。

在讨论数据链路层的功能时，常常在两个对等的数据链路层之间画出一个数字管道，而**在这条数字管道上传输的数据单位是帧**。虽然我们知道在物理层之间传送的是比特流，而在物理传输媒体上传送的是信号（电信号或光信号），但有时为了方便我们也常说，“在某条链路（而没有说数据链路）上传送数据帧”。其实这已经隐含地假定了我们是在数据链路层上来观察问题。如果没有数据链路层的协议，我们在物理层上就只能看见链路上传送的比特串，根本不能找出一个帧的起止比特，当然更无法识别帧的结构。有时我们也会不太严格地说，“在某条链路上传送分组或比特流”，但这显然是在网络层或物理层上讨论问题。

也有人采用另外的术语。这就是将链路分为物理链路和逻辑链路。物理链路就是上面所说的链路，而逻辑链路就是上面的数据链路，是物理链路加上必要的通信协议。这两种划分方法实质上是一样的。

早期的数据通信协议曾叫作通信**规程**(procedure)。因此在数据链路层，规程和协议是同义语。

数据链路层的主要功能如下（有些协议可以省略其中的一些）：

（1）**链路管理**　当网络中的两个结点要进行通信时，数据的发方必须确知收方是否已经处在准备接收的状态。为此，通信的双方必须先要交换一些必要的信息。或者用我们的术语，必须先建立一条数据链路。同样地，在传输数据时要维持数据链路，而在通信完毕时要释放数据链路。数据链路的建立、维持和释放就叫做**链路管理**。

（2）**帧定界**　在数据链路层，数据的传送单位是帧。数据一帧一帧地传送，就可以在出现差错时，将有差错的帧再重传一次，而避免了将全部数据都进行重传。帧定界是指收方应当能从收到的比特流中准确地区分出一帧的开始和结束在什么地方。帧定界也可称为

帧同步。

（3）**流量控制**　发方发送数据的速率必须使收方来得及接收。当收方来不及接收时，就必须及时控制发方发送数据的速率。这种功能称作流量控制(flow control)。

（4）**差错控制**　在计算机通信中，一般都要求有极低的比特差错率。为此，广泛地采用了编码技术。编码技术有两大类。一类是**前向纠错**，即收方收到有差错的数据帧时，能够自动将差错改正过来。这种方法的开销较大，不大适合于计算机通信。另一类是**差错检测**，即收方可以检测出收到的帧有差错（但并不知道是哪几个比特错了）。当检测出有差错的帧时就立即将它丢弃，但接下去有两种选择：一种方法不进行任何处理（要处理也是由高层进行），另一种方法则是由数据链路层负责重传丢弃的帧。这两种方法都是很常用的。

（5）**将数据和控制信息区分开**　在许多情况下，数据和控制信息处于同一帧中。因此一定要有相应的措施使收方能够将它们区分开来。

（6）**透明传输**　所谓透明传输就是不管所传数据是什么样的比特组合，都应当能够在链路上传送。当所传数据中的比特组合恰巧出现了与某一个控制信息完全一样时，必须有可靠的措施，使收方不会将这种比特组合的数据误认为是某种控制信息。只要能做到这点，数据链路层的传输就被称为是透明的。

（7）**寻址**　必须保证每一帧都能送到正确的目的站。收方也应知道发方是哪个站。

当 OSI 确定了应当有一个数据链路层后，又出现了许多种采用不同媒体接入控制的局域网。这些局域网无法使用一种统一的数据链路层协议。于是局域网的数据链路层就分解为两个子层。本章实际上是讨论最基本的数据链路层协议，其基本概念对学习全课程都很重要。至于更为复杂的局域网的数据链路层协议将在第 4 章进行讨论。下面先讨论最简单的停止等待协议。

3.2　停止等待协议

停止等待(stop-and-wait)协议是最简单但也是最基本的数据链路层协议。很多有关协议的基本概念都可以从这个协议中学习到。我们先从最简单的情况讲起。

3.2.1　完全理想化的数据传输

当两个主机进行通信时，应用进程要将数据从应用层逐层往下传，经物理层到达通信线路。通信线路将数据传到远端主机的物理层后，再逐层向上传，最后由应用层交给远程的应用进程。但现在为了把主要精力放在数据链路层的协议上，可以采用一个简化的模型（图 3-1），即把数据链路层以上的各层用一个主机来代替，而物理层和通信线路则等效成一条简单的数据链路。

在发方和收方的数据链路层分别有一个发送缓存和接收缓存。若进行全双工通信，则在每一方都要同时设有发送缓存和接收缓存。缓存就是一个存储空间，它是必不可少的。这是因为在通信线路上数据是以比特流的形式串行传输的，但在计算机内部数据的传输则是以字节（或若干个字节）为单位并行传输的。计算机在发送数据时，先以并行方式将数据写入发送缓存，然后以串行方式从发送缓存中按顺序读出比特，发送到通信线路上。在接收数据时，计算机先从通信线路上将串行传输的比特流按顺序存入接收缓存，然后再以并行方式按字节（或若干个字节）将数据从接收缓存读出。

图 3-1 所示的简化模型对于一个计算机网络中任意一条数据链路上的数据传输情况都是适用的。在网络内部，各交换结点的数据链路层的上面只有一个网络层。对于这种交换结点，网络层就相当于简化模型中的主机。

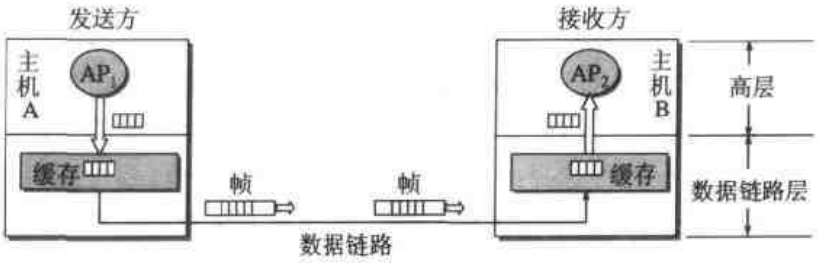


图 3-1 两台计算机通过一条数据链路进行通信的简化模型

为了深入理解数据链路层的协议，我们先从一种假想的、完全理想化的数据传输过程开始讨论。为了和后面的讨论相衔接，我们假定数据传输是以帧为单位。

所谓**完全理想化的数据传输**就是基于以下两个假定：

假定 1：链路是理想的传输信道，所传送的任何数据既不会出差错也不会丢失。

假定 2：不管发方以多快的速率发送数据，收方总是来得及收下，并及时上交主机。

第一个假定很容易理解。对第二个假定则需加以解释。

我们假设主机 A 连续不断地向主机 B 发送数据。在收方，主机 B 的数据链路层也就将收到的数据逐帧交给主机 B。在理想情况下，收方数据链路层的缓存每存满一帧就向主机 B 交付一帧。如果没有专门的流量控制协议，则收方并没有办法控制发方的发送速率，而收方也很难做到：向主机交付数据的速率永远不会低于发方发送数据的速率。若收方数据链路层向主机交付数据的速率略低于发方发送数据的速率，则收方的缓存中暂时存放的数据帧就会逐渐堆积起来，最后造成缓存溢出和数据帧丢失。因此，上述第二个假定就相当于认为：**接收端向主机交付数据的速率永远不会低于发送端发送数据的速率。**

在这样理想化的条件下，数据的传输就非常简单（不需要有流量控制，也不需要有差错控制）。但下面我们要逐步去掉这些完全理想化的假定。

3.2.2 具有最简单流量控制的数据链路层协议

现在去掉上述的第二个假定。但是，仍然保留第一个假定，即主机 A 向主机 B 传输数据的信道仍然是无差错的理想信道。然而现在不能保证接收端向主机交付数据的速率永远不低于发送端发送数据的速率。

为了使收方的接收缓存在任何情况下都不会溢出，在最简单的情况下，就是发方每发送一帧就暂时停下来，等待收方的确接收完毕后再发送下一帧。收方收到数据帧后就交付给主机，然后发一信息给发方，表示接收的任务已经完成。这时，发方才再发送下一个数据帧。在这种情况下，收方的接收缓存的大小只要能够装得下一个数据帧即可。显然，用这样的方法收发双方能够同步得很好，发方发送的数据流受收方的控制。这里应强调一下，**由收方控制发方的数据流，乃是计算机网络中流量控制的一个基本方法。**

现将以上具有最简单流量控制的数据链路层协议写成算法如下：

假定：链路是理想的传输信道，即所传送的任何数据既不会出差错也不会丢失。

在发送结点：

（1）从主机取一个数据帧。

（2）将数据帧送到数据链路层的发送缓存。

（3）将发送缓存中的数据帧发送出去。

（4）等待。

（5）若收到由接收结点发过来的信息（此信息的格式与内容可由双方事先商定好），则从主机取一个新的数据帧，然后转到(2)。

在接收结点：

（1）等待。

（2）若收到由发送结点发过来的数据帧，则将其放入数据链路层的接收缓存。

（3）将接收缓存中的数据帧上交主机。

（4）向发送结点发一信息，表示数据帧已经上交给主机。

（5）转到(1)。

图 3-2 是前面所述的两种情况的对比。图 3-2(a)是不需要任何协议的理想化情况。主机 A 将数据帧（图中用 DATA 表示）连续发出，而不管发送速率有多快，收方总能够跟得上，收到一帧即交付给主机 B。显然，这种完全理想化情况的传输效率是很高的。

图 3-2(b)是由收方控制发方发送速率的情况。发方每发完一帧就必须停下来，等待收方的信息。这里要指出，由于假定了数据在传输过程中不会出差错，因此收方将数据帧交给主机 B 后向发方主机 A 发送的信息，不需要有任何具体的内容，即不需要说明所收到的数据是否是正确无误的。这相当于只要发回一个不需要装入任何信件的空信封就能起到流量控制的作用。

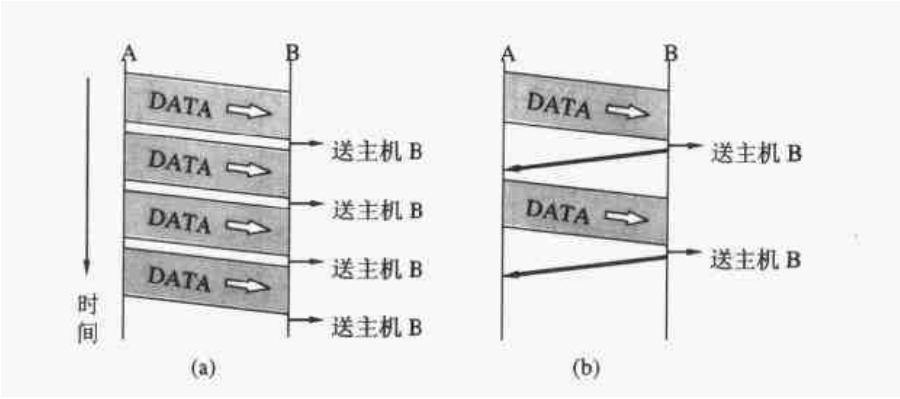


图 3-2　不需要任何数据链路层协议的数据传输

(a)和具有最简单的流量控制的数据链路层协议(b)

3.2.3　实用的停止等待协议

现在去掉前面的两个假定，讨论实用的数据链路层协议。这就是说，传输数据的信道不能保证使所传的数据不产生差错，并且还需要对数据的发送端进行流量控制。

图 3-3(a)画的是数据在传输过程中不出差错的情况。收方在收到一个正确的数据帧后，即交付给主机 B，同时向主机 A 发送一个**确认帧** ACK (ACKnowledgement)。当主机 A 收到确认帧 ACK 后才能发送一个新的数据帧。这样就实现了收方对发方的流量控制。

现在假定数据帧在传输过程中出现了差错。由于通常都在数据帧中加上了**循环冗余检验**

CRC (Cyclic Redundancy Check)，所以结点 B 很容易用硬件检验出收到的数据帧是否有差错（使用循环冗余检验进行差错检测的原理在后面的 3.2.4 小节讨论）。当发现差错时，结点 B 就向主机 A 发送一个否认帧 NAK (Negative Acknow ledgement)，以表示主机 A 应当重传出现差错的那个数据帧。图 3-3(b)画出了主机 A 重传数据帧。如多次出现差错，就要多次重传数据帧，直到收到结点 B 发来的确认帧 ACK 为止。为此，在发送端必须暂时保存已发送过的数据帧的副本。当通信线路质量太差时，则主机 A 在重传一定的次数后（如 8 次或 16 次，这要事先设定好），即不再进行重传，而是将此情况向上一层报告。

有时链路上的干扰很严重，或由于其他一些原因，结点 B 收不到结点 A 发来的数据帧。这种情况称为帧丢失（图 3-3(c)）。发生帧丢失时结点 B 当然不会向结点 A 发送任何确认帧。如果结点 A 要等收到结点 B 的确认信息后再发送下一个数据帧，那么就将永远等待下去。于是就出现了死锁现象。同理，若结点 B 发过来的确认帧丢失，也会同样出现这种死锁现象。

要解决死锁问题，可在结点 A 发送完一个数据帧时，就启动一个**超时计时器**(timeout timer)。计时器又称为**定时器**[MINGCI94]。若到了超时计时器所设置的重传时间 t_{out} 而仍收不到结点 B 的任何确认帧，则结点 A 就重传前面所发送的这一数据帧（见图 3-3(c),(d)）。如果在重传时间 t_{out} 内收到确认，则将超时计时器清零并停止计时。显然，超时计时器设置的重传时间应仔细选择确定。若重传时间选得太短，则在正常情况下也会在对方的确认信息回到发送方之前就过早地重传数据。若重传时间选得太长，则往往要白白等掉许多时间。**一般可将重传时间选为略大于“从发完数据帧到收到确认帧所需的平均时间”。**

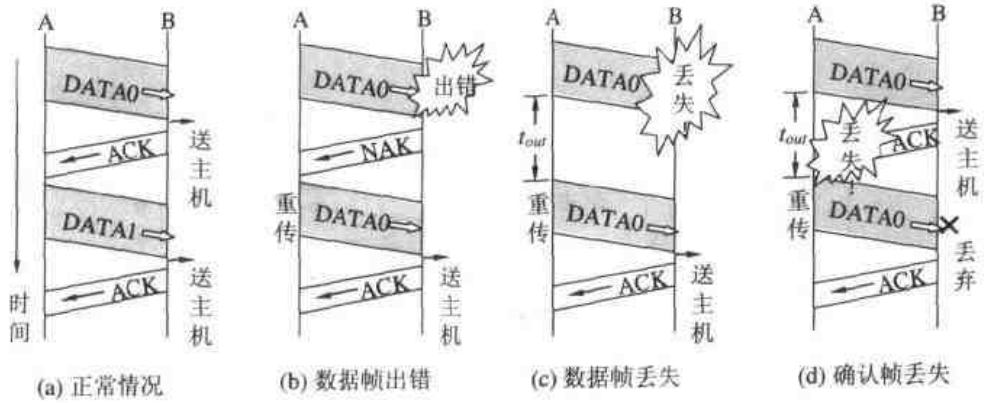


图 3-3　数据帧在链路上传输的几种情况

然而现在问题并没有完全解决。当出现数据帧丢失时，超时重传的确是一个好办法。但是若丢失的是确认帧，则超时重传将使主机 B 收到两个同样的数据帧。由于主机 B 现在无法识别重复的数据帧，因而在主机 B 收到的数据中出现了另一种差错——**重复帧**。重复帧也是一种不允许出现的差错。

要解决重复帧的问题，必须使每一个数据帧带上不同的发送序号。每发送一个新的数据帧就把它的发送序号加 1。若结点 B 收到发送序号相同的数据帧，就表明出现了重复帧。这时应当丢弃这重复帧，因为已经收到过同样的数据帧并且也交给了主机 B。但应注意，此时结点 B 还必须向结点 A 发送一个确认帧 ACK，因为结点 B 已经知道结点 A 还没有收到上一次发过去的确认帧 ACK。

我们知道，任何一个编号系统的序号所占用的比特数一定是有限的。因此，经过一段时

间后，发送序号就会重复。例如，当发送序号占用 3 bit 时，就可组成 8 种不同的发送序号，从 000 到 111。当数据帧的发送序号为 111 时，下一个发送序号就又是 000。因此，要进行编号就要考虑序号到底要占用多少个比特。序号占用的比特数越少，数据传输的额外开销就越小。对于停止等待协议，由于每发送一个数据帧就停止等待，因此用一个比特来编号就够了。一个比特可以有 0 和 1 两种不同的序号。这样，数据帧中的发送序号（以后记为 N(S)，S 表示发送）就以 0 和 1 交替的方式出现在数据帧中。每发一个新的数据帧，发送序号就和上次发送的不一样。用这样的方法就可以使收方能够区分开新的数据帧和重传的数据帧了。

从以上的讨论可以看出，虽然物理层在传输比特时会出现差错，但**由于数据链路层的停止等待协议采用了有效的检错重传机制，数据链路层对上面的网络层就提供了可靠传输的服务。**

3.2.4 循环冗余检验的原理

在数据链路层传送的帧中，广泛使用了循环冗余检验 CRC 的检错技术。下面我们用一个具体的例子来说明循环冗余检验的原理。

假设待传送的数据 $M = 1010001101$(共 k bit)。我们在 M 的后面再添加供差错检测用的 n bit **冗余码**一起发送。在所要发送的数据后面增加一些冗余码，虽然增大了数据传输的开销，但却可以进行差错检测。在传输可能出现差错时，付出这种代价还是值得的。

这 n bit 冗余码是这样得出的。用二进制的**模 2 运算**[①]进行 2^n 乘 M 的运算，这相当于在 M 后面添加 n 个 0。得到的$(k + n)$bit 的数除以事先选定好的长度为$(n + 1)$bit 的数 P，得出商是 Q 而**余数**是 R，余数 R 比除数 P 少 1 个比特。至于 P 是怎样选定的，下面还要介绍。在图 3-4 所示的例子中，$n = 5$, $P = 110101$，模 2 运算的结果是：商 $Q = 1101010110$，而余数 $R = 01110$。现在将得到的余数 R 就作为冗余码添加在数据 M 的后面发送出去，即发送的数据是 1010001101**01110**，或 $2^nM + R$。

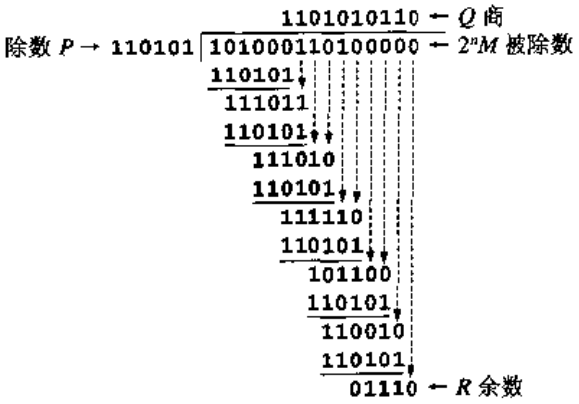


图 3-4 循环冗余检验的原理说明

为检测差错而在数据后面添加上的冗余码常称为**帧检验序列** FCS (Frame Check Sequence)。帧检验序列就是要保证收到的数据和发送的数据完全相同。这里应当注意，循环冗余检验 CRC 和帧检验序列 FCS 并不等同。CRC 是一种常用的检错方法，而 FCS 是添加在

① 注：用模 2 运算进行加法时不进位，例如，1111 + 1010 = 0101。减法和加法一样，按加法规则计算。

数据后面的冗余码，它可以用 CRC，但也不一定选用 CRC 这种方法。

如果数据在传输过程中不产生误码，则接收端收到的应当是 2^nM+R。将这个数除以 P(模 2 运算)后，得出的余数显然应当是 0 (读者可以自己进行类似图 3-4 的运算。被除数现在是 101000110101110，而除数是 $P=110101$，看余数是否为 0)。若数据在传输过程中出现误码，则在接收端进行以上的运算后，一般就不会得出余数为 0 的结果。

一种较方便的方法是用多项式来表示循环冗余检验过程，即使用多项式相应的系数来表示上述二进制数字中的 1 和 0。例如，可以用多项式 $P(X)=X^5+X^4+X^2+1$ 来表示上面的除数 P（称为**生成多项式**）。因此，在接收端进行的运算就可以写为

$$\frac{X^nM(X)+R(X)}{P(X)}$$

只要得出的余数 R 不为 0，就表示检测到了差错（注意：这种检测方法并不能确定究竟是哪一个或哪几个比特出现了差错），然后就**丢弃这个出现差错的帧**。

那么，能不能说只要得出的余数 R 是 0，就一定没有出现差错呢？不行！因为在某种非常特殊的比特差错组合下，也可能非常碰巧地使得余数 R 恰好为 0。但只要经过严格的挑选，并使用位数足够多的除数 P，那么出现检测不到的差错的概率就可以是个极小的数值。现在广泛使用的 $P(X)$有以下几种：

CRC-16 = $X^{16}+X^{15}+X^2+1$

CRC-CCITT = $X^{16}+X^{12}+X^5+1$

CRC-32 = $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$

应当注意，仅用循环冗余检验 CRC 差错检测技术只能做到**无差错接受**(accept)。所谓“无差错接受”就是指：“**凡是接受的帧（即不包括丢弃的帧），我们都能以非常接近于 1 的概率认为这些帧在传输过程中没有产生差错**”，或更简单些，这是指“**凡是接受的帧均无传输差错**”（丢弃的帧都不属于接受的帧）。而要做到“**可靠传输**”（即发送什么就收到什么）就**必须再加上确认和重传机制**。

3.2.5 停止等待协议的算法

为了对上面所述的停止等待协议有一个完整而准确的理解，下面给出此协议的算法，读者应弄清算法中的每一个步骤。这里不使用否认帧（实用的数据链路层协议大都是这样的），而且确认帧带有序号 n。我们按照网络协议习惯的表示法，ACKn 表示“第 $n-1$ 号帧已经收到，现在期望接收第 n 号帧”。因此，ACK1 表示“0 号帧已收到，现在期望接收下一帧，即 1 号帧”，而 ACK0 表示“1 号帧已收到，现在期望接收下一帧，即 0 号帧”。

在发送结点：

（1）从主机取一个数据帧，送交发送缓存。

（2）V(S)←0。　　{发送状态变量 V(S)初始化}

（3）N(S)←V(S)。　　{将发送状态变量值写入数据帧中的发送序号 N(S)}

（4）将发送缓存中的数据帧发送出去。　　{这个数据帧的副本仍保留在发送缓存中}

（5）设置超时计时器。　　{选择适当的超时重传时间 t_{out} }

（6）等待。　　{等待以下(7)和(8)这两个事件中最先出现的一个}

（7）收到确认帧 ACKn，{这里隐含地表示已经通过了 CRC 的硬件差错检测}

若 n=1-V(S)，则：　{已发送的数据帧被接收方确认}

从主机取一个新的数据帧，放入发送缓存；

V(S)←[1 – V(S)]；　　{更新发送状态变量，使用下一个序号}

转到(3)。

否则，丢弃这个确认帧，转到(6)。　　{这表明已发送的数据帧没有被接收方确认}

（8）若超时计时器时间到，则转到(4)。　　{重传已发送的数据帧}

在接收结点：

（1）V(R)←0。　　{接收状态变量初始化，其数值等于欲接收的数据帧的发送序号}

（2）等待。

（3）收到一个数据帧：　　{这里隐含地表示已经通过了 CRC 的硬件差错检测}

若 N(S)= V(R)，则执行(4)；　　{收到正确序号的数据帧}

否则丢弃此数据帧，然后转到(6)。　　{丢弃的帧就是重复帧}

（4）将收到的数据帧中的数据部分送交上层软件，也就是图 3-1 模型中的主机。

（5）V(R)←[1 – V(R)]。　　{更新接收状态变量，准备接收下一个数据帧}

（6）n←V(R)；

发送确认帧 ACK，转到(2)。　　{期望接收 n 号数据帧，在它之前的帧收到了}

从以上算法可知，停止等待协议在收发两端各设置一个本地状态变量(仅占 1 bit)。状态变量的概念很重要，一定要弄清它们的作用。总之，停止等待协议的要点如下：

（1）在发送端，在发送数据帧之前，都必须将发送状态变量 V(S)的值(即 0 或 1)写到数据帧的发送序号 N(S)上。只有收到序号正确的确认帧 ACKn 后，才更新发送状态变量 V(S)一次(将 1 变成 0，或 0 变成 1)，并发送新的数据帧。

（2）在接收端，每接收到一个数据帧，就要将发送端在数据帧上设置的发送序号 N(S)与本地的接收状态变量 V(R)相比较。若二者相等就表明是新的数据帧，就收下，并发送确认。否则为重复帧，就必须丢弃。但这时仍须向发送端发送确认帧 ACKn，而接收状态变量 V(R)和确认序号 n 都不变，和已发送过的确认帧是一样的。请注意，现在跳过了步骤(5)。

（3）连续出现相同发送序号的数据帧，表明发送端进行了超时重传。连续出现相同序号的确认帧，表明接收端收到了重复帧。

（4）发送端在发送完数据帧时，必须在其发送缓存中暂时保留这个数据帧的副本。这样才能在出差错时进行重传。只有确认对方已经收到这个数据帧时，才可以清除这个副本。

（5）实用的 CRC 检验器都是用硬件完成的。CRC 检验器能够自动丢弃检测到的出错帧。因此所谓的“丢弃出错帧”，对上层软件或用户来说都是感觉不到的。由于发送端对出错的数据帧进行重传是自动进行的，所以这种差错控制体制常简称为 ARQ (Automatic Repeat reQuest)，直译是**自动重传请求**，但意思是**自动请求重传**。

3.2.6 停止等待协议的定量分析

下面对停止等待协议进行定量分析（图 3-5）。我们仍采用图 3-1 所示的半双工通信模型。我们设结点 A 向结点 B 发送数据帧。结点 B 只发送确认帧而不发送否认帧，也不发送自己的数据帧。全双工通信的定量分析要稍复杂些。

设 t_f 是一个数据帧的**发送时间**，且数据帧的长度是固定不变的。显然，数据帧的发送时间 t_f 是数据帧的长度 l_f(bit)与数据的发送速率 C (bit/s)之比，即

$$t_f = l_f/C = l_f/C \text{ (s)} \qquad (3\text{-}1)$$

发送时间 t_f 也就是数据帧的发送时延。数据帧沿链路传到结点 B 还要经历一个**传播时延** t_p，它是信号（电磁波）在物理链路上传播所造成的时延。结点 B 收到数据帧要花费时间进行处理，此时间称为**处理时间** t_{pr}。结点 B 接着发送确认帧 ACK，其发送时间为 t_a，传播时延为

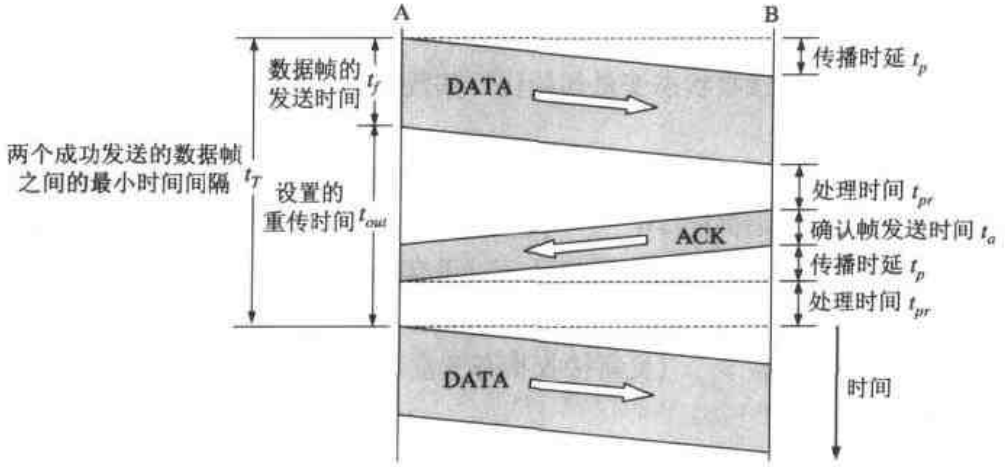


图 3-5　停止等待协议中数据帧和确认帧的发送时间关系

t_p（我们设信道的双向传播时延都是一样的）。结点 A 收到确认帧后也要花费处理时间，我们设这个时间和处理数据帧的时间一样，都是 t_{pr}。然后才接着发送下一个数据帧。为方便起见，我们设**重传时间**为

$$t_{out} = t_p + t_{pr} + t_a + t_p + t_{pr} \tag{3-2}$$

重传时间的作用是：数据帧发送完毕后若经过了这样长的时间还没有收到确认帧，就重传这个数据帧。为研究问题方便起见，我们设上式右端的处理时间 t_{pr} 和确认帧的发送时间 t_a 都远小于传播时延 t_p，这样就可简单地将**重传时间取为两倍的传播时延**，即

$$t_{out} = 2t_p \tag{3-3}$$

因此，**两个发送成功的数据帧之间的最小时间间隔** t_T 为

$$t_T = t_f + t_{out} = t_f + 2t_p \tag{3-4}$$

如遇发生差错，则成功发送 1 个数据帧所需的时间显然要超过 t_T。

现在设数据帧出现差错（包括帧丢失）的概率为 p，但假设确认帧不会出现差错（我们设确认帧很短，因而确认帧出差错的概率也就较小）。此外，允许重传的次数不受限制。这样，可以得出**正确传送一个数据帧所需的平均时间** t_{av} 来。推导的主要步骤如下：

$$t_{av} = t_T(1 + \text{一个帧的平均重传次数})$$

$$\begin{aligned}\text{一帧的平均重传次数} &= \{1 \times P[\text{重传次数为 1}] + 2 \times P[\text{重传次数为 2}] + 3 \times P[\text{重传次数为 3}] \\ &\quad + \ldots\} \\ &= \{1 \times P[\text{第 1 次发送出错}] \times P[\text{第 2 次发送成功}] \\ &\quad + 2 \times P[\text{第 1, 2 次发送出错}] \times P[\text{第 3 次发送成功}] \\ &\quad + 3 \times P[\text{第 1, 2, 3 次发送出错}] \times P[\text{第 4 次发送成功}] + \ldots\} \\ &= p(1-p) + 2p^2(1-p) + 3p^3(1-p) + \ldots\end{aligned}$$

这里 P[X]是出现事件 X 的概率。这样就可得出正确传送一个数据帧所需的平均时间

$$t_{av} = t_T + (1-p)\sum_{i=1}^{\infty} ip^i t_T = t_T/(1-p) \tag{3-5}$$

不难看出，当传输差错率增大时，t_{av} 也随之增大。当无差错时，$p=0$，$t_{av}=t_T$。

每秒成功发送的最大帧数就是链路的**最大吞吐量**λ_{max}。显然，

$$\lambda_{max} = 1/t_{av} = (1-p)/t_T \tag{3-6}$$

在发送端，设数据帧的实际到达率为 λ（即每秒到达 λ 个帧），则 λ 不应超过最大吞吐量λ_{max}，即

$$\lambda \le (1-p)/t_T \tag{3-7}$$

用时间 t_f 进行归一化，得出**归一化的吞吐量** ρ 为

$$\rho \equiv \lambda t_f \le (1-p)/\alpha < 1 \tag{3-8}$$

其中参数α是 t_T 的**归一化时间**：

$$\alpha \equiv t_T/t_f \ge 1 \tag{3-9}$$

当重传时间远小于发送时间时，$\alpha \approx 1$，此时的归一化吞吐量

$$\rho \le 1-p, \tag{3-10}$$

以上所讨论的停止等待协议 ARQ 的优点是：比较简单，但缺点是：通信信道的利用率不高，也就是说，信道还远远没有被数据比特填满。为了克服这一缺点，就产生了另外两种协议，即连续 ARQ 和选择重传 ARQ。这将在下面 3.3 和 3.4 节进一步讨论。

3.3 连续 ARQ 协议

3.3.1 连续 ARQ 协议的工作原理

我们先用图 3-6 所示的简单例子来讨论连续 ARQ 协议的工作原理。它的要点就是在发送完一个数据帧后，不是停下来等待确认帧，而是可以连续再发送若干个数据帧。如果这时收到了接收端发来的确认帧，那么还可以接着发送数据帧。由于减少了等待时间，整个通信的吞吐量就提高了。

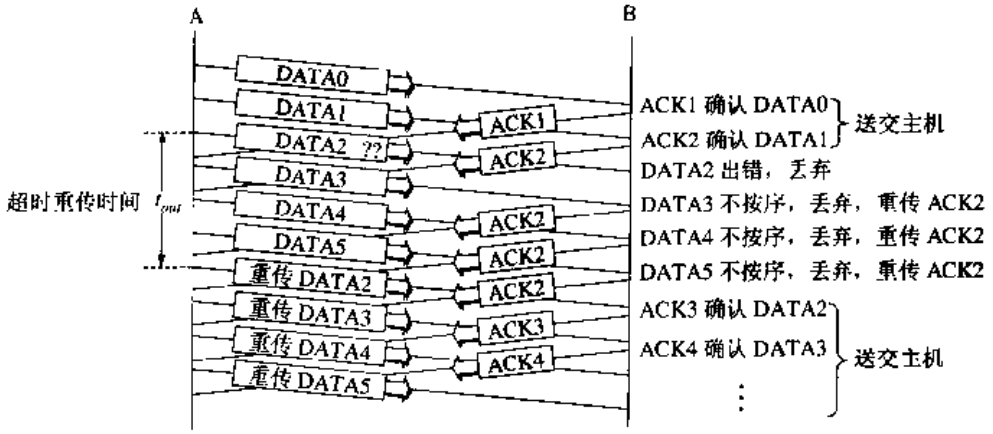


图 3-6 连续 ARQ 协议的工作原理：对出错数据帧的处理

如图 3-6 所示，结点 A 向结点 B 发送数据帧。当结点 A 发完 0 号帧后，不是停止等待，而是继续发送后续的 1 号帧、2 号帧等。A 每发送完一帧就要为该帧设置超时计时器。由于

连续发送了许多帧，所以确认帧必须要指明是对哪一帧进行确认。在图 3-6 中，ACKn 表示对第$(n-1)$号帧的确认。这表示对发送方说："我已正确收到了第$(n-1)$号帧，下一次我期望收到第 n 号帧"。

结点 B 正确地收到了 0 号帧和 1 号帧，并送交其主机。现在设 2 号帧出了差错，于是结点 B 的 CRC 检验器就自动将有差错的 2 号帧丢弃，然后就等待发送端超时重传。

这里要注意以下四点：

（1）接收端只**按序接收**数据帧。虽然在有差错的 2 号帧之后接着又收到了正确的 3 个数据帧，但接收端都必须将这些帧丢弃，因为在这些帧前面有一个 2 号帧还没有收到。接收端虽然丢弃了这些不按序的无差错帧，但应重复发送已经发送过的最后一个确认帧 ACK2（这是防止已经发送的确认帧 ACK2 丢失了）。

（2）ACK1 表示确认 0 号帧 DATA0，并期望下次收到 1 号帧；ACK2 表示确认 1 号帧 DATA1，并期望下次收到 2 号帧。依此类推。

（3）结点 A 在每发送完一个数据帧时都要**设置该帧的超时计时器**。如果在所设置的超时时间 t_{out} 内收到确认帧，就立即将超时计时器清零。但若在所设置的超时时间 t_{out} 到了而仍未收到确认帧，就要重传相应的数据帧（仍需重新设置超时计时器）。在等不到 2 号帧的确认而重传 2 号数据帧时，虽然结点 A 已经发完了 5 号帧，但仍必须向回走，将 2 号帧及其以后的各帧全部进行重传。正因为如此，连续 ARQ 又称为 Go-back-N ARQ，意思是当出现差错必须重传时，要向回走 N 个帧，然后再开始重传。

（4）以上讲述的仅仅是连续 ARQ 协议的工作原理。协议在具体实现时还有许多细节。例如，用一个计时器就可实现相当于 N 个独立的超时计时器的功能（见习题 3-14）。

从这里不难看出，连续 ARQ 协议一方面因连续发送数据帧而提高了信道的利用率，但另一方面，在重传时又必须把原来已传送正确的数据帧进行重传（仅因这些数据帧的前面有一个数据帧出了错），这种做法又使传送效率降低。由此可见，若传输信道的传输质量很差因而误码率较大时，连续 ARQ 协议不一定优于停止等待协议。

若 2 号数据帧不是出现差错而是丢失了，则情况也是类似的，读者可自行分析。

3.3.2 连续 ARQ 协议的吞吐量

可以很方便地导出连续 ARQ 协议的吞吐量公式。

我们假定在不出现差错时，成功地发送一个数据帧所需的时间是 t_f（而不是像图 3-5 中所示的需时间 t_T）。当出现差错时，重传一个数据帧所需的时间设为 t_T。所以只要参照停止等待协议的公式(3-5)，就不难得出，在连续 ARQ 协议的情况下，正确传送一个数据帧所需的平均时间是

$$t_{av} = t_f + (1-p)\sum_{i=1}^{\infty} ip^i t_T = t_f[1+(\alpha-1)p]/(1-p) \tag{3-11}$$

这里参数 α 仍为比值 t_T/t_f，t_T 略大于 $t_f + t_{out}$。

在发送结点处于饱和状态下，吞吐量的最大值是

$$\lambda_{\max} = 1/t_{av} = (1-p)/t_f[1+(\alpha-1)p] \tag{3-12}$$

而归一化的吞吐量为

$$\rho = \lambda t_f \le (1-p)/[1+(\alpha-1)p] \tag{3-13}$$

我们应注意到，若传播时延、重传时间等都远小于一个数据帧的发送时间 t_f，则 $\alpha \approx 1$，这时(3-13)式将和(3-10)式一样。

下面是几个简单的例子。

【例 1】 若数据帧的差错率 $p = 0.01$，而参数 $\alpha = 4$，则对于停止等待协议，$\rho \leq 0.99/4$，但对于连续 ARQ 协议，$\rho \leq 0.96$。故即使在数据帧的差错率高达 0.01 时，连续 ARQ 的效率也比停止等待协议的高。

【例 2】 考虑在广域网上传数据。设数据帧长为 1200 bit，线路传输速率为 9.6 kb/s。求出数据帧的发送时间 $t_f = 125$ ms。设链路长度为 160 km。若传播时延为 1 ms，它显然远小于数据帧的发送时间。若确认帧的发送时间和结点对数据帧和确认帧的处理时间都远小于 125 ms，则 t_{out} 也应远小于 t_f。这时，停止等待协议与连续 ARQ 协议区别很小。

现在假定通信是全双工的。这时，结点 B 没有必要专门发送确认帧 ACK 或 NAK。当结点 B 向结点 A 发送数据时，把确认信息捎带传过去即可。这样做可以提高效率。在这种情况下，发送端收到对方确认的最短时间是 $2t_p + t_f$，这里 t_p 是传播时延。为了防止不必要的超时重传，可以把 t_{out} 取得稍大些，例如，取 $t_{out} = 2t_p + 2t_f$。于是得出

$$t_T = t_f + t_{out} = 2t_p + 3t_f$$

$$\alpha = t_T / t_f = 3 + 2t_p / t_f$$

在这种情况下，连续 ARQ 协议明显优于停止等待协议。

如果将线路速率从 9.6 kb/s 提高到 56 kb/s，则对 1200 bit 长的数据帧，

$$t_f = 1200/56\,000 \approx 21 \text{ ms}$$

对于 1600 km 的线路，$2t_p$ 约为 20 ms，而 $t_{out} = 2t_p + 2t_f = 62$ ms。此时 $\alpha = 1 + 62/21 = 4$。在这种情况下，连续 ARQ 协议比停止等待协议强得多。

【例 3】 两个卫星地球站通过卫星进行通信。从发送站到接收站的传播时延一般在 250～270 ms 之间，这里取传播时延 $t_p = 250$ ms。设数据帧长为 1200 bit，而线路速率为 4.8 kb/s。此时，$t_f = 250$ ms。取 $t_{out} = 2t_p + 2t_f$，得出 $\alpha = t_T / t_f = 5$。若线路速率为 9.6 kb/s，则 $t_f = 125$ ms，$\alpha = 7$。若线路速率再提高到 48 kb/s，则 $t_f = 25$ ms，$\alpha = 23$。很明显，在卫星通信时，由于传播时延很大，停止等待协议就很不适用。这时必须采用连续 ARQ 协议。

3.3.3 滑动窗口的概念

在前面已经介绍了连续 ARQ 的概念。下面我们进行更深入的讨论。

在使用连续 ARQ 协议时，有以下两个问题要解决：

（1）当未被确认的数据帧的数目太多时，只要有一帧出了差错，就可能要有很多的数据帧需要重传，这必然就要白白花费较多的时间，因而增大开销。

（2）为了对所发送出去的大量数据帧进行编号，每个数据帧的发送序号也要占用较多的比特数，这样又增加了一些不必要开销。

因此，在连续 ARQ 议中，应当将已发送出去但未被确认的数据帧的数目加以限制。这就是本节滑动窗口所要研究的内容。

我们从停止等待协议中已经得到了启发。在停止等待协议中，无论发送多少帧，只需使用 1 个比特来编号就足够了。发送序号循环使用 0 和 1 这两个序号。对于连续 ARQ 协议，也可采用同样的原理，即循环重复使用已收到确认的那些帧的序号。这时只需要在控制信息中用有限的几个比特来编号就够了。当然这还要加入适当的控制机制才行。这就是要在发送

端和接收端分别设定所谓的**发送窗口**和**接收窗口**。下面先讨论发送窗口。

发送窗口用来对发送端进行流量控制，而发送窗口的大小 W_T 代表在还没有收到对方确认信息的情况下发送端最多可以发送多少个数据帧。显然，停止等待协议的发送窗口大小是1，表明只要发送出去的某个数据帧未得到确认，就不能再发送下一个数据帧。发送窗口的概念最好用图形来说明。

我们现在设发送序号用 3 比特来编码，即发送序号可以有 8 个不同的序号，从 0 到 7。又假定发送窗口 $W_T = 5$，这表示在未收到对方确认信息的情况下，发送端最多可以发送出 5 个数据帧。发送窗口的规则归纳如下：

（1）发送窗口内的帧是允许发送的帧，而不考虑有没有收到确认。发送窗口右侧所有的帧都是不允许发送的帧。图 3-7(a)说明了这一情况。

（2）每发送完一个帧，允许发送的帧数就减 1。但发送窗口的位置不改变。图 3-7(b)有三种不同的帧：已经发送了帧（最左边的 0 号帧）；允许发送的帧（共 4 个，左边的 1 号帧至 4 号帧）；以及不允许发送的帧（5 号帧和以后的帧）。

（3）如果所允许发送的 5 个帧都发送完了，但还没有收到任何确认，那么就不能再发送任何帧了。图 3-7(c)表示这种情况。这时，发送端就进入等待状态。

（4）每收到对一个帧的确认，发送窗口就向前（即向右方）滑动一个帧的位置。图 3-7(d)表示收到了对前三个帧的确认，因此发送窗口可向右方滑动三个帧的位置。在图 3-7(d)中共有四种不同的帧：已发送且已收到了确认的帧（最左边的 0～2 号帧）；已发送但未收到确认的帧（左边的 3～4 号帧）；还可以继续发送的帧（5～7 号帧）；以及不允许发送的帧（右边的 0 号帧和以后的帧）。

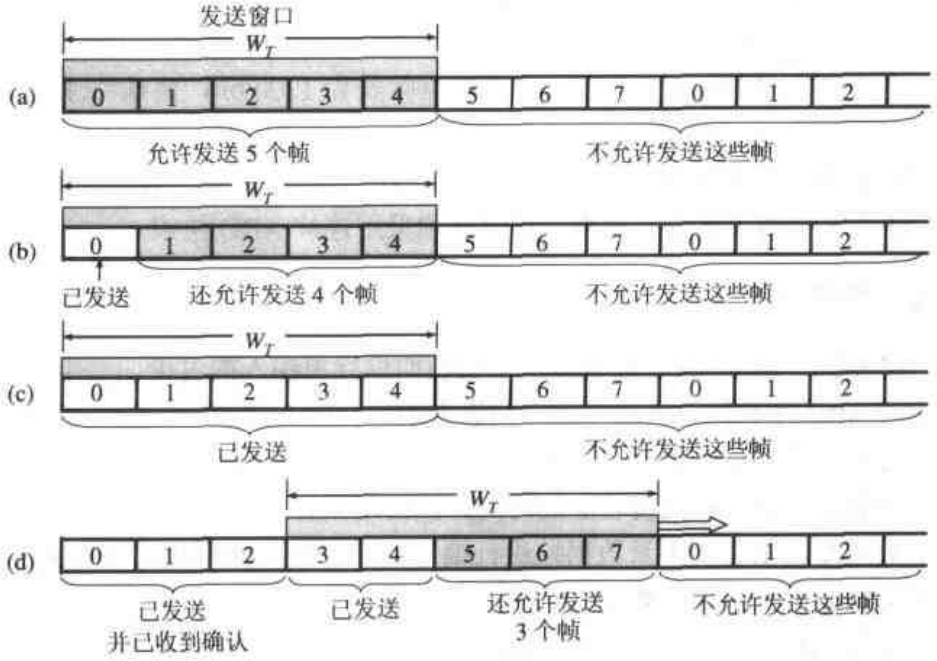


图 3-7　发送窗口控制发送端的发送速率

(a)允许发送 0～4 号共 5 个帧；(b)允许发送 1～4 号共 4 个帧；

(c)不允许发送任何帧；(d)允许发送 5～7 号共 3 个帧

为了减少开销，连续 ARQ 协议还规定接收端不一定每收到一个正确的数据帧就必须立

即发回一个确认帧，而是可以在连续收到好几个正确的数据帧以后，才对最后一个数据帧发确认信息，或者可以在当自己有数据要发送时才将对以前正确收到的帧加以捎带确认。这就是说，对某一数据帧的确认就表明该数据帧和这以前所有的数据帧均已正确无误地收到了。这样做可以使接收端少发送一些确认帧，因而减少了开销。例如，在图 3-7(d)中，接收端可以只发送一个对 2 号帧的确认，表示 2 号帧和它以前的 0 号和 1 号帧都已经正确收到了。

同理，在接收端设置接收窗口是为了控制可以接收哪些数据帧而不可以接收哪些帧。**在接收端只有当收到的数据帧的发送序号落入接收窗口内才允许将该数据帧收下。**若接收到的数据帧落在接收窗口之外，则一律将其丢弃。在连续 ARQ 协议中，接收窗口的大小 $W_R = 1$。接收窗口的规则很简单，归纳如下：

（1）只有当收到的帧的序号与接收窗口一致时才能接收该帧。否则，就丢弃它。

（2）每收到一个序号正确的帧，接收窗口就向前（即向右方）滑动一个帧的位置。同时向发送端发送对该帧的确认。

图 3-8(a)表明一开始接收窗口处于 0 号帧处，接收端准备接收 0 号帧。一旦收到 0 号帧，接收窗口即向前滑动一个帧的位置（图 3-8(b)），准备接收 1 号帧，同时向发送端发送对 0 号帧的确认信息。当陆续收到 1 号至 3 号帧后，接收窗口的位置应如图 3-8(c)所示。

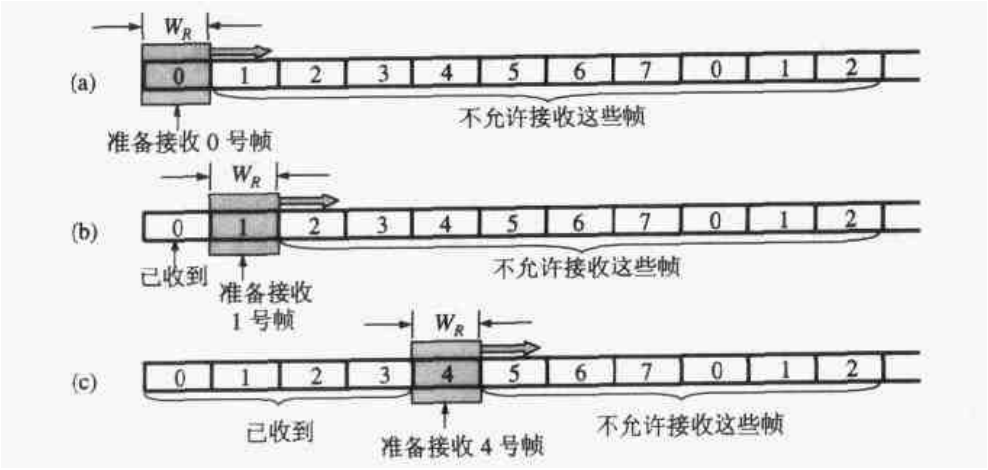


图 3-8　接收窗口 W_R 的意义

(a)准备接收 0 号帧；(b)准备接收 1 号帧；(c)准备接收 4 号帧

不难看出，只有在接收窗口向前滑动时（与此同时也发送了确认），发送窗口才有可能向前滑动。发送端若没有收到该确认，发送窗口就不能滑动。

正因为收发两端的窗口按照以上规律不断地向前滑动，因此这种协议又称为**滑动窗口协议**。当发送窗口和接收窗口的大小都等于 1 时，就是我们最初讨论的停止等待协议。

下面讨论当数据帧的发送序号所占用的比特数一定时，发送窗口的最大值是多少。初看起来，问题好像很简单。例如用 3 比特可编出 8 个不同的序号，因而发送窗口的最大值似乎应为 8。但实际上，设置发送窗口为 8 将使协议在某些情况下无法工作。现在我们就来说明这一点。

现在设发送窗口 $W_T = 8$。设发送端发送完 0～7 号共 8 个数据帧。因发送窗口已满，发送暂停。假定这 8 个数据帧均已正确到达接收端，并且对每一个数据帧，接收端都发送出确认帧。下面考虑两种不同的情况。

第一种情况是：所有的确认帧都正确到达了发送端，因而发送端接着又发送 8 个新的数据帧，其编号应当是 0～7。请注意，序号是循环使用的。所以序号虽然相同，但 8 个帧都是新的帧。

第二种情况是：所有的确认帧都丢失了。经过一段由超时计时器控制的时间后，发送端重传这 8 个旧的数据帧，其编号仍为 0～7。

问题已经十分明显了。接收端第二次收到编号为 0～7 的 8 个数据帧时，无法判定：这是 8 个新的数据帧，或这是 8 个旧的、重传的数据帧。

因此，将发送窗口设置为 8 显然是不行的。像上述那样的证明方法很有用。这就是当我们要证明某个协议是错误时，只要设法找出一个具体例子说明该协议不能正确工作就算证明完毕。但反过来若要证明某个协议是正确的就很困难。就算能够举出 1000 个例子说明协议能正常工作也还不充分，因为很可能该协议用到第 1001 个例子就出现问题。

可以证明，当用 n 个比特进行编号时，若接收窗口的大小为 1，则只有在发送窗口的大小 $W_T \leq 2^n - 1$ 时，连续 ARQ 协议才能正确运行（见习题 3-9）。这就是说，当采用 3 bit 编码时，发送窗口的最大值是 7 而不是 8。这对一般的陆地链路已足够大了。但对于卫星链路，由于其传播时延很大，发送窗口也必须适当增大才能使信道利用率不致太低。这时常用 7 bit 编码，因而发送窗口的大小可达 127。在这种情况下，所有已发送出去的但尚未被确认的数据帧都必须保存在发送端的缓存中，以便在出差错时进行重传。当然，这就要占用相当大的存储器空间。相反，对于停止等待协议，发送窗口 $W_T = 1$，发送端只要花费 1 个数据帧的存储空间即可。

顺便指出，上述的这种对已发送过的数据帧的保存，是使用一个先进先出的队列。发送端每发完一个新的数据帧就将该帧存入这个队列。当队列长度达到发送窗口大小 W_T 时，即停止再发送新的数据帧。当按照协议进行重传（重传 1 帧或多帧）时，队列并不发生变化。只有当收到对应于队首的帧的确认时，才将队首的数据帧清除。若队列变空，则表明全部已发出的数据帧均已得到了确认。

3.3.4 信道利用率

由于每个数据帧都必须包括一定的控制信息（如帧的序号、地址、同步信息以及其他的一些控制信息），所以即使连续不停地发送数据帧，**信道利用率**（即扣除全部的控制信息后的数据率与信道容量之比）也不可能达到 100 %。当出现差错时（这是不可避免的），数据帧的不断重传将进一步使信道利用率降低。

很明显，若数据帧的帧长取得很短，那么控制信息在每一帧中所占的比例就增大，因而额外开销增大，这就导致信道利用率的下降。但反过来，若帧长取得太长，则数据帧在传输过程中出错的概率就增大，于是重传次数将增大，这也会使信道利用率下降。由此可见，存在一个最佳帧长，在此帧长下信道的利用率最高。

要定量地分析这问题，就必须知道链路的差错特性。对于卫星信道，**每个比特的出错可视为独立的**（因为差错主要是由卫星到地球的传输途径中的随机噪声引起的）。误比特率记为 p_b。设数据帧长为 l_f（即数据部分加上控制信息），则数据帧的差错率或误帧率（即 1 帧中至少有 1 个比特出错的概率）为

$$p = 1 - (1 - p_b)^{l_f} \tag{3-14}$$

对于很小的 p_b，上式可近似为

$$p \approx l_f p_b << 1 \qquad (3\text{-}15)$$

上述的这种误码特性并不适用于陆地上的链路，因为对陆地上的链路，**突发性误码是主要的**。当发生突发性误码时，一连串的比特都要出错。但是，一些实验表明，在产生突发性误码时，误帧率与帧长成正比[BURT72]。这样，当差错率很低时，公式(3-15)既可用于卫星链路，又可用于陆地上的链路。

设发送端工作在饱和状态的发送速率为每秒发送λ_{max}帧。设每帧中数据为 l_d bit 而控制信息为 l_h bit。由(3-12)式可求出**平均有效数据率** D（指最后交付给主机的平均数据率）为

$$D = \lambda_{max} l_d = (1-p) l_d / t_f [1 + (\alpha - 1)p] \qquad (3\text{-}16)$$

令 $t_f = l_f / C = (l_d + l_h)/C$，其中 C 为链路容量（也就是链路的带宽或链路的最高数据率），则可得出连续 ARQ 协议的单位信道容量的平均有效数据率，即通常所说的信道利用率 U

$$U = \frac{D}{C} = \left(\frac{l_d}{l_d + l_h}\right)\left[\frac{1-p}{1+(\alpha-1)p}\right] \le 1 \qquad (3\text{-}17)$$

公式(3-17)就是信道利用率与帧长的关系。信道利用率越接近于 1，信道的利用就越充分。由于误帧率 p 以及参数 a 都与帧长有关，因此只有在近似的条件下信道利用率与帧长才能写成简单的数学表达式。

3.4 选择重传 ARQ 协议

为进一步提高信道的利用率，可设法只重传出现差错的数据帧或者是计时器超时的数据帧。但这时必须加大接收窗口，以便先收下发送序号不连续但仍处在接收窗口中的那些数据帧。等到所缺序号的数据帧收到后再一并送交主机。这就是**选择重传 ARQ 协议**。

使用选择重传 ARQ 协议可以避免重复传送那些本来已经正确到达接收端的数据帧。但我们付出的代价是在接收端要设置具有相当容量的缓存空间，这在许多情况下是不够经济的。正因如此，选择重传 ARQ 协议在目前就远没有连续 ARQ 协议使用得那么广泛。今后存储器芯片的价格会更加便宜，选择重传 ARQ 协议还是有可能受到更多的重视。

对于选择重传 ARQ 协议，接收窗口显然不应该大于发送窗口。若用 n 比特进行编号，则可以证明，接收窗口的最大值受下式的约束（见习题 3-10）

$$W_R \le 2^n/2 \qquad (3\text{-}18)$$

当接收窗口 W_R 为最大值时，$W_T = W_R = 2^n/2$。例如在 $n = 3$ 时，可以算出 $W_T = W_R = 4$。

3.5 面向比特的链路控制规程 HDLC

3.5.1 HDLC 协议概述

在计算机通信的早期人们就已发现，对于经常产生误码的实际链路，只要加上合适的控制规程，就可以使通信变为比较可靠的。那时 ARPANET 和 IBM 公司分别使用了各自的控制规程，它们分别是：IMP-IMP 协议和 BSC 规程（也可称为 BISYNC，即 BInary SYNchronous Communication 的缩写）。这些规程都是数据链路层的协议。

上述的两种协议都是**面向字符**的。所谓面向字符就是说在链路上所传送的数据必须是由规定字符集（例如 ASCII 码）中的字符所组成。不仅如此，在链路上传送的控制信息也必须

由同一个字符集中的若干指定的控制字符构成。这种面向字符的链路控制规程在计算机网络的发展过程中曾起了重要的作用。

但随着计算机通信的发展，这种面向字符的链路控制规程就逐渐暴露出其弱点。就以著名的 BSC 规程来说，其主要限制是：

（1）通信线路的利用率低，因为它采用的是停止等待协议，收发双方交替地工作。

（2）所有通信的设备必须使用同样字符代码，而不同版本的 BSC 规程要求使用不同的代码。

（3）只对数据部分进行差错控制，若控制部分出错就无法控制，因而可靠性较差。

（4）不易扩展。每增加一种功能就需要设定一个新的控制字符。

此外，它还存在其他一些缺点。由此可见，需要设计出一种新的链路控制规程来代替旧的面向字符的链路规程。

1974 年，IBM 公司推出了著名的体系结构 SNA。在 SNA 的数据链路层规程采用了**面向比特**的规程 SDLC (Synchronous Data Link Control)。后来 ISO 把 SDLC 修改后称为 HDLC (High-level Data Link Control)，译为**高级数据链路控制**，作为国际标准 ISO 3309。我国的相应国家标准是 GB 7496。CCITT 则将 HDLC 再修改后称为**链路接入规程 LAP** (Link Access Procedure)，并作为 X.25 建议书的一部分（即有关数据链路层协议的部分）。不久，HDLC 的新版本又把 LAP 修改为 LAPB，“B”表示平衡型(Balanced)，所以 LAPB 叫做**链路接入规程（平衡型）**。

HDLC 可适用于链路的两种基本配置，即**非平衡配置**与**平衡配置**。非平衡配置的特点是由一个**主站**(primary station)控制整个链路的工作。主站发出的帧叫做**命令**(command)。受控的各站叫做**次站**或**从站**(secondary station)。次站发出的帧叫做**响应**(response)。在多点链路中，主站与每一个次站之间都有一个分开的逻辑链路。

平衡配置的特点是链路两端的两个站都是**复合站**(combined station)。复合站同时具有主站与次站的功能。因此每个复合站都可以发出命令和响应。

对于非平衡配置，只有主站才能发起向次站的数据传输，而次站只有在主站向它发送命令帧进行**探询**(polling，也常称为**轮询**)时，才能以响应帧的形式回答主站。主站还负责链路的初始化、链路的建立和释放，以及差错恢复等。

平衡配置的特点是每个复合站都可以平等地发起数据传输，而不需要得到对方复合站的允许。

3.5.2 HDLC 的帧结构

1．各字段的意义

数据链路层的数据传送是以帧为单位的。一个帧的结构具有固定的格式（见图 3-9）。从网络层交下来的分组，变成为数据链路层的数据。这就是图 3-9 中的信息字段。信息字段的长度没有具体规定。数据链路层在信息字段的头尾各加上 24 bit 的控制信息，这样就构成了一个完整的帧。下面分别介绍控制信息中各字段的意义。

我们知道，物理层要解决比特同步的问题。但是，数据链路层要解决帧同步的问题。所谓帧同步就是从收到的比特流中正确无误地判断出一个帧从哪个比特开始以及到哪个比特结束。为此，HDLC 规定了在一个帧的开头（即首部中的第一个字节）和结尾（即尾部中的最

后一个字节）各放入一个特殊的标记，作为一个帧的边界。这个标记就叫做**标志字段 F (Flag)**。标志字段 F 为 6 个连续 1 加上两边各一个 0 共 8 bit。在接收端，只要找到标志字段，就可以很容易地确定一个帧的位置。

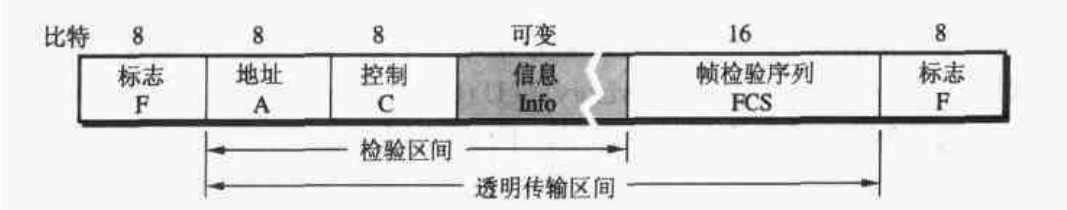


图 3-9 HDLC 的帧结构

在两个标志字段之间的比特串中，如果碰巧出现了和标志字段 F 一样的比特组合，那么就会在处理 HDLC 帧首部时误认为是找到了一个帧的边界。为了避免出现这种错误，HDLC 采用**零比特填充法**使一帧中两个 F 字段之间不会出现 6 个连续 1。

零比特填充的具体做法是：在发送端，当一串比特流数据尚未加上标志字段时，先用硬件扫描整个帧（用软件也能实现，但要慢些）。只要发现有 5 个连续 1，则立即填入一个 0。因此经过这种零比特填充后的数据，就可以保证在数据中不会出现 6 个连续 1。在接收一个帧时，先找到 F 字段以确定一个 HDLC 帧的边界。接着再用硬件对其中的比特流进行扫描。每当发现 5 个连续 1 时，就将这 5 个连续 1 后的一个 0 删除，以还原成原来的比特流（图 3-10）。这样就保证了在所传送的数据比特流中，不管出现什么样的比特组合，也不至于引起对帧边界的判断错误。

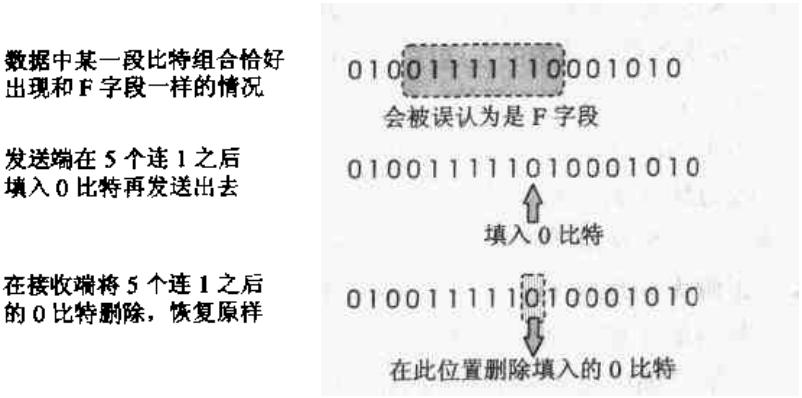


图 3-10 零比特的填充与删除

采用零比特填充法就可传送任意组合的比特流，或者说，就可实现数据链路层的**透明传输**。在图 3-9 两个 F 字段之间注明的“透明传输区间”就是这个意思。

当连续传输两个帧时，前一个帧的结束标志字段 F 可以兼作后一帧的起始标志字段。当暂时没有信息传送时，可以连续发送标志字段，使收端可以一直和发端保持同步。

地址字段 A 也是 8 个比特。在使用非平衡方式传送数据时，地址字段总是写入次站的地址。但在平衡方式时，地址字段总是填入确认站的地址。

全 1 地址是广播方式，而全 0 地址是无效地址。因此，有效的地址共有 254 个。这对一般的多点链路是足够的。但考虑在某些情况下，例如使用分组无线电，用户可能很多，所以地址字段就做成可扩展的。这时用地址字段的比特 1 表示扩展比特，其余 7 个比特为地址比特。当某个地址字段的第 1 比特为 0 时，表示下一个地址字段的后 7 个比特也是地址比特。当此地址字段的第 1 比特为 1 时，即表示这已是最后一个地址字段了。

帧检验序列字段 FCS 共 16 bit。它采用的生成多项式是 CRC-CCITT。所检验的范围是从地址字段的第 1 个比特起，到信息字段的最末 1 个比特为止。图 3-9 标志出了这个检验范围。

控制字段 C 共 8 bit，是最复杂的字段。HDLC 的许多重要功能都靠控制字段来实现。根据其最前面两个比特的取值，可将 HDLC 帧划分为三大类，即**信息帧、监督帧和无编号帧**，其简称分别是 I (Information)，S (Supervisory)和 U (Unnumbered)。图 3-11 是对应于这三种帧的控制字段以及控制字段中的各比特的作用。下面分别介绍这三种帧的特点。

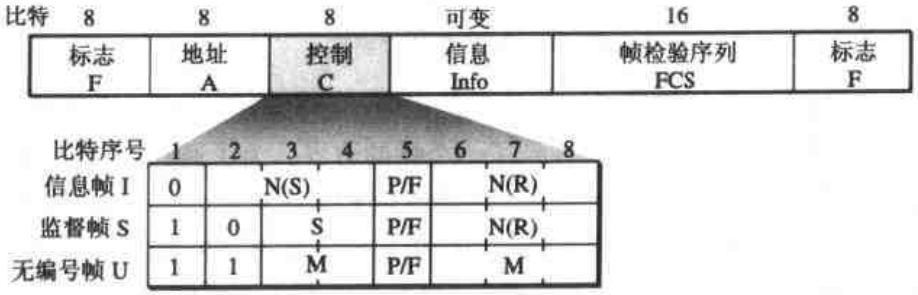


图 3-11　控制字段的结构

2．信息帧

若控制字段的第 1 比特为 0，则该帧为**信息帧**。比特 2～4 为发送序号 N(S)，而比特 6～8 为接收序号 N(R)。N(S)表示**当前发送的信息帧的序号**，而 N(R)表示本站**所期望收到的帧的发送序号**（注意：对方发送的信息帧中的发送序号是由对方确定并填入的，而这里的 N(R)则是由本站填入的）。由于是全双工通信，所以通信的每一方都各有一个 N(S)和 N(R)，因此通信的双方总共有两个 N(S)和两个 N(R)。因此，当讨论到 N(S)和 N(R)时，一定要弄清这是在发送方还是在接收方填入的序号。

这里要强调指出，N(R)带有确认的意思。它表示序号为 [N(R)– 1] (mod 8)的帧以及在这以前的各帧都已正确无误地收妥了。这里 mod 8(模 8）的意思是：序号只有 8 个数值，当序号增加到 7 时，若再加 1 就又回到了 0。

顺便指出，在信息帧中设有接收序号 N(R)这一字段，就表示不必专门为收到的信息帧发送确认确认帧。可以在本站有信息帧发送时，将确认信息放在其接收序号 N(R)中让本站发送信息帧时将确认信息**捎带走**(piggybacking)。例如，在一连收到对方 N(S)= 0～3 共 4 个信息帧后，可在即将发送的信息帧中将接收序号 N(R)置为 4，表示 3 号帧及其以前的各帧均已正确收到，而期望接收的是发送序号 N(S)= 4 的信息帧。采用这种捎带的方法可以提高信道的利用率。

控制字段的第 5 个比特是**探询/终止**(Poll/Final)比特，简称 P/F 比特。主站发出的命令帧中若将 P 比特置为 1 则表示要求对方立即发送响应。在对方确认的帧中若将 F 比特置为 1 则表示要发送的数据已经发送完毕。

3．监督帧

若控制字段的第 1～2 比特为 1 0，则对应的帧即为**监督帧 S**。监督帧共有四种，取决于第 3～4 比特的值（见图 3-11 中标有 S 的二比特）。表 3-1 是这四种监督帧的名称和功能。

表 3-1 四种监督帧的名称和功能

第 3～4 比特	帧　名	功　能
0 0	RR (Receive Ready) 接收准备就绪	准备收下一帧 确认序号为 N(R)－1 及其以前的各帧
1 0	RNR (Receive Not Ready) 接收未就绪	暂停接收下一帧 确认序号为 N(R)－1 及其以前的各帧
0 1	REJ (Reject) 拒绝	从 N(R)起的所有帧都被否认 但确认序号为 N(R)－1 及其以前的各帧
1 1	SREJ (Selective Reject) 选择拒绝	只否认序号为 N(R)的帧 但确认序号为 N(R)－1 及其以前的各帧

上述四种监督帧中，前三种用在连续 ARQ 协议中，而最后一种只用于选择重传 ARQ 协议中（较少使用）。

所有的监督帧都不包含要传送的数据信息，因此它只有 48 bit 长。显然，监督帧不需要有发送序号 N(S)。但监督帧中的接收序号 N(R)却是至关重要的。在前两种监督帧中的 N(R)都具有同样的含义，因此这两种监督帧都相当于以前提到过的确认帧 ACK。REJ 则相当于以前提到过的否认帧 NAK，而在 REJ 帧中的 N(R)表示所否认的帧号。不过这种否认帧还带有某种确认信息，即确认序号为 N(R)–1 及其以前的各帧均已正确收到。

我们应当注意到，RR 帧和 RNR 帧还具有流量控制的作用。RR 帧表示已做好接收帧的准备，希望对方继续发送，而 RNR 帧则表示希望对方停止发送（这可能是由于来不及处理到达的帧，或缓存已满）。

4．无编号帧

若控制字段的第 1～2 比特都是 1 时，这个帧就是无编号帧 U。无编号帧本身不带编号，即无 N(S)和 N(R)字段，而是用 5 bit (图 3-12 中标有 M 的第 3, 4, 6, 7, 8 比特)来表示不同功能的无编号帧。虽然总共可以有 32 个不同组合，但实际上目前只定义了 15 种无编号帧。无编号帧主要起控制作用，可在需要时随时发出。

3.6 因特网的点对点协议 PPP

3.6.1 PPP 协议的工作原理

虽然 HDLC 协议在历史上曾经起过很大的作用，但现在全世界使用得最多的数据链路层协议却是另一个协议——点对点协议 PPP (Point-to-Point Protocol)。下面就介绍 PPP 协议。

用户接入因特网的一般方法有两种。一种是用户使用拨号电话线接入因特网，另一种是使用专线接入（这通常是用户人数较多的单位）。不管用哪一种方法，在传送数据时都需要有数据链路层协议。

图 3-12 是用户拨号入网的示意图。因特网服务提供者 ISP 是一个能够提供用户拨号入网的经营机构。ISP 拥有路由器与因特网相连（一般都用高速专线），并且和电信公司的电话交换机有专线相连。用户在某一个 ISP 缴费注册后（有的 ISP 是出售上网卡，有的自动将上网和市话的费用计入用户的住宅电话费里），即可用家中的计算机通过调制解调器、电话线接入

到该 ISP (例如，用计算机拨号码为 163 或 169 的 ISP)。ISP 在收到用户的接入呼叫后，就分配给该用户一个临时的 IP 地址（IP 地址将在第 6 章中详细讨论）。一个 ISP 拥有很多的调制解调器，并申请得到了很多个可供分配的 IP 地址，使许多用户能够同时拨通该 ISP 接入到因特网。用户拨通 ISP 后，经过 ISP 的识别用户名和口令的过程后，就获得了一个临时的 IP 地址，使其计算机成为接在因特网上的主机，这样就可以使用因特网所提供的各种服务。当用户结束通信并发出释放连接的请求时，ISP 就将刚才分配给该用户的 IP 地址收回，以便再分配给下次拨号入网的其他用户使用。

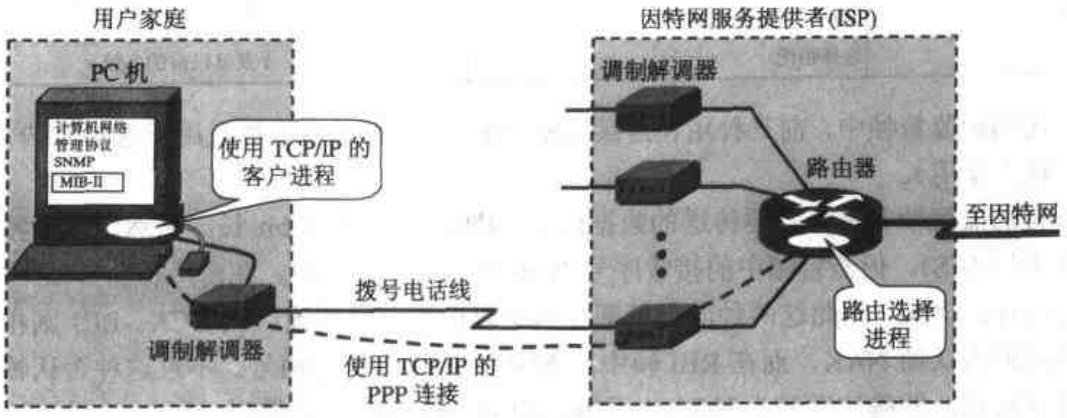


图 3-12　PC 机用拨号方式入网

早在 1984 年因特网就已经开始使用一个简单的**面向字符协议 SLIP** (Serial Line Internet Protocol) [RFC 1055]。但 SLIP 协议的缺点很多，如：

（1）SLIP 没有差错检测的功能。若 SLIP 帧在传输中出了差错，只能靠高层进行纠正。

（2）通信的每一方必须事先知道对方的 IP 地址。这对拨号入网的用户很不方便。

（3）SLIP 仅支持 IP，而不支持其他的协议。

（4）SLIP 并未成为因特网的标准协议，存在多种互不兼容的版本，影响网络的互连。

为了克服 SLIP 的这些缺点，1992 年制订了 PPP 协议。经过 1993 年和 1994 年的修订，现在的 PPP 协议已成为因特网的正式标准[RFC 1661]。PPP 协议有三个组成部分：

（1）一个将 IP 数据报封装到串行链路的方法。PPP 既支持异步链路（无奇偶检验的 8 比特数据），也支持面向比特的同步链路。IP 数据报在 PPP 帧中就是其信息部分。这个信息部分的长度受**最大接收单元 MRU** (Maximum Receive Unit)的限制。MRU 的默认值是 1500 字节。

（2）一个用来建立、配置和测试数据链路连接的**链路控制协议 LCP** (Link Control Protocol)。通信的双方可协商一些选项。在[RFC 1661]中定义了 11 种类型的 LCP 分组。

（3）一**套网络控制协议 NCP** (Network Control Protocol)[①]，其中的每一个协议支持不同的网络层协议，如 IP, OSI 的网络层，DECnet, 以及 AppleTalk 等。

3.6.2　PPP 协议的帧格式

PPP 的帧格式和 HDLC 的相似（见图 3-13）。PPP 帧的前 3 个字段和最后两个字段和

① 注：TCP 的早期版本也叫做 NCP, 但它和这里所讨论的 NCP 没有关系。

HDLC 的格式是一样的。标志字段 F 仍为 0x7E (符号“0x”表示它后面的字符是用十六进制表示的。十六进制的 7E 的二进制表示是 01111110)。地址字段 A 只置为 0xFF (即 11111111)，表示所有的站都接收这个帧。因为 PPP 只用于点对点链路，地址字段实际上并不起作用。控制字段 C 通常置为 0x03(即 00000011)。这表示 PPP 帧不使用编号（若与前面的图 3-11 比较就可发现，PPP 的控制字段和 HDLC 的无编号帧 U 的控制字段一样，即控制字段的最低两位都是 1，不过在图 3-11 中，最低位是画在最左边)。

PPP 不是面向比特而是面向字节的，因而所有的 PPP 帧的长度都是整数个字节。

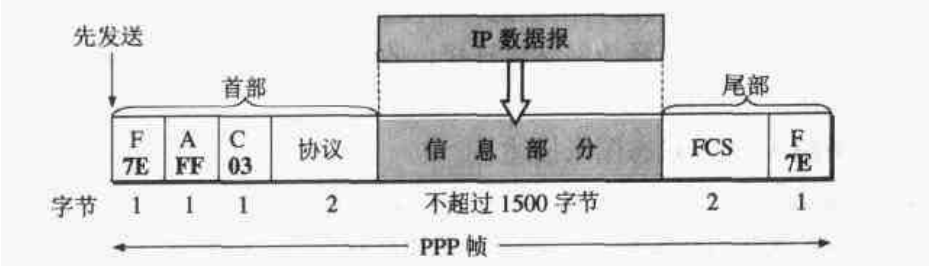


图 3-13 PPP 帧的格式

PPP 与 HDLC 不同的是多了一个 2 个字节的协议字段。当协议字段为 0x0021 时，PPP 帧的信息字段就是 IP 数据报。若为 0xC021，则信息字段是 PPP 链路控制数据，而 0x8021 表示这是网络控制数据。在[RFC 1700]中定义了 PPP 使用的协议字段的代码。

当信息字段中出现和标志字段一样的比特(0x7E)组合时，就必须采取一些措施使这种如同标志字段一样的比特组合不出现在信息字段中。当 PPP 用在同步传输链路时，协议规定**采用硬件来完成比特填充**（和 HDLC 的做法一样）。但当 PPP 用在异步传输时，它就使用一种**特殊的字符填充法**。具体的做法是将信息字段中出现的每一个 0x7E 字节转变成为 2 字节序列(0x7D, 0x5E)。若信息字段中出现一个 0x7D 的字节，则将其转变成为 2 字节序列(0x7D, 0x5D)。若信息字段中出现 ASCII 码的控制字符（即数值小于 0x20 的字符），则在该字符前面要加入一个 0x7D 字节，同时将该字符的编码加以改变。例如，0x03 (在控制字符中是“传输结束”ETX)就要变为 0x31。这些在 RFC 1662 中均有详细的规定。这样做的目的是防止这些表面上的 ASCII 码控制符（在被传输的数据中当然已不是控制符了）被错误地解释为控制符。

在 RFC 1661 定义的 PPP 不提供使用序号和确认的可靠传输。所谓“可靠传输”是指所传送的帧“无差错”、“不丢失”和“不重复”。要做到这一点，就应当在数据链路层中使用序号和确认机制，例如，像 HDLC 那样。PPP 协议之所以不使用序号和确认机制是出于以下的考虑：

第一，若使用能够实现可靠传输的数据链路层协议(如 HDLC)，开销就要增大。**在数据链路层出现差错的概率不大时，使用比较简单的 PPP 协议较为合理。**

第二，在因特网环境下，PPP 的信息字段放入的数据是 IP 数据报。假定我们采用了能实现可靠传输但十分复杂的数据链路层协议，然而当数据帧在路由器中从数据链路层上升到网络层后，仍有可能因网络拥塞而被丢弃（IP 层提供的是“尽最大努力”的交付)。因此，**数据链路层的可靠传输并不能够保证网络层的传输也是可靠的。**

第三，PPP 协议在帧格式中有帧检验序列 FCS 字段。对每一个收到的帧，PPP 都要使用硬件进行 CRC 检验。若发现有差错，则丢弃该帧（一定不能把有差错的帧交付给上一层)。端到端的差错检测最后由高层协议负责。因此，PPP 协议可保证**无差错接受**。

在噪声较大的环境下，如无线网络，则应使用有编号的工作方式，这样就可以提供可靠传输服务。这种工作方式定义在[RFC 1663]中，这里不再讨论。

3.6.3 PPP 协议的工作状态

当用户拨号接入 ISP 时，路由器的调制解调器对拨号做出确认，并建立一条物理连接。这时，PC 机向路由器发送一系列的 LCP 分组（封装成多个 PPP 帧）。这些分组及其响应选择了将要使用的一些 PPP 参数。接着就进行网络层配置，NCP 给新接入的 PC 机分配一个临时的 IP 地址。这样，PC 机就成为因特网上的一个主机了。

当用户通信完毕时，NCP 释放网络层连接，收回原来分配出去的 IP 地址。接着，LCP 释放数据链路层连接。最后释放的是物理层的连接。

上述过程可用图 3-14 的状态图来描述。

PPP 链路的起始和终止状态永远是图 3-14 中的“**静止状态**”，这时并不存在物理层的连接。当检测到调制解调器的载波信号，并建立物理层连接后，PPP 就进入链路的“**建立状态**”。这时 LCP 开始协商一些**配置选项**，即发送 LCP 的**配置请求帧**(configure-request)。这是个 PPP 帧，其协议字段置为 LCP，而信息字段包含特定的配置请求。链路的另一端可以发送以下几种响应：

（1）**配置确认帧**(configure-ack)：所有选项都接受。

（2）**配置否认帧**(configure-nac)：所有选项都理解但不能接受。

（3）**配置拒绝帧**(configure-reject)：选项有的无法识别或不能接受，需要协商。

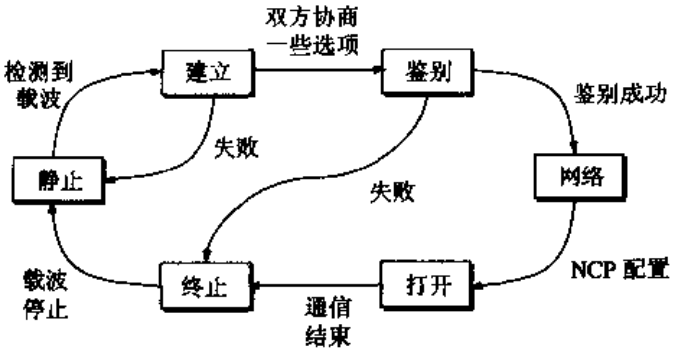


图 3-14　使用 PPP 协议的状态图

LCP 配置选项包括链路上的最大帧长、所使用的**鉴别协议**(authentication protocol)的规约（如果有的话），以及不使用 PPP 帧中的地址和控制字段（因为这两个字段的值是固定的，没有任何信息量，可以跳过去不用）。

协商结束后就进入“**鉴别状态**”。若通信的双方鉴别身份成功，则进入“**网络状态**”。这就是 PPP 链路的两端互相交换网络层特定的网络控制分组。如果在 PPP 链路上运行的是 IP 协议，则使用 IP **控制协议** IPCP (IP Control Protocol)来对 PPP 链路的每一端配置 IP 协议模块（如分配 IP 地址）。和 LCP 分组封装成 PPP 帧一样，IPCP 分组也封装成 PPP 帧（其中的协议字段为 0x8201）在 PPP 链路上传送。当网络层配置完毕后，链路就进入可进行数据通信的“**打开状态**”。两个 PPP 端点还可发送**回送请求** LCP 分组(echo-request)和**回送回答** LCP 分组(echo-reply)以检查链路的状态。数据传输结束后，链路的一端发出**终止请求** LCP 分组(terminate-request)请求终止链路连接，而当收到对方发来的**终止确认** LCP 分组

(terminate-ack)后，就转到“终止状态”。当载波停止后则回到“静止状态”。

习题

3-01 数据链路（即逻辑链路）与链路（即物理链路）有何区别？“电路接通了”与“数据链路接通了”的区别何在？

3-02 数据链路层中的链路控制包括哪些功能？

3-03 考察停止等待协议算法。在接收结点，当执行步骤(4)时，若将“否则转到(7)”改为“否则转到(8)”，将产生什么结果？

3-04 试导出公式(3-5)。

3-05 试导出停止等待协议的信道利用率公式。

3-06 信道速率为 4 kb/s。采用停止等待协议。传播时延 $t_p = 20$ ms。确认帧长度和处理时间均可忽略。问帧长为多少才能使信道利用率达到至少 50%？

3-07 在停止等待协议中，确认帧是否需要序号？请说明理由。

3-08 试写出连续 ARQ 协议的算法。

3-09 试证明：当用 n 个比特进行编号时，若接收窗口的大小为 1，则只有在发送窗口的大小 $W_T \leq 2^n - 1$ 时，连续 ARQ 协议才能正确运行。

3-10 试证明：对于选择重传 ARQ 协议，若用 n 比特进行编号，则接收窗口的最大值受公式(3-18)的约束。

3-11 在选择重传 ARQ 协议中，设编号用 3 bit。再设发送窗口 $W_T = 6$ 而接收窗口 $W_R = 3$。试找出一种情况，使得在此情况下协议不能正确工作。

3-12 在连续 ARQ 协议中，设编号用 3 bit，而发送窗口 $W_T = 8$。试找出一种情况，使得在此情况下协议不能正确工作。

3-13 在什么条件下，选择重传 ARQ 协议和连续 ARQ 协议在效果上完全一致？

3-14 在连续 ARQ 协议中，若 $W_T = 7$，则发送端在开始时可连续发送 7 个数据帧。因此，在每一帧发出后，都要置一个超时计时器。现在计算机里只有一个硬时钟。设这 7 个数据帧发出的时间分别为 $t_0, t_1, ..., t_6$，且 t_{out} 都一样大。试问如何实现这 7 个超时计时器（这叫软时钟法）？

3-15 卫星信道的数据率为 1 Mb/s。取卫星信道的单程传播时延为 0.25 秒。每一个数据帧长都是 2000 bit。忽略误码率、确认帧长和处理时间。试计算下列情况下的信道利用率：

（1）停止等待协议。

（2）连续 ARQ 协议，$W_T = 7$。

（3）连续 ARQ 协议，$W_T = 127$。

（4）连续 ARQ 协议，$W_T = 255$。

3-16 试简述 HDLC 帧各字段的意义。HDLC 用什么方法保证数据的透明传输？

3-17 HDLC 帧可分为哪几个大类？试简述各类帧的作用。

3-18 HDLC 规定，接收序号 N(R)表示序号为[N(R)– 1] (mod 8)的帧以及在这以前的各帧都已正确无误地收妥了。为什么不定义“N(R)表示序号为 N(R) (mod 8)的帧以及在这以前的各帧都已正确无误地收妥了”？

3-19 PPP 协议的主要特点是什么？为什么 PPP 不使用帧的编号？PPP 适用于什么情况？

3-20 要发送的数据为 1101011011。采用 CRC 的生成多项式是 $P(X) = X^4 + X + 1$。试求应添加在数据后面的余数。

数据在传输过程中最后一个 1 变成了 0，问接收端能否发现？

若数据在传输过程中最后两个 1 都变成了 0，问接收端能否发现？

3-21 一个 PPP 帧的数据部分（用十六进制写出）是 7D 5E FE 27 7D 5D 7D 5D 65 7D 5E。试问真正的数据是什么（用十六进制写出）？

3-22 有两条长度均为 1000 km 的链路 AB 和 BC。现在从 A 用停止等待协议向 C 发送数据，中途经过 B 转发。链路带宽为 1.5 Mb/s，链路的误码率 $p = 10^{-6}$。链路只允许传送长度不超过 2 KB 的帧。每一个帧的首部和尾部的开销为 32 字节。信号在链路上的传播速率为 2×10^5 km/s。试计算从 A 成功发送长度为 64 KB 的数据所需的平均时间。忽略结点对数据的处理时间。忽略所有确认帧的处理时间和发送时间，并认为确认帧不会出错。

计算出的时间比直接向链路发送 64 KB 数据需要的发送时间大多少？

3-23 有一比特串 0110111111111100 用 HDLC 协议传送。经过零比特填充后变成怎样的比特串？若接收端收到的 HDLC 帧的数据部分是 000111011111010111110110，问删除发送端加入的零比特后变成怎样的比特串？

第 4 章　局　域　网

局域网是计算机网络的重要组成部分。本章重点介绍最常用的局域网——以太网。在简要给出局域网的概念后，从传统总线式以太网入手，详细讨论以太网使用的 CSMA/CD 协议和 MAC 帧的结构。对其中的 MAC 地址进行比较深入的讨论。接着介绍使用集线器的以太网和使用网桥及以太网交换机对以太网进行扩展。还对快速以太网(100Mb/s)和吉比特以太网以及 10 吉比特以太网的基本特点进行了讨论。在本章的最后是无线局域网，它已引起人们日益增长的关注。

4.1　局域网概述

自 20 世纪 70 年代末，微型计算机由于价格不断下降而获得了日益广泛的使用，这就促使计算机局域网技术得到了飞速发展，并在计算机网络中占有非常重要的地位。

局域网最主要的特点是：**网络为一个单位所拥有，且地理范围和站点数目均有限**。在局域网刚刚出现时，局域网比广域网具有较高的数据率、较低的时延和较小的误码率。但随着光纤技术在广域网中普遍使用，现在广域网也具有很高的数据率和很低的误码率。

一个工作在多用户系统下的小型计算机，也基本上可以完成局域网所能做的工作。二者相比，局域网具有如下的一些主要优点：

（1）能方便地共享昂贵的外部设备、主机以及软件、数据。从一个站点可访问全网。

（2）便于系统的扩展和逐渐地演变，各设备的位置可灵活调整和改变。

（3）提高了系统的可靠性、可用性和残存性。

局域网可按网络拓扑进行分类。图 4-1(a)是**星形网**。由于**集线器**(hub)的出现和双绞线大量用于局域网中，星形以太网以及多级星形结构的以太网获得了非常广泛的应用。图 4-1(b)是**环形网**，最典型的就是**令牌环形网**(token ring)，它又称为令牌环。图 4-1(c)为**总线网**，各站直接连在总线上。总线网可使用两种协议。一种是传统以太网使用的 CSMA/CD，而另一种是令牌传递总线网，即物理上是总线网而逻辑上是令牌环形网。前一种总线网现在已演进为星形网，而后一种令牌传递总线网早已退出了市场。图 4-1(d)是**树形网**，它是总线网的变型，都属于使用广播信道的网络，但这主要用于频分复用的宽带局域网。局域网经过了近三十年

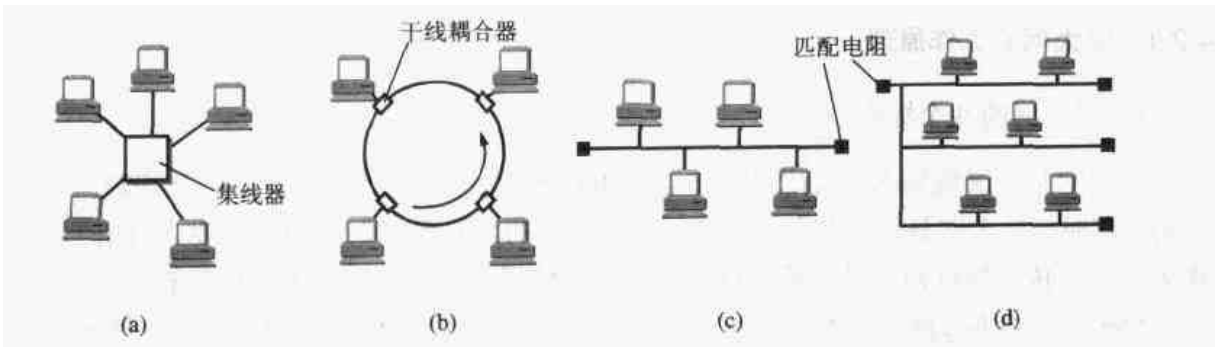


图 4-1 局域网的拓扑：(a)星形网；(b)环形网；(c)总线网；(d)树形网

的发展，尤其是在快速以太网(100 Mb/s)和吉比特以太网(1 Gb/s)、10 吉比特以太网(10 Gb/s)进入市场后，以太网已经在局域网市场中占据了绝对优势。现在以太网几乎成为了局域网的同义词，因此本章主要讨论以太网技术。

局域网可使用多种传输媒体。双绞线最便宜，原来只用于低速(1～2 Mb/s)基带局域网。现在 10 Mb/s 甚至 100 Mb/s 乃至 1Gb/s 的局域网也可使用双绞线。双绞线已成为局域网中的主流传输媒体。50 Ω同轴电缆可用到 10 Mb/s，而 75 Ω同轴电缆可用到几百 Mb/s。光纤具有很好的抗电磁干扰特性和很宽的频带，主要用在环形网中，其数据率可达 100 Mb/s 甚至达到 10 Gb/s。现在技术发展很快，点到点线路使用光纤也已相当普遍。

局域网要着重考虑的一个问题就是如何使众多用户能够合理而方便地共享通信媒体资源。媒体共享技术大体上可分为两大类：

（1）**静态划分信道**，如在第 2 章中已经介绍过的频分复用、时分复用、波分复用和码分复用等。用户只要得到了信道就不会和别的用户发生冲突。但这种划分信道的方法代价较高，不适合于局域网和某些广播信道的网络使用。

（2）**动态媒体接入控制**，它又称为**多点接入**(multiple access)，其特点是信道并非在用户通信时固定分配给用户。这里又分为以下两类：

- **随机接入**　随机接入的特点是所有的用户可随机地发送信息。但如果恰巧有两个或更多的用户在同一时刻发送信息，那么在共享媒体上就要产生**碰撞**（即发生了冲突），使得这些用户的发送都失败。因此，必须有解决碰撞的网络协议。
- **受控接入**　受控接入的特点是用户不能随机地发送信息而必须服从一定的控制。这类的典型代表有分散控制的令牌环局域网和集中控制的多点线路**探询**(polling)，或称为**轮询**。

属于随机接入的以太网将在本章重点讨论。在卫星通信中常使用的随机接入的 ALOHA 系统在本章的附录 B 中介绍，供有兴趣的读者参考。属于受控接入的分散控制的光纤分布式数据接口 FDDI 将在 4.7.2 节中进行简单的介绍。探询技术则由于目前使用得较少，本书不再讨论。

4.2　传统以太网

由于现在以太网的数据率已演进到每秒百兆比特、吉比特甚至 10 吉比特，因此通常就用“**传统以太网**”来表示最早进入市场的 10 Mb/s 速率的以太网。下面我们从传统以太网入手来讨论以太网的基本原理。

4.2.1　以太网的工作原理

1．以太网的两个标准

以太网是美国施乐(Xerox)公司的 Palo Alto 研究中心（简称为 PARC）于 1975 年研制成功的。那时，以太网是一种基带总线局域网，当时的数据率为 2.94 Mb/s。以太网用无源电缆作为总线来传送数据帧，并以曾经在历史上表示传播电磁波的**以太**(Ether)来命名。1976 年 7 月，Metcalfe 和 Boggs 发表他们的以太网里程碑论文[METC76]。1980 年 9 月，DEC 公司、英特尔(Intel)公司和施乐公司联合提出了 10 Mb/s 以太网规约的第一个版本 DIX V1(DIX 是这

三个公司名称的缩写）。1982 年又修改为第二版规约（实际上也就是最后的版本），即 DIX Ethernet V2，成为世界上第一个局域网产品的规约。

在此基础上，IEEE 802 委员会[①]的 802 工作组于 1983 年制订了第一个 IEEE 的以太网标准，其编号为 802.3，数据率为 10Mb/s。802.3 局域网对以太网标准中的帧格式作了很小的一点更动，但允许基于这两种标准的硬件实现可以在同一个局域网上互操作。以太网的两个标准 DIX Ethernet V2 与 IEEE 的 802.3 标准只有很小的差别，因此很多人也常将 802.3 局域网简称为“以太网”（本书也经常不严格区分它们，虽然严格说来，“以太网”应当是指符合 DIX Ethernet V2 标准的局域网）。

出于厂商们在商业上的激烈竞争，IEEE 的 802 委员会未能形成一个统一的、“最佳的”局域网标准，而是被迫制订了几个不同的局域网标准，如 802.4 令牌总线网、802.5 令牌环网等。为了使数据链路层能更好地适应多种局域网标准，802 委员会就将局域网的数据链路层拆成两个子层，即**逻辑链路控制** LLC (Logical Link Control)子层和**媒体接入控制** MAC (Medium Access Control)子层。与接入到传输媒体有关的内容都放在 MAC 子层，而 LLC 子层则与传输媒体无关，不管采用何种协议的局域网对 LLC 子层来说都是透明的（见图 4-2 所示）。

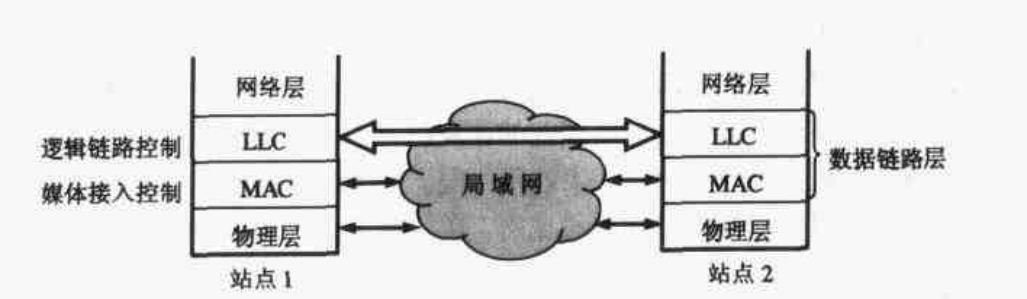


图 4-2　局域网对 LLC 子层是透明的

然而到了 20 世纪 90 年代后，激烈竞争的局域网市场逐渐明朗。以太网在局域网市场中已取得了垄断地位，并且几乎成为了局域网的代名词。由于因特网发展很快而 TCP/IP 体系经常使用的局域网是 DIX Ethernet V2 而不是 802.3 标准中的几种局域网，因此现在 802 委员会制定的逻辑链路控制子层 LLC（即 802.2 标准）的作用已经不大了，很多厂商生产的网卡上就仅装有 MAC 协议而没有 LLC 协议。本章在介绍以太网时一般都不考虑 LLC 子层。这样对以太网工作原理的讨论会更加简洁。

2. 网卡的作用

首先我们从一般的概念上讨论一下计算机是怎样连接到局域网上的。

计算机与外界局域网的连接是通过主机箱内插入一块网络接口板（或者是在笔记本电脑中插入一块 PCMCIA 卡）。网络接口板又称为通信适配器(adapter)或**网络接口卡 NIC** (Network Interface Card)，但现在更多的人愿意使用更为简单的名称“**网卡**”。网卡上面装有处理器和存储器（包括 RAM 和 ROM）。网卡和局域网之间的通信是通过电缆或双绞线以串行传输方

① 注：IEEE 802 委员会是专门制定局域网和城域网标准的机构。目前其下属的活跃工作组只有六个，即 802.1：桥接/体系结构；802.3：CSMA/CD；802.11：无线局域网；802.15：无线个人网；802.16：宽带无线接入；802.17：弹性分组环（Resilient Packet Ring）。其余的都已经暂时或完全停止了活动。所有 802 标准都可以免费从因特网上下载[W-IEEE802]。

式进行的。而网卡和计算机之间的通信则是通过计算机主板上的 I/O 总线以并行传输方式进行的（见图 4-3）。因此，网卡的一个重要功能就是要进行串行/并行转换。由于网络上的数据率和计算机总线上的数据率并不相同，因此在网卡中必须装有对数据进行缓存的存储芯片。在安装网卡时必须将管理网卡的设备驱动程序安装在计算机的操作系统中。这个驱动程序以后就会告诉网卡，应当从存储器的什么位置将多大的数据块发送到局域网，或者应当在存储器的什么位置上将局域网传送过来的数据块存储下来。网卡还要能够实现以太网协议。

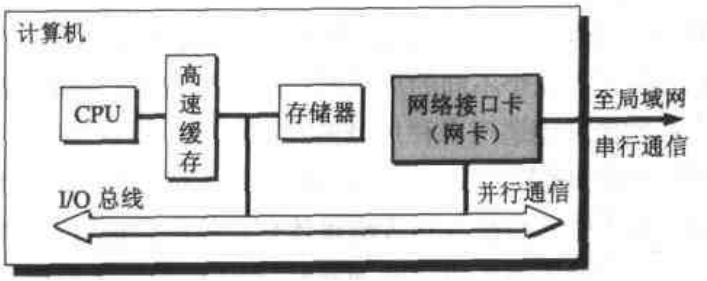


图 4-3　计算机通过网卡和局域网进行通信

网卡不是独立的自治单元，因为网卡本身不带电源而是必须使用所插入的计算机的电源，并受该计算机的控制。因此网卡可看成为一个**半自治单元**。当网卡收到一个有差错的帧时，它就将这个帧丢弃而不必通知它所插入的计算机。当网卡收到一个正确的帧时，它就使用中断来通知该计算机并交付给协议栈中的网络层。当计算机要发送一个 IP 数据报时，就由协议栈向下交给网卡组装成帧后发送到局域网。关于网卡后面还要讨论。

3．CSMA/CD 协议

当初提出以太网的方案是基于下面的思路：要寻找很简单的方法将一些相距较近的计算机互相连接起来，使它们可以很方便和很可靠地进行较高速率的数据通信。

最初的以太网是将许多计算机都连接到一根总线上。当初认为这样的连接方法既简单又可靠，因为总线上没有有源器件。在那个时代普遍认为有源器件不可靠，而无源的电缆线才是最可靠的。

总线的特点是：当一台计算机发送数据时，总线上的所有计算机都能检测到这个数据。这种通信方式是广播通信。但我们并不总是希望使用广播通信。为了在总线上实现一对一的通信，可以使每一台计算机拥有一个与其他计算机都不同的地址。在发送数据帧时，在帧的首部写明接收站的地址。现在的电子技术可以很容易做到：仅当数据帧中的目的地址与计算机的地址一致时，该计算机才能接收这个数据帧。计算机对不是发送给自己的数据帧，则一律不接收（即丢弃）。

图 4-4 是上述概念的示意图。设现在计算机 B 向 D 发送数据。总线上的每一个工作的计算机都能检测到 B 发送的数据信号。但由于只有计算机 D 的地址与数据帧首部写入的地址一致，因此只有 D 才接收这个数据帧，而其他所有的计算机(A, C 和 E)都检测到不是发送给它们的数据帧，因此它们就丢弃这个数据帧而不能够收下来。这样，具有广播特性的总线上就实现了一对一的通信。总线两端的匹配电阻是为了吸收在总线上传播的电磁波信号的能量，避免在总线上产生有害的电磁波反射。

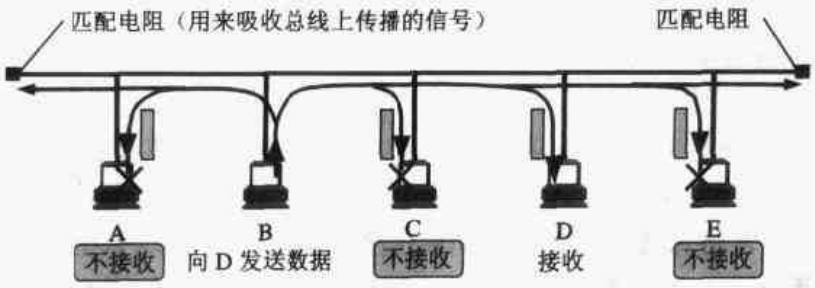


图 4-4　B 向 D 发送数据，在局域网上只有计算机 D 才能接收此数据

人们也常把局域网上的计算机称为“**主机**”、“**工作站**”、“**站点**”或“**站**”。

为了通信的简便，以太网采取了两种重要的措施：

第一、采用较为灵活的**无连接**的工作方式，即不必先建立连接就可以直接发送数据。

第二、以太网对发送的数据帧**不进行编号，也不要求对方发回确认**。这样做的理由是局域网信道的质量很好，因信道质量产生差错的概率是很小的。

因此，**以太网提供的服务是不可靠的交付，即尽最大努力的交付**。当目的站收到有差错的数据帧时（例如，用 CRC 查出有差错），就丢弃此帧，其他什么也不做。**差错的纠正由高层来决定**。例如，如果高层使用 TCP 协议，那么 TCP 就会发现丢失了一些数据。于是经过一定的时间后，TCP 就将这些数据重新传递给以太网进行重传。**但以太网并不知道这是一个重传的帧，而是当作一个新的数据帧来发送**。

剩下的一个重要问题就是如何协调总线上各计算机的工作。我们知道，总线上只要有一台计算机在发送数据，总线的传输资源就被占用。因此，**在同一时间只能允许一台计算机发送信息**，否则各计算机之间就会互相干扰，结果大家都无法正常发送数据。

以太网采用的协调方法是使用一种特殊的协议，即**载波监听多点接入/碰撞检测** CSMA/CD (Carrier Sense Multiple Access with Collision Detection)。下面是 CSMA/CD 的要点。

“**多点接入**”就是说明这是总线型网络，许多计算机以多点接入的方式连接在一根总线上。协议的实质是“载波监听”和“碰撞检测”。

“**载波监听**”是指每一个站在发送数据之前先要检测一下总线上是否有其他计算机在发送数据，如果有，则暂时不要发送数据，以免发生碰撞。

这里要指出，以太网标准规定各计算机发送的数据都使用曼彻斯特编码的信号（在第 2 章已讲过，使用曼彻斯特编码是为了便于接收端提取位同步信号）。因此总线上并没有什么“载波”。因此，“载波监听”就是用电子技术检测总线上有没有其他计算机发送的数据信号。

“**碰撞检测**”就是计算机边发送数据边检测信道上的信号电压大小。当几个站同时在总线上发送数据时，总线上的信号电压摆动值将会增大（互相叠加）。当一个站检测到的信号电压摆动值超过一定的门限值时，就认为总线上至少有两个站同时在发送数据，表明产生了碰撞。所谓“碰撞”就是发生了冲突。因此“碰撞检测”也称为“**冲突检测**”。在发生碰撞时，总线上传输的信号产生了严重的失真，无法从中恢复出有用的信息来。因此，每一个正在发送数据的站，一旦发现总线上出现了碰撞，就要立即停止发送，免得继续浪费网络资源，然后等待一段随机时间后再次发送。

既然每一个站在发送数据之前已经监听到信道为“**空闲**”，那么为什么还会出现数据在总线上的碰撞呢？这是因为电磁波在总线上总是以有限的速率传播的。因此当某个站监听到

总线是空闲时，也可能总线并非真正是空闲的。图 4-5 所示的例子可以说明这种情况。设图中的局域网两端的站 A 和 B 相距 1 km，用同轴电缆相连。**电磁波在 1 km 电缆的传播时延约为 5μs**（这个数字应当记住）。因此，A 向 B 发出的信息，在约 5μs 后才能传送到 B。换言之，B 若在 A 发送的信息到达 B 之前发送自己的帧（因为这时 B 的载波监听检测不到 A 所发送的信息），则必然要在某个时间和 A 发送的帧发生碰撞。碰撞的结果是两个帧都变得无用。在局域网的分析中，常将总线上的**单程端到端传播时延**记为 τ。发送数据的站希望尽早知道是否发生了碰撞。那么，A 发送数据后，**最迟要经过多长时间才能知道自己发送的数据和其他站发送的数据有没有发生碰撞？**从图 4-5 不难看出，这个时间最多是**两倍的总线端到端的传播时延**(2τ)，或总线的**端到端往返传播时延**。由于局域网上任意两个站之间的传播时延有长有短，因此局域网按最坏情况设计，即取总线两端的两个站之间的传播时延（这两个站之间的距离最大）为端到端传播时延。

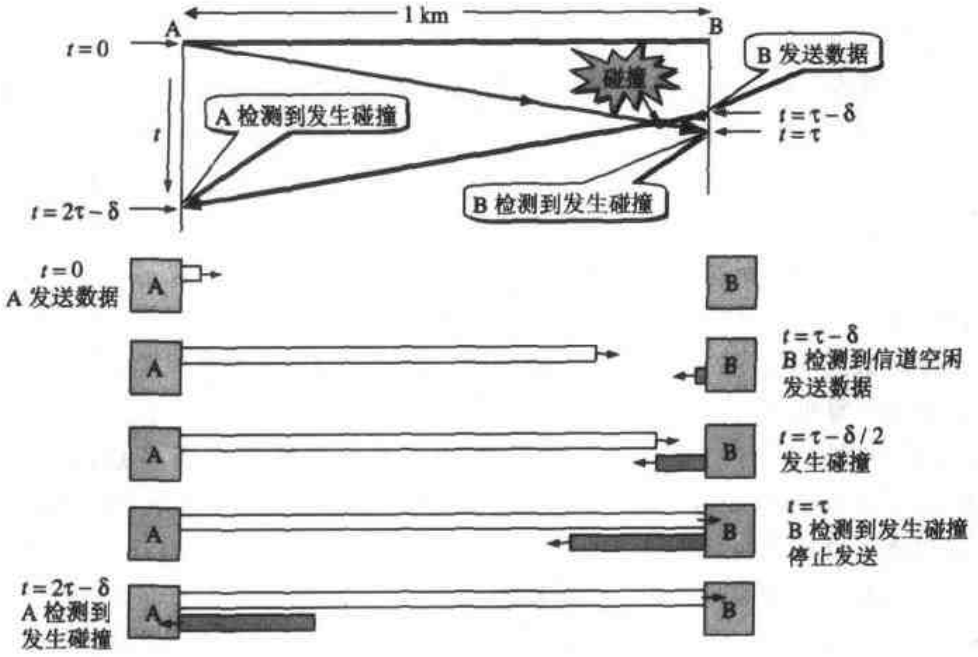


图 4-5　传播时延对载波监听的影响

显然，在使用 CSMA/CD 协议时，一个站**不可能同时进行发送和接收**。因此使用 CSMA/CD 协议的以太网不可能进行全双工通信而只能进行**双向交替通信（半双工通信）**。

下面是图 4-5 中的一些重要的时刻。

在 $t = 0$ 时，A 发送数据。B 检测到信道为空闲。

在 $t = \tau - \delta$ 时，A 发送的数据还没有到达 B 时，由于 B 检测到信道是空闲，因此 B 发送数据。

经过时间 $\delta / 2$ 后，即在 $t = \tau - \delta / 2$ 时，A 发送的数据和 B 发送的数据发生了碰撞。

在 $t = \tau$ 时，B 检测到发生了碰撞，于是停止发送数据。

在 $t = 2\tau - \delta$ 时，A 也检测到发生了碰撞，因而也停止发送数据。

A 和 B 发送数据均失败，它们都要推迟一段时间再重新发送。

由此可见，**每一个站在自己发送数据之后的一小段时间内，存在着遭遇碰撞的可能性。**这一小段时间是**不确定的**，它取决于另一个发送数据的站到本站的距离。如果发生了碰撞，

发送数据的站就必须推迟一段时间重新发送。因此，以太网**不能保证**一定在某一时间之内能够将自己的数据帧成功地发送出去（因为还不知道会不会产生碰撞）。以太网的这一特点称为**发送的不确定性**。如果希望在以太网上发生碰撞的机会很小，必须使整个以太网的平均通信量远小于以太网的最高数据率。

4. 争用期

从图 4-5 可看出，最先发送数据帧的 A 站，在发送数据帧后至多经过时间 2τ 就可知道所发送的数据帧是否遭受了碰撞。这就是 $\delta \to 0$ 的情况。因此以太网的端到端往返时延 2τ 称为**争用期**(contention period)，它是一个很重要的参数。争用期又称为**碰撞窗口**(collision window)。这是因为一个站在发送完数据后，只有通过争用期的“考验”，即**经过争用期这段时间还没有检测到碰撞，才能肯定这次发送不会发生碰撞**。

现在考虑一种情况。当某个站正在发送数据时，有另外两个站有数据要发送。这两个站进行载波监听，发现总线忙，于是就等待。当它们发现总线变为空闲时，就立即发送自己的数据。但这必然再次产生碰撞。经碰撞检测发现了碰撞，就停止发送。然后再重新发送。……这样下去，一直不能发送成功。因此必须设法解决这个问题。

以太网使用**截断二进制指数类型**(truncated binary exponential type)的**退避算法**来解决这一问题。截断二进制指数类型退避算法很简单，就是让发生碰撞的站在停止发送数据后，不是立即再发送数据，而是推迟（这叫作**退避**）一个随机的时间。这样做是为了使重传时再次发生冲突的概率减小。具体做法是：

（1）确定基本退避时间，一般是取为争用期 2τ。

（2）定义参数 k，它等于重传次数，但 k 不超过 10。因此，$k = \text{Min}[\text{重传次数}, 10]$。

（3）从离散的整数集合$[0,1,\cdots,(2^k-1)]$中随机地取出一个数，记为 r。重传所需的时延就是 r 倍的基本退避时间。

（4）当重传达 16 次仍不能成功时（这表明同时打算发送数据的站太多，以致连续发生冲突），则丢弃该帧，并向高层报告。

例如，在第 1 次重传时，$k=1$，$r=0$ 或 1。因此重传的站可选择的重传推迟时间是 0 或 2τ，在这两个时间中随机选择一个。若再发生碰撞，则在第 2 次重传时，$k=2$，$r=0, 1, 2, 3$。因此重传推迟的时间是在 0, 2τ, 4τ和 6τ这 4 个数之间随机地选取一个。依此类推。若连续多次发生冲突，就表明可能有较多的站参与争用信道，因此，各站应在更大的整数集合中随机选择自己的退避时间，这样才能减小再次冲突的概率。

使用上述退避算法可使重传需要推迟的平均时间随重传次数而增大（这也称为**动态退避**），有利于整个系统的稳定。

以太网取 51.2 μs 为争用期的长度。对于 10 Mb/s 以太网，在争用期内可发送 512 bit，即 64 字节。因此以太网在发送数据时，如果前 64 字节没有发生冲突，那么后续的数据就不会发生冲突，以太网就认为这个数据帧的发送是成功的。换句话说，如果发生冲突，就一定是在发送的前 64 字节之内。由于一检测到冲突就立即中止发送，这时已经发送出去的数据一定小于 64 字节，因此以太网规定了**最短有效帧长**为 64 字节，**凡长度小于 64 字节的帧都是由于冲突而异常中止的无效帧**。

需要指出，以太网的端到端时延实际上是小于争用期的一半(即 25.6 μs)。争用期被规定为 51.2 μs，不仅是考虑了以太网的端到端时延，而且还包括其他的许多因素，如可能存在的

转发器所增加的时延，以及强化碰撞（见后面）的干扰信号的持续时间等。

但以太网的这种退避算法会产生“**捕获效应**”，它引起某些站往往**不能公平地**“捕获”到总线，因而使这些站不能公平地发送数据。例如，设网上只有两个站 A 和 B，它们都有大量数据等待发送。现在 A 和 B 同时发送了数据，因而发生碰撞，双方都要推迟重传。A 和 B 都要在 $r = 0$ 或 1 中随机选择地一个数。假定 A 选择了 0 而 B 选择了 1。这样，A 马上就可进行重传，但 B 要等待。A 发送完数据后立刻可接着发送第二帧数据。这就和 B 又发生了碰撞。请注意：A 仍然是在 $r = 0$ 或 1 中随机选择地一个数，但 B 因为是再次发生碰撞，因此 B 要在 $r = 0, 1, 2, 3$ 中（共 4 个数）随机选择地一个。这样，A 先获得发送权的概率就较大。假定 A 又先获得了发送权。以后 A 再次和 B 发生碰撞。A 仍然是在 $r = 0$ 或 1 中随机选择地一个，但 B 就要在 $r = 0, 1, 2, \ldots, 7$（共 8 个数）中随机选择地一个。所以很可能 B 又比 A 晚发送数据。这种不公平现象一直继续下去，直到 A 将其发送队列中的数据全部发送完毕，或 B 重传达 16 次后将其计数器复位，重新和 A 一起公平地争用信道。

为解决这一问题，IEEE 的 802.3w 工作组在 1994 年提出了一个新的退避算法，即**二进制对数仲裁方法** BLAM (Binary Logarithmic Arbitration Method)。虽然 BLAM 改善了原来老的退避算法，但由于现在人们已将兴趣转移到全双工以太网（不使用 CSMA/CD），因此 BLAM 并没有成为以太网的标准。

以太网还采取一种叫做**强化碰撞**的措施。这就是当发送数据的站一旦发现发生了碰撞时，除了立即停止发送数据外，还要再继续发送若干比特的**人为干扰信号**(jamming signal)，以便让所有用户都知道现在已经发生了碰撞（图 4-6）。

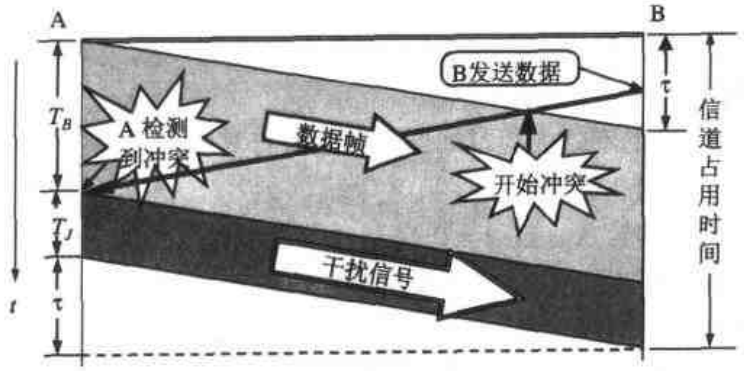


图 4-6　人为干扰信号的加入

从图 4-6 可以看出，A 站从发送数据开始到发现碰撞并停止发送的时间间隔是 T_B。A 站得知碰撞已经发生时所发送的强化碰撞的干扰信号的持续时间是 T_J。图中的 B 站在得知发生碰撞后，也要发送干扰信号，但为简单起见，图 4-6 没有画出 B 站所发送的干扰信号。发生碰撞使 A 浪费时间 $T_B + T_J$。可是整个信道被占用的时间还要增加一个单程端到端的传播时延 τ。因此总线被占用的时间是 $T_B + T_J + \tau$。

4.2.2　传统以太网的连接方法

传统以太网可使用的**传输媒体**有四种，即铜缆（粗缆或细缆）、铜线（双绞线）或光缆。这样，以太网就有四种不同的物理层。图 4-7 在 MAC 层下面给出了对应于这四种传输媒体的物理层，即 10BASE5(粗缆)、10BASE2(细缆)、10BASE-T(双绞线)和 10BASE-F(光缆)。这

里“BASE”表示电缆上的信号是基带信号，采用曼彻斯特编码（见图 2-8）。BASE 前面的数字“10”表示数据率为 10Mb/s，而后面的数字 5 或 2 表示每一段电缆的最大长度为 500 米或 200 米（实际上是 185 米）。“T”代表双绞线，而“F”代表光纤。目前使用得最广泛的是双绞线传输媒体。



图 4-7　以太网的四种不同的物理层

图 4-8 给出了用铜缆或铜线连接到以太网的示意图。图 4-8(a)是 10BASE5 以太网的连接方法。这种以太网称为**粗缆以太网**（电缆直径为 10mm，特性阻抗为 50Ω）。图 4-8(b)是细缆以太网，图 4-8(c)是使用集线器的双绞线以太网。

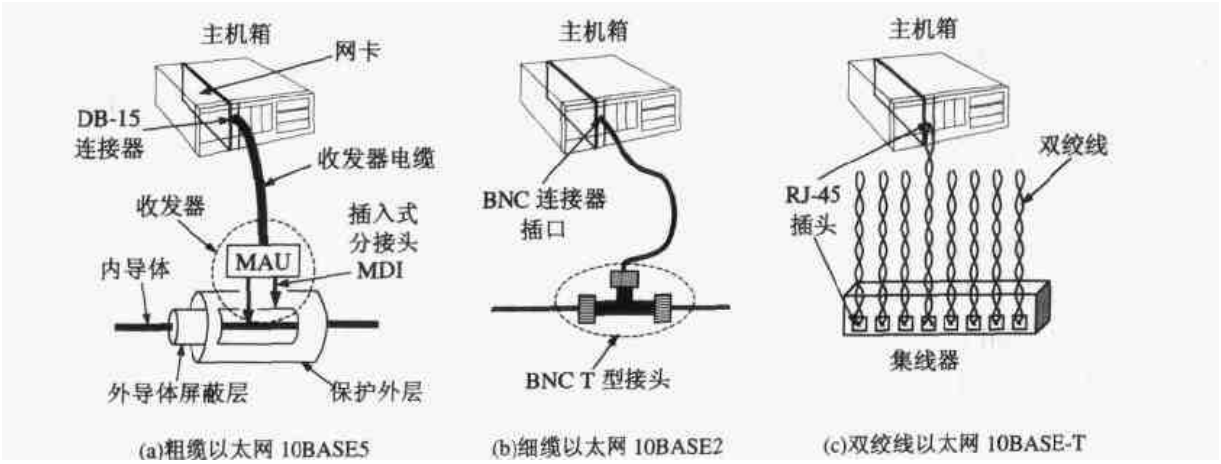


图 4-8　几种以太网的连接方法

粗缆以太网是最初使用的以太网，其网卡通过 DB-15 型**连接器**(15 针)与**收发器电缆**(transceiver cable)相连，收发器电缆的正式名称是 **AUI 电缆**。AUI 是连接单元接口(Attachment Unit Interface)的缩写。收发器电缆的另一端连接到**收发器**(transceiver)。收发器电缆的长度不能超过 50 m。收发器由两部分组成。一部分是含有电子元器件的**媒体连接单元 MAU** (Medium Attachment Unit)。另一部分是没有电子元器件的插入式分接头(tap)，称为**媒体相关接口 MDI** (Medium Dependent Interface)，它直接插入到电缆中（不用切断电缆）就能和同轴电缆的内部导线有良好的接触连接。

粗缆以太网的网卡包括了处理通信所用到的数字电路，如地址确认和差错检测。网卡还使用总线与主机交换数据，并使用中断机制来通知 CPU 其操作已经结束。这种网卡不包括模拟硬件，也不处理模拟信号。

收发器的功能有以下这样一些：

（1）从计算机经收发器电缆得到数据向同轴电缆发送，或反过来，从同轴电缆接收数据经收发器电缆送给计算机。

（2）检测在同轴电缆上发生的数据帧的碰撞。

（3）在同轴电缆和电缆接口的电子设备之间进行电气隔离。

（4）当收发器或所连接的计算机出故障时，保护同轴电缆不受其影响。

上述最后一个功能叫做**超长控制**(jabber control)。当收发器或所连接的计算机出故障时，就有可能向总线上不停地发送无规律的数据，使总线上所有的站都不能工作。为了避免这种现象，必须对所有的站发送的数据帧的长度设一上限。当检测到某个数据帧的长度超过此上限值时，即认为该站出了故障，接着就自动禁止该站向总线发送数据。

为什么以太网要限制同轴电缆的长度呢？这是因为信号沿总线传播时必然产生衰减。若总线太长，则经总线传播的信号将会衰减变得很弱，以致影响载波监听和碰撞检测的正常工作。因此，以太网所用的这种同轴电缆的最大长度被限制为 500 m。若实际网络需要跨越更长的距离，就必须采用**转发器**(repeater)将信号放大并整形后再转发出去。转发器又称为**中继器**，它工作在物理层。转发器的作用是消除信号由于经过一长段电缆而造成的失真和衰减，使信号的波形和强度达到所要求的指标。用转发器连接起来的几个网段仍然是一个局域网。

下面研究一下以太网的最大作用距离。

考虑到电缆的衰减和时延等因素，在 802.3 标准中对传输系统的要求做了详细的规定。例如，在任意两个站之间最多可以有三个同轴电缆段。所以在图 4-9 中画出了由三段同轴电缆连接成的以太网**网段**(segment)。802.3 标准还规定，接到转发器的点到点链路的总长度不能超过 1000 m。在图 4-9 中，网段 1 和网段 2 各通过一个转发器经过 750 m 的点到点链路相连，而网段 2 和网段 3 各通过一个转发器经过 250 m 的点到点链路相连。这样，三段 500 m 长的电缆，加上总长度为 1000 m 的点到点链路，共 2500 m。如果算上 6 节各 50 m 长的收发器电缆，则整个局域网的最大作用距离达到 2.8 km。为了减少两个转发器之间较长链路上带来的干扰，可采用光纤链路。相关的标准叫做**转发器间的光纤链路 FOIRL** (Fiber Optic Inter-Repeater Link)，其最大长度为 1 km，上面传送 10 Mb/s 的基带信号。与光纤相连的媒体连接单元叫做 FOMAU (Fiber Optic Medium Attactment Unit)。

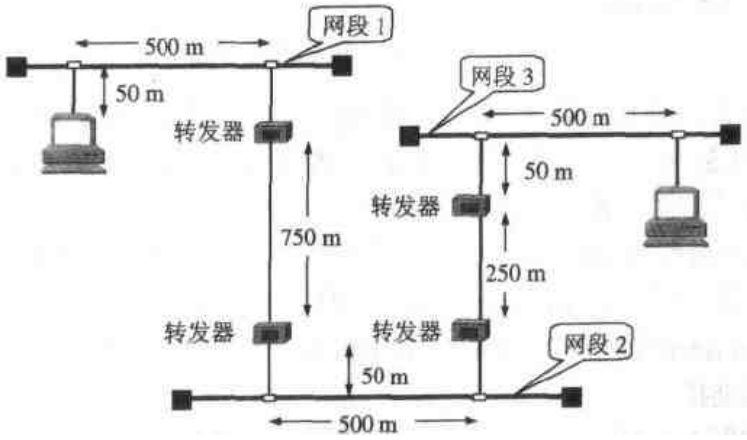


图 4-9　用转发器扩展以太网的作用距离

以太网还规定一个网上的最大站数为 1024。实际上这样大的数字是达不到的。因为按照以太网的规约，每个同轴电缆段最多只能安装 100 个站。

为了克服 10BASE5 粗缆以太网的布线很贵且安装不便的主要缺点，1985 年细缆以太网 10BASE2 的标准问世了。10BASE2 细缆以太网的特点是：

第一，采用更便宜的直径为 5 mm 的细同轴电缆（特性阻抗仍为 50 W），可代替粗同轴电缆。细缆在布线转角处易于转弯，并可直接连接到机箱。网络的每个段最长为 185 m，因

此这种细缆局域网就简记为 10BASE2 (2 表示距离约为 200 m)。

第二，将媒体连接单元 MAU 和媒体相关接口 MDI 都安装在网卡上，取消了外部的 AUI 电缆。细缆直接用标准 BNC T[①]型接头连接到网卡上的 BNC 连接器的插口。但细缆在装上 BNC 接头时必须先切断电缆，如图 4-8(b)所示。

随着集成度的提高，网卡上芯片的个数不断减少。虽然现在各厂家生产的网卡种类繁多，但其功能大同小异。网卡的功能主要有以下三个：

（1）**数据的封装与解封**　发送时将上一层交下来的数据加上首部和尾部，成为以太网的帧。接收时将以太网的帧剥去首部和尾部，然后送交上一层。

（2）**链路管理**　主要是 CSMA/CD 协议的实现。

（3）**编码与译码**　即曼彻斯特编码与译码。

人们在 10BASE2 局域网的实际使用过程中发现了它的缺点，这就是当细缆总线上某个电缆接头处发生短路或开路时，整个网络就无法工作，且确定故障点相当麻烦（尤其是当总线上的站点数很多时）。这就使得网络的可靠性很差。此外，细缆布线不够方便，价格也仍较高。**考虑到便于维护局域网**，人们又像电话网那样使用星形网拓扑，不用电缆而使用无屏蔽双绞线。每个站需要用两对双绞线，分别用于发送和接收。在星形网的中心则增加了一种可靠性非常高的设备，叫做**集线器**(hub)。双绞线以太网总是和集线器配合使用的。图 4-8(c)是使用集线器时的连网方法。由于集线器使用了大规模集成电路芯片，因此这样的硬件设备的可靠性已大大提高了。实践证明，这比使用具有大量机械接头的无源电缆要可靠得多。

1990 年 IEEE 制订出星形网 10BASE-T 的标准 802.3i。“10”代表 10 Mb/s 的数据率，T 代表双绞线。但 10BASE-T 的通信距离稍短，每个站到集线器的距离不超过 100 m。这种 10 Mb/s 速率的无屏蔽双绞线星形网的出现，既降低了成本，又提高了可靠性。**10BASE-T 双绞线以太网的出现，是局域网发展史上的一个非常重要的里程碑**，它为以太网在局域网中的统治地位奠定了牢固的基础。

使双绞线能够传送高速数据的主要措施是将双绞线的绞合度做得非常精确。这样不仅可使特性阻抗均匀以减少失真，而且大大减少了电磁波辐射和无线电频率的干扰。在多对双绞线的电缆中，还要使用更加复杂的绞合方法。

集线器的一些特点如下：

（1）从表面上看，使用集线器的局域网在物理上是一个星型网，但由于集线器是使用电子器件来模拟实际电缆线的工作，因此整个系统仍然像一个传统的以太网那样运行。也就是说，**使用集线器的以太网在逻辑上仍是一个总线网，各工作站使用的还是 CSMA/CD 协议，并共享逻辑上的总线**。网络中的各个计算机必须竞争对传输媒体的控制，并且**在一个特定时间至多只有一台计算机能够发送数据**。因此，这种 10BASE-T 以太网又称为**星型总线**(star-shaped bus)或**盒中总线**(bus in a box)。

（2）一个集线器有许多端口，例如，8 至 16 个，每个端口通过 RJ-45 插头（与电话机使用的插头相似，但略大一些）用两对双绞线与一个工作站上的网卡相连（这种插座可连接 4 对双绞线，实际上只用 2 对，即发送和接收各使用一对双绞线）。因此，一个集线器很像一个**多端口的转发器**。

---

① 注：BNC 来自 Bayonet Neil-Concelman，因此 B 表示是“卡口类型”的接头，而 NC 是发明这种接头的两个人名的缩写。

（3）**集线器和转发器都是工作在物理层**，它的每个端口都具有发送和接收数据的功能。当集线器的某个端口接收到工作站发来的比特时，就简单地将该比特向所有其他端口转发。若两个端口同时有信号输入（即发生碰撞），那么所有的端口都收不到正确的帧。图 4-10 是具有三个端口的集线器的示意图。

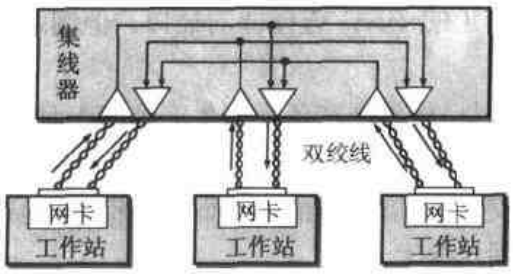


图 4-10　具有三个端口的集线器

（4）集线器采用了专门的芯片，进行自适应串音回波抵消。这样就可使端口转发出去的较强信号不致对该端口接收到的较弱信号产生干扰（这种干扰即近端串音）。每个比特在转发之前还要进行再生整形并重新定时。

集线器本身必须非常可靠。现在的**堆叠式**(stackable)集线器由 4～8 个集线器堆叠起来使用。一般都有少量的容错能力和网络管理功能。模块化的机箱式智能集线器有很高的可靠性。它全部的网络功能都以模块方式实现。各模块均可进行热插拔，出故障时不断电即可更换或增加新模块。集线器上的指示灯还可显示网络上的故障情况，给网络的管理带来了很大的方便。由于集线器具有上述这些优点，早先使用的粗缆和细缆以太网已经成为了历史并已退出了市场。

802.3 标准还扩展到可以支持宽带局域网。相应的标准是 10BROAD36。“36”表示网络的最大跨度为 3600 m，但每一个网段的长度不能超过 1800 m。数据率仍为 10 Mb/s。传输媒体为标准的 75 Ω的 CATV 电缆。基带的曼彻斯特码还要经过差分移相键控（即 DPSK）调制后才发送到电缆上。调制后的 10 Mb/s 的信号占据 14 MHz 的带宽。

802.3 标准还可使用光纤作为传输媒体，相应的标准是 10BASE-F 系列，F 代表光纤。

4.2.3　以太网的信道利用率

下面我们讨论以太网的最大信道利用率。

我们假定：

（1）总线上共有 N 个站，每个站发送帧的概率都是 p。

（2）争用期长度为 2τ，即端到端传播时延的两倍。检测到碰撞后不发送干扰信号。

（3）帧长为 L (bit)，数据发送速率为 C (b/s)，因而帧的发送时间为 $L/C = T_0$(s)。

这样，一个帧从开始发送，经碰撞后再重传数次，到发送成功且信道转为空闲（即再经过时间 τ 使得信道上无信号在传播）时为止，共需如图 4-11 所示的平均时间为 T_{av}。

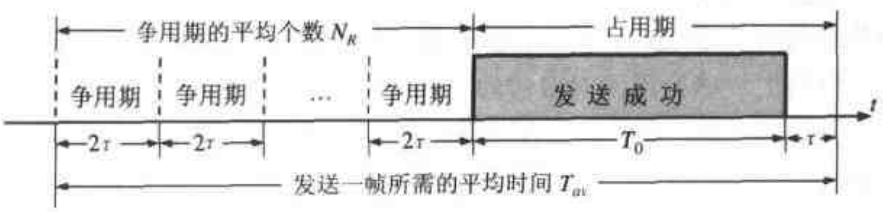


图 4-11　发送一个帧所需的平均时间 T_{av}

令 A 为某个站发送成功的概率，则

$$A = \text{P[某个站发送成功]} = Np(1-p)^{N-1} \tag{4-1}$$

显然，某个站发送失败的概率为 $1-A$。因此，

$$\text{P[争用期为 } j \text{ 个]} = \text{P[发送 } j \text{ 次失败但下一次成功]} = (1-A)^j A \tag{4-2}$$

争用期的平均个数等于帧重发的次数 N_R：

$$N_R = \sum_{j=0}^{\infty} j(1-A)^j A = (1-A)/A \tag{4-3}$$

由此求出以太网的信道利用率（它又称为归一化吞吐量）为：

$$S = \frac{T_0}{T_{av}} = \frac{T_0}{2\tau N_R + T_0 + \tau} = \frac{1}{1 + a(2A^{-1} - 1)} \tag{4-4}$$

这里

$$a = \frac{\tau}{T_0} \tag{4-5}$$

是一个非常重要的参数。参数 a 是总线的**单程传播时延与帧的发送时延之比**。

从(4-4)式可看出，若设法使 A 为最大，则可获得最大的信道利用率。将(4-4)式对 p 求极大值，得出当 $p = 1/N$ 时可使 A 等于其极大值 A_{max}：

$$A_{max} = \left[1 - \frac{1}{N}\right]^{N-1} \tag{4-6}$$

当 $N \to \infty$ 时，$A_{max} = 1/e \approx 0.368$。实际上，只要有十几个站，$A_{max}$ 就接近于 0.368 这个极限值了。这点从下面具体的数值计算即可看出：

N	2	4	8	16	32	64	128	256
A_{max}	0.5	0.422	0.393	0.380	0.374	0.371	0.369	0.369

将(4-6)式中的 A_{max} 值代入(4-4)式，即得出信道利用率的最大值 S_{max}。

图 4-12 画的是在不同帧长下 S_{max} 与总线上的站数的关系曲线。这里设总线长度为 1 km，因而单程端到端的时延约为 5μs。设数据率为 5Mb/s。帧长分别为 1024, 512, 256 和 128 bit。与帧长相对应的 a 值也注在曲线上。$N = 1$ 未画出，因为一个站无法进行通信,没有意义。从图 4-12 可以看出，只要站数 N 不是太小，S_{max} 就与 N 的数值基本无关。但我们应注意到，若帧越短，则参数 a 就越大，信道利用率的最大值 S_{max} 就越小。

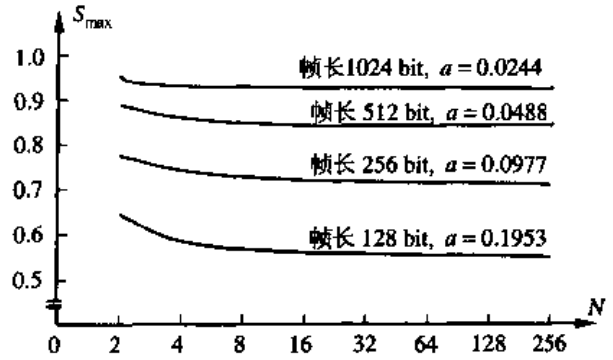


图 4-12　帧长与站数对信道利用率最大值的影响

这里要特别强调的是：图 4-12 所示的 S_{max} 曲线，是**假定了各站发送帧的概率均为理**

想的最佳值，并在此假定下求出 A_{max}，然后再算出 S_{max}。若不是这样，则 $A<A_{max}$，因而相应的 S 值也必然小于图中所示的 S_{max}。所以图 4-12 所示的 S_{max} 乃是实际上的信道利用率 S 所能达到的一个**理想化的极限值**。当取 $A = A_{max} = e^{-1} \approx 0.368$ 时，(4-4)式可简化为：

$$S_{max} \approx \frac{1}{1+4.44a} \qquad N \to \infty \tag{4-7}$$

这样就可更清楚地看出参数 a 对信道利用率最大值 S_{max} 的影响。若 $a \to 0$，则信道利用率的最大值可达到 100%。这里的物理意义是很清楚的。当 $a \to 0$ 时，只要一发生碰撞，就立即被检测出来，并立即停止发送，信道的资源不会浪费。

顺便指出，在一些文献或书籍中的类似公式里，参数 a 前面的系数可能不是 4.44。读者应自己找出为什么会有这个差别。

我们再强调一下公式(4-7)的前提。这就是假定各站都以同样的概率（此概率可使发送的成功率最大）随机地发送数据（因而不可避免地要产生碰撞）。在这种情况下，根据(4-3)式，争用期的平均个数 N_R 是(1 – 0.368)/0.368 = 1.72。

现在我们假设一种更加理想化的情况。假定以太网上的各站发送数据都不会产生碰撞（这显然已经不是 CSMA/CD 而是需要使用一种特殊的调度方法），并且能够非常有效地利用网络的传输资源，即总线一旦空闲就有某一个站立即发送数据。这样，发送一帧占用线路的时间是 $T_0 + \tau$，而帧本身的发送时延是 T_0。于是我们可计算出极限信道利用率 S_M 为

$$S_M = T_0/(T_0 + \tau) = 1/(1 + a) \tag{4-8}$$

(4-8)式的意义并非表明我们可以得到这种极限信道利用率。(4-8)式指出了**参数 a 远小于 1 才能得到尽可能高的极限信道利用率**。反之，若参数 a 远大于 1，则极限信道利用率就远小于 1，而这时实际的信道利用率就更小了。

图 4-13 可以用来说明参数 $a > 1$ 时的信道利用情况。图中设 $a = 4$，即信道的端到端传播时延 τ 是帧的发送时间 T_0 的 4 倍。假定在 $t = 0$ 时 A 站开始发送一个帧。当 $t = T_0$ 时，A 发送的第一个比特刚走完信道全长的 1/4。当 $t = 4T_0$ 时，A 发送的第一个比特正好到达 B 站。再经过时间 T_0(即当 $t = 5T_0$ 时)信道才恢复到空闲状态。

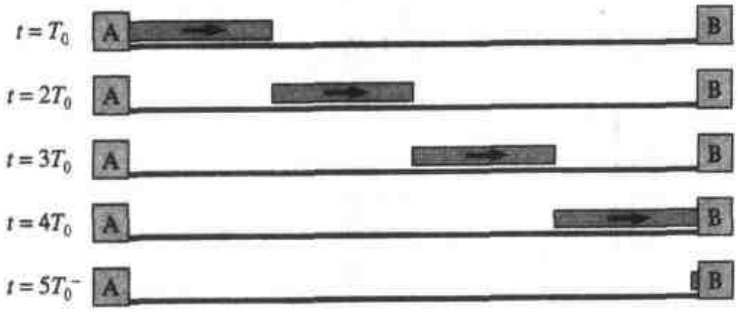


图 4-13　$a > 1$ 时的信道利用情况($a = 4$)

从图 4-13 可看出，参数 $a = 4$ 使得信道利用率很低。在数据传送的时间，信道并没有被比特填满。因此局域网若工作在这种状态就不可能获得较高的信道利用率。

参数 a 可以很容易地和第 1 章介绍的时延带宽积联系起来。考虑到 T_0 是帧长 L 与数据的发送速率 C 之比，于是参数 a 可写为：

$$a=\frac{\tau}{T_0}=\frac{\tau}{L/C}=\frac{\tau C}{L} \tag{4-9}$$

(4-9)式的分子正是第 1 章介绍的时延带宽积，或以比特为单位的信道长度，而分母是**以比特为单位的帧长**（帧长有时也用帧的发送时间来度量，这就是 T_0）。对于典型的以太网，如数据率 C =10 Mb/s，传播时延τ = 5 μs，帧长 L = 5000 bit，则算出参数 a = 0.01。这表明以比特为单位的信道长度只有帧长的 1/100。因此，发送端发送数据帧时，其数据比特很容易就把信道填满。图 4-14 画出了参数 a = 0.01 时的信道利用情况。可以看出，当 $t = 0.5\tau$时（即仅仅发送了帧长的 1/200），信道就已经被数据帧的比特填满了一半。当 $t = \tau$时，信道已被数据比特完全填满，一直到 $t = 100\tau$时都是这样。当 $t = 101\tau$时，信道才达到空闲状态。

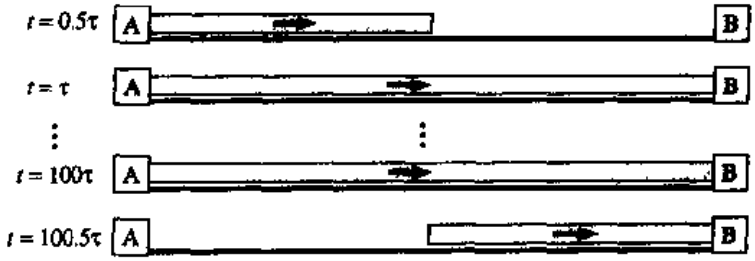


图 4-14　参数 a = 0.01 时的信道利用情况

4.3　以太网的 MAC 层

4.3.1　MAC 层的硬件地址

在局域网中，**硬件地址**又称为**物理地址**或 MAC 地址（因为这种地址用在 MAC 帧中）。

大家知道，在所有计算机系统的设计中，**标识系统**(identification system)[①]**都是一个核心**问题。在标识系统中，**地址**就是为识别某个系统的一个非常重要的标识符。在讨论地址问题时，很多人常常引用著名文献[SHOC78]给出的如下定义：

“名字指出我们所要寻找的那个资源，地址指出那个资源在何处，路由告诉我们如何到达该处。”

这个非形式的定义固然很简单，但有时却嫌不够准确。**严格地讲，名字应当与系统的所在地无关**。这就像我们每一个人的名字一样，不随我们所处的地点而改变。**但是 802 标准为局域网规定了一种 48 bit 的全球地址**（一般都简称为“**地址**”），是指局域网上的每一台计算机所**插入的网卡上固化在 ROM 中的地址**。因此，

（1）假定连接在局域网上的一台计算机的网卡坏了而我们更换了一个新的网卡，那么这台计算机的局域网的“地址”也就改变了，虽然这台计算机的地理位置一点也没有变化，所

① 注：名词 identification 原来的标准译名是“标识”[MINGCI94]。《现代汉语词典》给出“标识”的读音是“biaozhi”而不是“biaoshi”，并且说明这个词的意思是：同“标志”。现在教育部国家语言文学工作委员会发布“第一批异形词整理表”规定今后不再使用“标识”而应当用“标志”。但[MINGCI94]又将 flag 译为“标志”。这样，若 identification 和 flag 均译为“标志”就会引起混乱。因此，本书采取这样的做法：作为动词用时，我们使用“标志”，但作为名词使用时，我们用“标识”(identification)和“标志”(flag)。请读者注意。

接入的局域网也没有任何改变。

（2）假定我们将位于南京的某局域网上的一台笔记本电脑转移到北京，并连接在北京的某局域网上。虽然这台电脑的地理位置改变了，但只要电脑中的网卡不变，那么该电脑在北京的局域网中的“地址”仍然和它在南京的局域网中的“地址”一样。

由此可见，**802 标准所说的“地址”严格地讲应当是每一个站的“名字”或标识符**。但鉴于大家都早已习惯了将这种 48 bit 的“名字”称为“地址”，所以本书也采用这种习惯用法，尽管这种说法并不太严格。但读者一定要弄清这个概念。

在制定局域网的地址标准时，首先遇到的问题就是应当用多少个比特来表示一个网络的地址字段。为了减少不必要的开销，地址字段的长度应当尽可能地短些，当然它的长度应当够用。起初人们觉得用两个字节(共 16 bit)表示地址就够了，因为这一共可表示 6 万多个地址。但是，由于局域网的迅速发展，而处在不同地点的局域网之间又经常需要交换信息，这就希望在各地的局域网中的站具有互不相同的物理地址。为了使用户在买到网卡并把机器连到局域网后马上就能工作，而不需要等待网络管理员给他先分配一个地址，802 标准规定 MAC 地址字段可采用 6 字节(48 bit)或 2 字节(16 bit)这两种中的一种。6 字节的地址字段对于一个局部范围内使用的局域网的确是太长了一些，这会增加额外开销。但是由于 6 字节的地址字段可使全世界所有局域网上的站都具有不相同的地址，因此这种标准深受广大用户的欢迎。现在的局域网实际上使用的都是 6 字节 MAC 地址。

现在 IEEE 的**注册管理委员会 RAC** (Registration Authority Committee)是局域网全球地址的法定管理机构，它负责分配地址字段的 6 个字节中的前 3 个字节(即高位 24 bit)。世界上凡要生产局域网网卡的厂家都必须向 IEEE 购买由这 3 个字节构成的一个号（即地址块），这个号的正式名称是**机构惟一标识符 OUI** (Organizationally Unique Identifier)，通常也叫做**公司标识符**(company_id)。目前的价格是 1250 美元买一个地址块。例如，3Com 公司生产的网卡的 MAC 地址的前 6 个字节是 02-60-8C[①]。地址字段中的后 3 个字节(即低位 24 bit)则是由厂家自行指派，称为**扩展标识符**(extended identifier)，只要保证生产出的网卡没有重复地址即可。可见用一个地址块可以生成 2^{24} 个不同的地址。用这种方式得到的 48 bit 地址称为 MAC-48，它的通用名称是 EUI-48，这里 EUI 表示**扩展的惟一标识符**(Extended Unique Identifier)。EUI-48 的使用范围更广，不限于硬件地址，例如，用于软件接口。但应注意，24 bit 的 OUI 不能够单独使用来标志一个公司，因为一个公司可能有几个 OUI，也可能有几个小公司合起来购买一个 OUI。在生产网卡时这种 6 字节的 MAC 地址已被固化在网卡的只读存储器(ROM)中。因此 MAC 地址也常常叫作**硬件地址**(hardware address)或**物理地址**[②]。可见“MAC 地址”实际上就是**网卡地址**或**网卡标识符** EUI-48。当这块网卡插入到某台计算机后，网卡上的标识符 EUI-48 就成为这台计算机的 MAC 地址了。

IEEE 规定地址字段的第一字节的最低位为 I/G 比特，I/G 表示 Individual/Group。当 I/G

① 注：这里的 02-60-8C 是十六进制数字在局域网地址中的一种标准记法。每 4 个二进制数字用一个十六进制数字表示，而每两个十六进制数字与它后面两个十六进制数字之间用连字符隔开。另一种记法是在 0x 后面写上一连串的十六进制数字，如 0x02608C。

② 注：地址有平面地址(flat address)和层次地址(hierarchical address)两大类。平面地址也叫作非层次地址，就是在分配地址时按顺序号一个个地挨着分配。层次地址是将整个地址再划分为几个部分，而每部分按一定的规律分配号码。像我们的电话号码就是一种层次号码，电信网中的交换机按照国家号→区号→局号→用户号的顺序可以准确地找到用户的电话。但局域网的 6 字节的地址是一种平面地址。网卡根据全部的 48 bit 地址决定接收或丢弃所收到的 MAC 帧。

比特为 0 时，地址字段表示一个**单个站地址**。当 I/G 比特为 1 时表示**组地址**，用来进行多播。因此，IEEE 只分配地址字段前三个字节中的 23 bit。当 I/G 比特分别为 0 和 1 时，一个地址块可分别生成 2^{24} 个单个站地址和 2^{24} 个组地址。需要指出，有的书将上述最低位写为“第一比特”，但这样有些含糊不清，因为“第一”的定义是不明确的。这点可从图 4-15 看出。在图 4-15 的上面部分给出了一个十六进制表示的 EUI-48 地址：AC-DE-48-00-00-80。在这六个字节中，第一字节是 AC，而第六字节是 80。这种记法没有二义性，因为大家都是将第一字节写在最左边，将第六字节写在最右边。

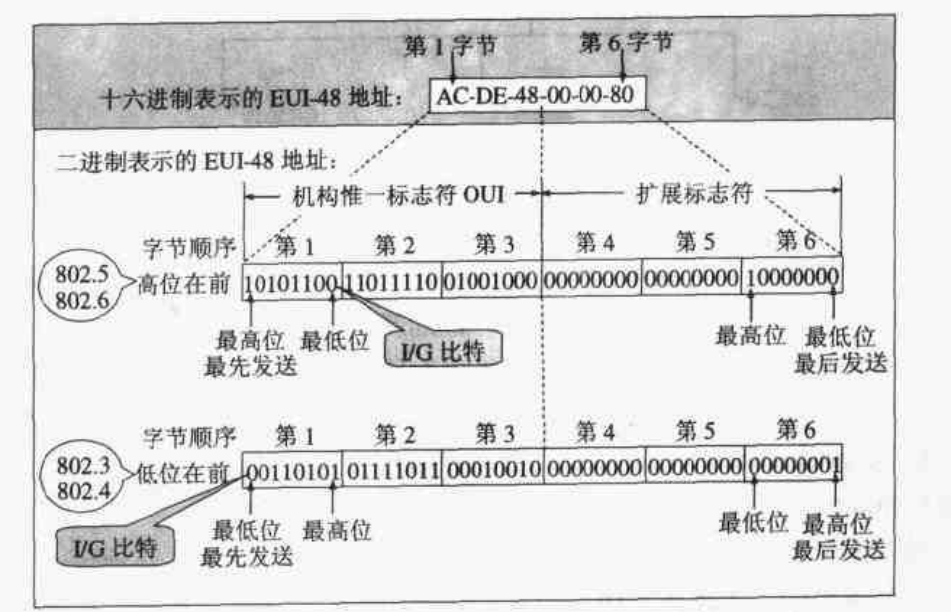


图 4-15　EUI-48 地址的十六进制和二进制记法

但 IEEE 制订的二进制 EUI-48 地址就有两种不同的记法。

第一种记法是 802.5 和 802.6 采用的标准。这种记法将每一个字节的高位写在最左边，如第一字节 AC 就变成为 10101100。这种记法和平常习惯的二进制数字的记法一致，即每一个字节的高位比特写在最左边。在发送数据时是按照字节的顺序发送，而每一个字节先发送其最高位比特。

另一种记法是 802.3 和 802.4 采用的标准。这种记法将每一个字节的高位比特写在最右边，如第一字节 AC 就变成为 00110101(这看起来好像是十六进制的 35)。在发送数据时也是按照字节的顺序发送，但每一个字节先发送其最低位比特。

造成这种不同的记法是因为在制订 IEEE 802 标准时各公司都坚持要使标准和自己公司原有的产品标准相兼容。从图 4-15 可看出，在二进制表示的 EUI-48 地址中，高位在前和低位在前两种不同二进制记法中，I/G 比特的位置是不一样的。

IEEE 还考虑到可能有人并不愿意向 IEEE 的 RAC 购买机构惟一标识符 OUI。为此，IEEE 将地址字段第 1 字节的最低第 2 位规定为 G/L 比特，表示 Global/Local。当 G/L 比特为 1 时是**全球管理**（保证在全球没有相同的地址），厂商向 IEEE 购买的 OUI 都属于**全球管理**。当地址字段的 G/L 比特为 0 时是**本地管理**，这时用户可任意分配网络上的地址。采用 2 字节地址字段时全都是本地管理。但应当指出，以太网几乎不使用这个 G/L 比特。

这样，在全球管理时，对每一个站的地址可用 46 位的二进制数字来表示（最低位为 0 和最低第 2 位为 1 时）。剩下的 46 位组成的地址空间可以有超过 70 万亿个地址，可保证世界

上的每一个网卡都可有一个惟一的地址。

由于网卡是插在计算机中，因此网卡上的硬件地址就可用来标志插有该网卡的计算机。同理，当路由器用网卡连接到局域网时，网卡上的硬件地址就用来标志插有该网卡的路由器的某个接口。图 4-16 表示用网卡上的硬件地址来标志局域网上的计算机和路由器。图中的路由器由于同时连接到两个网络上，因此它有两块网卡和两个硬件地址。

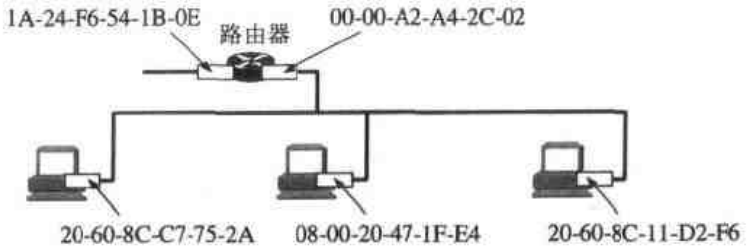


图 4-16　用网卡上的硬件地址标志局域网上的计算机和路由器

网卡从网络上每收到一个 MAC 帧就首先用硬件检查 MAC 帧中的 MAC 地址。如果是发往本站的帧则收下，然后再进行其他的处理。否则就将此帧丢弃，不再进行其他的处理。这样做就不浪费主机的处理机和内存资源。这里“发往本站的帧”包括以下三种帧：

（1）**单播**(unicast)帧（一对一），即收到的帧的 MAC 地址与本站的硬件地址相同。

（2）**广播**(broadcast)帧（一对全体），即发送给所有站点的帧（全 1 地址）。

（3）**多播**(multicast）帧（一对多），即发送给一部分站点的帧。

所有的网卡都至少应当能够识别前两种帧，即能够识别单播和广播地址。有的网卡可用编程方法识别多播地址。当操作系统启动时，它就将网卡初始化，使网卡能够识别某些多播地址。显然，只有目的地址才能使用广播地址和多播地址。

4.3.2　两种不同的 MAC 帧格式

常用的以太网 MAC 帧格式有两种标准，一种是 DIX Ethernet V2 标准，另一种是 IEEE 的 802.3 标准。图 4-17 画出了这两种不同的 MAC 帧格式。为便于理解，图中假定网络层使用的是 IP 协议。实际上使用其他的协议也是可以的。

现在 MAC 帧最常用的是以太网 V2 的格式，它较为简单，由 5 个字段组成。前两个字段分别为 6 字节长的**目的地址**和**源地址**字段。第三个字段是 2 字节的**类型字段**，用来标志上一层使用的是什么协议，以便把收到的 MAC 帧的数据上交给上一层的这个协议。施乐公司负责管理这个类型字段的代码分配。例如，当类型字段的值是 0x0800 时，就表示上层使用的是 IP 数据报。若类型字段的值为 0x8137，则表示该帧是由 Novell IPX 发过来的。第四个字段是**数据字段**，但它的正式名称是 **MAC 客户数据字段**，其长度在 46 到 1500 字节之间（46 字节是这样得出的：最小长度 64 字节减去 18 字节的首部和尾部就得出数据字段的最小长度）。最后一个字段是 4 字节的**帧检验序列 FCS**。当传输媒体的误码率为 1×10^{-8} 时，MAC 子层可使未检测到的差错小于 1×10^{-14}。

当数据字段的长度小于 46 字节时，MAC 子层就会在数据字段的后面加入一个整数字节的填充字段，以保证以太网的 MAC 帧长不小于 64 字节。我们应当注意到，MAC 帧的首部并没有指出数据字段的长度是多少。在有填充字段的情况下，接收端的 MAC 子层在剥去首部和尾部后就将数据字段和填充字段一起交给上层协议。现在的问题是：上层协议如何知道

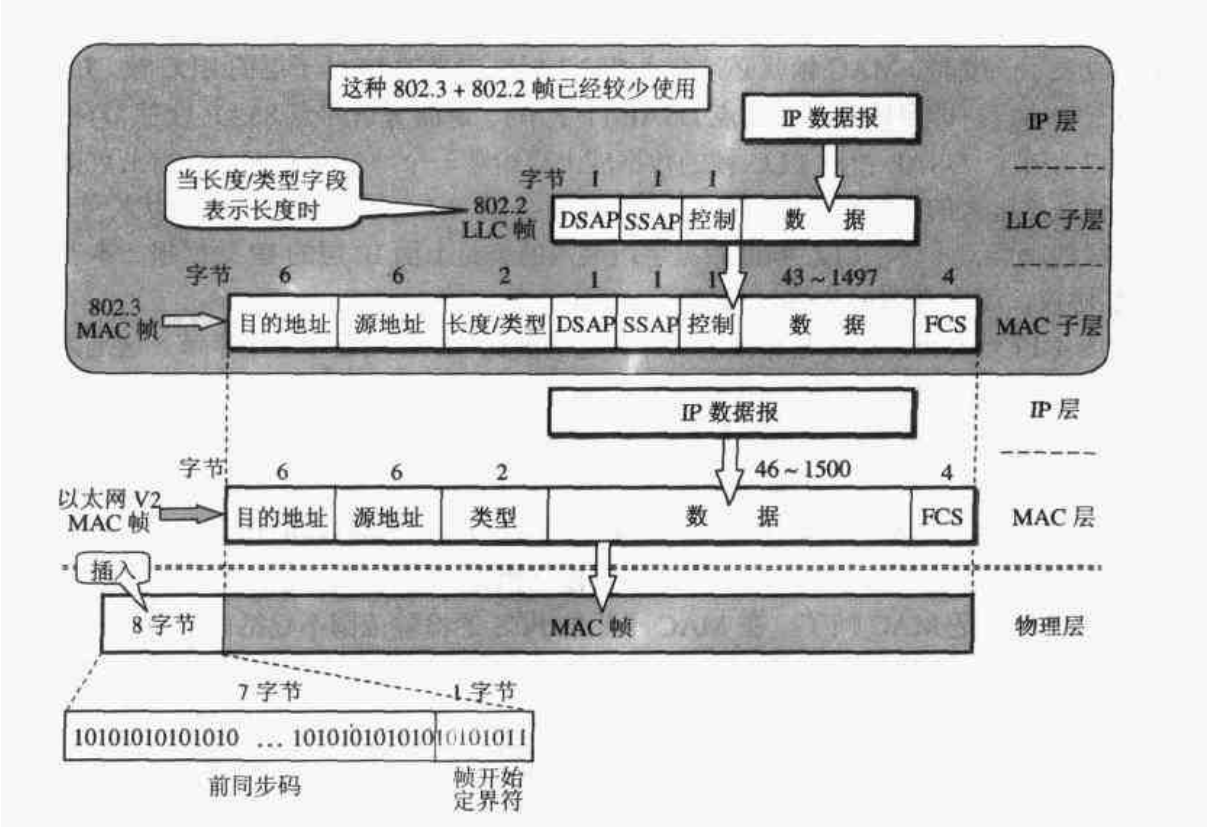


图 4-17　两种不同的 MAC 帧格式

数据字段的长度以便将没有用处的填充字段丢弃呢？答案是：上层协议必须具有识别有效的数据字段长度的功能。例如，当上层使用 IP 协议时，其首部就有一个“总长度”字段，它等于 MAC 帧中数据字段的长度（见 6.2.4 节）。这样，IP 协议可以很容易地将多余的填充字段丢弃。

然而 IEEE 802.3 标准规定的 MAC 帧则较为复杂[①]。它和以太网 V2 的 MAC 帧的区别是：

（1）第三个字段是**长度/类型字段**。根据长度/类型字段的数值大小，这个字段可以表示 MAC 帧的**数据字段长度**（请注意：**不是整个 MAC 帧的长度**），也可以等同于以太网 V2 的类型字段。具体地讲：

- 若长度/类型字段的数值小于 MAC 帧的数据字段的最大值 1500 (字节)，这个字段就表示 MAC 帧的数据字段长度。
- 若长度/类型字段的数值大于 0x0600(相当于十进制的 1536)，那么这个数值就不可能表示以太网有效的数据字段长度，因而这个字段就表示类型。

当长度/类型字段表示类型时，802.3 的 MAC 帧和以太网 V2 的 MAC 帧一样。当长度/

① 注：802.3 标准的 MAC 帧格式之所以较为复杂，是因为 802 委员会不仅制定 802.3 标准，而且还要制定另外几个标准，如 802.4 和 802.5 等。802 委员会的出发点是：(1) MAC 层自己就应当能够知道有效的数据字段的长度，以利于不同类型局域网之间的互通，因此需要有一个字段指出数据字段的长度；(2) 用 LLC 子层首部中的 DSAP 和 SSAP 字段以及控制字段不仅可以实现以太网 V2 中的类型字段的功能，而且还可使局域网实现面向连接的服务。为了使 802.3 协议能够更好地和以太网 V2 协议相兼容，802 委员会又制定了 802.3 子网接入协议 SNAP (Sub-Network Access Protocol)。使用 SNAP 协议时，DSAP 和 SSAP 字段都置为 0xAA，然后在 LLC 帧的控制字段后面再增加两个字段。第一个是 OUI 字段（3 字节），置为全 0，而第二个字段（2 字节）和以太网 V2 的类型字段一样。但实践证明，这样做过于繁琐，白白地在 MAC 帧中增加了 8 个字节的开销，这是因为 802.4 和 802.5 局域网以及面向连接的服务方式现在都很少有人愿意使用。

类型字段表示长度时，MAC 帧就必须装入 802.2 标准定义的 LLC 子层的 LLC 帧。LLC 帧的首部有三个字段，即**目的服务访问点** DSAP (1 字节)、**源服务访问点** SSAP (1 字节)和**控制字段**(1 或 2 字节)。DSAP 指出 LLC 帧的数据应上交给哪一个协议，而 SSAP 指出数据是从哪一个协议发送过来的。控制字段则指出 LLC 帧的类型。LLC 帧首部的作用和以太网 V2 帧的类型字段的功能差不多。LLC 帧的数据字段装入的就是上面 IP 层的 IP 数据报。本书不再讨论 LLC 协议，可参考[NAUG99]一书的第 4 章。

从图 4-17 可看出，在传输媒体上实际传送的要比 MAC 帧还多 8 个字节。这是因为当一个站在刚开始接收 MAC 帧时，由于尚未与到达的比特流达成同步，因此 MAC 帧的最前面的若干个比特就无法接收，结果使整个的 MAC 成为无用的帧。为了达到比特同步，从 MAC 子层向下传到物理层时还要在帧的前面插入 8 字节（由硬件生成），它由两个字段构成。第一个字段共 7 个字节，称为**前同步码**（1 和 0 交替的码）。前同步码的作用是使接收端在接收 MAC 帧时能够迅速实现比特同步。第二个字段是**帧开始定界符**，定义为 10101011，表示在这后面的信息就是 MAC 帧了。在 MAC 子层的 FCS 的检验范围不包括前同步码和帧开始定界符。顺便指出，在广域网中使用同步传输的 HDLC 规程时则不需要用前同步码，因为在同步传输时收发双方的比特同步总是一直保持着的。

802.3 标准规定凡出现下列情况之一的即为无效的 MAC 帧：

（1）MAC 客户数据字段的长度与长度字段的值不一致；

（2）帧的长度不是整数个字节；

（3）用收到的帧检验序列 FCS 查出有差错；

（4）收到的帧的 MAC 客户数据字段的长度不在 46～1500 字节之间。考虑到 MAC 帧首部的长度是 18 字节，可以得出有效的 MAC 帧长度为 64～1518 字节之间。

对于检查出的无效 MAC 帧就简单地丢弃。以太网不负责重传丢弃的帧。

当 MAC 客户数据字段的长度小于 46 字节时，则应加以填充（内容不限）。这样，整个 MAC 帧（包含 14 字节首部和 4 字节尾部）的最小长度是 64 字节，或 512 bit。

MAC 子层的标准还规定了帧间最小间隔为 9.6μs，相当于 96 bit 的发送时间。这就是说，一个站在检测到总线开始空闲后，还要等待 9.6 μs 才能发送数据。这样做是为了使刚刚收到数据帧的站的接收缓存来得及清理，做好接收下一帧的准备。

4.4 扩展的局域网

在许多情况下，一个单位往往拥有许多个局域网，因而需要实现局域网之间的通信。这时就需要使用一些中间设备将这些局域网连接起来。如果使用路由器进行互连（在网络层进行互连），那么这是第 6 章将要讨论的网络互连问题。本节要讨论的是在物理层或数据链路层将局域网进行扩展。这种扩展的局域网在网络层看来仍然是一个网络。

4.4.1 在物理层扩展局域网

在物理层扩展局域网是使用转发器和集线器。在 4.2.2 节我们已经介绍了转发器和集线器的使用方法。这里再讨论一下用几个集线器连接成更大范围的多级星型结构的局域网。例如，一个学院的三个系各有一个 10BASE-T 局域网（图 4-18(a)），可通过一个主干集线器互相连接起来，成为一个更大的扩展的局域网（图 4-18(b)）。

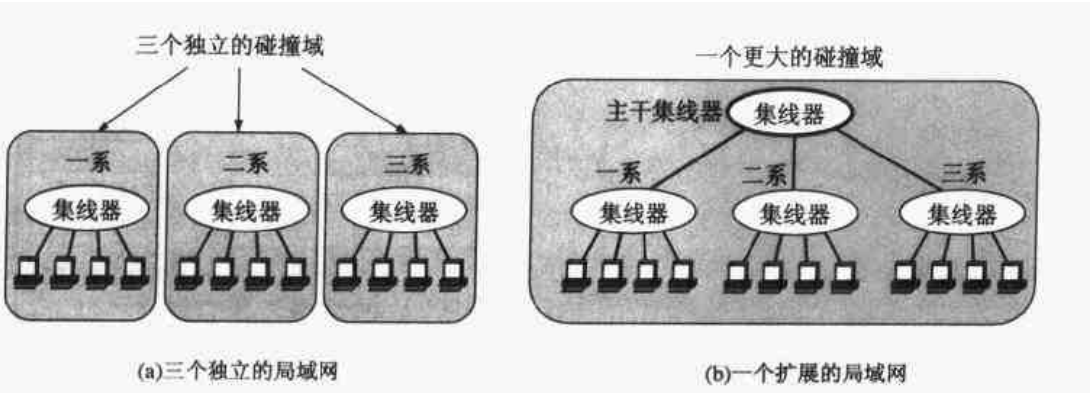


图 4-18　用多个集线器连成更大的局域网

这样做可以有以下两个好处。第一，使学院不同系的局域网上的计算机能够进行跨系的通信。第二，扩大了局域网覆盖的地理范围。例如，在一个系的 10BASE-T 局域网中，主机与集线器的最大距离是 100 m，因而两个主机之间的最大距离是 200 m。但在通过主干集线器相连接后，不同系的主机之间的距离就可扩展了，因为集线器之间的距离可以是 100 m（使用双绞线）或甚至更远（如使用光纤）。

但这种多级结构的集线器局域网也带来了一些缺点。

（1）如图 4-18(a)所示的例子，在三个系的局域网互连起来之前，每一个系的 10BASE-T 局域网是一个独立的**碰撞域**（collision domain，又称为**冲突域**），即在任一时刻，在每一个系的局域网中只能有一个站在发送数据。每一个系的局域网的最大吞吐量是 10 Mb/s，因此三个系总的最大吞吐量共有 30 Mb/s。在三个系的局域网通过集线器互连起来后就组成了一个更大的、共同的碰撞域，如图 4-18(b)所示，而三个系合起来的最大吞吐量还是 10 Mb/s。

（2）如果不同的系使用不同的以太网技术（如数据率不同），那么就不可能用集线器将它们互连起来。如果在图 4-18 中，一个系使用 10 Mb/s 的网卡，而另外两个系使用 10/100 Mb/s 的网卡，那么用集线器连接起来后，大家都只能工作在 10 Mb/s 的速率。集线器基本上是个多端口的转发器，它并不能将帧进行缓存。

4.4.2　在数据链路层扩展局域网

在数据链路层扩展局域网是使用**网桥**。网桥工作在数据链路层，它根据 MAC 帧的目的地址对收到的帧进行转发。网桥具有过滤帧的功能。当网桥收到一个帧时，并不是向所有的端口转发此帧，而是先检查此帧的目的 MAC 地址，然后再确定将该帧转发到哪一个端口[NETW88]。

1．网桥的内部结构

图 4-19 给出了一个网桥的内部结构要点。最简单的网桥有两个端口（即接口）。复杂些的网桥可以有更多的端口。网桥的每个端口与一个网段（这里所说的网段就是普通的局域网）相连。在图中所示的网桥，其端口 1 与网段 A 相连，而端口 2 则连接到网段 B。

网桥从端口接收网段上传送的各种帧。每当收到一个帧时，就先暂存在其缓存中。若此帧未出现差错，且欲发往的目的站 MAC 地址属于另一个网段，则通过查找**转发表**，将收到的帧送往对应的端口转发出去。若该帧出现差错，则丢弃此帧。因此，仅在同一个网段中通信的帧，不会被网桥转发到另一个网段去，因而不会加重整个网络的负担。例如，设图 4-19

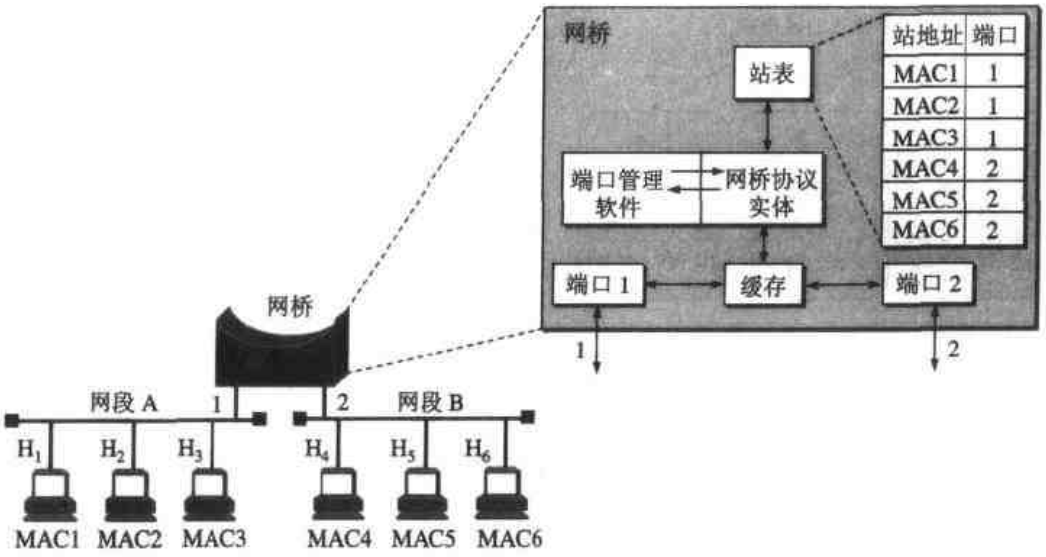


图 4-19　网桥的工作原理

中网段 A 上的三个站 H_1、H_2 和 H_3 的 MAC 地址分别为 MAC1、MAC2 和 MAC3，而网段 B 上的三个站 H_4、H_5 和 H_6 的 MAC 地址分别为 MAC4、MAC5 和 MAC6。若网桥的端口 1 收到站 H_1 发给站 H_5 的帧，则在查找转发表后，将此帧送到端口 2 转发给网段 B，然后再传给站 H_5。若端口 1 收到站 H_1 发给站 H_2 的帧，由于目的站对应的端口就是该帧进入网桥的端口 1，表明不需要经过网桥转发，于是丢弃此帧。

网桥是通过内部的端口管理软件和网桥协议实体来完成上述操作的。转发表也叫做**转发数据库**或**路由目录**。至于转发表是怎样建立起来的，后面还要讨论。

使用网桥可以带来以下好处：

（1）**过滤通信量**。网桥工作在链路层的 MAC 子层，可以使局域网各网段成为隔离开的**碰撞域**，从而减轻了扩展的局域网上的负荷，同时也减小了在扩展的局域网上的帧平均时延。工作在物理层的转发器就没有网桥的这种过滤通信量的功能。

（2）**扩大了物理范围**，因而也增加了整个局域网上工作站的最大数目。

（3）**提高了可靠性**。当网络出现故障时，一般只影响个别网段。

（4）**可互连不同物理层、不同 MAC 子层和不同速率**（如 10 Mb/s 和 100 Mb/s 以太网）**的局域网**。

当然，网桥也有一些缺点，例如：

（1）由于网桥对接收的帧要先存储和查找转发表，然后才转发，这就**增加了时延**。

（2）在 MAC 子层并**没有流量控制功能**。当网络上的负荷很重时，网桥中的缓存的存储空间可能不够而发生溢出，以致产生帧丢失的现象。

（3）具有不同 MAC 子层的网段桥接在一起时，网桥在转发一个帧之前，必须修改帧的某些字段的内容，以适合另一个 MAC 子层的要求。这也耗费时间，因而**增加时延**。

（4）网桥只适合于用户数不太多（不超过几百个）和通信量不太大的局域网，否则有时还会因传播过多的广播信息而产生网络拥塞。这就是所谓的**广播风暴**。

尽管如此，网桥仍获得了很广泛的应用，因为它的优点还是主要的。

有时在两个网桥之间还可使用一段点到点链路。图 4-20 说明了这种情况。

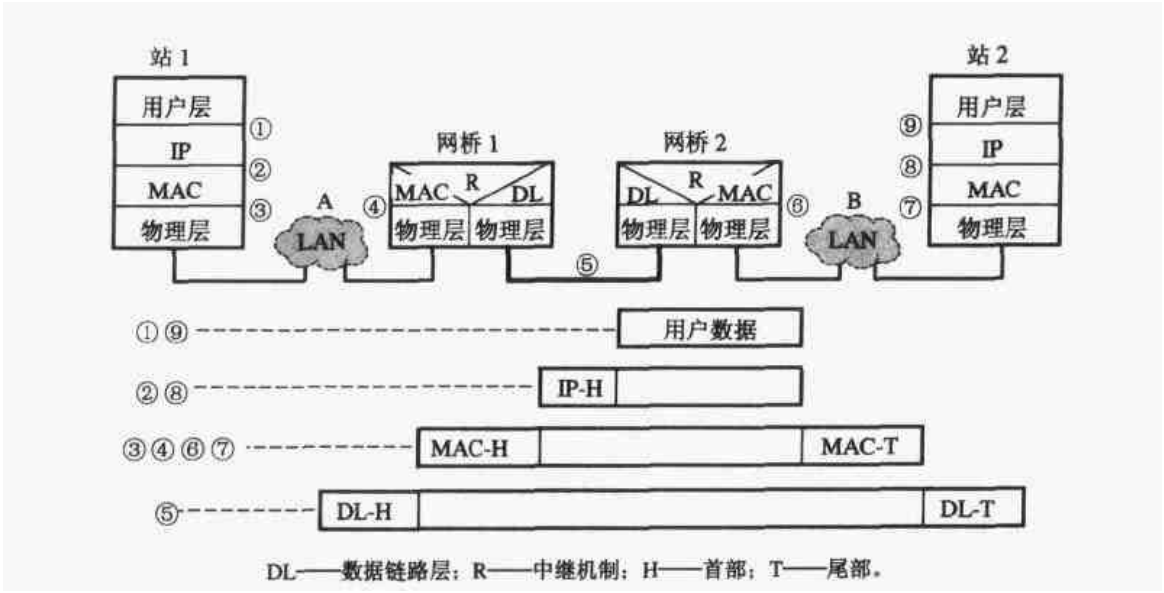


图 4-20　两个网桥之间有点到点的链路

图中的局域网 A 和 B 通过两个网桥（网桥 1 和网桥 2）和一段点到点链路相连。为简单起见，我们省略了 802.3 的 LLC 子层，同时将 IP 层以上看成是用户层。图 4-20 的上面部分是从体系结构来看网桥的作用，而图的下面部分表示用户数据从站 1 传到站 2 经过各层次时数据帧的首部的变化。这里只需要指出以下几点。

当站 1 向站 2 发送数据帧时，其 MAC 帧首部中的源地址和目的地址分别是站 1 和站 2 的硬件地址，相当于图中的③和④所对应的图。当网桥 1 通过点对点链路转发数据帧时，若链路采用 HDLC 协议，则要在数据帧的头尾分别加上链路层协议的首部 DL-H 和尾部 DL-T（对应于图中的⑤）。在数据帧离开网桥 2 时，还要剥去这个首部 DL-H 和尾部 DL-T（如⑥），然后经过局域网 B 到达站 2。

请读者分清网桥和集线器（或转发器）的不同。集线器在转发一个帧时，不对传输媒体进行检测。但网桥在转发一个帧之前必须执行 CSMA/CD 算法。若在发送过程中出现碰撞，就必须停止发送和进行退避。在这一点上网桥的接口很像一个网卡。但网桥却没有网卡。我们知道，网卡在发送一个帧之前一定要将本网卡上的源地址写到 MAC 帧的地址字段中。但由于网桥没有网卡，因此**网桥并不改变它转发的帧的源地址**。

2．透明网桥

目前使用得最多的网桥是**透明网桥**(transparent bridge)。“透明”是指局域网上的站点并不知道所发送的帧将经过哪几个网桥，因为网桥对各站来说是看不见的。透明网桥是**一种即插即用设备**，其标准是 IEEE 802.1D。

当一个网桥刚刚连接到局域网上时，其转发表显然是空的。这时若网桥收到一个帧，它将怎样处理该帧呢？网桥应当按照以下算法处理该帧和建立起自己的转发表。

（1）从端口 x 收到无差错的帧（如有差错即丢弃），在转发表中查找目的站 MAC 地址。

（2）如有，则查找出到此 MAC 地址应当走的端口 d，然后进行(3)，否则转到(5)。

（3）如到这个 MAC 地址去的端口 d = x，则丢弃此帧（因为这表示不需要经过网桥进行转发）。否则从端口 d 转发此帧。

（4）转到(6)。

（5）向网桥除 x 以外的所有端口转发此帧（这样做可保证找到目的站）。

（6）如源站不在转发表中，则将源站 MAC 地址加入到转发表，登记该帧进入网桥的端口号，设置计时器。然后转到(8)。如源站在转发表中，则执行(7)。

（7）更新计时器。

（8）等待新的数据帧。转到(1)。

这时，网桥就在转发表中登记以下三个信息：

（1）**站地址**：登记收到的帧的源 MAC 地址。

（2）**端口**：登记收到的帧进入该网桥的端口号。

（3）**时间**：登记收到的帧进入该网桥的时间（图 4-19 中的转发表省略了这一项）。

网桥在这样的转发过程中就可逐渐将其转发表建立起来。这里特别要注意的是，转发表中的 MAC 地址是根据源 MAC 地址写入的，但在进行转发时是将此 MAC 地址当作目的地址。这是因为网桥的转发表根据的原理就是：**如果网桥现在能够从端口 x 收到从源地址 A 发来的帧，那么以后就可以从端口 x 将一个帧转发到目的地址 A。**

局域网的拓扑经常会发生变化。局域网上的工作站和网桥可能时而接通电源时而关掉电源。为了使转发表能反映出整个网络的最新拓扑，所以还要将每个帧到达网桥的时间登记下来，以便在转发表中保留网络拓扑的最新状态信息。具体的方法是，网桥中的端口管理软件周期性地扫描转发表中的项目。只要是在一定时间（例如几分钟）以前登记的都要删除。这样就使得网桥中的转发表能反映当前网络拓扑状态。

透明网桥使用了一个**支撑树**(spanning tree)算法，即互连在一起的网桥在进行彼此通信后，就能找出原来的网络拓扑的一个子集，在这个子集里整个连通的网络中不存在回路，即在任何两个站之间只有一条路径。一旦支撑树确定了，网桥就会将某些接口断开，以确保从原来的拓扑得出一个支撑树。

为什么要找出一个支撑树呢？就是为了避免产生转发的帧在网络中不断地**兜圈子**。可以看图 4-21 所示的简单例子。这里用两个网桥将两个局域网互连起来。设站 A 发送一个帧 F，它经过这两个网桥（见箭头①和②）。假定帧 F 的目的地址均不在这两个网桥的转发表中，因此两个网桥都转发帧 F（见箭头③和④）。我们将经网桥 1 和网桥 2 转发的帧 F 在到达局域网 2 以后，分别记为 F_1 和 F_2。接着 F_1 传到网桥 2（见箭头⑤）而 F_2 传到了网桥 1（见箭头⑥）。网桥 2 和网桥 1 分别收到 F_1 和 F_2 后，又将其转发到局域网 1。结果引起一个帧在网络中不停地兜圈子，从而使网络资源不断地白白消耗了。

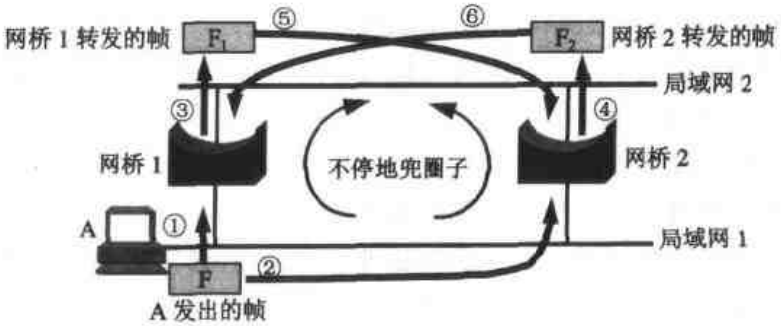


图 4-21　网桥引起的兜圈子

为了得出能够反映网络拓扑发生变化时的支撑树，每隔几秒钟每一个网桥要广播其标识号（由生产网桥的厂家设定的一个惟一的序号）和它所知道的其他所有在网上的网桥。支撑

树算法选择一个网桥作为支撑树的根（例如，选择一个最小序号的网桥），然后以最短路径为依据，找到树上的每一个结点。但这样并不能充分利用全部的可用资源（因为某些路径未被利用，为的是消除兜圈子现象），同时也不一定将每一个帧沿最佳的路由传送，因此可能导致稍大些的时延。

当互连局域网的数目非常大时，支撑树的算法可能要花费很多的时间。这时可将大的互连网划分为多个较小的互连网，然后得出多个支撑树。

3. 源路由网桥

透明网桥的最大优点就是容易安装，一接上就能工作。但是，网络资源的利用不充分。因此另一种由发送帧的源站负责路由选择的网桥就问世了，这就是**源路由**(source route)**网桥**。

源路由网桥是在发送帧时将详细的路由信息放在帧的首部中。

这里的关键是源站用什么方法才能知道应当选择什么样的路由。

为了发现合适的路由，源站以广播方式向欲通信的目的站发送一个**发现帧**(discovery frame)作为探测之用。发现帧将在整个扩展的局域网中沿着所有可能的路由传送。在传送过程中，每个发现帧都记录所经过的路由。当这些发现帧到达目的站时，就沿着各自的路由返回源站。源站在得知这些路由后，从所有可能的路由中选择出一个最佳路由。以后凡从这个源站向该目的站发送的帧的首部，都必须携带源站所确定的这一路由信息。

发现帧还有另一个作用，就是帮助源站确定整个网络可以通过的帧的最大长度。

源路由网桥对主机则不是透明的，主机必须知道网桥的标识以及连接到哪一个网段上。使用源路由网桥可以利用最佳路由。若在两个局域网之间使用并联的源路由网桥，则可使通信量较平均地分配给每一个网桥。用透明网桥则只能使用支撑树，它一般并不是最佳路由，也不能在不同的链路中进行负载的均衡。

4. 多端口网桥——以太网交换机

1990 年问世的**交换式集线器**(switching hub)，可明显地提高局域网的性能。交换式集线器常称为以太网**交换机**(switch)或**第二层交换机**（表明这种交换机工作在数据链路层）。

“交换机”并无准确的定义和明确的概念，而现在的很多交换机已混杂了网桥和路由器的功能。著名网络专家 Perlman 认为：“交换机”应当是一个**市场名词**，而交换机的出现的确使数据的转发更加快速了（见[PERL00]的第 5 章）。由于交换机这一名词已经广泛地使用了，因此我们也使用这个名词。下面简单地介绍其特点。

从技术上讲，网桥的端口数很少，一般只有 2～4 个，而以太网交换机通常都有十几个端口。因此，**以太网交换机实质上就是一个多端口的网桥**，可见**交换机工作在数据链路层**（和工作在物理层的转发器和集线器不同）。此外，以太网交换机的每个端口都直接与一个单个主机或另一个集线器相连（注意：普通网桥的端口往往是连接到局域网上的一个主机），并且一般都工作在全双工方式。当主机需要通信时，交换机能同时连通许多对的端口，使每一对相互通信的主机都能像独占通信媒体那样，进行**无碰撞地传输数据**。通信完成后就断开连接。以太网交换机由于使用了专用的交换结构芯片，其交换速率就较高。

对于普通 10 Mb/s 的共享式以太网，若共有 N 个用户，则每个用户占有的平均带宽只有总带宽(10 Mb/s)的 N 分之一。在使用以太网交换机时，虽然在每个端口到主机的带宽还是 10 Mb/s，但由于一个用户在通信时是独占而不是和其他网络用户共享传输媒体的带宽，因此

对于拥有 N 对端口的交换机的总容量为 N×10 Mb/s。这正是交换机的最大优点。

从共享总线以太网或 10BASE-T 以太网转到交换式以太网时，所有接入设备的软件和硬件、网卡等都不需要作任何改动。也就是说，所有接入的设备继续使用 CSMA/CD 协议。此外，只要增加集线器的容量，整个系统的容量是很容易扩充的。

以太网交换机一般都具有多种速率的端口，例如，可以具有 10 Mb/s，100 Mb/s 和 1 Gb/s 的端口的各种组合，这就大大方便了各种不同情况的用户。

图 4-22 举出了一个简单的例子。图中的以太网交换机有三个 10 Mb/s 端口分别和学院三个系的 10BASE-T 局域网相连，还有三个 100 Mb/s 的端口分别和电子邮件服务器、万维网服务器以及一个连接因特网的路由器相连。

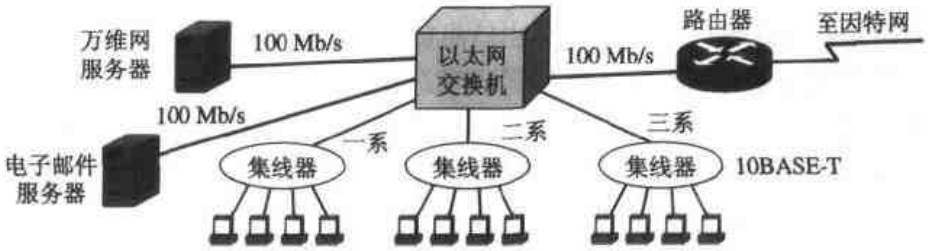


图 4-22　用以太网交换机扩展局域网

有些以太网交换机对收到的帧采用存储转发方式进行转发，但也有一些交换机采用**直通**(cut-through)的交换方式。直通交换不必将整个数据帧先缓存后再进行处理，而是在接收数据帧的同时就立即按数据帧的目的 MAC 地址决定该帧的转发端口,因而提高了帧的转发速度。如果在这种交换机的内部采用基于硬件的交叉矩阵，交换时延非常小。直通交换的一个缺点是它不检查差错就直接将帧转发出去，因此有可能也将一些无效帧转发给其他的站。在某些情况下，仍需要采用基于软件的存储转发方式进行交换，例如，当需要进行线路速率匹配、协议转换或差错检测时。现在有的厂商已生产出能支持两种交换方式的以太网交换机。以太网交换机的发展与建筑物结构化布线系统的普及应用密切相关。在结构化布线系统中，广泛地使用了以太网交换机。

4.5　虚拟局域网

4.5.1　虚拟局域网的概念

在 IEEE 802.1Q 标准中对**虚拟局域网 VLAN** (Virtual LAN)是这样定义的：

虚拟局域网 VLAN 是由一些局域网网段构成的与物理位置无关的逻辑组，而这些网段具有某些共同的需求。每一个 VLAN 的帧都有一个明确的标识符，指明发送这个帧的工作站是属于哪一个 VLAN。

利用以太网交换机可以很方便地实现虚拟局域网 VLAN。这里要指出，虚拟局域网其实只是局域网**给用户提供的一种服务**，而**并不是一种新型局域网**。

图 4-23 画的是使用了四个交换机的网络拓扑。设有 10 个工作站分配在三个楼层中，构成了三个局域网，即：

LAN_1：(A_1, A_2, B_1, C_1)，　LAN_2：(A_3, B_2, C_2)，LAN_3：(A_4, B_3, C_3)

但这 10 个用户划分为三个工作组，也就是说划分为三个虚拟局域网 VLAN。即：

$VLAN_1$：(A_1, A_2, A_3, A_4)，$VLAN_2$：(B_1, B_2, B_3)；$VLAN_3$：(C_1, C_2, C_3)。

从图 4-23 可看出，每一个 VLAN 的工作站可处在不同的局域网中，也可以不在同一层楼中。

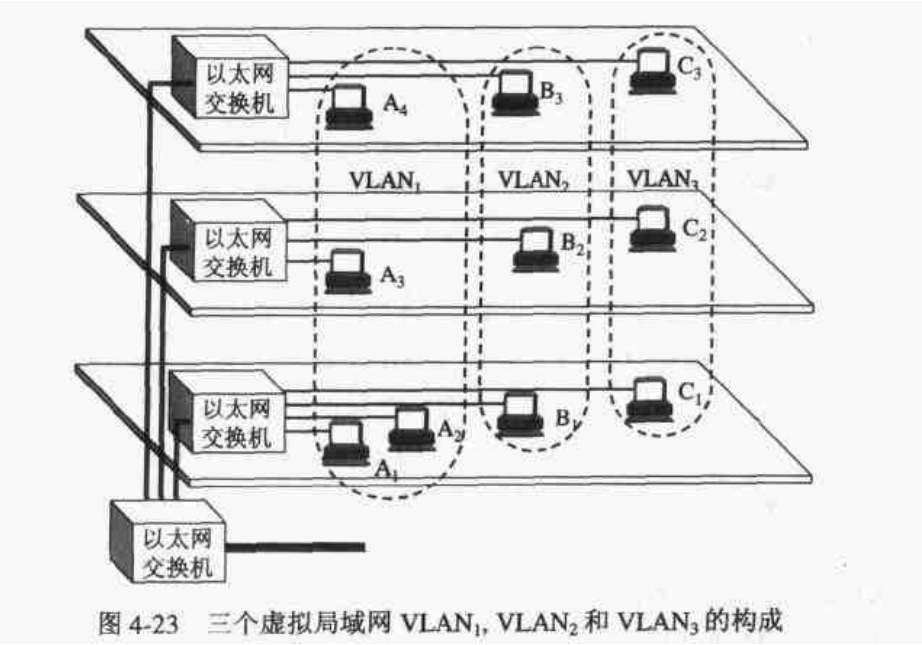


图 4-23　三个虚拟局域网 $VLAN_1$, $VLAN_2$ 和 $VLAN_3$ 的构成

利用交换集线器可以很方便地将这 10 个工作站划分为三个虚拟局域网：$VLAN_1$, $VLAN_2$ 和 $VLAN_3$。在虚拟局域网上的每一个站都可以听到同一个虚拟局域网上的其他成员所发出的广播。例如，工作站 B_1～B_3 同属于虚拟局域网 $VLAN_2$。当 B_1 向工作组内成员发送数据时，工作站 B_2 和 B_3 将会收到广播的信息，虽然它们没有和 B_1 连在同一个集线器上。相反，B_1 发送数据时，工作站 A_1, A_2 和 C_1 都不会收到 B_1 发出的广播信息，虽然它们都与 B_1 连接在同一个集线器上。交换式集线器不向虚拟局域网以外的工作站传送 B_1 的广播信息。这样，虚拟局域网限制了接收广播信息的工作站数，使得网络不会因传播过多的广播信息（即所谓的“广播风暴”）而引起性能恶化。在共享传输媒体的局域网中，网络总带宽的绝大部分都是由广播帧消耗的。

由于虚拟局域网是用户和网络资源的逻辑组合，因此可按照需要将有关设备和资源非常方便地重新组合，使用户从不同的服务器或数据库中存取所需的资源。

以太网交换机的种类很多。例如，“具有第三层特性的第二层交换机”和“多层交换机”。前者具有某些第三层的功能，如数据报的分片和对多播通信量的管理，而后者可根据第三层的 IP 地址对分组进行过滤。

4.5.2　虚拟局域网使用的以太网帧格式

1988 年 IEEE 批准了 802.3ac 标准，这个标准定义了以太网的帧格式的扩展，以便支持虚拟局域网。虚拟局域网协议允许在以大网的帧格式中插入一个 4 字节的标识符（见图 4-24），称为 VLAN 标记(tag)，用来指明发送该帧的工作站属于哪一个虚拟局域网。如果还使用原来的以太网帧格式，那么就无法划分虚拟局域网。

VLAN 标记字段的长度是 4 字节，插入在以太网 MAC 帧的源地址字段和长度/类型字段之间。VLAN 标记的前两个字节和原来的长度/类型字段的作用一样，但它总是设置为 0x8100（这个数值大于 0x0600，因此不是代表长度），称为 802.1Q 标记类型。当数据链路层检测到 MAC

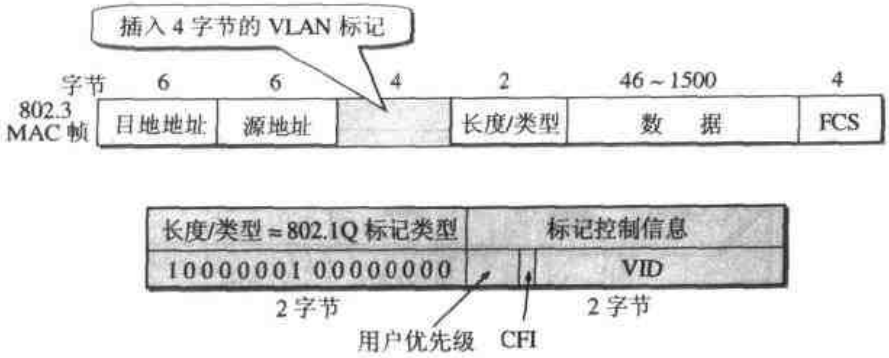


图 4-24 在以太网的帧格式中插入 VLAN 标记

帧的源地址字段后面的长度/类型字段的值是 0x8100 时，就知道现在插入了 4 字节的 VLAN 标记。于是就接着检查后两个字节的内容。在后面的两个字节中，前 3 个比特是用户优先级字段，接着的一个比特是**规范格式指示符 CFI** (Canonical Format Indicator)[①]，最后的 12 bit 是该虚拟局域网 VLAN **标识符** VID (VLAN ID)，它惟一地标志了这个以太网帧是属于哪一个 VLAN。

由于用于 VLAN 的以太网帧的首部增加了 4 个字节，因此以太网的最大长度从原来的 1518 字节（1500 字节的数据加上 18 字节的首部）变为 1522 字节。

4.6 高速以太网

速率达到或超过 100 Mb/s 的以太网称为**高速以太网**。下面简单介绍几种高速以太网技术。

4.6.1 100BASE-T 以太网

在 20 世纪 80 年代，很少有人想到以太网还会升级。然而在 1992 年 9 月 100 Mb/s 以太网的设想提出后仅过了 13 个月，100 Mb/s 以太网的产品就问世了。

100BASE-T 是在双绞线上传送 100 Mb/s 基带信号的星型拓扑以太网，仍使用 IEEE 802.3 的 CSMA/CD 协议，它又称为**快速以太网**(Fast Ethernet)。用户只要更换一张网卡，再配上一个 100 Mb/s 的集线器，就可很方便地由 10BASE-T 以太网直接升级到 100 Mb/s，而不必改变网络的拓扑结构。所有在 10BASE-T 上的应用软件和网络软件都可保持不变。100BASE-T 的网卡有很强的自适应性，能够自动识别 10 Mb/s 和 100 Mb/s。

1995 年 IEEE 已将 100BASE-T 的快速以太网定为正式的国际标准，其代号为 802.3u，是对现行的 802.3 标准的补充。快速以太网的标准得到了所有的主流网络厂商的支持。

100BASE-T 容易掌握，可以使用交换式集线器提供很好的服务质量，可以在全双工方式下工作而无冲突发生。因此，CSMA/CD 协议对全双工方式工作的快速以太网是不起作用的（但在半双工方式工作时则一定要使用 CSMA/CD 协议）。可能读者会问，不使用 CSMA/CD 协议为什么还能够叫作以太网呢？这是因为快速以太网使用的 MAC 帧格式仍然是 802.3 标准

① 注：所谓“规范格式”就是指地址的十六进制表示中每一个字节的最低位（见 IEEE 802 标准）代表规范格式地址中相应字节的最低位。“非规范格式”就是指地址的十六进制表示中每一个字节的最高位（见 IEEE 802 标准）代表规范格式地址中相应字节的最低位。CFI 置 1 表示是规范格式。

规定的帧格式。

然而 802.3u 的标准未包括对同轴电缆的支持。这表示想从细缆以太网升级到快速以太网的用户必须重新布线。但现在大多数安装场地正在向无屏蔽双绞线(UTP)布线过渡，因此这一问题将会逐渐地淡化。

100 Mb/s 以太网的新标准改动了原 10 Mb/s 以太网的某些规定。这里最主要的原因是要在数据发送速率提高时使参数 a 仍保持不变（或保持为较小的数值）。在 4.2.3 节已经给出了参数 a 的公式，现在再给出在下面：

$$a=\frac{\tau}{T_0}=\frac{\tau}{L/C}=\frac{\tau C}{L}$$

可以看出，当数据率 C 提高到 10 倍时，为了保持参数 a 不变，可以将帧长 L 也增大到 10 倍，也可以将网络电缆长度（因而使 τ）减小到原有数值的十分之一。

在 100 Mb/s 的以太网中采用的方法是保持最短帧长不变，但将一个网段的最大电缆长度减小到 100 m。帧间时间间隔从原来的 9.6 μs 改为现在的 0.96 μs。新标准还规定了以下三种不同的物理层标准：

（1）100BASE-TX　使用 2 对 UTP 5 类线或屏蔽双绞线(STP)，其中一对用于发送，另一对用于接收。信号的编码采用“多电平传输 3(MLT-3)”的编码方法，使信号的主要能量集中在 30 MHz 以下，以便减少辐射的影响。MLT-3 用三元制进行编码，即用正、负和零三种电平传送信号。其编码规则是：

- 当输入一个 0 时，下一个输出值不变。
- 当输入一个 1 时，下一个输出值要变化：若前一个输出值为正值或负值，则下一个输出值为零；若前一个输出值为零，则下一个输出值与上次的一个非零输出值的符号相反。

（2）100BASE-FX　使用 2 根光纤，其中一根用于发送，另一根用于接收。信号的编码采用 4B/5B-NRZI 编码。NRZI 即不归零 1 制（当“1”出现时信号电平在正值与负值之间变化一次），4B/5B 编码就是将数据流中的每 4 bit 作为一组(Block)，然后按编码规则将每一个组转换成为 5 bit，其中至少有 2 个“1”，保证信号码元至少发生两次跳变。

在标准中将上述的 100BASE-TX 和 100BASE-FX 合在一起都称为 100BASE-X。

（3）100BASE-T4　使用 4 对 UTP 3 类线或 5 类线，这是为已使用 UTP 3 类线的大量用户而设计的。信号的编码采用 8B6T-NRZ（不归零）的编码方法。8B6T 编码是将数据流中的每 8 bit 作为一组，然后按编码规则转换为每组 6 bit 的三元制(Ternary)码元。它同时使用 3 对线同时传送数据（每一对线以 $33\frac{1}{3}$ Mb/s 的速率传送数据），用 1 对线作为碰撞检测的接收信道。

4.6.2　吉比特以太网

在 1995 年以前，很多人都想不到以太网能工作在吉比特的速率（即 Gb/s 量级的速率）。他们认为在这样的速率下惟一能使用的技术恐怕只有 ATM 了。然而到了 1996 年夏季，吉比特以太网（又称为**千兆以太网**）的产品已经问市。IEEE 在 1997 年通过了吉比特以太网的标准 802.3z，它在 1998 年成为了正式标准。由于吉比特以太网仍使用 CSMA/CD 协议并与现有的以太网兼容，这就使得 ATM 在局域网的范围更加缺乏竞争力。虽然吉比特以太网也是一种高速局域网，但由于它发展很快，而且还能继续升级为 10 吉比特以太网，因此我们将它单

列为一节来讨论。

吉比特以太网的标准 802.3z 考虑了以下几个要点：

（1）允许在 1 Gb/s 下全双工和半双工两种方式工作。

（2）使用 802.3 协议规定的帧格式。

（3）在半双工方式下使用 CSMA/CD 协议（全双工方式不需要使用 CSMA/CD 协议）。

（4）与 10BASE-T 和 100BASE-T 技术向后兼容。

吉比特以太网可用作现有网络的主干网，也可在高带宽的应用场合中（如医疗图像或 CAD 的图形等）用来连接工作站和服务器。

吉比特以太网的物理层使用两种成熟的技术：一种来自现有的以太网，另一种则是 ANSI 制订的**光纤通道**(Fibre Channel)。采用成熟技术就能大大缩短吉比特以太网标准的开发时间。

吉比特以太网的物理层共有以下两个标准[CUNN99]：

（1）1000BASE-X（802.3z 标准）

1000BASE-X 标准是基于光纤通道的物理层，即 FC-0 和 FC-1。使用的媒体有三种：

- 1000BASE-SX　SX 表示短波长（使用 850 nm 激光器）。使用纤芯直径为 62.5 μm 和 50 μm 的多模光纤时，传输距离分别为 275 m 和 550 m。
- 1000BASE-LX　LX 表示长波长（使用 1300 nm 激光器）。使用纤芯直径为 62.5 μm 和 50 μm 的多模光纤时，传输距离为 550 m。使用纤芯直径为 10 μm 的单模光纤时，传输距离为 5 km。
- 1000BASE-CX　CX 表示铜线。使用两对短距离的屏蔽双绞线电缆，传输距离为 25 m。

（2）1000BASE-T(802.3ab 标准）

1000BASE-T 是使用 4 对 5 类线 UTP，传送距离为 100 m。

吉比特以太网在工作在半双工方式时，就必须进行碰撞检测。由于数据率提高了，因此只有减小最大电缆长度或增大帧的最小长度，才能使参数 a 保持为较小的数值。若将吉比特以太网最大电缆长度减小到 10 m，那么网络的实际价值就大大减小。而若将最短帧长提高到 640 字节，则在发送短数据时开销又嫌太大。因此吉比特以太网仍然保持一个网段的最大长度为 100 m，但采用了“**载波延伸**”(carrier extension)的办法，使最短帧长仍为 64 字节（这样可以保持兼容性），同时将争用时间增大为 512 字节。凡发送的 MAC 帧长不足 512 字节时，就用一些特殊字符填充在帧的后面，使 MAC 帧的发送长度增大到 512 字节，但这对有效载荷并无影响（见图 4-25）。接收端在收到以太网的 MAC 帧后，要将所填充的特殊字符删除后才向高层交付。当原来仅 64 字节长的短帧填充到 512 字节时，所填充的 448 字节就造成了很大的开销。

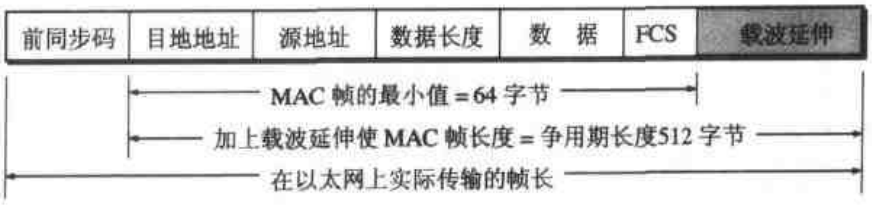


图 4-25　在短 MAC 帧后面加上载波延伸

为此，吉比特以太网还增加一种功能称为**分组突发**(packet bursting)。这就是当很多短帧

要发送时，第一个短帧要采用上面所说的载波延伸的方法进行填充。但随后的一些短帧则可一个接一个地发送，它们之间只需留有必要的帧间最小间隔即可。这样就形成可一串分组的突发，直到达到 1500 字节或稍多一些为止（见图 4-26，图中最后加入一个分组使整个以太网的长度略大于 1500 字节）。



图 4-26　分组突发可连续发送多个短分组

当吉比特以太网工作在全双工方式时（即通信双方可同时进行发送和接收数据），不使用载波延伸和分组突发。

吉比特以太网交换机可以直接与多个图形工作站相连。也可用作百兆以太网的主干网，与百兆比特或吉比特以太网集线器相连，然后再和大型服务器连接在一起。图 4-27 是吉比特以太网的一种配置举例。

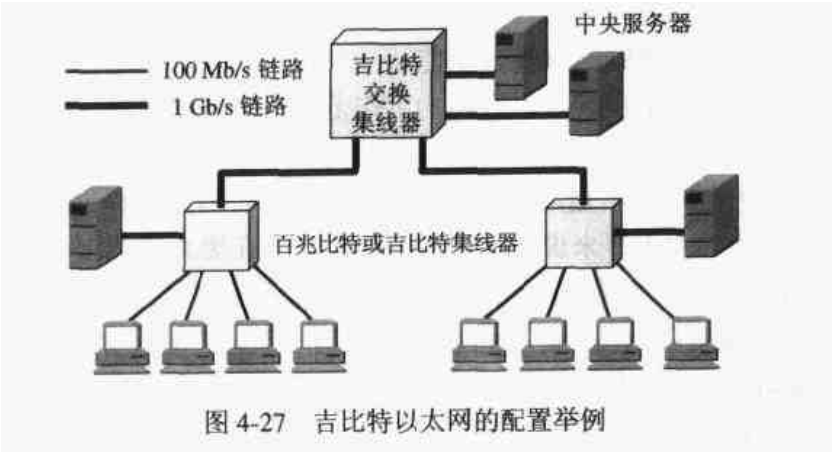


图 4-27　吉比特以太网的配置举例

4.6.3　10 吉比特以太网

就在吉比特以太网标准 802.3z 通过后不久，在 1999 年 3 月，IEEE 成立了**高速研究组 HSSG (High Speed Study Group)**，其任务是致力于 10 吉比特以太网的研究[W-10GE]。10 吉比特以太网(10GE)的标准由 IEEE 802.3ae 委员会进行制定，10 吉比特以太网的正式标准已在 2002 年 6 月完成。10 吉比特以太网也就是万兆以太网。

10 吉比特以太网并非将吉比特以太网的速率简单地提高到 10 倍。这里有许多技术上的问题要解决。下面是 10 吉比特以太网的主要特点。

10 吉比特以太网的帧格式与 10 Mb/s，100 Mb/s 和 1 Gb/s 以太网的**帧格式完全相同**。10 吉比特以太网还保留了 802.3 标准规定的**以太网最小和最大帧长**。这就使用户在将其已有的以太网进行升级时，仍能和较低速率的以太网很方便地通信。

由于数据率很高，10 吉比特以太网不再使用铜线而**只使用光纤作为传输媒体**。它使用长距离（超过 40 km）的光收发器与单模光纤接口，以便能够工作在广域网和城域网的范围。

10 吉比特以太网也可使用较便宜的多模光纤，但传输距离为 65～300 m。

10 吉比特以太网**只工作在全双工方式，因此不存在争用问题，也不使用 CSMA/CD 协议。**这就使得 10 吉比特以太网的传输距离不再受进行碰撞检测的限制而大大提高了。

吉比特以太网的物理层是使用已有的光纤通道的技术，而 10 吉比特以太网的物理层则是新开发的。10 吉比特以太网有两种不同的物理层：

（1）**局域网物理层** LAN PHY。局域网物理层的数据率是 10.000 Gb/s（这表示是精确的 10 Gb/s），因此一个 10 吉比特以太网交换机可以支持正好 10 个吉比特以太网端口。

（2）可选的**广域网物理层** WAN PHY。广域网物理层具有另一种数据率，这是为了和所谓的“Gb/s”的 SONET/SDH（即 OC-192/STM-64）相连接。我们知道，OC-192/STM-64 的准确数据率并非精确的 10 Gb/s（平时是为了简单就称它是 10 Gb/s 的速率）而是 9.95328 Gb/s。在去掉帧首部的开销后，其有效载荷的数据率只有 9.584 64 Gb/s。因此，为了使 10 吉比特以太网的帧能够插入到 OC-192/STM-64 帧的有效载荷中，就要使用可选的广域网物理层，其数据率为 9.953 28 Gb/s。显然，SONET/SDH 的“10 Gb/s”速率不可能支持 10 个吉比特以太网的端口，而只是能够与 SONET/SDH 相连接。

需要注意的是，10 吉比特以太网并没有 SONET/SDH 的同步接口而只有异步的以太网接口。因此，10 吉比特以太网在和 SONET/SDH 连接时，出于经济上的考虑，它只是具有 SONET/SDH 的某些特性，如 OC-192 的链路速率、SONET/SDH 的组帧格式等，但 WAN PHY 与 SONET/SDH 并不是全部都兼容的。例如，10 吉比特以太网没有 TDM 的支持，没有使用分层的精确时钟，也没有完整的网络管理功能。

由于 10 吉比特以太网的出现，以太网的工作范围已经从局域网（校园网、企业网）扩大到城域网和广域网，从而实现了端到端的以太网传输。这种工作方式的好处是：

（1）以太网是一种经过证明是成熟的技术，无论是因特网服务提供者 ISP 还是端用户都很愿意使用以太网。当然对 ISP 来说，使用以太网还需要在更大的范围进行试验。

（2）以太网的互操作性也很好，不同厂商生产的以太网都能可靠地进行互操作。

（3）在广域网中使用以太网时，其价格大约只有 SONET 的五分之一和 ATM 的十分之一。以太网还能够适应多种的传输媒体，如铜缆、双绞线以及各种光缆。这就使具有不同传输媒体的用户在进行通信时不必重新布线。

（4）端到端的以太网连接使帧的格式全都是以太网的格式，而不需要再进行帧的格式转换，这就简化了操作和管理。但是，以太网和现有的其他网络，如帧中继或 ATM 网络，仍然需要有相应的接口才能进行互连。

回顾过去的历史，我们看到 10 Mb/s 以太网最终淘汰了速率比快 60%的 16 Mb/s 的令牌环，100 Mb/s 的快速以太网也使曾经是最快的局域网/城域网的 FDDI 变成历史。吉比特以太网和 10 吉比特以太网的问世，使以太网的市场占有率进一步地得到提高，使得 ATM 在城域网和广域网中的地位受到更加严峻的挑战。10 吉比特以太网是 IEEE 802.3 标准在速率和距离方面的自然演进。以太网从 10 Mb/s 到 10 Gb/s 的演进证明了以太网是：

（1）可扩展的（从 10 Mb/s 到 10 Gb/s）。

（2）灵活的（多种媒体、全/半双工、共享/交换）。

（3）易于安装。

（4）稳健性好。

4.7 其他种类的高速局域网

4.7.1 100VG-AnyLAN 局域网

标准为 802.12 的 100VG-AnyLAN 也是一种使用集线器的 100 Mb/s 高速局域网，它常简写为 100VG。VG 代表 Voice Grade（话音级，这种局域网可使用 3 类线），而 Any 则表示能使用多种传输媒体，并可支持 IEEE 802.3 和 802.5 的数据帧。100VG 的产品在 1994 年推出，其标准是 802.12。

100VG 是一种无碰撞局域网，能更好地支持多媒体传输。在网络上可获得高达 95%的吞吐量。在媒体接入控制 MAC 子层运行一种新的协议，叫做**需求优先级**(demand priority)协议。各工作站有数据要发送时，要向集线器发出请求。每个请求都标有优先级别。一般的数据为低优先级，而对时间敏感的多媒体应用的数据（如话音、活动图像）则可定为高优先级。集线器使用一种循环仲裁过程来管理网络的结点，因而可保证对时间敏感的一些应用提供所需的实时服务。

100VG 是惠普(HP)公司推出的。但事实证明 100VG 的生命力并不强。这主要是因为：

（1）100VG 不是以太网，它和广大用户使用的以太网并不兼容。

（2）100VG 要求使用 4 对芯线，但有的安装场地只有 2 对芯线可供使用。

（3）100VG 不支持全双工方式，因此其速率只能是 100 Mb/s。

（4）100VG 基本上是 HP 公司的专有技术，而主要的集线器和网卡制造厂商都是支持 100BASE-T。

这些因素使 100VG 在市场的激烈竞争中失败了。

4.7.2 光纤分布式数据接口 FDDI

光纤分布式数据接口 FDDI (Fiber Distributed Data Interface)是一个使用光纤作为传输媒体的令牌环形网。它先是在 ANSI 的标准委员会 X3T9.5 通过为美国的标准，随后被 ISO 通过为国际标准 ISO 9314。FDDI 也常被划分在城域网 MAN 的范围。FDDI 的产品在 1988 年问世。

FDDI 主要是用作校园环境的主干网，其主要特性如下：

（1）使用基于 IEEE 802.5 令牌环标准的 MAC 协议，分组长度最大为 4500 字节。

（2）利用多模光纤进行传输，并使用有容错能力的双环拓扑。

（3）数据率为 100 Mb/s，光信号码元传输速率为 125 Mbaud。

（4）可以安装 1000 个物理连接（若都是双连接站，则为 500 个站）。最大站间距离为 2 km（多模光纤），环路长度为 100 km，即光纤总长度为 200 km。

（5）具有动态分配带宽的能力，故能同时提供同步和异步数据服务。

FDDI 采取了**自恢复**措施，可以大大地提高网络的可靠性。这种措施是使用两个数据传输方向相反的环路。在正常情况下，只有一个方向的环路在工作。这个工作的环路叫做**主环**，而另一个不工作的环路叫做**次环**（图 4-28(a)）。当环路出现故障时，例如，A 和 B 之间的链路断开了（图 4-28(b)），那么 FDDI 可自动重新配置，同时启动次环工作，并在 A 站和 B 站将主环和次环接通，使整个网络的四个站点仍然保持连通。当站点出现故障时，例如站点 A 不能工作了（图 4-28(c)），那么 FDDI 同样可启动次环工作，并在 B 站和 D 站将主环和次环接通，使站点 B, C 和 D 保持连通。当出故障的链路或站点修好后，整个 FDDI 网络又恢复到原来的主环工作状态。

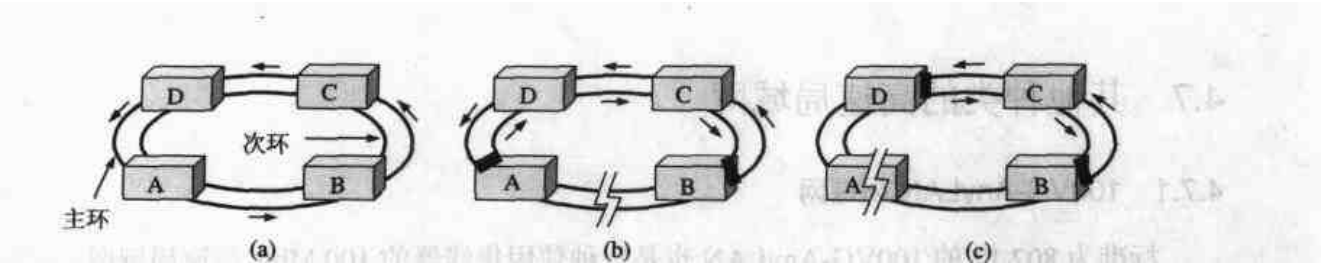


图 4-28　具有双环的 FDDI：(a)正常情况；(b)链路出故障；(c)站点出故障

不难看出，当主环和次环都工作时，FDDI 环路的总长度大约增加一倍。当出现多处故障时，FDDI 将变为多个分离的小环形网继续工作。

拥有速率为 100 Mb/s 的 FDDI 在 20 世纪 90 年代初期曾获得了较快的发展，也曾被预测为“下一代的局域网”。然而 FDDI 从未拥有过很大的市场。这是因为 FDDI 的芯片过于复杂因而价格昂贵。自从快速以太网大量进入市场后，在 100 Mb/s 局域网的领域中，已很少有人愿意再使用 FDDI 了。

4.7.3　高性能并行接口 HIPPI

高性能并行接口 HIPPI (HIgh-Performance Parallel Interface)主要用于超级计算机与一些外围设备（如海量存储器、图形工作站等）的高速接口。1987 年设计的 HIPPI 的数据传送标准是 800 Mb/s。这是因为对于 1024 × 1024 像素的画面，若每个像素使用 24 bit 的色彩编码和每秒 30 个画面，则总的数据率为 750 Mb/s。以后，又制定了 1600 Mb/s 和 6.4 Gb/s 的数据率标准。HIPPI 是一个 ANSI 标准。

4.7.4　光纤通道

在 1987 年设计 HIPPI 时，光纤还很贵，因此只好用廉价的双绞线进行数据的传输。现在光纤技术已很成熟，因此产生了**光纤通道**(Fibre Channel)。这里“光纤通道”已是一个专用名词（并非任何使用光纤的连接都可称为“光纤通道”），很多英文资料在提到这一名词时是用大写的 F 和 C。

光纤通道可处理数据通道和网络的连接。它可用来传送数据通道，包括 HIPPI，SCSI 以及 IBM 主机所用的复用器通道，也可用来传送网络的分组，如 IEEE 802、IP 以及 ATM 的分组。光纤通道的基本结构是与输入和输出端口连接的一个交叉式交换机。光纤通道支持三种服务类：第一类服务是纯电路交换，保证按序交付；第二类服务是保证交付的分组交换；第三类服务是不保证交付的分组交换。

光纤通道的物理媒体可使用单模光纤和多模光纤(50 μm 和 62.5 μm 两种)。使用单模光纤可传输 10 km，而使用多模光纤的传输距离为 175 m 至 10 km，取决于传输速率。光纤通道还可使用视频电缆(传输距离为 25～100 m)、小同轴电缆(传输距离为 10～35 m)以及屏蔽双绞线 STP (传输距离为 50～100 m，分别对应于数据率为 200 和 100 Mb/s)。

4.8　无线局域网

4.8.1　无线局域网的组成

在局域网刚刚问世后的一段时间，无线局域网的发展比较缓慢，其原因是：价格贵、数

据率低、安全性较差，以及使用登记手续复杂（使用无线电频率必须得到有关部门的批准）。但自 20 世纪 80 年代末以来，由于人们工作和生活节奏的加快以及移动通信技术的飞速发展，无线局域网也已开始进入市场。无线局域网提供了移动接入的功能，这就给许多需要发送数据但又不能坐在办公室的工作人员提供了方便。当一个工厂跨越的面积很大时，若要将各个部门都用电缆连接成网，其费用可能很高。但若使用无线局域网，不仅节省了投资，而且建网的速度也会较快。另外，当大量持有笔记本电脑的用户在一个地方同时要求上网时（如在图书馆或购买股票的大厅里），若用电缆连网，恐怕连铺设电缆的位置都很难找到。而用无线局域网则比较容易。例如，清华大学图书馆已实现了无线局域网上网。进入图书馆的读者只需请图书馆的工作人员在其笔记本电脑中插入一个无线上网卡并进行简单的设置后。即可在图书馆阅览室的任何角落连接到因特网。该无线局域网使用 802.11b 协议，数据率为 11 Mb/s，这对查询图书资料和收发电子邮件都是足够快的。

请读者注意，**便携站**(portable station)和**移动站**(mobile station)表示的意思并不一样。便携站当然是便于移动的，但便携站在工作时其位置是固定不变的。而移动站不仅能够移动，而且还可以在移动的过程中进行通信（正在进行的应用程序感觉不到计算机位置的的变化，也不因计算机位置的移动而中断运行）。移动站一般都是使用电池供电。

1997 年 IEEE 制订出无线局域网的协议标准 802.11，ISO/IEC 也批准了这一标准，其编号为 ISO/IEC 8802-11[W-802.11]。有关无线局域网的标准都可从因特网下载[W-IEEE802]。802.11 是一个非常复杂的标准，本节只能介绍其中的主要特点。

无线局域网可分为两大类。第一类是**有固定基础设施的**，第二类是无**固定基础设施的**。所谓“固定基础设施”是指预先建立起来的、能够覆盖一定地理范围的一批固定基站。大家经常使用的蜂窝移动电话就是利用电信公司预先建立的、覆盖全国的大量固定基站来接通用户手机拨打的电话。

对于第一类有固定基础设施的无线局域网，802.11 标准规定无线局域网的最小构件是**基本服务集 BSS** (Basic Service Set)。一个基本服务集 BSS 包括一个基站和若干个移动站，所有的站在本 BSS 以内都可以直接通信，但在和本 BSS 以外的站通信时都必须通过本 BSS 的基站。一个基本服务集 BSS 所覆盖的地理范围叫作一个**基本服务区 BSA** (Basic Service Area)。基本服务区 BSA 和无线移动通信的蜂窝小区相似。在无线局域网中，一个基本服务区 BSA 的范围可以有几十米的直径。

在 802.11 标准中，基本服务集里面的基站叫做**接入点 AP** (Access Point)，但其作用和网桥相似。一个基本服务集可以是孤立的，也可通过接入点 AP 连接到一个主干**分配系统 DS** (Distribution System)，然后再接入到另一个基本服务集，这样就构成了一个**扩展的服务集 ESS** (Extended Service Set)（图 4-29）。分配系统的作用就是使扩展的服务集 ESS 对上层的表现就像一个基本服务集 BSS 一样。分配系统可以使用以太网（这是最常用的）、点对点链路或其他无线网络。扩展服务集 ESS 还可为无线用户提供到非 802.11 无线局域网（例如，到有线连接的因特网）的接入。这种接入是通过叫做**门桥**(portal)的设备来实现的。门桥也是 802.11 定义的新名词，其实它的作用就相当于一个网桥。在一个扩展服务集内的几个不同的基本服务集也可能有相交的部分。在图 4-29 中还给出了移动站 A 从某一个基本服务集漫游到另一个基本服务集，而仍然可保持与另一个移动站 B 进行通信。当然 A 在不同的基本服务集所使用的接入点 AP 并不相同。基本服务集的服务范围是由移动设备所发射的电磁波的辐射范围确定的，在图 4-29 中用一个椭圆来表示基本服务集的服务范围，当然实际上的服务范围可能是很不规则的几何形状。

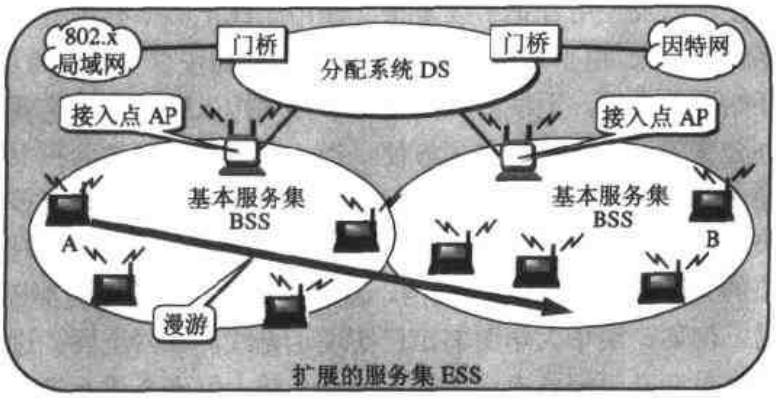


图 4-29　IEEE 802.11 的基本服务集 BSS 和扩展服务集 ESS

802.11 标准并没有定义如何实现漫游，但定义了一些基本的工具。例如，一个移动站若要加入到一个基本服务集 BSS，就必须先选择一个接入点 AP，并与此接入点建立**关联**(association)。此后，这个移动站就可以通过该接入点来发送和接收数据。若移动站使用**重建关联**(reassociation)服务，就可将这种关联转移到另一个接入点。当使用**分离**(dissociation)服务时，就可终止这种关联。移动站与接入点建立关联的方法有两种。一种是被动扫描，即移动站等待接收接入站周期性发出的**信标帧**(beacon frame)。另一种是主动扫描，即移动站主动发出**探测请求帧**(probe request frame)，然后等待从接入点发回的**探测响应帧**(probe response frame)。

另一类无线局域网是无固定基础设施的无线局域网，它又叫作**自组网络**(ad hoc network)[①]。这种自组网络没有上述基本服务集中的接入点 AP 而是由一些处于平等状态的移动站之间相互通信组成的临时网络（图 4-30）。图中还画出了当移动站 A 和 E 通信时，是经过 A→B，B→C，C→D 和最后 D→E 这样一连串的存储转发过程。因此在从源结点 A 到目的结点 E 的路径中的移动站 B，C 和 D 都是转发结点，这些结点都具有路由器的功能。由于自组网络没有预先建好的网络固定基础设施（基站），因此自组网络的服务范围通常是受限的，而且自组网络一般也不和外界的其他网络相连接。移动自组网络也就是**移动分组无线网络**。

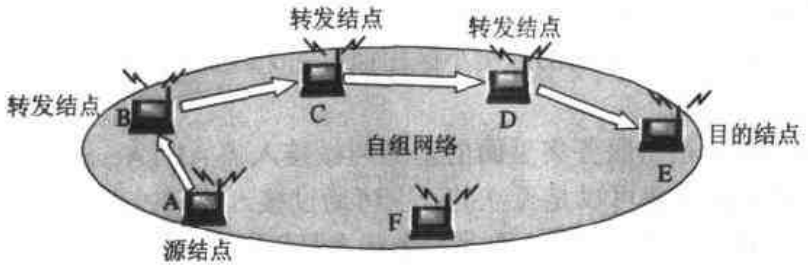


图 4-30　由处于平等状态的一些便携机构成的自组网络

自组网络通常是这样构成的：一些可移动的设备发现在它们附近还有其他的可移动设备，并且要求和其他移动设备进行通信。由于便携式电脑的大量普及，自组网络的组网方式已受到人们的广泛关注。由于在自组网络中的每一个移动站都要参与到网络中的其他移

① 注：拉丁语 ad hoc 本来的意思是“仅为此目的(for this purpose only)”，并且通常还有“临时的”含义。译成中文就是“特定的”。直译 ad hoc network 就是“特定网络”，但由于这种网络的组成并不需要使用固定的基础设施，因此可意译为“自组网络”，表明仅依靠移动站自身而不需要固定基站就能组成网络。

动站的路由的发现和维护，同时由移动站构成的网络拓扑有可能随时间变化得很快，因此在固定网络中行之有效的一些路由选择协议对移动自组网络已不适用，这样，路由选择协议在自组网络中就引起了特别的关注。另一个重要问题是多播。在移动自组网络中往往需要将某个重要信息同时向多个移动站传送。这种多播比固定结点网络的多播要复杂得多，需要有实时性好而效率又高的多播协议。在移动自组网络中，安全问题也是一个更为突出的问题。在 IETF 下面设有一个专门研究移动自组网络的工作组 MANET (Mobile Ad-hoc NETworks) [W-MANET]，读者可在 MANET 网站查阅到有关移动自组网络的技术资料。

移动自组网络在军用和民用领域都有很好的应用前景。在军事领域中，由于战场上往往没有预先建好的固定接入点，但携带了移动站的战上就可以利用临时建立的移动自组网络进行通信。这种组网方式也能够应用到作战的地面车辆群和坦克群，以及海上的舰艇群、空中的机群。由于每一个移动设备都具有路由器的转发分组的功能，因此分布式的移动自组网络的生存性非常好。在民用领域，开会时持有笔记本电脑的人可以利用这种移动自组网络方便地交换信息，而不受笔记本电脑附近没有电话线插头的限制。当出现自然灾害时，在抢险救灾时利用移动自组网络进行及时的通信往往也是很有效的，因为这时事先已建好的固定网络基础设施（基站）可能已经都被破坏了。

顺便指出，**移动自组网络和移动 IP 并不相同**。移动 IP 技术使漫游的主机可以用多种方式连接到因特网。漫游的主机可以直接连接到或通过无线链路连接到固定网络上的另一个子网。支持这种形式的主机移动性需要地址管理和增加协议的互操作性，但移动 IP 的核心网络功能仍然是基于在固定互联网中一直在使用的各种路由选择协议。但移动自组网络是将移动性扩展到无线领域中的自治系统，它具有自己特定的路由选择协议，并且可以不和因特网相连。即使在和因特网相连时，移动自组网络也是以**残桩网络**(stub network)方式工作的。所谓“残桩网络”就是通信量可以进入残桩网络，也可以从残桩网络发出，但不允许外部的通信量穿越残桩网络。

下面我们对有固定基础设施的无线局域网进行更加细致的讨论。

4.8.2 802.11 标准中的物理层

802.11 标准规定的物理层相当复杂，1997 年制订了第一部分，叫做 802.11。在 1999 年又制订了剩下的两部分，即 802.11a 和 802.11b。

（1）802.11 的物理层有以下三种实现方法：

- **跳频扩频 FHSS**　**跳频扩频 FHSS** (Frequency Hopping Spread Spectrum)是扩频技术中常用的一种。它使用 2.4 GHz 的 ISM 频段（即 2.4000～2.4835 GHz）[①]。共有 79 个信道可供跳频使用。第一个频道的中心频率为 2.402GHz，以后每隔 1 MHz 一个信道。因此每个信道可使用的带宽为 1 MHz。当使用二元高斯移频键控 GFSK 时，基本接入速率为 1 Mb/s。当使用 4 元 GFSK 时，接入速率为 2 Mb/s。
- **直接序列扩频 DSSS**　**直接序列扩频 DSSS** (Direct Sequence Spread Spectrum)是另一种重要的扩频技术。它也使用 2.4 GHz 的 ISM 频段。当使用二元相对移相键控时，

① 注：ISM 为 Industrial, Scientific, and Medical（工业、科学与医药）的缩写。

基本接入速率为 1 Mb/s。当使用 4 元相对移相键控时，接入速率为 2 Mb/s。

- 红外线 IR　　**红外线 IR** (InfraRed)的波长为 850～950 nm，可用于室内传送数据。接入速率为 1～2 Mb/s。

（2）802.11a 的物理层工作在 5GHz 频带，不采用扩频技术而是采用**正交频分复用** OFDM，它也叫做**多载波调制技术**（载波数可多达 52 个）。可以使用的数据率为 6, 9, 12, 18, 24, 36, 48 和 56 Mb/s。

（3）802.11b 的物理层使用工作在 2.4 GHz 的直接序列扩频技术，数据率为 5.5 或 11 Mb/s。

4.8.3　802.11 标准中的 MAC 层

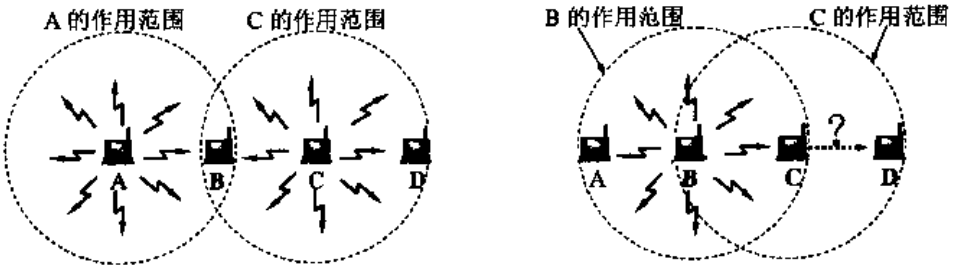
1．CSMA/CA 协议

虽然 CSMA/CD 协议已成功地应用于使用有线连接的局域网，但无线局域网却不能简单地搬用 CSMA/CD 协议。这里主要有两个原因：

第一，CSMA/CD 协议要求一个站点在发送本站数据的同时还必须不间断地检测信道，以便发现是否有其他的站也在发送数据，这样才能实现“碰撞检测”的功能。但在无线局域网的设备中要实现这种功能就花费过大。

第二，更重要的是，即使我们能够实现碰撞检测的功能，并且当我们在发送数据时检测到信道是空闲的，在接收端仍然有可能发生碰撞。这就表明，碰撞检测对无线局域网没有什么用处。

产生这种结果是由无线信道本身特点决定的。具体地说，这是由于无线电波能够向所有的方向传播，且其传播距离受限。当电磁波在传播过程中遇到障碍物时，其传播距离就更加受到限制。图 4-31 的例子表示了无线局域网的特殊问题。图中画有 4 个无线移动站，并假定无线电信号传播的范围是以发送站为圆心的一个圆形面积。



(a) A 和 C 同时向 B 发送信号，发生碰撞；　　(b) B 向 A 发送信号，影响 C 向 D 发送数据

图 4-31　无线局域网的问题

图 4-31(a)表示站 A 和 C 都想和 B 通信。但 A 和 C 相距较远，彼此都接收不到对方发送的信号。当 A 和 C 检测不到无线信号时，就都以为 B 是空闲的，因而都向 B 发送自己的数据。结果 B 同时收到 A 和 C 发来的数据，发生了碰撞。可见在无线局域网中，在发送数据前未检测到媒体上有信号还不能保证在接收端能够成功地接收到数据。这种未能检测出媒体上已存在的信号的问题叫做**隐蔽站问题**(hidden station problem)。

当移动站之间有障碍物时也有可能出现上述问题。例如，三个站 A, B 和 C 彼此距离都差不多，但 A 和 C 之间有一个座山，因此 A 和 C 都不能检测到对方发出的信号。若 A 和 C 同时向 B 发送数据就会发生碰撞（但 A 和 C 并不知道），则 B 无法正常接收。

图 4-31(b)给出了另一种情况。站 B 向 A 发送数据。而 C 又想和 D 通信。但 C 检测到媒体上有信号，于是就不敢向 D 发送数据。其实 B 向 A 发送数据并不影响 C 向 D 发送数据。这就是**暴露站问题**(exposed station problem)。在无线局域网中，在不发生干扰的情况下，可允许同时多个移动站进行通信。这点与总线式局域网有很大的差别。

除以上两个原因外，无线信道还由于传输条件特殊，造成信号强度的动态范围非常大。这就使发送站无法使用碰撞检测的方法来确定是否发生了碰撞。

因此，无线局域网不能使用 CSMA/CD，而只能使用改进的 CSMA 协议。

改进的办法是将 CSMA 增加一个**碰撞避免**(Collision Avoidance)功能。于是 802.11 就使用 CSMA/CA 协议。而在使用 CSMA/CA 的同时还增加使用确认机制。

下面在讨论 CSMA/CA 协议之前，我们先介绍 802.11 的 MAC 层。

802.11 标准设计了独特的 MAC 层（图 4-32）。它通过**协调功能**(Coordination Function)来确定在基本服务集 BSS 中的移动站在什么时间能发送数据或接收数据。802.11 的 MAC 层在物理层的上面，它包括两个子层。在下面的一个子层是**分布协调功能 DCF** (Distributed Coordination Function)。DCF 在每一个结点使用 CSMA 机制的分布式接入算法，让各个站通过争用信道来获取发送权。因此 DCF 向上提供争用服务。另一个子层叫做**点协调功能 PCF** (Point Coordination Function)。PCF 是选项，自组网络就没有 PCF 子层。PCF 使用集中控制的接入算法（一般在接入点 AP 实现集中控制），用类似于探询的方法将发送数据权轮流交给各个站，从而避免了碰撞的产生。对于时间敏感的业务，如分组话音，就应使用提供无争用服务的点协调功能 PCF。

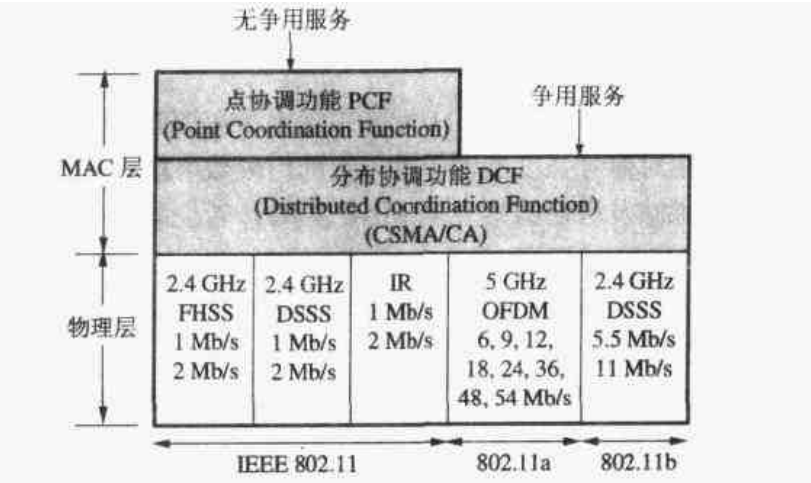


图 4-32　802.11 的 MAC 层

为了尽量避免碰撞，802.11 规定，所有的站在完成发送后，必须再等待一段很短的时间（继续监听）才能发送下一帧。这段时间的通称是**帧间间隔 IFS** (InterFrame Space)。帧间间隔的长短取决于该站打算发送的帧的类型。高优先级帧需要等待的时间较短，因此可优先获得发送权，但低优先级帧就必须等待较长的时间。若低优先级帧还没来得及发送而其他站的高优先级帧已发送到媒体，则媒体变为忙态因而低优先级帧就只能再推迟发送了。这样就减少了发生碰撞的机会。常用的三种帧间间隔如下（见图 4-33）：

（1）**SIFS**，即**短**(Short)帧间间隔，长度为 28 μs。SIFS 是最短的帧间间隔，用来分隔开属于一次对话的各帧。一个站应当能够在这段时间内从发送方式切换到接收方式。使用 SIFS

的帧类型有：ACK 帧、CTS 帧（在后面第 2 小节中讲）、由过长的 MAC 帧分片后的数据帧[①]，以及所有回答 AP 探询的帧和在 PCF 方式中接入点 AP 发送出的任何帧。

（2）**PIFS**，即**点协调功能帧间间隔**（比 SIFS 长），是为了在开始使用 PCF 方式时（在 PCF 方式下使用，没有争用）优先获得接入到媒体中。PIFS 的长度是 SIFS 加一个**时隙**(slot)长度（其长度为 50 μs），即 78 μs。时隙的长度是这样确定的：在一个基本服务集 BSS 内，当某个站在一个时隙开始时接入到媒体时，那么在下一个时隙开始时，其他站就都能检测出信道已转变为忙态。

（3）**DIFS**，即**分布协调功能帧间间隔**（最长的 IFS），在 DCF 方式中用来发送数据帧和管理帧。DIFS 的长度比 PIFS 再多一个时隙长度，因此 DIFS 的长度为 128 μs。

CSMA/CA 协议的原理可用图 4-33 来说明。

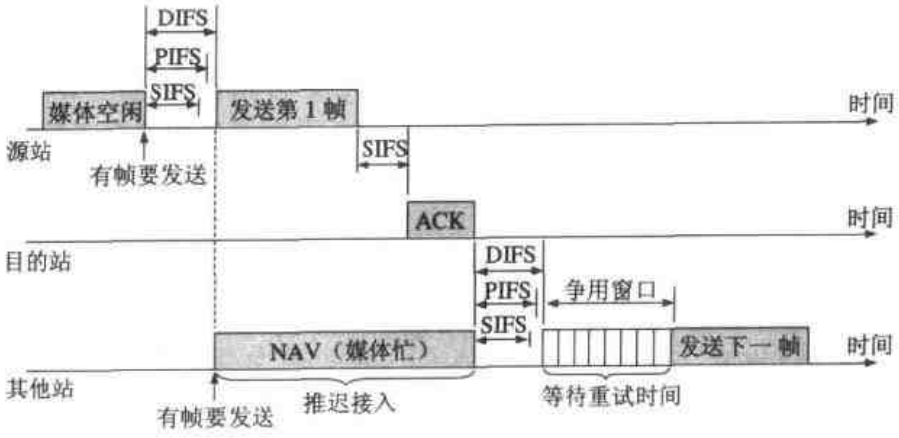


图 4-33　CSMA/CA 协议的工作原理

欲发送数据的站先检测信道。在 802.11 标准中规定了在物理层的空中接口进行物理层的载波监听。通过收到的相对信号强度是否超过一定的门限数值就可判定是否有其他的移动站在信道上发送数据。当源站发送它的第一个 MAC 帧时，若检测到信道空闲，则在等待一段时间 DIFS 后就可发送。

为什么信道空闲还要再等待呢？就是考虑到可能有其他的站有高优先级的帧要发送。如有，就要让高优先级帧先发送。

现在假定没有高优先级帧要发送，因而源站发送了自己的数据帧。目的站若正确收到此帧，则经过时间间隔 SIFS 后，向源站发送确认帧 ACK。若源站在规定时间内没有收到确认帧 ACK（由重传计时器控制这段时间），就必须重传此帧，直到收到确认为止，或者经过若干次的重传失败后放弃发送。

802.11 标准还采用了一种叫做**虚拟载波监听**(Virtual Carrier Sense)的机制，这就是让源站将它要占用信道的时间（包括目的站发回确认帧所需的时间）通知给所有其他站，以便使其他所有站在这一段时间都停止发送数据。这样就大大减少了碰撞的机会。“虚拟载波监听”是表示其他站并没有监听信道，而是由于其他站收到了“源站的通知”才不发送数据。这种效果好像是其他站都监听了信道。所谓“源站的通知”就是源站在其 MAC 帧首部中的第二个

① 注：因为无线信道的误码率比有线信道的高得多，因此无线局域网的 MAC 帧长应当短些，以便在出错重传时减小开销。这样，就必须将太长的帧进行分片。

字段“**持续时间**”中填入了在本帧结束后还要占用信道多少时间（以微秒为单位），包括目的站发送确认帧所需的时间。

当一个站检测到正在信道中传送的 MAC 帧首部的“持续时间”字段时，就调整自己的**网络分配向量** NAV (Network Allocation Vector)。NAV 指出了必须经过多少时间才能完成数据帧的这次传输，才能使信道转入到空闲状态。因此，信道处于忙态，或者是由于物理层的载波监听检测到信道忙，或者是由于 MAC 层的虚拟载波监听机制指出了信道忙。

图 4-33 指出，当信道从忙态变为空闲时，任何一个站要发送数据帧时，不仅都必须等待一个 DIFS 的间隔，而且还要进入**争用窗口**，并计算随机退避时间以便再次重新试图接入到信道。**请读者注意**，在以太网的 CSMA/CD 协议中，碰撞的各站执行退避算法是在发生了碰撞之后；但在 802.11 的 CSMA/CA 协议中，因为没有像以太网那样的碰撞检测机制，因此在**信道从忙态转为空闲时，各站就要执行退避算法**。这样做就减少了发生碰撞的概率（当多个站都打算占用信道）。802.11 也是使用二进制指数退避算法，但具体做法稍有不同。这就是：第 i 次退避就在 2^{2+i} 个时隙中随机地选择一个。这就是说，第 1 次退避是在 8 个时隙（而不是 2 个）中随机选择一个，而第 2 次退避是在 16 个时隙（而不是 4 个）中随机选择一个。

当某个想发送数据的站使用退避算法选择了争用窗口中的某个时隙后，就根据该时隙的位置设置一个**退避计时器**(backoff timer)。当退避计时器的时间减小到零时，就开始发送数据。也可能当退避计时器的时间还未减小到零时而信道又转变为忙态，这时就冻结退避计时器的数值，重新等待信道变为空闲，再经过时间 DIFS 后，继续启动退避计时器（从剩下的时间开始）。这种规定有利于继续启动退避计时器的站更早地接入到信道中。

应当指出，当一个站要发送数据帧时，仅在下面的情况下才**不使用**退避算法：检测到信道是空闲的，并且这个数据帧是它想发送的第一个数据帧。除此以外的所有情况，都必须使用退避算法。具体来说，就是：

（1）在发送它的第一个帧之前检测到信道处于忙态。

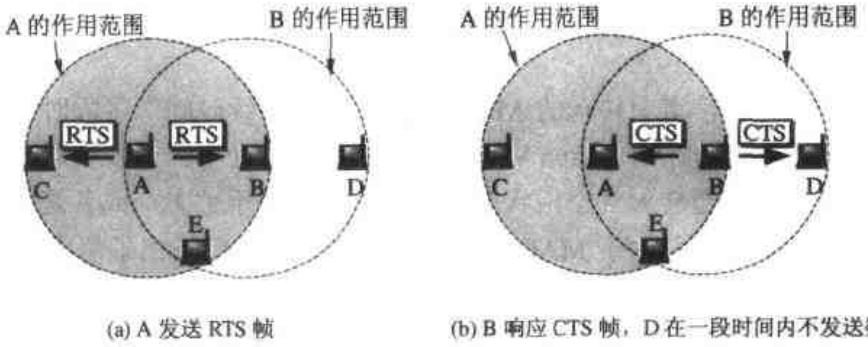
（2）在每一次的重传后。

（3）在每一次的成功发送后。

2. 对信道进行预约

为了更好地解决隐蔽站带来的碰撞问题，802.11 允许要发送数据的站对信道进行**预约**。具体的做法是这样的：如图 4-34(a)所示，源站 A 在发送数据帧之前先发送一个短的控制帧，叫做**请求发送 RTS** (Request To Send)，它包括源地址、目的地址和这次通信（包括相应的确认帧）所需的持续时间。若信道空闲，则目的站 B 就发送一个响应控制帧，叫做**允许发送 CTS** (Clear To Send)，如图 4-34(b)所示，它也包括这次通信所需的持续时间（从 RTS 帧中将此持续时间复制到 CTS 帧中）。A 收到 CTS 帧后就可发送其数据帧。下面讨论在 A 和 B 两个站附近的一些站将做出的反应。

C 处于 A 的传输范围内，但不在 B 的传输范围内。因此 C 能够收到 A 发送的 RTS，但经过一小段时间后，C 不会收到 B 发送的 CTS 帧。这样，在 A 向 B 发送数据时，C 也可以发送自己的数据给其他的站而不会干扰 B。请读者注意，C 收不到 B 的信号表明 B 也收不到 C 的信号。



(a) A 发送 RTS 帧　　(b) B 响应 CTS 帧，D 在一段时间内不发送数据

图 4-34　CSMA/CA 协议中的 RTS 和 CTS 帧

再观察 D。D 收不到 A 发送的 RTS 帧，但能收到 B 发送的 CTS 帧。因此 D 知道 B 将要和 A 通信，因此 D 在 A 和 B 通信的一段时间内不能发送数据，因而不会干扰 B 接收 A 发来的数据。

至于站 E，它能收到 RTS 和 CTS，因此 E 和 D 一样，在 A 发送数据帧和 B 发送确认帧的整个过程中都不能发送数据。

可见这种协议实际上就是在发送数据帧之前先对信道进行预约一段时间。

使用 RTS 和 CTS 帧会使整个网络的效率有所下降。但这两种控制帧都很短，其长度分别为 20 字节和 14 字节，与数据帧（最长可达 2346 字节）相比开销不算大。相反，若不使用这种控制帧，则一旦发生碰撞而导致数据帧重发，则浪费的时间就更多。虽然如此，但协议还是设有三种情况供用户选择：一种是使用 RTS 和 CTS 帧；另一种是只有当数据帧的长度超过某一数值时才使用 RTS 和 CTS 帧（显然，当数据帧本身就很短时，再使用 RTS 和 CTS 帧只能增加开销）；还有一种是不使用 RTS 和 CTS 帧。

虽然协议经过了精心设计，但碰撞仍然会发生。例如，B 和 C 同时向 A 发送 RTS 帧。这两个 RTS 帧发生碰撞后，使得 A 收不到正确的 RTS 帧因而 A 就不会发送后续的 CTS 帧。这时，B 和 C 像以太网发生碰撞那样，各自随机地推迟一段时间后重新发送其 RTS 帧。推迟时间的算法也是使用二进制指数退避。

图 4-35 给出了 RTS 和 CTS 帧以及数据帧和 ACK 帧的传输时间关系。在除源站和目的站以外的其他站中，有的在收到 RTS 帧后就设置其网络分配向量 NAV，有的则在收到 CTS 帧或数据帧后才设置其 NAV。因此图中画出了几种不同的 NAV 的设置。

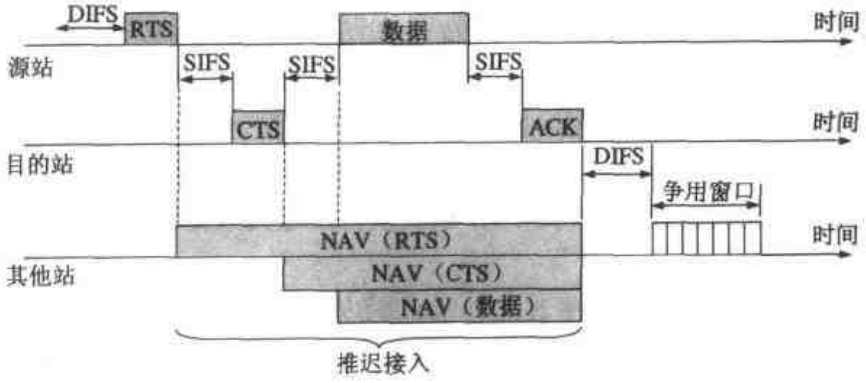


图 4-35　RTS 和 CTS 帧以及数据帧和 ACK 帧的传输时间关系

习题

4-01 局域网的主要特点是什么？为什么局域网采用广播通信方式而广域网不采用呢？

4-02 DIX 以太网和 802.3 以太网的帧格式有何异同之处？

4-03 一个 7 层楼，每层有一排共 15 间办公室。每个办公室的墙上设有一个插座。所有的插座在一个垂直面上构成一个正方形栅格网的结点，相邻插座间的垂直和水平距离均为 4 m。设任意两个插座之间都允许连上电缆（垂直、水平、斜线、……均可）。现要用电缆将它们连成：

（1）集线器在中央的星形网；

（2）以太网；

（3）令牌环形网。

试计算每种情况下所需的电缆长度。

4-04 数据率为 10 Mb/s 的以太网在物理媒体上的码元传输速率是多少波特？

4-05 以太网上只有两个站，它们同时发送数据，产生了碰撞。于是按二进制指数类型退避算法进行重传。重传次数记为 i，$i = 1, 2, 3, \ldots$。试计算第 1 次重传失败的概率、第 2 次重传失败的概率、第 3 次重传失败的概率，以及一个站成功发送数据之前的平均重传次数 I。

4-06 试说明 10BASE5，10BASE2，10BASE-T，1BASE5，10BROAD36 和 FOMAU 所代表的意思。

4-07 10 Mb/s 以太网升级到 100 Mb/s、1 Gb/s 和 10 Gb/s 时，都需要解决哪些技术问题？为什么以太网能够在发展的过程中淘汰掉自己的竞争对手，并使自己的应用范围从局域网一直扩展到城域网和广域网？

4-08 有 10 个站连接到以太网上。试计算以下三种情况下每一个站所能得到的带宽。

（1）10 个站都连接到一个 10 Mb/s 以太网集线器；

（2）10 个站都连接到一个 100 Mb/s 以太网集线器；

（3）10 个站都连接到一个 10 Mb/s 以太网交换机。

4-09 100 个站分布在 4 km 长的总线上。协议采用 CSMA/CD。总线速率为 5 Mb/s，帧平均长度为 1000 bit。试估算每个站每秒钟发送的平均帧数的最大值。传播时延为 5 μs/km。

4-10 在以下的条件下，分别重新计算上题，并解释所得结果。

（1）总线长度减小到 1 km。

（2）总线速率加倍。

（3）帧长变为 10 000 bit。

4-11 假定 1 km 长的 CSMA/CD 网络的数据率为 1 Gb/s。设信号在网络上的传播速率为 200 000 km/s。求能够使用此协议的最短帧长。

4-12 有一个使用集线器的以太网，每个站到集线器的距离为 d，数据发送速率为 C，帧长为 12 500 字节，信号在线路上的传播速率为 2.5×10^8 m/s。距离 d 为 25 m 或 2500 m，发送速率为 10 Mb/s 或 10 Gb/s。这样就有 4 种不同的组合。试利用公式(4-9)分别计算这 4 种不同情况下参数 a 的数值，并进行简单讨论。

4-13 为什么早期的以太网选择总线拓扑结构而不使用星形拓扑结构，但现在却改为使用星形

拓扑结构？

4-14 试比较以太网的 MAC 层协议和 HDLC 协议的相似点和不同点。

4-15 假定一个以太网上的通信量中的 80%是在本局域网上进行的，而其余的 20%的通信量是在本局域网和因特网之间进行的。另一个以太网的情况则反过来。这两个以太网一个使用以太网集线器，而另一个使用以太网交换机。你认为以太网交换机应当用在哪一个网络上？

4-16 以太网使用的 CSMA/CD 协议是以争用方式接入到共享信道。这与传统的时分复用 TDM 相比优缺点如何？

4-17 使用 CSMA/CD 协议时，若线路长度为 100 m，信号在线路上传播速率为 2×10^8 m/s。数据的发送速率为 1 Gb/s。试计算帧长分别为 512 字节、1500 字节和 64 000 字节时的参数 a 的数值，并进行简单讨论。

4-18 以太网交换机有何特点？用它怎样组成虚拟局域网？

4-19 网桥的工作原理和特点是什么？网桥与转发器以及以太网交换机有何异同？

4-20 以太网交换机最主要的特点是什么？用它怎样组成虚拟局域网？

4-21 FDDI 的主要特点有哪些？和以太网相比，优缺点各有哪些？

4-22 同时连接在两个总线局域网上的四个站以相等的概率随意选择一个总线向其他站发送数据。假定时间被划分为长度为 1 秒的时隙，而这四个站开始发送数据的时间只能选择在每一个时隙的开始，并且发送的帧长也是 1 秒。每一个站发送数据的概率都是 p。如果一个总线上同时有两个站发送数据就会产生冲突。试问概率 p 等于多少才能使产生冲突的概率最小？

4-23 现有五个站分别连接在三个局域网上，并且用两个网桥连接起来，如图 4-36 所示。每一个网桥的两个端口号都标明在图上。在一开始，两个网桥中的转发表都是空的。以后有以下各站向其他的站发送了数据帧，即 H_1 发送给 H_5，H_3 发送给 H_2，H_4 发送给 H_3，H_2 发送给 H_1。试将有关数据填写在表 4-1 中。

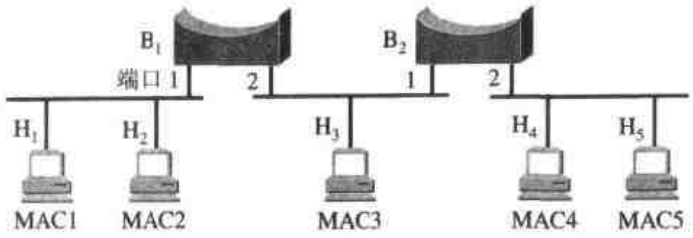


图 4-36 习题 4-23 的图

表 4-1 习题 4-23 的表

发送的帧	网桥 1 的转发表		网桥 2 的转发表		网桥 1 的处理	网桥 2 的处理
	站地址	端口	站地址	端口	（转发？丢弃？登记？）	（转发？丢弃？登记？）
$H_1\rightarrow H_5$						
$H_3\rightarrow H_2$						
$H_4\rightarrow H_3$						
$H_2\rightarrow H_1$						

4-24 无线局域网的 MAC 协议有哪些特点？为什么在无线局域网中不能使用 CSMA/CD 协议而必须使用 CSMA/CA 协议？结合隐蔽站问题和暴露站问题说明 RTS 帧和 CTS 帧的作用。

4-25 为什么在无线局域网上发送数据帧后要对方必须发回确认帧，而以太网就不需要对方发回确认帧？

4-26 无线局域网的 MAC 协议中的 SIFS，PIFS 和 DIFS 的作用是什么？

4-27 解释无线局域网中的名词：BSS, ESS, AP, BSA, Portal, DCF, PCF 和 NAV。

第5章 广 域 网

本章先讨论广域网基本概念，包括广域网所提供的两种服务——数据报和虚电路。接着要讨论分组转发，即网络交换结点怎样知道应通过哪条路径(path)才能将数据转发到所要通信的目的主机。这就要查找转发表。本章只讨论查找转发表的简单过程，而转发表是如何建立的则在下一章讨论。最后介绍三种广域网，即 X.25 网、帧中继网和采用 ATM 技术的广域网。

5.1 广域网的基本概念

5.1.1 广域网的构成

当主机之间的距离较远时，例如，相隔几十或几百公里，甚至几千公里，局域网显然就无法完成主机之间的通信任务。这时就需要另一种结构的网络，即**广域网**。

广域网由一些结点交换机以及连接这些交换机的链路组成。结点交换机执行将分组存储转发的功能。结点之间都是点到点连接，但为了提高网络的可靠性，通常一个结点交换机往往与多个结点交换机相连。受经济条件的限制，广域网都不使用局域网普遍采用的多点接入技术。从层次上考虑，广域网和局域网的区别很大，因为局域网使用的协议主要在数据链路层（还有少量物理层的内容），而广域网使用的协议在网络层。在广域网中的一个重要问题就是路由选择和分组转发。

然而广域网并没有严格的定义。通常广域网是指覆盖范围很广（远远超过一个城市的范围）的长距离网络。由于广域网的造价较高，一般都是由国家或较大的电信公司出资建造。广域网是因特网的核心部分，其任务是通过长距离（例如，跨越不同的国家）运送主机所发送的数据。连接广域网各结点交换机的链路都是高速链路，其距离可以是几千公里的光缆线路，也可以是几万公里的点对点卫星链路。因此广域网首先要考虑的问题是它的通信容量必须足够大，以便支持日益增长的通信量。

图 5-1 表示相距较远的局域网通过路由器与广域网相连，组成了一个覆盖范围很广的互联网。这样，局域网就可通过广域网与另一个相隔很远的局域网进行通信。互联网和路由器的工作原理将在第 6 章中讨论。路由器是一种特殊用途的主机，在图中将它画在两种网络之外。其实它也可同时画在两个网络之中，因为它既属于局域网也属于广域网。

像图 5-1 所示的互联网，即使覆盖范围很广，一般也不称它为广域网，因为在这种网络中，不同网络的“互连”才是它的最主要的特征。互联网必须使用路由器来连接，而广域网指的是单个的网络，它使用结点交换机连接各主机而不是用路由器来连接各网络。结点交换机和路由器都是用来转发分组，它们的工作原理相似。但区别是：结点交换机是在**单个网络**中转发分组，而路由器是在多个网络构成的**互联网**中转发分组。

广域网和局域网都是互联网的重要组成构件。尽管它们的价格和作用距离相差很远，但从互联网的角度来看，广域网和局域网却都是平等的(在学完第 6 章后就能更好地理解这点)。

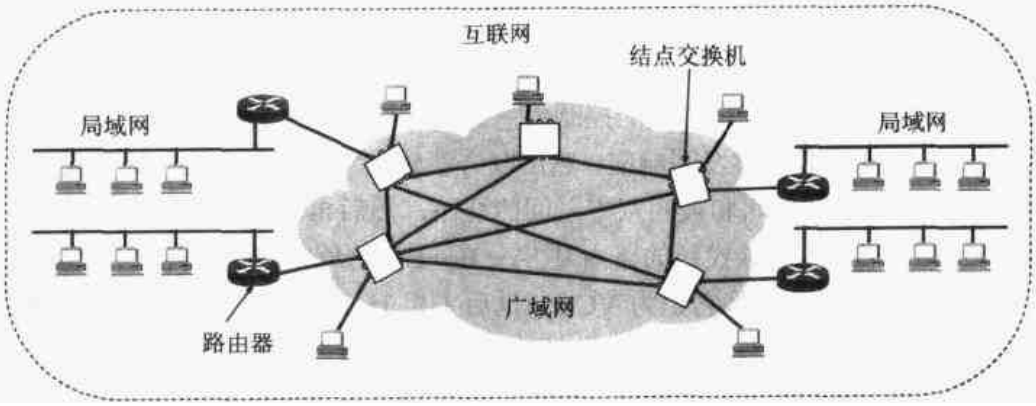


图 5-1　由局域网和广域网组成互联网

这里的一个关键就是广域网和局域网有一个共同点：连接在一个广域网或一个局域网上的主机在该网内进行通信时，**只需要使用其网络的物理地址即可**。

5.1.2　数据报和虚电路

从层次上看，广域网中的最高层就是网络层。网络层为接在网络上的主机所提供的服务可以有两大类，即**无连接的网络服务和面向连接的网络服务**。这两种服务的具体实现就是通常所谓的**数据报服务**和**虚电路服务**。

图 5-2(a)和(b)分别画出了网络提供数据报服务和提供虚电路服务的特点。网络层的用户是运输层实体，但为方便起见，可用主机作为网络层的用户。

我们先讨论数据报服务。

网络提供数据报服务的特点是：网络随时都可接受主机发送的分组（即数据报）。网络为每个分组独立地选择路由。网络只是**尽最大努力地**将分组交付给目的主机，但网络对源主机**没有任何承诺**。网络不保证所传送的分组不丢失，也不保证按源主机发送分组的先后顺序以及在多长的时限内必须将分组交付给目的主机。当需要把分组按发送顺序交付给目的主机时，在目的站还必须把收到的分组缓存一下，等到能够按顺序交付主机时再进行交付。当网络发生拥塞时，网络中的某个结点可根据当时的情况将一些分组丢弃（请注意，网络并不是随意丢弃分组）。所以，数据报提供的服务是不可靠的，它不能保证服务质量。实际上**“尽最大努力交付”**的服务就是**没有质量保证的服务**。图 5-2(a)表示主机 H_1 向 H_5 发送的分组，可以看出，有的分组可经过结点 A-B-E，而另一些则可能经过结点 A-C-E，或 A-C-B-E。在

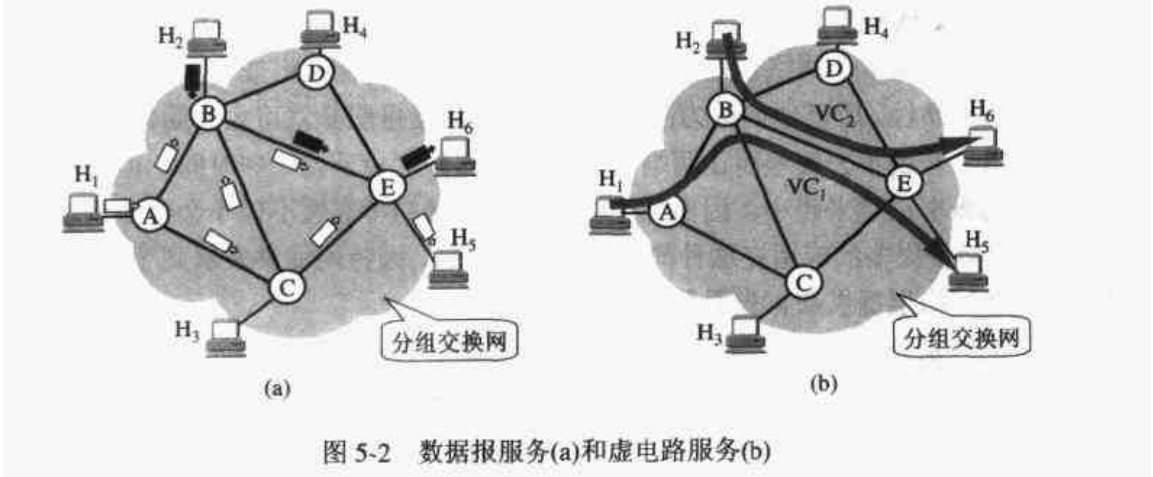


图 5-2　数据报服务(a)和虚电路服务(b)

一个网络中可以有多个主机同时发送数据报，例如主机 H_2 经过结点 B-E 与主机 H_6 通信。

再看网络提供虚电路服务的情况。先设图 5-2(b)中主机 H_1 要和主机 H_5 通信。于是，主机 H_1 先向主机 H_5 发出一个特定格式的控制信息分组，要求进行通信，同时也寻找一条合适的路由。若主机 H_5 同意通信就发回响应，然后双方就建立了虚电路并可传送数据了。这点很像电话通信，先拨号建立主叫和被叫双方之间的通路，然后再通话。

在图 5-2(b)中，我们设寻找到的路由是 A→B→E。这就是我们要建立的**虚电路**(Virtual Circuit)：H_1→A→B→E→H_5(将它记为 VC_1)。以后主机 H_1 向主机 H_5 传送的所有分组都必须沿着这条虚电路传送。在数据传送完毕后，还要将这条虚电路释放掉。

需要注意的是，由于采用了存储转发技术，所以这种虚电路就和电路交换的连接有很大的不同。在电路交换的电话网上打电话时，两个用户在通话期间自始至终地占用一条端到端的物理信道。但当我们占用一条虚电路进行主机通信时，由于采用的是存储转发的分组交换，所以只是断续地占用一段又一段的链路，虽然我们感觉到好像（但并没有真正地）占用了一条端到端的物理电路。建立虚电路的好处是可以在数据传送路径上的各交换结点预先保留一定数量的资源（如带宽、缓存），作为对分组的存储转发之用。

假定还有主机 H_2 和主机 H_6 通信。所建立的虚电路为经过 B→E 两个结点的 VC_2。

在虚电路建立后，网络向用户提供的服务就好像在两个主机之间建立了一对穿过网络的数字管道（收发各用一条）。所有发送的分组都按发送的前后顺序进入管道，然后按照先进先出的原则沿着此管道传送到目的站主机。因为是全双工通信，所以每一条管道只沿着一个方向传送分组。这样，到达目的站的分组顺序就与发送时的顺序一致，因此网络提供虚电路服务对通信的**服务质量** QoS (Quality of Service)有较好的保证。

网络所提供的上述这两种服务的思路来源不同。

虚电路服务的思路来源于传统的电信网。电信网将其用户终端（电话机）做得非常简单，而**电信网负责保证可靠通信的一切措施**，因此**电信网的结点交换机复杂而昂贵**。

数据报服务使用另一种完全不同的新思路。它力求使**网络生存性好和使对网络的控制功能分散，因而只能要求网络提供尽最大努力的服务**。但这种网络要求使用较复杂且有相当智能的主机作为用户终端。**可靠通信由用户终端中的软件（即 TCP）来保证**。

从 20 世纪 70 年代起，关于网络层究竟应当采用数据报服务还是虚电路服务，在网络界一直在进行争论。问题的焦点就是**网络要不要提供网络端到端的可靠通信？**OSI 一开始就按照电信网的思路来对待网络，坚持“网络提供的服务必须是非常可靠的”这样一种观点，因此 OSI 在网络层（以及其他的各个层次）采用了虚电路服务。

然而美国 ARPANET 的一些专家则认为，根据多年的实践证明，不管用什么方法设计网络，网络（这可能由多个网络互连而成）提供的服务并不可能做得非常可靠，用户主机仍要负责端到端的可靠性。所以他们认为：**让网络只提供数据报服务就可大大简化网络层的结构**。当然，网络出了差错不去处理而让两端的主机来处理肯定会延误一些时间，但技术的进步使得网络出错的概率已越来越小，因而让主机负责端到端的可靠性不但不会给主机增加更多的负担，反而能够使更多的应用在这种简单的网络上运行。因特网能够发展到今天这样的规模，充分说明了在网络层提供数据报服务是非常成功的。

除以上的区别外，数据报服务和虚电路服务还都各有一些优缺点。

根据统计，网络上传送的报文长度，在很多情况下都很短。若采用 128 个字节为分组长度，则往往一次传送一个分组就够了。这样，用数据报既迅速又经济。若用虚电路，为了传

送一个分组而建立虚电路和释放虚电路就显得太浪费网络资源了。

为了在交换结点进行存储转发，在使用数据报时，每个分组必须携带完整的地址信息。但在使用虚电路的情况下，每个分组不需要携带完整的目的地址，而仅需要有个很简单的虚电路号码的标志，这就使分组的控制信息部分的比特数减少，因而减少了额外开销。

对待差错处理和流量控制，这两种服务也是有差别的。在使用数据报时，主机承担端到端的差错控制和流量控制。在使用虚电路时，分组按顺序交付，网络可以负责差错控制和流量控制。

数据报服务对军事通信有其特殊的意义。这是因为每个分组可独立地选择路由。当某个结点发生故障时，后续的分组就可另选路由，因而提高了可靠性。但在使用虚电路时，结点发生故障就必须重新建立另一条虚电路。数据报服务还很适合于将一个分组发送到多个地址（即广播或多播）。这一点正是当初 ARPANET 选择数据报的主要理由之一。

表 5-1 归纳了虚电路服务与数据报服务的主要区别。

表 5-1　虚电路服务与数据报服务的对比

对比的方面	虚电路服务	数据报服务
思路	可靠通信应当由网络来保证	可靠通信应当由用户主机来保证
连接的建立	必须有	不要
目的站地址	仅在连接建立阶段使用，每个分组使用短的虚电路号	每个分组都有目的站的全地址
分组的转发	属于同一条虚电路的分组均按照同一路由进行转发	每个分组独立选择路由进行转发
当结点出故障时	所有通过出故障的结点的虚电路均不能工作	出故障的结点可能会丢失分组，一些路由可能会发生变化
分组的顺序	总是按发送顺序到达目的站	到达目的站时不一定按发送顺序
端到端的差错处理和流量控制	可以由分组交换网负责，也可以由用户机负责	由用户主机负责

5.2　广域网中的分组转发机制

我们知道分组交换网的分组转发是基于查表的，本节就是讨论这种查表的机制。这一节所讨论的分组转发机制，也是下一章学习互联网的路由选择的基础[COME01]。

在讨论转发机制之前要先说明一下，“**转发**”(forwarding)和“**路由选择**”(routing)这两个名词的使用在过去有些混乱。现在的文献倾向于将它们区分开来[PETE00]。

转发就是当交换结点收到分组后，根据其目的地址查找**转发表**(forwarding table)，并找出应从结点的哪一个接口将该分组发送出去。

路由选择则是构造**路由表**(routing table)[①]的过程。

路由表是根据一定的路由选择算法得到的，而**转发表**又是根据路由表构造出的。

总之，**路由选择协议负责搜索分组从某个结点到目的结点的最佳传输路由，以便构造路**

① 注：名词 route 和 routing 的标准译名是“路由”和“路由选择”[MINGCI94]。但 routing table 的标准译名是“路由表”而不是“路由选择表”，这可能是为了使译名更简洁些。

由表。从路由表再构造出转发分组的转发表。**分组是通过转发表进行转发的。**

需要注意的是，当讨论一些原理时，为了使讨论更简单些，在许多文献中没有严格区分“转发”和“路由选择”，也不一定使用“转发表”这一名词（例如，在转发分组时不是说“查找转发表”而是说“查找路由表”）。这样做并不影响对问题实质的理解。

5.2.1 在结点交换机中查找转发表

在讨论转发表之前，应先知道广域网是怎样给接入到网络的每一台主机进行编址的。这个问题很重要，因为没有地址的主机是无法在网络中进行通信的。

1. 层次结构的地址结构

第 4 章讨论的局域网采用了**平面地址结构**(flat addressing)。对不需要进行路由选择的局域网，这种结构非常方便。然而在广域网中，分组往往要经过许多的结点交换机的存储转发才到达目的地。在广域网中每一个结点交换机中都有一个转发表，里面存放了到达每一个主机的路由。显然，广域网中的主机数越多，查找转发表就越费时间。为了减少查找转发表所花费的时间，在广域网中一般都采用**层次地址结构**(hierarchical addressing)。

最简单的层次结构地址就是把一个用二进制数表示的主机地址划分为前后两部分。前一部分的二进制数表示该主机所连接的分**组交换机的编号**，而后一部分的二进制数表示所连接的分组交换机的**端口号**，或**主机的编号**（图 5-3）。网络中的结点交换机分为两种：

（1）仅和其他结点交换机相连接；

（2）除了和其他结点交换机相连接，还要和用户主机相连接。

这两种交换机的主要区别是：第一种交换机的连接端口都是高速端口（因为交换机之间的线路速率较高），而第二种交换机还要有一些和主机连接的低速端口。

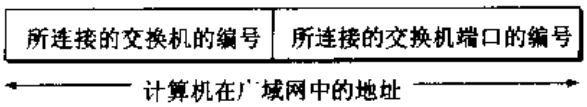


图 5-3 最简单的层次地址

例如在图 5-4 中有三个交换机，其编号分别为 1, 2 和 3。每个交换机所连接的主机也按接入的低速端口编上号码。这样，交换机 1 所接入的两个主机的地址就分别记为[1, 1]和[1, 3]（标注在图中的主机旁边）。如果我们用 8 个比特来表示主机地址，而交换机的编号和低速端口的编号都各用 4 个比特，那么上述的两个主机的地址就分别是 00010001 和 00010011。不难看出，采用这种编址方法，在整个广域网中的每一台主机的地址一定是惟一的。实际上，在许多情况下，用户和应用程序不必知道这个地址是分层结构的，而可以将这样的地址简单地看成是一个数。

下面我们先看一下结点交换机是怎样转发分组的。

为简单起见，图 5-4 中只画出了结点交换机 2 中的转发表，并只给出了转发表中最重要的两个内容，即一个分组将要发往的目的站，以及分组发往的**下一跳**(next hop)[①]。

① 注：术语 hop 的标准译名是“跳”。一个分组在网络中经过一个个的结点交换机存储转发，最后到达目的地，也就是“一跳”接“一跳”地向前转发。“下一跳”也可称为“下一站”，但“下一站”不够准确，因为“站”一般是指连接在网络上的主机（即工作站或站）而不是指网络中的结点交换机，而“跳”则带有转发的含义。有时在转发表上没有写上“下一跳”而是写上“转发端口”。这实质上是一样的。因为“转发端口”和“下一跳”是一一对应的。

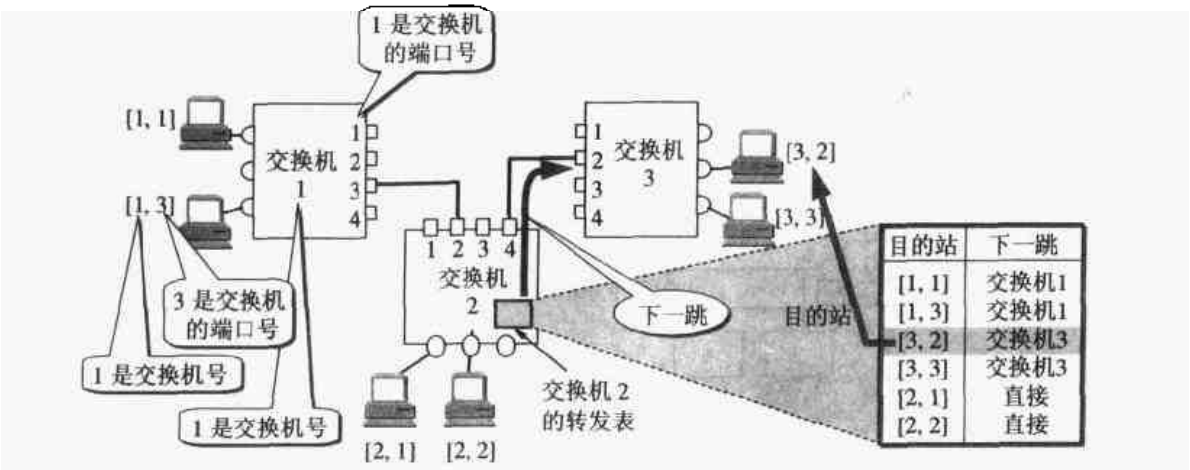


图 5-4 主机在广域网中的地址和交换机中的转发表

例如，图 5-4 中有一个欲发往主机[3, 2]的分组到达了交换机 2。在转发表的第 3 行找出下一跳应为“交换机 3”。于是按照转发表的这个指示将该分组转发到交换机 3。如果分组的目的地是直接连接在本交换机上的主机，则不需要再将分组转发到别的交换机，这时转发表上注明的就是：“直接”。例如，有一个欲发往主机[2, 1]的分组到达了交换机 2。查找转发表后在第 5 行找出，其下一跳应为“直接”，表明该分组已经到达了最后一个交换机，而目的主机就连接在这个交换机上。

读者应当注意，**转发表中没有源站地址这一项**。这是因为**在分组转发中的下一跳只取决于数据报中的目的站地址，而与源站地址无关**。这是一个很重要的概念，应记住。

2. 按照目的站的交换机号确定下一跳

仔细再看一下图 5-4 就可发现，这种转发表还可进行简化。这是因为只要转发表中目的站的交换机号相同，那么查出的“下一跳”就是相同的。因此在确定下一跳时，我们可以不必根据目的站的完整地址，而是可以仅仅根据目的站地址中的交换机号。所以，若将转发表中的“目的站”定义为“目的主机地址中的交换机号”（即不管主机的编号是多少），那么转发表就可进一步简化。例如，图 5-4 交换机 2 中的转发表可压缩为 3 行，即将交换机号相同的行合并。

若每个交换机连接 10 台主机，则简化后的转发表就只有原来的十分之一大小。

采用两个层次的编址方案可使转发分组时只根据分组的第一部分地址（交换机号），即**在进行分组转发时，只根据**收到的分组的主机地址**中的交换机号**。只有当分组到达与目的主机相连的结点交换机时，交换机才检查第二部分地址（主机号），并通过合适的低速端口将分组交给目的主机。在互联网环境下也是采用这种分层次转发分组的原理。

5.2.2 在路由表中使用默认路由

从上面的讨论可知，所谓广域网的路由问题就是要解决分组在各交换机中应如何进行转发。前面所提到的转发表就是为了解决广域网的路由问题而在交换机中专门设置的。因此，在专门研究广域网的路由问题时，可用图论中的“图(graph)”来表示整个广域网，用“结点”表示广域网上的结点交换机，用连接结点与结点的“边”表示广域网中的链路。至于连接在结点交换机上的主机由于与分组转发无关（因为现在我们是根据主机所连接到的交换机号进

行分组的转发），因此在图中一律不画上主机而只剩下各结点交换机。这样得出的较简明的图对于讨论分组转发是非常清晰的。

图 5-5 左边是一个具有 4 个结点交换机的例子，而右边则是对应的图。图中结点表示交换机，圆圈中的数字就是结点交换机号，连接两结点的边表示连接交换机的链路。

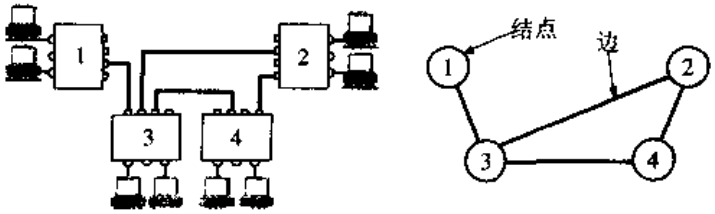


图 5-5 用图表示一个广域网

根据图 5-5 所示的图，可得出每一个结点中的转发表（图 5-6）。在“下一跳”下面的“直接”，表示只通过“本交换机”直接发往所连接的主机，而不要再转发到其他结点。

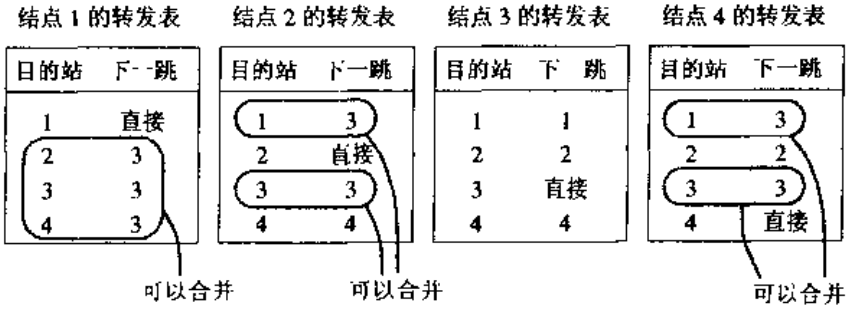


图 5-6 图 5-5 中每一个结点的转发表

分析图 5-6 所示的转发表，就可发现还可再将转发表进一步简化。例如，在结点 1 的转发表中，当目的站为 2，3 或 4 时，分组都是转发到结点 3，因而“下一跳”这一列中的“3”是重复出现的。为什么会出现这种情况呢？只要看一下图 5-5 就知道了。结点 1 只有一条链路连接到结点 3。从结点 1 发往其他任何结点的分组都只能先转发到结点 3。

在较小的网络中，转发表中重复的项目不多。但很大的广域网的转发表中就有可能出现很多的重复项目。这会导致搜索转发表时花费较长的时间。为了减少转发表中的重复项目，可以用一个**默认路由**(default route)代替所有的具有相同“下一跳”的项目。默认路由比其他项目的优先级低。若转发分组时找不到明确的项目对应，才使用默认路由。图 5-7 是使用了默认路由的简化转发表，其中有下划线的是默认路由。

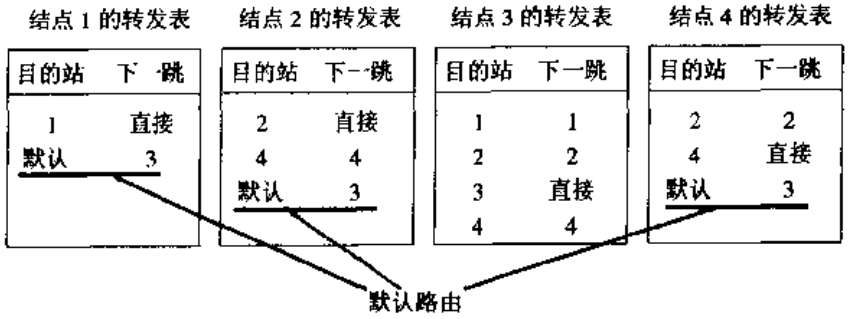


图 5-7 使用了默认路由的简化转发表

从图 5-7 可看出，只有超过一个以上的目的站有相同的下一跳时，使用默认路由才会使转发表更加简洁，才能减少查找转发表的时间。

到目前为止，我们已经讨论了怎样查找转发表，找到下一跳。然而我们还没有讨论转发表中的各项目是怎样写入的。像图 5-5 所示的简单网络，我们稍加观察就能写出所有结点的转发表。但对于大型广域网（如有上百个或更多的结点）情况就不同了。在这种情况下就必须使用合适的路由算法。所谓“**路由算法**”就是**用于产生路由表的算法**。有关路由算法的问题我们将在下一章进行讨论。

5.3 拥塞控制

拥塞控制是广域网和互连网中的一个很重要的问题。本节从一般意义上介绍拥塞控制的意义和拥塞控制的基本原理。

5.3.1 拥塞控制的意义

在计算机网络中的链路容量（即带宽）、交换结点中的缓存和处理机等，都是网络的资源。在某段时间，若对网络中某一资源的需求超过了该资源所能提供的可用部分，网络的性能就要变坏。这种情况就叫做拥塞(congestion)。可以将出现资源拥塞的条件写成如下的关系式：

$$\sum \text{对资源的需求} > \text{可用资源} \qquad (5\text{-}1)$$

若网络中有许多资源同时产生拥塞，网络的性能就要明显变坏，整个网络的吞吐量将随输入负荷的增大而下降。

有人可能会说：“只要增加一些资源，例如，将结点缓存的存储空间扩大，或将链路更换为更高速率的链路，或将结点处理机的运算速度提高，就可以解决网络拥塞的问题。”其实不然。这是因为网络拥塞是一个非常复杂的问题。简单地采用上述做法，在许多情况下，不但不能解决拥塞问题，而且还可能使网络的性能更坏。

网络拥塞往往是由许多因素引起的。例如，当某个结点缓存的容量太小时，到达该结点的分组因无存储空间暂存而不得不被丢弃。现在设想将该结点缓存的容量扩展到非常大。于是凡到达该结点的分组均可在这缓存的队列中排队，不受任何限制。由于输出链路的容量和处理机的速度并未提高，因此在这队列中的绝大多数分组的排队等待时间将会很长很长，结果上层软件只好将它们进行重传（因为早就超时了）。由此可见，简单地扩大缓存的存储空间同样会造成网络资源的严重浪费，因而解决不了网络拥塞的问题。

又如，处理机处理的速率太慢可能引起网络的拥塞。简单地将处理机的速率提高，可能会使上述情况缓解一些，但往往又会将瓶颈转移到其他地方。问题的实质往往是整个系统的各个部分不匹配。只有所有的部分都平衡了，问题才会得到解决。

拥塞常常使问题趋于恶化。如果一个路由器没有足够的缓存空间，它就会丢弃一些新到的分组。但当分组被丢弃时，发送这一分组的相邻路由器就会重传这一分组，甚至可能还要重传多次。发送端在未收到确认之前必须保留所发分组的副本以便进行可能的重传。可见在接收端产生的拥塞反过来会引起发送端缓存的拥塞。

拥塞控制与流量控制的关系密切，它们之间也存在着一些差别。拥塞控制所要做的都有一个前提，就是**网络能够承受现有的网络负荷**。拥塞控制是一个**全局性的过程**，涉及到所有

的主机、所有的路由器，以及与降低网络传输性能有关的所有因素。

相反，**流量控制往往指在给定的发送端和接收端之间的点对点通信量的控制**。流量控制所要做的就是抑制发送端发送数据的速率，以便使接收端来得及接收。流量控制几乎总是存在着从接收端到发送端的某种直接反馈，使发送端知道接收端是处于怎样的状况。

可以用一个简单例子说明这种区别。设有一个光纤网络，其链路传输速率为 1000 Gb/s。有一个巨型计算机向一个 PC 机以 1 Gb/s 的速率传送一个文件。显然，网络本身的带宽是足够大的，因而不存在产生拥塞的问题。但流量控制却是必须的，因为巨型计算机必须经常停下来，以便使 PC 机来得及接收。

但如果有另一个网络，其链路传输速率为 1 Mb/s，而有 1000 台大型计算机连接在这个网络上。假定其中的 500 台计算机分别向其余的 500 台计算机以 100 kb/s 的速率发送文件。那么现在的问题已不是接收端的大型计算机是否来得及接收，而是整个网络的输入负载是否超过网络所能承受的。

拥塞控制和流量控制之所以常常被弄混，是因为某些拥塞控制算法是向发送端发送控制报文，并告诉发送端，网络已出现麻烦，必须放慢发送速率。这点又和流量控制是很相似的。

进行拥塞控制需要付出代价。这首先需要获得网络内部流量分布的信息。在实施拥塞控制时，还需要在结点之间交换信息和各种命令以便选择控制的策略和实施控制。这样就产生了额外开销。拥塞控制有时需要将一些资源（如缓存、带宽等）分配给个别用户（或一些类别的用户）单独使用，这样就使得网络资源不能更好地实现共享。十分明显，在设计拥塞控制策略时，必须全面衡量得失。

在图 5-8 中的横坐标是**提供的负载**(offered load)，代表单位时间内输入给网络的分组数目。因此提供的负载也称为**输入负载或网络负载**。纵坐标是**吞吐量**(throughput)，代表单位时间内从网络输出的分组数目。具有理想拥塞控制的网络，在吞吐量饱和之前，网络吞吐量应等于提供的负载，故吞吐量曲线是 45°的斜线。但当提供的负载超过某一限度时，由于网络资源受限，吞吐量不再增长而保持为水平线，即吞吐量达到饱和。这就表明提供的负载中有一部分损失掉了（例如，输入到网络的某些分组被某个结点丢弃了）。虽然如此，在这种理想的拥塞控制作用下，网络的吞吐量仍然维持在其所能达到的最大值。

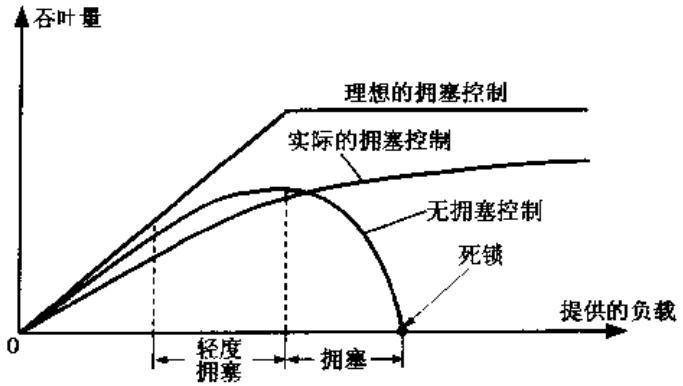


图 5-8　拥塞控制所起的作用

但是，实际网络的情况就很不相同了。从图 5-8 可看出，随着提供的负载的增大，网络吞吐量的增长速率逐渐减小。也就是说，在网络吞吐量还未达到饱和时，就已经有一部分的

输入分组被丢弃了。当网络的吞吐量明显地小于理想的吞吐量时，网络就进入了**轻度拥塞**的状态。更值得注意的是，当提供的负载达到某一数值时，网络的吞吐量反而随提供的负载的增大而下降，这时**网络就进入了拥塞状态**。当提供的负载继续增大到某一数值时，网络的吞吐量就下降到零，网络已无法工作。这就是所谓的死锁(deadlock)。

死锁中有一种是**直接死锁**，即由互相占用了对方需要的资源而造成的死锁。例如两个结点 A 和 B 都有大量的分组要发往对方，但两个结点中的缓存在发送之前就已经全部被待发分组占满了。这样，当每个分组到达对方时，由于没有地方存放，只好被丢弃。发送分组的一方因收不到对方发来的确认信息，只能将发送过的分组依然保存在自己结点的缓存中。这两个结点就这样一直互相僵持着，谁也无法成功地发送出一个分组。

还有一种死锁是由于路由器的缓存的拥塞而引起的**重装死锁**(reassembly deadlock)。图 5-9 为重装死锁的例子。设三个报文 A，B 和 C 经过路由器 P，Q 和 R 发往主机 H。每一个报文由四个分组构成。又设每个路由器的缓存只能容纳 4 个分组。从图 5-9 可看出，路由器 R 已为报文 A 预留了 4 个分组的缓存。由于分组 A_3 还未到达，所以目前还不能交付给主机 H。目前分组 A_3 暂存于路由器 P 的缓存中，它无法转发到路由器 Q，因为路由器 Q 的缓存已全占满了。路由器 Q 的缓存中的任何一分组都不能向前转发，因为路由器 R 的缓存全是给报文 A 预留的。

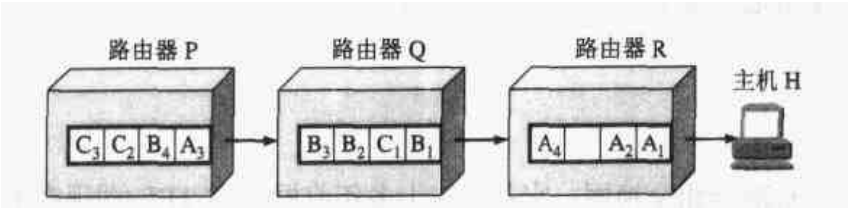


图 5-9　重装死锁举例

加上合适的拥塞控制后，网络就不易出现拥塞现象和死锁。图 5-8 表明，我们付出的代价就是：当提供的负载较小时，有拥塞控制的吞吐量反而比无拥塞控制时要小。

在分组交换网中，网络性能恶化有时是由于网络的资源白白地被浪费了。最常被浪费的网络资源是通信信道带宽和结点的存储空间。

5.3.2　拥塞控制的一般原理

从原理上讲，寻找拥塞控制的方案无非是寻找使不等式(5-1)不再成立的条件。这或者是增大网络的某些可用资源（如业务繁忙时增加一些链路，增大链路的带宽，或使额外的通信量从另外的通路分流），或减少一些用户对某些资源的需求（如拒绝接受新的建立连接的请求，或要求用户减轻其负荷，这属于降低服务质量）。但正如上面所讲过的，在采用某种措施时，还必须考虑到该措施所带来的其他影响。

实践证明，拥塞控制是很难设计的，因为它是一个**动态的**（而不是静态的）问题。当前网络正朝着高速化的方向发展，这很容易出现缓存不够大而造成分组的丢失。但分组的丢失是网络发生拥塞的征兆而不是原因。在许多情况下，甚至正是拥塞控制本身成为引起网络性能恶化甚至发生死锁的原因。**这点应特别引起重视**。

由于计算机网络是一个很复杂的系统，因此可以从控制理论的角度来看拥塞控制这个问题。这样，从大的方面看，可以分为**开环控制**和**闭环控制**两种方法。开环控制方法就是在设计网络时事先将有关发生拥塞的因素考虑周到，力求网络在工作时不产生拥塞。但一旦整个

系统运行起来，就不再中途进行改正了。

闭环控制是基于反馈环路的概念。属于闭环控制的有以下几种措施：

（1）监测网络系统以便检测到拥塞在何时、何处发生。

（2）将拥塞发生的信息传送到可采取行动的地方。

（3）调整网络系统的运行以解决出现的问题。

有很多的方法可用来监测网络的拥塞。主要的一些指标是：由于缺少缓存空间而被丢弃的分组的百分数；平均队列长度；超时重传的分组数；平均分组时延；分组时延的标准差等等。上述这些指标的上升都标志着拥塞的增长。

一般在监测到拥塞发生时，要将拥塞发生的信息传送到产生分组的源站。当然，通知拥塞发生的分组同样会使网络更加拥塞。

另一种方法是在路由器转发的分组中保留一个比特或字段，用该比特或字段的值表示网络没有拥塞或产生了拥塞。也可以由一些主机或路由器周期性地发出分组，以询问拥塞是否发生。

此外，过于频繁地采取行动以缓和网络的拥塞会使系统产生不稳定的振荡。但过于迟缓地采取行动又不具有任何实用价值。因此，要采用某种折中的方法。但选择正确的时间常数是相当困难的。

一些更加具体的防止网络拥塞的方法将在 7.4.4 节介绍。

5.4 X.25 网

X.25 网就是 X.25 分组交换网，它是在二十多年前根据 CCITT (即现在的 ITU-T)的 X.25 建议书实现的计算机网络。X.25 网在推动分组交换网的发展中曾做出了很大的贡献。但是，现在已经有了性能更好的网络来代替它，如帧中继网或 ATM 网。

X.25 只是一个对公用分组交换网接口的规约。X.25 所讨论的都是**以面向连接的虚电路服务为基础**。这一概念如图 5-10 所示。图中画的是一个数据终端设备 DTE 同时和另外两个 DTE 进行通信的情况。网络中的虚线代表两条虚电路。图中还画出了三个 DTE 与数据电路端接设备 DCE 的接口，X.25 所规定的正是关于这一接口的标准。



图 5-10　X.25 规定了 DTE-DCE 的接口

DTE 与 DCE 的接口实际上也就是 DTE 和公用分组交换网的接口。由于 DCE 通常是用户设施，因此可将 DCE 画在网络外面（图 5-10）。

图 5-11 表示 X.25 接口的三个层次。最下面是物理层，接口标准是 X.21 建议书。第二层是数据链路层，接口标准是平衡型链路接入规程 LAPB，它就是第 3 章介绍的 HDLC 的一个子集。第三层是分组层（不叫网络层），在这一层上，在 DTE 与 DCE 之间可建立多条逻辑信

道(0～4095 号)。这样可以使一个 DTE 同时和网上其他多个 DTE 建立虚电路并进行通信。从第一层到第三层，数据传送的单位分别是“比特”、“帧”和“分组”。X.25 还规定了在经常需要进行通信的两个 DTE 之间可以建立**永久虚电路**。

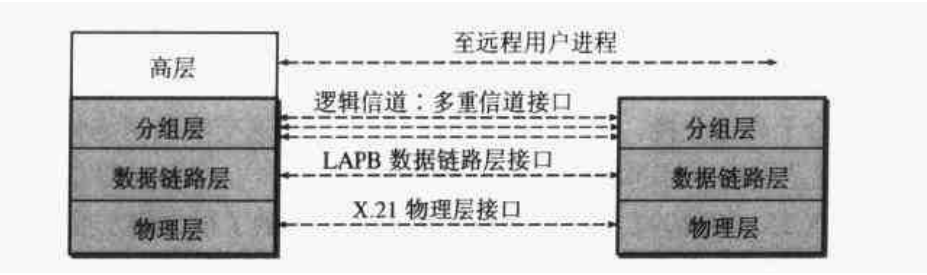


图 5-11　X.25 的层次关系

从以上的简单介绍就可看出，X.25 分组交换网和以 IP 协议为基础的因特网在设计思想上有着根本的差别。因特网是无连接的，只提供尽最大努力交付的数据报服务，无服务质量可言。而 X.25 网是面向连接的，能够提供可靠交付的虚电路服务，能保证服务质量。正因为 X.25 网能保证服务质量，在二十多年前它曾经是颇受欢迎的一种计算机网络。

大家知道，在二十多年前，计算机的价格很贵，许多用户只用得起廉价的哑终端（连硬盘都没有）。当时通信线路的传输质量一般都较差，误码率较高。X.25 网的设计思路是将智能做在网络内。X.25 网在每两个结点之间的传输都使用带有编号和确认机制的 HDLC 协议，而网络层使用具有流量控制的虚电路机制，可以向用户的哑终端提供可靠交付的服务。但是到了 20 世纪 90 年代，情况就发生了很大的变化。通信主干线路已大量使用光纤技术，数据传输质量大大提高使得误码率降低好几个数量级，而 X.25 十分复杂的数据链路层协议和分组层协议已成为多余的。PC 机的价格急剧下降使得无硬盘的哑终端退出了通信市场。这正好符合因特网当初的设计思想：网络应尽量简单而智能应尽可能放在网络以外的用户端。虽然因特网只提供尽最大努力交付的服务，但具有足够智能的用户 PC 机完全可以实现差错控制和流量控制，因而因特网仍能向用户提供端到端的可靠交付。

这样，到了 20 世纪末，无连接的、提供数据报服务的因特网最终演变成为全世界最大的计算机网络，而 X.25 分组交换网却退出了历史舞台。

值得注意的是，当利用现有的一些 X.25 网来支持因特网的服务时，X.25 网就表现为数据链路层的链路。图 5-12 说明了这一情况。设路由器 B 和 C 之间是 X.25 网络。在 B 和 C 之间建立的 X.25 虚电路就相当于 IP 层下面的数据链路层。在有的计算机网络文献中，常把支持因特网的广域网（包括 X.25 网、帧中继网和 ATM 网）都看成是 IP 层下面的数据链路层。不过在单独讨论广域网的问题时，广域网还是应当属于网络层。本书就是这样处理广域网的。



图 5-12　X.25 虚电路相当于 IP 层下面的数据链路层

5.5 帧中继 FR

5.5.1 帧中继的工作原理

在 20 世纪 80 年代后期，许多应用都迫切要求增加分组交换服务的速率。然而 X.25 网络的体系结构并不适合于高速交换。可见需要研制一种支持高速交换的网络体系结构。**帧中继 FR** (Frame Relay)就是为这一目的而提出的[STAL00]。帧中继在许多方面非常类似于 X.25，它被称为第二代的 X.25。在 1992 年帧中继问世后不久就得到了很大的发展。

在 X.25 网络发展初期，网络传输设施基本是借用了模拟电话线路，这种线路非常容易受到噪声的干扰而产生误码。为了确保传输无差错，X.25 在每个结点都需要作大量的处理。例如，X.25 的数据链路层协议 LAPB 保证了帧在结点间无差错传输。在网络中的每一个结点，只有当收到的帧已进行了正确性检查后，才将它交付给第 3 层协议。对于经历多个网络结点的帧，这种处理帧的方法会导致较长的时延。除了数据链路层的开销，分组层协议为确保在每个逻辑信道上按序正确传送，还要有一些处理开销。在一个典型的 X.25 网络中，分组在传输过程中在每个结点大约有 30 次左右的差错检查或其他处理步骤。

今天的数字光纤网比早期的电话网具有低得多的误码率，因此，我们完全可以简化 X.25 的某些差错控制过程。如果减少结点对每个分组的处理时间，则各分组通过网络的时延亦可减少，同时结点对分组的处理能力也就增大了。

帧中继就是一种减少结点处理时间的技术。帧中继的原理很简单。当帧中继交换机收到一个帧的首部时，**只要一查出帧的目的地址就立即开始转发该帧**。因此在帧中继网络中，一个帧的处理时间比 X.25 网约减少一个数量级。这样，帧中继网络的吞吐量要比 X.25 网络的提高一个数量级以上。

那么若出现差错该如何处理呢？显然，只有当整个帧被收下后该结点才能够检测到比特差错。但是当结点检测出差错时，很可能该帧的大部分已经转发出去了。

解决这一问题的方法实际上非常简单。当检测到有误码时，结点要立即中止这次传输。当中止传输的指示到达下个结点后，下个结点也立即中止该帧的传输，并丢弃该帧。即使上述出错的帧已到达了目的结点，用这种丢弃出错帧的方法也不会引起不可弥补的损失。不管是上述的哪一种情况，源站将用高层协议请求重传该帧。帧中继网络纠正一个比特差错所用的时间当然要比 X.25 网分组交换网稍多一些。因此，**仅当帧中继网络本身的误码率非常低时，帧中继技术才是可行的**。

当正在接收一个帧时就转发此帧，通常被称为**快速分组交换**(fast packet switching)。快速分组交换在实现的技术上有两大类，它是根据网络中传送的帧长是可变的还是固定的来划分。在快速分组交换中，**当帧长为可变时就是帧中继；当帧长为固定时**（这时每一个帧叫做一个信元）就是**信元中继**(Cell Relay)，像异步传递方式 ATM 就属于信元中继。

图 5-13(a)和(b)分别是一般分组交换网络和帧中继这两种方式在层次上的对比。前者的概念已在前面讲过。这里要指出的是，对于一般的分组交换网，其数据链路层具有完全的差错控制。但对于**帧中继网络，不仅其网络中的各结点没有网络层，并且其数据链路层只具有有限的差错控制功能**。只有在通信两端的主机中的数据链路层才具有完全的差错控制功能。图 5-13(b)中带阴影的部分表示**帧中继网络只有最低的两层**。

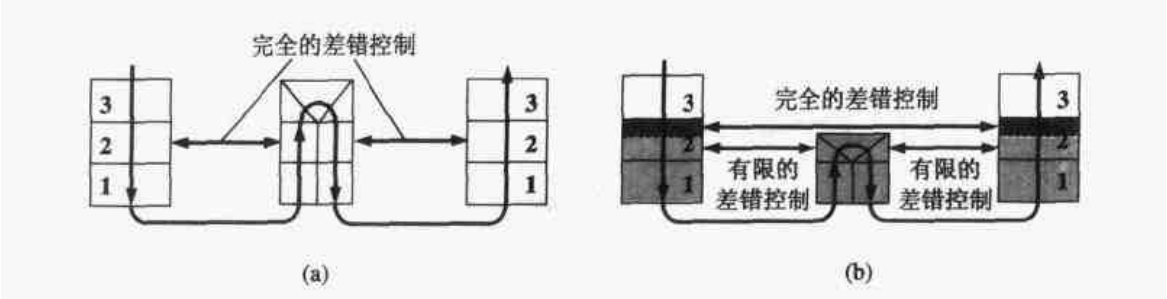


图 5-13　从层次关系上比较一般分组交换网(a)与帧中继方式(b)

图 5-14 比较了两种情况下从源站到目的站传送一帧在网络的各链路上所要传送的信息。若在传送的过程中出现了差错而导致分组的重传，则二者的差别就还要大。图 5-14(a)是一般分组交换网的情况，每一个结点在收到一个数据帧后都要向前一个结点发回确认帧，而目的站最后还要向源站发回确认，这也要逐站进行确认（即对确认帧的确认）。图 5-14(b)是帧中继的情况，它的中间站只转发数据帧而不发送确认帧，即中间站没有逐段的链路控制能力。只有在目的站收到数据帧后才向源站发回端到端的确认。因此帧中继在数据传输的过程中省略掉了很多的确认过程。

帧中继的数据链路层也没有流量控制能力。帧中继的流量控制由高层来完成。

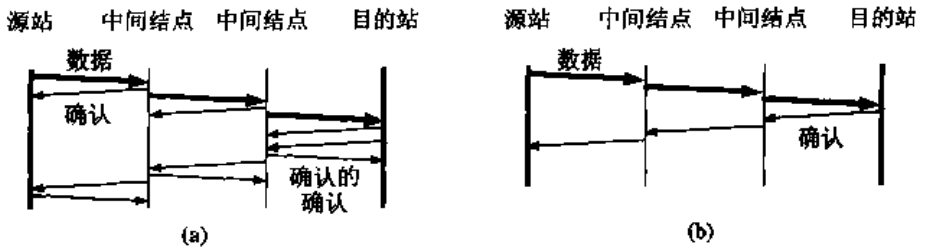


图 5-14　一般分组交换网的存储转发方式(a)与帧中继方式(b)的对比

帧中继的呼叫控制信令是在与用户数据分开的另一个逻辑连接上传送的（即**共路信令或带外信令**，见附录 C）。这点和 X.25 很不相同。X.25 使用带内信令，即呼叫控制分组与用户数据分组都在同一条虚电路上传送。

帧中继的逻辑连接的复用和交换都在第二层处理，而不是像 X.25 在第三层处理。

帧中继网络向上提供面向连接的虚电路服务。虚电路一般分为交换虚电路 SVC 和永久虚电路 PVC 两种，但帧中继网络通常为相隔较远的一些局域网提供链路层的永久虚电路服务。永久虚电路的好处是在通信时可省去建立连接的过程。图 5-15(a)是一个例子。帧中继网络有 4 个帧中继交换机。帧中继网络与局域网相连的交换机相当于 DCE，而与帧中继网络相连的路由器则相当于 DTE。当帧中继网络为其两个用户提供帧中继虚电路服务时，对两端的用户来说，帧中继网络所提供的虚电路就好像在这两个用户之间有一条直通的专用电路（图 5-15(b)）。**用户看不见帧中继网络中的帧中继交换机**。

下面是帧中继网络的工作过程。

当用户在局域网上传送的 MAC 帧传到与帧中继网络相连接的路由器时，该路由器就剥去 MAC 帧的首部，将 IP 数据报交给路由器的网络层。网络层再将 IP 数据报传给帧中继接口卡。帧中继接口卡将 IP 数据报加以封装，加上帧中继帧的首部（其中包括帧中继的虚电路号），进行 CRC 检验和加上帧中继帧的尾部。然后帧中继接口卡将封装好的帧通过向电信公

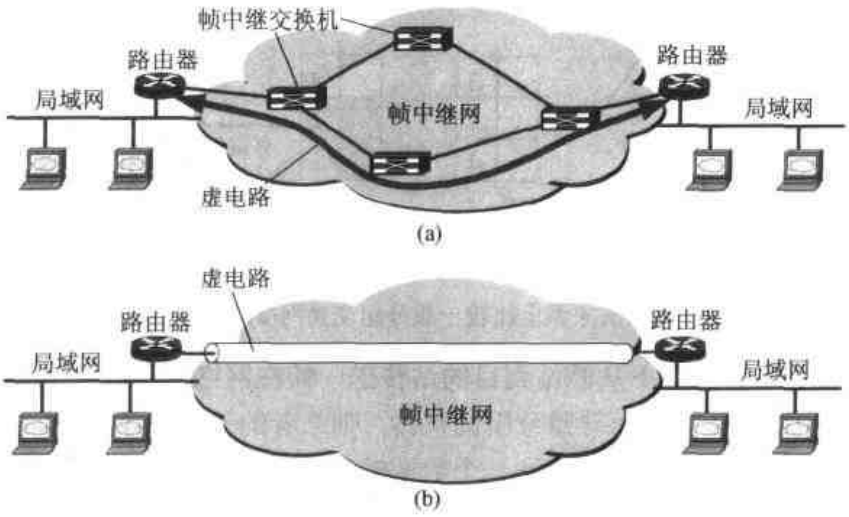


图 5-15　帧中继网络提供的服务：(a)帧中继提供虚电路服务；(b)虚电路像一条专用电路

司租来的专线发送给帧中继网络中的帧中继交换机。帧中继交换机在收到一个帧时，就按虚电路号对帧进行转发（若检查出有差错则丢弃）。当这个帧被转发到虚电路的终点路由器时，该路由器剥去帧中继帧的首部和尾部，加上局域网的首部和尾部，交付给连接在此局域网上的目的主机。目的主机若发现有差错，则报告上层的 TCP 协议处理。

图 5-16 进一步给出了帧中继服务的几个主要组成部分。用户通过帧中继**用户接入电路**(User Access Circuit)连接到帧中继网络，常用的用户接入电路的速率是 64 kb/s 和 2.048 Mb/s(或 T1 速率 1.544 Mb/s)。理论上也可使用 T3 或 E3 的速率。帧中继用户接入电路又称为**用户网络接口** UNI (User-to-Network Interface)。UNI 有两个端口。在用户的一侧叫做**用户接入端口**(User Access Port)，而在帧中继网络一侧的叫做**网络接入端口**(Network Access Port)。用户接入端口就是在**用户屋内设备** CPE (Customer Premises Equipment)中的一个物理端口（例如，一个路由器端口）。一个 UNI 中可以有一条或多条虚电路（永久的或交换的）。图中的 UNI 画有两条永久虚电路：PVC1 和 PVC2。从用户的角度来看，一条永久虚电路 PVC 就是跨接在两个用户接入端口之间。**每一条虚电路都是双向的**，并且每一个方向都有一个指派的 CIR。CIR 就是**承诺的信息速率**(Committed Information Rate)。为了区分开不同的 PVC，每一条 PVC 的两个端点都各有一个**数据链路连接标识符** DLCI (Data Link Connection Identifier)。

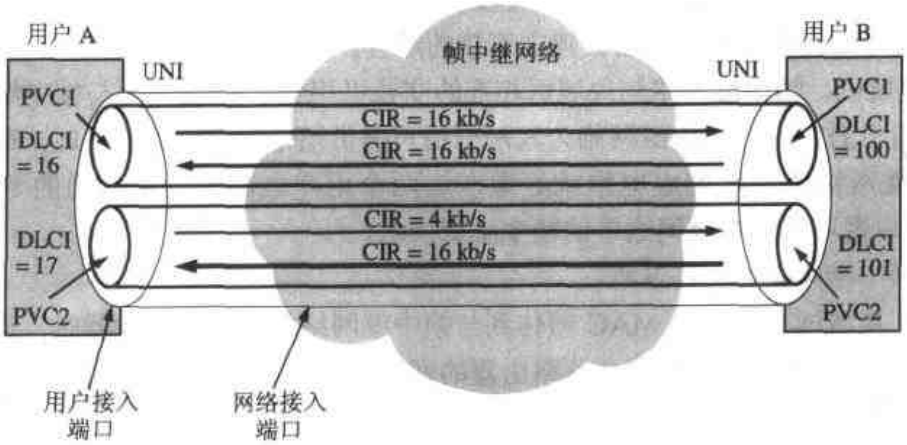


图 5-16　帧中继服务的几个主要组成部分

下面我们归纳一下帧中继的主要优点：

（1）**减少了网络互连的代价**。当使用专用帧中继网络时，将不同的源站产生的通信量复用到专用的主干网上，可以减少在广域网中使用的电路数。多条逻辑连接复用到一条物理连接上可以减少接入代价。

（2）**网络的复杂性减少但性能却提高了**。与 X.25 相比，由于网络结点的处理量减少，由于更加有效地利用高速数据传输线路，帧中继明显改善了网络的性能和响应时间。

（3）由于使用了国际标准，**增加了互操作性**。帧中继的简化的链路协议实现起来不难。接入设备通常只需要一些软件修改或简单的硬件改动就可支持接口标准。现有的分组交换设备和 T1/E1 复用器都可进行升级，以便在现有的主干网上支持帧中继。

（4）**协议的独立性**。帧中继可以很容易地配置成容纳多种不同的网络协议（如 IP, IPX 和 SNA 等）的通信量。可以用帧中继作为公共的主干网，这样可统一所使用的硬件，也更加便于进行网络管理。

根据帧中继的特点，可以知道帧中继适用于大文件（如高分辨率图像）的传送、多个低速率线路的复用，以及局域网的互连。

5.5.2 帧中继的帧格式

图 5-17 画的是帧中继的帧格式。这种格式与 HDLC 帧格式类似，其最主要的区别是没有控制字段。这是因为帧中继的逻辑连接只能携带用户的数据，并且没有帧的序号，也不能进行流量控制和差错控制。

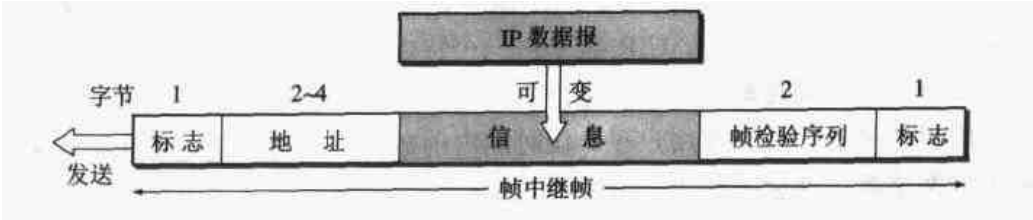


图 5-17 帧中继的帧格式

下面简单介绍其各字段的作用。

（1）**标志** 是一个 01111110 的比特序列，用于指示一个帧的起始和结束。它的惟一性是通过比特填充法来确保的。

（2）**信息** 是长度可变的用户数据。

（3）**帧检验序列** 包括 2 字节的 CRC 检验。当检测出差错时，就将此帧丢弃。

（4）**地址** 一般为 2 字节，但也可扩展为 3 或 4 字节。

地址字段中的几个重要部分是：

- **数据链路连接标识符 DLCI** DLCI 字段的长度一般为 10 bit(采用默认值 2 字节地址字段)，但也可扩展为 16 bit(用 3 字节地址字段)，或 23 bit(用 4 字节地址字段)，这取决于扩展地址字段的值。DLCI 的值用于标识永久虚电路(PVC)、呼叫控制或管理信息。
- **前向显式拥塞通知 FECN** (Forward Explicit Congestion Notification) 若某结点将 FECN 置为 1，表明与该帧在同方向传输的帧可能受网络拥塞的影响而产生时延。
- **反向显式拥塞通知 BECN** (Backward Explicit Congestion Notification) 若某结点将 BECN 置为 1 即指示接受者，与该帧反方向传输的帧可能受网络拥塞的影响产生时延。

• **可丢弃指示 DE** (Discard Eligibility)　在网络发生拥塞时，为了维持网络的服务水平就必须丢弃一些帧。显然，网络应当先丢弃一些相对比较不重要的帧。帧的重要性体现在 DE 比特。DE 比特为 1 的帧表明这是**较为不重要的低优先级帧，在必要时可丢弃**。而 DE = 0 的帧为高优先级帧，希望网络尽可能不要丢弃这类帧。用户采用 DE 比特就可以比通常允许的情况多发送一些帧，并将这些帧的 DE 比特置 1（表明这是较为次要的帧）。

应当注意：**数据链路连接标识符 DLCI 只具有本地意义**。在一个帧中继的连接中，在连接两端的用户网络接口 UNI 上所使用的两个 DLCI 是各自独立选取的。帧中继可同时将多条不同 DLCI 的逻辑信道复用在一条物理信道中。

5.5.3　帧中继的拥塞控制

帧中继的拥塞控制实际上是网络和用户共同负责来实现的。网络（即交换机的集合）能够非常清楚地监视全网的拥塞程度，而用户则在限制通信量方面是最有效的。帧中继使用的拥塞控制方法有以下三种：

（1）**丢弃策略**。当拥塞足够严重时，网络就要被迫将帧丢弃。这是网络对拥塞的最基本的响应。但在具体操作时应当对所有用户都是公平的。

（2）**拥塞避免**。在刚一出现轻微的拥塞迹象时可采取拥塞避免的方法。这时，帧中继网络应当有一些信令机制及时地使拥塞避免过程开始工作。

（3）**拥塞恢复**。在已出现拥塞时，拥塞恢复过程可阻止网络彻底崩溃。当网络由于拥塞开始将帧丢弃时（这时高层软件能够发现这一问题），拥塞恢复过程就应开始工作。

为了进行拥塞控制，帧中继采用了一个概念，叫做**承诺的信息速率 CIR** (Committed Information Rate)，其单位为 b/s。CIR 就是对一个特定的帧中继连接，用户和网络共同协商确定的关于**用户信息传送速率的门限数值**。CIR 数值越高，帧中继用户向帧中继的服务提供者交纳的费用也就越多。只要端用户在一段时间内的数据传输速率超过 CIR，在网络出现拥塞时，帧中继网络就可能会丢弃用户所发送的某些帧。虽然使用了“承诺的”这一名词，但当数据传输速率不超过 CIR 时，网络并不保证一定不发生帧丢弃。当网络拥塞已经非常严重时，网络可以对某个连接只提供比 CIR 还差的服务。当网络必须将一些帧丢弃时，网络将首先选择超过其 CIR 值的那些连接上的帧来丢弃。请注意：CIR 并非数据率的瞬时值。CIR 是端用户根据在某一段**测量时间间隔** T_c 内（这段时间的长短没有国际标准，通常由帧中继网络提供者确定）所发送的数据量所测量出来的**平均数据率**。时间间隔 T_c **越大，通信量超过平均数据率**的波动可能就越大。

每个帧中继结点都应使通过该结点的所有连接的 CIR 的总和不超过该结点的容量，即不能超过该结点的**接入速率**(access rate)。

对于永久虚电路连接，每一个连接的 CIR 应在连接建立时即确定下来。对于交换虚电路连接，CIR 的参数应在呼叫建立阶段协商确定。

当拥塞发生时，应当丢弃什么样的帧呢？这就是检查一个帧的可丢弃指示 DE 字段。若数据的发送速率超过 CIR，则结点交换机就将所收到的帧的 DE 比特都置为 1，并转发此帧。这样的帧，可能会通过网络，但也可能在网络发生拥塞时被丢弃。若结点交换机在收到一个帧时，其数据发送速率已超过网络所设定的最高速率，则立即将其丢弃。

总之，帧中继网络的拥塞控制的原则是：

（1）若数据率小于 CIR，则在该连接上传送的所有帧均被置为 DE = 0（这表明在网络发

生拥塞时尽量不要丢弃 DE = 0 的帧）。这在一般情况下传输是有保证的。

（2）若数据率仅在不太长的时间间隔大于 CIR，则网络可以将这样的帧置为 DE = 1，并在可能的情况下进行传送（即不一定丢弃，视网络的拥塞程度而定）。

（3）若数据率超过 CIR 的时间较长，以致注入到网络的数据量超过了网络所设定的最高门限值，则应立即丢弃该连接上传送的帧。

下面用简单数字说明 CIR 的意义。设某个结点的接入速率为 64 kb/s。该结点使用的一条虚电路被指派的 CIR = 32 kb/s，而 CIR 的测量时间间隔 T_c = 500 ms。再假定帧中继网络的帧长 L = 4000 bit。这就表示在 500 ms 的时间间隔内，这条虚电路只能够发送 $CIR \times T_c / L = 4$ 个高优先级的帧中继帧，其 DE = 0。这就是说，这 4 个高优先级帧在网络中的传输是有保证的，但由于 CIR 的数值只是接入速率的一半，因此用户在 500 ms 内还可再发送 4 个低优先级的帧，其 DE = 1。

帧中继还可利用显式信令避免拥塞。上面讲过，在帧中继的地址字段中有两个指示拥塞的比特，即前向显式拥塞通知 FECN 和反向显式拥塞通知 BECN。我们设帧中继网络的两个用户 A 和 B 之间已经建立了一条双向通信的连接。当两个方向都没有拥塞时，则在两个方向传送的帧中，FECN 和 BECN 都应为零。反之，若这两个方向都发生了拥塞，则不管是哪一个方向，FECN 和 BECN 都应置为 1。当只有一个方向发生拥塞而另一个方向无拥塞时，FECN 和 BECN 中的哪一个应置为 1，则取决于帧是从 A 传送的 B 还是从 B 传送到 A。

网络可以根据结点中待转发的帧队列的平均长度是否超过门限值来确定是否发生了拥塞。

用户也可以根据收到的显式拥塞通知信令采用相应的措施。当收到 BECN 信令时的处理方法比较简单。用户只要降低数据发送的速率即可。但当用户收到一个 FECN 信令时，情况就较复杂，因为这需要用户通知这个连接的对等用户来减少帧的流量。帧中继协议所使用的核心功能并不支持这样的通知，因此需要在高层来进行相应的处理。

5.6 异步传递方式 ATM

5.6.1 ATM 的基本概念

现有的电路交换和分组交换在实现宽带高速的交换任务时，都表现有一些缺点。

对于电路交换，当数据的传输速率及其突发性变化非常之大时，交换的控制就变得十分复杂。对于分组交换，当数据传输速率很高时，协议数据单元在各层的处理成为很大的开销，无法满足实时性很强的业务的时延要求。特别是，基于 IP 的分组交换网不能保证服务质量。

但电路交换的实时性和服务质量都很好而分组交换的灵活性很好，因此，人们曾经设想过“未来最理想的”一种网络应当是宽带综合业务数字网 B-ISDN（见附录 C），它采用另一种新的交换技术，这种技术结合了电路交换和分组交换的优点。虽然在今天看来 B-ISDN 并没有成功，但 ATM 技术还是获得了相当广泛的应用，并在因特网的发展中起到了重要的作用。

异步传递方式 ATM (Asynchronous Transfer Mode)[①]就是建立在电路交换和分组交换的基

①注：请不要和银行使用的自动取款机 ATM (Automatic Teller Machine)弄混。这种机器与本节讨论的 ATM 无关。

础上的一种面向连接的快速分组交换技术，它采用**定长分组**作为传输和交换的单位。在 ATM 中这种定长分组叫做**信元**(cell)。

首先我们应当弄清 ATM 名词中“**异步**”的含义。

在第 2 章的 2.6 节中我们已经讲过，目前在高速数字通信系统中，在物理层是使用**同步传输**的 SDH。这里的“同步”是指网络中各链路上的比特流都是受同一个非常精确的主时钟的控制。

我们知道，SDH 传送的同步比特流被划分为一个个固定时间长度的帧（请注意，这是时分复用的时间帧，而不是数据链路层的帧）。当用户的 ATM 信元需要传送时，就可插入到 SDH 的一个帧中，但每一个用户发送的信元在每一帧中的相对位置并不是固定不变的。如果用户有很多信元要发送，就可以接连不断地发送出去。只要 SDH 的帧有空位置就可以将这些信元插入进来。因此，这和第 2 章的 2.5.1 节介绍的异步时分复用是一样的原理。也就是说，ATM 名词中的“异步”是指将 ATM 信元“**异步插入**”到同步的 SDH 比特流中。

如果是使用同步插入（即同步时分复用），则用户在每一帧中所占据的时隙的相对位置是固定不变的，即用户只能周期性地占用每一个帧中分配给自己的固定时隙（一个时隙可以是一个或多个字节），而不能再使用其他的已分配给别人的空闲时隙。

ATM 的主要优点如下：

（1）**选择固定长度的短信元作为信息传输的单位，有利于宽带高速交换**。信元长度为 53 字节，其首部（可简称为**信头**）为 5 字节。长度固定的首部可使 ATM 交换机的功能尽量简化，只用硬件电路就可对信元进行处理，因而缩短了每一个信元的处理时间。在传输实时话音或视频业务时，短的信元有利于减小时延，也节约了结点交换机为存储信元所需的存储空间。

（2）**能支持不同速率的各种业务**。ATM 允许终端有足够多比特时就去利用信道，从而取得灵活的带宽共享。来自各终端的数字流在链路控制器中形成完整的信元后，即按先到先服务的规则，经统计复用器，以统一的传输速率将信元插入一个空闲时隙内。链路控制器调节信息源进网的速率。不同类型的服务都可复用在一起，高速率信源就占有较多的时隙。交换设备只需按网络最大速率来设置，它与用户设备的特性无关。

（3）**所有信息在最低层是以面向连接的方式传送，保持了电路交换在保证实时性和服务质量方面的优点**。但对用户来说，ATM 既可工作于确定方式（即承载某种业务的信元基本上周期性地出现），以支持实时型业务；也可以工作于统计方式（即信元不规则地出现），以**支持**突发型业务。

（4）ATM 使用光纤信道传输。由于光纤信道的误码率极低，且容量很大，因此**在 ATM 网内不必在数据链路层进行差错控制和流量控制**（**放在高层处理**），因而明显地提高了信元在网络中的传送速率。

ATM 的一个明显缺点就是**信元首部的开销太大**，即 5 字节的信元首部在整个 53 字节的信元中所占的比例相当大。

由于 ATM 具有上述的许多优点，因此在 ATM 技术出现后，不少人曾认为 ATM 必然成为未来的宽带综合业务数字网 B-ISDN 的基础。但实际上 ATM 只是用在因特网的许多主干网中。ATM 的发展之所以不如当初预期的那样顺利，主要是因为 ATM 的技术复杂且价格较高，同时 ATM 能够直接支持的应用不多。与此同时，无连接的因特网发展非常快，各种应用与因特网的衔接非常好。在 100 Mb/s 的快速以太网和吉比特以太网推向市场后，10 吉比特以太网又问世了。这就进一步削弱了 ATM 在因特网高速主干网领域的竞争能力。

一个 ATM 网络包括两种网络元素，即 ATM **端点**(endpoint)和 ATM **交换机**。

ATM 端点又称为 ATM **端系统**，即在 ATM 网络中能够产生或接收信元的源站或目的站。ATM 端点通过点到点链路与 ATM 交换机相连。

ATM 交换机就是一个快速分组交换机（交换容量高达数百 Gb/s），其主要构件是**交换结构**(switching fabric)、若干个高速输入端口和输出端口、以及必要的缓存（见图 5-18，这里没有画出缓存）。该 ATM 交换机有 4 个输入端口和 4 个输出端口。在交换结构的作用下（后面的 5.6.3 节还要讨论和 ATM 交换机的工作过程），从输入端口 a 进入到 ATM 交换机的信元，经过交换结构从输出端口 g 离开 ATM 交换机，而从输入端口 c 进入到 ATM 交换机的信元，经过交换结构后从端口 f 离开 ATM 交换机。

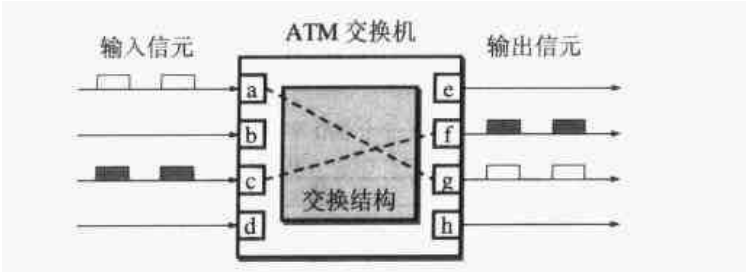


图 5-18　ATM 交换机的交换结构和输入端口、输出端口

由于 ATM 标准并不对 ATM 交换机的具体交换结构做出规定，因此现在已经出现了多种类型的 ATM 交换结构。限于篇幅，本书不讨论 ATM 交换机的交换结构的具体实现方法。

最简单的 ATM 网络可以只有一个 ATM 交换机，并通过一些点到点链路与各 ATM 端点相连。较小的 ATM 网络只拥有少量的 ATM 交换机，一般都连接成网格状网络以获得较好的连通性。大型 ATM 网络则拥有较多数量的 ATM 交换机，并按照分级的结构连成网络。

5.6.2　ATM 的协议参考模型和信元结构

1．ATM 的协议参考模型

制定 ATM 标准的最主要的组织机构有 ITU-T 和 ATM 论坛(ATM Forum)[W-ATM]以及 IETF 等。下面介绍 ATM 的协议参考模型（图 5-19）。

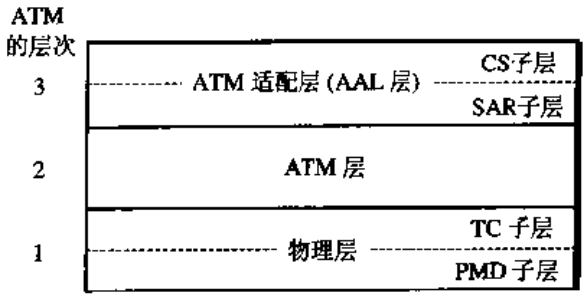


图 5-19　ATM 的协议参考模型

ATM 的协议参考模型共有三层，**大体上**与 OSI 的最低两层相当（但**无法严格**对应）。

（1）**物理层**

物理层又分为两个子层。靠下面的是**物理媒体相关**(Physical Medium Dependent)**子层**，即

PMD 子层。PMD 子层的上面是传输汇聚(Transmission Convergence)子层，即 TC 子层。

- **PMD 子层**　负责在物理媒体上正确传输和接收比特流。它完成只和媒体相关的功能，如线路编码和解码、比特定时以及光电转换等。对不同的传输媒体 PMD 子层是不同的。可供使用的传输媒体有：铜线(UTP 或 STP)、同轴电缆、光纤（单模或多模）或无线信道等。
- **TC 子层**　实现信元流和比特流的转换，包括速率适配（空闲信元的插入）、信元定界与同步、传输帧的产生与恢复等。在发送时，TC 子层将上面的 ATM 层交下来的信元流转换成比特流，再交给下面的 PMD 子层。在接收时，TC 子层将 PMD 子层交上来的比特流转换成信元流，标记出每一个信元的开始和结束，并交给 ATM 层。TC 子层的存在使得 ATM 层实现了与下面的传输媒体完全无关。典型的 TC 子层就是 SONET/SDH。

图 5-20 给出了一个例子，说明 ATM 的信元流是怎样装入到一个 STM-1 帧（STM-1 的速率就是 OC-3 的速率）中。这是一个 9 行×270 列的字节排列。

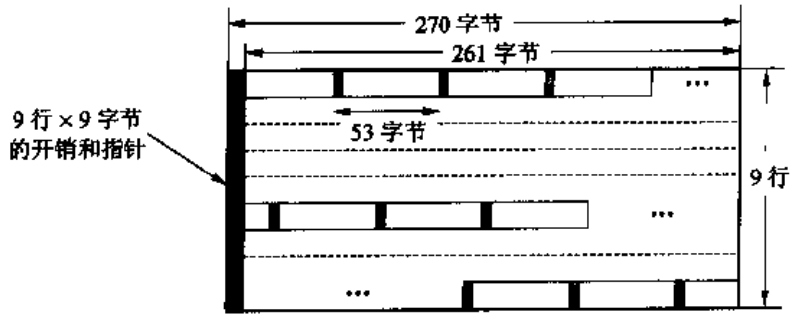


图 5-20　ATM 的信元流装入一个 STM-1 帧的例子

一个 STM-1 帧共有 270 × 9 字节（由于每帧的字节数太大，因此将一个帧画成为长方形结构，每帧有 9 行，每行 270 字节），即 2430 字节，或 19 440 bit。每秒共发送 8000 帧，因此数据率为 155.520 Mb/s。在每一行的 270 字节中的前 9 字节作为开销(overhead)和指针(pointer)之用（其格式相当复杂，这里从略），剩下的空间就用来放入 ATM 信元，因此这部分空间叫做有效载荷(payload，有人也将此名词译为“净负荷”），它共有 9 行，每行 261 字节，总数是 2349 字节。由于 2349 并不是 53 的整数倍，一个信元会跨过有效载荷区域的边界，因此需要用指针来指明一个信元的边界在何处。

ATM 物理层中的 TC 子层的许多功能类似于 OSI 模型的数据链路层。

（2）ATM 层

主要完成交换和复用功能，与传送 ATM 信元的物理媒体或物理层无关。

每一个 ATM 连接都用信元首部中的两级标号来识别。第一级标号是虚通路标识 VCI (Virtual Channel Identifier)，第二级标号是虚通道标识符 VPI (Virtual Path Identifier)[①]。

一个虚通路 VC 是在两个或两个以上的端点之间的一个运送 ATM 信元的通信通路。

一个虚通道 VP 包含有许多相同端点的虚通路，而这许多虚通路都使用同一个虚通道标识符 VPI。在一个给定的接口，复用在一条链路上的许多不同的虚通道，用它们的虚通道标

① 注：这两个名词的译名比较混乱。有人将 Virtual Path 译为“虚通路”或“虚路径”，而将 Virtual Channel 译为“虚通道”或“虚信道”。但使用它们的英文缩写 VPI 和 VCI 则是无二义性的。请读者在阅读中文的 ATM 资料时特别注意。

识符 VPI 来识别。而复用在一个虚通道 VP 中的不同的虚通路，用它们的虚通路标识符 VCI 来识别。图 5-21 表示了使用 VPI 和 VCI 来标识 VP 和 VC 的方法。

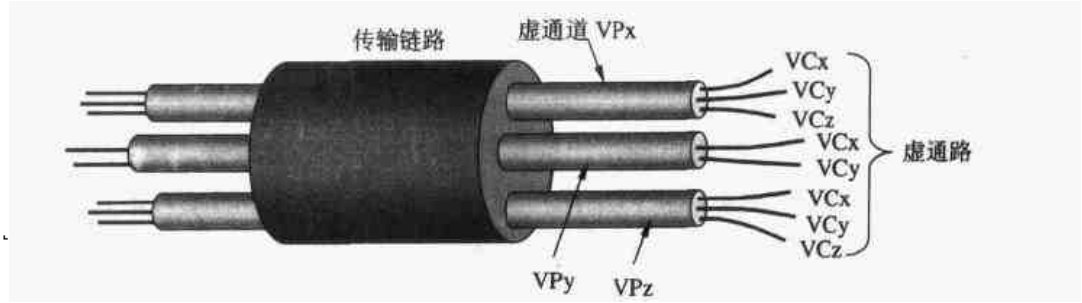


图 5-21　ATM 连接的标识符 VCI 和 VPI

在一个给定的接口上，属于两个不同的 VP 的两个 VC，可以具有相同的 VCI。如图中所示的三个不同的虚通道 VP 都可以使用相同的虚通路标识符 VCx 或 VCy。因此，要同时使用 VPI 和 VCI 这两个参数才能完全识别一个虚通路 VC。应当强调指出，一个给定的 VCI 值没有端到端的意义。VP 在经过集中器或交换机时，其 VPI 也会改变。关于这点后面还要讨论。

ATM 层的功能是：

- 信元的复用与分用；
- 信元的 VPI/VCI 转换（就是将一个入信元的 VPI/VCI 转换成新的数值）；
- 信元首部的产生与提取；
- 流量控制。

（3）ATM 适配层

ATM 传送和交换的是 53 字节固定长度的信元。但是上层的应用程序向下层传递的并不是 53 字节长的信元。例如，在因特网的 IP 层传送的是各种长度的 IP 数据报。因此当 IP 数据报需要在 ATM 网络上传送时，就需要有一个接口，它能够将 IP 数据报装入一个个 ATM 信元，然后在 ATM 网络中传送。这个接口就是在 ATM 层上面的 **ATM 适配层**，记为 **AAL** (ATM Adaptation Layer)。

AAL 层的作用就是**增强 ATM 层所提供的服务，并向上面高层提供各种不同的服务**。图 5-22 表示不同的信号源发出的信号（话音、视频、数据等）通过 AAL 层后都变成了固定长度的信元（53 字节长），然后再交给 ATM 层。

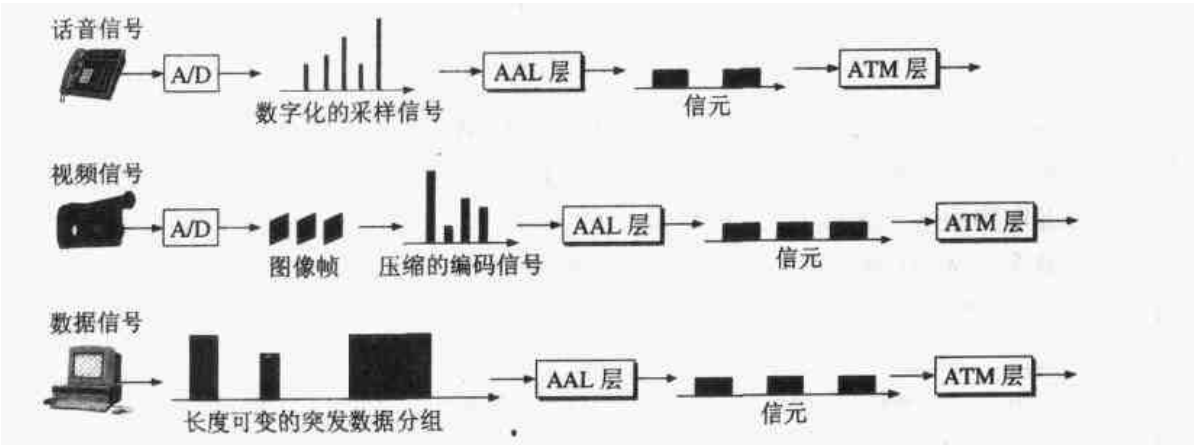


图 5-22　AAL 层将 53 字节长的信元交给 ATM 层

ITU-T 的 I.362 规定了 AAL 向上提供的服务是：

- 将用户的应用数据单元 ADU[①]划分为信元或将信元重装成为应用数据单元 ADU；
- 对比特差错进行检测和处理；
- 处理丢失和错误交付的信元；
- 流量控制和定时控制。

ITU-T 最初规定了 ATM 网络可向用户提供四种类别(class)的服务，从 A 类到 D 类。服务类别的划分是根据：**比特率是固定的还是可变的；源站和目的站的定时是否需要同步；是面向连接还是无连接**。后来 ITU-T 将 AAL 改分为四种类型(type)，并且一种类型的 AAL 可支持不止一种类别的服务。随后 ITU-T 发现没有必要划分开类型 3 和类型 4。于是将这两个类型合并，取名类型 3/4。但实践证明需要增加一种新的类型，即类型 5，用来支持各类服务。从目前 ATM 的使用情况来看，AAL5 是受到计算机界普遍欢迎的的一个 ATM 适配层。

为了方便，AAL 层又划分为两个子层，即 CS 子层和 SAR 子层，CS 子层在上面。

- **汇聚子层 CS** (Convergence Sublayer)　使 ATM 系统可以对不同的应用（如文件传送、点播视像等）提供不同的服务。每一个 AAL 用户通过相应的服务访问点 SAP（即应用程序的地址）接入到 AAL 层。在 CS 子层形成的协议数据单元叫做 CS-PDU。
- **拆装子层 SAR** (Segmentation And Reassembly)　在发送时，将 CS 子层传下来的协议数据单元 CS-PDU 划分成为长度为 48 字节的单元，交给 ATM 层作为信元的有效载荷。在接收时，SAR 子层进行相反的操作，将 ATM 层交上来的 48 字节长的有效载荷装配成 CS-PDU。这样，SAR 子层就使得 ATM 层与上面的应用无关。

需要强调的是，AAL 层的功能只能驻留在 ATM 端点之中，而在 ATM 交换机中只有物理层和 ATM 层。图 5-23 表示出了这个概念。对应于每一个 ATM 交换机的物理层都画了两个，这表示在不同的链路可采用不同的物理传输媒体。

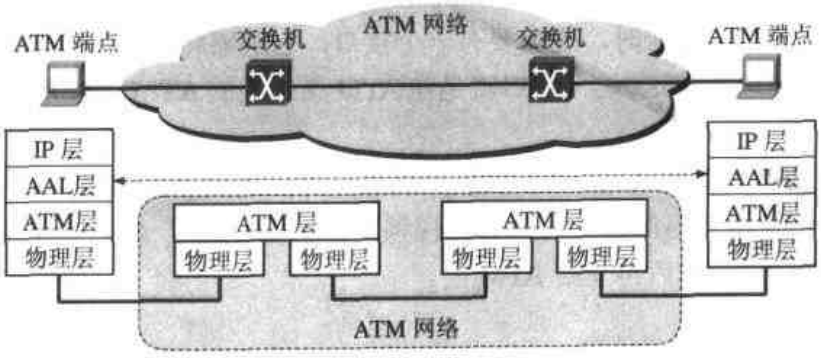


图 5-23　AAL 层只能驻留在 ATM 端点之中

从图 5-23 我们可以看出，当孤立地观察一个 ATM 网络时，ATM 网络像一个广域网，因为它可以覆盖很大的地理范围，有自己网络的硬件地址和进行信元转发的结点交换机，并且向上提供虚电路服务。因此有关 ATM 技术的内容可以放在广域网这一章来讨论。不过从 IP 层来看，整个的 ATM 网络又相当于两个 IP 结点之间的一条数据链路。从这个角度看，整个 ATM 网络又好像是处在数据链路层。这就是为什么在前面曾指出，ATM 体系结构中的层次

① 注：请注意，应用数据单元 ADU 并不是协议栈中的应用层的数据单元。在 ATM 的标准术语中，在 AAL 层上面的层次的数据单元都叫做应用数据单元 ADU。如果 AAL 层的上面是 IP 层，则 IP 数据报就是这里所说的应用数据单元 ADU。

和 OSI 的层次很难有严格的对应关系。

以上所述可归纳为图 5-24 所示的砂漏模型（即形状是上下宽、中间窄的砂漏）。这个模型说明 ATM 层可以支持上面 AAL 层的许多种不同的服务，而在 ATM 层下面也可以有许多种不同的物理层能够支持 ATM。

我们还要指出，上面所讨论的 ATM 协议参考模型假定了 ATM 网络拓扑是由点到点链路构成的。当 ATM 网络包括共享媒体的拓扑时（如使用了已有的局域网），则在 ATM 层和物理层之间还应加上一个局域网的 MAC 层。由于 ATM 网络最好使用点到点的连接，因此在 ATM 的协议参考模型中一般都不画出 MAC 层。

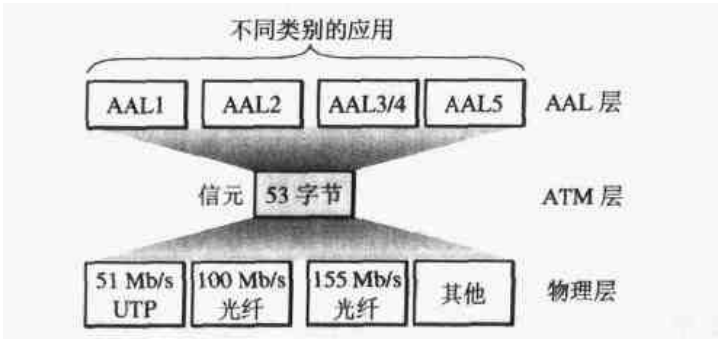


图 5-24　ATM 的砂漏模型

最后要指出，ATM 协议参考模型和 OSI 的七层协议模型（或原理体系结构的五层协议模型）并没有严格的对应关系。

2. ATM 的信元结构

信元长度主要由两个互相矛盾的因素决定。当采用分组交换时，分组越长，分组首部的额外开销就相对越小。对于某些应用，例如分组话音通信或分组图像通信，过长的分组将导致时延增大（因为实时话音或图像的比特信息要等到装配成一个完整的分组后才能发送到线路上）因而使通信质量变坏。当时延较大时，如果再加上有回声，则恢复后的话音质量还要更差。因此，在选择最佳分组长度时应折中处理。

ATM 信元采用固定长度的分组，它由 5 个字节的首部和 48 字节的信息字段组成。信元首部包含着在 ATM 网络中传递信息所需的信息。图 5-25 为 ATM 信元的结构。

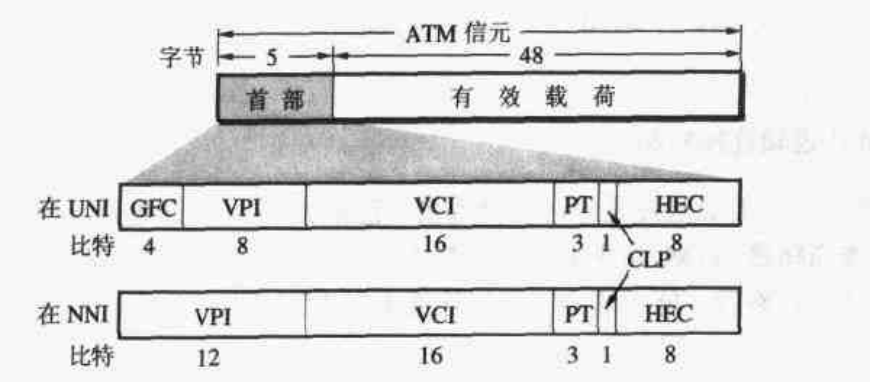


图 5-25　ATM 信元的结构

ATM 信元有两种不同的首部，分别对应于用户到网络接口 UNI (User-to-Network Interface)

和**网络到网络接口** NNI (Network-to-Network Interface)。在这两种接口上的 ATM 信元首部仅仅是前两个字段不同，后面的字段全都一样。

下面介绍 ATM 信元首部中各字段的作用。

（1）**通用流量控制 GFC** (Generic Flow Control)　4 bit 字段，通常置为 0。根据 ITU-T 标准，GFC 用来在共享媒体上进行接入流量控制，现在的点到点配置不需要这一字段的接入控制功能。GFC 字段完成的是类似局域网 MAC 子层的功能，与 ATM 层的处理无关。

（2）**VPI/VCI**　即**路由字段**，总共 24 bit。VPI/VCI 字段和帧中继中的 DLCI 字段的作用相似。对于用户到网络接口 UNI，VPI/VCI 字段中的虚通道标识符 VPI 占 8 bit，而虚通路标识符 VCI 占 16 bit。因此理论上一个主机最多可建立 256 个虚通道，而每一个虚通道所包含的虚通路数最多可达 65 536。对于网络到网络接口 NNI，则 VPI/VCI 字段中的 VPI 和 VCI 各占 12 bit。需要注意的是，许多 VPI/VCI 取值组合已经被定义了特定的功能，比如 VPI = 0, VCI = 5，用来作为点到点连接和一点到多点连接的默认的信令[KWOK98]。顺便指出，使用 VPI 和 VCI 信息的交换机称为 ATM **交换机**，而仅使用 VPI 信息的就是 ATM **交连机**(cross connect switch)，它是一种特殊的 ATM 交换机。

（3）**有效载荷类型 PT** (Payload Type)　3 bit 字段，用来区分该信元是用户信息或非用户信息。此字段又称为**有效载荷类型指示 PTI** (I 表示 Indicator)。第一个比特为 0 表示是用户数据信元，第二个比特表示有无遭受到拥塞（该比特由网络的 ATM 交换机填写），第三个比特用来区分服务数据单元 SDU 的类型。[①]例如，一个信元发出时 PT 为 000，可能在到达时就变成了 010，因而目的站就知道网络出现了拥塞。

（4）**信元丢失优先级 CLP** (Cell Loss Priority)　1 bit 字段，指示信元的丢失优先级（这个比特和帧中继的 DE 比特的作用相似）。当网络负荷很重时，ATM 交换机首先丢弃 CLP = 1 的信元以缓解网络可能出现的拥塞。除端系统可直接指定 CLP 值，网络还可能将违反通信量合约(contract)的信元的 CLP 从 0 改为 1，这个过程称为“**打标记**”(tagging)。

（5）**首部差错控制 HEC** (Header Error Control)　8 bit 字段。HEC 是首部的第五个字节。HEC 只对首部的前四个字节（但不包括有效载荷部分）进行循环冗余检验，并将检验的结果放在 HEC 字段中。HEC 可进行多个比特的检错和单个比特的纠错。值得注意的是 HEC 虽属 ATM 信元首部，却是由物理层产生和进行检验的。

在 ATM 的各字段中，GFC 字段涉及到 MAC 子层（选项）；VPI 和 VCI 字段、PT 字段以及 CLP 字段涉及到 ATM 层；HEC 字段涉及到物理层；有效载荷字段涉及到 AAL 层。

5.6.3　ATM 的逻辑连接机制

在 ATM 中使用的虚通路是一种逻辑连接，它和 X.25 中的虚电路或帧中继中的数据链路连接相似。**虚通路是 ATM 网络中的一个基本交换单元**。两个端用户要进行通信，首先必须建立虚通路连接，然后才能在这个端到端连接上以固定信元长度和可变速率进行全双工的通

① 注：这里的服务数据单元 SDU 是指从上面的 SAR 子层交下来的 48 字节的有效载荷。PT 字段的第三个比特在 ITU-T 的文档中定义为“ATM 用户到 ATM 用户(AAU)指示比特”，其意义是一样的。

信。数据传送完毕后再释放连接。

1．逻辑连接的建立和释放

我们以图 5-26 的简单 ATM 网络为例来说明逻辑连接建立和释放的大致过程。

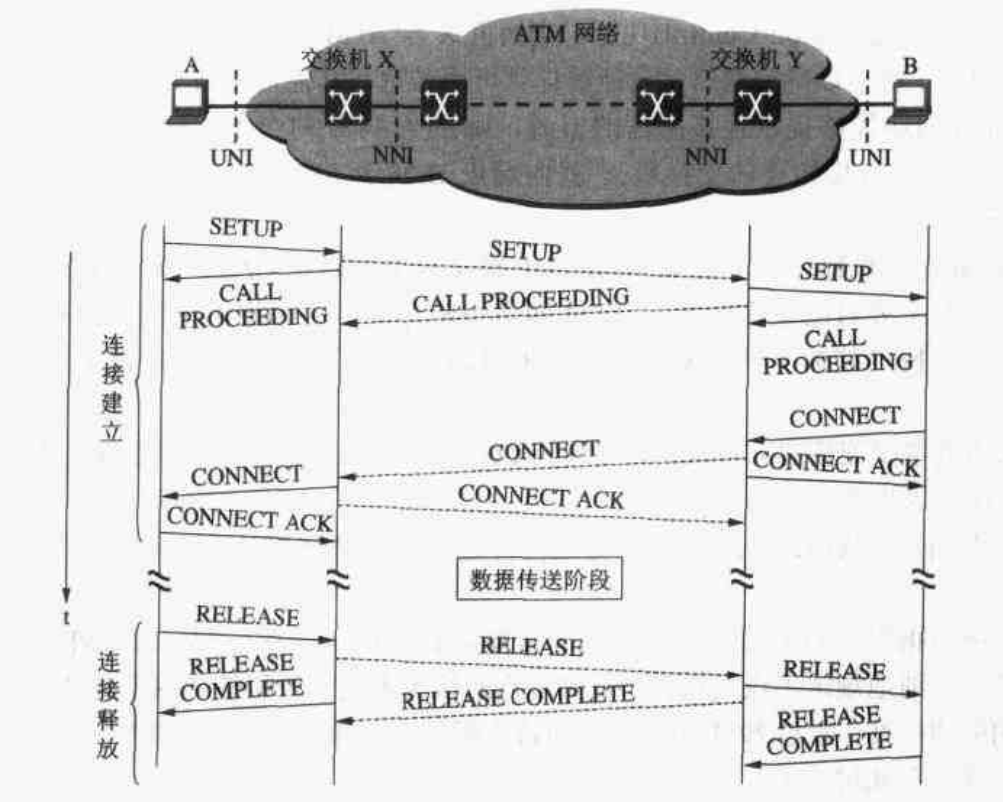


图 5-26　两个端点通过 ATM 网络建立逻辑连接

ATM 用于连接建立和连接释放的主要信令报文(signaling message)都在图 5-26 中用到了，各种报文的意义如表 5-2 所示。需要指出的是，每一种报文可以用一个或多个信元来传送。报文的内容包括报文类型、长度以及一些参数（如 ATM 地址[①]、在每一个方向所需的带宽和服务质量 QoS，以及对该连接所指派的 VPI/VCI 等）。

表 5-2　ATM 的主要信令报文

报 文 类 型	当主机发送时的意义	当网络发送时的意义
SETUP	请求建立连接	有一个入呼叫(incoming call)
CALL PROCEEDING	收到入呼叫	连接建立的请求正在进行处理
CONNECT	接受入呼叫	呼叫请求已被接受
CONNECT ACK	对 CONNECT 报文的确认	对 CONNECT 报文的确认
RELEASE	请求释放连接	一个端点发出了连接释放请求
RELEASE COMPLETE	对 RELEASE 的确认	对 RELEASE 的确认

① 注：ATM 的编址使用 ISO 制订的 20 个字节的网络服务访问点 NSAP 编码。这里又包括几种不同的格式。ITU-T 的 E.164 规定的 8 字节地址是 NSAP 编码格式中的一个字段[KWOK98]。

这里需要解释几点：

一个连接可能要通过 ATM 网络中的许多个交换机。为简单起见，在图 5-26 中表示报文传送过程的虚线箭头代表了逻辑连接中可能要经过很多个交换机。表 5-2 中的“入呼叫”是常用的名词，它与“连接建立请求”是同样的意思。

与端点 A 相连接的 ATM 交换机 X 使用 UNI 协议对信令报文进行处理，并完成呼叫准许控制 CAC (Call Admission Control)功能。交换机 X 与 ATM 网络中的其他交换机则使用 NNI 协议进行协同工作，以确定网络是否能够提供可用带宽和其他资源来支持此连接。

若 ATM 网络能够提供连接所需的资源，则从 A 到 B 的一条物理路由才能确定下来。这时，与端点 B 直接相连的交换机 Y 就向端点 B 发送 SETUP 报文，表示有一个入呼叫到来。

各 ATM 交换机或端点 B 在收到 SETUP 报文后所发送的 CALL PROCEEDING 报文是一个选项，其主要目的是在较大的 ATM 网络中防止主叫方因迟迟未能收到被叫方的应答而采取超时重发的措施。响应 CALL PROCEEDING 报文可使发送 SETUP 报文的结点耐心等待。

被叫方若接受连接建立请求，则应响应 CONNECT 报文。其余的信令报文的意义从图上都可反映出，这里不再解释。

我们要指出，ATM 标准没有指明任何特定的路由选择算法，这由各使用单位自行确定。

最后应当指出，ATM 的信令过程是非常复杂的。ATM 论坛已制订出 UNI 3.1 和 UNI 4.0 信令标准，详细地规定了点到点和一点到多点的信令报文格式和整个的控制过程。ITU-T 也制订了相应的标准，如 Q.2931, Q.2110（即特定服务面向连接协议 SSCOP）和 Q.2130 等建议书。这些就不在此讨论了。

2．虚通路标识符 VCI 和虚通道标识符 VPI 的转换

前面已经讲过，ATM 信元在 ATM 网络中传输时，一定是在某一个特定的虚连接上按序传送的。因此，ATM 信元的首部一定要有这个虚连接的标识符 VPI/VCI，以便惟一地标识该信元属于哪一个虚通路。若 ATM 交换机收到一个信元但其标识符 VPI/VCI 与该交换机所知道的任何虚通道/虚通路标识符均无联系，则 ATM 交换机就丢弃此信元。

那么 ATM 信元是怎样得到其标识符 VPI/VCI 呢？

设端点 A 与端点 B 进行通信，它们之间建立了一条逻辑连接。端点 A 必须给新建立的连接指派一个标识符 VPI/VCI。显然，这个标识符 VPI/VCI 不能和端点 A 正在使用的其他标识符 VPI/VCI 重复。然而端点 B 可同时接收其他一些端点发来的信元，这些端点都是独立地选择自己的标识符 VPI/VCI 值。这样，端点 B 很可能收到来自不同端点的但具有相同 VPI/VCI 的信元，结果使端点 B 无法确定这些信元的源端点。

为了解决上述矛盾，最简单的方法就是使所有的 VPI/VCI 值只在每一段物理链路上具有惟一的值。当信元在 ATM 网络中传送时，每经过一段链路，其 VPI/VCI 值都可能改变数值。在下面图 5-27 的例子中，我们设从端点 A 到端点 B 经过 ATM 交换机 X, Y 和 Z。图中用较粗的线表示信元通过的这条特定路径 A→X→Y→Z→B。

设端点 A 选择 VPI/VCI = 3/17（这种记法表示 VPI = 3 而 VCI = 17）作为供信元从端点 A 到交换机 X 之间传送时使用的标识符，而信元从交换机 X 的端口 4 进入该交换机。假定交换

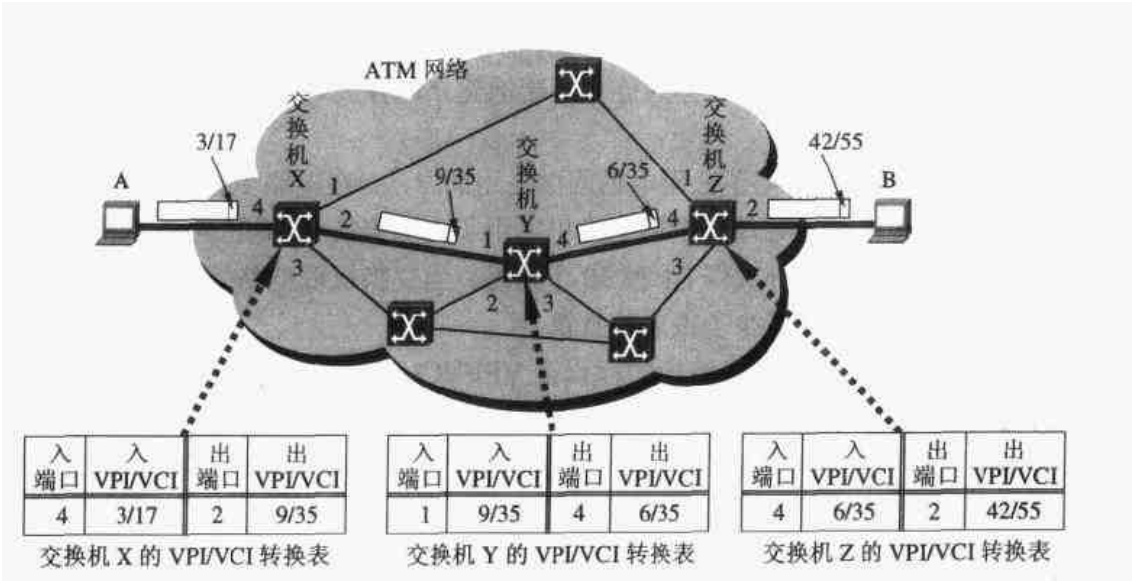


图 5-27　端点 A 通过 ATM 交换机 X, Y 和 Z 与端点 B 建立了一条逻辑连接

机 X 再从其端口 2 向交换机 Y 转发此信元，并将 VPI/VCI 转换为交换机 X 未在使用的 VPI/VCI 号，如 9/35。交换机 Y 从自己的端口 1 收到此信元，再从端口 4 向交换机 Z 进行转发。现在假定 Y 是一个交连机，因此它只改变虚通道标识符 VPI 的数值而不改变虚通路标识符 VCI 的数值。因此通过交连机 Y 后，信元的 VPI/VCI 从 9/35 转换为 6/35。最后，交换机 Z 从其端口 4 收到信元，再从自己的另一个端口 2 向端点 B 进行转发，并将 VPI/VCI 从 6/35 转换为 42/55。这样，一个 ATM 信元从端点 A 经过交换机 X→Y→Z 最后到端点 B，其标识符 VPI/VCI 的数值所经历的变化是：

VPI/VCI：3/17 → 9/35 → 6/35 → 42/55

在第 1 章 1.2.1 节我们提到过分组交换是“**基于标记的**”就是这个意思。ATM 采用的是快速分组交换，它所基于的标记就是这里的标识符 VPI/VCI。

由此可见，在每一个交换机中都应当有一个 VPI/VCI 的**转换表**（这种转换表也称为**信元的转发表**），其中至少要有 4 个参数，即入端口号、入 VPI/VCI 值、出端口号以及出 VPI/VCI 值。应当注意的是，在上述的 VPI/VCI 转换表中存放的是已经建立的虚连接的信元转发信息。但是在初次建立一个新的连接时，一个 ATM 交换机还需要先建立一个这样的 VPI/VCI 转换表来进行信元的转发。

上述这种 VPI/VCI 的指派和转换方法有两个好处。第一，**能够解决 ATM 网络的可扩缩性(scalability)**。由于给某一条链路指派一个 VPI/VCI 值并不需要网络的全局信息，因此每个交换机只要保证与它直接相连的链路具有惟一的 VPI/VCI 值即可。第二，使 VPI/VCI 值只具有局部意义可大大增加全网可供指派的 VPI/VCI 值的数目。否则当 ATM 网络的端点数增多时，每个端点所能分配到的 VPI/VCI 值的数目就会减少。这点与蜂窝无线电在不同蜂窝小区的频率重复使用是很相似的。

我们应注意到，端点 A 发出的信元的 VPI/VCI 值与端点 B 收到的信元的 VPI/VCI 值是不一样的。例如在上述例子中，端点 B 收到的信元的 VPI/VCI = 42/55。但 B 并不知道，也不需要知道 A 发出的信元的 VPI/VCI = 3/17。其实只要 B 收到 VPI/VCI = 42/55 的信元就知道是从 A 经过这条连接传送过来的。只有网络中的交换机才清楚 VPI/VCI 的转换过程。

在一个较大的 ATM 网络中，通过 ATM 交换机的虚通路可能有几千条。在 VPI/VCI 转换表中每一条虚通路都要占据一行。当虚连接不断地建立和释放时，对交换机的内存和处理机来说都是相当大的负担。

ATM 在解决上述问题借用了公用电话网中采用的方法。在电路交换的电话网中，交换机要对几千条的电话连接进行处理。电话交换机采用了**交叉连接交换机**(crossconnect switch)对群路进行交换。例如，一个一次群 E1 有 30 个话路。当交换机以群路为单位进行交换时，就比以单个话路为单位的交换减少了很多的开销。

与此相似，ATM 采用了两级结构的标识符 VPI/VCI。假定有许多的 ATM 连接，它们的源点和终点在地理上都比较集中，那么这些连接就可以使用一个共同的虚通道标识符 VPI。这样，在 ATM 网络中的交换机就只需将 VPI 值进行转换，而不管每条连接的 VCI 值是多少。这样的 VPI 转换表要比 VPI/VCI 转换表简单得多。

5.6.4 AAL 层举例：AAL5

下面我们以使用得较多的 AAL5 层为例来说明 AAL 层的作用。

AAL5 现已成为最重要的 AAL 协议类型。所有的支持**交换虚连接** SVC (Switched Virtual Connection)的 ATM 交换机和端点都必须支持 AAL5，因为标准化组织规定 ATM 的信令协议建立在 AAL5 之上。

AAL5 从 AAL3/4 简化得来，它去掉了后者的复用、每信元的 CRC 检验、序号和缓存分配等功能。因此 AAL5 又称为 SEAL (Simple Efficient Adaptation Layer)，表明 AAL5 既简单又高效。

AAL5 采用了检错能力很强的 32 bit 的 CRC 检验（像以太网和 FDDI 帧使用的那种循环冗余检验），能够检出比特差错、信元丢失和插入错误。另外，AAL5 的 CS-PDU 还采用了一个长度字段以检测信元丢失或插入错误。

任何分段重装协议都必须能够辨别各个分段的先后顺序，以及开始和结尾分段的位置。ATM 网络采用了面向连接方式保持信元的传送顺序。但开始和结尾分段的确定则需专门的机制完成。AAL5 协议并不采用给每个分段编号的方法来完成这个功能，而是利用了 ATM 层信元首部中的 PT 字段。若信元是一个分组的最后一个分段，则 PT = 001，而其他信元的 PT = 000。这样就使得 SAR-PDU 不包含任何开销。

图 5-28 表示用户数据（例如，可以是常用的 IP 数据报）先传递给 AAL5 的汇聚子层 CS，作为汇聚子层的数据。汇聚子层的协议数据单元 CS-PDU 只有一个 8 字节的**尾部**而无首部，其尾部由三个字段组成：

（1）**保留**　2 字节。其中前一个字节是用户到用户字段，不属于 AAL 层。

（2）**长度**　2 字节。它指明 AAL5 的 CS-PDU 中用户数据的字节长度。为了使用户数据加上尾部后能够成为 48 字节的整数倍（因为还要分割为整数个信元），在尾部之前有一个**填充字段**。显然，填充字段的长度是 0～47 字节。2 字节的长度字段允许 CS-PDU 的用户数据部分的长度从 1 到 $2^{16}-1$（即从 1 到 65 535）。然而实际上当 IP 数据报要在 AAL5 上传送时，TCP/IP 的标准对 IP 数据报规定的默认长度是 9180 字节。若超过这个数值，就必须经过通信双方事先的商定。

（3）**CRC 检验**　4 字节。

AAL5 的拆装子层 SAR 的协议数据单元 SAR-PDU 没有首部和尾部，对每个信元几乎没

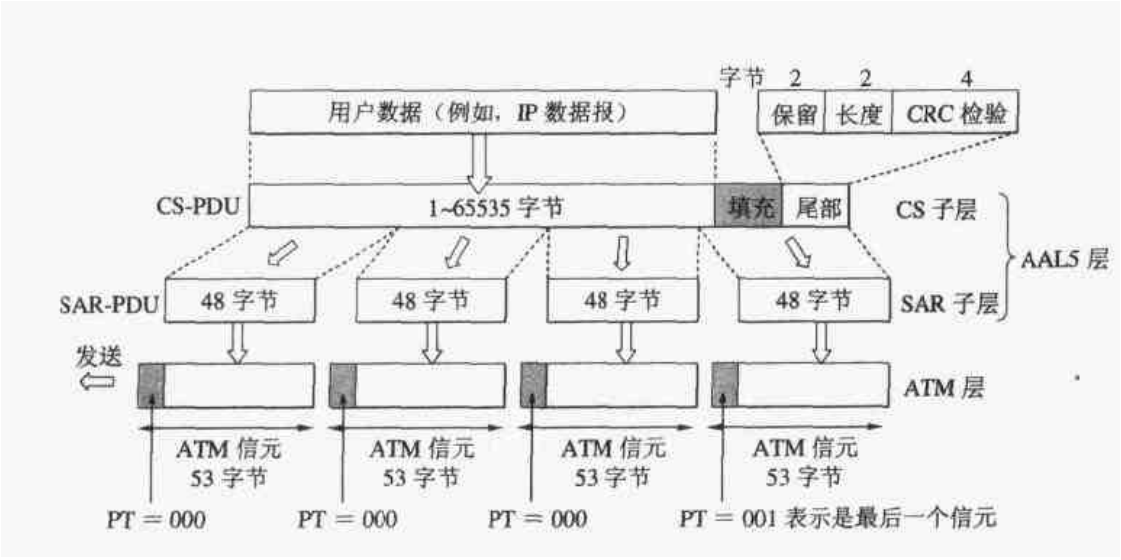


图 5-28　AAL5 将用户数据分割为 48 字节的数据块传递给 ATM 层

有 SAR 协议的处理。SAR-PDU 由 CS 子层交来的 48 字节的数据块组成。从图 5-28 可看出，一个 CS-PDU 被划分为整数个 48 字节的 SAR-PDU。每个 SAR-PDU 再加上 5 个字节的 ATM 信元首部就构成了一个完整的 ATM 信元。

在 ATM 领域还有很多的问题值得关注，例如，有关 ATM 的通信量的一些问题可参考本书的附录 D。

习题

5-01　试从多个方面比较虚电路和数据报这两种服务的优缺点。

5-02　设有一分组交换网。若使用面向连接的虚电路，则每一分组必须有 3 字节的分组首部，而每个网络结点必须为虚电路保留 8 字节的存储空间来识别虚电路。但若使用无连接的数据报，则每个分组要有 15 字节的分组首部，而结点就不需要保留转发表的存储空间。设每段链路每传 1 兆字节需 0.01 元。购买结点存储器的代价为每字节 0.01 元，而存储器的寿命为 2 年工作时间（每周工作 40 小时）。假定一条虚电路的每次平均使用时间为 1000 秒，而在此时间内发送 200 分组，每个分组平均要经过 4 段链路。试问：采用哪种方案（虚电路或数据报）更为经济？相差多少？

5-03　假定分组交换网中所有结点的处理机和主机均正常工作，所有的软件也正确无误。试问一个分组是否可能被投送到错误的目的结点（不管这个概率有多小）？

如果一个网络中所有链路的数据链路层协议都能正确工作，试问从源结点到目的结点之间的端到端通信是否一定也是可靠的？

5-04　广域网中的主机为什么采用层次结构方式进行编址？

5-05　一个数据报分组交换网允许各结点在必要时将收到的分组丢弃。设结点丢弃一个分组的概率为 p。现有一个主机经过两个网络结点与另一个主机以数据报方式通信，因此两个主机之间要经过 3 段链路。当传送数据报时，只要任何一个结点丢弃分组，则源点主机最终将重传此分组。试问：

（1）每一个分组在一次传输过程中平均经过几段链路？

（2）每一个分组平均要传送几次？

（3）目的主机每收到一个分组，连同该分组在传输时被丢弃的传输，平均需要经过几

段链路？

5-06 一个分组交换网其内部采用虚电路服务，沿虚电路共有 n 个结点交换机，在交换机中为每一个方向设有一个缓存，可存放一个分组。在交换机之间采用停止等待协议，并采用以下的措施进行拥塞控制。结点交换机在收到分组后要发回确认，但条件是：

（1）接收端已成功地收到了该分组；

（2）有空闲的缓存。

设发送一个分组需 T 秒（数据或确认），传输的差错可忽略不计，主机和结点交换机之间的数据传输时延也可忽略不计。试问：分组交付给目的主机的速率最快为多少？

5-07 假定要在误码率 $p = 10^{-6}$ 的链路上传送长度为 10^6 bit 的报文。整个报文就是一个大的分组。数据率为 1 Mb/s。若使用停止等待协议，并忽略分组首部的开销、传播时延和确认分组出错的概率，试求这个分组正确到达终点所需的平均时间。

若将此报文划分为 1000 个 1000 bit 长的分组，然后逐个发送到终点，试证明，所有分组正确到达终点所需的平均时间将比上面得出的时间减少约 63%。

5-08 流量控制在网络工作中具有何意义？流量控制与拥塞控制有何异同之处？

5-09 为什么说，“只要任意增加一些资源就可以解决网络拥塞的问题”是不正确的？

5-10 死锁是怎样形成的？有什么措施可用来防止死锁？

5-11 有 AB 和 BC 两条链路。A 经过 B 向 C 发送数据。B 收到 A 发来的数据时，可以先向 C 转发再向 A 发确认，也可以把这顺序反过来。也就是说，B 要做的三件事的顺序是：收数据→转发→发确认，或：收数据→发确认→转发。现假定 B 在做完第二件事后处理机即出故障，存储器中所存信息全部丢失，但很快又恢复了工作。试证明：只有采用端到端发确认信息的方法（即从 C 向 A 发确认信息），才能保证在任何情况下数据都能从 A 经 B 正确无误地交付到 C。

5-12 为什么 X.25 分组交换网会发展到帧中继？帧中继有什么优点？试从层次结构上以及结点交换机需要进行的处理过程进行讨论。

5-13 帧中继的数据链路连接标识符 DLCI 的用途是什么？什么是“本地意义”？

5-14 帧中继的拥塞控制是怎样进行的？承诺的信息速率 CIR 在拥塞控制中其何作用？

5-15 ATM 的主要优点是什么？UNI 和 NNI 有何不同？

5-16 试说明 ATM 的物理层、ATM 层和 AAL 层的作用（包括各子层的作用）。

5-17 ATM 中的 VCI 和 VPI 各有何用处？试举例说明，并分别计算在 UNI 和 NNI 接口上一共可以有多少条 ATM 连接？

5-18 试述 ATM 连接的建立过程。

5-19 ATM 信元的结构是怎样的？信元首部的结构分为哪些字段？各有何用处？

5-20 在传送一个 ATM 信元时，其路由选择决定是在什么时候作出的？

5-21 若 ATM 信元采用可变长度，那么会有何优点和缺点？

5-22 试判断以下各种说法的正确性。给出“是”或“否”的答案。

（1）ATM 的差错控制是对整个信元进行检验。

（2）ATM 信元的寻址是根据信元首部中的目的站地址。

（3）ATM 提供固定的服务质量保证。

（4）ATM 的信元采用动态路由选择。

（5）TCP 不能在 ATM 上运行。

（6）ATM 在每次传送数据之前必须先建立连接。

（7）ATM 的信元的长度是可变的。

5-23 在一个 ATM 网络的源端点和目的端点之间有三个 ATM 交换机。现在要建立一条虚通路。问一共需要发送多少个报文？

5-24 在同一幢大厦中有 N 个用户，他们都通过一个 ATM 网络使用同一个远地的计算机。每个用户平均每小时产生 L 行的通信量（输入和输出），平均每行长度为 P 字节（不包括 ATM 首部）。用户应交纳的费用是：对所传送的数据，每字节为 C 分钱，而使用 ATM 的虚通路，每小时 X 分钱。若将 N 个用户都复用到一条 ATM 虚通路上，则每一行的数据还要增加 2 字节的开销。假定一条虚通路的带宽足够 N 个用户使用。试问在什么条件下复用会更加便宜？

5-25 一个 1024 字节的报文用 ATM 的 AAL5 来传送，试问数据传输的效率是多少？

5-26 当收到一个 ATM 信元时要用 HEC 字段进行循环冗余检验。已知 ATM 所使用的循环冗余检验生成多项式 $P(X)= X^8 + X^2 + X +1$。试问当一个 ATM 信元的首部出现差错但仍能通过循环冗余检验的概率是多少？当一连两个首部出现差错的 ATM 信元仍能通过循环冗余检验的概率是多少？

5-27 试述 CBR、rt-VBR、nrt-VBR、UBR 以及 ABR 的主要特点。

第6章 网络互连

本章讨论网络互连问题，也就是讨论多个网络通过路由器互连成为一个**互连网**（或**互联网**）的各种问题。互连网的核心内容是网际协议 IP，这是本书的一个重点内容。只有较深入地掌握了 IP 协议的主要内容，才能理解因特网是怎样工作的。我们还要重点讨论因特网的路由选择协议，以及介绍因特网控制报文协议 ICMP 和因特网组管理协议 IGMP。最后简要地讨论下一代的网际协议 IPv6 的主要内容。

本章最重要的概念是：

（1）IP 地址与物理地址的关系。

（2）传统的分类的 IP 地址（包括子网掩码）和无分类域间路由选择 CIDR。

（3）路由选择协议的工作原理。

对上述概念务必弄清楚。

6.1 路由器在网际互连中的作用

在现实世界中，单一的网络无法满足各种用户的多种需求。因此，我们经常使用的计算机网络往往由许多种不同类型的网络互连而成。通常在谈到“互连”时，就已暗示这些通过各种网络相互连接的计算机不仅仅在物理上是连通的，更重要的是它们能进行通信。那么，这些网络是怎样连接起来的呢？是通过许许多多的路由器。在互联网中，路由器起到了关键的作用。本节首先讨论路由器是怎样构成的，接着介绍路由器的关键构件——交换结构，最后给出关于互联网（互连网）和因特网的概念。

6.1.1 路由器的构成

我们先简单地再说明一下路由器的作用。

当主机 A 要向另一个主机 B 发送数据报时，先要检查目的主机 B 是否与源主机 A 连接在同一个网络上。如果是，就将数据报**直接交付**给目的主机 B 而不需要通过路由器。

但如果目的主机 C 或 D 与源主机 A 不是连接在同一个网络上，则应将数据报发送给本网络上的某个路由器，由该路由器按照转发表指出的路由将数据报转发给下一个路由器。这就叫作**间接交付**。在数据报传输路径上的最后一个路由器与目的主机在同一个网络中，由该路由器把数据报直接交付给目的主机 C 或 D。可见离开路由器就无法在互联网上传送数据。图 6-1 是直接交付和间接交付的示意图。

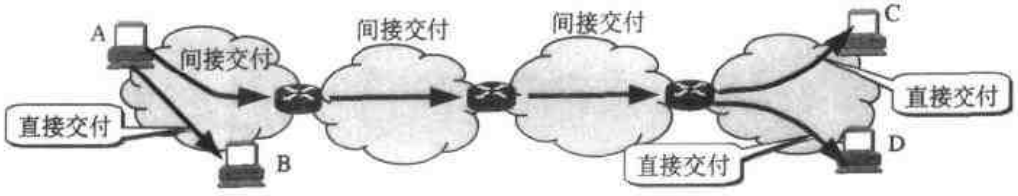


图 6-1 直接交付(A 发送到 B)和间接交付(A 发送到 C 或 D)

路由器是一种具有多个输入端口和多个输出端口的专用计算机，其任务是转发分组。也就是说，将路由器某个输入端口收到的分组，按照分组要去的目的地（即目的网络），将该分组从某个合适的输出端口转发给下一跳路由器。下一跳路由器也按照这种方法处理分组，直到该分组到达目的地为止。路由器的转发分组正是网络层的主要工作。图 6-2 给出了一种典型的路由器的构成框图。

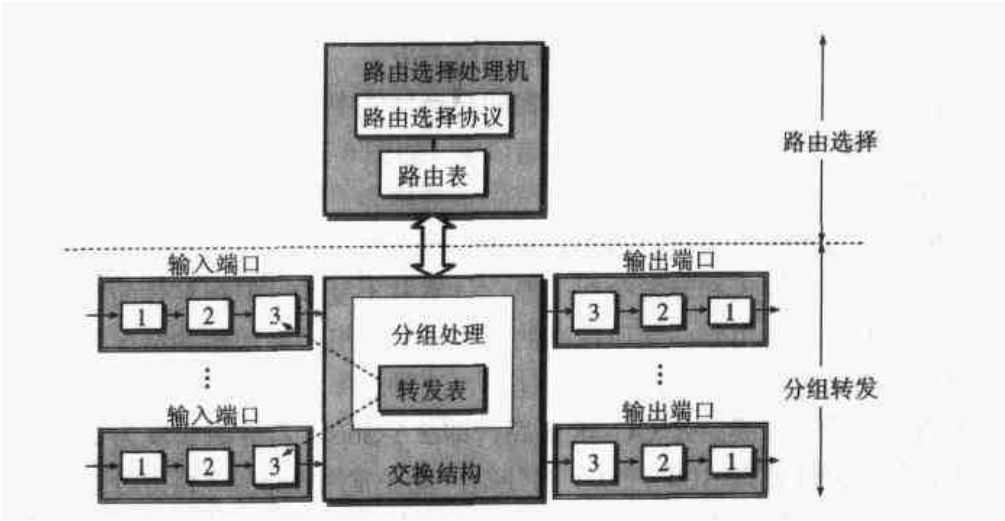


图 6-2　典型的路由器的结构（图中的数字 1～3 表示相应层次的构件）

从图 6-2 可以看出，整个的路由器结构可划分为两大部分：**路由选择**部分和**分组转发**部分。

路由选择部分也叫做**控制**部分，其核心构件是路由选择处理机。路由选择处理机的任务是根据所选定的路由选择协议构造出路由表，同时经常或定期地和相邻路由器交换路由信息而不断地更新和维护路由表。关于怎样根据路由选择协议构造和更新路由表，我们将在后面几节详细讨论。

分组转发部分是本节所要讨论的问题，它由三部分组成：**交换结构**、一组**输入端口**和一组**输出端口**。下面分别讨论每一部分的组成。

交换结构(switching fabric)又称为**交换组织**，它的作用就是根据**转发表**(forwarding table)对分组进行处理，将某个输入端口进入的分组从一个合适的输出端口转发出去。交换结构本身就是一种网络，但这种网络完全包含在路由器之中，因此交换结构可看成是“在路由器中的网络”。

在上一章已经提到过“转发”和“路由选择”的区别。在互联网中，“**转发**”就是路由器根据转发表将用户的 IP 数据报从合适的端口转发出去。但“**路由选择**”则是按照复杂的分布式算法，根据从各相邻路由器所得到的关于整个网络的拓扑变化情况，动态地改变所选择的路由。路由表是根据路由选择算法得出的。因此路由表一般仅包含从目的网络到下一跳（以 IP 地址表示，下一节就讨论什么是 IP 地址）的映射。而转发表是从路由表得出的。转发表必须包含完成转发功能所必须的信息。这就是说，在转发表的每一行必须包含从要到达的目的网络到输出端口和某些 MAC 地址信息（如下一跳的以太网地址）的映射。MAC 地址需要通过 ARP 协议才能得出（这将在 6.2.3 节介绍）。将转发表和路由表用不同的数据结构实现会带来一些好处，这是因为在转发分组时，转发表的结构应当使查找过程最优化，但路由表则需要对网络拓扑变化的计算最优化。路由表总是用软件实现的，但转发表则甚至可用特殊的硬件来实现。请读者注意，在讨论路由选择的原理时，往往不去区分转发表和路由表的区别，而是笼统地使用路由表这一名词。

在路由器的输入和输出端口里面都各有三个方框，用方框中的 1, 2 和 3 分别代表物理层、数据链路层和网络层的处理模块。物理层进行比特的接收。数据链路层则按照链路层协议接收传送分组的帧。在将帧的首部和尾部剥去后，分组就被送入网络层的处理模块。若接收到的分组是路由器之间交换路由信息的分组（如 RIP 或 OSPF 分组等），则将这种分组送交路由器的路由选择部分中的路由选择处理机。若接收到的是数据分组，则按照分组首部中的目的地址查找转发表，根据得出的结果，分组就经过交换结构到达合适的输出端口。一个路由器的输入端口和输出端口就做在路由器的**线路接口卡**上。

输入端口中的查找和转发功能在路由器的交换功能中是最重要的。为了使交换功能分散化，往往将复制的转发表放在每一个输入端口中（如图 6-2 中的虚线箭头所示）。路由选择处理机负责对各转发表的副本进行更新。这些副本常称为“**影子副本**”(shadow copy)。分散化交换可以避免在路由器中的某一点上出现瓶颈。

以上介绍的查找转发表和转发分组的概念虽然并不复杂，但在具体的实现中还是会遇到不少困难。问题就在于路由器必须以很高的速率转发分组。最理想的情况是输入端口的处理速率能够跟上线路将分组传送到路由器的速率。这种速率称为**线速**(line speed 或 wire speed)。可以粗略地估算一下。设线路是 OC-48 链路，即 2.5 Gb/s。若分组长度为 256 字节，那么线速就应当达到每秒能够处理 100 万以上的分组。现在常用 Mpps（百万分组每秒）为单位来说明一个路由器对收到的分组的处理速率有多高。在路由器的设计中，怎样提高查找转发表的速率已经成为一个十分重要的研究课题。

当一个分组正在查找转发表时，后面又紧跟着从这个输入端口收到另一个分组。这个后到的分组就必须在队列中排队等待，因而产生了一定的时延。图 6-3 给出了在输入端口的队列中排队的分组的示意图。

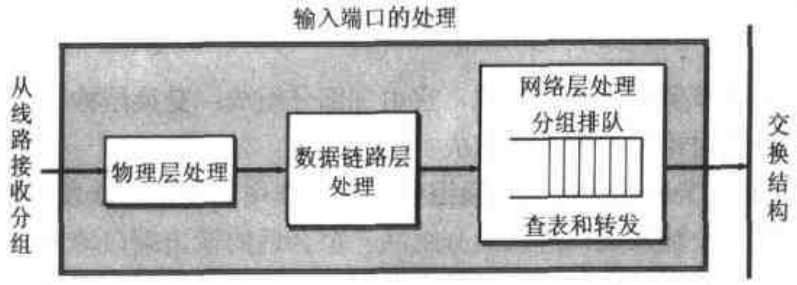


图 6-3　输入端口对线路上收到的分组的处理

我们再来观察在输出端口上出现什么情况（图 6-4）。输出端口从交换结构接收分组，然后将它们发送到路由器外面的线路上。在网络层的处理模块中设有一个缓存，实际上它就是一个队列。当交换结构传送过来的分组的速率超过输出链路的发送速率时，来不及发送的分

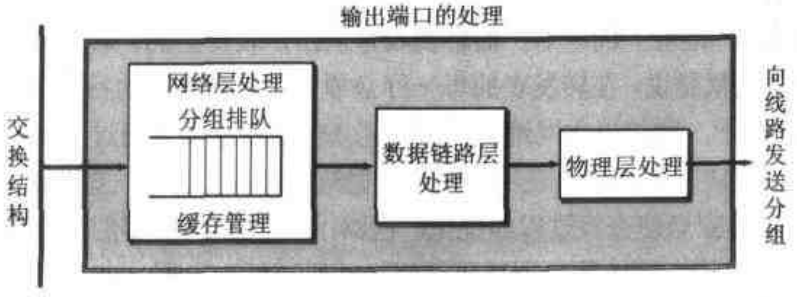


图 6-4　输出端口将交换结构传送过来的分组发送到线路上

组就必须暂时存放在这个队列中。数据链路层处理模块将分组加上链路层的首部和尾部，交给物理层后发送到外部线路。

从以上的讨论可以看出，分组在路由器的输入端口和输出端口都可能会在队列中排队等候处理。若分组处理的速率赶不上分组进入队列的速率，则队列的存储空间最终必定减少到零，这就使后面再进入队列的分组由于没有存储空间而只能被丢弃。以前我们提到过的分组丢失就是发生在路由器中的输入或输出队列产生溢出。当然，设备或线路出故障也可能使分组丢失。

6.1.2 交换结构

交换结构是路由器的关键构件[KURO01]。正是这个交换结构将分组从一个输入端口转移到某个合适的输出端口。实现这样的交换有多种方法，图 6-5 给出了三种常用的交换方法。这三种方法都是将输入端口 I_1 收到的分组转发到输出端口 O_2。下面简单介绍它们的特点。

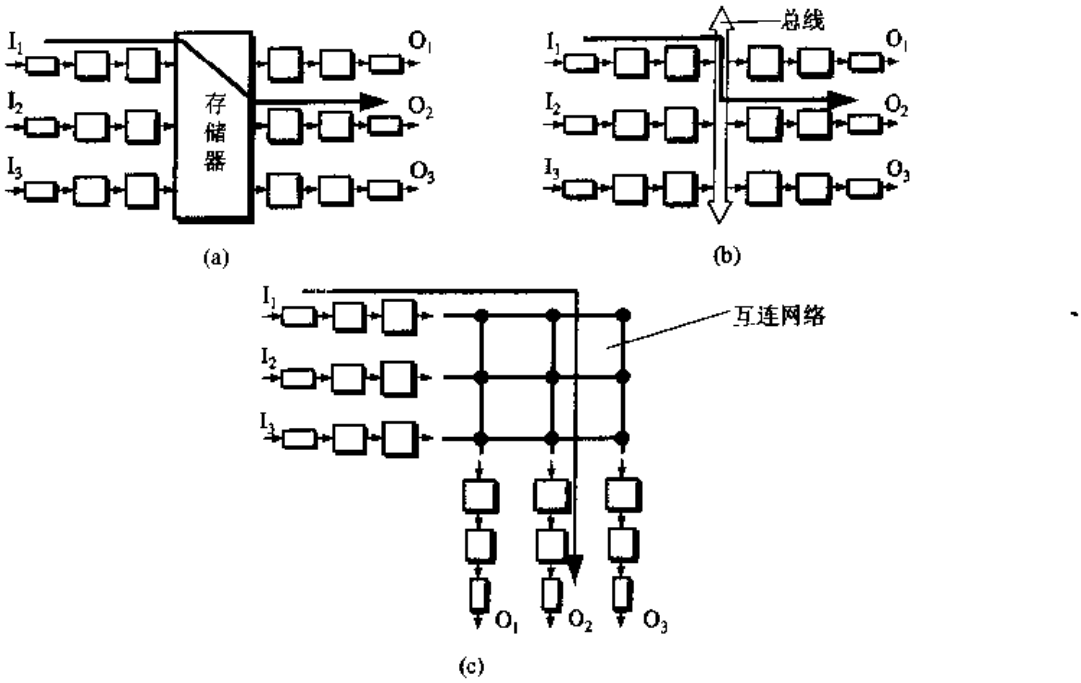


图 6-5 三种常用的交换方法：(a)通过存储器；(b)通过总线；(c)通过互联网络

1．通过存储器进行交换

最早使用的路由器就是利用普通的计算机，用计算机的 CPU 作为路由器的路由选择处理机。路由器的输入和输出端口的功能和传统的操作系统中的 I/O 设备一样。当路由器的某个输入端口收到一个分组时，就用中断方式通知路由选择处理机。然后分组就从输入端口复制到存储器中。路由器处理机从分组首部提取目的地址，查找路由表，再将分组复制到合适的输出端口的缓存中。若存储器的带宽（读或写）为每秒 M 个分组，那么路由器的交换速率（即分组从输入端口传送到输出端口的速率）一定小于 $M/2$。这是因为存储器对分组的读和写需要花费的时间是同一个数量级。

许多现代的路由器也通过存储器进行交换，图 6-5(a)的示意图表示分组通过存储器进行交换。与早期的路由器的区别就是目的地址的查找和分组在存储器中的缓存都是在输入端口

中进行的。Cisco 公司的 Catalyst 8500 系列交换机（有的公司把路由器也称为交换机）和 Bay Network 公司的 Accelar 1200 系列路由器就采用了共享存储器的方法。

图 6-5(b)是通过总线进行交换的示意图。采用这种方式时，数据报从输入端口通过共享的总线直接传送到合适的输出端口，而不需要路由选择处理机的干预。但是，由于总线是共享的，因此在同一时间只能有一个分组在总线上传送。当分组到达输入端口时若发现总线忙（因为总线正在传送另一个分组），则被阻塞而不能通过交换结构，并在输入端口排队等待。因为每一个要转发的分组都要通过这一条总线，因此路由器的转发带宽就受总线速率的限制。现代的技术已经可以将总线的带宽提高到每秒吉比特的速率，因此许多的路由器产品都采用这种通过总线的交换方式。例如，Cisco 公司的 Catalyst 1900 系列交换机就使用了带宽达到 1 Gb/s 的总线 (叫做 Packet Exchange Bus)。

图 6-5(c)画的是通过纵横交换结构(crossbar switch fabric)进行交换。这种交换机构常称为**互连网络**(interconnection network)，它有 $2N$ 条总线，可以使 N 个输入端口和 N 个输出端口相连接，这取决于相应的交叉结点是使水平总线和垂直总线接通还是断开。当输入端口收到一个分组时，就将它发送到与该输入端口相连的水平总线上。若通向所要转发的输出端口的垂直总线是空闲的，则在这个结点将垂直总线与水平总线接通，然后将该分组转发到这个输出端口。但若该垂直总线已被占用（有另一个分组正在转发到同一个输出端口），则后到达的分组就被阻塞，必须在输入端口排队。采用这种交换方式的路由器例子是 Cisco 公司的 12000 系列交换路由器，它使用的互连网络的带宽达 60 Gb/s。

6.1.3 互联网与因特网

互连在一起的网络要进行通信，会遇到许多问题需要解决，如：

- 不同的寻址方案；
- 不同的最大分组长度；
- 不同的网络接入机制；
- 不同的超时控制；
- 不同的差错恢复方法；
- 不同的状态报告方法；
- 不同的路由选择技术；
- 不同的用户接入控制；
- 不同的服务（面向连接服务和无连接服务）；
- 不同的管理与控制方式；等等。

从一般的概念来讲，将网络互相连接起来要使用一些**中间设备**（或**中间系统**），ISO 的术语称之为**中继**(relay)系统。根据中继系统所在的层次，可以有以下五种不同的中继系统：

（1）物理层中继系统，即**转发器**(repeater)。

（2）数据链路层中继系统，即**网桥**或**桥接器**(bridge)。

（3）网络层中继系统，即**路由器**(router)。

（4）网桥和路由器的混合物**桥路器**(brouter)。桥路器是一种兼有网桥和路由器的功能的产品。实际上，严格的网桥或严格的路由器产品是较少见的。不过此名词用得不普遍。

（5）在网络层以上的中继系统，即称为**网关**(gateway)。用网关连接两个不兼容的系统就要在高层进行协议的转换。

当中继系统是转发器或网桥时，一般并不称之为网络互连，因为这仅仅是把一个网络扩大了，而这仍然是一个网络。网关由于比较复杂，目前使用得较少。因此一般讨论互联网时都是指用路由器进行互连的互联网络。前面已经说过，路由器其实就是一台专用计算机，用来在互联网中进行路由选择。**由于历史的原因，许多有关 TCP/IP 的文献将网络层使用的路由器称为网关**（在本书中，有时也这样用）。对此请读者加以注意。

用网关进行网络互连为什么会很复杂呢？可以设想有 N 个网络要进行互连。每两个网络之间需要一个协议转换器。N 个网络共需 $N(N-1)$ 个协议转换器。（考虑到每两个网络之间就经过一个网关）。因此当网络数 N 很大时，所需的协议转换器的数量与网络数 N 的平方成正比，因此需要设计出非常多的协议转换器才行。

因特网在 IP 层采用了标准化协议。图 6-6(a)表示有许多计算机网络通过一些路由器进行互连。由于参加互连的计算机网络都使用相同的网际协议 IP (Internet Protocol)，因此可以将互连以后的计算机网络看成如图 6-6(b)所示的一个**虚拟互连网络**(internet)。所谓虚拟互连网络也就是逻辑互连网络，它的意思就是互连起来的各种物理网络的异构性本来是客观存在的，但是我们利用 IP 协议就可以使这些性能各异的网络从用**户看起来好像是一个统一的网络**。这种使用 IP 协议的虚拟互连网络可简称为 **IP 网**。使用虚拟互连网络的好处是：当互联网上的主机进行通信时，就好像在一个网络上通信一样，它们看不见互连的各具体的网络的异构细节（如具体的编址方案、路由选择协议等等）。

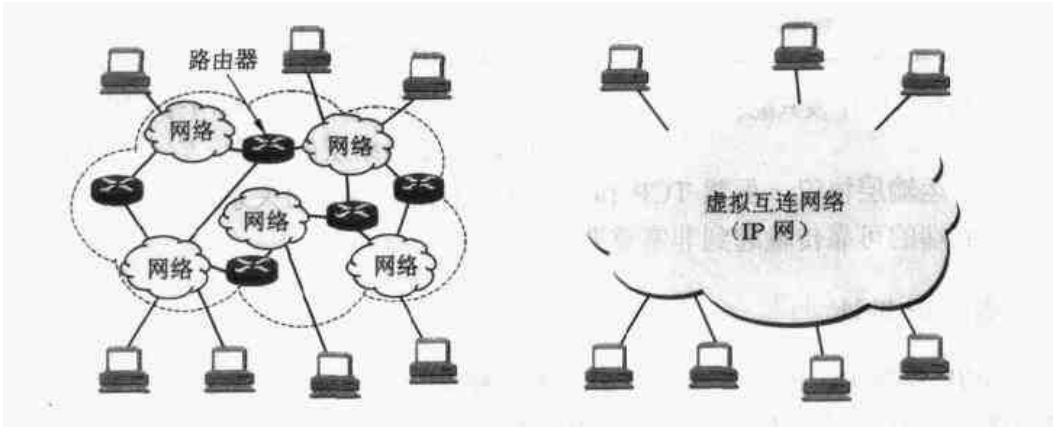


图 6-6　互连网络的概念(a)实际的互连网络；(b)虚拟互连网络(IP 网）

请读者注意以下两个意思不一样的名词 internet 和 Internet [RFC 1208]：

以小写字母 i 开始的 **internet（互联网或互连网）是一个通用名词，它泛指由多个计算机网络互连而成的虚拟网络。**

以大写字母 I 开始的的 **Internet (因特网)则是一个专用名词，它指当前全球最大的、开放的、由众多网络相互连接而成的特定计算机网络，它采用 TCP/IP 协议族，且其前身是美国的 ARPANET**。在讨论到因特网时，有时是指虚拟网络，有时也指具体的物理网络。

当然，在互联网中使用网际协议 IP 进行互连并非是惟一的一种方法。例如，当多个 X.25 分组交换网进行互连时，只要使互连的网络接口符合 ITU-T 的 X.75 建议书即可。然而目前全世界使用 IP 协议的因特网用户却远远超过了使用 X.25 和 X.75 的用户。

从下一节起开始讨论因特网的核心协议，即**网际协议 IP**。

6.2 因特网的网际协议 IP

网际协议 IP 是 TCP/IP 体系中两个最主要的协议之一[STEV94][COME00][FORO00]，也是最重要的因特网标准协议[RFC 791]之一。与 IP 协议配套使用的还有四个协议：

- **地址解析协议 ARP** (Address Resolution Protocol)
- **逆地址解析协议 RARP** (Reverse Address Resolution Protocol)
- **因特网控制报文协议 ICMP** (Internet Control Message Protocol)
- **因特网组管理协议 IGMP** (Internet Group Management Protocol)

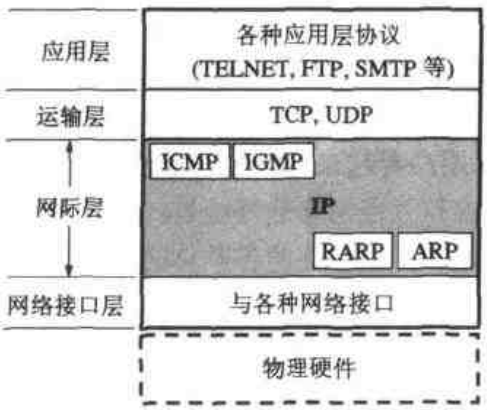


图 6-7 网际协议 IP 及其配套协议

图 6-7 画出了这四个协议和网际协议 IP 的关系。在这一层中，ARP 和 RARP 画在最下面，因为 IP 经常要使用这两个协议。ICMP 和 IGMP 画在这一层的上部，因为它们要使用 IP 协议。这四个协议将在后面陆续介绍。由于网际协议 IP 是用来使互连起来的许多计算机网络能够进行通信，因此 TCP/IP 体系中的网络层常常称为**网际层** (internet layer)，或 **IP 层**。

顺便指出，有时会听到这样的说法："我们用 TCP/IP 协议进行网络互连"。请读者注意，IP 协议是负责网络互连的网络层的核心协议，而 TCP 不是网络层的协议，它是与网际协议 IP 配套使用的一个运输层协议。虽然 TCP 和网络互连并没有直接的关系，但 TCP 协议对保证互连网络中端到端的可靠传输起到非常重要的作用。

6.2.1 分类的 IP 地址

在 TCP/IP 体系中，IP 地址是一个最基本的概念，一定要把它弄清楚。有关 IP 最重要的文档就是[RFC 791]，它很早就成为了因特网的正式标准。

1．IP 地址及其表示方法

我们把整个因特网看成为一个**单一的、抽象的网络**。IP 地址就是给每个连接在因特网上的主机（或路由器）分配一个在全世界范围是惟一的 32 bit 的标识符。IP 地址的结构使我们可以在因特网上很方便地进行寻址。IP 地址现在由**因特网名字与号码指派公司 ICANN** (Internet Corporation for Assigned Names and Numbers)进行分配[①]。

IP 地址的编址方法共经过了三个历史阶段。这三个阶段是：

（1）**分类的 IP 地址**。这是最基本的编址方法，在 1981 年就通过了相应的标准协议。

（2）**子网的划分**。这是对最基本的编址方法的改进，其标准[RFC 950]在 1985 年通过。

（3）**构成超网**。这是比较新的无分类编址方法。1993 年提出后很快就得到推广应用。

① 注：我国用户可向亚太网络信息中心 APNIC (Asia Pacific Network Information Center)申请 IP 地址（要缴费）。

本节只讨论最基本的分类 IP 地址。后两种方法将在 6.3 节中讨论。

所谓“分类的 IP 地址”就是将 IP 地址划分为若干个固定类，每一类地址都由两个固定长度的字段组成，其中一个字段是**网络号 net-id**，它标志主机（或路由器）所连接到的网络，而另一个字段则是**主机号 host-id**，它标志该主机（或路由器）。或者说，这种两级的 IP 地址可以记为：

$$\text{IP 地址} ::= \{ <\text{网络号}>, <\text{主机号}> \} \qquad (6\text{-}1)$$

图 6-8 给出了各种 IP 地址的网络号字段和主机号字段，这里 A 类、B 类和 C 类地址是最常用的。

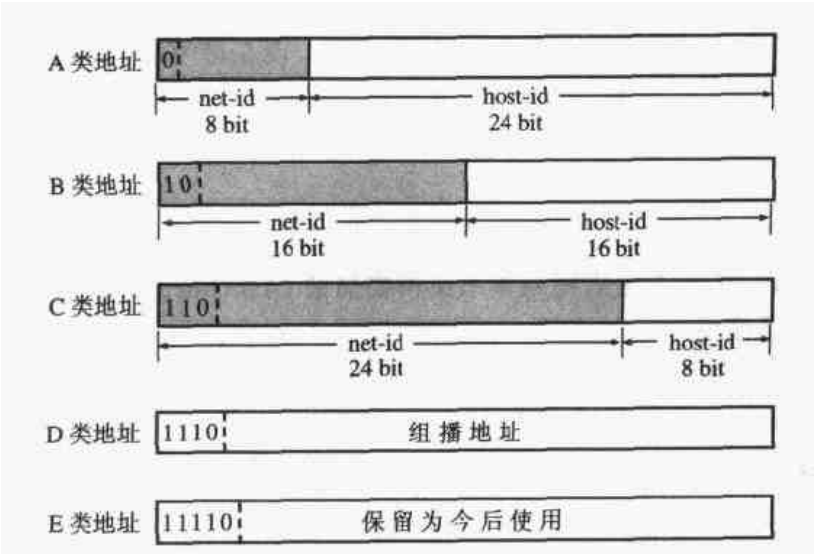


图 6-8　IP 地址中的网络号字段和主机号字段

从图 6-8 可以看出：

- A 类、B 类和 C 类地址的网络号字段 net-id(在图中这个字段是灰色的)分别为 1，2 和 3 字节长，而在网络号字段的最前面有 1～3 bit 的类别比特，其数值分别规定为 0，10 和 110。
- A 类、B 类和 C 类地址的主机号字段分别为 3 个、2 个和 1 个字节长。

这里要指出，由于近年来已经广泛使用不分类 IP 地址进行路由选择，A 类、B 类和 C 类地址的区分已成为历史[RFC 1812]，但由于很多文献和资料都还使用传统的分类 IP 地址，因此我们在这里还要从分类 IP 地址讲起。

路由器为转发分组而查找转发表时，很重要的一点就是查找路由表花费的时间要尽量短。在 5.2.2 节讨论广域网时已经讲过简化转发表的办法就是根据目的主机所连接的目的结点交换机的位置来寻找路由。当找到目的结点交换机后，就可直接交付给目的主机。在互连网中也是类似的。如果将到达所有目的主机的路由全都写入到转发表中，就会有很多路由是重复的，因此没有必要这样做。我们应将重复的路由合并。转发表只使用 IP 地址中的网络号 net-id 来查找路由。只要 IP 数据报能够正确到达目的网络，就可在这个网络上直接交付给目的主机而不再需要经过其他路由器进行转发了。

因此，路由器转发分组的步骤是：

（1）先按所要找的 IP 地址中的网络号 net-id 把目的网络找到。虽然 IP 地址的网络字段有三种不同的长度，但根据 IP 地址中最前面的类别比特，就可以很容易地确定网络字段的准

确字节数。然而以后我们会知道还有更加简便的方法。

（2）当分组到达目的网络后，再利用主机号 host-id 将数据报直接交付给目的主机。

因此，按照整数字节划分 net-id 字段和 host-id 字段，就可以使路由器在收到一个分组时能够更快地将地址中的网络号提取出来。

从 IP 地址的结构来看，IP 地址并不仅仅是一个主机的号，而是指出了**连接到某个网络上的某个主机**。如果一个主机的地理位置保持不变，但现在只改变连接的线路，即连接到另外一个网络，那么这个主机的 IP 地址就必须改变。

将 IP 地址划分为三个类别，当初是这样考虑的。各种网络的差异很大，有的网络拥有很多主机，而有的网络上的主机则很少。将 IP 地址划分为 A 类、B 类和 C 类可更好地满足不同用户的要求。A 类 IP 地址的网络号数不多。现在能够申请到的 IP 地址只有 B 类和 C 类两种。当某个单位申请到一个 IP 地址时，实际上只是获得了一个网络号 net-id。具体的各个主机号 host-id 则由该单位自行分配，只要做到在该单位管辖的范围内无重复的主机号即可。

除上述的三类 IP 地址外，还有两类使用得较少的地址，即 D 类和 E 类地址（图 6-8）。D 类地址是多播地址，主要留给**因特网体系结构委员会 IAB** (Internet Architecture Board)使用。E 类地址保留在今后使用。

在主机或路由器中存放的 IP 地址都是 32 bit 的二进制代码。为了提高可读性，在写出给人看的 IP 地址时，往往每隔 8 bit 插入一个空格。但这样还是不方便。于是我们常常将 32 bit 的 IP 地址中的每 8 bit 用其等效的十进制数字表示，并且在这些数字之间加上一个点。这就叫做**点分十进制记法**(dotted decimal notation)。图 6-9 表示了这种方法，这是一个 B 类 IP 地址。显然，128.11.3.31 比 10000000 00001011 00000011 00011111 读起来要方便得多。

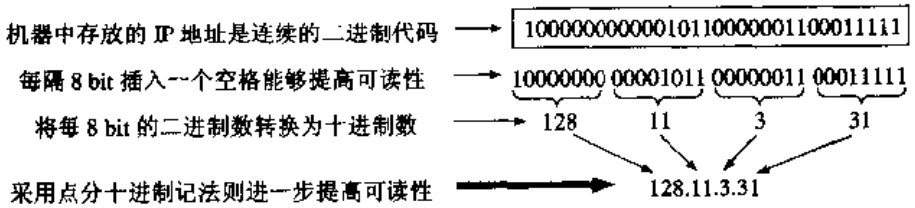


图 6-9　采用点分十进制记法能够提高可读性

2．常用的三种类别的 IP 地址

下面我们再对 A 类、B 类和 C 类地址进行较深入的讨论。

A 类地址的 net-id 字段占一个字节，只有 7 个比特可供使用（该字段的第一个比特已固定为 0），但可提供使用的网络号是 126 个(即 $2^7 - 2$)。减 2 的原因是：第一，IP 地址中的全 0 表示“**这个**(this)”。net-id 字段为全 0 的 IP 地址是个保留地址，意思是“**本网络**”。第二，net-id 字段为 127(即 01111111)保留作为本地软件**环回测试**(loopback test)本主机之用（后面三个字节的二进制数字可任意填入，但不能都是 0 或都是 1，即除了 127.0.0.0 和 127.255.255.255 以外都可以用）。A 类地址的 host-id 字段为 3 个字节，因此每一个 A 类网络中的最大主机数是 16 777 214 (即 $2^{24}-2$)。这里减 2 的原因是：全 0 的 host-id 字段表示该 IP 地址是“**本主机**”所连接到的**单个网络地址**（例如，一主机的 IP 地址为 5.6.7.8，则该主机所在的网络地址就是 5.0.0.0），而全

1 表示“**所有的**(all)”，因此全 1 的 host-id 字段表示该网络上的所有主机[①]。

整个 A 类地址空间共有 2^{31} (即 2147483648)个地址，而 IP 地址全部的地址空间共有 2^{32} (即 4294967296)个地址。可见 A 类地址占有整个 IP 地址空间的 50%。

B 类地址的 net-id 字段有 2 字节，但前面两个比特(1 0)已经固定了，只剩下 14 个比特可以变化，因此 B 类地址的网络数为 16 384 (即 2^{14})。请注意，这里不存在减 2 的问题，因为 net-id 字段最前面的两个比特(1 0)使得后面的 14 个比特无论怎样排列也不可能出现使整个 2 字节的 net-id 字段成为全 0 或全 1。B 类地址的每一个网络上的最大主机数是 65 534 (即 $2^{16}-2$)。这里需要减 2 是因为要扣除全 0 和全 1 的主机号。整个 B 类地址空间共有 1 073 741 824 (即 2^{30})个地址，占整个 IP 地址空间的 25%。

C 类地址有 3 个字节的 net-id 字段，最前面的 3 个比特是(1 1 0)，还有 21 个比特可以变化，因此 C 类地址的网络总数是 2 097 152 (即 2^{21}，这里也不需要减 2)。每一个 C 类地址的最大主机数是 254 (即 2^8-2)。整个 C 类地址空间共有 536870912 (即 2^{29})个地址，占整个 IP 地址的 12.5%。

这样，我们就可得出表 6-1 所示的 IP 地址的使用范围。

表 6-1　IP 地址的使用范围

网络类别	最大网络数	第一个可用的网络号	最后一个可用的网络号	每个网络中的最大主机数
A	126 (2^7-2)	1	126	16 777 214
B	16 384 (2^{14})	128.0	191.255	65 534
C	2 097 152 (2^{21})	192.0.0	223.255.255	254

表 6-2 给出了一般不使用的 IP 地址，这些地址只能在特定的情况下使用。

表 6-2　一般不使用的特殊 IP 地址

net-id	host-id	源地址使用	目的地址使用	代表的意思
0	0	可以	不可	在本网络上的本主机
0	host-id	可以	不可	在本网络上的某个主机
全 1	全 1	不可	可以	只在本网络上进行广播（各路由器均不转发）
net-id	全 1	不可	可以	对 net-id 上的所有主机进行广播
127	任何数	可以	可以	用作本地软件环回测试之用

IP 地址具有以下一些重要特点：

（1）每一个 IP 地址都由网络号和主机号两部分组成。从这个意义上说，IP 地址是一种分等级的地址结构。分两个等级的好处是：第一，IP 地址管理机构在分配 IP 地址时只分配网络号（第一级），而剩下的主机号（第二级）则由得到该网络号的单位自行分配。这样就方便了 IP 地址的管理。第二，路由器仅根据目的主机所连接的网络号来转发分组（而不考虑目的主机号），这样就可以使路由表中的项目数大幅度减少，从而减小了路由表所占的存储空间。

（2）实际上 IP 地址是标志一个主机（或路由器）和一条链路的接口。当一个主机同时连接到两个网络上时，该主机就必须同时具有两个相应的 IP 地址，其网络号 net-id 必须是不

① 注：关于全 1 和全 0 还可以再举两个例子。例如，B 类地址 128.7.255.255 表示“在网络 128.7 上的所有主机”。而 A 类地址 0.0.0.35 则表示“在这个网络上的主机 35”。

同的。这种主机称为**多接口主机**(multihomed host)。由于一个路由器至少应当连接到两个网络，因此一个路由器至少应当有两个不同的 IP 地址。

（3）按照因特网的观点，**用转发器或网桥连接起来的若干个局域网仍为一个网络**，因此这些局域网都具有同样的网络号 net-id。

（4）在 IP 地址中，所有分配到网络号 net-id 的网络（不管是第 4 章讨论的范围很小的局域网，还是第 5 章讨论的可能覆盖很大地理范围的广域网）都是平等的。

图 6-10 画出了 3 个局域网(LAN_1, LAN_2 和 LAN_3)通过 3 个路由器(R_1, R_2 和 R_3)互连起来所构成的一个互联网（此互联网用虚线圆角方框表示）。其中局域网 LAN_2 是由两个网段通过网桥 B 互连的。图中的小圆圈表示需要有一个 IP 地址。

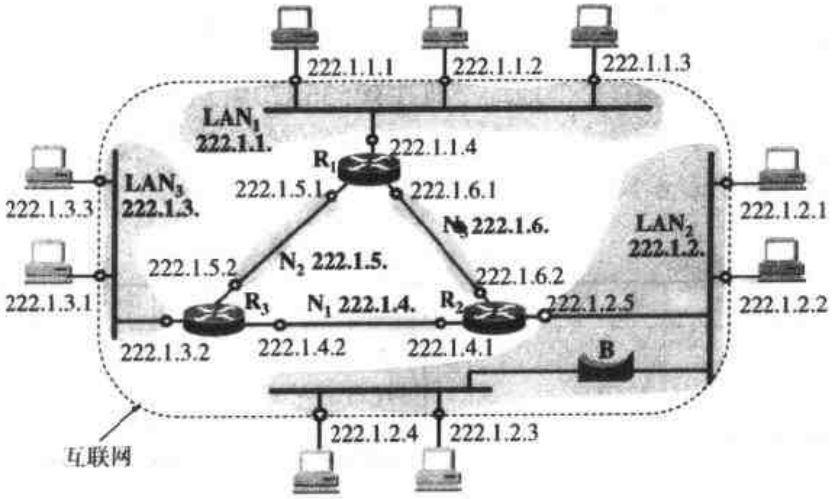


图 6-10　互联网中的 IP 地址

我们应当注意到：

（1）在同一个局域网上的主机或路由器的 IP 地址中的网络号必须是一样的。图中所示的网络号就是 IP 地址中的 net-id，这也是文献中常见的一种表示方法。另一种表示方法是用主机号 host-id 为全 0 的网络 IP 地址。

（2）用网桥（它只在链路层工作）互连的网段仍然是一个局域网，只能有一个网络号。

（3）路由器总是具有两个或两个以上的 IP 地址。即路由器的每一个接口都有一个不同网络号的 IP 地址。

（4）当两个路由器直接相连时，在连线两端的接口处，可以指明也可以不指明 IP 地址。如指明了 IP 地址，则这一段连线就构成了一种只包含一段线路的特殊“网络”（如图中的 N_1, N_2 和 N_3）。之所以叫做“网络”是因为它有 IP 地址。但为了节省 IP 地址资源，对于这种由一段连线构成的特殊“网络”，现在也常常不指明 IP 地址。

6.2.2　IP 地址与硬件地址

在学习 IP 地址时，很重要的一点就是要弄懂主机的 IP 地址与**硬件地址**[4]的区别。

---

[4] 注：在局域网中，由于硬件地址已固化在网卡上的 ROM 中，因此常常将硬件地址称为**物理地址**。因为在局域网的 MAC 帧中的源地址和目的地址都是硬件地址，因此硬件地址又称为 MAC 地址。在本书中，物理地址、硬件地址和 MAC 地址常常作为同义词。但应注意，有时，如在 X.25 网中，计算机的硬件地址并不是固化在 ROM 中的。

图 6-11 说明了这两种地址的区别。从层次的角度看，**物理地址是数据链路层和物理层使用的地址，而 IP 地址是网络层和以上各层使用的地址。**

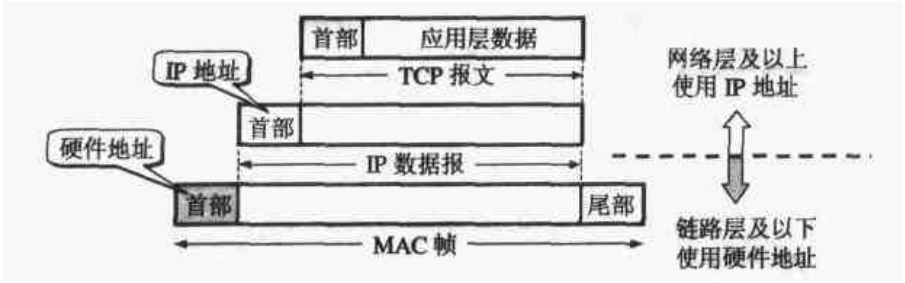


图 6-11 IP 地址与硬件地址的区别

在发送数据时，数据从高层下到低层，然后才到通信链路上传输。使用 IP 地址的 IP 数据报一旦交给了数据链路层，就被封装成 MAC 帧了。MAC 帧在传送时使用的源地址和目的地址都是硬件地址，这两个硬件地址都写在 MAC 帧的首部中。

连接在通信链路上的设备（主机或路由器）在接收 MAC 帧时，其根据是 MAC 帧首部中的硬件地址。在数据链路层看不见隐藏在 MAC 帧的数据中的 IP 地址。只有在剥去 MAC 帧的首部和尾部后将 MAC 层的数据上交给网络层后（这时 MAC 层的数据就变成了 IP 数据报），网络层才能在 IP 数据报的首部中找到源 IP 地址和目的 IP 地址。

总之，**IP 地址放在 IP 数据报的首部，而硬件地址则放在 MAC 帧的首部。在网络层和网络层以上使用的是 IP 地址，而数据链路层及以下使用的是硬件地址。**在图 6-11 中，当 IP 数据报放入数据链路层的 MAC 帧中以后，整个的 IP 数据报就成为 MAC 帧的数据，因而在**数据链路层看不见数据报的 IP 地址。**

图 6-12(a)画的是三个局域网用两个路由器 R_1 和 R_2 互连起来。现在主机 H_1 要和主机 H_2 通信。这两个主机的 IP 地址分别是 IP_1 和 IP_2，而它们硬件地址分别为 HA_1 和 HA_2 (HA 表示 Hardware Address)。通信的路径是：H_1→经过 R_1 转发→再经过 R_2 转发→H_2。路由器 R_1 因同时连接到两个局域网上，因此它有两个硬件地址，即 HA_3 和 HA_4。同理，路由器 R_2 也有两个硬件地址 HA_5 和 HA_6。

图 6-12(b)特别强调了 IP 地址与硬件地址的区别。表 6-3 归纳了这种区别。

这里要强调指出的是：

（1）**在 IP 层抽象的互联网上只能看到 IP 数据报。**虽然 IP 数据报要经过路由器 R_1 和 R_2 的两次转发，但在它的首部中的源地址和目的地址始终分别是 IP_1 和 IP_2。图中的数据报上写的“从 IP_1 到 IP_2”就表示前者是源地址而后者是目的地址。数据报中间经过的两个路由器的 IP 地址并不出现在 IP 数据报的首部中。

（2）虽然在 IP 数据报首部有源站 IP 地址，但**路由器只根据目的站的 IP 地址的网络号进行路由选择。**

（3）在**具体的物理网络的链路层，只能看见 MAC 帧**（在 X.25 网的链路层则是 HDLC 帧）。IP 数据报被封装在 MAC 帧中。MAC 帧在不同网络上传送时，其 MAC 帧首部中的源地址和目的地址要发生变化。开始在 H_1 到 R_1 间传送时，MAC 帧首部中写的是从硬件地址 HA_1 发送到硬件地址 HA_3，路由器 R_1 收到此 MAC 帧后，在转发时要改变首部中的源地址和目的地址，将它们换成从硬件地址 HA_4 发送到硬件地址 HA_5。路由器 R_2 收到此帧后，再改变一次 MAC 帧的首部，填入从 HA_6 发送到 HA_2，然后在 R_2 到 H_2 之间传送。MAC 帧的首部

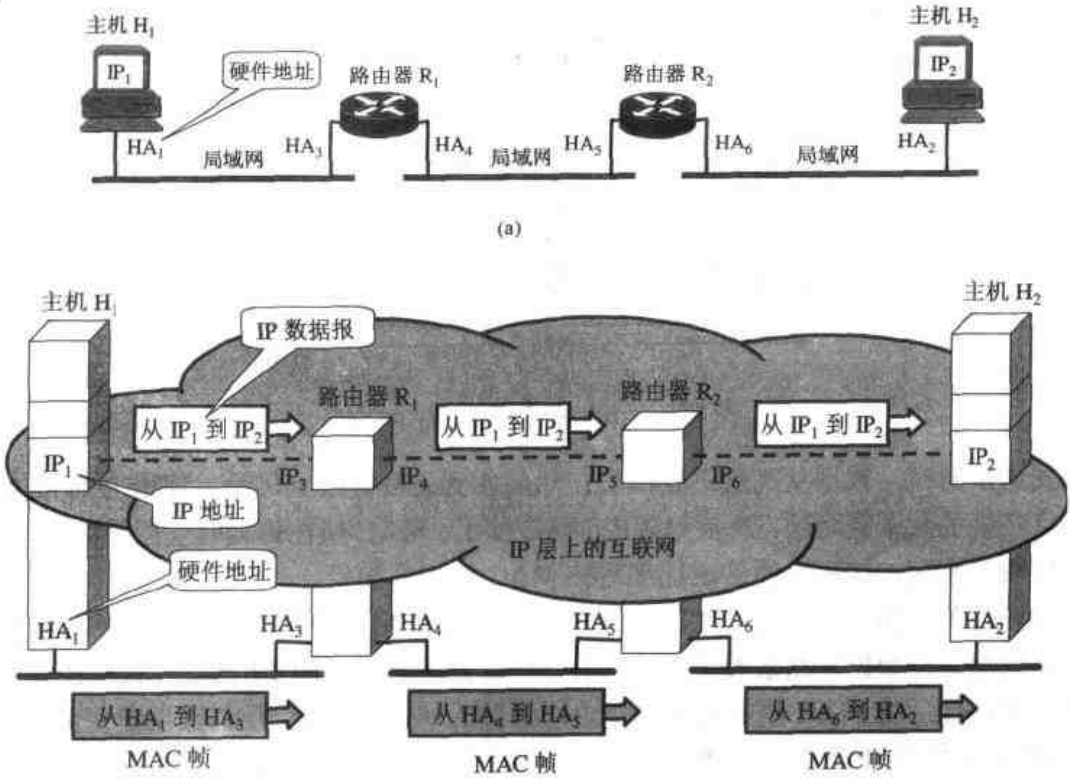


图 6-12　从不同层次上看 IP 地址和硬件地址：

(a)网络配置；(b)不同层次、不同区间的源地址和目的地址

表 6-3　图 6-12(b)中不同层次、不同区间的源地址和目的地址

	在网络层 写入 IP 数据报首部的		在数据链路层 写入 MAC 帧首部的	
	源地址	目的地址	源地址	目的地址
从 H_1 到 R_1	IP_1	IP_2	HA_1	HA_3
从 R_1 到 R_2	IP_1	IP_2	HA_4	HA_5
从 R_2 到 H_2	IP_1	IP_2	HA_6	HA_2

的这种变化，在上面的 IP 层上也是看不见的。

（4）尽管互连在一起的网络的硬件地址体系各不相同，**但 IP 层抽象的互联网却屏蔽了下层这些很复杂的细节。只要我们在网络层上讨论问题，就能够使用统一的、抽象的 IP 地址研究主机和主机或路由器之间的通信。**上述的这种“屏蔽”概念是一个很有用、很普遍的基本概念。例如，计算机中广泛使用的图形用户界面使得用户只需简单地点击几下鼠标就能让电脑完成很多任务。实际上电脑要完成这些任务必须运行很多的程序。但这些复杂的过程全都被设计良好的图形用户界面屏蔽掉了，使用户看不见这些复杂过程。

以上这些概念是计算机网络的精髓所在，对这些重要概念务必仔细思考和掌握。

细心的读者会发现，还有两个重要问题还没有解决：

（1）主机或路由器怎样知道应当在 MAC 帧的首部填入什么样的硬件地址？

（2）路由器中的路由表是怎样得出的？

第一个问题就是下一节所要讲的内容，而第二个问题将在 6.5 节详细讨论。

6.2.3 地址解析协议 ARP 和逆地址解析协议 RARP

上面讲的 IP 地址是不能直接用来进行通信的。这是因为 IP 地址只是主机在抽象的网络层中的地址。若要将网络层中传送的数据报交给目的主机，还要传到链路层转变成 MAC 帧后才能发送到实际的网络上。因此，不管网络层使用的是什么协议，在实际网络的链路上传送数据帧时，最终还是必须使用硬件地址。

由于 IP 地址有 32 bit，而局域网的硬件地址是 48 bit，因此它们之间不存在简单的映射关系。此外，在一个网络上可能经常会有新的主机加入进来，或撤走一些主机。更换网卡也会使主机的硬件地址改变。可见在主机中应存放一个从 IP 地址到硬件地址的映射表，并且这个映射表还必须能够经常动态更新。**地址解析协议** ARP 很好地解决了这些问题。

每一个主机都设有一个 ARP **高速缓存**(ARP cache)，里面有所在的局域网上的各主机和路由器的 IP 地址到硬件地址的映射表，这些都是该主机目前知道的一些地址。那么主机怎样知道这些地址呢？我们可以通过下面的例子来说明。当主机 A 欲向本局域网上的某个主机 B 发送 IP 数据报时，就先在其 ARP 高速缓存中查看有无主机 B 的 IP 地址。如有，就可查出其对应的硬件地址，再将此硬件地址写入 MAC 帧，然后通过局域网将该 MAC 帧发往此硬件地址。

也有可能查不到主机 B 的 IP 地址的项目。这可能是主机 B 才入网，也可能是主机 A 刚刚加电，其高速缓存还是空的。在这种情况下，主机 A 就自动运行 ARP，然后按以下步骤找出主机 B 的硬件地址。

① ARP 进程在本局域网上广播发送一个 ARP 请求分组（具体格式从略）。图 6-13(a)是主机 A 广播发送 ARP 请求分组的示意图。ARP 请求分组的主要内容是表明："我的 IP 地址是 209.0.0.5，硬件地址是 00-00-C0-15-AD-18。我想知道 IP 地址为 209.0.0.6 的主机的硬件地址。"

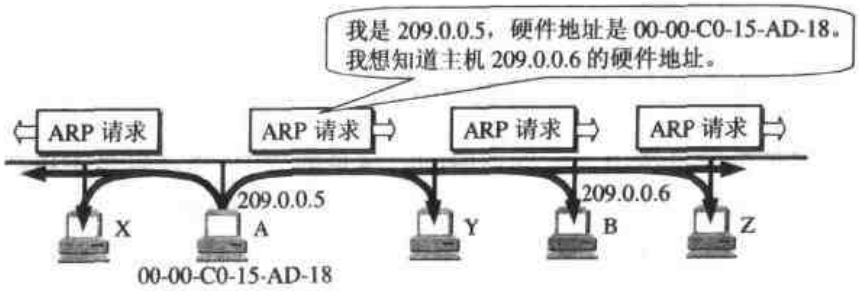
② 在本局域网上的所有主机上运行的 ARP 进程都收到此 ARP 请求分组。

③ 主机 B 在 ARP 请求分组中见到自己的 IP 地址，就向主机 A 发送 ARP 响应分组（其格式从略），并写入自己的硬件地址。其余的所有主机都不理睬这个 ARP 请求分组，见图 6-13(b)。ARP 响应分组的主要内容是表明："我的 IP 地址是 209.0.0.6，我的硬件地址是 08-00-2B-00-EE-0A。"请注意：虽然 ARP 请求分组是广播发送的，但 ARP 响应分组是普通的单播，即从一个源地址发送到一个目的地址。

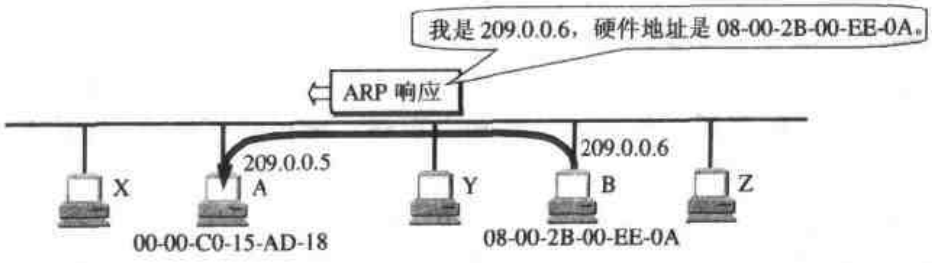
④ 主机 A 收到主机 B 的 ARP 响应分组后，就在其 ARP 高速缓存中写入主机 B 的 IP 地址到硬件地址的映射。

当主机 A 向 B 发送数据报时，很可能以后不久主机 B 还要向 A 发送数据报，因而主机 B 也可能要向 A 发送 ARP 请求分组。为了减少网络上的通信量，主机 A 在发送其 ARP 请求分组时，就将自己的 IP 地址到硬件地址的映射写入 ARP 请求分组。当主机 B 收到 A 的 ARP 请求分组时，就将主机 A 的这一地址映射写入主机 B 自己的 ARP 高速缓存中。这对主机 B 以后向 A 发送数据报时就更方便了。

可见 ARP 高速缓存非常有用。如果不使用 ARP 高速缓存，那么任何一个主机只要进行一次通信，就必须在网络上用广播方式发送 ARP 请求分组，这就使网络上的通信量大大增加。



(a) 主机 A 发送 ARP 请求分组



(b) 主机 B 向 A 发送 ARP 响应分组

图 6-13　地址解析协议 ARP 的工作原理：

ARP 将已经得到的地址映射保存在高速缓存中，这样就使得该主机下次再和具有同样目的地址的主机通信时，可以直接从高速缓存中找到所需的硬件地址而不必再用广播方式发送 ARP 请求分组。

ARP 将保存在高速缓存中的每一个映射地址项目都设置**生存时间**（例如，10～20 分钟）。凡超过生存时间的项目就从高速缓存中删除掉。设置这种地址映射项目的生存时间是很重要的。设想有一种情况。主机 A 和 B 通信。A 的 ARP 高速缓存里保存有 B 的物理地址。但 B 的网卡突然坏了，B 立即更换了一块，因此 B 的硬件地址就改变了。A 还要和 B 继续通信。A 在其 ARP 高速缓存中查找到 B 原先的硬件地址，并使用该硬件地址向 B 发送数据帧。但 B 原先的硬件地址已经失效了，因此 A 无法找到主机 B。但是过了一段时间，A 的 ARP 高速缓存中已经删除了 B 原先的硬件地址（因为它的生存时间到了），于是 A 重新广播发送 ARP 请求分组，又找到了 B。

这里需要指出，ARP 是解决**同一个局域网上**的主机或路由器的 IP 地址和硬件地址的映射问题。如果所要找的主机和源主机不在同一个局域网上，例如，图 6-12 所示的主机 H_1 和 H_2，那么主机 H_1 就无法解析出主机 H_2 的硬件地址（实际上主机 H_1 也不需要知道远程主机 H_2 的硬件地址）。主机 H_1 发送给 H_2 的 IP 数据报需要通过与主机 H_1 连接在同一个局域网上的路由器 R_1 来转发。因此主机 H_1 这时需要的是将路由器 R_1 的 IP 地址 IP_3 解析为硬件地址 HA_3，以便能够将 IP 数据报传送到转发该数据报的路由器 R_1。以后，R_1 从路由表找出了下一跳路由器 R_2，同时使用 ARP 解析出 R_2 的硬件地址 HA_5。于是 IP 数据报按照硬件地址 HA_5 转发到路由器 R_2。路由器 R_2 在转发这个 IP 数据报时用类似方法解析出目的主机 H_2 的硬件地址 HA_2，使 IP 数据报最终交付给主机 H_2。

这里我们还要指出，这种从 IP 地址到硬件地址的解析是自动进行的，**主机的用户对这种地址解析过程是不知道的**。只要主机或路由器要和本网络上的另一个已知 IP 地址的主机或路由器进行通信，ARP 协议就会自动地将该 IP 地址解析为链路层所需要的硬件地址。

有的读者可能会产生这样的问题：既然在网络链路上传送的帧最终是按照硬件地址找到目的主机的，那么为什么我们不直接使用硬件地址进行通信，而是要使用抽象的 IP 地址并调用 ARP 来寻找出相应的硬件地址呢？

这个问题必须弄清楚。

由于全世界存在着各式各样的网络，它们使用不同的硬件地址。要使这些异构网络能够互相通信就必须进行非常复杂的硬件地址转换工作，因此几乎是不可能的事。但统一的 IP 地址把这个复杂问题解决了。连接到因特网的主机都拥有统一的 IP 地址，它们之间的通信就像连接在同一个网络上那样简单方便，因为调用 ARP 来寻找某个路由器或主机的硬件地址都是由计算机软件自动进行的，对用户来说是看不见这种调用过程的。

设想有两个主机可以直接使用硬件地址进行通信（具体实现方法暂不必管）。再假定其两个主机的网卡都同时坏了，然后又都更换了一块，因此它们的硬件地址也都改变了。这时，这两个主机怎样能够知道对方的硬件地址呢？显然很难。但 IP 地址是独立于主机或路由器的硬件地址的。硬件地址的改变不会影响使用 IP 协议的主机的通信。

因此，在虚拟的 IP 网络上用 IP 地址进行通信给广大的计算机用户带来很大的方便。

在进行地址转换时，有时还要用到逆地址解析协议 RARP。逆地址解析协议 RARP 使只知道自己硬件地址的主机能够知道其 IP 地址。这种主机往往是无盘工作站。这种无盘工作站一般只要运行其 ROM 中的文件传送代码，就可用下行装载方法从局域网上其他主机得到所需的操作系统和 TCP/IP 通信软件，但这些软件中并没有 IP 地址。无盘工作站要运行 ROM 中的 RARP 来获得其 IP 地址。RARP 的工作过程大致如下。

为了使 RARP 能工作，在局域网上至少有一个主机要充当 RARP 服务器，无盘工作站先向局域网发出 RARP 请求分组（在格式上与 ARP 请求分组相似），并在此分组中给出自己的硬件地址。

RARP 服务器有一个事先做好的从无盘工作站的硬件地址到 IP 地址的映射表，当收到 RARP 请求分组后，RARP 服务器就从这映射表查出该无盘工作站的 IP 地址。然后写入 RARP 响应分组，发回给无盘工作站。无盘工作站用此方法获得自己的 IP 地址。

ARP 和 RARP 都已经成为因特网标准协议，其 RFC 文档分别为[RFC 826]和[RFC 903]。

6.2.4 IP 数据报的格式

IP 数据报的格式能够说明 IP 协议都具有什么功能。在 TCP/IP 的标准中，各种数据格式常常以 32 bit（即 4 字节）为单位来描述。图 6-14 是 IP 数据报的完整格式。

从图 6-14 可看出，一个 IP 数据报由首部和数据两部分组成。首部的前一部分是固定长度，共 20 字节，是所有 IP 数据报必须具有的。在首部的固定部分的后面是一些可选字段，其长度是可变的。下面介绍首部各字段的意义。

1．IP 数据报首部的固定部分中的各字段

（1）**版本**　占 4 bit，指 IP 协议的版本。通信双方使用的 IP 协议的版本必须一致。目前广泛使用的 IP 协议版本号为 4（即 IPv4）。以前的 3 个版本目前已不使用。

（2）**首部长度**　占 4 bit，可表示的最大数值是 15 个单位（一个单位为 4 字节），因此 IP 的首部长度的最大值是 60 字节。当 IP 分组的首部长度不是 4 字节的整数倍时，必须利用最后一个填充字段加以填充。因此数据部分永远在 4 字节的整数倍时开始，这样在实现 IP 协

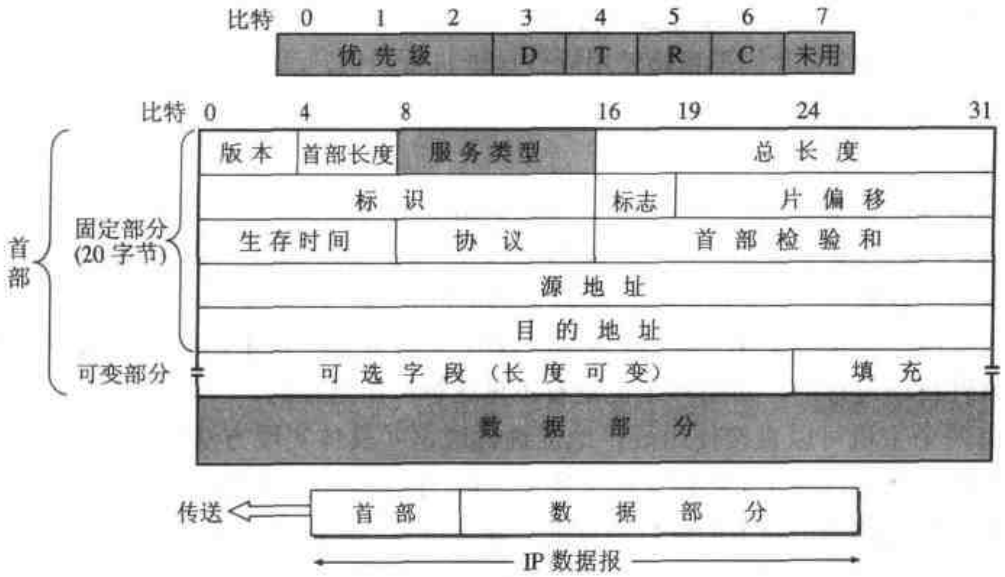


图 6-14　IP 数据报的格式

议时较为方便。首部长度限制为 60 字节的缺点是有时（如源站路由选择）不够用。但这样做是希望用户尽量减少开销。最常用的首部长度就是 20 字节，即不使用任何选项。

（3）**服务类型**　　占 8 bit，用来获得更好的服务，其意义见图 6-14 的上面部分所示。

- 前三个比特表示**优先级**，它可使数据报具有 8 个优先级中的一个。
- 第 4 个比特是 D 比特，表示要求有更低的时延。
- 第 5 个比特是 T 比特，表示要求有更高的吞吐量。
- 第 6 个比特是 R 比特，表示要求有更高的可靠性（即在数据报传送的过程中，被路由器丢弃的概率要更小些）。
- 第 7 个比特是 C 比特，是新增加的，表示要求选择代价更小的路由。
- 最后一个比特目前尚未使用。

在相当长一段时期内并没有什么人使用**服务类型 TOS** (Type Of Service)字段。直到最近，当需要将实时多媒体信息在因特网上传送时，服务类型字段才重新引起大家的重视。

（4）**总长度**　　总长度指首部和数据之和的长度，单位为字节。总长度字段为 16 bit，因此数据报的最大长度为 65 535 字节(即 64 KB)。

在 IP 层下面的每一种数据链路层都有其自己的帧格式，其中包括**帧格式中的数据字段的最大长度**，这称为**最大传送单元** MTU (Maximum Transfer Unit)。当一个 IP 数据报封装成链路层的帧时，此数据报的总长度（即首部加上数据部分）一定不能超过下面的数据链路层的 MTU 值。表 6-4 给出了不同链路层协议的 MTU 值。

表 6-4　不同链路层协议的 MTU 值

协　　议	MTU（字节）
Hyperchannel	65 535
令牌环(16 Mb/s)	17 914
令牌环(4 Mb/s)	4 464
FDDI	4 352
以太网	1 500

续表

协　议	MTU（字节）
X.25	576
PPP	296

虽然使用尽可能长的数据报会使传输效率提高，但由于以太网的普遍应用，所以实际上使用的数据报长度很少有超过 1500 字节的，而有时数据报长度还被限制在 576 字节。当数据报长度超过网络所容许的最大传送单元 MTU 时，就必须将过长的数据报进行分片后才能在网络上传送（见后面的“片偏移”字段）。这时，数据报首部中的“总长度”字段不是指未分片前的数据报长度，而是指分片后每片的首部长度与数据长度的总和。

（5）**标识**(identification)　占 16 bit，它是一个计数器，用来产生数据报的标识。但这里的“标识”并没有序号的意思，因为 IP 是无连接服务，数据报不存在按序接收的问题。当 IP 协议发送数据报时，它就将这个计数器的当前值复制到标识字段中。当数据报由于长度超过网络的 MTU 而必须分片时，这个标识字段的值就被复制到所有的数据报片的标识字段中。相同的标识字段的值使分片后的各数据报片最后能正确地重装成为原来的数据报。

（6）**标志**(flag)　占 3 bit。目前只有前两个比特有意义。

- 标志字段中的最低位记为 **MF** (More Fragment)。MF=1 即表示后面“**还有分片**”的数据报。MF = 0 表示这已是若干数据报片中的最后一个。
- 标志字段中间的一位记为 **DF** (Don't Fragment)，意思是“**不能分片**”。只有当 DF = 0 时才允许分片。

（7）**片偏移**　片偏移指出：较长的分组在分片后，某片在原分组中的相对位置。也就是说，相对于用户数据字段的起点，该片从何处开始。片偏移以 8 个字节为偏移单位。这就是说，每个分片的长度一定是 8 字节(64 bit)的整数倍。

下面举一个例子。

【例】 一数据报的数据部分为 3800 字节长（使用固定首部），需要分片为长度不超过 1420 字节的数据报片。因固定首部长度为 20 字节，因此每个数据报片的数据部分长度不能超过 1400 字节。于是分为 3 个数据报片，其数据部分的长度分别为 1400, 1400 和 1000 字节。原始数据报首部被复制为各数据报片的首部，但必须修改有关字段的值。图 6-15 表示分片的结果。表 6-5 是各数据报的首部中与分片有关的字段中的数值，其中标识字段的值是任意给定的。具有相同标识的数据报片在目的站就可无误地重装成原来的数据报。

现在假定数据报片 2 经过某个网络时还要再进行分片，即划分为数据报片 2-1（携带数据 800 字节）和数据报片 2-2（携带数据 600 字节）。那么这两个数据报片的总长度、标识、MF、DF 和片偏移分别为：820, 12345, 1, 0, 175；620, 12345, 1, 0, 275。

（8）**生存时间**　生存时间字段记为 TTL (Time To Live)，即数据报在网络中的寿命，其单位为秒。生存时间的建议值是 32 秒。但也可设定为 3～4 秒，甚至 255 秒。

（9）**协议**　占 8 bit，协议字段指出此数据报携带的数据是使用何种协议，以便使目的主机的 IP 层知道应将数据部分上交给哪个处理过程。图 6-16 表示 IP 层需要根据这个协议字段的值将所收到的数据交付到正确的地方。

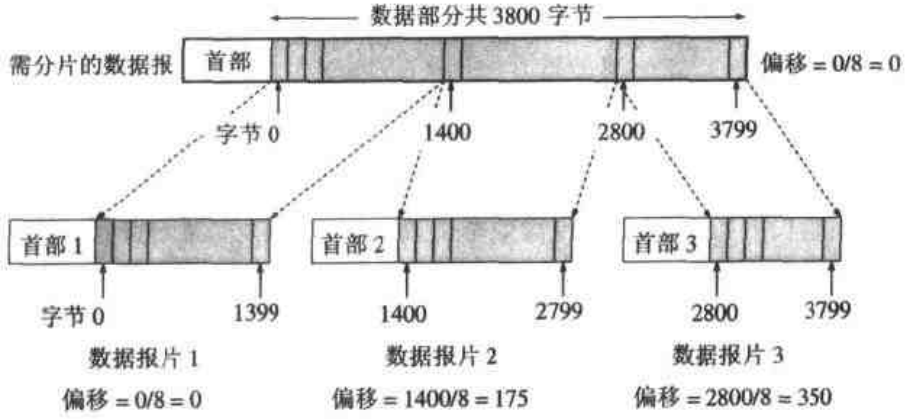


图 6-15　数据报的分片

表 6-5　IP 数据报首部中与分片有关的字段中的数值。

	总长度	标识	MF	DF	片偏移
原始数据报	4 000	12345	0	0	0
数据报片 1	1 420	12345	1	0	0
数据报片 2	1 420	12345	1	0	175
数据报片 3	1 020	12345	0	0	350

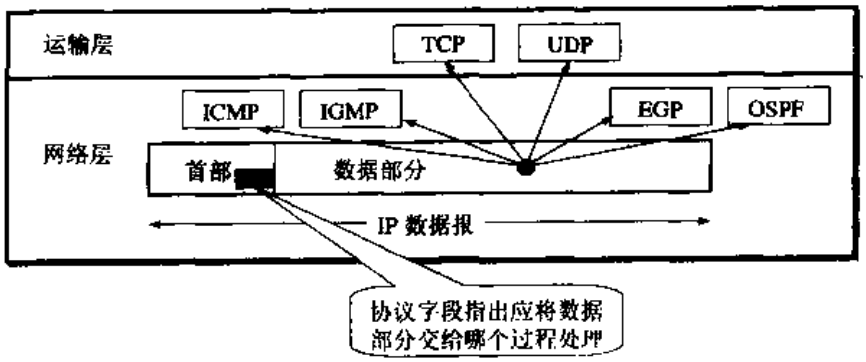


图 6-16　协议字段告诉 IP 层应当如何交付数据

常用的一些协议和相应的协议字段值如下：

协议名	ICMP	IGMP	TCP	EGP	IGP	UDP	IPv6	OSPF
协议字段值	1	2	6	8	9	17	41	89

（10）**首部检验和**　此字段只检验数据报的首部，不包括数据部分。这是因为数据报每经过一个结点，结点处理机都要重新计算一下首部检验和（一些字段，如生存时间、标志、片偏移等都可能发生变化）。如将数据部分一起检验，计算的工作量就太大了。

为了减小计算检验和的工作量，IP 首部的检验和不采用 CRC 检验码而采用下面的简单计算方法：在发送端，先将 IP 数据报首部划分为许多 16 bit 字的序列，并将检验和字段置零。用反码算术运算将所有 16 bit 字相加后，将得到的和的反码写入检验和字段。接收端收到数据报后，将首部的所有 16 bit 字再使用反码算术运算相加一次。将得到的和取反码，即得出接收端检验和的计算结果。若首部未发生任何变化，则此结果必为 0，于是就保留这个

数据报。否则即认为出差错，并将此数据报丢弃。图 6-17 说明了 IP 数据报首部检验和的计算过程。

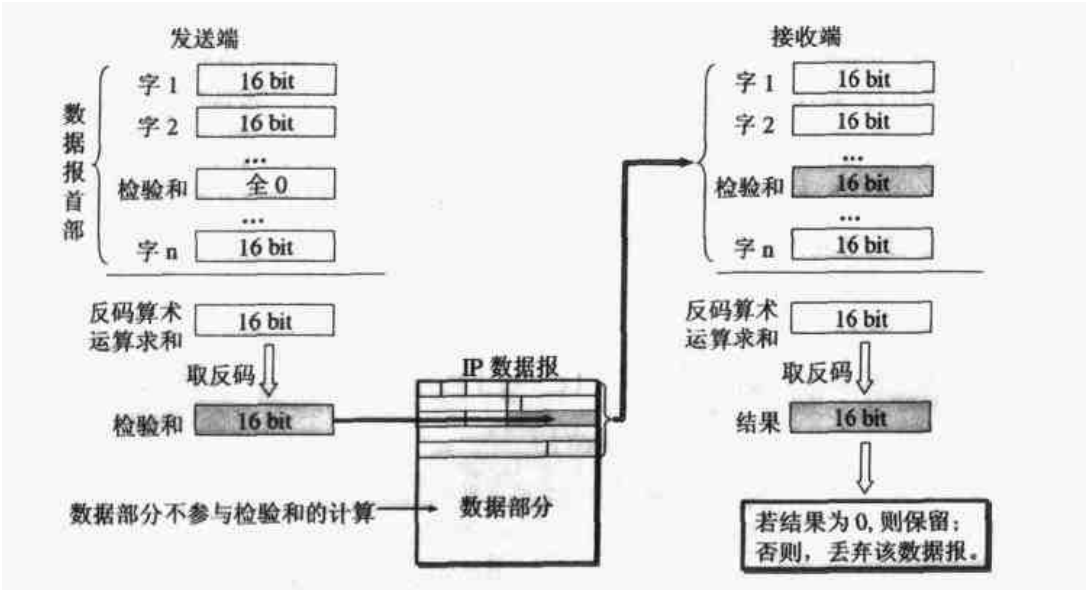


图 6-17　IP 数据报首部检验和的计算过程

（11）**源地址**　　占 4 字节。

（12）**目的地址**　　占 4 字节。

2．IP 数据报首部的可变部分

IP 首部的可变部分就是一个选项字段。选项字段用来支持排错、测量以及安全等措施，内容很丰富。此字段的长度可变，从 1 个字节到 40 个字节不等，取决于所选择的项目。某些选项项目只需要 1 个字节，它只包括 1 个字节的选项代码。但还有些选项需要多个字节，这些选项一个个拼接起来，中间不需要有分隔符，最后用全 0 的填充字段补齐成为 4 字节的整数倍。

增加首部的可变部分是为了增加 IP 数据报的功能，但这同时也使得 IP 数据报的首部长度成为可变的。这就增加了每一个路由器处理数据报的开销。实际上这些选项很少被使用。新的 IP 版本 IPv6 就将 IP 数据报的首部长度做成固定的。因此这里不再继续讨论这些选项的细节。有兴趣的读者可参阅[RFC 791]。

6.2.5　IP 层转发分组的流程

在因特网中路由器的作用和第 5 章广域网中的结点交换机非常相似。但路由器和结点交换机还有些区别。这就是：

（1）路由器是用来连接不同的网络，而结点交换机只是在一个特定的网络中工作。

（2）路由器是专门用来转发分组的，而结点交换机还可接上许多个主机。

（3）路由器使用统一的 IP 协议，而结点交换机使用所在广域网的特定协议。

（4）路由器根据目的网络地址找出下一跳（即下一个路由器），而结点交换机则根据目的站所接入的交换机号找出下一跳（即下一个结点交换机）。

图 6-18(a)是一个路由表的简单的例子。有 4 个 A 类网络通过 3 个路由器连接在一起。每

一个网络上都可能有成千上万个主机。可以想像，若按查找目的主机号来制作路由表，则所得出的路由表就会过于庞大。但若按主机所在的网络地址来制作路由表，那么每一个路由器中的路由表就只包含 4 个项目。以路由器 R_2 的路由表为例。由于 R_2 同时连接在网络 2 和网络 3 上，因此只要目的站在这两个网络上，都可通过接口 0 或 1 由路由器 R_2 直接交付（当然还要利用地址解析协议 ARP 才能找到这些主机相应的硬件地址）。若目的站在网络 1 中，则下一跳路由器应为 R_1，其 IP 地址为 20.0.0.7。路由器 R_2 和 R_1 由于同时连接在网络 2 上，因此从路由器 R_2 将分组转发到路由器 R_1 是很容易的。同理，若目的站在网络 4 中，则路由器 R_2 应将分组转发给 IP 地址为 30.0.0.1 的路由器 R_3。

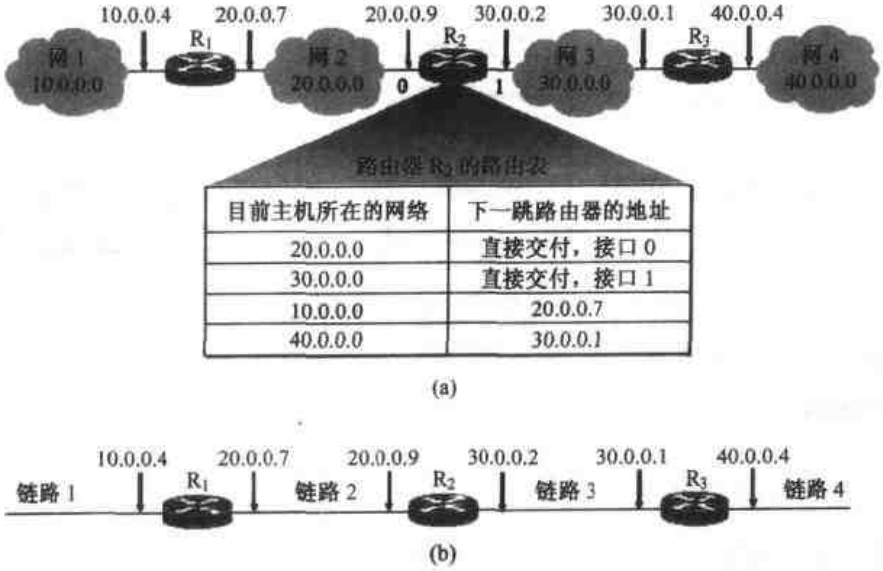


图 6-18　路由表举例：(a)路由器 R_2 的路由表；(b)将网络简化为一条链路

可以将整个的网络拓扑简化为图 6-18(b)所示的那样。在简化图中，网络变成了一条链路，但每一个路由器旁边都注明其 IP 地址。使用这样的简化图，可以使我们不用关心某个网络内部的拓扑以及连接在该网络上有多少台计算机，因为这些对于研究分组转发问题并没有什么关系。这样的简化图强调了在互联网上转发分组时，是从一个路由器转发到下一个路由器。

总之，在路由表中，对每一条路由最主要的是以下两项[①]：

（目的网络地址，下一跳地址）

于是，我们就根据目的网络地址来确定下一跳路由器，这样做的结果是：

（1）IP 数据报首先要设法找到目的主机所在目的网络上的路由器（间接交付）。

① 注：一个实际的路由表还会有其他的一些内容，例如，标志、参考计数、使用情况以及接口等。“标志”可以设置多个字符以说明不同的意思。如 U 表示该路由是可用的，G 表示下一跳地址是一个路由器因而是间接交付（如不设置 G，则表示直接交付），H 表示该路由是到一个主机（如不设置 H，则表示该路由是到一个网络）。“参考计数”是给出正在使用该路由的活动进程的个数。“使用情况”显示出通过该路由的分组数。“接口”是本地接口的名字，指出分组应当从哪一个接口转发。

（2）只有到达最后一个路由器时，才试图向目的主机进行直接交付。

虽然因特网所有的分组转发都是**基于目的主机所在的网络**，但在大多数情况下都允许有这样的特例，即对特定的目的主机指明一个路由。这种路由叫做**特定主机路由**。采用特定主机路由可使网络管理人员能更方便地控制网络和测试网络，同时也可在需要考虑某种安全问题时采用这种特定主机路由。在对网络的连接或路由表进行排错时，指明到某一个主机的特殊路由就十分有用。

和结点交换机路由表的情况相似，路由器也可采用默认路由以减少路由表所占用的空间和搜索路由表所用的时间。

根据以上所讲的，在因特网中某一个路由器的 IP 层所执行的**分组转发算法**如下：

（1）从数据报的首部提取目的站的 IP 地址 *D*，得出目的网络地址为 *N*。

（2）若 *N* 就是与此路由器直接相连的某个网络地址，则这种交付为**直接交付**，即不需要再经过其他的路由器。这时就直接通过该网络将数据报交付给目的站 *D*（这里包括将目的主机地址 *D* 转换为具体的硬件地址，将数据报封装为 MAC 帧，再发送此帧）；否则就是间接交付，执行(3)。

（3）若路由表中有目的地址为 *D* 的特定主机路由，则将数据报传送给路由表中所指明的下一跳路由器；否则，执行(4)。

（4）若路由表中有到达网络 *N* 的路由，则将数据报传送给路由表中所指明的下一跳路由器；否则，执行(5)。

（5）若路由表中有一个默认路由，则将数据报传送给路由表中所指明的默认路由器；否则，执行(6)。

（6）报告转发分组出错。

这里我们再强调指出，在 IP 数据报的首部中没有地方可以用来指明“下一跳路由器的 IP 地址”。在 IP 数据报的首部写上的 IP 地址是源 IP 地址和目的 IP 地址，而没有中间经过的路由器的 IP 地址。既然 IP 数据报中没有下一跳路由器的 IP 地址，那么待转发的数据报又怎样能够找到下一跳路由器呢？

当路由器收到一个待转发的数据报，在从路由表得出下一跳路由器的 IP 地址后，不是将下一跳路由器的 IP 地址填入 IP 数据报，而是送交下层的网络接口软件。网络接口软件负责将下一跳路由器的 IP 地址转换成硬件地址（使用 ARP），并将此硬件地址放在链路层的 MAC 帧的首部，然后根据这个硬件地址找到下一跳路由器。由此可见，当发送一连串的数据报时，上述的这种查找路由表、计算硬件地址、写入 MAC 帧的首部等过程，将不断地重复进行，造成了一定的开销。

那么，能不能在路由表中不使用 IP 地址而直接使用硬件地址呢？不行。我们一定要弄清楚，使用抽象的 IP 地址，本来就是为了隐蔽各种底层网络的复杂性而便于分析和研究问题，这样就不可避免地要付出些代价，例如在选择路由时多了一些上述的开销。但反过来，如果在路由表中直接使用硬件地址，那就会带来更多的麻烦。

上面所讨论的是 IP 层怎样根据路由表的内容进行分组转发，而没有涉及到路由表一开始是如何建立的以及路由表中的内容应如何进行更新。但是在进一步讨论路由选择之前，我们还要先介绍划分子网和构造超网这两个非常重要的概念。

6.3 划分子网和构造超网

6.3.1 划分子网

1．从两级 IP 地址到三级 IP 地址

在今天看来，在 ARPANET 的早期，IP 地址的设计确实不够合理。例如：

第一，**IP 地址空间的利用率有时很低。**

每一个 A 类地址网络可连接的主机数超过 1 千万，而每一个 B 类地址网络可连接的主机数也超过 6 万。然而有些网络对连接在网络上的计算机数目有限制，根本达不到这样大的数值。例如 10BASE-T 以太网规定其最大结点数只有 1024。这样的以太网若使用一个 B 类地址就浪费 6 万多个 IP 地址，地址空间的利用率还不到 2%，而其他单位的主机无法使用这些被浪费的地址。据统计，超过半数的 B 类地址网络所连接的主机还不到 50 台，而这些单位并不愿意申请一个足够使用的 C 类地址（理由是考虑到今后可能的发展）。IP 地址的浪费，还会使 IP 地址空间的资源过早地被用完。

从网络的吞吐量考虑，将大量主机安装在一个网络上往往会影响网络的性能。当网络上工作的主机数小于一定数值时，网络的吞吐量和网络上工作的主机数大约成正比。但是当网络上工作的主机数超过一定数值时，拥塞就可能产生，这就导致网络的吞吐量增加缓慢，甚至反而会随主机数的增加而下降。因此，从提高网络的吞吐量考虑，一个网络上的主机数也不应太多。这一因素也使得 IP 地址空间的利用率不可能很高。

第二，给每一个物理网络分配一个网络号会**使路由表变得太大**因而使网络性能变坏。

每一个路由器都应当能够从路由表查出应怎样到达其他网络的下一跳路由器。因此，互联网中的网络数越多，路由器的路由表的项目数也就越多。这样，即使我们拥有足够多的 IP 地址资源可以给每一个物理网络分配一个网络号，也会导致路由器中的路由表中的项目数过多。这不仅增加了路由器的成本（需要更多的存储空间），而且使查找路由时耗费更多的时间，同时也使路由器之间定期交换的路由信息急剧增加，因而使路由器和整个因特网的性能都下降了。

第三，**两级的 IP 地址不够灵活。**

有时情况紧急，一个单位需要在新的地点马上开通一个新的网络。但是在申请到一个新的 IP 地址之前，新增加的网络是不可能连接到因特网上工作的。我们希望有一种方法，使本单位能随时灵活地增加本单位的网络，而不必事先到因特网管理机构去申请新的网络号。原来的两级的 IP 地址无法做到这一点。

为解决上述问题，从 1985 年起在 IP 地址中又增加了一个**“子网号字段”**，使两级的 IP 地址变成为三级的 IP 地址，它能够较好地解决上述问题，并且使用起来也很灵活。这种做法叫作**划分子网**(subnetting) [RFC 950]，或**子网寻址**或**子网路由选择**。划分子网已成为因特网的正式标准协议。

划分子网的基本思路如下：

（1）一个拥有许多物理网络的单位，可将所属的物理网络划分为若干个**子网**(subnet)。划分子网纯属一个单位内部的事情，本单位以外的网络看不见这个网络是由多少个子网组成，因为这个单位对外仍然表现为一个没有划分子网的网络。

（2）划分子网的方法是从网络的主机号借用若干个比特作为子网号 subnet-id，而主机号 host-id 也就相应减少了若干个比特。于是两级的 IP 地址在本单位内部就变为三级的 IP 地址：网络号 net-id、子网号 subnet-id 和主机号 host-id。或者可以用以下记法来表示：

$$\text{IP 地址 ::= \{ <网络号>, <子网号>, <主机号>\}} \qquad (6\text{-}2)$$

（3）凡是从其他网络发送给本单位某个主机的 IP 数据报，仍然是根据 IP 数据报的目的网络号 net-id 找到连接在本单位网络上的路由器。但此路由器在收到 IP 数据报后，再按目的网络号 net-id 和子网号 subnet-id 找到目的子网，将 IP 数据报交付给目的主机。

下面用例子说明划分子网的概念。图 6-19 表示一个单位拥有一个 B 类 IP 地址，网络地址是 145.13.0.0 (net-id 是 145.13)。凡目的地址为 145.13.x.x 的数据报都被送到这个网络上的路由器 R_1。

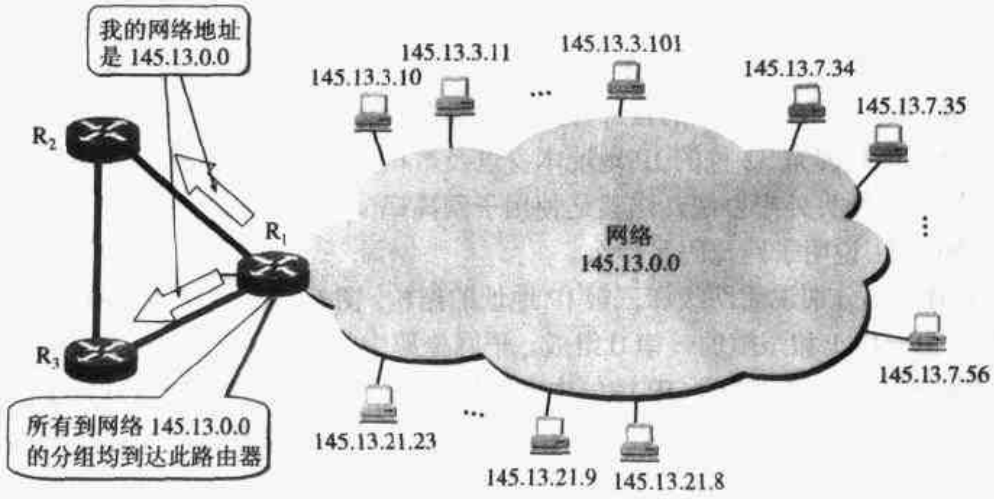


图 6-19　一个 B 类网络 145.13.0.0

现将图 6-19 的网络划分为三个子网（图 6-20）。这里假定子网号 subnet-id 占用 8 位，因

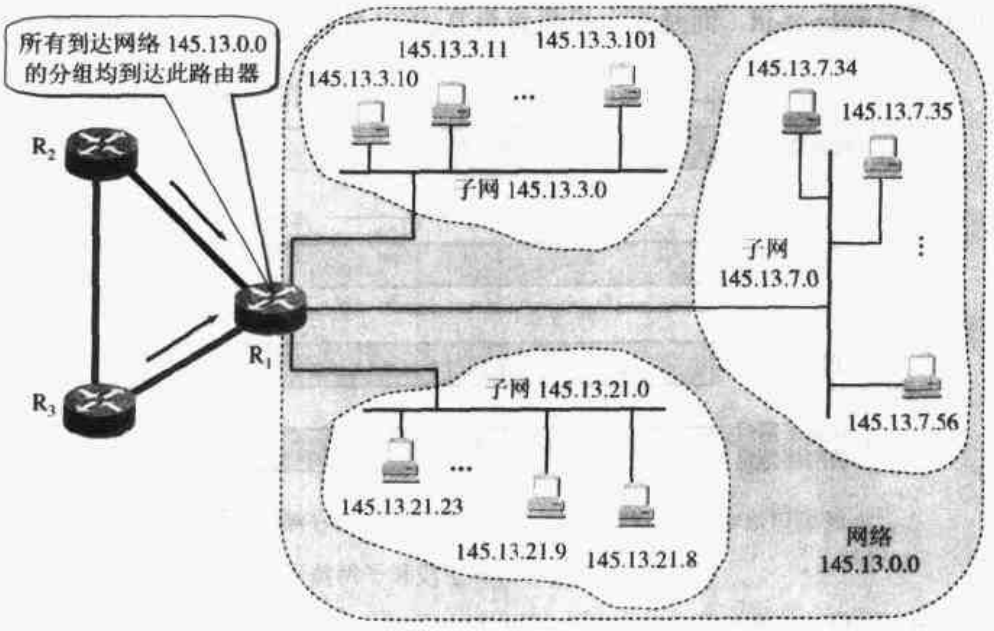


图 6-20　将图 6-19 的网络划分为三个子网，但对外仍是一个网络

此在增加了子网号后，主机号 host-id 就只有 8 位。所划分的三个子网分别是：145.13.3.0，145.13.7.0 和 145.13.21.0。在划分子网后，整个网络对外部仍表现为一个网络，其网络地址仍为 145.13.0.0。但网络 145.13.0.0 上的路由器 R_1 在收到数据报后，再根据数据报的目的地址将其转发到相应的子网。

总之，当没有划分子网时，IP 地址是两级结构，地址的网络号字段也就是 IP 地址的“**因特网部分**”，而主机号字段是 IP 地址的“**本地部分**”。

划分子网后 IP 地址就变成了三级结构。请注意，划分子网只是将 IP 地址的**本地部分进行再划分**，而不改变 IP 地址的因特网部分。

2. 子网掩码

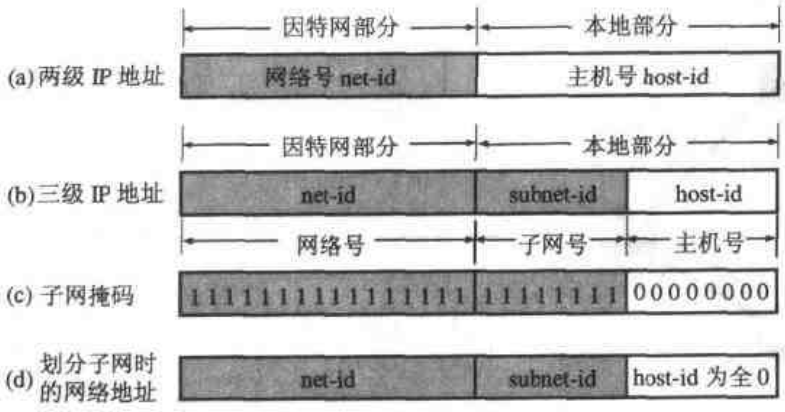
细心的读者可能会发现一个问题，这就是所有子网上的主机和外界的通信都必须通过原来网络上的路由器 R_1（见图 6-20）。这种通信方式显然是低效的。有没有可能让每一个子网上的主机能够通过本子网上的路由器直接和外界通信呢？答案是可以的，但还要采用一些措施。

我们知道，从一个 IP 数据报的首部并**无法判断**源主机或目的主机所连接的网络是否进行了子网的划分。这是因为 32 位的 IP 地址本身以及数据报的首部都没有包含任何有关子网划分的信息。因此必须另外想办法，这就是使用**子网掩码**(subnet mask)。

我们用图 6-21 说明子网掩码是什么。

图 6-21(a)和(b)分别说明两级和三级 IP 地址的结构。图 6-21(c)说明子网掩码和 IP 地址都是 32 bit 长，由一串 1 和跟随的一串 0 组成。子网掩码中的 1 对应于 IP 地址中的网络号和子网号，而子网掩码中的 0 对应于 IP 地址中主机号。虽然 RFC 文档中没有规定子网掩码中的一串 1 必须是连续的，但却极力推荐人家在子网掩码中选用**连续的** 1 以免出现可能发生的差错。

图 6-21(d)表示在划分子网的情况下，网络地址（即子网地址）就是将主机号 host-id 置为 0 的 IP 地址。这也就是将子网掩码和 IP 地址逐比特相“**与**”(AND)的结果（计算机进行这种逻辑运算是很容易的）。这里要注意：**网络地址**（在划分子网时常称为**子网地址**）并不是仅仅是一个子网号 subnet-id，而是将主机号置为 0 的 IP 地址。



(a)两级 IP 地址；(b)三级 IP 地址；(c)子网掩码；(d)划分子网时的网络地址。

图 6-21　IP 地址的各字段和子网掩码

使用子网掩码的好处就是：不管网络有没有划分子网，不管网络字段 net-id 的长度是 1 字节、2 字节或 3 字节，只要将子网掩码和 IP 地址进行逐比特的“**与**”运算(AND)，就立即

得出网络地址来。这样在路由器处理到来的分组时就可采用同样的算法。

这里还要弄清一个问题，这就是：在不划分子网时，既然没有子网，为什么还要使用子网掩码？这就是为了简化路由器的路由选择算法。现在因特网的标准规定：所有的网络都必须有一个子网掩码，同时在路由器的路由表中也必须有子网掩码这一栏。如果一个网络不划分子网，那么该网络的子网掩码就使用**默认子网掩码**。默认子网掩码中 1 比特的位置和 IP 地址中的网络号字段正好相对应。因此，若将默认的子网掩码和某个不划分子网的 IP 地址逐比特相“与”(AND)，就得出该 IP 地址的网络地址来。这样做可以不用查找该地址的类别比特就能知道这是哪一类的 IP 地址。显然，

A 类地址的默认子网掩码是 255.0.0.0，或 0xFF000000。

B 类地址的默认子网掩码是 255.255.0.0，或 0xFFFF0000。

C 类地址的默认子网掩码是 255.255.255.0，或 0xFFFFFF00。

图 6-22 是这三类 IP 地址的网络地址和相应的默认子网掩码。

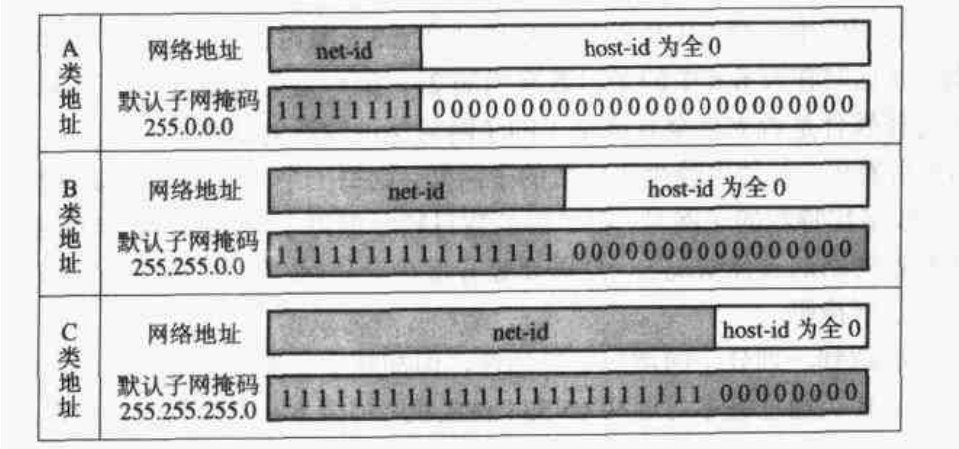


图 6-22　A 类、B 类和 C 类 IP 地址的默认子网掩码

子网掩码是一个网络或一个子网的重要属性。在[RFC 950]成为因特网的正式标准后，路由器在和相邻路由器交换路由信息时，必须将自己所在网络（或子网）的子网掩码告诉相邻路由器。在路由器的路由表中的每一个项目，除了要给出目的网络地址外，还必须同时给出该网络的子网掩码。若一个路由器连接在两个子网上就拥有两个网络地址和两个子网掩码。

我们以一个 B 类地址为例，说明可以有多少种子网划分的方法。在采用固定长度子网时，所划分的所有子网的子网掩码都是相同的（见表 6-6）。

表 6-6　B 类地址的子网划分选择（使用固定长度子网）

子网号的比特数	子 网 掩 码	子 网 数	主机数/子网
2	255.255.192.0	2	16 382
3	255.255.224.0	6	8190
4	255.255.240.0	14	4094
5	255.255.248.0	30	2046
6	255.255.252.0	62	1022
7	255.255.254.0	126	510
8	255.255.255.0	254	254

续表

子网号的比特数	子 网 掩 码	子 网 数	主机数/子网
9	255.255.255.128	510	126
10	255.255.255.192	1022	62
11	255.255.255.224	2046	30
12	255.255.255.240	4094	14
13	255.255.255.248	8190	6
14	255.255.255.252	16 382	2

在表 6-6 中，子网数是根据子网号 subnet-id 计算出来的。若 subnet-id 有 n bit，则共有 2^n 种可能的排列。除去全 0 和全 1 这两种情况，就得出表中的子网数。

表中的“子网号的比特数”中没有 0、1、15 和 16 这 4 种情况，因为这没有意义。但是请读者注意，虽然根据已成为因特网标准协议的[RFC 950]文档，子网号不能为全 1 或全 0，但随着无分类域间路由选择 CIDR 的广泛使用（在 6.3.3 节讨论），现在全 1 和全 0 的子网号也可以使用了（这时在表 6-6 中的子网数应当加 2），但一定要谨慎使用，要弄清你的路由器所用的路由选择软件是否支持全 0 或全 1 的子网号这种较新的用法。

从表 6-6 可看出，若使用较少比特数的子网号，则每一个子网上可连接的主机数就较大。反之，若使用较多比特数的子网号，则子网的数目较多但每个子网上可连接的主机数就较小。因此我们可根据网络的具体情况（一共需要划分多少个子网，每个子网中最多有多少个主机）来选择合适的子网掩码。

我们还应注意到，划分子网增加了灵活性，但却减少了能够连接在网络上的主机总数。例如，本来一个 B 类地址最多可连接 65 534 台主机，但表 6-6 中任意一行的最后两项的乘积一定小于 65 534。

对 A 类和 C 类地址的子网划分也可得出类似的表格，读者可自行算出。

我们将在 6.3.2 节进一步讨论使用了子网掩码后应怎样查找路由表。

6.3.2 使用子网掩码的分组转发过程

在 6.2.5 节的分组转发算法中的第(1)条就是：“从数据报的首部提取目的站的 IP 地址 D，得出目的网络地址为 N”。在不划分子网的两级 IP 地址下，从 IP 地址得出网络地址是个很简单的事。但在划分子网的情况下，从 IP 地址却不能惟一地得出网络地址来，这是因为网络地址取决于那个网络所采用的子网掩码，但数据报的首部并没有提供子网掩码的信息。因此分组转发的算法也必须做相应的改动。

图 6-23 画出了包括三个子网的网络拓扑。各子网的网络地址和子网掩码都已标注在图中。作为例子，图中还给出了路由器 R_1 的路由表。我们注意到，使用子网划分后，路由表中的每行所包括的主要内容是：目的网络地址、子网掩码和下一跳地址。

我们假定图 6-23 中的主机 H_1 要向某一个主机发送一个分组。首先，主机 H_1 应判断是采用直接交付还是间接交付。主机 H_1 采用的方法是：将分组的目的地址和主机 H_1 自己的子网掩码进行逐比特相“与”的运算。

若运算的结果等于主机 H_1 的网络地址，则说明目的主机与主机 H_1 是连接在同一个子网上，因此可以直接交付而不需要找下一跳的路由器来转发。

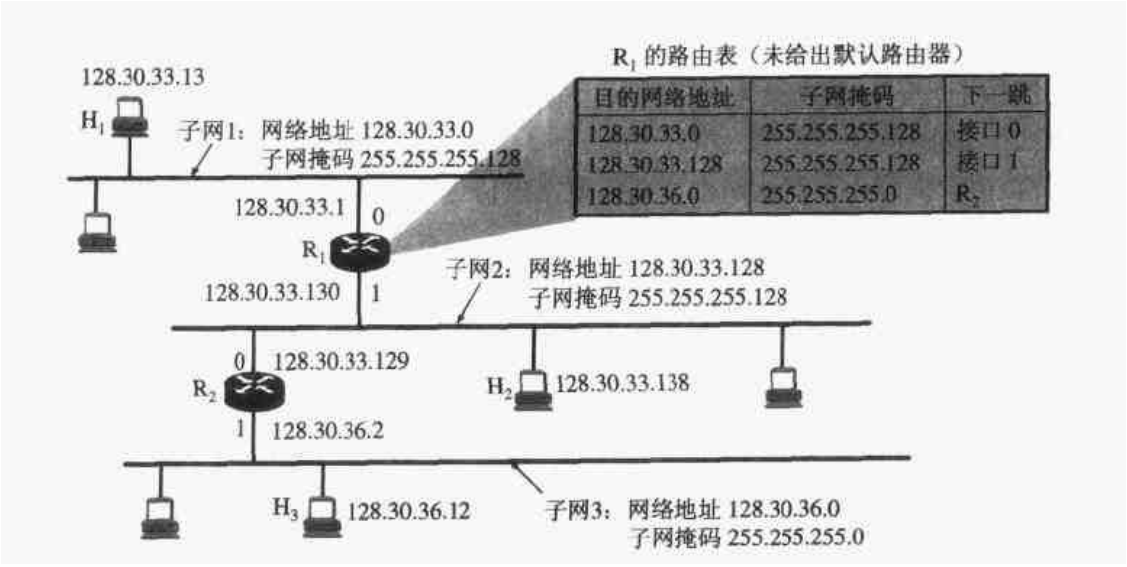


图 6-23　划分子网后分组的转发举例

但若“与”运算的结果不等于 H_1 的网络地址，则表明应采用间接交付，必须将该分组交给本子网上的一个路由器进行转发。

例如，现在假定主机 H_1 要发送的分组是给 H_2，即所发送的分组的目的地址是 H_2 的 IP 地址 128.30.33.138。主机 H_1 要进行的操作是将本子网的“子网掩码 255.255.255.128”与 H_2 的“IP 地址 128.30.33.138”逐比特相“与”，得出 128.30.33.128，它不等于 H_1 的网络地址 (128.30.33.0)，这说明 H_2 与 H_1 不在同一个子网上。因此 H_1 知道不能将分组直接交付给 H_2，而必须将分组交给子网上的默认路由器 R_1，由 R_1 来转发。

下面讨论路由器 R_1 在收到一个分组后应如何查找其路由表。为简单起见，在 R_1 的路由表没有画出默认路由器。

路由器 R_1 先找路由表中的第一行，看看这一行的网络地址和收到的分组的网络地址是否匹配。因为并不知道收到的分组的网络地址，因此只能试试看。这就是用这一行（子网 1）的“子网掩码 255.255.255.128”和收到的分组的“目的地址 128.30.33.138”逐比特相“与”，得出 128.30.33.128。如果这个数值和这一行给出的目的网络地址一致，就说明收到的分组是发送给本子网上的某个主机。但现在比较的结果是不一致。

因为和路由表第一行的比较结果是“不匹配”，所以用同样方法继续往下找第二行。用第二行的“子网掩码 255.255.255.128”和该分组的“目的地址 128.30.33.128”逐比特相“与”，结果也是 128.30.33.128。但这个结果和第二行的目的网络地址相匹配，说明这个网络（子网 2）就是收到的分组所要寻找的目的网络。于是不需要再找下一个路由器进行间接交付了，R_1 将分组从接口 1 直接交付给主机 H_2（它们都在一个子网上）。

这样，我们得出在划分子网的情况下路由器转发分组的算法：

（1）从收到的数据报的首部提取目的 IP 地址 D。

（2）先判断是否为直接交付。对路由器直接相连的网络逐个进行检查：用各网络的子网掩码和 D 逐比特相“与”，看结果是否和相应的网络地址匹配。若匹配，则将分组进行直接交付（需要将 D 转换成物理地址，将数据报封装成帧发送出去），转发任务结束。否则就是间接交付，执行(3)。

（3）若路由表中有目的地址为 D 的特定主机路由，则将数据报传送给路由表中所指明的下一跳路由器；否则，执行(4)。

（4）对路由表中的每一行（目的网络地址，子网掩码，下一跳地址），将其中的子网掩码和 D 逐比特相“与”，其结果为 N。若 N 与该行的目的网络地址匹配，则将数据报传送给该行指明的下一跳路由器；否则，执行(5)。

（5）若路由表中有一个默认路由，则将数据报传送给路由表中所指明的默认路由器；否则，执行(6)。

（6）报告转发分组出错。

6.3.3 无分类编址 CIDR

1．网络前缀

划分子网在一定程度上缓解了因特网在发展中遇到的困难。然而在 1992 年因特网仍然面临三个必须尽早解决的问题，这就是：

（1）B 类地址在 1992 年已分配了近一半，眼看就要在 1994 年 3 月全部分配完毕！

（2）因特网主干网上的路由表中的项目数急剧增长（从几千个增长到几万个）。

（3）整个 IPv4 的地址空间最终将全部耗尽。

当时预计前两个问题将在 1994 年变得非常严重。因此 IETF 很快地就研究出采用**无分类编址**的方法来解决前两个问题，而第三个问题属于更加长远的问题，由 IETF 的 IPv6 工作组负责研究解决。

其实早在 1987 年，RFC 1009 就指明了在一个划分子网的网络中可同时使用几个不同的子网掩码。使用**变长子网掩码** VLSM (Variable Length Subnet Mask)可进一步提高 IP 地址资源的利用率。在 VLSM 的基础上又进一步研究出**无分类编址**方法，它的正式名字是**无分类域间路由选择** CIDR (Classless Inter-Domain Routing，CIDR 的读音是“sider”)。在 1993 年形成了 CIDR 的 RFC 文档：RFC 1517～1519 和 1520。现在 CIDR 已成为因特网建议标准协议。

CIDR 最主要的特点有两个：

（1）CIDR **消除了传统的 A 类、B 类和 C 类地址以及划分子网的概念**，因而可以更加有效地分配 IPv4 的地址空间，并且可以在新的 IPv6 使用之前容许因特网的规模继续增长。CIDR 使用各种长度的“**网络前缀**”(network-prefix)来代替分类地址中的网络号和子网号，而不是像分类地址中只能使用 1 字节、2 字节和 3 字节长的网络号。CIDR 不再使用“子网”的概念而使用**网络前缀**，使 IP 地址从三级编址（使用子网掩码）又回到了两级编址，但这已是**无分类的两级编址**。它的记法是：

$$\text{IP 地址} ::= \{\text{<网络前缀>}, \text{<主机号>}\} \tag{6-3}$$

CIDR 还使用“**斜线记法**”(slash notation)，它又称为 CIDR 记法，即在 IP 地址后面加上一个斜线“/”，然后写上网络前缀所占的比特数（这个数值对应于三级编址中子网掩码中比特 1 的个数）。例如，128.14.46.34/20，表示在这个 32 bit 的 IP 地址中，前 20 bit 表示网络前缀，而后面的 12 bit 为主机号。有时需要将点分十进制的 IP 地址写成二进制表示的地址才能看清楚网络前缀和主机号。例如，上述地址的前 20 bit 是 10000000 00001110 0010（这就是网络前缀），而后面的 12 bit 是 1110 00100010 (这就是主机号 host-id)。

（2）CIDR 将网络前缀都相同的连续的 IP 地址组成“**CIDR 地址块**”。一个 CIDR 地址块是由地址块的起始地址（即地址块中地址数值最小的一个）和地址块中的地址数来定义的。CIDR 地址块也可用斜线记法来表示。例如，128.14.32.0/20 表示的地址块共有 2^{12} 个地址（因

为斜线后面的 20 是网络前缀的比特数，所以主机号的比特数是 12，因而地址数就是 2^{12}），而该地址块的起始地址是 128.14.32.0。在不需要指出地址块的起始地址时，也可将这样的地址块简称为“/20 地址块”。上面的地址块的最小地址和最大地址是：

最小地址	128.14.32.0	10000000 00001110 00100000 00000000
最大地址	128.14.47.255	10000000 00001110 00101111 11111111

当然，这两个全 0 和全 1 的主机号地址一般并不使用。通常只使用在这两个地址之间的地址。

当我们见到斜线记法表示的地址时，一定要根据上下文弄清它是指一个单个的 IP 地址还是指一个地址块。

由于一个 CIDR 地址块可以表示很多地址，所以在路由表中就利用 CIDR 地址块来查找目的网络。这种地址的聚合常称为**路由聚合**(route aggregation)，它使得路由表中的一个项目可以表示很多个（例如上千个）原来传统分类地址的路由。路由聚合也称为**构成超网**(supernetting)。如果没有采用 CIDR，则在 1994 和 1995 年，因特网的一个路由表就会超过 7 万个项目，而使用了 CIDR 后，在 1996 年一个路由表的项目数才只有 3 万多个。路由聚合有利于减少路由器之间的路由选择信息的交换，从而提高了整个因特网的性能。

CIDR 虽然不使用子网了，但仍然使用“**掩码**”这一名词（但不叫子网掩码）。对于/20 地址块，它的掩码是：11111111 11111111 11110000 00000000(20 个连续的 1)。斜线记法中的数字就是掩码中 1 的个数。

CIDR 记法有几种等效的形式，例如，10.0.0.0/10 可简写为 10/10，也就是将点分十进制中低位连续的 0 省略。10.0.0.0/10 相当于指出 IP 地址 10.0.0.0 的掩码是 255.192.0.0。

比较清楚的表示方法是直接使用二进制。例如，10.0.0.0/10 可写为：

00001010 00xxxxxx xxxxxxxx xxxxxxxx

这里的 22 个 x 可以是任意值的主机号（但全 0 和全 1 的主机号一般不使用）。因此 10/10 可表示包含有 2^{22} 个 IP 地址的地址块，这些地址块都具有相同的网络前缀 00001010 00。

另一种简化表示方法是在网络前缀的后面加一个星号*，如：

00001010 00*

意思是：在星号*之前是网络前缀，而星号*表示 IP 地址中的主机号，可以是任意值。

对于前缀比特数不是 8 的整数倍时，需要比较小心地对待。

表 6-7 给出了最常用的 CIDR 地址块。表中的 K 表示 2^{10} 即 1024。网络前缀小于 13 或大于 27 都较少使用。在“包含的地址数”中，没有将全 1 和全 0 的主机号除外。

从表 6-7 可看出，除最后几行外，CIDR 地址块都包含了**多个 C 类地址**，这就是“**构成超网**”这一名词的来源。

表 6-7 常用的 CIDR 地址块

CIDR 前缀长度	点分十进制	包含的地址数	包含的分类的网络数
/13	255.248.0.0	512 K	8 个 B 类或 2048 个 C 类
/14	255.252.0.0	256 K	4 个 B 类或 1024 个 C 类
/15	255.254.0.0	128 K	2 个 B 类或 512 个 C 类
/16	255.255.0.0	64 K	1 个 B 类或 256 个 C 类

续表

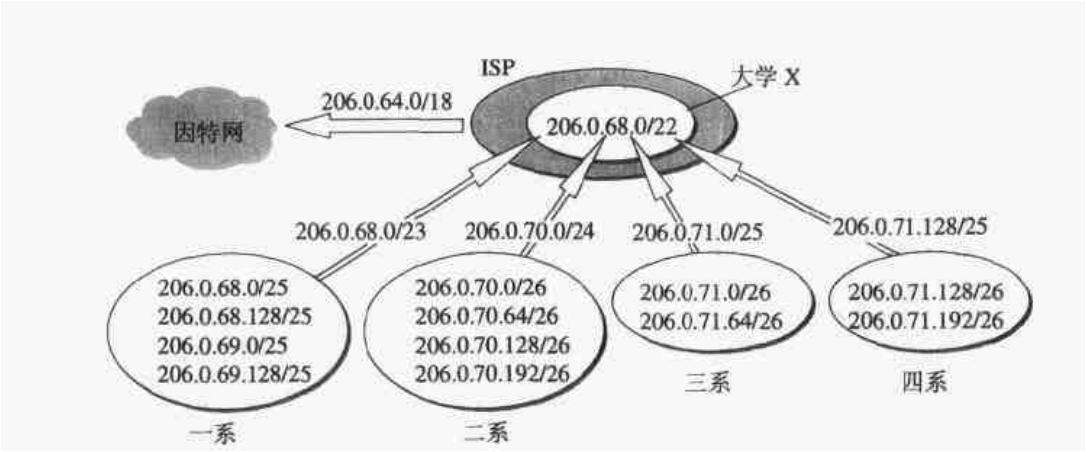
CIDR 前缀长度	点分十进制	包含的地址数	包含的分类的网络数
/17	255.255.128.0	32 K	128 个 C 类
/18	255.255.192.0	16 K	64 个 C 类
/19	255.255.224.0	8 K	32 个 C 类
/20	255.255.240.0	4 K	16 个 C 类
/21	255.255.248.0	2 K	8 个 C 类
/22	255.255.252.0	1 K	4 个 C 类
/23	255.255.254.0	512	2 个 C 类
/24	255.255.255.0	256	1 个 C 类
/25	255.255.255.128	128	1/2 个 C 类
/26	255.255.255.192	64	1/4 个 C 类
/27	255.255.255.224	32	1/8 个 C 类

应当注意，在配置基于 CIDR 的网络时，可能有一些主机本来是使用分类的 IP 地址。它们可能不允许将网络前缀设置成比原来分类地址的子网掩码的 1 比特长度更短。例如，将网络配置成 200.25.16.0/20 时就可能不行，因为这原来是一个 C 类地址，其子网掩码的长度至少是 24 bit。只有在主机的软件支持 CIDR 时，网络前缀才能比原来的掩码长度短。但是，若将 200.25.16.0/20 配置成 16 个/24 地址块就不会有问题，因为不支持 CIDR 的主机会将本地的 /24 解释成 C 类网络。

使用 CIDR 的一个好处就是可以更加有效地分配 IPv4 的地址空间，因此现在的因特网服务提供者 ISP 都愿意使用 CIDR。在分类地址的环境中，因特网服务提供者 ISP 向其客户分配 IP 地址时（这里指的是固定 IP 地址用户而不是拨号上网的用户），只能以 /8, /16 或 /24 为单位来分配。但在 CIDR 环境，ISP 可根据每个客户的具体情况进行分配。例如，某 ISP 已拥有地址块 206.0.64.0/18（相当于有 64 个 C 类网络）。现在某大学需要 800 个 IP 地址。在不使用 CIDR 时，ISP 或者可以给大学分配一个 B 类地址（但这将浪费 64 734 个 IP 地址），或者分配 4 个 C 类地址（但这会在各个路由表中出现对应于该大学的 4 个相应的项目）。然而在 CIDR 环境下，ISP 可以给该大学分配一个地址块 206.0.68.0/22，它包括 1024（即 2^{10}）个 IP 地址，相当于 4 个连续的 C 类 /24 地址块，占该 ISP 拥有的地址空间的 1/16。这样，地址空间的利用率显然提高了。像这样的地址块有时也称为一个“**编址域**”或“**域**”(domain)。显然，用 CIDR 分配的地址块中的地址数一定是 2 的整数次幂。

这个大学可自由地对本校的各系分配地址块，而各系还可再划分本系的地址块，如图 6-24 所示。CIDR 的地址块分配有时不易看清，这是因为网络前缀和主机号的界限不是恰好出现在整数字节处。只要写出地址的二进制表示（从图中的地址块的二进制表示中可看出，实际上只需要将其中的一个关键字节转换为二进制的表示即可），弄清网络前缀的比特数，就不会把地址块的范围弄错。

从图 6-24 可以清楚地看出**地址聚合**的概念。这个 ISP 共拥有 64 个 C 类网络。如果不采用 CIDR 技术，则在与该 ISP 的路由器交换路由信息的每一个路由器的路由表中，就需要有 64 个项目。但采用地址聚合后，就只需用路由聚合后的一个项目 206.0.64.0/18 就能找到该 ISP。同理，大学共有 4 个系。在 ISP 内的路由器的路由表中，也是需使用 206.0.68.0/22 这一个项目。



单位	地址块	二进制表示	地址数
ISP	206.0.64.0/18	11001110.00000000.01*	16384
大学	206.0.68.0/22	11001110.00000000.010001*	1024
一系	206.0.68.0/23	11001110.00000000.0100010*	512
二系	206.0.70.0/24	11001110.00000000.01000110.*	256
三系	206.0.71.0/25	11001110.00000000.01000111.0*	128
四系	206.0.71.128/25	11001110.00000000.01000111.1*	128

图 6-24　CIDR 地址块划分举例

从图 6-24 下面表格中的二进制地址可看出，将四个系的路由聚合为大学的一个路由（即构成超网），是将网络前缀缩短。**网络前缀越短，其地址块所包含的地址数就越多**。而在三级结构的 IP 地址中，划分子网是使网络前缀变长。

2．最长前缀匹配

在使用 CIDR 时，由于采用了网络前缀这种记法，IP 地址由网络前缀和主机号这两个部分组成，因此在路由表中的项目也要有相应的改变。这时，每个项目由“**网络前缀**”和“**下一跳地址**”组成。但是在查找路由表时可能会得到不止一个匹配结果。这样就带来一个问题：我们应当从这些匹配结果中选择哪一条路由呢？

正确的答案是：**应当从匹配结果中选择具有最长网络前缀的路由**。这叫作**最长前缀匹配**(longest-prefix matching)，这是因为网络前缀越长，其地址块就越小，因而路由就越具体(more specific)。最长前缀匹配又称为**最长匹配**或**最佳匹配**。为了说明最长前缀匹配的概念，我们仍以前面的例子来讨论。

假定大学下属的四系现在希望 ISP 将转发给四系的数据报直接发到四系而不要经过大学的路由器，但又不愿意改变自己原来使用的 IP 地址块。因此，在 ISP 的路由器的路由表中，至少要有以下两个项目，即 206.0.68.0/22（大学）和 206.0.71.128/25（四系）。现在假定 ISP 收到一个数据报，其目的 IP 地址为 D = 206.0.71.130。将 D 和路由表中这两个项目的掩码逐比特相“与”。将所得的逐比特相“与”的结果按顺序写在下面。

D 和 11111111 11111111 11111100 00000000 逐比特相“与”= 206.0.68.0/22　匹配

D 和 11111111 11111111 11111111 10000000 逐比特相“与”= 206.0.71.128/25 匹配

不难看出，现在同一个 IP 地址 D 可以在路由表中找到两个目的网络（大学和四系）和该地址相匹配。根据最长前缀匹配的原理，应当选择后者，将收到的数据报转发到后一个目的网络（四系），即选择两个匹配的地址中更具体的一个。

从以上的讨论可以看出，如果 IP 地址的分配一开始就采用 CIDR，那么我们可以按网络所在的地理位置来分配地址块，这样就可大大减少路由表中的路由项目。例如，可以将世界划分为四大地区，每一地区分配一个 CIDR 地址块[TANE96]：

地址块 194/7　(194.0.0.0 至 195.255.255.255)　分配给欧洲；

地址块 198/7　(198.0.0.0 至 199.255.255.255)　分配给北美州；

地址块 200/7　(200.0.0.0 至 201.255.255.255)　分配给中美洲和南美洲；

地址块 202/7　(202.0.0.0 至 203.255.255.255)　分配给亚洲和太平洋地区。

上面的每一个地址块包含有约 3200 万个地址。这种分配地址的方法就使得 IP 地址与地理位置相关联。它的好处是可以大大压缩路由表中的项目数。例如，凡是从中国发往北美的数据报（不管它是地址块 198/7 中的哪一个地址）都先送交位于美国的一个路由器，因此在路由表中使用一个项目就行了。

但是，在使用 CIDR 之前因特网的地址管理机构没有按地理位置来分配 IP 地址。现在要把已分配出的 IP 地址收回再重新分配是十分困难的事，因为这牵涉到很多正在工作的主机必须改变其 IP 地址。尽管这样，CIDR 的使用已经推迟了 IP 地址将要耗尽的日期。

3．使用二叉线索查找路由表

使用 CIDR 后，由于要寻找最长前缀匹配，使路由表的查找过程变得更加复杂了。当路由表的项目数很大时，怎样设法减小路由表的查找时间就成为一个非常重要的问题。例如，连接路由器的线路的速率为 10 Gb/s，而分组的平均长度为 2000 bit，那么路由器就应当平均每秒钟能够处理 500 万个分组（常记为 5 Mpps）。或者说，路由器处理一个分组的平均时间只有 200 ns(1 ns = 10^{-9} 秒)。因此，查找每一个路由所需的时间应当是非常短的。可见在路由表中必须使用很好的数据结构和使用先进的快速查找算法，这一直是人们积极研究的热门课题。

对无分类编址的路由表的最简单的查找算法就是对所有可能的前缀进行循环查找。例如，给定一个目的地址 D。对每一个可能的网络前缀长度 M，路由器从 D 中提取前 M 个比特，并假定所提取的比特构成了一个网络前缀，然后查找路由表中的网络前缀。所找到的最长匹配就对应于要查找的路由。

这种最简单的算法的明显缺点就是查找的次数太多。最坏的情况是路由表中没有这个路由。在这种情况下，算法仍要进行 32 次（具有 32 bit 的网络前缀是一个特定主机路由）。就是要找到一个传统的 B 类地址（即 /16），也要查找 16 次。对于经常使用的默认路由，这种算法都要经历 31 次的不必要的查找。

为了进行更加有效的查找，通常是将无分类编址的路由表存放在一种层次的数据结构中，然后自上而下地按层次进行查找。这里最常用的就是**二叉线索**(binary trie)[①]，它是一种特殊结构的树。IP 地址中从左到右的比特值决定了从根结点逐层向下层延伸的路径，而二叉线索中的各个路径就代表路由表中存放的各个地址。

图 6-25 用一个例子来说明二叉线索的结构。图中给出了 5 个 IP 地址。为了简化二叉线索的结构，可以先找出对应于每一个 IP 地址的**惟一前缀**(unique prefix)。所谓惟一前缀就是在

① 注：线索(trie)来自 retrieval（检索），读音与“try”相同。

表中所有的 IP 地址中，该前缀是惟一的。这样就可以用这些惟一前缀来构造二叉线索。在进行查找时，只要能够和惟一前缀相匹配就行了。

32 bit 的 IP 地址	惟一前缀
01000110 00000000 00000000 00000000	0100
01010110 00000000 00000000 00000000	0101
01100001 00000000 00000000 00000000	011
10110000 00000010 00000000 00000000	10110
10111011 00001010 00000000 00000000	10111

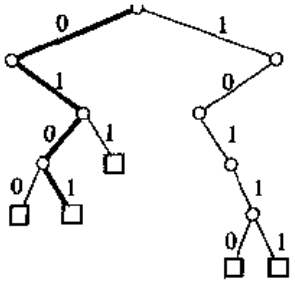


图 6-25　用 5 个前缀构成的二叉线索

从二叉线索的根结点自顶向下的深度最多有 32 层，每一层对应于 IP 地址中的一位。一个 IP 地址存入二叉线索的规则很简单。先检查 IP 地址左边的第一位，如为 0，则第一层的结点就在根结点的左下方；如为 1，则在右下方。然后再检查地址的第二位，构造出第二层的结点。依此类推，直到惟一前缀的最后一位。由于惟一前缀一般都小于 32 位，因此用惟一前缀构造的二叉线索的深度往往不到 32 层。图中较粗的折线就是前缀 0101 在这个二叉线索中的路径。二叉线索中的小圈圈是中间结点，而在路径终点的小方框是叶结点（也叫作外部结点）。每个叶结点代表一个惟一前缀。结点之间的连线旁边的数字表示这条边在惟一前缀中对应的比特是 0 或 1。

假定有一个 IP 地址是 10011011 01111010 00000000 00000000，需要查找该地址是否在此二叉线索中。我们从最左边查起。很容易发现，查到第三个字符（即前缀 10 后面的 0）时，在二叉线索中就找不到匹配的，说明这个地址不在这个二叉线索中。

以上只是给出了二叉线索这种数据结构的用法，而并没有说明“与惟一前缀匹配”和“与网络前缀匹配”的关系。显然，要将二叉线索用于路由表中，还必须使二叉线索中的每一个叶结点包含所对应的网络前缀和子网掩码。当搜索到一个叶结点时，就必须将寻找匹配的目的地址和该叶结点的子网掩码进行逐比特的“与”运算，看结果是否与对应的网络前缀相匹配。若匹配，就按下一跳的接口转发该分组。否则，就丢弃该分组。

总之，二叉线索只是提供了一种可以快速在路由表中找到匹配的叶结点的机制。但这是否和网络前缀匹配，还要和子网掩码进行一次逻辑与的运算。

为了提高二叉线索的查找速度，广泛使用了各种压缩技术。例如，在图 6-25 中的最后两个地址，其最前面的 4 个比特都是 1011。因此，只要一个地址的前 4 个比特是 1011，就可以跳过前面的 4 个比特（即压缩了 4 个层次）而直接从第 5 个比特开始比较。这样就可以减少查找的时间。当然，制作经过压缩的二叉线索需要更多的计算，但由于每一次查找路由表时都可以提高查找速度，因此这样做还是值得的。

6.4　因特网控制报文协议 ICMP

为了提高 IP 数据报交付成功的机会，在网际层使用了**因特网控制报文协议 ICMP** (Internet Control Message Protocol) [RFC 792]。ICMP 允许主机或路由器报告差错情况和提供有关异常情况的报告。ICMP 是因特网的标准协议。但 ICMP 不是高层协议，而是 IP 层的协议。ICMP 报文作为 IP 层数据报的数据，加上数据报的首部，组成数据报发送出去。ICMP

报文格式如图 6-26 所示。

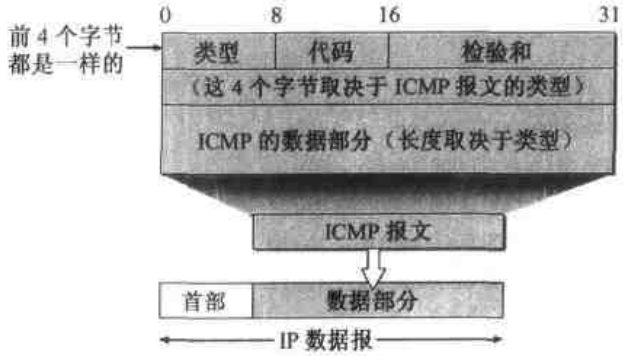


图 6-26　ICMP 报文的格式

ICMP 报文的种类有两种，即 **ICMP 差错报告报文**和 **ICMP 询问报文**。

ICMP 报文的前 4 个字节是统一的格式，共有三个字段：即类型、代码和检验和。接着的 4 个字节的内容与 ICMP 的类型有关。再后面是数据字段，其长度取决于 ICMP 的类型。ICMP 报文的类型字段的值与 ICMP 报文类型的对应关系如表 6-8 所示。

表 6-8　类型字段的值与 ICMP 报文的类型的关系

ICMP 报文种类	类型的值	ICMP 报文的类型
差错报告报文	3	终点不可达
	4	源站抑制(Source quench)
	11	时间超过
	12	参数问题
	5	改变路由(Redirect)
询问报文	8 或 0	回送(Echo)请求或回答
	13 或 14	时间戳(Timestamp)请求或回答
	17 或 18	地址掩码(Address mask)请求或回答
	10 或 9	路由器询问(Router solicitation)或通告

ICMP 报文的代码字段是为了进一步区分某种类型中的几种不同的情况。检验和字段用来检验整个 ICMP 报文。我们应当还记得，IP 数据报首部的检验和并不检验 IP 数据报的内容，因此不能保证经过传输的 ICMP 报文不产生差错。

ICMP 差错报告报文共有 5 种，即：

（1）**终点不可达**　终点不可达分为：网络不可达、主机不可达、协议不可达、端口不可达、需要分片但 DF 比特已置为 1，以及源路由失败等六种情况，其代码字段分别置为 0 至 5。当出现以上六种情况时就向源站发送终点不可达报文。

（2）**源站抑制**　当路由器或主机由于拥塞而丢弃数据报时，就向源站发送源站抑制报文，使源站知道应当将数据报的发送速率放慢。

（3）**时间超过**　当路由器收到生存时间为零的数据报时，除丢弃该数据报外，还要向源站发送时间超过报文。当目的站在预先规定的时间内不能收到一个数据报的全部数据报片时，就将已收到的数据报片都丢弃，并向源站发送时间超过报文。

（4）**参数问题**　当路由器或目的主机收到的数据报的首部中有的字段的值不正确时，

就丢弃该数据报，并向源站发送参数问题报文。

（5）**改变路由（重定向）**　路由器将改变路由报文发送给主机，让主机知道下次应将数据报发送给另外的路由器（可通过更好的路由）。

下面对改变路由报文进行简短的解释。我们知道，在因特网中各路由器之间要经常交换路由信息，以便动态更新各自的路由表。但在因特网中主机的数量远大于路由器的数量。主机如果也像路由器那样经常交换路由信息，就会产生很大的附加通信量，因而大大浪费了网络资源。所以，出于效率的考虑，连接在网络上的主机的路由表一般都采用人工配置，并且主机不和连接在网络上的路由器定期交换路由信息。在主机刚开始工作时，一般都在路由表中设置一个默认路由器的 IP 地址。不管数据报要发送到哪个目的地址，都一律先将数据报传送给网络上的这个默认路由器，而这个默认路由器知道到每一个目的网络的最佳路由。如果默认路由器发现主机发往某个目的地址的数据报的最佳路由不应当经过默认路由器而是应当经过网络上的另一个路由器 R 时，就用改变路由报文将此情况告诉主机。于是，该主机就在其路由表中增加一个项目：到某某目的地址应经过路由器 R（而不是默认路由器）。

所有的 ICMP 差错报告报文中的数据字段都具有同样的格式（图 6-27）。将收到的需要进行差错报告的 IP 数据报的首部和数据字段的前 8 个字节提取出来，作为 ICMP 报文的数据字段。再加上相应的 ICMP 差错报告报文的前 8 个字节，就构成了 ICMP 差错报告报文。提取收到的数据报的数据字段的前 8 个字节是为了得到运输层的端口号（对于 TCP 和 UDP）以及运输层报文的发送序号（对于 TCP）。这些信息对源站通知高层协议是有用的（端口的作用将在 7.2.2 中介绍）。整个 ICMP 报文作为 IP 数据报的数据。

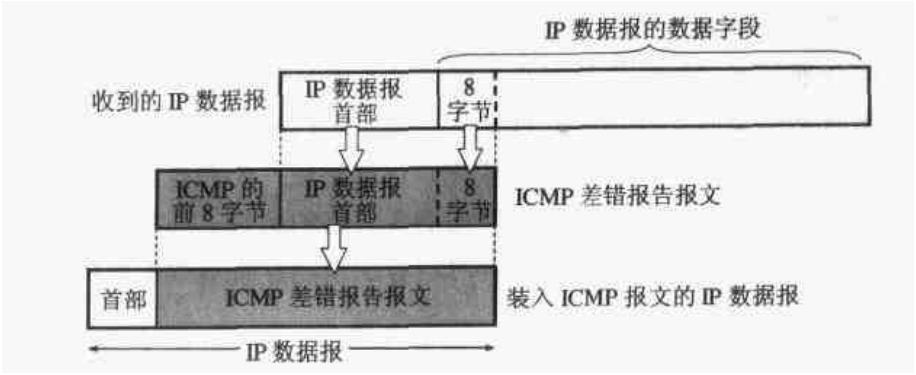


图 6-27　ICMP 差错报告报文的数据字段的内容

下面是不应发送 ICMP 差错报告报文的几种情况。

- 对 ICMP 差错报告报文不再发送 ICMP 差错报告报文。
- 对第一个分片的数据报片的所有后续数据报片都不发送 ICMP 差错报告报文。
- 对具有多播地址的数据报都不发送 ICMP 差错报告报文。
- 对具有特殊地址（如 127.0.0.0 或 0.0.0.0）的数据报不发送 ICMP 差错报告报文。

ICMP 询问报文有四种，即**回送请求和回答、时间戳请求和回答、掩码地址请求和回答**，以及**路由器询问和通告**。

ICMP 回送请求报文是由主机或路由器向一个特定的目的主机发出的询问。收到此报文的机器必须给源主机发送 ICMP 回送回答报文。这种询问报文用来测试目的站是否可达以及了解其有关状态。在应用层有一个很常用的服务叫做 **PING** (Packet InterNet Groper)，用来测

试两个主机之间的连通性[①]。PING 使用了 ICMP 回送请求与回送回答报文。PING 是应用层直接使用网络层 ICMP 的一个例子。它没有通过运输层的 TCP 或 UDP。

ICMP 时间戳请求报文是请某个主机或路由器回答当前的日期和时间。在 ICMP 时间戳回答报文中有一个 32 bit 的字段，其中写入的整数代表从 1900 年 1 月 1 日起到当前时刻一共有多少秒。时间戳请求与回答可用来进行时钟同步和测量时间。

主机使用 ICMP 地址掩码请求报文可向子网掩码服务器得到某个接口的地址掩码。

主机使用 ICMP 路由器询问和通告报文可了解连接在本网络上的路由器是否正常工作。主机将路由器询问报文进行广播（或多播）。收到询问报文的一个或几个路由器就使用路由器通告报文广播其路由选择信息。

6.5 因特网的路由选择协议

本节将讨论几种常用的路由选择协议，也就是要讨论路由表中的路由是怎样得出的。

6.5.1 有关路由选择协议的几个基本概念

1．理想的路由算法

路由选择协议的核心就是路由算法，即需要何种算法来获得路由表中的各项目。一个理想的路由算法应具有如下的一些特点[BELL86]：

（1）**算法必须是正确的和完整的**。这里，“正确”的含义是：沿着各路由表所指引的路由，分组一定能够最终到达的目的网络和目的主机。

（2）**算法在计算上应简单**。进行路由选择的计算必然要增加分组的时延。因此，路由选择的计算不应使网络通信量增加太多的额外开销。若为了计算合适的路由必须使用网络其他路由器发来的大量状态信息时，开销就会过大。

（3）**算法应能适应通信量和网络拓扑的变化，这就是说，要有自适应性**。当网络中的通信量发生变化时，算法能自适应地改变路由以均衡各链路的负载。当某个或某些结点、链路发生故障不能工作，或者修理好了再投入运行时，算法也能及时地改变路由。有时称这种自适应性为“**稳健性**”(robustness)。[②]

（4）**算法应具有稳定性**。在网络通信量和网络拓扑相对稳定的情况下，路由算法应收敛于一个可以接受的解，而不应使得出的路由不停地变化。

（5）**算法应是公平的**。这就是说，算法应对所有用户（除对少数优先级高的用户）都是平等的。例如，若使某一对用户的端到端时延为最小，但却不考虑其他的广大用户，这就明显地不符合公平性的要求。

（6）**算法应是最佳的**。这里的“最佳”是指以最低的代价来实现路由算法。这里特别需要注意的是，在研究路由选择时，需要给每一条链路指明一定的**代价**(cost，也可以将 cost 译为“费用”)，而这里“**代价**”**并不是指“钱”**，而是由一个或几个因素综合决定的一种度量

---

① 注：Windows 用户可在接入因特网后转入 MS DOS，再键入 PING www.xxx.yyy.zzz（这里 www.xxx.yyy.zzz 是被测试主机的 IP 地址域名），就可知道该主机是否可达，建议读者亲自实践一下。

② 注：Robustness 一词在自动控制界的标准译名是“鲁棒性”，但在[MINGCI94]则译为“稳健性”。

(metric)，如链路长度、数据率、链路容量、是否要保密、传播时延等，甚至还可以是一天中某一个小时内的通信量、结点的缓存被占用的程度、链路差错率等。可以根据用户的具体情况来设置每一条链路的“代价”。由此可见，不存在一种绝对的最佳路由算法。**所谓“最佳”只能是相对于某一种特定要求下得出的较为合理的选择而已。**

一个实际的路由选择算法，应尽可能接近于理想的算法。在不同的应用条件下，对以上提出的六个方面也可有不同的侧重。

应当指出，路由选择是个非常复杂的问题，因为它是网络中的所有结点共同协调工作的结果。其次，路由选择的环境往往是不断变化的，而这种变化有时无法事先知道，例如，网络中出了某些故障。此外，当网络发生拥塞时，就特别需要有能缓解这种拥塞的路由选择策略，但恰好在这种条件下，很难从网络中的各结点获得所需的路由选择信息。

倘若从路由算法能否随网络的通信量或拓扑自适应地进行调整变化来划分，则只有两大类，即**静态路由选择策略**与**动态路由选择策略**。静态路由选择也叫做**非自适应路由选择**，其特点是简单和开销较小，但不能及时适应网络状态的变化。动态路由选择也叫做**自适应路由选择**，其特点是能较好地适应网络状态的变化，但实现起来较为复杂，开销也比较大。

2．分层次的路由选择协议

因特网采用的路由选择协议主要是自适应的（即动态的）、分布式路由选择协议。由于以下两个原因，因特网采用分层次的路由选择协议：

（1）因特网的规模非常大，现在就已经有几百万个路由器互连在一起。如果让所有的路由器知道所有的网络应怎样到达，则这种路由表将非常大，处理起来也太花时间。而所有这些路由器之间交换路由信息所需的带宽就会使因特网的通信链路饱和。

（2）许多单位不愿意外界了解自己单位网络的布局细节和本部门所采用的路由选择协议（这属于本部门内部的事情），但同时还希望连接到因特网上。

为此，因特网将整个互联网划分为许多较小的**自治系统**(autonomous system)，一般简称为 AS。一个自治系统是一个互联网，其最重要的特点就是**自治系统有权自主地决定在本系统内应采用何种路由选择协议**。一个自治系统内的所有网络都属于一个行政单位（例如，一个公司，一所大学，政府的一个部门，等等）来管辖。但一个自治系统的所有路由器在本自治系统内都必须是连通的。如果一个部门管辖两个网络，但这两个网络要通过其他的主干网才能互连起来，那么这两个网络并不能构成一个自治系统。它们还是两个自治系统。这样，因特网就把路由选择协议划分为两大类，即：

（1）**内部网关协议** IGP (Interior Gateway Protocol)　即在一个自治系统内部使用的路由选择协议，而这与在互联网中的其他自治系统选用什么路由选择协议无关。目前这类路由选择协议使用得最多，如 RIP 和 OSPF 协议。

（2）**外部网关协议** EGP (External Gateway Protocol)　若源站和目的站处在不同的自治系统中（这两个自治系统使用不同的内部网关协议），当数据报传到一个自治系统的边界时，就需要使用一种协议将路由选择信息传递到另一个自治系统中。这样的协议就是外部网关协议 EGP。在外部网关协议中目前使用最多的是 BGP-4。

自治系统之间的路由选择也叫做**域间路由选择**(interdomain routing)，而在自治系统内部的路由选择叫做**域内路由选择**(intradomain routing)。

图 6-28 为三个自治系统互连在一起的示意图，在自治系统内各路由器之间的网络就省略了，而用一条链路表示路由器之间的网络。每个自治系统运行本自治系统的内部路由选择协议 IGP，但每个自治系统都有一个或多个路由器除运行本系统的内部路由选择协议外，还运行自治系统间的路由选择协议 EGP。在图 6-28 中，能运行自治系统间的路由选择协议的有 R_1, R_2 和 R_3 三个路由器。在图中将这类路由器画得稍大些以示区别。假定图中自治系统 A 的主机 H_1 要向自治系统 B 的主机 H_2 发送数据报，那么在各自治系统内使用的是各自的内部网关协议 IGP(例如，分别使用 RIP 和 OSPF)，而在路由器 R_1 和 R_2 之间则必须使用外部网关协议 EGP(例如，使用 BGP-4)。

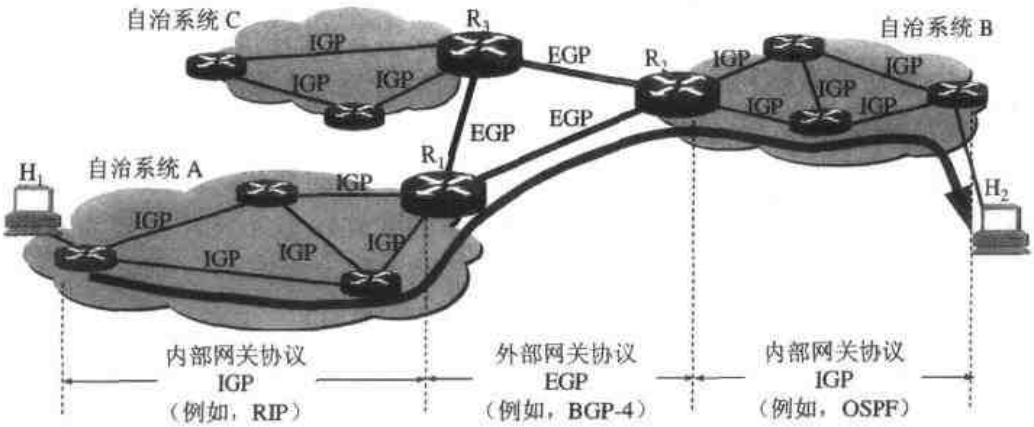


图 6-28　自治系统和内部网关协议、外部网关协议

这里我们要指出两点。

（1）因特网的早期 RFC 文档中未使用“路由器”而是使用“网关”这一名词。但是在新的 RFC 文档中又使用了“路由器”这一名词，因此有的书将 IGP 和 EGP 分别改为 IRP（内部路由器协议）和 ERP（外部路由器协议）。为了方便读者查阅 RFC 文档，本书仍使用 RFC 原先使用的名字 IGP 和 EGP。

（2）RFC 采用的这两个名词 IGP 和 EGP 是**协议类别**的名称。但 RFC 在使用 EGP 这个名词时出现了一点混乱，因为最早的一个外部网关协议的**协议名字**正好也是 EGP [RFC 827]。后来发现该 RFC 提出的 EGP 有不少缺点，就设计了一种更好的外部网关协议，叫做**边界网关协议** BGP (Border Gateway Protocol)，用来取代旧的 RFC 827 外部网关协议 EGP。实际上，旧的协议 EGP 和新的协议 BGP 都属于外部网关协议 EGP 这一类别。因此在遇到名词 EGP 时，应弄清它是指旧的[RFC 827]协议 EGP 还是指外部网关协议 EGP 这个类别。

总之，使用分层次的路由选择方法，可将因特网的路由选择协议划分为：

- **内部网关协议** IGP：具体的协议有多种，如 RIP 和 OSPF 等。
- **外部网关协议** EGP：目前使用的协议就是 BGP。

对于比较大的自治系统，还可将所有的网络再进行一次划分。例如，可以构筑一个链路速率较高的主干网和许多速率较低的区域网。每个区域网通过路由器连接到主干网。在一个区域内找不到目的站时，就通过路由器经过主干网到达另一个区域网，或者通过外部路由器到别的自治系统中去查找。下面对这两类协议分别进行介绍[HUIT-95]。

6.5.2 内部网关协议 RIP

1．工作原理

路由信息协议 RIP (Routing Information Protocol)是内部网关协议 IGP 中最先得到广泛使用的协议[RFC 1058]。RIP 是一种分布式的**基于距离向量的路由选择协议**，是因特网的标准协议，其最大优点就是简单。

RIP 协议要求网络中的每一个路由器都要维护从它自己到其他每一个目的网络的距离记录（因此，这是一组距离，即“距离向量”）。RIP 协议将“**距离**”定义如下：

从一路由器到直接连接的网络的距离定义为 1。从一路由器到非直接连接的网络的距离定义为所经过的路由器数加 1。“加 1”是因为到达目的网络后就进行直接交付，而到直接连接的网络的距离已经定义为 1。例如在前面讲过的图 6-18 中，路由器 R_1 到网 1 或网 2 的距离都是 1（直接连接），而到网 3 的距离是 2，到网 4 的距离是 3。

RIP 协议的“距离”也称为“**跳数**” (hop count)[①]，因为每经过一个路由器，跳数就加 1。RIP 认为一个好的路由就是它通过的路由器的数目少，即“距离短”。RIP 允许一条路径最多只能包含 15 个路由器。因此“距离”的最大值为 16 时即相当于不可达。可见 **RIP 只适用于小型互联网**。

需要注意的是，到直接连接的网络的距离也可定义为 0（采用这种定义的理由是：路由器在和直接连接在该网络上的主机通信时不需要经过另外的路由器。既然每经过一个路由器要将距离加 1，那么不再经过路由器的距离就应当为零）。作者编写的其他版本的教材**过去也曾使用过这种定义**。但两种不同的定义对实现 RIP 协议并无影响，因为重要的是要找出最短距离，将所有的距离都加 1 或减 1 对选择最佳路由都是一样的。

RIP 不能在两个网络之间同时使用多条路由。RIP 选择一个具有最少路由器的路由（即最短路由），哪怕还存在另一条高速（低时延）但路由器较多的路由。

本节讨论的 RIP 协议和下一节要讨论的 OSPF 协议都是分布式路由选择协议。它们的共同特点就是每一个路由器都要不断地和其他一些路由器交换路由信息。我们一定要弄清以下三个要点，即**和哪些路由器交换信息？交换什么信息？在什么时候交换信息？**

（1）**仅和相邻路由器交换信息**。两个路由器是相邻的，如果它们之间的通信不需要经过另一个路由器。换言之，两个相邻路由器在同一个网络上都有自己的接口。RIP 协议规定，不相邻的路由器不交换信息。

（2）交换的信息是**当前本路由器所知道的全部信息，即自己的路由表**。因此，交换的信息就是：“到本自治系统中所有网络的(最短)距离，以及到每个网络应经过的下一跳路由器”。至于本路由器怎样获得这些信息以及路由表是否完整，都是不重要的。

（3）**按固定的时间间隔**交换路由信息，例如，每隔 30 秒。然后路由器根据收到的路由信息更新路由表。当网络拓扑发生变化时，路由器也及时向相邻路由器通告拓扑变化后的路由信息。

这里要强调一点：路由器在**刚刚开始工作**时，只知道到直接连接的网络的距离（此距离定义为 1）。以后，每一个路由器也只和**数目非常有限**的相邻路由器交换并更新路由信息。但经过若干次的更新后，所有的路由器最终都会知道到达本自治系统中任何一个网络的最短距

① 注：这里的“距离”实际上指的是“最短距离”，但为方便起见往往省略“最短”二字。

离和下一跳路由器的地址。看起来 RIP 协议有些奇怪，因为“我的路由表中的信息要依赖于你的，而你的信息又依赖于我的。”然而事实证明，RIP 协议的收敛(convergence)过程较快。所谓收敛就是在自治系统中所有的结点都得到正确的路由选择信息的过程。

路由表中最主要的信息就是：到某个网络的距离（即最短距离），以及应经过的下一跳地址。路由表更新的原则是找出到每个目的网络的最短距离。这种更新算法又称为距离向量算法。下面就是 RIP 协议使用的距离向量算法。

2．距离向量算法

收到相邻路由器（其地址为 X）的一个 RIP 报文：

（1）先修改此 RIP 报文中的所有项目：将“下一跳”字段中的地址都改为 X，并将所有的“距离”字段的值加 1（见后面的解释 1）。

（2）对修改后的 RIP 报文中的每一个项目，重复以下步骤：

若项目中的目的网络不在路由表中，则将该项目添加到路由表中（见解释 2）。

否则

若下一跳字段给出的路由器地址是同样的，则将收到的项目替换原路由表中的项目（见解释 3）。

否则

若收到的项目中的距离小于路由表中的距离，则进行更新（见解释 4），

否则，什么也不做

（3）若 3 分钟还没有收到相邻路由器的更新路由表，则将此相邻路由器记为不可达的路由器，即将距离置为 16（距离为 16 表示不可达）。

（4）返回。

上面给出的距离向量算法的基础就是 Bellman-Ford 算法（或 Ford-Fulkerson 算法）。这种算法的要点是这样的：设 X 是结点 A 到 B 的最短路径上的一个结点。若将路径 A→B 拆成两段路径 A→X 和 X→B，则每一段路径 A→X 和 X→B 也都分别是结点 A 到 X 和结点 X 到 B 的最短路径。

下面是对上述距离向量算法的 4 点解释。

解释 1：这样做是为了便于进行本路由表的更新。设从位于地址 X 的相邻路由器发来的 RIP 报文的某一个项目是：“Net2, 3, Y”，意思是“我到网络 Net2 的距离是 3，要经过的下一跳路由器的地址是 Y”，那么本路由器就可推断出：“若我将下一跳路由器选为在地址 X 的路由器，则我到网络 Net2 的距离应为 3 + 1 = 4”。于是，本路由器就将收到的 RIP 报文的这一个项目修改为“Net2, 4, X”，作为下一步进行比较时使用（只有和路由表中原有的项目比较后才能知道是否需要更新）。读者可注意到，收到的项目中的 Y 对本路由器是没有用的，因为 Y 不是本路由器的路由的下一跳路由器地址。

解释 2：表明这是新的目的网络，应当加入到路由表中。例如，本路由表中没有到目的网络 Net2 的路由，那么在路由表中就要加入新的项目“Net2, 4, X”。

解释 3：为什么要替换呢？因为这是最新的消息，要以最新的消息为准。到目的网络的距离有可能增大或减小，但也可能没有改变。例如，不管原来路由表中的项目是“Net2, 3, X”还是“Net2, 5, X”，都要更新为现在的“Net2, 4, X”。

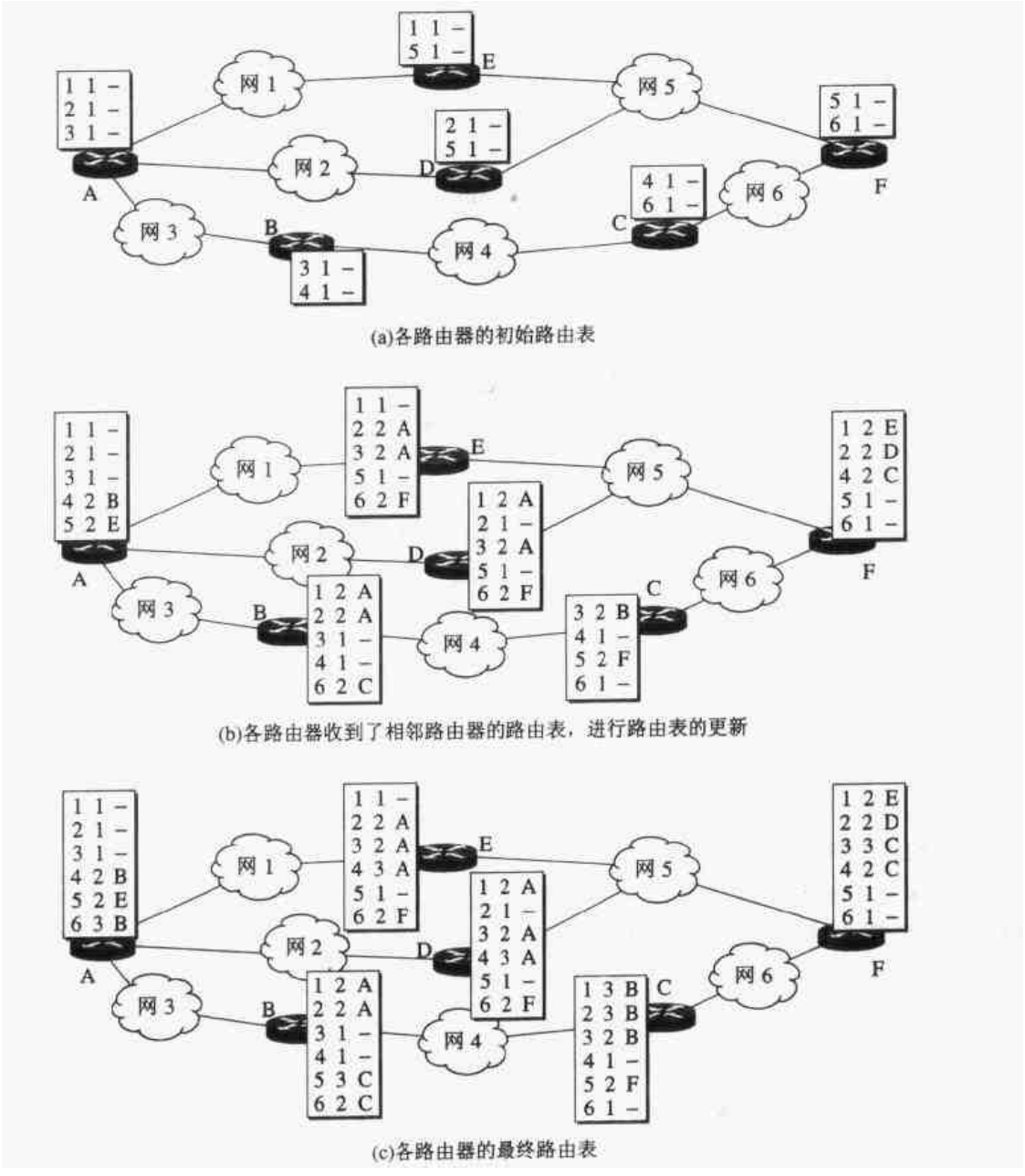
解释 4：例如，若路由表中已有项目“Net2, 5, P”，就要更新为“Net2, 4, X”。因为更新

后到网络 Net2 的距离更短了（从 5 减到 4）。

RIP 协议让互联网中的所有路由器都和自己的相邻路由器不断交换路由信息，并不断更新其路由表，使得从**每一个路由器到每一个目的网络的路由都是最短的**（即跳数最少）。这里还应注意：虽然所有的路由器最终都拥有了整个自治系统的全局路由信息，但由于每一个路由器的位置不同，它们的路由表当然也应当是不同的。

图 6-29 说明了使用 RIP 协议的各路由器，其路由表的初始数据和最终数据。

图 6-29(a)给出了一个简单的网络拓扑，共有 6 个网络（网 1 至网 6），通过 6 个路由器(A 至 F)互连起来。当 RIP 协议刚刚开始工作时，各路由器的路由表中的内容如图(a)所示。路由表中的每一行都包括三个字符，它们从左到右分别代表：**目的网络**，从本路由器到该目的网络的**跳数**（即最短距离），以及**下一跳路由器**（如果是直接交付就不经过路由器而是用一条短横线表示）。在初始状态下，路由表的行数取决于该路由器与多少个网络直接相连。



(a)各路由器的初始路由表

(b)各路由器收到了相邻路由器的路由表，进行路由表的更新

(c)各路由器的最终路由表

图 6-29　RIP 协议的工作原理

图 6-29(b)是各路由器收到了相邻路由器的路由表，进行路由表更新后的情况（假定所有的路由器都同时向其相邻的路由器发送自己的路由表）。图 6-29(c)是各路由器再更新一次路由表后得出最终的路由表中的内容。这些路由表中的每一行都指出：到某个网络的距离是多少，以及下一跳是哪一个路由器。对于更复杂的网络，路由表要经过更多次的更新才能达到最终的数值。有时，到达同一个目的网络可以经过不同的下一跳路由器（但跳数是同样的）。这时可任选一个下一跳路由器。

RIP 协议使用运输层的用户数据报 UDP 进行传送（使用 UDP 的端口 520。端口的意义见 7.2.2 节）。因此 RIP 协议的位置应当在应用层。但转发 IP 数据报的过程是在网络层完成的。

3．RIP 协议的报文格式

现在较新的 RIP 版本是 1998 年 11 月公布的 RIP2 [RFC 2453]（已成为因特网标准协议），新版本协议本身并无多大变化，但性能上有些改进。RIP2 可以支持变长子网掩码和 CIDR。此外，RIP2 还提供简单的鉴别过程支持多播。

图 6-30 是 RIP2 的报文格式，它和 RIP1 的首部相同，但后面的路由部分不一样。

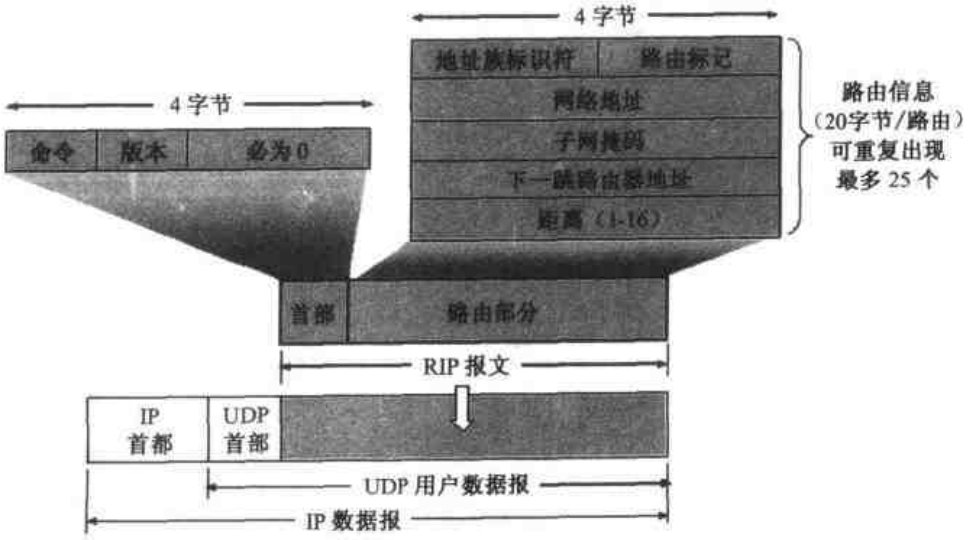


图 6-30　RIP2 的报文格式

RIP 报文由首部和路由部分组成。

RIP 的首部占 4 个字节，其中的命令字段指出报文的意义。例如，1 表示请求路由信息，2 表示对请求路由信息的响应或未被请求而发出的路由更新报文。首部后面的“必为 0”是为了 4 字节字的对齐。

RIP2 报文中的路由部分由若干个路由信息组成。每个路由信息需要用 20 个字节。**地址族标识符**（又称为**地址类别**）字段用来标志所使用的地址协议。如采用 IP 地址就令这个字段的值为 2（原来考虑 RIP 也可用于其他非 TCP/IP 协议的情况）。**路由标记**填入自治系统的号码，这是考虑使 RIP 有可能收到本自治系统以外的路由选择信息。再后面指出某**个网络地址**、该网络的**子网掩码**、**下一跳路由器地址**以及**到此网络的距离**。一个 RIP 报文最多可包括 25 个路由，因而 RIP 报文的最大长度是 $4 + 20 \times 25 = 504$ 字节。如超过，必须再用一个 RIP 报文来传送。

RIP2 还具有简单的鉴别功能。若使用鉴别功能，则将原来写入第一个路由信息（20 字

节）的位置用作鉴别。这时应将地址族标识符置为全 1（即 0xFFFF），而路由标记写入鉴别类型，剩下的 16 字节为鉴别数据。在鉴别数据之后才写入路由信息，但这时最多只能再放入 24 个路由信息。

RIP 存在的一个问题是当网络出现故障时，要经过比较长的时间才能将此信息传送到所有的路由器。我们可以用图 6-31 的简单例子来说明。设三个网络通过两个路由器互连起来。都已建立了各自的路由表。路由表只给出了一行我们感兴趣的内容。每一行的三个字符的意思和图 6-29 的约定相同。因此，路由器 R_1 中的“1, 1, –”表示“到网 1 的距离是 1，直接交付”。路由器 R_2 中的“1, 2, R_1”表示“到网 1 的距离是 2，下一跳经过 R_1”。

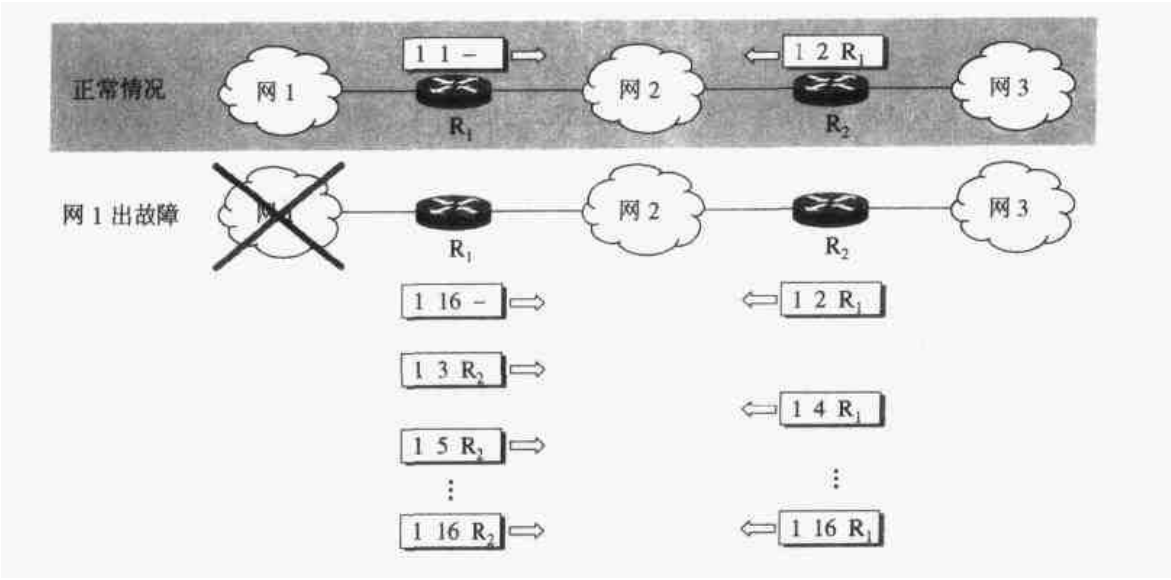


图 6-31　RIP 协议的缺点：坏消息传播得慢

现在假定路由器 R_1 到网 1 的链路出了故障，R_1 无法到达网 1。于是路由器 R_1 将到网 1 的距离改为 16（16 就表示到网 1 不可达），因而在 R_1 的路由表中的相应项目变为“1, 16, –”。但是，很可能要经过 30 秒钟后 R_1 才将此更新信息发送给 R_2。然而 R_2 可能已将自己的路由表发送给 R_1，其中有“1, 2, R_1”这一项。

R_1 收到 R_2 的更新报文后，误认为可经过 R_2 到达网 1，于是将收到的路由信息“1, 2, R_1”修改为：“1, 3, R_2”，表明“我到网 1 的距离是 3，下一跳经过 R_2”。R_1 用“1, 3, R_2”更新路由表中的项目“1, 16, –”，并将此更新信息发送给 R_2。

同理，R_2 以后又更新自己的路由表为“1, 4, R_1”，表明 “我到网 1 距离是 4，下一跳经过 R_1”。

这样不断更新下去，直到 R_1 和 R_2 到网 1 的距离都增大到 16 时，R_1 和 R_2 才知道网 1 是不可达的。RIP 协议的这一特点叫做：**好消息传播得快，而坏消息传播得慢**。网络出故障的传播时间往往需要较长的时间（例如数分钟），这是 RIP 的一个主要缺点。

但如果一个路由器发现了更短的路由，那么这种更新信息就传播得很快。

为了使坏消息传播得更快些，可以采取多种措施。例如，让路由器记录收到某特定路由信息的接口，而不让同一路由信息再通过此接口向反方向传送。

总之，RIP 协议最大的优点就是**实现简单，开销较小**。但 RIP 协议的缺点也较多。首先，RIP 限制了网络的规模，它能使用的最大距离为 15（16 表示不可达）。其次，路由器

之间交换的路由信息是路由器中的完整路由表，因而随着网络规模的扩大，开销也就增加。最后，“坏消息传播得慢”，使更新过程的收敛时间过长。因此，对于规模较大的网络就应当使用下一节所述的 OSPF 协议。然而目前在规模较小的网络中，使用 RIP 协议的仍占多数。

6.5.3 内部网关协议 OSPF

1．OSPF 协议的基本特点

这个协议的名字是**开放最短路径优先** OSPF (Open Shortest Path First)。它是为克服 RIP 的缺点在 1989 年开发出来的。OSPF 的原理很简单，但实现起来却较复杂。“**开放**”表明 OSPF 协议不是受某一家厂商控制，而是公开发表的。“**最短路径优先**”是因为使用了 Dijkstra 提出的**最短路径算法**（见附录 E）。OSPF 的第二个版本 OSPF2 已成为因特网标准协议[RFC 2328]（OSPF2 的文档长达 224 页，而 RIP2 的文档才 38 页）。关于 OSPF 可参阅专著[MOY98], [HUIT95]。

请注意：OSPF 只是一个协议的名字，**它并不表示其他的路由选择协议不是“最短路径优先”**。实际上，所有的在自治系统内部使用的路由选择协议（包括 RIP 协议）都是要寻找一条最短的路径。

OSPF 最主要的特征就是使用分布式的**链路状态协议**(link state protocol)，而不是像 RIP 那样的距离向量协议。和 RIP 协议相比，OSPF 的三个要点和 RIP 的都不一样：

（1）向本自治系统中**所有路由器**发送信息。这里使用的方法是**洪泛法**(flooding)，这就是路由器通过所有输出端口向所有相邻的路由器发送信息。而每一个相邻路由器又再将此信息发往其所有的相邻路由器（但不再发送给刚刚发来信息的那个路由器）。这样，最终整个区域中所有的路由器都得到了这个信息的一个副本。更具体的做法后面还要讨论。我们应注意，RIP 协议是仅仅向自己相邻的几个路由器发送信息。

（2）发送的信息就是与本路由器相邻**的所有路由器**的**链路状态**，但这只是路由器所知道的**部分信息**。所谓“链路状态”就是说明本路由器都和哪些路由器相邻[①]，以及该链路的“度量”(metric)。OSPF 将这个“**度量**”用来表示费用、距离、时延、带宽等等。这些都由网络管理人员来决定，因此较为灵活。有时为了方便就称这个度量为“**代价**”（参见 6.5.1 节）。我们应注意，对于 RIP 协议，发送的信息是：“到所有网络的距离和下一跳路由器”。

（3）只有当链路状态**发生变化时**，路由器才用洪泛法向所有路由器发送此信息。而不像 RIP 那样，不管网络拓扑有无发生变化，路由器之间都要定期交换路由表的信息。

从上述的三个方面可以看出，OSPF 和 RIP 的工作原理相差较大。

由于各路由器之间频繁地交换链路状态信息，因此所有的路由器最终都能建立一个**链路状态数据库**(link-state database)，这个数据库实际上就是**全网的拓扑结构图**。这个拓扑结构图在全网范围内是**一致的**（这称为**链路状态数据库的同步**）。因此，每一个路由器都知道全网共有多少个路由器，以及哪些路由器是相连的，其代价是多少，等等。每一个路由器使用链路状态数据库中的数据，构造出自己的路由表（例如，使用 Dijkstra 的最短路径路由算法）。我们注意到，RIP 协议的每一个路由器虽然知道到所有的网络的距离以及下一跳路由器，但却

① 注：在前面我们已经说过，在讨论路由器之间是如何交换路由信息时，最好将路由器之间的网络简化为一条链路。OSPF 的“链路状态”中的“链路”实际上就是指“和这两个路由器都有接口的网络”。

不知道全网的拓扑结构（只有到了下一跳路由器，才能知道再下一跳应当怎样走）。

OSPF 的链路状态数据库能较快地进行更新，使各个路由器能及时更新其路由表。OSPF 的**更新过程收敛得快**是其重要优点。

为了使 OSPF 能够用于规模很大的网络，OSPF 将一个自治系统再划分为若干个更小的范围，叫作**区域**(area)。图 6-32 就表示一个自治系统划分为 4 个区域。每一个区域都有一个 32 bit 的区域标识符（用点分十进制表示）。当然，一个区域也不能太大，在一个区域内的路由器最好不超过 200 个。

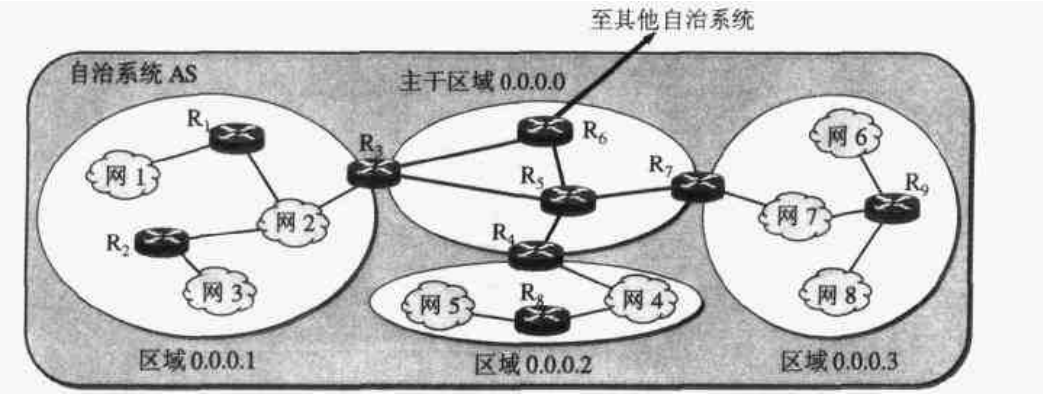


图 6-32　OSPF 划分为两种不同的区域

划分区域的好处就是将利用洪泛法交换链路状态信息的范围局限于每一个区域而不是整个的自治系统，这就减少了整个网络上的通信量。在一个区域内部的路由器只知道本区域的完整网络拓扑，而不知道其他区域的网络拓扑的情况。

为了使每一个区域能够和本区域以外的区域进行通信，OSPF 使用**层次结构的区域划分**。在上层的区域叫作**主干区域**(backbone area)。主干区域的标识符规定为 0.0.0.0。主干区域的作用是用来连通其他在下层的区域。从其他区域来的信息都由**区域边界路由器**(area border router)进行概括。在图 6-32 中，路由器 R_3、R_4 和 R_7 都是区域边界路由器，而显然，每一个区域至少应当有一个区域边界路由器。在主干区域内的路由器叫做**主干路由器**(backbone router)，如 R_3、R_4、R_5、R_6 和 R_7。一个主干路由器可以同时是区域边界路由器，如 R_3、R_4 和 R_7。在主干区域内还要有一个路由器专门和本自治系统外的其他自治系统交换路由信息。这样的路由器叫作**自治系统边界路由器**（如图中的 R_6）。

采用分层次划分区域的方法虽然使交换信息的种类增多了，同时也使 OSPF 协议更加复杂了。但这样做却能使每一个区域内部交换路由信息的通信量大大减小，因而使 OSPF 协议能够用于规模很大的自治系统中。这里我们再一次地看到划分层次在网络设计中的重要性。

OSPF 不用 UDP 而是**直接用 IP 数据报传送**（其 IP 数据报首部的协议字段值为 89），可见 OSPF 的位置在网络层。OSPF 构成的数据报很短。这样做可减少路由信息的通信量。数据报很短的另一好处是可以不必将长的数据报分片传送。分片传送的数据报只要丢失一个，就无法组装成原来的数据报，而整个数据报就必须重传。

OSPF 分组使用 24 字节的固定长度首部（见图 6-33），分组的数据部分可以是 5 种类型分组中的一种。下面简单介绍 OSPF 首部各字段的意义。

（1）**版本**　　当前的版本号是 2。

（2）**类型**　可以是 5 种类型分组中的一种。

（3）**分组长度**　包括 OSPF 首部在内的分组长度，以字节为单位。

（4）**路由器标识符**　标志发送该分组的路由器的接口的 IP 地址。

（5）**区域标识符**　分组属于的区域的标识符。

（6）**检验和**　用来检测分组中的差错。

（7）**鉴别类型**　目前只有两种：0（不用）和 1（口令）。

（8）**鉴别**　鉴别类型为 0 时就填入 0。鉴别类型为 1 则填入 8 个字符的口令。

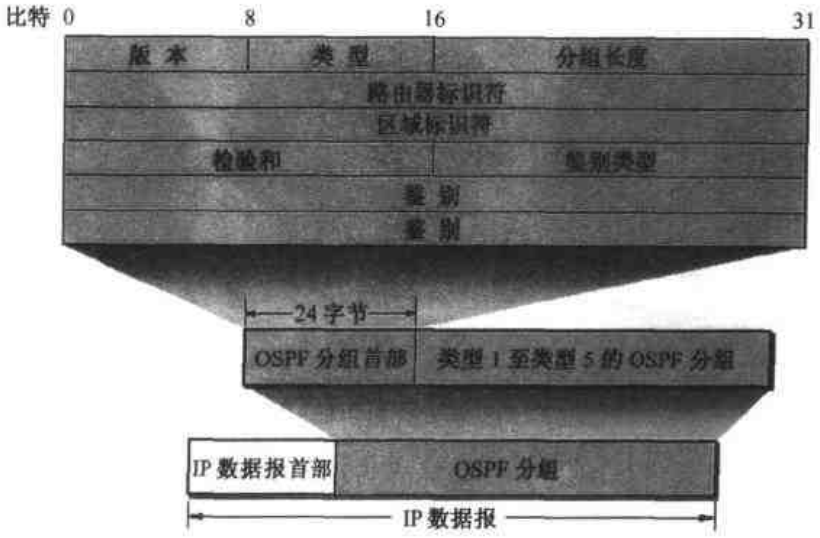


图 6-33　OSPF 分组用 IP 数据报传送

除了以上的几个基本特点外，OSPF 还具有下列的一些特点：

（1）OSPF 对不同的链路可根据 IP 分组的不同服务类型 TOS 而设置成不同的代价。例如，高带宽的卫星链路对于非实时的业务可设置为较低的代价，但对于时延敏感的业务就可设置为非常高的代价。因此，OSPF **对于不同类型的业务可计算出不同的路由**。链路的代价可以是 1 至 65535 中的任何一个无量纲的数，因此十分灵活。商用的 OSPF 实现通常是根据链路带宽来计算链路的代价。这种灵活性是 RIP 所没有的。

（2）如果到同一个目的网络有多条相同代价的路径，那么可以将通信量分配给这几条路径。这叫作多路径间的**负载平衡**(load balancing)。在代价相同的多条路径上分配通信量是流量工程中的简单形式。RIP 只能找出到某个网络的一条路径。

（3）所有在 OSPF 路由器之间交换的分组（例如，链路状态更新分组）都具有鉴别的功能，因而保证了仅在可信赖的路由器之间交换链路状态信息。

（4）OSPF 支持可变长度的子网划分和无分类的编址 CIDR。

（5）由于网络中的链路状态可能经常发生变化，因此 OSPF 让每一个链路状态都带上一个 32 bit 的**序号**，序号越大状态就越新。OSPF 规定，链路状态序号增长的速率不得超过每 5 秒钟 1 次。这样，全部序号空间在 600 年内不会产生重复号。

2．OSPF 的五种分组类型

OSPF 共有以下五种分组类型：

（1）**类型 1，问候**(Hello)分组，用来发现和维持邻站的可达性。

（2）**类型 2，数据库描述**(Database Description)分组，向邻站给出自己的链路状态数据库中的所有链路状态项目的摘要信息。

（3）**类型 3，链路状态请求**(Link State Request)分组，向对方请求发送某些链路状态项目的详细信息。

（4）**类型 4，链路状态更新**(Link State Update)分组，用洪泛法对全网更新链路状态。这种分组是最复杂的，也是 OSPF 协议最核心的部分。路由器使用这种分组将其链路状态通知给邻站。链路状态更新分组共有五种不同的链路状态，这里从略。

（5）**类型 5，链路状态确认**(Link State Acknowledgment)分组，对链路更新分组的确认。

OSPF 规定，每两个相邻路由器每隔 10 秒钟要交换一次问候分组。这样就能确知哪些邻站是可达的。对相邻路由器来说，“可达”是最基本的要求，因为只有可达邻站的链路状态信息才存入链路状态数据库（路由表就是根据链路状态数据库计算出来的）。在正常情况下，网络中传送的绝大多数 OSPF 分组都是问候分组。若有 40 秒钟没有收到某个相邻路由器发来的问候分组，则可认为该相邻路由器是不可达的，应立即修改链路状态数据库，并重新计算路由表。

其他的四种分组都是用来进行链路状态数据库的同步。所谓同步就是指不同路由器的链路状态数据库的内容是一样的。两个同步的路由器叫做“**完全邻接的**”(fully adjacent)路由器。不是完全邻接的路由器表明它们虽然在物理上是相邻的，但其链路状态数据库并没有达到一致。

当一个路由器刚开始工作时，它只能通过问候分组得知它有哪些相邻的路由器在工作，以及将数据发往相邻路由器所需的“代价”。如果所有的路由器都把自己的本地链路状态信息对全网进行广播，那么各路由器只要将这些链路状态信息综合起来就可得出链路状态数据库。但这样做开销太大，因此 OSPF 采用另外的办法。

OSPF 让每一个路由器用数据库描述分组和相邻路由器交换本数据库中已有的链路状态摘要信息。摘要信息主要就是指出有哪些路由器的链路状态信息（以及其序号）已经写入了数据库。经过与相邻路由器交换数据库描述分组后，路由器就使用链路状态请求分组，向对方请求发送自己所缺少的某些链路状态项目的详细信息。通过一系列的这种分组交换，全网同步的链路数据库就建立了。图 6-34 给出了 OSPF 的基本操作，说明了两个路由器需要交换各种类型的分组。

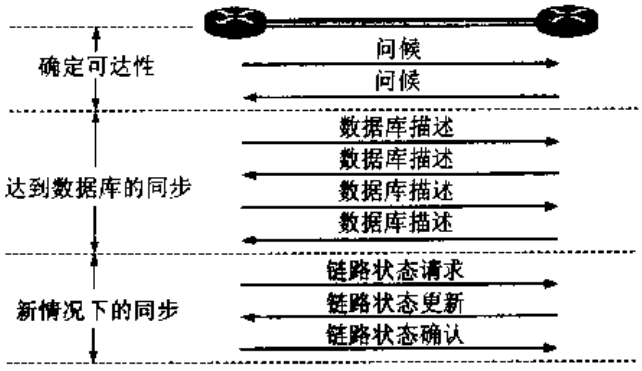


图 6-34　OSPF 的基本操作

在网络运行的过程中，只要一个路由器的链路状态发生变化，该路由器就要使用链路状态更新分组，用洪泛法对全网更新链路状态。OSPF 使用的是**可靠的洪泛法**，其要点见图 6-35 所示。设路由器 R 用洪泛法发出链路状态更新分组。图中用小箭头表示更新分组。第一次先

发给相邻的三个路由器。这三个路由器将收到的分组再进行转发时，要将其上游路由器除外。可靠的洪泛法是在收到更新分组后要发送确认。确认的发送故意迟延一些时间，为的是希望可以少发送几个确认分组。图中的空心箭头表示确认分组。

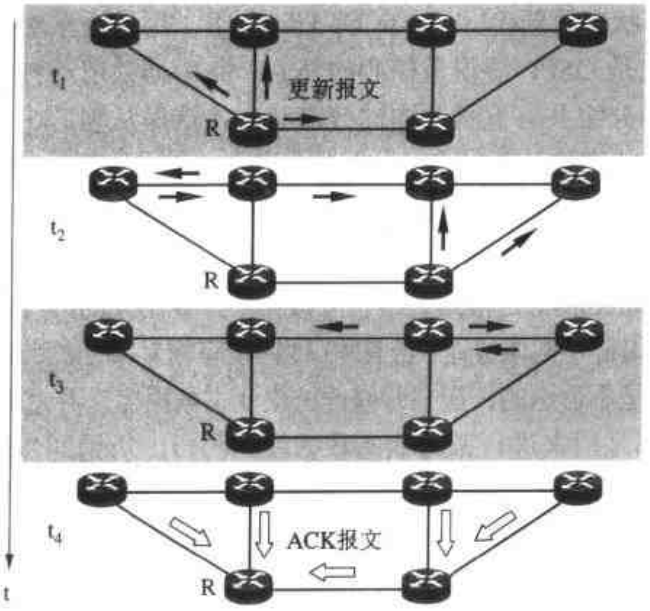


图 6-35　用可靠的洪泛法发送更新分组

为了确保链路状态数据库与全网的状态保持一致，OSPF 还规定每隔一段时间，如 30 分钟，要刷新一次数据库中的链路状态。

由于一个路由器的链路状态只涉及到与相邻路由器的连通状态，因而与整个互联网的规模并无直接关系。因此当互联网规模很大时，OSPF 协议要比距离向量协议 RIP 好得多。由于 OSPF 没有“坏消息传播得慢”的问题，据统计，其响应网络变化的时间小于 100 ms。

若 N 个路由器连接在一个以太网上，则每个路由器要向其他$(N-1)$个路由器发送链路状态信息，因而共有$(N-1)^2$ 个链路状态要在这个以太网上传送。OSPF 协议对这种多点接入的局域网采用了**指定的路由器**(designated router)的方法，使广播的信息量大大减少。指定的路由器代表该局域网上所有的链路向连接到该网络上的各路由器发送状态信息。

OSPF 支持三种网络的连接：

（1）两个路由器之间的点对点连接；

（2）具有广播功能的局域网；

（3）无广播功能的广域网。

每一种网络都可能带有多个路由器。图 6-36(a)是包括这三种网络连接的一个自治系统的例子。每一条链路的旁边标注了代价（或距离、时延等）。图中没有画出主机，因为主机一般都不运行 OSPF 协议。

通过各路由器之间的交换链路状态信息，可得出该互联网的链路状态数据库。实际的数据库是一个表，但我们可以用图 6-36(b)所示的**有向图**表示该数据库。其中每一个路由器、局域网或广域网都抽象为一个结点，而每条链路则用两条不同方向的边表示。OSPF 规定，从网络到路由器的代价为 0，不标注在图上。每个路由器中的路由表可从这个链路状态数据库

导出。例如，要计算路由器 F 的路由表，可先算出如图 6-37 所示的以 F 为根的最短路径树。根据最短路径树就很容易地得出路由表来。

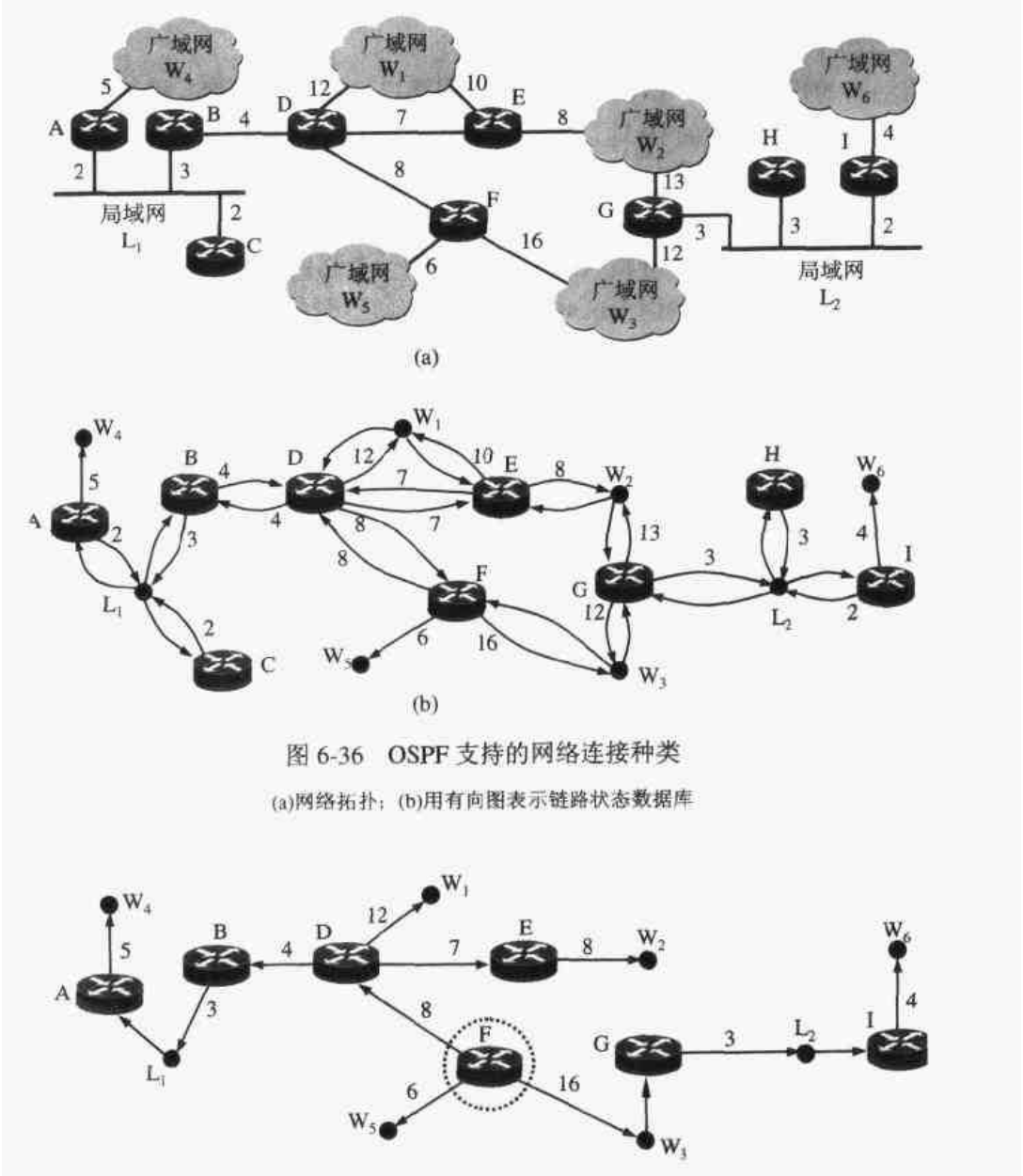


图 6-36　OSPF 支持的网络连接种类

(a)网络拓扑；(b)用有向图表示链路状态数据库

图 6-37　以路由器 F 为根的最短路径树（图 6-36 的例子）

目前大多数路由器厂商都支持 OSPF，并开始在一些网络中取代旧的 RIP。而链路状态路由选择协议也已用在其他的一些非 TCP/IP 体系中，如 NetWare 的 NLSP，IBM 的 APPN，以及 ATM 论坛的 PNNI 路由选择协议。

6.5.4　外部网关协议 BGP

1989 年，公布了新的外部网关协议——**边界网关协议** BGP。BGP 是**不同自治系统的路由器之间交换路由信息**的协议。BGP 的较新版本是 1995 年发表的 BGP-4 [RFC 1771-1772]，它已成为因特网草案标准协议。本书后面都将 BGP-4 简写为 BGP。

我们首先应当弄清，在不同自治系统之间的路由选择为什么不能使用前面讨论过的内部

网关协议，如 RIP 或 OSPF?

我们知道，内部网关协议（如 RIP 或 OSPF）主要是设法使数据报在一个自治系统中尽可能有效地从源站传送到目的站。在一个自治系统内部并不需要考虑其他方面的策略。然而 BGP 使用的环境却不同。这主要是因为以下的三个原因：

第一，**因特网的规模太大，使得自治系统之间路由选择非常困难**。连接在因特网主干网上的路由器，必须对任何有效的 IP 地址都能在路由表中找到匹配的目的网络。目前主干网路由器中的路由表的项目数早已超过了 5 万个网络前缀。这些网络的性能相差很大。如果用最短距离（即最少跳数）找出来的路径，可能并不是应当选用的路径。例如，有的路径的使用代价很高或很不安全。如果使用链路状态协议，则每一个路由器必须维持一个很大的链路状态数据库。对于这样大的主干网用 Dijkstra 算法计算最短路径时花费的时间也太长。

第二，**对于自治系统之间的路由选择，要寻找最佳路由是很不现实的**。由于各自治系统是运行自己选定的内部路由选择协议，使用本自治系统指明的路径代价，因此，当一条路径通过几个不同的自治系统时，要想对这样的路径计算出有意义的代价是不可能的。例如，对某个自治系统来说，代价为 1000 可能表示一条比较长的路由。但对另一个自治系统代价为 1000 却可能表示不可接受的坏路由。因此，自治系统之间的路由选择只可能交换“可达性”信息（即“可到达”或“不可到达”）。例如，告诉相邻路由器：“到达目的网络 N 可经过自治系统 A”。

第三，**自治系统之间的路由选择必须考虑有关策略**。例如，自治系统 A 要发送数据报到自治系统 B，本来最好是经过自治系统 C。但自治系统 C 不愿意让这些数据报通过本系统的网络，因为“这是他们的事情，和我们没有关系。”但另一方面，自治系统 C 愿意让某些相邻的自治系统的数据报通过自己的网络，特别是对那些付了服务费的某些自治系统更是如此。因此，自治系统之间的路由选择协议应当允许使用多种路由选择策略。这些策略包括政治、安全或经济方面的考虑。例如：我国国内的站点在互相传送数据报时不应经过国外兜圈子，特别是，不要经过某些对我国的安全有威胁的国家。这些策略都是由网络管理人员对每一个路由器进行设置的，但这些策略并不是自治系统之间的路由选择协议本身。还可举出一些策略的例子，如：“仅在到达下列这些地址时才经过自治系统 A”，“自治系统 A 和 B 相比时应优先通过 A”，等等。显然，使用这些策略是为了找出较好的路径而不是最佳路径。

由于上述情况，边界网关协议 BGP 只能是力求寻找一条能够到达目的网络且比**较好**的路由（不能兜圈子），而**并非要寻找一条最佳路由**。BGP 采用了**路径向量**(path vector)**路由选择协议**，它与距离向量协议和链路状态协议都有很大的区别。

在配置 BGP 时，每一个自治系统的管理员要选择至少一个路由器作为该自治系统的“**BGP 发言人**”[①]。一般说来，两个 BGP 发言人都是通过一个共享网络连接在一起的，而 BGP 发言人往往就是 BGP **边界路由器**，但也可以不是 BGP 边界路由器。

一个 BGP 发言人与其他自治系统中的 BGP 发言人要交换路由信息，就要先建立 TCP 连接（端口号为 179），然后在此连接上交换 BGP 报文以建立 **BGP 会话**(session)，利用 BGP 会话交换路由信息，如增加了新的路由，或撤消过时的路由，以及报告出差错的情况等等。使用 TCP 连接能提供可靠的服务，也简化了路由选择协议。使用 TCP 连接交换路由信息的两

① 注：[RFC 1771]使用了一个新名词——BGP speaker（BGP 发言人）。“BGP 发言人”表明该路由器可以代表整个自治系统和其他自治系统交换路由信息。虽然 BGP 协议允许使用任何其他台计算机用作 BGP 发言人，但大多数的自治系统实际上是在一个路由器上运行 BGP 协议的。

个 BGP 发言人，彼此成为对方的邻站(neighbor)或对等站(peer)。

图 6-38 表示 BGP 发言人和自治系统 AS 的关系的示意图。在图中画出了三个自治系统中的五个 BGP 发言人。每一个 BGP 发言人除了必须运行 BGP 协议外，还必须运行该自治系统所使用的内部网关协议，如 OSPF 或 RIP。

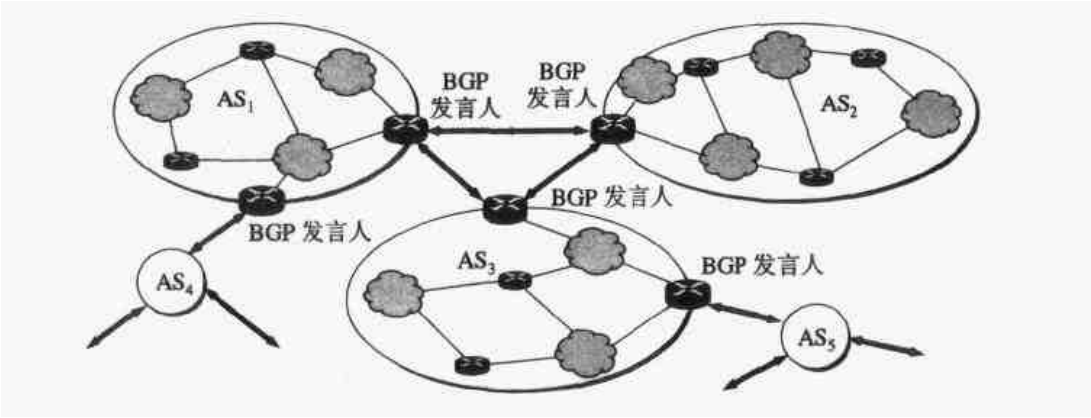


图 6-38　BGP 发言人和自治系统 AS 的关系

BGP 所交换的网络可达性的信息就是要到达某个网络（用网络前缀表示）所要经过的一系列的自治系统。当 BGP 发言人互相交换了网络可达性的信息后，各 BGP 发言人就根据所采用的策略从收到的路由信息中找出到达各自治系统的比较好的路由。图 6-39 表示一个 BGP 发言人构造出的自治系统连通图，它是树形结构，不存在回路。

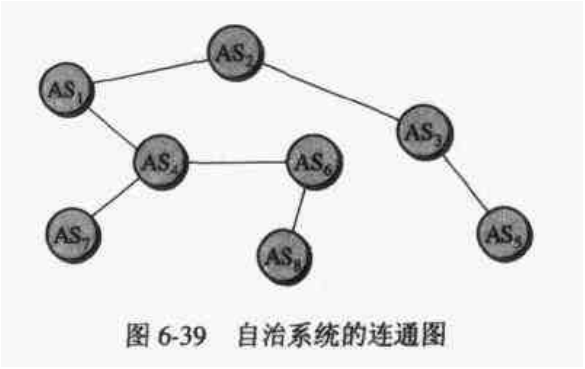


图 6-39　自治系统的连通图

在第 1 章的 1.2.2 节我们已经介绍了当前因特网的多级结构特点（图 1-8）。这种多级结构的网络拓扑决定了 BGP 路由选择协议的特点。

图 6-40 给出了一个 BGP 发言人交换路径向量的例子。自治系统 AS_2 的 BGP 发言人通知主干网的 BGP 发言人："要到达网络 N_1, N_2, N_3 和 N_4 可经过 AS_2。"主干网在收到这个通知后，就发出通知："要到达网络 N_1, N_2, N_3 和 N_4 可沿路径(AS_1, AS_2)。"同理，主干网还可发出通知："要到达网络 N_5, N_6 和 N_7 可沿路径(AS_1, AS_3)。"

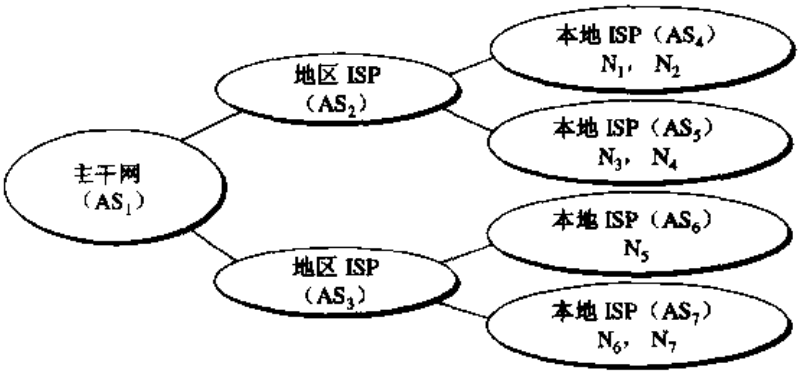


图 6-40　BGP 发言人交换路径向量的例子

从上面的讨论可看出，BGP 协议交换路由信息的结点数量级是自治系统数的量级，这要比这些自治系统中的网络数少很多。要在许多自治系统之间寻找一条较好的路径，就是要寻找正确的 BGP 发言人（或边界路由器），而在每一个自治系统中 BGP 发言人（或边界路由器）的数目是很少的。这样就使得自治系统之间的路由选择不致过分复杂。

BGP 支持 CIDR，因此 BGP 的路由表也就应当包括目的网络前缀、下一跳路由器，以及到达该目的网络所要经过的各个自治系统序列。由于使用了路径向量的信息，就可以很容易地避免产生兜圈子的路由。如果一个 BGP 发言人收到了其他 BGP 发言人发来的路径通知，它就要检查一下本自治系统是否在此通知的路径中。如果在这条路径中，就不能采用这条路径（因为会兜圈子）。

在 BGP 刚刚运行时，BGP 的邻站是交换整个的 BGP 路由表。但以后只需要在发生变化时更新有变化的部分。这样做对节省网络带宽和减少路由器的处理开销方面都有好处。

BGP-4 共使用四种报文，即：

（1）打开(Open)报文，用来与相邻的另一个 BGP 发言人建立关系。

（2）更新(Update)报文，用来发送某一路由的信息，以及列出要撤消的多条路由。

（3）保活(Keepalive)报文，用来确认打开报文和周期性地证实邻站关系。

（3）通知(Notificaton)报文，用来发送检测到的差错。

若两个邻站属于两个不同的自治系统，而其中一个邻站打算和另一个邻站定期地交换路由信息，这就应当有一个商谈的过程（因为很可能对方路由器的负荷已经很重了因而不愿意再加重负担）。因此，一开始向邻站进行商谈时就必须发送打开报文。如果邻站接受这种邻站关系，就用保活报文响应。这样，两个 BGP 发言人的邻站关系就建立了。

一旦邻站关系建立了，就要继续维持这种关系。双方中的每一方都需要确信对方是存在的，且一直在保持这种邻站关系。为此，这两个 BGP 发言人彼此要周期性地交换保活报文（一般每隔 30 秒）。保活报文只有 19 字节长（只用 BGP 报文的通用首部），因此不会造成网络上太大的开销。

更新报文是 BGP 协议的核心内容。BGP 发言人可以用更新报文撤消它以前曾经通知过的路由，也可以宣布增加新的路由。撤消路由可以一次撤消许多条，但增加新路由时，每个更新报文只能增加一条。

BGP 可以很容易地解决距离向量路由选择算法中的“坏消息传播得慢”这一问题。当某个路由器或链路出故障时，由于 BGP 发言人可以从不止一个邻站获得路由信息，因此很容易选择出新的路由。距离向量算法往往不能给出正确的选择，是因为这些算法不能指出哪些邻站到目的站的路由是独立的。

图 6-41 给出了 BGP 报文的格式。四种类型的 BGP 报文的首部都是一样的，长度为 19 字节。分为三个字段。标记(marker)字段为 16 字节长，用来鉴别收到的 BGP 报文（设立这一字段是假定将来有人会发明出合理的鉴别方案）。当不使用鉴别时，标记字段要置为全 1。长度字段指出包括首部在内的整个BGP报文以字节为单位的长度，最小值是19，最大值是4096。类型字段的值为 1 到 4，分别对应于上述四种 BGP 报文中的一种。

打开报文共有 6 个字段，即**版本**（1 字节，现在的值是 4）、**本自治系统编号**（2 字节，使用全球惟一的 16 bit AS 代码）、**保持时间**（2 字节，以秒计算的保持为邻站关系的时间）、**BGP 标识符**（4 字节，通常就是该路由器的 IP 地址）、**可选参数长度**（1 字节）和**可选参数**。

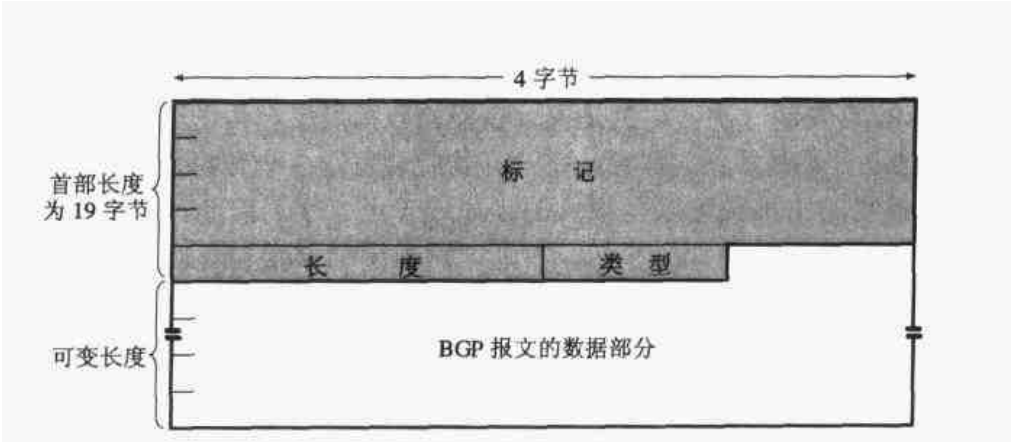


图 6-41　BGP 报文的格式

更新报文共有 5 个字段，即**不可行路由长度**（2 字节，指明下一个字段的长度）、**撤消的路由**（列出所有要撤消的路由）、**路径属性总长度**（2 字节，指明下一个字段的长度）、**路径属性**（定义在这个报文中增加的路径的属性）和**网络层可达性信息 NLRI** (Network Layer Reachability Information)。最后这个字段定义发出此报文的网络，包括网络前缀的比特数、IP 地址前缀。

保活报文只有 BGP 的 19 字节长的首部，没有数据部分。

通知报文有 3 个字段，即差错代码（1 字节）、差错子代码（1 字节）和差错数据（给出有关差错的诊断信息）

6.6　IP 多播和因特网组管理协议 IGMP

6.6.1　IP 多播的基本概念

1988 年 Steve Deering 首次在其博士学位论文中提出 IP 多播的概念。1992 年 3 月 IETF 在因特网范围首次试验 IETF 会议声音的多播，当时有 20 个网点可同时听到会议的声音。IP 多播是需要在因特网上增加更多的智能才能提供的一种服务。现在 IP 多播（multicast，以前曾译为组播）已成为因特网的一个热门课题。这是由于有许多的应用需要由一个源点发送到许多个终点，即一对多的通信。例如，实时信息的交付（如新闻、股市行情等），软件更新，交互式会议等。随着因特网的用户数目的急剧增加，以及多媒体通信的开展，有更多的业务需要多播来支持。关于 IP 多播可参考[W-MCAST]。

在第 4 章曾提到过多播。局域网的多播是用硬件实现的。当以太网上的 PC 机收到一个帧时，用 PC 机网卡硬件就可判断该帧的目的地址是否属于以下三种地址之一：

（1）本网卡的硬件地址（单播）。

（2）全 1 的目的地址（广播）。

（3）地址的第一字节的最低位为 1 的组地址，且本站已加入到该组（多播）。

如果是，就收下该帧。否则就丢弃。

在因特网上向多个目的站发送同样的数据报可以有两种方法。一种方法是采用**单播**，即一次向一个目的站发送数据报，这样的发送共进行多次。另一种方法是采用**多播**。图 6-42 表示多播的特点。图中三个主机 A, C 和 D 构成一个多播组 G。现在主机 X 向多播组 G 的三个主机进行多播（主机 X 可以不属于该多播组，也不一定要知道这个多播组中都

包括哪些成员）。主机 X 在进行多播时只发送一个数据报，只是到了路由器 R_2 才进行复制，然后到了 R_6 再复制一次。这就是说，多播的数据报仅在传送路径分岔时才将数据报复制后继续转发。若不是多播，则源站在一开始就要发送 3 个数据报，分别发给主机 A, C 和 D。构成多播组的主机数可以是很大的，如成千上万个。因此，采用多播协议可明显地减轻网络中各种资源的消耗。显然，在因特网范围的多播要靠路由器来实现，因此这些路由器必须增加一些能够识别多播的软件。能够运行多播协议的路由器称为**多播路由器**(multicast router)。多播路由器可以是一个单独的路由器，也可以是运行多播软件的普通路由器。

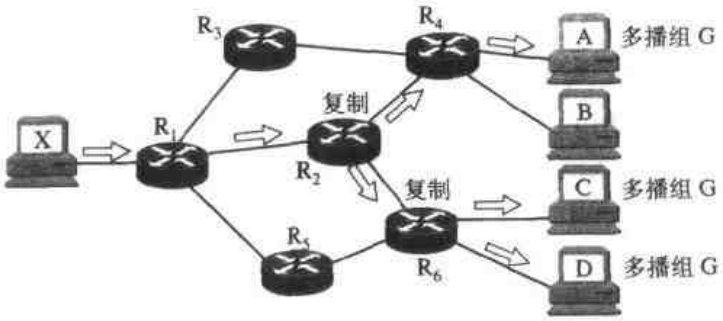


图 6-42　多播可明显地减少网络中资源的消耗

为了适应交互式音频和视频信息的多播，从 1992 年起，在因特网上开始试验虚拟的**多播主干网** MBONE (Multicast Backbone On the InterNEt)。MBONE 可以将分组传播给不在一起但属于一个组的许多个主机。现在多播主干网的规模已经很大，有几千个多播路由器。

在因特网上进行多播就叫做 **IP 多播**。IP 多播具有以下的一些特点：

（1）**多播使用组地址**

IP 使用 D 类地址支持多播。D 类 IP 地址的前缀是 1110，因而地址范围是 224.0.0.0 到 239.255.255.255。每一个 D 类地址标志一组主机。D 类地址可用来标志各个**主机组**(host group)的共有 28 bit，因此可以标志 2^{28} 个多播组（超过 2 亿 5 千万个多播组）。当某进程向某个 D 类地址发送数据报时，就是向该组中的每一个主机发送同样的数据报，但都是“尽最大努力交付”，而某些组内成员可能收不到这个数据报。

显然，**多播地址只能用于目的地址，而不能用于源地址**。

（2）**永久组地址**

下面是曾由**因特网号码指派管理局** IANA (Internet Assigned Numbers Authority)所分配的几个永久组地址的例子：

224.0.0.0　基地址（保留）

224.0.0.1　在本子网上的所有参加多播的主机和路由器

224.0.0.2　在本子网上的所有参加多播的路由器

224.0.0.3　未指派

224.0.0.4　DVMRP 路由器

……

224.0.0.19 至 224.0.0.225　未指派

239.192.0.0 至 239.251.255.255　限制在一个组织的范围

239.252.0.0 至 239.255.255.255　限制在一个地点的范围

（3）动态的组成员

主机组中的成员是动态的。临时组地址则是在每一次使用前都必须创建主机组。一个进程可请求其主机参加某个特定的组，或在任意时间退出该组。当一个主机新加入某一个主机组时，它就向多播地址中的所有主机发送报文，声明其组员关系。本地的多播路由器收到此报文后，就将此报文转发到因特网中其他的多播路由器。当主机上最后一个进程退出某个组时，该主机即不再属于那个组了。每一个主机都知道当前它的各进程属于哪些组。由于组内成员的关系是动态的，因此本地的多播路由器要周期性地向本地网络上的主机进行轮询，以确定哪些主机仍留在组内。若经过几次轮询在一个组内已没有主机是其中的成员，多播路由器就认为该网络中已经没有主机属于该组，以后也就不再向其他的多播路由器通告组内成员的状况。

（4）使用硬件进行多播

由于因特网是由许多网络互连起来的，其中有些网络是以太网，这些以太网本身就具有硬件多播能力，因此当多播数据报传送到这些以太网时，以太网就利用硬件进行多播，交付给属于该组成员的主机。这样的主机在一个以太网上可能有多个。

因特网号码指派管理局 IANA 拥有的以太网地址块的高 24 bit 为 00-00-5E，也就是说，整个以太网硬件地址的范围是从 00-00-5E-00-00-00 到 00-00-5E-FF-FF-FF。在 4.3.1 已讲过，以太网硬件地址字段中的第 1 字节的最低位为 1 时即为多播地址。IANA 用其中的一半作为多播地址，即 IANA 拥有的以太网多播地址的范围是从 01-00-5E-00-00-00 到 01-00-5E-7F-FF-FF。这样，以太网中只有 23 bit 用作多播地址。这只能和 D 类 IP 地址中的 23 bit 有一一对应的关系。D 类 IP 地址可供分配的有 28 bit，可见在这 28 bit 中的前 5 bit 不能用来构成以太网硬件地址（图 6-43）。例如，IP 多播地址 224.128.64.32（即 E0-80-40-20）和另一个 IP 多播地址 224.0.64.32（即 E0-00-40-20）转换成以太网的硬件多播地址都是 01-00-5E-00-40-20。由于多播 IP 地址与以太网硬件地址的映射关系不是惟一的，因此到了 IP 模块还需要利用软件进行过滤，把不是本主机要接收的数据报丢弃。

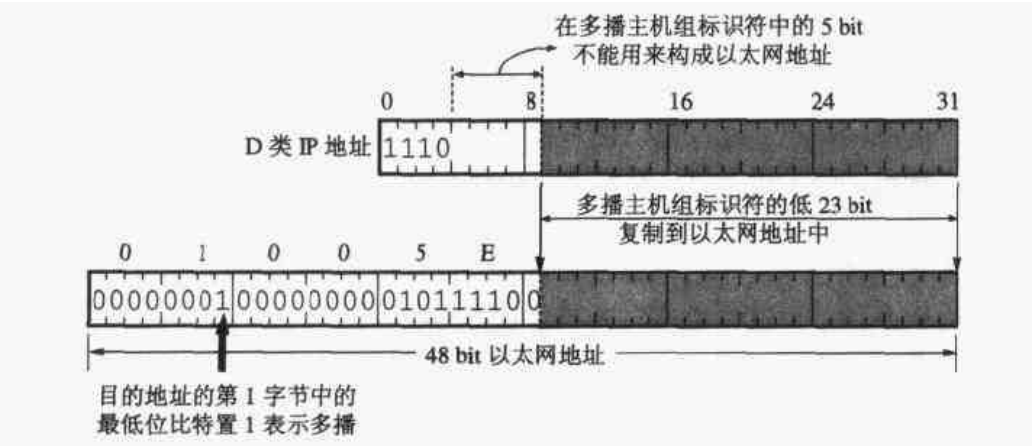


图 6-43　D 类 IP 地址与以太网多播地址的映射关系

6.6.2　因特网组管理协议 IGMP

因特网组管理协议 IGMP (Internet Group Management Protocol)是在多播环境下使用的协

议，它位于网际层。IGMP 就是用来帮助多播路由器识别加入到一个多播组的成员主机。IGMP 的第一个版本[RFC 1112]很早就成为了因特网标准协议。由于多播业务的增多，IGMP 协议也就日益引起人们的重视。1997 年公布了 IGMP 的第二个版本[RFC 2236]（有时也记为 IGMPv2），它已成为因特网建议标准协议。

和 ICMP 相似，IGMP 使用 IP 数据报传递其报文（即 IGMP 报文加上 IP 首部构成 IP 数据报），但它也向 IP 提供服务。因此，我们不把 IGMP 看成是一个单独的协议，而是属于整个网际协议 IP 的一个组成部分。

从概念上讲，IGMP 可分为两个阶段。

第一阶段：当某个主机加入新的多播组时，该主机应向多播组的多播地址发送一个 IGMP 报文，声明自己要成为该组的成员。本地的多播路由器收到 IGMP 报文后，将组成员关系转发给因特网上的其他多播路由器。

第二阶段：因为组成员关系是动态的，因此本地多播路由器要周期性地探询本地局域网上的主机，以便知道这些主机是否还继续是组的成员。只要对某个组有一个主机响应，那么多播路由器就认为这个组是活跃的。但一个组在经过几次的探询后仍然没有一个主机响应，则多播路由器就认为本网络上的主机已经都离开了这个组，因此也就不再将该组的成员关系转发给其他的多播路由器。

IGMP 设计得很仔细，避免了多播控制信息给网络增加大量的开销。IGMP 采用的一些具体措施如下：

（1）在主机和多播路由器之间的所有通信都是使用 IP 多播。只要有可能，携带 IGMP 报文的数据报都用硬件多播来传送。因此在支持硬件多播的网络上，没有参加 IP 多播的主机不会收到 IGMP 报文。

（2）多播路由器在探询组成员关系时，只需要对所有的组发送一个请求信息的询问报文，而不需要对每一个组发送一个询问报文（虽然也允许对一个特定组发送询问报文）。默认的询问速率是每 125 秒发送一次（通信量并不太大）。

（3）当同一个网络上连接有几个多播路由器时，它们能够迅速和有效地选择其中的一个来探询主机的成员关系。因此，网络上多个多播路由器并不会引起 IGMP 通信量的增大。

（4）在 IGMP 的询问报文中有一个数值 *N*，它指明一个最长响应时间（默认值为 10 秒）。当收到询问时，主机在 0 到 *N* 之间随机选择发送响应所需经过的时延。因此，若一个主机同时参加了几个多播组，则主机对每一个多播组选择不同的随机数。对应于最小时延的响应最先发送。

（5）同一个组内的每一个主机都要监听响应，只要有本组的其他主机先发送了响应，自己就可以不再发送响应了。这样就抑制了不必要的通信量。

要理解上面最后一点并不难。多播路由器并不需要保留组成员关系的准确记录，因为向局域网上的组成员转发数据报是使用硬件多播。因此，多播路由器只需知道网络上是否至少还有一个主机是本组成员即可。当询问报文通过多播到达组内每个成员后，各主机就计算出随机时延，并且时延最小的最先发送响应。因为响应也是按组的多播地址发送的，因此最先发送的响应能够被组内所有其他成员收到。一个组的成员只要知道了有其他主机已经发送了对本组的响应，就取消自己原来准备发送的响应。这样，对询问报文实际上每一个组只有一个主机发送响应。

主机在多播中的不同状态可用图 6-44 来说明。图中的圆圈表示主机的状态。圆圈之间带箭头的弧线表示状态变迁。弧线旁边标注的字，在斜线前面的表示所发生的事件，斜线后面有下划线的表示所采取的动作。状态变迁图没有画出主机发送或接收的报文。

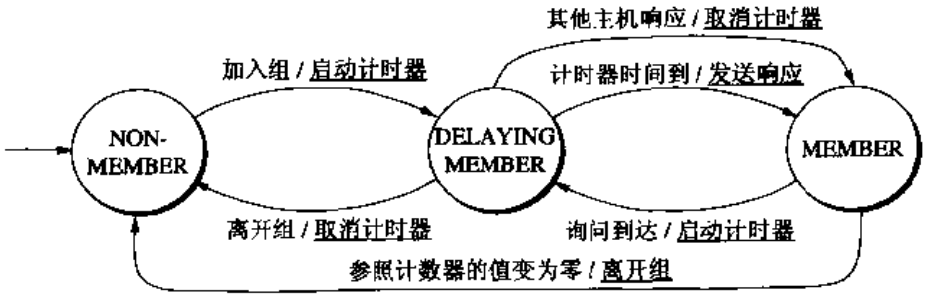


图 6-44　主机在多播中的几种状态

当一个主机还没有加入多播组时，就处在**非成员状态**(NON-MEMBER)。当要加入某个多播组或收到来自多播路由器的询问报文时，就启动计时器，并进入**迟延成员状态**(DELAYING-MEMBER)。这时有两种可能。若计时器到时间，该主机就发送一个响应，并进入**成员状态**(MEMBER)。但若先收到另一个主机的响应，就取消自己的计时器，并进入成员状态。多播路由器每隔 125 秒就产生一个询问报文。

每一个主机都保留有记录其组成员关系信息的表。当主机中的某个应用程序加入到一个新的组时，IGMP 软件就分配一个项目，填入该组的信息。在这些信息中，IGMP 维持一个组参照计数器，并将其值置为 1。若有另一个应用程序也加入到这个组，IGMP 就将此计数器加 1。到应用程序终止执行或退出该组时，IGMP 就将计数器减 1。当组参照计数器的值降为零时，主机就通知多播路由器："我离开这个多播组了"。于是主机又回到非成员状态。

图 6-44 省略了一些细节。例如，当主机处于迟延成员状态时到达了一个询问报文，则协议要求主机将其计时器复位。此外，IGMPv2 还应当和旧的版本 IGMPv1 向后兼容。

IGMP 的报文格式如图 6-45 所示。

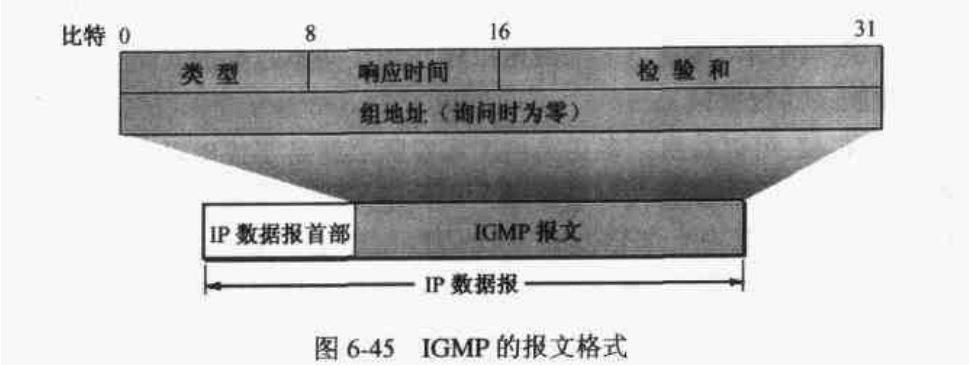


图 6-45　IGMP 的报文格式

IGMP 报文只有 8 个字节，共分为四个字段。

（1）**类型**　　见表 6-9。目前有三种 IGMPv2 类型和一种 IGMPv1 类型。

（2）**响应时间**　　以 1/10 秒为单位。默认值是 10 秒。

（3）**检验和**　　对整个 IGMP 报文进行检验，其算法和 IP 数据报的相同。

表 6-9 IGMP 报文的类型

类　型	组 地 址	意　义	版　本
0x11	不使用（填入 0）	一般的成员关系询问	IGMPv2 使用
	使用	特定的成员关系询问	IGMPv2 使用
0x16	使用	成员关系报告	IGMPv2 使用
0x17	使用	离开组	IGMPv2 使用
0x12	使用	成员关系报告	IGMPv1 使用

（4）**组地址**　当对所有的组发出询问时，组地址字段就填入零。当询问特定的组时，路由器就填入该组的组地址。主机发送成员关系的报告时填入自己的组地址。

6.6.3 多播路由选择

上面介绍的 IGMP 协议指明了主机如何与本地的多播路由器进行交互，但并没有指明在多播路由器之间应如何交换组成员关系的信息，也没有指明怎样保证每一个多播数据报能够可靠地到达所有的组成员。

虽然在 TCP/IP 中 IP 的多播已成为标准协议，但在多播路由器中路由信息的传播则尚未标准化。这主要是因为多播转发和路由选择和单播的情况很不相同而且相当复杂。

多播路由选择之所以相当复杂是由于以下三个原因：

（1）即使网络拓扑不发生变化（包括网络中机器的故障状态也没有发生变化），但由于某个应用程序加入或离开了一个多播组，多播路由都会发生变化。

（2）多播转发要求路由器不仅要检查目的地址，而且还要检查源地址，以便确定什么时候需要将多播数据报进行复制并转发多播数据报副本。

（3）多播数据报可以由不是多播组成员的主机产生，并且可能通过没有任何组成员的网络。

在进行多播的过程中，组成员是不断动态变化的。例如在收听网上广播时，随时会有人加入或离开多播组。如果让网上所有的路由器实时而精确地掌握各个组成员关系的动态信息，那么网络上的多播控制信息就会非常大，造成极大的网络资源浪费。但如反过来，每个路由器得到的信息不完整，那么路由选择就会滞后于组成员关系的变化。因此，多播的设计就是要采取折衷的方法，即：多播组成员关系的信息不要传播得太快，同时多播路由器也不会做出最佳的路由判决（例如，或者将多播数据报转发到某些不必要的网络上，或者没有使所有的组成员都能收到多播数据报）。

从前面的图 6-42 可以看出，多播路由选择实际上就是要找出以源主机为根结点的多播树。在多播树上，每一个数据报在每条链路上只传送一次（不兜圈子）。

在因特网上使用的第一个多播路由选择协议是**距离向量多播路由选择协议** DVMRP (Distance Vector Multicast Routing Protocol)，它到现在还在使用[RFC 1075]。

由于在 UNIX 系统中实现 RIP 的程序叫做 routed，所以在 UNIX 系统上处理多播路由的程序就在 routed 的前面加表示多播(multicast)的字母 m，叫做 mrouted，它使用 DVMRP 在路由器之间传播路由信息，但它不使用标准的路由表。mrouted 只能在 UNIX 的一个特殊版本（多播内核）上运行。UNIX 的多播内核有一个特殊的路由表以及需要转发多播数据报的一些代码。当多播数据报在传输的过程中，若遇到有不运行多播软件的路由器或网络时，就要采用一种**隧道技术**(tunneling)。图 6-46 是对隧道技术的说明。

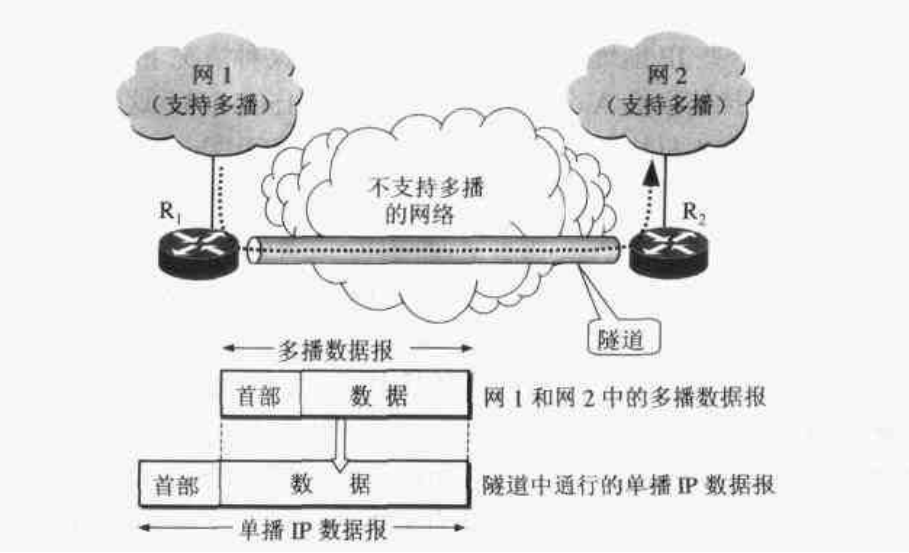


图 6-46 隧道技术在多播中的应用

图 6-46 表示网 1 中的主机向网 2 中的一些主机进行多播。但路由器 R_1 或 R_2 并不运行多播软件，因而不能按多播地址转发数据报。为此，路由器 R_1 就对多播数据报进行再次封装，即再加上普通数据报首部，使之成为向单一目的站发送的**单播**(unicast)数据报（以前所讨论的非多播和非广播的数据报都是单播数据报），然后通过“**隧道**”(tunnel)从 R_1 发送到 R_2。这种隧道叫做 **mrouted tunnel**。单播数据报到达路由器 R_2 后，再由路由器 R_2 剥去其首部，使它又恢复成原来的多播数据报，继续向多个目的站转发。这一点和英吉利海峡隧道运送汽车的情况相似。英吉利海峡隧道不允许汽车在隧道中行驶。但是，可以把汽车放置在隧道中行驶的电气火车上来通过隧道。过了隧道后，汽车又可以继续在公路上行驶。这种使用隧道技术传送数据报又叫做 IP 中的 IP (IP-in-IP)。

现在还提出来好几种多播路由选择协议，如**核心基干树** CBT (Core Based Tree) [RFC 2189, 2201]，**开放最短通路优先的多播扩展 MOSPF** (Multicast Extensions to OSPF) [RFC 1548]，以及**协议无关多播-稀疏方式 PIM-SM** (Protocol Independent Multicast-Sparse Mode) [RFC 2362] 和**协议无关多播-密集方式 PIM-DM** (Protocol Independent Multicast-Dense Mode)。但这些协议都还没有成为因特网的正式标准。读者可注意这方面的进展。

6.7 虚拟专用网 VPN 和网络地址转换 NAT

6.7.1 虚拟专用网 VPN

由于 IP 地址的紧缺，一个机构能够申请到的 IP 地址数往往远小于本机构所拥有的主机数。实际上，出于安全等原因，一个机构内的很多主机并不需要接入到外部的因特网，它们主要是和内部的其他主机进行通信（例如，在大型商场或宾馆中有很多用于营业和管理的计算机。显然这些计算机并不需要和因特网相连）。假定在一个机构内部的计算机通信也是采用 TCP/IP 协议，那么从原则上讲，对于这些仅在机构内部使用的计算机就可以由本机构**自行分配**其 IP 地址。这就是说，让这些计算机使用仅在本机构有效的 IP 地址（这种地址称为**本地地址**），而不需要向因特网的管理机构申请全球惟一的 IP 地址（这种地址称为**全球地址**）。这样就可以大大节约宝贵的全球 IP 地址资源。

但是，如果任意选择一些IP地址作为本地地址，那么在某种情况下可能会引起一些麻烦。例如，一个不和因特网连接的主机A分配到一个本地地址150.1.2.3。这个地址不需要在因特网地址管理机构注册，但在本机构内必须是惟一的。然而正巧因特网上有一个主机，其IP地址就是150.1.2.3，而且这个主机要和本机构的某个具有全球地址的主机通信。这样就会出现地址的二义性问题。

为了解决这一问题，[RFC 1918]指明了一些**专用地址**(private address)。这些地址只能用于一个机构的内部通信，而不能用于和因特网上的主机通信。换言之，专用地址只能用作本地地址而不能用作全球地址。在因特网中的所有路由器对目的地址是专用地址的数据报一律不进行转发。[RFC 1918]指明的专用地址是：

（1）10.0.0.0到10.255.255.255　（或记为10/8，它又称为24 bit块）

（2）172.16.0.0到172.31.255.255　（或记为172.16/12，它又称为20 bit块）

（3）192.168.0.0到192.168.255.255　（或记为192.168/16，它又称为16 bit块）

上面的三个地址块分别相当于一个A类网络、16个连续的B类网络和256个连续的C类网络。A类地址本来早已用完了，而上面的地址10.0.0.0本来是分配给ARPANET的。由于ARPANET已经关闭停止运行了，因此这个地址就用作专用地址。

采用这样的专用IP地址的互连网络称为**专用互连网**或**本地互连网**，或更简单些，就叫做**专用网**。显然，全世界可能有很多的专用互连网络具有相同的专用IP地址，但这并不会引起麻烦，因为这些专用地址仅在本机构内部使用。专用IP地址也叫做**可重用地址**(reusable address)。

有时一个很大的机构有许多部门分布在相距很远的一些地点，而在每一个地点都有自己的专用网。假定这些分布在不同地点的专用网需要经常进行通信。这时，可以有两种方法。第一种方法是租用电信公司的线路为本机构专用。这种方法的好处是简单方便，但线路的租金太高。第二种方法是利用因特网（即公用互连网）来实现本机构的专用网，因此这样的专用网又称为**虚拟专用网** VPN (Virtual Private Network)。“虚拟”即“好像是”但实际上不是，因为现在是因特网（而并没有用专线）来连接分散在各地的本地网络。VPN只是在效果上和真正的专用网一样。图6-48说明如何使用隧道技术实现虚拟专用网。

假定某个机构在两个相隔较远的部门A和B建立了专用网，其网络地址分别为专用地址10.1.0.0和10.2.0.0。现在这两个部门需要通过因特网进行通信。

显然，每一个部门至少要有一个路由器具有合法的全球IP地址，如图6-47(a)中的路由器R_1和R_2。这两个路由器和因特网的接口地址必须是合法的全球IP地址。路由器R_1和R_2在和专用网内部网络的接口地址则是专用网的本地地址。

现在设部门A的主机X要向部门B的主机Y发送数据报，源地址是10.1.0.1而目的地址是10.2.0.3。这个数据报作为本机构的内部数据报从X发送到与外界连接的路由器R_1。路由器R_1收到内部数据报后将整个的内部数据报进行加密，然后重新填加上数据报的首部封装成为在因特网上发送的外部数据报，其源地址是路由器R_1的全球地址125.1.2.3，而目的地址是路由器R_2的全球地址194.4.5.6。路由器R_2收到数据报后将其数据部分取出进行解密，恢复出原来的内部数据报，并转发给主机Y。

由于在因特网上传送的外部数据报的数据部分（即内部数据报）是加密的，因此在因特网上所经过的所有路由器都不知道内部数据报的内容。图中在路由器R_1和R_2之间的隧道表

明了这一概念。关于加密的具体方法见第 9 章。

如图 6-47(b)所示的由部门 A 和 B 的内部网络所构成的虚拟专用网 VPN 又称为**内联网**(Intranet)，表示部门 A 和 B 都是在同一个机构的内部。但有时一个机构需要和某些外部机构共同建立一个虚拟专用网。这样的 VPN 又称为**外联网**(Extranet)。这里再强调一下，内联网和外联网都是采用因特网技术的，即都是基于 TCP/IP 协议的。

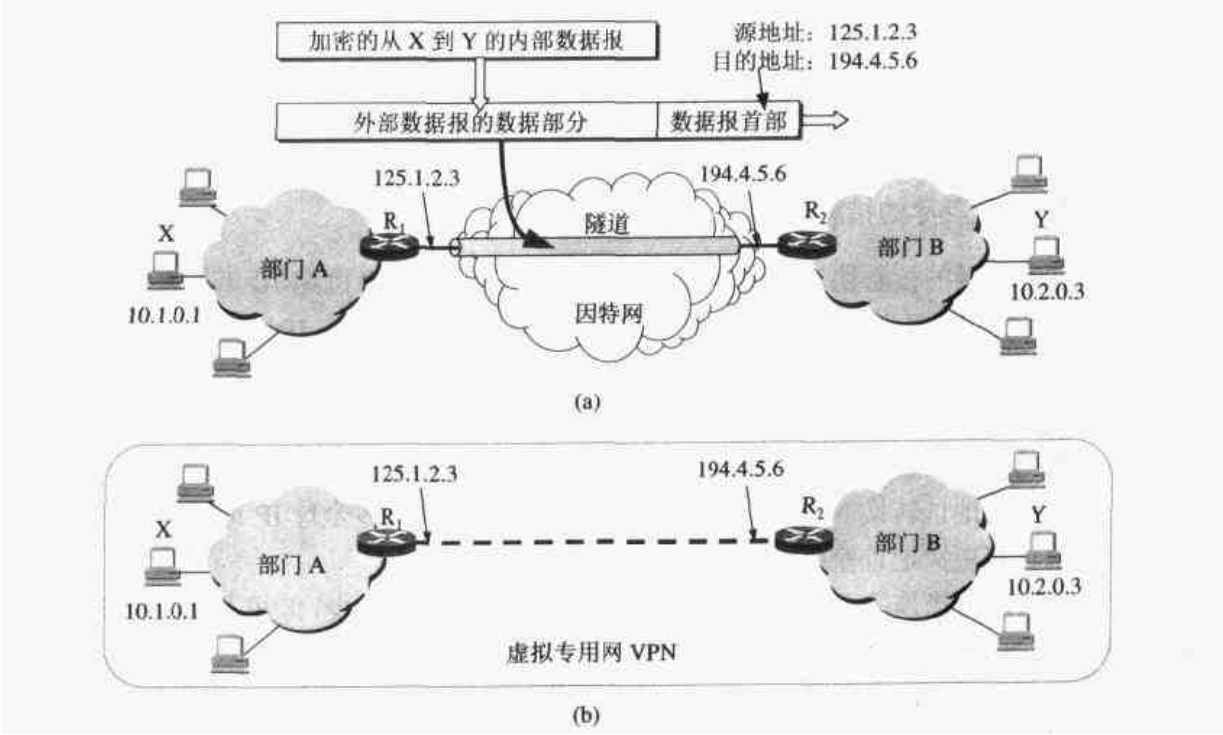


图 6-47　用隧道技术实现虚拟专用网：(a)使用隧道技术；(b)构成虚拟专用网

6.7.2　网络地址转换 NAT

下面讨论另一种情况，就是在专用网内部的一些主机本来已经分配到了本地 IP 地址，但现在又想和因特网上的主机通信（并不需要加密），那么应当采取什么措施呢？

最简单的办法就是设法再申请一些全球 IP 地址。但这在很多情况下是不容易做到的，因为全球 IP 地址已所剩不多了。目前使用得最多的方法是采用网络地址转换。

网络地址转换 NAT (Network Address Translation)方法是在 1994 年提出的。这种方法需要在专用网连接到因特网的路由器上安装 NAT 软件。装有 NAT 软件的路由器叫做 NAT 路由器，它至少有一个有效的外部全球地址 IP_G。这样，所有使用本地地址的主机在和外界通信时都要在 NAT 路由器上将其本地地址转换成 IP_G 才能和因特网连接。

例如，当内部主机 X 用其本地地址 IP_X 和因特网上的主机 Y 通信时，它所发送的数据报必须经过 NAT 路由器。NAT 路由器将数据报的源地址 IP_X 转换成自己的全球地址 IP_G，但目的地址 IP_Y 保持不变，然后发送到因特网。当 NAT 路由器从因特网收到主机 Y 发回的数据报时，知道数据报中的源地址是 IP_Y 而目的地址是 IP_G。根据原来的记录（这个记录叫做 NAT 转换表），NAT 路由器知道这个数据报是要发送给主机 X 的，因此 NAT 路由器将目的地址 IP_G 转换为 IP_X，转发给最终的内部主机 X。

如果 NAT 路由器具有多个全球 IP 地址，那么就可以同时将多个本地地址转换为全球 IP

地址，因而可以使多个拥有本地地址的主机能够和因特网上的主机进行通信。

还有一种 NAT 转换表将运输层的端口号也利用上（端口号将在 7.2.2 节讨论）。这样就可以用一个全球 IP 地址使多个拥有本地地址的主机同时和因特网上的不同主机进行通信[COME00]。

有关 NAT 的详细讨论见[RFC 3235, 3027, 3022, 2993, 2663]和[W-NAT]。网站[W-NAT]是 IETF 关于 NAT 的工作组的相关资料。

6.8 下一代的网际协议 IPv6(IPng)

6.8.1 解决 IP 地址耗尽的措施

IP 协议是因特网的核心协议。现在使用的 IP(即 IPv4)是在 20 世纪 70 年代末期设计的，无论从计算机本身发展还是从因特网规模和网络传输速率来看，现在 IPv4 已很不适用了。这里最主要的问题就是 32 bit 的 IP 地址不够用。

要解决 IP 地址耗尽的问题，可以采用以下三个措施：

（1）采用无分类编址 CIDR（见 6.3.3 节），使 IP 地址的分配更加合理。

（2）采用网络地址转换 NAT 方法（见 6.7.2 节），可节省许多全球 IP 地址。

（3）采用具有更大地址空间的新版本的 IP 协议，即 IPv6。

尽管上述前两项措施的采用使得 IP 地址耗尽的日期推后了不少，但却不能从根本上解决 IP 地址即将耗尽的问题。因此，治本的方法应当是上述的第三种方法。

及早开始过渡到 IPv6 的好处是：有更多的时间来规划平滑过渡；有更多的时间培养 IPv6 的专门人才；及早提供 IPv6 服务比较便宜。因此现在有些 ISP 已经开始了进行 IPv6 的过渡。

IETF 早在 1992 年 6 月就提出要制订**下一代的 IP**，即 Ipng (IP Next Generation)。IPng 现正式称为 IPv6。1998 年 12 月发表的[RFC 2460～2463]已成为因特网草案标准协议。应当指出，换一个新版的 IP 并非易事。世界上许多团体都从因特网的发展中看到了机遇，因此在新标准的制订过程中出于自身的经济利益而产生了激烈的争论。到目前为止，IPv6 还只是草案标准阶段。有关向 IPv6 转换的进展情况可在[W-NGTRANS]找到。

下面是 IPv6 的简介。

6.8.2 IPv6 的基本首部

IPv6 仍支持无连接的传送，但将协议数据单元 PDU 称为分组，而不是 IPv4 的数据报。为方便起见，本书仍采用数据报这一名词([COME00]和[TANE96]也是这样做的)。

IPv6 所引进的主要变化如下：

（1）**更大的地址空间**。IPv6 将地址从 IPv4 的 32 bit 增大到了 128 bit，使地址空间增大了 2^{96} 倍。这样大的地址空间在可预见的将来是不会用完的。

（2）**扩展的地址层次结构**。IPv6 由于地址空间很大，因此可以划分为更多的层次。

（3）**灵活的首部格式**。IPv6 数据报的首部和 IPv4 的并不兼容。IPv6 定义了许多可选的扩展首部，不仅可提供比 IPv4 更多的功能，而且还可提高路由器的处理效率，这是因为路由器对扩展首部不进行处理（除逐跳扩展首部外）。

（4）**改进的选项**。IPv6 允许数据报包含有选项的控制信息，因而可以包含一些新的选项。我们知道，IPv4 所规定的选项是固定不变的。

（5）**允许协议继续扩充**。这一点很重要，因为技术总是在不断地发展（如网络硬件的更新）而新的应用也还会出现。但我们知道，IPv4 的功能是固定不变的。

（6）**支持即插即用（即自动配置）**。

（7）**支持资源的预分配**。IPv6 支持实时视像等要求保证一定的带宽和时延的应用。

IPv6 将首部长度变为固定的 40 字节，称为**基本首部**(base header)。将不必要的功能取消了，首部的字段数减少到只有 8 个（虽然首部长度增大了一倍）。此外，还取消了首部的检验和字段（考虑到数据链路层和运输层都有差错检验功能）。这样就加快了路由器处理数据报的速度。

IPv6 数据报在**基本首部**的后面允许有零个或多个**扩展首部**(extension header)，再后面是数据（见图 6-48 所示）。但请注意，所有的扩展首部都不属于数据报的首部。所有的扩展首部和数据合起来叫做数据报的**有效载荷**(payload)或**净负荷**。

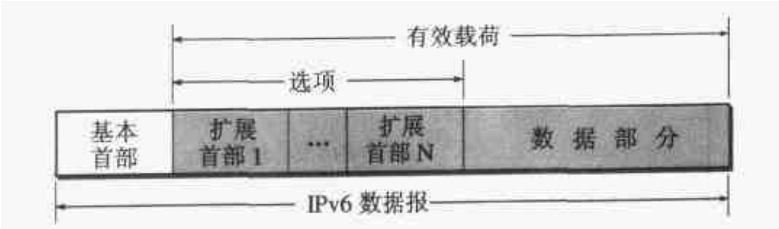


图 6-48　具有多个可选扩展首部的 IPv6 数据报的一般形式

为了和 IPv4 数据报的首部进行对比，我们在图 6-49(a)重新画出 IPv4 数据报的首部，其中在 IPv6 数据报中取消的字段加上灰色方框的记号（共 8 个），而有变化的字段加上灰色椭圆框记号（共 5 个）。图 6-49(b)是 IPv6 数据报的基本首部。在基本首部后面是**有效载荷**，它包括运输层的数据和可能选用的扩展首部。

下面解释 IPv6 基本首部中的各字段的作用。

（1）**版本**(version)　　占 4 bit。它指明了协议的版本，对 IPv6 该字段总是 6。

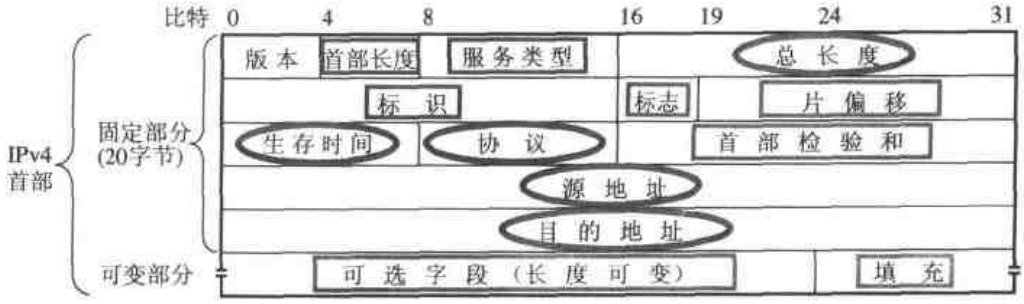
（2）**通信量类**(traffic class)　　占 8 bit。这是为了区分不同的 IPv6 数据报的类别或优先级。目前正在进行不同的通信量类性能的实验。

（3）**流标号**(flow label)　　占 20 bit。IPv6 一个新的机制是支持资源预分配，并且允许路由器将每一个数据报与一个给定的资源分配相联系。IPv6 提出流(flow)的抽象概念。所谓“流”就是互联网络上从特定源点到特定终点（单播或多播）的一系列数据报（如实时音频或视频传输），而在这个“流”所经过的路径上的路由器都保证指明的服务质量。所有属于同一个流的数据报都具有同样的流标号。

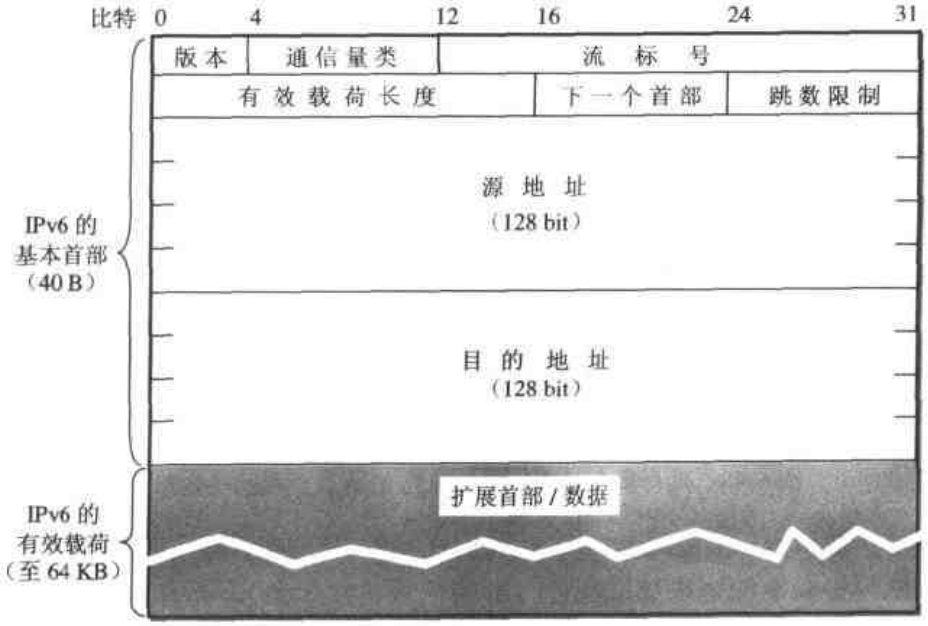
（4）**有效载荷长度**(payload length)　　占 16 bit。它指明 IPv6 数据报除基本首部以外的字节数（所有扩展首部都算在有效载荷之内）。这个字段的最大值是 64 KB。

（5）**下一个首部**(next header)　　占 8 bit。它相当于 IPv4 的协议字段或可选字段。

- 当 IPv6 数据报没有扩展首部时，下一个首部字段的作用和 IPv4 的协议字段一样，它的值指出了基本首部后面的数据应交付给 IP 上面的哪一个高层协议（例如，6 或 17 分别表示应交付给 TCP 或 UDP）。
- 当出现扩展首部时，下一个首部字段的值就标识后面第一个扩展首部的类型。



(a)



(b)

图 6-49　IPv4 首部(a)和 40 字节长的 IPv6 基本首部

（6）**跳数限制(hop limit)**　　占 8 bit。用来防止数据报在网络中无限期地存在。源站在每个数据报发出时即设定某个跳数限制。每个路由器在转发数据报时，要先将跳数限制字段中的值减 1。当跳数限制的值为零时，就要将此数据报丢弃。

（7）**源地址**　　占 128 bit。是数据报的发送站的 IP 地址。

（8）**目的地址**　　占 128 bit。是数据报的接收站的 IP 地址。

下一节我们先讨论 IPv6 的扩展首部。

6.8.3　IPv6 的扩展首部

1．扩展首部及下一个首部字段

大家知道，IPv4 的数据报如果在其首部中使用了选项，那么沿数据报传送的路径上的每一个路由器都必须对这些选项进行一一检查，然而实际上很多的选项在途中的路由器上是不需要检查的(因为它们并不使用这些选项的信息)。这就降低了路由器处理数据报的速度。IPv6 将原来 IPv4 首部中选项的功能都放在扩展首部中，并将扩展首部留给路径两端的源站和目的站的主机来处理，而数据报途中经过的**路由器都不处理这些扩展首部**（只有一个首部例外，

即逐跳选项扩展首部），这样就大大提高了路由器的处理效率。

在[RFC 2460]中定义了以下六种扩展首部：

①逐跳选项；②路由选择；③分片；④鉴别；⑤封装安全有效载荷；⑥目的站选项。

每一个扩展首部都由若干个字段组成，它们的长度也各不同。但所有扩展首部的第一个字段都是 8 bit 的“下一个首部”字段。此字段的值指出了在该扩展首部后面的字段是什么。当使用多个扩展首部时，应按以上的先后顺序出现。高层首部总是放在最后面。

图 6-50(a)表示当数据报不包含扩展首部时，固定首部中的下一个首部字段就相当于 IPv4 首部中的协议字段，此字段的值指出后面的有效载荷应当交付给上一层的哪一个进程。例如，当有效载荷是 TCP 报文段时（固定首部中下一个首部字段的值就是 6，这个数值和 IPv4 中协议字段填入的值一样），后面的有效载荷就被交付给上层的 TCP 进程。

图 6-50(b)表示在基本首部后面有两个扩展首部的情况。所有扩展首部中的第一个字段“下一个首部”的值都是指出了跟随在此扩展首部后面的是何种首部。例如，第一个扩展首部是路由选择首部，其“下一个首部字段”的值就指出后面的扩展首部是分片扩展首部，而分片扩展首部的“下一个首部字段”的值又指出再后面的首部是 TCP/UDP 的首部。

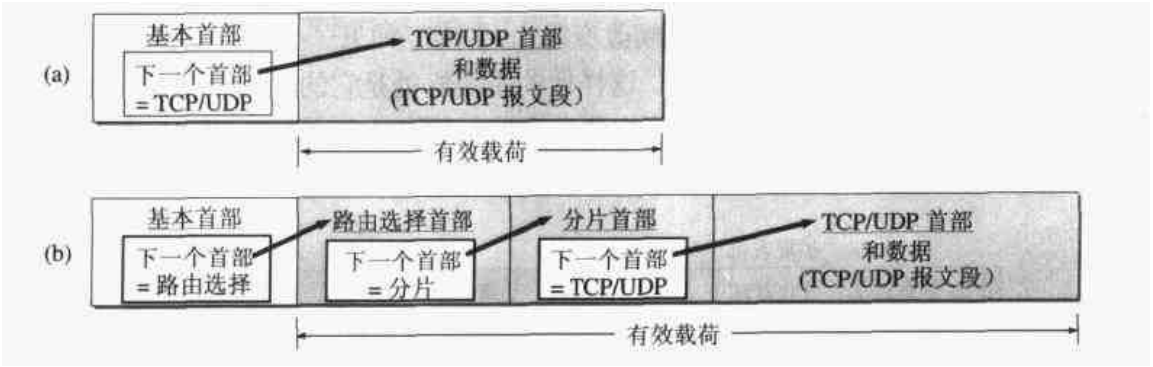


图 6-50　IPv6 的扩展首部：(a)无扩展首部；(b)有两个扩展首部

2．扩展首部举例

下面以分片扩展首部为例来说明扩展首部的作用。

IPv6 将分片限制为由源站来完成。源站可以采用保证的最小 MTU (1280 字节)，或者在发送数据前完成路径最大传送单元发现(Path MTU Discovery)，以确定沿着该路径到目的站的最小 MTU。当需要分片时，源站在发送数据报前先将数据报分片，保证每个数据报片都小于此路径的 MTU。因此，分片是端到端的，路径途中的路由器不允许进行分片。

IPv6 基本首部中不包含用于分片的字段，而是在需要分片时，源站在每一数据报片的基本首部的后边插入一个小的分片扩展首部，它的格式如图 6-51 所示。

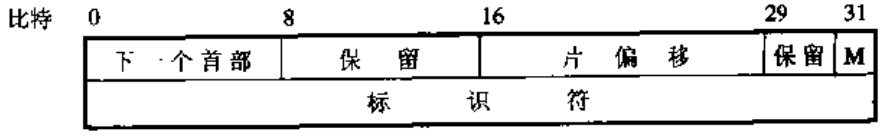


图 6-51　分片扩展首部的格式

IPv6 保留了 IPv4 分片的大部分特征，其分片扩展首部共有以下几个字段：

（1）**下一个首部**(8 bit)　　指明紧接着这个扩展首部的下一个首部。

（2）**保留**(10 bit)　　为今后使用。该字段在第 8～15 bit 和第 29～30 bit。

（3）**片偏移**(13 bit)　　指明本数据报片在原来的数据报中的偏移量，以 8 个字节为表示单位。可见每个数据报片的长度必须是 8 个字节的整数。

（4）**M**(1 bit)　　M = 1 表示后面还有数据报片。M = 0 则表示这已是最后一个数据报片。

（5）**标识符**(32 bit)　　由源站产生的、用来惟一地标志数据报的一个 32 bit 数。每产生一个新数据报，就将这个标识符加 1。采用 32 bit 的标识符，可使得在源站发送到同样的目的站的数据报中，在数据报的生存时间内无相同的标识符（即使是高速网络）。

下面用具体数字加以说明。假定有一个 IPv6 数据报，其有效载荷长度为 3000 字节。现在要将此数据报用下层的以太网传送，而以太网的最大传送单元 MTU 是 1500 字节，因此必须进行分片。我们分成三个数据报片，两个 1400 字节长，最后一个是 200 字节长。分片需要在 IPv6 的基本首部后面增加一个分片扩展首部。分片的结果如图 6-52 所示。

采用端到端分片的方法可以减少路由器的开销，因而允许路由器在单位时间内处理更多的数据报。然而，端到端的分片方法有一个重要的后果：它改变了因特网的基本假设。

因特网原来被设计为允许在任何时候改变路由。例如，如果一个网络或者路由器出故障，那么就可以重新选择另一条不同的路由。这样做的主要好处是它的灵活性。然而 IPv6 就不能这样容易地改变路由，因为改变路由可能也要改变路径的最大传送单元 MTU。如果新路径的 MTU 小于原来路径的 MTU，那么就要想办法解决这个问题。

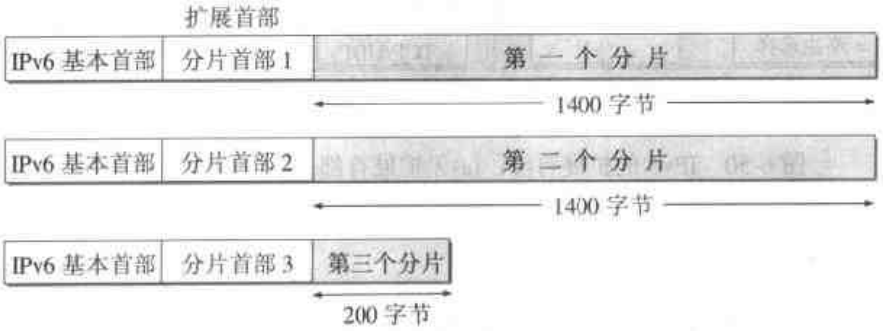


图 6-52　IPv6 数据报分片举例

为此，IPv6 允许中间的路由器采用隧道技术来传送太长的数据报。当路径途中的路由器需要对数据报进行分片时，路由器既不插入数据报片扩展首部，也不改变基本首部中的各个字段。相反，这个路由器创建一个全新的数据报，然后将这个新的数据报分片，并在各个数据报片中插入扩展首部和新的基本首部。最后，路由器将每个数据报片发送给最终的目的站，而在目的站将收到的各个数据报片收集起来，组装成原来的数据报，再从中抽取出数据部分。图 6-53 说明了上述这种采用隧道技术的封装方法。

6.8.4　IPv6 的地址空间

1．128 bit 的地址空间

一般来讲，一个 IPv6 数据报的目的地址可以是以下三种基本类型地址之一：

（1）**单播**(unicast)　　单播就是传统的点对点通信。

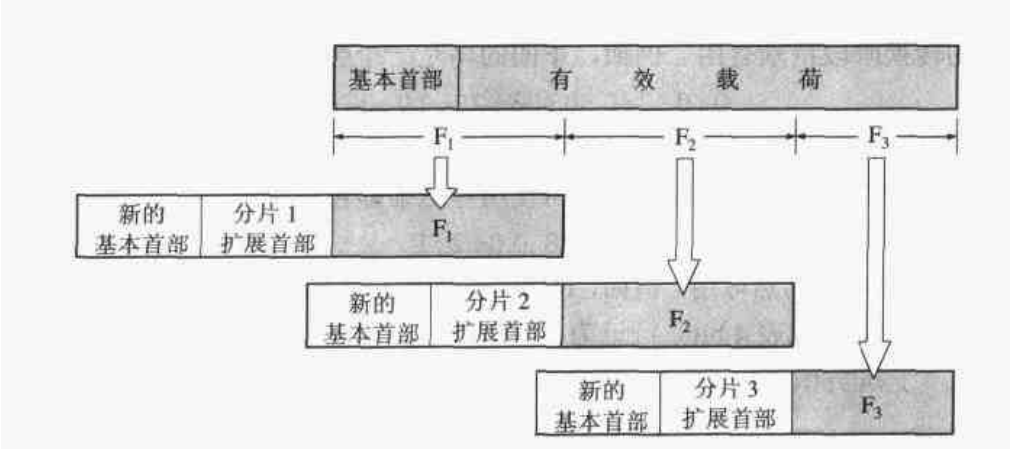


图 6-53 用隧道技术将一个 IPv6 数据报分成 3 个数据报片

（2）**多播(multicast)** 多播是一点对多点的通信，数据报交付到一组计算机中的每一个。IPv6 没有采用广播的术语，而是将广播看作多播的一个特例。

（3）**任播(anycast)** 这是 IPv6 增加的一种类型。任播的目的站是一组计算机，但数据报在交付时只交付给其中的一个，通常是距离最近的一个。

IPv6 将实现 IPv6 的主机和路由器均称为**结点**，并将 IPv6 地址分配给结点上面的**接口**。一个接口可以有多个单播地址。一个结点接口的单播地址可用来惟一地标志该结点。

同 IPv4 一样，IPv6 把一个地址与特定的网络连接（而不是与特定的计算机）相关联，因此，和两个或多个网络相连接的 IPv6 路由器就具有两个或多个 IP 地址，而和一个网络只有一条连接的 IPv6 主机则只需一个 IP 地址。为了地址分配和修改的方便，IPv6 允许给一个给定的网络指派多个前缀，也允许对一个主机的给定接口同时指派多个地址。

在 IPv6 中，每个地址占 128 bit，地址空间大于 3.4×10^{38}。如果整个地球表面（包括陆地和水面）都覆盖着计算机，那么 IPv6 允许每平方米拥有 7×10^{23} 个 IP 地址。如果地址分配速率是每微秒分配 100 万个地址，则需要 10^{19} 年的时间才能将所有可能的地址分配完毕。可见在想像得到的将来，IPv6 的地址空间是不可能用完的。

巨大的地址范围还必须使维护互联网的人易于阅读和操纵这些地址。IPv4 所用的点分十进制记法现在也不够方便了。例如，一个用点分十进制记法的 128 比特的地址：

104.230.140.100.255.255.255.255.0.0.17.128.150.10.255.255

为了使地址再稍简洁些，IPv6 使用**冒号十六进制记法**(colon hexadecimal notation，简写为 colon hex)，它把每个 16 bit 的值用十六进制值表示，各值之间用冒号分隔。例如，如果前面所给的点分十进制数记法的值改为冒号十六进制记法，就变成了：

68E6:8C64:FFFF:FFFF:0:1180:960A:FFFF

这里将 0000 中的前三个 0 省略了。例如，三个 0 后面一个 F(000F)可缩写为 F。

冒号十六进制记法还包含两个技术使它尤其有用。首先，冒号十六进制记法可以允许**零压缩**(zero compression)，即一连串连续的零可以为一对冒号所取代，例如：

FF05:0:0:0:0:0:0:B3

可以写成：

FF05::B3

为了保证零压缩有一个不含混的解释，规定在任一地址中只能使用一次零压缩。该技术对已建议的分配策略特别有用，因为会有许多地址包含连续的零串。

其次，冒号十六进制记法可结合有点分十进制记法的后缀。我们下面会看到这种结合在

IPv4 向 IPv6 的转换阶段特别有用。例如，下面的串是一个合法的冒号十六进制记法：

```
0:0:0:0:0:0:128.10.2.1
```

请注意，在这种记法中，虽然为冒号所分隔的每个值是一个 16 bit 的量，但每个点分十进制部分的值则指明一个字节(8 bit)的值。再使用零压缩即可得出：

```
::128.10.2.1
```

CIDR 的斜线表示法仍然可用。例如，60 bit 的前缀 12AB00000000CD3（十六进制表示的 15 个字符，每个字符代表 4 bit）可记为：

```
12AB:0000:0000:CD30:0000:0000:0000:0000/60
12AB::CD30:0:0:0:0/60
12AB:0:0:CD30::/60
```

2．地址空间的分配

IPv6 将 128 bit 地址空间分为两大部分。第一部分是可变长度的**类型前缀**，它定义了地址的目的。第二部分是地址的其余部分，其长度也是可变的。图 6-54 表示了 IPv6 的地址结构。IPv6 的地址类型前缀如表 6-9 所示[RFC 2373]（目前是因特网的建议标准）。



图 6-54　IPv6 的地址结构

表 6-9　IPv6 的地址分配方案

类型前缀（二进制）	地址的类型	占地址空间的份额
0000 0000	保留（与 IPv4 兼容）	1/256
0000 0001	未指派	1/256
0000 001	保留给 NSAP 地址	1/128
0000 010	保留给 IPX 地址	1/128
0000 011	未指派	1/128
0000 1	未指派	1/32
0001	未指派	1/16
001	可聚合的全球单播地址	1/8
010	未指派	1/8
011	未指派	1/8
100	未指派	1/8
101	未指派	1/8
110	未指派	1/8
1110	未指派	1/16
1111 0	未指派	1/32
1111 10	未指派	1/64
1111 110	未指派	1/128
1111 1110 0	未指派	1/512

续表

类型前缀（二进制）	地址的类型	占地址空间的份额
1111 1110 10	本地链路单播地址	1/1024
1111 1110 11	本地网点单播地址	1/1024
1111 1111	多播地址	1/256

前缀为 0000 0000 是保留一小部分地址与 IPv4 兼容的，这是因为必须要考虑到在比较长的时期 IPv4 和 IPv6 将会同时存在，而有的结点不支持 IPv6。因此数据报在这两类结点之间转发时，就必须进行地址的转换。图 6-55 表示将 IPv4 地址嵌入到 IPv6 地址的两种形式。若地址的前 96 bit 都是 0，而最低 32 bit 是 IPv4 地址，则这种地址叫做“IPv4 兼容的 IPv6 地址”，用在自动隧道技术机制中。使用这样的地址的结点既支持 IPv4 也支持 IPv6。但若地址的前 80 bit 都是 0，接着的 16 bit 都是 1，然后是 IPv4 地址，则这种地址叫做“IPv4 映射的 IPv6 地址”。使用这种地址的结点不支持 IPv6。

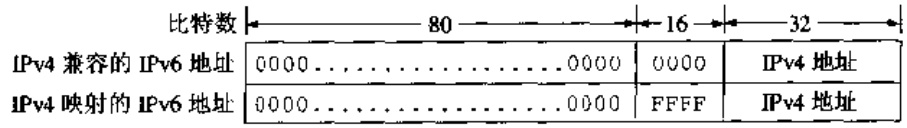


图 6-55　IPv4 地址放在 IPv6 地址的最低 32 bit 处（兼容地址和映射地址）

IPv6 和 IPv4 的最重要的变化之一就是单播地址所使用的划分策略，以及由此产生的多级地址体系。我们知道 IPv4 的地址是两级结构，它的地址被划分为一个全球惟一的前缀和一个后缀[①]。IPv6 扩展了地址的分级概念，它使用以下的三个等级（图 6-56）：

（1）**第一级（顶级）**　指明全球都知道的公共拓扑。

（2）**第二级（地点级）**　指明单个的地点。

（3）**第三级**　指明单个的网络接口。

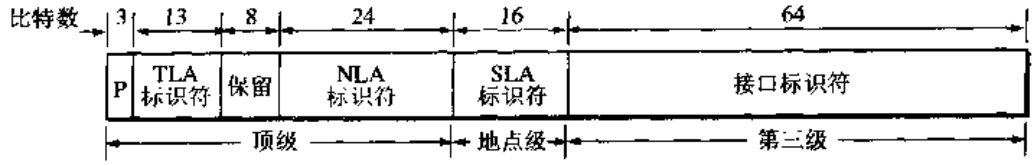


图 6-56　IPv6 单播地址的等级结构

IPv6 的地址体系采用多级体系是充分考虑到怎样使路由器可更快地查找路由。最低的两级很容易理解，因为它们都对应于可标志的实体。

IPv6 地址的最低的第三级对应于计算机和网络的单个接口。与 IPv4 不同，IPv6 地址的后缀有 64 bit 之多，它足够大，因而可以将各种接口的硬件地址直接进行编码。这样，IPv6 不需要使用 ARP 来将 IP 地址解析为硬件地址。IPv6 使用一个叫做**邻站发现协议**(neighbor discovery protocol)使一个结点能够确定哪些计算机是和它相邻接的（在 ICMP 新版本 ICMPv6 中使用这个协议）。为了保证可操作性，所有的计算机都必须对硬件地址使用同样的编码方法。

① 注：采用 CIDR 后，将网络号字段和子网号字段合在一起，称为网络前缀，而在网络前缀后面的主机号就称为**后缀**(suffix)。

因此，IPv6 还指明了各种形式的硬件地址的精确编码方法。

IEEE 定义了一个标准的 64 bit 全球惟一地址格式 EUI-64。它和 4.3.1 节介绍的 EUI-48 相似。EUI-64 的前三个字节(24 bit)仍为公司标识符，但后面的扩展标识符是五个字节(40 bit)。当一个 EUI-64 硬件地址需要转换为 IPv6 地址时，只要将它放入 IPv6 地址中的接口标识符字段中即可。但要将公司标识符的第 1 字节的最低第 2 位（即 G/L 比特）置为 1（因为这时是全球管理的 IP 地址，G/L 比特必须是 1）。

较为复杂的是当需要将 48 bit 的以太网硬件地址转换为 IPv6 地址。图 6-57 表示地址转换的方法。图中上面的地址是 48 bit 的 IEEE 802 以太网地址（每一个字节的高位在前），其中的前 24 bit 是公司标识符（用字母 c 表示），但第一字节的最低位是 I/G 比特（用字母 g 表示），而第一字节的最低第二位是 G/L 比特（图中假定是 0）。

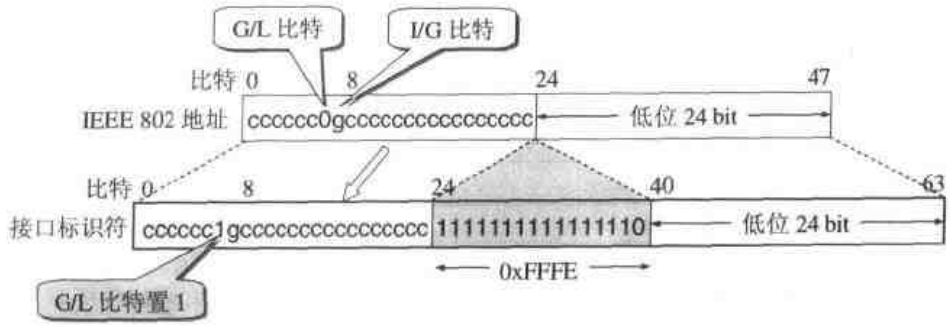


图 6-57　将以太网地址转换为 IPv6 地址

从图 6-57 可看出，将 48 bit 的以太网地址放入到 IPv6 地址中的 64 bit 的接口标识符时，应增加 16 个比特才行。IPv6 规定这 16 比特的十六进制值是 0xFFFE，并且应插入在以太网地址前 24 bit 的公司标识符之后。此外，公司标识符的第一字节的最低第二位必须置为 1。以太网地址后 24 bit 的扩展标识符则复制到接口标识符的最后 24 bit。

IPv6 地址中间的第二级对应于在一个地点的一组计算机和网络，它们通常是相距较近的且都归一个单位来管理。图 6-56 中的 SLA 级表示 Site Level Aggregation，即**地点级聚合**，它和 IPv4 中的子网字段相似。

IPv6 地址最前面的第一级（顶级）有四个字段：

（1）P 字段，占 3 bit，即**格式前缀**(format Prefix)，对可聚合的全球单播地址，格式前缀是 001。

（2）**顶级聚合标识符** TLA ID，占 13 bit。TLA 即 Top-Level Aggregation，该标识符指派给 ISP 或拥有这些地址的**汇接点**(exchange)。

（3）**保留**字段，占 8 bit。

（4）**下一级聚合标识符** NLA ID，占 16 bit。NLA 即 Next-Level Aggregation，该标识符指派给一个特定的**用户**(subscriber)。

6.8.5　从 IPv4 向 IPv6 过渡

由于现在整个因特网上使用老版本 IPv4 的路由器的数量太大，因此，“规定一个日期，从这一天起所有的路由器一律都改用 IPv6”，显然是不可行的。这样，向 IPv6 过渡**只能采用逐步演进的办法**，同时，还必须使新安装的 IPv6 系统能够**向后兼容**。这就是说，IPv6 系统必须能够接收和转发 IPv4 分组，并且能够为 IPv4 分组选择路由。

下面介绍两种向 IPv6 过渡的策略，即使用双协议栈和使用隧道技术[RFC 2473, 2529, 2893, 3056]。

双协议栈(dual stack)是指在完全过渡到 IPv6 之前，使一部分主机（或路由器）装有两个协议栈，一个 IPv4 和一个 IPv6（如图 6-58）。因此双协议栈主机（或路由器）既能够和 IPv6 的系统通信，又能够和 IPv4 的系统进行通信。双协议栈的主机（或路由器）记为 IPv6/IPv4，表明它具有两种 IP 地址：一个 IPv6 地址和一个 IPv4 地址。

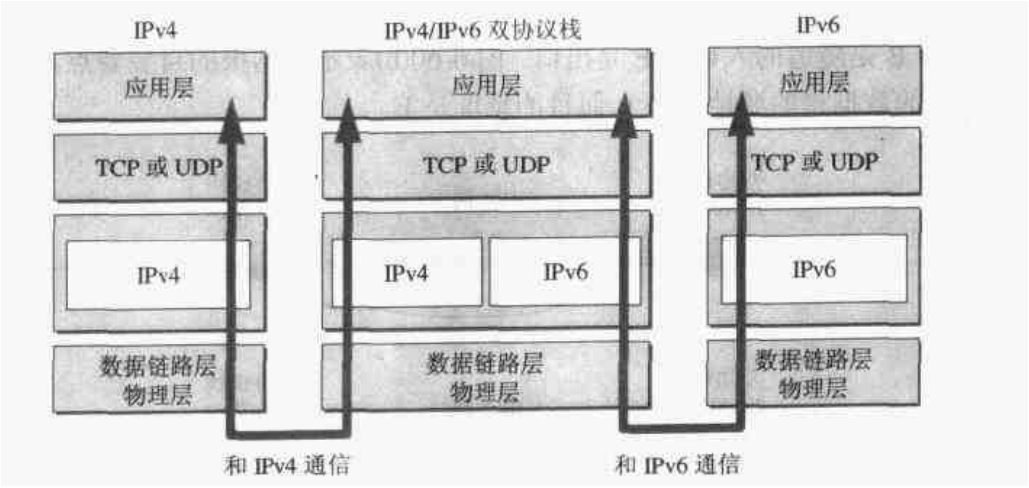


图 6-58　双协议栈

双协议栈主机在和 IPv6 主机通信时是采用 IPv6 地址，而和 IPv4 主机通信时就采用 IPv4 地址。但双协议栈主机怎样知道目的主机是采用哪一种地址呢？它是使用域名系统 DNS（这将在 8.2 节中讨论）来查询。若 DNS 返回的是 IPv4 地址，双协议栈的源主机就使用 IPv4 地址。但当 DNS 返回的是 IPv6 地址，源主机就使用 IPv6 地址。

图 6-59 所示的情况是源主机 A 和目的主机 F 都使用 IPv6，所以 A 向 F 发送 IPv6 数据报，路径是 A→B→C→D→E→F。中间 B 到 E 这段路径是 IPv4 网络，因此路由器 B 不能向 C 转发 IPv6 数据报，因为 C 只使用 IPv4 协议。由于 B 是 IPv6/IPv4 路由器，因此路由器 B 将 IPv6 数据报首部转换为 IPv4 数据报首部后发送给 C。等到 IPv4 数据报到达 IPv4 网络的出口路由器 E 时（E 也是 IPv6/IPv4 路由器），再恢复成原来的 IPv6 数据报。需要注意的是：IPv6 首

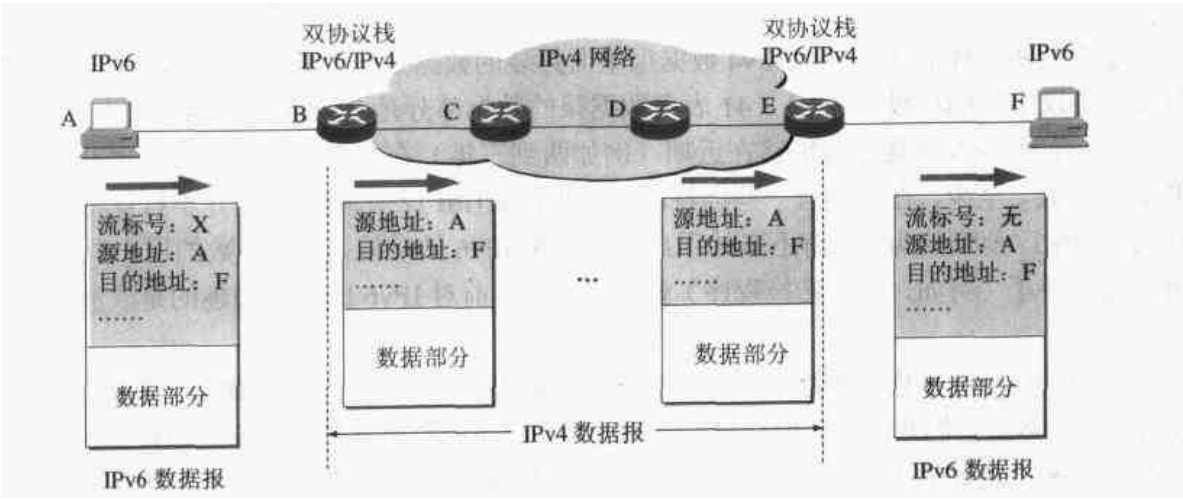


图 6-59　使用双协议栈进行从 IPv4 到 IPv6 的过渡

部中的某些字段却无法恢复。例如，原来 IPv6 首部中的流标号 X 在最后恢复出的 IPv6 数据报中只能变为空缺。这种信息的损失是使用首部转换方法所不可避免的。

向 IPv6 过渡的另一种方法是隧道技术(tunneling)。图 6-60 给出了隧道技术的工作原理。这种方法的要点就是在 IPv6 数据报要进入 IPv4 网络时，将 IPv6 数据报封装成为 IPv4 数据报（整个的 IPv6 数据报变成了 IPv4 数据报的数据部分）。然后 IPv6 数据报就在 IPv4 网络的隧道中传输。当 IPv4 数据报离开 IPv4 网络中的隧道时再将其数据部分（即原来的 IPv6 数据报）交给主机的 IPv6 协议栈。图 6-60(a)表示在 IPv4 网络中打通了一个从 B 到 E 的“IPv6 隧道”，路由器 B 是隧道的入口而 E 是出口。图 6-60(b)表示数据报的封装要点。请读者注意，在隧道中传送的数据报的源地址是 B 而目的地址是 E。

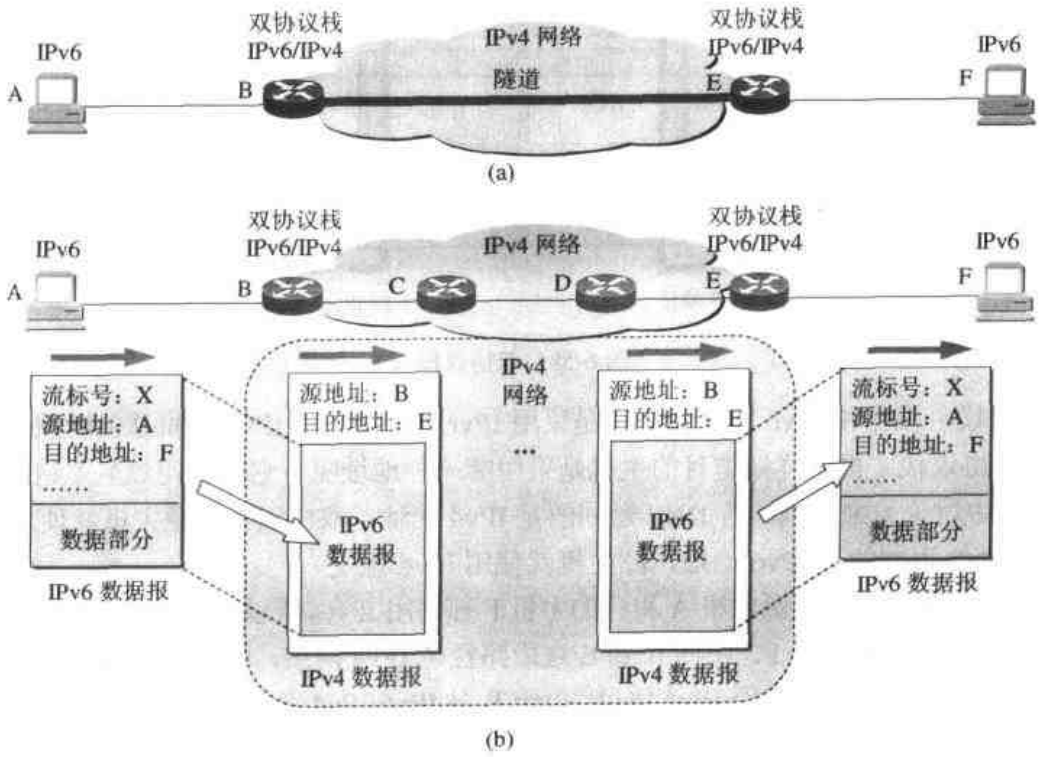


图 6-60　使用隧道技术进行从 IPv4 到 IPv6 的过渡

(a)在 IPv4 网络的隧道中传送 IPv6 数据报；(b)隧道不改变 IPv6 数据报的首部

要使双协议栈的主机知道 IPv4 数据报里面封装的数据是一个 IPv6 数据报，就必须将 IPv4 首部的协议字段的值设置为 41（41 表示数据报的数据部分是 IPv6 数据报）。

现在有不少人怀疑是否能够在近期（例如两到三年）在整个因特网范围实现从 IPv4 到 IPv6 的过渡。至少，在北美有一些因特网服务提供者(ISP)表示他们近期并不打算将其路由器升级到 IPv6。他们认为，只有不多的用户需要使用 IPv6 的功能，而对多数的用户只要对 IPv4 协议打些补丁（例如，地址转换程序）就可以了。目前对 IPv6 比较感兴趣的是欧洲和亚洲的一些用户。

从 20 世纪 90 年代初期起，就陆续出现了许多新的网络层协议，如 IPv6，多播协议，以及 RSVP 等，然而它们并没有立即获得广泛的应用。这里的原因就是：将新的协议引入网络层就像改造一座已建好的大楼的地基（大楼里面已有人办公和居住）。如果不暂时将这些人迁出大楼，甚至拆除大楼的某些部分，就很难进行大楼地基的改造。相反，因特网上的应用层协议却

能比较容易地添加上去，就像我们可以较容易地改变大楼内中一些房间里的装璜那样。因此，在可预见的将来，作为因特网基础的网络层的改变估计会比应用层的改变缓慢得多。

6.8.6 ICMPv6

和 IPv4 一样，IPv6 也需要使用 ICMP。但旧版本的、适合于 IPv4 的 ICMP 并不能满足 IPv6 全部的要求。因此，叫做 ICMPv6 的新版本在 1998 年底也问世了[RFC 2463]，它目前还只是草案标准协议。

ICMPv6 的报文格式和 IPv4 使用的 ICMP 的相似（见图 6-26），即前 4 个字节的字段名称都是一样的，但 ICMPv6 将第 5 个字节起的后面部分作为报文主体。ICMPv6 将其报文种类划分为两大类，即**差错报文**(error message)和**提供信息的报文**(informational message)，并取消了使用得很少的 ICMP 报文。差错报文的类型字段的最高位是 0，因此其类型字段的值是 0 到 127。提供信息的报文的类型字段的最高位是 1，其值是 128 到 255。但在[RFC 2463]中只定义了 6 种类型的的 ICMPv6 报文（见表 6-10 所示）。

表 6-10 [RFC 2463]定义的 ICMPv6 报文

ICMP 报文种类	类型的值	ICMP 报文的类型
差错报告报文	1	目的站不可达
	2	分组太长
	3	时间超过
	4	参数问题
提供信息的报文	128	回送(Echo)请求
	129	回送回答

ICMPv6 报文的前面是 IPv6 首部和零个或更多的 IPv6 扩展首部。在 ICMPv6 前面的一个首部中的“下一个首部字段”的值应当置为 58。请注意：这和 IPv4 中标志 ICMP 的值不同，在 IPv4 中标志 ICMP 的值是 1。

习题

6-01 网络互连有何实际意义？进行网络互连时，有哪些共同的问题需要解决？

6-02 小写和大写开头的英文名字 internet 和 Internet 在意思上有何重要区别？

6-03 作为中间系统，转发器、网桥、路由器和网关有何区别？

6-04 试简单说明下列协议的作用：
IP、ARP、RARP 和 ICMP。

6-05 IP 地址分为几类？各如何表示？IP 地址的主要特点是什么？

6-06 试根据 IP 地址的规定，计算出表 6-1 中的各项数据。

6-07 试说明 IP 地址与硬件地址的区别。为什么要使用这两种不同的地址？

6-08 IP 地址方案与我国的电话号码体制的主要不同点是什么？

6-09 （1）子网掩码为 255.255.255.0 代表什么意思？
（2）一网络的现在掩码为 255.255.255.248，问该网络能够连接多少个主机？
（3）一 A 类网络和一 B 类网络的子网号 subnet-id 分别为 16 bit 和 8 bit 的 1，问这两个网络的子网掩码有何不同？

（4）一个 B 类地址的子网掩码是 255.255.240.0。试问在其中每一个子网上的主机数最多是多少？

（5）一个 A 类网络的子网掩码为 255.255.0.255，它是否为一个有效的子网掩码？

（6）某个 IP 地址的十六进制表示是 C22F1481，试将其转换为点分十进制的形式。这个地址是哪一类 IP 地址？

（7）C 类网络使用子网掩码有无实际意义？为什么？

6-10 试辨认以下 IP 地址的网络类别。

（1）128.36.199.3

（2）21.12.240.17

（3）183.194.76.253

（4）192.12.69.248

（5）89.3.0.1

（6）200.3.6.2

6-11 IP 数据报中的首部检验和并不检验数据报中的数据。这样做的最大好处是什么？坏处是什么？

6-12 当某个路由器发现一数据报的检验和有差错时，为什么采取丢弃的办法而不是要求源站重传此数据报？计算首部检验和为什么不采用 CRC 检验码？

6-13 在因特网中将 IP 数据报分片传送的数据报在最后的目的主机进行组装。还可以有另一种做法，即数据报片通过一个网络就进行一次组装。试比较这两种方法的优劣。

6-14 一个 3200 bit 长的 TCP 报文传到 IP 层，加上 160 bit 的首部后成为数据报。下面的互联网由两个局域网通过路由器连接起来。但第二个局域网所能传送的最长数据帧中的数据部分只有 1200 bit。因此数据报在路由器必须进行分片。试问第二个局域网向其上层要传送多少比特的数据（这里的“数据”当然指的是局域网看见的数据）？

6-15 （1）有人认为：“ARP 协议向网络层提供了转换地址的服务，因此 ARP 应当属于数据链路层。”这种说法为什么是错误的？

（2）试解释为什么 ARP 高速缓存每存入一个项目就要设置 10～20 分钟的超时计时器。这个时间设置得太大或太小会出现什么问题？

（3）至少举出两种不需要发送 ARP 请求分组的情况（即不需要请求将某个目的 IP 地址解析为相应的硬件地址）。

6-16 设某路由器建立了如下路由表（这三列分别是目的网络、子网掩码和下一跳路由器，若直接交付则最后一列表示应当从哪一个接口转发出去）：

128.96.39.0	255.255.255.128	接口 0
128.96.39.128	255.255.255.128	接口 1
128.96.40.0	255.255.255.128	R_2
192.4.153.0	255.255.255.192	R_3
*（默认）	–	R_4

现共收到 5 个分组，其目的站 IP 地址分别为：

（1）128.96.39.10

（2）128.96.40.12

（3）128.96.40.151

（4）192.4.153.17

（5）192.4.153.90

试分别计算其下一跳。

6-17 某单位分配到一个 B 类 IP 地址，其 net-id 为 129.250.0.0。该单位有 4000 多台机器，分布在 16 个不同的地点。如选用子网掩码为 255.255.255.0，试给每一个地点分配一个子网号码，并算出每个地点主机号码的最小值和最大值

6-18 一个数据报长度为 4000 字节（固定首部长度）。现在经过一个网络传送，但此网络能够传送的最大数据长度为 1500 字节。试问应当划分为几个短些的数据报片？各数据报片的数据字段长度、片偏移字段和 MF 标志应为何数值？

6-19 分两种种情况（使用子网掩码和使用 CIDR）写出因特网的 IP 层查找路由的算法。

6-20 试找出可产生以下数目的 A 类子网的子网掩码（采用连续掩码）。

（1）2,（2）6,（3）30,（4）62,（5）122,（6）250。

6-21 以下有 4 个子网掩码。哪些是不推荐使用的？

（1）176.0.0.0,（2）96.0.0.0,（3）127.192.0.0,（4）255.128.0.0。

6-22 有如下的 4 个/24 地址块，试进行最大可能的聚合。

212.56.132.0/24,

212.56.133.0/24,

212.56.134.0/24,

212.56.135.0/24。

6-23 有两个 CIDR 地址块 208.128/11 和 208.130.28/22。是否有哪一个地址块包含了另一个地址？如果有，请指出，并说明理由。

6-24 一个自治系统有 5 个局域网，其连接图如图 6-61 所示。LAN_2 至 LAN_5 上的主机数分别为：91, 150, 3 和 15。该自治系统分配到的 IP 地址块为 30.138.118/23。试给出每一个局域网的地址块（包括前缀）。

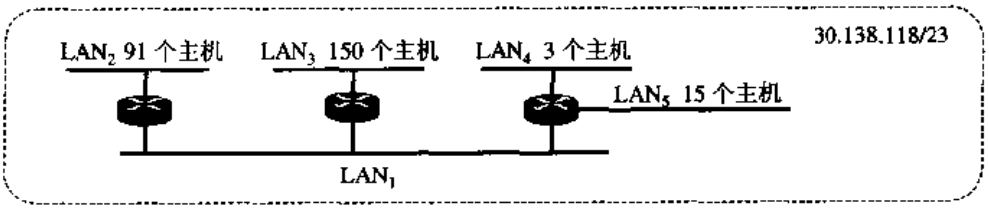


图 6-61　习题 6-24 的图

6-25 一个大公司有一个总部和三个下属部门。公司分配到的网络前缀是 192.77.33/24。公司的网络布局如图 6-62 所示。总部共有五个局域网，其中的 LAN_1～LAN_4 都连接到路由器 R_1 上，R_1 再通过 LAN_5 与路由器 R_5 相连。R_5 和远地的三个部门的局域网 LAN_6～LAN_8 通过广域网相连。每一个局域网旁边标明的数字是局域网上的主机数。试给每一个局域网分配一个合适的网络前缀。

6-26 以下地址中的哪一个和 86.32/12 匹配？请说明理由。

（1）86.33.224.123；（2）86.79.65.216；（3）86.58.119.74；（4）86.68.206.154。

6-27 以下的地址前缀中的哪一个地址 2.52.90.140 匹配？请说明理由。

（1）0/4；（2）32/4；（3）4/6；（4）80/4。

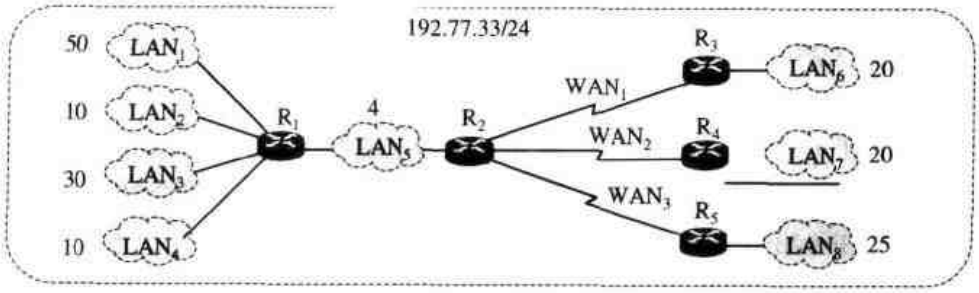


图 6-62　习题 6-25 的图

6-28　下面的那样前缀中的哪一个和地址 152.7.77.159 及 152.31.47.252 都匹配？请说明理由。

（1）152.40/13；（2）153.40/9；（3）152.64/12；（4）152.0/11。

6-29　与下列掩码相对应的网络前缀各有多少比特？

（1）192.0.0.0；（2）240.0.0.0；（3）255.224.0.0；（4）255.255.255.252。

6-30　一个网络有几个子网，其中的一个已经分配了子网掩码 74.178.247.96/29。试问下列网络前缀中的哪些不能再分配给其他的子网？

（1）74.178.247.120/29；（2）74.178.247.64/29；（3）74.178.247.80/28；

（4）74.178.247.104/29。

6-31　IGP 和 EGP 这两类协议的主要区别是什么？

6-32　试简述 RIP、OSPF 和 BGP 路由选择协议的主要特点。

6-33　RIP 使用 UDP，OSPF 使用 IP，而 BGP 使用 TCP。这样做有何优点？为什么 RIP 周期性地和邻站交换路由信息而 BGP 却不这样做？

6-34　假定网络中的路由器 B 的路由表有如下的项目（这三列分别表示“目的网络”、“距离”和“下一跳路由器”）

N_1	7	A
N_2	2	C
N_6	8	F
N_8	4	E
N_9	4	F

现在 B 收到从 C 发来的路由信息（这两列分别表示“目的网络”和“距离”）：

N_2	4
N_3	8
N_5	4
N_8	3
N_9	5

试求出路由器 B 更新后的路由表（详细说明每一个步骤）。

6-35　假定网络中的路由器 A 的路由表有如下的项目（格式同上题）：

N_1	4	B
N_2	2	C
N_3	1	F
N_4	5	G

现在 A 收到从 C 发来的路由信息（格式同上题）：

N_1	2
N_2	1
N_3	3

试求出路由器 A 更新后的路由表（详细说明每一个步骤）。

6-36 IGMP 协议的要点是什么？隧道技术是怎样使用的？

6-37 建议的 IPv6 没有首部检验和。这样做的优缺点是什么？

6-38 在 IPv4 首部中有一个“协议”字段，但在 IPv6 的固定首部中却没有。这是为什么？

6-39 当使用 IPv6 时，是否 ARP 协议需要改变？如果需要改变，那么应当概念性的改变还是技术性的改变？

6-40 设每隔 1 微微秒就分配出 100 万个 IPv6 地址。试计算大约要用多少年才能将 IPv6 地址空间全部用光。可以和宇宙的年龄（大约有 100 亿年）进行比较。

6-41 试将以下的 IPv6 地址用零压缩方法写成简洁形式：

（1）0000:0000:0F53:6382:AB00:67DB:BB27:7332

（2）0000:0000:0000:0000:0000:0000:004D:ABCD

（3）0000:0000:0000:AF36:7328:0000:87AA:0398

（4）2819:00AF:0000:0000:0000:0035:0CB2:B271

6-42 从 IPv4 过渡到 IPv6 的方法有哪些？

第7章 运 输 层

运输层是整个网络体系结构中的关键层次之一。本章讨论TCP/IP体系中的运输协议UDP和TCP。TCP比UDP复杂得多。最重要的是应弄清TCP的各种机制（如面向连接的可靠服务、序号、确认、窗口、拥塞控制等），以及TCP有关连接管理和状态图的概念。

7.1 运输层协议概述

从通信和信息处理的角度看，**运输层向它上面的应用层提供通信服务**，它属于面向通信部分的最高层，同时也是用户功能中的最低层。在通信子网中没有运输层。运输层只存在于通信子网以外的主机中（见图7-1）。

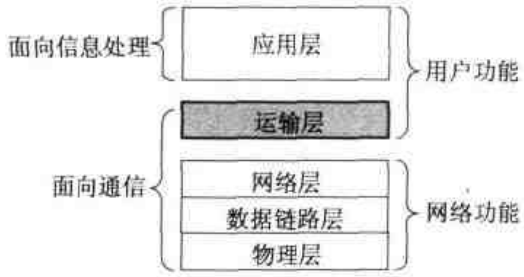


图7-1 运输层在层次体系结构中的地位

下面再通过图7-2的示意图来说明运输层的作用。设局域网1上的主机A和局域网2上的主机B通过互连的广域网进行通信。既然我们的IP协议能够将源主机发送出的分组按照首部中的目的地址送交到目的主机，那么，为什么还需要再设置一个运输层呢？

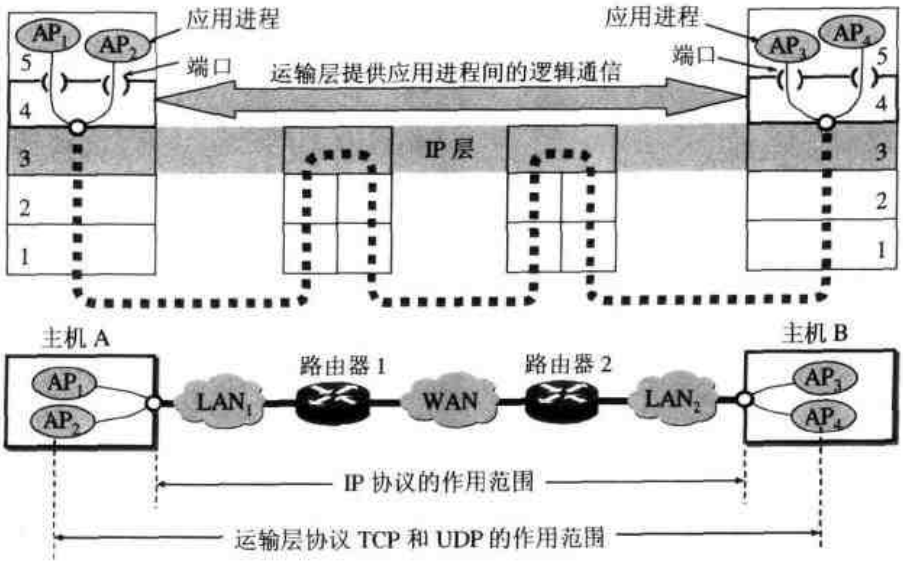


图7-2 运输层为相互通信的应用进程提供了逻辑通信

严格地讲，两个主机进行通信实际上就是两个主机中的**应用进程互相通信**。IP协议虽然能把分组送到目的主机，但是这个分组还停留在主机的网络层而没有交付给主机中的应用进程（请注意：IP地址是标志在因特网中的一个主机，而**不是标志主机中的应用进程**）。由于通信的两个端点是源主机和目的主机中的应用进程，因此应用进程之间的通信又称为**端到端**

的通信。在一个主机中经常有多个应用进程同时分别和另一个主机中的多个应用进程通信。例如，某用户在使用浏览器查找某网站的信息时，其主机的应用层运行浏览器客户进程。如果在浏览网页的同时，还要用电子邮件给网站发送反馈意见，那么主机的应用层就还要运行电子邮件的客户进程。在图 7-2 中，主机 A 的应用进程 1 和主机 B 的应用进程 3 通信，而与此同时，应用进程 2 也和对方的应用进程 4 通信。因此，运输层的一个很重要的功能就是**复用和分用**。应用层不同进程的报文通过不同的端口（在后面的 7.2.2 节还要详细讨论端口的概念）向下交到运输层，再往下就共用网络层提供的服务。当这些报文沿着图中的虚线到达目的主机后，目的主机的运输层就使用其分用功能，通过不同的端口将报文分别交付到相应的应用进程。图 7-2 中两个运输层之间有一个粗的双向箭头，写明“**运输层提供应用进程间的逻辑通信**”。“逻辑通信”的意思是：**运输层之间的通信好像是沿水平方向传送数据。但事实上这两个运输层之间并没有一条水平方向的物理连接。要传送的数据是沿着图中虚线方向传送的。**

从这里可以看出网络层和运输层有很大的区别。**运输层为应用进程之间提供端到端的逻辑通信，但网络层是为主机之间提供逻辑通信**（见图 7-3）。如果运输层只有这一点复用和分用功能，那么取消运输层而让网络层增加这一点功能也未尝不可。然而正如下面就要指出的，运输层还具有网络层无法代替的许多其他重要功能。



图 7-3　运输层协议和网络层协议的主要区别

其次，运输层还要对收到的报文进行**差错检测**。大家应当还记得，在网络层，IP 数据报首部中的检验和字段，只检验首部是否出现差错而不检查数据部分。

再次，根据应用的不同，运输层需要有两种不同的运输协议，即**面向连接的 TCP 和无连接的 UDP**，而网络层无法同时实现这两种协议。

OSI 使用了简洁的抽象方法将运输层与其上下层之间的关系归纳如图 7-4 所示。

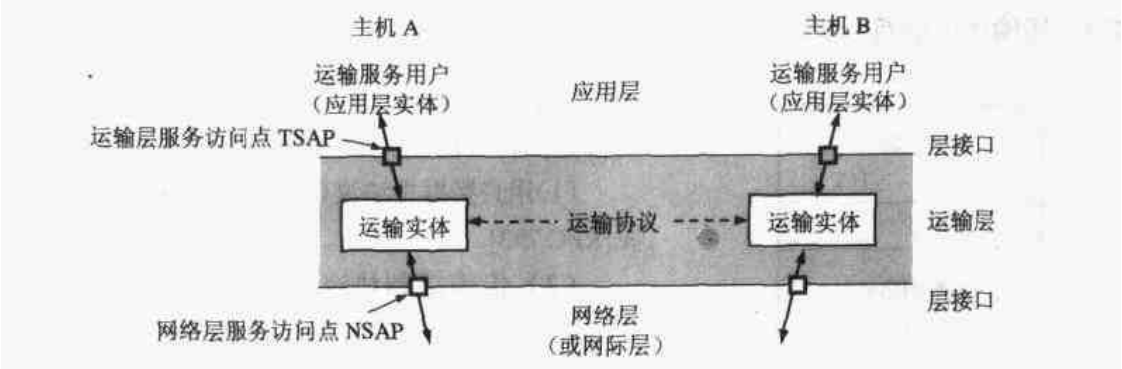


图 7-4　运输层与其上下层之间的关系的 OSI 表示法

根据 OSI 的观点，运输层中向应用层提供运输服务的是**运输实体**。使用运输服务的是**运输服务用户**（也就是应用层中的各种应用进程，或应用层实体，但不是使用计算机的最终用户）。运输层中的两个对等运输实体之间的通信遵循着运输协议。**运输协议保证了运输层能够向应用层提供运输服务**。运输层提供的运输服务也使用了下面网络层向上提供的网络服务。TSAP 和 NSAP 分别是运输层服务访问点和网络层服务访问点，它们都是**层与层之间交换信息的抽象接口**。

我们要再次强调指出，**运输层向高层用户屏蔽了下面通信子网的细节**（如网络拓扑、所采用的协议等），**它使应用进程看见的就是好像在两个运输层实体之间有一条端到端的逻辑通信信道**，但这条逻辑通信信道对上层的表现却因运输层使用的不同协议而有很大的差别。当运输层**采用面向连接的 TCP 协议时，尽管下面的网络是不可靠的**（即只提供尽最大努力服务），但这种逻辑通信信道就相当于**一条全双工的可靠信道**。但当运输层采用**无连接的 UDP 协议**时，这种逻辑通信信道则是**一条不可靠信道**。在图 7-5 中我们将可靠信道画成一个管道，这意味着报文在这样的“管道”中运输时，可以做到无差错、按序（接收的顺序和发送的顺序一样）、无丢失和无重复。对不可靠信道就用一个云状网络来表示。不可靠信道的特点就是**不保证交付**，即：接收时可能不按序、可能会出现丢失和重复，但我们不能将这里所说的“不可靠信道”理解为“收下来的数据不可靠，因为里面有差错”。我们知道，运输层检查出到达的报文有差错时就将其丢弃因而不收下有差错的报文。

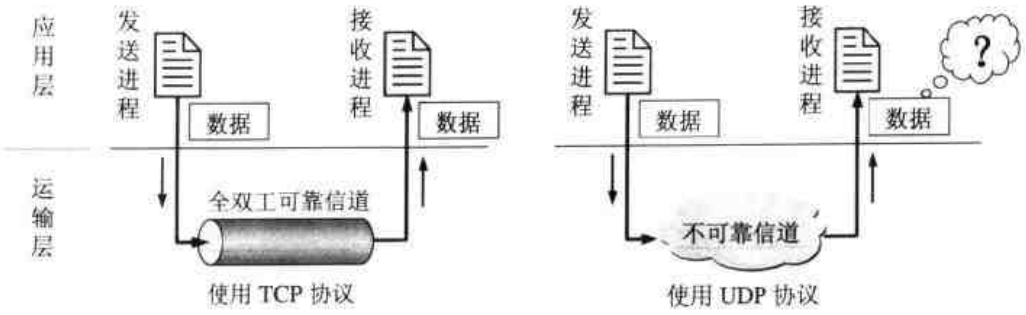


图 7-5　运输层向上提供可靠的和不可靠的逻辑通信信道

最后再强调一下，我们说运输层提供可靠的交付，是指运输层将数据可靠地交付给接收端的**应用层**。

7.2　TCP/IP 体系中的运输层

7.2.1　运输层中的两个协议

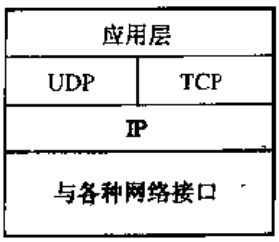


图 7-6　TCP/IP 体系中的运输层协议

TCP/IP 的运输层有两个不同的协议（图 7-6），它们都是因特网的正式标准，即：

（1）**用户数据报协议** UDP (User Datagram Protocol)，见[RFC 793]。

（2）**传输控制协议** TCP (Transmission Control Protocol)，见[RFC 768]。

按照 OSI 的术语，两个对等运输实体在通信时传

送的数据单位叫作**运输协议数据单元** TPDU (Transport Protocol Data Unit)。但在 TCP/IP 体系中，则根据所使用的协议是 TCP 或 UDP，分别称之为 **TCP 报文段**(segment)或 **UDP 报文**或**用户数据报**。

UDP 在传送数据之前**不需要先建立连接**。远地主机的运输层在收到 UDP 报文后，不需要给出任何确认。虽然 UDP 不提供可靠交付，但在某些情况下 UDP 是一种最有效的工作方式。

TCP 则**提供面向连接的服务**。在传送数据之前必须先建立连接，数据传送结束后要释放连接。TCP **不提供广播或多播服务**。由于 TCP 要提供可靠的、面向连接的运输服务，因此不可避免地增加了许多的开销，如确认、流量控制、计时器以及连接管理等。这不仅使协议数据单元的首部增大很多，还要占用许多的处理机资源。

这里还要强调两点：

（1）运输层的 UDP 用户数据报与网际层的 IP 数据报有很大的区别。IP 数据报要经过互连网中许多路由器的存储转发，但 **UDP 用户数据报是在运输层的端到端抽象的逻辑信道中传送的**。这个逻辑信道虽然也是尽最大努力交付（因而不可靠），但运输层的这个逻辑信道并不经过路由器（运输层看不见路由器），因为路由器只有下三层协议而没有运输层。IP 数据报虽然经过路由器进行转发，但用户数据报只是 IP 数据报中的数据，因此路由器看不见有用户数据报经过它。这两种数据报不能弄混。

（2）TCP 是运输层的连接，它和网络层中的虚电路（如 X.25 所使用的）完全不同。**TCP 报文段是在运输层抽象的的端到端逻辑信道中传送，这种信道是可靠的全双工信道。但这样的信道却不知道究竟经过了哪些路由器**，而这些路由器也根本不知道上面的运输层是否建立了 TCP 连接。当 IP 数据报的传输路径中增加或减少了一些路由器时，上层的 TCP 连接都不会发生变化，因为上层的 TCP 根本不知道下层所发生的这些事情。然而在 X.25 建立的虚电路所经过的交换结点中，都必须保存 X.25 虚电路的状态信息。

7.2.2 端口的概念

UDP 和 TCP 都使用了与应用层接口处的**端口**(port)与上层的应用进程进行通信。端口是个非常重要的概念，因为应用层的各种进程是通过相应的端口与运输实体进行交互。因此在运输协议数据单元（即 TCP 报文段或 UDP 用户数据报）的首部中都要写入**源端口号**和**目的端口号**。当运输层收到 IP 层交上来的数据，就要根据其目的端口号来决定应当通过哪一个端口上交给目的应用进程。图 7-7 中在应用层和运输层之间的小方框就代表端口。其实在其他各层之间的信息交互都必须通过类似的端口，但这里没有将它们画出。

用 OSI 的术语，图中的端口就是**运输层服务访问点** TSAP。**端口的作用就是让应用层的各种应用进程都能将其数据通过端口向下交付给运输层，以及让运输层知道应当将其报文段**中的**数据向上通过端口交付给应用层相应的进程**。从这个意义上讲，端口是用来标志应用层的进程。图 7.7 强调了运输层 TCP 和 UDP 的复用和分用的概念。由于使用了复用和分用技术，在运输层与网络层的交互中已看不见各种应用进程，而只有 TCP 报文段或 UDP 用户数据报。IP 层也使用类似的复用和分用技术，因而在网络层和链路层的交互中也只看得见 IP 数据报。上述概念在网络中是十分重要的（见图 7-7）。

端口用一个 16 bit **端口号**进行标志。对于不同的计算机，端口的具体实现方法可能有很大的差别，因为这取决于计算机的操作系统。因此端口的基本概念就是：应用层的源进程将

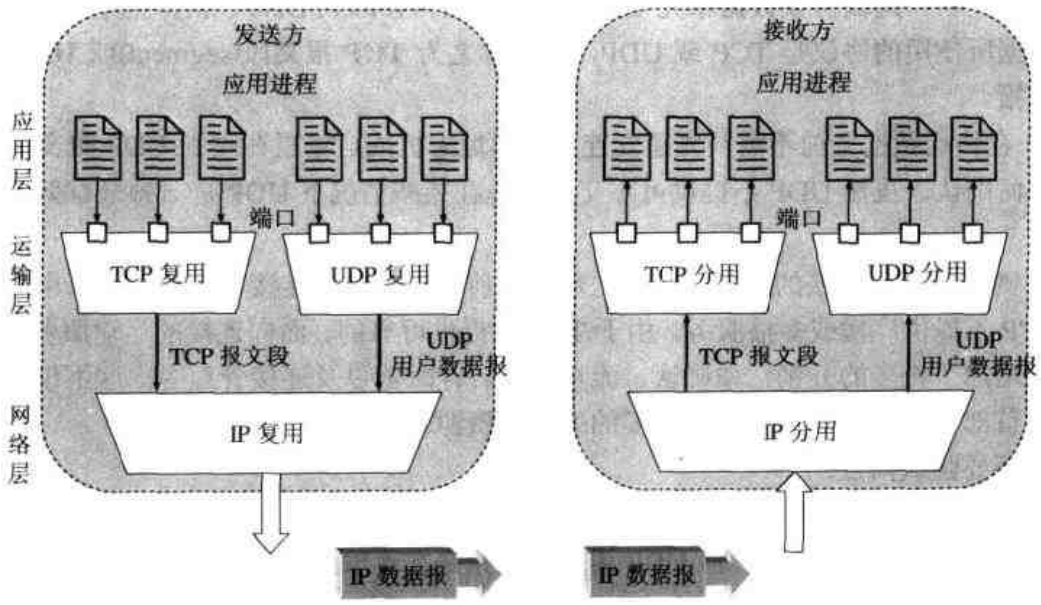


图 7-7　端口在进程之间的通信中所起的作用

报文发送给运输层的某个端口，而应用层的目的进程从端口接收报文。**但端口号只具有本地意义**，即端口号只是为了标志**本计算机**应用层中的各进程。在因特网中不同计算机的相同端口号是**没有联系**的。16 bit 的端口号可允许有 64K 个端口号，这个数目对一个计算机来说是足够用的。

端口号分为两类。一类是由**因特网指派名字和号码公司** ICANN 负责分配给一些常用的应用层程序固定使用的**熟知端口**(well-known port)，其数值一般为 0～1023，见[RFC 1700]。例如：

应用程序	FTP	TELNET	SMTP	DNS	TFTP	HTTP	SNMP	SNMP(trap)
熟知端口	21	23	25	53	69	80	161	162

“熟知”就表示这些端口号是 TCP/IP 体系确定并公布的，因而是所有用户进程都知道的。当一种新的应用程序出现时，必须为它指派一个熟知端口，否则其他的应用进程就无法和它进行交互。在应用层中的各种不同的服务器进程不断地检测分配给它们的熟知端口，以便发现是否有某个客户进程要和它通信。另一类则是**一般端口**，用来随时分配给请求通信的客户进程。

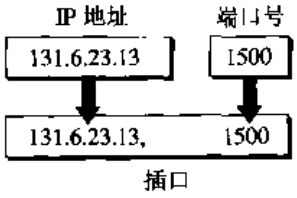


图 7-8　插口和端口、IP 地址的关系

为了在通信时不致发生混乱，就必须把端口号和主机的 IP 地址结合在一起使用。一个 TCP 连接由它的两个端点来标志，而每一个端点又是由 IP 地址和端口号决定的。因此，TCP 使用“**连接**”(而不仅仅是“端口”)作为最基本的抽象，同时将 TCP 连接的端点称为**插口**(socket)，或**套接字**、**套接口**[①]。插口的概念并不复

① 注：Socket 一词目前没有标准译名。虽然广为流行的译名还有“套接字”和“套接口”，但编者认为译名“插口”可能较为准确，因此本书使用不甚流行的译名“插口”。

杂，但非常重要。图 7-8 给出了插口和端口、IP 地址的关系。例如，若 IP 地址 = 131.6.23.13，端口号 = 1500，则：

插口 = (131.6.23.13, 1500)。

上面讲的是面向连接的 TCP。若使用无连接的 UDP，虽然在相互通信的两个进程之间没有一条虚连接，但发送端 UDP 一定有一个发送端口而在接收端 UDP 也一定有一个接收端口，因而也同样可使用插口的概念。这样才能区分多个主机中同时通信的多个进程。

值得注意的是，插口这个名词很容易使人将一些概念弄混淆，因为同**一个名词 socket 却有多种不同的意思**。例如：

（1）允许应用程序访问连网协议的**应用编程接口 API** (Application Programming Interface)，也就是在运输层和应用层之间的一种接口，称为 socket API，并简称为 socket。

（2）在 socket API 中使用的一个**函数名**也叫作 socket。

（3）调用 socket 函数的**端点**称为 socket，如“创建一个**数据报** socket”。

（4）调用 socket 函数时其**返回值**称为 socket 描述符，可简称为 socket。

（5）在操作系统内核中连网协议的 Berkeley 实现，称为 socket **实现**。

上面的这些 socket 的意思都和本章中的 socket（指 IP 地址和端口号的**组合**）不同。

下面将分别介绍 UDP 和 TCP 的要点。UDP 比较简单，本章主要的**篇幅**是讨论 TCP。

7.3 用户数据报协议 UDP

7.3.1 UDP 概述

用户数据报协议 UDP 只在 IP 的数据报服务之上增加了很少一点的功能，这就是端口的功能（有了端口，运输层就能进行复用和分用）和差错检测的功能。虽然 UDP 用户数据报只能提供不可靠的交付，但 UDP 在某些方面有其特殊的优点，例如：

（1）发送数据之前不需要建立连接（当然发送数据结束时也没有连接需要释放），因此减少了开销和发送数据之前的时延。

（2）UDP 不使用拥塞控制，也不保证可靠交付，因此主机不需要维持具有许多参数的、复杂的连接状态表。

（3）UDP 用户数据报只有 8 个字节的首部开销，比 TCP 的 20 个字节的首部要短。

（4）由于 UDP 没有拥塞控制，因此网络出现的拥塞不会使源主机的发送速率降低。这对某些实时应用是很重要的。很多的实时应用（如 IP 电话、实时视频会议等）要求源主机以恒定的速率发送数据，并且允许在网络发生拥塞时丢失一些数据，但却不允许数据有太大的时延。UDP 正好适合这种要求。

表 7-1 给出了一些应用和应用层协议主要使用的运输层协议(UDP 或 TCP)。

表 7-1 使用 UDP 和 TCP 协议的各种应用和应用层协议

应　用	应用层协议	运输层协议
名字转换	DNS	UDP
文件传送	TFTP	UDP
路由选择协议	RIP	UDP
IP 地址配置	BOOTP, DHCP	UDP

续表

应　用	应用层协议	运输层协议
网络管理	SNMP	UDP
远程文件服务器	NFS	UDP
IP 电话	专用协议	UDP
流式多媒体通信	专用协议	UDP
多播	IGMP	UDP
电子邮件	SMTP	TCP
远程终端接入	TELNET	TCP
万维网	HTTP	TCP
文件传送	FTP	TCP

虽然某些实时应用需要使用没有拥塞控制的 UDP，但当很多的源主机同时都向网络发送高速率的实时视频流时，网络就有可能发生拥塞，结果大家都无法正常接收。因此 UDP 不使用拥塞控制功能可能会引起网络产生严重的拥塞问题。

还有一些使用 UDP 的实时应用需要对 UDP 的不可靠的传输进行适当的改进以减少数据的丢失。在这种情况下，应用进程本身可在不影响应用的实时性的前提下增加一些提高可靠性的措施，如采用前向纠错或重传已丢失的报文。

UDP 与应用层之间的端口都是用报文队列来实现的，见图 7-9。

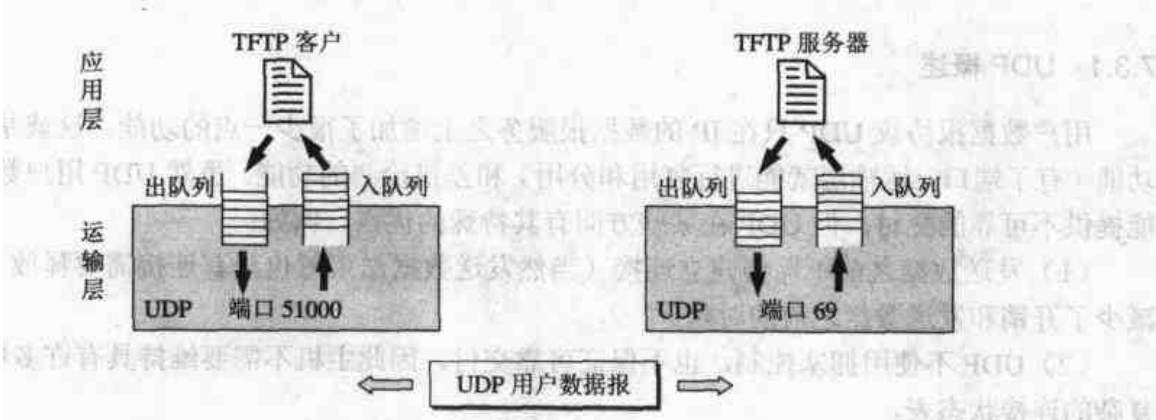


图 7-9　UDP 中的队列

我们假定应用层使用简单文件传送协议 TFTP。TFTP 使用 UDP 传送报文，但 TFTP 服务器和 TFTP 客户所使用的端口是不一样的。

TFTP 服务器进程是一直在运行着，等待 TFTP 客户进程的服务请求。因此，服务器的端口必须使用熟知端口号。TFTP 服务器的熟知端口号是 69。在 TFTP 客户端，当进程启动时，就向操作系统请求一个临时的一般端口号，例如，51000。然后操作系统就为该进程创建两个队列：**入队列**和**出队列**。只要进程在执行，这两个队列就一直存在。当进程终止时，入队列和出队列以及临时端口号就一起被撤消。

客户进程将报文发送到出队列中。UDP 按队列中报文的先后顺序进行发送。在传送到 IP 层之前要给报文加上 UDP 首部，其中的目的端口是 TFTP 的熟知端口 69。然后 UDP 数据报就传送给 IP 层。出队列也有可能溢出。若发生溢出，则操作系统通知应用层的 TFTP 客户进程暂停发送。

但客户端收到来自 IP 层的报文时，UDP 先检查报文中的目的端口号是否正确。如不正确，UDP 就丢弃该报文，并请 ICMP 发送“端口不可达”差错报文给服务器端。若目的端口号正确，UDP 就将这收到的报文放在入队列的队尾，客户进程按报文到达的先后顺序将其一一取走。入队列也有可能发生溢出。若发生溢出，UDP 就丢弃收到的报文，但不通知对方。这里没有流量控制机制来通知发送方降低发送速率。UDP 还请 ICMP 发送“端口不可达”差错报文给服务器端。

在服务器端，创建队列的机制与客户端不同。但这里我们只讨论最简单的情况，即服务器使用熟知端口号创建其入队列和出队列，并一直使用这两个队列进行和外界的通信。

服务器在收到报文时，UDP 先检查到达的用户数据报的目的端口号是否为 69。如果是，就将此用户数据报放进入队列的队尾。如果端口号不正确，就丢弃此报文，并请 ICMP 协议发送“端口不可达”差错报文给客户端。不论从哪个客户发送来的报文，都在同一个入队列中排队。若入队列溢出，则丢弃队尾的报文（不通知对方），并请 ICMP 发送“端口不可达”差错报文给客户端。

当服务器要回答客户的请求时，就将要发送的报文传送到出队列，并使用请求服务的报文的源端口号作为回答报文的目的端口号。在添加上 UDP 首部后，将其传送给 IP 层。若出队列溢出，则操作系统通知服务器进程暂停发送。

7.3.2 UDP 用户数据报的首部格式

用户数据报 UDP 有两个字段：数据字段和首部字段。首部字段很简单，只有 8 个字节（图 7-10），由 4 个字段组成，**每个字段都是两个字节**。各字段意义如下：

（1）**源端口**　　源端口号。

（2）**目的端口**　　目的端口号。

（3）**长度**　　UDP 用户数据报的长度。

（4）**检验和**　　防止 UDP 用户数据报在传输中出错。

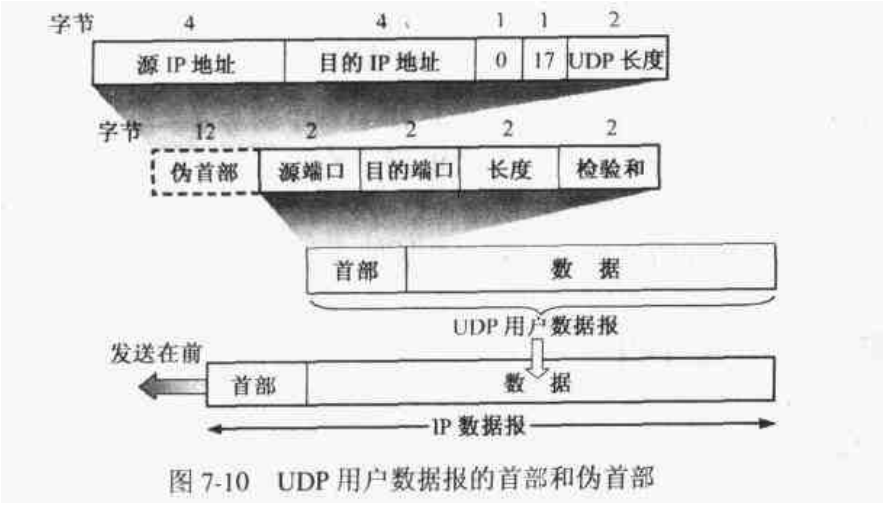


图 7-10　UDP 用户数据报的首部和伪首部

UDP 用户数据报首部中检验和的计算方法有些特殊。在计算检验和时，要在 UDP 用户数据报之前增加 12 个字节的**伪首部**。所谓“伪首部”是因为这种伪首部并不是 UDP 用户数据报真正的首部。只是在计算检验和时，临时和 UDP 用户数据报连接在一起，得到一个过渡的 UDP 用户数据报。检验和就是按照这个过渡的 UDP 用户数据报来计算的。伪首部既不

向下传送也不向上递交，而仅仅是为了计算检验和。图 7-10 的最上面给出了伪首部各字段的内容。

UDP 计算检验和的方法和计算 IP 数据报首部检验和的方法相似。但不同的是：IP 数据报的检验和只检验 IP 数据报的首部，但 UDP 的检验和是**将首部和数据部分一起都检验**。在发送端，首先是先将全零放入检验和字段。再将伪首部以及 UDP 用户数据报看成是由许多 16 bit 的字串接起来。若 UDP 用户数据报的数据部分不是偶数个字节，则要填入一个全零字节（但此字节不发送）。然后按二进制反码计算出这些 16 bit 字的和[①]。将此和的二进制反码写入检验和字段后，发送此 UDP 用户数据报。在接收端，将收到的 UDP 用户数据报连同伪首部（以及可能的填充全零字节）一起，按二进制反码求这些 16 bit 字的和。当无差错时其结果应为全 1。否则就表明有差错出现，接收端就应将此 UDP 用户数据报丢弃（也可以上交给应用层，但附上出现了差错的警告）。图 7-11 给出了一个计算 UDP 检验和的例子。这里假定用户数据报的长度是 15 字节，因此要添加一个全 0 的字节。读者可以自己检验一下在接收端是怎样对检验和进行检验的。不难看出，这种简单的差错检验方法的检错能力并不强，但它的好处是简单，处理起来较快。

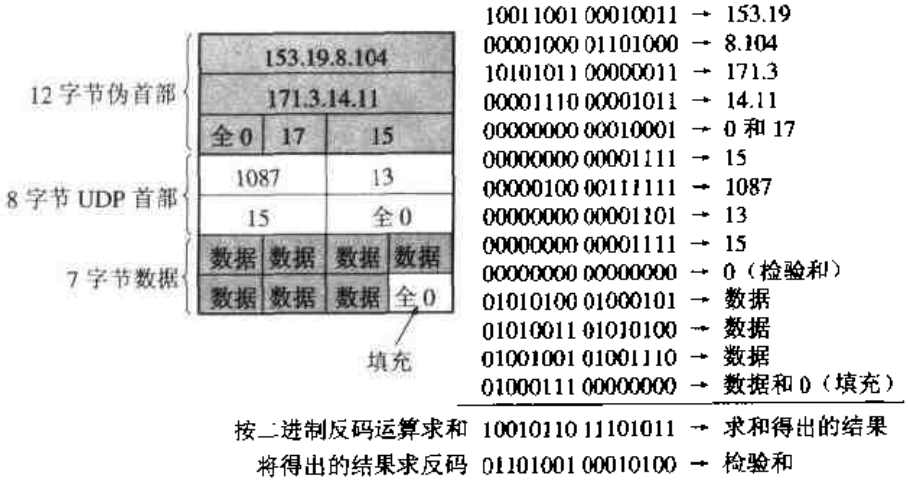


图 7-11　计算 UDP 检验和的例子

伪首部的第 3 字段是全零，第 4 个字段是 IP 首部中的协议字段的值。以前已讲过，对于 UDP，此协议字段值为 17。第 5 字段是 UDP 用户数据报的长度。因此我们可以看出，这样的检验和，既检查了 UDP 用户数据报的源端口号和目的端口号以及 UDP 用户数据报的数据部分，又检查了 IP 数据报的源 IP 地址和目的地址。

7.4　传输控制协议 TCP

7.4.1　TCP 概述

TCP 是 TCP/IP 体系中面向连接的运输层协议，它提供全双工的和可靠交付的服务。一

① 注：两个数进行二进制反码求和的运算很简单。它的规则是从低位到高位逐列进行计算。0 和 0 相加是 0，0 和 1 相加是 1，1 和 1 相加是 0 但要产生一个进位 1，加到下一列。若最高位相加后产生进位，则最后得到的结果要加 1。

定要记住，TCP 与 UDP 最大的区别就是：TCP 是**面向连接**的，而 UDP 是**无连接**的。

图 7-12 是 TCP 发送报文段的过程的示意图。为了突出示意图的要点，我们只画出了一个方向的数据流。实际上，只要建立了 TCP 连接，就能支持同时双向通信的数据流。

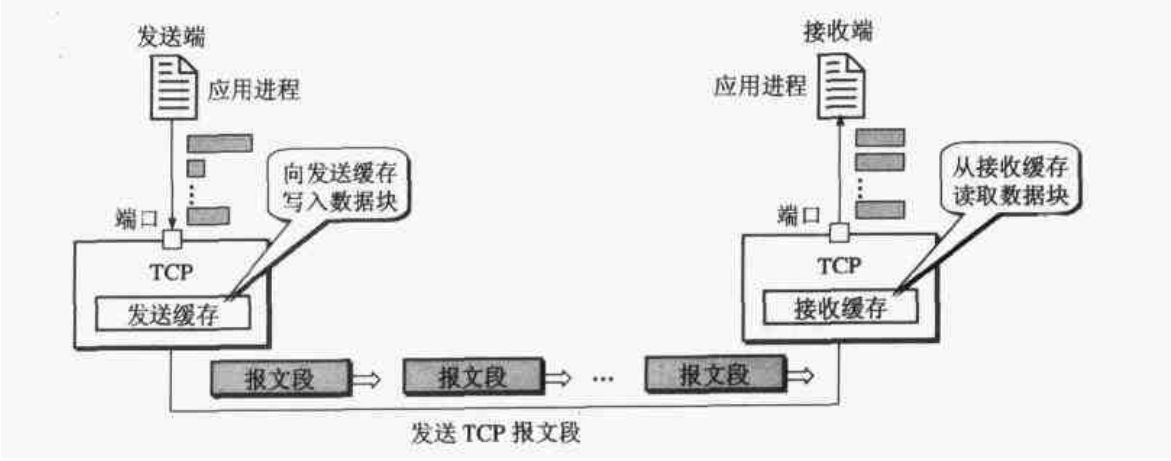


图 7-12　TCP 发送报文段的示意图

在图 7-12 中的每一个端口也是用两个队列（一个入队列和一个出队列）来实现的。这里为简单起见，在表示端口的小方框中，没有画出这两个队列。

TCP 连接的建立和释放过程将在后面的 7.4.7 节讨论。这里只是简单地说明在 TCP 连接建立后数据传输的大致过程。由于通信是全双工方式，因此 TCP 连接的任何一方（不论是客户端还是服务器端）都能够发送和接收数据。因此在图中就只标明数据的发送端和接收端。数据传输的过程和客户服务器方式没有什么关系。

发送端的应用进程按照自己产生数据的规律，不断地将数据块（其长短可能各异）陆续写入到 TCP 的发送缓存中。TCP 再从发送缓存中取出一定数量的数据，将其组成 **TCP 报文段**(segment)逐个传送给 IP 层，然后发送出去。图中表示的是在 TCP 连接上传送一个个 TCP 报文段，而没有画出 IP 层或链路层的动作（这样会太烦琐）。接收端从 IP 层收到 TCP 报文段后，先将其暂存在接收缓存中，然后让接收端的应用进程从接收缓存中将数据块逐个读取。

以上就是 TCP 的数据传输的简单过程。有了这些基本概念后，下面我们就开始介绍 TCP 的报文段首部的格式，然后再讨论保证数据传送可靠、按序、无丢失和不重复的一些机制。

7.4.2　TCP 报文段的首部

一个 TCP 报文段分为首部和数据两部分（图 7-13）。应当指出，TCP 的全部功能都体现在它首部中各字段的作用。因此，只有弄清 TCP 首部各字段的作用才能掌握 TCP 的工作原理。本节先大致介绍各字段的作用，然后在后续的几节结合相关的字段详细讨论 TCP 的主要机制。

TCP 报文段首部的前 20 个字节是固定的，后面有 4N 字节是根据需要而增加的选项（N 必须是整数）。因此 TCP 首部的最小长度是 20 字节。

首部固定部分各字段的意义如下：

（1）**源端口和目的端口**　　各占 2 个字节。前面已讲过，端口是运输层与应用层的服务接口。运输层的复用和分用功能都要通过端口才能实现。

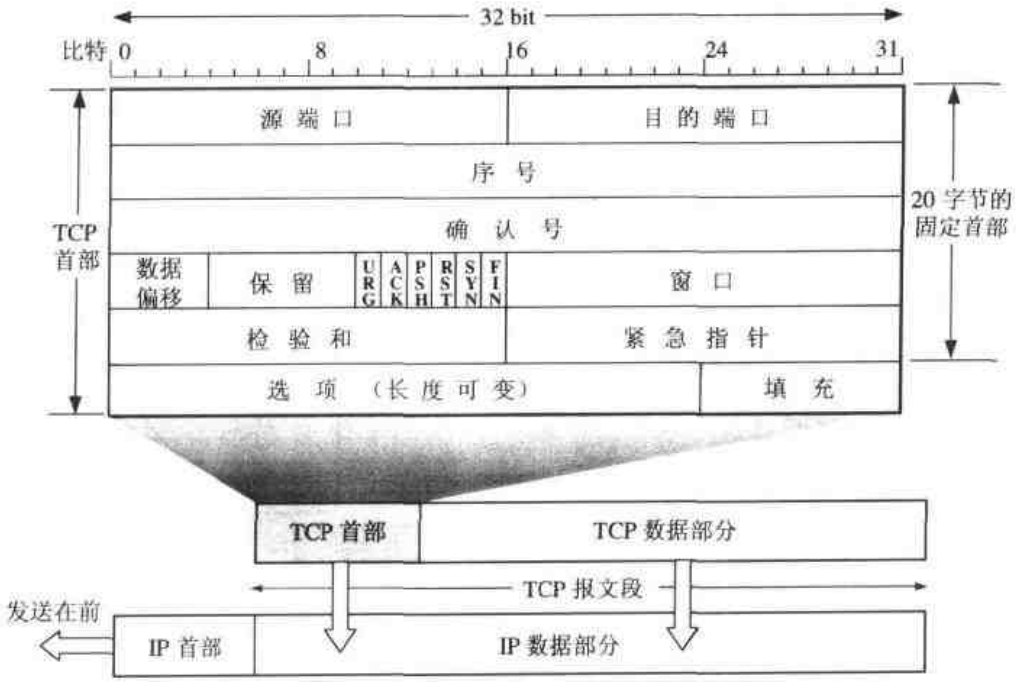


图 7-13 TCP 报文段的首部

（2）**序号** 占 4 字节。TCP 是面向数据流的。TCP 传送的报文可看成为连续的数据流。TCP 把在一个 TCP 连接中传送的数据流中的**每一个字节都编上一个序号**。整个数据的起始序号在连接建立时设置。首部中的序号字段的值则指的是**本报文段**所发送的数据的第一个字节的序号。例如，一报文段的序号字段的值是 301，而携带的数据共有 100 字节。这就表明：本报文段的数据的最后一个字节的序号应当是 400。我们还可看到，下一个报文段的数据序号应当从 401 开始，因而下一个报文段的序号字段值应为 401。

（3）**确认号** 占 4 字节，是期望收到对方的下一个报文段的数据的第一个字节的序号，也就是期望收到的下一个报文段首部的序号字段的值。例如，A 正确收到了 B 发送过来的一个报文段，其序号字段的值是 501，而数据长度是 200 字节，这就表明 A 已正确收到了 B 发送的序号在 501 至 700 之间的数据。因此，A 期望收到 B 的下一个报文段的首部中的序号字段应为 701，于是 A 在发送给 B 的响应报文段中将首部中的确认号置为 701。

由于序号字段有 32 bit 长，可对 4 GB（即 4 千兆字节）的数据进行编号。这样就可保证当序号重复使用时，旧序号的数据早已在网络中消失了。

（5）**数据偏移** 占 4 bit，它指出 TCP 报文段的数据起始处距离 TCP 报文段的起始处有多远。这实际上就是 TCP 报文段首部的长度。由于首部长度不固定（因首部中还有长度不确定的选项字段），因此数据偏移字段是必要的。但应注意，“数据偏移”的单位不是字节而是 32 bit 字（即以 4 字节长的字为计算单位）。由于 4 bit 能够表示的最大十进制数字是 15，因此数据偏移的最大值是 60 字节，这也是 TCP 首部的最大长度。

（6）**保留** 占 6 bit，保留为今后使用，但目前应置为 0。

下面有 6 个比特是说明本报文段性质的**控制**比特，它们的意义见(7)~(12)。

（7）**紧急**比特 URG (URGent) 当 URG = 1 时，表明紧急指针字段有效。它告诉系统此报文段中有紧急数据，应尽快传送（相当于高优先级的数据），而不要按原来的排队顺序来传送。例如，已经发送了很长的一个程序要在远地的主机上运行。但后来发现了一些问题，需要取消该程序的运行。因此用户从键盘发出中断命令(Control + C)。如果不使用紧急数据，那

么这两个字符将存储在接收 TCP 缓存的末尾。只有在所有的数据被处理完毕后这两个字符才被交付到接收应用进程。这样做就浪费了许多时间。

当使用紧急比特并将 URG 置 1 时，发送应用进程就告诉发送 TCP 这两个字符是紧急数据。于是发送 TCP 就将这两个字符插入到报文段的数据的最前面，其余的数据都是普通数据。这时要与首部中第 5 个 32 bit 字中的一半“**紧急指针**”(Urgent Pointer)字段配合使用。紧急指针指出在本报文段中的紧急数据的最后一个字节的序号。紧急指针使接收方知道紧急数据共有多少个字节。紧急数据到达接收端后，当所有紧急数据都被处理完时，TCP 就告诉应用程序恢复到正常操作。值得注意的是，即使窗口为零时也可发送紧急数据。

（8）**确认比特 ACK**　只有当 ACK = 1 时确认号字段才有效。当 ACK = 0 时，确认号无效。

（9）**推送比特 PSH(PuSH)**　当两个应用进程进行交互式的通信时，有时在一端的应用进程希望在键入一个命令后立即就能够收到对方的响应。在这种情况下，TCP 就可以使用推送(push)操作。这时，发送端 TCP 将推送比特 PSH 置 1，并立即创建一个报文段发送出去。接收 TCP 收到推送比特置 1 的报文段，就尽快地（即“推送”向前）交付给接收应用进程，而不再等到整个缓存都填满了后再向上交付。PSH 比特也可叫做**急迫比特**。

虽然应用程序可以选择推送操作，但推送操作还是往往不被人们使用。TCP 可以选择或不选择这个操作。

（10）**复位比特 RST (ReSeT)**　当 RST = 1 时，表明 TCP 连接中出现严重差错（如由于主机崩溃或其他原因），必须释放连接，然后再重新建立运输连接。复位比特还用来拒绝一个非法的报文段或拒绝打开一个连接。复位比特也可称为重建比特或重置比特。

（11）**同步比特 SYN**　在连接建立时用来同步序号。当 SYN = 1 而 ACK = 0 时，表明这是一个连接请求报文段。对方若同意建立连接，则应在响应的报文段中使 SYN = 1 和 ACK = 1。因此，同步比特 SYN 置为 1，就表示这是一个连接请求或连接接受报文。关于连接的建立和释放，后面还要进行讨论。

（12）**终止比特 FIN(FINal)**　用来释放一个连接。当 FIN = 1 时，表明此报文段的发送端的数据已发送完毕，并要求释放运输连接。

（13）**窗口**　占 2 字节。窗口字段用来控制对方发送的数据量，**单位为字节**。大家知道，计算机网络经常是用接收端的接收能力的大小来控制发送端的数据发送量。TCP 也是这样。TCP 连接的一端根据设置的缓存空间大小确定自己的接收窗口大小，然后通知对方以确定对方的发送窗口的上限。将 TCP 连接的两端分别记为 A 和 B。若 A 确定自己的接收窗口为 WIN，则 A 发送给 B 的 TCP 报文段的窗口字段中写入 WIN 的数值。这就是告诉 B 的 TCP，“你(b)在未收到我(a)的确认时所能够发送的数据量的上限就是从本首部中的确认号开始的 WIN 个字节。”所以 A 所设定的 WIN 既是 A 的接收窗口，同时也就是 B 的发送窗口的上限值。例如，A 在发送给 B 的报文段的首部中将窗口字段的值 WIN 置为 500，将确认号置为 201。这就是告诉 B：“你(b)在未收到确认的情况下，最多可向我(a)发送序号从 201 开始到 700 共 500 字节的数据”。B 在收到此报文段后，就用此窗口数值 500 作为 B 的发送窗口的上限值。但应注意，B 向 A 发送的报文段的首部也有一个窗口字段，但这是根据 B 的接收能力来确定 A 的发送窗口上限，一定不要弄混。

（14）**检验和**　占 2 字节。检验和字段检验的范围包括首部和数据这两部分。和 UDP 用户数据报一样，在计算检验和时，要在 TCP 报文段的前面加上 12 字节的伪首部。

伪首部的格式与图 7-10 中 UDP 用户数据报的伪首部一样。但应将伪首部第 4 个字段中的 17 改为 6(TCP 的协议号是 6)，将第 5 字段中的 UDP 长度改为 TCP 长度。接收端收到此报文段后，仍要加上这个伪首部来计算检验和。若使用 IPv6，则相应的伪首部也要改变。

（15）**选项**　长度可变。TCP 只规定了一种选项，即**最大报文段长度 MSS** (Maximum Segment Size)[①]。MSS 告诉对方 TCP："我的缓存所能接收的报文段的**数据字段**的最大长度是 MSS 个字节。"当没有使用选项时，TCP 的首部长度是 20 字节。

MSS 的选择并不太简单。若选择较小的 MSS 长度，网络的利用率就降低。设想在极端的情况下，当 TCP 报文段只含有 1 字节的数据时，在 IP 层传输的数据报的开销至少有 40 字节（包括 TCP 报文段的首部和 IP 数据报的首部）。这样，对网络的利用率就不会超过 1/41。到了数据链路层还要加上一些开销。但反过来，若 TCP 报文段非常长，那么在 IP 层传输时就有可能要分解成多个短数据报片。在目的站要将收到的各个短数据报片装配成原来的 TCP 报文段。当传输出错时还要进行重传。这些也都会使开销增大。一般认为，MSS 应尽可能大些，只要在 IP 层传输时不需要再分片就行。在连接建立的过程中，双方都将自己能够支持的 MSS 写入这一字段。在以后的数据传送阶段，MSS 取双方提出的较小的那个数值。若主机未填写这项，则 MSS 的默认值是 536 字节长。因此，所有在因特网上的主机都应能接受的报文段长度是 536 + 20 = 556 字节。

7.4.3　TCP 的数据编号与确认

TCP 协议是**面向字节的**。TCP 将所要传送的整个报文（这可能包括许多个报文段）看成是**一个个字节组成的数据流**，并**使每一个字节对应于一个序号**。在连接建立时，双方要商定**初始序号**。TCP 每次发送的报文段的首部中的序号字段数值表示该报文段中的数据部分的**第一个字节的序号**。

TCP 的确认是**对接收到的数据的最高序号**（即收到的数据流中的最后一个序号）**表示确认**。但接收端返回的确认号是已收到的数据的最高序号加 1。也就是说，**确认号表示接收端期望下次收到的数据中的第一个数据字节的序号**。

TCP 传输的可靠是由于使用了序号和确认。当 TCP 发送一报文段时，它同时也在自己的重传队列中存放一个副本。若收到确认，则删除此副本。若在计时器时间到之前没有收到确认，则重传此报文段的副本。TCP 的确认并不保证数据已由应用层交付给了端用户，而只是表明在接收端的 TCP 收到了对方所发送的报文段。。

由于 TCP 连接能提供全双工通信，因此通信中的每一方都不必专门发送确认报文段，而可以在传送数据时顺便把确认信息捎带传送。这样做可以提高传输效率。

在发送端，TCP 是怎样决定发送一个报文段的时机呢？

TCP 有三种基本机制来控制报文段的发送。第一种机制是 TCP 维持一个变量，它等于**最大报文段长度** MSS。只要发送缓存从发送进程得到的数据达到 MSS 字节时，就组装成一个

---

① 注：最大报文段长度 MSS 这个名词**很容易引起误解**。MSS 是 TCP 报文段中的**数据字段的最大长度**。数据字段加上 TCP 首部才等于整个的 TCP 报文段。所以 MSS 并不是 TCP 报文段的最大长度，而是：

MSS = TCP 报文段长度 − TCP 首部长度

TCP 报文段，然后发送出去。第二种机制是发送端的应用进程指明要求发送报文段，即 TCP 支持的**推送**(push)操作。第三种机制是发送端的一个计时器时间到了，这时就把当前已有的缓存数据装入报文段发送出去。

但是，**如何控制 TCP 发送报文段的时机**仍然是一个较为复杂的问题。

例如，一个交互式用户使用一条 TELNET 连接（运输层为 TCP 协议）。设用户只发一个字符。加上 20 字节的首部后，得到 21 字节长的 TCP 报文段。再加上 20 字节的 IP 首部，形成 41 字节长的 IP 数据报。在接收端 TCP 立即发出确认，构成的数据报是 40 字节长（假定没有数据发送）。若用户要求远地主机回送这一字符，则又要发回 41 字节长的 IP 数据报和 40 字节长的确认 IP 数据报。这样，用户仅发一个字符时线路上就需传送总长度为 162 字节共 4 个报文段。当线路带宽并不富裕时，这种传送方法的效率的确不高。因此应适当推迟发回确认报文，并尽量使用捎带确认的方法。

在 TCP 的实现中广泛使用 Nagle 算法[NAGL84]。算法如下：若发送端应用进程将欲发送的数据逐个字节地送到发送端的 TCP 缓存，则发送端就将第一个字符（一个字符的长度是一个字节）先发送出去，将后面到达的字符都缓存起来。当接收端收到对第一个字符的确认后，再将缓存中的所有字符装成一个报文段发送出去，同时继续对随后到达的字符进行缓存。只有在收到对前一个报文段的确认后才继续发送下一个报文段。当字符到达较快而网络速率较慢时，用这样的方法可明显地减少所用的网络带宽。算法还规定，当到达的字符已达到窗口大小的一半或已达到报文段的最大长度时，就立即发送一个报文段。

但有时不宜采用 Nagle 算法。例如在因特网上使用 X-Windows，并将鼠标移动的信息传到远地主机。若采用 Nagle 算法会使用户感到无法忍受。这时最好关闭这个算法。

另一个问题叫做**糊涂窗口综合症**(silly window syndrome) [RFC 813]，有时也会使 TCP 的性能变坏。设想一种情况：接收端的缓存已满，而交互式的应用进程一次只从接收缓存中读取一个字符（这样就在缓存产生仅 1 个字节的空位子），然后接收端向发送端发送确认，并将窗口设置为 1 个字节（但发送的数据报是 40 字节长）。接着，发送端又发来 1 个字符（但发来的 IP 数据报是 41 字节长）。接收端发回确认，仍然将窗口设置为 1 个字节。这样进行下去，使网络的效率很低。

要解决这个问题，可**让接收端等待一段时间**，使得或者缓存已能有足够的空间容纳一个最长的报文段，或者**缓存已有一半的空间处于空的状态**。只要出现这两种情况之一，接收端就发出确认报文，并向发送端通知当前的窗口大小。此外，发送端也不要发送太小的报文段，而是将数据积累成足够大的报文段，或达到接收端缓存的空间的一半大小。

上述两种方法可配合使用。使得在发送端不发送很小的报文段的同时，接收端也不要在缓存刚刚有了一点小的空位置就急忙将一个很小的窗口大小通知给发送端。

若发送方在规定的设置时间内没有收到确认，就要将未被确认的报文段重新发送。接收方若收到有差错的报文段，则丢弃此报文段（不发送否认信息）。若收到重复的报文段，也要将其丢弃，但要发回（或捎带发回）确认信息。这与数据链路层的情况相似。

若收到的报文段无差错，只是未按序号，那么应如何处理？TCP 对此未作明确规定，而是让 TCP 的实现者自行确定。或者将不按序的报文段丢弃，或者先将其暂存于接收缓存内，待所缺序号的报文段收齐后再一起上交应用层。如有可能，采用后一种策略对网络的性能会更好些。例如，发送端每个报文中含有 100 字节的数据，且一连发送了 8 个报文段，其序号

分别为 1, 101, 201,...,701。设接收端正确收到了其中的 7 个，而未收到序号为 201 的报文段。接收端可以将序号为 301 到 701 的 5 个报文段先进行暂存，而发回确认号为 201 的报文段(即序号为 200 及这以前的都已正确收到了)。当发送端重传的序号为 201 的报文段正确到达接收端后，接收端就发回确认号为 801 的确认，从而提高了传输效率。

7.4.4 TCP 的流量控制与拥塞控制

1．滑动窗口的概念

为了提高报文段的传输效率，TCP 采用**大小可变的滑动窗口**进行流量控制。窗口大小的单位是**字节**。在 TCP 报文段首部的窗口字段写入的数值就是当前给对方设置的发送窗口数值的上限。

发送窗口在连接建立时由双方商定。但在通信的过程中，接收端可根据自己的资源情况，随时动态地调整对方的发送窗口上限值（可增大或减小）。这种**由接收端控制发送端**的做法，在计算机网络中经常使用。图 7-14 表示的是在 TCP 中使用的窗口概念。

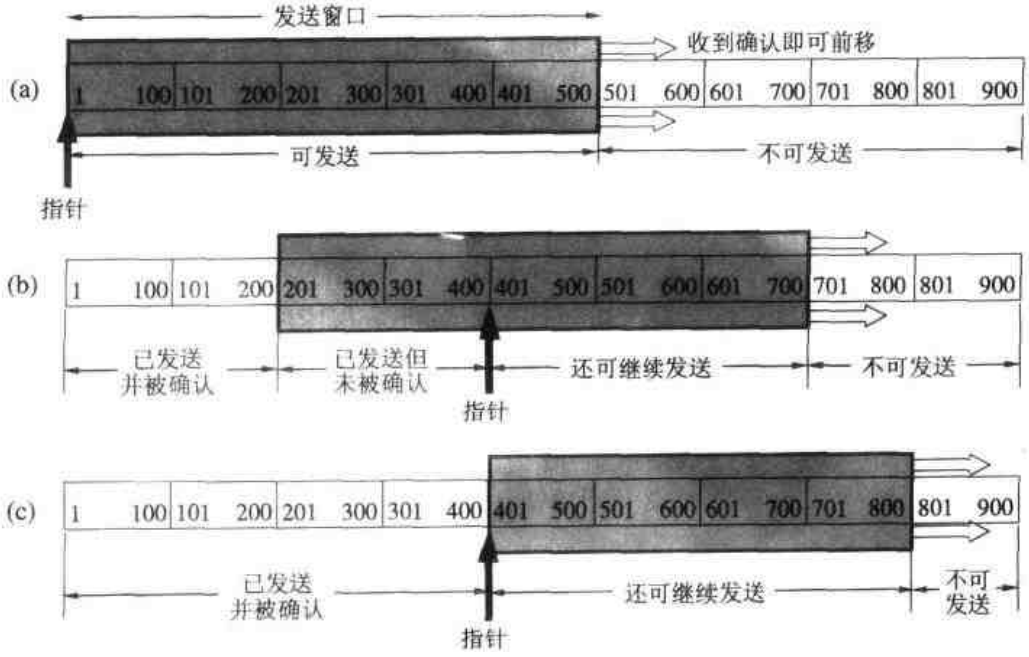


图 7-14　TCP 中的窗口概念

(a)发送窗口大小为 500 字节；(b)发送了 400 字节，收到的确认号为 201；(c)收到的确认号为 401

图 7-14(a)表示发送端要发送 900 字节长的数据，划分为 9 个 100 字节长的报文段，而发送窗口确定为 500 字节。发送端只要收到了对方的确认，发送窗口就可前移。发送 TCP 要维护一个指针。每发送一个报文段，指针就向前移动一个报文段的距离。当指针移动到发送窗口的最右端（即窗口前沿）时就不能再发送报文段了。

图 7-14(b)表示发送端已发送了 400 字节的数据，但只收到对前 200 字节数据的确认，同时窗口大小不变。我们注意到，现在发送端还可发送 300 字节。

图 7-14(c)表示发送端收到了对方对前 400 字节数据的确认，但对方将窗口减小到 400 字节。于是，发送端最多还可发送 400 字节的数据。

下面通过图 7-15 的例子说明利用可变窗口大小进行流量控制。

设主机 A 向主机 B 发送数据。双方确定的窗口值是 400。再设每一个报文段为 100 字节长，序号的初始值为 1（见图中第一个箭头上的 SEQ = 1。图中右边的注释可帮助理解整个的过程）。我们应注意到，主机 B 进行了三次流量控制。第一次将窗口减小为 300 字节，第二次又减为 200 字节，最后一次减至零，即不允许对方再发送数据了。这种暂停状态将持续到主机 B 重新发出一个新的窗口值为止。



图 7-15　利用可变窗口进行流量控制举例

发送端利用发送窗口（这个窗口取决于对方的接收窗口）调节向网络注入分组的速率不仅仅是为了使接收端来得及接收，而且还是为了对网络进行拥塞控制。我们知道，拥塞发生在通过网络传输的分组数量开始接近网络对分组的处理能力时。从这个角度看，拥塞控制的目标就是将网络中的分组数量维持在一定的水平之下。若网络中的分组数量超过这个水平，网络的性能就会急剧恶化。拥塞控制也是运输层必须解决的一个非常复杂的问题。

在讨论拥塞控制时需要特别注意以下两点。

第一，我们在网络层观察问题时，经常讲到主机发送分组（或 IP 数据报）和路由器转发分组（或 IP 数据报）。但在运输层讨论问题时，我们就说“主机发送报文段”。这两种说法并没有多大实质性差别，因为运输层的报文段就是分组（或 IP 数据报）的数据部分。在讨论网络的各种问题时也经常同时使用“报文段”和“分组”这两个名词，但应注意它们是不同层次的数据传送单位。

第二，在每一个运输连接上报文段是断续发送的（因为要受发送窗口的制约，有时要停止发送而等待对方的确认）。这样就有了两种速率。一种是链路层的数据率，另一种是从运输层看到的数据注入速率。例如，主机连接到网络的链路速率是 2 Mb/s。主机只要发送报文段，在链路上的数据传送速率就是 2 Mb/s。但当主机暂停发送报文段时，则链路处于空闲状态。现在假定主机在 1 秒钟内发送 100 个报文段，而每个报文段含有 1000 字节的数据。从运输层看，这相当于主机在这 1 秒钟向网络注入了 100000 字节的数据，即 0.8 Mb/s。实际上，一个报文段除了数据部分还有 20 字节长的首部。但通常在运输层观察向网络注入数据时往往不考虑首部的 20 字节，这是因为运输层只对数据部分的每个字节进行编号，而运输层的窗口大小

都是对报文段中的数据部分而言的。实际上从链路层注入到网络的数据量，还必须把运输层和网络层的首部以及链路层的首部和尾部都加上去。

为了更好地在运输层进行拥塞控制，1999 年公布的因特网建议标准[RFC 2581]定义了以下四种算法，即**慢开始**(slow-start)、**拥塞避免**(congestion avoidance)、**快重传**(fast retransmit)和**快恢复**(fast recovery)。下面就介绍这些算法的要点[PETE00]。

2．慢开始和拥塞避免

根据以上所述，发送端的主机在确定发送报文段的速率时，既要根据**接收端的接收能力**，又要从全局考虑**不要使网络发生拥塞**。因此，对于每一个 TCP 连接，需要有以下两个状态变量：

（1）**接收端窗口** rwnd (receiver window)　这是接收端根据其目前的接收缓存大小所许诺的最新的窗口值，**是来自接收端的流量控制**。接收端将此窗口值放在 TCP 报文的首部中的窗口字段，传送给发送端。**接收端窗口**又称为**通知窗口**(advertised window)。

（2）**拥塞窗口** cwnd (congestion window)　这是发送端根据自己估计的网络拥塞程度而设置的窗口值，**是来自发送端的流量控制**。

没有人会告诉发送端："你的拥塞窗口应设置为多大。"发送端确定拥塞窗口的原则是这样的：只要网络没有出现拥塞，发送端就使拥塞窗口再增大一些，以便将更多的分组发送出去。但只要网络出现拥塞，发送端就使拥塞窗口减小一些，以减少注入到网络中的分组数。发送端又是如何知道网络发生了拥塞呢？我们知道，当网络发生拥塞时，路由器就要丢弃分组。因此只要发送端没有按时收到应当到达的确认报文 ACK，就可以认为网络出现了拥塞。现在通信线路的传输质量一般都很好，因传输出差错而丢弃分组的概率是很小的（远小于 1 %）。下面继续讨论发送端如何具体控制拥塞窗口 cwnd 的大小。

如上所述，发送端的发送窗口的上限值应当取为接收端窗口 rwnd 和拥塞窗口 cwnd 这两个变量中较小的一个，即应按以下公式确定：

$$\text{发送窗口的上限值} = \text{Min}\,[\text{rwnd}, \text{cwnd}] \qquad (7\text{-}1)$$

(7-1)式告诉我们：当 rwnd < cwnd 时，是接收端的接收能力限制发送窗口的最大值。但当 cwnd < rwnd 时，则是网络的拥塞限制发送窗口的最大值。也就是说，TCP 发送端的发送速率是受目的主机或网络中较慢的一个的制约。也就是说，rwnd 和 cwnd 中较小的一个控制数据的传输。

慢开始算法的原理是这样的。当主机开始发送数据时，如果立即将较大的发送窗口中的全部数据字节都注入到网络，那么由于这时还不清楚网络的状况，因而就有可能引起网络拥塞。经验证明，较好的方法是试探一下，即**由小到大逐渐增大发送端的拥塞窗口数值**。通常在刚刚开始发送报文段时可先将拥塞窗口 cwnd 设置为一个最大报文段 MSS 的数值[①]。而在每收到一个**对新的报文段的确认**后，将拥塞窗口增加至多一个 MSS 的数值。用这样的方法逐步增大发送端的拥塞窗口 cwnd，可以使分组注入到网络的速率更加合理。

① 注：[RFC 2581]规定在一开始 cwnd 应设置为不超过 2 × MSS 个字节，并且在一开始也不能超过两个报文段。但通常就将 cwnd 设置为一个 MSS。

下面用例子说明慢开始算法的原理。为说明原理的方便起见，我们用报文段的个数作为窗口大小的单位。此外，还假定接收端窗口 rwnd 足够大，因此发送窗口只受发送端的拥塞窗口的制约。

在一开始，发送端先设置 cwnd = 1，发送第一个报文段 M_0，接收端收到后发回 ACK_1（表示期望收到下一个报文段 M_1）。发送端收到 ACK_1 后，将 cwnd 从 1 增大到 2，于是发送端可以接着发送 M_1 和 M_2 两个报文段。接收端收到后发回 ACK_2 和 ACK_3。发送端每收到一个对新报文段的确认 ACK，就使发送端的拥塞窗口加 1，因此现在发送端的 cwnd 又从 2 增大到 4，并可发送 M_3～M_6 共 4 个报文段。可见慢开始的“慢”并不是指 cwnd 的增长速率慢。但即使 cwnd 增长得很快，和一开始就将 cwnd 设置为较大的数值相比，使用慢开始算法可以使发送端在开始发送时向网络注入的分组数大大减少。这对防止网络出现拥塞是非个常有力的措施。

为了防止拥塞窗口 cwnd 的增长引起网络拥塞，还需要另一个状态变量，即**慢开始门限** ssthresh（如何设置 ssthresh，后面还要讲）。慢开始门限 ssthresh 的用法如下：

当 cwnd < ssthresh 时，使用上述的慢开始算法。

当 cwnd > ssthresh 时，停止使用慢开始算法而改用拥塞避免算法。

当 cwnd = ssthresh 时，既可使用慢开始算法，也可使用拥塞避免算法。

具体的做法是：

拥塞避免算法使发送端的拥塞窗口 cwnd 每经过一个往返时延 RTT 就增加一个 MSS 的大小（而不管在时间 RTT 内收到了几个 ACK）。这样，拥塞窗口 cwnd 按线性规律缓慢增长，比慢开始算法的拥塞窗口增长速率缓慢得多。

无论在慢开始阶段还是在拥塞避免阶段，只要发送端发现网络出现拥塞（其根据就是没有按时收到 ACK 或收到了重复的 ACK），就要将慢开始门限 ssthresh 设置为出现拥塞时的发送窗口值（即接收端窗口和拥塞窗口中数值较小的一个）的一半（但不能小于 2）[①]。这样设置的考虑就是：既然出现了网络拥塞，那就要减少向网络注入的分组数。然后将拥塞窗口 cwnd 重新设置为 1，并执行慢开始算法。这样做的目的就是要迅速减少主机发送到网络中的分组数，使得发生拥塞的路由器有足够时间把队列中积压的分组处理完毕。

图 7-16 说明了上述拥塞控制的具体过程。

（1）当 TCP 连接进行初始化时，将拥塞窗口置为 1。前面已说过，**为了便于理解**，图 7-16 中的窗口单位不使用字节而使用**报文段**。慢开始门限的初始值设置为 16 个报文段，即 ssthresh = 16。发送端的发送窗口不能超过拥塞窗口 cwnd 和接收端窗口 rwnd 中的最小值。我们假定接收端窗口足够大，因此现在发送窗口的数值等于拥塞窗口的数值。

（2）在执行慢开始算法时，拥塞窗口 cwnd 的初始值为 1。以后发送端每收到一个对新报文段的确认 ACK，就将发送端的拥塞窗口加 1，然后开始下一次的传输（图 7-16 的横坐标是传输次数）。因此拥塞窗口 cwnd 随着传输次数按指数规律增长。当拥塞窗口 cwnd 增长到慢开始门限值 ssthresh 时（即当 cwnd = 16 时），就改为执行拥塞避免算法，拥塞窗口按线性规律增长。

---

① 注：[RFC 2581]给出了根据已发送出但还未被确认的数据字节数来设置 ssthresh 的新的计算公式。但许多教科书在讨论拥塞控制原理时仍使用原来的“将 ssthresh 设置为出现拥塞时的发送窗口值的一半”。

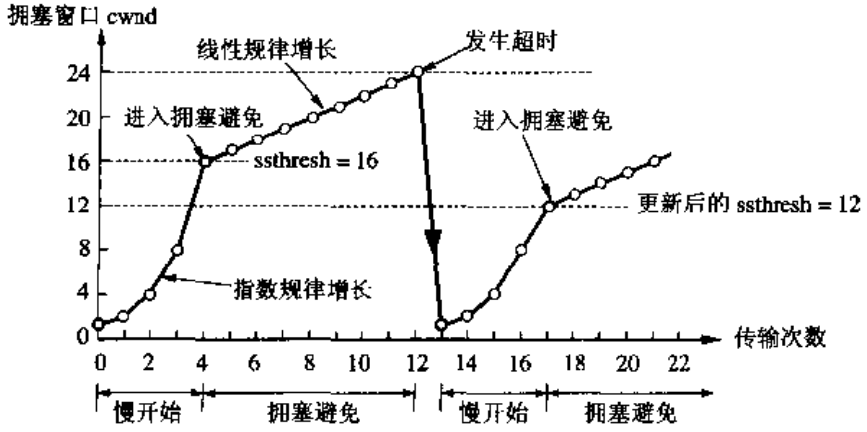


图 7-16 慢开始和拥塞避免算法的实现举例

（3）假定拥塞窗口的数值增长到 24 时，网络出现超时（表明网络拥塞了）。更新后的 ssthresh 值变为 12（即发送窗口数值 24 的一半），拥塞窗口再重新设置为 1，并执行慢开始算法。当 cwnd = 12 时改为执行拥塞避免算法，拥塞窗口按按线性规律增长，每经过一个往返时延就增加一个 MSS 的大小。

在 TCP 拥塞控制的文献中经常可看见“**乘法减小**”(multiplicative decrease)和“**加法增大**”(additive increase)这样的提法。“乘法减小“是指不论在慢开始阶段还是拥塞避免阶段，只要出现一次超时（即出现一次网络拥塞），就将慢开始门限值 ssthresh 设置为当前的拥塞窗口值**乘以** 0.5。当网络频繁出现拥塞时，ssthresh 值就下降得很快，以大大减少注入到网络中的分组数。而“加法增大”是指执行**拥塞避免算法**后，当收到对所有发送出的报文段的确认就将拥塞窗口 cwnd **增加**一个 MSS 大小，使拥塞窗口缓慢增大，以防止网络过早出现拥塞。

这里要再强调一下，“**拥塞避免**”并非指完全能够避免了拥塞。利用以上的措施要完全避免网络拥塞还是不可能的。“拥塞避免”是说在拥塞避免阶段将拥塞窗口控制为按线性规律增长，**使网络比较不容易出现拥塞**。

3．快重传和快恢复

上面讲的慢开始和拥塞避免算法是在 TCP 中最早使用的拥塞控制算法。但后来人们发现这种拥塞控制算法还需要改进，因为有时一条 TCP 连接会因等待重传计时器的超时而空闲较长的时间。为此以后又增加了两个新的拥塞控制算法。这就是**快重传**和**快恢复**。

下面结合一个例子来说明快重传的工作原理。

假定发送端发送了报文段 M_1～M_4 共 4 个报文段。接收端每收到一个报文段后都要立即发出确认 ACK 而不要等待自己发送数据时才将 ACK 捎带上。当接收端收到了 M_1 和 M_2 后，就发出确认 ACK_1 和 ACK_2。假定由于网络拥塞使 M_3 丢失了。接收端后来收到下一个 M_4，发现其序号不对，但仍收下放在缓存中，同时发出确认，不过发出的是重复的 ACK_2（不能够发送 ACK_4，因为 ACK_4 表示 M_4 和 M_3 都已经收到了）。这样，发送端知道现在可能是网络出现了拥塞造成分组丢失，但也可能是报文段 M_3 尚滞留在网络中的某处，还要经过较长的时延才能到达接收端。发送端接着发送 M_5 和 M_6。接收端收到了 M_5 和 M_6 后，也还要分别发出重复的 ACK_2。这样，发送端共收到了接收端的四个 ACK_2，其中三个是重复的。快重传算法规定，发送端只要一连收到三个重复的 ACK 即可断定有分组丢失了，就应立即重传丢失的报文段 M_3 而不必继续等待为 M_3 设置的重传计时器的超时。不难看出，快重传并非取消重

传计时器，而是在某些情况下可更早地重传丢失的报文段。

与快重传配合使用的还有**快恢复**算法。当不使用快恢复算法时，发送端若发现网络出现拥塞就将拥塞窗口降低为 1，然后执行慢开始算法。但这样做的缺点是网络不能很快地恢复到正常工作状态。快恢复算法可以较好地解决这一问题，其具体步骤如下：

（1）当发送端收到连续三个重复的 ACK 时，就重新按照前面讲过的“乘法减小”重新设置慢开始门限 ssthresh。这一点和慢开始算法是一样的。

（2）与慢开始不同之处是拥塞窗口 cwnd 不是设置为 1，而是设置为 ssthresh + 3 × MSS。这样做的理由是：发送端收到三个重复的 ACK_2 表明有三个分组已经离开了网络，它们不会再消耗网络的资源。这三个分组是停留在接收端的缓存中（接收端发送出三个重复的 ACK 就证明了这个事实）。可见现在网络中并不是堆积了分组而是减少了三个分组。因此，将拥塞窗口扩大些并不会加剧网络的拥塞。

（3）若收到的重复的 ACK 为 n 个($n > 3$)，则将 cwnd 设置为 ssthresh + n × MSS。

（4）若发送窗口值还容许发送报文段，就按拥塞避免算法继续发送报文段。

（5）若收到了确认新的报文段的 ACK，就将 cwnd 缩小到 ssthresh。

在采用快恢复算法时，慢开始算法只是在 TCP 连接建立时才使用。

采用这样的流量控制方法使得 TCP 的性能有明显的改进[STEV94][RFC 2581]。

7.4.5 TCP 的重传机制

重传机制是 TCP 中最重要和最复杂的问题之一。**TCP 每发送一个报文段，就对这个报文段设置一次计时器**。只要计时器设置的重传时间到但还没有收到确认，就要重传这一报文段。

由于 TCP 的下层是一个互连网环境，发送的报文段可能只经过一个高速率的局域网，但也可能是经过多个低速率的广域网，并且 IP 数据报所选择的路由还可能会发生变化。图 7-17 画出了数据链路层和运输层的往返时延概率分布的对比。往返时延就是从数据发出到收到对方的确认所经历的时间。对于数据链路层，其往返时延的方差很小，因此将超时时间设置为如图中的 T_1 即可。**但对于运输层来说，其往返时延的方差很大**。若将超时时间设置为图中的 T_2，则很多报文段的重传时间是太早了，给网络增加了许多不应有的负荷。但若将超时时间选为图中的 T_3，则显然会使网络的传输效率降低很多。

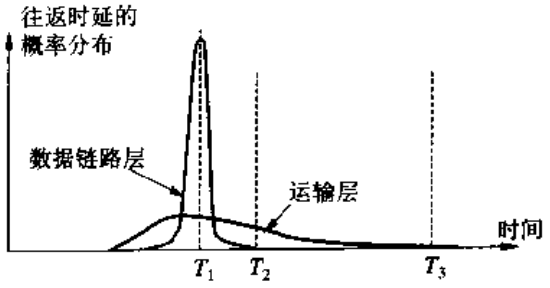


图 7-17　数据链路层和运输层的往返时延的概率分布的差别

那么，运输层的超时计时器的重传时间究竟应设置为多大？

TCP 采用了一种自适应算法。这种算法记录每一个报文段发出的时间，以及收到相应的确认报文段的时间。这两个时间之差就是**报文段的往返时延**。将各个报文段的往返时延样本

加权平均，就得出**报文段的平均往返时延 RTT**。每测量到一个新的往返时延样本，就按下式重新计算一次平均往返时延 RTT：

$$\text{平均往返时延 RTT} = \alpha \times (\text{旧的 RTT}) + (1 - \alpha) \times (\text{新的往返时延样本}) \qquad (7\text{-}2)$$

在上式中，$0 \le \alpha < 1$。若 α 很接近于 1，表示新算出的平均往返时延 RTT 和原来的值相比变化不大，而新的往返时延样本的影响不大（RTT 值更新较慢）。若选择 α 接近于零，则表示加权计算的平均往返时延 RTT 受新的往返时延样本的影响较大（RTT 值更新较快）。典型的 α 值为 7/8。

显然，**计时器设置的超时重传时间 RTO (RetransmissionTime-Out)应略大于上面得出的平均往返时延 RTT**，即：

$$\text{RTO} = \beta \times \text{RTT} \qquad (7\text{-}3)$$

这里 β 是个大于 1 的系数。实际上，系数β是很难确定的。若取β很接近于 1，发送端可以很及时地重传丢失的报文段，因此效率得到提高。但若报文段并未丢失而仅仅是增加了一点时延，那么过早地重传未收到确认的报文段，反而会加重网络的负担。因此 TCP 原先的标准推荐将 β 值取为 2。

上面所说的往返时间的测量，实现起来相当复杂。试看下面的例子。

如图 7-18 所示。发送出一个 TCP 报文段 1。设定的重传时间到了，还没有收到确认。于是重传此报文段，即图中的报文段 2。经过了一段时间后，收到了确认报文段 ACK。现在的问题是：**如何判定此确认报文段是对原来的报文段 1 的确认，还是对重传的报文段 2 的确认？**由于重传的报文段 2 和原来的报文段 1 完全一样，因此源站在收到确认后，就无法做出正确的判断，而正确的判断对确定平均往返时延 RTT 的值关系很大。

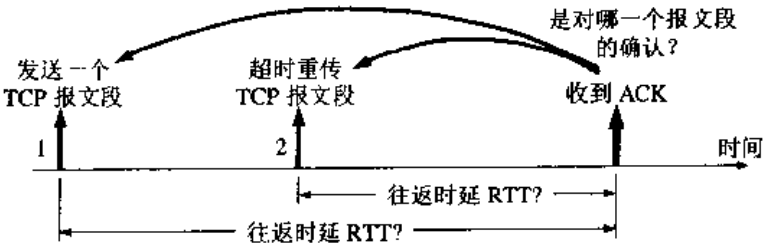


图 7-18　收到的确认报文段 ACK 是对哪一个报文段的确认？

若收到的确认是对重传报文段 2 的确认，但却被源站当成是对原来的报文段 1 的确认，那么这样计算出的往返时延样本和重传时间就会偏大。如果后面再发送的报文段又是经过重传后才收到确认报文段，那么按此方法得出的重传时间就越来越长。

同样，若收到的确认是对原来的报文段 1 的确认，但被当成是对重传报文段 2 的确认，则由此计算出的往返时延样本和重传时间都会偏小。这就必然导致报文段的重传。这样就有可能导致重传时间越来越短。

根据以上所述，Karn 提出了一个算法：**在计算平均往返时延 RTT 时，只要报文段重传了，就不采用其往返时延样本。这样得出的平均往返时延 RTT 和重传时间就较准确。**

但是，这又引起新的问题。设想出现这样的情况：报文段的时延突然增大了很多。因此在原来得出的重传时间内，不会收到确认报文段。于是就重传报文段。但根据 Karn 算法，不考虑重传的报文段的往返时延样本。这样，重传时间就无法更新。

因此要对 Karn 算法进行修正。方法是：报文段每重传一次，就将重传时间增大一些：

新的重传时间 $=\gamma\times$(旧的重传时间)　　(7-4)

系数 γ 的典型值是 2 。当不再发生报文段的重传时，才根据报文段的往返时延更新平均往返时延 RTT 和重传时间的数值。实践证明，这种策略较为合理。

7.4.6 采用随机早期丢弃 RED 进行拥塞控制

上一节讨论的 TCP 拥塞控制并没有和网络层采取的策略联系起来。其实，它们之间是有着密切的关系。

例如，假定一个路由器对某些数据报的处理时间特别长（如在队列中总是将它们排在最后面），那么这就可能使这些数据报中的 TCP 报文段要经过很长的时间才能到达终点，结果引起发送端对这些报文段的重传。而根据我们在前面所讲的，重传就会使 TCP 连接的发送端认为是在网络中发生了拥塞。于是在 TCP 的发送端就采取了拥塞控制措施，但实际上网络并没有发生拥塞。

网络层的策略对 TCP 拥塞控制影响最大的就是路由器的数据报丢弃策略。在最简单的情况下，路由器的队列都是按照“先进先出”的规则 FIFO (First In First Out)处理到来的数据报。由于队列长度总是有限的，因此当队列已满时，以后再到达的所有数据报（如果能够继续排队，这些数据报都将排在队列的尾部）将都被丢弃。这就叫做**尾部丢弃策略** (tail-drop polocy)。

路由器的尾部丢弃会导致上层的 TCP 进入拥塞控制的慢开始状态，使 TCP 连接的发送端突然将数据的发送速率降低到很小的数值。更为严重的是，在网络中通常有很多的 TCP 连接（它们有不同的源点和终点），这些连接中的报文段通常是复用在网络层的 IP 数据报中传送。在这种情况下，若发生了路由器中的尾部丢弃，就可能会同时影响到很多条 TCP 连接，结果使这许多 TCP 连接在**同一时间**突然都进入到慢开始状态。这在 TCP 的术语中称为**全局同步**(global syncronization)。全局同步使得全网的通信量突然下降了很多，而在网络恢复正常后，其通信量又突然增大很多。

为了避免发生网络中的全局同步现象，可以在路由器采用**随机早期丢弃** RED (Random Early Discard)的措施。RED 还有几个不同的名称，即 Random Early Drop，或 Random Early Detection（随机早期检测）。实现 RED 的要点如下：

使路由器的队列维持两个参数，即队列长度最小门限 TH_{min} 和最大门限 TH_{max}。RED 对每一个到达的数据报都先计算平均队列长度 L_{AV}（后面要讲如何计算）。若平均队列长度小于最小门限 TH_{min}，则将新到达的数据报放入队列进行排队。若平均队列长度超过最大门限 TH_{max}，则将新到达的数据报丢弃。若平均队列长度在最小门限 TH_{min} 和最大门限 TH_{max} 之间，则按照某一概率 p 将新到达的数据报丢弃。

图 7-19 说明了以上参数的意义。实际上，RED 就是将路由器的数据报到达队列划分为三个区域。进行正常排队的区域、以概率 p 丢弃的区域和必须丢弃的区域。

随机早期丢弃 RED 中的“随机”就体现在上述算法中的第三条。也就是说，RED 不是等到已经发生网络拥塞后才将所有在队列尾部的数据报全部丢弃，而是在检测到网络拥塞的早期征兆时（即路由器的平均队列长度超过一定的门限值时），就先以概率 p 丢弃个别的数据报，让拥塞控制只在个别的 TCP 连接上进行，因而避免发生全局性的拥塞控制。

这样，使 RED 正常工作的关键就是要选择好三个参数：最小门限 TH_{min}、最大门限 TH_{max} 和概率 p。

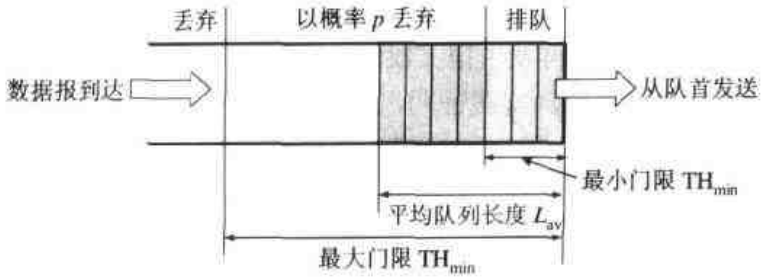


图 7-19　RED 将路由器的到达队列划分成为三个区域

最小门限 TH_{min} 必须足够大，以保证连接路由器的输出链路有较高的利用率。而最大门限 TH_{max} 和最小门限 TH_{min} 之差也应当足够大，使得在一个 TCP 往返时延 RTT 中队列的正常增长仍在最大门限 TH_{max} 之内。经验证明，**使最大门限 TH_{max} 等于最小门限 TH_{min} 值的两倍是合适的**。如果门限值设定得不合适，则 RED 也可能会引起类似于尾部丢弃那样的全局振荡。

在 RED 的操作中最复杂的就是丢弃概率 p 的选择，因为概率 p 不是常数。对每一个到达的数据报，都必须计算丢弃概率 p 的数值。概率 p 的数值取决于当前的平均队列长度 L_{AV} 和所设定的两个门限值 TH_{min} 和 TH_{max}。更具体些就是根据下面三条原则来确定：

（1）当平均队列长度 L_{AV} 小于最小门限 TH_{min} 时，数据报的丢弃概率 $p = 0$。

（2）当平均队列长度 L_{AV} 超过最大门限 TH_{max} 时，丢弃概率 $p = 1$。

（3）当平均队列长度 L_{AV} 在 TH_{min} 和 TH_{max} 之间时，丢弃概率 p 应在 0 到 1 之间，例如，可以按照线性规律变化，从 0 变到 p_{max}。

图 7-20 表示了丢弃概率 p 与两个门限值 TH_{min} 和 TH_{max} 的关系。

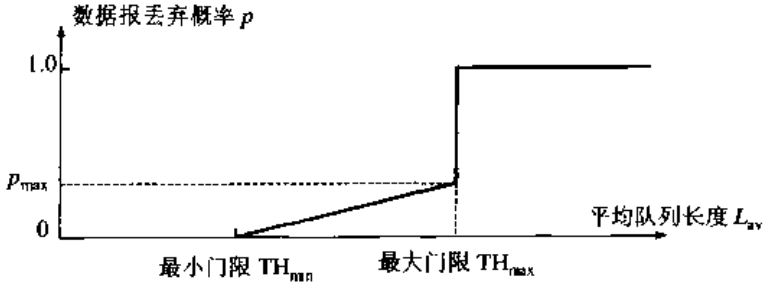


图 7-20　丢弃概率 p 与两个门限值 TH_{min} 和 TH_{max} 的关系

为什么要使用“平均队列长度”呢？我们知道，计算机数据具有突发性的特点，因此路由器中的队列长度经常会出现很快的起伏变化。如果丢弃概率 p 是按照瞬时队列长度来计算，那就可能会出现一些不合理的现象。例如，很短的突发数据不太可能使队列溢出，因此对于这类数据，如果仅因为瞬时队列长度超过了门限值 TH_{min} 就将其丢弃就会产生不必要的拥塞控制。图 7-21 是瞬时队列长度和平均队列长度的区别的示意图。

为此，RED 采用了和计算平均往返时延 RTT 类似的**加权平均**的方法来计算**平均队列长度** L_{AV}，并根据这个平均队列长度 L_{AV} 求出数据报的丢弃概率 p。公式(7-5)给出了平均队列长度 L_{AV} 的计算方法。

$$\text{平均队列长度 } L_{AV} = (1 - \delta)\times(\text{旧的 } L_{AV}) + \delta\times(\text{当前的队列长度样本}) \qquad (7\text{-}5)$$

公式中的 δ 是在 0 到 1 之间的数。若 δ 足够小，则平均队列长度 L_{AV} 取决于队列长度的长期变化趋势，而不受持续时间短的数据突发的影响。

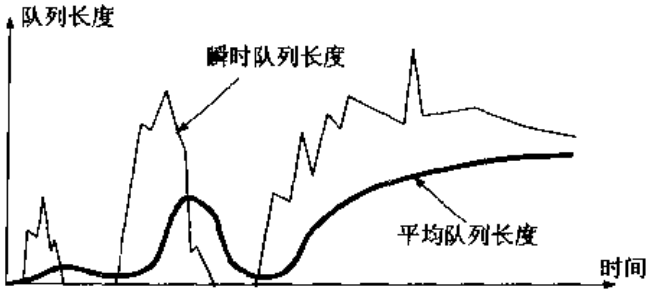


图 7-21　瞬时队列长度和平均队列长度的区别

根据在因特网上的实践经验，丢弃概率 p 的计算方法现在又有了一些改进，其目的是使数据报的丢弃间隔相对地更加均匀些。具体的做法是先按照前面给出的方法算出过渡的丢弃概率 p_{temp}，然后按下面改进的公式计算出丢弃概率：

$$p = p_{temp}/(1 - count \times p_{temp}) \qquad (7\text{-}6)$$

这里 *count* 是一个变量，它代表新到达的数据报有多少个已经进入了队列（没有被丢弃）。显然，过渡的丢弃概率应为：

$$p_{temp} = p_{max} \times (L_{av} - TH_{min})/(TH_{max} - TH_{min}) \qquad (7\text{-}7)$$

下面用个具体的例子来说明这点。设 $p_{max} = 0.02$，而变量 *count* 的初始值为 0。再假定平均队列长度正好在两个门限之间，计算出过渡的丢弃概率 $p_{temp} = 0.01$。由于在开始时变量 $count = 0$，因此算出 $p = p_{temp} = 0.01$。也就是说，现在到达的数据报进入路由器的队列的概率是 0.99。但随着数据报的不断进入队列，变量 *count* 的值不断增大，由(7-6)式算出的丢弃概率也逐渐增大。假定一连有 50 个数据报进入了队列而没有被丢弃，这就使得丢弃概率增大到一倍，即 $p = 0.02$。再假定一连 99 个数据报都没有被丢弃。那么这时由(7-6)式算出丢弃概率 $p = 1$（设平均队列长度一致保持不变），表明下一个数据报肯定要被丢弃。从这里可看出，使丢弃概率 p 不仅与平均队列长度有关，而且还随着队列中不被丢弃的数据报数目的增多而逐渐增大，就可以避免数据报的丢弃过于集中。

总之，随机早期丢弃 RED 好处就是当平均队列长度超过门限值 TH_{min} 时，就会有少量的数据报被丢弃，这就使得有少量的 TCP 连接会减小其窗口值，使得到达路由器的数据报的数量减少。结果，队列平均长度就减小了，从而避免了网络拥塞的发生。应当注意到，网络的吞吐量仍然保持在较高的数值，因此丢弃的数据报的数量是很少的。

我们还应注意到，路由器在某一时刻的瞬时队列长度完全可能远远超过平均队列长度。如果按照式(7-6)和式(7-7)算出的丢弃概率很小，但路由器的队列已经没有空间可接纳新到达的数据报，那么这时 RED 的操作和“尾部丢弃“方式是一样的。RED 只是在可能的条件下尽量使“尾部丢弃”不要发生。

我们还可看出，RED 机制使得路由器可以更好地管理其队列长度。但多长的队列是最佳长度仍然有待于进一步的研究。

RED 工作得很有效，IETF 已经推荐在因特网中的路由器使用 RED 机制[RFC 2309]。

7.4.7　TCP 的运输连接管理

1．运输连接的三个阶段

TCP 是面向连接的协议。运输连接是用来传送 TCP 报文的。TCP 的运输连接的建立和

释放是每一次面向连接的通信中必不可少的过程。因此，运输连接就有三个阶段，即：**连接建立**、**数据传送**和**连接释放**。运输连接的管理就是使运输连接的建立和释放都能正常地进行。

在连接建立过程中要解决以下三个问题：

（1）要使每一方能够确知对方的存在。

（2）要允许双方协商一些参数（如最大报文段长度，最大窗口大小，服务质量等）。

（3）能够对运输实体资源（如缓存大小，连接表中的项目等）进行分配。

TCP 的连接和建立都是采用客户服务器方式。主动发起连接建立的应用进程叫做**客户**(client)，而被动等待连接建立的应用进程叫做**服务器**(server)。

设主机 B 中运行一个服务器进程（图 7-22），它先发出一个**被动打开**(passive open)命令，告诉它的 TCP 要准备接受客户进程的连接请求。然后服务器进程就处于“**听**”(listen)的状态，不断检测是否有客户进程要发起连接请求。如有，即作出响应。

设客户进程运行在主机 A 中。它先向其 TCP 发出**主动打开**(active open)命令，表明要向某个 IP 地址的某个端口建立运输连接。

主机 A 的 TCP 向主机 B 的 TCP 发出连接请求报文段，其首部中的同步比特 SYN 应置为 1，同时选择一个序号 x，表明在后面传送数据时的第一个数据字节的序号是 x。在图 7-22 中，一个从 A 到 B 的箭头上标有“SYN，SEQ = x”就是这个意思。

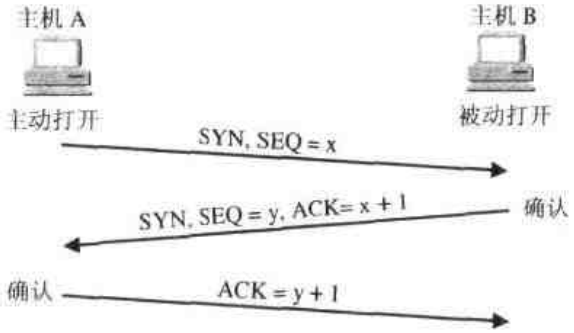


图 7-22　用三次握手建立 TCP 连接

主机 B 的 TCP 收到连接请求报文段后，如同意，则发回确认。在确认报文段中应将 SYN 置为 1，确认号应为 x + 1，同时也为自己选择一个序号 y。

主机 A 的 TCP 收到此报文段后，还要向 B 给出确认，其确认号为 y + 1。

运行客户进程的主机 A 的 TCP 通知上层应用进程，连接已经建立（或打开）。

当运行服务器进程的主机 B 的 TCP 收到主机 A 的确认后，也通知其上层应用进程，连接已经建立。

连接建立采用的这种过程叫做**三次握手**(three-way handshake)，或**三次联络**[①]。

为什么要发送这第三个报文段呢？这主要是为了防止已失效的连接请求报文段突然又传送到了主机 B，因而产生错误。

所谓“已失效的连接请求报文段”是这样产生的。考虑这样一种情况。主机 A 发出连接

① 注：广为流行的译名“三次”(three-way)并不准确。这里的“三次”是指：A 发送一个报文给 B，B 发回确认，然后 A 再加以确认。来回共三次，但这三个报文合起来构成连接建立的“一次握手”。

请求，但因连接请求报文丢失而未收到确认。主机 A 于是再重传一次。后来收到了确认，建立了连接。数据传输完毕后，就释放了连接。主机 A 共发送了两个连接请求报文段，其中的第二个到达了主机 B。

现假定出现另一种情况，即主机 A 发出的第一个连接请求报文段并没有丢失，而是在某些网络结点滞留的时间太长，以致延误到在这次的连接释放以后才传送到主机 B。本来这是一个已经失效的报文段。但主机 B 收到此失效的连接请求报文段后，就误认为是主机 A 又发出一次新的连接请求。于是就向主机 A 发出确认报文段，同意建立连接。

主机 A 由于并没有要求建立连接，因此不会理睬主机 B 的确认，也不会向主机 B 发送数据。但主机 B 却以为运输连接就这样建立了，并一直等待主机 A 发来数据。主机 B 的许多资源就这样白白浪费了。

采用三次握手的办法可以防止上述现象的发生。例如在刚才的情况下，主机 A 不会向主机 B 的确认发出确认。主机 B 收不到确认，连接就建立不起来。

在数据传输结束后，通信的双方都可以发出释放连接的请求。

设图 7-23 中的主机 A 的应用进程先向其 TCP 发出连接释放请求，并且不再发送数据。TCP 通知对方要释放从 A 到 B 这个方向的连接，将发往主机 B 的 TCP 报文段首部的终止比特 FIN 置 1，其序号 x 等于前面已传送过的数据的最后一个字节的序号加 1。

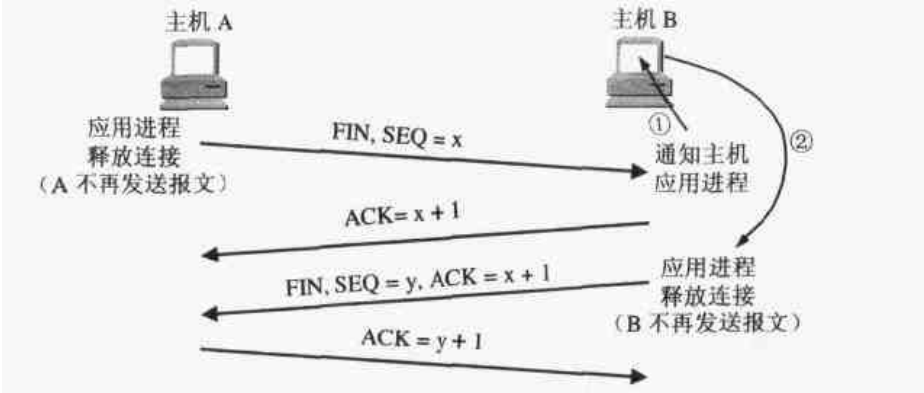


图 7-23　TCP 连接释放的过程

主机 B 的 TCP 收到释放连接通知后即发出确认，其序号为 x + 1，同时通知高层应用进程，见图 7-23 中的箭头①。这样，从 A 到 B 的连接就释放了，连接处于**半关闭(half-close)**状态，相当于主机 A 向主机 B 说："我已经没有数据要发送了。但你如果还发送数据，我仍接收。"

此后，主机 B 不再接收主机 A 发来的数据。但若主机 B 还有一些数据要发往主机 A，则可以继续发送。主机 A 只要正确收到数据，仍应向主机 B 发送确认。

在主机 B 向主机 A 的数据发送结束后，其应用进程就通知 TCP 释放连接，见图 7-23 中的箭头②。主机 B 发出的连接释放报文段必须将终止比特 FIN 置 1，并使其序号 y 等于前面已传送过的数据的最后一个字节的序号加 1，还必须重复上次已发送过的 ACK = x + 1。主机 A 必须对此发出确认，给出 ACK = y + 1。这样才把从 B 到 A 的反方向连接释放掉。主机 A 的 TCP 再向其应用进程报告，整个连接已经全部释放。

读者可发现，上述连接释放过程，和连接建立时的三次握手在本质上是一致的。

7.4.8 TCP 的有限状态机

为了管理因特网，在网络管理中心设有**管理信息库** MIB (Management Information Base)。管理信息库存放着各主机的 TCP **连接表**(Connection Table)，其格式如表 7-2 所示。TCP 连接表对每个连接都登记了其连接信息。除本地和远地的 IP 地址和端口号外，还要记录每一个连接所处的状态。

表 7-2 TCP 连接表

	连接状态	本地 IP 地址	本地端口	远地 IP 地址	远地端口
连接 1					
连接 2					
…					
连接 *n*					

TCP 将连接可能于的状态及各状态可能发生的变迁，画成图 7-24 所示的有限状态机。图中每一个方框即 TCP 可能具有的状态。方框中的字是 TCP 标准使用的状态名。上面所说的 TCP 连接表要记录的“连接状态”就是指每一个连接是处在上述有限状态机中的哪一个状态。状态之间的箭头表示可能发生的状态变迁。箭头旁边的字，表明是什么原因引起这种变迁，

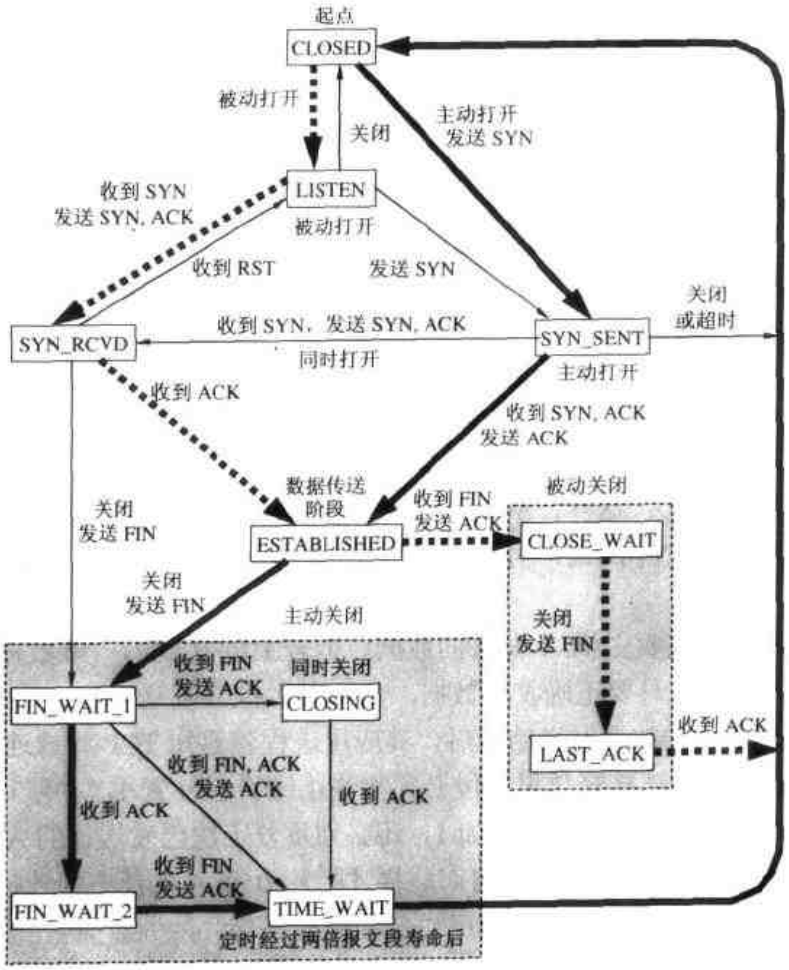


图 7-24 TCP 的有限状态机

或表明发生状态变迁后又出现什么动作。请注意有三种不同的箭头。粗实线箭头表示对客户进程的正常变迁。粗虚线箭头表示对服务器进程的正常变迁。所谓正常变迁即典型变迁。另一种细线箭头表示非典型变迁。下面对此进行简单的解释。

一个 TCP 连接有两个端点。在 TCP 的有限状态机中，有时是指某个端点，而有时则是指另一个端点。对这一点，在分析状态变迁时应特别注意。

我们从连接还未建立时的关闭状态 CLOSED 开始。图 7-24 中的状态较多，最好先顺着粗实线箭头看下去，这是从客户进程开始的变迁过程。设一个主机的客户进程发起连接请求（主动打开），这时本地 TCP 实体就创建**传输控制模块** TCB[①]，发送一个 SYN 置为 1 的报文，因而进入 SYN_SENT 状态。应注意的是，可以有好几个连接代表多个进程同时打开，因此状态是针对每一个连接的。当收到来自进程的 SYN 和 ACK 时，TCP 就发送出三次握手中的最后的一个 ACK，接着就进入连接已经建立的状态 ESTABLISHED。这时就可以发送和接收数据了。

当应用进程结束数据传送时，就要释放已建立的连接。设运行客户进程的主机的本地 TCP 实体发送 FIN 置为 1 的报文，等待着确认 ACK 的到达。这时状态变为 FIN_WAIT_1（见主动关闭的虚线方框中左上角）。当运行客户进程的主机收到确认 ACK 时，则一个方向的连接已经关闭了，状态变为 FIN_WAIT_2。

当运行客户进程的主机收到运行服务器进程的主机发送的 FIN 置为 1 的报文后，应响应确认 ACK。这时另一条连接也关闭了。但是 TCP 还要等待一段时间（此时间取为报文段在网络中的寿命的两倍），TCP 才删除原来建立的连接记录，返回到初始的 CLOSED 状态。这样做是为了保证原来连接上面的所有分组都从网络中消失了。

现在从服务器进程来分析状态图的变迁（看图中粗虚线箭头）。服务器进程发出被动打开，进入听状态 LISTEN。当收到 SYN 置为 1 的连接请求报文后，发送确认 ACK，并且报文中的 SYN 也置为 1，然后进入 SYN_RCVD 状态。在收到三次握手中的最后一个确认 ACK 时，就转为 ESTABLISHED 状态，进入数据传送阶段。

当客户进程的数据已经传送完毕，就发送出 FIN 置为 1 的报文给服务器进程（见标有被动关闭的虚线方框），进入 CLOSE_WAIT 状态。服务器进程发送 FIN 报文段给客户进程，状态变为 LAST_ACK 状态。当收到客户进程的 ACK 时，服务器进程就释放连接，删除连接记录，状态回到原来的 CLOSED 状态。

还有一些状态变迁，例如连接建立过程中的从 LISTEN 到 SYN_SENT 和从 SYN_SENT 到 SYN_RCVD。读者可分析在什么情况下会出现这样的变迁（见习题 7-15）。

为了强调在正常的 TCP 连接建立和关闭的变迁过程，图 7-25 画出了客户进程和服务器进程所经历的状态变化。实际上，它就是将图 7-22 和图 7-23 合并起来，同时将状态的名字注明在图上。图中画在左边的是客户进程进行主动打开和主动关闭，而画在右边的是服务器进程进行被动打开和被动关闭。读者可与图 7-24 进行对照。

---

① 注：TCB 是传输控制程序块(Transmission Control Block)，它存储了每一个连接中的一些重要信息，如：TCP 连接表，到发送和接收缓存的指针，到重传队列的指针，当前的发送和接收序号等等。

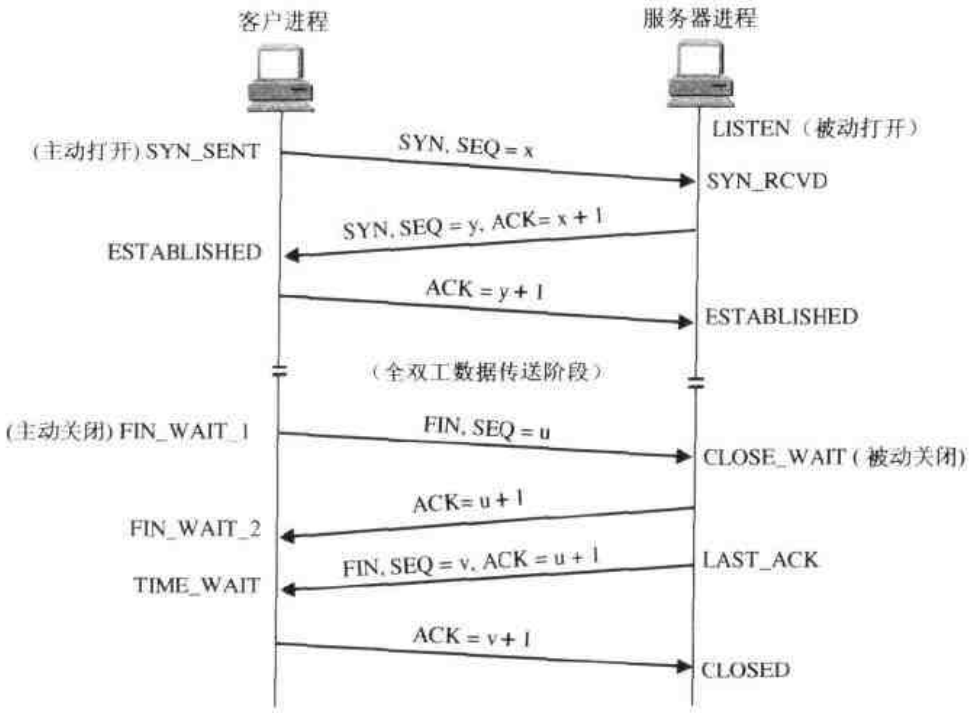


图 7-25　TCP 的正常的连接建立和关闭

习题

7-01　（1）试说明运输层的作用。网络层提供数据报或虚电路服务对上面的运输层有何影响？

（2）当应用程序使用面向连接的 TCP 和无连接的 IP 时，这种传输是面向连接的还是无连接的？

（3）接收端收到有差错的 UDP 用户数据报时应如何处理？

7-02　试用示意图来解释运输层的复用。一个给定的运输连接能否分裂成许多条虚电路？解释之。画图说明许多个运输用户复用到一条运输连接上，而这条运输连接又复用到若干条网络连接（虚电路）上。

7-03　试以具体例子说明为什么一个运输连接可以有多种方式释放。可以设两个互相通信的用户分别连接在网络的两结点上。

7-04　解释为什么突然释放运输连接就可能会丢失用户数据而使用 TCP 的连接释放方法就可保证不丢失数据。

7-05　试用具体例子说明为什么在运输连接建立时要使用三次握手。说明如不这样做可能会出现什么情况。

7-06　一个 TCP 报文段的数据部分最多为多少个字节？为什么？如果用户要传送的数据的字节长度超过 TCP 报文段中的序号字段可能编出的最大序号，问还能否用 TCP 来传送？

7-07　主机 A 和 B 使用 TCP 通信。在 B 发送过的报文段中，有这样连续的两个：ACK = 120 和 ACK = 100。这可能吗（前一个报文段确认的序号还大于后一个的）？试说明理由。

7-08　在使用 TCP 传送数据时，如果有一个确认报文段丢失了，也不一定会引起与该确认报文段对应的数据的重传。试说明理由（可结合上一题讨论）。

7-09　在 7.4.3 节曾讲过，若收到的报文段无差错，只是未按序号，则 TCP 对此未作明确规定，而是让 TCP 的实现者自行确定。试讨论两种可能的方法的优劣：

（1）将不按序的报文段丢弃；

（2）先将不按序的报文段暂存于接收缓存内，待所缺序号的报文段收齐后再一起上交应用层。

7-10 设 TCP 使用的最大窗口为 64 KB，即 64×1024 字节，而传输信道的带宽可认为是不受限制的。若报文段的平均往返时延为 20 ms，问所能得到的最大吞量是多少？

7-11 试计算一个包括 5 段链路的运输连接的单程端到端时延。5 段链路程中有 2 段是卫星链路，有 3 段是广域网链路。每条卫星链路又由上行链路和下行链路两部分组成。可以取这两部分的传播时延之和为 250 ms。每一个广域网的范围为 1 500 km，其传播时延可按 150 000 km/s 来计算。各数据链路速率为 48 kb/s，帧长为 960 bit。

7-12 重复 7-11 题，但假定其中的一个陆地上的广域网的传输时延为 150 ms。

7-13 用 TCP 传送 512 字节的数据。设窗口为 100 字节，而 TCP 报文段每次也是传送 100 字节的数据。再设发送端和接收端的起始序号分别选为 100 和 200，试画出类似于图 7-15 的工作示意图。从连接建立阶段到连接释放都要画上。

7-14 在图 7-23 中所示的连接释放过程中，主机 B 能否先不发送 ACK = x + 1 的确认？（因为后面要发送的连接释放报文段中仍有 ACK = x + 1 这一信息）

7-15 在图 7-24 中，在什么情况下会发生从状态 LISTEN 到状态 SYN_SENT，以及从状态 SYN_SENT 到状态 SYN_RCVD 的变迁？

7-16 什么是 Karn 算法？在 TCP 的重传机制中，若不采用 Karn 算法，而是在收到确认时都认为是对重传报文段的确认，那么由此得出的往返时延样本和重传时间都会偏小。试问：重传时间最后会减小到什么程度？

7-17 若一个应用进程使用运输层的用户数据报 UDP。但继续向下交给 IP 层后，又封装成 IP 数据报。既然都是数据报，是否可以跳过 UDP 而直接交给 IP 层？哪些功能 UDP 提供了但 IP 没有提供？

7-18 使用 TCP 对实时话音数据的传输有没有什么问题？使用 UDP 在传送数据文件时会有什么问题？

7-19 TCP 在进行流量控制时是以分组的丢失作为产生拥塞的标志。有没有不是因拥塞而引起的分组丢失的情况？如有，请举出三种情况。

7-20 一个应用程序用 UDP，到了 IP 层将数据报再划分为 4 个数据报片发送出去。结果前两个数据报片丢失，后两个到达目的站。过了一段时间应用程序重传 UDP，而 IP 层仍然划分为 4 个数据报片来传送。结果这次前两个到达目的站而后两个丢失。试问：在目的站能否将这两次传输的 4 个数据报片组装成为完整的数据报？假定目的站第一次收到的后两个数据报片仍然保存在目的站的缓存中。

7-21 为什么在 TCP 首部中的最开始的 4 个字节是 TCP 的端口号？

7-22 为什么在 TCP 首部中有一个首部长度字段，而 UDP 的首部中就没有这个字段？

7-23 一个 UDP 用户数据报的数据字段为 8192 字节。要使用以太网来传送。试问应当划分为几个数据报片？说明每一个数据报片的数据字段长度和片偏移字段的值。

7-24 在 TCP 的拥塞控制中，什么是慢开始、拥塞避免、快重传和快恢复算法？这里每一种算法各起什么作用？“乘法减小”和“加法增大”各用在什么情况下？

7-25 一打字员平均每分钟打 100 字，而每个字平均有 6 个字符。打字员所用的 PC 机作为客户端，它和另一端的服务器进行通信，在运输层使用 Nagle 算法。试问在以下两种

情况下，从客户端发往服务器的 TCP 报文段序列中的每一个报文段包含多少个字符？

（1）客户和服务器都在同一个局域网上，往返时延 RTT = 20 ms。

（2）客户和服务器通过广域网相连，往返时延 RTT = 100 ms。

7-26 设 TCP 的 ssthresh 的初始值为 8（单位为报文段）。当拥塞窗口上升到 12 时网络发生了超时，TCP 使用慢开始和拥塞避免。试分别求出第 1 次到第 15 次传输的各拥塞窗口大小。

7-27 通信信道带宽为 1 Gb/s，端到端时延为 10 ms。TCP 的发送窗口为 65 535 字节。试问：可能达到的最大吞吐量是多少？信道的利用率是多少？

7-28 网络允许的最大报文段长度为 128 字节，序号用 8 bit 表示，报文段在网络中的寿命为 30 秒。求每一条 TCP 连接所能达到的最高数据率。

7-29 若 TCP 中的序号采用 64 bit 编码，而每一个字节有其自己的序号，试问：在 75 Tb/s 的传输速率下（这是光纤信道理论上可达到的数据率），分组的寿命应为多大才不会使序号发生重复？

7-30 一个 TCP 连接下面使用 256 kb/s 的链路，其端到端时延为 128 ms。经测试，发现吞吐量只有 120 kb/s。试问发送窗口是多少？

7-31 设源站和目的站相距 20 km，而信号在传输媒体中的传播速率为 200 km/ms。若一个分组长度为 1 KB，而其发送时间等于信号的往返传播时延，求数据的发送速率。

7-32 一 UDP 用户数据报的首部的十六进制表示是：06 32 00 45 00 1C E2 17。试求源端口、目的端口、用户数据报的总长度、数据部分长度。这个用户数据报是从客户发送给服务器还是从服务器发送给客户？使用 UDP 的这个服务器程序是什么？

7-33 已知 TCP 的往返时延的当前值是 30 ms。现在收到了三个接连的确认报文段，它们比相应的数据报文段的发送时间分别滞后的时间是：26 ms, 32 ms 和 24 ms。设 $\alpha = 0.9$。试计算新的估计的往返时延值 RTT。

第8章 应 用 层

在前七章我们已经详细地讨论了计算机网络提供通信服务的过程。但是我们还没有讨论这些通信服务是如何提供给应用进程来使用的。本章就是讨论各种应用进程通过什么样的应用层协议来使用网络所提供的这些通信服务。

这里需要再强调一下，每个应用层协议都是为了解决某一类应用问题，而问题的解决又往往是**通过位于不同主机中的多个应用进程之间的通信和协同工作来完成的**。应用层的具体内容就是**规定应用进程在通信时所遵循的协议**。

应用层的许多协议都是基于**客户服务器方式**。这里再明确一下。**客户**(client)和**服务器**(server)都是指通信中所涉及的两个**应用进程**。客户服务器方式所描述的是进程之间服务和被服务的关系。这里最主要的特征就是：**客户是服务请求方，服务器是服务提供方**。

下面开始讨论许多应用协议都要使用的域名系统。在介绍了文件传送协议和远程登录协议后，就讨论用户最常用的因特网电子邮件。然后重点介绍万维网的工作原理及其主要协议。由于万维网的出现使因特网得到了飞速的发展，因此万维网在本章中占有最大的篇幅，也是本章的重点。在本章的最后，介绍有关网络管理方面的问题。对应用层更深入的学习可参阅[COME00][COME97][TANE96][STAL00]及有关标准。

8.1 域名系统 DNS

8.1.1 域名系统概述

许多应用层软件经常直接使用**域名系统** DNS (Domain Name System)，但计算机的用户只是间接而不是直接使用域名系统。

用户与因特网上某个主机通信时，显然不愿意使用很难记忆的长达 32 位二进制主机地址。即使是点分十进制 IP 地址也并不太容易记忆。相反，大家愿意使用某种易于记忆的主机名字。早在 ARPANET 时代，整个网络上只有数百台计算机，那时使用一个叫做 hosts 的文件，列出所有主机名字和相应的 IP 地址。只要用户输入一个主机名字，计算机就能很快地将这个主机名字解析成机器能够识别的二进制 IP 地址。

虽然从理论上讲，可以只使用一个域名服务器，使它装入因特网上所有的主机名，并回答所有对 IP 地址的查询。然而这种做法并不可取。因为随着因特网规模的扩大，这样的域名服务器肯定会因过负荷而无法正常工作，而且一旦域名服务器出现故障，整个因特网就会瘫痪。1983 年因特网开始采用层次结构的命名树作为主机的名字，并使用分布式的**域名系统** DNS [RFC 1034, 1035]。这两个文档早已成为因特网的正式标准。

因特网的域名系统 DNS 被设计成为一个联机分布式数据库系统，并采用客户服务器方式。DNS 使大多数名字都在本地解析，仅少量解析需要在因特网上通信，因此系统效率很高。由于 DNS 是分布式系统，即使单个计算机出了故障，也不会妨碍整个系统的正常运行。

名字到域名的解析是由若干个域名服务器程序完成的。域名服务器程序在专设的结点上

运行，而人们也常把运行该程序的机器称为**域名服务器**。

域名的解析过程如下：当某一个应用进程需要将主机名解析为 IP 地址时，该应用进程就成为域名系统 DNS 的一个客户，并将待解析的域名放在 DNS 请求报文中，以 UDP 数据报方式发给本地域名服务器（使用 UDP 是为了减少开销）。本地的域名服务器在查找域名后，将对应的 IP 地址放在回答报文中返回。应用进程获得目的主机的 IP 地址后即可进行通信。

若本地域名服务器不能回答该请求，则此域名服务器就暂时成为 DNS 中的另一个客户，并向其他域名服务器发出查询请求。这种过程直至找到能够回答该请求的域名服务器为止。上述的这种查找过程，后面还要进一步讨论。

8.1.2 因特网的域名结构

早期的因特网使用了非等级的名字空间，其优点是名字简短。但当因特网上的用户数急剧增加时，用非等级的名字空间来管理一个很大的而且是经常变化的名字集合是非常困难的。因此因特网后来就采用了层次树状结构的命名方法，就像全球邮政系统和电话系统那样。采用这种命名方法，任何一个连接在因特网上的主机或路由器，都有一个**惟一的层次结构的名字**，即**域名**(domain name)。这里，“**域**”(domain)是名字空间中一个可被管理的划分。域还可以继续划分为子域，如二级域、三级域等等。

域名的结构由若干个分量组成，各分量之间用点（请注意，是小数点的点）隔开：

…．三级域名．二级域名．顶级域名

各分量分别代表不同级别的域名。每一级的域名都由英文字母和数字组成（不超过 63 个字符，并且不区分大小写字母），级别最低的域名写在最左边，而级别最高的顶级域名则写在最右边。完整的域名不超过 255 个字符。域名系统既不规定一个域名需要包含多少个下级域名，也不规定每一级的域名代表什么意思。各级域名由其上一级的域名管理机构管理，而最高的顶级域名则由因特网的有关机构管理。用这种方法可使每一个名字都是惟一的，并且也容易设计出一种查找域名的机制。需要注意的是，域名只是个**逻辑概念**，并不代表计算机所在的物理地点。总之，变长的域名和使用有助记忆的字符串，是为了便于使用。而 IP 地址是定长的 32 位二进制数字，是为了便于机器进行处理。这里需要注意，域名中的“点”和点分十进制 IP 地址中的“点”并无一一对应的关系。点分十进制 IP 地址中一定是包含三个“点”，但域名中“点”的数目则不一定正好是三个。

在 1998 年以后，非赢利组织 ICANN 成为因特网的域名管理机构[W-ICANN]。

现在顶级域名 TLD (Top Level Domain)有三大类：

（1）**国家顶级域名** nTDL：采用 ISO 3166 的规定。如：.cn 表示中国，.us 表示美国，.uk 表示英国，等等。国家顶级域名又常记为 ccTLD (cc 表示国家代码 country-code)，现在使用的国家顶级域名约有 200 个左右。

（2）**国际顶级域名** iTDL：采用 .int。国际性的组织可在 .int 下注册。

（3）**通用顶级域名** gTDL：根据[RFC1591]规定，最早的顶级域名共六个，即：

.com 表示公司企业，.net 表示网络服务机构，.org 表示非赢利性组织，.edu 表示教育机构（美国专用），.gov 表示政府部门（美国专用），.mil 表示军事部门（美国专用）。

由于因特网上用户的急剧增加，从 2000 年 11 月起，ICANN 又新增加了七个通用顶级域名[W-NewTLD]，即：

.aero 用于航空运输企业，.biz 用于公司和企业，.coop 用于合作团体，.info 适用

于各种情况，.museum 用于博物馆，.name 用于个人，.pro 用于会计、律师和医师等自由职业者。

在国家顶级域名下注册的二级域名均由该国家自行确定。例如，荷兰就不再设二级域名，其所有机构均注册在顶级域名 .nl 之下。又如顶级域名为 .jp 的日本，将其教育和企业机构的二级域名定为 .ac 和 .co(而不用 .edu 和 .com)。

我国则将二级域名划分为“类别域名”和“行政区域名”两大类。其中“类别域名”6 个，分别为：.ac 表示科研机构；.com 表示工、商、金融等企业；.edu 表示教育机构；.gov 表示政府部门；.net 表示互联网络、接入网络的信息中心(NIC)和运行中心(NOC)；.org 表示各种非盈利性的组织。“行政区域名”34 个，适用于我国的各省、自治区、直辖市。例如：.bj 为北京市；.sh 为上海市；.js 为江苏省；等等。在我国，在二级域名 .edu 下申请注册三级域名则由中国教育和科研计算机网网络中心负责。在二级域名 .edu 之外的其他二级域名下申请注册三级域名的，则应向中国互联网网络信息中心 CNNIC 申请。关于我国的互联网络发展情况以及各种规定，均可在 CNNIC 的网址上找到[W-CNIC]。

图 8-1 是因特网名字空间的结构，它实际上是一个倒过来的树，树根在最上面而没有名字。树根下面一级的结点就是最高一级的顶级域结点。在顶级域结点下面的是二级域结点。最下面的叶结点就是单台计算机。图 8-1 列举了一些域名作为例子。凡是在顶级域名 .com 下注册的单位都获得了一个二级域名。例如图中的例子有：中央电视台，以及 IBM、惠普、摩托罗拉等公司。在顶级域名 .cn（中国）下的二级域名的例子是：4 个行政区域名香港、江苏省、上海市、北京市，以及我国规定的 6 个类别域名。这些二级域名是我国规定的，凡在其中的某一个二级域名下注册的单位就可以获得一个三级域名。图中给出的在 .edu 下面的三级域名有：清华大学、北京大学、复旦大学、上海交通大学、东南大学等。一旦某个单位拥有了一个域名，它就可以自己决定是否要进一步划分其下属的子域，并且不必将这些子域的划分情况报告上级机构。图中画出了在顶级域名 .com 下的中央电视台自己划分的三级域名 mail。在清华大学下的四级域名的例子是：mail，csnet1，ep 等。域名树的树叶就是单台计算机的名字，它不能再继续往下划分子域了。

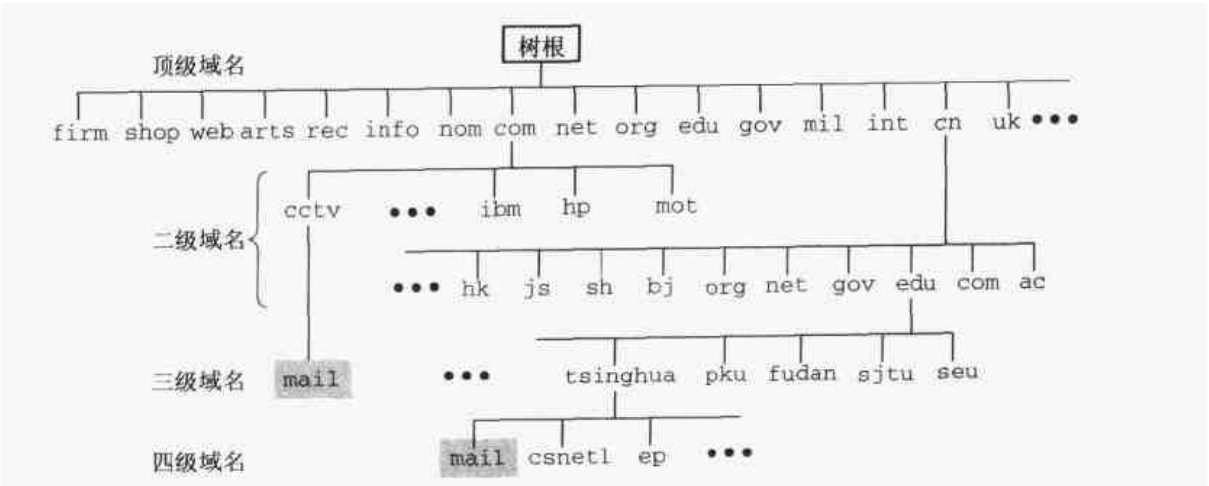


图 8-1 因特网的名字空间

应当注意，虽然中央电视台和清华大学都各有一台计算机取名为 mail，但它们的域名并不一样，因为前者是 mail.cctv.com，而后者是 mail.tsinghua.edu.cn。因此，即

使在世界上还有很多单位的计算机取名为 mail，但是它们在因特网中的域名却都必须是惟一的。

这里还要强调指出，因特网的名字空间是按照机构的组织来划分的，与物理的网络无关，与 IP 地址中的“子网”也没有关系。

8.1.3 用域名服务器进行域名解析

每一个域名服务器不但能够进行一些域名到 IP 地址的**解析**[①]，而且还必须具有连向其他域名服务器的信息。当自己不能进行域名到 IP 地址的转换时，就能够知道到什么地方去找别的域名服务器。

因特网上的域名服务器系统也是按照域名的层次来安排的。每一个域名服务器都只对域名体系中的一部分进行管辖。现在共有以下三种不同类型的域名服务器：

（1）**本地域名服务器**(local name server)：每一个因特网服务提供者 ISP，或一个大学，甚至一个大学里的系，都可以拥有一个**本地域名服务器**，它有时也称为**默认域名服务器**。当一个主机发出 DNS 查询报文时，这个查询报文就首先被送往该主机的本地域名服务器。当 PC 机使用 Windows 9x 时，打开“控制面板”，选择“网络”，再选择“配置”中的 TCP/IP，再选择“属性”，就可看见“DNS 配置”一项。其中的 DNS 服务器就是这里所说的本地域名服务器。本地域名服务器离用户较近，一般不超过几个路由器的距离。当所要查询的主机也属于同一个本地 ISP 时，该本地域名服务器立即就能将所查询的主机名转换为它的 IP 地址，而不需要再去询问其他的域名服务器。

（2）**根域名服务器**(root name server)：目前在因特网上有十几个**根域名服务器**，大部分都在北美。当一个本地域名服务器不能立即回答某个主机的查询时（因为它没有保存被查询主机的信息），该本地域名服务器就以 DNS 客户的身份向某一个根域名服务器查询。若根域名服务器有被查询主机的信息，就发送 DNS 回答报文给本地域名服务器，然后本地域名服务器再回答发起查询的主机。但当根域名服务器没有被查询主机的信息时，它一定知道某个保存有被查询主机名字映射的授权域名服务器的 IP 地址（见下一段）。通常**根域名服务器用来**管辖顶级域（如 .com）。根域名服务器并不直接对顶级域下面所属的所有的域名进行转换，但它一定能够找到下面的所有二级域名的域名服务器。

（3）**授权域名服务器**(authoritative name server)：每一个主机都必须在授权域名服务器处注册登记。通常，一个主机的授权域名服务器就是它的本地 ISP 的一个域名服务器。实际上，为了更加可靠地工作，一个主机最好有至少两个授权域名服务器。许多域名服务器同时充当本地域名服务器和授权域名服务器。授权域名服务器总是能够将其管辖的主机名转换为该主机的 IP 地址。

因特网允许各个单位根据本单位的具体情况将本单位的域名划分为若干个域名服务器**管辖区**(zone)，一般就在各管辖区中设置相应的**授权域名服务器**。例如，图 8-2 中的 abc 公司有下属部门 x 和 y，而部门 x 下面又分三个分部门 u，v 和 w，而 w 下面还有其下属的部门 t。图 8-2 是管辖区的划分举例。可以看出，管辖区是“域”的子集。

图 8-3 表示查询 IP 地址的过程。假定域名为 m.xyz.com 的主机想知道另一个域名为

① 注：在 TCP/IP 的文档中，这种地址转换常称为**地址解析**(address resolution)。解析就是转换的意思。

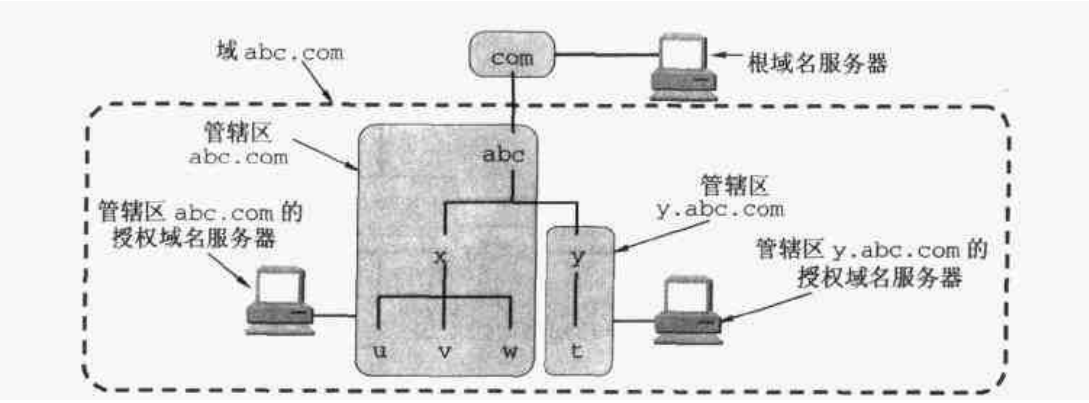


图 8-2　域名服务器管辖区的的划分举例

t.y.abc.com 的主机的 IP 地址。于是向其本地域名服务器 dns.xyz.com 查询。由于查询不到，就向根域名服务器 dns.com 查询。根据被查询的域名中的“abc.com”再向授权域名服务器 dns.abc.com 发送查询报文，最后再向授权域名服务器 dns.y.abc.com 查询。以上的查询过程见图中的①→②→③→④的顺序。得到结果后，按照图中的⑤→⑥→⑦→⑧的顺序将回答报文传送给本地域名服务器 dns.xyz.com。总共要使用 8 个 UDP 报文。这种查询方法叫作**递归查询**。

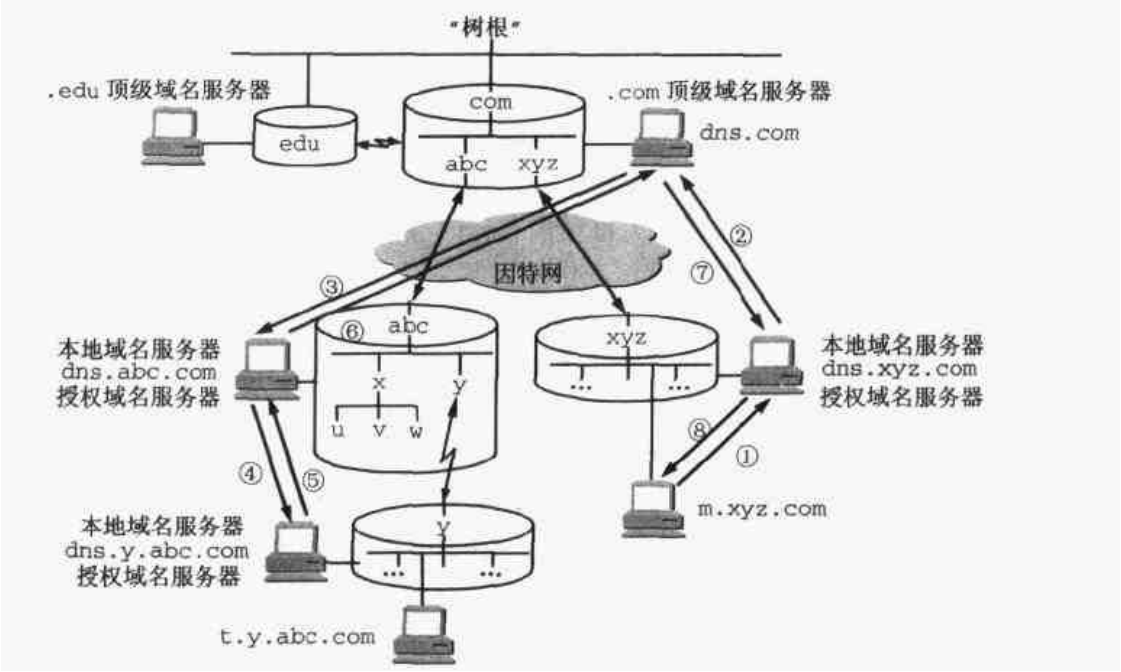


图 8-3　域名转换的递归查询过程举例

为了减轻根域名服务器的负担，根域名服务器在收到图 8-3 中的查询②后，可以直接将下属的授权域名服务器 dns.abc.com 的 IP 地址返回给本地域名服务器 dns.xyz.com，然后让本地域名服务器直接向授权域名服务器 dns.abc.com 进行查询。以后的过程如图 8-4 所示。这就是**递归与迭代相结合**的查询方法。可以看出，对根域名服务器来说，负担是减轻了一半。

以上的例子说明这种查询 IP 地址的方法需要进行改进。第一，大多数的名字转换是本地名字，即与查询的机器在同一个空间划分中。从顶级域名服务器经过几层才能找到所要查询

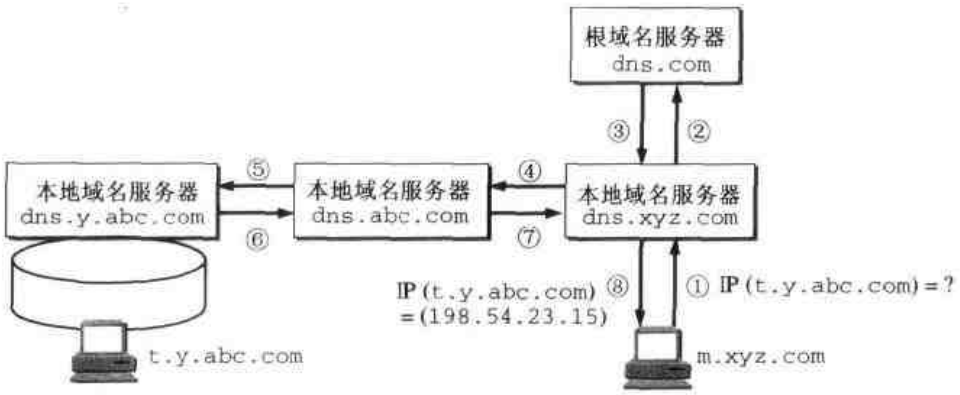


图 8-4　递归与迭代相结合的查询

的 IP 地址，使得查询效率不高。第二，顶级域名服务器的负荷很可能会过载。第三，顶级域名服务器若出现故障，整个名字的转换就无法进行。

使用**名字的高速缓存**可优化查询的开销。每个域名服务器都维护一个高速缓存，存放最近用过的名字以及从何处获得名字映射信息的记录。当客户请求域名服务器转换名字时，服务器首先按标准过程检查它是否被授权管理该名字。若未被授权，则查看自己的高速缓存，检查该名字是否最近被转换过。域名服务器向客户报告缓存中有关名字与地址的**绑定(binding)**信息，并标志为非授权绑定，以及给出获得此绑定的服务器 S 的域名。本地服务器同时也将服务器 S 与 IP 地址的绑定告知客户。因此，客户可很快收到回答，但有可能信息已是过时的了。如果强调高效，客户可选择接受非授权的回答信息并继续进行查询 。如果强调准确性，客户可与授权服务器联系，并检验名字与地址间的绑定是否仍有效。

由于名字到地址的绑定并不经常改变，高速缓存可在域名系统中很好地运作。为保持高速缓存中的内容正确，域名服务器应为每项内容计时并处理超过合理时间的项（例如，每个项目只存放两天）。当域名服务器已从缓存中删去某项后又被请求查询该项信息，就必须重新到授权管理该项的服务器获取绑定信息。当授权服务器回答一个请求时，在响应中都有指明绑定有效存在的时间值。增加此时间值可减少网络开销，而减少此时间值可提高域名转换的准确性。

不但在本地域名服务器中需要高速缓存，在主机中也很需要。许多主机在启动时从本地域名服务器下载名字和地址的全部数据库，维护存放自己最近使用的域名的高速缓存，并且只在从缓存中找不到名字时才使用域名服务器。维护本地域名服务器数据库的主机自然应该定期地检查域名服务器以获取新的映射信息，而且主机必须从缓存中删掉无效的项。由于域名改动并不频繁，大多数网点不需花太多精力就能维护数据库的一致性。

在每个主机中保留一个本地域名服务器数据库的副本，可使本地主机上的域名转换特别快。这也意味着万一本地服务器出故障，本地网点也有一定的保护措施。此外，它减轻了域名服务器的计算负担，使得服务器可为更多机器提供名字解析服务。

8.2　文件传送协议

8.2.1　概述

文件传送协议 FTP (File Transfer Protocol) [RFC 959]是因特网上使用得最广泛的文件传

送协议。FTP 提供交互式的访问，允许客户指明文件的类型与格式（如指明是否使用 ASCII 码），并允许文件具有存取权限（如访问文件的用户必须经过授权，并输入有效的口令）。FTP 屏蔽了各计算机系统的细节，因而适合于在异构网络中任意计算机之间传送文件。文档[RFC 959]很早就成为了因特网的正式标准。

在因特网发展的早期阶段，用 FTP 传送文件约占整个因特网的通信量的三分之一，而由电子邮件和域名系统所产生的通信量还要小于 FTP 所产生的通信量。只是到了 1995 年，WWW 的通信量才首次超过了 FTP。

在下面 8.2.2 和 8.2.3 节分别介绍基于 TCP 的 FTP 和基于 UDP 的 TFTP，它们都是文件共享协议中的一大类，即**复制整个文件**，其特点是：若要存取一个文件，就必须先获得一个本地的文件副本。如果要修改文件，只能对文件的副本进行修改，然后再将修改后的文件副本传回到原结点。

文件共享协议中的另一大类是**联机访问**(on-line access)。联机访问意味着允许多个程序同时对一个文件进行存取。和数据库系统不同之处是用户不需要调用一个特殊的客户进程，而是由操作系统提供对远地共享文件进行访问的服务，就如同对本地文件的访问一样。这就使用户可以用远地文件作为输入和输出来运行任何应用程序，而操作系统中的文件系统则提供对共享文件的**透明存取**。透明存取的优点是：将原来用于处理本地文件的应用程序用来处理远地文件时，不需要对该应用程序作明显的改动。属于文件共享协议的有**网络文件系统** NFS (Network File System)[COME00]。网络文件系统 NFS 最初是在 UNIX 操作系统环境下实现文件和目录的共享。NFS 可使本地计算机共享远地的资源，就像这些资源在本地一样。由于 NFS 原先是 SUN 公司在 TCP/IP 网络上创建的，因此目前 NFS 主要应用在 TCP/IP 网络上。然而现在 NFS 也可在 OS/2, MS-Windows, NetWare 等操作系统上运行。NFS 还没有成为因特网的正式标准，现在的版本 4(NFSv4)是 2000 年底发表的[RFC 3010]，目前还只是建议标准。限于篇幅，本书不讨论 NFS 的详细工作过程。

8.2.2 FTP 的基本工作原理

网络环境中的一项基本应用就是将文件从一台计算机中复制到另一台可能相距很远的计算机中。初看起来，在两个主机之间传送文件是很简单的事情。其实这往往非常困难。原因是众多的计算机厂商研制出的文件系统多达数百种，且差别很大。经常遇到的问题是：

（1）计算机存储数据的格式不同。

（2）文件的目录结构和文件命名的规定不同。

（3）对于相同的文件存取功能，操作系统使用的命令不同。

（4）访问控制方法不同。

文件传送协议 FTP 只提供文件传送的一些基本的服务，它使用 TCP 可靠的运输服务。FTP 的主要功能是减少或消除在不同操作系统下处理文件的不兼容性。

FTP 使用客户服务器方式。一个 FTP 服务器进程可同时为多个客户进程提供服务。FTP 的服务器进程由两大部分组成：一个**主进程**，负责接受新的请求；另外有若干个**从属进程**，负责处理单个请求。

主进程的工作步骤如下：

（1）打开熟知端口（端口号为 21），使客户进程能够连接上。

（2）等待客户进程发出连接请求。

（3）启动从属进程来处理客户进程发来的请求。从属进程对客户进程的请求处理完毕后即终止，但从属进程在运行期间根据需要还可能创建其他一些子进程。

（4）回到等待状态，继续接受其他客户进程发来的请求。主进程与从属进程的处理是并发地进行。

FTP 的工作情况如图 8-5 所示。图中的圆圈表示在系统中运行的进程。图中的服务器端有两个从属进程：**控制进程**和**数据传送进程**。为简单起见，服务器端的主进程没有画上。在客户端除控制进程和数据传送进程外，还有一个用户界面进程用来和用户接口。

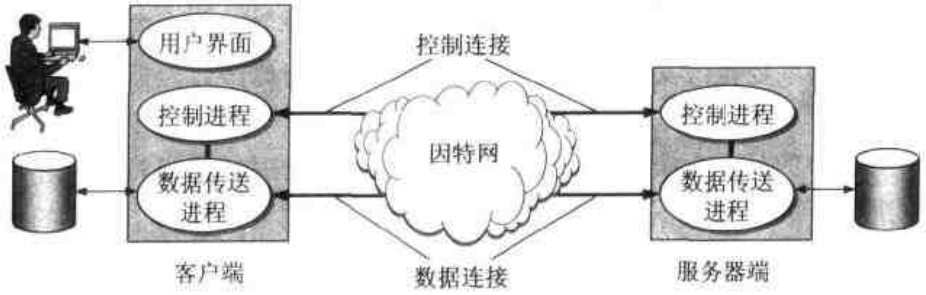


图 8-5　FTP 使用的两个 TCP 连接

在进行文件传输时，FTP 的客户和服务器之间要建立两个连接：“**控制连接**”和“**数据连接**”。控制连接在整个会话期间一直保持打开，FTP 客户所发出的传送请求通过控制连接发送给服务器端的控制进程，但控制连接并不用来传送文件。**实际用于传输文件的是“数据连接”**。服务器端的控制进程在接收到 FTP 客户发送来的文件传输请求后就创建“**数据传送进程**”和“**数据连接**”，数据连接用来连接到客户端和服务器端的数据传送进程，数据传送进程实际完成文件的传送，在传送完毕后关闭“数据传送连接”，并结束运行。

当客户进程向服务器进程发出建立连接请求时，要寻找连接服务器进程的熟知端口(21)，同时还要告诉服务器进程自己的另一个端口号码，用于建立数据传送连接。接着，服务器进程用自己传送数据的熟知端口(20)与客户进程所提供的端口号码建立数据传送连接。由于 FTP 使用了两个不同的端口号，所以数据连接与控制连接不会发生混乱。

使用两个独立的连接的主要好处是使协议更加简单和更容易实现，同时在传输文件时还可以利用控制连接（例如，客户发送请求终止传输）。

FTP 一般都是交互式地工作。作为例子，图 8-6 给出了用户机器上显示出的信息。但最左边的编号[01]～[15]并不是屏幕信息而是教材编者增加的，目的是便于在后面加上相应的说明。图中的黑体字是用户键入的字符。在用户键入完毕后还要键入回车键。

图中的各行信息的解释如下：

[01] 用户要用 FTP 和远地主机（网络信息中心 NIC 上的主机）建立连接。

[02] 本地 FTP 发出的连接成功信息。

[03] 从远地服务器返回的信息，220 表示“服务就绪”。

[04] 本地 FTP 提示用户键入名字。用户键入的名字表示“匿名”。在因特网上有许多文件免费向公众提供。用户不需要键入自己的真实姓名而只需键入 anonymous 即可。

[05] 数字 331 表示“用户名正确”，需要口令。

[06] 本地 FTP 提示用户键入口令。用户这时可键入 guest 作为匿名的口令，也可以键入自己的电子邮件地址，即耶鲁大学数学系名为 xyz 的主机上的 abc。

```
[01] ftp nic.ddn.mil
[02] connected to nic.ddn.mil
[03] 220 nic FTP server (Sunos 4.1)ready.
[04] Name: anonymous
[05] 331 Guest login ok, send ident as password.
[06] Password: abc@xyz.math.yale.edu
[07] 230 Guest login ok, access restrictions apply.
[08] ftp> cd rfc
[09] 250 CWD command successful.
[10] ftp> get rfc1261.txt nicinfo
[11] 200 PORT command successful.
[12] 150 ASCII data connection for rfc1261.txt
     (128.36.12.27,1401) (4318 bytes).
[13] 226 ASCII Transfer complete.
     local: nicinfo remote: rfc1261.txt
     4488 bytes received in 15 seconds (0.3 Kbytes/s).
[14] ftp> quit
[15] 221 Goodbye.
```

图 8-6 FTP 的屏幕信息举例

[07] 数字 230 表示用户已经注册完毕。

[08] “ftp>”是 FTP 的提示信息。用户键入的是将目录改变为包含 RFC 文件的目录。

[09] 字符 CWD 是 FTP 的标准命令，代表 Change Working Directory。

[10] 用户要求将名为 rfc1261.txt 的文件复制到本地主机上，并改名为 nicinfo。

[11] 字符 PORT 是 FTP 的标准命令，表示要建立数据连接。200 表示“命令正确”。

[12] 数字 150 表示“文件状态正确，即将建立数据连接”。

[13] 数字 226 是“释放数据连接”。现在一个新的本地文件已产生。

[14] 用户键入退出命令。

[15] 表明 FTP 工作结束。

FTP 并非对所有的数据传输都是最佳的。例如，计算机 A 上运行的应用程序要在远地计算机 B 的一个很大的文件末尾添加一行信息。若使用 FTP，则应先将此文件从计算机 B 传送到计算机 A。添加上这一行信息后，再用 FTP 将此文件传送到计算机 B。来回传送这样大的文件很花时间。实际上这种传送是不必要的，因为计算机 A 并没有使用该文件的内容。

然而网络文件系统 NFS 则采用另一种思路。**NFS 允许应用进程打开一个远地文件，并能在该文件的某一个特定的位置上开始读写数据。**这样，NFS 可使用户只复制一个大文件中的一个很小的片段，而不需要复制整个大文件。对于上述例子，计算机 A 中的 NFS 客户软件，将要添加的数据和在文件后面写数据的请求一起发送到远地的计算机 B 中的 NFS 服务器。NFS 服务器更新文件后返回应答信息。**在网络上传送的只是少量的修改数据。**

8.2.3 简单文件传送协议 TFTP

TCP/IP 协议族中还有一个**简单文件传送协议** TFTP (Trivial File Transfer Protocol)，它是一个很小且易于实现的文件传送协议。TFTP 的版本 2 是因特网的正式标准[RFC 1350]。虽然 TFTP 也使用客户服务器方式，但它使用 UDP 数据报，因此 TFTP 需要有自己的差错改正措施。TFTP 只支持文件传输而不支持交互。TFTP 没有一个庞大的命令集，没有列目录的功能，

也不能对用户进行身份鉴别。

TFTP 的主要优点有两个。第一，TFTP 可用于 UDP 环境。例如，当需要将程序或文件同时向许多机器下载时就往往需要使用 TFTP。第二，TFTP 代码所占的内存较小。这对较小的计算机或某些特殊用途的设备是很重要的。这些设备不需要硬盘，只需要固化了 TFTP, UDP 和 IP 的小容量只读存储器即可。当接通电源后，设备执行只读存储器中的代码，在网络上广播一个 TFTP 请求。网络上的 TFTP 服务器就发送响应，其中包括可执行二进制程序。设备收到此文件后将其放入内存，然后开始运行程序。这种方式增加了灵活性，也减少了开销。

TFTP 的主要特点是：

（1）每次传送的数据 PDU 中有 512 字节的数据，但最后一次可不足 512 字节。

（2）数据 PDU 也称为**文件块**(block)，每个块按序编号，从 1 开始。

（3）支持 ASCII 码或二进制传送。

（4）可对文件进行读或写。

（5）使用很简单的首部。

TFTP 的工作很像停止等待协议。发送完一个文件块后就等待对方的确认，确认时应指明所确认的块编号。发完数据后在规定时间内收不到确认就要重发数据 PDU。发送确认 PDU 的一方若在规定时间内收不到下一个文件块，也要重发确认 PDU。这样就可保证文件的传送不致因某一个数据报的丢失而告失败。

在一开始工作时。TFTP 客户进程发送一个读请求 PDU 或写请求 PDU 给 TFTP 服务器进程，其熟知端口号码为 69。TFTP 服务器进程要选择一个新的端口和 TFTP 客户进程进行通信。若文件长度恰好为 512 字节的整数倍，则在文件传送完毕后，还必须在最后发送一个只含首部而无数据的数据 PDU。若文件长度不是 512 字节的整数倍，则最后传送数据 PDU 的数据字段一定不满 512 字节，这正好可作为文件结束的标志。

TFTP 共有五种协议数据单元 PDU，即：读请求 PDU、写请求 PDU、数据 PDU、确认 PDU 和差错 PDU。图 8-7 是这五种 PDU 的格式。

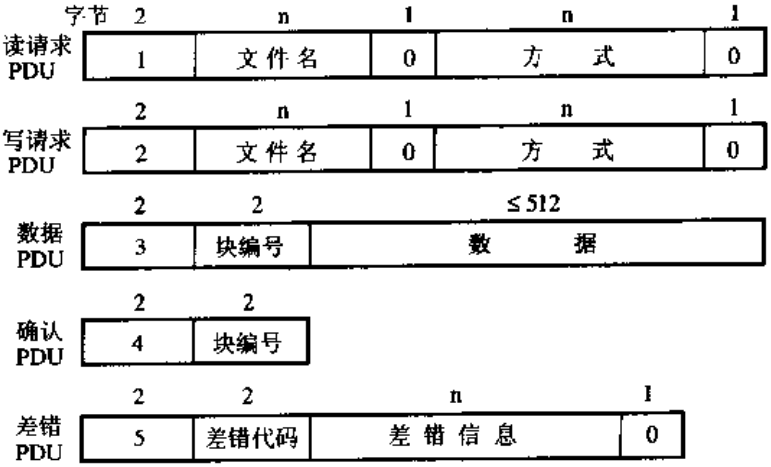


图 8-7 TFTP PDU 的格式

每种 PDU 的第一个字段是操作码字段，分别填入 1 至 5。读或写 PDU 中的方式字段用来区分不同类型的文件（如 ASCII 文件或二进制文件等）。在出现差错时就结束文

件的传送。

8.3 远程终端协议 TELNET

TELNET 是一个简单的远程终端协议[RFC 854]，它也是因特网的正式标准。用户用 TELNET 就可在其所在地通过 TCP 连接注册（即登录）到远地的另一个主机上（使用主机名或 IP 地址）。TELNET 能将用户的击键传到远地主机，同时也能将远地主机的输出通过 TCP 连接返回到用户屏幕。这种服务是透明的，因为用户感觉到好像键盘和显示器是直接连在远地主机上。

TELNET 并不复杂，以前应用得很多。现在由于 PC 机的功能越来越强，用户已较少使用 TELNET 了。TELNET 也使用客户服务器方式。在本地系统运行 TELNET 客户进程，而在远地主机则运行 TELNET 服务器进程。和 FTP 的情况相似，服务器中的主进程等待新的请求，并产生从属进程来处理每一个连接。

TELNET 能够适应许多计算机和操作系统的差异。例如，对于文本中一行的结束，有的系统使用 ASCII 码的回车(CR)，有的系统使用换行(LF)，还有的系统使用两个字符，回车-换行(CR-LF)。又如，许多系统在中断一个程序时使用 Control-C (^C)，但也有系统使用 ESC 按键。为了适应这种差异，TELNET 定义了数据和命令应怎样通过因特网。这些定义就是所谓的**网络虚拟终端** NVT (Network Virtual Terminal)。图 8-8 说明了 NVT 的意义。客户软件把用户的击键和命令转换成 NVT 格式，并送交服务器。服务器软件把收到的数据和命令，从 NVT 格式转换成远地系统所需的格式。向用户返回数据时，服务器把远地系统的格式转换为 NVT 格式，本地客户再从 NVT 格式转换到本地系统所需的格式。

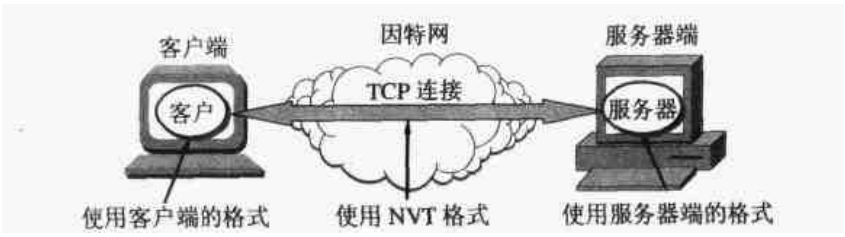


图 8-8 TELNET 使用网络虚拟终端 NVT 格式

NVT 的格式定义很简单。所有的通信都使用 8 bit 的字节。在运转时，NVT 使用 7 位 ASCII 码传送数据，而当高位置 1 时用作控制命令。ASCII 码共有 95 个可打印字符（如字母、数字、标点符号）和 33 个控制字符。所有可打印字符在 NVT 中的意义和在 ASCII 码中一样。但 NVT 只使用了 ASCII 码的控制字符中的几个（见表 8-1）。

表 8-1 TELNET NVT 使用的 ASCII 码控制字符

ASCII 控制码	十进制值	意　义
NUL	0	无操作
BEL	7	声音或可视信号
BS	8	左移一个字符位置
HT	9	右移到下一个水平制表符位置
LF	10	垂直下移到下一行

续表

ASCII 控制码	十进制值	意　义
VT	11	下移到下一个垂直制表符位置
FF	12	移到下一页的顶部
CR	13	移到当前行的左边界
其他控制		无操作

此外，NVT 还定义了两字符的 CR-LF 为标准的行结束控制符。当用户键入回车按键时，TELNET 的客户就将其转换为 CR-LF 再进行传输，而 TELNET 服务器要将 CR-LF 转换为远地机器的行结束字符。

虽然 TELNET 的 NVT 的功能非常简单，但 TELNET 定义了自己的一些控制命令。通过 TELNET 的选项协商(Option Negotiation)，TELNET 客户和 TELNET 服务器还可商定使用更多的终端功能。TELNET 规定，协商的双方是平等的。协商的对话方式有以下四种：

（1）DO（选项代码）　　表示要求对方执行该选项
　　WILL（选项代码）　　同意执行此选项
（2）DO（选项代码）　　表示要求对方执行该选项
　　WON'T（选项代码）　　不同意，状态不变
（3）WILL（选项代码）　　表示我想执行该选项
　　DO（选项代码）　　同意执行此选项
（4）WILL（选项代码）　　表示我想执行该选项
　　DON'T（选项代码）　　不同意，状态不变
　　WON'T（选项代码）　　证实状态不变

以上的 WILL，WON'T，DO 和 DON'T 是 TELNET 的协商命令，它们的十进制值分别是 251～254（ASCII 码对应的十进制值是 0～127）。

TELNET 将十进制值为 255 的代码规定为 IAC (Interpret As Command，意思是“解释为命令”)。这是一个特殊的代码。凡出现在 IAC 之后的一个字节就是 TELNET 的命令。如果要发送的数据中恰好出现和 IAC 一样的组合，则在它的前面要增加一个 IAC。因此在收到的数据中出现一连两个 IAC 时，就去掉一个，面剩下的一个是数据。TELNET 的一些命令（十进制值 240～250）和选项的代码都可在有关手册中查到，这里从略。

8.4　电子邮件

8.4.1　概述

大家知道，可以进行实时通信的电话有两个严重缺点。第一，电话通信的主叫和被叫双方必须同时在场。但据一些统计资料，大约有 70 %的业务电话不能在第一次呼叫时直接传到被叫人。第二，一些不是十分紧迫的电话也常常不必要地打断人们正在进行的工作。

电子邮件(e-mail)是因特网上使用得最多的和最受用户欢迎的一种应用。电子邮件将邮件发送到 ISP 的邮件服务器，并放在其中的收信人**邮箱**(mail box)中，收信人可随时上网到 ISP

的邮件服务器进行读取。上述的性质相当于利用因特网为用户设立了存放邮件的信箱，因此e-mail 有时也称为“**电子信箱**”。电子邮件不仅使用方便，而且还具有传递迅速和费用低廉的优点。据有的公司报道，使用电子邮件后可提高劳动生产率 30 %以上。现在电子邮件不仅可传送文字信息，而且还可附上声音和图像。由于电子邮件的广泛使用，现已很少有人愿意到邮电局去打电报，因为这种传统电报既贵又慢，且十分不方便。

最初的电子邮件系统的功能很简单。邮件无标准的内部结构格式，计算机很难对邮件进行处理。用户接口也不好。用户将邮件编辑完毕后必须退出邮件编辑程序，再调用文件传送程序方能传送已编辑好的邮件。但经过人们的努力，在 1982 年就制定出 ARPANET 上的电子邮件标准：**简单邮件传送协议** SMTP (Simple Mail Transfer Protocol)[RFC 821]和**因特网文本报文格式**[RFC 822]，它们都已成为因特网的正式标准。两年以后，CCITT 制定了**报文处理系统** MHS 的标准，即 X.400 建议书。以后 OSI 又在此基础上制定了一个**面向报文的电文交换系统** MOTIF (Message Oriented Text Interchange System)的标准。在 1988 年，CCITT 参照 MOTIF 修改了 X.400。X.400 是一个功能很强的电子邮件标准。

由于因特网的 SMTP 只能传送可打印的 7 位 ASCII 码邮件，因此在 1993 年又提出了**通用因特网邮件扩充** MIME (Multipurpose Internet Mail Extensions)。1996 年经修订后已成为因特网的草案标准[RFC 2045～2049]。MIME 在其邮件首部中说明了邮件的数据类型（如文本、声音、图像、视像等）。在 MIME 邮件中可同时传送多种类型的数据。这在多媒体通信的环境下是非常有用的。

经过 10 年的竞争，全世界都已广泛地使用了因特网的电子邮件系统，而 X.400 却基本上消声匿迹了。因此本书不再讨论 X.400。

一个电子邮件系统应具有图 8-9 所示的三个主要组成构件，这就是**用户代理**、**邮件服务器**，以及电子邮件使用的**协议**，如 SMTP 和 POP3（或 IMAP）等。在图 8-9 中只画出了 POP3。

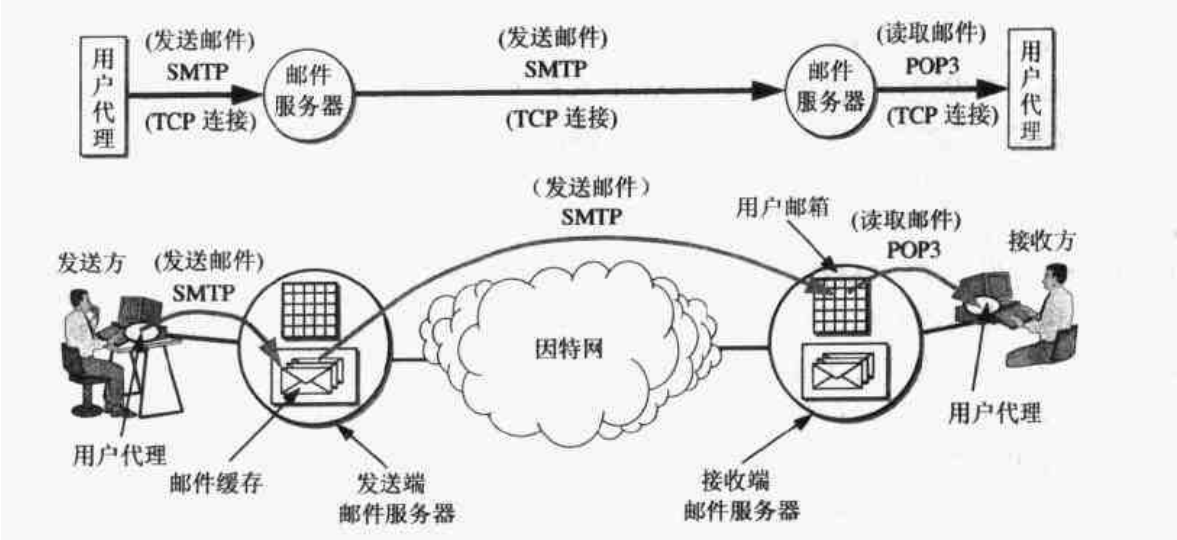


图 8-9　电子邮件的最主要的组成构件

用户代理 UA (User Agent)就是用户与电子邮件系统的接口，在大多数情况下它就是在用户 PC 机中运行的程序。用户代理使用户能够通过一个很友好的接口（目前主要是用窗口界面）来发送和接收邮件。现在可供大家选择的用户代理有很多种。例如，微软公司的 Outlook

Express 和我国张小龙制作的 Foxmail，都是很受欢迎的电子邮件用户代理。

用户代理至少应当具有以下三个功能：

（1）**撰写**。给用户提供很方便地编辑信件的环境。例如，应让用户能创建便于使用的通讯录（有常用的人名和地址）。回信时不仅能很方便地从来信中提取出对方地址，并自动地将此地址写入到邮件中合适的位置，而且还能方便地对来信提出的问题进行答复（系统自动将来信复制一份在用户撰写回信的窗口中，因而用户不需要再输入来信中的问题）。

（2）**显示**。能方便地在计算机屏幕上显示出来信（包括来信附上的声音和图像）。

（3）**处理**。处理包括发送邮件和接收邮件。收信人应能根据情况按不同方式对来信进行处理。例如，阅读后删除、存盘、打印、转发等，以及自建目录对来信进行分类保存。有时还可在读取信件之前先查看一下邮件的发信人和长度等，对于不愿收的信件可直接在邮箱中删除。

邮件服务器是电子邮件系统的核心构件，因特网上所有的 ISP 都有邮件服务器。邮件服务器的功能是发送和接收邮件，同时还要向发信人报告邮件传送的情况（已交付、被拒绝、丢失等）。邮件服务器按照客户服务器方式工作。邮件服务器需要使用**两个不同的协议**。一个协议用于**发送邮件**，即 SMTP 协议，而另一个协议用于**接收邮件**，即**邮局协议** POP (Post Office Protocol)。

这里应当注意，一个邮件服务器既可以作为客户，也可以作为服务器。例如，当邮件服务器 A 向另一个邮件服务器 B 发送邮件时，邮件服务器 A 就作为 SMTP 客户，而 B 是 SMTP 服务器。当邮件服务器 A 从另一个邮件服务器 B 接收邮件时，邮件服务器 A 就作为 SMTP 服务器，而 B 是 SMTP 客户。

下面就是一封电子邮件的发送和接收过程（结合图 8-9 来看）。

（1）发信人调用用户代理来编辑要发送的邮件。用户代理用 SMTP 将邮件传送给发送端邮件服务器。

（2）发送端邮件服务器将邮件放入邮件缓存队列中，等待发送。

（3）运行在发送端邮件服务器的 SMTP 客户进程，发现在邮件缓存中有待发送的邮件，就向运行在接收端邮件服务器的 SMTP 服务器进程发起 TCP 连接的建立。

（4）当 TCP 连接建立后，SMTP 客户进程开始向远程的 SMTP 服务器进程发送邮件。如果有多个邮件在邮件缓存中，则 SMTP 客户一一将它们发送到远程的 SMTP 服务器。当所有的待发送邮件发完了，SMTP 就关闭所建立的 TCP 连接。

（5）运行在接收端邮件服务器中的 SMTP 服务器进程收到邮件后，将邮件放入收信人的用户邮箱中，等待收信人在他方便时进行读取。

（6）收信人在打算收信时，调用用户代理，使用 POP3（或 IMAP）协议将自己的邮件从接收端邮件服务器的用户邮箱中的取回（如果邮箱中有来信的话）。

细心的读者可能会想到这样的问题，即：如果让图 8-9 中的邮件服务器程序就在发送方和接收方的 PC 机中运行，那么是否就可以将两个用户端的用户代理程序省略？

答案是“不行”。这是因为并非所有的计算机都能运行邮件服务器程序。有些计算机可能没有足够的存储器来运行允许程序在后台运行的操作系统，或是可能没有足够的 CPU 能力来运行邮件服务器程序。更重要的是，邮件服务器程序必须不间断地运行，每天 24 小时都必须不间断地连接在因特网上，否则就可能使很多外面发来的邮件丢失。这样看来，让用户的

PC 机运行邮件服务器程序显然是很不现实的（一般用户在不使用 PC 机时就将机器关闭）。让来信暂时存储在 ISP 的邮件服务器中，而当用户方便时就从邮件服务器的用户信箱中读取来信，则是一种比较合理的做法。

电子邮件由**信封**(envelope)和**内容**(content)两部分组成。电子邮件的传输程序根据邮件信封上的信息来传送邮件。用户在从自己的邮箱中读取邮件时才能见到邮件的内容。

在邮件的信封上，最重要的就是收信人的地址。TCP/IP 体系的电子邮件系统规定**电子邮件地址**(e-mail address)的格式如下：

收信人邮箱名@邮箱所在主机的域名　　(8-1)

在(8-1)式中，符号“@”读作“at”，表示“在”的意思。收信人邮箱名又简称为**用户名**(user name)，是收信人自己定义的字符串标识符。但应注意，标志收信人邮箱名的字符串在邮箱所在计算机中必须是惟一的。电子邮件的用户一般采用容易记忆的字符串，不宜使用难于记忆的字符串作为邮箱名，这只会给自己和别人增加麻烦。例如，本书编者的电子邮件地址为：xiexr@public1.ptt.js.cn。邮箱所在的主机的域名是 public1.ptt.js.cn，这就是中国电信的 ISP（用 163 拨号）在南京市的主机域名，这个域名在整个因特网的范围内必须是惟一的。当作者到这个 ISP 申请电子邮件账号时，ISP 必须保证 xiexr 这个用户名在该域名的范围内是惟一的。若已经有其他的用户选用了这个用户名，那么后来的登记者就只能选用另一个用户名。

由于一个主机的域名在因特网上是惟一的，而每一个邮箱名在该主机中也是惟一的，因此在因特网上的每一个人的电子邮件地址都是惟一的。这一点对保证电子邮件能够在整个因特网范围内的准确交付是十分重要的。

8.4.2 简单邮件传送协议 SMTP

下面介绍 SMTP 的一些主要特点。

使用 SMTP 时，收信人可以是和发信人连接在同一个本地网络上的用户，也可以是因特网上其他网络的用户，或者是与因特网相连但不是 TCP/IP 网络上的用户。

SMTP 没有规定发信人应如何将邮件提交给 SMTP，以及 SMTP 应如何将邮件投递给收信人。至于邮件内部的格式，邮件如何存储，以及邮件系统应以多快的速度来发送邮件，SMTP 也都未做出规定。SMTP 所规定的就是在两个相互通信的 SMTP 进程之间应如何交换信息。由于 SMTP 使用客户服务器方式，因此负责发送邮件的 SMTP 进程就是 SMTP 客户，而负责接收邮件的 SMTP 进程就是 SMTP 服务器。

SMTP 规定了 14 条命令和 21 种应答信息。每条命令用 4 个字母组成，而每一种应答信息一般只有一行信息，由一个 3 位数字的代码开始，后面附上（也可不附上）很简单的文字说明。下面通过 SMTP 通信的三个阶段介绍几个最主要的命令和响应信息。

1．连接建立

发信人先将要发送的邮件送到邮件缓存。SMTP 客户每隔一定时间（例如 30 分钟）对邮件缓存扫描一次。如发现有邮件，就使用 SMTP 的熟知端口号码(25)与目的主机的 SMTP 服务器建立 TCP 连接。在连接建立后，SMTP 服务器要发出“220 Service ready（服务就绪）”。然后 SMTP 客户向 SMTP 服务器发送 HELO 命令，附上发送方的主机名。SMTP 服务器若有

能力接收邮件，则回答："250 OK"，表示已准备好接收。若 SMTP 服务器不可用，则回答"421 Service not available（服务不可用）"。

如在一定时间内（例如三天）发送不了邮件，则将邮件退还发信人。

这里要强调指出，上面所说的连接并不是在发信人和收信人之间建立的。连接是在发送主机的 SMTP 客户和接收主机的 SMTP 服务器之间建立的。发信人和收信人都可以在其主机上做自己的工作，而 SMTP 客户和 SMTP 服务器都在后台工作。

SMTP 不使用中间的邮件服务器。不管发送端和接收端的邮件服务器相隔有多远，不管在邮件的传送过程中要经过多少个路由器，TCP 连接总是在发送端和接收端这两个邮件服务器之间直接建立。当接收端邮件服务器出故障而不能工作时，发送端邮件服务器只能等待一段时间后再尝试和该邮件服务器建立 TCP 连接，而不能先找一个中间的邮件服务器建立 TCP 连接。

2．邮件传送

邮件的传送从 MAIL 命令开始。MAIL 命令后面有发信人的地址。如：MAIL FROM: <xiexr@public1.ptt.js.cn>。若 SMTP 服务器已准备好接收邮件，则回答"250 OK"。否则，返回一个代码，指出原因。如：451（处理时出错），452（存储空间不够），500（命令无法识别）等。

下面跟着一个或多个 RCPT 命令，取决于将同一个邮件发送给一个或多个收信人，其格式为 RCPT TO: <收信人地址>。每发送一个命令，都应当有相应的信息从 SMTP 服务器返回，如："250 OK"，表示指明的邮箱在接收端的系统中。或"550 No such user here（无此用户）"，即不存在此邮箱。

RCPT 命令的作用就是：先弄清接收端系统是否已做好接收邮件的准备，然后才发送邮件。这样做是为了避免浪费通信资源，不致发送了很长的邮件但以后才知道是因地址错误而白白浪费了许多通信资源。

再下面就是 DATA 命令，表示要开始传送邮件的内容了。SMTP 服务器返回的信息是："354 Start mail input; end with <CRLF>.<CRLF>"。这里<CRLF>是"回车换行"的意思。若不能接收邮件，则返回 421（服务器不可用），500（命令无法识别）等。接着 SMTP 客户就发送邮件的内容。发送完毕后，再发送<CRLF>.<CRLF>（两个回车换行中间用一个点隔开）表示邮件内容结束。实际上在服务器端看到的可打印字符只是一个英文的句点。若邮件收到了，则 SMTP 服务器返回信息"250 OK"，或返回差错代码。

虽然 SMTP 使用 TCP 连接试图使邮件的传送可靠，但它并不能保证不丢失邮件。没有端到端的确认返回到收信人处。差错指示也不保证能传送到收信人处。然而基于 SMTP 的电子邮件通常都被认为是可靠的。

3．连接释放

邮件发送完毕后，SMTP 客户应发送 QUIT 命令。SMTP 服务器返回的信息是"221（服务关闭）"，表示 SMTP 同意释放 TCP 连接。邮件传送的全部过程即结束。

这里再强调一下，上述的 SMTP 客户与服务器交互的过程都被电子邮件系统的用户代理屏蔽了，**使用电子邮件的用户是看不见这些过程的**。

8.4.3 电子邮件的信息格式

一个电子邮件分为**信封**和**内容**两大部分。[RFC 822]只规定了邮件内容中的**首部**(header)格式，而对邮件的**主体**(body)部分则让用户自由撰写。用户写好首部后，邮件系统将自动地将信封所需的信息提取出来并写在信封上。所以用户不需要填写电子邮件信封上的信息。

邮件内容首部包括一些关键字，后面加上冒号。最重要的关键字是：To 和 Subject。

“To:”后面填入一个或多个收信人的电子邮件地址。在电子邮件软件中，用户把经常通信的对象姓名和电子邮件地址写到**地址簿**(address book)中。当撰写邮件时，只需打开地址簿，点击收信人名字，收信人的电子邮件地址就会自动地填入到合适的位置上。

“Subject:”是邮件的**主题**。它反映了邮件的主要内容。主题类似于文件系统的文件名，便于用户查找邮件。

邮件首部还有一项是**抄送**“Cc:”。这两个字符来自“Carbon copy”，意思是留下一个“**复写副本**”。这是借用旧的名词，表示应给某某人发送一个邮件副本。

有些邮件系统允许用户使用关键字 Bcc (Blind carbon copy)来实现**盲复写副本**。这是使发信人能将邮件的副本送给某人，但不希望此事为收信人知道。Bcc 又称为**暗送**。

首部关键字还有“From”和“Date”，表示**发信人的电子邮件地址**和**发信日期**。这两项一般都由邮件系统自动填入。

另一个关键字是“Reply-To”，即对方回信所用的地址。这个地址可以与发信人发信时所用的地址不同。例如有时到外地借用他人的邮箱给自己的朋友发送邮件，但仍希望对方将回信发送到自己的邮箱。这一项可以事先设置好，不需要在每次写信时进行设置。

8.4.4 邮件读取协议 POP3 和 IMAP

现在常用的邮件读取协议有两个，即邮局协议第 3 个版本 POP3 和**因特网报文存取协议** IMAP (Internet Message Access Protocol)。现分别讨论如下。

邮局协议 POP 是一个非常简单、但功能有限的邮件读取协议。邮局协议 POP 最初公布于 1984 年[RFC 918]。经过几次的更新，现在使用的是它的第三个版本 POP3 [RFC 1939]，它已成为因特网的正式标准。大多数的 ISP 都支持 POP。

POP 也使用客户服务器的工作方式。在接收邮件的用户 PC 机中必须运行 POP 客户程序，而在用户所连接的 ISP 的邮件服务器中则运行 POP 服务器程序。当然，这个 ISP 的邮件服务器还必须运行 SMTP 服务器程序，以便接收发送方邮件服务器的 SMTP 客户程序发来的邮件。这些请参阅图 8-9。POP 服务器只有在用户输入鉴别信息（用户名和口令）后才允许对邮箱进行读取。应当注意的是，邮件服务器只能向其他邮件服务器传输电子邮件，但 POP 服务器还能向用户提供邮箱内容的信息。

对依靠拨号连接的用户来说，POP 协议是使用得最普遍的。当用户拨号上网连接成功后，就可以运行 POP 客户程序，并与所连接的 ISP 邮件服务器的 POP 服务器程序建立 TCP 连接（POP 客户和 POP 服务器之间要交互一些命令与响应，但对收信人来说都是透明的），然后就可以接收电子邮件了。

POP3 协议的一个特点就是只要用户从 POP 服务器读取了邮件，POP 服务器就将该邮件删除。这在某些情况下就不够方便。例如，某用户在办公室的计算机上接收了一些邮件，还

来不及写回信，就马上携带笔记本电脑出差。当他打开笔记本电脑写回信时，却无法再看到原先在办公室收到的邮件(除非他事先将这些邮件复制到笔记本电脑中)。为了解决这一问题，POP3 进行了一些功能扩充，其中包括使用户能够事先设置邮件读取后仍然在 POP 服务器中存放的时间[RFC 2449]。目前[RFC 2449]还只是因特网建议标准。

下面介绍因特网报文存取协议 IMAP，它比 POP3 复杂得多。IMAP 和 POP 都按客户服务器方式工作，但它们有很大的差别。现在较新的版本是 1996 年的版本 4，即 IMAP4 [RFC 2060]，它目前还只是因特网的建议标准。

在使用 IMAP 时，所有收到的邮件同样是先送到 ISP 的邮件服务器的 IMAP 服务器。而在用户的 PC 机上运行 IMAP 客户程序，然后与 ISP 的邮件服务器上的 IMAP 服务器程序建立 TCP 连接。用户在自己的 PC 机上就可以操纵 ISP 的邮件服务器的邮箱，就像在本地操纵一样，因此 IMAP 是一个联机协议。当用户 PC 机上的 IMAP 客户程序打开 IMAP 服务器的邮箱时，用户就可看到邮件的首部。若用户需要打开某个邮件，则该邮件才传到用户的计算机上。用户可以根据需要为自己的邮箱创建便于分类管理的层次式的邮箱文件夹，并且能够将存放的邮件从某一个文件夹中移动到另一个文件夹中。用户也可按某种条件对邮件进行查找。在用户未发出删除邮件的命令之前，IMAP 服务器邮箱中的邮件一直保存着。这样就省去了用户 PC 机硬盘上的大量存储空间。

IMAP 最大的好处就是用户可以在不同的地方使用不同的计算机（例如，使用办公室的计算机、或家中的计算机，或在外地使用笔记本计算机）随时上网阅读和处理自己的邮件。IMAP 还允许收信人只读取邮件中的某一个部分。例如，收到了一个带有视像附件（此文件可能很大）的邮件。用户使用的是无线上网，信道的传输速率很低。为了节省时间，可以先下载邮件的正文部分，待以后有时间再读取或下载这个很长的附件。

IMAP 的缺点是如果用户没有将邮件复制到自己的 PC 机上，则邮件一直是存放在 IMAP 服务器上。因此用户需要经常与 IMAP 服务器建立连接（因而许多用户要考虑到所花费的上网费）。

最后再强调一下，不要将邮件读取协议 POP 或 IMAP 与邮件传送协议 SMTP 弄混。发信人的用户代理向源邮件服务器发送邮件，以及源邮件服务器向目的邮件服务器发送邮件，都是使用 SMTP 协议。而 POP 或 IMAP 则是用户从目的邮件服务器上读取邮件所使用的协议。

8.4.5 通用因特网邮件扩充 MIME

1. MIME 概述

前面所述的电子邮件协议 SMTP 有以下缺点：

（1）SMTP 不能传送可执行文件或其他的二进制对象。人们曾试图将二进制文件转换为 SMTP 使用的 ASCII 文本，例如流行的 UNIX UUencode/UUdecode 方案，但这些均未形成正式标准或事实上的标准。

（2）SMTP 限于传送 7 位的 ASCII 码。许多其他非英语国家的文字（如中文、俄文，甚至带重音符号的法文或德文）就无法传送。即使在 SMTP 网关将 EBCDIC 码（即扩充的二/十进制交换码）转换为 ASCII 码时也会遇到一些麻烦。

（3）SMTP 服务器会拒绝超过一定长度的邮件。

（4）某些 SMTP 的实现并没有完全按照[RFC 821]的 SMTP 标准。常见的问题如下：

- 回车、换行的删除和增加。
- 超过 76 个字符时的处理：截断或自动换行。
- 后面多余空格的删除。
- 将制表符 tab 转换为若干个空格。

于是在这种情况下就提出了**通用因特网邮件扩充** MIME [RFC 2045～2049]。MIME 并没有改动 SMTP 或取代它。MIME 的意图是继续使用目前的[RFC 822]格式，但增加了邮件主体的结构，并定义了传送非 ASCII 码的编码规则。也就是说，MIME 邮件可在现有的电子邮件程序和协议下传送。图 8-10 表示 MIME 和 SMTP 的关系。

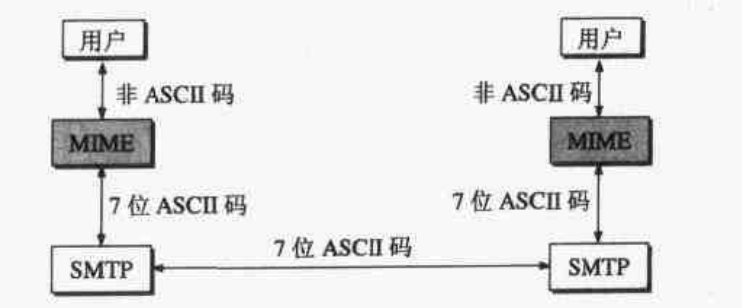


图 8-10　MIME 和 SMTP 的关系

MIME 主要包括以下三部分内容：

（1）5 个新的邮件首部字段，它们可包含在[RFC 822]首部中。这些字段提供了有关邮件主体的信息。

（2）定义了许多邮件内容的格式，对多媒体电子邮件的表示方法进行了标准化。

（3）定义了传送编码，可对任何内容格式进行转换，而不会被邮件系统改变。

为适应于任意数据类型和表示，每个 MIME 报文包含告知收信人数据类型和使用编码的信息。MIME 将增加的信息加入到[RFC 822]邮件首部中。下面是 MIME 增加的 5 个新的邮件首部的名称及其意义（有的可以是选项）。

- MIME-Version: 标志 MIME 的版本。现在的版本号是 1.0。若无此行，则为英文文本。
- Content-Description: 这是可读字符串，说明此邮件是什么。和邮件的主题差不多。
- Content-Id: 邮件的惟一标识符。
- Content-Transfer-Encoding: 在传送时邮件的主体是如何编码的。
- Content-Type: 说明邮件的性质。

上述的前三项的意思很清楚，因此下面只对上述的后两项进行介绍。

2．内容传送编码

下面介绍三种常用的**内容传送编码** Content-Transfer-Encoding。

最简单的编码就是 7 位 ASCII 码，而每行不能超过 1000 个字符。MIME 对这种由 ASCII 码构成的邮件主体不进行任何转换。

另一种编码称为 quoted-printable，这种编码方法适用于当所传送的数据中只有少量的非 ASCII 码，例如汉字。这种编码方法的要点就是对于所有可打印的 ASCII 码，除特殊字符等

号“=”外，**都不改变**。等号“=”和不可打印的 ASCII 码以及非 ASCII 码的数据的编码方法是：先将每个字节的二进制代码用两个十六进制数字表示，然后在前面再加上一个等号“=”。例如，汉字的“系统”的二进制编码是：11001111 10110101 11001101 10110011（共有 32 bit，但这四个字节都不是 ASCII 码），其十六进制数字表示为：CFB5CDB3。用 quoted-printable 编码表示为：=CF=B5=CD=B3，这 12 个字符都是可打印的 ASCII 字符，它们的二进制编码[①]需要 96bit，和原来的 32bit 相比，开销达 200%。而等号“=”的二进制代码为 00111101，即十六进制的 3D，因此等号“=”的 quoted-printable 编码为“=3D”。

对于任意的二进制文件，可用 base64 编码。这种编码方法是先将二进制代码划分为一个个 24 bit 长的单元，然后将每一个 24 bit 单元划分为 4 个 6 bit 组。每一个 6 bit 组按以下方法转换成 ASCII 码。6 bit 的二进制代码共有 64 种不同的值，从 0 到 63。用 A 表示 0，用 B 表示 1，等等。26 个大写字母排列完毕后，接下去再排 26 个小写字母，再后面是 10 个数字，最后用 + 表示 62，而用 / 表示 63。再用两个连在一起的等号 == 和一个等号 = 分别表示最后一组的代码只有 8 或 16 比特。回车和换行都忽略，它们可在任何地方插入。作为 base64 编码的例子，假设有二进制代码，共 24 bit：01001001 00110001 01111001。先划分为 4 个 6 bit 组，即 010010 010011 000101 111001。对应的 base64 编码为：STF5。最后，将 STF5 用 ASCII 编码进行发送，即 01010011 01010100 01000110 00110101。我们不难看出，24 bit 的二进制代码采用 base64 编码后变成了 32 bit，开销为 25%。当需要传送的数据大部分都是 ASCII 码时，最好还是采用 quoted-printable 编码。

3．内容类型

MIME 标准规定 Content-Type 后面的说明中必须含有两个标识符，即内容**类型**(type)和**子类型**(subtype)，中间用“/”分开。

MIME 标准定义了 7 个基本内容类型和 15 种子类型。除了内容类型和子类型，MIME 允许发信人和收信人自己定义专用的内容类型。但为避免可能出现名字冲突，标准要求为专用的内容类型选择的名字要以字符串 X-开始。表 8-2 列出了 7 种基本内容类型和 15 种子类型，以及简单的说明[②]。

表 8-2　可出现在 MIME Content-Type 说明中的七种基本类型及其意义

内 容 类 型	子 类 型	说　明
Text(正文)	plain	无格式的文本
	richtext	有少量格式命令的文本
Image(图像)	gif	GIF 格式的静止图像
	jpeg	JPEG 格式的静止图像
Audio(音频)	basic	可听见的声音
Video(视频)	mpeg	MPEG 格式的影片

① 注：ASCII 码本来定义为 7 位码。但最初为了增加检错能力就增加了一个奇偶检验位，因而使用一个字节(8 bit)表示一个 ASCII 码。于是 ASCII 码就变成了 8 位码，但其最高位一定是 0，而后面的 7 位就是最初定义的 ASCII 编码。

② 注：GIF (Graphics Interchange Format)和 JPEG (Joint Photographic Expert Group)都是静止图像（如照片）压缩的标准，而 MPEG (Motion Picture Experts Group)是活动图像（如电影）的压缩标准。PostScript 是 Adobe Systems 开发的一种编程语言，用于描述打印出的页面。

续表

内容类型	子类型	说明
Application(应用)	octet-stream	不间断的字节序列
	postscript	PostScript 可打印文档
Message(报文)	rfc822	MIME RFC 822 邮件
	partial	为传输将邮件分割开
	external-body	邮件必须从网上获取
multipart(多部分)	mixed	按规定顺序的几个独立部分
	alternative	不同格式的同一邮件
	parallel	必须同时读取的几个部分
	digest	每一个部分是一个完整的 RFC 822 邮件

MIME 的内容类型中的 multipart 是很有用的，因为它使邮件增加了相当大的灵活性。标准为 multipart 定义了 4 种可能的子类型，每个子类型都提供重要功能。

（1）mixed 子类型允许单个报文含有多个相互独立的子报文，每个子报文可有自己的类型和编码。mixed 子类型报文使用户能够在单个报文中附上文本、图形和声音，或者用额外数据段发送一个备忘录，类似商业信笺含有的附件。在 mixed 后面还要用到一个关键字，即 Boundary=，此关键字定义了分隔报文各部分所用的字符串（由邮件系统定义），只要在邮件的内容中不会出现这样的字符串即可。当某一行以两个连字符“--”开始，后面紧跟上述的字符串，就表示下面开始了另一个子报文。

（2）alternative 子类型允许单个报文含有同一数据的多种表示。当给多个使用不同硬件和软件系统的收信人发送备忘录时，这种类型的 multipart 报文很有用。例如，用户可同时用普通的 ASCII 文本和格式化的形式发送文本，从而允许拥有图形功能的计算机用户在察看图形时选择格式化的形式。

（3）parallel 子类型允许单个报文含有可同时显示的各个子部分（如图像和声音子部分必须一起播放）。

（4）digest 子类型允许单个报文含有一组其他报文（如从讨论中收集电子邮件报文）。

下面显示了一个 MIME 邮件，它包含有一个简单解释的文本和含有非文本信息的照片。邮件中第一部分的注解说明第二部分含有一张照片。

```
From: xiexr@public1.ptt.js.cn
To: xyz@public.bta.net.cn
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=qwertyuiop

--qwertyuiop
XYZ:
你要的图片在此邮件中，收到后请回信。
                                        谢希仁

--qwertyuiop
Content-Type: image/gif
Content-Transfer-Encoding: base64
```

...data for the image（图像的数据）...

--qwertyuiop--

上面最后一行表示 boundary 的字符串后面还有两个连字符“--”，表示整个 multipart 的结束。

8.5 万维网 WWW

8.5.1 概述

万维网 WWW (World Wide Web)并非某种特殊的计算机网络。**万维网是一个大规模的、联机式的信息储藏所**，英文简称为 Web。万维网用链接的方法能非常方便地从因特网上的一个站点访问另一个站点（也就是所谓的“**链接到另一个站点**”），从而主动地按需获取丰富的信息。图 8-11 说明了万维网提供分布式服务的特点。

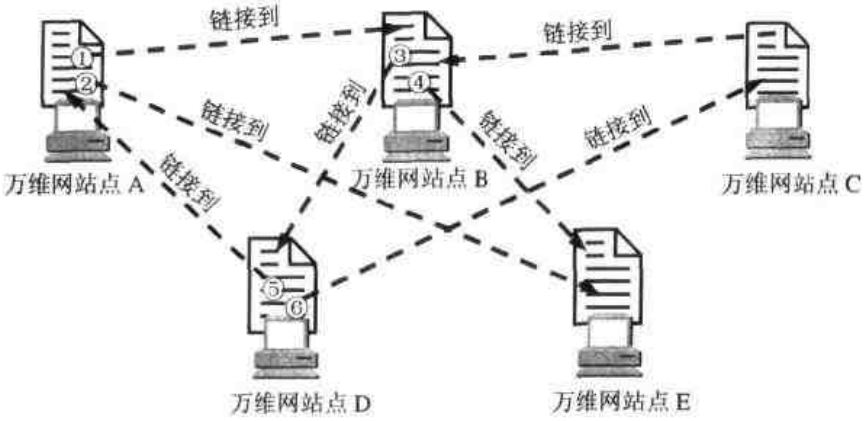


图 8-11 万维网提供分布式服务

图 8-11 画出了五个万维网上的站点，它们可以相隔数千公里，但都必须连接在因特网上。每一个万维网站点都存放了许多文档。在这些文档中有一些地方的文字是用特殊方式显示的（例如用不同的颜色，或添加了下划线），而当我们将鼠标移动到这些地方时，鼠标的箭头就变成了一只手的形状。这就表明这些地方有一个**链接**（这种链接有时也称之为**超链**），如果我们在这些地方点击鼠标，就可以从这个文档链接到可能相隔很远的另一个文档。经过一定的时延（几秒钟、几分钟甚至更长，取决于所链接的文档的大小和网络的拥塞情况），在我们的屏幕上就能将远方传送过来的文档显示出来。例如，站点 A 的某个文档中有两个地方①和②可以链接到其他的站点。当我们点击链接①时，就可链接到站点 B 的某个文档。若点击②则可链接到站点 E。站点 B 的文档也有两个地方③和④有链接。若点击链接③就可链接到站点 D，而点击链接④就链接到站点 E，但从 E 的这个文档已不能继续链接到其他任何的站点。站点 D 的文档中有两个地方⑤和⑥有链接，可以分别链接到 A 和 C。

正是由于万维网的出现，使因特网从仅由少数计算机专家使用变为普通百姓也能利用的信息资源。万维网的出现使网站数按指数规律增长。据统计，在 1998 年，万维网的通信量已超过整个因特网上通信量的 75%。因此，万维网的出现是因特网发展中一个非常重要的里程碑。

理解万维网最好的方法就是自己上因特网操作。本节不介绍万维网的具体操作步骤，而

是着重讨论有关万维网的基本概念和原理。

万维网是日内瓦的欧洲原子核研究委员会CERN(这是法文缩写)的Tim Berners-Lee最初于1989年3月提出的。开发万维网的动机是为了使分布在好几个国家的物理学家们更方便地协同工作（他们需要经常交换各种报告、图形、照片等）。1993年2月，第一个图形界面的浏览器(browser)开发成功，名字叫做Mosaic。1995年著名的Netscape Navigator浏览器上市。目前最受用户欢迎的浏览器是Netscape公司的Navigator和微软公司的Internet Explorer。

万维网是一个分布式的**超媒体**(hypermedia)系统，它是**超文本**(hypertext)系统的扩充。一个超文本由多个信息源链接成，而这些信息源的数目实际上是不受限制的。利用一个链接可使用户找到另一个**文档**，而这又可链接到其他的文档（依次类推）。这些**文档**可以位于世界上任何一个接在因特网上的超文本系统中。超文本是万维网的基础。

超媒体与**超文本**的区别是文档内容不同。超文本文档仅包含文本信息，而超媒体文档还包含其他表示方式的信息，如图形、图像、声音、动画，甚至活动视频图像。

分布式的和非分布式的超媒体系统有很大区别。在非分布式系统中，各种信息都驻留在单个计算机的磁盘中。由于各种文档都可从本地获得，因此这些文档之间的链接可进行一致性检查。因此一个非分布式超媒体系统能够保证所有的链接都是有效的和一致的。

但万维网将大量信息分布在整个因特网上。每台计算机上的文档都独立进行管理。对这些文档的增加、修改、删除或重新命名都不需要（实际上也不可能）通知到因特网上成千上万的结点。这样，万维网**文档**之间的链接就经常会不一致。例如，计算机A上的文档X本来包含了一个指向计算机B上的文档Y的链接。若计算机B的管理员在某日删除了文档Y，那么计算机A的上述链接显然就失效了。

万维网以客户服务器方式工作。上面所说的浏览器就是在用户计算机上的万维网客户程序。万维网文档所驻留的计算机则运行服务器程序，因此这个计算机也称为万维网服务器。**客户程序向服务器程序发出请求，服务器程序向客户程序送回客户所要的万维网文档**。在一个客户程序主窗口上显示出的万维网文档称为**页面**(page)。

从以上所述可以看出，万维网必须解决以下几个问题：

（1）怎样标志分布在整个因特网上的万维网文档？

（2）用什么样的协议来实现万维网上各种超链的链接？

（3）怎样使不同作者创作的不同风格的万维网文档都能在因特网上的各种计算机上显示出来，同时使用户清楚地知道在什么地方存在着超链？

（4）怎样使用户能够很方便地找到所需的信息？

为了解决第一个问题，万维网使用**统一资源定位符** URL (Uniform Resource Locator)来标志万维网上的各种**文档**，并使每一个文档在整个因特网的范围内具有惟一的标识符URL。为了解决上述的第二个问题，就要使万维网客户程序与万维网服务器程序之间的交互遵守严格的协议，这就是**超文本传送协议** HTTP (HyperText Transfer Protocol)。HTTP是一个应用层协议，它使用TCP连接进行可靠的传送。为了解决上述的第三个问题，万维网使用**超文本标记语言** HTML (HyperText Markup Language)，使得万维网页面的设计者可以很方便地用一个超链从本页面的某处链接到因特网上的任何一个万维网页面，并且能够在自己的计算机屏幕上将这些页面显示出来。最后，为了在万维网上方便地查找信息，用户可使用各种的**搜索工具**。

下面我们将进一步讨论万维网的几个重要概念，如 URL，HTTP 和 HTML 等。

8.5.2 统一资源定位符 URL

1．URL 的格式

在[RFC 1738]和[RFC 1808]中（这两个文档目前是因特网的建议标准），对 URL 是这样定义的：

“统一资源定位符 URL 是对可以从因特网上得到的资源的位置和访问方法的一种简洁的表示。URL 给资源的位置提供一种抽象的识别方法，并用这种方法给资源定位。只要能够对资源定位，系统就可以对资源进行各种操作，如存取、更新、替换和查找其属性。”

上述的“资源”是指在因特网上可以被访问的任何对象，包括文件目录、文件、文档、图像、声音等，以及与因特网相连的任何形式的数据。“资源”还包括电子邮件的地址和 USENET 新闻组[①]，或 USENET 新闻组中的报文。

URL 相当于一个文件名在网络范围的扩展。因此 URL 是与因特网相连的机器上的任何可访问对象的一个指针。由于对不同对象的访问方式不同（如通过 WWW，FTP 等），所以 URL 还指出读取某个对象时所使用的访问方式。这样，URL 的一般形式如下（即由以冒号隔开的两大部分组成，并且在 URL 中的字符对大写或小写没有要求）：

<URL 的访问方式>://<主机>:<端口>/<路径>　　(8-2)

在式(8-2)冒号左边的<URL 访问方式>中，最常用的有三种，即 ftp(文件传送协议 FTP)，http(超文本传送协议 HTTP)和 news (USENET 新闻)。

(8-2)式冒号的右边部分，<主机>一项是必须的，而<端口>和<路径>则有时可省略。

下面我们简单介绍使用得较多的前两种 URL。

2．使用 FTP 的 URL

使用 FTP 访问站点的 URL 的最简单的形式见下面的例子：

ftp://rtfm.mit.edu

这里 rtfm.mit.edu 就是在麻省理工学院 MIT 的匿名服务器 rtfm 的因特网域名。如果不使用域名而是把该服务器的点分十进制的 IP 地址写在两个斜杠后面也是可以的。假定我们要直接访问上面的服务器中在目录 pub 下的一个文件 abc.txt，那么该文件的 URL 就是：

ftp://rtfm.mit.edu/pub/abc.txt

而该目录 pub 的 URL 是：

ftp://rtfm.mit.edu/pub/

某些 FTP 服务器要求用户要提供用户名和口令，那么这时就要在<host>项之前填入用户名和口令。FTP 的默认端口号是 21，一般可省略。但有时也可使用另外的端口号。

FTP 已使用了超过 20 年。世界上已有很多 FTP 服务器使广大用户能利用它下载所需的

① 注：USENET 新闻组实际上是一个世界范围的电子公告牌(Bulletin Board)，用于发布公告、新闻和各种文章供大家使用。USENET 的价值不亚于电子邮件。现在 USENET 已组织了 6 千多个小组，分为不同的专题，参加者可对有兴趣的专题进行讨论，可根据自己的观点在因特网上发表评论或对文章增添新的内容。

各种文件。万维网的出现并没有想取消 FTP，而是要使 FTP 用起来更加方便（FTP 的用户界面不太友好，不易使用）。也许以后 FTP 也可能会消失，这是因为 HTTP 服务器可以做 FTP 所能做的一切工作。

3. 使用 HTTP 的 URL

对于万维网的网点的访问要使用 HTTP 协议。HTTP 的 URL 的一般形式是：

http://<主机>:<端口>/<路径>

HTTP 的默认端口号是 80，通常可省略。若再省略文件的<路径>项，则 URL 就指到因特网上的某个主页(home page)。主页是个很重要的概念，它可以是以下几种情况之一：

（1）一个 WWW 服务器的最高级别的页面。

（2）某一个组织或部门的一个定制的页面或目录。从这样的页面可链接到因特网上的与本组织或部门有关的其他站点。

（3）由某一个人自己设计的描述他本人情况的 WWW 页面。

例如，要查有关清华大学的信息，就可先进入到清华大学的主页，其 URL 为[①]：

http://www.tsinghua.edu.cn

这里省略了默认的端口号 80。我们从清华大学的主页入手，就可以通过许多不同的超链找到所要查找的各种有关清华大学各个部门的信息。

更复杂一些的路径是指向层次结构的从属页面。例如：

http://www.tsinghua.edu.cn/chn/yxsz/index.htm

是清华大学的“院系设置”页面的 URL。注意：上面的 URL 中使用了指向文件的路径，而文件名就是最后的 index.htm。后缀 htm（有时可写为 html）表示这是一个用超文本标记语言 HTML 写出的文件。

虽然 URL 里面的字母不分大小写，但有的页面为了读者看起来方便，故意用了一些大写字母，实际上这对使用 Windows 的 PC 用户是没有关系的。

用户使用 URL 并非仅仅能够访问万维网的页面，而且还能够通过 URL 使用其他的因特网应用程序，如 FTP，Gopher，TELNET，电子邮件以及新闻组等。更重要的是，用户在使用这些应用程序时，只使用一个程序，即浏览器。这显然是非常方便的。

还有另一个通用的万维网标识符，即**通用资源标识符** URI (Universal Resource Identifier)。请注意，这里的 U 是 Universal 而不是 Uniform 的缩写。URI 的规格说明[RFC 1630]定义了对任意命名和编址方式进行编码的语法。URI 的概念和许多细节还在发展之中。

前面所讲的 URL 有一个固有的弱点，这就是 URL 必须指向一个特定的主机。如果有的页面频繁地被访问，那么自然就希望将此页面复制几份存放在不同地点，这样就可能减少网络的通信量。但使用 URL 时必须指明该页面的所在地。例如，我们不能说：“我希望访问页面 abc，但我不想知道此页面在何处”。要解决这问题就要用到 URI，因为 URI 的好处是它使一个资源的名字与其位置无关，甚至与访问的方法都无关。URI 包括了 URL 和统一资源名字 URN (Uniform Resource Name)。因此 URI 可看成是一种广义的 URL。而 URL 只是 URI 的一

① 注：Tsinghua 是清华大学创立时所用的拼音名字（那时拼音 ts 和现在的汉语拼音字母 q 的发音一样）。由于国外都早已知道 Tsinghua 这个名字，因此现在就不使用标准的汉语拼音 qinghua。

种类型，在 URL 中指明了访问的协议以及一个特定的因特网地址。

8.5.3 超文本传送协议 HTTP

1．HTTP 的操作过程

为了使超文本的链接能够高效率地完成，需要用 HTTP 协议来传送一切必须的信息。从层次的角度看，HTTP 是**面向事务的**(transaction-oriented)[②]应用层协议，它是万维网上能够可靠地交换文件（包括文本、声音、图像等各种多媒体文件）的重要基础。

万维网的大致工作过程如图 8-12 所示。

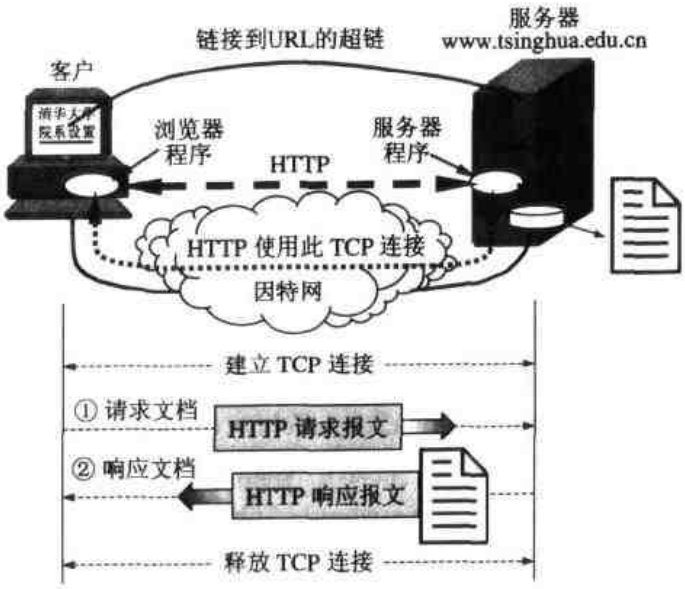


图 8-12 万维网的工作过程

每个万维网网点都有一个服务器进程，它不断地监听 TCP 的端口 80，以便发现是否有浏览器（即客户进程）向它发出连接建立请求。一旦监听到连接建立请求并建立了 TCP 连接之后，浏览器就向服务器发出浏览某个页面的请求，服务器接着就返回所请求的页面作为响应。最后，TCP 连接就被释放了。在浏览器和服务器之间的请求和响应的交互，必须按照规定的格式和遵循一定的规则。这些格式和规则就是超文本传送协议 HTTP。

HTTP 规定在 HTTP 客户与 HTTP 服务器之间的每次交互都由一个 ASCII 码串构成的请求和一个“类 MIME (MIME-like)”的响应组成。虽然大家都使用 TCP 连接进行传送，但标准并没有这样明确规定。

用户浏览页面的方法有两种。一种方法是在浏览器的地址窗口中键入所要找的页面的 URL。另一种方法是在某一个页面中用鼠标点击一个可选部分，这时浏览器自动在因特网上找到所要链接的页面。

假定图 8-12 中的用户用鼠标点击了屏幕上的一个可选部分。他使用的超链指向了“清华大学院系设置”的页面，其 URL 是 http://www.tsinghua.edu.cn/chn/yxsz/index.htm。下面更具

② 注：所谓**事务**(transaction)就是指一系列的信息交换，而这一系列的信息交换是一个不可分割的整体，即要么所有的信息交换都完成，要么一次交换都不进行。

体地说明在用户点击鼠标后所发生的几个事件：

（1）浏览器分析超链指向页面的 URL。

（2）浏览器向 DNS 请求解析 www.tsinghua.edu.cn 的 IP 地址。

（3）域名系统 DNS 解析出清华大学服务器的 IP 地址为 166.111.4.100。

（4）浏览器与服务器建立 TCP 连接（在服务器端 IP 地址是 166.111.4.100，端口是 80）

（5）浏览器发出取文件命令：GET /chn/yxsz/index.htm。

（6）服务器 www.tsinghua.edu.cn 给出响应，将文件 index.htm 发送给浏览器。

（7）TCP 连接释放。

（8）浏览器显示“清华大学院系设置”文件 index.htm 中的所有文本。

许多浏览器在下载文件时，往往只下载其中的文本部分。这样可使下载的速度加快。文件中原来嵌入图像或声音的地方只用一个小图标来显示。用户若要下载这些图像或声音，可用鼠标再分别点击这些图标。每点击一次鼠标，就重复执行一次类似于上面的 8 个步骤。也就是先建立 TCP 连接，再使用 TCP 连接传送命令和传送文件，最后释放 TCP 连接。虽然这许多步骤看来繁琐，但这样做实现起来却较为容易。

HTTP 是一个**面向事务**的客户服务器协议。虽然 HTTP 使用了 TCP，但[RFC 1945]定义的 HTTP 1.0 协议是**无状态的**(stateless)。也就是说，同一个客户第二次访问同一个服务器上的页面时，服务器的响应与第一次被访问时的相同（假定现在服务器还没有将该页面更新），因为服务器不记得曾经访问过的这个客户，也不记得曾经服务过多少次。HTTP 的无状态特性很适合它的典型应用。用户在使用万维网时，往往要读取一系列的网页，而这些网页又可能分布在许多相距很远的服务器上。将 HTTP 协议做成无状态的，可使读取网页信息完成得较迅速。HTTP 协议**本身也是无连接的**，虽然它使用了面向连接的 TCP 向上提供的服务。

在许多情况下，用户的个人计算机并不是一直连接在因特网上，而是通过拨号方式经过因特网服务提供者 ISP 再连接到因特网上的。在这种情况下，用户先要使用 PPP 协议与 ISP 接通，待 ISP 分配给用户一个临时的 IP 地址后，用户才能使用 WWW 浏览器。

从 HTTP 的观点来看，上述的万维网浏览器就是一个 HTTP 客户，而在万维网服务器等待 HTTP 请求的进程常称为 HTTP daemon[①]，有的文献将它缩写为 HTTPD。HTTP daemon 在收到 HTTP 客户的请求后，经过一些必要的处理，将所需的文件返回给 HTTP 客户。

HTTP 仍在不断地发展。现在的较新版本是 1999 年公布的 HTTP/1.1 [RFC 2616]，它已成为因特网草案标准。而下一代 HTTP (HTTP-NG)则正在研究之中[W-HTTP]。

2. 万维网高速缓存

万维网高速缓存(Web cache)是一种网络实体，它能代表浏览器发出 HTTP 请求，因此万维网高速缓存又称为**代理服务器**(proxy server)。万维网高速缓存将最近的一些请求和响应暂存在本地磁盘中。当与暂时存放的请求相同的新请求到达时，万维网高速缓存就将暂存的响应发送出去，而不需要按 URL 的地址再去因特网访问该资源。万维网高速缓存可在客户或服务器端工作，也可在中间系统上工作。下面我们用例子说明它的作用。

① 注：名词 daemon(又称 demon)来自 UNIX 中的一个术语，可译为**守护程序**。它指的是一种程序或进程，或一个大程序或进程的一个部分，它经常处于后台工作，但当某种条件满足时，daemon 就被激活并开始进行处理。基于任何平台的万维网服务器常称为 HTTPD 服务器或 HTTP 服务器，读者可以将 daemon 理解为服务器上的一个服务进程。

设图 8-13 中的校园网有许多人用 PC 机的浏览器访问因特网上不同的服务器。校园网的路由器 R_1 用 2 Mb/s 专线连接到因特网上的某个路由器 R_2。先假定不使用万维网高速缓存。现在估算一下访问因特网上的服务器的时延。

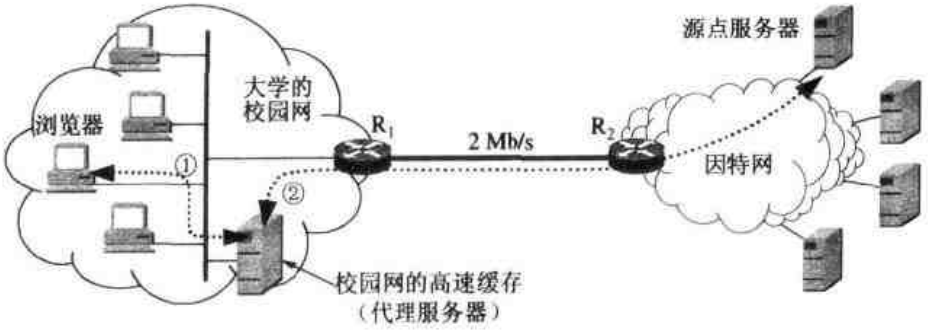


图 8-13　使用高速缓存可减少访问因特网服务器的时延

若校园网使用 10 Mb/s 以太网，平均每秒产生 20 个请求，每个请求得到的返回信息平均为 100 kbit，则每秒返回的平均信息量为 20 × 100 kbit = 2 Mbit，可知校园网以太网上的通信量强度为 0.1，这时以太网上的时延很小，一般仅几十毫秒。在因特网内，从路由器 R_2 转发 HTTP 请求报文开始，到含有响应信息的数据报传送到 R_2 为止所需的时间，在正常情况下为 2 秒左右（这就是所谓的“因特网时延”）。再观察一下 R_1 和 R_2 两个路由器之间的链路上的时延。这时在 2 Mb/s 链路上的通信量强度已达到 1。这表明在这条链路上的时延已增长到非常大的数值（例如长达几分钟以上）。用户显然无法忍受这样大的时延。

将两个路由器之间的专线增大到 10 Mb/s 就能解决这一瓶颈问题。但这要增加租用电路的费用。因此最好采用其他更加经济的方法，即在校园网增加一个**万维网高速缓存**（这不算太贵）。这时，校园网内各浏览器访问因特网上的服务器的过程是：

（1）浏览器访问因特网的服务器时，要先与校园网的高速缓存建立 TCP 连接，并向高速缓存发出 HTTP 请求报文（图 8-13 中的①）。

（2）若高速缓存已经存放了所请求的对象，则将此对象放入 HTTP 响应报文中返回给浏览器。

（3）否则，高速缓存就代表发出请求的用户浏览器，与因特网上的**源点服务器**(origin server)建立 TCP 连接（图 8-13 中的②），并发送 HTTP 请求报文。

（4）源点服务器将所请求的对象放在 HTTP 响应报文中返回给校园网的高速缓存。

（5）高速缓存收到此对象后，先复制在其本地存储器中（为今后使用），然后再将该对象放在 HTTP 响应报文中，通过已建立的 TCP 连接（图 8-13 中的①），返回给请求该对象的浏览器。

我们注意到高速缓存有时是作为服务器（当接受浏览器的 HTTP 请求时），但有时却作为客户（当向因特网上的源点服务器发送 HTTP 请求时）。

假定高速缓存的**命中率**[1]为 0.4，那就表明有 40%的 HTTP 请求可以由高速缓存来响应(通

[1] 注：**命中率**(hit rate)是高速缓存可以响应浏览器的请求数与总的请求数之比。通常命中率为 0.2~0.7 之间。

过 10 Mb/s 局域网），而 60%的 HTTP 请求仍需经过 2 Mb/s 的链路连接到因特网。这样，2 Mb/s 链路的通信量强度就从 1 下降到 0.6（一般通信量强度小于 0.8 就可接受），因而在此链路上的时延就大大降低，例如，只有几十毫秒。可以看出，在使用高速缓存的情况下，2 Mb/s 链路已不再成为访问因特网服务器的瓶颈。这时的平均访问时延是 0.4 × (几十毫秒) + 0.6 × (2 秒 + 几十毫秒)，已经小于典型的因特网时延(2 秒)了。

3．HTTP 的报文结构

了解 HTTP 功能最好的方法就是研究 HTTP 的报文结构（图 8-14）。HTTP 有两类报文：

（1）请求报文——从客户向服务器发送请求报文，见图 8-14(a)。

（2）响应报文——从服务器到客户的回答，见图 8-14(b)。

由于 HTTP 是**面向正文的**(text-oriented)，因此在报文中的每一个字段都是一些 ASCII 码串，因而每个字段的长度都是不确定的。

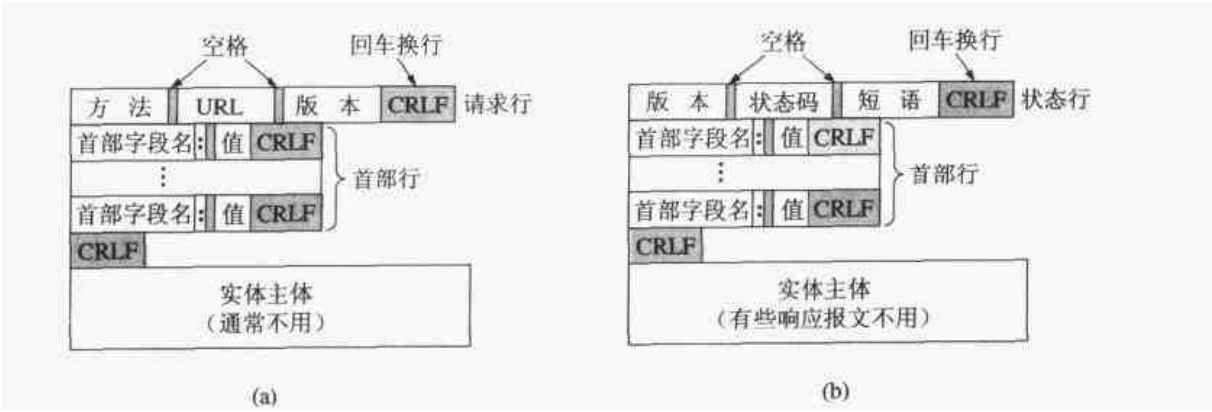


图 8-14　HTTP 的报文结构：(a)请求报文；(b)响应报文

HTTP 请求报文和响应报文都是由三个部分组成。可以看出，这两种报文格式的区别就是开始行不同。

（1）**开始行**，用于区分是请求报文还是响应报文。在请求报文中的开始行叫做**请求行**(Request-Line)，而在响应报文中的开始行叫做**状态行**(Status-Line)。在开始行的三个字段之间都以空格分隔开，最后的“CR”和“LF”分别代表“回车”和“换行”。

（2）**首部行**，用来说明浏览器、服务器或报文主体的一些信息。首部可以有好几行，但也可以不使用。在每一个首部行中都有首部字段名和它的值，每一行在结束的地方都要有“回车”和“换行”。整个首部行结束时还有一空行将首部行和后面的实体主体分开。

（3）**实体主体**(entity body)。在请求报文中一般都不用这个字段，而在响应报文中也可能没有这个字段。

下面先介绍 HTTP 请求报文最主要的一些主要特点。

请求报文的第一行“请求行”只有三个内容，即**方法**，**请求资源的 URL**，以及 HTTP 的**版本**。

请注意：这里的名词“**方法**”是面向对象技术中使用的专门名词。所谓“**方法**”就是对所请求的对象进行的**操作**，因此**这些方法实际上也就是一些命令**。因此，请求报文的类型是由它所采用的**方法**决定的。表 8-3 给出了请求报文中常用的几种方法。

表 8-3 HTTP 请求报文的一些方法

方法（操作）	意义
OPTION	请求一些选项的信息
GET	请求读取由 URL 所标志的信息
HEAD	请求读取由 URL 所标志的信息的首部
POST	给服务器添加信息（例如，注释）
PUT	在指明的 URL 下存储一个文档
DELETE	删除指明的 URL 所标志的资源
TRACE	用来进行环回测试的请求报文
CONNECT	用于代理服务器

对于我们在图 8-12 中的例子，即要链接到“清华大学院系设置”的页面。HTTP 的请求报文的开始行（即请求行）应当是（请注意在 GET 后面和 HTTP/1.1 前面的空格）：

```
GET http://www.tsinghua.edu.cn/chn/yxsz/index.htm HTTP/1.1
```

下面是一个请求报文的例子：

```
GET /chn/yxsz/index.htm HTTP/1.1     {请求行使用了相对 URL}
Host: www.tsinghua.edu.cn    {此行是首部行的开始。这行给出主机的域名}
Connection: close      {告诉服务器发送完请求的文档后就可释放连接}
User-Agent: Mozilla/5.0    {表明用户代理是使用 Netscape 浏览器}
Accept-Language: cn       {表示用户希望优先得到中文版本的文档}
{请求报文的最后还有一个空行}
```

在请求行使用了相对 URL（即省略了主机的域名）是因为下面的首部行（第 2 行）给出了主机的域名。第 3 行是告诉服务器不使用**持续连接**(persistent connection)，表示浏览器希望服务器在传送完所请求的对象后即关闭 TCP 连接（若使用持续连接就是建立一次 TCP 连接可连续传送多个请求的对象）。这个请求报文没有实体主体。

再看一下 HTTP 响应报文的主要特点。

每一个请求报文发出后，都能收到一个响应报文。响应报文的第一行就是状态行。

状态行包括三项内容，即 HTTP 的版本，状态码，以及解释状态码的简单短语。

状态码(Status-Code)都是三位数字的，分为 5 大类共 33 种。例如，

1xx 表示通知信息的，如请求收到了或正在进行处理。

2xx 表示成功，如接受或知道了。

3xx 表示重定向，表示要完成请求还必须采取进一步的行动。

4xx 表示客户的差错，如请求中有错误的语法或不能完成。

5xx 表示服务器的差错，如服务器失效无法完成请求。

下面三种状态行在响应报文中是经常见到的。

```
HTTP/1.1 202 Accepted              {接受}
HTTP/1.1 400 Bad Request           {错误的请求}
```

```
Http/1.1 404 Not Found                    {找不到}
```

若请求的网页从 http://www.ee.xyz.edu/index.html 转移到了一个新的地址，则响应报文的状态行和一个首部行就是下面的形式：

```
HTTP/1.1 301 Moved Permanently                    {永久性地转移了}
Location: http://www.xyz.edu/ee/index.html        {新的 URL}
```

8.5.4 超文本标记语言 HTML

1. HTML 概述

现在计算机使用的字处理器种类繁多而版本各异，某一台计算机屏幕上显示出的文件，在另一台机器上就未必能显示出来。万维网要使任何一台计算机都能显示出任何一个万维网服务器上的页面，就必须解决页面制作的标准化问题。超文本标记语言 HTML 就是一种制作万维网页面的标准语言，它消除了不同计算机之间信息交流的障碍。

超文本标记语言 HTML (HyperText Markup Language)中的 Markup 的意思就是“**设置标记**”。因此 HTML 也常译为**超文本置标语言**。这就像在出版社图书编辑经常要在书稿文档上写上各种版式记号，指明在何处应当用何种字体等等。因此也有人将 HTML 译为超文本排版语言。

ISO 早在 1986 年就已制定了一个标准 ISO 8879，即 SGML (Standard Generalized Markup Language)。这是一个描述标记语言的标准[W-SGML]。SGML 是一个非常复杂的、功能丰富的系统，有很多种选项，很适合于需要精确文档标准的大型组织。然而 SGML 的过分复杂使它很不适合于简单快捷的万维网出版。由于 HTML 非常易于掌握且实施简单，因此它很快就成为万维网的重要基础[RFC 1866]。官方的 HTML 标准由 W3C (即 WWW Consortium)负责制定。有关 HTML 的一些参考资料见[W-HTML1]至[W-HTML4]。现在最新的版本是 HTML 4.0。更新的版本正在研究之中。

HTML 定义了许多用于排版的命令，即“标签”(tag)[①]。例如，<I>表示后面开始用斜体字排版，而</I>则表示斜体字排版到此结束。HTML 就将各种标签嵌入到万维网的页面中。这样就构成了所谓的 HTML 文档。HTML 文档是一种可以用任何文本编辑器（例如，Windows 的记事本 Notepad）创建的 ASCII 码文件。但应注意，仅当 HTML 文档是以.html 或.htm 为后缀时，浏览器才对这样的 HTML 文档的各种标签进行解释。如果 HTML 文档改换以.txt 为其后缀，则 HTML 解释程序就不对标签进行解释，而浏览器只能看见原来的文本文件。

当浏览器从服务器读取某个页面的 HTML 文档后，就按照 HTML 文档中的各种标签，根据浏览器所使用的显示器的尺寸和分辨率大小，重新进行排版并恢复出所读取的页面。现有的一些字处理软件都不具有像 HTML 这样的功能。

目前已开发出了很好的制作万维网页面的软件工具，使我们能够像使用 Word 字处理器

---

① 注：在[MINGCI93]中，将 tag 和 flag 两个名词都译为“标志”，由于目前已有较多的作者将 tag 译为“标签”，并考虑到最好与 flag 的译名有所区别，故将 tag 译为标签。实际上“标签”的意思也还比较准确。因为一个 HTML 文档与浏览器所显示的内容相比，主要就是增加了许多的标签。

那样很方便地制作各种页面。然而学习一些 HTML 的基本概念仍是必要的。这是因为在对已有的万维网页面进行修改时，往往要查看其源代码，即查看其 HTML 文档。直接在 HTML 文档上对页面进行修改，有时是很必要的。

建议每一个读者都使用一下浏览器上的编辑器来编写一个很简单的页面。然后仔细观察浏览器显示的页面和相应的 HTML 文档的关系，同时也和编辑器上显示的内容进行对比。这样做可能是学习 HTML 最好的方法。

2．HTML 的格式与标签

元素(element)是 HTML 文档结构的基本组成部分。一个 HTML 文档本身就是一个元素。每个 HTML 文档由两个主要元素组成：**首部**(head)和**主体**(body)，主体紧接在首部的后面。首部包含文档的**标题**(title)，以及系统用来标识文档的一些其他信息。标题相当于文件名。用户可使用标题来搜索页面和管理文档。文档的主体是 HTML 文档的最主要的部分，文档所包含的主要信息都在主体中。当浏览器工作时，在浏览器的最上面的标题条显示出文档的标题，而在浏览器最大的主窗口中显示的就是文档的主体。

主体部分往往又由若干个更小的元素组成，如**段落**(paragraph)、**表格**(table)、和**列表**(list)等。

HTML 用一对标签（即一个开始标签和一个结束标签）或几对标签来标识一个元素。开始标签由一个小于字符“<”、一个标签名、和一个大于字符“>”组成。结束标签和开始标签的区别只是在小于字符的后面要加上一个斜杠字符“/”。虽然**标签名并不区分大写和小写**（例如，<TITLE>和<title>或<TitLE>是等效的），但习惯上大家都愿意用大写字符表示一个标签名。有一些标签可以将结束标签省略。

并非所有的浏览器都支持所有的 HTML 标签。若某一个浏览器不支持某一个 HTML 标签，则浏览器将忽略此标签，但在一对不能识别的标签之间的文本仍然会被显示出来。

下面是一个简单例子，用来说明 HTML 文档中标签的用法。

```
<HTML>                                  {HTML 文档开始}
<HEAD>                                  {首部开始}
    <TITLE>一个 HTML 的例子</TITLE>      {“一个 HTML 的例子”是标题}
</HEAD>                                 {首部结束}
<BODY>                                  {主体开始}
    <H1>HTML 很容易掌握</H1>             {“HTML 很容易掌握”是 1 级题头}
    <P>这是第一个段落。虽然很            {<P>和</P>之间的文字是一个段落}
    短，但它仍是一个段落。</P>           {实际上，标签</P>经常可以省略不用}
    <P>这是第二个段落。</P>              {<P>和</P>之间的文字是一个段落}
</BODY>                                 {主体结束}
</HTML>                                 {HTML 文档结束}
```

这里需要说明一下，当浏览器显示 HTML 文档时，连续的空格、回车和换行都被当成是一个空格。浏览器在显示文本时，会根据显示器的尺寸在适当的地方自动换行，而和 HTML 文档的一个段落里面的换行（如上面例子中在“虽然很”后面的换行）没有关系。

题头(heading)是在主体中的标题[1]，共分为6级，1级最高，6级最低。级别越高的题头所用的字也越大（编辑器自动设置题头字体的大小）。题头标签<Hn>中的n表示题头的级别。

在有的标签名后面还可加上属性，如：ALIGN=CENTER（居中），ALIGN=RIGHT（右对齐）。默认的属性是左对齐ALIGN=LEFT。

在HTML中有三个字符具有特殊的意义，即：

< 表示一个标签的开始；

> 表示一个标签的结束；

& 表示转义序列的开始。

因此，当这三个字符在文件中出现时，在HTML文档中就要将其转换为**转义序列**(escape sequence)。每个转义序列都以字符“&”开始，以分号“;”结束。这三个字符<, >, &所对应的转义序列分别为下面引号中的字符序列：“<”，“>”和“&”[2]。这样，在浏览器中就可以显示各种非ASCII码了。例如字符©是一个版权符号，在HTML文档中就用©来表示。顺便指出，在转义序列中的字符是区分大小写的。

表8-4给出一些常用的HTML标签以及简要的说明。

表8-4 一些常用的HTML标签

标 签	说 明
<HTML>...</HTML>	声明这是用HTML写成的万维网文档
<HEAD>...</HEAD>	定义页面的首部
<TTLE>...</TITLE>	定义页面的标题，此标题并不在浏览器的显示窗口中显示
<BODY>...</BODY>	定义页面的主体
<Hn>...</Hn>	定义一个n级题头
<B>...</B>	设置...为黑体字
<I>...</I>	设置...为斜体字
<UL>...</UL>	设置...为无序列表，列表中每一个项目前面出现一个圆点
<OL>...</OL>	设置...为编号列表
<MENU>...</MENU>	设置...为菜单
<LI>	开始一个列表项目，</LI>可不用
 	强制换行
<P>	一个段落开始，与下个段落空一行或缩进几个字符。</P>可不用。
<HR>	强制换行，同时画出一条水平线
<PRE>...</PRE>	设置...为已排版的文本，浏览器显示时不再进行排版
<IMG SRC="...">	插入一张图像，其文件名为...
<A HREF="...">X</A>	定义一个超链。超链的起点为X，终点为“...”

最前面的7个标签不需要再进行解释。标签<UL>表示**无序列表**(Unordered List)，在列表中的每一个项目都不编号，而是在项目前面出现一个圆点。标签<OL>则表示**编号列表**

(1) 注：英文title和heading本来都是标题的意思。但为了将这两个名词区分开，我们将前者译为“标题”，而将后者译为“题头”。

(2) 注：lt表示less than，gt表示greater than，而amp表示ampersand，即符号&。

(Ordered List)，列表中的项目都按顺序编号。无论是无序列表还是编号列表，都可以嵌套使用。标签<MENU>是使列表中的项目前面既没有圆点，也没有编号，因此在屏幕上更加简洁。若浏览器不支持<MENU>标签，则在每个项目前仍然使用圆点。再后面的 4 个标签意思都很清楚，不用解释。

当有的文本的版面已经过精心的排版而浏览器在进行显示时可能会改变版面的格式时，就要进行已排版文本的设置。标签<PRE>表示**已排版**(PREformatted)。

HTML 允许在万维网页面中插入图像。一个页面本身带有的图像称为**内含图像**(inline image)。标签<IMG>即表明在当前位置装入一个**内含图像**(IMaGe)，其**来源**(SouRCe)是“文件...”。HTML 标准并没有规定该图像的格式。实际上，大多数浏览器都支持 GIF 和 JPEG 文件。从理论上讲，用户可以设计一个浏览器，使它能够支持多种格式的图像文件。但按照这种浏览器创作出的万维网页面，很可能使别人的浏览器什么也看不见。这里的原因就是很多种格式的图像占据的存储空间太大，因而这种图像在因特网传送时就很浪费时间。例如，一幅**位图文件**(.bmp)可能要占用 500 KB～700 KB 的存储空间。但若将此图像改存为经压缩的.gif 格式，则可能只有十几个千字节，大大减少了存储空间。

在插入图像时，在标签<IMG>中还可使用一些参数。例如，参数 ALIGN 给图像**定位**，并将与图像一起出现的文字放在合适的地方（与图像的顶部、或中部、或底部对齐）。参数 HEIGHT 和 WIDTH 是指明图像装入时在屏幕上显示时的大小，一般用**像素**(pixel)的数目表示。如<IMG SRC=portrait.gif HEIGHT=100 WIDTH=65>表示装入一个文件名为 portrait.gif 的图像，其高度和宽度分别为 100 和 65 个像素。

HTML 还可插入**表格**(table)。这就要使用标签<TABLE>。与此标签配套使用的还有如：<CAPTION>（表格的标题），<TR>（表格的行），<TH>或<TD>（表格每格中应填入的数据）等等，这里不再详细介绍。

表 8-4 中最后一项是最重要的一个标签，我们在下一小节专门讨论关于超链的问题。

8.5.5 万维网页面中的超链

1．链接到其他网点上的页面

没有超链就没有万维网。在前面的图 8-11 中我们已经给出了链接的初步概念。现在我们要比较详细地介绍 HTML 关于超链的一些规定。

每个超链有一个**起点**和**终点**。超链的起点说明在万维网页面中的什么地方可引出一个超链。在一个页面中，超链的起点可以是一个字或几个字，或是一幅图，或是一段文字。在浏览器所显示的页面上，超链的起点是很容易识别的。对于以文字作为超链的起点时，这些文字往往用不同的颜色显示（例如，一般的文字用黑色字时，超链起点往往使用蓝色字），甚至还加上下划线（一般由浏览器来设置）。当我们将鼠标移动到一个超链的起点时，表示鼠标位置的箭头就变成了一只手。这时只要点击鼠标，这个超链就被激活。

定义一个超链的标签是<A>。字符 A 表示**锚**(Anchor)。建立一个超链时好像抛出一个锚，并将这个锚扎到超链的终点。

HTML 规定，在 HTML 文档中定义一个超链的语法是：

<A HREF="... ">X</A> (8-3)

这里一定要弄清链接的起点和终点。

链接的起点就是(8-3)式中的 X，它可以是一个或多个字符。

链接的终点则放在(8-3)式中的 HREF=" ... "的引号中。引号中的 ... 就是链接终点的统一资源定位符 URL。HREF 与前面的字符 A 之间应有一个空格。H 代表超文本，而 REF 代表 REFerence，是“访问”或“引用”的意思。

例如，我们有一个页面的文档中的某一行提到清华大学，但没有详细介绍。这时就可以将该行中“清华大学”这四个字作为一个链接的起点。在浏览时只要将鼠标的位置放在这个超链起点，点击一下，就可进入到清华大学的主页和了解清华大学的详细情况。有关这个链接的 HTML 文档就是：

<A HREF="http://www.tsinghua.edu.cn">清华大学</A>

如果这个超链的起点不是“清华大学”这四个字而是一幅清华大学的照片（假定此照片的文件名为 tsinghua.gif），则有关这个超链的 HTML 文档就应当是：

<A HREF="http://www.tsinghua.edu.cn"><IMG SRC="tsinghua.gif"></A>

2．链接到一个本地文件

上面的例子是超链的终点是其他网点上的页面。这种链接方式叫做**远程链接**。在许多情况下，超链可以指向自己的计算机中的某一个文件。这叫做**本地链接**。

在进行本地链接时，在 HREF=的后面不需要写很长的、完整的 URL（包括具有完整目录路径的文件名）。这是因为在使用 URL 时可进行许多的简化：

（1）当协议(http://）被省略时，就认为与当前页面的协议相同。

（2）当主机域名被省略时，就认为是当前的主机域名。

（3）当目录路径被省略时，就认为是当前目录（对于远程链接，就认为是主机的默认根目录）。

（4）当文件名被省略时，就认为是当前文件（对于远程链接，就认为是对方服务器上默认的文件名，通常是一个名为 index.html 的文件）。

使用这种简化的方法，在 HREF=的后面使用了**相对路径名**。相反，使用完整的 URL 找一个文件是使用**绝对路径名**。

使用相对路径名的好处不仅是可以少键入一些字符，而且也便于目录的改动。例如，在某一级目录下创建了许多可互相链接的文档。若需要改动一下目录结构并将此目录移至另一个目录下，那么以前创建的使用绝对路径名的链接全都要改动，否则就链接不上了（当下载许多相互链接的文档后，若改变其文档目录，就常常发生这种链接不上的情况）。但若使用相对路径名，则原来的链接关系可以不必去改动。

从一个页面也可将其他页面上的图像链接到自己的页面上，这种图像叫做外部图像(external image)。从链接的起点 X 链接到外部图像的相应的 HTML 语法和(8-3)式一样，只不过是将引号中的 ... 写为外部图像的文件名。例如，MyImage.gif。

HTML 还支持读取其他站点的声音或视像文件。这时应将(8-3)式中要读取的文件名写为后缀为.wav, .mpeg, 或.mov 的文件名即可。

3．链接到本文件中的某个地方

假定有个很长的文件（后缀为.html）可在浏览器中显示。当需要查找某些内容时，往往要利用窗口边上的滚动条在几千行的信息中反复来回查找。这显然很不方便。比较好的办法

是在文件的一开始放入一个详细目录。目录中的每一节都是一个超链的起点。只要用鼠标点击目录中某一节的超链起点，就能立即将所要找的那一节显示在屏幕的最上方。

这种情况和前面所讨论的不同之处就是链接的终点不同，因为现在链接的终点不是一个统一资源定位符 URL，而是一个文件中指明的特定地方（例如某个段落的开始行）。为了标识这个链接的终点，HTML 将这种链接的终点称为**命名锚**(named anchor)，因为链接的终点像被一个锚扎定一样。但在一个文件中可能会有很多的链接终点。为了区分这些链接终点，就必须给每一个链接终点命名。图 8-15 画出了远程链接和在本文件中链接的区别。

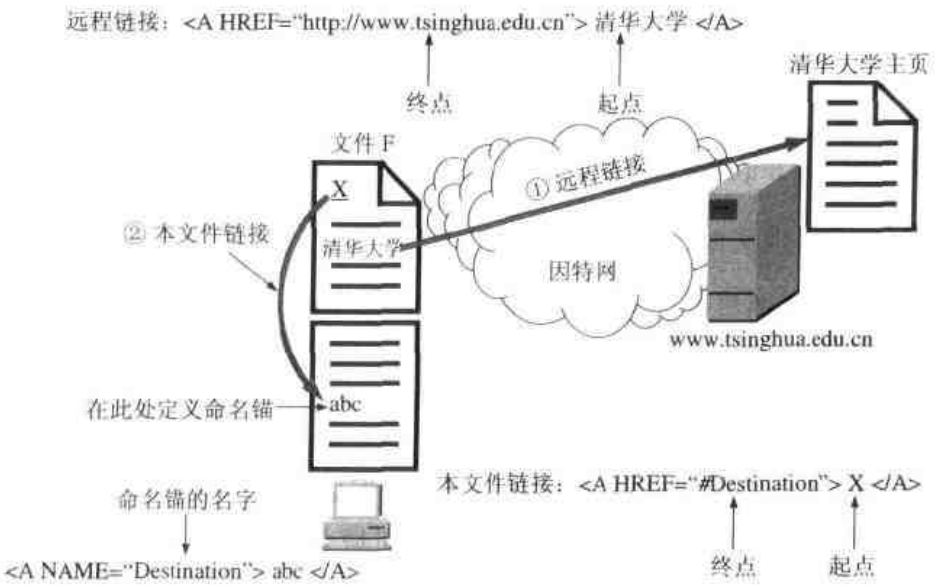


图 8-15　远程链接和本文件链接

从图 8-15 可看出，在设置远程链接时，由于远程文档的 URL 已经知道了，因此按照(8-3)式就可将链接设置好。但要设置在本文件内的链接时，就必须在要链接到的地方（这个地方在本文件中），先定义一个命名锚。HTML 规定一个命名锚的定义语法如下：

<A NAME="...">Y</A>　　（8-4）

这里的 Y 就是被指明为该链接终点的一个或多个字符，而 NAME=的后面引号中的 ... 写入我们给命名锚取的名字。例如，对于图 8-15 的例子中，**链接的终点**在字符 abc 的前面，而**命名锚的名字**取为 Destination，因此(8-4)式就变为：

<A NAME="Destination"> abc </A>　　（8-5）

HTML 规定链接到一个命名锚的 HTML 文档的语法是：

<A HREF="#...">X</A>　　（8-6）

(8-6)式中在字符 # 后面的省略号 ... 就是**命名锚的名字**，而 X 是**链接的起点**。例如，当上述链接的起点选为文档中的 X 时，在链接起点对应的 HTML 语句就是：

<A HREF="#Destination"> X </A>　　（8-7）

不难看出，(8-7)式和(8-5)式的作用是不同的。(8-7)式定义了一个本文件中的链接的起点和终点的命名锚，而(8-5)式是在链接的终点定义命名锚的名字。

命名锚也可插入到本地的其他 HTML 文件中（但在其他网点中的别人的文件中插入自己

设置的命名锚是不允许的的）。这时应在式(8-6)中的字符 # 前加上该文件的名字。

4．浏览器的结构

浏览器的结构要比服务器的结构复杂得多。服务器只是重复地执行一个简单的任务：等待浏览器打开一个连接，按照浏览器发来的请求向浏览器发送页面，关闭连接，并等待浏览器（也可能是另外的浏览器）的下一个请求。但浏览器却包含若干个大型软件组件，它们协同在一起工作。图 8-16 是一个浏览器的主要组成部分。

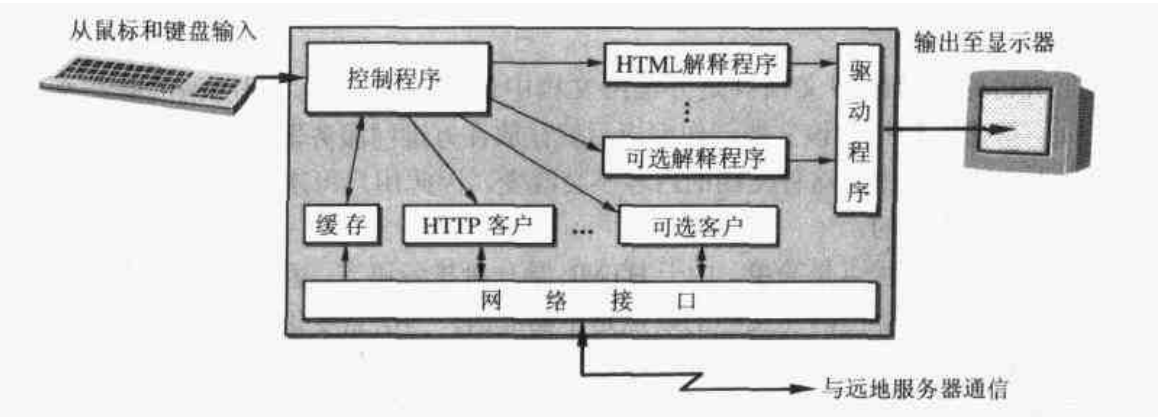


图 8-16 浏览器的主要组成部分

从图 8-16 可看出，一个浏览器有一组客户、一组解释程序，以及管理这些客户和解释程序的控制程序。控制程序是其中的核心部件，它解释鼠标的点击和键盘的输入，并调用有关的组件来执行用户指定的操作。例如，当用户用鼠标点击一个超链的起点时，控制程序就调用一个客户从所需文档所在的远地服务器上取回该文档，并调用解释程序向用户显示该文档。

HTML 解释程序是必不可少的，而其他的解释程序则是可选的。HTML 解释程序的输入就是符合 HTML 语法的文档。解释程序将 HTML 规格转换为适合用户显示硬件的命令来处理版面的细节。例如，当遇到一个强制换行标签
，解释程序就输出一个新的行。

HTML 解释程序对页面中所有的可选项（即所有超链的起点）都保存有其位置信息。当用户的鼠标点击某个选项时，浏览器就根据当前光标位置和存储的位置信息来决定哪个选项被用户选中。

前面已经讲过，浏览器的任务不仅是浏览。许多浏览器还包含一个 FTP 客户，用来获取文件传送服务。一些浏览器也包含一个电子邮件客户，使浏览器能够发送和接收电子邮件。现在的浏览器都设计得很好，它使用户看不见许多细节，用户也并不知道他执行了一个可选客户，如 FTP 客户或 SMTP 客户。

在浏览器中还设有一个缓存。浏览器将它取回的每一个页面副本都放入本地磁盘的缓存中。当用户用鼠标点击某个选项时，浏览器首先检查磁盘的缓存。若缓存中保存了该项，那么浏览器就直接从缓存中得到该项副本而不必从网络来获取。在这种情况下就可明显地改善浏览器的运行特性。对于网络连接较为缓慢的用户，这种缓存就显得更加重要。因为从网络上取回一个很大的文件所需的时间将大大超过从本地磁盘直接读取的时间。

然而使用缓存也带来了一些问题。首先，缓存要占用磁盘大量的空间。其次，浏览器性能的改善只有在用户再次查看缓存中的页面时才有帮助。实际上，用户在进行浏览时，一般

会及时将有保存价值的页面存储下来（只需点击几下鼠标即可）。因此缓存中保存的大部分今后不再查看的文件并不会改善浏览器的性能。相反，由于浏览器要耗费时间来将这些文件白白地存储在磁盘上，这反而降低了浏览器的效率。

为了改善浏览器的特性，许多浏览器允许用户调整缓存策略。例如，用户可设置缓存的时间限制，并在此时间限制到期后在缓存中删除这些文件。

8.5.6 动态万维网文档与 CGI 技术

1．动态文档的概念

上面所讨论的万维网文档只是万维网文档中最基本的一种，即所谓的**静态文档**(static document)。静态文档是指该文档创作完毕后就存放在万维网服务器中，在被用户浏览的过程中，内容不会改变。由于这种文档的内容不会改变，因此用户每次对静态文档的读取所得到的返回结果都是相同的。

静态文档的最大优点是简单。由于 HTML 是一种排版语言，因此静态文档可以由不懂程序设计的人员来创建。但静态文档的缺点是不够灵活。当信息变化时就要由文档的作者手工对文档进行修改。可见对变化频繁的文档不适于作成静态文档。

动态文档(dynamic document)是指文档的内容是在浏览器访问万维网服务器时才由应用程序动态创建。当浏览器请求到达时，万维网服务器要运行另一个应用程序，并将控制转移到此应用程序。接着，该应用程序对浏览器发来的数据进行处理，并输出 HTTP 格式的文档，万维网服务器将应用程序的输出作为对浏览器的响应。由于对浏览器每次请求的响应都是临时生成的，因此用户通过动态文档所看到的内容可根据需要不断变化。可见动态文档的主要优点是具有报告当前最新信息的能力。例如，动态文档可用来报告股市行情、天气预报或民航售票情况等内容。但动态文档的创建难度比静态文档的高，因为动态文档的开发不是直接编写文档本身，而是编写用于生成文档的应用程序，这就要求动态文档的开发人员必须会编程，而所编写的程序还要通过大范围的测试，以保证输入的有效性。

动态文档和静态文档之间的主要差别体现在服务器一端。这主要是**文档内容的生成方法不同**。而从浏览器的角度看，这两种文档并没有区别。动态文档和静态文档的内容都遵循 HTML 所规定的格式，，浏览器仅根据在屏幕上看到的内容并无法判定服务器送来的是哪一种文档，只有文档的开发者才知道。

从以上所述可以看出，要实现动态文档就必须在以下两个方面对万维网服务器的功能进行扩充：

（1）应增加另一个应用程序，用来处理浏览器发来的数据，并创建动态文档。

（2）应增加一个机制，用来使万维网服务器将浏览器发来的数据传送给这个应用程序，然后万维网服务器能够解释这个应用程序的输出，并向浏览器返回 HTML 文档。

图 8-17 是扩充了功能的万维网服务器的示意图。这里增加了一个机制，叫做**通用网关接口** CGI (Common Gateway Interface)。CGI 是一种标准，它定义了动态文档应如何创建，输入数据应如何提供给应用程序，以及输出结果应如何使用。

在万维网服务器中新增加的应用程序叫做 CGI 程序。取这个名字的原因是：万维网服务器与 CGI 的通信是遵循着 CGI 标准。“通用”是因为这个标准所定义的规则对其他任何语言都是通用的。“网关”二字的出现是因为 CGI 程序还可能要访问其他的服务器资源，如数据

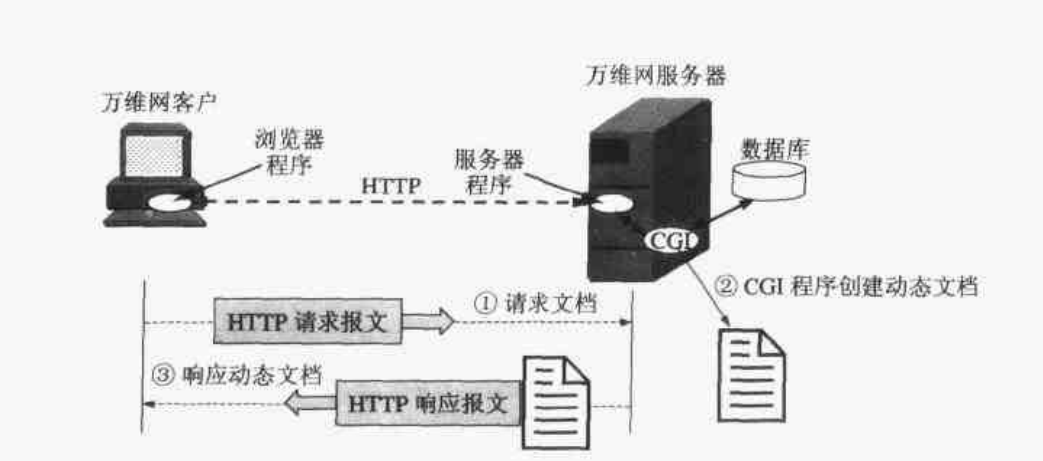


图 8-17　扩充了功能的万维网服务器

库或图形软件包，因而 CGI 程序的作用有点像一个网关。也有人将 CGI 程序简称为网关程序。“接口”是因为有一些已定义好的变量和调用等可供其他 CGI 程序使用。请读者注意：在看到 CGI 这个名词时，应弄清是指 CGI 标准，还是指 CGI 程序。

CGI 程序的正式名字是 CGI **脚本**(script)。按照计算机科学的一般概念，“**脚本**”[①]指的是一个程序，它被另一个程序（解释程序）而不是计算机的处理机来解释或执行。有一些语言专门作为**脚本语言**(script language)，如 Perl, REXX（在 IBM 主机上使用），JavaScipt 以及 Tcl/Tk 等。脚本也可用一些常用的编程语言写出，如 C，C++等。使用脚本语言可更容易地和更快地进行编码，这对一些有限功能的小程序是很合适的。但一个脚本运行起来要比一般的编译程序要慢，因为它的每一条指令先要被另一个程序来处理（这就要一些附加的指令），而不是直接被指令处理器来处理。脚本不一定是一个独立的程序，它可以是一个动态装入的库，甚至是服务器的一个子程序。

CGI 程序又称为 cgi-bin 脚本，这是因为在许多万维网服务器上，为便于找到 CGI 程序，就将 CGI 程序放在/cgi-bin 的目录下。

2．表单

从 HTML 2.0 开始就增加了“表单”项目。“**表单**”(form)用来将用户数据从浏览器传递给万维网服务器。在创建动态文档时，表单和 CGI 程序经常配合使用。表单在浏览器的屏幕出现时，就有一些方框和按钮，可供用户选择和点取。有的方框可让用户输入数据。

下面我们通过一个简单的例子说明图 8-18(a)中的表单是怎样构成的。在图 8-18(a)中用户已输入了数据。图 8-18(b)是表单对应的 HTML 文档。

HTML 定义一个表单是在 HTML 文档的主体中插入表单的标签<FORM>和</FORM>。在这两个标签中间的一些标签就是用来指明此表单中所包含的的一些项目。

在<FORM>标签中首先要指明一个 ACTION 参数。ACTION 参数后面的引号中指出在万维网服务器中的 CGI 程序在何处。一般就是指明一个 URL 或 URI。本例中是将表单发给域名为 web.com 的一个主机上的名为 form 的一个 CGI 程序，它在主机的/cgi-bin 目录下。后面的一个参数是 METHOD（这两个参数的顺序并不重要），说明对表单所采用的方法，即数据

① 注：**脚本**(script)一词还有其他的意思。例如在多媒体开发程序中使用“脚本”来表示编程人员输入的一系列指令，这些指令指明多媒体文件应按什么顺序执行。



(a)一个表单

```
<BODY>
 <H3>情况调查表</H3>
 <FORM ACTION="http://web.com/cgi-bin/form" METHOD=POST><P>
  你使用Internet的水平：
   初学<INPUT NAME="level"TYPE=RADIO VALUE="low">
    一般<INPUT NAME="level"TYPE=RADIO VALUE="middle">
   熟练<INPUT NAME="level"TYPE=RADIO VALUE="high"><P>
  已掌握的内容：
   <LI><INPUT NAME="http"TYPE=CHECKBOX>HTTP</LI>
   <LI><INPUT NAME="html"TYPE=CHECKBOX>HTML</LI>
   <LI><INPUT NAME= java"TYPE=CHECKBOX>JAVA</LI><P>
  其他情况：<BR>
   <TEXTAREA NAME="others"ROWS='5'COLS='60"WRAP='virtual"></TEXTAREA><P>
  你的电子邮件地址：<INPUT TYPE=TEXT NAME='email' SIZE=40><P>
   <INPUT TYPE=SUBMIT VALUE="填完按此按钮">
   <INPUT TYPE=RESET VALUE="清除所填信息"><P>
 </FORM>
</BODY>
```

(b)表单对应的 HTML 文档

图 8-18　一个简单的例子

是如何在浏览器和服务器之间传送的。现在最常用的方法是 POST 和 GET。

本例选择的方法是 POST。当使用 POST 方法时，表单中所有的变量及其值都按一定的规律放在报文的主体中。而使用 GET 方法时，则将这些变量及其值放在 URL 的后面，中间用一个问号“?”隔开。

由于表单的格式是早已确定的，因此从浏览器向 CGI 程序就只需发送用户输入的数据。表单中用户要输入数据的地方都称为变量，并都必须取个名字，而用户所输入的数据就是该变量的值。这样，浏览器向 CGI 程序发送的数据格式为：

name1=variable1&name2=variable2&name3=variable3&...

其中 name*i* 为第 *i* 个变量，variable*i* 是第 *i* 个变量的值。字符 & 是个特殊的字符，用来将各个字段（即变量及其值）分隔开。当变量值中有空格字符出现时，则用字符 + 代替。

当变量值中有标点符号等特殊字符出现时，则用 %nn 表示。这里 nn 是该特殊字符在 ASCII 码中的十六进制序号。例如，空格字符也可用 %20 表示，字符 @ 用 %40 表示，等号 = 用 %3d 表示。

在 HTML 文档中用标签<INPUT>表示需要用户输入数据的项目。在图 8-18(a)中的“初学”、“一般”和“熟练”这几个字的后面都各有一个小圆圈，从<INPUT>标签中的 NAME="..." 可看出它们的名字都是 level，而类型 TYPE 是 RADIO（即无线电按钮），它又称为**单选按钮**（无线电的频道按钮有很多个，但每次只能按下一个）。所以用户可在这三个小圆圈中任意选中一个（但只能一个）。现在用户选取的是“熟练”，因此这部分的数据是 level=high。

接着有三个小方框，它们的类型都是 CHECKBOX（检查框），它又称为**复选框**，即可选取的小方框的数目没有限制，因此每一个检查框都有自己的名字。现在用户选取了前两个。凡选中的小方框的变量值都是 on。

再后面有一个文本区 TEXTAREA（文本区不用 INPUT），取名为 others。创建表单的人已经规定了文本区的大小是 5 行 60 列，并且有一个参数 WRAP="virtual"。这个参数的意思是当用户在文本区输入数据时能自动换行。

再往后是一个文本框，取名为 e-mail，其长度为 40 个字符。

最后是两个按钮。第一个按钮的类型是 SUBMIT（提交），取名为“填完按此按钮”。这个名字也就是在屏幕上看见的按钮上的字。当用户点取此按钮时，浏览器即向服务器发送填写的数据。另一个按钮的类型是 RESET（复位），取名为“清除所填信息”。用户若认为所填写的数据不合适，在按此按钮后，表单即恢复到刚开始时的样子。

这样，在用户点取提交按钮后，从浏览器向服务器发送的报文的主体部分就是：

level=high&http=on&html=on&others=&e-mail=xyz%40public1.ptt.js.cn

用户没有选择的复选框，以及没有写入数据的文本区，在发送的数据中都将其忽略。若用户在一个表单中填写了很多数据，那么最后发送的数据将是很长的一行。CGI 程序负责解释这一长串的数据。CGI 程序调用一些子程序即可将每一个字段分隔开，并将其中的特殊字符恢复成原来的 ASCII 码。

在表单中还可加入下拉式菜单。这时要使用<SELECT>标签。菜单中的每一项要用一个<OPTION>标签指明，此标签中的 NAME="..."指明了该菜单上出现的名字。

3. CGI 标准

1998 年 5 月，CGI 的标准化工作才进入到因特网草案阶段[W-CGI]，形成了 CGI 1.1 版本。然而要成为正式 CGI 标准仍须一定的时间。

当 CGI 程序被调用时，服务器就将一些参数传递给 CGI 程序，参数的值可由浏览器提供。使用参数的好处是可以用一个 CGI 程序就能够产生许多只有细节不同的动态文档。由于最初设计的 CGI 是在运行 UNIX 操作系统的服务器上使用的，因此在服务器将这些参数传递给 CGI 程序时，不是使用一般的命令行(command-line)方式，而是将这些参数信息置于 UNIX 的**环境变量**(environment variable)中，然后调用 CGI 程序。CGI 程序从环境变量中将值提取出来。例如，服务器指派 URL 的后缀（即 URL 中在问号字符？以后的字符序列）为环境变量 QUERY_STRING 的值。

CGI 规定了 18 个环境变量，这些环境变量在 CGI 的因特网草案中使用的名词是**元变量**(meta-variable)。因为有的操作系统可能不提供类似于 UNIX“环境变量”的机制，因此，标

准草案中采用了“元变量”这一与系统无关的名词。这就使得 CGI 的具体实现者不受限制，可以根据所使用操作系统的具体情况，采用他们认为最方便的方法向 CGI 程序传递参数。

对所有的请求都要使用的元变量有：

- SERVER_SOFTWARE　指明回答请求的服务器软件的名字和版本。
- SERVER_NAME　指明运行服务器的计算机域名或 IP 地址。
- GATEWAY_INTERFACE　指明服务器运行的 CGI 软件版本。

还有一些元变量是根据请求才使用的。如：

- QUERY_STRING　在 URL 中的问号?以后的部分所带的信息。
- CONTENT_LENGTH　指明浏览器发送过来的字符串共有多少个字节。
- REMOTE_ADDR　发出请求的浏览器所驻留的计算机的 IP 地址。
- SCRIPT_NAME　在 URL 中服务器域名后面的路径。

当 CGI 程序要将数据返回给浏览器时，必须将这些数据写到其标准输出。CGI 程序输出的报文的首部的第一行必须是：

```
Content-type: text/html
```

而第二行必须是一个空行，从第三行起才是 HTML 文档。服务器在确认 CGI 程序发来的报文首部正确无误后，就加上 HTTP 有关首部信息，将 HTML 文档返回给浏览器。

为了说明这点，下面给出用 C 语言编写的一个 CGI 程序的小例子。

```
#include <stdio.h>

main () {
    printf ("Content-type: text/html\n\n") ;
    printf ("<html>") ;
    printf ("Thank you!") ;
    printf ("</html>") ; }
```

在程序运行后，得到的输出是：

```
Content-type: text/html

<html>Thank you!</html>
```

服务器收到 CGI 程序的输出后，在这个输出的前面加上必要的 HTTP 首部，就向刚才发起请求的浏览器发送如下的信息：

```
HTTP/1.0 200 OK
SERVER: SERVERNAME/1.0
MIME-version: 1.0
Content-type: text/html
```

```
<html>Thank you!</html>
```

浏览器收到的响应报文的主体就是<html>Thank you!</html>这一行信息，而最后在屏幕上显示的就是 Thank you!这几个字符。

每当浏览器的请求到达时，服务器就调用 CGI 程序。由于服务器不能保留浏览器请求的历史数据，所以服务器无法告诉 CGI 程序某个用户是否曾经发出过请求。但历史数据对 CGI 程序进行交互对话是很有用的。因为当一个用户再次发出请求时，历史数据可避免让用户重复地回答问题。

CGI 程序在两次调用之间所存储的信息叫做**状态信息**(state information)。状态信息若用于浏览器的多次调用，则应将这种信息长期存储在磁盘文件中。但状态信息若仅用于浏览器的一次运行中，则可将这类短期状态信息放在其 URL 中。下面看两个例子。

【例 1】 在本地磁盘文件存放长期状态信息。

CGI 程序可在本地文件中保存浏览器 IP 地址的列表。当浏览器调用 CGI 程序时，CGI 程序可从元变量 REMOTE_ADDR 中获得浏览器所在的计算机的 IP 地址，然后检索存放浏览器 IP 地址的本地文件。若找不到此 IP 地址，则将此浏览器的 IP 地址附加在该文件的后面。CGI 程序可根据浏览器是首次还是再次访问，在其输出文档中反映出该浏览器的这一状态。

【例 2】 在 URL 中存放短期状态信息。

假定我们希望在第一次访问某网点时，屏幕上出现如下的两行信息：

这是您的首次访问。

点击此处可刷新页面。

上面的第二行信息是一个超链的起点，使用不同的颜色来显示。只要点击此超链起点，屏幕内容就被刷新。屏幕刷新后上面这两行信息中的第一行应改变。屏幕上应显示：

您已经刷新了页面 1 次。

点击此处可刷新页面。

以后每点击一次超链的起点，页面就刷新一次，而显示的刷新了页面的次数就自动加 1。

我们假定服务器的域名是 web.com，而 CGI 程序在主机的目录/cgi-bin 下的文件 form 中。要解决这个问题，可在 CGI 程序中设一个变量 n，并编写一小段程序。当 CGI 程序被调用时，就将元变量 QUERY_STRING 的值赋给变量 n。CGI 程序使用了下面的语句进行判断：

- 若变量 n 为空值(null)，则 CGI 程序将变量 n 赋值为 1，并执行语句(8-8)至(8-10)。这里我们仍使用 C 语言的写法。

```
printf("这是您的首次访问。<BR>")                                    (8-8)
printf("<A HREF=\"http://web.com/cgi-bin/form?", n, "\">")  (8-9)
printf("点击此处可刷新页面。</A>")                                  (8-10)
```

请注意：在语句(8-9)中有一个变量 n。另外，这里将<HTML>和</HTML>都省略了。

- 若变量 n 为整数值，则 CGI 程序先执行语句(8-11)。请注意其中的变量 n。

```
printf ("您已经刷新了页面", n, "次。<BR>")                          (8-11)
```

接着，CGI 程序将变量 n 赋值为 n+1，然后仍执行语句(8-9)和(8-10)。

我们不难看到，当用户第一次访问该网点时，其 URL 为 http://web.com/cgi-bin/form，因此变量 n 为空值。这时 CGI 程序将变量 n 置为 1，因而输出的 HTML 文档为：

这是您的首次访问。

<A HREF="http://web.com/cgi-bin/form?1">

点击此处可刷新页面。</A>

这样，当用户点击超链起点“点击此处可刷新页面。”时，实际上就将 URL 的后缀置成 1。这里的 1 就是元变量 QUERY_STRING 的当前值。CGI 程序再次执行时，先将 1 赋给变量 n，使式(8-11)对应的 HTML 文档成为：

您已经刷新了页面 1 次。

接着，CGI 程序将变量 n 加 1 后，输出与式(8-9)和式(8-10)相对应的 HTML 文档：

<A HREF="http://web.com/cgi-bin/form?2">

点击此处可刷新页面。</A>

读者应注意，这种产生动态文档的 CGI 程序只是根据元变量的值得出新的文档。如果一个用户故意手工键入以下的 URL：

http://web.com/cgi-bin/form?98765

那么 CGI 程序会（不正确地）报告页面已经被刷新了 98765 次。

8.5.7 活动万维网文档

随着 HTTP 和万维网浏览器的发展，上一节所述的动态文档已明显地不能满足发展的需要。虽然使用表单可以获得一些双向的交互，但这种交互功能是很有限的。这是因为，动态文档一旦建立，它所包含的信息内容也就固定下来而无法及时刷新屏幕。另外，像动画之类的显示效果，动态文档也无法提供。

1．活动文档的创建

有两种技术可用于浏览器屏幕显示的连续更新。一种技术称为**服务器推送**(server push)，这种技术是将所有的工作都交给服务器。服务器不断地运行与动态文档相关联的应用程序，定期更新信息，并发送更新过的文档。

尽管从用户的角度看，这样做是达到了连续更新的目的，但这有很大的缺点。

首先，为了满足很多客户的请求，服务器就要运行很多的服务器推送程序。这将造成过多的服务器开销。其次，服务器推送技术要求服务器为每一个浏览器客户维持一个不释放的 TCP 连接。随着 TCP 连接的数目增加，每一个连接所能分配到的网络带宽就下降，这就导致网络传输时延的增大。

另一种提供屏幕连续更新的技术是**活动文档**(active document)技术。这种技术是将所有的工作都转移给浏览器端。每当浏览器请求一个活动文档时，服务器就返回一段程序副本，使该程序副本在浏览器端运行。这时，活动文档程序可与用户直接交互，并可连续地改变屏幕的显示。只要用户运行活动文档程序，活动文档的内容就可以连续地改变。由于活动文档技术不需要服务器的连续更新传送，对网络带宽的要求也不会太高。

从传送的角度看，浏览器和服务器都把活动文档看成是静态文档。在服务器上的活动文档的内容是不变的，这点和动态文档是不同的。浏览器可在本地缓存一份活动文档的副本。活动文档还可处理成压缩形式，便于存储和传送。另一点要注意的是，活动文档本身并不包括其运行所需的全部软件，大部分的支持软件是事先存放在浏览器中。图 8-19 说明了活动文档的创建过程。

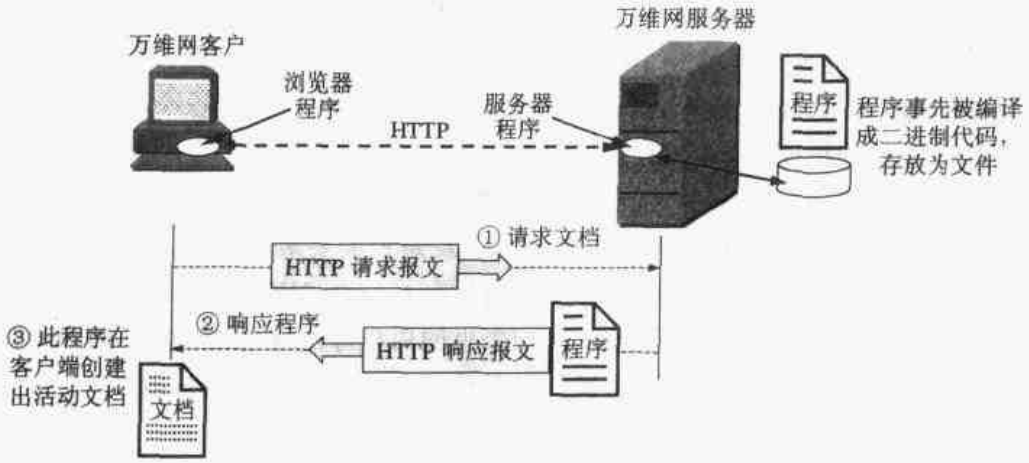


图 8-19　活动文档由服务器发送过来的程序在客户端创建

2．用 Java 技术创建活动文档

由美国 Sun 公司开发的 Java 语言是一项用于创建和运行活动文档的技术。在 Java 技术中使用了一个新的名词“小应用程序”(applet)[①]来描述活动文档程序。当用户从万维网服务器下载一个嵌入了 Java 小应用程序的 HTML 文档后，用户可在浏览器的显示屏幕上点击某个图像，然后就可看到动画的效果，或是在某个下拉式菜单中点击某个项目，然后就可看到根据用户键入的数据所得到的计算结果。实际上，Java 技术是活动文档技术的一部分。

Java 技术共有三个主要组成部分：

（1）**程序设计语言**。Java 包含一个新的程序设计语言，它既可用来编写传统的计算机程序，也可用来编写 Java 小应用程序。

（2）**运行(runtime)环境**。Java 系统还定义了一个运行 Java 程序所必须的运行环境，其中包括一个 Java 虚拟机（简称为 JVM），该软件定义了 Java 二进制代码的执行模型。

（3）**类库(class library)**。为了更容易编写 Java 小应用程序，Java 提供了强大的类库支持。

Java 是一种面向对象的高级语言，编程方便直观。Java 是从 C++派生出来的，它省略了 C++很多复杂的、很少用的语言特点。但 Java 和 C 或 C++并不兼容。Java 的每一个数据项都有一个确定的类型。对数据的操作严格按照该数据的类型来进行。

Java 的编译程序将源程序转换成 Java **字节码**(bytecode)，这是一种与机器无关的二进制代码。计算机程序调用**解释程序**(interpreter)读取字节码，并解释执行。

Java 语言、字节码以及 Java 运行系统都被设计成与计算机硬件无关。一旦形成了字节码，就可在任何计算机上运行，并都产生相同的输出。为什么要使 Java 小应用程序与具体的机器无关呢？这是因为，首先，在因特网上用户使用的计算机种类繁多。Java 小应用程序与机器无关可使在任何计算机上运行的浏览器程序能够下载并运行活动文档。其次，这样做可保证活动文档在所有的浏览器上产生同样的正确输出。第三，可大大地降低活动文档的创建和测试费用，因为不必为每一种计算机都制作一个副本。

Java 运行环境包括一些设施，可允许小应用程序操纵用户的显示，而 Java 类库则包含提

① 注：在 Java 语言出现之前就已经有了 applet 这一名词。小应用程序 applet 通常被嵌入在操作系统或一个较大的应用程序之中。在万维网技术中，此名词常常指的是 Java 小应用程序。

供高级图形接口的软件。这些合在一起，就称为**抽象窗口工具箱** AWT (Abstract Window Toolkit)执行[②]。为什么需要这样的一个 AWT 呢？有这样两个原因。首先，使用小应用程序主要是为了复杂的显示。只要静态显示不能满足要求时就要使用小应用程序。其次，一个控制图形显示的程序还必须指明许多的细节。例如，若要显示一个图形，程序就必须决定：是将此图形显示在已有的窗口内，还是另创建一个新的窗口。若创建新的窗口，那么程序就要指明窗口的题头、大小、颜色、窗口所放的位置，以及是否需要滚动条等。

由于 Java 小应用程序可能需要和静态文档进行交互，在 AWT 中还包括执行常规的万维网浏览器操作的类。例如，给定一个 URL，一个小应用程序能够使用 AWT 的类来读取和显示一个 HTML 文档，读取和显示一个图像，以及读取和播放一个声音片段。

3．Java 解释程序

在前面的图 8-16 中我们曾讲到，一个浏览器可以包含一个或多个解释程序。最早的浏览器只有一个 HTML 解释程序，用来显示静态或动态文档。但运行 Java 的浏览器则需要两个解释程序，即 HTML 解释程序和 Java 小应用程序解释程序。

解释程序是一个复杂的程序，其核心是一个模仿计算机的简单循环。解释程序维持一个指令指针，在初始化时指在小应用程序的开始处。在每一次循环操作时，解释程序在指令指针指向的地址读取字节码。然后解释程序对字节码进行解码，并完成指明的操作。例如，如果解释程序找到字节码 add 和两个整数型操作数，就将这两个整数相加，然后更新指令指针，继续下一个循环。

解释程序除了应具备基本的指令解码功能，还必须包括对 Java 运行环境的支持。也就是说，一个 Java 解释程序必须能够在屏幕上显示图形，接入到因特网，以及执行 I/O 操作等。此外，解释程序必须设计成使得小应用程序能够利用浏览器的设施来读取和显示静态和动态文档。因此，在浏览器中的 Java 解释程序必须能够与浏览器中的 HTTP 客户以及 HTML 解释程序进行通信。

8.5.8 万维网上的信息检索系统

1．搜索引擎的工作原理

万维网是一个大规模的、联机式的信息储藏所。那么，应当采用什么方法才能找到所需的信息呢？如果已经知道存放该信息的网点，那么只要在浏览器的 Location 框内键入该网点的 URL 和回车键，就可进入该网点。但是，若不知道要找的信息在何网点，那就要使用万维网的检索系统。在万维网中用来进行搜索的程序叫做**搜索引擎**(search engine)。有时还使用其他一些名词，如 spider, crawler, worms, robot 或 knowbots (knowledge robot)。

万维网可看成是一个很大的图。每一个页面是图上的一个**结点**，而链接这些页面的超链则是一些**弧**（有时也称为**边**）。能够访问到所有的结点的算法早已有了，但现在的难点是：结点的数量非常之多，并且整个万维网的情况处在不断变化之中。

要在万维网上进行检索，就要将所有万维网页面标题中的关键词作成索引。下面介绍制

② 注：AWT 所代表的英文还有：Alternative Window Toolkit, Advanced Window Toolkit, 和 Applet Widge Toolkit 等。

作索引的算法。我们需要三种数据结构。首先，我们需要一个很大的线性数组 url_table，用于存放已经找到的 URL。url_table 包括几百万个项，应当做到每一个页面都有一项。这个线性数组 url_table 要使用虚存，将不太常用的项存放在磁盘中。每个项有两个指针。其中一个指向页面的 URL，而另一个则指向页面的标题。页面的 URL 和标题的字符串长度都是可变的，因此存放它们要使用的数据结构是**堆(heap)**。“堆”是虚存中的一个很大的非结构化的数据块，页面的 URL 和标题的字符串可不断地追加到堆的后面。堆是我们使用的第二个数据结构。

由于 URL 的数量太大，查找起来很不方便。因此我们采用了第三种数据结构，即**散列表(hash table)**[1]，其长度为 n。任何一个 URL 经过散列函数散列后都产生一个小于 n 的非负整数。所有散列函数值相同的 URL（不妨设散列值为 k），都链接到以散列码 k 为标识的一个链表中。后面将要讨论的算法可以保证：将某个 URL 放入 url_table 的同时也将它放入散列表。而散列表的主要用途就是可以迅速确定一个 URL 是否已经在 url_table 中。这三种数据结构见图 8-20 所示。

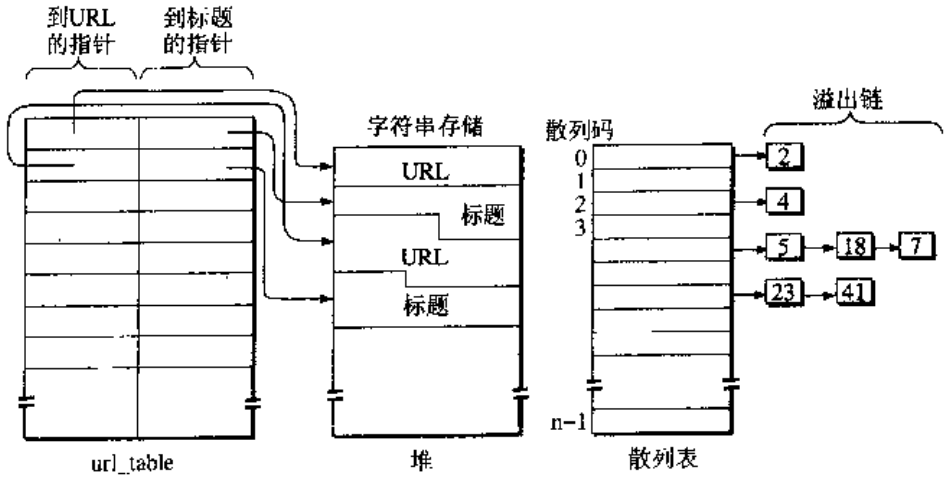


图 8-20　一个简单搜索引擎中使用的数据结构

要制作索引就要首先进行搜索。搜索引擎的核心是一个递归过程 process_url，它将一个 URL 字符串作为其输入。过程 process_url 首先散列 URL，看它是否已经在 url_table 中。如在，就继续散列下一个 URL。每一个 URL 只被处理一次。

若该 URL 不在 url_table 中，则读取该页面。然后将该页面的 URL 和标题复制到堆中，并将指向这两个字符串的指针存入 url_table。与此同时，URL 也要进入散列表。

接着，过程 process_url 从该页面抽取所有的超链。对每一个超链调用一次 process_url，并将超链的 URL 作为 process_url 的输入。

在运行搜索引擎时，我们是从某一个 URL 开始来调用过程 process_url。当过程运行完毕时，所有从该 URL 可到达的页面都已进入了 url_table，而这个搜索阶段就结束了。

上述算法的缺点就是它先从深度上搜索。这样就会进入一种循环状态（或许在搜索过几千个超链之后）。比较好些的算法是先收集每一个页面上的所有超链，将已经处理过的除外，其余的都留下。接着就从广度上进行搜索。这就是说，先搜索该页面上的所有超链，再搜索

[1] 注：有关散列表的内容可参考有关数据结构的教科书。

这些超链所指向的所有页面上的所有超链。

搜索阶段结束后，就进入第二阶段，将关键词制成索引。制作索引的过程就是将 url_table 中的项目逐项地进行处理。对于 url_table 中的每一个项目，检查其标题，并将所有不在“非用词表”(stop list)中的词挑选出来。所谓非用词表就是这些词不应当制作在索引中（非用词表包含了如前置词、连接词、冠词等以及一些没有什么价值的词）。每选择一个词，就在索引中写一行，包括这个词，以及现在对应的 url_table 中的项目。当整个的 url_table 都已扫描过，索引就制成了，因此可按词进行检索。索引存放在磁盘上。

用户使用浏览器进行检索时，是在一个表格的输入框中键入一个或多个关键词，然后点击“提交”按钮。这个动作形成了一个 POST 请求。服务器收到请求后，调用 CGI 程序，读取用户键入的关键词，并在索引中查找这些关键词，最后对应于每一个关键词可找出相应的一组 URL。若用户使用了布尔运算符 AND 或 OR，那么 CGI 程序可给出经过布尔运算后的检索结果。

CGI 程序根据用户提出的关键词，检索出用户所需的所有标题及其 URL。然后形成一个表格，作为对 POST 请求的响应，返回给浏览器。用户可根据浏览器所显示的页面，继续点击他所感兴趣的超链。

然而一个实用的搜索引擎必须解决好以下的一些问题：

（1）某些 URL 已经变成是陈旧的。例如，所指向的页面已不复存在。这时，服务器会返回一个差错码。

（2）某些主机可能暂时不可达（例如，主机出故障）。这时 TCP 连接将迟迟不能建立。要避免过长时间的等候，应设置一个超时时间。超时时间设得太短，可能使有效的 URL 不能找到。但超时时间设得太长，整个的搜索会明显地缓慢下来。

（3）并非从某一个页面开始即可访问到所有的页面。因此在一开始时，最好从一个尽可能大的 URL 集合开始。

（4）一些文档并不能够制成索引，因为它们包括了很多图像和声音。但制作索引只能对文本类型的文档，这时可使用 HEAD 方法将其 MIME 首部取回。对 text 以外的类型不再进行搜索。

（5）大约有 20%的万维网页面没有标题，或无有用的标题（例如，“某某的网页”）。在这种情况下，只能从整个的超文本中来抽取关键词。可以根据每一个词（除去在非用词表中的）出现的频度来挑选关键词，例如，选择出现频度最高的 10 至 20 个。

（6）搜索引擎的内存或磁盘空间可能不够用。整个搜索过程也可能太长。解决这个问题可采用另一种完全不同的策略来进行搜索。这就是每一个服务器只搜索本服务器上的网页，形成一个本地索引。然后由一个中心网点将其他服务器上的本地索引收集在一起，形成一个上索引。这样就可大大减少占用的存储空间、CPU 时间，以及网络的带宽。但让其他服务器运行别人的程序有时会遭到拒绝。

2. 一些著名的搜索引擎

在万维网上已经有了好几个著名的搜索引擎。它们的共同特点是索引量很大，而且还可采用归类信息方式，非常便于查找。当前最有名的搜索引擎是：

Google(http://www.google.com)

Yahoo(http://www.yahoo.com)

由于各搜索引擎收集的站点专业分类方法和搜索速度都不同，有时仅使用一、二个搜索引擎还是难于找到所需的信息。因此现在有一种**智能元搜索引擎**(Smart Meta Search Engine)能够同时对若干个搜索引擎进行搜索，例如：

Mamma(http://www.mamma.com)

MetaCrawler(http://www.metacrawler)

All4one(http://www.all4one.com）

在因特网上搜索信息需要经验的积累。要多实践才能掌握从因特网获取信息的技巧。

8.6 引导程序协议 BOOTP 与动态主机配置协议 DHCP

8.6.1 引导程序协议 BOOTP

为了将软件协议做成通用的和便于移植，协议软件的编写者不会将所有的细节都固定在源代码中。相反，他们将协议软件参数化。这就使得在很多台计算机上使用同一个经过编译的二进制代码成为可能。一台计算机和另一台计算机的许多区别，都可以通过一些不同的参数来体现。在软件协议运行之前，必须给每一个参数赋值。

在协议软件中给这些参数赋值的动作叫做**协议配置**。一个软件协议在使用之前必须是已正确配置的。具体的配置信息有哪些则取决于协议栈。例如，连接到因特网的计算机的协议软件需要配置的项目包括：

（1）IP 地址。

（2）子网掩码。

（3）默认路由器的 IP 地址。

（4）域名服务器的 IP 地址。

这些信息通常存储在一个配置文件中，计算机在引导过程中可以对这个文件进行存取。但是，对于一个无盘工作站或一个有盘的计算机在第一次引导时应如何处理呢？

在无盘工作站的情况下，操作系统和连网软件可以存储在只读存储器(ROM)中。但是制造厂家并不知道 IP 地址等信息，这些信息取决于该机器所连接到的网络。因此这些信息不能存储在 ROM 中。

还有一种情况，就是计算机可能经常改变在网络上的位置（尤其是便携式计算机的大量使用，有时在家中上网，有时在实验室上网），用人工进行协议配置既不方便，又容易出错。因此，需要采用自动协议配置的方法。

引导程序协议 BOOTP (BOOTstrap Protocol)也称为**自举协议**。BOOTP 使用 UDP 来使一个无盘工作站自动获取配置信息。BOOTP 目前还是因特网的草案标准，最早的文档是 1985 年的[RFC 951]，在 1993 年的更新版本是[RFC 1542]。

BOOTP 使用客户服务器工作方式。为了获取配置信息，协议软件广播一个 BOOTP 请求报文。这时，BOOTP 报文作为 UDP 用户数据报的数据，UDP 用户数据报再作为 IP 数据报的数据。收到请求报文的 BOOTP 服务器查找发出请求的计算机的各项配置信息，将配置信息放入一个 BOOTP 回答报文中，并将回答报文返回给提出请求的计算机。这样，一台计算机就获得了所需的配置信息。

由于计算机发送 BOOTP 请求报文时自己还没有 IP 地址，因此它使用全 1 广播地址（只

在本网络上广播）作为目的地址，而用全 0 地址作为源地址。这时，BOOTP 服务器可使用广播方式将回答报文返回给该计算机，或使用收到广播帧上的硬件地址进行单播。

以前已经讲过，可使用不同的协议来获取 TCP/IP 协议配置信息的不同部分。如用 RARP 可获取 IP 地址，而用 ICMP 可获取子网掩码。但 BOOTP 功能要强些，可只发送一个广播报文就可获取所需的全部配置信息。

8.6.2 动态主机配置协议 DHCP

然而 BOOTP 并没有彻底解决配置问题，因为 BOOTP 是一个**静态配置协议**而不是**动态配置协议**。当 BOOTP 服务器收到来自某个主机的请求时，就在其信息库中查找该主机的已经确定的地址绑定信息。但当一个主机移动到另一个网络或需要一个临时的 IP 地址时，BOOTP 就无能为力了，除非管理员手工修改数据库中的信息。

动态主机配置协议 DHCP (Dynamic Host Configuration Protocol)比 BOOTP 更进了一步。它提供了一种机制，称为**即插即用连网**(plug-and-play networking)。这种机制允许一台计算机加入新的网络和获取 IP 地址而不用手工参与。实际上，DHCP 并不是一个新的协议而是扩展了的 BOOTP。DHCP 与 BOOTP 是向后兼容的，并且它们所使用的报文格式都很相似。DHCP 最新的 RFC 文档是 1997 年的[RFC 2131, 2132]，目前还是因特网草案标准。

DHCP 对运行客户软件和服务器软件的计算机都适用。当运行客户软件的计算机移至一个新的网络时，就可使用 DHCP 获取其配置信息而不需要手工干预。DHCP 给运行服务器软件而位置固定的计算机指派一个永久地址，而当这计算机重新启动时其地址不改变。

DHCP 使用客户服务器方式，需要 IP 地址的主机在启动时就向 DHCP 服务器广播发送**发现报文**(DHCPDISCOVER)（将目的 IP 地址置为全 1，即 255.255.255.255），这时该主机就成为 DHCP 客户。发送广播报文是因为现在还不知道 DHCP 服务器在什么地方，因此还要发现(DISCOVER)DHCP 服务器的 IP 地址。这个主机目前还没有自己的 IP 地址，因此它将 IP 数据报的源 IP 地址设为全 0。这样，在本地网络上的所有主机都能够收到这个广播报文，但只有 DHCP 服务器才对此广播报文进行回答。DHCP 服务器先在其数据库中查找该计算机的配置信息。若找到，则返回找到的信息。若找不到，则从服务器的 IP 地址池(address pool)中取一个地址分配给该计算机。DHCP 服务器的回答报文叫做**提供报文**(DHCPOFFER)，表示“提供”了 IP 地址等配置信息。

但是我们并不愿意在每一个网络上都设置一个 DHCP 服务器，因为这样会使 DHCP 服务器的数量太多。因此现在是使每一个网络至少有一个 DHCP **中继代理**(relay agent)（见图 8-21），它配置了 DHCP 服务器的 IP 地址信息。当 DHCP 中继代理收到主机 A 以**广播**形式发送的发现报文后，就以**单播**方式向 DHCP 服务器转发此报文，并等待其回答。收到 DHCP 服务器回答的提供报文后，DHCP 中继代理再将此提供报文发回给主机 A。需要注意的是，图 8-21 是



图 8-21 DHCP 中继代理以单播方式转发发现报文

个示意图。实际上，DHCP 报文只是 UDP 用户数据报的数据，它还要加上 UDP 首部、IP 数据报首部，以及以太网的 MAC 帧的首部和尾部后，才能在链路上传送。

DHCP 服务器分配给 DHCP 客户的 IP 地址的临时的，因此 DHCP 客户只能在一段有限的时间内使用这个分配到的 IP 地址。DHCP 协议称这段时间为**租用期**(lease period)，但并没有具体规定租用期应取为多长或至少为多长，这个数值应由 DHCP 服务器自己决定。例如，一个校园网的 DHCP 服务器可将租用期设定为 1 小时。DHCP 服务器在给 DHCP 发送的提供报文的选项中给出租用期的数值。按照[RFC 1533]的规定，租用期用 4 字节的二进制数字表示，单位是秒。因此可供选择的租用期范围从 1 秒到 136 年。DHCP 客户也可在自己发送的报文中（例如，发现报文）提出对租用期的要求。

DHCP 的详细工作过程见图 8-22 所示。DHCP 客户使用的 UDP 端口是 68，而 DHCP 服务器使用的 UDP 端口是 67（和 BOOTP 是一样的）。这两个 UDP 端口都是熟知端口。

下面按照图 8-22 中的注释编号（①至⑨）对其工作过程进行简单的解释。

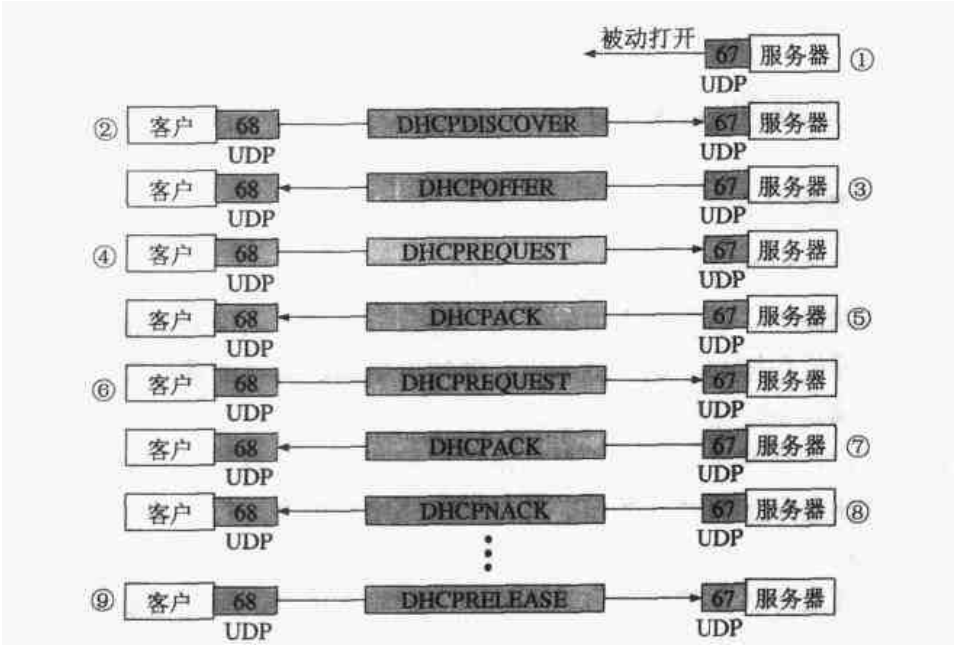


图 8-22　DHCP 协议的工作过程

① DHCP 服务器被动打开 UDP 端口 67，等待客户端发来的报文。

② DHCP 客户从 UDP 端口 68 发送 DHCP 发现报文。

③ 凡收到 DHCP 发现报文的 DHCP 服务器都发出 DHCP 提供报文，因此 DHCP 客户可能收到多个 DHCP 提供报文。

④ DHCP 客户从几个 DHCP 服务器中选择其中的一个，并向所选择的 DHCP 服务器发送 DHCP 请求报文。

⑤ 被选择的 DHCP 服务器发送确认报文 DHCPACK。从这时起，DHCP 客户就可以使用这个 IP 地址了。这种状态叫做**已绑定状态**，因为在 DHCP 客户端的 IP 地址和硬件地址已经完成绑定，并且可以开始使用得到的临时 IP 地址了。

DHCP 客户现在要根据服务器提供的租用期 T 设置两个计时器 T1 和 T2，它们的超时时间分别是 0.5T 和 0.875T。当超时时间到就要请求更新租用期。

⑥ 租用期过了一半（T1 时间到），DHCP 发送请求报文 DHCPREQUEST 要求更新租用期。

⑦ DHCP 服务器若同意，则发回确认报文 DHCPACK。DHCP 客户得到了新的租用期，重新设置计时器。

⑧ DHCP 服务器若不同意，则发回否认报文 DHCPNACK。这时 DHCP 客户必须立即停止使用原来的 IP 地址，而必须重新申请 IP 地址（回到步骤②）。

若 DHCP 服务器不响应步骤⑥的请求报文 DHCPREQUEST，则在过了租用期的 87.5% 时，DHCP 客户必须重新发送请求报文 DHCPREQUEST（重复步骤⑥），然后又继续后面的步骤。

⑨ DHCP 客户可随时提前终止服务器所提供的租用期，这时只需向 DHCP 服务器发送释放报文 DHCPRELEASE 即可。

DHCP 很适合于经常移动位置的计算机。当计算机使用 Windows 操作系统时，若点击控制面板的网络图标就可以添加 TCP/IP 协议。然后点击“属性”按钮，在“IP 地址”这一项下面有两种方法可供选择：一种是“自动获得一个 IP 地址”，另一种是“指定 IP 地址”。若选择前一种，就表示是使用 DHCP 协议。

8.7 简单网络管理协议 SNMP

8.7.1 网络管理的基本概念

虽然网络管理还没有精确定义，但它的内容可归纳为：

网络管理包括对硬件、软件和人力的使用、综合与协调，以便对网络资源进行监视、测试、配置、分析、评价和控制，这样就能以合理的价格满足网络的使用需求，如实时运行性能，服务质量等。网络管理常简称为网管。

我们可以看到，网络管理并不是指对网络进行行政上的管理。

网络是一个非常复杂的分布式系统。这是因为网络有很多运行着多种协议的结点，而这些结点还在相互通信和交换信息。网络的状态总是不断地变化着。可见我们必须使用网络来管理网络。我们需要有一种协议来读取这些结点上的状态信息，有时还要将一些新的状态信息写入到这些结点上。

下面简单介绍网络管理模型中的主要构件（见图 8-23）。

管理站是整个网络管理系统的核心，它通常是个有良好图形界面的高性能的工作站，并由网络管理员直接操作和控制。所有向被管设备发送的命令都是从管理站发出的。管理站也常称为**网络运行中心** NOC (Network Operations Center)。在管理站中的关键构件是**管理程序**（如图 8-23 中有字母 M 的椭圆形图标所示）。管理程序在运行时就成为**管理进程**。管理站（硬件）或管理程序（软件）都可称为**管理者**(manager)，所以这里的 manager 不是指人而是指机器或软件。网络管理员(administrator)才是指人。大型网络往往实行多级管理，因而有多个管理者，而一个管理者一般只管理本地网络的设备。

在网络中有很多的**被管设备**（包括设备中的软件）。被管设备可以是主机、路由器、打印机、集线器、网桥或调制解调器等。在每一个被管设备中可能有许多**被管对象**(Managed Object)。被管对象可以是被管设备中的某个硬件（例如，一块网络接口卡），也可以是某些硬件或软件（例如，路由选择协议）的配置参数的集合。被管设备有时可称为**网络元素或网元**。在被管设备中也会有一些不能被管的对象。

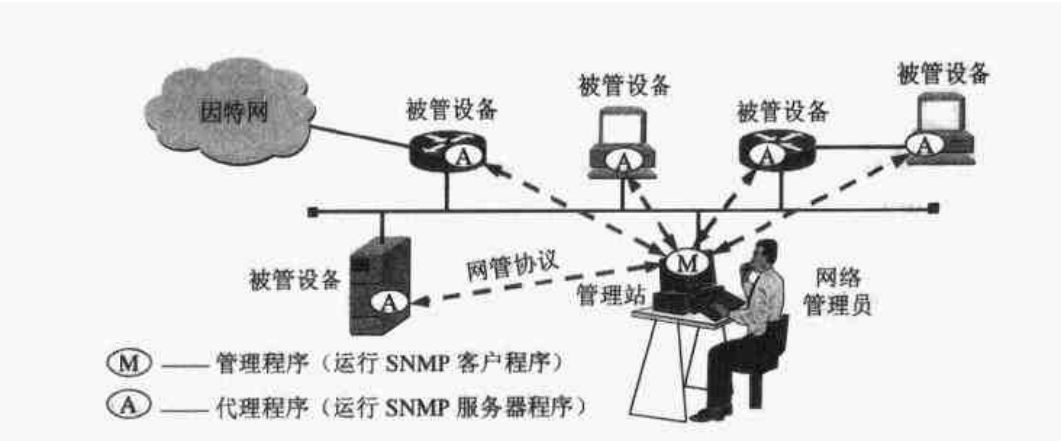


图 8-23　网络管理的一般模型

被管对象必须维持可供管理程序读写的若干控制和状态信息。这些信息总称为**管理信息库 MIB (Management Information Base)**，而管理程序就使用 MIB 中这些信息的值对网络进行管理（如读取或重新设置这些值）。

在每一个被管设备中都要运行一个程序以便和管理站中的管理程序进行通信。这些运行着的程序叫做**网络管理代理程序**，简称为**代理**(agent)（如图 8-23 中有字母 A 的几个椭圆形图标所示）。代理程序在管理程序的命令和控制下在被管设备上采取本地的行动。

在图 8-23 中还有一个重要构件就是**网络管理协议**，简称为**网管协议**。需要注意，并不是网管协议本身来管理网络。网管协议就是管理程序和代理程序之间进行通信的规则。网络管理员利用网管协议通过管理站对网络中的被管设备进行管理。

管理程序和代理程序按客户服务器方式工作。管理程序运行 SNMP **客户程序**，向某个代理程序发出请求（或命令），代理程序运行 SNMP **服务器程序**，返回响应（或执行某个动作）。在网管系统中往往是一个（或少数几个）客户程序与很多的服务器程序进行交互。

OSI 很早就在其总体标准中提出了网络管理标准的框架，即 ISO 7498-4。ITU-T 在网络管理方面紧密地和 ISO 合作，制订了与 ISO 7498-4 相对应的 X.700 系列建议书。在 OSI 网络管理标准中，将网络管理分为**系统管理**（管理整个 OSI 系统）、**层管理**（只管理某一个层次）和**层操作**（只对一个层次中管理通信的一个实例进行管理）。在系统管理中，提出了**管理的五个功能域**：

（1）**故障管理**(Fault Management)　对网络中被管对象故障的检测、定位和排除。故障并非一般的差错，而是指网络已无法正常运行，或出现了过多的差错。网络中的每一个设备都必须有一个预先设定好的故障门限（此门限必须能够调整），以便确定是否出了故障。

（2）**配置管理**(Configuration Management)　用来定义、识别、初始化、监控网络中的被管对象，改变被管对象的操作特性，报告被管对象状态的变化。

（3）**计费管理**(Accounting Management)　记录用户使用网络资源的情况并核收费用，同时也统计网络的利用率。

（4）**性能管理**(Performance Management)　以网络性能为准则，保证在使用最少网络资源和具有最小时延的前提下，网络能提供可靠、连续的通信能力。

（5）**安全管理**(Security Management)　保证网络不被非法使用。

这五个管理功能域简称为 FCAPS，基本上覆盖了整个网络管理的范围。

8.7.2　简单网络管理协议 SNMP 概述

关于网络管理有一个基本原理，这就是：

若要管理某个对象，就必然会给该对象添加一些软件或硬件，但这种“添加”必须对原有对象的影响尽量小些。

简单网络管理协议 SNMP (Simple Network Management Protocol)正是按照这样的基本原理来设计的[STAL96]。

SNMP 发布于 1988 年。次年就有 70 个以上的厂家（包括 IBM, HP, Sun 等著名公司）宣布支持 SNMP。OSI 虽然已制订出许多的网络管理标准，但当时（到现在也很少）却没有符合 OSI 网管标准的产品，在这种情况下，IETF 在 1990 年制订出的网管标准 SNMP [RFC 1155, 1157]就变成了因特网的正式标准。由于以后又有了新的版本 SNMPv2 和 SNMPv3，因此原来的 SNMP 又称为 SNMPv1。

SNMP 最重要的指导思想就是**要尽可能简单**。SNMP 的基本功能包括监视网络性能、检测分析网络差错和配置网络设备等。在网络正常工作时，SNMP 可实现统计、配置、和测试等功能。当网络出故障时，可实现各种差错检测和恢复功能。虽然 SNMP 是在 TCP/IP 基础上的网络管理协议，但也可扩展到其他类型的网络设备上。

图 8-24 是使用 SNMP 的典型配置。整个系统必须有一个管理站。管理进程和代理进程利用 SNMP 报文进行通信，而 SNMP 报文又使用 UDP 来传送。图中有两个主机和一个路由器。这些协议栈中带有阴影的部分是原来这些主机和路由器所具有的，而没有阴影的部分则是为实现网络管理而增加的。

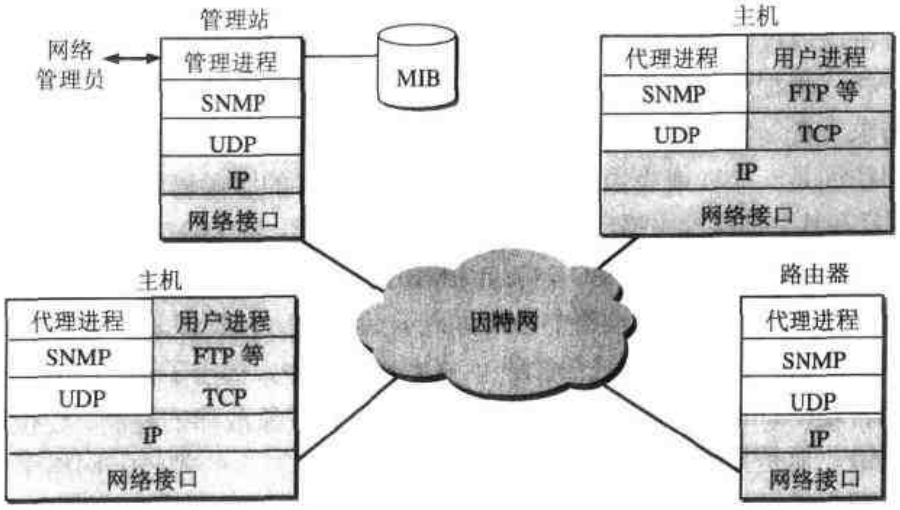


图 8-24 SNMP 的配置

若网络元素使用的不是 SNMP 而是另一种网络管理协议，SNMP 协议就无法控制该网络元素。这时可使用**委托代理**(proxy agent)。委托代理能提供如协议转换和过滤操作等功能对被管对象进行管理。

SNMP 的网络管理由三个部分组成，即管理信息库 MIB，管理信息结构 SMI 以及 SNMP 本身。下面先介绍管理信息库。

8.7.3 管理信息库 MIB

管理信息库 MIB 是一个网络中所有可能的被管对象的集合的数据结构[RFC 1212]。只有在 MIB 中的对象才是 SNMP 所能够管理的。例如，路由器应当维持各网络接口的状态、入分组和出分组的流量、丢弃的分组和有差错的报文的统计信息，而调制解调器则应当维持发

送和接收的字符数、波特率和接受的呼叫等统计信息。那么在 MIB 中就必须有上面这样一些信息。SNMP 的管理信息库采用和域名系统 DNS 相似的树形结构，它的根在最上面，根没有名字。图 8-25 画的只是管理信息库的一部分，它又称为**对象命名树**(object naming tree)。

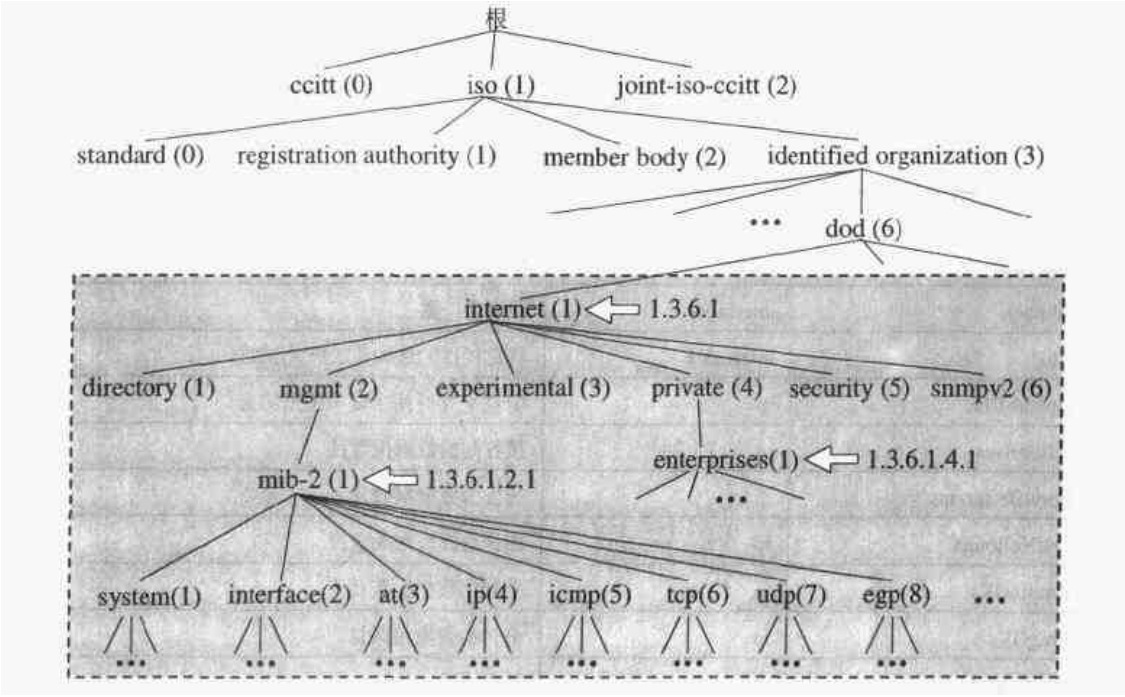


图 8-25　管理信息库的对象命名树举例

对象命名树的顶级对象有三个，都是世界上著名的标准制订单位，即 ISO，CCITT（现在已是 ITU-T）和这两个组织的联合体。在 ISO 的下面有 4 个结点，其中的一个（标号为 3）是被标志的组织。在其下面有一个美国国防部 dod (Department of Defense)的子树（标号为 6），再下面就是 internet（标号为 1）。在只讨论 internet 中的对象时，可只画出 internet 以下的子树（图中带阴影的虚线方框），并在 internet 结点旁边标注上{1.3.6.1}即可。

在 internet 结点下面的第二个结点是 mgmt（管理），标号为 2。再下面是管理信息库 mib。1991 年定义了新版本 MIB-II [RFC 1213]，故该结点名现改用 mib-2，其**对象标识符**(Object Identifier)为{1.3.6.1.2.1}，或{internet(1) .2.1}。

表 8-5 给出了结点 mib-2 所包含的信息类别举例，

表 8-5　结点 mib-2 所包含的信息类别举例

类　别	标　号	所包含的信息
system	(1)	主机或路由器的操作系统
interfaces	(2)	各种网络接口
address translation	(3)	地址转换（例如，ARP 映射）
ip	(4)	IP 软件
icmp	(5)	ICMP 软件
tcp	(6)	TCP 软件
udp	(7)	UDP 软件
egp	(8)	EGP 软件

应当指出，MIB 的定义与具体的网络管理协议无关，这对于厂商和用户都有利。厂商可以在产品（如路由器）中包含 SNMP 代理软件，并保证在定义新的 MIB 项目后该软件仍遵守标准。用户可使用同一网络管理客户软件来管理具有不同版本的 MIB 的多个路由器。当然，一个没有新的 MIB 项目的路由器不能提供这些项目的信息。

图 8-25 所示的对象命名树的大小并没有限制。下面给出若干 MIB 变量的例子（见表 8-6），以便更好地理解 MIB 的意义。这里的“变量”是指一个特定对象的一个实例。

表 8-6 MIB 变量的例子

MIB 变量	所属类别	意义
sysUpTime	system	距上次重启动的时间
ifNumber	interfaces	网络接口数
ifMtu	interfaces	特定接口的最大传送单元 MTU
ipDefaultTTL	ip	IP 在生存时间字段中使用的值
ipInReceives	ip	接收到的数据报数目
ipForwDatagrams	ip	转发的数据报数目
ipOutNoRoutes	ip	路由选择失败的数目
ipReasmOKs	ip	重装的数据报数目
ipFragOKs	ip	分片的数据报数目
ipRoutingTable	ip	IP 路由表
icmpInEchos	icmp	收到的 ICMP 回送请求数目
tcpRtoMin	tcp	TCP 允许的最小重传时间
tcpMaxConn	tcp	允许的最大 TCP 连接数目
tcpInSegs	tcp	已收到的 TCP 报文段数目
udpInDatagrams	udp	已收到的 UDP 数据报数目

上面列举的大多数项目的值可用一个整数来表示。但 MIB 也定义了更复杂的结构。例如，MIB 变量 ipRoutingTable 则定义一个完整的路由表。还有其他一些 MIB 变量定义了路由表项目的内容，并允许网络管理协议访问路由器中的单个项目，包括前缀、地址掩码以及下一跳地址等。当然，MIB 变量只给出了每个数据项的逻辑定义，而一个路由器使用的内部数据结构可能与 MIB 的定义不同。当一个查询到达路由器时，路由器上的代理软件负责 MIB 变量和路由器用于存储信息的数据结构之间的映射。

值得注意的是，MIB 中的对象{1.3.6.1.4.1}，即 enterprises（企业），其所属结点数已超过 3000。例如，IBM 为{1.3.6.1.4.1.2}，Cisco 为{1.3.6.1.4.1.9}，Novell 为{1.3.6.1.4.1.23}等。世界上任何一个公司、学校只要用电子邮件发往 iana-mib@isi.edu 进行申请即可获得一个结点名。这样，各厂家就可以定义自己的产品的被管对象名，使它能用 SNMP 进行管理。因此，从理论上讲，全世界所有的连接到因特网的设备都可以纳入到 MIB 的数据结构中。

8.7.4 SNMPv1 的五种协议数据单元

SNMPv1 规定了五种协议数据单元 PDU（也就是 SNMP 报文），用来在管理进程和代理之间的交换。实际上，SNMP 的操作只有两种基本的管理功能，即：

（1）“读”操作，用 get 报文来检测各被管对象的状况；

（2）“写”操作，用 set 报文来改变各被管对象的状况。

SNMP 的这些功能通过探询操作来实现，即 SNMP 管理进程定时向被管设备周期性地发送探询信息。上述时间间隔可通过 SNMP 的管理信息库 MIB 来建立。探询的好处是：第一，可使系统相对简单。第二，能限制通过网络所产生的管理信息的通信量。但探询管理协议不够灵活，而且所能管理的设备数目不能太多。探询系统的开销也较大。如探询频繁而并未得到有用的报告，则通信线路和计算机的 CPU 周期就被浪费了。

但 SNMP 不是完全的探询协议，它允许不经过询问就能发送某些信息。这种信息称为**陷阱(trap)**，表示它能够捕捉“事件”。但这种陷阱信息的参数是受限制的。

当被管对象的代理检测到有事件发生时，就检查其门限值。代理只向管理进程报告达到某些门限值的事件（这就叫做**过滤**）。这种方法的好处是：第一，仅在严重事件发生时才发送陷阱；第二，陷阱信息很简单且所需字节数很少。

总之，使用探询（至少是周期性地）以维持对网络资源的实时监视，同时也采用陷阱机制报告特殊事件，使得 SNMP 成为一种有效的网络管理协议。

SNMP 使用无连接的 UDP，因此在网络上传送 SNMP 报文的开销较小。但 UDP 是不保证可靠交付的。这里还要指出，SNMP 使用 UDP 的方法有些特殊。在运行代理程序的服务器端用熟知端口 161 来接收 get 或 set 报文和发送响应报文（与熟知端口通信的客户端使用临时端口），但运行管理程序的客户端则使用熟知端口 162 来接收来自各代理的 trap 报文。

SNMPv1 共定义了表 8-7 所列的 5 种类型的协议数据单元：

表 8-7　SNMPv1 定义的协议数据单元类型

PDU 编号	PDU 名称	用　途
0	get-request	用来查询一个或多个变量的值
1	get-next-request	允许在一个 MIB 树上检索下一个变量，此操作可反复进行
2	get-reponse	对 get/set 报文作出响应，并提供差错码、差错状态等信息
3	set-request	对一个或多个变量的值进行设置
4	trap	向管理进程报告代理中发生的事件

图 8-26 画出了 SNMPv1 的报文格式。可以看出，一个 SNMP 报文由三个部分组成，即**版本**、**共同体**(community)和 **SNMP PDU**。版本字段写入“版本号减 1”。对于 SNMPv1 则应写入 0。共同体字段就是一个字符串，作为管理进程和代理进程之间的明文口令，常用的是 6 个字符“public”。SNMP PDU 由三个部分组成，即 PDU 类型（写入表 8-7 所示的 PDU 编号）、get/set 首部或 trap 首部，以及变量绑定(variable-bindings)（变量绑定指明一个或多个变量的名和对应的值。在 get 或 get-next 报文中，变量的值应忽略）。

get/set 首部具有如下的字段：

（1）**请求标识符**(request ID)　由管理进程设置的一个整数值。代理进程在发送 get-response 报文时也要返回此请求标识符。管理进程可同时向许多代理发出 get 报文，这些报文都使用 UDP 传送，先发送的有可能后到达。设置了请求标识符可使管理进程能够识别返回的响应报文对应于哪一个请求报文。

（2）**差错状态**(error status)　由代理进程回答时填入 0～5 中的一个数字（见表 8-8 的描述）。

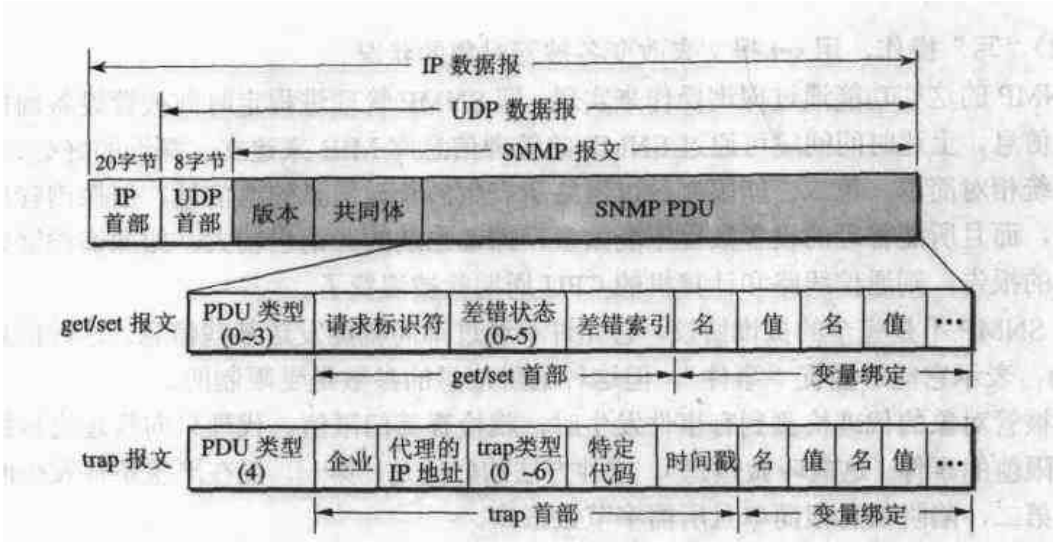


图 8-26　SNMP 的报文格式

表 8-8　差错状态描述

差错状态	名　字	说　明
0	noError	一切正常
1	tooBig	代理无法将回答装入到一个 SNMP 报文之中
2	noSuchName	操作指明了一个不存在的变量
3	badValue	一个 set 操作指明了一个无效值或无效语法
4	readOnly	管理进程试图修改一个只读变量
5	genErr	某些其他的差错

（3）**差错索引**(error index)　当出现 noSuchName, badValue, 或 readOnly 的差错时，由代理进程在回答时设置的一个整数，它指明有差错的变量在变量列表中的偏移。

trap 首部的字段如下：

（1）**企业**(enterprise)　填入产生陷阱报文的网络设备的对象标识符。此对象标识符肯定是在图 8-25 的对象命名树上的 enterprises 结点{1.3.6.1.4.1}下面的一棵子树上。

（2）**陷阱类型**　此字段正式的名称是 generic-trap，共分为表 8-9 所列的 7 种（类型 0~6）：

表 8-9　陷阱类型描述

陷阱类型	名　字	说　明
0	ColdStart	代理进行了初始化
1	warmStart	代理进行了重新初始化
2	linkDown	一个接口从工作状态变为故障状态
3	linkUp	一个接口从故障状态变为工作状态
4	authenticationFailure	从 SNMP 管理进程接收到具有一个无效共同体的报文
5	egpNeighborLoss	一个 EGP 相邻路由器变为故障状态
6	enterpriseSpecific	代理自定义的事件，需要用后面的“特定代码”来指明

当使用上述类型 2, 3 和 5 时，在报文后面变量部分的第一个变量应标识相应的接口。

（3）**特定代码**(specific-code)　指明代理自定义的事件（若陷阱类型为 6），否则为 0。

（4）**时间戳**(timestamp)　指明自代理进程初始化到陷阱报告的事件发生所经历的时间，

单位为 10 ms。例如，时间戳为 1908 表明在代理进程初始化后 1908 ms 发生了该事件。

8.7.5 管理信息结构 SMI

管理信息结构 SMI (Structure of Management Information)是 SNMP 的另一个重要组成部分[RFC 1155]。SMI 标准指明了所有的 MIB 变量必须使用**抽象语法记法** 1 (ASN.1)来定义。ASN.1 有两个主要特点：一个是人们阅读的文档中使用的记法，另一个是同一信息在通信协议中使用的紧凑编码表示。这种记法使得数据的含义不存在任何可能的二义性。例如，使用 ASN.1 的协议设计者不能简单地说“一个具有整数值的变量”，而必须说明该变量的准确格式和整数取值的范围。当网络中的计算机对数据项并不都使用相同的表示时，采用这种精确的记法就尤其重要。下面结合 SNMP 对 ASN.1 进行简单的介绍。

1．局部语法、传送语法与抽象语法

我们知道，任何数据都具有两种重要的属性，即**值**(value)与**类型**(type)。这里“值”是某个值集合中的一个元素，而“类型”则是值集合的名字。如果给定一种类型，则这种类型的一个值就是该类型的一个具体实例。在计算机中引入数据类型的概念可使程序设计更加方便。然而这只是从比较高的层次上所看到的情况。

从较低的层次上看，任何类型的数据最终都被表示成计算机中的比特串。一个比特串本身并不能说明自己表示的是什么类型的数据。一个比特串所代表的意义是由该比特串的程序设计者确定的。计算机硬件体系结构的不同也会带来对比特串解释上的差异。例如，VAX 系列计算机和 Intel 80x86 系列计算机，对浮点数的表示就采用了不同的格式。

上述这种对比特串的不同解释，是由于它们所使用的“**语法**”不同。“**语法**”实际上就是“**符号串解释方法**”。

在计算机网络中互相通信的计算机常常采用不同的“**局部语法**”(local syntax)。局部语法的差异决定了同一数据对象在不同的计算机上被表示为不同的符号串。当一个应用程序在多个计算机上实现时，往往需要把一个数据对象从一台计算机传送到另一台计算机。为了保证程序语义的正确性，必然要对比特串的格式进行变换，把符合发送方局部语法的比特串转换成为符合接收方局部语法的比特串。这就是**语法转换**。这可由发送方或接收方完成，也可由双方协作完成。由于局部语法的不同所带来的问题在其他层次同样存在。在应用层引入 ASN.1 这种形式语言，主要是因为应用层需要用到非常多种的数据类型。

当格式转换工作由双方中的一方完成时，只需要进行一次转换，但执行转换工作的一方必须要精确知道对方的局部语法。要求一台计算机能实现很多种类型的语法转换是非常困难的。

OSI 采用两次转换语法的方法，即由发送方和接收方共同协作完成**语法转换**。为此，定义了一种标准语法，即“**传送语法**”(transfer syntax)。发送方把符合自己局部语法的比特串转换为符合传送语法的比特串，接收方再把此比特串转换为符合自己局部语法的比特串（即通过“局部语法→传送语法→局部语法”的两次转换）。在采用这种标准的传送语法时，不仅要传送数据对象的“**值信息**”，还需要传送关于该对象的“**类型信息**”。

OSI 提出的 ASN.1 (Abstract Syntax Notation One，抽象语法记法 1)是一种数据类型描述语言，具有类似于面向对象程序设计语言中所提供的类型机制，它可定义任意复杂结构的数据类型，而不同的数据类型之间还可以有继承关系。ISO 原以为还会制定出更多的抽象语法记法，但实际上，到目前为止并没有第二个抽象语法记法出现。因此 ASN.1 似应写为 ASN。**抽象语**

法只描述数据的结构形式且与具体的编码格式无关，同时也不涉及这些数据结构在计算机内如何存放。

在定义应用层协议时，报文的结构很复杂。如仍用自然语言描述，则既不方便又容易出错。为此，引入了抽象语法记法 ASN.1 来描述报文的结构。其实，ASN.1 语言和 C 语言中的 struct 结构并无本质区别，只是可描述的数据类型更多而已。这里“抽象”的含义是指 ASN.1 不够具体，仅从 ASN.1 的描述还无法惟一地确定报文在发送时的比特流，因为可能使用的编码方案有多种。

ISO 在制订 ASN.1 的同时也为它定义了一种标准的编码方案，即**基本编码规则** BER (Basic Encoding Rule)。BER 指明了每种数据类型中每个数据值的表示。在发送端用 BER 编码，可将用 ASN.1 所表述的报文转换成惟一的比特序列。在接收端用 BER 进行解码，就可得到该比特序列所表示的 ASN.1 报文。协议实现者可根据实际情况选择合适的编码和解码程序，ASN.1 对此未做统一规定。

这样，ASN.1 就有以下两个标准：

（1）抽象语法记法 1(ASN.1)　　ISO 8824　　ITU-T X.208

（2）ASN.1 的基本编码规则　　ISO 8825　　ITU-T X.209

ASN.1 和 ASN.1 基本编码规则的区别就是：ASN.1 是用来定义各种应用协议数据单元的数据类型的工具，是描述抽象语法的一种语言。ASN.1 基本编码规则用于描述各应用协议数据单元类型所代表的数据值。但很多人在使用 ASN.1 这一术语时，也笼统地包含了抽象语法和上述的基本编码规则。

总之，**抽象语法**是协议设计者所使用的工具，用于将设计者的思想记录下来，便于交流和讨论；**传送语法**用于数据在线路上的**传输**；而**局部语法**则用于数据在端系统中的存储。计算机通信的最终目的是传递数据的语义。因此一个数据无论采用何种表示方式，其语义不应改变。这里“数据的语义”指的是数据在被使用时所表现出来的性质。

由此可见，采用上述的几种语法可解决分布式异构环境下数据表示的两个重要问题：

（1）在不同系统之间的数据交换有了一个共同的表示。

（2）在一个系统内部，一个应用程序可使用某种特殊的数据表示。但使用抽象语法和传送语法可自动地解决互相协作的应用实体表示的差异。

虽然因特网不采用 OSI 的体系结构，但因特网的网络管理协议 SNMP 还是采用了 ISO 制定的 ASN.1 标准，因为 ASN.1 是一个制订得比较成功的标准。

2．抽象语法记法 ASN.1 的要点

ASN.1 的词法有这样一些约定：

（1）标识符（即值的名或字段名）、数据类型名和模块名由大写或小写字母、数字、以及连字符组成。

（2）ASN.1 固有的数据类型全部由大写字母组成。

（3）用户自定义的数据类型名和模块名的第一个字母用大写，后面至少要有一个非大写字母。

（4）标识符(identifier)的第一个字母用小写，后面可用数字、连字符以及一些大写字母以增加可读性。

（5）多个空格或空行都被认为是一个空格。

（6）注释由两个连字符（--）表示开始，由另外两个连字符或行结束符表示结束。

和程序设计语言中的数据定义一样，ASN.1 也把数据类型分为**简单类型**和**构造类型**两种。表 8-10 为 SNMP 所用到的 ASN.1 的部分类型名称及其主要特点。

表 8-10　SNMP 所用到的 ASN.1 的部分类型名称及其主要特点

分类	标　记	类 型 名 称	主 要 特 点
简	UNIVERSAL 2	INTEGER	取整数值数据类型
单	UNIVERSAL 4	OCTET STRING	取八位位组序列值的数据类型①
类	UNIVERSAL 5	NULL	只取空值的数据类型（用于尚未获得数据的情况下）
型	UNIVERSAL 6	OBJECT IDENTIFIER	与信息对象相关联的值的集合
构	UNIVERSAL 16	SEQUENCE	取值为多个数据类型的按序组成的值
造	UNIVERSAL 16	SEQUENCE-OF	取值为同一数据类型的按序组成的值
类	无标记	CHOICE	可选择多个数据类型中的某一个数据类型
型	无标记	ANY	可描述事先还不知道的任何类型的任何值

在表 8-10 中的第二列是标记(tab)。ASN.1 规定每一个数据类型应当有一个能够惟一被识别的标记，以便能无二义性地标识各种数据类型。标记有两个分量，一个分量是标记的类(class)，另一个分量是非负整数。标记共划分为以下的 4 类(class)：

（1）**通用类**(Universal)　由 ASN.1 分配给所定义的最常用的一些数据类型，它与具体的应用无关。表 8-10 中给出的类型都是通用类。

（2）**应用类**(Application-wide)　与某个特定应用相关联的类型（被其他标准所定义）。

（3）**上下文类**(Context-specific)　上下文所定义的类型，它属于一个应用的子集。

（4）**专用类**(Private)　保留为一些厂家所定义的类型，在 ASN.1 标准中未定义。

下面举出一些简单的例子。当使用 ASN.1 定义 SNMP 报文中的 get-request-PDU 时，可写出（可参阅图 8-26）：

```
SNMP-Message ::= SEQUENCE {                          --“::=”表示“定义为”
    version        INTEGER { version-1(0)}           --整数取值为 0，表示 version-1
    community      OCTET STRING                      --共同体的名字
    data           ANY }                             --数据部分放入各种 PDU
Get-request-PDU ::= [0] IMPLICIT PDU                 --[0]表示上下文类，编号为 0
PDU ::= SEQUENCE {
    request-id         INTEGER,                      --变量 request-id 的类型是 INTEGER
    error-status       INTEGER { noError(0),         --若取值为 0，则表示 noError
                                 tooBig(1),          --若取值为 1，则表示 tooBig
                                 noSuchName(2),      --若取值为 2，则表示 noSuchName
                                 badValue(3),        --若取值为 3，则表示 badValue
                                 readOnly(4),        --若取值为 4，则表示 readOnly
                                 genErr(5)},         --若取值为 5，则表示 genErr
```

① 注：八位位组(octet)是一个无二义性的术语，它表示八个比特构成的序列。通用的不严格的表示是“字节”。虽然目前大多数计算机采用 1 字节表示 8 比特，但的确有少数计算机的 1 字节不是 8 比特。

```
    error-index          INTEGER,
    variable-bindings    VarBindList }
VarBindList ::= SEQUENCE OF VarBind
VarBind ::= SEQUENCE {name     ObjectName,
                      value    ObjectSyntax}
```

从以上可看出，一个 SNMP 报文定义为一个包含版本、共同体以及数据的序列，而数据可以是任何一种 PDU。Get-request-PDU 定义为编号为 0 的上下文类的 PDU（编号是为了区分不同的 PDU，其他几种 PDU 的编号为 1 至 4）。而类型 PDU 的定义又写在后面。这里要提一下隐式标记 IMPLICIT，这是为了在进行编码时可省去对 IMPLICIT 后面的类型的编码，使最后得出的编码更加简洁。

PDU 定义为包括 request-id 等 4 个变量的序列。变量 variable-bindings 的类型是 VarBindList，它定义为 VarBind 的序列。而 VarBind 又进一步定义为序偶{name, value}的序列。

上面的最后两个类型 ObjectName 和 ObjectSyntax 在 SMI 中都有严格的定义，这里从略。用非形式语言来说，ObjectName 的类型就是类型 OBJECT IDENTIFIER，而类型 ObjectSyntax 则包括了各种 ASN.1 使用的简单类型和 ASN.1 定义的几种应用类。

3．ASN.1 的基本编码规则

ASN.1 规定了对各种数据值都采用所谓的 TLV 方法进行编码。这种方法把各种数据元素表示为以下三个字段组成的八位位组序列（见图 8-27）：

（1）T 字段，即标识符八位位组(identifier octet)，用于标识标记。

（2）L 字段，即长度用八位位组(length octet)，用于标识后面 V 字段的长度。

（3）V 字段，即内容八位位组(content octet)，用于标识数据元素的值。

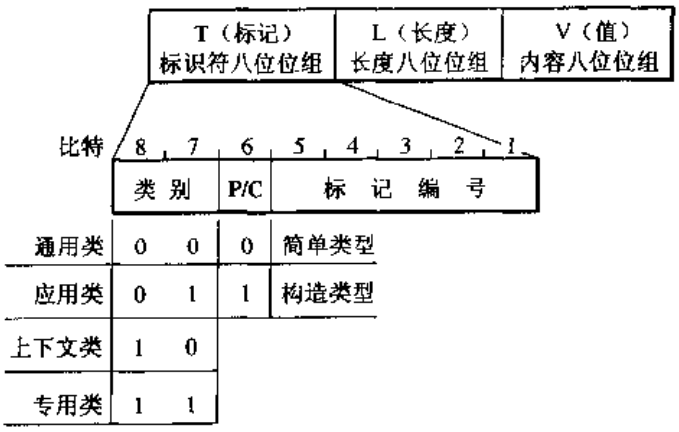


图 8-27　用 TLV 方法进行编码

T 字段的比特 8～7 代表类别（第 8 比特在最左边），即用 00, 01, 10 和 11 分别代表通用类、应用类、上下文类和专用类。比特 6 是 P/C 比特。P/C = 0 为简单类型，而 P/C = 1 为构造类型。比特 5～1 为标记的编号（二进制整数，比特 5 为最高位），编号的范围为 0～30。当标记编号大于 30 时，T 字段就扩展为多个字节（因很少用到这种情况，这里从略）。

表示数据元素长度的 L 字段由 1 个或多个字节组成。当 L 字段仅为 1 字节时，其比特 8 为 0，因而长度指示最多为 126（字节）（127 暂不用，为保留值）。当长度超过 126 时，L 扩

展为多个字节。此时第 1 个 L 字节的比特 8 置为 1，而比特 7～1 表示后续字节的字节数（用二进制整数表示），比特 7 为最高位。这时所有的后续字节并置起来的二进制整数，即为所指示的数据元素的长度。例如，当长度为 133 字节时，L 字段由两个字节组成，其值为 L=10000001 10000101。若用 16 进制写出时，L=81 85。当写出一串 16 进制数字时，常常在每两个数字之间加一个空格，以改进可读性。

TLV 方法中的 V 字段可嵌套其他数据元素的(T, L, V)字段，并可多重嵌套。下面举出一个例子。我们将 SNMP Get-request 报文的编码写出，如图 8-28 所示。这里假定需要从代理进程获得数据项目 sysDescr 的信息，而 sysDescr 的对象标识符是 1.3.6.1.2.1.1.1。由于 sysDescr 已是对象命名树的树叶，并且是个简单的标量而不是数组，因此还要在对象标识符的最后加上一个后缀“.0”。即 sysDescr 的对象标识符是 1.3.6.1.2.1.1.1.0。

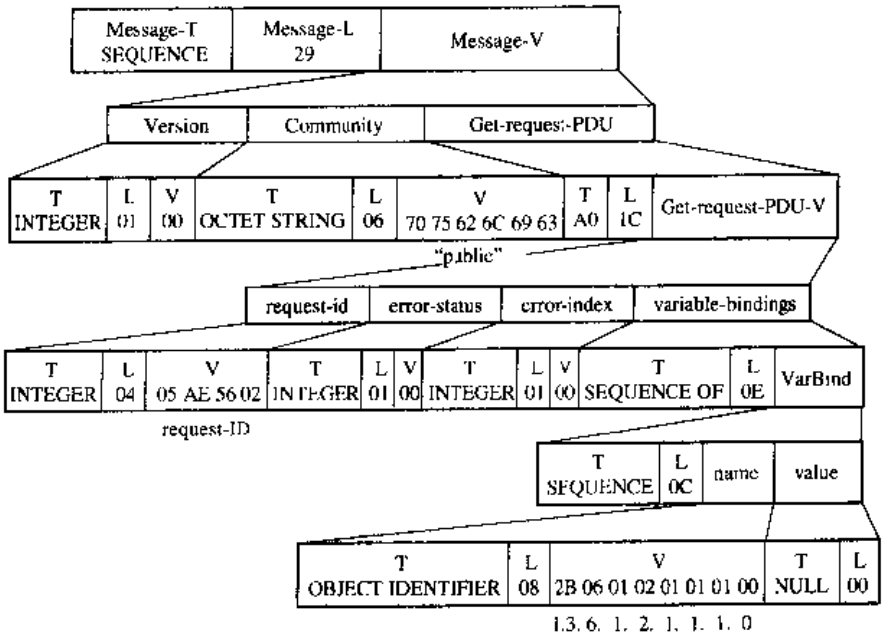


图 8-28　SNMP 的 Get-request 报文的 ASN.1 编码

这样，可将前面的 SNMP get-request 报文的编码写出如下。

30 29	{类型 SEQUENCE，长度 $29_{16}=41_{10}$}
02 01 00	{类型 INTEGER，长度 01_{16}，版本 = 0}
04 06 70 75 62 6C 69 63	{类型 OCTET STRING，长度 06_{16}，“public”}
A0 1C	{类型“上下文结构类型”，长度 $1C_{16}$}
02 04 05 AE 56 02	{类型 INTEGER，长度 04_{16}，request-id = 05 AE 56 02}
02 01 00	{类型 INTEGER，长度 01_{16}，error status = 00_{16}}
02 01 00	{类型 INTEGER，长度 01_{16}，error index = 00_{16}}
30 0E	{类型 SEQUENCE OF，长度 $0E_{16}$}
30 0C	{类型 SEQUENCE，长度 $0C_{16}$}
06 08 2B 06 01 02 01 01 01 00	{类型 OBJECT IDENTIFIER，长度 08_{16}，sysDescr}
05 00	{类型 NULL，长度 00_{16}}

只要对照编码规则，就不难弄清以上的编码。这里只需说明以下几点。

（1）为了方便阅读，编码一律用 16 进制数来表示。

（2）要特别注意在 V 字段中出现的嵌套。例如，最上面的是 SNMP 报文的 TLV 表示。一开始只有报文的类型是已知的(SEQUENCE)，而长度要在最后才能算出。报文的 V 字段很复杂，按照 SNMP 的格式先划分成三个部分。每个部分都进一步按 TLV 方式进行编码。而对于 get-request-PDU 的 V 字段还要进行多次的嵌套。

（3）由于对象命名树的顶级结点仅 3 个，而每一个顶级结点下面的二级结点数不超过 39 个，因此在进行编码时，将顶级和二级结点合并成一个**子标识符**(sub-identifier)。算法是：

若顶级结点和二级结点的值分别为 X 和 Y，子网得出的子标识符的值为 40X + Y。这样就得出 sysDescr 在进行编码时的对象标识符为 43.6.1.2.1.1.1.0（即占两个字符的 1.3 压缩为占一个字符的 43），节省了一个字符的空间。

（4）最后得到的用十六进制表示的编码如下所示：

30 29 02 01 00 04 06 70 75 62 6C 69 63 A0 1C 02 04 05 AE 56 02 02 01 00 02 01 00 30 0E 30 0C 06 08 2B 06 01 02 01 01 01 00 05 00

这就是作为 UDP 用户数据报的数据部分的一个完整的 SNMP 报文。

8.7.6 SNMPv2 和 SNMPv3

SNMP (现在将第一个版本的 SNMP 称为 SNMPv1)的优点是非常简单，因此得到了广泛的推广使用。但其主要缺点是：

（1）不能有效地传送大块的数据。

（2）不能将网络管理的功能分散化。

（3）安全性不够好。

本来因特网还有一个远期的网络管理标准 CMOT (Common Management information service and protocol Over TCP/IP)，意思是“在 TCP/IP 上的公共管理信息服务与协议”。虽然 CMOT 使用了 OSI 的网络管理标准 CMIS/CMIP，但现在还远未达到实用阶段。相反，SNMP 却在因特网中被广泛地使用。1996 年发布 IETF 发布了 8 个 SNMPv2 文档[RFC 1901～1908]。但是 SNMPv2 在安全方面的设计却过分复杂，使得有些人不愿意接受它。

SNMPv2 增加了一个叫做 get-bulk-request 的命令，可一次从路由器的路由表中读取许多行的信息，而不是像 SNMPv1 那样，一次只能读取一行的信息。即使在读取整个的路由表信息的同时，还可再读取其他某些变量的信息。

SNMPv2 的另一个特点是使用非原子的 get 命令。SNMPv1 在使用 get 命令读取多个变量的信息时，只要有一个变量的值不能返回，整个的 get 命令就被拒绝。这时，管理进程就要减少变量的数目，重新发送 get 命令。SNMPv2 的 get 命令允许返回部分的变量值，这就提高了效率，减少了网络上的通信量。

当网络规模扩大时，只用一个网络管理站对全网集中管理是不适宜的。SNMPv2 采用了分散化的管理方法。在一个网络中可以有多个顶级管理站，叫做**管理服务器**。每一个这样的管理服务器管理网络中的一部分代理进程，并指派若干个代理进程使之具有管理其他代理进程的功能。这就是**中间管理进程**，或**网元管理进程**、**管理进程/代理进程**。这种体系结构分散了处理功能，使得网络总的通信量明显地降低。

为了支持这样的配置，SNMPv2 增加了一个 inform 命令和一个管理进程到管理进

程的 MIB (manager-to-manager MIB)。使用这种 inform 命令可以使管理进程之间互相传送有关的事件信息而不需要经过请求。这样的信息则定义在管理进程到管理进程的 MIB 中。

在 1998 年 1 月 IETF 发表了 SNMPv3 的有关文档[RFC 2271-2275]。仅隔 15 个月后就更新为[RFC 2571-2575]。SNMPv3 最大的改进就是安全特性。也就是说，只有被授权的人员才有资格执行网络管理的功能（如关闭某一条链路）和读取有关网络管理的信息（如读取一个配置文件的内容）。

目前 SNMPv3 和 SNMPv2 都还是因特网草案标准。关于 SNMPv2 和 SNMPv3 的进一步的学习可参阅[SUBR01] [HARN98]和[W-SNMPv3]。

8.8 应用进程跨越网络的通信

在这以前我们已经讨论了因特网使用的几种常用的应用层协议，这些应用协议使广大用户可以更加方便地利用因特网的资源。

现在的问题是：如果我们还有一些特定的应用需要因特网的支持，但这些应用又不能直接使用已经标准化的因特网应用协议，那么我们应当做哪些工作？要回答这个问题实际上就是要了解下面要介绍的**系统调用**和**应用编程接口**。

8.8.1 系统调用和应用编程接口

大多数操作系统使用**系统调用**(system call)的机制在应用程序和操作系统之间传递控制权。对程序员来说，每一个系统调用和一般程序设计中的函数调用非常相似，只是系统调用是将控制权传递给了操作系统。图 8-29 说明了多个应用进程使用系统调用的机制。

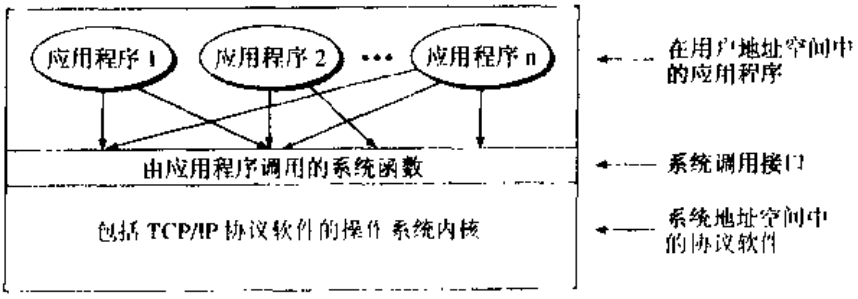


图 8-29 多个应用进程使用系统调用的机制

当某个应用进程启动系统调用时，控制权就从应用进程传递给了系统调用接口。此接口再将控制权传递给计算机的操作系统。操作系统将此调用转给某个内部过程，并执行所请求的操作。内部过程一旦执行完毕，控制权就又通过系统调用接口返回给应用进程。总之，只要应用进程需要从操作系统获得服务，就要将控制权传递给操作系统，操作系统在执行必要的操作后将控制权返回给应用进程。因此，系统调用接口实际上就是应用进程的控制权和操作系统的控制权进行转换的一个接口。由于应用程序在使用系统调用之前要编写一些程序，特别是需要设置系统调用中的许多参数，因此这种系统调用接口又称为**应用编程接口** API (Application Programming Interface)。

由于 TCP/IP 协议族被设计成能运行在多厂商的环境中，因此 TCP/IP 标准没有规定应用程序与 TCP/IP 协议软件如何接口的细节，而是允许系统设计者能够选择有关 API 的具体实

现细节。目前只有几种可供应用程序使用 TCP/IP 的应用编程接口 API。这里最著名的就是美国加利福尼亚大学伯克利分校为 Berkeley UNIX 操作系统定义了一种 API，它又称为**插口接口**(socket interface)。微软公司在其操作系统中采用了**插口接口** API，形成了一个稍有不同的 API，并称之为 Windows Socket。AT&T 为其 UNIX 系统 V 定义了一种 API，简写为 TLI (Transport Layer Interface)。

我们已经知道，若要让计算机做某件事情，就要编写使计算机能理解的程序。在网络环境下的计算机应用都有一个共同特点，这就是：在不同地点的计算机要通过网络进行通信。也就是说，某个计算机要读取另一个地点的计算机中的数据，或者要将数据从某个计算机写入到另一个地点的计算机中。现在我们就要把这种“读取”、“写入”的过程和上面所说的系统调用联系起来。

我们知道，应用进程之间是通过端口和运输层进行交互的。但是两个计算机的通信还必须指明它们的 IP 地址。因此，我们也常用**插口**(socket)来表示应用层和运输层之间的接口。图 8-30 强调了这一概念。图中假定了运输层使用的是 TCP 协议（如使用 UDP 协议，情况也是类似的，只是 UDP 是无连接的。通信的两端仍然是用两个插口来标志）。

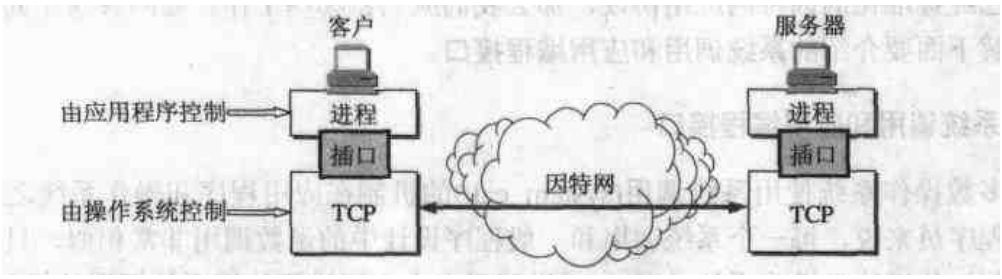


图 8-30　应用进程通过插口接入到网络

当应用进程需要使用网络进行通信时就发出一个系统调用，请求操作系统为其创建一个“插口”，其实际效果是请求操作系统把网络通信所需要的一些系统资源（存储器空间、CPU 时间、网络带宽等等）分配给该应用进程。操作系统为这些资源的总和用一个号码来表示，然后把这个号码返回给应用进程。此后，应用进程所进行的网络操作（建立连接、收发数据、调整网络通信参数等）都必须使用这个号码。所以，几乎所有的网络系统调用都把该号码作为第一个参数。在处理系统调用的时候，通过这个参数，操作系统就可以识别出应该使用哪些资源来完成应用进程所请求的服务。通信完毕后，应用进程通过一个关闭插口的系统调用通知操作系统回收与该“号码”相关的所有资源。由此可见，插口是应用进程为了获得网络通信服务而与操作系统进行交互时使用的一种机制。

我们也可以将插口看成是**应用进程和网络之间的接口**，因为插口既包含有运输层与应用层之间的端口号，又包含有机器的 IP 地址。插口和应用编程接口 API 是性质不同的接口。API 是从**程序设计的角度**定义了许多标准的系统调用函数。应用进程只要使用标准的系统调用函数就可得到操作系统的服务。因此在这个意义上讲，API 是**应用程序和操作系统之间的接口**。

我们还应当记住：在插口以上的进程是受应用程序控制的，而在插口以下的 TCP 协议软件以及 TCP 使用的缓存和一些必要的变量等，则是受计算机操作系统的控制。因此，只要应用程序使用 TCP/IP 协议进行通信，它就必须通过插口与操作系统交互（这就要使用系统调用函数）并请求其服务。我们应当注意到，应用程序的开发者对插口以上的应用进程具有完全的控制，但对插口以下的运输层却只有少量的控制，例如，选择运输层协议和一些运输层的

参数（如最大缓存空间和最大报文长度等）。

下面我们通过两个例子来说明两个进程通过系统调用接口进行通信的过程。但需要先介绍一下服务器的两种工作方式。

8.8.2 服务器的两种工作方式

服务器都可工作在两种不同的方式：**循环方式**(iterative mode)和**并发方式**(concurrent)。循环方式就是在计算机中一次只运行一个服务器进程。当有多个客户进程请求服务时，服务器进程就按请求的先后顺序依次做出响应。并发方式则可在计算机中同时运行多个服务器进程，而每一个服务器进程都对某个特定的客户进程做出响应。

服务器可使用 UDP 无连接的运输层协议，也可以使用 TCP 这样的面向连接的运输层协议。因此，从理论上讲，我们可以有四种不同的服务器：无连接循环服务器；无连接并发服务器；面向连接循环服务器和面向连接并发服务器。不过在实际上人们只使用第一种和第四种。我们将在下面分别介绍这两种服务器的特点。

至于客户进程这里就不再专门讨论，因为客户进程的工作过程比较简单，而现在的计算机都能够支持客户进程的并发运行。

1．无连接循环服务器

使用无连接的 UDP 的服务器通常都工作在循环方式，这种工作方式的特点如图 8-31 所示。最主要的特点就是：**一个服务器在同一时间只能向一个客户提供服务。**

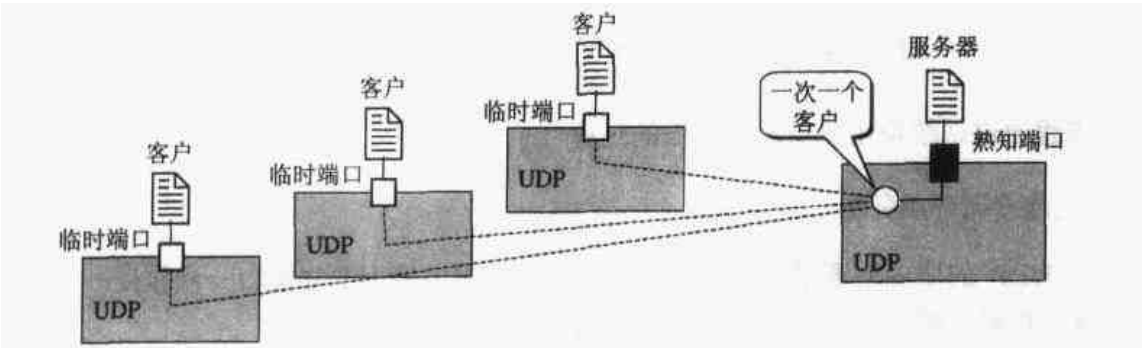


图 8-31 无连接循环服务器的特点

服务器收到客户的请求后，就发送 UDP 用户数据报响应该客户。但对其他客户发来的请求则暂时不予理睬，这些请求都在服务器端的一个队列中排队等候服务器的处理。当服务器进程处理完毕一个请求时，就从队列中读取来自下一个客户的请求，然后继续处理。

这样的服务器只使用一个熟知端口。所有请求服务的客户都通过这个端口得到服务器的响应。每一个客户则使用自己创建的临时端口（端口号自己设定）。

2．面向连接并发服务器

使用面向连接的 TCP 的服务器通常都是工作在并发方式。这表示**一个服务器在同一时间可以向多个客户提供服务**。由于 TCP 的通信是面向连接的，因此在服务器和多个客户之间必须建立多条 TCP 连接，而每一条 TCP 连接要在其数据传送完毕后才能释放。

使用 TCP 的服务器只能有一个熟知端口（这样才能使各地的客户找到这个服务器），但

建立多条连接又必须有多个端口。因此并发服务器采用这样的工作方式：**主服务器**（就是原来的服务器）在熟知端口等待客户发出的请求。主服务器一旦收到客户的请求，就立即创建一个**从属服务器**，并指明从属服务器使用一个临时端口和该客户建立 TCP 连接，然后主服务器继续在原来的熟知端口等待向其他客户提供服务。假定再有一个客户提出服务请求，主服务器就再创建出一个从属服务器，并指明它使用另一个临时端口和这个客户建立 TCP 连接。图 8-32 表示了这种情况。图中的主服务器共创建了三个从属服务器，它们分别在三个临时端口和三个不同地点的客户建立三条 TCP 连接。客户都使用临时端口。

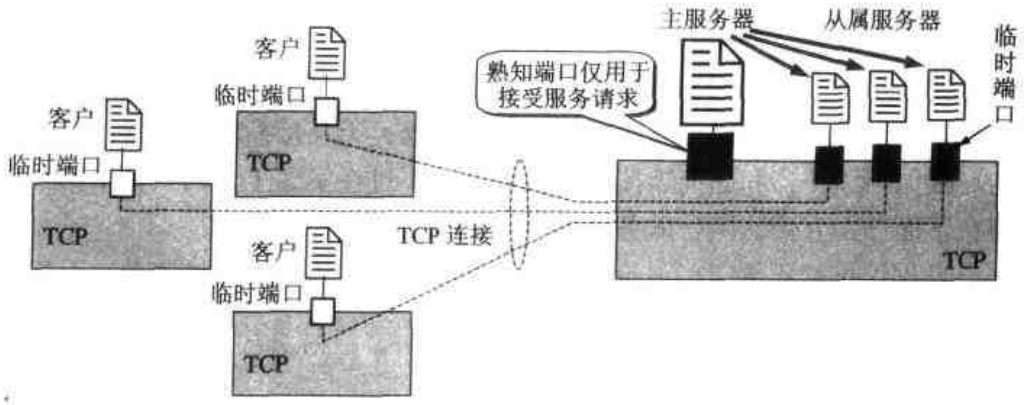


图 8-32　面向连接并发服务器的特点

从图 8-32 可看出，面向连接并发服务器使用熟知端口就是为了接受服务请求。并发服务器之所以能够在同一时间和多个客户建立多条 TCP 连接，是因为创建了多个从属服务器。主服务器有时又称为**父服务器**，而从属服务器又称为**子服务器**。

8.8.3　进程通过系统调用接口进行通信的过程

1．无连接循环服务器

使用 UDP 的服务器通常是无连接循环服务器。服务器用熟知端口一次接受一个客户的服务请求。进程之间的通信过程如图 8-33 所示。

图中所示的过程都很清楚，这里只对所用到的插口系统调用函数进行简单的解释。

（1）socket( )调用，是用来创建一个插口。这个调用返回一个整数（即前面 8.8.1 节提到的“号码”），其标准名称是**“插口描述符”**，该描述符惟一地定义了这个新创建的插口。

（2）bind( )调用，指定插口所使用的 IP 地址和端口号，又称“本地插口地址”。

（3）recvfrom( )调用，将到达插口的入队列中的下一个数据报提取出来。

（4）sendto( )调用，将一个数据报从出队列中取出，并用 UDP 发送给远地机器的一个进程。远地机器的插口地址是从上面的 recvfrom( )调用得到的。

（5）close( )调用，是用来关闭一个插口。

在客户端往往不需要启动 bind( )调用，因为操作系统会给新创建的插口指明一个本地插口地址（使用本地 IP 地址和临时的端口号）。

2．面向连接并发服务器

使用 TCP 的服务器都是面向连接并发服务器，这种服务器可同时和多个客户建立连接，

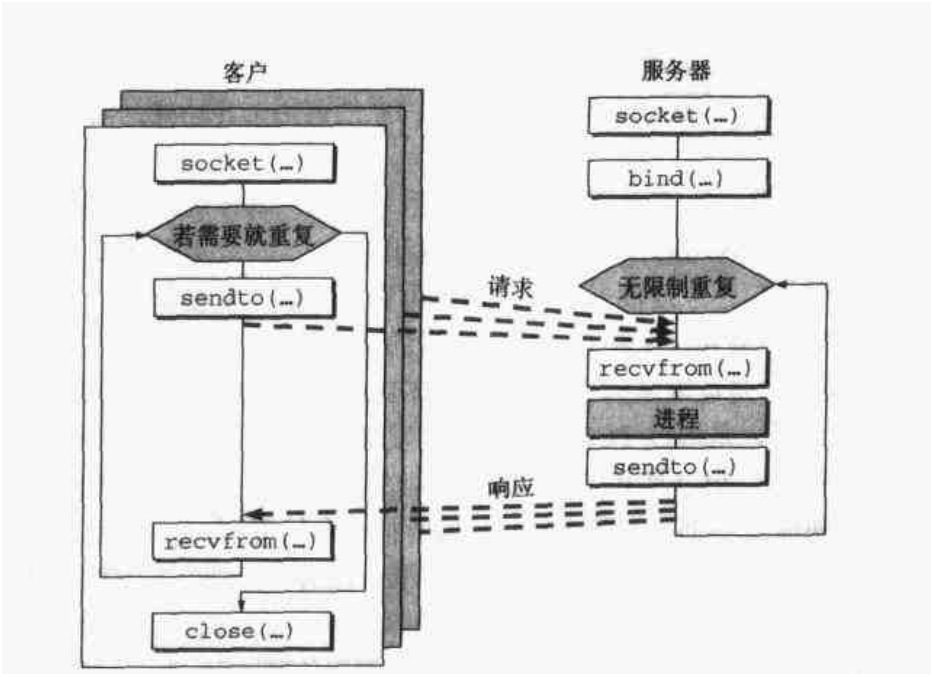


图 8-33　无连接循环服务器的通信过程

但服务器只使用一个熟知端口。服务器为每一条连接都要设置一个缓存，用来存放来自客户的报文段。这种服务器的最重要的特点是主服务器会创建多个从属服务器。图 8-34 是面向连接并发服务器的通信过程。

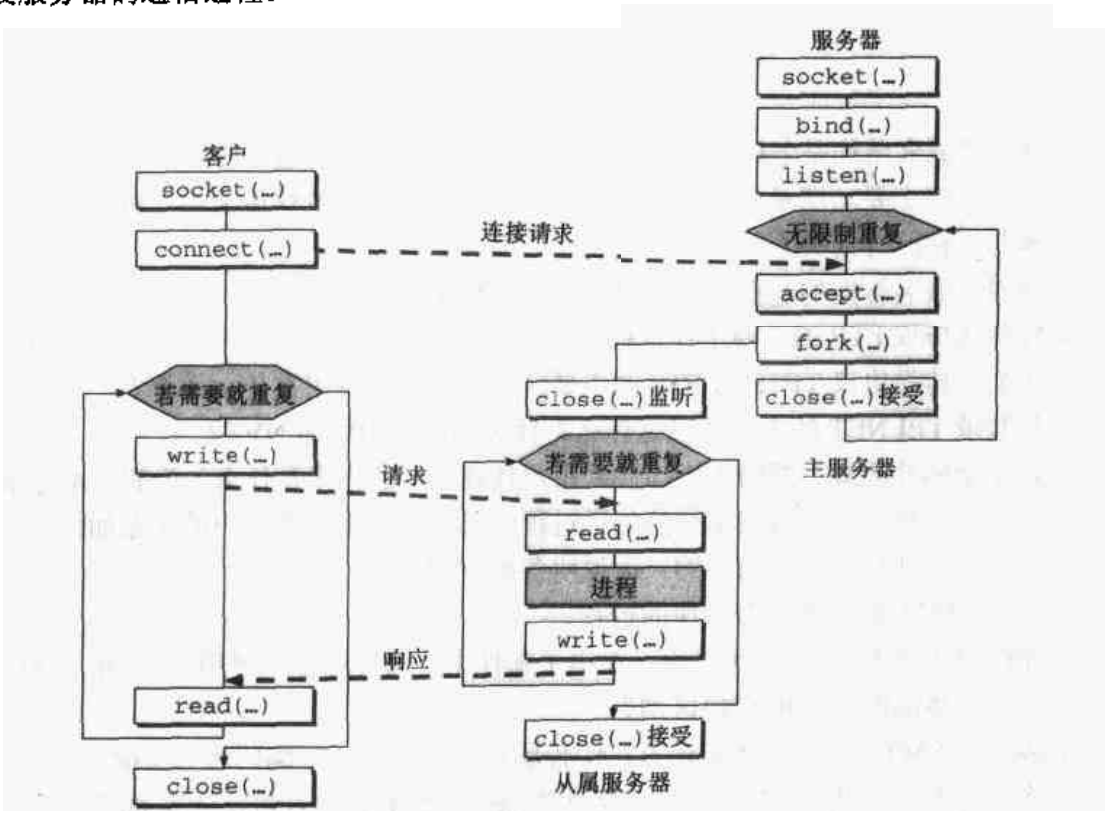


图 8-34　面向连接并发服务器的通信过程

这里对图中新出现的系统调用函数简单解释如下。

(1) listen( )调用，是仅为 TCP 服务器使用的系统调用，其作用是使已经创建的插口变成被动插口，即**监听插口**。监听插口的用处不是和远地插口建立连接，而只是等待远地客

户发出的连接请求。Listen( )调用通知操作系统：服务器已做好接受连接的准备。

（2）connect( )调用，通常是客户进程使用的系统调用，其功能是向远地进程（通常是服务器）请求建立连接。

（3）accept( )调用，是 TCP 服务器使用的系统调用，其作用是从入队列中提取最前面的连接请求。accept( )调用的作用是创建一个新的插口，叫做**接受插口**。此后，客户将只和这个新创建的接受插口建立连接并通信，而不再和监听插口发生联系。

（4）fork( )调用，创建一个和自己完全一样的从属进程（或子进程）。

（5）read( )调用，是读取从远地机器通过 TCP 连接传送到缓存中的数据。

（6）write( )调用，是通过 TCP 连接将数据发送到远地机器的缓存中。

我们应当注意在 fork( )调用完成后的那个时刻，客户同时连接到主服务器和从属服务器。主服务器和从属服务器都具有两个插口：监听插口和接受插口。因此，在 fork( )调用完成后有两个不同的 close( )调用。在 close( )方框中注有“接受”二字的表明主服务器发出 close( )调用来关闭其接受插口，但其监听插口仍然是打开的，因为还要等待下一个客户的连接请求。在 close( )方框中注有“监听”二字的表明从属服务器发出 close( )调用来关闭其监听插口，但其接受插口仍然是打开的。

从属服务器用 read( )调用读取入缓存中的数据，经过处理后，用 write( )调用将结果发送给客户。在所有的服务结束后，从属服务器启动 close( )调用以便关闭这个为通信用的插口。

客户端的过程很简单，这里不再讨论。

习题

8-01 因特网的域名结构是怎样的？它与目前的电话网的号码结构有何异同之处？

8-02 域名系统的主要功能是什么？域名系统中的根服务器和授权服务器有何区别？授权服务器与管辖区有何关系？

8-03 举例说明域名转换的过程。域名服务器中的高速缓存的作用是什么？

8-04 文件传送协议 FTP 的主要工作过程是怎样的？主进程和从属进程各起什么作用？

8-05 简单文件传送协议 TFTP 与 FTP 的主要区别是什么？各用在什么场合？

8-06 远程登录 TELNET 的主要特点是什么？什么叫做虚拟终端 NVT？

8-07 试述电子邮件的最主要的组成部件。用户代理 UA 的作用是什么？没有 UA 行不行？

8-08 电子邮件的信封和内容在邮件的传送过程中起什么作用？和用户的关系如何？

8-09 电子邮件的地址格式是怎样的？请说明各部分的意思。

8-10 试简述 SMTP 通信的三个阶段的过程。

8-11 试述邮局协议 POP 的工作过程。在电子邮件中，为什么必须使用 POP 和 SMTP 这两个协议？IMAP 与 POP 有何区别？

8-12 MIME 与 SMTP 的关系是怎样的？什么是 quoted-printable 编码和 base64 编码？

8-13 一个二进制文件共 3072 字节长。若使用 base64 编码，并且每发送完 80 字节就插入一个回车符 CR 和一个换行符 LF，问一共发送了多少个字节？

8-14 试将数据 11001100 10000001 00111000 进行 base64 编码，并得出最后传送的 ASCII 数据。

8-15 试将数据 01001100 10011101 00111001 进行 quoted-printable 编码，并得出最后传送的 ASCII 数据。这样的数据用 quoted-printable 编码后，其编码开销有多大？

8-16 电子邮件系统需要将人们的电子邮件地址编成目录以便于查找。要建立这种目录应将人名划分为几个标准部分（例如，姓、名）。若要形成一个国际标准，那么必须解决哪些问题？

8-17 电子邮件系统使用 TCP 传送邮件。为什么有时我们会遇到邮件发送失败的情况？为什么有时对方会收不到我们发送的邮件？

8-18 解释以下名词。各英文缩写词的原文是什么？
WWW，URL，URI，HTTP，HTML，CGI，浏览器，超文本、超媒体、超链，页面、表单(form)，活动文档，搜索引擎。

8-19 假定一个超链从一个万维网文档链接到另一个万维网文档时，由于万维网文档上出现了差错而使得超链指向一个无效的计算机名字。这时浏览器将向用户报告什么？

8-20 当使用鼠标点取一个万维网文档时，若该文档除了有文本外，还有一个本地.gif 图像和两个远地.gif 图像。试问：需要使用哪个应用程序，以及需要建立几次 UDP 连接和几次 TCP 连接？

8-21 你所使用的浏览器的高速缓存有多大？请进行一个实验：访问几个万维网文档，然后将你的计算机与网络断开，然后再回到你刚才访问过的文档。你的浏览器的高速缓存能够存放多少个页面？

8-22 试创建一个万维网页面，它有一个标题(title)。然后观察浏览器如何使用此标题。

8-23 用你的浏览器读取一个万维网页面，然后利用浏览器菜单上的选项查看该页面的源程序。和本书介绍的 HTML 的格式与标签进行对比，看是否能够看懂。

8-24 假定某文档中有这样几个字：下载 RFC 文档。我们希望当点击到这几个字的地方时就能够链接到下载RFC 文档的网站页面 http://www.ietf.org/rfc.html，试写出有关的 HTML 语句。

8-25 某页面的 URL 为 http://www.abc.net/file/file.html。此页面中有一个网络拓扑结构简图(map.gif)和一段简单的解释文字。我们希望能够从这张简图或者从这段文字中的“网络拓扑”链接到该网络拓扑的详细情况的主页 http://www.topology.net/index.html。试写出两种相应的 HTML 语句。

8-26 领会 HTML 要点的一个好办法就是用 Windows 中的记事本(Notepad)键入一条 HTML 语句，然后另存为.htm 或.html 格式的文件（不要存为.txt 格式），然后再用浏览器（如 IE）打开这个文件，观察所得到的效果。你能够实现上一题的链接吗？

8-27 在浏览器中应当有几个可选解释程序。试给出一些可选解释程序的名称。

8-28 一个万维网网点有 1 千万个页面，平均每个页面有 10 个超链。读取一个页面平均要 100 ms。问要检索整个网点所需的最少时间。

8-29 一个文件夹中有两个文件 X 和 Y。从文件 X 中的某处有一个超链的起点：“文件 Y”。点击“文件 Y”就可以链接到文件 Y。这个超链的相应 HTML 语句是：

<A HREF="Y.html">文件 Y</A>

现在将文件 X 移动到另一个文件夹中。再打开文件 X 并点击“文件 Y”这个超链起点，发现已无法找到文件 Y（但文件 Y 并未移动位置）。试解决这个问题。

8-30 BOOTP 和 DHCP 协议有什么关系？这两个协议都用再什么情况下？当一台计算机第一次运行引导程序时，其 ROM 中有没有该主机的 IP 地址、子网掩码或某个域名服务器的 IP 地址？

8-31 什么是网络管理？为什么说网络管理是当今网络领域中的热门课题？

8-32 解释下列术语：网络元素、被管对象、管理进程、代理进程、管理信息库和综合网络管理系统。

8-33 OSI 的五个管理功能域都有哪些内容？

8-34 SNMP 使用 UDP 传送报文。为什么不使用 TCP？

8-35 为什么 SNMP 的管理进程使用探询掌握全网状态属于正常情况，而代理进程用陷阱向管理进程报告属于较少发生的异常情况？

8-36 SNMP 使用哪五种操作？SNMP 在 get 报文中设置了请求标识符字段，为什么？

8-37 什么是管理信息库 MIB？为什么要使用 MIB？

8-38 什么是管理信息结构 SMI？它的作用是什么？

8-39 用 ASN.1 基本编码规则对以下数组(SEQUENCE-OF)进行编码。假定每一个数字占用 4 个字节。

2345

1236

122

1236

8-40 SNMP 要发送一个 Get-request 报文，以便向一个路由器获取 ICMP 的 icmpInParmProbs 的值。在 icmp 中变量 icmpInParmProbs 的标号是(5)，它是一个计数器，用来统计收到的类型为参数问题的 ICMP 差错报告报文的数目。试给出这个 Get-request 报文的编码。

8-41 对象 tcp 的 OBJECT IDENTIFIER 是什么？

8-42 试说明抽象语法、传送语法和局部语法的用途。为什么要使用这几种不同的语法？

8-43 在 ASN.1 中，IP 地址(IPAddress)的类别是应用类。若 IPAddress = 131.21.14.2，试求其 ASN.1 编码。

8-44 什么是应用编程接口 API？它是应用程序和谁的接口？

8-45 试比较服务器的两种不同工作方式：循环方式和并发方式。当服务器使用 UDP 或 TCP 协议时，服务器常工作在何种方式？

8-46 为什么在服务器端除了使用熟知端口外还需要使用临时端口？

8-47 面向连接的进程和无连接的进程在向远地进程发送数据时要使用哪一个系统调用？

8-48 图 8-35 表示了各应用协议在层次中的位置。

(1) 简单讨论一下为什么有的应用层协议要使用 TCP 而有的却要使用 UDP？

(2) 为什么 MIME 画在 SMTP 之上？

(3) 为什么路由选择协议 RIP 放在应用层？

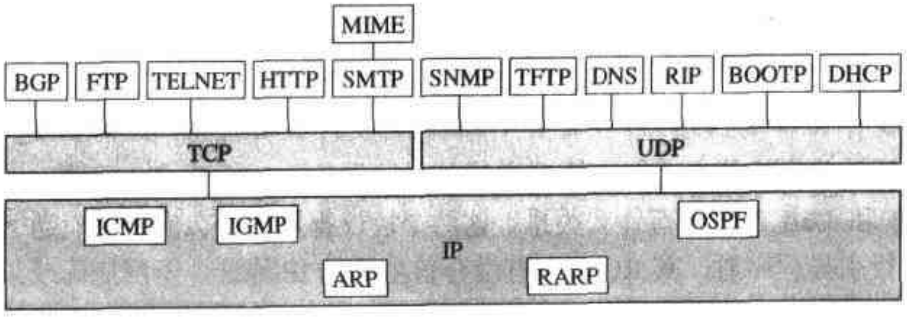


图 8-35 习题 8-48 的图

第 9 章　计算机网络的安全

随着计算机网络的发展，网络中的安全问题也日趋严重。当网络的用户来自社会各个阶层与部门时，大量在网络中存储和传输的数据就需要保护。本章将对计算机网络安全问题的基本内容进行初步的讨论。

9.1　网络安全问题概述

本节讨论计算机网络面临的安全性威胁、安全的内容和一般的数据加密模型。

9.1.1　计算机网络面临的安全性威胁

计算机网络上的通信面临以下的 4 种威胁：

（1）**截获**(interception)　攻击者从网络上窃听他人的通信内容。

（2）**中断**(interruption)　攻击者有意中断他人在网络上的通信。

（3）**篡改**(modification)　攻击者故意篡改网络上传送的报文。

（4）**伪造**(fabrication)　攻击者伪造信息在网络上传送。

上述四种威胁可划分为两大类，即**被动攻击**和**主动攻击**（图 9-1）。在上述情况中，**截获信息的攻击称为被动攻击，而更改信息和拒绝用户使用资源的攻击称为主动攻击。**

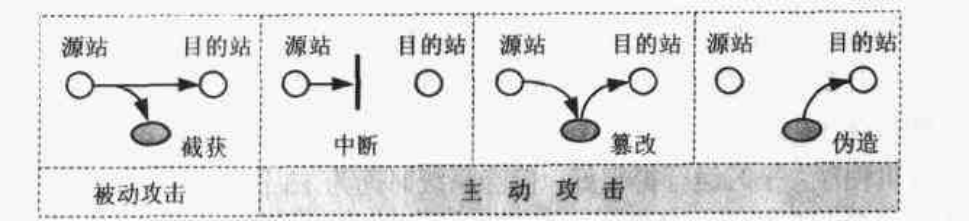


图 9-1　对网络的被动攻击和主动攻击

在被动攻击中，攻击者只是观察和分析某一个**协议数据单元** PDU（这里使用 PDU 这一名词是考虑到所涉及到的可能是不同的层次）而不干扰信息流。即使这些数据对攻击者来说是不易理解的，他也可通过观察 PDU 的协议控制信息部分，了解正在通信的协议实体的地址和身份，研究 PDU 的长度和传输的频度，以便了解所交换的数据的性质。这种被动攻击又称为**通信量分析**(traffic analysis)。

主动攻击是指攻击者对某个连接中通过的 PDU 进行各种处理。如有选择地更改、删除、延迟这些 PDU（当然也包括记录和复制它们）。还可在稍后的时间将以前录下的 PDU 插入这个连接（即**重放攻击**）。甚至还可将合成的或伪造的 PDU 送入到一个连接中去。

所有主动攻击都是上述各种方法的某种组合。但从类型上看，主动攻击又可进一步划分为三种，即：

（1）**更改报文流**　包括对通过连接的 PDU 的真实性、完整性和有序性的攻击。

（2）**拒绝报文服务**　指攻击者或者删除通过某一连接的所有 PDU，或者将双方或单方

的所有 PDU 加以延迟。在 2000 年 2 月 7 日至 9 日美国几个著名网站遭黑客袭击，使这些网站的服务器一直处于“忙”的状态，因而拒绝向发出请求的客户提供服务。这种攻击方式被称为**拒绝服务** DoS (Denial of Service)。若从因特网上的成百上千的网站集中攻击一个网站，则称为**分布式拒绝服务** DdoS (Distributed Denial of Service)。

（3）**伪造连接初始化**　攻击者重放以前已被记录的合法连接初始化序列，或者伪造身份而企图建立连接。

对于主动攻击，可以采取适当措施加以检测。但对于被动攻击，通常却是检测不出来的。根据这些特点，可得出计算机网络通信安全的五个目标如下：

（1）防止析出报文内容。

（2）防止通信量分析。

（3）检测更改报文流。

（4）检测拒绝报文服务。

（5）检测伪造初始化连接。

对付被动攻击可采用各种数据加密技术，而对付主动攻击，则需将加密技术与适当的鉴别技术相结合。

还有一种特殊的主动攻击就是**恶意程序**(rogue program)的攻击。恶意程序种类繁多，对网络安全威胁较大的主要有以下几种：

（1）**计算机病毒**(computer virus)，一种会“传染”其他程序的程序，“传染”是通过修改其他程序来把自身或其变种复制进去完成的。

（2）**计算机蠕虫**(computer worm)，一种通过网络的通信功能将自身从一个结点发送到另一个结点并启动运行的程序。

（3）**特洛伊木马**(Trojan horse)，一种程序，它执行的功能超出所声称的功能。如一个编译程序除了执行编译任务以外，还把用户的源程序偷偷地拷贝下来，则这种编译程序就是一种特洛伊木马。计算机病毒有时也以特洛伊木马的形式出现。

（4）**逻辑炸弹**(logic bomb)，一种当运行环境满足某种特定条件时执行其他特殊功能的程序。如一个编辑程序，平时运行得很好，但当系统时间为 13 日又为星期五时，它删去系统中所有的文件，这种程序就是一种逻辑炸弹。

这里讨论的计算机病毒是狭义的，也有人把所有的恶意程序泛指为计算机病毒。例如 1988 年 10 月的“Morris 病毒”入侵美国因特网。舆论说它是“计算机病毒入侵美国计算机网”，而计算机安全专家却称之为“因特网蠕虫事件”。

限于篇幅，我们在这里只讨论有关网络安全的最基本的概念。

9.1.2　计算机网络安全的内容

由上一节的内容可知，计算机网络安全主要有以下一些内容：

1．保密性

为用户提供安全可靠的保密通信是计算机网络安全最为重要的内容。尽管计算机网络安全不仅仅局限于保密性，但不能提供保密性的网络肯定是不安全的。网络的保密性机制除为用户提供保密通信以外，也是许多其他安全机制的基础。例如，接入控制中登录口令的设计、安全通信协议的设计以及数字签名的设计等，都离不开密码机制。

2．安全协议的设计

人们一直希望能设计出安全的计算机网络，但不幸的是，网络的安全性是不可判定的[DENN82]。目前在安全协议的设计方面，主要是针对具体的攻击（如假冒）设计安全的通信协议。但如何保证所设计出的协议是安全的？协议安全性的保证通常有两种方法，一种是用形式化方法来证明，另一种是用经验来分析协议的安全性。形式化证明的方法是人们所希望的，但一般意义上的协议安全性也是不可判定的，只能针对某种特定类型的攻击来讨论其安全性。对复杂的通信协议的安全性，形式化证明比较困难，所以主要采用找漏洞的分析方法。对于简单的协议，可通过限制敌手的操作（即假定敌手不会进行某种攻击）来对一些特定情况进行形式化的证明，当然，这种方法有很大的局限性。

3．接入控制

接入控制(access control)也叫做**访问控制**或**存取控制**。必须对接入网络的权限加以控制，并规定每个用户的接入权限。由于网络是个非常复杂的系统，其接入控制机制比操作系统的访问控制机制更复杂（尽管网络的接入控制机制是建立在操作系统的访问控制机制之上），尤其在高安全性级别的**多级安全性**(multilevel security)情况下更是如此。

所有上述计算机网络安全的内容都与密码技术紧密相关。如在保密通信中，要用加密算法来对消息进行加密，以对抗可能的窃听；安全协议中的一个重要内容就是要论证协议的安全性取决于加密算法的强度；在接入控制系统的设计中，也要用到加密技术。

9.1.3 一般的数据加密模型

一般的数据加密模型如图 9-2 所示。**明文 X** 用**加密算法 E** 和**加密密钥 K** 得到**密文** $Y = E_K(X)$。在传送过程中可能出现密文截取者。到了收端，利用**解密算法 D** 和**解密密钥 K**，解出明文为 $D_K(Y) = D_K(E_K(X)) = X$。**截取者**又称为**攻击者**，或**入侵者**。在这里我们假定加密密钥和解密密钥都是一样的。但实际上它们可以是不一样的（即使不一样，这两个密钥也必然有某种相关性）。密钥通常是由一个**密钥源**提供。当密钥需要向远地传送时，一定要通过另一个安全信道。

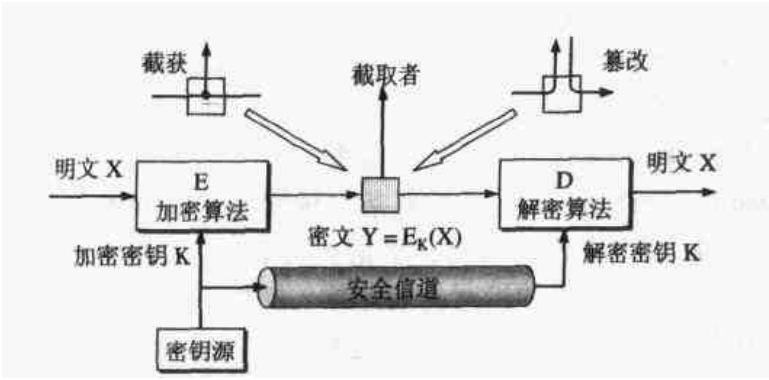


图 9-2　一般的数据加密模型

密码编码学(cryptography)是密码体制的设计学，而**密码分析学**(cryptanalysis)则是在未知密钥的情况下从密文推演出明文或密钥的技术。密码编码学与密码分析学合起来即为**密码学**(cryptology)。

如果不论截取者获得了多少密文，但在密文中都没有足够的信息来惟一地确定出对应的明文，则这一密码体制称为**无条件安全的**，或称为**理论上是不可破的**。在无任何限制的条件下，目前几乎所有实用的密码体制均是可破的。因此，人们关心的是要研制出**在计算上（而不是在理论上）是不可破的密码体制**。如果一个密码体制中的密码不能被可以使用的计算资源破译，则这一密码体制称为**在计算上是安全的**。

早在几千年前人类就已经有了通信保密的思想和方法。但直到 1949 年，信息论创始人香农(C. E. Shannon)发表著名文章[SHAN49]，论证了一般经典加密方法得到的密文几乎都是可破的。密码学的研究曾面临着严重的危机。但从 20 世纪 60 年代起，随着电子技术、计算技术的迅速发展以及结构代数、可计算性和计算复杂性理论等学科的研究，密码学又进入了一个新的发展时期。在 20 世纪 70 年代后期，美国的**数据加密标准** DES (Data Encryption Standard)和**公开密钥密码体制**(public key crypto-system)的出现，成为近代密码学发展史上的两个重要里程碑。

9.2 常规密钥密码体制

所谓**常规密钥密码体制**，即**加密密钥与解密密钥是相同的密码体制**。这种加密系统又称为**对称密钥系统**。我们先介绍在常规密钥密码体制中的两种最基本的密码。

9.2.1 替代密码与置换密码

在早期的常规密钥密码体制中，有两种常用的密码，即**替代密码和置换密码**。

替代密码(substitution cipher)的原理可用一个例子来说明。例如，将字母 a, b, c, d, …, w, x, y, z 的自然顺序保持不变，但使之与 D, E, F, G, …, X, A, B, C 分别对应（即相差 3 个字符，见表 9-1）。

表 9-1 字母 a,b,c,…等与 D, E, F,…等相对应

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	y	w	x	y	z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

若明文为小写字母 caesar cipher，则对应的密文为大写字母 FDHVDU FLSKHU（此时密钥为 3，因为对应的大写字母向左位移了 3 个字母的位置）。由于英文字母中各字母出现的频度早已有人进行过统计，所以根据字母频度表可以很容易对这种替代密码进行破译。目前替代密码只是作为复杂的编码过程中的一个中间步骤。

置换密码(transposition cipher)则是按照某一规则重新排列消息中的比特或字符的顺序。例如，以 CIPHER 这个字作为密钥。根据英文字母在 26 个字母中的先后顺序，我们可以得出密钥中的每一个字母的**相对先后顺序**，例如，因为没有 A 和 B，因此 C 为第 1。同理，E 为第 2，H 为第 3，……，R 为第 6。于是得出密钥字母的相对先后顺序为 145326。形成密文的规律如下：若密钥中的数字 i 在密钥中的顺序是第 j 个，则表示第 i 次读取第 j 列的字符。具体来说，数字 1 在密钥中排为第 1 个，因此第 1 次读取第 1 列的字符 a b a。数字 2 在密钥中排在第 5 个，因此第 2 次读取第 5 列的字符 c n u。下面依次读取第 4 列、第 2 列、第 3 列和第 6 列。下面将明文也以 6 个字符为一组从上到下写在密钥下，如：

密钥	C	I	P	H	E	R	
顺序	1	4	5	3	2	6	（注：此顺序与密钥等价，但不如密钥便于记忆）
明文	a	t	t	a	c	k	（注：明文的意思是“四时开始进攻”）
	b	e	g	i	n	s	
	a	t	f	o	u	r	

这样得出密文为 abacnuaiotettgfksr。接收者按密钥中的字母顺序按列写下，按行读出，即得明文。由于这种密码很容易破译，所以置换密码也是作为加密过程中的中间步骤。

从得到的密文序列的结构来划分，则有**序列密码**与**分组密码**两种不同的密码体制。序列码体制是将明文 X 看成是连续的比特流（或字符流）$x_1x_2\cdots$，并且用密钥序列 $K = k_1k_2\cdots$ 中的第 i 个元素 k_i 对明文中的 x_i 进行加密，即

$$E_K(X) = E_{k1}(x_1)E_{k2}(x_2)\cdots$$

图 9-3 给出了序列密码的框图。在开始工作时种子 I_0 对密钥序列产生器进行初始化。

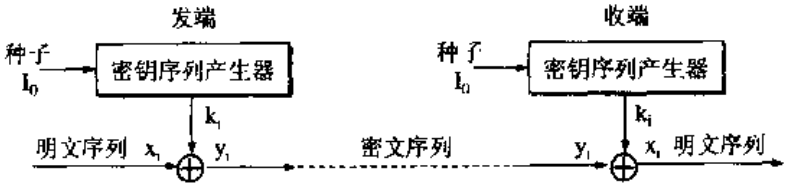


图 9-3　序列密码体制

k_i, x_i 和 y_i 均为 1 bit（或均为 1 个字符），并按照模 2 进行运算，得出：

$$y_i = E_{ki}(x_i) = x_i \oplus k_i \tag{9-1}$$

在收端，对 y_i 的解密算法为：

$$D_{ki}(y_i) = y_i \oplus k_i = (x_i \oplus k_i) \oplus k_i = x_i \tag{9-2}$$

序列密码又称为**密钥流密码**。这种体制的保密性完全在于密钥的随机性。如果密钥是真正的随机数，则这种体制就是理论上不可破的。这也可称为一次一密乱码本体制。

严格的一次一密乱码本体制所需的密钥量不存在上限，很难实用化。密码学家就试图以某种方法模仿这种一次一密乱码本体制。目前常使用**伪随机序列**作为密钥序列。关键是序列的周期要足够长，且序列要有很好的随机性（这很难寻找）。现在周期小于 10^{10} 的序列很少被采用，而周期长达 10^{50} 的序列也不罕见。这种伪随机序列一般用 n 级移位寄存器来构成。

另一种密码体制与序列密码不同。它将明文划分成固定的 n 比特的数据组，然后以组为单位，在密钥的控制下进行一系列的线性或非线性的变化而得到密文。这就是**分组密码** (block cipher)。图 9-4 为分组密码体制的框图。分组密码一次变换一组数据。分组密码算法的一个重要特点就是：**当给定一个密钥后，若明文分组相同，那么所变换出密文分组也相同**。

分组密码的一个重要优点是不**需要同步**，因而在分组交换网中有着广泛的用途。分组密码中最有名的就是美国的**数据加密标准** DES 和**国际数据加密算法** IDEA。

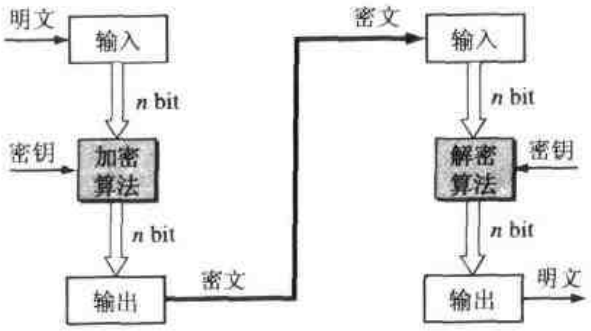


图 9-4　分组密码体制

9.2.2　数据加密标准 DES

数据加密标准 DES 属于常规密钥密码体制。它由 IBM 公司研制出，于 1977 年被美国定为联邦信息标准后，在国际上引起了极大的重视。ISO 曾将 DES 作为数据加密标准。

DES 是一种分组密码。在加密前，先对整个的明文进行分组。每一个组长为 64 bit。然后对每一个 64 bit 二进制数据进行加密处理，产生一组 64 bit 密文数据。最后将各组密文串接起来，即得出整个的密文。使用的密钥为 64 bit(实际密钥长度为 56 bit，有 8 bit 用于奇偶校验)，其加密算法如图 9-5 所示。

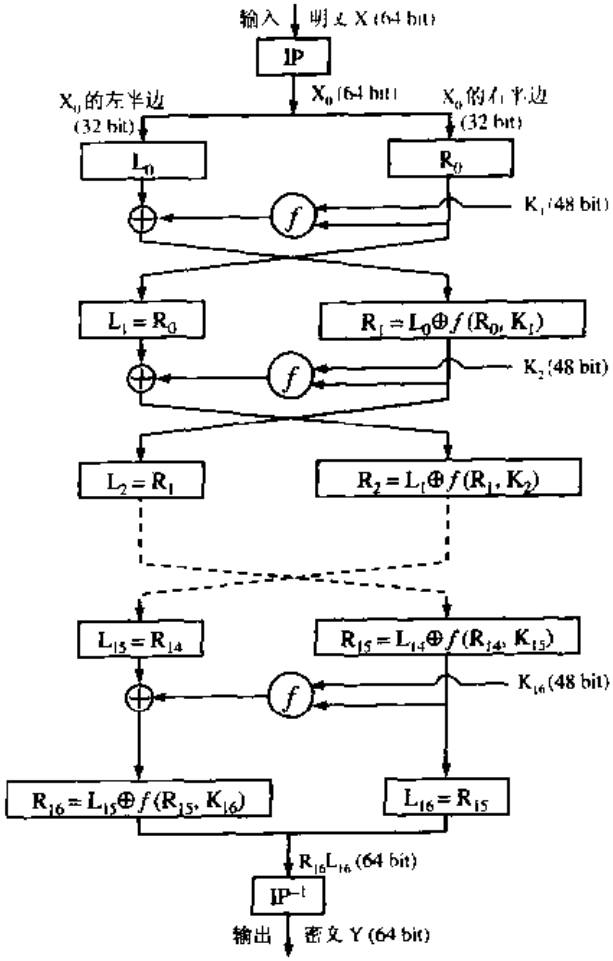


图 9-5　DES 加密标准

64 bit 的明文 X 进行初始置换 IP 后得出 X_0，其左半边 32 bit 和右半边 32 bit 分别记为 L_0 和 R_0。然后再经过 16 次的迭代。如果用 X_i 表示第 i 次的迭代结果，同时令 L_i 和 R_i 分别代表 X_i 的左半边和右半边（各 32 bit），则从图 9-5 可看出，

$$L_i = R_{i-1} \tag{9-3}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{9-4}$$

式中 i = 1, 2, …, 16, 而 K_i 是 48 bit 密钥（从原来的 64 bit 密钥经过若干次变换而得出）。(9-4)式即 **DES 加密方程**。每次迭代要进行函数 f 的变换、模 2 加运算以及左右半边交换。在最后的一次迭代之后，左右半边没有交换。这是为了使算法既能加密又能解密。最后一次的变换是 IP 的逆变换 IP^{-1}，其输入是 $R_{16}L_{16}$。Y 即为输出的密文。

DES 加密中起关键作用的是函数 f。它是个非常复杂的变换（其复杂变化的细节都未画出）。$f(R_{i-1}, K_i)$先将 32 bit 的 R_{i-1} 进行变换，扩展成 48 bit，记为 $E(R_{i-1})$。48 bit 的 $E(R_{i-1})$ 与 48 bit 的 K_i 按比特进行“模 2 加”，所得结果顺序地划分为 8 个 6 bit 长的组 $B_1, B_2, \cdots, B_8$，即

$$E(R_{i-1}) \oplus K_i = B_1 B_2 \cdots B_8$$

然后将 6 bit 长的组经过称为“**S 变换**”的替代转换为长 4 bit 的组（S 也是一个复杂的函数）。或写为 $B_j \rightarrow S_j(B_j)$，(j = 1, 2, …,8)。这里要用到 8 个不同的 S 函数$(S_1, S_2, \cdots, S_8)$。将所得的 8 个 4 bit 长的 $S_j(B_j)$按顺序排好，再进行一次置换，即得出 32 bit 的 $f(R_{i-1}, K_i)$。

解密过程和加密相似，但生成 16 个密钥的顺序正好相反。

上述的 DES 的一个明显的缺点就是它实际上就是一种单字符替代，而这种字符的长度是 64 bit。也就是说，对于 DES 算法，相同的明文就产生相同的密文。这对 DES 的安全性来说是不利的。为了提高 DES 的安全性，可采用加密分组链接的方法（图 9-6）。

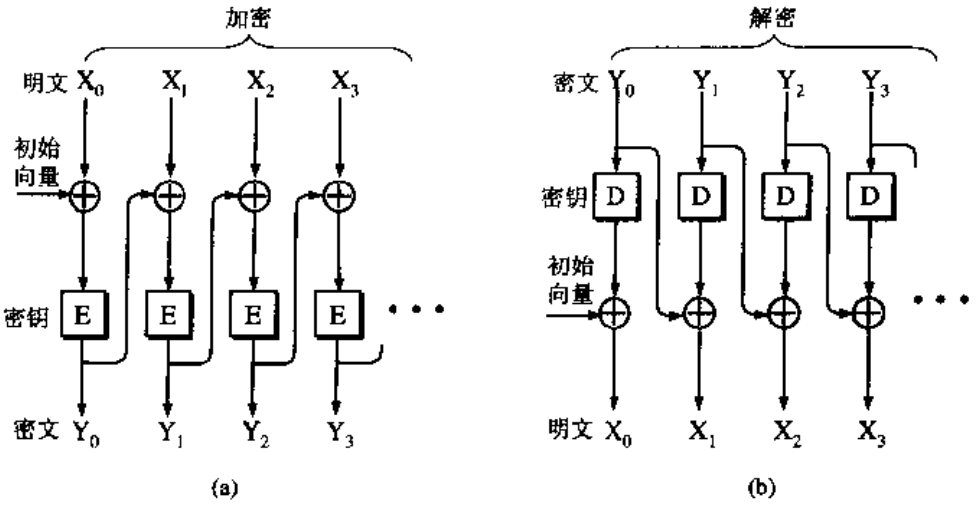


图 9-6　加密分组的链接（方框 E 和 D 分别代表加密和解密操作）

图 9-6(a)是加密过程。64 bit 的明文分组 X_0 先和初始向量逐比特进行异或运算，然后进行加密操作（这不一定限于采用 DES 加密算法），得到密文 Y_0。再将 Y_0 和下一个明文分组 X_1 进行异或运算后再加密，得出密文 Y_1。以后都用相同的方法进行操作。不难看出，即使出现明文相同的分组，加密后得到的密文也是不一样的。这就给攻击者的破译增加了难度。

图 9-6(b)是解密过程。密文分组 Y_0 先解密，再和初始向量进行异或运算，得出明文 X_0。下一个密文分组在经过解密后，要和密文 Y_0 进行异或运算后才得出第二个明文分组 X_1。以后的步骤读者可从图中很容易地弄清楚。

DES 的保密性仅取决于对密钥的保密，而算法是公开的。尽管人们在破译 DES 方面取得了许多进展[BIHA90]，但至今仍未能找到比穷举搜索密钥更有效的方法。

在 DES 的算法中，令人们怀疑最多的就是上面提到的非常复杂的 S 函数。由于 DES 的设计者一直未公布 S 函数的整个设计过程，因此人们很担心破译者可能会利用 S 函数的薄弱环节来攻破 DES。尽管如此，到目前为止，DES 加密算法仍是目前最好的加密算法之一。无论是个人或商业的应用（如银行系统的自动取款机 ATM），信赖 DES 仍然是合理的。学习数据加密的人仍然仔细研究 DES，因为 DES 是世界上第一个公认的实用密码算法标准，它曾对密码学的发展做出了重大贡献。

目前较为严重的问题是 DES 的密钥的长度。56 bit 长的密钥意味着共有 2^{56} 种可能的密钥，也就是说，共约有 7.6×10^{16} 种密钥。假设一台计算机 1 μs 可执行一次 DES 加密，同时假定平均只需搜索密钥空间的一半即可找到密钥，那么破译 DES 要超过 1 千年。

但现在已经设计出来搜索 DES 密钥的专用芯片。例如在 1999 年有一批在因特网上合作的人借助于一台不到 25 万美元的专用计算机，在略大于 22 小时的时间就破译了 56 bit 密钥的 DES。若用价格为 100 万美元或 1000 万美元的机器，则预期的搜索时间分别为 3.5 小时或 21 分钟。

一种叫做**三重 DES** (Triple DES)是 Tuchman 提出的，并在 1985 年成为美国的一个商用加密标准[RFC 2420]。三重 DES 使用两个密钥，执行三次 DES 算法（图 9-7）。图中的方框 E 和 D 分别表示执行加密和解密算法。因此加密时是 E-D-E，解密时是 D-E-D。

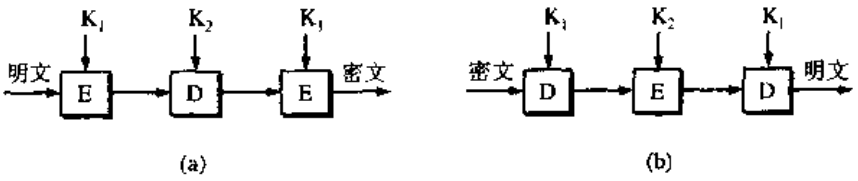


图 9-7　三重 DES

为什么在加密和解密过程中只使用两个而不是三个密钥呢？这是因为两个密钥合起来的长度已有 112 bit，这对商用已足够长（至少目前是这样）。如使用总长为 168 bit 的三个密钥，则产生不必要的开销。

为什么在加密时采用 E-D-E（加密-解密-加密）而不是 E-E-E（加密-加密-加密）呢？这是为了与现有的 DES 系统向后兼容。其实，加密和解密过程都是在两个 64 bit 的数之间的一种映射。从密码的角度来看，这两种映射的作用是一样的。但是，使用 E-D-E 的好处是当 $K_1 = K_2$ 时，三重 DES 的效果就和现有的 DES 一样。这有利于逐渐地推广使用三重 DES。但目前还没有关于攻破三重 DES 的报道。

关于近几年来如何改进 DES 的建议和和新的算法已有不少，可参阅[TANE96]。

在 DES 之后出现了**国际数据加密算法 IDEA** (International Data Encryption Algorithm) [LAI90]。IDEA 使用 128 bit 密钥，因而更加不容易被攻破。IDEA 和 DES 相似，也是先将明文划分成一个个 64 bit 长的数据分组，然后经过 8 次迭代和一次变换，得出 64 bit 密文。对于每一次的迭代，每一个输出比特都与每一个输入比特有关。

计算指出，当密钥长度为 128 bit 时，若每微秒可搜索 1 百万次，则破译 IDEA 密码需要花费 5.4×10^{18} 年。这显然是比较安全的。

9.3 公开密钥密码体制

9.3.1 公开密钥密码体制的特点

公开密钥密码体制的概念是由 Stanford 大学的研究人员 Diffie 与 Hellman 于 1976 年提出的[DIFF76]。所谓**公开密钥密码体制就是使用不同的加密密钥与解密密钥，是一种“由已知加密密钥推导出解密密钥在计算上是不可行的”密码体制。**

公开密钥密码体制的产生主要是因为两个方面的原因，一是由于常规密钥密码体制的**密钥分配**(distribution)问题，另一是由于对**数字签名**的需求。

在常规密钥密码体制中，加解密的双方使用的是相同的密钥。但怎样才能做到这一点呢？一种是事先约定，另一种是用信使来传送。在高度自动化的大型计算机网络中，用信使来传送密钥显然是不合适的。如果事先约定密钥，就会给密钥的管理和更换都带来了极大的不便。若使用高度安全的**密钥分配中心** KDC (Key Distribution Center)，也会使得网络成本增加。

对数字签名的强烈需要也是产生公开密钥密码体制的一个原因。在许多应用中，人们需要对纯数字的电子信息进行签名，表明该信息确实是某个特定的人产生的。

公开密钥密码体制提出不久，人们就找到了三种公开密钥密码体制。目前最著名的是由美国三位科学家 Rivest, Shamir 和 Adleman 于 1976 年提出并在 1978 年正式发表的 **RSA 体制**，它是基于数论中大数分解问题的体制[RIVE78]。

在公开密钥密码体制中，**加密密钥**（即**公开密钥**）PK 是公开信息，而**解密密钥**（即**秘密密钥**）SK 是需要保密的。加密算法 E 和解密算法 D 也都是公开的。虽然秘密密钥 SK **是由公开密钥 PK 决定的，但却不能根据 PK 计算出 SK。**

在继续讨论公开密钥密钥密码体制之前，先要澄清一下几个容易产生的错误概念。第一个错误概念就是认为“公开密钥加密方法要比传统的加密方法更加安全”。实际上，**任何加密方法的安全性取决于密钥的长度，以及攻破密文所需的计算量**。在这方面，公开密钥密码体制并不具有比传统加密体制更加优越之处。第二个错误概念就是认为“公开密钥密码体制是一种通用技术，它已使得传统的密码体制成为陈旧的。”实际上正好相反，由于目前公开密钥加密算法的**开销较大**，在可见的将来还看不出来要放弃传统的加密方法。最后的一个错误概念就是认为“公开密钥的密钥分配实现起来很简单。”实际上，这还**需要密钥分配协议**，具体的分配过程并不比采用传统加密方法时更为简单。

公开密钥算法的特点如下：

（1）发送者用加密密钥 PK 对明文 X 加密后，在接收者用解密密钥 SK 解密，即可恢复出明文，或写为：

$$D_{SK}(E_{PK}(X)) = X \tag{9-5}$$

解密密钥是接收者专用的秘密密钥，对其他人都保密。

此外，加密和解密的运算可以对调，即 $E_{PK}(D_{SK}(X)) = X$。

（2）加密密钥是公开的，但不能用它来解密，即

$$D_{PK}(E_{PK}(X)) \neq X \tag{9-6}$$

（3）在计算机上可以容易地产生成对的 PK 和 SK。

（4）从已知的 PK 实际上不可能推导出 SK，即从 PK 到 SK 是“计算上不可能的”。

（5）加密和解密算法都是公开的。

上述过程如图 9-8 所示。

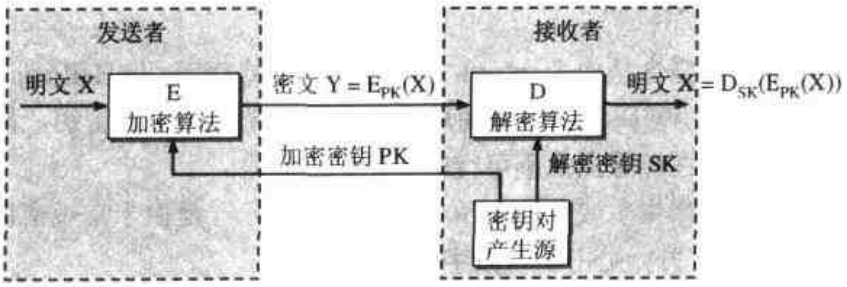


图 9-8　公开密钥密码体制

9.3.2　RSA 公开密钥密码体制

RSA 公开密钥密码体制所根据的原理是：**根据数论，寻求两个大素数比较简单，而将它们的乘积分解开则极其困难**。在这一体制中，每个用户有两个密钥：加密密钥 PK = $\{e, n\}$和解密密钥 SK = $\{d, n\}$。用户把加密密钥公开，使得系统中的任何其他用户都可以使用，而对解密密钥中的 d 则保密。这里，n 为两个大素数 p 和 q 的乘积（素数 p 和 q 一般为 100 位以上的十进数），e 和 d 满足一定的关系（下面将介绍）。当敌手已知 e 和 n 时并不能求出 d。

（1）加密算法

若用整数 X 表示明文，用整数 Y 表示密文（X 和 Y 均小于 n），则加密和解密运算为：

加密：　　$Y = X^e \bmod n$　　（9-7）

解密：　　$X = Y^d \bmod n$　　（9-8）

（2）密钥的产生

现在讨论 RSA 公开密钥密码体制中每个参数是如何选择和计算的。

① 计算 n。用户秘密地选择两个大素数 p 和 q，计算出 $n = pq$。n 称为 RSA 算法的模数。明文必须能够用小于 n 的数来表示。实际上 n 是几百比特长的数。

② 计算 $\phi(n)$。用户再计算出 n 的欧拉函数

$$\phi(n) = (p - 1)(q - 1) \tag{9-9}$$

$\phi(n)$定义为不超过 n 并与 n 互素的数的个数。

③ 选择 e。用户从$[0, \phi(n) - 1]$中选择一个与$\phi(n)$互素的数 e 作为公开的加密指数。

④ 计算 d。用户计算出满足下式的 d

$$ed = 1 \bmod \phi(n) \tag{9-10}$$

作为解密指数。

⑤ 得出所需要的公开密钥和秘密密钥：

公开密钥（即加密密钥）　　PK = $\{e, n\}$

秘密密钥（即解密密钥）　　SK = $\{d, n\}$

（3）正确性的例子说明

限于篇幅，我们这里不去证明RSA加密算法的正确性。但可以用一个简单的例子来说明(9-8）式的解密运算能正确地恢复明文。

设选择了两个素数（这里显然无法选择100位的大素数），$p=7, q=17$。

计算出 $n=pq=7\times17=119$。

计算出$\phi(n)=(p-1)(q-1)=96$。

从[0, 95]中选择一个与96互素的数 e。我们选 $e=5$。然后根据(9-10)式，

$$5d=1 \bmod 96$$

解出 d。不难得出，$d=77$, 因为 $ed=5\times77=385=4\times96+1=1 \bmod 96$。

于是，公开密钥 PK = (e, n) = {5, 119}, 而秘密密钥 SK = {77, 119}。

现在对明文进行加密。首先将明文划分为一个个分组，使得每个明文分组的二进制值不超过 n, 即不超过119。现在设明文 X = 19。

用公开密钥加密时，先计算 $X^e=19^5=2476099$。再除以119，得出商为20807，余数为66。这就是对应于明文19的密文Y的值。

在用秘密密钥 SK = {77, 119}进行解密时，先计算 $Y^d=66^{77}=1.27...\times10^{140}$。再除以119，得出商为 $1.06...\times10^{138}$，余数为19。此余数即解密后应得出的明文X。

以上过程如图9-9所示。

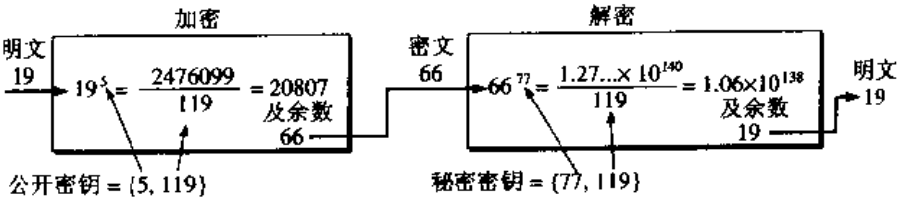


图9-9 RSA算法举例

若选 p 和 q 大于100位十进制数，则 n 大于200位十进制数或大于664位二进制数。这样就可一次对83个字符（每字符8 bit编码）进行加密。我们可以注意到，对于RSA算法，同样的明文映射为同样的密文。

RSA体制的保密性在于对大数进行因数分解很花时间。一个 b 位二进制数 n 的因数分解大约需要的机器周期数为：$\exp\{[\ln(n)\ln(\ln(n))]^{1/2}\}$。若机器周期为1 μs，则 b 分别为100, 200, 300, 500和1000 bit时，分解 $n=2^b$ 所需时间分别为30秒、3天、9年、100万年和 6×10^{15} 年。

RSA算法的三位提出者用129位十进制数字作为其模数 n，并预言要经过 40×10^{15} 年方能攻破。然而最近一个世界范围的研究组在因特网上用1600台协同工作的计算机仅用了8个月就攻破了[LEUT94]。这件事并不是说明RSA是不可靠的，而是告诉我们，在使用RSA加密时，必须选择足够长的密钥。对于当前的计算机水平，一般认为只要选择1024 bit长的密钥（相当于约300位十进制数字）就可认为是无法攻破的。

9.3.3 数字签名

书信或文件是根据亲笔签名或印章来证明其真实性。但在计算机网络中传送的文电又如何盖章呢？这就是数字签名所要解决的问题。数字签名必须保证以下三点：

（1）接收者能够核实发送者对报文的签名。

（2）发送者事后不能抵赖对报文的签名。

（3）接收者不能伪造对报文的签名。

现在已有多种实现数字签名的方法。但采用公开密钥算法要比采用常规密钥算法更容易实现。下面就来介绍这种数字签名。

发送者 A 用其秘密解密密钥 SKA 对报文 X 进行运算，将结果 $D_{SKA}(X)$传送给接收者 B。（读者可能要问：报文 X 还没有加密，怎么能够进行解密呢？其实“解密”仅仅是一个数学运算。发送者此时的运算并非想将报文 X 加密而是为了进行数字签名）。B 用已知的 A 的公开加密密钥得出 $E_{PKA}(D_{SKA}(X)) = X$。因为除 A 外没有别人能具有 A 的解密密钥 SKA，所以除 A 外没有别人能产生密文 $D_{SKA}(X)$。这样，B 就相信报文 X 是 A 签名发送的（图 9-10）。

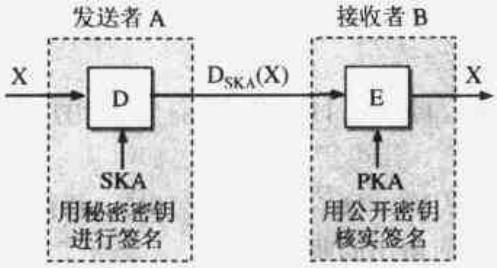


图 9-10　数字签名的实现

若 A 要抵赖曾发送报文给 B，B 可将 X 及 $D_{SKA}(X)$出示给第三者。第三者很容易用 PKA 去证实 A 确实发送 X 给 B。反之，若 B 将 X 伪造成 X'，则 B 不能在第三者前出示 $D_{SKA}(X')$。这样就证明了 B 伪造了报文。可见实现数字签名也同时实现了对报文来源的鉴别。

但上述过程仅对报文进行了签名。对报文 X 本身却未保密。因为截到密文 $D_{SKA}(X)$并知道发送者身份的任何人，通过查阅手册即可获得发送者的公开密钥 PKA，因而能破解电文内容。若采用图 9-11 所示的方法，则可同时实现秘密通信和数字签名。图中 SKA 和 SKB 分别为 A 和 B 的秘密密钥，而 PKA 和 PKB 分别为 A 和 B 的公开密钥。

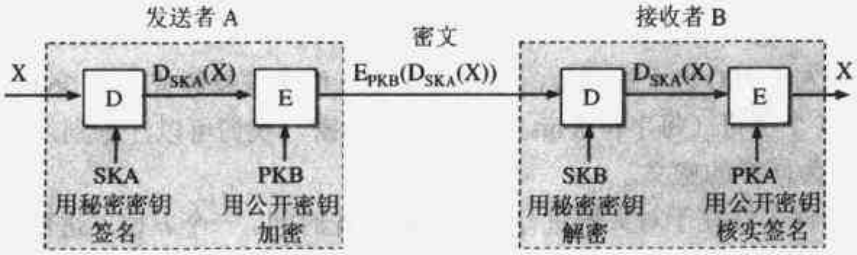


图 9-11　具有保密性的数字签名

9.4　报文鉴别

在信息的安全领域中，对付被动攻击的重要措施是加密，而对付主动攻击中的篡改和伪造则要用**报文鉴别**(message authentication)的方法。报文鉴别就是一种过程，它使得通信的接收方能够验证所收到的报文（发送者和报文内容、发送时间、序列等）的真伪。

使用加密就可达到报文鉴别的目的。但在网络的应用中，许多报文并不需要加密。例如，通知网络上所有的用户有关网络的一些情况。对于不需要加密的报文进行加密和解密，会使计算机增加很多不必要的负担（加密和解密要花费相当多的 CPU 时间）。当我们传送不需要加密的明文时，有时却有这样的要求：应当使接收者能用很简单的方法**鉴别报文的真伪**。

近年来，广泛使用**报文摘要** MD (Message Digest)来进行报文鉴别。发送端将可变长度的

报文 m 经过报文摘要算法运算后得出固定长度的报文摘要 H(m)。然后对 H(m)进行加密，得出 $E_K(H(m))$，并将其追加在报文 m 后面发送出去。接收端将 $E_K(H(m))$解密还原为 H(m)，再将收到的报文进行报文摘要运算，看得出的是否为此 H(m)。如不一样，则可断定收到的报文不是发送端产生的。报文摘要的优点就是：仅对短得多的定长报文摘要 H(m)进行加密比对整个长报文 m 进行加密要简单得多，但对鉴别报文 m 来说，其效果是一样的。也就是说，m 和 $E_K(H(m))$合在一起是不可伪造的，是**可检验的和不可抵赖的**。

报文摘要和以前讲过的循环冗余检验都是**多对一**(many-to-one)的**散列函数**(hash function)的例子。要做到不可伪造，报文摘要算法必须满足以下两个条件：

（1）任给一个报文摘要值 x，若想找到一个报文 y 使得 H(y)= x，则在**计算上是不可行的**。

（2）若想找到任意两个报文 x 和 y，使得 H(x) = H(y)，则在**计算上是不可行的**。

上述的两个条件表明：若(m, H(m))是发送者产生的报文和报文摘要对，则攻击者不可能伪造出另一个报文 y，使得 y 与 x 具有同样的报文摘要。发送者可以对 H(m)进行数字签名，使报文成为可检验的和不可抵赖的。

报文经过散列函数运算可以看成是没有密钥的加密运算。在接收端不需要（也无法）将报文摘要解密还原为明文报文。

图 9-12 表示了报文摘要是怎样使用的。图已经很清楚，不需要再进行解释。

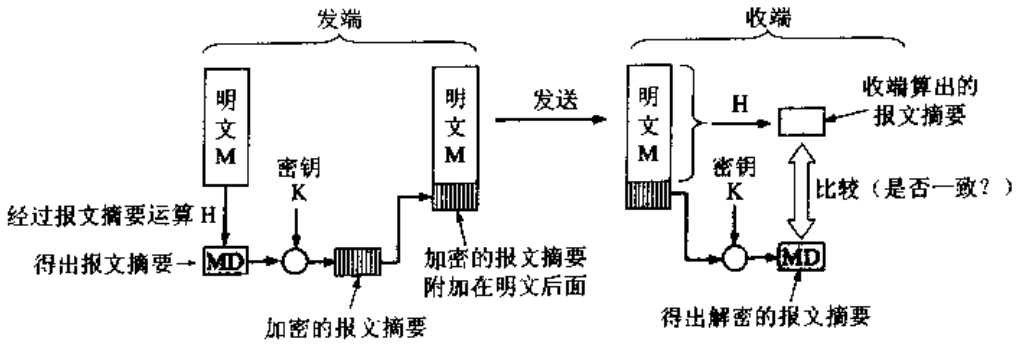


图 9-12　报文摘要是怎样使用的

[RFC 1321]提出的报文摘要算法 MD5 已获得了广泛的应用。它可对任意长的报文进行运算，然后得出 128 bit 的 MD 报文摘要代码。MD5 的算法大致的过程如下：

（1）先将任意长的报文按模 2^{64} 计算其余数(64 bit)，追加在报文的后面。这就是说，最后得出的 MD 代码已包含了报文长度的信息，

（2）在报文和余数之间填充 1～512 bit，使得填充后的总长度是 512 的整数倍。填充比特的首位是 1，后面都是 0。

（3）将追加和填充后的报文分割为一个个 512 bit 的数据块，512bit 的报文数据分成 4 个 128 bit 的数据块依次送到不同的散列函数进行 4 轮计算。每一轮又都按 32 bit 的小数据块进行复杂的运算。一直到最后计算出 MD5 报文摘要代码。

这样得出的 MD5 代码中的每一个比特，都与原来报文中的每一个比特有关。Rivest 提出一个猜想，即根据给定的 MD5 代码找出原来报文的难度，其所需的操作量级为 2^{128}。到目前为止，还没有任何分析可以证明这种猜想是错误的。

MD5 目前已在因特网上大量使用。另一种标准叫做**安全散列算法** SHA (Secure Hash Algorithm)和 MD5 相似，但码长为 160 bit。它也是用 512 bit 长的数据块经过复杂运算得

出的。SHA 比 MD5 更安全，但计算起来却比 MD5 要慢些。新的一个版本 SHA-1 也已制定出来。

9.5 密钥分配

由于密码算法是公开的，网络的安全性就完全基于密钥的安全保护上。因此在密码学中出现了一个重要的分支——**密钥管理**。密钥管理包括：密钥的产生、分配、注入、验证和使用。本节只讨论密钥的分配。

密钥分配是密钥管理中最大的问题。密钥必须通过最安全的通路进行分配。例如，可以派非常可靠的信使携带密钥分配给互相通信的各用户。这种方法称为**网外分配方式**。但随着用户的增多和通信量的增大，密钥更换频繁（密钥必须定期更换才能做到可靠），派信使的办法将不再适用。这时应采用**网内分配方式**，即对密钥自动分配。

目前常用的密钥分配方式是设立**密钥分配中心** KDC (Key Distribution Center)，通过 KDC 来分配密钥。图 9-13 为一种对常规密钥进行分配的方法，我们假定用户 A 和 B 都是 KDC 的注册用户，他们分别拥有与 KDC 通信的主密钥 KA 和 KB。密钥分配分为三个步骤（如图中带箭头直线上的①, ②和③所示）。

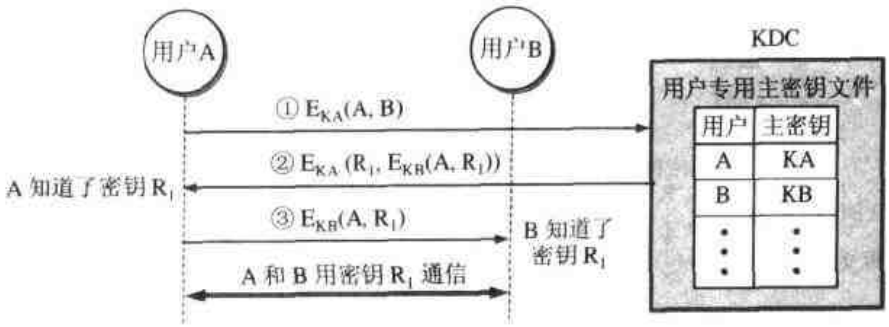


图 9-13 常规密钥分配协议

① 首先，用户 A 向 KDC 发送用自己的密钥 KA 加密的报文 $E_{KA}(A, B)$，说明想和用户 B 通信。

② KDC 用随机数产生一个“一次一密”密钥 R_1 供 A 和 B 这次的通信使用，然后向 A 发送回答报文，这个回答报文用 A 的密钥 KA 加密，报文中有密钥 R_1 和请 A 转给 B 的报文 $E_{KB}(A, R_1)$，但报文 $E_{KB}(A, R_1)$是用 B 的密钥加密的，因此 A 无法知道报文 $E_{KB}(A, R_1)$的内容（A 没有 B 的密钥 KB，也不需要知道此报文的内容）。

③ 当 B 收到 A 转来的报文 $E_{KB}(A, R_1)$并使用自己的密钥 KB 解密后，就知道 A 要和他通信，同时也知道和 A 通信应当使用的密钥 R_1。

此后，A 和 B 就可使用密钥 R_1 进行这次的通信了。

KDC 还可在报文中加入时间戳，以防止报文的截取者利用以前已记录下的报文进行重放攻击。密钥 R_1 是一次性的，因此保密性较高。而 KDC 分配给用户的密钥，如 KA 和 KB，都应定期更换以减少攻击者破译密钥的机会。[RFC 1510]描述了目前最出名的密钥分配协议 Kerberos，是美国麻省理工学院 MIT 开发出的。

在公开密钥体制中，如果每个用户都具有其他用户的公开密钥，就可实现安全通信。看

来好像可以随意公布用户的公开密钥。其实不然。设想用户 A 要欺骗用户 B。A 可以向 B 发送一份伪造是 C 发送的报文。A 用自己的秘密密钥进行数字签名，并附上 A 自己的公开密钥，谎称这公开密钥是 C 的。B 如何知道这个公开密钥不是 C 的呢？显然，这需要有一个值得信赖的机构来将公开密钥与其对应的实体（人或机器）进行**绑定**(binding)。这样的机构就叫作**认证中心 CA** (Certification Authority)，它一般由政府出资建立。每个实体都有 CA 发来的**证书**(certificate)，里面有公开密钥及其拥有者的标识信息（人名或 IP 地址）。此证书被 CA 进行了数字签名。任何用户都可从可信的地方（如代表政府的报纸）获得认证中心 CA 的公开密钥，此公开密钥用来验证某个公开密钥是否为某个实体所拥有（通过向 CA 查询）。有的大公司，如 Netscape，也提供认证中心服务。

9.6 电子邮件的加密

一封电子邮件在传送的过程中可能要经过许多路由器，其中的任何一个路由器都有可能对转发的邮件进行阅读。从这个意义上讲，电子邮件是没有什么隐私可言的。为此，人们研究了如何对电子邮件进行加密。下面要介绍的是两个著名的安全电子邮件系统。

9.6.1 PGP

PGP (Pretty Good Privacy)是 Zimmermann 于 1995 年开发出的。它是一个完整的电子邮件安全软件包，包括加密、鉴别、电子签名和压缩等技术。PGP 并没有使用什么新的概念，它只是将现有的一些算法（如 MD5、RSA，以及 IDEA 等）综合在一起而已。由于包括源程序的整个软件包可以从因特网免费下载[W-PGP1]，因此 PGP 在 MS-DOS/Windows 以及 UNIX 等平台上得到了广泛的应用。但是如果要将 PGP 用于商业，那么还需要到指定网站 http://www.pgpinternational.com/ 获得商用许可证才行。

值得注意的是，虽然 PGP 已被广泛使用，但 PGP 并不是因特网的正式标准。

图 9-14 是 PGP 的工作原理。用户 A 向用户 B 发送一个电子邮件明文 P，用 PGP 进行加密。假定 A 和 B 都有 RSA 的秘密密钥 D_X 和公开密钥 E_X，都有对方的公开密钥。

明文 P 先经过 MD5 运算，再用 RSA 的秘密密钥 D_A 对报文摘要 MD5 进行加密，得出 H。明文 P 和 RSA 的输出 H 拼接在一起，成为另一个报文 P1。经 ZIP 程序压缩后，得出 P1.Z。

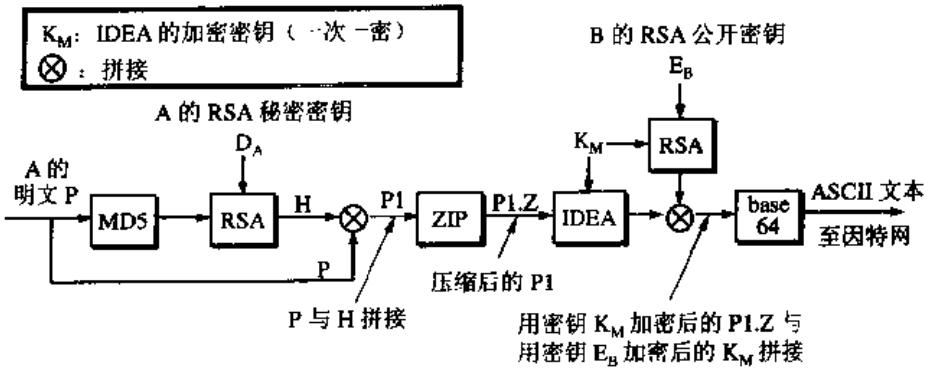


图 9-14 PGP 的加密过程

下一步是对 P1.Z 进行 IDEA 加密，使用的是一次一密的加密密钥，即 128 bit 的 K_M。此外，密钥 K_M 再经过 RSA 加密，其密钥是 B 的公开密钥 D_B。加密后的 K_M 与加密后的 P1.Z 拼接在一起，用 base64 进行编码，然后得出 ASCII 码的文本（只包含 52 个字母、10 个数字和 3 个符号+, /, =）发送到因特网上。

用户 B 收到加密的邮件后，先进行 base64 解码，并用其 RSA 秘密密钥解出 IDEA 的密钥。用此密钥恢复出 P1.Z。对 P1.Z 进行解压后，还原出 P1。B 接着分开明文 P 和加了密的 MD5，并用 A 的公开密钥解出 MD5。若与 B 自己算出的 MD5 一致，则可认为 P 是从 A 发来的邮件。

从图 9-14 可看出，在两个地方使用了 RSA：对 128 bit 的 MD5 加密和对 128 bit 的 IDEA 密钥加密。虽然 RSA 的运算很慢，但这里只对数量不大的 256 bit 进行加密。

PGP 支持三种 RSA 密钥长度：384 bit（偶尔使用），512 bit（商业用）和 1024 bit（军用）。

PGP 很难被攻破。根据计算，仅破译其中的 RSA 部分（密钥为 1024 bit 长，使用 1000 MIPS 的计算机）就需要 3 亿年[W-PGP2]。因此在目前可以认为 PGP 是足够安全的。

PGP 的报文格式如图 9-15 所示。可以看出，PGP 报文由三个部分组成，即报文的 IDEA 密钥部分，签字部分，以及报文本身。密钥部分不仅是密钥，而且还有密钥的标识符，因为用户可能拥有多个公开密钥。

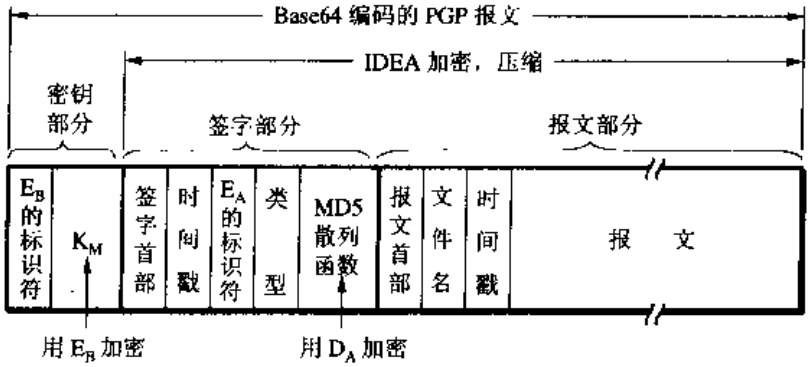


图 9-15 PGP 的报文格式

签字部分从一个首部开始，接下去是一个时间戳，然后是发信人的公开密钥（用于对 MD5 签名）。再后面的类型标识所使用的加密算法。

报文部分也包括一个首部。文件名是当收信人将信件存盘时使用的默认文件名。在时间戳后面才是报文本身。

密钥管理是 PGP 系统的一个关键。每个用户在其所在地要维持两个数据结构：**秘密密钥环**(private key ring)和**公开密钥环**(public key ring)。秘密密钥环包括一个或几个用户自己的秘密密钥-公开密钥对。这样做是为了使用户可经常更换自己的密钥。每一对密钥有对应的标识符。发信人将此标识符通知收信人，使收信人知道应当用哪一个公开密钥进行解密。公开密钥环包括用户的一些经常通信对象的公开密钥。

9.6.2 PEM

PEM (Privacy Enhanced Mail)是因特网的邮件加密建议标准，由 4 个 RFC 文档来描述：

（1）RFC 1421：报文加密与鉴别过程。

（2）RFC 1422：基于证书的密钥管理。

（3）RFC 1423：PEM 的算法、工作方式和标识符。

（4）RFC 1424：密钥证书和相关的服务。

PEM 的功能和 PGP 的差不多，都是对基于[RFC 822]的电子邮件进行加密和鉴别。

报文在使用 PEM 之前先要经过规范形式的处理（即对空格、制表符、回车、换行等的处理）。接着使用 MD5 得出报文摘要，与报文拼接在一起后，用 DES 加密。加密后的报文可再用 base64 编码，然后发送给收信人。

和 PGP 相似，每个报文都是使用一次一密的方法进行加密，并且密钥也是放在报文中一起在网络上传送。当然对密钥还必须加密。可以使用 RSA 或三重 DES。实际上大多数人愿意使用 RSA。

PEM 有比 PGP 更加完善的密钥管理机制。由**认证中心**(Certificate Authority)发布证书，上面有用户姓名、公开密钥以及密钥的使用期限。每个证书有一个惟一的序号。证书还包括用证书管理机构的秘密密钥签了名的 MD5 散列函数。这种证书与 ITU-T X.509 关于公开密钥证书的建议书以及 X.400 的名字体系相符合。

PGP 也有类似的密钥管理机制（但 PGP 没有使用 X.509）。问题是：用户是否信任这种证书管理机构？PEM 对这个问题解决得较好，它用的方法是设立一些政策认证管理机构 PCA (Policy Certification Authority)来证明这些证书，然后由因特网政策登记管理机构 IPRA (Internet Policy Registration Authority)对这些 PCA 进行认证。

9.7 链路加密与端到端加密

从网络传输的角度看，通常有两种不同的加密策略，即链路加密与端到端加密。现分别讨论如下：

9.7.1 链路加密

在采用链路加密的网络中，每条通信链路上的加密是独立实现的。通常对每条链路使用不同的加密密钥（图 9-16，图中的 E 和 D 分别表示加密和解密）。当某条链路受到破坏时不会导致其他链路上传送的信息被析出。加密算法常采用序列密码。由于 PDU 中的协议控制信息和数据都被加密，这就掩盖了源结点和目的结点的地址。若在结点间保持连续的密文序列，则 PDU 的频度和长度也能得到掩盖。这样就能防止各种形式的通信量分析。由于不需要传送额外的数据，采用这种技术不会减少网络的有效带宽。由于只要求相邻结点之间具有相同的密钥，因而密钥管理易于实现。链路加密对用户来说是透明的，因为加密的功能是由通信子网提供的。

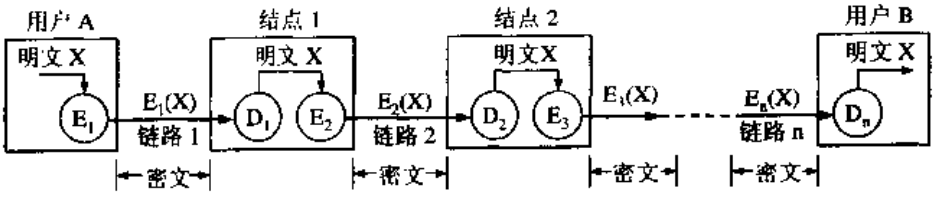


图 9-16 链路加密

由于报文是以明文形式在各结点内加密的，所以结点本身必须是安全的。一般认为网络的源结点和目的结点在物理上都是安全的，但所有的中间结点（包括可能经过的路由器）则未必都是安全的。因此必须采取有效措施。对于采用动态自适应路由的网络，一个被攻击者掌握的结点可以设法更改路由使有意义的 PDU 经过此结点。这样将导致大量信息的泄露，因而对整个网络的安全造成威胁。

链路加密的最大缺点是在中间结点暴露了信息的内容。在网络互连的情况下，仅采用链路加密是不能实现通信安全的。此外，链路加密也不适用于广播网络，因为它的通信子网没有明确的链路存在。若将整个 PDU 加密将造成无法确定接收者和发送者。由于上述原因，除非采取其他措施，否则在网络环境中链路加密将受到很大的限制，可能只适用于局部数据的保护。

9.7.2 端到端加密

端到端加密是在源结点和目的结点中对传送的 PDU 进行加密和解密，其过程如图 9-17 所示。可以看出，报文的安全性不会因中间结点的不可靠而受到影响。

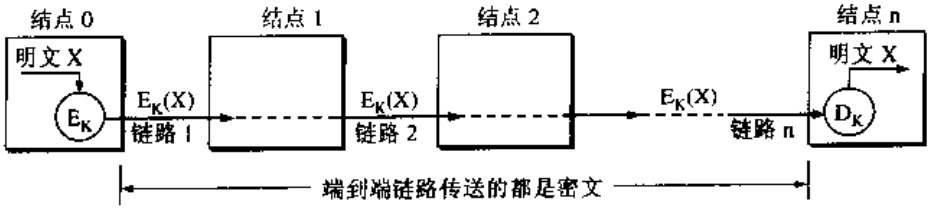


图 9-17 端到端加密

端到端加密应在运输层或其以上各层来实现。若选择在运输层进行加密，可以使安全措施对用户来说是透明的。这样可不必为每一个用户提供单独的安全保护，但容易遭受运输层以上的攻击。当选择在应用层实现加密时，用户可根据自己的特殊要求来选择不同的加密算法，而不会影响其他用户。这样，端到端加密更容易适合不同用户的要求。端到端加密不仅适用于互连网环境，而且同样也适用于广播网。

在端到端加密的情况下，PDU 的控制信息部分（如源结点地址、目的结点地址、路由信息等）不能被加密，否则中间结点就不能正确选择路由。这就使得这种方法易于受到通信量分析的攻击。虽然也可以通过发送一些假的 PDU 来掩盖有意义的报文流动（这称为报文填充），但这要以降低网络性能为代价。由于各结点必须持有与其他结点相同的密钥，这就需要在全网范围内进行密钥管理和分配。

为了获得更好的安全性，可将链路加密与端到端加密结合在一起使用。链路加密用来对 PDU 的目的地址 B 进行加密，而端到端加密则提供了对端到端的数据进行保护。

9.8 因特网商务中的加密

因特网商务就是通过因特网来进行商务活动，如购物、订票、股票交易等。近年来，在商务安全方面较出名的有两个协议，即已在许多因特网交易中使用的**安全插口层 SSL** (Secure Socket Layer)和有很强竞争潜力的**安全电子交易 SET** (Secure Electronic Transaction)。本节内容主要取自[KURO01]。

9.8.1 安全插口层 SSL

SSL 又称为**安全套接层**，是 Netscape 公司开发的协议，可对万维网客户与服务器之间传送的数据进行加密和鉴别。它在双方的联络阶段协商将要使用的加密算法（如用 DES 或 RSA）和密钥，以及客户与服务器之间的鉴别。在联络阶段完成之后，所有传送的数据都使用在联络阶段商定的会话密钥。SSL 不仅被所有常用的浏览器和万维网服务器所支持，而且也是**运输层安全协议** TLS (Transport Layer Security)的基础[RFC 2246]。

SSL 和 TLS 并不仅限于万维网的应用，它们还可用于 IMAP 邮件存取的鉴别和数据加密。SSL 可看成是在应用层和运输层之间的一个层（图 9-18）。在发送方，SSL 接收应用层的数据（如 HTTP 或 IMAP 报文），对数据进行加密，然后将加了密的数据送往 TCP 插口。在接收方，SSL 从 TCP 插口读取数据，解密后将数据交给应用层。



图 9-18 安全插口层 SSL 的位置

SSL 提供以下三个功能：

（1）**SSL 服务器鉴别** 允许用户证实服务器的身份。具有 SSL 功能的浏览器维持一个表，上面有一些可信赖的认证中心 CA 和它们的公开密钥。当浏览器要和一个具有 SSL 功能的服务器进行商务活动时，浏览器就从服务器得到含有服务器的公开密钥的证书。此证书是由某个认证中心 CA 发出的（此 CA 在客户的表中）。这就使得客户在提交其信用卡之前能够鉴别服务器的身份。

（2）**加密的 SSL 会话** 客户和服务器交互的所有数据都在发送方加密，在接收方解密。SSL 还有检测攻击者有无窃听传送的数据的功能。

（3）**SSL 客户鉴别** 允许服务器证实客户的身份。这个信息对服务器是重要的。例如，当银行将保密的有关财务信息发送某顾客时，就必须检验接收者的身份。

下面通过一个简单的例子说明 SSL 的工作原理。

假定 A 有一个使用 SSL 的安全网页。B 上网时用鼠标点击到这个安全网页的链接（这种安全网页的 URL 的协议部分不是 http 而是 https）。接着，服务器和浏览器就进行握手协议，其主要过程如下：

（1）浏览器向服务器发送浏览器的 SSL 版本号和密码编码的参数选择(preference)（因为浏览器和服务器要协商使用哪一种对称密钥算法）。

（2）服务器向浏览器发送服务器的 SSL 版本号、密码编码的参数选择及服务器的证书。证书包括服务器的 RSA 公开密钥。此证书用某个认证中心的秘密密钥加密。

（3）浏览器有一个可信赖的 CA 表，表中有每一个 CA 的公开密钥。当浏览器收到服务器发来的证书时，就检查此证书是否在自己的可信赖的 CA 表中。如不在，则后面的加密和鉴别连接就不能进行下去。如在，浏览器就使用 CA 的公开密钥对证书解密，这样就得到了服务器的公开密钥。

（4）浏览器随机地产生一个对称会话密钥，并用服务器的公开密钥加密，然后将加密的会话密钥发送给服务器。

（5）浏览器向服务器发送一个报文，说明以后浏览器将使用此会话密钥进行加密。然后浏览器再向服务器发送一个单独的加密报文，表明浏览器端的握手过程已经完成。

（6）服务器也向浏览器发送一个报文，说明以后服务器将使用此会话密钥进行加密。然后服务器再向浏览器发送一个单独的加密报文，表明服务器端的握手过程已经完成。

（7）SSL 的握手过程至此已经完成，下面就可开始 SSL 的会话过程。浏览器和服务器都使用这个会话密钥对所发送的报文进行加密。

由于 SSL 简单且开发得较早，因此目前在因特网商务中使用得比较广泛。但 SSL 并非专门为信用卡交易而设计的，它只是在客户与服务器之间提供了一般的安全通信。SSL 还缺少一些措施来防止在因特网商务中出现各种可能的欺骗行为。

9.8.2 安全电子交易 SET

安全电子交易 SET 则是专为在因特网上进行安全信用卡交易的协议。它最初是由两个著名信用卡公司 Visa 和 MasterCard 于 1996 年开发的，世界上许多具有领先技术的公司也参与了。1997 年底成立了实体 SETCo，目的是在全球推广使用 SET。

SET 的主要特点是：

（1）SET 是**专为与支付有关的报文进行加密的**，它不能像 SSL 那样对任意的数据（如正文或图象）进行加密。

（2）SET 协议涉及到**三方**，即**顾客、商家和商业银行**。所有在这三方之间交互的敏感信息都被加密。

（3）SET 要求这三方都有证书。在 SET 交易中，**商家看不见顾客传送给商业银行的信用卡号码**。这是 SET 的一个最关键的特性。

在一个 SET 交易中要使用 3 个软件，即

（1）**浏览器钱包**。这个软件集成在浏览器中，它给顾客在购物时提供信用卡和证书的存储和管理的地方，并响应从商家发来的 SET 报文，提示顾客选择信用卡进行支付。

（2）**商家服务器**。这是在万维网上提供商品交易的实现引擎。它处理持卡人的交易，并与商业银行通信。

（3）**支付网关**(acquirer gateway)，是商业银行使用的软件，处理信用卡的交易，包括授权和支付，是个相当复杂的软件。

下面以顾客 B 到公司 A 用 SET 购买物品为例来说明 SET 的工作原理。这里涉及到两个银行，即 A 的银行（公司 A 的支付银行）和 B 的银行（给 B 发出信用卡的银行）。

（1）B 告诉 A 他想用信用卡购买公司 A 的物品。

（2）A 将物品清单和一个惟一的交易标识符发送给 B。

（3）A 将其商家的证书，包括商家的公开密钥发送给 B。A 还向 B 发送其银行的证书，包括银行的公开密钥。这两个证书都用一个认证中心 CA 的秘密密钥进行加密。

（4）B 使用认证中心 CA 的公开密钥对这两个证书解密。于是 B 有了 A 的公开密钥和 A 的银行的公开密钥。

（5）B 生成两个数据包：给 A 用的**定货信息** OI (Order Information)和给 A 的银行用的**购买指令** PI (Purchase Instruction)。OI 包括交易标识符和将要使用的信用卡的类别，但**不包含** B 的信用卡号码。PI 则包括交易标识符、B 的信用卡号码以及 B 同意向 A 付出的款数。OI 用 A 的公开密钥加密，而 PI 用 A 的银行的公开密钥加密。B 将加密后的 OI 和 PI 发送给 A。请注意，PI 虽然是给 A 的银行用的，但并不是由 B 直接发送给 A 的银行。

（6）A 生成对信用卡**支付请求**(payment request)的**授权请求**(authorization request)，它包括

交易标识符。

（7）A 用银行的公开密钥将一个报文加密发送给银行，此报文包括授权请求、从 B 发过来的 PI 数据包，以及 A 的证书。

（8）A 的银行收到此报文，将其解密。A 的银行要检查此报文有无被篡改，以及检查在授权请求中的交易标识符是否与 B 的 PI 数据包给出的一致。

（9）A 的银行通过传统的银行信用卡信道向 B 的银行发送请求支付授权的报文。

（10）一旦 B 的银行准许支付，A 的银行就像 A 发送响应（加密的）。此响应包括交易标识符。

（11）若此次交易被批准，A 就向 B 发送响应报文。此报文作为收据，并通知 B：“支付已被接受，所购物品即将发出”。

SET 的特点中的很重要的一点就是购物人的信用卡号码不向商家暴露。我们注意到在上面的第(5)项中，B 使用银行的密钥对其信用卡号码加密。

9.9 因特网的网络层安全协议族 IPsec

1. IPsec 与安全关联 SA

1998 年 11 月公布了因特网网络层安全的系列 RFC [RFC 2401～1411][W-IPsec]。其中最重要的就是描述 IP 安全体系结构的[RFC 2401]和提供 IPsec 协议族概述的[RFC 2411]。IPsec 就是“IP 安全(Security)协议”的缩写。

网络层保密是指所有在 IP 数据报中的数据都是加密的。此外，网络层还应提供**源站鉴别**(source authentication)，即当目的站收到 IP 数据报时，能确信这是从该数据报的源 IP 地址的主机发来的。在 IPsec 中最主要的两个部分就是：**鉴别首部 AH** (Authentication Header)和**封装安全有效载荷 ESP** (Encapsulation Security Payload)。AH 提供源站鉴别和数据完整性，但不能保密。而 ESP 比 AH 复杂得多，它提供源站鉴别、数据完整性和保密。

在使用 AH 或 ESP 之前，先要从源主机到目的主机建立一条网络层的逻辑连接。此逻辑连接叫做**安全关联 SA** (Security Association)。这样，**IPsec 就将传统的因特网无连接的网络层转换为具有逻辑连接的层**。安全关联是一个**单向连接**。如进行双向的安全通信则需要建立两个安全关联。一个安全关联 SA 由一个三元组惟一地确定，它包括：

（1）安全协议（使用 AH 或 ESP）的标识符。

（2）此单向连接的源 IP 地址。

（3）一个 32 bit 的连接标识符，称为**安全参数索引 SPI** (Security Parameter Index)。

对于一个给定的安全关联 SA，每一个 IPsec 数据报都有一个存放 SPI 的字段。通过此 SA 的所有数据报都使用同样的 SPI 值。

2. 鉴别首部 AH

在使用鉴别首部 AH 时，将 AH 首部插在原数据报数据部分的前面，同时将 IP 首部中的协议字段置为 51（图 9-19）。此字段原来是为了区分在数据部分是何种协议（如 TCP, UDP 或 ICMP）。在传输过程中，中间的路由器都不查看 AH 首部。当数据报到达目的站时，目的站主机才处理 AH 字段，以鉴别源主机和检查数据报的完整性[RFC 2402]。

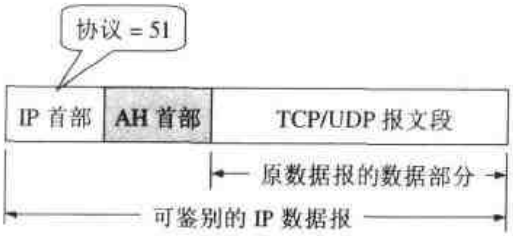


图 9-19　AH 首部的安全数据报中的位置

AH 首部具有如下的一些字段：

（1）**下一个首部**(8 bit)。标志紧接着本首部的下一个首部的类型（如 TCP 或 UDP）。

（2）**有效载荷长度**(8 bit)。即鉴别数据字段的长度，以 32 bit 字为单位。

（3）**安全参数索引** SPI (32 bit)。标志一个安全关联。

（4）**序号**(32 bit)。鉴别数据字段的长度，以 32 bit 字为单位。

（5）**保留**(16 bit)。为今后用。

（6）**鉴别数据**（可变）。为 32 bit 字的整数倍，它包含了经数字签名的报文摘要（对原来的数据报进行报文摘要运算）。因此可用来鉴别源主机和检查 IP 数据报的完整性。

3．封装安全有效载荷 ESP

在 ESP 首部中，有标识一个安全关联的**安全参数索引** SPI (32 bit)和**序号**(32 bit)。在 ESP 尾部中有**下一个首部**(8 bit，作用和 AH 首部的一样)。ESP 尾部和原来数据报的数据部分一起进行加密（图 9-20），因此攻击者无法得知所使用的运输层协议。ESP 的**鉴别数据**和 AH 中的鉴别数据是一样的。因此，用 ESP 封装的数据报既有鉴别源站和检查数据报完整性的功能，又能提供保密。

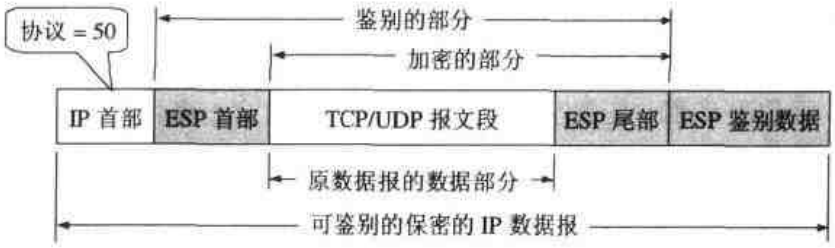


图 9-20　在 IP 数据报中的 ESP 的各字段

9.10　防火墙

防火墙(firewall)是由软件、硬件构成的系统，用来在两个网络之间实施接入控制策略。这里特别需要注意的是，这个接入控制策略是由使用防火墙的单位自行制订的。这种安全策略应当最适合本单位的需要。图 9-21 是防火墙在互连的网络中的位置。一般都将防火墙内的网络称为“**可信赖的网络**”(trusted network)，而将外部的因特网称为“**不可信赖的网络**”(untrusted network)。防火墙可用来解决内联网和外联网的安全问题。

防火墙的功能有两个：一个是**阻止**，另一个是**允许**。“阻止”就是阻止某种类型的通信量通过防火墙（从外部网络到内部网络，或反过来）。“允许”的功能与“阻止”恰好相反。

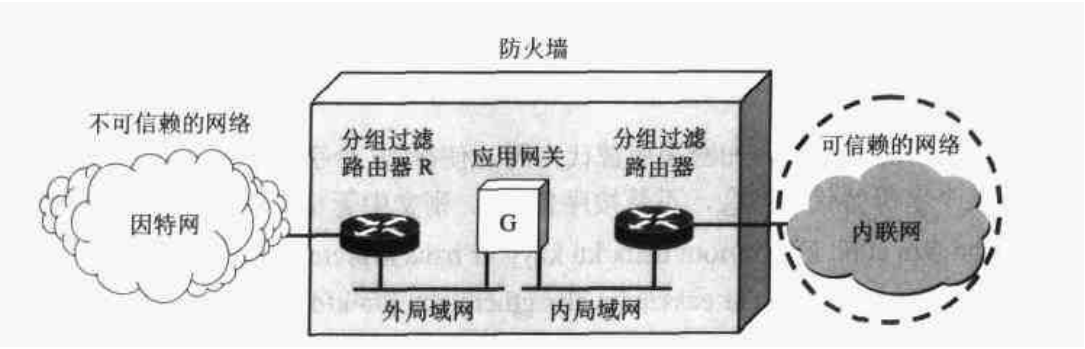


图 9-21　防火墙在互连网络中的位置

可见防火墙必须能够识别通信量的各种类型。不过在大多数情况下防火墙的主要功能是“阻止”。

但是，“绝对阻止所不希望的通信”和“绝对防止信息泄露”一样，是很难做到的。简单地购买一个商用的防火墙往往不能得到所需要的保护，但正确地使用防火墙则可将安全风险降低到可接受的水平。

防火墙技术一般分为两类，即：

（1）**网络级防火墙**　主要是用来防止整个网络出现外来非法的入侵。属于这类的有**分组过滤**(packet filtering)和**授权服务器**(authorization server)。前者检查所有流入本网络的信息，然后拒绝不符合事先制订好的一套准则的数据，而后者则是检查用户的登录是否合法。

（2）**应用级防火墙**　从应用程序来进行接入控制。通常使用**应用网关或代理服务器**(proxy server)来区分各种应用。例如，可以只允许通过访问万维网的应用，而阻止 FTP 应用的通过。

图 9-21 所画的防火墙就同时具有这两种技术。它包括两个分组过滤路由器和一个应用网关，它们通过两个局域网连接在一起。

这两个分组过滤路由器都是标准的路由器，但增加了一些功能，这就是对每一个通过的分组进行检查。这两个路由器中的一个专门检查进入内联网的分组，而另一个则检查出去的。符合条件的分组就能通过，否则就丢弃。使用两个局域网的原因就是使穿过防火墙的各种分组必须经过分组过滤路由器和应用网关的检查，而没有任何其他的路径。

分组过滤是靠查找系统管理员所设置的表格来实现的。表格列出了可接受的、或必须进行阻挡的目的站和源站，以及其他的一些通过防火墙的规则。

我们知道，TCP 的端口号指出了在 TCP 上面的应用层服务。例如，端口号 23 是 TELNET，端口号 119 是 USENET，等等。所以如果在因特网进入防火墙的分组过滤路由器中将所有目的端口号为 23 的入**分组**(incoming packet)都进行阻拦，那么所有外单位用户就不能使用 TELNET 登录到本单位的主机上。同理，如果某公司不愿意其雇员在上班时花费大量时间去看因特网的 USENET 新闻，就可将目的端口号为 119 的**出分组**(outgoing packet)阻拦住，使其无法发送到因特网。

阻拦出分组要麻烦些，因为有时它们不使用标准的端口号。例如 FTP 常常是动态地分配端口号。阻拦 UDP 更困难，因为事先不容易知道 UDP 想做什么。许多分组过滤路由器干脆将所有的 UDP 全部阻拦。

应用网关是从应用层的角度来检查每一个分组。例如，一个邮件网关在检查每一个邮件时，要根据邮件的首部或报文的大小，甚至是报文的内容（例如，有没有某些像“导弹”“核弹头”等关键词）来确定该邮件能否通过防火墙。

习题

9-01 试破译下面的密文诗。加密采用替代密码，使得 26 个字母（从 a 到 z）中的每一个用其他某个字母替代（注意，不是按序替代）。密文中无标点符号。空格未加密。

kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm ur mfudm zhx
mftnm zhx mdzythc pzq ur ezsszcdm zhx gthcm zhx pfa kfd mdz tm sutythc
fuk zhx pfdkfdi ntcm fzld pthcm sok pztk z stk kfd uamkdim eitdx sdruid
pd fzld uoi efzk rui mubd ur om zid uok ur sidzkf zhx zyy ur om zid rzk
hu foiia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk

9-02 下面一段密文本来是连续的字串，只是为了便于阅读将它分成每五个一组。明文是一般计算机教科书中的一段话，因此也许有“computer”这个字出现。加密采用的是置换密码，明文中无空格，无标点符号。试破译之。

aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo gorch srsoc nntii
imiha oofpa gsivt tpsit lbolr otoex

9-03 仔细阅读 DES 的细节，弄清加密过程中的每一个步骤。

9-04 常规密钥体制与公开密钥体制的特点各如何？各有何优缺点？

9-05 密钥分配存在哪些问题？试举出一种密钥分配的方法。

9-06 链路加密与端到端加密各有何特点？各用在什么场合？

9-07 对网络安全的威胁都有哪些？有哪些安全措施？

9-08 采用 DES 加密算法和加密分组链接的方法。在传输过程中，某一个密文分组 C_i 中的一个 0 变成了 1。试问：在对应的明文中会出现多少个错误？

9-09 在上题中，若不是一个 0 变成了 1 而是在 C_i 中多出了一个 0。试分析明文中会出现什么样的错误？

9-10 试述国际数据加密算法 IDEA 的加密过程。

9-11 使用 RSA 公开密钥体制进行加密。

（1）若 $p=7$ 而 $q=11$，试列出 5 个有效的 e。

（2）若 $p=13$，$q=31$，而 $e=7$，问 d 是多少？

（3）若 $p=5$，$q=11$，而 $d=27$，试求 e。设 26 个英文字母可用其字母序号来代替，如 a = 1, b = 2, …, z = 26。试将“abcdefghij”进行加密。

（4）若 $p=5$ 而 $q=11$，而 $e=7$，试求 d，并将报文“RSA”进行加密（报文中的字母用其字母序号代替），然后再解密，看是否能够恢复出原来的明文。

提示：可利用以下公式：

$$(a \times b) \bmod n = ((a \bmod n) \times (b \bmod n)) \bmod n$$

9-12 试述数字签名的原理。

9-13 电子邮件可以用什么方法进行加密？

9-14 为什么需要进行报文鉴别？报文的保密性与完整性有何区别？什么是 MD5？

9-15 试简述 SSL 和 SET 的工作过程。

9-16 因特网的网络层安全协议族 IPsec 都包含哪些主要内容？

9-17 试述防火墙的工作原理和所提供的功能。什么叫做网络级防火墙和应用级防火墙？

第 10 章　因特网的演进

10.1　概述

计算机网络最初是为传送数据信息设计的。传统因特网认为**所有的主机都是平等的**，这符合当时的实际情况。因特网 IP 层提供的“**尽最大努力交付**”服务以及**每一个分组独立选择路由**的策略，对传送数据信息也是很合适的。当我们从因特网下载文件时，过长的网络响应时间虽然令人颇为烦恼，但这至少不会对我们产生有害的结果。下载文件的数据传输显然**不能容忍差错或丢失出现**，否则我们得到的文件将没有什么用处。因特网使用的 TCP 协议可以很好地解决网络不能提供可靠交付这一问题。

然而技术的进步使许多主机对所传送的信息有了不同的要求。一些用户开始利用因特网传送多媒体信息。使用电路交换的公用电话网来传送多媒体信息早已是相当成熟的技术。例如视频会议（又称为电视会议）一般都是使用电路交换的公用电话网。使用电路交换的好处是：一旦连接建立了，经过压缩处理的多媒体信息在电话线路上的**传输质量有保证**。美中不足的是向电信公司租用电话线路的**价格太高**。

于是人们想到了因特网：能否利用现在的因特网以低廉的价格来传送多媒体信息呢？

多媒体信息（包括声音和图像信息）与不包括声音和图像的数据信息有很大的区别，其中最主要的两个特点如下。

第一，多媒体信息的信息量往往很大。

含有音频或视频的多媒体信息的信息量一般都很大，下面是简单的说明。

对于打电话的声音信息，如采用标准的 PCM 编码（用 8 kHz 的速率采样），而每一个采样脉冲用 8 bit 编码，则得出的声音信号的数据率就是 64 kb/s，已经大于目前常用的调制解调器所支持的 56 kb/s 速率。对于高质量的立体声音乐 CD 信息，虽然它也使用 PCM 编码，但其采样速率为 44.1 kHz，而每一个采样脉冲用 16 bit 编码，因此这种双声道立体声音乐信号的数据率超过了 1.4 Mb/s。

再看一下数码相机照出的照片。假定分辨率设置为 1280×960（这只是中等质量的照片），于是一张照片的像素数是 1 228 800 个。若每个像素用 24 bit 进行编码，则一张照片需要 29 491 200 bit。换算成字节数，约合 3.52 MB（这里 1 B = 8 bit，1 M = 2^{20}）。

活动图像的信息量就更大。例如要传送不压缩的彩色电视信号，则数据率超过 250 Mb/s。

因此在网络上传送多媒体信息都无例外地采用各种信息压缩技术。例如在话音压缩方面的标准有：移动通信的 GSM (13 kb/s)，IP 电话使用的 G.729 (8 kb/s)和 G.723.1 (6.4 kb/s 和 5.3 kb/s)。在立体声音乐的压缩技术有 MP3 (128 kb/s 或 112 kb/s)。在视频信号方面有：VCD 质量的 MPEG 1 (1.5 Mb/s)和 DVD 质量的 MPEG 2 (3 Mb/s～6 Mb/s)。由于多媒体信息压缩技术本身还不是计算机网络技术，限于篇幅，本书将不讨论有关数据压缩方面的内容。

第二，在传输多媒体数据时，对时延和时延抖动均有较高的要求。

多媒体数据往往是**实时数据**(real time data)，它的含义是：在发送实时数据的同时，在接

收端**边接收边播放**。而**非实时数据**则是先把全部音频或视频数据接收和存储完毕后再进行播放。

我们知道，模拟的多媒体信号只有经过数字化后才能在因特网上传送。这就是要经过采样和模数转换变为数字信号，然后将一定数量的比特组装成分组。这些分组在发送时的时间间隔都是**恒定的**，通常称这样的分组为**等时的**(isochronous)。这种等时分组进入因特网的速率也是恒定的。但传统的**因特网本身是非等时的**。这是因为在使用 IP 协议的因特网中，每一个分组是独立地选择路由，因而这些分组在接收端的到达速率一般都会变成**非恒定的**。如果我们在接收端对这些以非恒定速率到达的分组边接收边还原，那么就一定会产生很大的失真。图 10-1 说明了因特网是非等时的这一特点。

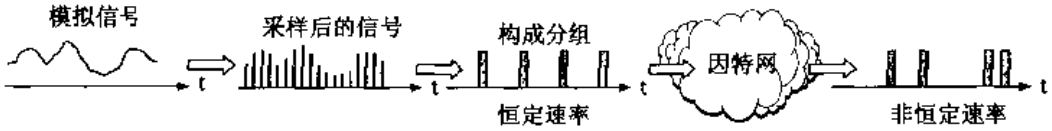


图 10-1　因特网是非等时的

要解决这一问题，通常是在接收端设置适当大小的缓存，当缓存中的分组数达到一定的数量后再以恒定速率按顺序将这些分组读出进行还原播放。图 10-2 说明了缓存的作用。

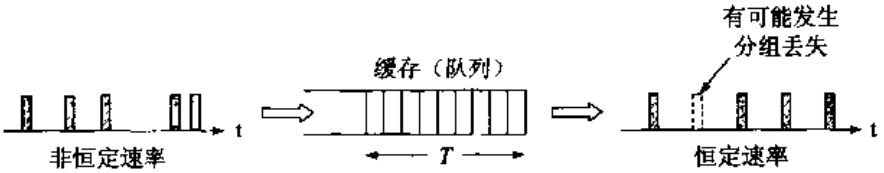


图 10-2　缓存将非恒定速率的分组变换为恒定速率的

从图 10-2 可看出，缓存实际上就是一个先进先出的队列。图中标明的 T 叫做**播放时延**，这就是从最初的分组开始到达缓存算起，经过时间 T 后就按固定时间间隔将缓存中的分组按先后顺序依次读出，然后就还原成原来的多媒体通信号。我们看到，缓存使所有到达的分组都经受了迟延。如果所有的分组都是以恒定速率到达，那么缓存仅仅是使接收端的播放时间推迟了时间 T。但实际上分组是以非恒定速率到达，因此早到达的分组在缓存中停留的时间较长，而晚到达的分组在缓存中停留的时间就较短。由于从缓存中取出分组是按照固定的时钟节拍进行，因此，以非恒定速率到达的分组，经过缓存后再以恒定速率读出，就能够在一定程度上**消除了时延的抖动**。但我们付出的代价是增加了时延。

然而我们还有一些问题没有讨论。

首先，时间 T 应当选为多大？时间 T 选择得越大，就可以消除更大的时延抖动，但所有分组经受的平均时延也增大了，而这对某些实时应用（如视频会议）是很不利的。当然这对单向传输的视频节目问题并不太大（如从网上下载一段视频节目，只要耐心多等待一段时间用来将分组放入缓存即可）。如果时间 T 选择得太小，那么消除时延抖动的效果就较差。因此时间 T 的选择必须折中考虑。在传送**时延敏感**(delay sensitive)的实时数据时，不仅**传输时延不能太大**，而且**时延抖动也必须受到限制**。

其次，在因特网上传输实时数据的分组时有可能会出现差错或甚至丢失。如果利用 TCP 协议对这些出错或丢失的分组进行重传，那么时延就会大大增加。因此实时数据的传输在运

输层就采用 UDP 协议而不使用 TCP 协议。这就是说，对于传送实时数据，**我们宁可丢失少量分组**（当然不能丢失太多），**也不要太晚到达的分组**。在连续的音频或视频数据流中，很少量分组的丢失对播放效果的影响并不大（因为这是由人来进行主观评价的），因而是可以容忍的。**丢失容忍**(loss tolerant)也是实时数据的另一个重要特点。

由于分组的到达可能不按序，但将分组还原和播放时又应当是按序的。因此在发送多媒体分组时还应当给每一个分组加上**序号**。这表明还应当有相应的协议支持才行。

还有一种情况，就是要使接收端能够将节目中本来就存在的正常的短时间停顿（如音乐中停顿几拍）和因某些分组的较大迟延造成的“停顿”区分开来。这就需要增加一个**时间戳**(timestamp)，以便告诉接收端应当在什么时间播放哪个分组。

因此我们必须解决这样的问题：若想在因特网上传送实时多媒体信息，应当如何改造现有的因特网使它能够适应多媒体信息的传送？

对这个问题网络界一直有较大的争论，众说纷纭。有人认为，只要大量使用光缆，网络的时延和时延抖动就可以足够小。再加上使用具有大容量高速缓存的高速路由器，在因特网上传送实时数据就不会有问题。也有人认为，必须将因特网改造为能够对端到端通信所需的带宽资源实现**预留**(reservation)，从而根本改变因特网的协议栈——**从无连接的网络转变为面向连接的网络**。还有人认为，部分改动因特网的协议栈所付出的代价较小，而这也能够使多媒体信息在因特网上的传输质量得到改进。

尽管上述的争论仍在继续，但因特网的一些新的协议也不断在出现。下面我们有选择地讨论因特网演进中的若干问题。

10.2 因特网的多媒体体系结构

为了实现多媒体通信，需要使用一些新的协议。图 10-3 是因特网的多媒体体系结构。这些协议可分成三类，即直接传送声音或视像数据的、与服务质量有关的以及与信令有关的。下面就对这些新增加的应用层协议进行简单的介绍。

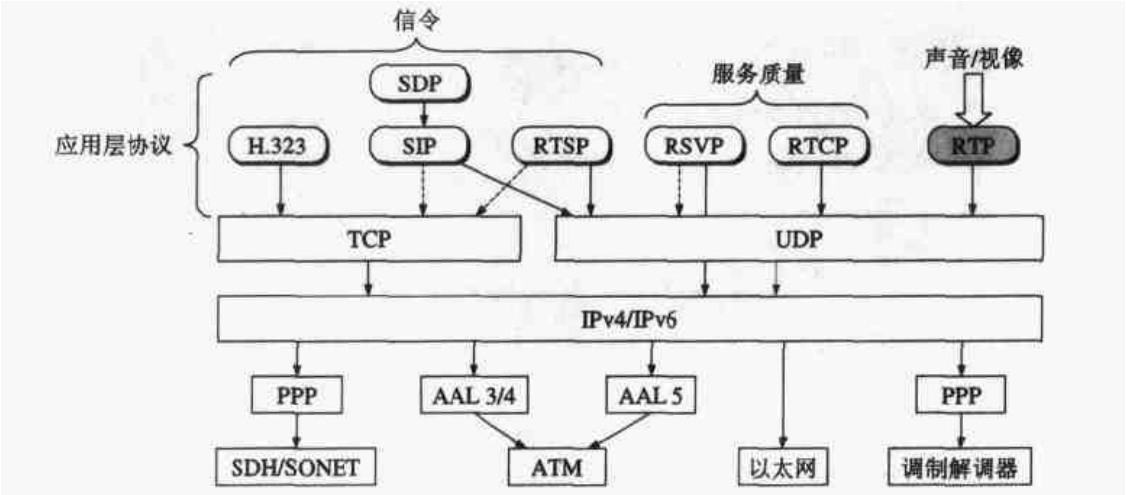


图 10-3 因特网的多媒体体系结构

10.2.1 实时运输协议 RTP

实时运输协议 RTP (Real-time Transport Protocol)是 IETF 的 AVT 工作组(Audio/Video

Transport WG)开发的协议[W-AVT]。

RTP [RFC 1889, 1890]为实时应用提供端到端的运输，但不提供任何服务质量的保证。需要发送的多媒体数据块（声音或视像）经过压缩编码处理后，先送给 RTP 封装成为 RTP 分组（也可称为 RTP 报文[①]），RTP 分组再装入运输层的 UDP 用户数据报，然后再向下递交给 IP 层。RTP 现已成为因特网建议标准，并且已被广泛使用。RTP 同时也是 ITU-T 的标准(H.225.0)。实际上，RTP 是一个协议框架，因为它只包含了实时应用的一些共同的功能。RTP 自己并不对多媒体数据块做任何处理，而只是向应用层提供一些附加的信息，让应用层知道应当如何进行处理。

图 10-3 将 RTP 协议画在应用层。这是因为从应用开发者的角度看，RTP 应当是应用层的一部分。在应用的发送端，开发者必须编写用 RTP 封装分组的程序代码，然后将 RTP 分组交给 UDP 插口接口。在接收端，RTP 分组通过 UDP 插口接口进入应用层后，还要利用开发者编写的程序代码从 RTP 分组中将应用数据块提取出来。

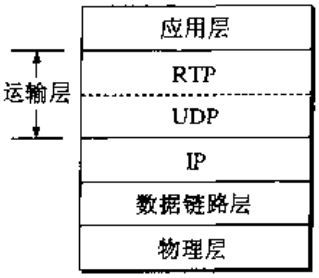


图 10-4 RTP 也可看成是运输层的一个子层

然而 RTP 的名称又隐含地表示它是一个运输层协议。这样划分也是可以的，因为 RTP 封装了多媒体应用的数据块，并且由于 RTP 向多媒体应用程序提供了服务（如时间戳和序号），因此也可以将 RTP 看成是在 UDP 之上的一个运输层的子层，如图 10-4 所示。

RTP 还有两点值得注意。首先，RTP 分组只包含 RTP 数据，而控制是由另一个配套使用的 RTCP 协议提供（这在下一节介绍）。其次，RTP 在 1025 到 65535 之间选择一个未使用的偶数 UDP 端口号，而在同一次会话中的 RTCP 则使用下一个奇数 UDP 端口号。但端口号 5004 和 5005 则分别用作 RTP 和 RTCP 的默认端口号。

图 10-5 给出了 RTP 分组的首部格式，下面进行简单的介绍。

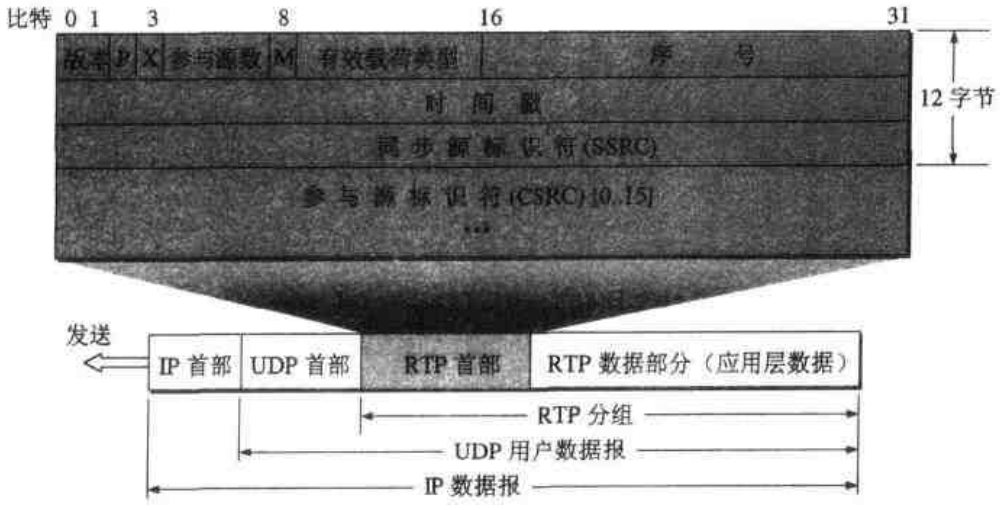


图 10-5 RTP 分组的首部格式

① 注：按惯例，在运输层或应用层的协议数据单元应当叫做报文，但相关 RFC 文档中都是使用 RTP packet 这一名词。为了和 RFC 文档一致，这里也使用“RTP 分组”，下一节的 RTCP 也按同样方法处理。

在 RTP 分组的首部中，前 12 个字节是必须的，而 12 字节以后的是可选的。下面按照各字段重要性的顺序来进行介绍。

（1）**有效载荷类型**(payload type)占 7 bit。这个字段指出后面的 RTP 数据属于何种格式的应用。收到 RTP 分组的应用层就根据此字段指出的类型进行处理。例如，对于音频有效载荷（每一种格式后面括弧中的数字就表示其有效载荷的编码）：μ律 PCM(0), GSM(3), LPC(7), A 律 PCM(8), G.722(9), G.728(15)等。对于视频有效载荷：活动 JPEG(26), H.261(31), MPEG1(32), MPEG2(33)等。

（2）**序号**　占 16 bit。对每一个发送出的 RTP 分组，其序号加 1。在一次 RTP 会话开始时的初始序号是随机选择的。序号使接收端能够发现丢失的分组，同时也能将失序的 RTP 分组重新按序排列好。例如，在收到序号为 60 的 RTP 分组后又收到了序号为 65 的 RTP 分组。那么就可推断出，中间还缺少序号为 61 至 64 的 4 个 RTP 分组。

（3）**时间戳**　占 32 bit。时间戳反映了 RTP 分组中的数据的第一个字节的采样时刻。在一次会话开始时时间戳的初始值也是随机选择的。即使是在没有信号发送时，时间戳的数值也要随时间而不断地增加。接收端使用时间戳可准确知道应当在什么时间还原哪一个数据块，从而消除时延的抖动。时间戳还可用来使视频应用中的声音和图像同步。在 RTP 协议中并没有规定时间戳的粒度(granularity)[①]，这取决于有效载荷的类型。因此 RTP 的时间戳又称为**媒体时间戳**，以强调这种时间戳的粒度取决于信号的类型。例如，对于 8 kHz 采样的话音信号，若每隔 20 ms 构成一个数据块，则一个数据块中包含有 160 个样本($0.02 \times 8000 = 160$)。因此发送端每发送一个 RTP 分组，其时间戳的值就增加 160。

（4）**同步源标识符**　占 32 bit。**同步源标识符** SSRC (Synchronous SouRCe identifier)是一个数，用来标志 RTP 流(stream)的来源。SSRC 与 IP 地址无关，在新的 RTP 流开始时随机地产生。由于 RTP 使用 UDP 传送，因此可以有多个 RTP 流（例如，使用几个摄像机从不同角度拍摄同一个节目所产生的多个 RTP 流）复用到一个 UDP 用户数据报中。SSRC 可使接收端的 UDP 能够将收到的 RTP 流送到各自的终点。两个 RTP 流恰好都选择同一个 SSRC 的概率是极小的。若发生这种情况，这两个源就都重新选择另一个 SSRC。

（5）**参与源标识符**　这是选项，最多可有 15 个。**参与源标识符** CSRC (Contributing SouRCe identifier)也是一个 32 bit 的数，但它是用来标志来源于不同地点的 RTP 流。在多播环境中，可以用中间的一个站（这样的站叫做**混合站** mixer）将多个发往同一个地点的 RTP 流混合成一个流（这样可节省通信资源）。在目的站再根据 CSRC 的数值将不同的 RTP 流分开。

（6）**参与源数**　占 4 bit。这个字段给出后面的参与源标识符的数目。

（7）**版本**　占 2 bit。当前使用的是版本 2。

（8）**填充 P**　占 1 bit。在某些特殊情况下需要对应用数据块加密，这往往要求每一个数据块有确定的长度。如不满足这种长度要求，就需要进行填充。这时就将 P 比特置 1，表示这个 RTP 分组的数据有若干填充字节。在数据部分的最后一个字节用来表示所填充的字节数。

（9）**扩展 X**　占 1 bit。若 M 置为 1，则表示在此 RTP 首部后面还有扩展首部。扩展首部很少使用，这里不再讨论。

---

① 注：粒度(granularity)用来说明一个对象或活动的特性，如大小、规模、详细程度或穿透深度。粒度可用于天文学和物理学，也经常用在信息技术中，例如，描述一张图像细节的精细程度。如果不熟悉所讨论问题的上下文，有时就不太容易准确地理解粒度的含义。

（10）标记 M　　占 1 bit。当 M 比特置 1 时就表示这个 RTP 分组具有特殊意义。例如，在传送视频流时用来表示每一帧的开始。

10.2.2 实时运输控制协议 RTCP

实时运输控制协议 RTCP (RTP Control Protocol)是与 RTP 配合使用的协议[RFC 1889, 1890]，实际上，RTCP 协议也是 RTP 协议的不可分割的部分。

RTCP 协议的主要功能是：服务质量的监视与反馈、媒体间的同步（如某一个 RTP 发送的声音和图像的配合），以及多播组中成员的标识。RTCP 分组（也可称为 RTCP 报文）也使用 UDP 来传送，但 RTCP 并不对声音或视像分组进行封装。由于 RTCP 分组很短，因此可将多个 RTCP 分组封装在一个 UDP 用户数据报中。RTCP 分组周期性地在网上传送，它带有发送端和接收端对服务质量的统计信息报告（如已发送的分组数和字节数、分组丢失率、分组到达时间间隔的抖动等）。

表 10-1 是 RTCP 使用的五种分组类型，它们都使用同样的格式。

表 10-1　RTCP 的五种分组类型

类型	缩写表示	意　义
200	SR	发送端报告
201	RR	接收端报告
202	SDES	源点描述
203	BYE	结束
204	APP	特定应用

结束分组 BYE 表示关闭一个数据流。

特定应用分组 APP 使应用程序能够定义新的分组类型。

接收端报告分组 RR 用来使接收端周期性地向所有的点用多播方式进行报告。接收端每收到一个 RTP 流（一次会话包含有许多的 RTP 流）就产生一个接收端报告分组 RR。RR 分组的内容有：所收到的 RTP 流的 SSRC；该 RTP 流的分组丢失率（若分组丢失率太高，发送端就应当适当降低发送分组的速率）；在该 RTP 流中的最后一个 RTP 分组的序号；分组到达时间间隔的抖动等。

发送 RR 分组有两个目的。第一，可以使所有的接收端和发送端了解当前网络的状态。第二，可以使所有发送 RTCP 分组的站点自适应地调整自己发送 RTCP 分组的速率，使得起控制作用的 RTCP 分组不要过多地影响传送应用数据的 RTP 分组在网络中的传输。通常是使 RTCP 分组的通信量不超过网络中的数据分组的通信量的 5%，而接收端报告分组的通信量又应小于所有 RTCP 分组的通信量的 75%。

发送端报告分组 SR 用来使发送端周期性地向所有接收端用多播方式进行报告。发送端每发送一个 RTP 流，就要发送一个发送端报告分组 SR。SR 分组的主要内容有：该 RTP 流的 SSRC；该 RTP 流中最新产生的 RTP 分组的时间戳和绝对时钟时间（或墙上时钟时间 wall clock time）；该 RTP 流包含的分组数；该 RTP 流包含的字节数。

绝对时钟时间是必要的。因为 RTP 要求每一种媒体使用一个流。例如，要传送视频图像和相应的声音就需要传送两个流。有了绝对时钟时间就可进行图像和声音的同步。

源点描述分组 SDES 给出会话中参加者的描述，它包含参加者的规范名 CNAME (Canonical NAME)。规范名是参加者的电子邮件地址的字符串。

10.2.3 实时流式协议 RTSP

实时流式协议 RTSP (Real-Time Streaming Protocol)是 IETF 的 MMUSIC 工作组(Multiparty MUltimedia SessIon Control WG)开发的协议[W-MMUSIC]，现已成为因特网建议标准[*RFC 2326*]。

RTSP 协议以客户服务器方式工作，它是一个**多媒体播放控制协议**，用来使用户在播放从因特网下载的实时数据时能够进行控制（像在影碟机上那样的控制），如：暂停/继续、后退、前进等。因此 RTSP 又称为“**因特网录像机遥控协议**”。

要实现 RTSP 的控制功能，我们不仅要有协议，而且要有专门的**媒体播放器**(media player)和**媒体服务器**(media server)。普通的万维网浏览器和服务器没有上述的播放实时多媒体节目的功能。在使用 RTSP 的播放器中比较著名的是苹果公司的 QuickTime。媒体播放器的主要功能是：解压缩、消除时延抖动、纠正差错以及具有良好的用户界面。媒体服务器与媒体播放器的关系是服务器与客户的关系。媒体服务器与普通的万维网服务器的最大区别就是媒体服务器支持流式(streaming)音频和视频的传送，因而在客户端的媒体播放器可以边下载边播放（当然需要先将节目存储一小段时间）。但从普通万维网服务器下载多媒体节目时，是先将整个文件下载完毕，然后再进行播放。由于整个文件的下载时间往往太长，使用户无法忍受。另外，普通的万维网服务器向客户端传送文件是基于 HTTP 的，客户请求的文件放在 HTTP 响应报文中，并用 TCP 传送给客户端。这种传送方式强调的是准确无误但不能及时（尤其是在发生拥塞时），也不适用于多播环境中。

RTSP 协议与前面讲的 RTP 和 RTCP 协议的关系如图 10-6 所示。

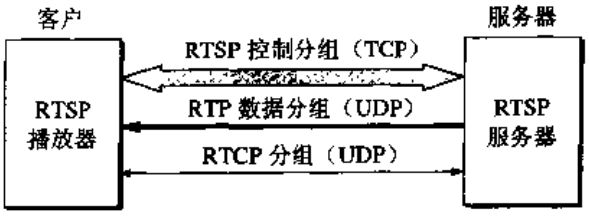


图 10-6 RTSP 与 RTP 和 RTCP 的关系

从图 10-6 可看出，音频和视频多媒体数据还是封装在 RTP 分组中来传送，而 RTCP 是为保证服务质量不可缺少的协议。RTSP 仅仅是使媒体播放器能够控制多媒体流的传送。因此，RTSP 又称为**带外协议**(out-of-band protocol)，而多媒体流是使用 RTP 在**带内**(in-band)传送的。可见 RTSP 信道有点像 FTP 的控制信道。RTSP 的语法和操作与 HTTP 协议的相似。[RFC 2326]还规定，RTSP 控制分组既可在 TCP 上传送，也可在 UDP 上传送。

10.3 IP 电话

IP 电话是在因特网上传送多媒体信息的一个特例——传送声音信息。通过 IP 电话的讨论可以更加清楚地了解在因特网上传送多媒体信息应当解决好哪些问题。

10.3.1 IP 电话概述

IP 电话有多个英文同义词。常见的有 VoIP (Voice over IP), Internet Telephony 和 VON

(Voice On the Net)。但 IP 电话的含义却有不同的解释。

狭义的 IP 电话就是指在 IP 网络上打电话。所谓“IP 网络”就是“使用 IP 协议的分组交换网”的简称。这里的网络可以是因特网，也可以是包含有传统的电路交换网的互连网，不过在互连网中至少要有一个 IP 网络。

广义的 IP 电话则不仅仅是电话通信，而且还可以是在 IP 网络上进行交互式多媒体实时通信（包括话音、视像等），甚至还包括即时传信 IM (Instant Messaging)。即时传信是在上网时就能从屏幕上得知有哪些朋友也正在上网。若有，则彼此可在网上即时交换信息（文字的或声音的），也包括使用一点对多点的多播技术。因此，IP 电话可看成是一个正在演进的多媒体服务平台，是话音、视像、数据综合的基础结构。在某些条件下（例如使用宽带的局域网），IP 电话的话音质量甚至还优于普通电话。

下面要讨论的是狭义的 IP 电话[COLL01][W-VoIP]。

其实 IP 电话并非新概念。早在 20 世纪 70 年代初期 ARPANET 刚开始运行不久，美国即着手研究如何在计算机网络上传送电话信息，即所谓的分组话音通信。但在很长一段时间里，分组话音通信发展得并不快。主要的原因是：

（1）缺少廉价的高质量、低速率的话音信号编解码软件和相应的芯片。

（2）计算机网络的传输速率和路由器处理速率均不够快，因而导致传输时延过大。

（3）没有保证实时通信服务质量 QoS (Quality of Service)的网络协议。

（4）计算机网络的规模较小，而通信网只有在具有一定规模后才能产生经济效益。

然而到了 20 世纪 90 年代中期，上述的几个问题才相继得到了较好地解决。于是美国的 VocalTec 在 1995 年初率先推出了实用化的 IP 电话。但是这种 IP 电话必须使用 PC 机。1996 年 3 月，IP 电话进入了一个转折点：VocalTec 公司成功地推出了 IP 电话网关(IP Telephony Gateway)，它是公用电话网[①]与 IP 网络的接口设备。有了这种 IP 电话网关，就可实现以下两种类型的电话通信：

（1）公用电话用户之间打 IP 电话要经过 IP 电话网关两次。

（2）多媒体 PC 机用户与公用电话用户打 IP 电话仅需经过 IP 电话网关一次。

在两个多媒体 PC 机用户之间的通话，当然就不需要经过 IP 电话网关。图 10-7 画出了这种通信方式，以及需要通过网关转换的两种方式。图 10-7 中间的一种情况是 PC 机和公用电话网用户打 IP 电话。最后一种情况是两个端点都是普通电话机，这当然是最方便的。读者应当特别注意在哪一部分是使用电路交换还是分组交换。

在公用电话网中信令(signaling)是至关重要的。电话交换机根据用户所拨打的号码就能够通过合适的路由找到相距很远的被叫用户，并在主叫和被叫之间建立起一条电路连接。这些都是依靠电话信令。我们听到的振铃声、忙音或一些录音提示，以及打完电话挂机释放连接，也都是由电话信令来处理的。现在电话网使用的信令就是 7 号信令 SS7。利用 IP 网络打电话同样也需 IP 网络能够识别的某种信令。但由于 IP 电话要使用已经存在的公用电话网，因此 IP 电话的信令必须在所有的功能上与原有的 7 号信令相兼容，这样才能使 IP 网络和公用电话网上的两种信令能够互相转换，因而能够做到互操作。

当然，IP 电话使用的话音编码也必须能够和普通电话的标准 PCM 编码互相转换。

---

① 注：公用电话网即公用电路交换电话网，又称为传统电话网或电信网。

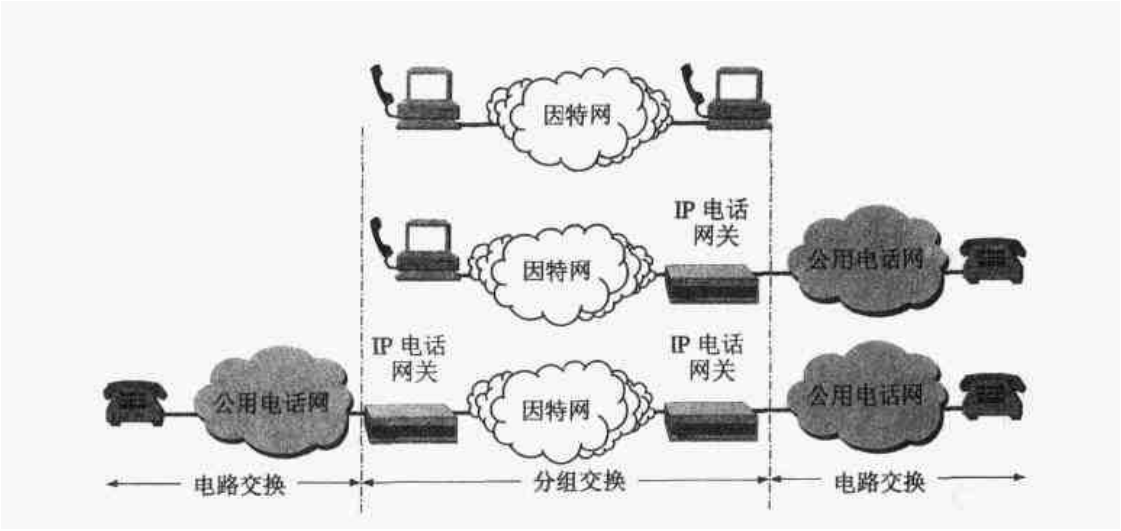


图 10-7　IP 电话网关的几种连接方法

解决上述问题的方法就是使用前面提到的 IP 电话网关（见图 10-7）。IP 电话网关的作用就是：

（1）在电话呼叫阶段和呼叫释放阶段进行电话信令的转换。

（2）在通话期间进行话音编码的转换。

现在 IP 电话有两套标准。一套是 ITU-T 定义的 H.323 协议。另一套是 IETF 提出的会话发起协议 SIP (Session Initiation Protocol)。下面对它们进行简单介绍。

10.3.2　H.323

H.323 是 ITU-T 于 1996 年制订的一个名称很长的建议书，1998 年的第二个版本改用的名称是“基于分组的多媒体通信系统”。基于分组的网络包括因特网、局域网、企业网、城域网和广域网。H.323 是因特网的端系统之间进行实时声音和视像会议的标准。H.323 包括系统和构件的描述、呼叫模型的描述、呼叫信令过程、控制报文、复用、话音编解码器、视像编解码器，以及数据协议等，但不保证服务质量 QoS。图 10-8 示意了连接在分组交换网上的 H.323 终端使用 H.323 协议进行多媒体通信。



图 10-8　H.323 终端使用 H.323 协议进行多媒体通信

H.323 标准指明了 4 种构件，使用这些构件连网就可以进行点对点或一点对多点的多媒体通信。

（1）H.323 终端　　这可以是一个 PC 机，也可以是运行 H.323 程序的单个设备。

（2）网关　　网关连接到两种不同的网络，使得 H.323 网络可以和非 H.323 网络（如公用电话网）进行通信。仅在一个 H.323 网络上通信的两个终端当然就不需要使用网关。

（3）网闸(gatekeeper)　网闸相当于整个 H.323 网络的大脑。所有的呼叫都要通过网闸，因为网闸提供地址转换、授权、带宽管理和计费功能。网闸还可以帮助 H.323 终端找到距离

公用电话网上的被叫用户最近的一个网关。

（4）**多点控制单元** MCU (Multipoint Control Unit)　MCU 支持三个或更多的 H.323 终端的音频或视频会议。MCU 管理会议资源、确定使用的音频或视频编解码器。

网关、网闸和 MCU 在逻辑上是分开的构件，但它们可实现在一个物理设备中。在 H.323 标准中将 H.323 终端、网关和 MCU 都称为 H.323 **端点**(end point)。

图 10-9 表示了利用 H.323 网关使因特网能够和公用电话网进行连接。

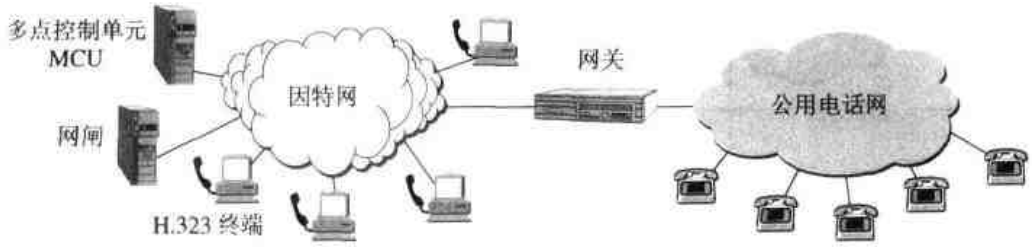


图 10-9　H.323 网关用来和非 H.323 网络进行连接

图 10-10 给出了 H.323 的体系结构。可以看出，H.323 是一个协议族，它与下层网络或运输协议无关。H.323 包括以下一些组成部分：

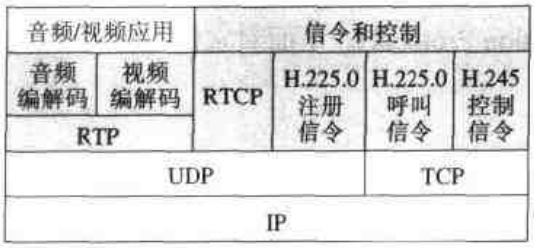


图 10-10　H.323 的协议体系结构

（1）**音频编解码器**　H.323 要求至少要支持 G.711 (64 kb/s 的 PCM)。建议支持如 G.722, G.723.1, G.728 和 G.729 等。

（2）**视频编解码器**　H.323 要求必须支持视频压缩标准 H.261 标准。

（3）H.255.0 **注册信令**　即**注册/接纳/状态** RAS (Registration/Admission/Status)。H.323 终端和网闸使用 RAS 来完成注册、接纳控制和带宽转换等功能。

（4）H.225.0 **呼叫信令**　用来在两个 H.323 端点之间建立连接。

（5）H.245 **控制信令**　用来交换端到端的控制报文以便管理 H.323 端点的运行。

（6）**实时运输协议** RTP 和**实时运输控制协议** RTCP　这两个协议前面已讨论。

H.323 的出发点是以已有的电路交换电话网为基础，增加了 IP 电话的功能（即远距离传输采用 IP 网络）。H.323 的信令也沿用原有电话网的信令模式，因此与原有电话网的连接比较容易。

10.3.3　会话发起协议 SIP

虽然 H.323 系列现在已被大部分生产 IP 电话的厂商采用，但由于 H.323 过于复杂（整个文档多达 736 页），不便于发展基于 IP 的新业务，因此 IETF 的 MMUSIC 工作组在 1999 年制订了另一套较为简单且实用的标准，即**会话发起协议 SIP** (Session Initiation Protocol)，其文

档仅 128 页[RFC 2543]，目前已成为因特网的建议标准[W-SIP]。

SIP 协议的出发点是以因特网为基础，而将 IP 电话视为因特网上的新应用。因此 SIP 协议只涉及到 IP 电话所需的信令和有关服务质量的问题，而没有提供像 H.323 那样多的功能。SIP 没有推荐具体使用的特定编解码器和进行实时数据传送所需的协议。虽然 SIP 没有指定使用 RTP 协议，但实际上大家还是选用 RTP 和 RTCP 作为配合使用的协议。

SIP 使用文本方式的客户服务器协议。SIP 系统只有两种构件，即**用户代理**(user agent)和**网络服务器**(network server)。用户代理包括两个程序，即**用户代理客户**(user agent client)和**用户代理服务器**(user agent server)，前者用来发起呼叫，而后者用来接受呼叫。网络服务器分为**代理服务器**(proxy server)和**重定向服务器**(redirect server)。代理服务器接受来自主叫用户的呼叫请求（实际上是来自用户代理客户的呼叫请求），并将其转发给下一跳代理服务器，最后将呼叫请求转发给被叫用户（实际上是转发给用户代理服务器）。重定向服务器不接受呼叫，它通过响应告诉客户下一跳代理服务器的地址，由客户按此地址向下一跳代理服务器重新发送呼叫请求。

SIP 还有一个配套协议是**会话描述协议** SDP (Session Description Protocol)。SDP 在电话会议的情况下特别重要，因为电话会议的参加者是**动态地加入和退出**。SDP 详细地指明了**媒体**编码、协议的端口号以及多播地址。SDP 现在也是因特网建议标准[RFC 2327]。

SIP 使用了 HTTP 的许多首部、编码规则、差错码以及一些鉴别机制，它比 H.323 具有更好的可扩缩性。由于 SIP 问世较晚，因此它现在比 H.323 占有的市场份额要小。对今后作为 IETF 标准的 SIP 协议的进展情况应当引起我们的注意。

10.3.4 IP 电话的通话质量

IP 电话的通话质量与电路交换电话网的通话质量有很大的差别。在电路交换电话网中任何两端之间的通话质量都是有保证的。但 IP 电话则不然。IP 电话的通话质量主要由两个因素决定。一个是**通话双方端到端的时延和时延抖动**，另一个是**话音分组的丢失率**。但这两个因素是**不确定的**，是取决于**当时网络上的通信量**。若网络上的通信量非常大以致发生了网络拥塞，那么端到端时延和时延抖动以及分组丢失率都会很高，这就导致 IP 电话的通话质量下降。因此，你打的 IP 电话的通话质量**取决于当时其他用户的行为**。我们应当注意到，电路交换电话网的情况不是这样。当电路交换电话网的通信量太大时，往往使我们无法拨通电话（听到的是忙音），即电话网拒绝对正在拨号的用户提供服务。但是只要我们拨通了电话，那么电信公司可以保证使户用满意的通话质量。

经验证明，在电话交谈中，端到端的时延不应超过 250ms，否则交谈者会能感到不自然。经过陆地公用电话网的时延一般只有 50～70 ms。经过同步卫星的电话端到端时延就超过 250 ms，因此一般人都不太适应经过卫星传送的大时延。IP 电话的时延通常都会超过 250 ms，因此 IP 电话必须努力减小端到端的时延。当通信线路产生回声时，则容许的端到端时延就更小些（有时甚至只容许几十毫秒的时延）。

IP 电话端到端时延是由以下几个因素造成的：

（1）话音信号进行模数转换要产生时延。

（2）已经数字化的话音比特流要积累到一定的数量才能够装配成一个话音分组，这也产生时延。

（3）话音分组的发送需要时间，此时间等于话音分组长度与通信线路的数据率之比。

（4）话音分组在因特网中经过许多路由器的存储转发时延。

（5）话音分组到达接收端在缓存中暂存所引起的时延。

（6）最后将话音分组还原成模拟话音信号的数模转换也要产生一定的时延。

（7）话音信号在通信线路上的传播时延。

（8）由终端设备的硬件和操作系统产生的接入时延。由 IP 电话网关引起的接入时延约为 20～40 ms，而用户 PC 机声卡引起的接入时延为 20～180 ms。若使用 V.34 调制解调器，则应再增加 20～40 ms 的时延（由于进行数字信号处理、均衡等）。

话音信号在通信线路上的传播时延一般都很小（卫星通信除外），通常可不予考虑。当采用高速光纤主干网时，上述第三项时延也不大。

第一、第二和第六项时延取决于话音编码的方法。很明显，在保证话音质量的前提下，话音信号的数码率应尽可能低些。为了能够在世界范围提供 IP 电话服务，话音编码就必须采用统一的国际标准。ITU-T 已制订出不少话音质量不错的低速率话音编码的标准。目前适合 IP 电话使用的 ITU-T 标准主要有以下两种：

（1）G.729　话音速率为 8 kb/s 的**共轭结构代数码激励线性预测声码器** CS-ACELP (Conjugate-Structure Algebraic-Code-Excited Linear Prediction)

（2）G.723.1　话音速率为 (5.3/6.3) kb/s 的为多媒体通信用的低速率声码器（由于 G 系列标准的号码已用完，故使用了小数点编号 723.1）

这两种标准的比较见表 10-2。

表 10-2　G.729 和 G.723.1 的主要性能比较

标　准	比特率(kb/s)	帧大小(ms)	处理时延(ms)	帧长(字节)	数字信号处理 MIPS
G.729	8	10	10	10	20
G.723.1	5.3/6.3	30	30	20/24	16

表中的比特率是输入为 64 kb/s 标准 PCM 信号时在编码器输出的数据率。帧大小是压缩到每一个分组中的话音信号时间长度。处理时间是对一个帧运行编码算法所需的时间。帧长是一个已编码的帧的字节数（不包括首部）。数字信号处理 MIPS 是用数字信号处理芯片实现编码所需的最小处理机速率（以每秒百万指令为单位）。如使用 PC 机的通用处理机，则所需的处理机 MIPS 还要高些。不难看出，G.723.1 标准虽然可得到更低的数据率，但其时延也更大些。

要减少上述第四和第五项时延较为困难。当网络发生拥塞而产生话音分组丢失时，还必须采用一定的策略（称为“**丢失掩蔽算法**”）对丢失的话音分组进行处理。例如，可使用前一个话音分组来填补丢失的话音分组的间隙。

接收端缓存空间和播放时延的大小，对话音分组丢失率和端到端时延也有很大的影响。图 10-11 说明了这一问题。话音质量可分为四个级别，即“长途电话质量”（这是最好的质量）、“良好”、“基本可用”和“不好”，各对应于图 10-11 中的一个区域。越接近坐标原点，话音质量就越好。目前 IP 电话的通话质量还不可能达到传统电话的质量。我们假定某 IP 电话的通话质量处在图中 B 点的位置。若增大接收端缓存空间并增大播放时延，则话音分组丢失率将减小，但端到端的时延将增大（如图中的 C 点）。继续增大播放时延，则话音分组丢失率将继续减小，趋向于网络所引起的丢失率（如图中的 D 点），但 D 点的端到端时延很大，话音质量很不好。反之，若将接收端缓存做得很小并减小播放时延，则端到端时延将减小，趋向于网络所引起的端到端时延（如图中的 A 点），但话音分组丢失率将会大大增加，话音质量也不好。

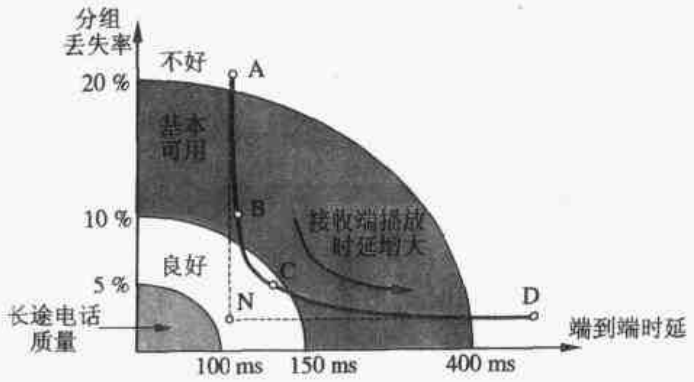


图 10-11　播放时延有一个最佳值

可见接收端的播放时延有一个最佳值。图中有一个点 N，相当于端到端时延和话音分组丢失率都是最小。但实际上不可能工作在这个点上。

据统计，当通话双方相距 3200 km 时，因特网上的时延约为 30 ~ 100 ms（传播和排队），而所有各环节的时延总和约为 100 ~ 262 ms（在两个 IP 电话网关之间）或 170 ~ 562 ms（在两个 PC 机之间）[KAST98]。可见为了减小时延，应尽可能不要直接用 PC 机打 IP 电话。

提高路由器的转发分组的速率对提高 IP 电话的质量也是很重要的。据统计，一个跨大西洋的 IP 电话一般要经过 20 ~ 30 个路由器。现在一个普通路由器每秒可转发 50 ~ 100 万个分组。若能改用吉比特路由器（又称为**线速路由器**），则每秒可转发 500 万至 6 000 万个分组（即交换速率达 60 Gb/s 左右）。这样还可进一步减少由网络造成的时延。

要提高 IP 电话的通话质量，还有一个方面值得注意，这就是下一节要讨论的内容——改进因特网的“尽最大努力交付”的服务。

10.4　改进“尽最大努力交付”的服务

上一节介绍了在应用层新增加的一些协议。使因特网更好地传送多媒体信息的另一种方法是改变因特网平等对待所有分组的思想，使对时延有较严格要求的实时话音或视像分组能够从网络得到更好的**服务质量** QoS。

下面我们先介绍提供服务质量的一般方法。

10.4.1　使因特网提供服务质量

根据 ITU-T 在建议书 E.800 中给出的定义，**服务质量 QoS 是服务性能的总效果，此效果决定了一个用户对服务的满意程度**。因此在最简单的意义上，有服务质量的服务就是能够满足用户的应用需求的服务，或者说，可提供一致的、可预计的数据交付服务。

在涉及到一些具体问题时，服务质量可用若干基本的性能指标来描述，包括可用性、差错率、响应时间、吞吐量、分组丢失率、连接建立时间、故障检测和改正时间等。服务提供者可向其用户保证某一种等级的服务质量。

我们已多次强调过，因特网的网络本身只能提供“尽最大努力交付”的服务。而要传送多媒体信息，网络又必须具有一定的服务质量。下面通过图 10-12 的例子说明应从哪些方面入手使因特网具有一定的服务质量[KURO01]。图中表示局域网上的两个主机 H_1 和 H_2 通过非常简单的网络（路由器 R_1 和 R_2 以及连接它们的链路）分别向远地另外两个主机 H_3 和 H_4 发

送数据。连接 R_1 和 R_2 的链路带宽是 1.5 Mb/s。现在考虑以下四种情况。

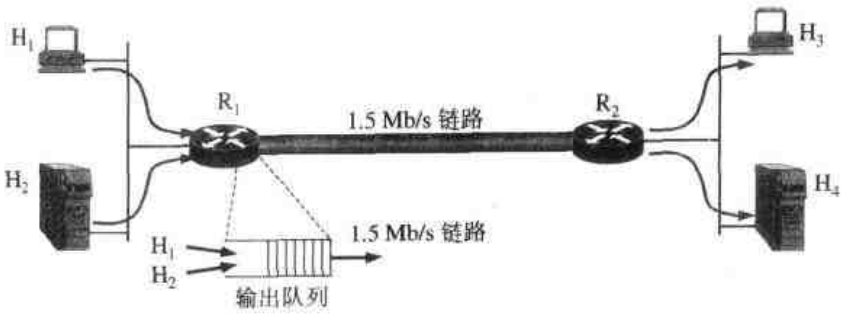


图 10-12　主机 H_1 和 H_2 分别向主机 H_3 和 H_4 发送数据

（1）一个 1 Mb/s 的实时音频数据和一个 FTP 文件数据

假定 H_1 向 H_3 传送 1 Mb/s 的实时音频数据而 H_2 向 H_4 传送 FTP 文件数据。两个主机发送的数据都在路由器 R_1 的输出队列中排队。显然，若突然有一个很大的 FTP 数据块来到 R_1，就会将输出队列全部占满。后面到达路由器 R_1 的实时音频分组就会被丢弃。显然这是不合理的。因此需要增加一个机制，就是**给不同性质的分组打上不同的标记**。这样当 H_1 和 H_2 的分组进入路由器 R_1 时，R_1 就能够识别实时数据分组，并使这些分组以高优先级进入输出队列，而仅在队列有多余空间时才准许低优先级的 FTP 的数据分组进入。

（2）一个 1 Mb/s 的实时音频数据和一个高优先级的 FTP 文件数据

假定 FTP 的用户用高价从 ISP 处购买了高优先级服务，而实时音频的用户只购买了低优先级服务。因此，仅根据分组自己的标记来确定其服务等级还不够合理。可见应当使路由器增加一种机制——**分类**(classification)，即路由器根据某些准则（例如，根据发送数据的地址）对输入分组进行分类，然后对不同类别的通信量给予不同的优先级。

（3）一个数据率异常的实时音频数据和一个 FTP 文件数据

假定上述的主机 H_1 的数据率突然不正常地增大到 1.5 Mb/s 或更高（这可能是出了故障或恶意破坏网络的正常运行），那么就会使主机 H_2 的 FTP 的低优先级数据无法通过路由器 R_1。因此，应当使路由器能够将某个数据流进行通信量的**管制**(policing)，使得这个数据流不要影响其他正常的数据流在网络中通过。例如，可以将 H_1 的数据率限定为 1 Mb/s。路由器 R_1 不停地监视 H_1 的数据率。只要 H_1 的数据率超过规定的 1 Mb/s，路由器 R_1 就将其中的某些分组丢弃，使其数据率不超过原来设定的门限。

为了更加合理地利用网络资源，应在路由器中再增加一种机制——**调度**(scheduling)。我们可以利用调度功能给实时音频和文件传送这两个应用分别分配 1.0 Mb/s 和 0.5 Mb/s 的带宽。这就好像在带宽为 1.5 Mb/s 的链路中划分出两个逻辑链路，其带宽分别为 1.0 Mb/s 和 0.5 Mb/s，因而对这两种应用都有相应的服务质量保证。

（4）H_1 和 H_2 都发送数据率为 1 Mb/s 的实时数据

在这种情况下到达路由器 R_1 的总数据率是 2 Mb/s，已超过了 1.5 Mb/s 链路的带宽。若使这两个主机发出的数据流**平等地**共享 1.5 Mb/s 链路的带宽，则每个数据流平均将丢失 25% 的分组，因而都变得没有用了。比较合理的做法是让一个数据流通过 1.5 Mb/s 的链路，而阻止另一个数据流的通过。这就需要另一种机制——**呼叫接纳**(call admission)。这里借用了电话网的术语，进一步的讨论见后面的 10.4.3 节）。在使用呼叫接纳机制时，一个数据流要预先声明它所需的服务质量，然后或者被准许进入网络（能得到所需的服务质量），或者被拒绝进入网

络（当所需的服务质量不能得到满足时）。

上面简单地说明了为了使因特网能够提供一定的服务质量，应当设法增加一些机制，即：分组的分类、管制、调度以及呼叫接纳。在后面的几节我们将陆续讨论这些问题。

10.4.2 调度和管制机制

调度和管制机制是使因特网能够提供服务质量的重要措施[STAL01]。下面先讨论调度机制。

1．调度机制

这里所说的“调度”就是指排队的规则。如果不采用专门的调度机制，那么在路由器的队列采用的默认排队规则就是先进先出 FIFO (First In First Out)。当队列已满时，后到达的分组就被丢弃。先进先出的最大缺点就是不能区分时间敏感分组和一般数据分组，并且也不公平，因为这使得排在长分组后面的短分组要等待很长的时间。就像在机场办理登机卡时，排在你前面的一个人持有 50 人的团体票，这时你只能耐心等待。

在先进先出的基础上增加按优先级排队，就能使优先级高的分组优先得到服务。图 10-13 是按优先级排队的例子。图中假定优先级分为两种，因此有两个队列：高优先级队列和低优先级队列。

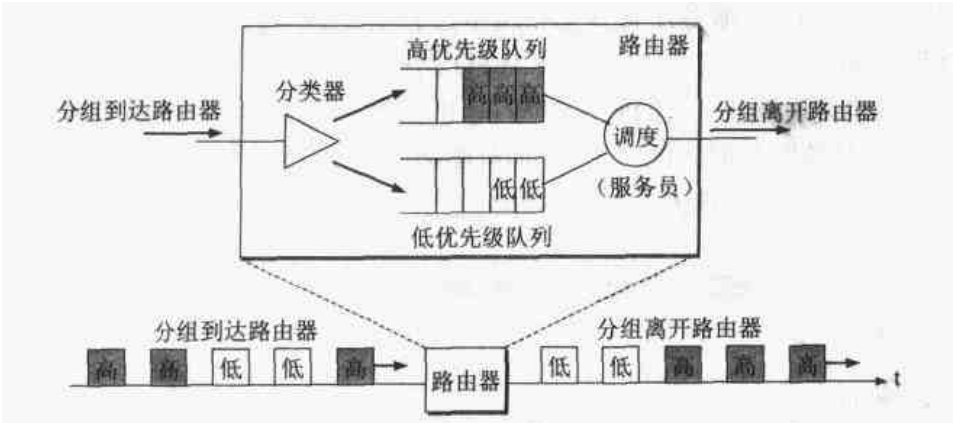


图 10-13 按优先级排队的例子

分组到达路由器后就由分类器（又称为分类程序）对其进行优先级分类，然后按照类别进入相应的队列。图中的圆圈表示调度，其作用是从队列中取走排在队首的分组。调度相当于排队论中的服务员。只要高优先级队列中有分组在内，就从高优先级队列中按照链路速率取出排在队首的分组。只有当高优先级队列已空时，才能轮到低优先级队列中的分组输出到链路上。在图 10-13 的下方给出两个低优先级的分组与三个高优先级的分组一起到达路由器。但离开路由器时，两个低优先级的分组已经处在三个高优先级分组的后面。

简单地按优先级排队会带来一个缺点，这就是在高优先级队列中总是有分组时，低优先级队列中的分组就长期得不到服务。这就不太公平。公平排队 FQ (Fair Queuing)可解决这一问题。公平排队是对每种类别的分组流设置一个队列，然后轮流使每一个队列一次只能发送一个分组。对于空的队列就跳过去。但公平排队也有不公平的地方，这就是长分组得到的服务时间长，而短分组就比较吃亏，并且公平排队并没有区分分组的优先级。

为了使高优先级队列中的分组有更多的机会得到服务，可增加队列“权重”的概念，这

就是**加权公平排队** WFQ (Weighted Fair Queuing)，其工作原理如图 10-14 所示。

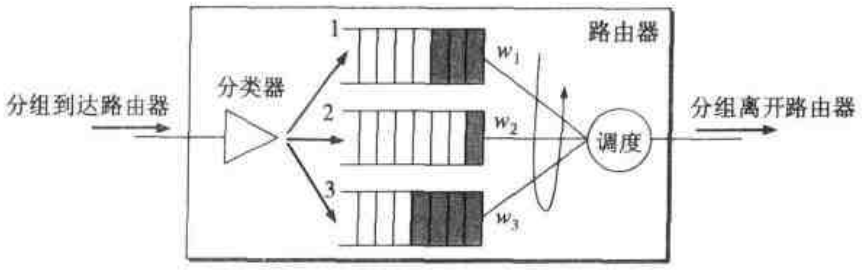


图 10-14　加权公平排队 WFQ

加权公平排队 WFQ 是这样工作的。分组到达后就将分组进行分类，然后送交与其类别对应的队列（这里假定分为三类）。三个队列按顺序依次将队首的分组发送到链路。遇到队列空就跳过去。但根据各类别的优先级的不同，每种队列分配到的服务时间也不同。可以给队列 i 指派一个权重 w_i。于是队列 i 得到的平均服务时间为 $w_i/(\Sigma w_j)$，这里Σw_j是对所有的非空队列的权重求和。这样，若路由器输出链路的数据率（即带宽）为 R，那么队列 i 将得到的有保证的数据率 R_i 应为

$$R_i = \frac{R \times w_i}{\Sigma w_j} \qquad (10\text{-}1)$$

加权公平排队 WFQ 在服务质量体系结构中占有重要的地位。当前的许多路由器产品都加入了 WFQ 调度的功能。为了更好地理解 WFQ 的概念，图 10-15 给出了一个简单的例子，并将 FIFO 的情况也同时画出。我们假定在 WFQ 的情况下，分配给分组流 1 的权重是 0.5（即得到服务的时间占总的服务时间的一半），而分配给其他 10 个分组流的权重都各为 0.05。这样，分组流 2～11 共 10 个分组流合起来的权重也是 0.5。

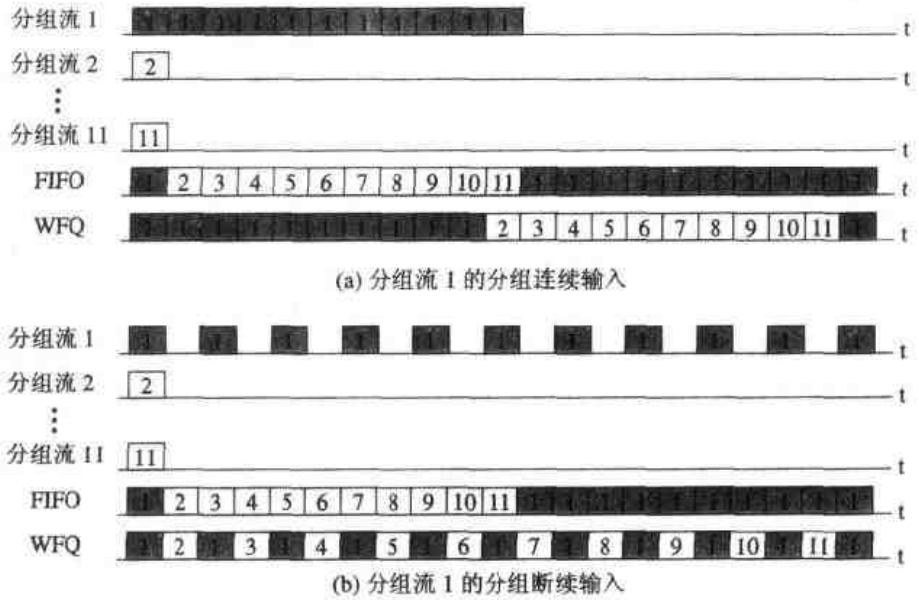


图 10-15　WFQ 与 FIFO 的比较

在使用先进先出规则时，只有一个队列，因此每个分组流的第一个分组共 11 个分组排在队首。在(a)和(b)两种情况下，FIFO 的结果都是一样的，即队列中前 11 分组发送端完毕后才

能发送分组流中剩下的分组。在使用 WFQ 时，在图(a)中分组流 1 先可以发送 10 个分组（但第 11 个分组还不能发送），而在图(b)中分组流 1 和其他的分组流交替地发送。不管是哪一种情况，分组流 1 都能够得到更多时间的服务。

2．管制机制

前面提到了使用管制机制提供服务质量。对一个数据流，我们可根据以下三个方面进行管制：

（1）**平均速率**　网络需要控制一个数据流的平均速率。这里的平均速率是指在一定的时间间隔内通过的分组数。但这个时间间隔的选择也说明了这个指标的严格程度。例如，限定数据流的平均速率为每秒 50 个分组和平均速率为每分钟 3000 个分组，虽然这两个指标的平均值都一样，但其严格程度却不同。假定有一个数据流，有一秒钟通过了 1000 个分组，但一分钟平均下来仍不超过 3000 个，那么这个数据流符合后面一个指标，但却远远不满足前面的指标。

（2）**峰值速率**　峰值速率限制了数据流在非常短的时间间隔内的流量。数学上的“瞬时值”在实际网络中无法测定。因此这里所说的“非常短的时间间隔”需要指明时间间隔是多少。例如，限定数据流的平均速率为每分钟 3000 个分组，但同时限定其峰值速率不超过每秒 1000 个分组。峰值速率也同时受到链路带宽的限制。

（3）**突发长度**　网络也限制在非常短的时间间隔内连续注入到网络中的分组数。

要在网络中对进入网络的分组流按以上三个指标进行管制，可使用非常著名的**漏桶管制器**(leaky bucket policer)（可简称为**漏桶**），其工作原理如图 10-16 所示。

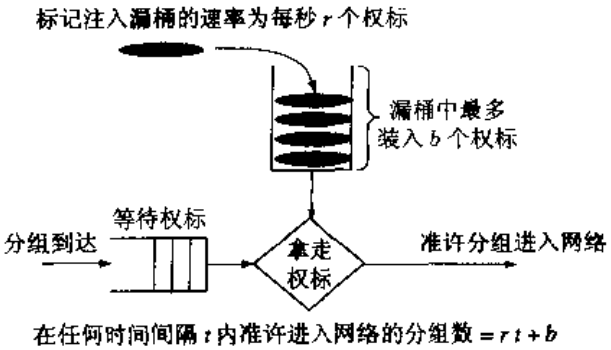


图 10-16　漏桶管制器的工作原理

漏桶是一个抽象的机制。在漏桶中可装入许多**权标**(token)，但最多装入 b 个权标。只要漏桶中的权标数小于 b 个，新的权标就以每秒 r 个权标的恒定速率加入到漏桶中。但若漏桶已装满了 b 个权标，则新的权标就不再装入而漏桶中就最多保持 b 个权标。

漏桶管制分组流进入网络的过程如下。分组进入网络前先要进入一个队列中等候漏桶中的权标。只要漏桶中有权标，就可从漏桶取走一个权标，然后就准许一个分组进入到网络。若漏桶已无权标，就要等新的权标注入到漏桶后，再将这个权标拿走才能准许下一个分组进入网络。这里请注意：“准许进入网络”并不等于说“已经进入了网络”，因为分组进入网络还需要时间，这取决于输出链路的带宽和分组在输出端的排队情况。

假定在时间间隔 t 中将漏桶中的全部 b 个权标都取走。但在这个时间间隔内漏桶又装入了 rt 个新的权标，因此在任何时间间隔 t 内准许进入网络的分组数的最大值为 $rt + b$。控制权

标进入漏桶的速率 r 就可对分组进入网络的速率进行管制。

3．漏桶机制与加权公平排队相结合

将漏桶机制与加权公平排队结合起来可以控制队列中的最大时延。现假定有 n 个分组流输入到一个路由器，复用后从一条链路输出。每一个分组流使用漏桶机制进行管制，漏桶参数为 b_i 和 r_i，$i=1, 2, \ldots, n$（见图 10-17）。

前面已经讲过，WFQ 可以使每一个分组流得到如公式(10-1)所示的有保证的带宽。那么当分组流通过漏桶后等待 WFQ 服务时，一个分组所经受的**最大时延**是多少？

现在考虑分组流 i。假定漏桶 i 已经装满了 b_i 个权标。这就表示分组流 i 不需要等待就可从漏桶中拿走 b_i 个权标，因此 b_i 个分组可以马上从路由器输出。但分组流 i 得到的带宽是由公式(10-1)给出。这 b_i 个分组中的最后一个分组所经受的时延最大，它等于传输这 b_i 个分组所需的时间 d_{max}，即 b_i 除以公式(10-1)给出的传输速率：

$$d_{max}=\frac{b_i\Sigma w_j}{R\times w_i} \tag{10-2}$$

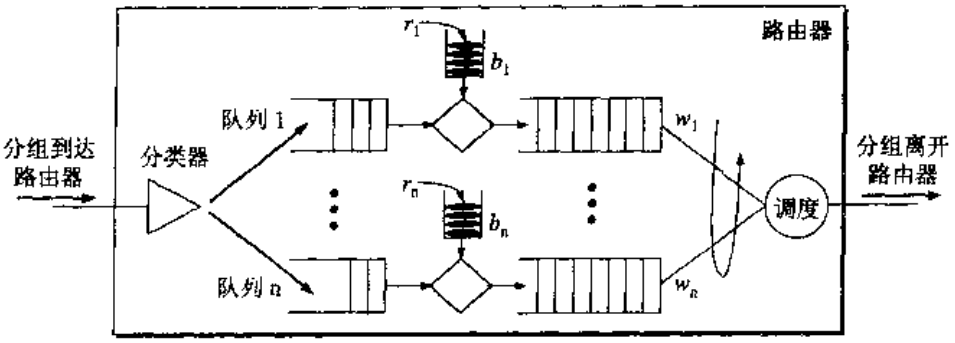


图 10-17　用漏桶机制进行管制

10.4.3　综合服务 IntServ 与资源预留协议 RSVP

最初试图在因特网中将因特网提供的服务划分为不同类别的是 IETF 提出的**综合服务** IntServ (Integrated Services)[RFC 2210～2215]和**资源预留协议** RSVP (ReSource reserVation Protocol) [RFC 2205～2209] [ZHAN93] [W-IntServ]，其中的某些 RFC 文档已成为因特网的建议标准。

IntServ 可对单个的应用会话提供服务质量的保证，其主要特点有二：

（1）**资源预留**。一个路由器需要知道不断出现的会话已经预留了多少资源（即链路带宽和缓存空间）。

（2）**呼叫建立**。一个需要服务质量保证的会话必须首先在源站到目的站的路径上的**每一个**路由器预留足够的资源，以保证其端到端的服务质量的要求。因此在一个会话开始之前必须先有一个**呼叫建立**（又称为**呼叫接纳**）过程，它需要在其分组传输路径上的每一个路由器都参加。每一个路由器都要确定该会话所需的本地资源是否够用，同时还不要影响到已经建立的会话的服务质量。

IntServ 定义了两类服务：

（1）**有保证的服务**(guaranteed service)，可保证一个分组在通过路由器时的排队时延有一

个严格的上限。

（2）**受控负载的服务**(controlled-load service)，可以使应用程序得到比通常的“尽最大努力”更加可靠的服务。

IntServ 共有以下四个组成部分：

（1）**资源预留协议 RSVP**，它是 IntServ 的**信令协议**。

（2）**接纳控制**(admission control)，用来决定是否同意对某一资源的请求。

（3）**分类器**(classifier)，用来将进入路由器的分组进行分类，并根据分类的结果将不同类别的分组放入特定的队列。

（4）**调度器**(scheduler)，根据服务质量要求决定分组发送的前后顺序。

一个会话必须首先声明它所需的服务质量，以便使路由器能够确定是否有足够的资源来满足该会话的需求。资源预留协议 RSVP 在进行资源预留时采用了多播树的方式。发送端发送 PATH 报文（即存储路径状态报文）给所有的接收端指明通信量的特性。每个中间的路由器都要转发 PATH 报文，而接收端用 RESV 报文进行响应。路径上的每个路由器对 RESV 报文（即资源预留请求报文）的请求都可以拒绝或接受。当请求被某个路由器拒绝时，路由器就发送一个差错报文给接收端，从而终止了这一信令过程。当请求被接受时，链路带宽和缓存空间就被分配给这个分组流，而相关的流(flow)状态信息就保留在路由器中。“流”是在多媒体通信中的一个常用的名词，一般定义为“**具有同样的源 IP 地址、源端口号、目的 IP 地址、目的端口号、协议标识符及服务质量需求的一连串分组**”。

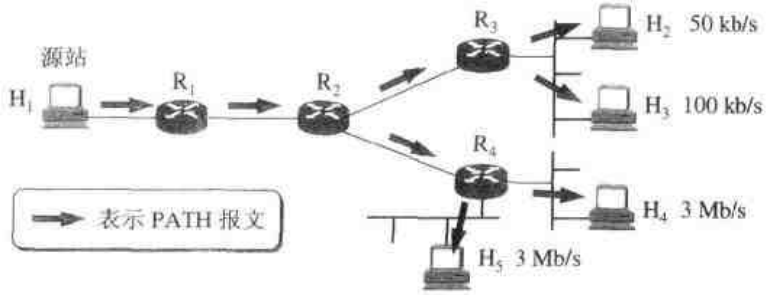
图 10-18 用一个简单例子说明 RSVP 协议的要点。设主机 H_1 要向因特网上的四个主机 H_2～H_5 发送多播视频节目，在图中这四个主机右边标注的数据率就是这些主机打算以这样的数据率来接收 H_1 发送的视频节目。这个视频节目可使用不同的数据率来接收。用较低数据率接收时，图像和声音的质量也就较差。

主机 H_1 先以多播方式从源点 H_1 向下游方向发送 PATH 报文，如图 10-18(a)所示。当 PATH 报文传送到多播路径终点的四个主机（即叶结点）时，每一个主机就向多播路径的上游发送 RESV（报文），指明在接收该多播节目时所需的服务质量等级。路由器若无法预留 RESV 报文所请求的资源，就返回差错报文。若能预留，则将下游传来的 RESV 报文合并构成新的 RESV 报文，传送给自己的上游路由器，最后传送到源点主机 H_1。这些情况如图 10-18(b)所示。因此，RSVP 协议是**面向终点的**。

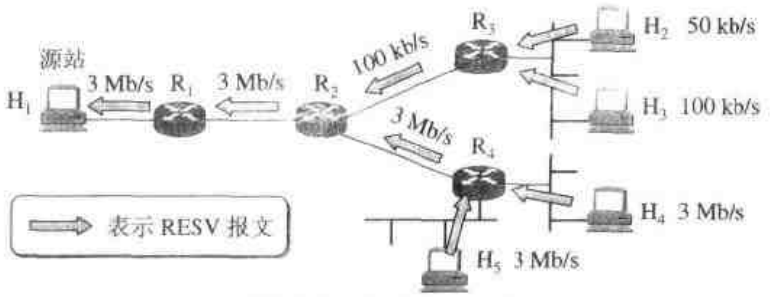
需要注意的是，路由器合并下游的 RESV 报文并不是把下游提出的预留数据率简单地相加而是取其中的较大的数值。例如，路由器 R_4 收到两个预留 3 Mb/s 的 RESV 报文，但 R_4 向 R_2 发送的 RESV 报文只要求预留 3 Mb/s 而不是 6 Mb/s（因为向下游方向发送数据是采用可以节省带宽的多播技术）。同理，R_3 向 R_2 发送的 RESV 报文要求预留 100 kb/s 而不是 150 kb/s。最后，R_1 向 H_1 发送的 RESV 报文要求预留 3 Mb/s。当 H_1 收到返回的 RESV 报文后，就开始发送视频数据报文了。

IntServ/RSVP 使得因特网的体系结构发生了根本的变化，因为这使得因特网不再是提供“尽最大努力交付”的服务。在有关服务质量的协议中，RSVP 是最复杂的。

IntServ/RSVP 所基于的概念是端系统中与分组流有关的状态信息。各路由器中的预留信息**只存储有限的**时间（这称为**软状态** soft-state），因而各终点对这些预留信息**必须定期进行更新**。我们还应注意到，RSVP 协议不是运输层协议而是个网络层的控制协议。RSVP 不携带应用数据。图 10-19 给出了在路由器中实现的 IntServ 体系结构。



(a) 源点用多播发送 PATH 报文



(a) 各终点向源点返回 RESV 报文

图 10-18　RSVP 协议的工作原理

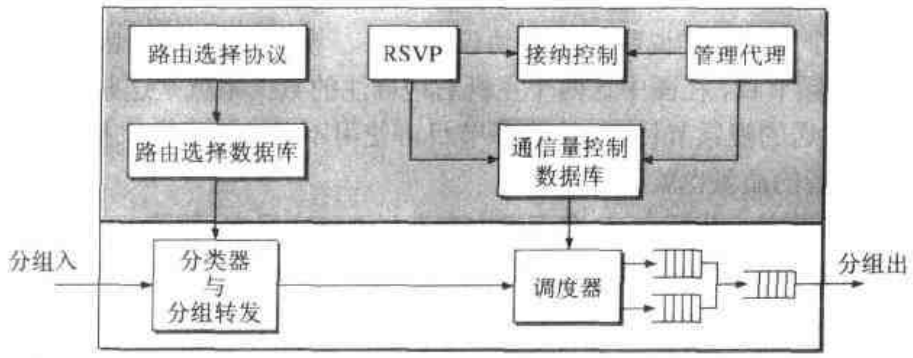


图 10-19　IntServ 体系结构在路由器中的实现

IntServ 体系结构分为前台和后台两个部分。前台部分画在下面，包括两个功能块，即分**类器与分组转发**和分组的**调度器**。每一个进入路由器的分组都要通过这两个功能块。后台部分画在上面（有灰色阴影的部分），包括四个功能块和两个数据库。这四个功能块是：①路由选择协议，负责维持路由选择数据库。由此可查找出对应于每一个目的地址和每一个流的下一跳地址。②RSVP 协议，为每一个流预留必要的资源，并不断地更新通信量控制数据库。③接纳控制，当一个新的流产生时，RSVP 就调用接纳控制功能块，以便确定是否有足够的资源可供这个流使用。④管理代理，用来修改通信量控制数据库和管理接纳控制功能块，包括设置接纳控制策略。

综合服务 IntServ 体系结构存在的主要问题是：

（1）**状态信息的数量与流的数目成正比**。例如，对于 OC-48 链路(2.5Gb/s)上的主干网路由器，通过 64 kb/s 的音频流的数目就超过 39000 个。如果对数据率再进行压缩，则流的数目就更多。因此在大型网络中，按每个流进行资源预留会产生很大的开销。

（2）IntServ 体系结构**复杂**。若要得到有保证的服务，**所有的路由器都必须装有** RSVP、

接纳控制、分类器和调度器。这种路由器称为 RSVP 路由器。在应用数据传送的路径中只要有一个路由器是非 RSVP 路由器，整个的服务就又变为“尽最大努力交付”了。

（3）综合服务 IntServ 所定义的**服务质量等级数量太少，不够灵活**。

10.4.4 区分服务 DiffServ

1．区分服务的基本概念

由于综合服务 IntServ 和资源预留协议 RSVP 都较复杂，很难在大规模的网络中实现，因此 IETF 提出了一种新的策略，即**区分服务** DiffServ (Differentiated Services) [RFC 2475] [W-DiffServ]。区分服务有时也简写为 DS。因此，具有区分服务功能的结点就称为 DS 结点。

区分服务 DiffServ 的要点如下：

（1）DiffServ 力图不改变网络的**基础结构，但在路由器中增加区分服务的功能**。因此，DiffServ 将 IP 协议中原有 8 bit 的 IPv4 的服务类型字段和 IPv6 的通信量类字段重新定义为区分服务字段 DS。路由器根据 DS 字段的值来处理分组的转发。因此利用 DS 字段的不同数值就可提供不同等级的服务质量。根据因特网的建议标准[RFC 2474]，DS 字段现在只使用占 6 bit 的前一部分——**区分服务码点** DSCP (Differentiated Services CodePoint)。再后面的两个比特记为 CU (Currently Unused)，表示暂时不使用。因此由 DS 字段的值所确定的服务质量实际上就是由 DSCP 的值来确定（见图 10-20）。

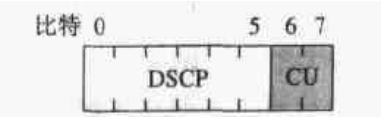


图 10-20 区分服务码点 DSCP 占 DS 字段的前 6bit

在使用 DS 字段之前，因特网的 ISP 要和用户商定一个**服务等级协定** SLA (Service Level Agreement)。在 SLA 中指明了被支持的服务类别（可包括吞吐量、分组丢失率、时延和时延抖动、网络的可用性等）和每一类别所容许的通信量。

（2）网络被划分为许多个 DS 域(DS Domain)。一个 DS 域在一个管理实体的控制下实现同样的区分服务策略。DiffServ **将所有的复杂性放在 DS 域的边界结点**(boundary node)**中，而使 DS 域内部路由器工作得尽可能地简单**。边界结点可以是主机、路由器或防火墙等。为了简单起见，下面只讨论边界结点是边界路由器的情况（原理都是一样的）。图 10-21 给出了 DS 域、**边界路由器**(boundary router)和**内部路由器**(interior router)的示意图。图中标有 B 的路由器都是边界路由器。

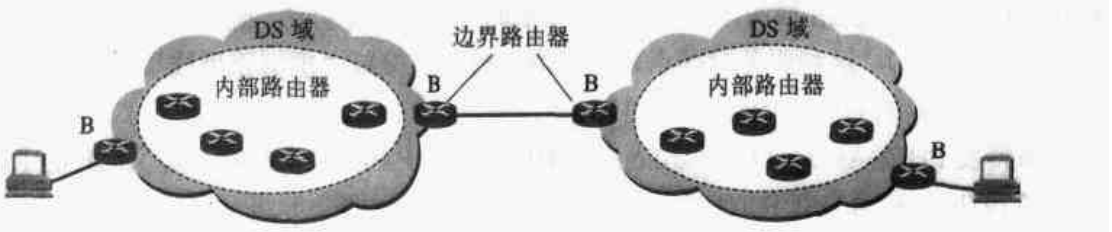


图 10-21 DS 域、边界路由器和内部路由器的示意图

（3）边界路由器中的功能较多，可分为**分类器**(classifier)和**通信量调节器**(conditioner)两大部分。调节器又由**标记器**(marker)、**整形器**(shaper)和**测定器**(meter)三个部分组成。分类器根据分组首部中的一些字段（如源地址、目的地址、源端口、目的端口或分组的标识等）对分组进行分类，然后将分组交给标记器。标记器根据分组的类别设置 DS 字段的值。以后在分组的转发过程中，就根据 DS 字段的值使分组得到相应的服务。测定器根据事先商定的 SLA 不断地测定分组流的速率（与事前商定的数值相比较），然后确定应采取的行动，例如，可重新打标记或交给整形器进行处理。整形器中设有缓存队列，可以将突发的分组峰值速率平滑为较均匀的速率，甚至丢弃一些分组。在分组进入内部路由器后，路由器就根据分组的 DS 值进行转发。图 10-22 给出了边界路由器中的各功能块的关系。

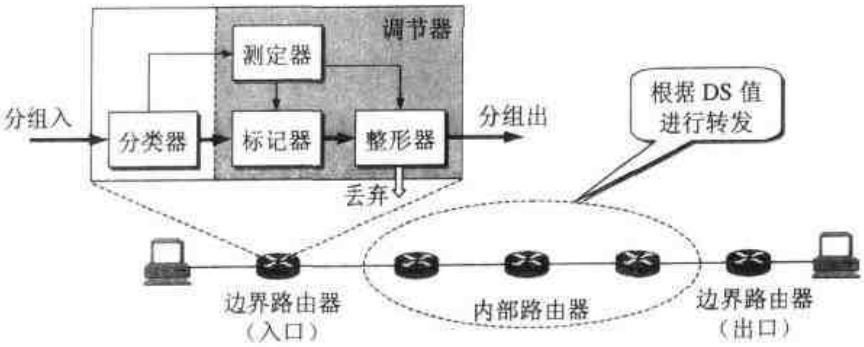


图 10-22　边界路由器中的各功能块的关系

（4）DiffServ 提供了一种**聚合**(aggregation)功能。DiffServ 不是为网络中的每一个流维持供转发时使用的状态信息，而是将若干个流根据其 DS 值聚合成少量的流。路由器对相同 DS 值的流都按相同的优先级进行转发。这就大大简化了网络内部的路由器的转发机制。区分服务 DiffServ 不需要使用 RSVP 信令。

2．每跳行为 PHB

DiffServ 定义了在转发分组时体现服务水平的**每跳行为 PHB** (Per-Hop Behavior)。所谓“**行为**”就是指在转发分组时路由器对分组是怎样处理的。“行为”的例子可以是：“首先转发这个分组”或“最后丢弃这个分组”。“每跳”是强调这里所说的行为只涉及到本路由器转发的这一跳的行为，而下一个路由器再怎样处理则与本路由器的处理无关。这和 IntServ/RSVP 考虑的服务质量是“端到端”的很不一样。

IETF 的 DiffServ 工作组已经定义了两种 PHB，即**迅速转发** PHB 和**确保转发** PHB。

迅速转发 PBH (Expedited Forwarding PHB)可记为 EF PHB，或 EF。定义 EF 的 RFC 文档是[RFC 3246]。EF 指明离开一个路由器的通信量的数据率必须等于或大于某一数值。因此 EF PHB 用来构造通过 DS 域的一个低丢失率、低时延、低时延抖动、确保带宽的端到端服务（即不排队或很少排队）。这种服务对端点来说，像点对点连接或“**虚拟租用线**”，又称为 Premium 服务。对应于 EF 的 DSCP 的值是 101110。

确保转发 PHB (Assured Forwarding PHB)可记为 AF PHB，或 AF。定义 AF 的 RFC 文档是[RFC 2597]。AF 用 DSCP 的比特 0～2 将通信量划分为四个等级（分别为 001, 010, 011 和 100），并给每一种等级提供最低数量的带宽和缓存空间。对于其中的每一个等级再用 DSCP 的比特 3～5 划分出三个“丢弃优先级”（分别为 010, 100 和 110，从最低丢弃优先级到最高

丢弃优先级)。当发生网络拥塞时，对于每一个等级的 AF，路由器就首先将“丢弃优先级”较高的分组丢弃。AF 可以与 7.4.6 节的 RED 结合起来使用（见[PETE00]）。

从以上所述可看出，区分服务 DiffServ 比较灵活，因为它并没有定义特定的服务或服务类别。当新的服务类别出现而旧的服务类别不再使用时，DiffServ 仍然可以工作。

10.5 多协议标记交换 MPLS

10.5.1 MPLS 的产生背景

在 20 世纪 90 年代问世的面向连接的 ATM 技术在传送实时数据时能够保证服务质量 QoS。这在当时是一种突破性的技术进展。但 ATM 网络未能取代现有的电信网络和计算机网络。这不仅是因为 ATM 网络价格昂贵，用 ATM 网络来替换现有的已投资非常巨大的电信网络和计算机网络是很不现实的，而且还因为 ATM 网络和上层的应用结合得很不好。相反，基于 IP 的因特网与各种应用已经结合得很好。因此 ATM 网络必须与 IP 网络相结合才有出路。由于因特网上通信量持续迅速增长[①]，因此在 90 年代中期 ATM 交换机已广泛地使用在宽带因特网的主干网中。图 10-23(a)表示 ATM 主干网是怎样和路由器相连接的示意图。

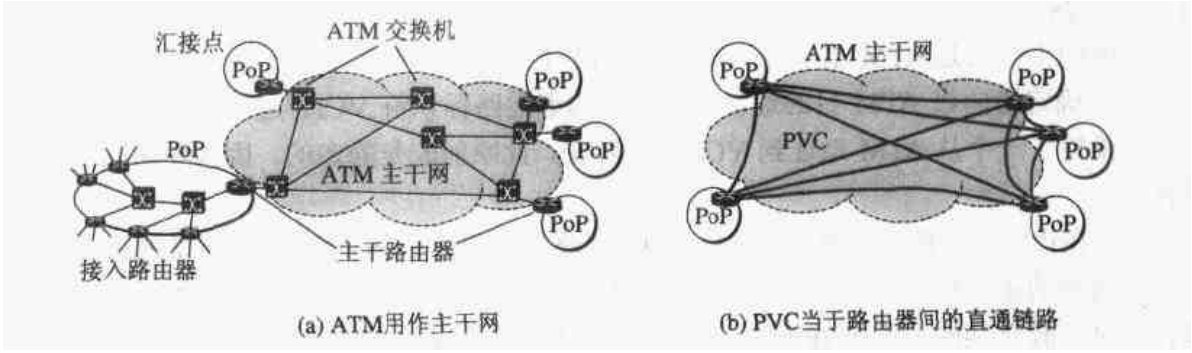


图 10-23 在因特网中使用 ATM

由于 ATM 主干网的速率很高，因此大量的低速路由器需要通过一些**汇接点** PoP (Point of Presence)进行汇接后才能和 ATM 主干网相连接。一个汇接点包含有两种路由器。一种是接入路由器(access router)或边沿路由器(border router)，这类路由器数量较多且速率较低。另一种是主干路由器(backbone router)或核心路由器(core router)，这类路由器在一个汇接点中一般只有一个，且速率较高。在汇接点中这两种路由器常通过 ATM 交换机相连。图 10-23(a)左下方放大了的汇接点表示出五个低速率的接入路由器通过 ATM 交换机汇接到一个高速率的主干路由器。

图 10-23(b)表示各汇接点的主干路由器之间跨越主干网的通信一般都是使用 ATM 网络的**永久虚通路** PVC，这样做的好处是避免在每一次使用虚通路时还要先建立虚通路和最后再

① 注：过去计算机网络有一个“80/20”规律，即 80%的通信量在局域网范围，而只有 20%的通信量在广域网上传送。因此过去的概念“局域网速率高，广域网速率低”与客观需求是适应的。然而现在由于因特网的发展，上述规律反过来了，变成为“20/80”，即 80%的通信量要经过广域网来传送。

释放这些虚通路。虽然ATM主干网中的交换机数可多达几十个甚至几百个，但正如图10-23(b)所示的那样，这些在汇接点之间的虚通路就像在IP层的下面有一条直通链路一样将主干网边上的主干路由器都互连起来（而不管实际的ATM虚通路要经过多少个ATM交换机）。这种把ATM看成是数据链路层并在其上面运行IP协议的方式称为**覆盖模型**。在具体实现时，覆盖模型又有传统IPOA (Classical IP Over ATM)，局域网仿真LANE (LAN Emulation)和MPOA (MultiProtocol Over ATM)等几种不同的方案。

显然，在ATM主干网边上的各主干路由器必须同时具有两种地址，一种是为了和IP网上的其他路由器通信的IP地址，而另一种则是为了和ATM主干网中的ATM交换机通信的ATM地址。ATM地址与IP地址不同，它使用ISO制订的20字节的网络服务访问点NSAP编码。有时我们根据IP数据报是进入ATM主干网还是从ATM主干网离开，将这些主干路由器分别称为“**入口路由器**”或“**出口路由器**”。

当IP数据报通过入口路由器要进入ATM主干网时，入口路由器要做以下几件事：

（1）根据数据报的目的地址从路由表中查找出下一跳路由器的IP地址，也就是ATM主干网上的某一个出口路由器的IP地址。

（2）入口路由器将ATM主干网看成是IP层下面的数据链路，并从ATM的ARP表中，利用刚才得到的出口路由器的IP地址，查找出该出口路由器的ATM地址。

（3）入口路由器将得到的出口路由器的ATM地址与IP数据报一起交给数据链路层（即ATM主干网）。

下面的工作就是ATM主干网的事了。ATM主干网要做两件事：

（1）确定通向该ATM目的地址的虚通路的虚通路标识符VCI。这很容易做到，因为在发送端维持了一个从ATM地址到VCI的映射表。此映射表是静态的，因为使用的是永久虚通路。

（2）在该虚通路的发送端（即入口路由器）将数据报分割并装配成53字节的ATM信元，而在虚通路的接收端（即出口路由器）将收到的ATM信元重装成原来的IP数据报。这里的过程还是比较复杂的，因为IP层先要将IP数据报交给AAL层。在AAL层根据不同的服务类别和所使用的AAL类型，加上相应的AAL首部后，将数据报分割成48字节的数据单元，交给ATM层。ATM层再加上5字节的首部，才得到最后的ATM信元。

然而这种覆盖模型很难协调IP和ATM网络的巨大差异，这主要是因为：**IP是无连接的，而ATM是面向连接的；IP只提供尽最大努力交付的服务，而ATM能确保服务质量QoS。**在20世纪90年代中期以前，扩大因特网主干网容量的惟一选择是使用ATM技术。但随着网络规模的不断扩大，和ATM主干网连接的主干路由器的数目就大大增多，以致需要建立非常多的永久虚通路，因而导致难以维持庞大的ATM地址到VCI的映射表。这就是所谓的N^2问题[①]。如果需要增加一个主干路由器，那么就需要从这个路由器再建立到所有其他主干路由器的永久虚通路。另外，从分组转换为信元时每一个信元的5字节首部开销相当大，这常称为**信元税**(cell tax)。另外，同时维护两种体系结构完全不同的网络也很不方便。ATM的发展又面临困难。

① 注：严格说来，应当是$N(N-1)$问题，这里的N是ATM主干网边上的主干路由器的数目。每两个主干路由器之间的永久虚通路都是每个方向各一条。例如，当主干路由器在5个的基础上增加1个时，虚通路数从20增加到30，增加了10条。但若主干路由器在100个的基础上增加1个时，虚通路数就从9900增加到10100，需要增加200条。

在这种情况下，IETF 于 1997 年成立了 MPLS 工作组，旨在开发出一种将第三层的路由选择功能与面向连接的第二层的交换功能综合在一起的新的协议标准，以便使 IP 和 ATM 结合得更好些。这种新的协议的名称是**多协议标记交换** MPLS (MultiProtocol Label Switching)[①]，它采用**综合模型**，能够克服上述的覆盖模型的一些缺点。IETF 还综合了许多公司的技术，如 Cisco 公司的**标记交换 TAG** (TAG Switching)，Ipsilon 公司的 **IP 交换**(IP Switching)等[COMM99]。2001 年 1 月 MPLS 终于成为因特网的建议标准[RFC 3031, 3032] [W-MPLS]。

然而现在市场的情况已有了很大的变化。以前在高速交换领域 ATM 交换机处于垄断地位。但现在的高速路由器的转发分组速率也可以和 ATM 交换机一样快。因此 ATM 已经不再是高速网络的惟一选择技术。这样，在同一网络中同时使用 IP 和 ATM 已不再成为必须的。相反，一些高速网络正在试验不用 ATM 而直接将 IP 数据报放入到 SONET/SDH 链路（这称为 IP over SONET/SDH）。在这种情况下，MPLS 仍然受到网络界的人士的高度重视。这里的原因就是 MPLS 在以下四个方面具有其特殊的功能：

（1）**支持面向连接的服务质量。**

（2）**支持流量工程，平衡网络负载。**

（3）**有效地支持虚拟专用网 VPN。**

（4）**支持多种网络协议。**

下面就开始讨论 MPLS 的基本工作原理。

10.5.2 MPLS 的工作原理

1．基本工作过程

在传统的 IP 网络中，分组每到达一个路由器，都必须查找路由表，并按照“最长前缀匹配”的原则找到下一跳的 IP 地址（而前缀的长度是不确定的）。当网络很大时，查找含有大量项目的路由表要花费很多的时间。在出现突发性的通信量时，往往还会使缓存溢出，这就引起分组丢失、传输时延增大和服务质量下降。

MPLS 的一个重要特点就是不用长度可变的 IP 地址前缀来查找转发表中的匹配项目，而**使用很简单的转发算法对打上固定长度“标记”的分组用硬件进行转发，这就使得**在分组转发的过程中省去了每到达一个结点都要上第三层用软件查找路由表的过程，因而分组转发的速率就大大地加快了。采用硬件技术对打上标记的分组进行转发就称为**标记交换**[②]。“交换”也表示在转发分组时不再上升到第三层，并且用软件分析第三层的 IP 首部和查找转发表，而**是根据第二层的标记用硬件进行转发。**MPLS 并不限于使用 ATM，它可以使用多种链路层协议，如 IPX, DECnet, PPP, 以太网以及帧中继等。因此这种标记交换是“**多协议**”的。图 10-24 是 MPLS 协议的基本原理的示意图。

MPLS 域(MPLS domain)是指该域中有许多相邻的路由器，并且所有的路由器都能支持 MPLS 技术。支持 MPLS 的标记交换功能的路由器称为**标记交换路由器** LSR (Label Switching Router)。LSR 之所以需要同时具有标记交换和路由选择这两种功能，是因为 LSR 使用其标记

① 注：label 的标准译名本来是“标号”，但目前在 MPLS 中将 label 译成为“标记”是最流行的。也还有译为“标签”的。

② 注：有的公司愿意用“第三层交换”表示“第三层的路由选择功能加上第二层的交换功能”，但这样的术语不够明确，因而编者不主张使用这一术语。

交换功能对分组进行快速转发。但在转发分组之前 LSR 需要使用路由选择功能构造分组转发表。

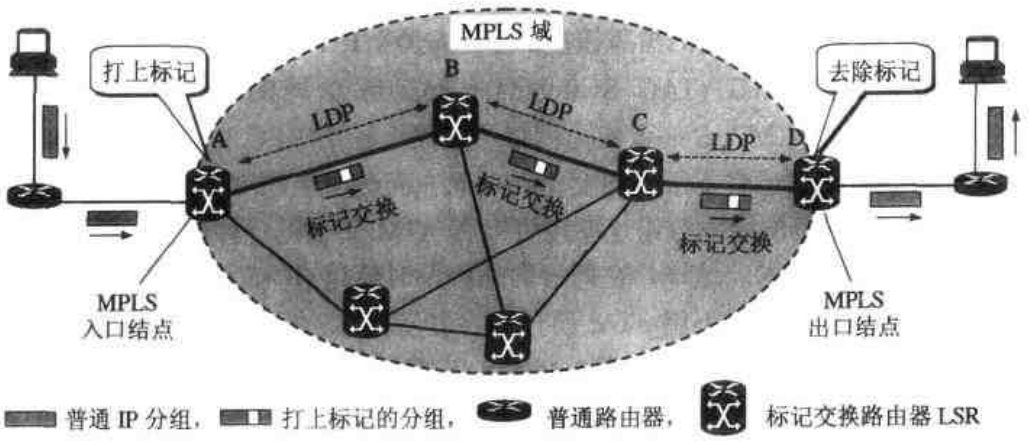


图 10-24　MPLS 协议的基本原理

这样，在图 10-24 中给出了 MPLS 的基本工作过程如下：

（1）MPLS 域中的各 LSR 使用专门的**标记分配协议 LDP** (Label Distribution Protocol) 交换报文，并找出和特定标记相对应的路径，即**标记交换路径** LSP (Label Switched Path)。例如在图中的路径 A→B→C→D。各 LSR 根据这些路径构造出分组转发表。这个过程和路由器构造自己的路由表相似，限于篇幅，这里不讨论转发表构造的详细步骤。

（2）当一个分组进入到 MPLS 域时，MPLS **入口结点**(ingress node)就将分组打上标记，并按照转发表将分组转发给下一个 LSR。

（3）以后的所有 LSR 都按照标记进行转发。每经过一个 LSR，要换一个新的标记。

（4）当分组离开 MPLS 域时，MPLS **出口结点**(egress node)就将分组的标记去除。再以后就按照一般分组的转发方法进行转发。

上述的这种由入口 LSR 来确定分组在进入 MPLS 域以后的转发路径称为**显式路由选择** (explicit routing)，它和因特网中通常使用的“每一个路由器逐跳进行路由选择”有着很大的区别。

下面再讨论 MPLS 中的几个重要概念。

2．转发等价类 FEC

MPLS 有个很重要的概念就是**转发等价类** FEC (Forwarding Equivalence Class)。所谓“转发等价类”就是路由器**按照同样方式对待**的分组的集合。这里“按照同样方式对待”表示从同样接口转发到同样的下一跳地址，并且具有同样服务类别和同样丢弃优先级等。FEC 的例子是：

（1）目的 IP 地址与某一个特定 IP 地址的前缀匹配的分组（这就相当于普通的 IP 路由器）；

（2）所有源地址与目的地址都相同的分组；

（3）具有某种服务质量需求的分组。

总之，划分 FEC 的方法不受什么限制，这都由网络管理员来控制，因此非常灵活。入口结点并不是给每一个分组指派一个不同的标记，而是将属于同样 FEC 的分组都指派同样的标记。FEC 和标记是一一对应的关系。

显然，FEC 可以有不同的粒度。细粒度的例子是为特定源主机和目的主机之间的特定应用指派的 FEC，而与特定出口 LSR（不管数据流是从哪一个源结点发送过来的）相关联的 FEC 则是粗粒度的例子。在这种情况下许多应用流聚合到出口 LSR 离开 MPLS 域，像一颗倒置的树，它的根在出口 LSR。

图 10-25 给出一个将 FEC 用于负载平衡的例子。图 10-25(a)的主机 H_1 和 H_2 分别向 H_3 和 H_4 发送大量数据。路由器 A 和 C 是数据传输必须经过的。但传统的路由选择协议只能选择最短路径 A→B→C，这就导致这段最短路径过载。

图 10-25(b)表示在 MPLS 的情况下，入口结点 A 可设置两种 FEC："源地址为 H_1 而目的地址为 H_3 的分组"和"源地址为 H_2 而目的地址为 H_4 的分组"，同时使前一种 FEC 的路径是 H_1→A→B→C→H_3，而后一种的路径是 H_2→A→D→E→C→H_4。这样可使网络的负载较为平衡。网络管理员采用自定义的 FEC 就可以更好地管理网络的资源。这种均衡网络负载的做法也称为**流量工程**(traffic engineering)[①]或**通信量工程**。

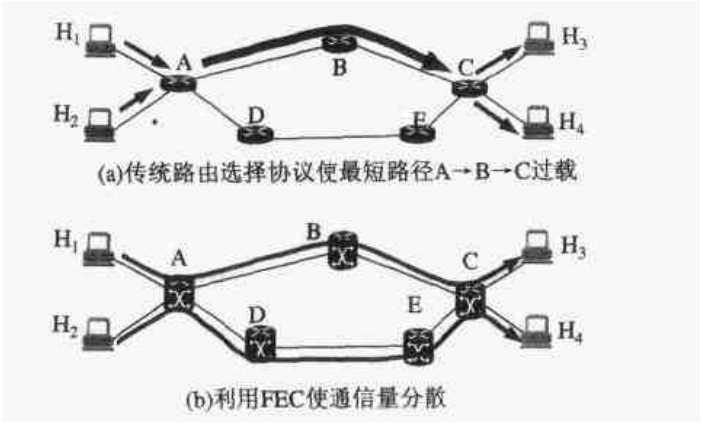


图 10-25　FEC 用于负载平衡

3．标记栈

MPLS 的一个重要功能就可以构成**标记栈**(label stack)。图 10-26 给出了 MPLS 标记的格式以及标记栈在 MPLS 帧中的位置。

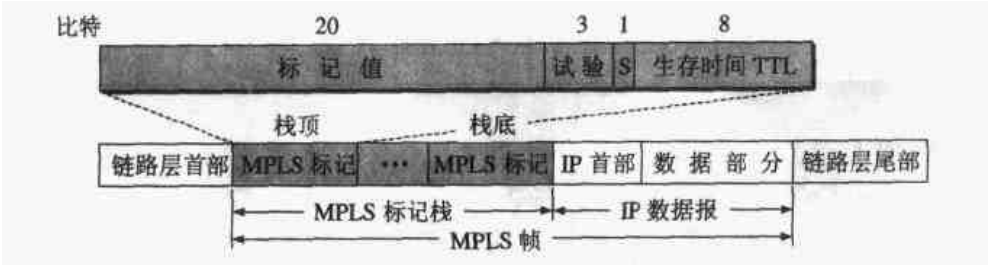


图 10-26　MPLS 的标记和标记栈

从图 10-26 可看出，MPLS 的标记共有 4 字节长，包括以下四个字段：

（1）**标记值**　　占 20 bit。

（2）**试验**　　占 3 bit。目前保留用作试验。

① 注：流量工程是对网络上的通信量进行测量、建模和控制，使网络运行的性能得到最优化。

（3）**栈 S**　　占 1 bit。当标记栈还有老的标记时就置为 1，否则为 0。

（4）**生存时间 TTL**　　占 8 bit，用来防止 MPLS 帧在 MPLS 域中兜圈子。

MPLS 标记一旦产生就压入到标记栈中，而整个标记栈放在数据链路层首部和 IP 首部之间。栈是一种后进先出的数据结构。MPLS 协议规定，标记栈的栈顶（最后进入栈的标记）最靠近数据链路层首部，而栈底最靠近 IP 首部。**在最简单的情况下，标记栈中只有一个标记。**

当数据链路层使用 ATM 时，标记值就是 ATM 首部中 VPI/VCI 的值。当使用帧中继时，标记值就是帧中继首部中的 DLCI 值。

MPLS 的标记栈用于当 MPLS 域出现嵌套的情况，见图 10-27。

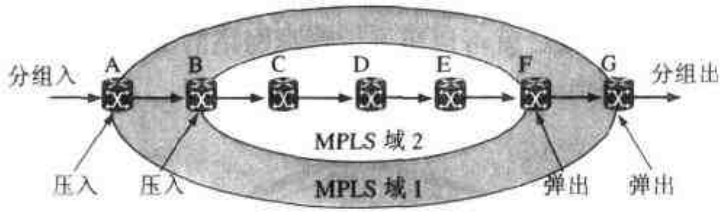


图 10-27　MPLS 标记栈的使用

假定分组进入 MPLS 域 1 的 LSR A，并且要经过 B 和 F 到达 G。这个分组在 MPLS 域 1 中的标记交换路径 LSP 是“A→B→F→G”。分组在到达入口结点 LSR A 时被压入一个标记。但分组到达 LSR B 时就进入了 MPLS 域 2。在域 2 中的标记交换路径 LSP 是“B→C→D→E→F”，因此 LSR B 要压入另一个标记。当分组到达 LSR F 时就弹出栈顶的标记。当分组到达 LSR G 时弹出标记栈剩下的一个标记。分组实际上通过的路径是“A→B→C→D→E→F→G”。

4．标记对换

标记交换路由器 LSR 在转发分组时要进行**标记对换**(label swapping)[①]。图 10-28 给出了使用标记对换进行分组转发的示意图。

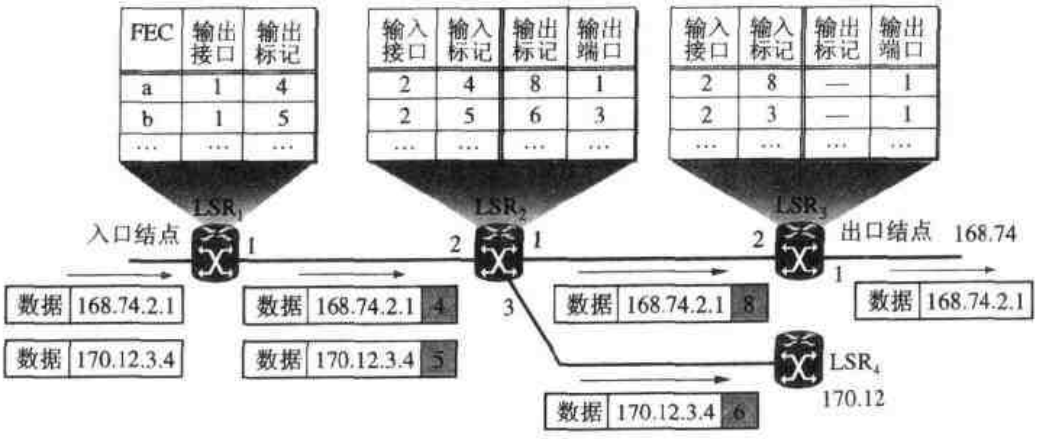


图 10-28　使用标记对换进行分组转发

① 注：这里使用]RFC 3031]中的标准词汇。“对换”和“交换”的意思相近，但“对换”强调是两个标记互相对换。MPLS 中的 LS 是“标记交换”，但标记交换路由器 LSR 实现的功能是“标记对换”。

标记对换的概念在图中已清楚地表示出。这里只需再强调一下，在入口结点对分组打标记，在出口结点要将分组中的标记拿走。只有在内部结点才进行标记对换。**标记只具有本地意义**。分组传送到下一个 LSR 就进行标记对换，原有的标记就消失了。

10.6 居民接入网 RAN

居民接入网 RAN (Residential Access Network)是近年来发展很快的一个领域，它又称为**居民宽带网** RBB (Residential BroadBand Network)。所谓居民接入网就是从居民住宅至某个 ISP 之间的一个中间网络，它起到接口作用，目的是使用户能够方便和经济地享用各种宽带多媒体信息。

现在居民接入网 RAN 往往是宽带上网的一个瓶颈。居民接入网有多种实现方案，而一些厂商也在寻找机会在许多国家或地区进行现场试验。但居民接入网到底应采用何种技术方案，目前既无定论，也无统一的国际标准。因此下面只能向读者概括介绍几种最主要的实现方案。本节未介绍无线接入网。但无线接入在目前也发展很快。对于一时难于铺设光缆或电缆的居民区，采用无线接入无疑是一种很好的方法，应得到足够的重视。

10.6.1 xDSL 技术

xDSL 技术就是用数字技术对现有的模拟电话用户线进行改造，使它能够承载宽带业务。DSL 是数字用户线(Digital Subscriber Line)的缩写。而字母 x 表示 DSL 的前缀可以是多种不同字母，用不同的前缀表示在数字用户线上实现的不同宽带方案。表 10-3 是 xDSL 的几种类型。其中 ADSL (Asymmetric Digital Subscriber Line)是**非对称数字用户线**[W-ADSL]，HDSL (High speed DSL)是**高速数字用户线**，SDSL (Single-line DSL)是 **1 对线的数字用户线**，VDSL (Very high speed DSL)是**甚高速数字用户线**，而 DSL 是使用 ISDN 用户线。

表 10-3 xDSL 的几种类型

xDSL	对称性	下行带宽	上行带宽	极限传输距离
ADSL	非对称	1.5 Mb/s	64 kb/s	4.6~5.5 km
ADSL	非对称	6~8 Mb/s	640 kb/s ~ 1 Mb/s	2.7~3.6 km
HDSL(2 对线)	对称	1.5 Mb/s	1.5 Mb/s	2.7~3.6 km
HDSL(1 对线)	对称	768 kb/s	768 kb/s	2.7~3.6 km
SDSL	对称	384 kb/s	384 kb/s	5.5 km
SDSL	对称	1.5 Mb/s	1.5 Mb/s	3 km
VDSL	非对称	12.96 Mb/s	1.6~2.3 Mb/s	1.4 km
VDSL	非对称	25 Mb/s	1.6~2.3 Mb/s	0.9 km
VDSL	非对称	52 Mb/s	1.6~2.3 Mb/s	0.3 km
DSL(ISDN)	对称	160 kb/s	160 kb/s	4.6~5.5 km

表 10-3 中的极限传输距离与用户线的有很大的关系。下面仅对 ADSL 进行简单介绍（有的文献还介绍**速率自适应 DSL**，即 RADSL (Rate-Adaptive DSL)，它是 ADSL 的一个子集，可自动调节线路速率）。

考虑到用户在使用万维网时需要的宽带业务主要是从因特网网点下载多媒体信息，而向万维网网点发送的信息仅需很小的带宽，因此 ADSL 将上行和下行带宽做成不对称的。这样

既经济又能满足需求。ADSL 仍使用现有的一对用户线（铜线），只是在用户线两端各安装一个 ADSL 调制解调器。该 ADSL 调制解调器用频分复用的方法，划分出 3 个频段（图 10-29）。最低的频段是 0～4 kHz，用来传送传统电话信号。然后是 20～50 kHz 的频段，用来传送上行数字信息。从 150～500 kHz 或 140 kHz 到 1.1 MHz 则用于速率为 1.5 Mb/s 或 6.3 Mb/s 下行数字信息的传送。还有一种方案是使下行频段的低端与上行频段重叠，这样可使下行频段更宽，但这时必须使用回波抵消技术。

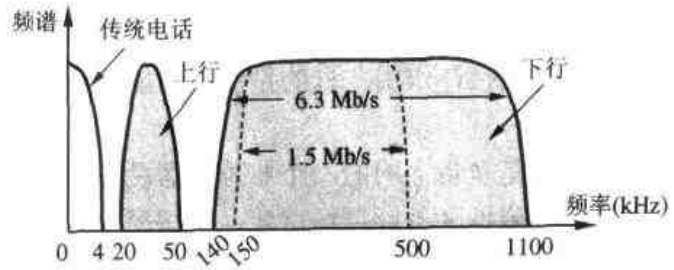


图 10-29　ADSL 调制解调器的频谱利用的划分

ADSL 主要采用两种调制技术。一种是**无载波振幅相位调制 CAP** (Carrierless Amplitude Phase)，而另一种是**离散多音调制 DMT** (Discrete Multi-Tone)。这两种技术的性能相差并不多。CAP 早就使用在第一代 ADSL 调制解调器中，在商品化方面更加成熟，目前已占领了市场。但 DMT 则被美国 ANSI 选择为 ADSL 的标准。DMT 将可用带宽划分为 256 个子信道(每个子信道带宽为 4 kHz)，然后将数据自适应地动态分配给每个子信道，这就使得在约 1 MHz 的可用频带内实现超过 6 Mb/s 的数据率。ADSL 技术还支持**前向纠错** FEC (Forward Error Correction)。实现 CAP 造成的时延为 4～10 ms，而 DMT 是 15～25 ms。

基于 ADSL 的接入网由以下三大部分组成：**数字用户线接入复用器** DSLAM (DSL Access Multiplexer)，用户线和用户家中的一些设施（图 10-30）。数字用户线接入复用器包括许多 ADSL 调制解调器。ADSL 调制解调器又称为**接入端接单元** ATU (Access Termination Unit)。由于 ADSL 调制解调器必须成对使用，因此在电话端局（或远端站）和用户家中所用的 ADSL 调制解调器分别记为 ATU-C (C 代表**端局** Central Office)和 ATU-R(R 代表**远端** Remote)。用户电话通过**电话分路器** PS (POTS Splitter)[①]和 ATU-R 连在一起，经用户线到端局，并再

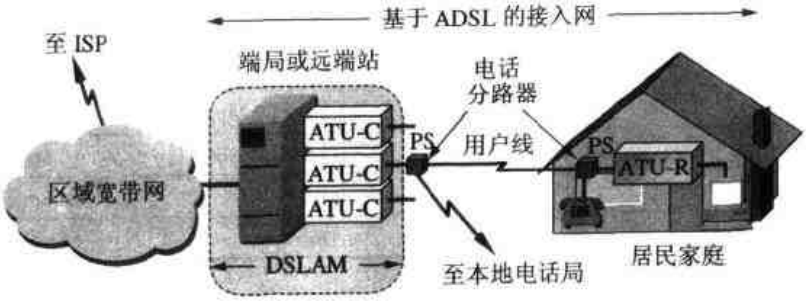


图 10-30　基于 ADSL 的接入网的组成

① 注：POTS 代表 Plain Old Telephone Service，即传统电话。

次经过一个电话分路器 PS 将电话连到本地电话交换机。电话分路器 PS 是无源的，它利用低通滤波器将电话信号与数字信号分开。电话分路器做成无源的是为了在停电时不影响传统电话的使用。一个 DSLAM 可支持多达 500～1000 个用户。若按 6 Mb/s 计算，则具有 1000 个端口的 DSLAM（这就需要用 1000 个 ATU-C）应有高达 6 Gb/s 的转发能力。因 ATU-C 要使用数字信号处理技术，因此 DSLAM 的价格较高。

ADSL 最大的好处就是可以利用现有电话网中的用户线。用户线在整个电话网的投资中约占一半，不能轻易丢弃。但 ADSL 的缺点是目前的价格仍偏高。

10.6.2 光纤同轴混合网(HFC 网)

一种叫做**光纤同轴混合网**(HFC 网)在 1988 年被提出。HFC 是 Hybrid Fiber Coax 的缩写。HFC 网是在目前覆盖面很广的有线电视网 CATV 的基础上开发的一种居民宽带接入网。HFC 网除可传送 CATV 外，还提供电话、数据和其他宽带交互型业务。

现有的 CATV 网是树形拓扑结构的同轴电缆网络，它采用模拟技术的频分复用对电视节目进行单向传输。而 HFC 网则需要对 CATV 网进行改造，其主要特点如下：

（1）HFC 网的主干线路采用光纤

CATV 网所使用的同轴电缆系统具有以下的一些缺点。首先，原有同轴电缆的带宽对居民所需的宽带业务来说仍嫌不足。其次，同轴电缆每 30 m 就要产生约 1 dB 的衰减，因此每隔约 600 m 就要加入一个放大器。大量放大器的接入将使整个网络的可靠性下降，因为任何一个放大器出了故障，其下游的用户就无法接收电视节目。第三，信号的质量在远离头端处较差，因为经过了可能多达几十次的放大所带来的失真将是很明显的。最后，要将电视信号的功率很均匀地分布给所有的用户，在设计上和操作上都是很复杂的。

因此，HFC 网将原 CATV 网中的同轴电缆主干部分改换为光纤，并使用**模拟光纤技术**（图 10-31）。在模拟光纤中采用光的振幅调制 AM，这比使用数字光纤更为经济。模拟光纤从头端连接到**光纤结点**(fiber node)，它又称为**光分配结点** ODN (Optical Distribution Node)。在光纤结点光信号被转换为电信号。在光纤结点以下就是同轴电缆。一个光纤结点可连接 1～6 根同轴电缆。采用这种网络结构后，从头端到用户家庭所需的放大器数目也就只有 4～5 个。这就大大提高了网络的可靠性和电视信号的质量。

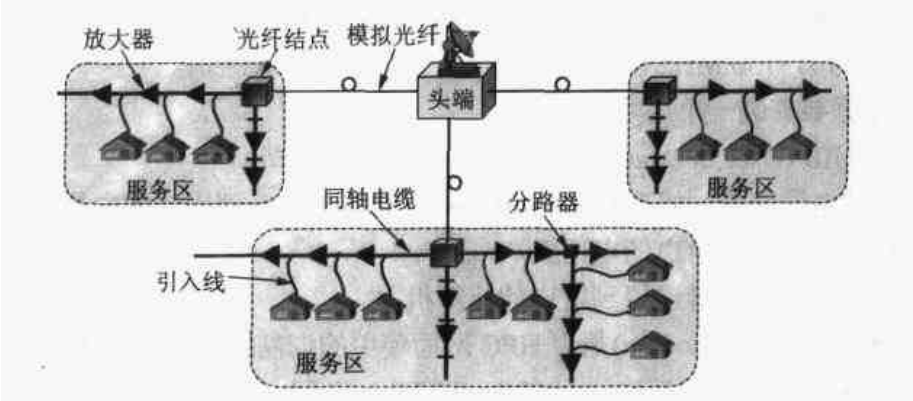


图 10-31 HFC 网的结构图

HFC 还要在头端增加一些智能，以便实现计费管理和安全管理，以及用选择性的寻址方法进行点对点的路由选择。此外，还要能适应两个方向的接入和分配协议。

（2）HFC 网采用结点体系结构

HFC 引入了**结点体系结构**(node architecture)的概念。这种体系结构的特点是：从头端到各个光纤结点用模拟光纤连接，构成星形网（图 10-5）。光纤结点以下是同轴电缆组成的树形网。连接到一个光纤结点的典型用户数是 500 左右，但不超过 2000。这样，一个光纤结点下的所有用户组成了一个**用户群**(cluster)，或称为**邻区**(neighborhood area)。光纤结点与头端的典型距离为 25 km，而从光纤结点到其用户群中的用户则不超过 2～3 km。

采用结点体系结构的好处首先是能够**提高网络的可靠性**。由于每一个用户群都独立于其他的用户群，因此某一个光纤结点或模拟光纤的故障不会影响其他的用户群。这也对整个网络可靠性的提高起了重要的作用。

结点体系结构的另一个优点是**简化了上行信道的设计**。HFC 网的上行信道是用户共享的。划分成若干个独立的用户群就可以使用价格较低的上行信道设备（因为共享上行信道的用户数减小了），同时每一个用户群可以采用同样的频谱划分而不致相互影响。这点与蜂窝无线电通信的频率重复使用是相似的。

（3）HFC 网具有比 CATV 网更宽的频谱，且具有双向传输功能

原来的 CATV 网的最高传输频率是 450 MHz，并且是用于电视信号的下行传输。HFC 网要具有双向传输功能，就必须扩展其传输频带。目前 HFC 网的频带划分还没有国际标准[①]。图 10-32 给出一种可供选择的例子。

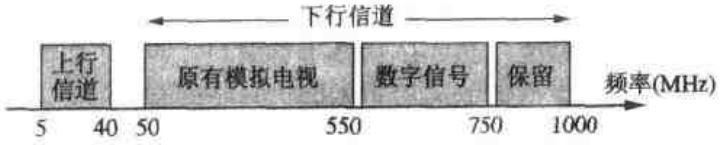


图 10-32　HFC 网频谱划分举例

从图 10-32 可看出，上行传输（从用户家庭到头端）是使用原来 CATV 并不使用的低端频段，带宽有 35 MHz 左右，这比 ADSL 的上行信道要宽得多。上行信道还可进一步划分为几个子频段，分别用于电话通信、数据通信以及对整个 HFC 网的监视。

下行传输(从头端到用户家庭)使用 50～750 MHz。其中原有模拟电视使用 50～550 MHz，但也可包含调频广播或数字广播，而各种数字信号的传输放在 550～750 MHz。这里“各种数字信号”包括数字电视信号和各种交互式业务的下行信息（如从万维网下载的多媒体信息或视频点播 VoD 信号）。750 MHz 以上保留给今后使用，如个人通信。

（4）每个家庭要安装一个用户接口盒

用户接口盒 UIB (User Interface Box)要提供三种连接，即：

- 使用同轴电缆连接到**机顶盒**(set-top box)，然后再连接到用户的电视机。
- 使用双绞线连接到用户的电话机。
- 使用电缆调制解调器连接到用户的计算机。

电缆调制解调器(cable modem)是为 HFC 网而使用的**调制解调器**。电缆调制解调器最大的特点就是传输速率高。其下行速率一般在 3～10 Mb/s 之间，最高可达 30 Mb/s，而上行速率

① 注：在阅读有关 HFC 的文献时请注意：我国采用的 PAL 制电视的频道带宽为 8 MHz。对于北美的 NTSC 制电视，频道带宽则为 6 MHz。

一般为 0.2～2 Mb/s，最高可达 10 Mb/s。然而电缆调制解调器比在普通电话线上使用的调制解调器要复杂得多，并且不是成对使用，而是只安装在用户端。

电缆调制解调器要有很好的抗干扰性能。在 HFC 网的上行频段正是无线电干扰和各种家电所产生的干扰较为集中的频段。此外，上行信号沿树形电缆向光纤结点传送时，噪声将不断地累计增大。许多厂商愿意采用正交相移键控 QPSK 作为上行信道中的调制手段，是因为 QPSK 具有良好的抗干扰性能。

电缆调制解调器的 MAC 子层协议还必须解决上行信道中可能出现的冲突问题。产生冲突的原因是因为 HFC 网的上行信道是一个用户群所共享的，而每个用户都可在任何时刻发送上行信息。这和以太网上争用信道是相似的。当所有的用户都要使用上行信道时，每个用户所能分配到的带宽就要减少。这在设计 HFC 网时应加以注意。

美国的有线电视实验室 CableLabs 制定的电缆调制解调器标准 DOCSIS (Data Over Cable Service Interface Specifications)的第一个版本 DOCSIS 1.0 已在 1998 年 3 月被 ITU-T 批准为国际标准。现在 DOCSIS 又有两个新的版本问世，即 DOCSIS 1.1 和 2.0 [W-CableLabs]。

HFC 网的最大优点是**它具有很宽的频带，并且能够利用已经有相当大的覆盖面的有线电视网**。但要将现有的 450 MHz 单向传输的有线电视网络改造为 750 MHz 双向传输的 HFC 网（还要将所有的用户服务区互连起来而不是一个个 HFC 网的孤岛），也需要相当的资金和时间。在电信政策方面也有一些需要协调解决的问题（主要是和电信网的关系）。现在使用 HFC 技术的宽带接入网工程已在许多地方启动，读者应注意这一动向。

10.6.3 FTTx 技术

除了上述的 xDSL 和 HFC 技术外，FTTx（即**光纤到……**）也是一种实现宽带居民接入网的方案。这里字母 x 可代表不同的意思（和 xDSL 中的字母 x 的作用相似）。

光纤到家 FTTH (Fiber To The Home)，即将光纤一直铺设到用户家庭可能是居民接入网最后的解决方法。但目前将光纤铺设到家还不是时机。这里有两个问题。第一，广大用户承担不了光纤到家的安装费用。这里包括铺设光缆的费用和安装在用户家中的光端机等接口设备的费用，以及应交给电信公司的月租费等。第二，现在很多用户还不需要使用这样大的带宽。目前因特网或各地区的信息网所能提供的信息并非必须使用光纤才行。因此 FTTH 只能说是今后的一个方向。

考虑中的 FTTH 将使用时分复用的方式进行双向传输，数据率为 155 Mb/s。对于上行信道需要有合适的 MAC 协议解决用户共享信道的问题。

当一幢大楼有较多用户需要使用宽带业务时，可采用**光纤到大楼** FTTB (Fiber To The Building)方案。光纤进入大楼后就转换为电信号，然后用电缆或双绞线分配到各用户。这种方案可支持大中型企业、商业或大公司高速率的宽带业务需求。它比 FTTH 要经济些。

但现在比较流行的是**光纤到路边** FTTC (Fiber To The Curb)。从路边到各个用户可使用星形结构的双绞线作为传输媒体。这可以根据具体的条件分批分阶段地实现最后的光纤到家的最后目标。FTTC 的传输速率为 155 Mb/s。FTTC 与交换局之间的接口采用 ITU-T 制订的接口标准 V5。

FTTx 还有许多其他种类，如**光纤到办公室**(Office) FTTO，**光纤到邻区**(Neighbor) FTTN，光纤到门户(Door) FTTD，光纤到楼层(Floor) FTTF，光纤到小区(Zone) FTTZ。这些就不再一一

讨论了。

10.6.4 以太网接入

由于以太网已经成功地从 10 Mb/s 的速率提高到 100 Mb/s、1 Gb/s 和 10 Gb/s，并且所覆盖的地理范围也从局域网扩展到了城域网和广域网，因此现在人们正在尝试使用以太网进行宽带接入。为此，IEEE 在 2001 年初成立了 802.3EFM 工作组[①]，专门研究以太网的宽带接入技术问题。

以太网接入的一个重要特点是它可以提供双向的宽带通信，并且可以根据用户对带宽的需求灵活地进行带宽升级（例如，将 10Mb/s 的以太网交换机更新为 100Mb/s 的，甚至 1Gb/s 的）。当城域网和广域网都采用吉比特以太网或 10 吉比特以太网时，采用以太网接入可以实现端到端的以太网传输，中间不需要再进行帧格式的转换。这就提高了数据的传输效率和降低了传输的成本。

以太网接入可以采用多种方案。图 10-33 给出的是一个例子——光纤到大楼 FTTB。每一个大楼的楼口都安装一个 100Mb/s 的以太网交换机（对于通信量不大的楼房也可使用 10 Mb/s 的以太网交换机），然后根据情况在每一个楼层安装一个 10 或 100 Mb/s 的以太网交换机。各大楼的以太网交换机通过光纤汇接到光结点汇接点。若干个光结点汇接点再通过吉比特以太网汇接到一个高速光结点汇接点（称为 GigaPoP），然后通过城域网连接到因特网的主干网。

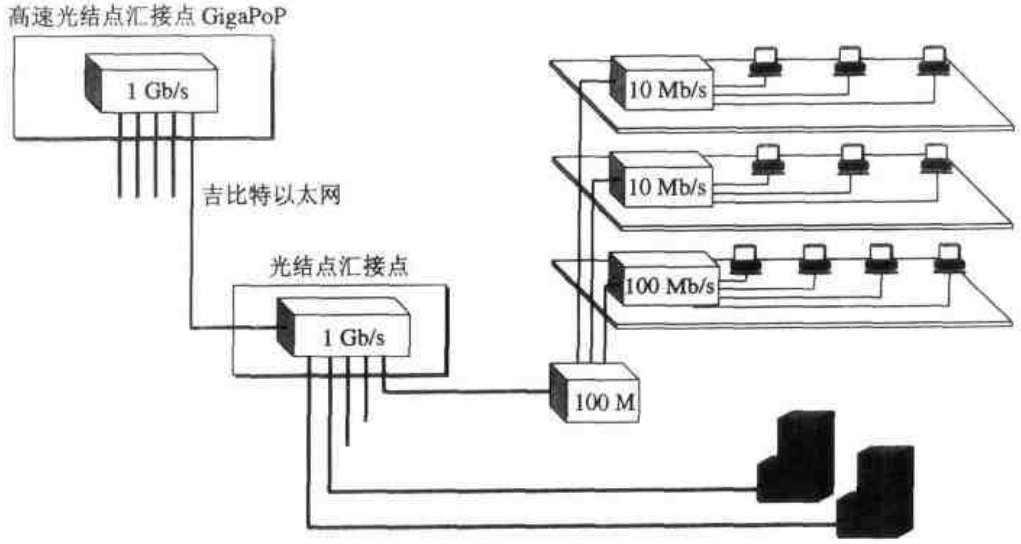


图 10-33 以太网接入举例——光纤到大楼 FTTB

当采用以太网接入时，如果大楼中使用电脑上网的用户数很少，那么这种接入方式就很难获得经济效益。对于上网用户非常密集的办公楼或居民小区，以太网接入是一个可供选择的宽带接入方法。

① 注：通信网的数字化是从主干网开始的，最后剩下的一段模拟线路是用户线，因此这一段用户线常称为是通信线路数字化过程中的“最后一英里”。802.3EFM 中的“EFM”表示“Ethernet in the First Mile”，意思是从用户端开始算，“第一英里采用以太网”，也就是说，EFM 表示“采用以太网接入”。

10.7 关于“三网融合”

我们都知道有一个著名的**穆尔定律**(Moore's law)[①]：**大规模集成电路芯片的集成度大约每18个月翻一番**。穆尔定律提出于1965年，修正于1975年。半导体集成电路的发展过程证明了穆尔定律的正确性。值得注意的是，专家们预言，或许一直到2010年穆尔定律可能仍然适用[SCHA97]。计算机网络由于还将不断得到功能更强大的计算机的支持，它所能提供的服务也将更多和更好。这种发展趋势是肯定的。

目前有关“三网合一”的问题一直受到人们的关注。这三种网（电信网络、有线电视网络和计算机网络）的规模现在都很大，但它们所使用的技术却相差很多，因此在短期内要用一种网络来代替这三种网络似不太可能。现在这三种网络都正在逐渐演变，都力图使自己也具有其他网络的优点，因此出现了“三网融合”的说法。“融合”(convergence)现在已成为热门词汇。但目前“融合”并无精确定义。通常人们是这样理解“融合”的意思：

（1）“融合”表示不同的网络平台可以提供基本上相似的服务。

（2）“融合”表示不同的消费设备（如电话、电视机、个人电脑）可以在一起工作。

图10-34中有阴影的部分表示这三种网络各自的优势所在，而箭头表示目前演变的方向。

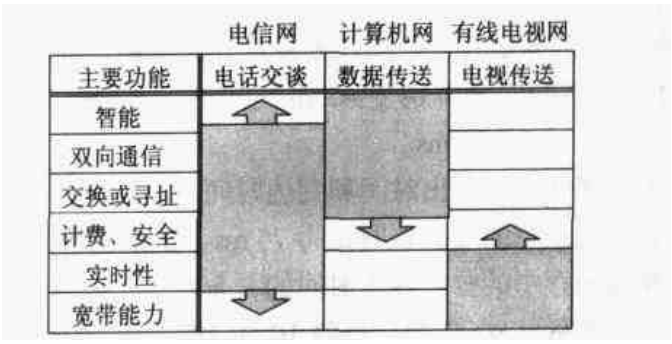


图10-34 三种网络的演变方向

三网融合远远不是技术问题，它牵涉到国家的政策、公民的文化素质和文化背景等因素，相当复杂。这里，国家的政策起到至关重要的作用。

在我国，《中共中央关于制定国民经济和社会发展第十个五年计划的建议》中明确指出：“抓紧发展和完善国家高速宽带传输网络，加快用户接入网建设，扩大利用互联网，促进电信、电视、计算机三网融合”。

这个文件表明现在国家的电信政策已顺应技术的发展潮流，适应社会主义市场经济的内在要求，明确了三网应当互相融合。因此，三网融合有着良好的前景。

然而在具体实现三网融合的过程中，如何处理好不同集团的利益分配问题，仍然是一个很复杂而困难的工作。读者应当经常关注这一领域中的动态。

在三网融合的宽带接入问题上，更重要的考虑因素是个人电脑的家庭拥有率、居民的文化水平和经济实力、以及从宽带网络上能够获取的信息。也就是说，在宽带接入方面最应当考虑的是：我国的居民到底最需要什么。我们知道，广大用户并不太关心在进行三网融合时

① 注：也有人将Moore译为“摩尔”。

哪些技术问题被解决了。实践证明，网络的速率（这里主要指接入网的速率）往往并非用户的首选因素。当上网速率超过一定数值时，用户就不太能够感觉出来。实际上，用户最关心的问题是：**我能够从宽带网得到什么**？从目前许多网络公司对宽带网络的运营情况来看，怎样使宽带网络能够向广大用户提供他们所需的内容应当是当务之急。这个问题应当引起大家的高度重视。

习题

10-01 多媒体数据和普通的文件数据都有哪些主要的区别？这些区别对多媒体数据在因特网上传送所用的协议有哪些影响？为什么说在传送多媒体数据时对时延和时延抖动都有较高的要求？

10-02 实时数据和等时的数据是一样的吗？为什么说因特网不是等时的？实时数据都有哪些特点？

10-03 因特网多媒体体系结构有哪些组成部分？有哪些新的协议？各有何特点？

10-04 为什么 RTP 协议同时具有运输层和应用层的特点？

10-05 RTP 协议能否提供应用分组的可靠传输？请说明理由。

10-06 在 RTP 分组的首部中为什么要使用序号、时间戳和标记？

10-07 携带实时音频信号的固定长度分组序列发送到因特网。每隔 10 ms 发送一个分组。前 10 个分组通过网络的时延分别是 45 ms，50 ms，53 ms，46 ms，30 ms，40 ms，46 ms，49 ms，55 ms 和 51 ms。

（1）用图表示出这些分组发出时间和到达时间。

（2）若在接收端还原时的端到端时延为 75 ms，试求出每一个分组经受的时延。

（3）画出接收端缓存中的分组数与时间的关系。

10-08 话音信号的采样速率为 8000 Hz。每隔 10 ms 将已编码的话音采样装配成话音分组。每一个话音分组在发送之前要加上一个时间戳。假定时间戳是从一个时钟得到的，该时钟每隔 Δ 秒将计数器加 1。试问能否将 Δ 取为 9 ms？如果行，请说明理由。如果不行，你认为 Δ 应取为多少？

10-09 在传送多媒体实时数据时，接收端的缓存空间的上限由什么因素决定？实时数据流的数据率和时延抖动对缓存空间上限的确定有何影响？

10-10 RTCP 协议和 RTSP 协议使用在什么场合？它们各有何主要特点？

10-11 IP 电话的两个主要标准各有何特点？影响 IP 电话话音质量的主要因素有哪些？

10-12 什么是服务质量 QoS？为什么说“因特网根本没有服务质量可言”？

10-13 在讨论服务质量时，管制、调度、呼叫接纳各表示什么意思？

10-14 试比较先进先出(FIFO)排队、公平排队(FQ)和加权公平排队(WFQ)的优缺点。

10-15 漏桶管制器的工作原理是怎样的？数据流的平均速率、峰值速率和突发长度各表示什么意思？

10-16 采用漏桶机制可以控制达到某一数值的、进入网络的数据率的持续时间。设漏桶最多可容纳 b 个权标。当漏桶中的权标数小于 b 个时，新的权标就以每秒 r 个权标的恒定速率加入到漏桶中。设分组进入网络的速率为 N pkt/s (pkt 代表分组)，试推导以此速率进入网络所能持续的时间 T。讨论一下为什么改变权标加入到漏桶中的速率就可以

控制分组进入网络的速率。

10-17 在上题中，设 $b = 250$ token，$r = 5000$ token/s，$N = 25000$ pkt/s。试求分组用这样的速率进入网络能够持续多长时间。若 $N = 2500$ pkt/s，重新计算本题。

10-18 试推导公式(10-2)。

10-19 综合服务 IntServ 由哪几个部分组成？有保证的服务和受控负载的服务有何区别？

10-20 试述资源预留协议 RSVP 的工作原理。

10-21 区分服务 DiffServ 与综合服务 IntServ 有何区别？区分服务的工作原理是怎样的？

10-22 在区分服务 DiffServ 中的每跳行为 PHB 是什么意思？EF PHB 和 AF PHB 有何区别？它们各适用于什么样的通信量？

10-23 假定一个发送端向 2^n 个接收端发送多播数据流，而数据流的路径是一个完全的二叉树，在此二叉树的每一个结点上都有一个路由器。若使用 RSVP 协议进行资源预留，问总共要产生多少个资源预留报文 RESV（有的在接收端产生，也有的在网络中的路由器产生）？

10-24 多协议标记交换 MPLS 的工作原理是怎样的？它有哪些主要的功能？

10-25 标记交换路由器 LSR 在转发 MPLS 分组时要进行标记对换。类似这样的标记对换是否也曾发生在 MPLS 以外的某种网络中？如有，请说明一下。

10-26 关于 MPLS 使用显式路由选择的几个问题：

（1）MPLS 能否使用显式路由选择以保证对特定流的 QoS 需求（如带宽或时延）？请说明理由。

（2）试给出两个例子分别在细粒度和粗粒度上使用 QoS 显式路由选择。

10-27 试讨论在 MPLS 域中的三种流的聚合程度：

（1）所有的分组都是流向同一个主机；

（2）所有的分组都流经同一个出口 LSR；

（3）所有的分组都具有同样的 CIDR 地址。

10-28 试比较网络在以下三种情况的可扩缩性：

（1）仅使用第三层转发：每一个路由器查找最长前缀匹配以确定下一跳；

（2）第三层转发和第二层 MPLS 转发；

（3）仅有第二层 MPLS 转发。

10-29 在 DiffServ 中的边界结点和 MPLS 中的入口结点是否都是同样性质的结点？DiffServ 中的边界路由器和 MPLS 入口结点的标记交换路由器一样吗？

10-30 什么是 RAN 和 RBB？试比较 xDSL、FTTx、HFC 和以太网接入技术的优缺点。

10-31 在图 10-29 中，为什么在 140～1100 kHz 不到 1 MHz 的带宽能够传送 6.3 Mb/s 的数据？

10-32 什么是穆尔定律？它对当前提出的“网络时代”有何影响？

10-33 什么是“三网融合”？这里都有哪些需要解决的问题？

附录 A　最基本的排队系统

掌握必要的排队论基础知识，对学习和研究计算机网络是非常重要的。限于篇幅，本附录只介绍最基本的排队系统，并推导出几个最常用的公式。进一步的学习可参阅教科书[HAMM86][KLEI75]。

A.1　稳定状态下的数据流

A.1.1　李特尔(Little)定律

设有一网络，其边界为一封闭曲线（如图 A-1 所示），设数据为长短不一的报文，报文随机地进入网络，再按其排队的先后顺序，发往其他地方。下面我们研究，在稳定状态下，这种网络中暂时存储的报文数目与哪些因素有关，它们的关系如何。

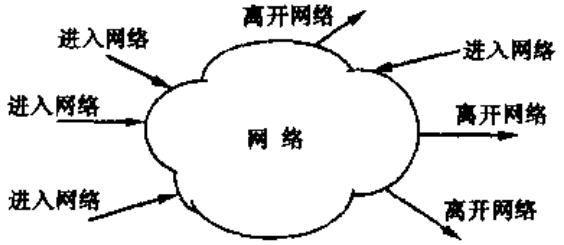


图 A-1　具有封闭边界的网络

设在时间间隔(0，t)内进入网络的报文数目为$\alpha(t)$，离开网络的报文数目为$\delta(t)$，它们的差值为 $N(t)$，即在此时间间隔里存储在网络中的报文数目：

$$N(t)=\alpha(t)-\delta(t) \tag{A-1}$$

典型的报文进入网络和离开网络的表示曲线如图 A-2 所示。这里“离开网络”是指报文发送完毕。

在时间间隔长度为 t 时，报文的平均到达率记为 λ_t，其表达式为

$$\lambda_t=\alpha(t)/t \tag{A-2}$$

还有一个参数需要计算，这就是所有的报文在网络中已经经历的时间 $\gamma(t)$，它就是在曲线 $\alpha(t)$ 和 $\delta(t)$之间的面积，即：

$$\gamma(t)=\int_0^t N(x)dx \tag{A-3}$$

显然，在时间间隔(0，t)内网络中的平均报文数目 N_t 应为：

$$N_t=\int_0^t N(x)dx/t=\gamma(t)/t \tag{A-4}$$

再算一下每一个报文在网络中所经历的平均时间 T_t。它应为：

$$T_t=\gamma(t)/\alpha(t) \tag{A-5}$$

由公式(A-2)，(A-4)和(A-5)可得出：

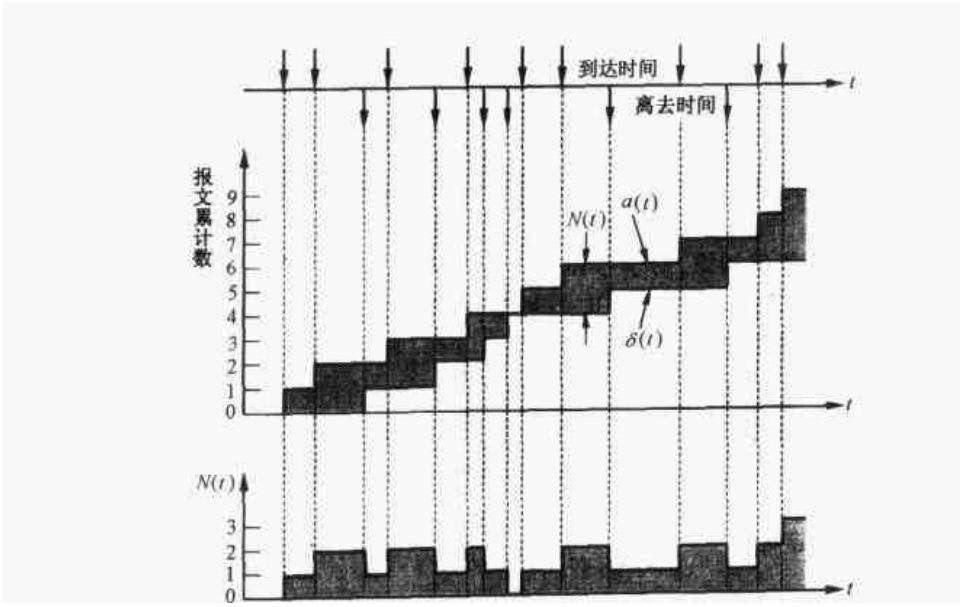


图 A-2　在稳定状态下累计进入网络和离开网络以及在网络中存储的报文数目

$$N_t = \lambda_t T_t \tag{A-6}$$

在以上推导中有一点不够精确。这就是在时间间隔(0, t)进入网络的报文一般总可能还有未能离开网络的。但随着时间 t 的增大，这类报文的影响就可以忽略不计。

设当 $t\to\infty$ 时，N_t，λ_t 和 T_t 都是有限值，则可令

$$\lambda = \lim_{t\to\infty}\lambda_t, \qquad T = \lim_{t\to\infty}T_t, \qquad N = \lim_{t\to\infty}N_t \tag{A-7}$$

这样就可将(A-6)式变为：

$$N = \lambda T \tag{A-8}$$

这就是著名的李特尔定律。李特尔定律告诉我们，在稳定状态下，**存储在网络中的报文平均数 N，等于报文的平均到达率 λ 乘以这些报文在网络中经历的平均时间 T**。

在使用李特尔定律时应注意到，网络的封闭边界可以由我们任意设定，只要 N，λ 和 T 都是属于同一个网络的即可。此外，报文按什么规律输入以及报文长度按什么规律分布，都不影响李特尔定律的成立。这点也是很重要的。

A.1.2　通信量强度

这一节将研究一个网络的**通信量强度**(traffic intensity)。

我们研究图 A-3 所示的网络，它有好几个报文输入端，但只有一个输出信道。对于更复杂的网络，只要单考虑发往某个信道的报文，就可得出图 A-3 所示的模型。我们还假设报文在队列中等待发送是按照先到先发送的原则进行的。下面研究在稳定状态下的一些重要关系式。

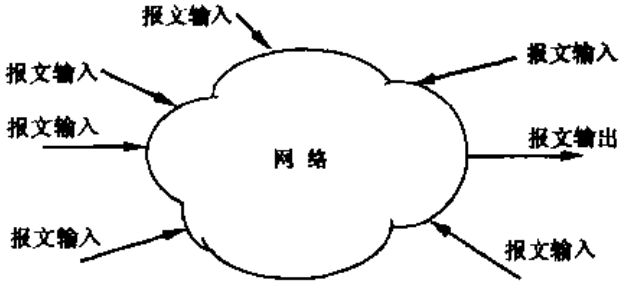


图 A-3　具有单个输出信道的网络

图 A-4(a)表示报文的到达和离去情况。我们要研究从第 i 个报文 M_i 发完到第 $i+1$ 个报文 M_{i+1} 发完这段时间内，网络内的报文数目 N 的变化情况。设 N_i 和 N_{i+1} 分别为 M_i 和 M_{i+1} 刚刚发送完毕时网络内的报文数目。我们应注意到，若 $N_i>0$，则 M_i 刚一发送完毕，就马上开始发送 M_{i+1}。对于图 A-4(a)的例子，在 M_{i+1} 的发送时间内共到达 3 个报文。但是，若像图 A-4(b)那样 $N_i=0$，则 M_i 刚一发送完毕，网络就处于空闲状态。所以在 M_{i+1} 的发送时间内共到达 2 个报文（第一个到达的报文现在就是 M_{i+1}）。若用 A_{i+1} 表示在 M_{i+1} 的发送时间内到达网络的报文数目，则从图 A-4 可得：

$$N_{i+1}=N_i+A_{i+1}-U(N_i) \tag{A-9}$$

其中 $U(N_i)$为单位阶跃函数，即：

$$U(N_i)=\begin{cases}1, & \text{当 } N_i>0\\ 0, & \text{当 } N_i=0\end{cases} \tag{A-10}$$

将(A-9)式两边取平均值，并考虑到在稳定状态下，$\overline{N}_{i+1}=\overline{N}_i$，则可得出：

$$\overline{U}(N_i)=\overline{A}_{i+1} \tag{A-11}$$

即 $U(N_i)$的平均值等于第 $i+1$ 个报文 M_{i+1} 的发送时间内网络的平均报文到达数 $\overline{A}_{i+1}$。

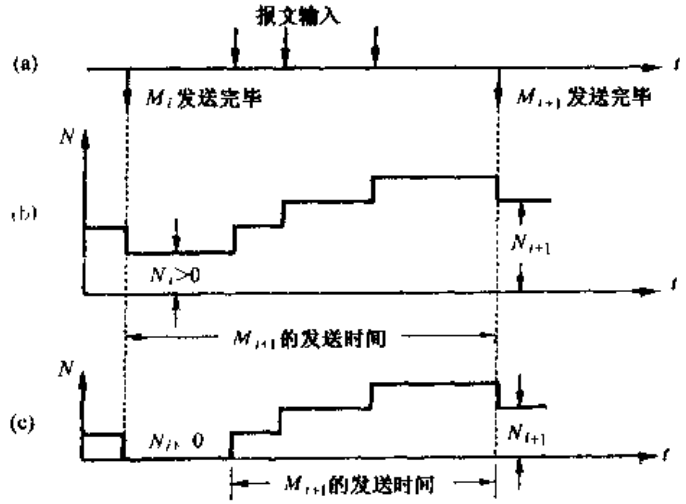


图 A-4　稳定状态下报文的发送时间

(a)报文的到达和发送；　(b)$N_i>0$；　(c)$N_i=0$

我们还应注意到，只要 $N_i>0$，就表明输出信道正在输出报文，信道处于忙的状态。而当 $N_i=0$ 时，网络内没有报文了，信道处于空闲状态。因此 $U(N_i)$可用来指示输出信道是忙还是闲，而 $U(N_i)$则是输出信道平均活跃性(Activity)的一个度量。所谓“活跃性”就是信道忙的程度。输出信道忙就表示到达的报文须排队等待一段时间才能发送出去。

如果不限于在报文刚刚发送完毕的时刻来研究问题，而将在任何时刻 t 网络内的报文数记为 $N(t)$，则 $\overline{U}(N(t))$ 就表示信道平均忙的程度。这个参数特别重要，通常称之为通信量强度或业务量强度，并记为 ρ，即

$$\overline{U}(N(t))=\rho \tag{A-12}$$

显然，

$$0\leqslant\rho\leqslant1 \tag{A-13}$$

ρ 是分析网络性能时的重要参数。

下面研究图 A-5 所示的简单例子。

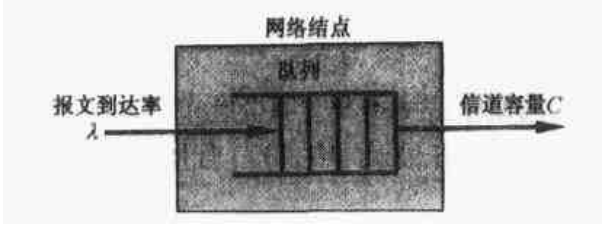


图 A-5　最简单的网络结点及输出信道

设报文(记为 msg)到达网络结点的平均到达率为 λ (msg/s)，报文的到达可认为是瞬时的，输出信道的容量为 C (b/s)。报文长度是随机的，其平均长度为 $1/\mu$ (bit)（采用 $1/\mu$ 是排队论中的习惯表示方法）。这样，每个报文的平均发送时间为 $1/\mu C$ (s)。换言之，信道的发送能力为 μC (msg/s)。但必须注意，当网络结点中的队列没有报文时，则信道上的输出应当为零，而不是 μC (msg/s)。在稳定状态下报文的平均输出速率一定等于平均输入速率。因此，若用 P[A] 表示事件 A 出现的概率，则

$$\lambda = P\ [\text{输出信道忙}] \cdot \mu C + P\ [\text{输出信道空闲}] \cdot 0$$

因此得出

$$\lambda = \rho \mu C \tag{A-14}$$

或

$$\rho = \lambda / \mu C \tag{A-15}$$

即通信量强度是报文平均到达率与输出信道所能提供的报文平均输出率之比。由(A-13)式可得出在稳定状态下，

$$\lambda \leqslant \mu C \tag{A-16}$$

其意义已十分清楚。后面将证明，在一个实际的网络中，报文的平均到达率 λ 不但不允许等于 μC，而且也不允许太接近于 μC 的值。

A.2　几种排队模型

本节将定量分析几种典型的排队模型。在排队论的文献中，常用“字母/字母/数字”这样的表示方法来描述某一类型的排队系统。第一个字母代表到达的规则，第二个字母代表服务规则，数字代表模型中平行的队列（即服务通道或发送信道）数目。我们最常遇到的几个字母是 M（负指数概率密度），G（一般概率密度）和 D（确定值）。

顺便指出：在排队理论中所用的术语和网络中的术语不同，但它们是互相对应的。如：顾客←→报文；服务员←→信道；服务时间←→报文发送时间。有时为方便起见我们对这两术语可以混用。

A.2.1　M/G/1 模型

M/G/1 表示到达规律是负指数概率密度，服务规则可以是任意的，而输出信道只有一个。M 也代表泊松(Poisson)过程，我们就先从泊松过程讲起。

1．泊松过程

如果在时间间隔 T 内到达 k 个报文的概率为

$$\text{P}[T\text{ 秒内 }k\text{ 个报文到达}]=\frac{(\lambda T)^k e^{-\lambda T}}{k!},\quad k=0,1,2,\ldots \tag{A-17}$$

其中 λ 为报文的平均到达率，则称这种到达为泊松过程。

现在推导到达时间间隔的概率密度函数 $\alpha(t)$。

$\alpha(t)\Delta t$ 表示 P[t 内无到达但在$(t, t+\Delta t)$内有 1 个到达](参见图 A-6)。

根据(A-17)式，

P[t 内无到达]$=e^{-\lambda t}$

P[$(t, t+\Delta t)$内有 1 个到达]$=(\lambda\Delta t)e^{-\lambda\Delta t}$

因此，

$$\alpha(t)\Delta t=e^{-\lambda t}\cdot\lambda\Delta t e^{-\lambda\Delta t}$$

当$\Delta t\to 0$ 时，得出

$$\alpha(t)=\lambda e^{-\lambda t} \tag{A-18}$$

这就是**负指数的概率密度函数**。从图 A-7 不难看出，负指数的规律表明短的到达时间间隔比长的间隔出现的更为频繁。

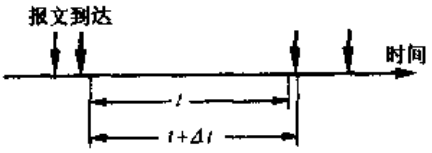


图 A-6　时间间隔Δt 内有 1 个到达

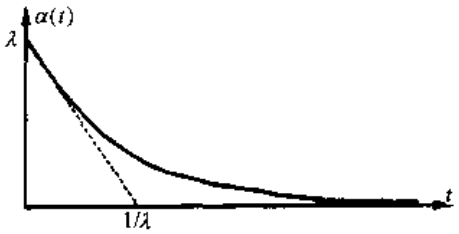


图 A-7　到达时间间隔的概率密度函数$\alpha(t)$

可以推导出平均到达时间间隔 a_1 正好是平均到达率 λ 的倒数：

$$a_1=\int_0^\infty t\alpha(t)dt=\frac{1}{\lambda} \tag{A-19}$$

到达时间间隔的方差为：

$$\sigma_a^2=\int_0^\infty t^2\alpha(t)dt-a_1^2=\frac{2}{\lambda^2}-\frac{1}{\lambda^2}=\frac{1}{\lambda^2} \tag{A-20}$$

我们看出，$\sigma_a=a_1=1/\lambda$。这是泊松过程的一个重要特点。

如果服务时间的概率密度也是负指数函数(记为 $b(t)$)，且平均服务时间为 $1/\mu$（注意：这样可得出与(A-18)式相似的公式），则

$$b(t)=\mu e^{-\mu t} \tag{A-21}$$

μ 也称为平均服务率。

用类似方法可导出服务时间的标准差 σ_b 与平均值 b_1 都等于 $1/\mu$，即

$$\sigma_b=b_1=1/\mu \tag{A-22}$$

这里还要指出，在排队论中，平均服务时间用 $1/\mu$。但在计算机网络中，$1/\mu$ 为平均报文长度，而平均服务时间为 $1/\mu C$，这点不要弄混。

关于泊松过程的其他重要特性，如无记忆特性以及和马尔可夫过程的关系等，可参阅教科书[KLEI75]。

2．扑拉切克-辛钦(Pollaczek-Khinchine)公式

现在我们来推导 M/G/1 排队模型中的最重要的扑拉切克-辛钦公式，简称 **P-K 公式**。

我们还要利用图 A-4，但现在假定报文输入是泊松过程，报文的平均到达率和平均长度分别为 λ 和 $1/\mu$，但报文长度的分布规律则是任意的。输出信道只有一个，其容量为 C。

我们从(A-9)式的差分方程着手。对于泊松过程，在任何一个报文发送时间内的到达数目，与其他报文发送时间的长短无关，也与网络内的报文数目无关。因此，(A-9)式中 A_{i+1} 的下标 $(i+1)$可以去掉。但要记住，A 与报文的发送时间（我们将它记为 Y，是个随机变量）有关。

将(A-9)式两端平方，再求平均值，得：

$$\overline{N_{i+1}^2}=\overline{N_i^2}+\overline{U^2(N_i)}+\overline{A^2}-\overline{2N_iU(N_i)}-\overline{2AU(N_i)}+\overline{2N_iA} \tag{A-23}$$

在稳态下，上式的前两项可消去，因为网络中的报文数的均方值与在哪个报文发完时进行统计无关。

根据(A-10)式和(A-12)式，可得出：

$$\overline{U^2(N_i)}=\overline{U(N_i)}=\rho \tag{A-24}$$

因为 $N_iU(N_i)=N_i$，所以

$$\overline{N_iU(N_i)}=\overline{N_i} \tag{A-25}$$

考虑到 A 与 N_i 互相独立，所以利用(A-11)式，得：

$$\overline{AU(N_i)}=\overline{A}\,\overline{U(N_i)}=\rho^2 \tag{A-26}$$

$$\overline{N_iA}=\overline{N_i}\,\overline{A}=\overline{N_i}\rho \tag{A-27}$$

将(A-24)至(A-27)式代入(A-23)式，得出：

$$0=\rho+\overline{A^2}-2\overline{N_i}-2\rho^2+2\overline{N_i}\rho \tag{A-28}$$

对于泊松过程，在稳定状态下，网络中的平均报文数目不是时间的函数，因此 $\overline{N_i}$ 可改写为 N。于是得出网络中平均报文数目为

$$N=\frac{\rho+\overline{A^2}-2\rho^2}{2(1-\rho)} \tag{A-29}$$

剩下的问题是计算 $\overline{A^2}$。A 是在一个报文的发送时间 Y 内报文的到达数目。所以计算 $\overline{A^2}$ 时先认为 Y 是固定的，然后再对 Y 求平均。设 Y 固定时 $\overline{A^2}$ 为 m，则

$$m=\sum_{A=1}^{\infty}A^2\frac{(\lambda Y)^A e^{-\lambda Y}}{A!}=\lambda Y+\lambda^2Y^2 \tag{A-30}$$

再设报文的发送时间的概率密度为 $b(Y)$（注意：现在不是负指数函数了），则

$$\begin{aligned}\overline{A^2}&=\int_0^{\infty}mb(Y)dY=\int_0^{\infty}(\lambda Y+\lambda^2Y^2)b(Y)dY\\&=\lambda\int_0^{\infty}Yb(Y)dY+\lambda^2\int_0^{\infty}Y^2b(Y)dY\end{aligned} \tag{A-31}$$

上式第一个积分即 Y 的平均值 $\overline{Y}$，它是 $1/\mu C$。第二个积分等于 Y 的方差 σ_Y^2 与 $\left(\overline{Y}\right)^2$ 之和。因此，

$$\overline{A^2}=\lambda/\mu C+\lambda^2(\sigma_Y^2+1/\mu^2C^2) \tag{A-32}$$

利用(A-15)式($\rho=\lambda/\mu C$)，并考虑到 σ_Y^2 实际上就是服务时间的方差 σ_b^2，可将(A-32)式改

写为：

$$\overline{A^2} = \rho + \rho^2 + \lambda^2\sigma_b^2 \tag{A-33}$$

将(A-33)式代入(A-29)式，即得出网络结点的队列中的平均报文数目：

$$N = \rho + \frac{\rho^2 + \lambda^2\sigma_b^2}{2(1-\rho)} \tag{A-34}$$

由(A-8)式的李特尔定律，可求出报文经网络所产生的平均时延 $T=N/\lambda$，即

$$T = \frac{1}{\mu C} + \frac{\rho + \lambda\mu C\sigma_b^2}{2\mu C(1-\rho)} \tag{A-35}$$

以上得出的 M/G/1 模型中的两个公式，即著名的 P-K 公式。

P-K 公式告诉我们，对于 M/G/1 排队系统，网络中存储的报文数的平均值以及报文时延的平均值，取决于报文到达率的平均值，以及报文发送时间的平均值和方差。至于报文发送时间的分布如何，是不需要知道的。

我们还可看出，当通信量强度 ρ 趋近于 1 时，网络中积压的报文的平均数目 N 和报文经网络引起的平均时延 T 都趋于无限大。由此可见，要使网络正常工作，通信量强度 ρ 一定不能太接近于 1。

我们还应注意到，当 ρ 不变时，N 和 T 都随 σ_b^2 线性增长。当 $\sigma_b^2 = 0$ 时，N 和 T 均达到最小值。$\sigma_b^2 = 0$ 相当于报文发送时间为常数，即报文都是等长的情况。

A.2.2 M/M/1 模型

M/M/1 模型实际上是 M/G/1 模型的一个特例，即报文发送时间也是泊松过程，或者说，报文处理时间具有负指数的概率密度函数。

在这种情况下，报文发送时间的概率密度函数是

$$b(t) = \mu C e^{-\mu C t} \tag{A-36}$$

报文发送时间的平均值和方差分别为：

$$b_1 = 1/\mu C \tag{A-37}$$

$$\sigma_b^2 = (1/\mu C)^2 \tag{A-38}$$

这样，(A-34)式变为：

$$N = \frac{\rho}{1-\rho} \tag{A-39}$$

(A-35)式变为：

$$T = \frac{1}{\mu C(1-\rho)} = \frac{1}{\mu C - \lambda} \tag{A-40}$$

由于 $1/\mu C$ 是一个报文的平均发送时间，因此，$T\mu C$ 表示报文的平均时延是发送时间的多少倍。将(A-40)式变为归一化的平均时延：

$$\mu CT = \frac{1}{1-\rho} \tag{A-41}$$

这样，N 和 μCT 仅仅是通信量强度的函数。当通信量强度 ρ 趋近 1 时，N 和 T 都趋于无限大，这点和 M/G/1 是相似的。

A.2.3 M/D/1 模型

在分组交换网中，若每个分组的长度是固定的，那么每个分组的发送时间是相等的。这就要用到 M/D/1 模型。

对于 M/D/1 模型，报文发送时间的平均值和方差分别为

$$b_1 = 1/\mu C \tag{A-42}$$

$$\sigma_b^2 = 0 \tag{A-43}$$

代入(A-34)和(A-35)式，得出

$$N = \rho + \frac{\rho^2}{2(1-\rho)} = \frac{\rho(2-\rho)}{2(1-\rho)} \tag{A-44}$$

$$T = \frac{1}{\mu C} + \frac{\rho}{2\mu C(1-\rho)} = \frac{2-\rho}{2\mu C(1-\rho)} \tag{A-45}$$

为了和 M/M/1 模型进行比较，我们将 M/D/1 和 M/M/1 两种排队模型所得出的结果画在图 A-8 中，横坐标都是通信量强度 ρ，而纵坐标则分别为 μCT 和 N。

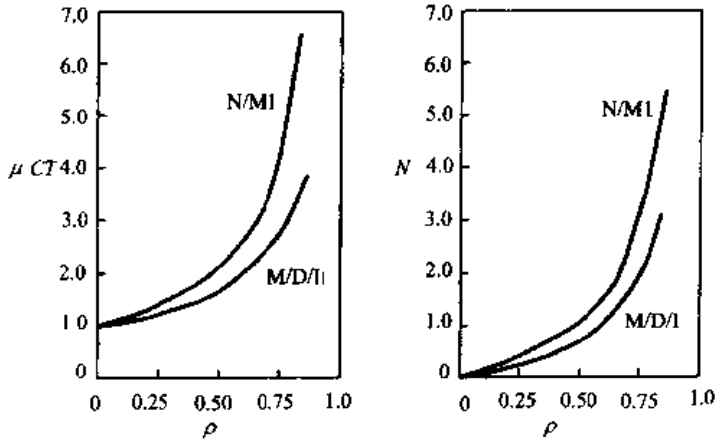


图 A-8 M/M/1 和 M/D/1 的性能对比

从图 A-8 可清楚地看出，M/M/1 的性能不如 M/D/1，因为网络中积压下来发送不出去的报文数目在 M/M/1 的情况下要多一些（当 ρ 接近 1 时要比 M/D/1 的加倍），这就使后面到达的报文具有较大的时延。不论哪一种情况，当 ρ 接近于 1 时，N 和 T 都趋向无限大，这正是所有排队系统共同特点。

我们还注意到，即使当 $\rho \to 0$ 时（即网络的通信量非常小），网络的平均时延亦不为零，而是等于 $\mu CT \to 1$，即时延 $T \to 1/\mu C$。这正是由于信道容量有限而使得发送一个报文需要一定的时间。由此可见，网络引起的平均时延由两部分组成，第一部分是 $1/\mu C$，即**报文的发送时间**，这是任何一种排队系统中都具有的一项；第二部分则是**报文的排队时间**，这是当报文到达时，在它前面的一个报文尚未发完而不得不参加排队所引起的平均时延。正是这一部分时延代表了一个排队系统的时延特性。很容易证明，M/M/1 的平均排队时间正好是 M/D/1 的 2 倍。

从(A-35)式还可看出，在 M/G/1 模型中，无论报文长度的分布服从什么规律，σ_b^2 总是非负的。因此在 M/G/1 模型中，**其特例 M/D/1 的性能是最佳的**。这就是为什么在讨论网络的性能分析时，总是喜欢用 M/D/1 模型作为一个比较的样板。

关于最基本的排队系统就介绍到这里。

附录 B　随机接入技术：ALOHA

在 20 世纪 70 年代初期夏威夷大学首次试验成功随机接入。这是为了使地理上分散的用户通过无线电来使用中心计算机。由于无线电信道是一个公用信道，一个站发送的信息可以同时被多个站收到，而每个站又是随机发送的，因此这种系统是一个随机接入系统。夏威夷大学早期研制的系统称为 ALOHA，是 Additive Link On-line HAwaii system 的缩写，而 ALOHA 恰好又是夏威夷方言的“你好”。下面先介绍纯 ALOHA。

B.1　纯 ALOHA

1．工作原理

纯 ALOHA 就是最原始的 ALOHA。它可以工作在无线信道，也可以工作在总线式网络中。为讨论其工作原理，我们采用如图 B-1 所示的模型。这个模型不仅可代表总线式网络，而且可以代表无线信道的情况。



图 B-1　ALOHA 系统的一般模型

图 B-2 表示一个 ALOHA 系统的工作原理。每一个站均自由地发送数据帧。为分析简单起见，今后帧的长度不是用比特而是用发送这个帧所需的时间来表示，在图 B-2 中用 T_0 代表这段时间。我们还设所有的站发送的帧都是定长的。

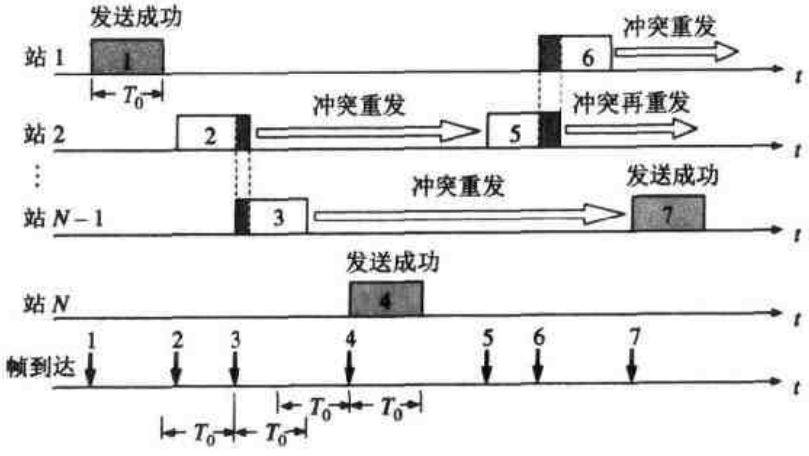


图 B-2　纯 ALOHA 系统的工作原理

当站 1 发送帧 1 时，其他的站都未发送数据，所以站 1 的发送必定成功。这里不考虑由信道不良而产生的误码。但随后站 2 和站 $N-1$ 发送的帧 2 和帧 3 在时间上重叠了一些。这就是以前提到过的“碰撞”。碰撞的结果是使碰撞的双方（有时也可能是多方）所发送的数据都出现差错，因而都必须进行重传。但是发生碰撞的各站不能马上进行重传，因为这样做就必然会继续产生碰撞。ALOHA 系统采用的重传策略是让各站等待一段随机的时间，然后再进行重传。如再发生碰撞，则需再等待一段随机的时间，直到重传成功为止。图中其余的一些帧的发送情况是帧 4 发送成功，而帧 5 和帧 6 发生碰撞。

2. 性能分析

下面我们来分析纯 ALOHA 的一些主要性能，这就是吞吐量和平均时延的计算。

为便于分析，我们在图 B-2 中用最下面的一个坐标将所有各站的发送情况都画在一起，用一个垂直向下的箭头表示某个帧的开始发送（可以和上面各站的发送情况对照来看）。从图中可看出，一个帧如欲发送成功，必须在该帧发送时刻之前和之后各一段时间 T_0 内（一共有 $2T_0$ 的时间间隔），没有其他帧的发送。否则就必产生碰撞而导致发送失败。例如，帧 3 发送时刻之前 T_0 的时间内，出现帧 2 的发送，因此帧 3 和帧 2 的发送都要失败。而帧 4 的发送时刻之前和之后的时间 T_0 内，没有其他帧的发送，因此帧 4 的发送必定成功。

我们可以把每发送一个帧看成是有一个帧到达 ALOHA 网络。这样，一个帧发送成功的条件，就是该帧与该帧前后的两个帧的到达时间间隔均大于 T_0。

我们设帧的到达服从泊松分布。但这并不完全符合实际情况。这是因为，虽然大量的站同时随机地发送数据帧时，在每个站的通信量都很小的条件下，整个系统的帧到达可看成是泊松过程，但在出现重传过程时，这样的到达过程就不再是泊松过程，而是一个与重传策略有关的较为复杂的过程。然而如果重传时的随机时延足够长，那么认为帧的到达（包括重传帧）是泊松过程仍是合理的。在这样的假定下，就可以使 ALOHA 系统的分析大为简化。

在有关 ALOHA 系统的文献中，一般都使用这样两个归一化的参数。它们是：

（1）**吞吐量** S　这又称为**吞吐率**，它等于在帧的发送时间 T_0 内成功发送的平均帧数。显然，$0 \le S \le 1$，而 $S = 1$ 是极限情况。在 $S = 1$ 时，帧一个接一个地发送出去，帧与帧之间没有空隙。这种情况虽然使信道的利用最为充分，但在众多用户随机发送帧的情况下是不可能实现的。但是，可以用 S 接近于 1 的程度来衡量信道的利用率是否充分。

当网络系统达到稳定状态时，在时间 T_0 内到达网络的平均帧数（即输入负载）应等于吞吐量 S。

（2）**网络负载**(offered load) G　从网络的角度看，G 等于在 T_0 内总共发送的平均帧数。这里包括发送成功的帧和因碰撞未发送成功而重传的帧。显然，$G \ge S$，而只有在不发生碰撞时，G 才等于 S。还应注意到，G 可以远大于 1。例如，$G = 10$，表示在 T_0 时间内网络共发送了 10 个帧，这当然会导致很多的碰撞。

在稳定状态下，吞吐量 S 与网络负载 G 的关系为：

$$S = G \cdot \text{P[发送成功]} \qquad \text{(B-1)}$$

这里 P[发送成功]是一个帧发送成功的概率，它实际上就是发送成功的帧在所发送的帧的总数中所占的比例。从图 B-2 可看出，若帧 4 要发送成功，帧 3 和帧 4 的时间间隔应大于 T_0，同时帧 4 和帧 5 的时间间隔也要大于 T_0。因此，若帧 4 要发送成功，必须在帧 4 到达的前后各一个 T_0 的时间内没有其他帧的到达。因为假定了帧的到达是泊松过程，因此在 $2T_0$ 的时间内

有 k 个到达的概率是：

$$\text{P[在 } 2T_0 \text{ 的时间内有 } k \text{ 个到达]} = \frac{(2G)^k}{k!}e^{-2G}, k = 0, 1, 2... \qquad \text{(B-2)}$$

在上式中，$2G$ 是在 $2T_0$ 的时间内的平均到达帧数。于是

$$\begin{aligned} S &= G \cdot \text{P[发送成功]} = G \cdot \text{P[在 } 2T_0 \text{ 的时间内有 0 个到达]} \\ &= G\frac{(2G)^0}{0!}e^{-2G} \\ &= Ge^{-2G} \end{aligned} \qquad \text{(B-3)}$$

这就是 Abramson 于 1970 年首次推导出的 ALOHA 吞吐量公式。

当 $G = 0.5$ 时，$S = 0.5e^{-1} \approx 0.184$。这是吞吐量 S 可能达到的极大值。这点从图 B-3 的吞吐量曲线可以看得很清楚。用求极值的方法也可很容易地得出这一结论。

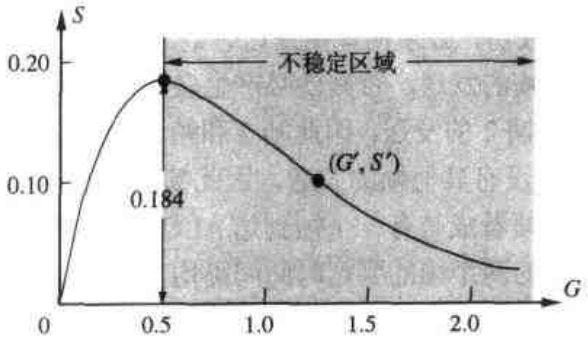


图 B-3　纯 ALOHA 的吞吐量与网络负载的关系曲线

(B-3)式是在假设系统工作在稳定状态下推导出来的。然而图 B-3 所示的曲线在 G 值大于 0.5 呈现负的斜率，因而这段区域是不稳定的。关于这点可做如下解释。设系统工作在 $G > 0.5$ 的某一个点上(G', S')。假定现在由于某种原因使网络负载 G 增大了一些。根据图中的曲线，吞吐量应下降。这表明成功发送的帧数减少而发生碰撞的帧数则增加。这种情况就引起更多的重传，因而使网络负载 G 进一步增大。这样恶性循环的结果，使工作点迅速沿曲线下降，直到吞吐量下降到零为止。这时，网络负载达到很大的数值。数据帧不断地发送、碰撞、重传……，但是并无有用的输出。整个系统完全不能工作了。可见，在纯 ALOHA 系统中，网络负载 G 一定不能超过 0.5。

一个理想随机接入系统的吞吐量 S 的极限值是 1。但纯 ALOHA 的吞吐量的极大值只能达到理想值的 18.4 %。实际上为安全起见，纯 ALOHA 的吞吐量 S 不应超过 10%。为了提高 ALOHA 系统的吞吐量，在纯 ALOHA 出现之后又有了多种改进的 ALOHA 系统。

虽然如此，在许多情况下，当需要进行突发式的交互性的数据通信时，采用纯 ALOHA 这样的方式可能既简单又便宜。当年夏威夷大学进行的实验也正是为这种环境而设计的。现在假定许多异步终端通过多点线路连到主机，线路的数据率为 4800 b/s。设每份报文有 60 个字符，而用户用键盘输入一份报文需 2 分钟（包括思考时间）。再设每个字符用 10 bit 进行编码，则每个终端的平均数据率仅 5 b/s。如采用 ALOHA 方式，取 $S = 0.1$，即仅利用信道容量的 10%，则信道的总数据率为 480 b/s。这样的系统一共可容纳 480/5 = 96 个交互式的用户，还是相当不错的。

下面讨论帧的时延。设发完一帧后要经过 R 倍的 T_0 后才能收到确认信息因而才能发送下

一帧。这样，在最好的情况下，发送一帧所需的时间是 $T_0(1+R)$。但若所发送的帧发生碰撞而必须重传，情况就不一样了。设由超时定时器决定重传需要经过的时间也是 R 倍的 T_0。但重传还要经过一段随机的时延。这样，从决定重传到重传完毕所需要的时间是 n 倍的 T_0，而 n 是一个从 1 到某一个事先确定的正整数 K 之间的随机选择出的一个整数（每次重传都要随机选择一次）。重传完毕后，再经过时间 RT_0 才能收到确认信息。图 B-4 画的是重传一次的情况。可以看出，当重传一次时，发送一帧所需的时间（从开始发送起到可以发送下一帧时为止）最小是 T_1，$T_1=T_0+RT_0+T_0+RT_0$；最大是 T_2，$T_2\approx T_0+RT_0+KT_0+RT_0$。若一个帧平均重传 N_R 次才能发送成功，则不难得出发送一个帧总共所需的平均时间为：

$$D = T_0[1+R+N_R(R+(K+1)/2)] \qquad \text{(B-4)}$$

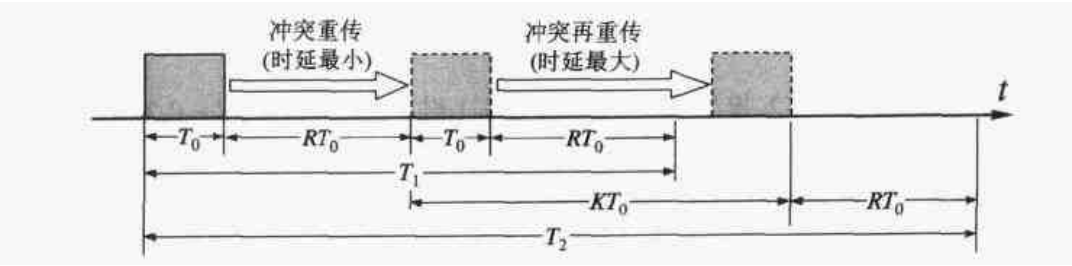


图 B-4　重传帧的时延时间

平均重传次数显然与整数 K 有关。不难想象，K 越小，重传时帧的碰撞机会就越大，因而重传次数也会增多。增大 K 值就可以减少再次碰撞的机会。但若使 K 值变得很大，则发送一帧的平均时延就会很大。理论分析表明，选择 $K=5$ 是一个很好的折衷。在这种情况下，重传次数 N_R 与 K 的关系不大。此时可得出：

$$G/S = 1+N_R \qquad \text{(B-5)}$$

再利用(B-3)式的结果，得出

$$N_R = e^{2G}-1 \qquad \text{(B-6)}$$

(B-6)式表示，当网络负载增大时，帧的重传次数将按指数规律增长。

B.2　时隙 ALOHA(S-ALOHA)

为了提高 ALOHA 系统的吞吐量，可以将所有各站在时间上都同步起来(这要付出代价)，并将时间划分为一段段等长的时隙(slot)，记为 T_0，同时规定，只能在每个时隙开始时才能发送一个帧。这样的 ALOHA 系统叫做时隙 ALOHA 或 S-ALOHA。

图 B-5 为两个站的时隙 ALOHA 的工作原理示意图。图中的一些向上的垂直箭头代表帧的到达。时隙的长度是使得每个帧正好在一个时隙内发送完毕。从图 B-5 可看出，每一个帧在到达后，一般都要在缓存中等待一段时间（这时间小于 T_0），然后才能发送出去。当在一个时隙内有两个或两个以上的帧到达时，则在下一个时隙将产生碰撞。碰撞后重传的策略与纯 ALOHA 的情况是相似的。

现在推导时隙 ALOHA 的吞吐量公式。吞吐量 S 与网络负载 G 的定义与纯 ALOHA 的相同。

参阅图 B-5。设一个帧在某个时隙开始之前到达。显然，此帧能够发送成功的条件是没有其他帧在同一时隙内到达。因此，

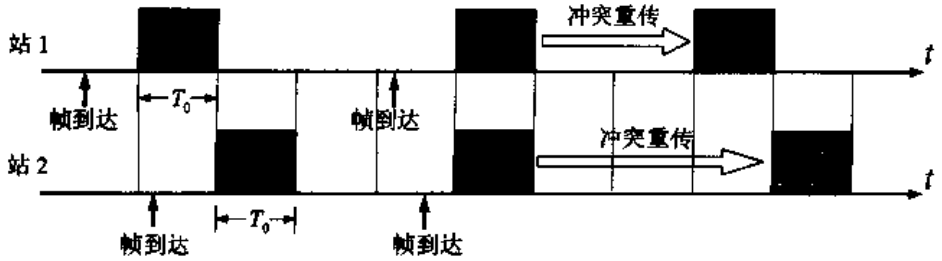


图 B-5　时隙 ALOHA 的工作原理

$$S = G \cdot \mathrm{P}[\text{发送成功}] = G \cdot \mathrm{P}[\text{在 } T_0 \text{ 的时间内有 0 个到达}]$$

$$= G\frac{(G)^0}{0!}\mathrm{e}^{-G}$$

$$= G\mathrm{e}^{-G} \qquad \text{(B-7)}$$

此公式为 Roberts 在 1972 推导出来的。当 $G=1$ 时，$S = S_{\max} = \mathrm{e}^{-1} \approx 0.368$。图 B-6 画出了(B-7)式表示的曲线。为便于比较，纯 ALOHA 的吞吐量且也画在同一坐标中。可以看出，对于时隙 ALOHA，不稳定区域位于 $G>1$ 的部分。

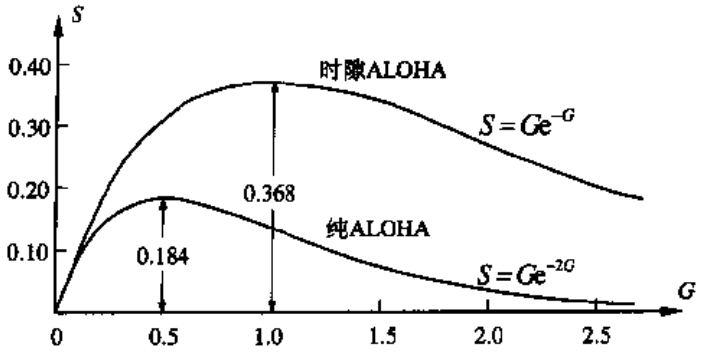


图 B-6　时隙 ALOHA 与纯 ALOHA 的吞吐量曲线

时隙 ALOHA 的发送一帧的平均时间的计算方法与纯 ALOHA 的相似。只是要注意，每个帧到达站的时间是随机的，到下一个时隙的到来平均要等待时间 $T_0/2$，因此现在要在(B-4)式右端两个地方加上 0.5，即

$$D/T_0 = 1.5 + R + N_R[R + 0.5 + (K+1)/2] \quad \text{（时隙 ALOHA）} \qquad \text{(B-8)}$$

这里 N_R 是帧的平均重传次数。当 K 很大时，N_R 与 K 基本无关。这样可以很容易求出：

$$N_R = \mathrm{e}^G - 1 \quad \text{（时隙 ALOHA）} \qquad \text{(B-9)}$$

若 K 不是很大，N_R 将与 K 有关，其关系的计算相当复杂，此处从略[KLEI75a]。实际上，只要 $K \geq 5$，(B-4)式和(B-8)式都还是相当准确的。

K 的大小对帧的时延有很大的影响。K 太大会使时延增大。但 K 太小又会使重传时的碰撞机会增大，这反而会增加重传次数。可见存在一个最佳的 K 值。但帧的时延对 K 值的选择并不灵敏（只要 S 不是太接近于极限值）。一般可取 $K=5$。

图 B-7 画的是两种 ALOHA 的归一化的帧平均传输时延 D/T_0 与吞吐量 S 的关系曲线。这是在忽略传播时延并令 $K=5$ 的条件下得出的。从两条曲线的对比可看出，当吞吐量很小时，纯 ALOHA 的性能要稍好一些。但当吞吐量增大时，纯 ALOHA 的时延会急剧上升（尤其是当 S 接近于 0.18 时），而对时隙 ALOHA 却可以在更高的吞吐量下工作。

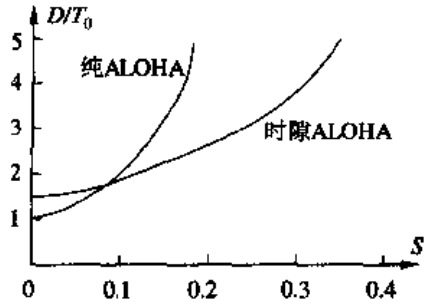


图 B-7　帧的平均传输时延与吞吐量的关系曲线（无传播时延，$K=5$）

最后还要强调一下，这两种 ALOHA 的吞吐量公式的推导，都是假定站的数目很大（理论上应为无穷大），而每一个站发送一个帧的概率很小（理论上应趋向于零），因为只有在这个条件下，各站随机地发送帧的总效应才相当于泊松过程。然而在实际上站的数目总是有限的。这样就产生一个问题：对于有限的站数，如使用前面推导的公式，究竟会带来多大的误差。

现在以时隙 ALOHA 为例，来研究有限站数的吞吐量公式。

假设共有 N 个站。各站独立地随机发送帧，一个时隙的长度正好可以发送一个帧。设 S_i 为站 i 在任一时隙成功发送一个帧的概率。于是，$1-S_i$ 为站 i 在任一时隙没有发送成功（发送失败或根本没有发送）的概率。再设 G_i 和 $1-G_i$ 分别为站 i 在任一时隙发送和不发送一个帧的概率。显然，对所有 i，我们有 $S_i \le G_i$。

因为各站发送帧是独立的，所以

$$S_j = G_j \prod_{\substack{i=1 \\ i \ne j}}^{N} (1-G_i) \qquad \text{(B-10)}$$

现在再设各站的统计特性都相同，即 $S_i = S/N$ 和 $G_i = G/N$，而 S 和 G 分别为整个系统的吞吐量和网络负载，则(B-10)式可化简为：

$$S = G(1-G/N)^{N-1} \qquad \text{(B-11)}$$

这就是有限站数的 ALOHA 系统的吞吐量公式。利用公式

$$\lim_{n\to\infty}(1+x/n)^n = e^x ,$$

(B-11)式在 $N \to \infty$ 时变为

$$S = \lim_{N\to\infty} G(1-G/N)^{N-1} = Ge^{-G} \qquad \text{(B-12)}$$

这正是前面导出的(B-7)式。

对(B-12)式的 S 求极值。得出当 $G=1$ 时，S 达极大值

$$S_{\max} = (1-1/N)^{N-1} \qquad \text{(B-13)}$$

表 B-1 列出了不同 N 值和相应的 $S_{\max}$ 值。

表 B-1　时隙 ALOHA 的最大吞吐量与站数的关系

N	1	2	3	5	10	20	100	∞
$S_{\max}$	1	0.5	0.444	0.410	0.387	0.377	0.370	0.368

从表中所列数值可以看出，只要有 20 个站（或更多些），就可以利用无穷多站的模型所得出的各种结论和公式。我们还可看出，只有在 $N=1$ 时，$S_{\max}$ 才等于 G，这时没有重传的

帧。随着站数的增多，S_{max} 值迅速下降，最后趋于 1/e。

习题

B-01 试用其他方法导出(B-3)式。例如，从“$G = S +$ 平均重传次数”出发，求出平均重传次数为 $G(1-e^{-2G})$，然后解出 S 来。（提示：计算至少发生一次冲突的概率。）

B-02 若干个终端用纯 ALOHA 随机接入协议与远端主机通信。信道速率为 2400 kb/s。每个终端平均每 2 分钟发送一个帧，帧长为 200 bit，问终端数目最多允许为多少？若采用时隙 ALOHA 协议，其结果又如何？若改变以下数据，分别重新计算上述问题：

（1）帧长变为 500 bit。

（2）终端每 3 分钟发送一个帧。

（3）线路速率改为 4800 b/s。

B-03 在纯 ALOHA 协议中，若系统工作在 $G = 0.5$ 的状态，求信道为空闲的概率。

B-04 在时隙 ALOHA 协议中，若帧长为 k 个时隙的时间，而帧可在任一时隙开始时发送出去。试计算此系统的吞吐量。由此导出 $k = 1$ 和 $k \to \infty$ 时的结果，并加以解释。

B-05 10000 个终端争用一条公用的时隙 ALOHA 信道。平均每个终端每小时发送帧 18 次。时隙长度为 125 μs。试求网络负载 G。

B-06 时隙 ALOHA 的时隙为 40 ms。大量用户同时工作，使网络每秒平均发送 50 个帧（包括重传的）。

（1）试计算第一次发送即成功的概率。

（2）试计算正好冲突 k 次然后才发送成功的概率。

（3）每个帧平均要发送多少次？

B-07 若时隙 ALOHA 系统有 10 % 的时隙是空闲的，问网络负载 G 和吞吐量 S 各等于多少？现在系统过载否？

B-08 一时隙 ALOHA 系统有 4 个站，各站在一个时隙内的帧发送率分别为 $G_1 = 0.1, G_2 = 0.5, G_3 = G_4 = 0.2$。试计算每一个站的吞吐量和整个系统的吞吐量以及空闲时隙所占的比例。

B-09 试证明：在采用时隙 ALOHA 协议时，各站都相同的有限用户系统的最大吞吐量发生在 $G = 1$ 时。

B-10 一站数很大的时隙 ALOHA 系统在工作时，其空闲时隙占 65 %。试求 S 和 G。

附录 C　综合业务数字网 ISDN

C.1　窄带综合业务数字网 N-ISDN

当人们认识到数字技术的优越性时，就将其用于模拟通信网中的传输系统或交换系统。于是人们就设想使各种不同业务的信息，经过数字化后，都在广域网中传送和交换。这就是**综合业务数字网 ISDN** (Integrated Services Digital Network)。后来由于又提出了**宽带综合业务数字网 B-ISDN**，因此 ITU-T（即前 CCITT）在 20 世纪 70 年代中期提出的 ISDN 就称为**窄带 ISDN**，或 **N-ISDN**。ISDN 的国际标准由 ITU-T 制订。

ISDN 的信令采用**共路信令** CCS (Common-Channel Signaling)。我们知道，信令系统是电信网的神经中枢，它使各交换机之间传递和交换必要的信息，使网络能够正常运行。传统的公用电话网的信令是采用**随路信令**（又称为**带内信令**），即局间信令和话音信号都在同一个电话网中的标准 4 kHz 话路中传送。这种信令功能较差，传送速率慢，且易受干扰。1976 年美国开始在交换局之间建造一个分组交换网用来传送共路信令。这个传送信令的网络就叫做**共路局间信令网**。共路局间信令网在使用程控交换局的基础上，利用高速链路以分组交换方式传送局间信令。一群话路（如几百条）可以用分时方式共享一条共路信令的链路。由于共路信令在逻辑和物理上均与电话信号相隔离，故又称为**带外信令**。共路信令主要用于：①呼叫建立、路由选择和呼叫释放；②内部数据库访问；③网络运行与支持；④计费。

7 号信令系统(SS7)是最新的共路信令系统。它的总目标是提供一个国际标准化的、具有普遍目的的共路信令系统，使具有程控交换机的数字通信网运行在最佳状态，并能提供一种按序的、无丢失、不重复和可靠的信息传输手段。

ISDN 最基本的概念就是在用户和 ISDN 之间的连线相当于一个**数字比特管道**。管道中的双向比特流可来自数字电话机或数字传真机等其他终端。这种数字比特管道用时分复用方式可支持多个独立**通路**(channel)[①]。复用的比特流的格式在接口标准中都有明确的规定。家庭用较小的带宽即可。而单位则要用较高的带宽，甚至多个数字比特管道。

ISDN 定义了一些标准化的通路，都各用一个英文字母表示。其中最常见的是 B 通路(64 kb/s 的数字 PCM 话音或数据通路)和 D 通路（16 或 64 kb/s 用作带外信令的数字通路）。在 ITU-T 规定的标准化组合中，以下两种是最重要的：

（1）**基本速率** 2B + D = 144 kb/s，这里 D 通路为 16 kb/s。这种速率是为了给家庭或小单位提供的服务。这里一个 B 通路用于电话，而另一个 B 通路用于传送数据。

（2）**一次群速率** 23B + D（美国和日本）或 30B + D（欧洲），这里 D 通路为 64 kb/s。一次群速率可适应北美的 T1 系统(1.544 Mb/s)或 E1 系统(2.048 Mb/s)。

B 通路可支持电路交换的数字电话和数据等业务，也可支持分组交换的数据。

---

① 注：对 channel 的标准译名是："信道"，又称"通路" [MINGCI93]。本书选用了后者。

ITU-T 将 ISDN 提供的业务分为基本业务和补充业务。基本业务又分为以下两种：

（1）**承载业务**(Bearer Service)　这是网络向用户提供的低层信息传递能力。

（2）**用户终端业务**(Teleservice)　这种业务不仅使用信息传递的低层功能，同时还包含高层功能。这是终端操作员利用终端实际上所获得的业务能力。

补充业务是对基本业务的改变或增添，通常可与多个基本业务结合供用户使用。

当 ISDN 的思想提出后，有不少人曾设想今后可能用 N-ISDN 来代替传统的电话网。然而事与愿违，由于技术发展得很快，N-ISDN 的标准刚一制订出来，其技术水平就已经不够先进了。ISDN 的 64 kb/s 的 B 通路速率就是和 10 Mb/s 的以太网相比也是很不相称的，更不用说用来传送宽带的图像信息了。这就使得 N-ISDN 的发展和当初设想的不一样，有些人甚至认为 N-ISDN 已经没有什么前途了。然而近几年来因特网的用户急剧增长，使得 N-ISDN 又找到了一些市场。这就是用户可以使用一条 B 通路上网，而用另一条 B 通路打电话。或者用整个基本速率共 144 kb/s 的数字链路接入到因特网。这就是电信部门宣传的“一线通”，它的一个很大的好处就是使只拥有一条电话线的用户在上网的同时，还能够接打电话，并且上网的速率比使用 56 kb/s 调制解调器的效果还要略好些。因此，在今后的一段时期，N-ISDN 还有可能会得到某种程度的发展，但这可能只是一种过渡的网络。

C.2　宽带综合业务数字网 B-ISDN

随着电子技术的飞速发展，特别是光纤技术、VLSI 技术、光盘存储技术、高速高分辨率工作站的出现，一方面，数据传输的速率已越来越快，另一方面，各种新的业务也不断涌现，使用户对高速网络的需求显得格外迫切（如传输高保真度的音频信号和电视电话就需要 Mb/s 级的传输速率，而传输高质量活动图像则需 Gb/s 级的传输速率）。现在千兆以太网和 10 千兆以太网已经问世，而每秒几十或上百吉比的传输速率也在广域网上实现了。

由于 N-ISDN 很难适应用户的宽带需求，因此在 N-ISDN 还远未广泛推广使用时，一种新型的**宽带综合业务数字网** B-ISDN (Broadband-ISDN)的思想就提出来了。

宽带综合业务数字网 B-ISDN 也是企图将各种业务，如话音、数据、图像以及活动图像都综合在一个宽带网络中进行传送和交换，包括了 N-ISDN 所有的业务功能[PRYC93]。B-ISDN 的最重要的任务就是要以全新的交换体制来支持所有可能的电信业务。更具体些，B-ISDN 与 N-ISDN 相比，具有以下的一些重大区别：

（1）N-ISDN 使用的是电路交换。只是在传送信令的 D 通路使用分组交换。B-ISDN 使用的交换方式是**快速分组交换**，即**异步传递方式** ATM。

（2）N-ISDN 是以目前正在使用的电话网为基础，其用户环路采用双绞线（铜线）。但在 B-ISDN 中，**其用户环路和干线都采用光缆**（短距离的通信也可使用双绞线）。

（3）N-ISDN 各通路的比特率是预先设置的。如 B 通路比特率为 64 kb/s。但 B-ISDN 使用**虚通路**的概念，其比特率只受用户到网络接口的物理比特率的限制。

（4）N-ISDN 无法传送高速图像，但 B-ISDN 可以传送服务质量有保证的高速图像。

虽然 B-ISDN 的想法看来也不错，但由于使用 IP 技术的因特网的飞速发展，以及由于 ATM 设备的过于昂贵，因此 B-ISDN 的发展远远不如当初设想的那样快。到现在，人们关心的是：传统电信网将如何演进到以 IP 为核心的下一代网络，而 B-ISDN 也已成为了历史名词。

附录 D　关于 ATM 的通信量

D.1　ATM 通信量的特点

ATM 网络的设计目标是在同一个网络中支持各种服务，而不同服务的**通信量(traffic)特性**和对服务质量的要求则各不相同。传统分组交换网或帧中继网的流量控制和拥塞控制机制无法实现 ATM 网络的设计目标。这是因为：

（1）在 ATM 网络中有许多的通信量并不适于进行流量控制。例如，当网络发生拥塞时，音频或视频数据的源端点并不能暂停产生信元。

（2）由于信元的传输时延远远小于信元在网络中的传播时延，进行拥塞控制的反馈就显得太慢。

（3）ATM 网络支持的应用所需的传输容量的变化范围很大，从每秒几千比特到每秒几百兆比特。简单的拥塞控制方法无法适应这样大的数据率变化范围。

（4）ATM 网络上的应用可以产生差异很大的通信量模式，从恒定比特率到可变比特率。传统的拥塞控制技术很难处理这样多种的通信量模式。

（5）ATM 网络上的不同应用需要不同的网络服务。如音频和视频数据需要对时延敏感的服务，而文件数据则需要对丢失敏感的服务。

（6）交换和传输的极高速率使得 ATM 网络在进行拥塞控制和通信量控制时，变得难于预测。有时甚至会在流量控制中产生很大的和无用的波动。

在 ATM 网络中对通信量控制有重要影响的是以下的两个特性：

（1）在物理媒体中存在大量正在传送的比特

现考虑 ATM 信元以 622 Mb/s 的速率向网络发送。在此速率下，向网络发送一个信元所需的时间是$(53\times 8)/(622\times 10^6)\approx 0.68\times 10^{-6}$s。假定一条连接长度为 2000 km，其传播时延为 10 ms（这里忽略 ATM 交换机带来的时延），因而往返时延为 20 ms。

现假定连接的一个端点 A 向另一个端点 B 传送一个大的文件。A 至少要经过往返时延 20 ms 才能收到 B 发现网络出现拥塞的报告。但在 20 ms 内，A 已经向网络注入了 N 个信元，而

$$N=(20\times 10^{-3})/(0.68\times 10^{-6})\approx 29.4\times 10^3$$

也就是说，在 A 能够对网络的拥塞作出响应之前，就已经向网络注入了近 3 万个信元，相当于在网络的媒体中注入了超过 12 Mb 的数据量。这就可以解释为什么许多传统的能够使用的拥塞控制技术不能用于 ATM 网络。

（2）信元的时延偏差

时延偏差也称为时延抖动（见 10.1 节）。产生信元时延偏差的主要原因有两个。一个是网络中的通信量过大，出现了拥塞。另一个原因是在物理层将 ATM 层交下来的信元在时间上有重叠（见图 D-1）。

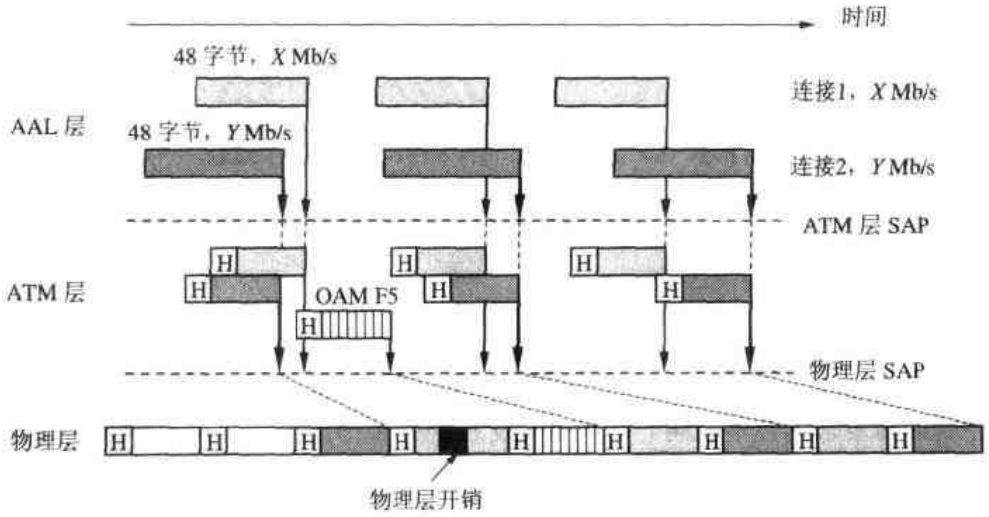


图 D-1　产生信元时延偏差的原因

假定两个端点之间建立了两个连接：连接 1 和连接 2，其数据率分别为 X Mb/s 和 Y Mb/s。这两个连接上都是每隔一定时间发送一个数据块。在 AAL 层最后形成的 PDU 都是 48 字节长，但在横坐标为时间的图上却不一样长，其长度分别是$(48 \times 8)/X$ 和$(48 \times 8)/Y$，长度的单位是微秒。AAL 层的 PDU 通过 ATM 层的服务访问点，交给了 ATM 层。再通过 ATM 层的服务访问点交给物理层。最后到物理层传送时，所有的信元必须一个接一个地串行传送。

现在可以看出产生信元时延的因素有这样一些：

（1）TM 信元形成后不能立即插入到物理层，因为物理层已经有了一个信元正在传送；

（2）连接 1 和连接 2 的信元在时间上可能会有一些重叠；

（3）物理层要加上必要的开销（如 SDH 帧首部中的控制字段）；

（4）ATM 层还要产生少量的 OAM 管理信元插在用户的信元流之中。

上述诸因素产生的信元时延都是不可预见的，因此信元的端到端时延不是恒定的。

针对以上 ATM 通信量的特点，需要设计一种新的通信量管理(traffic management)机制[STAL98]。这种通信量管理机制的要点就是当用户建立连接时都必须与网络达成一个**合约**(contract)：用户受合约规定的通信量特性的约束，而网络满足用户的服务质量要求。在合约中要使用下一节所讨论的一些参数。

D.2　ATM 通信量管理中的一些重要参数

描述用户的通信量特性的主要参数（对每一条虚连接）有以下四个：

（1）**峰值信元速率** PCR (Peak Cell Rate)　用户打算发送信元的最高速率。例如，发送端打算每 4 μs 向网络注入一个信元，则 PCR 为 250 000 信元/秒，虽然信元的实际发送时间可能只有 2.7 μs。

（2）**持续信元速率** SCR (Sustainable Cell Rate)　在一段时间内（但这段时间应远大于在峰值信元速率 PCR 下信元之间的时间间隔）的平均信元速率。对于恒定比特率的服务，SCR = PCR。但对于其他一些服务种类，SCR 可能远小于 PCR。PCR 与 SCR 的比值可用来度量数据流的突发性。请注意：持续信元速率 SCR 并不是在任意一段时间内的平均信元速率！SCR 不等于在任意一段很长的时间内发送的信元总数除以总的发送时间。在发送突发性数据时，

SCR 大于平均信元速率。SCR 是一个 ATM 连接上的平均信元速率的上限值。有了参数 SCR 就使得网络可以给许多可变比特率的用户合理地分配网络资源，而不必给每一个用户按照其 PCR 分配资源。

（3）**最大突发长度** MBS (Maximum Burst Size)　在 PCR 速率下可连续发送的信元最大数目。当用户按照 MBS 发送整块的数据时，数据块之间必须有足够的间隙使其持续数据率不超过参数 SCR。

（4）**最小信元速率** MCR (Minimum Cell Rate)　是用户能够接受的最小信元传送速率。若网络不能保证此 MCR 值，用户可拒绝此连接。在可用比特率 ABR 中，实际的信元速率在 MCR 与 PCR 之间，并且会动态地变化。MCR 值可以为零。

描述一个连接上的通信量的参数是：

容许的信元时延偏差 CDVT (Cell Delay Variation Tolerance)　每一个信元在网络中经受的时延均有差异。当传送话音或视像信息时，应对信元时延偏差规定其上限，否则就不能容许。此参数与 PCR 或 SCR 均无关。

描述 ATM 的服务质量 QoS 的主要参数有以下三个：

（1）**信元传送时延** CTD (Cell Transfer Delay)　准确地说，这是从一个信元的最后一个比特离开源 UNI 到该信元的第一个比特到达目的 UNI 所经历的时间。实际上，CTD 是一个变量，应当用它的概率密度函数来表示。图 D2 说明了这一情况。图中的的固定时延 minCTD 是每一个信元都要经历的时延，它包括经过物理媒体的传播时延、发送时延，以及 ATM 交换机带来的处理时延。在固定时延的基础上还要加上可变时延。这部分时延是由交换机对信元进行缓存和调度所引起的时延。对每一个连接都应规定一个最大信元传送时延 maxCTD，只要时延超过 maxCTD 的信元由于已经失去应用价值就应将其丢弃，或在某些情况下延迟交付。这部分信元所占的比例为 α。α 是一个很小的数，其量级约为 10^{-10}。参数 CDV 标识到达时间的分散程度。对于实时通信，参数 CDV 往往比比 CDT 更为重要。

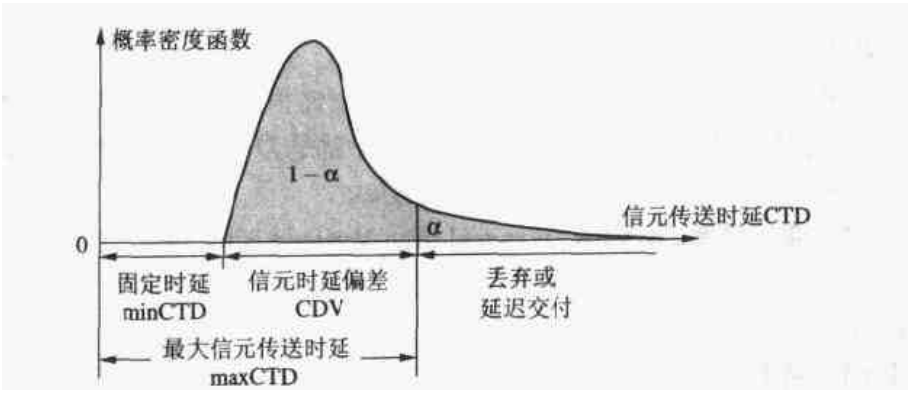


图 D-2　信元传送时延的概率密度函数

（2）**信元时延偏差** CDV (Cell Delay Variation)　在一个连接中 maxCTD 与 minCTD 之差。CDV 有时也称为**峰峰信元时延偏差**(peak-to-peak CDV)。

不应将 CDV 与 CDVT 弄混。参数 CDV 在连接建立时要进行协商（对于交换虚连接），而参数 CDVT 通常都是在 UNI 设置的，并且是不能进行协商的。

（3）**信元丢失率** CLR (Cell Loss Ratio)　在一个连接中丢失信元数与所有传输的信元数

之比。

ATM 的通信量管理机制分为**基于连接**的通信量管理和**基于信元**的通信量管理。前者是在连接建立阶段进行的，而后者则是在连接建立后在传送这种信元所使用的管理机制。

基于连接的通信量管理对于可以预测通信量特性的应用（比如实时连续流类型的服务）很有效。此类机制包括两种：**连接允许控制** CAC (Connection Admission Control)和网络资源管理。CAC 是指对一个新的连接请求，ATM 网络根据当时的资源情况和新连接的通信量特性，判断网络能否在接入新连接之后仍然保证已有连接的 QoS 和这个新连接的 QoS。若是，则允许接入，否则就拒绝接入。网络资源管理则沿着连接通路管理可用的网络带宽和所有交换机的缓存，使得所有允许接入的应用都能得到所需的网络资源。实际上，连接允许控制 CAC 需要得到网络资源管理的报告，才能知道网络资源是否能满足新的连接的需求。

基于信元的通信量管理包括**使用参数控制** UPC (Usage Parameter Control)、通信量整形 (Traffic Shaping)、调度、缓存管理和反馈控制。UPC 是根据用户与网络的通信量合约而进行的。UPC 的目的是监视用户是否违反了这个合约，并对违约信元进行适当的处理，例如，可丢弃违约信元，也可将信元首部的 CLP 比特从 0 变到 1，使该信元成为低优先级信元。常用的 UPC 算法就是著名的**漏桶算法**。在 ATM 论坛的通信量管理规范中，该算法被称为**一般信元速率算法** GCRA (Generic Cell Rate Algorithm)，用来定义用户通信量违约的标准。

D.3 ATM 服务的五个种类

为了便于通信量管理，ATM 论坛制将 ATM 的服务按照比特率的特点划分为以下 5 个**种类**(category)[W-TM4.0]：

（1）**恒定比特率** CBR (Constant Bit Rate)　用户提出所需的数据率，而吞吐量、时延和时延偏差均能满足要求。CBR 还适用于实时的视像传送系统。

（2）**实时可变比特率** rt-VBR (real-time Variable Bit Rate)　**可变比特率** VBR (Variable Bit Rate)并不是只有一个速率，它定义了一个正常使用的持续数据率和一个在峰值期间偶尔使用的更快的突发数据率（但不能一直使用这种最大速率）。实时可变比特率 rt-VBR 主要用于实时电视会议。这时，屏幕上的画面时而相对静止时而变化很快。当采用 MPEG 标准对视频信号进行压缩时，传输的比特率的变化就很大。rt-VBR 就是为了这种需要而提出的。这时，信元时延的平均值和最大偏差都必须受到严格的控制。

（3）**非实时可变比特率** nrt-VBR (non-real-time Variable Bit Rate)　和 rt-VBR 相似，但不指明时延偏差的上限，同时允许有少量的信元丢失率。属于这类的如多媒体电子邮件和存放在媒体上的视频信息。

（4）**不指明比特率** UBR (Unspecified Bit Rate)　用来支持“尽最大努力交付”的非实时应用。用户随时可发送数据，但服务质量 QoS 不能保证，网络对通信量也没有反馈机制。对于 UBR 也可指明 PCR 或 CDVT，但这都不是必须的。是否要用 UBR 对通信量进行调整，这要由网络来决定。网络在发生拥塞时可将 UBR 信元丢弃。

（5）**可用比特率** ABR (Available Bit Rate)　这类服务是对 UBR 的改进。在传送突发性的数据时，ABR 不仅将信元丢失率 CLR 降低到可接受的程度，而且对网络的可用资源也提供了更加有效的利用。我们知道，当使用恒定比特率传送突发性数据时，若按峰值负荷选择线路带宽，则在轻载时线路的容量将会浪费很多。但若按轻载选择线路带宽，则在重载时又

可能出现拥塞。ABR 的设计目的是使数据业务（不是实时业务）能够充分利用其他高优先级业务(CBR 和 VBR)剩下的可用带宽，并试图在所有的 ABR 用户之间以公平合理的方式动态地共享网络的可用带宽。因此，ABR 可提高网络的利用率而不会影响 CBR 和 VBR 连接的服务质量。当网络处于轻载时，ABR 用户可以按照峰值信元速率 PCR 来发送数据，因而提高了网络的效率。ABR 服务根据网络的当前负荷情况，依靠反馈控制机制调整源端点的发送速率。ABR 用户则按照这种反馈，调整自己的发送速率，因而可获得较小的信元丢失率 CLR(这点是 ABR 和 UBR 的主要区别)和较公平地共享网络的资源。当网络处于重载时，若 ABR 用户不能按照反馈机制降低信元的发送速率，则该 ABR 用户将遭受到明显的信元丢失。ABR 用户指明的通信量参数是峰值信元速率 PCR、容许的信元时延偏差 CDVT 和最小信元速率 MCR。MCR 是 ABR 服务必须给用户提供的最小带宽。若 MCR 为零，则对 ABR 用户就没有保证任何的带宽。但即使是这样，只要信道中还有剩余的带宽，ABR 的源端点也还是可发送数据的。表 D-1 是 TM 4.0 规定的 ATM 的五个服务种类及其属性。

表 D-1　ATM 的五个服务种类及其属性（表中的 ✓ 表示有这个参数或属性）

	CBR	rt-VBR	nrt-VBR	UBR	ABR
通信量参数：					
PCR, CVDT	✓	✓	✓	✓	✓
SCR, MBS, CDVT	–	✓	✓	–	–
MCR	–	–	–	–	✓
服务质量 QoS 参数：					
CLR	✓	✓	✓	–	✓
maxCTD, 峰峰 CDV	✓	✓	–	–	–
其他属性：					
带宽保证	✓	✓	✓	–	选项
适用于实时通信	✓	✓	–	–	–
适用于突发性通信	–	–	✓	✓	✓
用反馈进行流量控制	–	–	–	–	✓
常用的 AAL 类	1	2	3/4, 5	3/5, 5	5
使用场合	话音、视像	话音	数据	数据	数据
举例	T1 或 E1 电路	实时电视会议	多媒体电子邮件	传送文件	浏览网页

图 D-3 说明了 ATM 的几种不同比特率的区别。图的意思很清楚，不需要更多的解释。

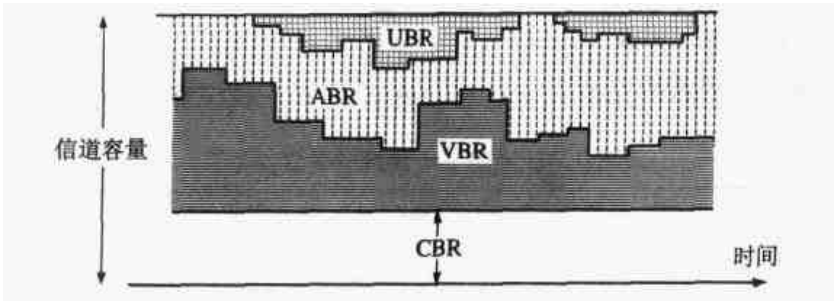


图 D-3　几种不同比特率的区别

附录 E　最短路径算法——Dijkstra 算法

在路由选择算法中都要用到求最短路径算法。最出名的求最短路径算法有两个，即 Bellman-Ford 算法和 Dijkstra 算法。这两种算法的思路不同，但得出的结果是相同的。我们在下面只介绍 Dijkstra 算法，它的已知条件是整个网络拓扑和各链路的长度。

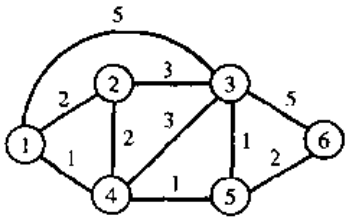


图 E-1　求最短路径算法的网络举例

应注意到，若将已知的各链路长度改为链路时延或费用，这就相当于求任意两结点之间具有最小时延或最小费用的路径。因此，求最短路径的算法具有普遍的应用价值。

下面以图 E-1 的网络为例来讨论这种算法，即寻找从源结点到网络中其他各结点的最短路径。为方便起见，设源结点为结点 1。然后一步一步地寻找，每次找一个结点到源结点的最短路径，直到把所有的点都找到为止。

令 $D(v)$为源结点（记为结点 1）到某个结点 v 的距离，它就是从结点 1 沿某一路径到结点 v 的所有链路的长度之和。再令 $l(i, j)$为结点 i 至结点 j 之间的距离。整个算法只有以下两个部分：

（1）初始化

令 N 表示网络结点的集合。先令 $N=\{1\}$。对所有不在 N 中的结点 v，写出

$$D(v)=\begin{cases} l(1,v) & \text{若结点 } v \text{ 与结点 1 直接相连} \\ \infty & \text{若结点 } v \text{ 与结点 1 不直接相连} \end{cases}$$

在用计算机进行求解时，可以用一个比任何路径长度大得多的数值代替∞。对于上述例子，可以使 $D(v)=99$。

（2）寻找一个不在 N 中的结点 w，其 $D(w)$值为最小。把 w 加入到 N 中。然后对所有不在 N 中的结点 v，用$[D(v), D(w)+l(w, v)]$中的较小的值去更新原有的 $D(v)$值，即：

$$D(v)\leftarrow \text{Min}[D(v),\ D(w)+l(w,\ v)] \qquad \text{(E-1)}$$

（3）重复步骤(2)，直到所有的网络结点都在 N 中为止。

表 E-1 是对图 E-1 的网络进行求解的详细步骤。可以看出，上述的步骤(2)共执行了 5 次。表中带圆圈的数字是在每一次执行步骤(2)时所寻找的具有最小值的 $D(w)$值。当第 5 次执行步骤(2)并得出了结果后，所有网络结点都已包含在 N 之中，整个算法即告结束。

表 E-1　计算图 E-1 的网络的最短路径

步骤	N	$D(2)$	$D(3)$	$D(4)$	$D(5)$	$D(6)$
初始化	{1}	2	5	1	∞	∞
1	{1, 4}	2	4	①	2	∞
2	{1, 4, 5}	2	3	1	②	4
3	{1, 2, 4, 5}	②	3	1	2	4
4	{1, 2, 3, 4, 5}	2	③	1	2	4
5	{1, 2, 3, 4, 5, 6}	2	3	1	2	④

现在我们对以上的最短路径树的找出过程进行一些解释。

因为选择了结点 1 为源结点，因此一开始在集合 N 中只有结点 1。结点 1 只和结点 2, 3 和 4 直接相连，因此在初始化时，在 $D(2)$，$D(3)$和 $D(4)$下面就填入结点 1 到这些结点相应的距离，而在 $D(5)$和 $D(6)$下面填入∞。

下面执行步骤 1。在结点 1 以外的结点中，找出一个距结点 1 最近的结点 w，这应当是 $w=4$，因为在 $D(2)$，$D(3)$和 $D(4)$中，$D(4)=1$，它的之值最小。于是将结点 4 加入到结点集合 N 中。这时，我们在步骤 1 这一行和 $D(4)$ 这一列下面写入①，数字 1 表示结点 4 到结点 1 的距离，数字 1 的圆圈表示结点 4 在这个步骤加入到结点集合 N 中了。

接着就要对所有不在集合 N 中的结点（即结点 2, 3, 5 和 6）逐个执行(E-1)式。

对于结点 2，原来的 $D(2)=2$。现在 $D(w)+l(w,v)=D(4)+l(4,2)=1+2=3>D(2)$。因此结点 2 到结点 1 距离不变，仍为 2。

对于结点 3，原来的 $D(3)=5$。现在 $D(w)+l(w,v)=D(4)+l(4,3)=1+3=4<D(3)$。因此结点 3 到结点 1 的距离要更新，从 5 减小到 4。

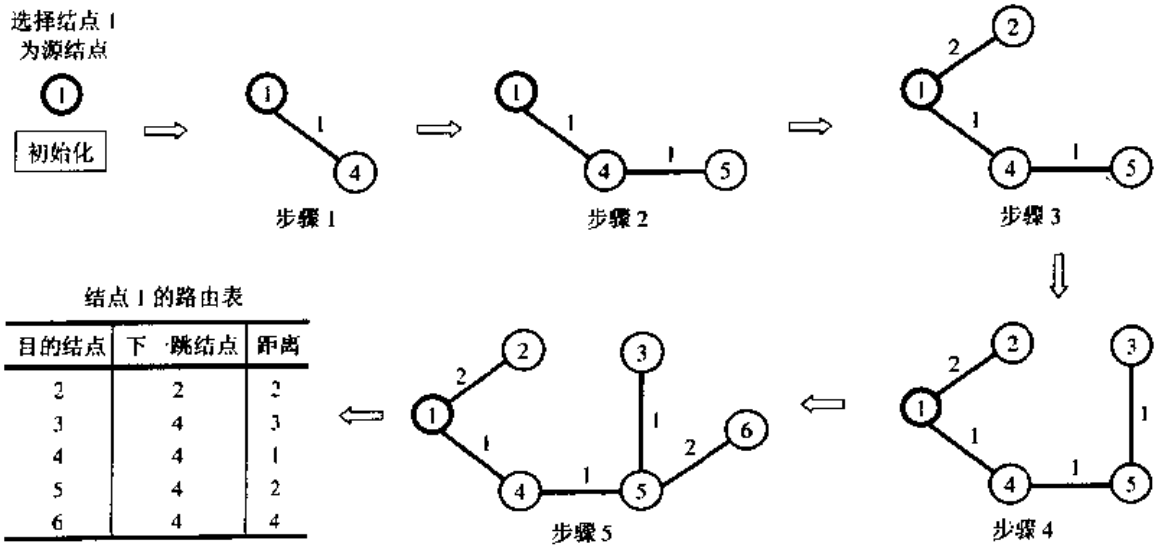
对于结点 5，原来的 $D(5)=\infty$。现在 $D(w)+l(w,v)=D(4)+l(4,5)=1+1=2<D(5)$。因此结点 5 到结点 1 的距离要更新，从∞减小到 2。

对于结点 6，现在到结点 1 的距离仍为∞。

步骤 1 的计算到此就结束了。

下面执行步骤 2。在结点 1 和 4 以外的结点中，找出一个距结点 1 最近的结点 w。现在有两个结点（结点 2 和 5）到结点 1 的距离一样，都是 2。我们选择结点 5（当然也可以选择结点 2，最后得出的结果还是一样的）。以后的详细步骤这里就省略了，读者可以自行完成剩下的步骤。

最后就得出以结点 1 为根的最短路径树。图 E-2 给出了各步骤执行后的结果。从最短路径树可清楚地找出从源结点（结点 1）到网内任何一结点的最短路径。图 E-2 还给出了在结点 1 的路由表。此路由表指出对于发往某个目的结点的分组，**从结点 1 发出后的下一跳结点**（在算法中常称为“后继结点”）和距离。当然，像这样的路由表，在所有其他各结点中都有一个。但这就需要分别以这些结点为源结点，重新执行算法，然后才能找出以这个结点为根的最短路径树和相应的路由表。



结点 1 的路由表

目的结点	下一跳结点	距离
2	2	2
3	4	3
4	4	1
5	4	2
6	4	4

图 E-2　用 Dijkstra 算法求出最短路径树的各个步骤和在结点 1 的路由表

附录 F　部分习题的解答

第 1 章

1-05　不需要。

1-10　分组交换时延较电路交换时延小的条件为：

$$(k-1)p/b < s, \qquad 当\ x >> p\ 时$$

1-11　写出总时延 D 的表达式，求 D 对 p 的导数，令其为零。解出

$$p = \sqrt{xh/(k-1)}$$

1-20　（1）发送时延为 100 s，传播时延为 5 ms。（2）发送时延为 1 μs，传播时延为 5 ms。

若数据长度大而发送速率低，则在总的时延中，发送时延往往大于传播时延。但若数据长度短而发送速率高，则传播时延又可能是总时延中的主要成分。

1-21

媒体长度 l	传播时延	媒体中的比特数	
		数据率 = 1 Mb/s	数据率 = 10 Gb/s
(1)0.1 m	4.35×10^{-10} s	4.35×10^{-4}	4.35
(2)100 m	4.35×10^{-7} s	0.435	4.35×10^{3}
(3)100 km	4.35×10^{-4} s	4.35×10^{2}	4.35×10^{6}
(4)5000 km	0.0217 s	2.17×10^{4}	2.17×10^{8}

1-22　数据长度为 100 字节时，数据传输效率为 63.3%。数据长度为 1000 字节时，传输效率为 94.5%。

第 2 章

2-06　信噪比应增大到约 100 倍。

如果在此基础上将信噪比 S/N 再增大到 10 倍，最大信息速率只能再增加 18.5%左右

2-12　假定连续传送且不出错。

若用 2.4 kb/s 速率，传 600 MB(= 600 × 1 048 576 × 8 = 5 033 164 800 bit，)需时间 24.3 天。若用 33.6 kb/s 速率传送，则需时间 1.73 天。比托人乘火车捎去要慢，且更贵。在计算机领域常用的单位 MB（兆字节）中的 M 并不是精确的 10^6，而是代表 2^{20}，即 1 048 576。在数据率的单位中，M 和 k 分别是精确的 10^6 和 10^3。

2-13　没有突破香农的极限速率。

2-14　“在数字传输时，若传输速率为每秒几个兆比特，则传输距离可达几公里。”这是指使用数字线路，同时其两端的设备并没有带宽的限制。

当我们使用调制解调器与 ISP 相连时，使用的是电话的用户线。这种用户线进入市话交换机，要经过交换机中的滤波器将带宽限制在 3400 Hz 以下，与数字线路的带宽相差很大。

2-17　A 和 D 发送 1，B 发送 0，而 C 未发送数据。

2-18　提示：设发送端和接收端的时钟周期分别为 X 和 Y。若接收端时钟稍慢，则最后一个采样必须发生在

停止比特结束之前，即 $9.5Y < 10X$。若接收端时钟稍快，则最后一个采样必须发生在停止比特开始之后，即 $9.5Y > 9X$。解出：$|(Y-X)/X| < 1/19 = 5.26\%$。因此收发双方频率相差 5%是可以正常工作的（但最好不要这样，因为太临界了）。

第 3 章

3-03 “否则”是指发送方发送的帧的 N(S)与接收方的状态变量 V(R)不同。

这表明发送方没有收到接收方发出的 ACK，于是重传上次的帧。

若“转到(8)”，则接收方要发送 NAK。发送方继续重传上次的帧。一直这样下去。

3-06 假定信道传输无差错。信道利用率为 50%，相当于帧的发送时间等于往返时延。得出帧长为 160 bit。

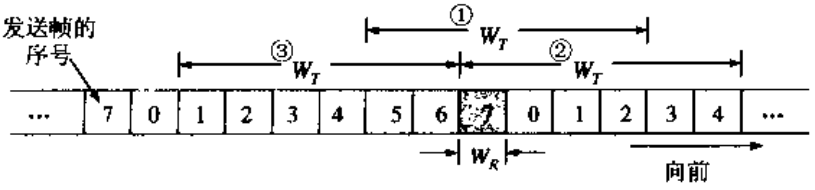
3-07 在一般情况下，确认帧可不用序号。但若超时时间的设置短了一些，则可能会出现问题，即有时发送方会分不清对哪一个帧的确认（请读者自己试试找出这种情况）。

3-09 提示：如题图 3-09 所示。设接收窗口正好在 7 号帧处（有阴影的帧）。发送窗口 W_T 的位置不可能比②更靠前，也不可能比③更靠后，也可能不是这种极端位置，如①。

对于①和②的情况，在 W_T 的范围内无重复序号，即 $W_T \le 2^n$。

对于③的情况，在 $W_T + W_R$ 的范围内无重复序号，即 $W_T + W_R \le 2^n$。

现在 $W_R = 1$，故发送窗口的最大值 $W_T \le 2^n - 1$。



题图 3-09

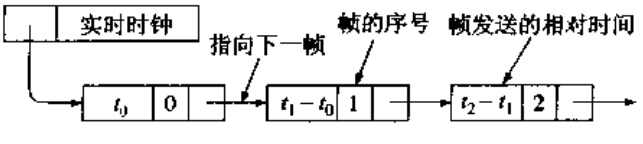
3-10 因 $W_T + W_R \le 2^n$，而 W_R 的最大值不能超过 W_T（否则无意义），故得出(3-18)式。

3-11 设想在发送窗口内的序号为 0, 1, 2, 3, 4, 5，而接收窗口等待后面的 6, 7, 0。接收端若收到 0 号帧，则无法判断是新帧或重传的（当确认帧丢失）。

3-12 设想在发送窗口内的序号为 0, 1, 2, 3, 4, 5, 6, 7。而接收窗口等待后面的 0。接收端若收到 0 号帧，则无法判断是新帧或重传的（当确认帧丢失）。

3-13 例如，当传输无差错时，或当 $W_R = 1$ 时。

3-14 用相对发送时间实现一个链表（题图 3-14）。



题图 3-14

3-15 （1）$U = 1/251$。（2）$U = 7/251$。（3）$U = 127/251$。（4）$U = 1$。

3-18 一般设置初始序号为 0 较为方便。

3-19 PPP 使用在线路质量不太差的情况下。

3-20 添加的检验序列是 1110。出现的两种差错都可以发现。

3-21 7E FE 27 7D 7D 65 7E

3-22 提示：需要将 64 KB 数据划分为 33 个帧。一帧中只要有一个比特出错就产生错误的帧。计算出误帧率 p_f = 0.016251。用公式(3-5)算出成功发送一帧的平均时间为 0.0212 秒。考虑到有两条链路和 33 个帧，因此总的平均发送时间为 1.4 秒。这个时间比直接向链路上发送 64 KB 数据要多用三倍的时间。

3-23 第一个比特串：经过零比特填充后变成 011011111011111000（加上下划线的 0 是填充的）。

另一个比特串：删除发送端加入的零比特后变成 000111011111-11111-110（连字符表示删除了 0）。

第 4 章

4-03 （1）集线器在第 4 层中央。电缆总长度为 1832 m。

（2）水平的电缆需 392 m，垂直的电缆需 24 m，共需 416 m。

（3）电缆采用螺旋形布线，经过的点为：(1, 1), (15, 1), (15, 7), (1, 7), (1, 2), (14, 2)等等。电缆总长度约为 462 m。

4-04 20 MBaud

4-05 提示：将第 i 次重传失败的概率记为 P_i，显然

$$P_i=(0.5)^k, \quad k=\min[i, 10]$$

故第 1 次重传失败的概率 $P_1=0.5$，第 2 次重传失败的概率 $P_2=0.25$，第 3 次重传失败的概率 $P_3=0.125$。

P[传送 i 次才成功] = P[第 1 次传送失败] P[第 2 次传送失败]…P[第 $i-1$ 次传送失败] P[第 i 次传送成功]

求{P[传送 i 次才成功]}的统计平均值，得出平均重传次数为 1.637。

4-08 （1）10 个站共享 10 Mb/s。（2）10 个站共享 100 Mb/s。（3）每一个站独占 10 Mb/s。

4-09 $a \approx 0.1$。 每个站每秒发送的平均帧数的最大值为 34。

4-10 （1）总线减到 1 km，每个站每秒发送的平均帧数的最大值为 44。

（2）总线速率加倍，a 加倍，每个站每秒发送的平均帧数的最大值为 53。

（3）帧长为 10 000 bit，$a=0.01$。每个站每秒发送的平均帧数的最大值为 4.8。

4-11 最短帧长为 10 000 bit 或 1250 字节。

4-12

	D=25 m		d=2500 m	
	C=10 Mb/s	C=10 Gb/s	C=10 Mb/s	C=10 Gb/s
a	10^{-5}	10^{-2}	10^{-3}	1

4-13 当时很可靠的星形拓扑结构较贵。人们都认为无源的总线结构更加可靠。但实践证明，连接有大量站点的总线式以太网很容易出现故障，而现在专用的 ASIC 芯片的使用可以将星形结构的集线器做得非常可靠。因此现在的以太网一般都使用星形结构的拓扑。

4-14 提示：从这些方面考虑，如组帧、差错控制、流控制、有无确认、面向连接或无连接、地址、网络拓扑等。

4-15 以太网交换机用在这样的以太网，其 20%通信量在本局域网内而 80%的通信量到因特网。

4-16 提示：从网络上负载轻重、灵活性以及网络效率等方面进行比较。

4-17 参数 a 的数值分别为：0.122, 0.0417 和 0.000977。可计算相应的最大吞吐量再进行比较。

4-22 0.5。

4-23

发送的帧	网桥 1 的转发表		网桥 2 的转发表		网桥 1 的处理	网桥 2 的处理
	站地址	端口	站地址	端口		
$H_1 \rightarrow H_5$	MAC1	1	MAC1	1	转发，写入转发表	转发，写入转发表
$H_3 \rightarrow H_2$	MAC3	2	MAC3	1	转发，写入转发表	转发，写入转发表
$H_4 \rightarrow H_3$	MAC4	2	MAC4	2	写入转发表，丢弃不转发	转发，写入转发表
$H_2 \rightarrow H_1$	MAC2	1			写入转发表，丢弃不转发	接收不到这个帧

第 5 章

5-02 四段链路表明需要五个结点交换机。虚电路占用 40 字节的存储器 1000 秒，共 40 000 字节秒，考虑到折旧费，价格是 2.67×10^{-3} 分。数据报在此期间的传输费用比虚电路多用 9.6×10^{-3} 分。可见本题的虚电路较为经济。

5-03 可能被错误投送。端到端的通信不一定可靠。

5-05 （1）$p^2 - 3p + 3$，（2）$1/(1-p)^2$，（3）$(p^2 - 3p + 3)/(1-p)^2$。

5-06 每 $2(n-1)T$ 秒交付一个分组。

5-07 提示：分组长度为 $L = 10^6$ bit，误码率 $p = 10^{-6}$。

P[报文正确到达终点] $=(1-p)^L \approx e^{-1}$。

发送 10^6 bit 的分组需要时间 1 s。正确到达终点平均要将分组传送 e 次，因此需要时间 2.718 s。

若分组长度为 1000 bit，P[一个分组正确到达终点] $=(1-p)^{1000} \approx e^{-0.001}$

发送 10^3 bit 的分组需要时间 0.001 s。1000 个分组正确到达终点平均传送 1001 次。因此，总共需要时间 1.001 s。

5-11 假定 B 收到数据后先向 C 转发，然后向 A 发送确认。B 转发完数据后就出故障，不能向 A 发送确认。A 以后超时重传，而 B 又恢复工作，从 A 收到数据，转发给 C（但 B 不知道是重复的数据，后来机器出故障，修好后重新启动和重新与 A 建立链路连接），然后向 A 发送确认。在这种情况下，C 收到重复的数据。同理，若 B 收到数据后先向 A 发送确认，再将数据转发给 C，则在向 A 发送确认后机器出故障的情况下，当修好机器恢复工作后，A 接着发送数据，造成 C 少受一次数据。

因此，没有端到端的确认是不可靠的。

5-17 在 UNI，VPI 最多有 256 条，每一 VP 可有最多 65 536 条 VC，故 ATM 连接最多可有 16 777 216 条。

在 NNI，VPI 最多有 4096 条，每一 VP 可有最多 65 536 条 VC，故 ATM 连接最多可有 2.68×10^8 条。

5-22 （1）否；（2）否；（3）否；（4）否；（5）否；（6）是。即使在使用永久虚通路时可以随时发送数据，但这仍然是在发送数据之前已经建立了连接（用手工配置的）；（7）否。

5-23 16 个。

5-24 当 $X > 2NLC/(N-1)$ 时复用会更加便宜。

5-25 报文长度加上 8 字节的尾部不是 48 字节的整数倍，因此必须进行填充。填充后要用 22 个信元传送。数据传送的效率是 87.8%。

5-26 ATM 首部共 40 bit，因此发送的信元首部的 40 个比特可用多项式 $B(X)$ 表示，其最高次方为 39。若信元在传输过程中出现误码，则可将误码的比特多项式记为 $E(X)$。收到的信元首部为 $R(X) = B(X) + E(X)$。显然，若 $E(X)$ 恰好能够被生成多项式 $P(X)$ 整除，则 HEC 发现不了这样的差错。因此，当 $E(X)$ 等于 $P(X)$ 乘以任何一个多项式 $M(X)$ 时，$E(X)$ 就能够被生成多项式 $P(X)$ 整除。可见要找的条件是 $E(X) = M(X)P(X)$。现在 $P(X)$ 的最高次方是 8，$E(X)$ 的最高方次是 39，因此 $M(X)$ 的最高方次是 31。可见这种误码共有 2^{31} 种。整个首部 40 bit 的比特排列共有 2^{40} 种。因此，

P[HEC 检测不出一个信元首部有差错] = $2^{31}/2^{40} = 2^{-9} = 1.953\times10^{-3}$。

P[HEC 检测不出一连两个信元首部有差错] = $(2^{-9})^2 = 2^{-9} = 3.81\times10^{-6}$。

第 6 章

6-09 （1）C 类地址对应的子网掩码默认值。但也可以是 A 类或 B 类地址的掩码，即主机号由最后 8 bit 决定，而路由器寻找网络由前 24 bit 决定。（2）6 个主机。（3）子网掩码一样，但子网数目不同。（4）最多可有 4094 个（不考虑全 0 和全 1 的主机号）。（5）有效，但不推荐这样使用。（6）1942.47.20.129，C 类。（7）有。对于小网络这样做还可进一步简化路由表。

6-10 （2）和（5）是 A 类，（1）和（3）是 B 类，（4）和（6）是 C 类。

6-11 好处：转发分组更快。缺点：数据部分出现差错时不能及早发现。

6-12 IP 首部中的源地址也可能变成错误的，请错误的源地址重传数据报是没有意义的。不使用 CRC 可减少路由器进行检验的时间。

6-13 在目的站而不是在中间的路由器进行组装是由于：（1）路由器处理数据报更简单些；（2）并非所有的数据报片都经过同样的路由器，因此在每一个中间的路由器进行组装可能总会缺少几个数据报片；（3）也许分组后面还要经过一个网络，它还要给这些数据报片划分成更小的片。如果在中间的路由器进行组装就可能会组装多次。

6-14 由于分片，共分为 4 个数据报片，故第二个局域网向上传送 3840 bit。

6-15 （1）不能说“ARP 向网络层提供了服务”，因为 ARP 本身是网络层的一部分（但 IP 使用 ARP）。数据链路层使用硬件地址而不使用 IP 地址，因此 ARP 不在数据链路层。

（2）当网络中某个 IP 地址和硬件地址的映射发生变化时，ARP 高速缓存中的相应的项目就要改变。例如，更换以太网网卡就会发生这样的事件。10 ~ 20 分钟更换一块网卡是合理的。超时时间太短会使 ARP 请求和响应分组的通信量太频繁，而超时时间太长会使更换网卡后的主机迟迟无法和网络上的其他主机通信。

（3）在源主机的 ARP 高速缓存中已经有了该目的 IP 地址的项目；源主机发送的是广播分组；源主机和目的主机使用点对点链路。

6-16 （1）接口 0，（2）R_2，（3）R_4，（4）R_3，（5）R_4。

6-18 3 个。数据字段长度分别为 1480, 1480 和 1020 字节。片偏移字段的值分别为 0, 185 和 370。MF 字段的值分别为 1, 1 和 0。

6-20 （1）255.128.0.0，（2）255.224.0.0，（3）255.248.0.0，（4）255.252.0.0，（5）255.254.0.0，（6）255.255.0.0。

6-21 只有（4）是推荐使用的。

6-22 共同前缀是 22 位，即：11010100 00111000 100001。

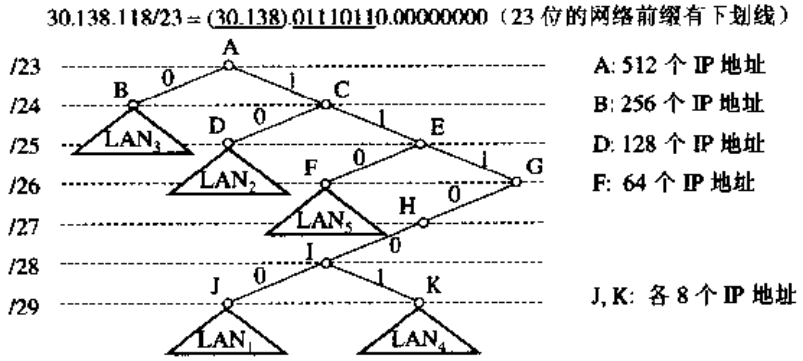
聚合的 CIDR 地址块是：212.56.132.0/22。

6-23 前一个地址块包含了后一个。写出这两个地址块的二进制表示就可看出。

6-24 分配网络前缀时应先分配地址数较多的前缀。题目没有说 LAN_1 上有几个主机，但至少需要三个地址给三个路由器用。本题的解答有很多种，下面给出两种不同的答案：

	第一组答案	第二组答案
LAN_1	30.138.119.192/29	30.138.118.192/27
LAN_2	30.138.119.0/25	30.138.118.0/25
LAN_3	30.138.118.0/24	30.138.119.0/24
LAN_4	30.138.119.200/29	30.138.118.224/27
LAN_5	30.138.119.128/26	30.138.118.128/27

可以将第一组答案用图 6-25 所示的二叉线索画出（见题图 6-24）。这样可看得清楚些。图中注明有 LAN 的三角形表示在三角形顶点下面所有的 IP 地址都包含在此局域网的网络前缀中。



题图 6-24

6-25 本题的解答有很多种，下面给出其中的一种答案（先选择需要较大的网络前缀）：

LAN_1：192.77.33.0/26。

LAN_3：192.77.33.64/27；LAN_6：192.77.33.96/27；LAN_7：192.77.33.128/27；LAN_8：192.77.33.160/27。

LAN_2：192.77.33.192/28；LAN_4：192.77.33.208/28。

LAN_5：192.77.33.224/29（考虑到以太网上可能还要再接几个主机，故留有余地）。

WAN_1：192.77.33.232/30；WAN_2：192.77.33.236/30；WAN_3：192.77.33.240/30。

6-26 观察地址的第二个字节 0x32 = 00100000，前缀 12 位，说明第二字节的前 4 位在前缀中。

给出的四个地址的第二字节的前 4 位分别为：0010, 0100, 0011 和 0100。因此只有（1）是匹配的。

6-27 前缀（1）和地址 2.52.90.140 匹配。

6-28 前缀（4）和这两个地址都匹配。

6-29 （1）/2；（2）/4；（3）/11；（4）/30。

6-30 （1）、（2）和（4）都可以再分配给其他子网，但（3）不能。

6-33 RIP 只和邻站交换信息，UDP 虽不保证可靠交付，但 UDP 开销小，可以满足 RIP 的要求。OSPF 使用可靠的洪泛法，并直接使用 IP，好处是很灵活性好和开销更小。BGP 需要交换整个的路由表（在开始时）和更新信息，TCP 提供可靠交付以减少带宽的消耗。

RIP 使用不保证可靠交付的 UDP，因此必须不断地（周期性地）和邻站交换信息才能使路由信息及时得到更新。但 BGP 使用保证可靠交付的 TCP,因此不需要这样做。

6-34 路由器 B 更新后的路由表如下：

N_1	7	A	无新信息，不改变。
N_2	5	C	相同的下一跳，更新。
N_3	9	C	新的项目，添加近来。
N_6	5	C	不同的下一跳，距离更短，更新。
N_8	4	E	不同的下一跳，距离一样，不改变。
N_9	4	F	不同的下一跳，距离更大，不改变。

6-35 路由器 A 更新后的路由表如下：

N_1	3	C	不同的下一跳，距离更短，改变。
N_2	2	C	相同的下一跳，距离一样，不变。
N_3	1	F	不同的下一跳，距离更大，不改变。
N_4	5	G	不同的下一跳，距离更大，不改变。

6-37 对首部的处理更简单。数据链路层已经将有差错的帧丢弃了，因此网络层可省去这一步骤。但可能遇到数据链路层检测不出来的差错（此概率极小）。

6-38 在 IP 数据报传送的路径上的所有路由器都不需要这一字段的信息。只有目的主机才需要协议字段。在 IPv6 使用“下一个首部”字段完成 IPv4 中的“协议”字段的功能。

6-39 从概念上讲没有改变，但因 IPv6 地址长度增大了，所以相应的字段都需要增大。

6-40 IPv6 的地址空间共有 2^{128} 个地址，或 3.4×10^{38}。1 秒钟分配 10^{18} 个地址，可分配 1.08×10^{13} 年。大约是宇宙年龄的 1000 倍。对于，地址空间的利用不会是均匀的。但即使只利用整个地址空间的 1/1000，那也是不可能用完的。

6-41 （1）::F53:6382:AB00:67DB:BB27:7332；

（2）::4D:ABCD；

（3）::AF36:7328:0:87AA:398；

（4）2819:AF::35:CB2:B271。

第 7 章

7-01 （2）都是。这要在不同层次来看。在运输层是面向连接的，在网络层则是无连接的。（3）丢弃。

7-06 65 495 字节。此数据部分加上 TCP 首部的 20 字节，再加上 IP 首部的 20 字节，正好是 IP 数据报的最大长度。当然，若 IP 首部包含了选择，则 IP 首部长度超过 20 字节，这时 TCP 报文段的数据部分的长度将小于 65 495 字节。

7-07 这完全可能。设想 A 连续发送两个报文段：（SEQ = 92, DATA 共 8 字节）和（SEQ =100, DATA 共 20 字节），均正确到达 B。B 连续发送两个确认：(ACK = 100)和(ACK = 120)。但前者在传送时丢失了。于是 A 超时重传（SEQ = 92, DATA 共 8 字节），而 B 再次收到该报文段后，发送(ACK = 100)。这样，在这个报文段之前发送的就是(ACK = 120)。

7-08 还未重传就收到了对更高序号的确认。

7-10 26.2 Mb/s。

7-11 630 ms。

7-12 760 ms。

7-17 IP 数据报只能找到目的主机而无法找到目的进程。UDP 提供对应用进程的复用和分用功能，以及提供对数据部分的差错检验。

7-18 UDP 不保证可靠交付，但 UDP 比 TCP 的开销要小很多。因此只要应用程序接受这样的服务质量就可以使用 UDP。如果话音数据不是实时播放（边接收边播放）就可以使用 TCP，因为 TCP 传输可靠。接收端用 TCP 将话音数据接收完毕后，可以在以后的任何时间进行播放。但假定是实时传输，则必须使用 UDP。

7-19 例如，当 IP 数据报在传输过程中需要分片，但其中的一个数据报片未能及时到达终点，而终点组装 IP 数据报已超时，因而只能丢弃该数据报；IP 数据报已经到达终点，但终点的缓存没有足够的空间存放此数据报；数据报在转发过程中经过一个局域网的网桥，但网桥在转发该数据报的帧时没有足够的差错空间而只好丢弃。

7-20 不行。重传时，IP 数据报的标识字段会有另一个标识符。仅当标识符相同的 IP 数据报片才能组装成一个 IP 数据报。前两个 IP 数据报片的标识符与后两个 IP 数据报片的标识符不同，因此不能组装成一个 IP 数据报。

7-21 在 ICMP 的差错报文中（见图 6-27）要包含 IP 首部后面的 8 个字节的内容，而这里面有 TCP 首部中的源端口和目的端口。当 TCP 收到 ICMP 差错报文时需要用这两个端口来确定是哪条连接出了差错。

7-22 TCP 首部除固定长度部分外，还有选项，因此 TCP 首部长度是可变的。UDP 首部长度是固定的。

7-23 6 个。数据字段的长度：前 5 个是 1480 字节，最后一个是 800 字节。片偏移字段的值分别是：0, 1480, 2960, 4440. 5920 和 7400。

7-25 （1）41 字符（2）41 或 42 字符。

7-26 拥塞窗口大小分别为：1, 2, 4, 8, 9, 10, 11, 12, 1, 2, 4, 6, 7, 8, 9。

7-27 最大吞吐量为 25.5 Mb/s。信道利用率为 2.55%。

7-28 8.704 kb/s。

7-29 3 星期零 1.78 天。

7-30 约为 7228 字节。

7-31 40.96 Mb/s。

7-32 源端口 1586，目的端口 69，UDP 用户数据报总长度 28 字节，数据部分长度 20 字节。此 UDP 用户数据报是从客户发给服务器。服务器程序是 TFTP。

7-33 三次算出平均往返时延分别为 29.6, 29.84 和 29.256 ms。RTT 变化多达 20%时，平均往返时延 RTT 的变化还不到 2.5%。

第 8 章

8-13 4200 字节。

8-14 01111010 01001001 01000101 00110100。

8-15 01001100 00111101 00111001 01000100 00111001。编码开销为 66.7%。

8-16 非常困难。例如，人名的书写方法，很多国家（如英、美等西方国家）是先写名再写姓。但像中国或日本等国家则先写姓再写名。有些国家的一些人还有中间的名。称呼也有非常多种类。还有各式各样的头衔。很难有统一的格式。

8-17 有时对方的邮件服务器不工作，邮件就发送不出去。对方的邮件服务器出故障也会使邮件丢失。

8-19 404 Not Found。

8-20 使用 HTTP；0 次；4 次。

8-24 <A HREF="http://www.ietf.org/rfc.html"> 下载 RFC 文档 </A>

8-25 链接的起点是文字：

<A HREF="http://www.topology.net/index.html">网络拓扑</A>。

链接的起点是图：

<A HREF="http://www.topology.net/index.html"> <IMG SRC="map.gif"> </A>

8-28 约 11.6 天。

8-29 在上述 HTML 语句中写入文件 Y 的绝对路径。

8-30 ROM 中没有所举出的任何一个。

8-39 整个的编码为

30 18

02 04 00 00 09 29

02 04 00 00 04 D4

02 04 00 00 00 7A

02 04 00 00 04 D4

8-40 变量 icmpInParmProbs 的对象标识符是 1.3.6.1.2.1.5.5，加上后缀“.0”。

30 29

02 01 00

04 06 70 75 62 6C 69 63

A0 1C

02 04 00 01 06 14

02 01 00

02 01 00

30 0E

30 0C

06 08 2B 06 01 02 01 05 05 00

05 00

8-41 {1.3.6.1.2.1.6}

8-43 40 04 83 15 0E 02

8-47 面向连接的进程使用系统调用 `write()`，
而无连接进程使用系统调用 `sendto()`。

第 9 章

9-01 the time has come the walrus said to talk of many things of ships and shoes and sealing wax of cabbages and kings of why the sea is boiling hot and whether pigs have wings but wait a bit the oysters cried before we have our chat for some of us are out of breath and all of us are fat no hurry said the carpenter they thanked him much for that

From *Through the looking glass* (Tweedledum and Tweedledee)

9-02 a digital computer is a machine that can solve problems for people by carrying out instructions given to it

9-08 C_i 中错了一个比特，会导致整个 P_i 出错。此外，P_{i+1} 中会错一个比特。但以后的明文不会出错。

9-09 从 P_{i+1} 起的所有明文块都会出错。

9-11 提示：可利用以下公式：

$(a \times b) \bmod n = ((a \bmod n) \times (b \bmod n)) \bmod n$，我们有：

$$17^{23} \bmod 55 = 17^{16+4+2+1} \bmod 55 = (17^{16} \times 17^4 \times 17^2 \times 17) \bmod 55$$

$$= [(17^{16} \bmod 55) \times (17^4 \bmod 55) \times (17^2 \bmod 55) \times (17 \bmod 55)] \bmod 55$$

$$(17^4 \bmod 55) = [(17^2 \bmod 55)(17^2 \bmod 55)] \bmod 55$$

（1）有效的 e 值为：7, 11, 13, 17 和 19。

（2）$d = 103$。

（3）$e = 3$。令 X 代表欲加密的明文字符的顺序号，即 X = 1, 2, ..., 10。

加密：$Y = X^3 \bmod 55$。对于明文 X = 1 至 10，得出与之对应的密文 Y = 1, 8, 27, 9, 15, 51, 13, 17, 14 和 10。

（4）$d = 23$。公开密钥是{7, 55}，秘密密钥是{23, 55}。

明文 X = RSA = 18, 19, 1。密文 Y = 17, 24, 1。

第 10 章

10-02 不一样。实时数据往往是等时的数据，但等时的数据不一定是实时数据。

10-07 （2）每一个分组经受的时延分别为（单位为 ms）：30, 25, 22, 29, 45, 35, 29, 26, 20 和 24。

（3）以时间 t 为横坐标，分组数 N 为纵坐标。

$t<45, N=0$；$45 \le t<60, N=1$；$60 \le t<70, N=2$；$70 \le t<73, N=3$；……$t>165, N=0$。

10-08 显然，Δ应小于话音分组长度 10 ms。如果将Δ取为 9 ms，则有：

时钟时间：0 9 18 27 36 45 54 63 72 81 90 99 108 …

计数器值：0 1 2 3 4 5 6 7 8 9 10 11 12 …

话音分组每隔 10 ms 产生一个，对应的时间戳值（即计数器值）为：

话音分组产生时间：0 10 20 30 40 50 60 70 80 90 100 110 …

应加上的时间戳值：0 1 2 3 4 5 6 7 8 10 11 12 …

我们看到时间戳值在 8 到 10 之间缺了一个。可见将Δ取为略小于话音分组长度 10 ms 是不行的。正确的做法是使 2Δ或 3Δ等于话音分组长度。当话音分组丢失时，时间戳值会相差 4Δ或 5Δ，由此来判定是否发生了分组丢失。

10-09 接收端的缓存空间的上限取决于还原播放时所容许的时延。当还原播放时所容许的时延已确定时，缓存空间的上限与实时数据流的数据率成正比。时延抖动越大，缓存空间也应更大。

10-16 $T = b\,l(N-r)$。

10-17 12.5 ms。当 $N = 2500$ pkt/s 时，$T =$ 任意长的时间，漏桶被权标装满后就不再增加权标。

10-23 按题意，此二叉树的叶结点有 2^n 个，故二叉树的深度为 $n+1$。每一个结点向其上游结点发送一个 RESV 报文，故总共发送 $2^{n+1}-1$ 个 RESV 报文。

10-25 例如，在 ATM 网络中每一个信元的首部中的 VPI/VCI 在通过 ATM 交换机时也发生虚通道号/虚通路号的对换。帧中继网络中也是类似的。

10-26 （1）可以。但有关这种 QoS 需求的信息应当使边沿路由器知道。

（2）细粒度：按照源点和终点间的每一个应用流定义 QoS 需求。

粗粒度：按照一组网络前缀或两个网络之间的应用流定义 QoS 需求。

10-27 （1）聚合粒度细；（2）聚合粒度稍粗些，出口 LSR 要检查每一个分组的首部，以便将其分配到合适的终点；（3）这是最粗的聚合粒度，许多网络中的流都将聚合为同一个流，而这种聚合路径通常都在 MPLS 的主干网中。

10-28 （1）当路由表很大时查找最长前缀匹配需要很长时间，这就限制了网络的规模；（2）若有相当多的分组使用 MPLS 就可缩短转发分组所需的时间，因而网络可扩展到较大的规模；（3）分组经受的时延最小，分组转发的速率不受路由表大小的影响。但网络结点无法处理没有 MPLS 标记的分组。

10-29 有相似之处，但具体的功能不同。

10-31 靠先进的编码，使得每一个码元带有多个比特的信息量。

附录 B

B-02 ALOHA 的容量为 $0.18 \times 2400 = 432$ b/s。终端速率为(5/3) b/s。

故允许终端数为 259 个。时隙 ALOHA：终端数加倍。

（1）帧长增加到 2.5 倍，以上答案应除以 2.5。

（2）终端速率减为原来的 2/3，以上答案应乘以 1.5。

（3）线路速率加倍，以上答案应加倍。

B-03 信道空闲的概率 $= \exp(-G) = \exp(-0.5) = 0.61$。

B-04 吞吐量 $= G\exp[(-G(2-1/k)]$

$k=1$ 相当于时隙 ALOHA。 $k\to\infty$，变成纯 ALOHA。

B-05 $G=1/160$

B-06 （1）$\exp(-2)$，（2）0.135×0.865^k，（3）每个帧平均发送 $\exp(G)=7.4$ 次。

B-07 $G=2.3$。 $S=0.23$。 已经过载了。

B-08 $S_1=0.032$, $S_2=0.288$, $S_3=S_4=0.072$。

B-09 写出 $S=f(G)$，对 G 求导数，令其为 0，解出 $G=1$。

B-10 $G=0.43$。$S=0.28$

附录 G　英文缩写词

（在每一行最后的括号中的数字是可找到解释该缩写词的章节编号）

AAL (ATM Adaptation Layer) ATM 适配层（5.6.2）

ABR (Available Bit Rate) 可用比特率（附录 D.3）

ACK (ACKnowledgement) 确认（3.2.3）

ADSL (Asymmetric Digital Subscriber Line) 非对称数字用户线（10.6.1）

AF PHB (Assured Forwarding Per-Hop Behavior) 确保转发每跳行为（10.4.4）

AH (Authentication Header) 鉴别首部（9.9）

ALOHA (Additive Link On-line HAwaii system) 一种随机接入系统（附录 B）

AN (Access Network) 接入网（1.3.2）

ANSI (American National Standards Institute) 美国国家标准协会（4.6.2）

AP (Access Point) 接入点（4.8.1）

API (Application Programming Interface) 应用编程接口（7.2.2）

APNIC (Asia Pacific Network Information Center) 亚太网络信息中心（6.2.1）

ARP (Address Resolution Protocol) 地址解析协议（6.2）

ARPA (Advanced Research Project Agency) 美国国防部远景研究规划局（高级研究计划署）（1.2.1）

ARQ (Automatic Repeat reQuest) 自动请求重发（3.2.5）

AS (autonomous system) 自治系统（6.5.1）

ASN.1 (Abstract Syntax Notation One) 抽象语法记法 1（8.8.5）

ATM (Asynchronous Transfer Mode) 异步传递方式（5.6）

ATU (Access Termination Unit) 接入端接单元（10.6.1）

ATU-C (Access Termination Unit Central Office) 端局接入端接单元（10.6.1）

ATU-R (Access Termination Unit Remote) 远端接入端接单元（10.6.1）

AUI (Attachment Unit Interface) 连接接口单元（4.2.2）

AWT (Abstract Window Toolkit) 抽象窗口工具箱（8.5.7）

BECN (Backward Explicit Congestion Notification) 反向显式拥塞通知（5.5.2）

BER (Basic Encoding Rule) 基本编码规则（8.8.5）

BGP (Border Gateway Protocol) 边界网关协议（6.5.1）

B-ISDN (Broadband-ISDN) 宽带综合业务数字网（附录 C.2）

BLAM (Binary Logarithmic Arbitration Method) 二进制对数仲裁方法（4.2.1）

BOOTP (BOOTstrap Protocol) 引导程序协议（8.6.1）

BSA (Basic Service Area) 基本服务区（4.8.1）

BSS (Basic Service Set) 基本服务集（4.8.1）

CA (Certification Authority) 认证中心（9.5）

CAC (Call Admission Control) 呼叫接纳控制（5.6.3）

CAC (Connection Admission Control) 连接准许控制（附录 D.2）

CAP (Carrierless Amplitude Phase) 无载波振幅相位调制（10.6.1）

CATV (Community Antenna TV, CAble TV) 有线电视（2.3.1）

CBR (Constant Bit Rate) 恒定比特率（附录 D.3）

CCIR (Consultative Committee, International Radio) 国际无线电咨询委员会（1.5.1）

CCITT (Consultative Committee, International Telegraph and Telephone) 国际电报电话咨询委员会（1.5.1）

CCS (Common-Channel Signaling) 共路信令（附录 C.1）

CDM (Code Division Multiplexing) 码分复用（2.5.3）

CDMA (Code Division Multiplex Access) 码分多址（2.5.3）

CDV (Cell Delay Variation) 信元时延偏差（附录 D.2）

CDVT (Cell Delay Variation Tolerance) 容许的信元时延偏差（附录 D.2）

CFI (Canonical Format Indicator) 规范格式指示符（4.5.2）

CGI (Common Gateway Interface) 通用网关接口（8.6.6）

CIDR (Classless InterDomain Routing) 无分类域间路由选择（6.3.3）

CIR (Committed Information Rate) 承诺的信息速率（5.5.1）

CLP (Cell Loss Priority) 信元丢失优先级（5.6.2）

CLR (Cell Loss Ratio) 信元丢失率（附录 D.2）

CMIP (Common Management Information Protocol) 公共管理信息协议（8.8.6）

CMOT (Common Management information service and protocol Over TCP/IP)
在 TCP/IP 上的公共管理信息服务与协议（8.8.6）

CNNIC (Network Information Center of China) 中国互联网络信息中心（8.2.2）

CPE (Customer Premises Equipment) 用户屋内设备（5.5.1）

CRC (Cyclic Redundancy Check) 循环冗余检验（3.2.3）

CS (Convergence Sublayer) 汇聚子层（5.6.2）

CS-ACELP (Conjugate-Structure Algebraic-Code-Excited Linear Prediction)
共轭结构代数码激励线性预测（声码器）（10.3.4）

CSMA/CD (Carrier Sense Multiple Access / Collision Detection)，载波监听多点接入/冲突检测（4.2.1）

CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance)，载波监听多点接入/冲突避免（4.7.3）

CSRC (Contributing SouRCe identifier) 参与源标识符（10.2.1）

CSU/DSU (Channel Service Unit/Data Service Unit) 信道服务单元/数据服务单元（2.7.1）

CTD (Cell Transfer Delay) 信元传送时延（附录 D.2）

CTS (Clear To Send) 允许发送（4.7.3）

DACS (Digital Access and Cross-connect System) 数字交接系统（2.6）

DARPA (Defense Advanced Research Project Agency) 美国国防部远景规划局（高级研究署）（1.2.1）

DAVIC (Digital Audio Visual Council) 数字视听协会（10.6.2）

DCE (Data Circuit-terminating Equipment) 数据电路端接设备（2.7.1）

DCF (Distributed Coordination Function) 分布协调功能（4.7.3）

DDoS (Distributed Denial of Service) 分布式拒绝服务（9.1.1）

DE (Discard Eligibility) 丢弃指示（5.5.2）

DES (Data Encryption Standard) 数据加密标准（9.2.2）

DHCP (Dynamic Host Configuration Protocol) 动态主机配置协议（8.6.2）

DiffServ (Differentiated Services) 区分服务（10.4.4）

DIFS (Distributed Coordination Function IFS) 分布协调功能帧间间隔（4.7.3）

DLCI (Data Link Connection Identifier) 数据链路连接标识符（5.5.1）

DMT (Discrete Multi-Tone) 离散多音（调制）（10.6.1）

DNS (Domain Name System) 域名系统（8.2.1）

DOCSIS (Data Over Cable Service Interface Specifications) 电缆数据服务接口规约（10.6.2）

DoS (Denial of Service) 拒绝服务（9.1.1）

DS (Distribution System) 分配系统（4.8.1）

DS (Differentiated Services) 区分服务（也写作 DiffServ）（10.4.4）

DSCP (Differentiated Services CodePoint) 区分服务码点（10.4.4）

DSL (Digital Subscriber Line) 数字用户线（10.6.1）

DSLAM (DSL Access Multiplexer) 数字用户线接入复用器（10.6.1）

DSSS (Direct Sequence Spread Spectrum) 直接序列扩频（4.8.2）

DTE (Data Terminal Equipment) 数据终端设备（2.7.1）

DVMRP (Distance Vector Multicast Routing Protocol) 距离向量多播路由选择协议（6.6.3）

DWDM (Dense WDM) 密集波分复用（2.5.2）

EBCDIC (Extended Binary-Coded Decimal Interchange Code) 扩充的二/十进制交换码（8.4.5）

EDFA (Erbium Doped Fiber Amplifier) 掺铒光纤放大器

EF PHB (Expedited Forwarding Per-Hop Behavior) 迅速转发每跳行为（10.4.4）

EGP (External Gateway Protocol) 外部网关协议（6.5.1）

EIA (Electronic Industries Association) 美国电子工业协会（2.3.1）

ESP (Encapsulating Security Payload) 封装安全有效载荷（9.9）

ESS (Extended Service Set) 扩展的服务集（4.8.1）

EUI (Extended Unique Identifier) 扩展的惟一标识符（4.3.1）

FCS (Frame Check Sequence) 帧检验序列（3.2.4）

FDDI (Fiber Distributed Data Interface) 光纤分布式数据接口（4.7.2）

FDM (Frequency Division Multiplexing) 频分复用（2.4.1）

FEC (Forwarding Equivalence Class) 转发等价类（10.5.2）

FEC (Forward Error Correction) 前向纠错（10.6.1）

FECN (Forward Explicit Congestion Notification) 前向显式拥塞通知（5.5.2）

FHSS (Frequency Hopping Spread Spectrum) 跳频扩频（4.8.2）

FIFO (First In First Out) 先进先出（7.4.6）

FOIRL (Fiber Optic Inter-Repeater Link) 转发器间的光纤链路（4.2.2）

FQ (Fair Queuing) 公平排队（10.4.2）

FR (Frame Relay) 帧中继（5.5.1）

FSK (Frequency Shift Keying) 移频键控（2.4.2）

FTP (File Transfer Protocol) 文件传送协议（8.3.1）

FTTB (Fiber To The Building) 光纤到大楼（10.6.3）

FTTC (Fiber To The Curb) 光纤到路边（10.6.3）

FTTH (Fiber To The Home) 光纤到家（10.6.3）

GCRA (Generic Cell Rate Algorithm) 一般信元速率算法（附录 D.2）

GIF (Graphics Interchange Format) 图形交换格式（8.5.5）

GII (Global Information Infrastructure) 全球信息基础结构，全球信息基础设施（1.1）

GFC (Generic Flow Control) 通用流量控制（5.6.2）

GSM (Group Special Mobile) 群组专用移动通信体制

HDLC (High-level Data Link Control) 高级数据链路控制（3.5.1）

HDSL (High speed DSL) 高速数字用户线（10.6.1）

HEC (Header Error Control) 首部差错控制（5.6.2）

HFC (Hybrid Fiber Coax) 光纤同轴混合（网）（10.6.2）

HIPPI (HIgh-Performance Parallel Interface) 高性能并行接口（4.7.3）

HSSG (High Speed Study Group) 高速研究组（4.6.3）

HTML (HyperText Markup Language) 超文本置标语言（8.6.1）

HTTP (HyperText Transfer Protocol) 超文本传送协议（8.6.1）

IAB (Internet Architecture Board) 因特网体系结构委员会（1.2.3）

IAC (Interpret As Command) 作为命令解释（8.4）

IAHC (Internet International Ad Hoc Committee) 因特网国际特别委员会（8.2.2）

IANA (Internet Assigned Numbers Authority) 因特网号码指派管理局（6.6.1）

ICANN (Internet Corporation for Assigned Names and Numbers) 因特网名字与号码指派公司（6.2.1）

ICMP (Internet Control Message Protocol) 因特网控制报文协议（6.2）

IDEA (International Data Encryption Algorithm) 国际数据加密算法（9.2.2）

IESG (Internet Engineering Steering Group) 因特网工程指导小组（1.2.3）

IETF (Internet Engineering Task Force) 因特网工程部（1.2.3）

IFS (InterFrame Space) 帧间间隔（4.7.3）

IGMP (Internet Group Management Protocol) 因特网组管理协议（6.6）

IGP (Interior Gateway Protocol) 内部网关协议（6.5.1）

IM (Instant Messaging) 即时传信（10.3.1）

IMAP (Internet Message Access Protocol) 因特网报文存取协议（8.5.4）

IMP (Interface Message Processor) 接口报文处理机（1.2.1）

IntServ (Integrated Services) 综合服务（10.4.3）

IP (Internet Protocol) 网际协议（1.5.3）

IPOA (IP Over ATM) IP 在 ATM 上运行（10.5.1）

IPsec (IP security) IP 安全协议（9.9）

IPX (Internet Packet Exchange) Novell 公司的一种连网协议（10.5.2）

IR (InfraRed) 红外技术（4.8.2）

IRTF (Internet Research Task Force) 因特网研究部（1.2.3）

ISDN (Integrated Services Digital Network) 综合业务数字网（附录 C.1）

ISO (International Organization for Standardization) 国际标准化组织（1.5.1）

ISOC (Internet Society) 因特网协会（1.2.3）

ISP (Internet Service Provider) 因特网服务提供者（1.2.2）

ITU (International Telecommunication Union) 国际电信联盟（1.5.1）

ITU-T (ITU Telecommunication Standardization Sector) 国际电信联盟电信标准化部门（1.5.1）

JPEG (Joint Photographic Expert Group) 联合图像专家组标准（8.5.5）

KDC (Key Distribution Center) 密钥分配中心（9.3.1）（9.5）

LAN (Local Area Network) 局域网（1.3.2）

LANE (LAN Emulation) 局域网仿真（10.5.1）

LAPB (Link Access Procedure Balanced) 链路接入规程(平衡型)（3.5.1）

LCP (Link Control Protocol) 链路控制协议（3.6.1）

LDP (Label Distribution Protocol) 标记分配协议（10.5.2）

LLC (Logical Link Control) 逻辑链路控制（4.2.1）

LSP (Label Switched Path) 标记交换路径（10.5.2）

LSR (Label Switching Router) 标记交换路由器（10.5.2）

MAC (Medium Access Control) 媒体接入控制（4.2.1）

MAN (Metropolitan Area Network) 城域网（1.3.2）

MAU (Medium Attachment Unit) 媒体连接单元（4.2.2）

MBONE (Multicast Backbone On the InterNEt) 多播主干网（6.6.1）

MBS (Maximum Burst Size) 最大突发长度（附录 D.2）

MCR (Minimum Cell Rate) 最小信元速率（附录 D.2）

MCU (Multipoint Control Unit) 多点控制单元（10.3.2）

MD (Message Digest) 报文摘要（9.4）

MDI (Medium Dependent Interface) 媒体相关接口（4.2.2）

MIB (Management Information Base) 管理信息库（7.4.8）

MIME (Multipurpose Internet Mail Extensions) 通用因特网邮件扩充（8.5.1）

MIPS (Million Instructions Per Second) 百万指令每秒（9.6.1）

MOTIF (Message Oriented Text Interchange System) 面向报文的电文交换系统（8.5.1）

MPEG (Motion Picture Experts Group) 活动图像专家组标准（8.5.5）

MPOA (MultiProtocol Over ATM) 多协议在 ATM 上运行（10.5.1）

MPLS (MultiProtocol Label Switching) 多协议标记交换（10.5.1）

MRU (Maximum Receive Unit) 最大接收单元（3.6.1）

MSS (Maximum Segment Size) 最长报文段（7.4.2）

MTU (Maximum Transfer Unit) 最大传送单元（6.2.4）

NAK (Negative AcKnowlegement) 否认帧（3.2.3）

NAP (Network Access Point) 网络接入点（1.2.2）

N-ISDN (Narrowband-ISDN) 窄带综合业务数字网（附录 C.1）

NAT (Network Address Translation) 网络地址转换（6.7.2）

NAV (Network Allocation Vector) 网络分配向量（4.7.3）

NCP (Network Control Protocol) 网络控制协议（3.6.1）

NFS (Network File System) 网络文件系统（8.3.1）

NIC (Network Interface Card) 网络接口卡、网卡（4.2.1）

NII (National Information Infrastructure) 国家信息基础结构，国家信息基础设施（1.1）

NLA ID (Next-Level Aggregation Identifier) 下一级聚合标识符（6.8.4）

NLRI (Network Layer Reachability Information) 网络层可达性信息（6.5.4）

NNI (Network-Node Interface) 网络结点接口（5.6.2）

NSF (National Science Foundation) （美国）国家科学基金会（1.2.2）

NVT (Network Virtual Terminal) 网络虚拟终端（8.4）

OAM (Operations, Administration and Maintenance) （网络）运行、处理（或管理）和维护（附录 D.1）

OC (Optical Carrier) 光载波（2.6）

ODN (Optical Distribution Node) 光分配结点（10.6.2）

OSI/RM (Open Systems Interconnection Reference Model) 开放系统互连基本参考模型（1.5.1）

OSI/RM (Open Systems Interconnection Reference Model) 开放系统互连基本参考模型（1.5.1）

OSPF (Open Shortest Path First) 开放最短通路优先（6.5.3）

OUI (Organizationally Unique Identifier) 机构惟一标识符（4.3.1）

PCF (Point Coordination Function) 点协调功能（4.7.3）

PCM (Pulse Code Modulation) 脉码调制（2.4.3）

PCR (Peak Cell Rate) 峰值信元速率（附录 D.2）

PDU (Protocol Data Unit) 协议数据单元（1.5.3）

PEM (Privacy Enhanced Mail) 因特网的正式邮件加密标准（9.6.2）

PGP (Pretty Good Privacy) 一种电子邮件加密技术（9.6.1）

PHB (Per-Hop Behavior) 每跳行为（10.4.4）

PIFS (P oint Coordination Function IFS) 点协调功能帧间间隔（4.7.3）

PING (Packet InterNet Groper) 因特网包（分组）探索程序（6.2.3）

PMD (Physical Medium Dependent) 物理媒体相关（子层）（5.6.2）

PoP (Point of Presence) 汇接点（10.5.1）

POP (Post Office Protocol) 邮局协议（8.5.1）

POTS (Plain Old Telephone Service) 传统电话（10.6.1）

PPP (Point-to-Point Protocol) 点对点协议（3.6）

PS (POTS Splitter) 电话分路器（10.6.1）

PSK (Phase Shift Keying) 移相键控（2.4.2）

PT (Payload Type) 有效载荷类型（5.6.2）

PTE (Path Terminating Element) 路径端接设备（2.6）

PTI (Payload Type Indicator) 有效载荷类型指示（5.6.2）

PVC (Permanent Virtual Circuit) 永久虚电路（5.6.2）

QAM (Quadrature Amplitude Modulation) 正交幅度调制（2.4.2）

QoS (Quality of Service) 服务质量（5.1.2）

RAC (Registration Authority Committee) 注册管理委员会（4.3.1）

RADSL (Rate-Adaptive DSL) 速率自适应数字用户线（10.6.1）

RAN (Residential Access Network) 居民接入网（10.6）

RARP (Reverse Address Resolution Protocol) 逆地址解析协议（6.2）

RBB (Residential BroadBand network) 居民宽带网（10.6）

RED (Random Early Detection) 随机早期检测（7.4.6）

RED (Random Early Discard, Random Early Drop) 随机早期丢弃（7.4.6）

RFC (Request For Comments) 请求评论（1.2.3）

RG (Research Group) 研究组（1.2.3）

RIP (Routing Information Protocol) 路由信息协议（6.5.2）

RSA (Rivest, Shamir and Adleman) 一种公开密钥算法的名称，用三个人名合成（9.3.2）

RSVP (Resource reSerVation Protocol) 资源预留协议（10.4.3）

RTCP (Real-time Transfer Control Protocol) 实时传送控制协议（10.2.2）
RTO (RetransmissionTime-Out) 超时重传时间（7.4.5）
RTP (Real-time Transfer Protocol) 实时传送协议（10.2.1）
RTS (Request To Send) 请求发送（4.7.3）
RTSP (Real-Time Streaming Protocol) 实时流式协议（10.2.3）
RTT (Round-Trip Time) 往返时延（1.4.3）（7.4.5）
SA (Security Association) 安全关联（9.9）
SAP (Service Access Point) 服务访问点（1.5.3）
SAR (Segmentation And Reassembly) 拆装（子层）（5.6.2）
SCR (Sustainable Cell Rate) 持续信元速率（附录 D.2）
SDH (Synchronous Digital Hierarchy) 同步数字系列（2.6）
SDLC (Synchronous Data Link Control) 面向比特的规程（3.5.1）
SDP (Session Description Protocol) 会话描述协议（10.3.3）
SDSL (Single-line DSL) 1 对线的数字用户线（10.6.1）
SDU (Service Data Unit) 服务数据单元（1.5.4）
SEAL (Simple Efficient Adaptation Layer) 简单高效适配层（5.6.4）
SET (Secure Electronic Transaction) 安全电子交易（9.8.2）
SHA (Secure Hash Algorithm) 安全散列算法（9.4）
SIFS (Short IFS) 短帧间间隔（4.7.3）
SIP (Session Initiation Protocol) 会话发起协议（10.3.3）
SLA (Site Level Aggregation) 地点级聚合（6.8.4）
SLA (Service Level Agreement) 服务等级协定（10.4.4）
SLIP (Serial Line Internet Protocol) 串行线路因特网协议（3.6.1）
SMI (Structure of Management Information) 管理信息结构（8.8.5）
SMTP (Simple Mail Transfer Protocol) 简单邮件传送协议（8.5.1）
SNA (System Network Architecture) 系统网络体系结构（1.5.1）
SNMP (Simple Network Management Protocol) 简单网络管理协议（8.8）
SONET (Synchronous Optical Network) 同步光纤网（2.6）
SPI (Security Parameter Index) 安全参数索引（9.9）
SSL (Secure Socket Layer) 安全插口层（9.8.1）
SSRC (Synchronous SouRCe identifier) 同步源标识符（10.2.1）
STDM (Statistic TDM) 统计时分复用（2.5.1）
STM (Synchronous Transfer Module) 同步传递模块（2.6）
STP (Shielded Twisted Pair) 屏蔽双绞线（2.3.1）
STS (Synchronous Transport Signal) 同步传送信号（2.6）
SVC (Switched Virtual Connection) 交换虚连接（5.6.4）
TC (Transmission Convergence) 传输汇聚（子层）（5.6.2）
TCB (Transmission Control Block) 传输控制程序块（7.4.8）
TCP (Transmission Control Protocol) 传输控制协议（1.5.3）（7.2）
TDM (Time Division Multiplexing) 时分复用（2.4.3）
TFTP (Trivial File Transfer Protocol) 简单文件传送协议（8.3.3）

TLA ID (Top-Level Aggregation IDentifier) 顶级聚合标识符（6.8.4）

TLD (Top Level Domain) 顶级域名（8.2.2）

TLS (Transport Layer Security) 运输层安全协议（9.8.1）

TOS (Type Of Service) 服务类型（6.2.4）

TPDU (Transport Protocol Data Unit) 运输协议数据单元（7.2.1）

TSS (Telecommunication Standardization Sector) 电信标准化部门（1.5.1）

TTL (Time To Live) 生存时间或寿命（6.2.4）

UA (User Agent) 用户代理（8.5.1）

UBR (Unspecified Bit Rate) 不指明比特率（附录 D.3）

UDP (User Datagram Protocol) 用户数据报协议（1.5.3）（7.2）

UIB (User Interface Box) 用户接口盒(10.6.2)

UNI (User-to-Network Interface) 用户网络接口（5.5.1）

UPC (Usage Parameter Control) 使用参数控制（附录 D.2）

URI (Universal Resource Identifier) 通用资源标识符（8.6.2）

URL (Uniform Resource Locator) 统一资源定位符（8.6.2）

URN (Uniform Resource Name) 统一资源名字（8.6.2）

UTP (Unshielded Twisted Pair) 无屏蔽双绞线（2.3.1）

VBR (Variable Bit Rate) 可变比特率（附录 D.3）

VC (Virtual Circuit) 虚电路（5.1.2）

VCI (Virtual Channel Identifier) 虚通路标识符（5.6.2）

VDSL (Very high speed DSL) 甚高速数字用户线（10.6.1）

VID (VLAN ID) VLAN 标识符（4.5.2）

VLAN (Virtual LAN) 虚拟局域网（4.5）

VLSM (Variable Length Subnet Mask) 变长子网掩码（6.3.3）

VoD (Video on Demand) 视频点播（10.6.2）

VoIP (Voice over IP) 在 IP 上的话音（10.3.1）

VON (Voice On the Net) 在因特网上的话音（10.3.1）

VP (Virtual Path) 虚通道（5.6.2）

VPI (Virtual Path Identifier) 虚通道标识符（5.6.2）

VPN (Virtual Private Network) 虚拟专用网（6.7.1）

VSAT (Very Small Aperture Terminal) 甚小孔径地球站（2.3.2）

WAN (Wide Area Network) 广域网（1.3.2）

WDM (Wavelength Division Multiplexing) 波分复用（2.5.2）

WFQ (Weighted Fair Queuing) 加权公平排队（10.4.2）

WG (Working Group) 工作组（1.2.3）

WWW (World Wide Web) 万维网（1.2.2）（8.6）

附录H　参考文献与网址

1．值得进一步深入阅读的计算机网络教材

[COME00] Comer, D., *Internetworking with TCP/IP*, Vol.1. 4th Ed., Prentice-Hall, 2000. 人民邮电出版社影印版。中译本：电子工业出版社，2001.

[COME01] Comer, D., *Computer Networks and Internets*, Prentice-Hall, 2001. 清华大学出版社影印版。旧版的中译本：电子工业出版社，1998.

[FORO00] Forouzan, B. A., *TCP/IP Protocol Suite*, McGraw-Hill, 2000. 中译本：清华大学出版社，2000.

[KURO01] Kurose, J. F. and Ross, K. W., *Computer Networking, A Top-Down Approach Featuring the Internet*, Pearson Education, 2001. 高等教育出版社影印版。

[PERL00] Perlman, R., *Interconnections: Bridges and Routers*, 2nd Ed. Addison-Wesley, 2000. 机械工业出版社影印版。有中译本：机械工业出版社，2000.

[PETE00] Peterson, L. L. and Davie, B. S., *Computer Networks, A Systems Approach*, 2nd Ed., Morgan Kaufmann, 2000. 机械工业出版社影印版。

[STEV94] Stevens, W. R., *TCP/IP Illustrated*, Vol.1. Addison-Wesley, 1994. 机械工业出版社影印版。中译本：机械工业出版社，2000.

[STAL00] Stallings, W., *Data and Computer Communications*, 6th Ed., Prentice-Hall, 2000. 高等教育出版社影印版。中译本：电子工业出版社，2001.

[TANE96] Tanenbaum, A. S., *Computer Networks*, 3rd Ed., Prentice-Hall, 1996. 清华大学出版社影印版。

2．参考文献

[BELL86] Bell, P. R., et al., "Review of Point-to-Point Network Routing Algorithms," *IEEE Commun. Magazine*, Vol.24, No.1, pp.34-38, Jan. 1986.

[BIHA90] Biham, E. and Shamir, A., "Differential Cryptanalysis of DES-like Cryptosystem," *CRYPTO'90*.

[BURT72] Burton, H. O., et al., "Error and Error Control", *Proc. IEEE*, Vol.60, No.11, pp.1293-1301, Nov. 1972.

[COLL01] Collins, D., *Carrier Grade Voice over IP*, McGraw-Hill, 2001.人民邮电出版社影印版。

[COMM90] *IEEE Commun. Magazine*, Special Issue on SONET/SDH, Vol.28, No.8, Aug. 1990.

[CUNN99] Cunningham, D. G. and Lane, W. G., "*Gigabit Ethernet Networking*", Macmillan Technical Publishing, 1999, 清华大学出版社影印。

[DAVI86] Davies, D. W., "The Origins of Packet Switching," *Computer Network Usage: Recent Experiences*, Eds. L. Csaba et al., North-Holland, 1986, pp.1-13.

[DENN82] Denning, D. E., *Cryptograph and Data Security*, Addison-Wesley, 1982.

[DIFF76] Diffie, W. and Hellman, M., "New Directions in Cryptography", *IEEE Trans.*, Vol.IT-22, No.6, pp.644-654, Nov. 1976.

[HARN98] Harnedy, S., *Total SNMP: Exploring the Simple Network Management Protocol*, 2nd Ed., Prentice

Hall, 1998.有中译本，电子工业出版社出版，1999.

[HUIT95] Huitema, C., *Routing in the Internet*, Prentice Hall, 1995.

[KAST98] Kastas, T. J., et. al., "Real-Time Voice Over Packet-Switched Networks," *IEEE Network*, Vol.12, No.1, pp.18-27, Jan/Feb 1998.

[KWOK98] Kwok, T., *ATM: The New Paradigm for Internet, Intranet, and Residential Broadband Services and Applications*, Prentice-Hall, 1998.

[LAI90] Lai, X., and Massey, J.: "A Proposal for a New Block Encryption Standard." *Advances in Cryptology —Eurocrypt '90 Proceedings*, New York: Springer-Verlag, pp.389-404, 1990.

[LEUT94] Leutwyler, K., "Superhack." *Scientific American*, July 1994.

[METC76] Metcalfe, R. M., et al., "Ethernet: Distributed Packet Switching for Local Computer Networks," *Commun. ACM*, Vol.19, No.7, pp.395-404, July 1976.

[MINGCI93] 电子学名词，科学出版社，1994 年 4 月。

[MINGCI94] 计算机科学技术名词，科学出版社，1994.

[MOY98] Moy, J. T., *OSPF: Anatomy of an Internet Routing Protocol*, Addison-Wesley, 1998.

[NAUG99] Naugle, M., *Network Protocols*, MaGraw-Hill, 1999. 有影印版，人民邮电出版社。

[NETW88] *IEEE Network*, Special Issue on Bridges, Vol.2, No.1, Jan. 1988.

[RIVE78] Rivest, R. L., Shamir, A. and Adleman, L., "A Method for Obtaining Digital Signature and Public Key Cryptosystems," *Commun. ACM*, Vol.21, No.2, pp.120-126, Feb. 1978.

[ROBE82] Roberts, L. G., "Packet Switching Economics," *J. Telecommun. Networks*, Vol.1, No.2, pp.213-218, Fall 1982.

[SCHA97] Schaller, R. R., "Moore's Law: Past, Present, and Future," *IEEE Spectrum*, Vol.34, No.6, pp.53-59, June 1997.

[SHAN49] Shannon, C. E., "Communication Theory of Secrecy Systems," *Bell Syst. Tech. J.*, Vol.28, pp.656-715, Oct. 1949.

[SHOC78] Shoch, J. F., "Inter-network Naming, Addressing, and Routing," *Compcon*, pp.72-79, Fall 1978.

[STAL96] Stallings, W., *SNMP, SNMPv2, and RMON*, Addison-Wesley, 1996.

[STAL01] Stallings, W., *High-Speed Networks and Internets: Performance and Quality of Service*, 2nd Ed., Macmillan, 1998. 有中译本，电子工业出版社出版，2002.

[SUBR01] Subramanian, M., *Network Management, Principle and Practice*, Pearson Education, 2001.有影印版，高等教育出版社。

[VITER95] Viterbi, A. J., CDMA Principles of Spread Spectrum Communication. Reading, MA: Addison-Wesley, 1995.

[ZHAN93] Zhang Lixia, "RSVP A New Resource ReServation Protocol," *IEEE Network Magazine*, Vol.7, pp.8-18, Sept/Oct. 1993.

3. 一些有参考价值的网址

[W-10GE] http://grouper.ieee.org/groups/802/3/10G_study/
http://www.10gea.org/

[W-ADSL] http://www.adsl.com/

[W-ATM] http://www.atmforum.com

[W-AVT] http://www.ietf.org/html.charters/avt-charter.html

[W-CableLabs] http://www.cablelabs.com

[W-CGI] http://Web.Golux.Com/coar/cgi/draft-coar-cgi-v11-00.html

[W-CNIC] http://www.cnnic.cn/

[W-DiffServ] http://www.ietf.org/html.charters/diffserv-charter.html

[W-HTML] http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html

http://www.w3.org/MarkUp/

[W-HTTP] http://www.w3.org/Protocols/HTTP-NG/Activity.html

[W-ICANN] http://www.icann.org/

[W-IEEE802] http://standards.ieee.org/getieee802/

[W-IntServ] http://www.ietf.org/html.charters/intserv-charter.html

[W-ISOC] http://www.isoc.org/ 这是因特网协会的主页，由此可链接到有关因特网的各种信息。

[W-MANET] http://www.ietf.org/html.charter/manet-charter.html

[W-MCAST] White Paper—The Evolution of Multicast, http://www.stardust.com

[W-MMUSIC] http://www.ietf.org/html.charters/mmusic-charter.html

[W-MPLS] http://www.ietf.org/html.charters/mpls-charter.html

[W-NAT] http://www.ietf.org/html.charters/nat-charter.html

[W-NewTLD] http://www.icann.org/tlds/

[W-NGTRANS] http://www.ietf/org/html.charters/ipngwg-charter.html

[W-PGP] http://www.pgpi.com/download/

[W-PGPATT] http://axion.physics.ubc.ca/pgp-attack.html

[W-RFC] http://www.ietf.org/rfc.html

http://www.ietf.org/rfc/rfcNNNN.txt 这里 NNNN 是所要下载的 RFC 编号

[W-SGML] http://www.w3.org/hypertext/WWW/MarkUp/SGML/ SGML

[W-SIP] http://www.ietf.org/html.charters/sip-charter.html

[W-SNMPv3] http://www ietf.org/html.charters/snmpv3-charter.html

[W-VoIP] http://www.ietf.org/html.charters/iptel-charter.html

[W-802.11] http://grouper.ieee.org/groups/802/11/index.html