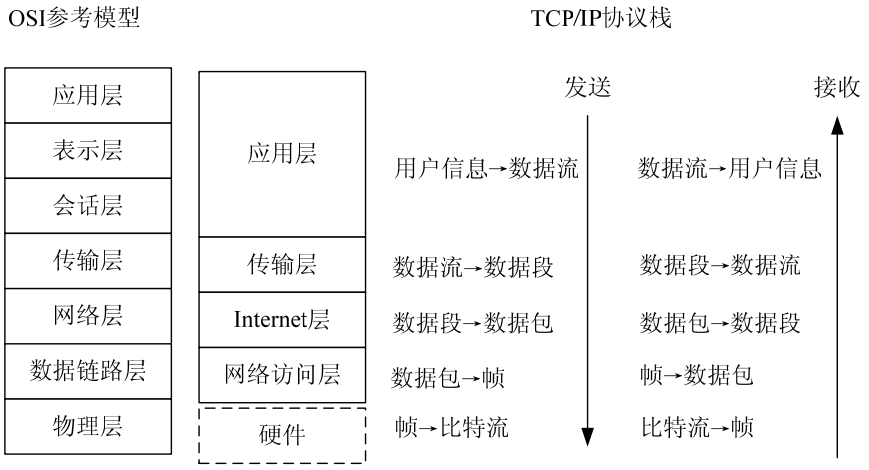
前言：
本文摘自人民邮电出版社《TCP/IP 管理及网络互联》，书号：7-115-12433-7，王群 王琳琳编著
http://www.china-pub.com/computers/common/info.asp?id=20412
http://www.china-pub.com/computers/common/info.asp?id=19332
本文仅用于南京工业大学思科网络学院内部学习交流，严禁抄袭！

1、通信协议

通信协议就是通信标准。它能够使不同硬件结构的及其进行通信、能够使用各种网络硬件、适用于各种不同的应用程序、适用于各种计算机操作系统。通信协议隐藏了通信的底层细节，因此我们可以撇开任何厂家的硬件来讨论通信问题。
TCP/IP(传输控制协议/网际协议，Transmission Control Protocol/Internet Protocol)是发展至今最成功的通信协议。刚开始时 TCP/IP 是美国国防部高等研究计划局 (DARPA) 开发研究计划的一部份，其原始目的是为 DARPA 提供通讯，现在它已被广泛应用于全球最大的开放式网络系统 Internet 之上，使全球数百万电脑得以互通联系。
TCP/IP 的成功和人们对 Internet 的广泛使用，TCP/IP 技术成为互连网络协议的市场标准，几乎所有厂商的设备都支持 TCP/IP。但是 TCP/IP 并不为某个厂商、专业协会或标准团体所拥有。
有关 TCP/IP 协议标准、Internet 的协议、协议修订的文档都出现在 Internet RFC 中，RFC 覆盖很多概念和细节，有些是标准，有些是关于新协议的建议。这一系列的技术告都可以从 Internet 上免费获得，其下载地址为：http://www.isi.edu/in-notes。

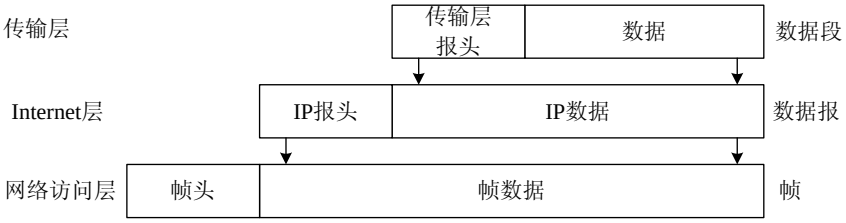
2、TCP/IP 协议栈与 OSI 参考模型的比较

TCP/IP 协议栈主要分成四层：应用层（Application Layer）、传输层（Transport Layer）、Internet 层（Internet Layer）、网络访问层（Network Interface Layer）。这个分层模型并非出自哪个标准委员会，而是来自一些对 TCP/IP 协议栈的研究工作。这四层大致对应 OSI 参考模型的七层。但是与 OSI 模型不同的是，TCP/IP 协议栈更加侧重于互连设备间的数据传送，而不是严格的功能层次的划分。两者的对比如下图所示：



在具体讲述 TCP/IP 协议栈之前，我们先从总体上讨论一下数据封装的过程：
①用户调用应用程序通过 TCP/IP 来访问相应的服务。应用层负责将这些应用程序信息转换成数据流，交给传输层处理。
②传输层的基本任务是提供端到端（End to End）的通信（即应用程序之间的通信）。传输层的协议负责系统地管理信息的流动，提供可靠或不可靠的传输服务。
在发送方，传输层将应用层提供的数据流分段（或称分组，即将数据流划分成小段），并将这些数据段加上标识，包括由哪个应用程序发出、由哪个应用程序处理、使用什么传输层协议、校验和、报文长度等等，这种标识称为传输层报文头，例如 TCP 报文头、UDP 报文头。
在接收方，传输层拨掉传输层报文头，利用报文头中的校验和来检验数据在传输过程中是否出错，以一定的顺序将数据段重新组装成数据流交给应用程序处理。
③Internet 层负责处理主机之间的通信。该层还要决定如何交付数据包，是交给网关（路由器），还是交给适当的本地端口。
在发送方，Internet 层将传输层提供的数据段封装到数据包（数据包）中，填入 IP 报头（包括源 IP 地址、目标 IP 地址、使用什么协议、校验和等等）。
在接收方，Internet 层通过读取 IP 头中的信息决定如何处理数据包。如果是路由器收到数据包，它通过校验和检验其有效性，决定是作本地处理还是转发该数据包；如果是目标主机收到该数据包，通过检验后，

它会去掉 IP 报头，交给传输层处理。
④网络访问层负责把 Internet 层提供的数据包封装成帧，帧头中包含源 MAC 地址、目标 MAC 地址、使用何种链路封装协议（如 HDLC、PPP）等信息，然后把帧通过选定的网络接口发送出去。
在接收方，该层读取帧头中的信息，如果是发给自己的，它拆开帧头，将数据包交给 Internet 层处理；如果不是发给自己的则丢弃该帧。该层还包括一些网络设备的驱动程序。
⑤最后硬件（网络设备）把帧转换成比特流通过传输介质将信息发送出去。
下图可以很好的描述数据封装的过程：



3、TCP/IP 协议栈的应用层

TCP/IP 应用层对应了 OSI 参考模型的上三层，它包括了一些服务。这些服务是和端用户相关的认证、数据处理及压缩，应用层还要告诉传输层哪个数据流是由哪个应用程序发出的。应用层主要包括以下 Internet 协议：

文件传输类：如 HTTP（超文本传输协议）、FTP（文件传输协议）、TFTP（简单文件传输协议）
远程登陆类：如 Telnet
电子邮件类：如 SMTP（简单邮件传输协议）
网络管理类：如 SNMP（简单网络管理协议）

域名解析类：如 DNS（域名服务）

HTTP

HTTP（Hypertext Transfer Protocol，超文本链接协议）是一个应用层的面向对象的协议，它适用于分布式超媒体信息系统。
WWW（World Wide Web，WWW，也简称为 Web）服务器使用的主要协议就是 HTTP。由于 HTTP 支持的服务不限于 WWW，还可以是其它服务，因此 HTTP 允许用户在统一的界面下，采用不同的协议访问不同的服务，如 FTP、SMTP、NNTP 等。

FTP

FTP（File Transfer Protocol，文件传输协议）是一个用于简化 IP 网络上系统之间文件传送的协议。采用 FTP 协议可使用户高效地从 Internet 上的 FTP 服务器下载大信息量的数据文件，以达到资源共享和传递信息的目的。
一个 FTP 站点可以是公用的，私有的，或者两者兼有之。我们可以为 FTP 帐号定义权限，让它可以访问整个 FTP 服务的目录结构，或者只是特定的区域。
FTP 服务器可以设置为语序任何人连接和传输文件，这种访问方式被称为匿名访问。当我们使用匿名方式登录到 FTP 站点时，系统默认使用“anonymous”作为用户名，“guest”或某个 e-mail 地址作为密码。匿名 FTP 经常用于发布大量的公用领域或共享软件。

TFTP

TFTP（Trivial File Transfer Protocol，简单文件传输协议）是基于 UDP 的应用。TFTP 在设计时是用于小文件传输的，它对内存和处理器的要求很低，速度快。但是 TFTP 不具备 FTP 的许多功能，它只能从文件服务器上获得或写入文件，不能列出目录，不进行认证，所以它没有建立连接的过程及错误恢复的功能，适用范围也不像 FTP 那么广泛。
一个最常见的 TFTP 的应用例子就是使用 TFTP 服务器来备份或恢复 Cisco 路由器、Catalyst 交换机的 IOS 镜像和配置文件。

DNS

域名管理系统 DNS（Domain Name System）是一台域名解析服务器。在互联网中我们通常用一些域名（例如 www.163.com、www.cisco.com、www.njupt.edu.cn）代替难记的 IP 地址（例如 202.106.168.104、198.133.219.25、202.119.230.10）以定位计算机和服务。DNS 服务器中包含了域名和相应的 IP 地址的映射。因此 DNS 的作用是：把域名转换成为网络可以识别的 IP 地址。
例如：

```
C:\>nslookup
Default Server:  dns.yc.js.cn
Address:  202.102.11.141

>www.cisco.com
Server:  dns.yc.js.cn
```

```
Address:  202.102.11.141

Non-authoritative answer:
Name:  www.cisco.com
Address:192.133.219.25
```

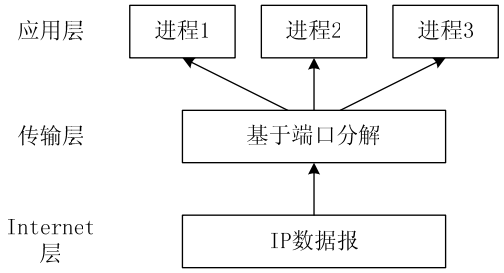
4、TCP/IP 协议栈的传输层

在进行本节的讲述之前，我们先提出以下几个问题：
①当我们在 Windows XP 中同时运行多个网络应用程序时，每个应用程序都会产生自己的数据流，传输层是用什么方法区分不同应用程序的数据流呢？
②在数据流被分段（分组）以后，传输层依靠什么来重新组装这些数据段呢？
③如果某个数据段在传输过程中丢失了或重复了，可靠的传输协议依据什么去要求重传这些数据或丢弃多余的数据呢？
带着这些问题，我们来讨论传输层所提供的服务。

传输层的主要功能是：分割并重新组装上层提供的数据流，为数据流提供端对端的传输服务。
在 TCP/IP 协议中，有两个传输层协议：传输控制协议（TCP）和用户数据包协议（UDP）。
TCP 是一个可靠的面向连接的协议，UDP 是不可靠的或无连接的协议。这种面向连接和无连接的通信方式的对比就像打电话和发明信片一样。打电话的双方在正式通话之前都会说“喂”，确定对方在线以后才开始通话，会话结束时都要说“再见”，然后才挂下电话。而发明信片却没有这种确信机制，发就发出去了，不管对方有没有收到。

端口号

每个应用程序都会产生自己的数据流，这些数据流可以把目标主机上相应的服务程序看作自己的目的地。对于传输层来说，它只需要知道目标主机上的哪个服务程序来响应这个应用程序，而不需要知道这个服务程序具体是干什么的。因此，传输层使用一个抽象的端口号来标识这些应用程序和服务程序。
端口号用来跟踪网络间同时发生的不同会话。TCP 和 UDP 可以同时接收多个应用程序送来的数据流，用端口号来标识他们，然后把他们送给 Internet 层处理；同时 TCP 和 UDP 接受来自 Internet 层送来的数据包，用端口号区分他们，然后送给适当的应用程序处理。 这是多路复用技术的体现，它可以确保正确的用户程序收到正确的数据（如下图）。因此，每个应用程序在发送数据前都会与操作系统进行协商，获得相应的源端口号和目标端口号。



在主机发送应用程序的数据之前，都必须确认端口号，如何分配这些端口号呢？
有两种情况：第一，使用中央管理机构统一分配的端口号。应用程序的开发者们都默认在 RFC1700 中定义的特殊端口号，在进行软件设计时，都要遵从 RFC1700 中定义的规则，不能随便使用已定义的端口号。例如任何 telnet 应用中的会话都要使用标准端口号 23。
第二，使用动态绑定。如果一个应用程序的会话没有涉及到特殊的端口号，那么系统将在一个特定的取值范围内随机地为应用程序分配一个端口号。在应用程序进行通信之前，如果不知道对方的端口号，就必须发送请求以获得对方的端口号。
TCP/IP 的设计者们采用一种混合方式实现端口地址的管理，终端系统利用源系统端口号来选择合适的应用程序，但是源系统的源端口号由源系统动态分配。

目前端口号的分配情况大致如下：

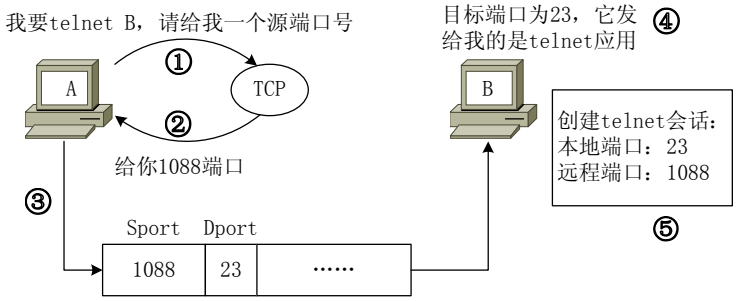
- 小于 255 的端口号用于公共应用
- 255～1023 是特定供应商应用程序的注册端口号
- 高于 1023 的端口号未作规定

下表描述了端口号的一些使用情况

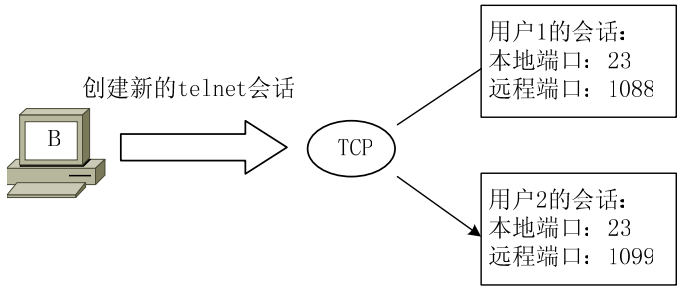
常用的应用层协议或应用程序	端口号	
	UDP	TCP

FTP		21、20
Telnet		23
SMTP		25
TFTP	69	
SNMP	161	
DNS	53	53
HTTP		80

下面举例说明端口号的使用过程：
主机 A 要 telnet 到主机 B，主机 A 首先向 TCP 请求一个可用端口，TCP 分配端口号为 1088 的端口给它，主机 A 将目标端口号置为 23。A 和 B 通信后，B 看到 A 过来的端口号为 23，就知道这是一个 telnet 应用，它会为它创建一个 telnet 会话。如下图所示：



假如同一系统中有多个 telnet 用户，会发生什么情况呢？当 A 上第二个 telnet 用户向 TCP 发出请求时，TCP 会选出另外一个可使用的端口号，如 1099，给第二个 telnet 用户。主机 B 上便会创建第二个 telnet 会话。如下图：



所以在同一 IP 地址上具有不同端口号的两个连接是不同的。IP 地址和端口号被用来唯一的确定数据连接的途径。

UDP

下图显示了 UDP 数据域头格式

源端口	目标端口
报文长度	校验和
数据	

UDP 头有 8 的字节，下面对它的头域进行一下简单的描述：

名称	描述
源端口（Source Port）	调用的端口号
目的端口（Destination Port）	被调用的端口号
头长度（HLEN）	记录 UDP 数据包中的八位组数目，它包括 UDP 头及 UDP 数据的长度，最小值为 8（UDP 数据部分长度为 0 时）
校验和（Checksum）	头标和数据域计算的校验和，这一项是可选的，为的是在高可靠性的网络上尽量减少开销
数据（Data）	上层协议的数据

UDP 为应用程序提供的是一种不可靠的、无连接的分组交付服务，UDP 报文可能出现丢失、重复、时延、乱

序、连接失效等问题。但是正是由于它不提供这种可靠性，所以它的开销很小。换句话说：UDP 提供了一种在高效可靠的网络上传输数据而不用消耗不必要的网络资源和处理时间的通信方式。使用 UDP 的协议包括 TFTP、SNMP、NFS、DNS、DHCP。UDP 很适合这种客户机向服务器发送简单服务请求的环境，因为这种服务的开销本来就很小，如果在开始或者结束时加入类似 TCP 三次握手的过程，网络的实际利用率将变得很低。

UDP 还可以用于操作信息的登陆。例如向日志服务器 syslog 发送日志信息，采用 UDP 不会导致多台设备向一台服务器发送日志信息而引起过载。

UDP 依靠上层协议提供可靠性，包括处理报文的丢失、重复、时延、乱序、连接失效等问题。如 real 的流格式媒体就是使用应用层协议来保证数据的正确传输。

TCP

在上一节中，我们提到：UDP 为应用程序提供的是一种不可靠的、无连接的分组交付服务。当网络硬件失效或者负担太重时，数据段可能会产生的丢失、重复、时延、乱序的现象，这些都会导致通信不正常。如果让应用程序来担负差错检测和恢复的工作，将给程序员带来很多复杂的工作，所以使用独立的通信协议来保证通信的可靠性是非常必要的。

传输控制协议 TCP 是在 RFC793 中定义的，它是一个面向连接的可靠的通信协议。

总的说来，TCP 主要提供以下服务：

●面向连接的虚电路：这有些和打电话相似，在开始传输之前，通信双方要进行三次握手建立连接的操作，以保证连接的可靠性。在传输过程中，通信双方的协议模块继续进行通信，以确保数据正确到达（例如接收方会用 ACK 应答发送方的报文段，发送方对未被应答的报文段提供重传）。如果在传输过程中通信失败了（例如传输路径上的某个网络接口失效），通信双方都会收到错误报告。在通信结束时，通信双方会使用改进的三次握手来关闭连接。

●面向流：当通信双方传输大量数据时，TCP 将数据流看作可分为字节的流，进行分段（分组），接收方将收到的报文段按原有顺序复原。

●流量控制，避免拥塞：为了提高传输效率和减少网络通信量（协议之间的通信），TCP 会尽量一次传输足够多的数据。

●多路复用技术：有端口号来实现。

●全双工连接：TCP 提供全双工连接，可以在一条连接上同时传输两个独立的、流向相反的数据流。

TCP 头格式

在传输层，数据被封装成数据段，TCP 连接用段实现网络间的通信，段的最大长度取决于输出接口的最大报文长度或系统间协商的结果。

下图显示了 TCP 数据域头格式。

<table>
<tr><td colspan="3">源端口（16）</td><td>目标端口（16）</td></tr>
<tr><td colspan="4">序号（32）</td></tr>
<tr><td colspan="4">应答号（32）</td></tr>
<tr><td>头长度（4）</td><td>保留（6）</td><td>编码位（6）</td><td>窗口（16）</td></tr>
<tr><td colspan="3">校验和（16）</td><td>紧急（16）</td></tr>
<tr><td colspan="4">可选项（如果有，0或32）</td></tr>
<tr><td colspan="4">数据（可变）</td></tr>
</table>

图 TCP 头格式

TCP 头有 20 的字节，下面对它的头域进行一下简单的描述：

名称	描述
源端口（Source Port）	调用的端口号
目的端口（Destination Port）	被调用的端口号
序列号（Sequence Number）	确保数据到达序列正确的编号
应答号（Acknowledgment Number）	期望的下一个 TCP 数据段
头长度（HLEN）	以 32 位字为单位的报头长度
保留（Reserved）	置为 0
代码位（Code Bits）	开始、终止会话之类的控制功能
窗口（Window）	用来控制流量
校验和（Checksum）	头标和数据域计算的校验和

紧急（Urgent Pointer）	指示紧急数据的末端
选项（Option）	当前定义项：TCP 段的最大值
数据（Data）	上层协议的数据

建立 TCP 连接：三次握手

TCP 是面向连接的，在面向连接的环境中，开始传输数据之前，在两个终端之间必须先建立一个连接。建立连接的过程可以确保通信双方在发送应用数据包之前已经准备号了传送和接收数据。对于一个要建立的连接，通信双方必须用彼此的初始化序列号 seq 和来自对方成功传输确认的应答号 ack 来同步。（ack 号指明希望收到的下一个八位组的编号）习惯上将同步信号写为 SYN，应答信号写为 ACK。

整个同步的过程称为三次握手，下图说明了这个过程：

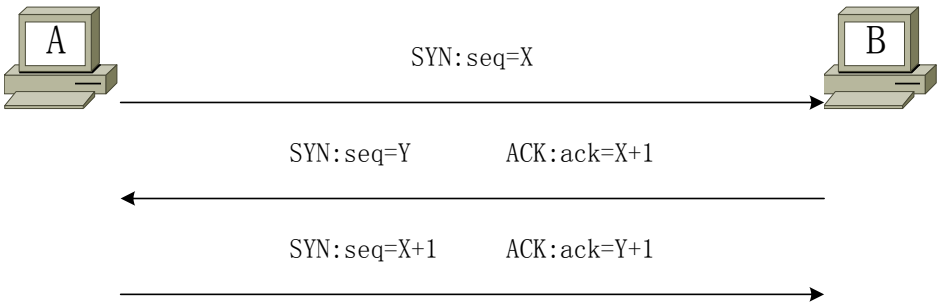


图 4 三次握手的过程

- 第一步：主机 A 发送 SYN 给主机 B：我的序列号 seq 是 X。
- 第二步：主机 B 发送 SYN、ACK 给主机 A：我的序列号 seq 是 Y，应答号 ack 是 X＋1（等待接收第 X＋1 号八位组）。
- 第三步：主机 A 发送 SYN、ACK 给主机 B：我的序列号 seq 是 X＋1，应答号 ack 是 Y＋1。

从此连接建立，开始传输数据。

TCP 是一种点对点的通信方式，任何一方都可以开始或终止通信。任何机器上的 TCP 都能被动地等待握手或主动的发起握手。一旦连接建立，数据可以对等地双向流动。

【注意】如果 TCP 使用 1 作为每次建立连接的初始化序列号，当本地系统重启后，远程系统会以为以前的连接依然存在。所以每次连接时，主机都会随即地选择一个初始化序列号，用它来辨别所传输的八位组在数据流中的位置。然后双方要对各自的序列号进行协商，因为接收方受到第一个 SYN 时，它并不知道这是否是一个被延迟的旧信号，所以它必须要求发送方验证这个 SYN。

一般情况下，TCP 使用最少量信息的报文段来实现三次握手，这对减少网络通信流量是有效的。

总之三次握手使通信双方做好了传输数据的准备，并且使通信双方统一了初始化序列号。

关闭 TCP 连接：改进的三次握手

对于一个已经建立的连接，TCP 使用改进的三次握手来结束通话。（使用一个带有 FIN 附加标记的报文段）TCP 关闭连接的步骤如下图所示：

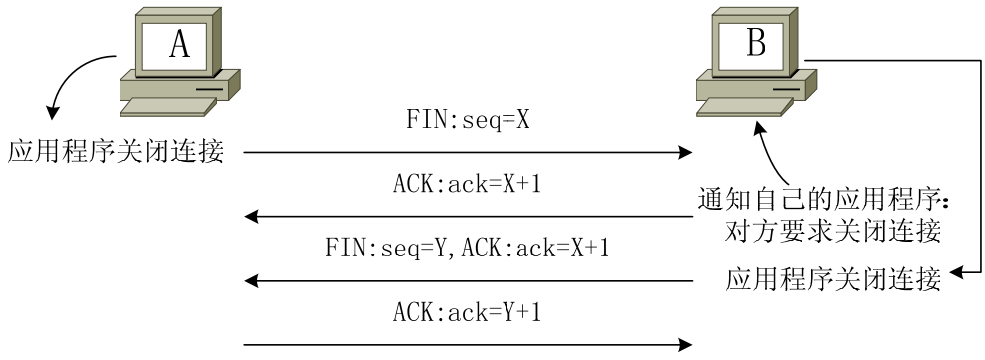


图　改进的三次握手

- 第一步：当主机 A 的应用程序通知 TCP 数据已经发送完毕时，TCP 向主机 B 发送一个带有 FIN 附加标记的报文段（FIN 理解为 finish）。
- 第二步：主机 B 收到这个 FIN 报文段之后，并不立即用 FIN 报文段回复主机 A，而是先向主机 A 发送一个确认 ACK，同时通知自己相应的应用程序：对方要求关闭连接（先发送 ACK 为了防止在这段时间内，对方重传 FIN 报文段）。
- 第三步：接主机 B 的应用程序告诉 TCP：我要彻底的关闭连接，TCP 向主机 A 送第二个 FIN 报文段。

- 第四步：主机 A 收到第二个 FIN 报文段后，向主机 B 发送一个 ACK 表示连接彻底关闭。

TCP 的可靠性

TCP 是面向流的，即数据段被当作字节的序列来进行传输。
在通过三次握手建立连接时，序列号被初始化。在传输过程中，TCP 继续使用这个序列号来标记每一个发送的数据段，每传送一个数据段，序列号加一。接收站点依据序列号重新组装所收到数据段。为什么要依靠序列号来重组数据段呢？例如在一个高速链路与低速链路并存的网络上，可能会出现高速链路上的数据段比低速链路上的数据段提前到达的情况，此时就必须依靠序列号来重组数据包，这就是序列号的作用之一。
在传输过程中确认号 ACK 的作用是告诉发送端哪些数据包已经成功接收。并且确认号会向发送端指出了接收端希望收到的下一个数据段的序列号。这种机制称为预期确认，即确认号等于下一个预期的位元组。
下图显示了数据包传输时顺序号和确认号所扮演的角色。

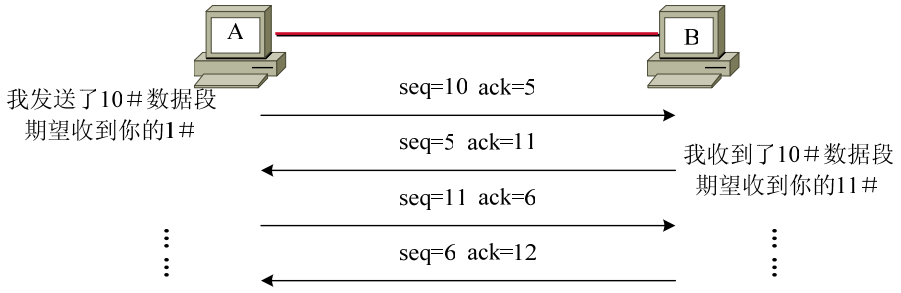


图　发送方等待每个数据段的确认信息

超时与重传

如果在传输过程中丢失了某个序列号的数据段，导致发送端在给定时间间隔内得不到那个数据段的应答，那么那个丢失数据段就会被要求重发。数据段会被保存在发送端的缓冲区中，直到发送端接受到应答号，它才会释放这个缓冲区。这种机制被称为肯定确认与重新传输 (Positive Acknowledgement and Retransmission, PAR) ，它是许多通信协议用来确保可信度的一种技术。
如下图：

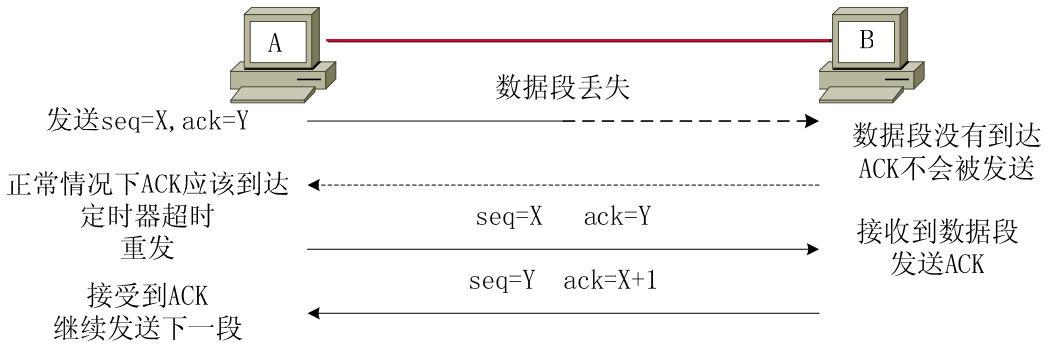


图　超时与重传

序列号的第二个作用就是消除网络中的重复包（同步复制）。例如在网络拥塞时，发送端迟迟没有收到接收端发来的某个数据段的 ACK 包，它可能会认为这个序列号的数据段丢失了，于是它会重新发送，这种情况可能会导致接收端在网络恢复正常后收到两个同样序列号的数据段，此时接收端会自动丢弃第二个一样数据段。
序列号和应答号为 TCP 提供了一种纠错机制，提高了 TCP 的可靠性。

滑动窗口与流量控制

TCP 窗口技术在 RFC793 和 RFC813 中有所说明。

窗口技术

在上节中，我们讲到为了实现可靠性，发送方发出一个数据段之后要等待接受方相应的确认信息，而不是直接发送下一个分组。
TCP 使用窗口技术来改善流量控制，以便通信双方能够充分利用带宽。滑动窗口允许发送方在收到接收方的确认之前发送多个数据段。窗口大小决定了在收到目的地确认之前，一次可以传送的数据段的最大数目。视窗大小越大，主机一次可以传输的数据段就越多。当主机传输视窗大小数目的数据段后，就必须等收到确认，才可以再传下面的数据段。例如，若视窗的大小为 1，则传完数据段后，都必须经过确认，才可以

再传下一个数据段；当窗口大小等于 3 时，发送方可以一次传输 3 个数据段，等待对方确认后，再传输下面三个数据段。

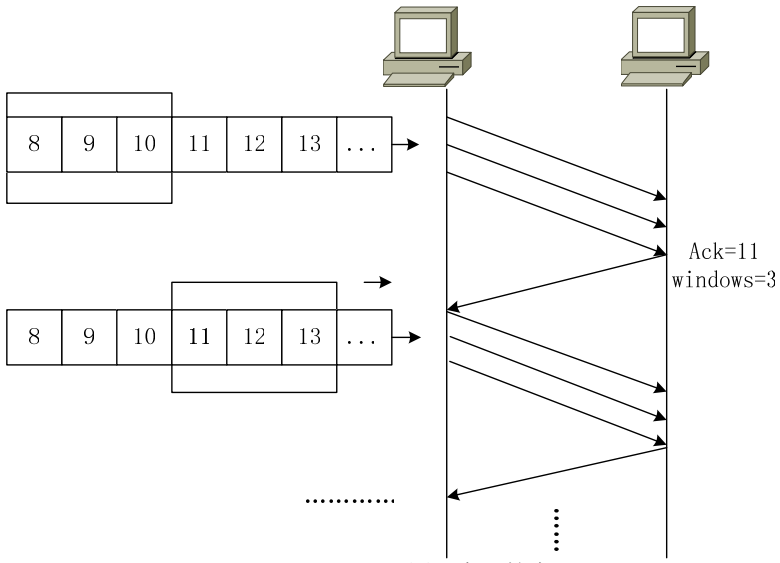


图　窗口技术

在上图中，窗口左边的是已经成功传输、被接收、确认的数据段，窗口中的是已经发送但还没有收到确认的数据段，窗口右边的是还没有发送的数据段。窗口技术大大提高了网络带宽的利用率。

流量控制

窗口的大小在通信双方连接期间是可变的，通信双方可以通过协商动态地修改窗口大小。在 TCP 的每个确认中，除了指出希望收到的下一个数据段的序列号之外，还包括一个窗口通告，通告中指出了接收方还能再收多少数据段（我们可以把通告看成接收缓冲区大小）。如果通告值增大，窗口大小也相应增大；通告值减小，窗口大小也相应减小。这种机制也可以防止网络拥塞。例如：当网络拥塞导致数据包丢失时，窗口大小自动减小一半，以保证数据传输。

下图中，发送方开始一次发送 3 个数据段，而接收方只能处理一个大小为 2 的窗口。接收方将丢弃第三个数据段将第三个数据段作为下一个期望收到的数据段，并且指定窗口大小为 2。在以后的传输中，发送方会自动调整窗口大小。

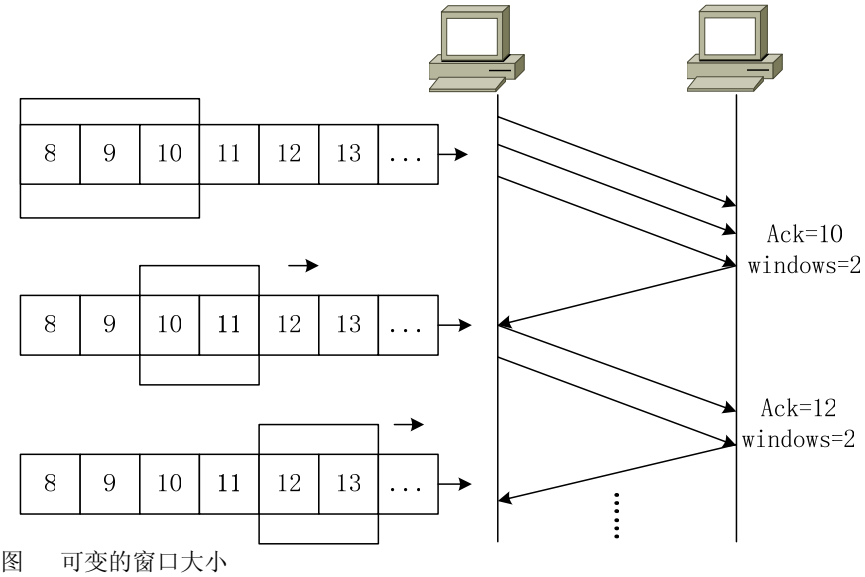


图　可变的窗口大小

TCP 不提供什么

TCP 的连接分为三个阶段：三次握手、数据传输、结束通话。其中“三次握手”和“结束通话”过程就需要交换不少数据，如果中间的“数据传输”只传输几个字节的信息的话，这种开销将显得十分昂贵。例如在进行 DNS 域名查询时，中间只要传输几个字节的消息，此时“三次握手”和“结束通话”的开销就显得非常大了，因此 DNS 使用 UDP 作为其传输层协议。

象 DNS 这种需要传输很少字节的通信被称为轻权通信，轻权通信一般交由 UDP 来完成。

TCP/IP 协议栈的 internet 层

TCP/IP 协议栈 internet 层提供寻址和路由选择协议，路由器主要工作在该层。TCP/IP 协议栈 internet 层对应 OSI 参考模型的网络层，该层主要运行以下几个协议：

- 网际协议（IP）：对数据包进行无连接的最佳的路由选择，它不关心数据包的内容，只查找一个路径将数据包送到目的地。
- 网际控制报文协议（ICMP）：提供控制和传递消息的功能
- 地址解析协议（ARP）：由已知的 IP 地址确定数据链路层的 MAC 地址
- 反向地址解析协议（RARP）：由已知的数据链路层 MAC 地址确定 IP 地址
- 动态主机配置协议（DHCP）：将 IP 地址和一些 TCP/IP 配置分配给网络中的计算机的一项服务和协议。

IP 头格式

IP 数据包由 IP 头和数据组成。IP 头的结构如下：

<table>
<tr><td>版本（4）</td><td>头长度(4)</td><td>服务类型（8）</td><td colspan="2">总长度（16）</td></tr>
<tr><td colspan="3">标识（16）</td><td>标志（3）</td><td>段位移（13）</td></tr>
<tr><td colspan="2">生存期（8）</td><td>协议（8）</td><td colspan="2">头校验和（16）</td></tr>
<tr><td colspan="5">源 IP 地址（32）</td></tr>
<tr><td colspan="5">目的 IP 地址（32）</td></tr>
<tr><td colspan="5">IP 选项（如果有，0 或 32）</td></tr>
<tr><td colspan="5">数据</td></tr>
</table>

图 10：IP 头格式

IP 头域有 20 字节，下面对它的头域进行一下简单的描述：

名称	描述
版本（VERS）：	表明了一个数据包采用的是因特网协议的哪个版本。对于 IPv4 ，这个域的值为 4。
头长度（HLEN）：	以字为单位的报头长度
服务类型（Type of Service）：	数据包的处理方式，前三位是优先级
总长度（Total Length）：	报头和数据的总长度
标识（Indentification）：	唯一的 IP 数据包值，可以理解为 IP 报文的序列号，用于识别潜在的重复报文等。
标志（Flags）：	指出数据包是否存在
段位移（Frag Offset）：	对数据包分片以允许互联网上的不同 MTU
生存期（TTL）(Time to Live)：	报头的存活时间，一旦该计数值减为 0，该报就被丢弃。TTL 用于限制一个 IP 包所经历的站数。正常设为 64，最大设为 255，TTL 每经过一个路由器减 1。当值为 0 时，数据包被丢弃。同时，路由器想发送者返回一个 ICMP 超时信息。通常数据包只会由于网络存在路由闭环而被丢弃。例如：当第一台路由器认为到达某一目的端的路径要经过第二台路由器，而第二台路由器又认为该路径应经过第一台路由器，这时会发生什么情况呢？当第一台路由器接收到一个发往该目的地址的数据包时，它会将数据包转发给第二台路由器，而第二台路由器就会将数据包重新转发给第一台路由器，然后第一台路由器又将包转发回第二台路由器。如果没有 TTL，这个包就会在这两台路由器构成的环路中永远转下去。这样的环路在大的网络中经常会出现。
协议（Protocol）：	发送数据包的上层（第四层）协议
头校验和（Header Checksum）：	报头上的完整性检查。头校验用来确认接收到的ＩＰ报头中有没有差错。头校验和只由Ｉ Ｐ报头中的各个域计算得来，而与 IP 包的净荷无关，ＩＰ包净荷的校验则是高层协议的工作。如果目的地计算的校验和与报文所含的校验和不同，那么这个数据包就会被丢弃。
源 IP 地址（Source IP address）：	标识通信终端设备的地址
目标 IP 地址（Destination IP address）：	标识通信终端设备的 IP 地址

IP 选项（IP Options）：	网络测试、调试、安全等功能选项
数据（Data）：	需要被传输的数据

数据包的大小、网络 MTU 及 TCP 最大报文段长度

本节中，我们将综合数据包大小、网络 MTU、TCP 最大报文段长度三个概念来进行讲述，加深读者的理解。

数据包的分片

我们知道，数据包是被封装在物理帧中传输的，对于网络硬件来说，它们对一个物理帧的可传输数据量都规定了一个上限值，这个上限值就是最大传输单元，即数据包的 MTU（maximum transfer unit）。例如：源于令牌环网的数据包最大传输单元（MTU）为 4500 字节、以太网的数据包最大传输单元为 1500 字节、FDDI 的数据包最大传输单元为 4770 字节。如果数据包的大小比互联网中最大网络的 MTU 要大，它是无法被封装到帧中去的；相反如果数据包的大小被限制为互连网中最小网络的 MTU，这种做法也是很不经济的（因为在大 MTU 的网络上，会造成带宽浪费）。

TCP/IP 怎样选择数据包的 MTU 呢？主要有以下两点：

1、TCP/IP 选择接近相连网络的 MTU 的值为初始数据包大小，例如：某主机连接在以太网上，那么 TCP/IP 会选择某个接近 1500 字节的值为初始数据包大小（如 1400、1440、1480）。

【注意】为什么会选择诸如 1400 这样的数值呢？因为 IP 数据包以 8 倍数的字节数来表示数据包的段位移，所以数据包大小肯定是 8 的倍数。

2、TCP/IP 同时也提供一种机制：在 MTU 较小的网络上，可以把大数据包划分成更小的数据包片（分片），即如果 TCP 选择初始数据包大小为 1500 字节，在中途有个 MTU 为 620 的网络，处于两种网络分界处的路由器会对数据包进行分片操作。如下图：

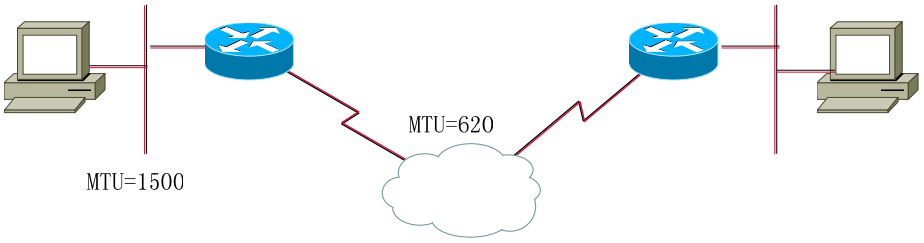


图　数据包需要分片

IP 报头中的标志和段位移就是用于将大 IP 包分割成几个称为片的小块，以保证它可以顺利地穿过无力处理大 IP 包的网络。其中，标识是发送端填写的值，以便接收端重组那些不得不分成几个片的包；段位移可以标识出每个分段偏移量（如下图所示），从而使目的系统可以正确的重组原来数据包。

【注意】这种分片不一定要把数据包分成相同大小的片，首先选择最接近网络 MTU 的 8 倍数的字节数为数据包大小；一般最后一片是前面分下来的零头，所以最后一片往往比其他片小。

分片以后所得的每个数据包的格式都与原来的数据包相同，都包含了原来数据包的 IP 报头，数据包片的总长度（报头＋数据）小于网络 MTU。如下图：

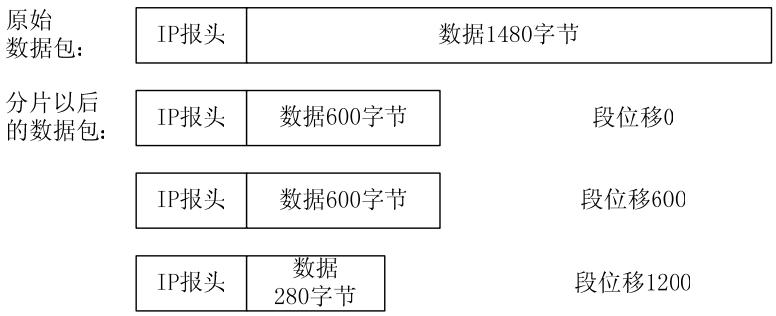


图 数据包分片

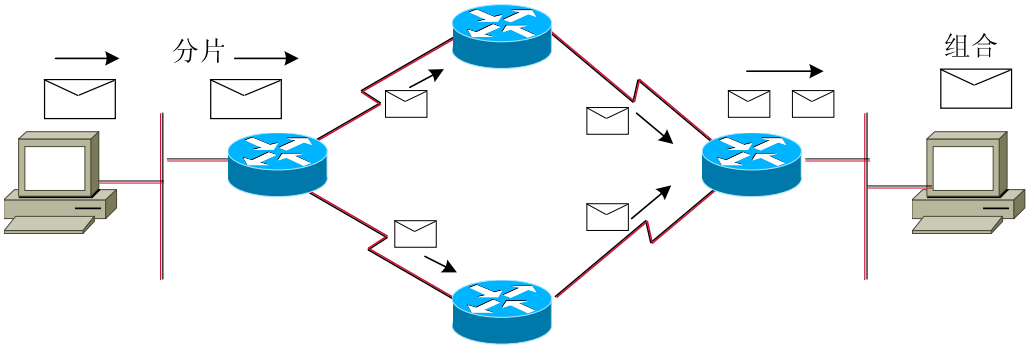
上图中源站点选择了 1500 字节作为数据包大小（20＋1480，其中 20 为 IP 报头长度），每个数据包经过两种网络的边界路由器时，被分成三片，每片的大小都小于或等于 620 字节，这样数据包就可以顺利地通过整个网络。那么数据包分片以后怎样重组呢？下面我们就来讨论这个问题。

数据包分片后的重组

首先，我们强调一点，数据包片的重组是发生在分片到达目标主机之后，而不是在通过 MTU 较小的网络之后的路由器上。一旦数据包分片之后，每个数据包片将被作为独立的数据包传输，即使在中途又遇到具有大 MTU 的网络，重组依然发生在分片到达目标主机之后。如果在传输过程中某个分片丢失了，目标主机将对丢弃整个数据包，所以分片增加了数据包丢失的概率，应尽量避免数据包分片。

虽然在目标主机上重组数据包片存在一些缺点，但是每个数据包片可以象数据包一样选择路由（可以在多路环境下实现负载平衡，即数据包的一部分分片走链路 1，一部分分片走链路 2，如下图），而且路由器不

负担数据包片的重组工作也有利于减轻路由器负担。



数据包分片

TCP 最大报文段长度

前面讲到传输层的 TCP 要对应用层生成的数据流进行分段，通信双方必须协商一个最大的报文段长度（Maximum segment size）。一般情况下 TCP 会自动计算选择合适的报文段长度，使生成的 IP 数据包与网络 MTU 相适应。

在互联网中选择合适报文段长度是很困难的，过大或者过小都会使网络性能变坏。一方面，报文段太小会降低网络利用率。因为 TCP 报文段是封装在 IP 数据包中的，而 IP 数据包又被封装在帧中，每一层的封装都要加上自己的头部，如果报文段很小，每一层的头部在数据帧中所占的比重就大，势必造成了带宽的浪费。

另一方面，报文段太大也会降低网络性能。大的报文段产生了大的数据包，如果网络中存在 MTU 较小的网络，TCP/IP 就会对数据包进行分片。我们知道数据包片并不像数据段那样能进行确认与重传，如果某个数据包片在传输过程中丢失了，整个数据包就要被重传，如果数据包分片太多，势必会影响网络性能。

理论上讲，TCP 报文段的最佳长度的要求是：尽可能携带更多的数据，而且 IP 数据包在传输过程中尽量不被分片。这个要求只是个理论要求，在真正实现时，还会遇到更多的问题，在这里就不详细的讨论了。

ICMP

ICMP（Internet Control Message Protocol，Internet 控制消息协议）经常被认为是 IP 层的一个组成部分。它携带于 IP 数据包中（如下图）。

20字节

图 ICMP 封装在 IP 数据包内部

ICMP 定义了一套差错报文和控制报文，用于主机与路由器之间交换不可达目的地址、网络拥塞、重定向到更好的路径、报文生命周期超时等信息。例如：用 ping 命令来查询 192.168.10.1 的主机是否在线，路由器会给你一个 ICMP 应答：目标端不可到达或者从 192.168.10.1 有回应。如下图所示：

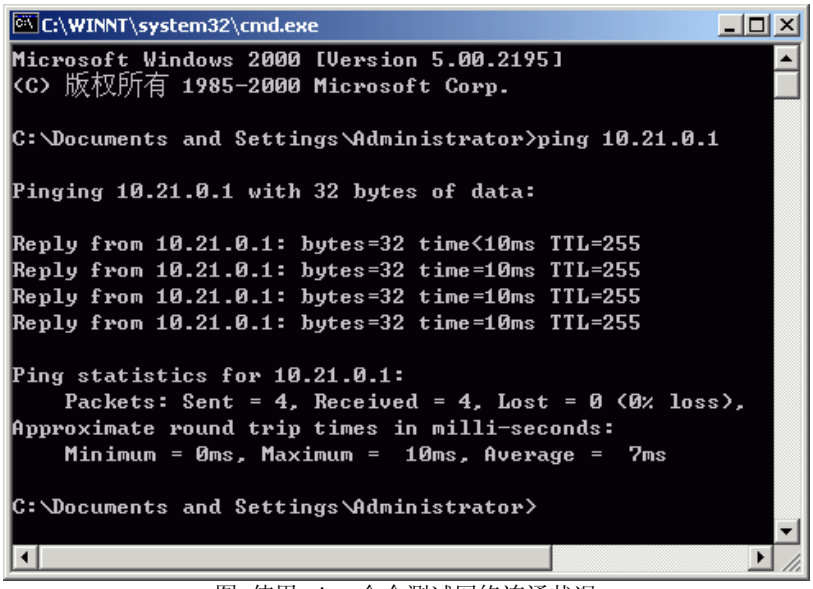


图 使用 ping 命令测试网络连通状况

ICMP 报文主要有两大类：查询报文和错误报文。查询报文是指 ICMP 响应请求、响应回答、路由器公告、地址屏蔽请求等。而绝大部分 ICMP 消息是错误报文，例如：目的地址不可到达、源地址消亡、生命周期超时等。

ICMP 报文的格式下图所示。所有报文的前 4 个字节都是一样的，但是剩下的其他字节则互不相同。

类型（8 位）	代码（8 位）	校验和（16 位）
（因不同的类型和代码而有不同的内容）		

图　ICMP 报文

类型字段可以有不同的值，以描述特定类型的 ICMP 报文。某些 ICMP 报文还使用代码字段的值来进一步描述不同的条件。各种类型的 ICMP 报文由下表所示，它由报文中的类型字段和代码字段来共同决定。表中的最后两列表明 ICMP 报文是一份查询报文还是一份错误报文。

类型	代码	描述	查询	错误
0	0	回显应答（Ping 应答）	√	
3		目标不可到达		√
3	0	网络不可到达		√
3	1	主机不可到达		√
3	2	协议不可到达		√
3	3	端口不可到达		√
3	4	禁止分割		√
3	5	源站选路失败		√
3	6	目标网络不认识		√
3	7	目标主机不认识		√
3	8	源主机被隔离（作废不用）		√
3	9	目标网络被强制禁止		√
3	10	目标主机被强制禁止		√
3	11	由于服务类型 TOS，网络不可到达		√
3	12	由于主机类型 TOS，主机不可到达		√
3	13	由于过滤，通信被强制禁止		√
3	14	主机越权		√
3	15	优先权中止生效		√
4	0	源端口被关闭（基本流控制）		√
5		重定向		√
5	0	对网络重定向		√
5	1	对主机重定向		√
5	2	对服务类型和网络重定向		√
5	3	对服务类型和主机重定向		√
8	0	请求回显（ping 请求）	√	

9	0	路由器通告	√	
10	0	路由器请求	√	
11		超时		√
11	0	传输期间生存时间（TTL）为 0（Traceroute）		√
11	1	在数据包组装期间生存时间为 0		√
12		参数问题		√
12	0	坏的 IP 首部（包括各种差错）		√
12	2	缺少必须的选项		√
13	0	时间戳请求	√	
14	0	时间戳应答	√	
15	0	信息请求	√	
16	0	信息应答	√	
17	0	地址掩码请求	√	
18	0	地址掩码应答	√	

【注意】目的地址是广播地址或多播地址（D 类地址）的 IP 数据包不会产生 ICMP 错误报文，这是为了防止 ICMP 错误报文对广播分组的响应所带来的广播风暴。

由于篇幅，我们不可能对所有类型的报文一一讨论，下面我们主要讲述目标不可到达、超时、重定向、回应请求与回应应答的 ICMP 报文。

ICMP 不可到达

ICMP 不可到达的报文类型值为 3，可以依据不同的代码值实现不同的不可达功能。下面我们用一个例子简要地描述一下几种常用的代码类型，对于我们没有讲到的代码类型，可以参阅相关的 RFC 文档。

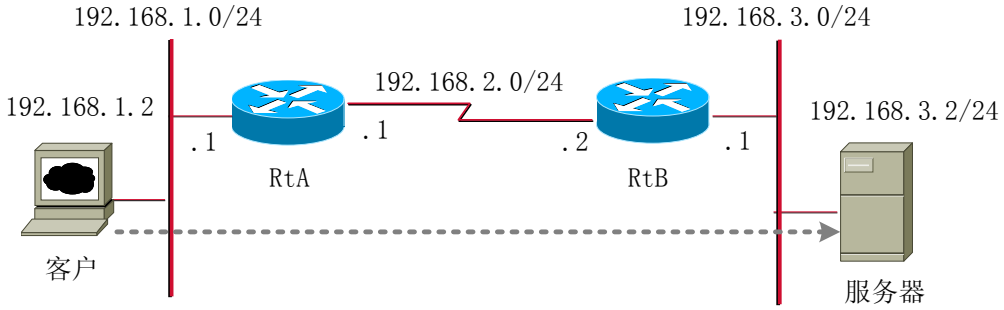


图　ICMP 不可到达

上图中左边的客户机试图连接右边的 WEB 服务器，它们以 TCP 作为传输协议。下面我们假设这个网络的某个部分出现故障，然后讨论不同情况下 ICMP 的操作。在这些例子中我们有可能提到路由的概念，读者可以参阅下一章的内容。

网络不可到达

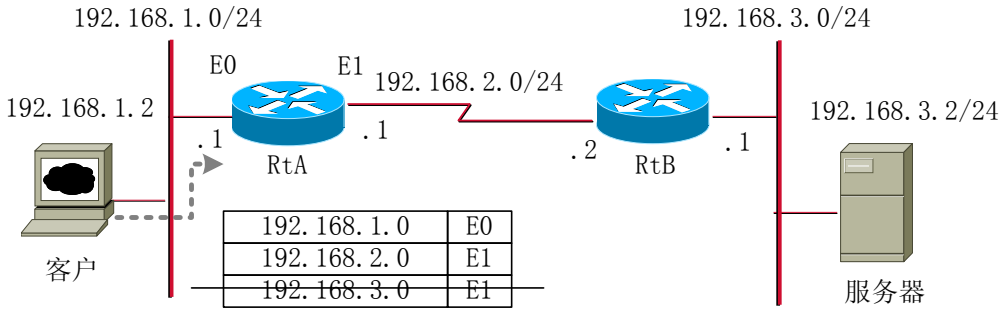


图 网络不可到达

如上图，若路由器 A 没有学习到到达 192.168.3.0/24 的路由，路由器 A 就会使用代码号为“0”的“网络不可到达”的代码向客户机返回一个 ICMP 消息，以响应客户机的目的地为 192.168.3.2/24 的包数据包。“网络不可到达”的代码用来表示某个网络（网段）不可到达。

主机不可到达

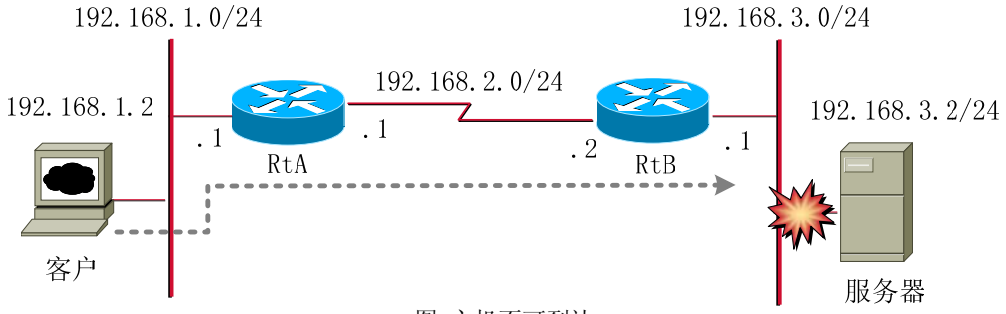


图 主机不可到达

如上图，若路由器 A 有到达 192.168.3.0/24 的路由，它会将数据包传给路由器 B。但是这个时候 WEB 服务器忽然 Down 机了，路由器 B 接受不到来自 WEB 服务器的信息，因此路由器 B 会使用代码号为“1”的“主机不可到达”的代码向客户机返回一个 ICMP 消息。“主机不可到达”的代码用来表示某单一主机不可到达。

禁止分割

在上面讲述 IP 层的数据包大小时，我们曾提到：在 MTU 较小的网络上，需要把大数据包划分成更小的数据包片（分片），以保证数据包正常通过网络。

假如路由器 A 在将客户机的数据发往服务器时，需要分割客户机的数据包，但是数据包的 IP 头中却设置了拒绝分片位，那么路由器将会向客户机返回一个代码为 4 的“禁止分割”的 ICMP 消息。

【注意】以上三种消息都是由路由器发送的。

协议不可到达

若数据包成功地到达了 WEB 服务器，但是服务器上不运行 TCP 或者 UDP 协议（基本上不可能有这种情况），那么 WEB 服务器将返回一个代码为 2 的“协议不可到达”的 ICMP 消息。

端口不可到达

若数据包成功地到达了 WEB 服务器，服务器上也运行 TCP 协议，但是服务器上的相关软件还没有运行，无法处理客户机连接，于是服务器上的 TCP/IP 将返回一个代码为 3 的“端口不可到达”的 ICMP 消息。

【注意】以上两种消息是由具体主机发送的。

下表总结了上述的 ICMP 不可到达的消息。

代码描述	场合	发送方
目标网络不可到达	路由表中无目标网络	路由器
目标主机不可到达	主机无响应	路由器
禁止分割	需要进行分割但设置了不分割比特位	路由器
协议不可到达	主机上无相关的传输层协议（TCP、UDP）	主机
端口不可到达	目标端口没有被应用程序打开	主机

ICMP 超时

超时 ICMP 报文与 IP 报头中 TTL 字段一起使用。我们简单讨论一下“传输期间 TTL 值为 0”的代码。

当数据包到达路由器时，路由器都需要把数据包 IP 头中的 TTL 的值减 1。当 TTL 值被减到 0 时，数据包就会被丢弃。此时丢弃这个包的路由器会返回一个代码为“传输期间 TTL 值为 0”的 ICMP 消息给原始发送者。TTL 值可以防止数据包在网络上被循环往复的传输。例如，当发生路由回路时，数据包可能在回路上被一直循环的传输。但是由于数据包每次经过路由器，TTL 字段的值都会减 1，因此当 TTL 值减为 0 时，循环包就会被自动丢弃。

Traceroute 命令可以让我们看到 IP 数据包从一台主机传输到另一台主机的过程中所经过的路由。如下例所示：

例：在 Windows2000 上使用 tracert 命令

```
c:\>tracert www.njupt.edu.cn
Tracing route to www.njupt.edu.cn [202.119.230.10]

over a maximum of 30 hops:

  1   <10 ms    10 ms    10 ms  10.21.0.1
  2   <10 ms   <10 ms   <10 ms  192.168.12.2
  3   <10 ms   <10 ms   <10 ms  192.168.11.1
  4   <10 ms   <10 ms   <10 ms  www.njupt.edu.cn [202.119.230.10]
Trace complete.
```

【注意】Tacreroute 是 Cisco 路由器上的命令格式，在 Windows2000 中，trace 命令写为“tracert”。

例：在 Cisco2600 系列路由器上使用 traceroute 命令

```
Router#traceroute www.njupt.edu.cn
```

```
Type escape sequence to abort.
Tracing the route to www.njupt.edu.cn (202.119.230.10)

  1 10.21.0.1 8 msec 4 msec 8 msec
  2 192.168.12.2 4 msec 0 msec 0 msec
  3 192.168.11.1 0 msec 0 msec 4 msec
  4 www.njupt.edu.cn (202.119.230.10) 0 msec *  0 msec
Router#
```

Traceroute 命令利用了“TTL 超时”的报文。当我们使用 Traceroute 命令时，发送方使用 UDP 故意发送一份 TTL 为 1 的 IP 数据包给目标主机，处理这份数据包的第一个路由器将 TTL 值减 1，然后丢弃该数据包，并向发送方返回一份超时 ICMP 报文。这样 trace 命令就得到了该路径中的第一个路由器的地址。

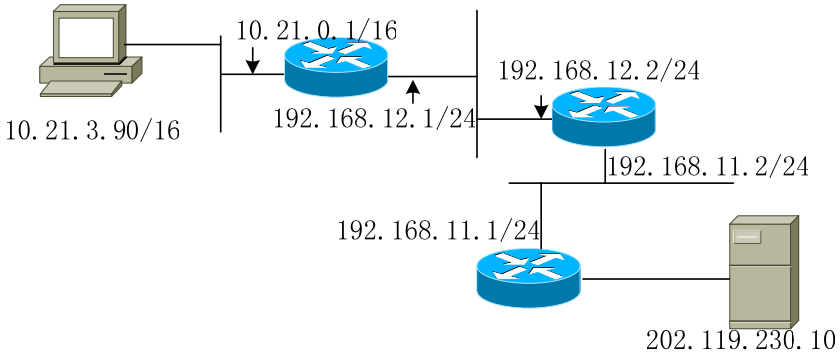


图 ICMP 超时消息的应用

然后 Traceroute 命令发送一份 TTL 值为 2 的数据包，第一个路由器将数据包转发给第二个路由器。而在第二个路由器上，数据包的 TTL 值会被减为 0，因此这个路由器将丢弃这个数据包并向发送方返回 ICMP 超时消息，这样我们就可以得到第二个路由器的地址。

Traceroute 命令继续上述过程直至数据包到达目标主机。

【注意】traceroute 命令如何判断数据包是否已经到达目标主机呢？

实际上执行 Traceroute 命令的设备发送原始数据包时，它会选择一个几乎不可能的值作为目标 UDP 端口号（大于 30000），目标主机的任何一个应用程序都不可能使用该端口。那么当该数据包到达目标主机时，目标主机的 UDP 将产生一份代码为“端口不可达”的 ICMP 报文。而之前路由器返回的是“超时”ICMP 报文，这样，Traceroute 命令只要区分所接收到的 ICMP 报文是“超时”还是“端口不可达”，就可以判断数据包是否已经到达目的地了。

ICMP 重定向

ICMP 重定向报文是一个非常重要的工具。我们知道：当主机向非本地子网发送数据包时，TCP/IP 会自动将数据包转发给它的默认网关(缺省路由器)。但是如果网络中还存在一个相对来说更好的本地路由器时，ICMP 重定向功能会通知主机以后将这些数据包发送给那个更好的路由器。如下图：

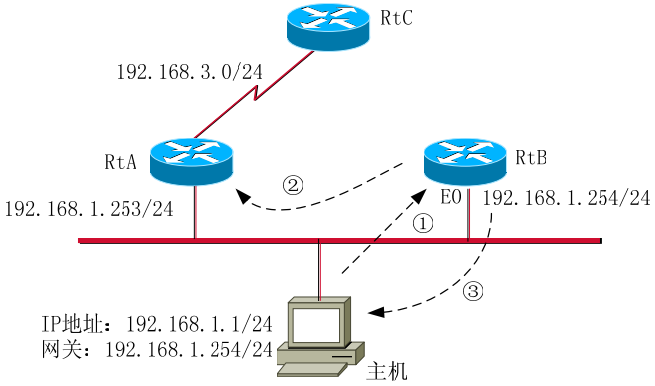


图　ICMP 重定向

①　主机使用路由器 B（192.168.2.0/24）作为默认网关，但是对于到某个子网的路由（例如此例中的 192.168.3.0/24），路由器 A 才是最好的选择。默认情况下，主机还是会将数据包转发给路由器 B。

②　路由器 B 从 E0 口收到主机发来的数据包并且检查自己的路由表，它发现路由器 A 是去往目标地址的下一跳，于是路由器 B 又从 E0 口把数据包转发给路由器 A。此时路由器 B 会检测到它正在发送数据的接口和此数据包到达的接口是同一个接口（即主机和两个路由器同处于一个本地子网中）。

③　路由器 B 发送一份 ICMP 重定向报文给主机，告诉它以后把这类数据包发送给路由器 A 而不是它自己。

ICMP 回应请求与回应应答

我们对 Windows2000 中的 ping 命令的使用应该很熟悉了，ping 命令负责发送和接收 ICMP 回应请求及回应应答报文。

实际上 ping 所产生的数据报文是 IP 网络中能够生成和寻址的最小报文，它很适合排除网络连接方面存在的问题。

Ping 命令有些类似声纳与雷达，典型的 ping 命令实现方法是发送方首先在欲发送的 ICMP 回应请求报文的数据域中存放当前时间；当目标主机收到回应请求后，它返回一个 ICMP 回应应答报文给发送方；发送方收到 ICMP 回应应答报文后将报文到达的时间减去回应请求的发送时间就得到往返时间。

ICMP 回应往返时间＝ICMP 回应应答报文到达时间－ICMP 回应请求报文的发送时间。

Cisco 路由器和 Catalyst 交换机上实现的扩展 ping 命令有很多实用的功能。例如：当用户在 Cisco 路由器上输入“ping”回车以后，会出现一个会话，在其中用户可以自己定制 ping 命令发送的报文长度、数目、超时值、源地址、目标地址等。如下图所示：

```
rtl#ping

Protocol [ip]:
Target IP address: 192.168.1.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:y
Source address or interface:192.168.1.2
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.1, time out is 2 secondes:
!!!!!
Success rate is 100 persent (5/5) , round-trip min/avg/max = 1/2/4 ms
```

ARP

介质访问地址(Media Access Control，MAC)对于每一台设备是全球唯一的，该地址被烧录在网卡（Network Interface Card，NIC)的硬件电路上。MAC 地址由 12 位 16 进制数表示，其中前 6 位标识网卡的制造厂商，后 6 位是网卡的序列号。

在以太网中，一个主机要和另一个主机进行直接通信，必须知道目标主机的 MAC 地址。

什么是 ARP

ARP（Address Resolution Protocol，地址解析协议）用来将 IP 地址映射到 MAC 地址，以便设备能在多路访问介质上通信。

可以举一个例子很好的说明 ARP 是如何工作的：老师要将一封信交给教室里的某个学生，但是她并不认识这个学生，她只知道这个学生的姓名（IP 地址），于是她对教室里所有的人说：“谁是小王，有你的信！”(ARP 请求)，当小王听到这个消息时（地址匹配），他站起来回答，然后老师就知道了他坐在几排几座（MAC 地址），最后把信送到他座位上。

在 ARP 协议的实现中还有一些应该注意的事项：

① 每台计算机上都有一个 ARP 缓冲，它保存了一定数量的从 IP 地址到 MAC 地址的映射，同时当一个 ARP 广播到来时，虽然这个 ARP 广播可能与它无关，但 ARP 协议软件也会把其中的物理地址与 IP 地址的映射记录下来，这样做的好处是能够减少 ARP 报文在局域网上发送的次数。

② 按照缺省设置，ARP 高速缓存中的项目是动态的，ARP 缓冲中 IP 地址与物理地址之间的映射并不是一旦生成就永久有效的，每一个 ARP 映射表项都有自己的寿命，如果在一段时间内没有使用，那么这个 ARP 映射就会从缓冲中被删除，这一点和交换机 MAC 地址表的原理是一样的。这种老化机制，可以大大减少 ARP 缓存表的长度，加快查询速度。

在以太网中，当主机要确定某个 IP 地址的 MAC 地址时，它会先检查自己的 ARP 缓冲表，如果目标地址不包含在该缓冲表中，主机就会发送一个 ARP 请求（广播形式），网段上的任何主机都可以接收到该广播，但是

只有目标主机才会响应此 ARP 请求。由于目标主机在收到 ARP 请求时可以学习到发送方的 IP 地址和 MAC 地址的映射，因此它采用一个单播消息来回应请求。这个过程可以用下图来表示：

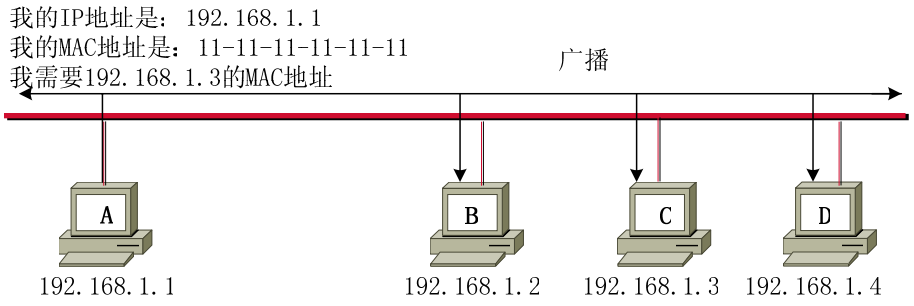


图　ARP 请求

在上图中，主机 A 以广播形式发送 ARP 请求查询 IP 地址为 192.168.1.3 的主机的 MAC 地址，网段上所有的主机都会收到该 ARP 请求。

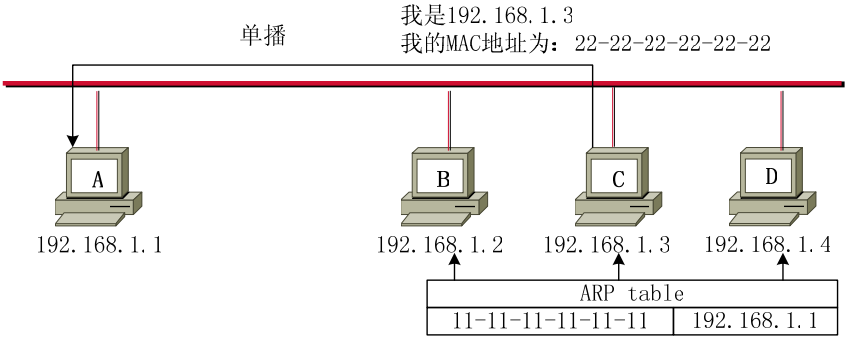


图 ARP 回应

主机 B、主机 D 收到主机 A 发来的 ARP 请求时，它们发现这个请求不是发给自己的，因此它们忽略这个请求，但是它们还是将主机 A 的 IP 地址和 MAC 地址的映射记录到自己的 ARP 表中。当主机 C 收到主机 A 发来的 ARP 请求时，它发现这个 ARP 请求是发给自己的，于是它用单播消息回应 ARP 请求，同时记录下其 IP 地址和 MAC 地址的映射。

通常 ARP 协议都在支持广播的网络上使用，例如以太网。但是 ARP 数据包不能跨网段使用，也就是说不能跨越路由器（路由器本身用作 ARP 代理除外）。当目标网络 IP 地址和源 IP 地址不在同一网段上时，就要使用代理 ARP，这部分内容将在下一章 IP 寻址中讲述。

在 PC 上使用 ARP 命令

Windows2000 支持 ARP 命令，使用 ARP 命令，你能够查看本地计算机的 ARP 高速缓存中的内容。此外，使用 ARP 命令，也可以以人工方式输入静态的网卡物理/IP 地址对，你可能会使用这种方式为缺省网关和本地服务器等常用主机进行这项操作，有助于减少网络上的信息量。

常用 ARP 命令选项：

arp -a 或 arp -g——用于查看高速缓存中的所有项目。“-a”和“-g”参数的结果是一样的，多年来“-g”一直是 UNIX 平台上用来显示 ARP 高速缓存中所有项目的选项，而 Windows 用的是 arp “－a”(“-a”可被视为 all，即全部的意思)，但它也可以接受比较传统的“-g”选项。

arp -a IP——如果你有多个网卡，那么使用 arp -a 加上接口的 IP 地址，就可以只显示与该接口相关的 ARP 缓存项目。

arp -s IP MAC——你可以向 ARP 高速缓存中人工输入一个静态项目。该项目在计算机引导过程中将保持有效状态，或者在出现错误时，人工配置的物理地址将自动更新该项目。

arp -d IP——使用本命令能够人工删除一个静态项目。

下面是一个 ARP 命令的输出结果：

```
C:\arp -a
Interface: 10.21.1.31 on Interface 0x5000004
Internet Address      Physical Address      Type
10.21.0.1             00-e0-fc-0c-1f-60     dynamic
10.21.0.127           00-e0-fc-0c-1f-60     dynamic
10.21.1.31            00-e0-4c-f8-db-85     static
10.21.2.177           00-0c-6e-3b-66-e8     dynamic
```

上例中“dynamic”表示此 ARP 项目是动态学习到的，“static”代表此 ARP 项目是静态指定的。

在 Cisco 路由器上使用 ARP 相关的命令

在 Cisco 路由器上，使用 show ip arp 命令可以查看路由器当前的 ARP 表。

```
Router#show ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.21.3.1               0  00e0.fc0c.1f60  ARPA   Ethernet0/0
Internet  10.21.0.1               0  00e0.fc0c.1f60  ARPA   Ethernet0/0
Internet  10.21.1.1               0  00e0.fc0c.1f60  ARPA   Ethernet0/0
Internet  10.21.2.154             -  0002.16f6.b640  ARPA   Ethernet0/0
Internet  10.21.0.207            26  00e0.fc0c.1f60  ARPA   Ethernet0/0
```

下表是对上面各输出参数的解释：

参数	解释
Protocol	网络地址所使用的协议，这里的协议是 IP。
Address	网络地址，这里指 IP 地址。
Age (min)	ARP 缓冲条目的寿命，其中“-”代表本地连接，即路由器接口的本身的缓冲条目。
Hardware Addr	MAC 地址。
Type	ARP 条目的封装类型，其中 ARPA 代表标准的以太网地址解析类型（RFC826 定义），snap 代表 FDDI 和 TokenRing 的地址解析类型（RFC1042 定义）。
Interface	路由器上学习到此 ARP 条目的接口。

【注意】ARP 不仅仅存在于 IP 网络中，其它网络（例如 IPX、Appletalk）也存在 ARP，本书只讨论 IP 网络中的 ARP。

我们也可以在 Cisco 路由器中手动地指定静态的 ARP 映射条目，通常可以为缺省网关和本地服务器等常用主机进行静态映射，这样有助于减少网络上的信息量，其命令格式为：

arp ip-address hardware-address type [alias]

下表是对 arp 命令参数的解释：

参数	解释
ip-address	IP 地址
hardware-address	MAC 地址
type	ARP 条目的封装类型，以太网中为 ARPA。
[alias]	可选参数，如果配置了这个参数，当 Cisco 路由器收到关于这个条目中的 IP 地址的 ARP 请求时，它会自动回应 ARP 请求，就象它自己拥有这个 IP 地址一样。

我们举例说明 ARP 命令的使用方法：

```
Router#conf t
Router(config)#arp 10.21.9.254 2222.2222.2222 arpa alias
```

例:ARP 命令的使用

要在 Cisco 路由器上配置静态 ARP 映射条目，首先应进入全局配置模式，然后在输入相应的命令，关于路由器上的 IOS 软件的操作方法，在本书第四章会详细讨论。

为了让读者加深理解，我们再次使用 show ip arp 命令查看配置静态条目以后的 ARP 表。

```
Router#show ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.21.9.254              -  2222.2222.2222  ARPA
Internet  10.21.3.1               0  00e0.fc0c.1f60  ARPA   Ethernet0/0
Internet  10.21.0.1               0  00e0.fc0c.1f60  ARPA   Ethernet0/0
Internet  10.21.1.1               0  00e0.fc0c.1f60  ARPA   Ethernet0/0
Internet  10.21.2.154             -  0002.16f6.b640  ARPA   Ethernet0/0
Internet  10.21.0.207            26  00e0.fc0c.1f60  ARPA   Ethernet0/0
```

RARP

我们知道通常主机的 IP 地址都是保存在硬盘中的，Windows 在启动时会找到它，但是对于那些把文件存放在远程服务器上的工作站来说，他们在启动时是如何获得 IP 地址的呢？

反向 ARP（Reverse Address Resolution Protocol，RARP）是 ARP 的逆过程，RARP 就是用于那些不知道自己 IP 地址的无盘工作站或者无配置的 cisco 路由器。使用 RARP 时,站点广播一个包含自己 MAC 地址的 RARP 请求，网络上所有的主机都会接收到该请求，但只有被授权的 RARP 服务器才能处理这个请求。RARP 服务器有一张映射表，它可以查出该 MAC 地址与哪个 IP 地址相对应，然后把响应发送给源站点。

下图可以清楚的说明 RARP 是如何工作的：

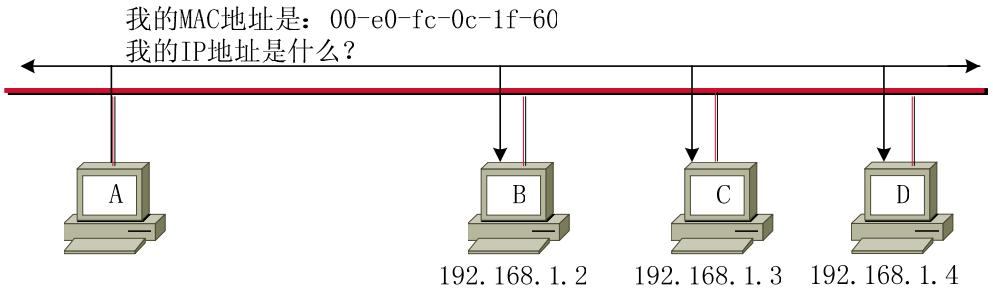


图　RARP 请求

图中主机 A 以广播形式发出 RARP 请求，网段上的所有主机都会接收到该请求。

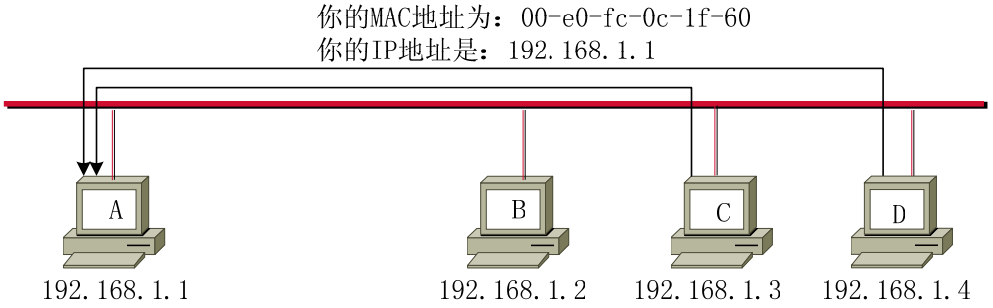


图 RARP 应答

如果某个网段上存在多个被授权的 RARP 服务器时，这些 RARP 服务器都会以单播方式响应 RARP 请求。不过当源主机接收到一个 RARP 应答时，它只接收第一个到达的 RARP 应答。

【注意】ARP 和 RARP 都是在数据链路层上实现的。

动态主机配置协议（DHCP）是现代 RARP 的实现，下面介绍 DHCP。

DHCP

DHCP 是将 IP 地址和一些 TCP/IP 配置分配给网络中的计算机的一项服务和协议。它克服了手动配置 TCP/IP 客户端及维护 IP 地址的局限性。

DHCP 是 Bootstrap Protocol（引导协议）的扩展。BOOTP 的主要限制是在于管理员必须手动为每个客户端在服务器上输入配置信息，DHCP 对 BOOTP 进行了改进，它专门设计了一个地址池，从地址池中动态地为客户机分配 IP 地址和 TCP/IP 配置。

使用 DHCP 有以下几点好处：

- DHCP 省去了很多维护工作，管理员无须到每台客户机上去设置 TCP/IP 设置、无须维护 IP 地址分配表。
- 当网络中的 IP 配置需要更换时（例如 IP 地址从 192.168.1.0/24 换到 10.10.0.0/16、或者 DNS 服务器地址需要更换等），使用 DHCP 您只需在 DHCP 服务器上更改相应的设置就可以轻松完成网络升级。
- DHCP 大大降低了 IP 地址冲突的可能性。

DHCP 通信过程

一台 DHCP 服务器可以是一台 Windows2000Server，也可以是一台 Cisco 路由器，本节我们以 Cisco 路由器为例对 DHCP 的工作过程进行讲解。

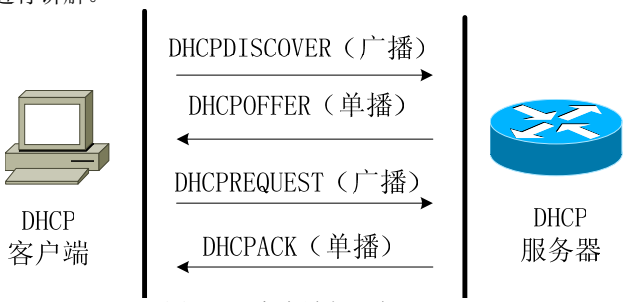


图 DHCP 客户端与服务器

如上图所示：

① DHCP 客户端首次初始化时会向 DHCP 服务器发送一个请求（DHCPDISCOVER），请求获得 IP 寻址信息，这个寻址信息包括 IP 地址、子网掩码、默认网关、DNS 服务器地址等，请求中同时也包含了客户机自己的 MAC 地址信息。DHCPDISCOVER 以广播形式发送，网段上的所有设备都会收到这个请求。

② 当 DHCP 服务器接收到请求时，它会从自己的地址池中选择一个 IP 地址分配给客户机，并且把其他

TCP/IP 配置一起发送过去（DHCPOFFER）。DHCPOFFER 以单播形式发送，因为它是针对某个具体主机的消息，DHCP 服务器可以从 DHCPDISCOVER 消息中获得客户机的 MAC 地址。

③ 当客户端接收到服务器所提供的信息时，它又以广播方式发送一个 DHCPREQUEST 消息，指明：我需要得到你的服务。

【注意】为什么还要以广播形式发送 DHCPREQUEST 消息呢？

如果一个网段上存在多个 DHCP 服务器，那么 DHCP 客户端可能会收到多个 DHCP 服务器响应的 DHCPOFFER 消息，DHCP 客户端只会选择最先收到的那个 DHCPOFFER 消息。以广播方式发送 DHCPREQUEST 消息有两个作用：一是通知那个服务器：我已经收到你所提供的 IP 地址，我需要你的服务；二是通知网络上其他 DHCP 服务器：我拒绝你们提供的 IP 寻址信息。

④ DHCP 服务器接收到 DHCPREQUEST 消息后，它会将所提供的 IP 地址和其他设置交给数据库，并且向 DHCP 客户端以单播形式发送一个 DHCPACK 消息，确认 DHCP 过程已经完成。

这样这个 IP 地址就会租给这个客户端一段时间，在租用期间，客户端每次登陆时都会向服务器发出这个 IP 地址的续定请求（DHCPREQUEST）。如果租用期到了，但是客户端没有续定，这个 IP 地址就会退回到 DHCP 服务器的地址池中等待重新分配。

DHCP 地址的分配类型

DHCP 的核心功能是分配 IP 地址，这个 IP 地址对于每个客户机都必须是唯一的。下面定义了三种类型的 IP 地址分配：

- 静态分配

管理员将某个 IP 地址固定地分配给某个主机，服务器接收到该主机的请求时就将该地址提供给这个主机，并且该地址只能分配给该主机。静态分配一般用于为网络中的服务器或者固定主机分配 IP 地址。

- 动态分配

管理员首先在 DHCP 服务器上定义一个地址池。当服务器收到 DHCP 请求时，它从地址池中取出一个 IP 地址分配给客户机（租借）。如果租用期到期，并且客户端没有续定，这个 IP 地址就会退回到 DHCP 服务器的地址池中等待重新分配，它有可能会分配给另外的主机。因此处于动态 DHCP 分配作用下的主机的 IP 地址不是固定的。

DHCP 还可以帮客户端指定 网关、子网掩码、DNS 服务器、WINS 服务器等项目，因此在客户端方面，除了将 DHCP 选项打勾之外，几乎无需做任何的 IP 环境设置。

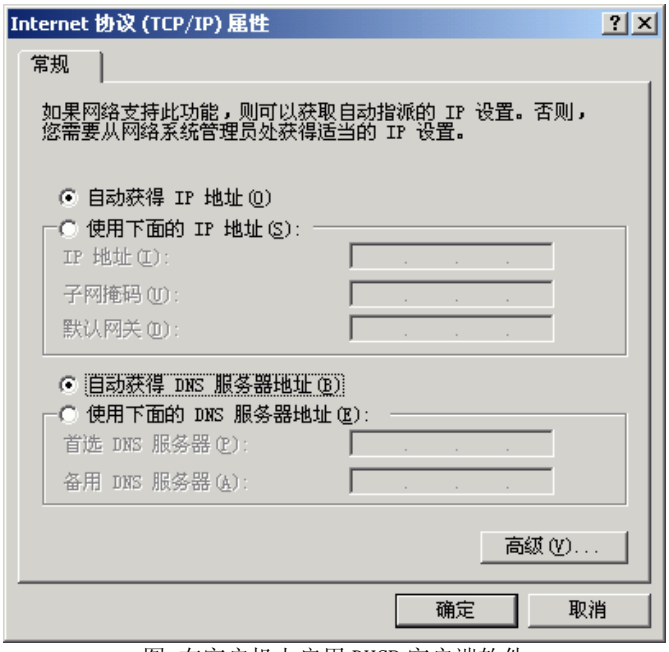


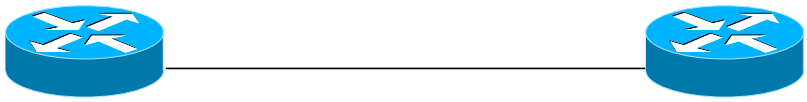
图 在客户机上启用 DHCP 客户端软件

Cisco 路由器中的 DHCP 配置

在 Cisco 的 IOS 软件中配置 DHCP 比 Windows2000 要复杂一些。

启用 Cisco IOS 的 DHCP 客户端软件

首先，我们讲讲如何在 Cisco IOS 中启用 DHCP 客户端软件，使路由器的接口可以从 DHCP 服务器自动获取 IP 地址。




```
Router#conf t
Router(config)#interface ethernet 0/0
Router(config-if)#ip address dhcp
Router(config-if)#no shutdown
*Mar  1 08:36:49.640: %LINK-3-UPDOWN: Interface Ethernet0/0, changed state to up
*Mar  1 08:36:50.642: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/0, changed
state to up
*Mar  1 08:37:01.704: %DHCP-6-ADDRESS_ASSIGN: Interface Ethernet0/0 assigned DHCP address
10.21.2.239, mask 255.255.0.0, hostname Router

Router(config-if)#
```

从上面的例子中我们可以看到：在接口配置模式下使用 ip address dhcp 命令，并且启用接口以后，DHCP 服务器为路由器的 Ethernet0/0 口分配了 10.21.2.239/16 的 IP 地址。

【注意】如果读者不理解这里提到的 IOS 操作命令，可以参阅本书第四章。

将 Cisco 路由器配置为 DHCP 服务器

下面先举个例子来说明如何将 Cisco 路由器配置为 DHCP 服务器，然后我们对例子中的命令进行解释。

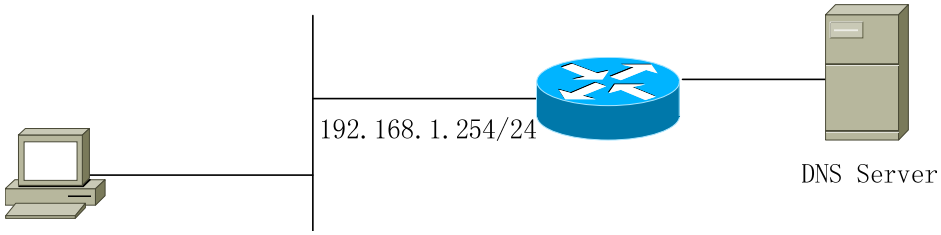


图 配置 DHCP 服务器

```
Router#conf t
Router(config)#ip dhcp pool test
Router(dhcp-config)#domain-name test
Router(dhcp-config)#network 192.168.1.0 255.255.255.0
Router(dhcp-config)#default-router 192.168.1.254
Router(dhcp-config)#dns-server 202.102.11.141
Router(dhcp-config)#lease 8 0 0
Router(dhcp-config)#exit
Router(config)#ip dhcp excluded-address 192.168.1.240 192.168.1.254
```

下面是对上面使用过的命令的解释：

命令	解释
ip dhcp pool pool-name	定义地址池，pool-name 为地址池的名称，此例中为“test”。
domain-name domain-name	定义 DHCP 作用域的域名，此例中为“test”。（可选）
network network-number [mask|/prefix-length]	定义地址池中的 IP 地址范围，此例中网络地址为 192. 168.1.0；子网掩码为 255.255.255.0，也可以写为/24。
default-router ip-address	定义客户机的默认网关，此例中为 192.168.1.254。
dns-server ip-address	定义 DNS 服务器地址，此例中为 202.102.11.141。
Lease {days [hours][minutes] | infinite}	定义 IP 地址租借期，此例中的“8 0 0”代表 8 天 0 小时 0 分钟。命令中的 infinite 参数代表永不过期，即租借期为无限长。
ip dhcp excluded-address low-address [high-address]	定义要从地址池中排除的地址范围，此例中为 192.168.1. 240～192.168.1.254。

我们也可以在 Cisco 路由器的 DHCP 服务器上定义静态的 DHCP 映射条目，这些静态映射一般用于网络中的服务器或者固定主机。

为了定义静态 DHCP 映射条目，我们必须为每条静态映射单独地定义一个的地址池，然后在新地址池中定义 IP 地址和 MAC 地址。

```
Router(config)#ip dhcp pool static1
Router(dhcp-config)#host 192.168.1.250 /24
Router(dhcp-config)#hardware-address 2222.2222.2222
```

下面是对以上命令的解释：

命令	解释
ip dhcp pool pool-name	定义地址池，pool-name 为地址池的名称，此例中为“test”。
host address [mask | /prefix-length]	定义固定主机的 IP 地址和子网掩码，本例中 IP 地址为 192.168.1.250，子网掩码为 255.255.255.0（/24）。
hardware-address hardware-address [type]	定义固定主机的 MAC 地址，本例中 MAC 地址为 2222.2222. 2222。后面的 type 可选参数用于定义网络类型，可以是 ethernet、ieee802 或者是相应的 RFC 文档号码。

检验 DHCP 服务器的配置

使用 show running-config 命令显示 DHCP 服务器的配置，下面是一个输出样例。

```
Router#show run
Building configuration...
Current configuration : 925 bytes
!
version 12.3
……….
ip dhcp excluded-address 192.168.1.240 192.168.1.250
!
ip dhcp pool test
   network 192.168.1.0 255.255.255.0
   default-router 192.168.1.254
   dns-server 192.168.1.253
   domain-name test
   lease 8
!
ip dhcp pool static1
   host 192.168.1.254 255.255.255.0
   hardware-address 2222.2222.2222
……….
end
Router#
```

使用 show ip dhcp server statistics 命令可以查看 DHCP 服务器的当前状态，例如：DHCPDISCOVER 等消息收发情况。

```
Router#show ip dhcp server statistics
Memory usage         15121
Address pools        2
Database agents      0
Automatic bindings   0
Manual bindings      1
Expired bindings     0
Malformed messages   0
Secure arp entries   0

Message              Received
BOOTREQUEST          0
DHCPDISCOVER         0
DHCPREQUEST          0
DHCPDECLINE          0
DHCPRELEASE          0
DHCPINFORM           0

Message              Sent
BOOTREPLY            0
DHCPOFFER            0
DHCPACK              0
DHCPNAK              0
```

子网规划部分

数据包在网络上传输时，数据包的目的 IP 地址是始终不变的，路由器的任务就是：根据自己的路由表选择到目标 IP 地址最佳路径的出口，然后重写数据包第二层的帧头（MAC 地址），让数据包发往下一跳。不变的 IP 地址是将数据包正确发往目的地的基础。

目前广泛应用的 IP 版本为：IPv4，它使用 32 位的二进制地址，每个地址由四个八位组构成，每个八位组被转换成十进制并用“.”来分割，即常说的“点分十进制表示法”。如下图所示：

32位

二进制	11000000	10101000	00001010	00000001
十进制	192	168	10	1

点分十进制　192.168.10.1

网络地址与主机地址

每一个利用 TCP/IP 通信的主机都需要一个唯一的 IP 地址，IP 地址都被分成网络地址和主机地址两部分，这种寻址策略有些类似街道（网络地址）和门牌号（主机地址）。如下图：

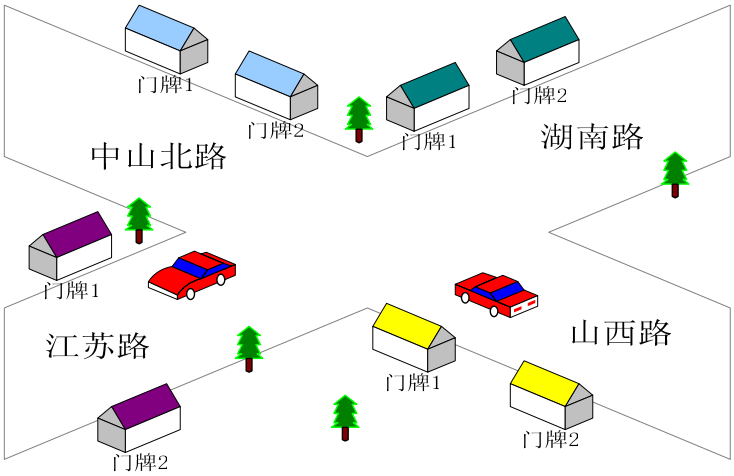


图　街道地址与门牌号

图中，每个房屋的确切地址都由自己的街道和门牌号来共同决定，对于整个城市来说，这个街道和门牌号都是唯一的，我们可以通过街道和门牌号找到自己的住所。例如：当我们要去往某处（江苏路 1 号）时，我们应先找到该处所在的街道（江苏路），然后再依靠门牌号寻找确切的地址（1 号）。

IP 地址和街道地址与门牌号的作用相似。IP 地址中的网络地址就好比街道地址，用来标识整个网段；主机地址就好比门牌号，用来标识某个确切的主机。如下图：

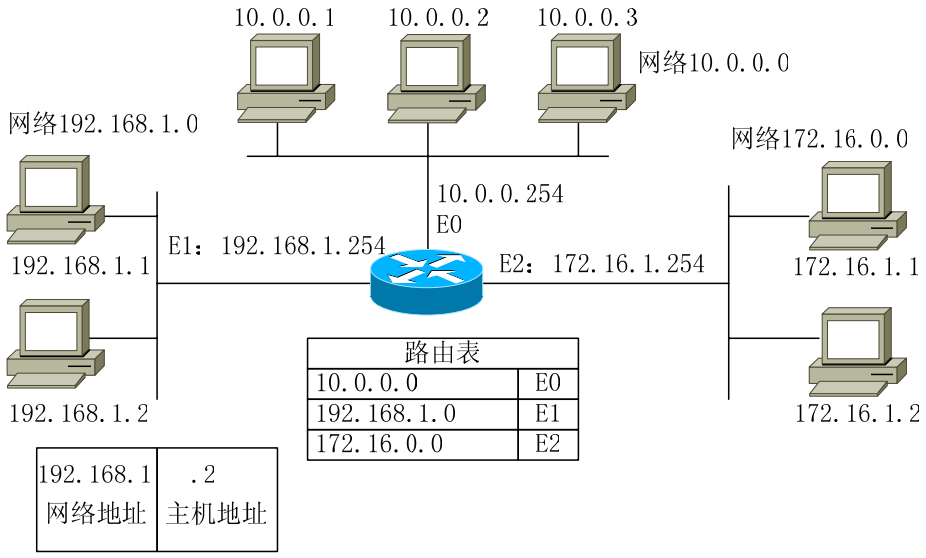


图 3 网络地址与主机地址

对于一个 IP 地址，外界只看它的网络地址，不关心其内部的网络结构。当外界要向某个主机发送数据包时，它只看主机 IP 地址中的网络地址，当数据包到达目标主机所在网段之后，再依靠主机地址把数据包发给目标主机。列如：路由器将对所有网络地址为 192.168.1.0 的数据包作同样处理。当路由器收到一个发往 192.168.1.2 的数据包时，它查询自己的路由表，发现通过 E1 口可以到达 192.168.1.0 的网络，于是路由器会直接将数据包发往 E1 口，而不关心目标 IP 地址的主机地址具体是多少；数据包到达 E1 口以后，路由器发现 E1 口本地连接着 192.168.1.0 的网络，于是它查找 ARP 表得到主机 192.168.1.1 的 MAC 地址，最后将 IP 数据包发送给目标主机。

IP 地址分类

IPv4 是基于 32 比特的地址方案，理论上可以支持 40 多亿台主机。为了适应不同的网络需求，IPv4 地址被分成五类，其分配由因特网地址授权委员会（IANA）统一管理。

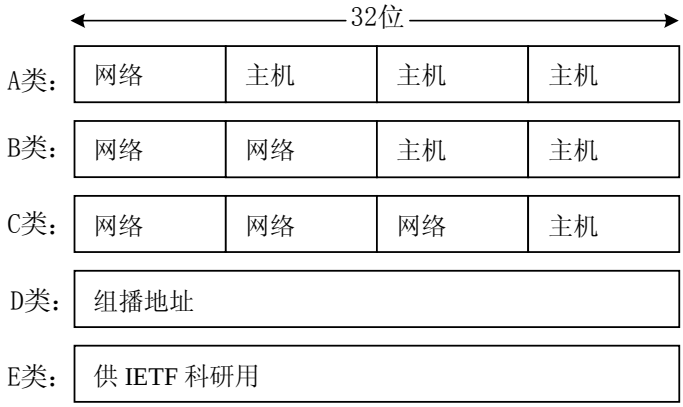


图　IP 地址分类

五类 IP 地址的前三类（A，B，C）被用来作全球唯一的单播地址；D 类和 E 类地址为组播和试验目的保留。【注意】目前，全球有 3 个区域因特网注册机构负责为 ISP 和组织分配成块的 IP 地址。其中美国因特网地址注册机构（ARIN）为北美洲、中美洲和南美洲提供服务；欧洲网络信息中心（PIPE NCC）为欧洲和非洲提供服务；亚太网络信息中心（APNIC）为亚洲地区提供服务。如果想得到关于这三个注册机构的详细信息，请登陆 www.arin.com、www.ripe.com、www.apnic.com。

子网掩码

网络设备如何区分网络地址和主机地址呢？这里就要引出子网掩码这个概念。子网掩码实际上是一个过滤码，将 IP 地址和子网掩码“按位求与”就可以过滤出 IP 地址中应该作为网络地址的那一部分。按位求与就是将 IP 地址中的每一位和相应的子网掩码位进行与运算。与运算的规则如下：

```
1 & 1=1
1 & 0=0
0 & 0=0
```

每一类 IP 地址都有缺省的子网掩码，A 类地址的子网掩码为 255.0.0.0，B 类地址的子网掩码为 255.255.0.0，C 类地址的子网掩码为 255.255.255.0。下面我们用 IP 地址 172.16.1.1 255.255.0.0 举例

说明子网掩码的作用：

	网络	网络	主机	主机
主机 172.16.1.1	10101100	00010000	00000001	00000001
子网掩码 255.255.255.0	11111111	11111111	00000000	00000000
与运算 172.16.0.0	10101100	00010000	00000000	00000000

图　缺省子网掩码

上例中经过与运算后被过滤出来的 172.16.0.0 就是 172.16.1.1 的网络地址。

通常情况下，在 IP 地址后面加“/n”来表示一个具体的 IP 地址。（n 是子网掩码中“1”的个数，如：子网掩码“255.255.255.0”，通常写成 “/24”）

A 类地址

A 类地址的缺省子网掩码是：255.0.0.0，它使用 IP 地址中的第一个八位组表示网络地址，其余三个八位组表示主机地址。A 类地址的结构使每个网络拥有的主机数非常多，因此 A 类地址是为巨型网络所设计的。A 类地址的第一个八位组的第一位总是被设置为 0,这就限制了 A 类地址的第一个八位组的值始终小于 127，也就是说仅有 127 个可能的 A 类网络。如下图：

0XXXXXXX	主机	主机	主机

（0～127）

图　A 类地址

实际上,A 类地址的范围是 1～126。虽然从理论上讲,127.x.x.x 和 0.x.x.x 也属于 A 类地址,但是 127.x.x.x 已经被保留作回路测试之用，网络 0.0.0.0 也保留用于广播地址（未知网络），所以它们不能分配给任何网络。

因为有三个八位组用于表示主机地址，所以每个 A 类网络的主机数的可能值是 16777216（2^{24}），但是由于全 0 的主机地址表示网络、全 1 的主机地址表示到这个网络的定向广播，所以实际的主机数比可能的主机数少 2。

【注意】主机地址的运算方法是：2^N-2（N 是主机部分的位数，例如：A 类地址就是 24）

B 类地址

B 类地址的缺省子网掩码是 255.255.0.0，B 类地址使用前两个八位组表示网络地址，后两个八位组表示主机地址。设计 B 类地址的目的是支持中到大型网络。

B 类地址的第一个八位组的前两位总是被设置为 10，所以 B 类地址的范围是从 128.0.0.0 到 191.255.0.0，如下图所示：

10XXXXXX	XXXXXXXX	主机	主机

（128～191）

图　B 类地址

B 类地址可能拥有的网络数是 16384（2^{14}）（实际上要减去两个特例），每个网络可能拥有的主机数是 65536（2^{16}）。

C 类地址

C 类地址的子网掩码是：255.255.255.0，C 类地址使用前三个八位组表示网络地址，最后一个八位组表示主机地址。设计 C 类地址的目的是支持大量的小型网络，因为这类地址拥有的网络数目很多，而每个网络所拥有的主机数却很少。

C 类地址的第一个八位组的前三位总是被设置为 110，所以 B 类地址的范围是从 192.0.0.0 到 223.255.255.0，如下图所示：

110XXXXX	XXXXXXXX	XXXXXXXX	主机

（192～223）

图　C 类地址

C 类地址可能拥有的网络数是 2097152（2^{21}），每个网络可能拥有的主机数是 256（2^8）。

D 类地址

D 类地址用于 IP 网络中的组播（多点广播）。它不像 A、B、C 类地址有网络号和主机号，一个组播地址标识了一个 IP 地址组。因此可以同时把一个数据流发送到多个接收端，这比为每个接收端创建一个数据流的流量小得多，它可以有效地节省网络带宽。

D 类地址的第一个八位组的前四位总是被设置成 1110，所以 D 类地址的范围是从 224.0.0.0 到 239.255.255.255，如下图所示

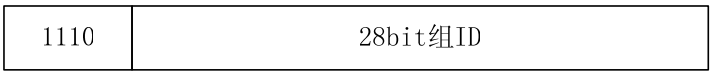


图　D 类地址

D 类地址拥有 268435456 个组（2^{28}），任何主机都可以自由的加入或离开任何组。

【注意】多播地址没有所谓的子网掩码。

E 类地址

E 类地址虽然被定义，但却为 IETF 保留作研究之用，因此 Internet 上没有可用的 E 类地址。

E 类地址的第一个八位组的前 4 位恒为 1，因此有效的地址范围从 240.0.0.0 到 255.255.255.255，如下图所示：

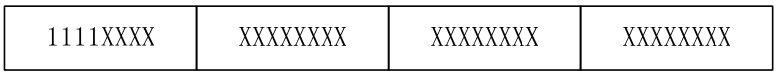


图 E 类地址

IP 寻址基础

刚刚在讲到网络地址和主机地址时，本书已经提到了一些 IP 寻址的内容，也许读者对其还有一些困惑，现在我们就来对 IP 寻址进行详细的讨论。

我们知道当某主机发送 IP 数据包给与其处于同一本地 IP 子网的另一台主机时，它只要直接把 IP 数据包送到本地网络上，然后对方就能收到。如下图：

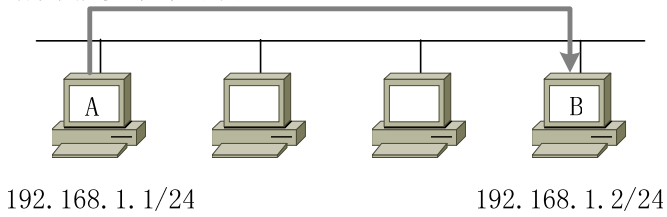


图 本地通信

当处于不同 IP 子网间的主机需要通信时，主机会先把 IP 数据包发送送给一个称为“缺省网关（default gateway）”的路由器上，然后由这个路由器把 IP 数据包送到目的地。 “缺省网关”是 TCP/IP 一个配置参数，它是处于本地网络上的某个路由器接口的 IP 地址。

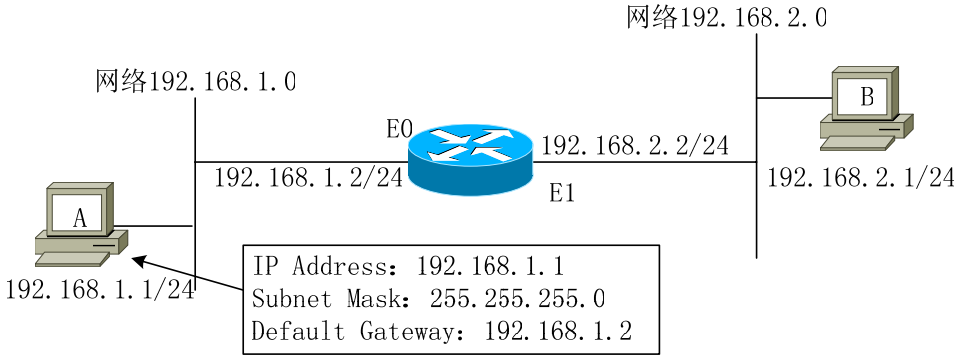


图 路由器可以将数据包从一个网段发送到另一个网段。

路由器转发 IP 数据包时，它只根据 IP 数据包目标 IP 地址的网络部分选择合适的接口送出数据包。路由器还要判断接口所连接的是否是目标子网，如果是，路由器就直接把数据通过接口送到网络上（例如上图中，主机 A 发往主机 B 的数据包到达路由器时，路由器发现它的 E1 口直接连接着目标网络，于是它将数据直接转发给主机 B）；否则，路由器会选择下一跳路由器来传输数据，这样一跳跳地传输，IP 数据包最终将被送到目的地。

路由器给人的感觉是一种复杂的设备，实际上从宏观的角度来看，路由器主要实现两样基本功能：

- 交换和转发功能，将数据从路由器的进入接口穿过路由器再送到外出接口。
- 路由功能，即寻址，学习和维护路由选择表，决定正确的转发路径。

寻址由路由选择算法来实现，为了判定最佳路径，路由选择算法维护着一个包含路由信息的路由选择表，路由选择表可以将目标网络和下一跳（nexthop）的关系告诉路由器。当路由器向目的地转发数据包时，它

就是通过这张路由选择表来确定所要使用的输出接口。
为了能够路由数据包，路由器需要知道以下信息：

- 目标地址
- 下一跳地址

下面我们来看一个更复杂一点的网络，我们知道：当一个数据包到达路由器时，路由器查看该数据包的目标 IP 地址的网络地址，然后从路由表中查询与目标网络地址最匹配的项。

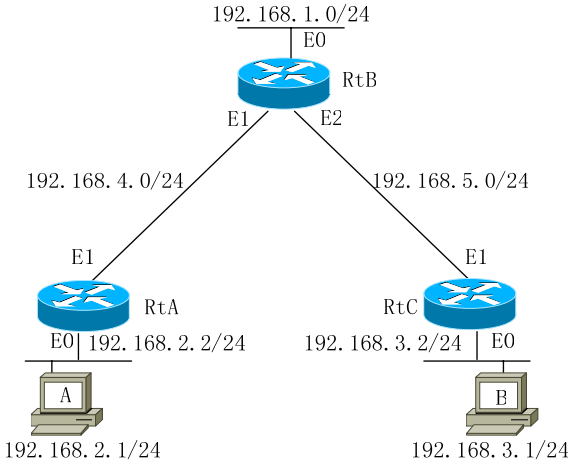


图　简单网络

上图中，假设主机 A 需要和主机 B 通信，数据传输的过程如下：
①　主机 A 先将数据发送给它的默认网关（路由器 A）。下面是主机 A 发送的数据帧的格式：

目标MAC地址	源MAC地址	目标IP地址	源IP地址	
RtA的E0口的MAC地址	主机A的MAC地址	192.168.3.1	192.168.2.1	数据……

主机 A 发送的数据帧中，以路由器 A 的 E0 口的 MAC 地址为目标 MAC 地址，而目标 IP 地址依然为主机 B 的 IP 地址。
②　路由器 A 收到主机 A 的数据后，它查询自己的路由表，它发现通过 E0 口可以到达主机 B 所在的网络（192.168.3.1）。路由器 A 的路由表如下：

网络号	出口
192.168.1.0	E1
192.168.2.0	E0
192.168.3.0	E1
192.168.4.0	E1
192.168.5.0	E1

③　路由器 A 重写数据包的帧头信息，将目标 MAC 地址改为路由器 B 的 E1 口的 MAC 地址，同时修改源 MAC 地址，然后将数据包从自己的 E1 口发送出去。下面是路由器 A 发送的数据帧的格式：

目标MAC地址	源MAC地址	目标IP地址	源IP地址	
RtB的E1口的MAC地址	RTA的E1口的MAC地址	192.168.3.1	192.168.2.1	数据……

④　数据被送到路由器 B 后，路由器 B 按照相同的方法把它送给路由器 C。路由器 B 的路由表如下：

网络号	出口
192.168.1.0	E0
192.168.2.0	E1
192.168.3.0	E2
192.168.4.0	E1
192.168.5.0	E2

⑤　路由器 C 发现 192.168.3.0 网络直接连接在它的 E0 口上，于是它将数据包发往 E0，在 E0 口上采用 ARP 来查询 192.168.3.1 的 MAC 地址，最后将数据包发往目的地。

代理 ARP

如果网络上所有的系统都配置成目的地均在本地网段，路由器将扮演远程系统代理的角色。如下图：

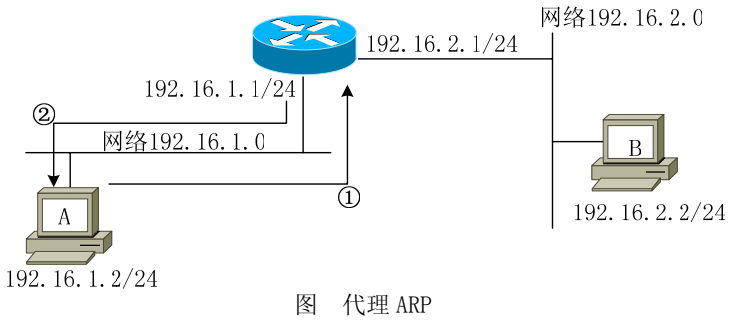


图　代理 ARP

①　主机 A：192.16.1.2/24 的试图与主机 B：192.16.2.2/24 进行通信，但是它不知道主机 B 的 MAC 地址，由于主机 A 被配置为目的地在本地网段，所以主机 A 直接发出 ARP 请求。(实际上运行代理 ARP 的路由器就充当默认网关的角色)

②　路由器回答来自主机 A 的 ARP 请求，它使用自己的 MAC 地址回答主机 A。于是主机 A 将数据包发送给路由器，路由器再将数据包转发到正确的目的地。

代理 ARP 可以简化主机的管理，不过却增加了网络的通信量，并且在路由器上需要较大的 ARP 缓存，因为每个不在本网的 IP 地址都将被创建一个映射表项。在使用代理 ARP 的主机看来，世界就像一个大的没有路由器物理网络。

IP 地址的几种特殊情况

IP 地址中的某些地址为特殊目的保留，规则大致如下：

网络地址和主机地址的特殊情况

网络地址部分不能设置为全 0 或全 1。当 IP 地址为 0.0.0.0 时，它代表一个未知网络。当 IP 地址为 255.255.255.255 时，它代表面向本地网络所有主机的广播，我们称为“泛洪广播”。路由器不会转发泛洪广播。

当主机地址部分全 0 时，它代表整个网段，例如：192.168.1.0。当主机地址部分全为 1 时，它代表一个面向这个网络的定向广播，例如：192.168.1.255，这个广播消息会发送给 192.168.1.0 网段的所有主机。

【注意】在 Cisco IOS 12.0 版本和其后的版本中，“no ip directed-broadcast”命令时缺省启用的，也就是说路由器会丢弃所有的定向广播，这对有效利用带宽时有利的。我们也可以使用“ip directed broadcast”启用定向广播。

公有地址和私有地址

公有地址（Public address）由因特网地址授权委员会（IANA）负责分配，使用这些公有地址可以直接访问因特网。

私有地址（Private address）属于非注册地址，专门为组织机构内部使用。

下表列出用于内部寻址的地址：

表 内部地址

类别	网络号	地址范围
A 类	10.0.0.0/8	10.0.0.0～10.255.255.255
B 类	172.16.0.0/12	172.16.0.0～172.31.255.255
C 类	192.168.0.0/16	192.168.0.0～192.168.255.255

为什么会有私有地址和共有地址的概念呢？这是为了减缓 IP 地址的耗尽及减少 Internet 路由表条目的数量而提出的一种解决方案。

当被设置成私有地址的主机要访问 Internet 时，就要经过网络地址转换(NAT)将私有地址转换为公有地址；同时这些主机从因特网上接收数据包时，也要将公用地址转换为私有地址。关于 NAT 技术将会在后面的章节中专门讲述。

回路地址

127.x.x.x 分配给回路地址，我们可以用它测试 TCP/IP 配置，因为它不需要任何网络连接。

在排除网络故障时，我们可以用回路地址来测试 TCP/IP 协议是否正常。如下图：

```
c:\>ping 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:

Reply from 127.0.0.1:byte=32 time<10ms TTL=64
Reply from 127.0.0.1:byte=32 time<10ms TTL=64
Reply from 127.0.0.1:byte=32 time<10ms TTL=64
Reply from 127.0.0.1:byte=32 time<10ms TTL=64

Ping statistics for 127.0.0.1:
    Packets: Sent = 4,Received = 4, Lost = 0 <0% loss>,
```

```
Approximate round trip time in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

由于 127.0.0.1 是回路地址，所以如果 TCP/IP 工作正常的话，这个地址始终是能 ping 通的。

子网划分

本节讲述子网划分的概念，主要涉及到以下几个方面：

- 子网划分的概念
- 为什么要进行子网划分
- 如何进行子网划分
- VLSM

子网划分概念

前面我们介绍了每类 IP 地址都有自己的缺省子网掩码，但是这些缺省子网掩码并不能让我们有效的利用 IP 地址。如果我们在缺省子网掩码后面再添加几位连续的“1”，使掩码中的全“1”位变得更长，会出现什么情况呢？

如下图：我们将 B 类网络 172.16.0.0/16 的子网掩码延长为 24 位，这样就可以“创造”出更多的新网络，不过每个网络所拥有主机数还是减少了。这种被额外创建的网络就是子网。

	网络	网络	主机	主机
网络 172.16.0.0	10101100	00010000	00000000	00000000

	网络	网络	网络	主机
子网掩码 255.255.255.0	11111111	11111111	11111111	00000000
新网络：172.16.1.0	10101100	00010000	00000001	00000000
172.16.2.0	10101100	00010000	00000010	00000000
			子网	

图 A　　对 B 类地址划分子网

为什么要进行子网划分

Internet 的发展快得让人难以置信，当前 IP 地址已经面临即将耗尽的挑战。如果不对 IP 地址进行子网划分，我们很可能浪费许多 IP 地址。

一个 B 类网络可以寻址 65534 个系统，但是对于一个单一的局域网而言，它的地址实在太多了。如下图：

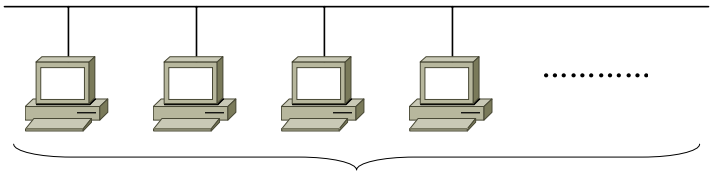


图　一个不切实际的网络

下图中，我们使用子网掩码 255.255.255.0 将 B 类网络 172.16.0.0/16 分成了许多更小的网络，这样可以 IP 地址的使用将更加有效。

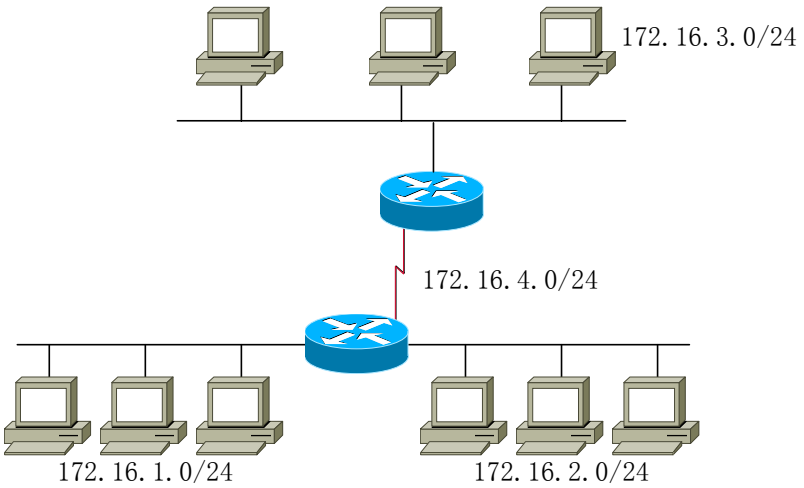


图　子网规划

这个例子中，网络 172.16.0.0 被划分成许多子网：172.16.1.0、172.16.2.0、172.16.3.0、172.16.4.0……划分子网的代价是减少了每个网络所能拥有的主机数目，但是这样却额外地“创造”出了更多的新网络。【注意】也许大家有个困惑：172.16.0.0/24 算不算一个子网呢？在 Cisco IOS 12.0 以前的版本中，缺省情况是不支持这种子网 0（所有子网地址部分为 0）的，必须依靠“ip subnet-zero”命令启用子网 0，不过在 IOS12.0 以后的版本中就不存在这个问题了。注意：其他厂商的设备有可能不支持子网 0。

也许读者还没有感觉到子网划分的意义何在，我们再举一个例子：在下图点到点的广域网链路中，每个网段仅需两个地址。如果我们在两台点对点连接的路由器之间使用 192.168.10.0/24 这个 C 类网络，那么我们将只使用了 192.168.10.1 和 192.168.10.2，其余的 252 个可用 IP 地址就全部浪费了。

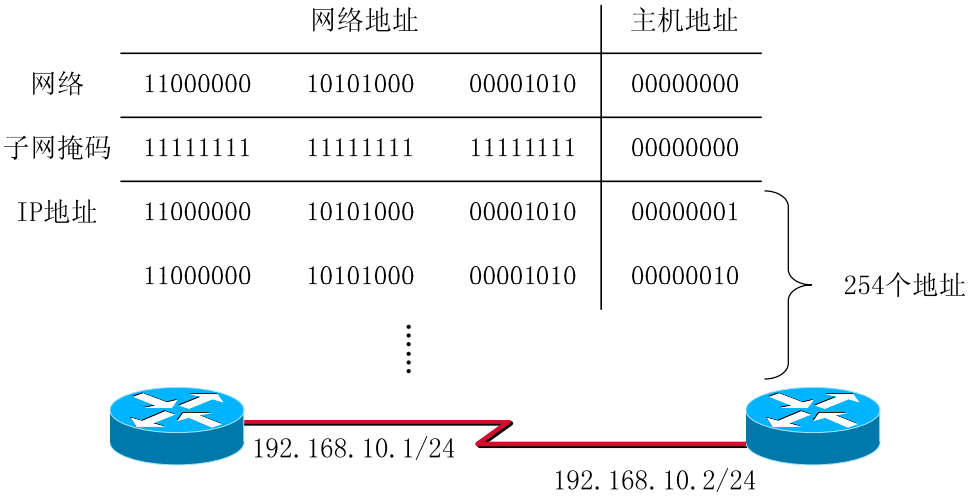


图 9 IP 地址的不合理规划

两台点对点连接的路由器之间只需要两个 IP 地址，其实，我们完全可以为它们定义一个只有两台主机的子网，这样就不会造成其余 252 个地址的浪费了。如下图：

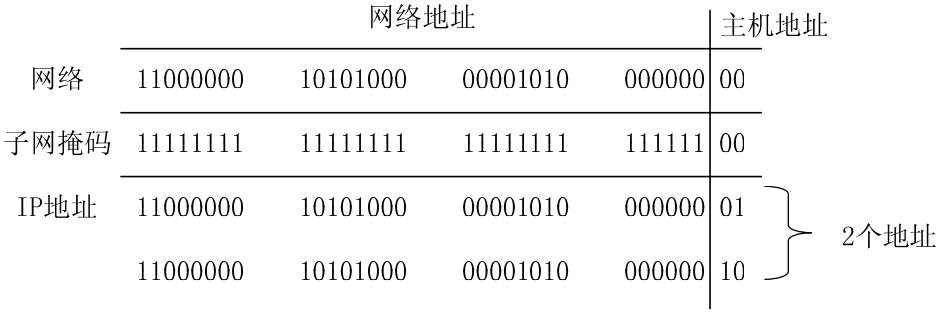


图 10 合理地规划 IP 地址

在上面的例子中，我们使用 192.168.10.0/30 这个子网对两台路由器间的链路进行 IP 地址分配，这个子网恰好只能容纳两台主机。这样就充分利用了 IP 地址，使得剩余的 IP 地址可以发挥应有的作用。

划分子网的目的之二是限制通信流量，我们知道网络主机能和与其处于相同网段上的其他主机进行通信，假如一个 A 类地址不进行子网划分，一台主机对这个网段发送一个定向广播，那么成千上万台主机将接收到这个定向广播，这种网络是绝对低效的。只有划分子网后用路由器去隔离广播，才能让网络发挥最好的性能。

子网规划的运算

首先我们给出进行子网规划时所运用到的公式：

计算创建的子网数量：划分子网后，新创建的子网数量=2^N（N 是缺省掩码被扩展的位数）。新建的子网中包含子网 0 和子网 1。 计算每个子网能容纳的主机数：2^N-2（N 是主机地址的可用位数）。

下面我们使用一个网络规划的例子来讨论子网划分的计算方法：

某公司欲在对其内部网络进行 VLAN 划分。他们决定采用 C 类网络：192.168.1.0/24，在这个网络中划分出 5 个子网，分别用于不同的 VLAN，每个 VLAN 能容纳的主机数为 20～25。

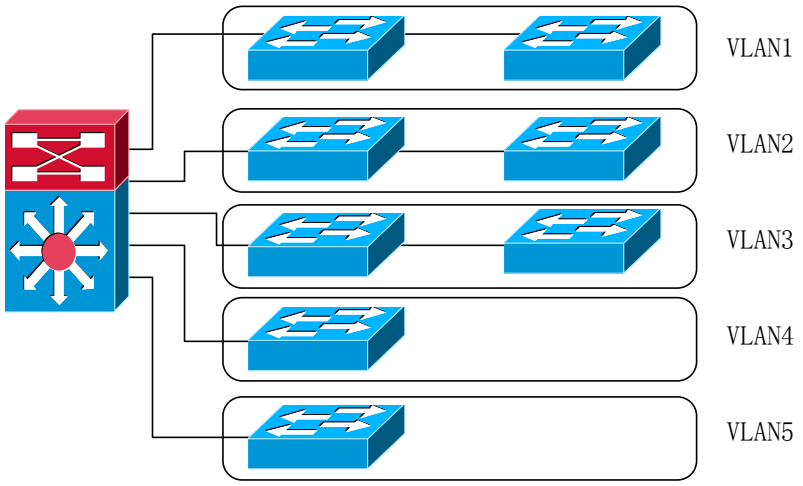


图　子网规划举例

①　确定子网位。

假设创建子网的数量=2^N（N 是缺省子网掩码被扩展的位数），则：$2^2<5<2^3$。

如果将掩码位扩展 2 位，则可以创建 4（2^2）个子网（其中包括子网 0 和子网 1），这显然是不符合要求的。因此我们将掩码位向后扩展 3 位，这样我们可以创建出 8（2^3）个子网。C 类网络的缺省掩码为 24 位长，因此新的子网掩码长度为 27 位。

②　验证主机位。

在使用这个子网掩码前，我们需要验证一下它是否满足每个子网所需的主机数目。使用 3 个子网位后，剩下的主机位数为 5，因此每个子网可以拥有 $2^5-2=30$ 台主机，这个数字足以满足该公司的需要。

③　确定子网的地址

我们知道最小的子网 ID 是 0，即子网 0，例如这个例子中通过子网划分，我们得到的第一个子网是 192.168.1.0/27，下图可以清楚地解释如何得到后面的子网 ID。

	192 . 168 . 1 . 0					
192.168.1.0/24	11000000	10101000	00000001	000	00000	原网络地址
255.255.255.224	11111111	11111111	11111111	111	00000	新子网掩码
192.168.1.0/27	11000000	10101000	00000001	000	00000	子网0
192.168.1.32/27	11000000	10101000	00000001	001	00000	
192.168.1.64/27	11000000	10101000	00000001	010	00000	
192.168.1.96/27	11000000	10101000	00000001	011	00000	
192.168.1.128/27	11000000	10101000	00000001	100	00000	
192.168.1.160/27	11000000	10101000	00000001	101	00000	
192.168.1.192/27	11000000	10101000	00000001	110	00000	
192.168.1.224/27	11000000	10101000	00000001	111	00000	最后一个子网
	缺省子网掩码位			子网ID		

图 计算子网 ID

【注意】其他参考书上可能认为划分子网时子网数目要减 2，因为他们认为全“0”和全“1”的子网是非法的。事实上，在相应的 RFC 文档中，已经承认了全“1”的子网，并且子网 0 也可以在 Cisco 路由器中启用。

④ 确定每个子网的主机地址

接下来的任务是要确定子每个网的主机地址，下图可以清楚的解释怎样确定主机地址，我们使用这个例子中的 192.168.1.32/27 子网进行举例。

	192 . 168 . 1 . 32					
192.168.1.32/27	11000000	10101000	00000001	001	00000	子网
255.255.255.224	11111111	11111111	11111111	111	00000	新子网掩码
192.168.1.33/27	11000000	10101000	00000001	001	00001	第一个主机
⋮	⋮	⋮	⋮		⋮	
192.168.1.62/27	11000000	10101000	00000001	001	11110	最后一个主机
192.168.1.63/27	11000000	10101000	00000001	001	11111	子网广播
	缺省子网掩码位			子网ID	主机ID	

图 计算主机地址空间

最后给出子网设计和地址分配。

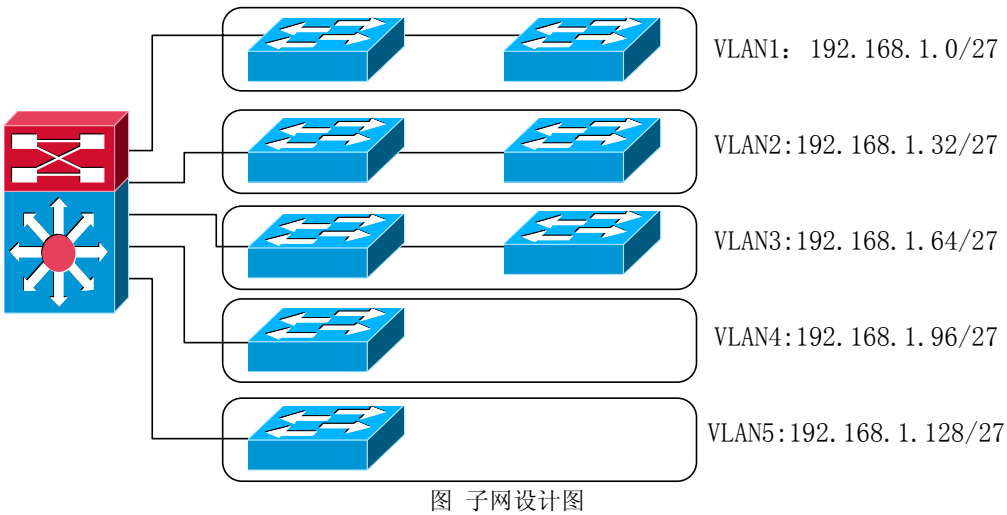


图 子网设计图

为方便读者进行网络设计，我们在此给出 A、B、C 类子网表。

表 A 类子网表

子网 ID 位数	子网掩码	子网数目	有效主机数目
2	255.192.0.0	4	4194302
3	255.224.0.0	8	2097150
4	255.240.0.0	16	1048574
5	255.248.0.0	32	524286
6	255.252.0.0	64	262142
7	255.254.0.0	128	313070
8	255.255.0.0	256	65534
9	255.255.128.0	512	32766
10	255.255.192.0	1024	16382
11	255.255.224.0	2048	8190
12	255.255.240.0	4096	4094
13	255.255.248.0	8192	2046
14	255.255.252.0	16384	1022
15	255.255.254.0	32768	510
16	255.255.255.0	65536	254
17	255.255.255.128	131072	126
18	255.255.255.192	262144	62
19	255.255.255.224	524288	30
20	255.255.255.240	1048576	14
21	255.255.255.248	2097152	6
22	255.255.255.252	4194304	2

表 B 类子网表

子网 ID 位数	子网掩码	子网数目	有效主机数目
2	255.255.192.0	4	16382
3	255.255.224.0	8	8190
4	255.255.240.0	16	4094
5	255.255.248.0	32	2046
6	255.255.252.0	64	1022
7	255.255.254.0	128	510
8	255.255.255.0	256	254
9	255.255.255.128	512	126
10	255.255.255.192	1024	62
11	255.255.255.224	2048	30
12	255.255.255.240	4096	14
13	255.255.255.248	8192	6
14	255.255.255.252	16384	2

表 C 类子网表

子网 ID 位数	子网掩码	子网数目	有效主机数目
2	255.255.255.192	4	62
3	255.255.255.224	8	30
4	255.255.255.240	16	14
5	255.255.255.248	32	6
6	255.255.255.252	64	2

【注意】子网数目中包含了子网 0 和子网 1。

VLSM

通过前面的介绍，我们知道可以通过子网划分将一个主类网络划分为许多小网络，这些小网络是否可以继续划分为更小的网络呢？例如：上一节中，我们将 192.168.1.0/24 用 27 位长的子网掩码划分成几个小网络，其实我们也可以再对其中的某个小网络（例如：192.168.1.160/27）使用 30 位长的子网掩码进行划分，这样我们又可以得到许多只拥有两个主机位的子网（例如 192.168.1.164/30）。这就是 VLSM。

VLSM 称为可变长度子网掩码，它可以为同一个主类网络提供包含多个子网掩码的能力。VLSM 可以帮助优化可用的地址空间。

下面举一个子网规划的例子。

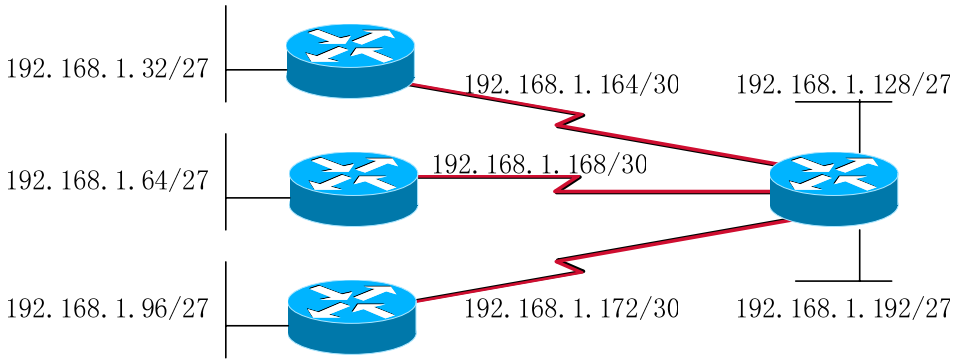


图 使用 VLSM 设计网络

如上图，我们依然使用 192.168.1.0/24 这个 C 类网络进行子网划分，子网划分结构如下：

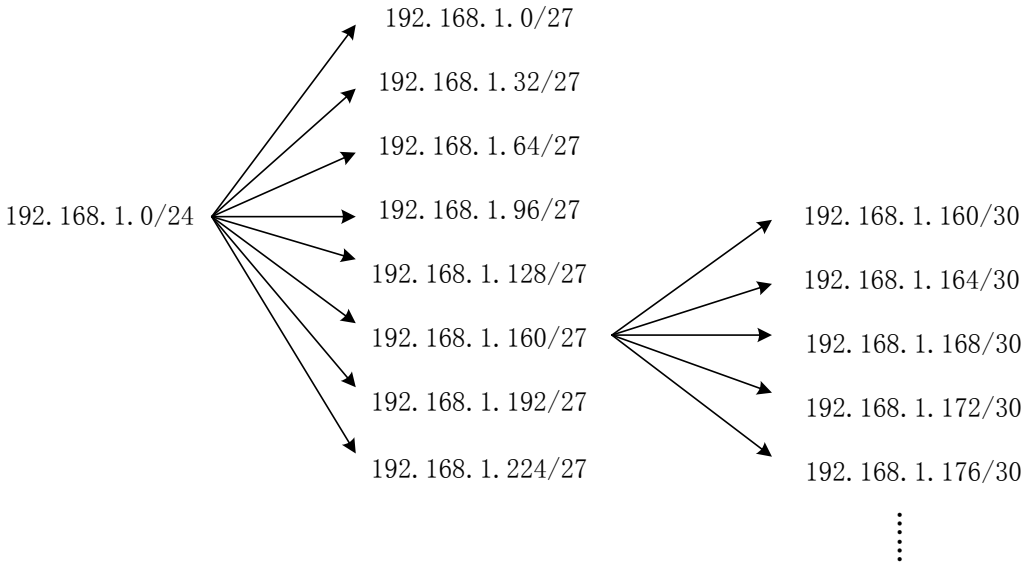


图 子网划分结构图

这个例子中，我们对 192.168.1.0/24 使用 27 位长的掩码进行划分之后，对 192.168.1.160/27 再使用 30 位长的子网掩码进行划分，这些小网络被用于路由器之间的点对点连接链路，有效的利用了 IP 地址空间。