第 5 章

传　输　层

传输层的作用是在通信子网提供的服务的基础上，为上层应用进程提供端到端的服务。传输层是客户程序和服务器程序之间的联络人，是一个进程到进程的连接。它是 Internet 上从一点到另一个点传输数据的端到端逻辑传输媒介。传输层使高层用户在相互通信时不必关心通信子网的实现细节和具体服务质量。传输层是网络体系结构的关键层次。

本章在讨论传输服务的基础上，主要阐述了传输层的两个重要协议：无连接的用户数据报协议 UDP 和面向连接的传输控制协议 TCP，重点是 TCP 为保证可靠传输而采用的连接管理、序号与确认、流量控制、拥塞控制等机制。

5.1　传输层概述

5.1.1　传输层功能及提供的服务

网络层协议提供了主机之间的逻辑通信，而两台主机进行通信实际上是两台主机中的应用进程间互相通信，传输层协议正是为运行在不同主机上的应用进程提供逻辑通信功能。从通信的角度看，传输层属于面向通信部分的最高层，但从用户功能来看，传输层又是用户功能中的最低层。传输层的功能归属如图 5.1 所示。

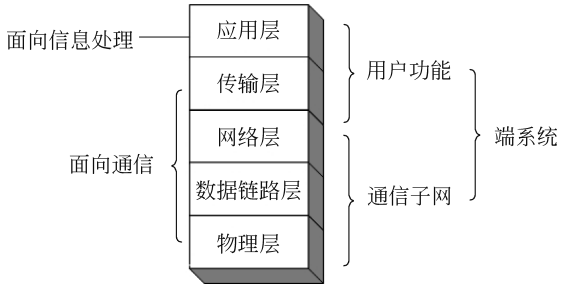

图 5.1　传输层的功能归属

从应用程序的角度看，通过逻辑通信，运行不同进程的主机好像直接相连；实际上这些主机是通过很多路由器和多种不同的链路相连。应用进程通过传输层提供的逻辑通信功能彼此发送报文，而不需要考虑承载报文的物理基础设施的细节，如图 5.2 所示。

一台主机上经常有多个应用进程同时分别与另一台主机上的多个应用进程通信。网

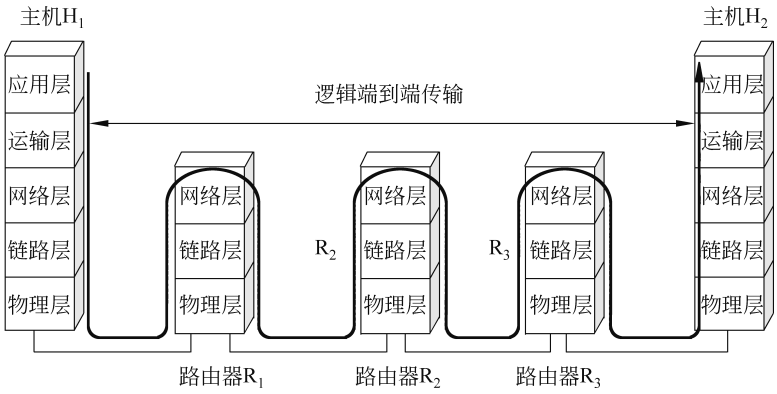

图 5.2 传输层为应用进程间提供逻辑通信

络层协议或网际互连协议能够将分组送达到目的主机，但它无法交付给主机中的某个对应的应用进程，因为在 TCP/IP 协议族中，IP 地址标识的是一台主机，并没有标识主机中的应用进程。因此，网络层是通过通信子网为主机之间提供逻辑的通信；而传输层依靠网络层的服务，在两台主机的应用进程间建立端到端的逻辑通信信道，如图 5.3 所示。

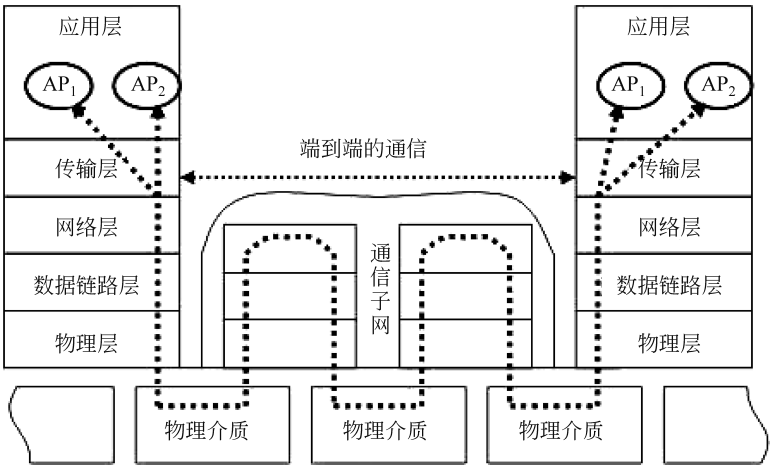

图 5.3 数据通信过程

传输层对整个报文段进行差错校验和检测。IP 在传输数据时并不保证传输的顺序，也不保证传输数据的质量。因此传输层协议要提供端到端的错误恢复与流量控制，对网络层出现的丢包、乱序或重复等问题做出反应。网络层如同邮递员，只负责将邮件从某个单位的收发室发送到另一个单位的收发室，传输层协议就像是将邮件从收发室传递到具体的个人的手中。另外，当上层的协议数据包的长度超过网络层所能承载的最大数据传输单元时，传输层提供必要的分段功能，并在接收方的对等层将分段合并。总之，传输层通过扩展网络层服务功能，为高层提供可靠数据传输，从而使系统之间实现高层资源的共享时不必再考虑数据通信方面的问题，即它是资源子网与通信子网的界面与桥梁，它完成资源子网中两结点间的逻辑通信，实现通信子网中端到端的透明传输。

5.1.2 应用进程、端口号与套接字

1. 应用进程、端口号

在计算机操作系统的术语中，进程即为运行着的程序。操作系统为每个进程分配一定的地址空间和 CPU 周期等资源，从而实现一台计算机上可以并发运行多个程序。传输层协议的首要任务是提供进程到进程通信(Process-to-process Communication)。进程是使用传输层服务的应用层实体。我们在讨论进程到进程通信如何实现之前，需要理解主机到主机通信与进程到进程通信的不同之处。

网络层负责计算机层次的通信(主机到主机通信)。IP 地址标识的是一台主机，网络层协议只把报文传递到目的计算机。然而，这是不完整的传递。报文仍然需要递交给目的主机中正确的进程。这正是传输层接管的部分。传输层协议负责将报文传输到正确的进程。图 5.4 给出了网络层和传输层的范围。其中 AP 表示应用进程。

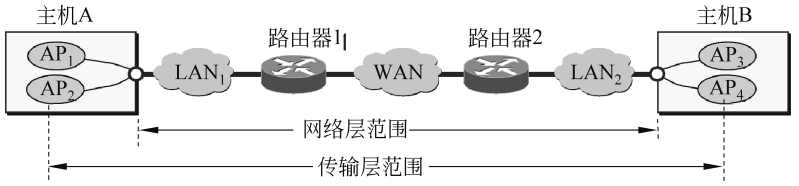

图 5.4 网络层与传输层作用范围

那么两个彼此通信的进程之间是如何相互识别的呢？对通信来说，必须定义本地主机、本地进程、远程主机以及远程进程。使用 IP 地址来定义本地主机和远程主机(见第 4 章)。为了定义进程，我们需要第二个标识符，称为端口号(port number)。两台计算机中的进程要互相通信，不仅需要知道双方的 IP 地址，通过 IP 地址可以找到对方的计算机(类似于学校的地址)，而且还需要知道对方的端口号，其标识了计算机中的应用进程(类似于信箱号)。通过学校地址和信箱号，就可以进行邮政通信了。

所以准确地说，端口号是操作系统为不同的网络应用提供的一个用于区分不同网络通信进程的标识。端口号是 16 位的二进制数，即位于 0～65 535 的整数。每个通信进程产生时都同时被设定一个端口号用来标识该进程，且端口号在同一个操作系统上是唯一的。客户进程向某个服务器请求一种服务时，请求信息中指明服务器某个特定的端口号，服务器便可以将所接收的服务请求提交给对应该端口号的服务进程。客户进程在发送服务请求时，随即也产生一个客户进程端口号，客户端与服务器就这样相互识别进行通信。

端口在传输层的作用有点类似 IP 地址在网络层的作用或 MAC 地址在数据链路层的作用，只不过 IP 地址和 MAC 地址标识的是主机，而端口标识的是网络应用进程。由于同一时刻一台主机上会有大量的网络应用进程在运行，所以需要大量的端口号来标识不同的进程。当传输层收到 IP 层交上来的数据(即 TCP 报文段或 UDP 用户数据报)时，就要根据其中首部的端口号来决定应当通过哪一个端口上交给应当接受此数据的应用进程。图 5.5 说明了端口在进程之间的通信中所起的作用。传输层具有复用和分用的功能(将在 5.1.3 小节介绍)。应用层所有的应用进程都可以通过传输层再传送到网络层，这

称为复用。传输层从网络层收到的数据报后必须交付给指明的应用进程，这就是分用。

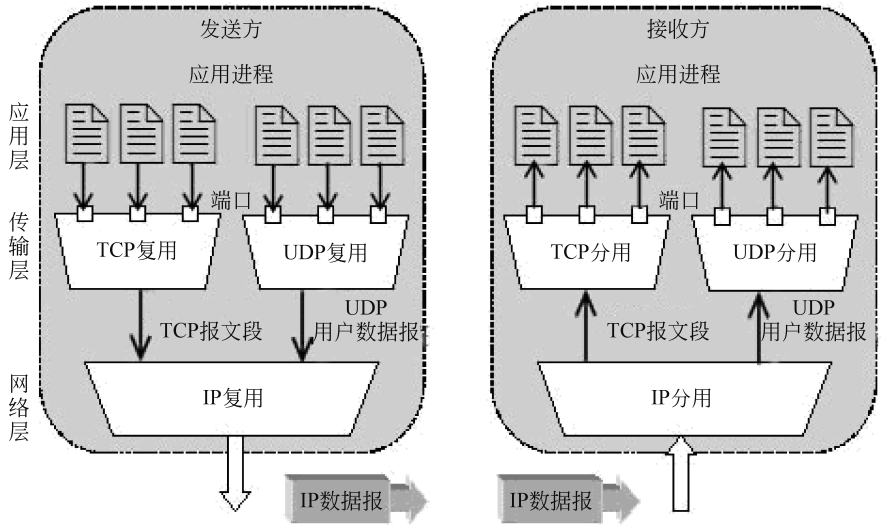

图 5.5 端口在进程之间的通信中所起的作用

端口可以分为两大类：服务器端使用的端口号和客户端使用的端口号。

(1) 服务器端使用的端口号又可以分为专用端口号和注册端口号。

- 专用端口号：也称为熟知端口号，端口号的范围是 0～1023，绑定于一些特定的服务，通常带有这些端口号的通信明确表明了某种服务的协议，这种端口号不可再重定义它的作用对象。
- 注册端口号：端口号的范围是 1024～49 151，多数没有明确的定义服务对象，不同程序可根据实际需要自己定义，比如远程控制软件和木马程序中都会有这些端口号的定义。

(2) 客户端使用的端口号：49 152～65 535，仅在客户进程运行时才动态分配，是留给客户进程暂时使用时选择。通信结束后被收回，供其他客户进程以后使用。

表 5.1 给出了常用的端口号及对应协议。在表 5.1 中我们可以看到常见的协议及它对应的端口号，如 DNS 的端口号是 53，HTTP 的端口号为 80 等。

表 5.1 常用的端口号及对应协议

端口号	应用程序	说　明
20	FTP_DATA	文件传输协议(数据)
21	FTP_CONTROL	文件传输协议(命令)
23	TELNET	远程连接
25	SMTP	简单邮件传输协议
53	DNS	域名解析服务
69	TFTP	简单文件传输协议

续表

端口号	应用程序	说　明
80	HTTP	超文本传输协议
110	POP3	邮局协议版本 3
161	SNMP	简单网络管理协议
179	BGP	边界网关协议
520	RIP	路由信息协议

2. 套接字

应用层通过传输层进行数据通信时，传输层会同时为多个应用进程提供并发服务，为了区分不同的应用程序进程和连接，计算机操作系统为应用进程与 TCP/IP 交互提供了被称为套接字（socket）的接口，用来区分不同应用程序进程间的网络通信和连接。Socket 的原意是“插座”，就像电话的插口一样，没有它就完全没办法通信。

在 TCP 协议簇中的传输层协议需要 IP 地址和端口号，它们各在一端建立一条连接。一个 IP 地址和一个端口号结合起来称为套接字地址（socket address）。

套接字地址 =（IP 地址：端口号）　　(5-1)

客户套接字地址唯一定义了客户进程，而服务器套接字地址唯一地定义了服务器进程。套接字可以看成在两个程序进行通信连接中的一个端点，如图 5.6 所示。一个程序将一段信息写入套接字中，该套接字将这段信息发送给另一个套接字中，使这段信息能传送到其他程序中。

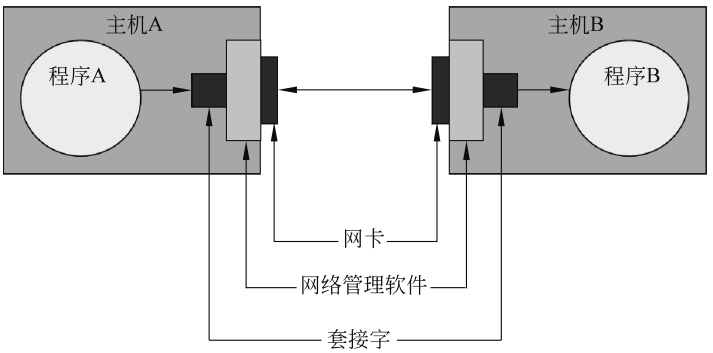

图 5.6　应用程序进程、套接字

两台计算机中的进程要互相通信，不仅需要知道双方的 IP 地址，而且还需要知道对方的端口号。将端口号拼接到 IP 地址后即构成了套接字。例如，IP 地址为 202.4.25.21，而端口号为 80，那么套接字就是（202.4.25.21：80）。如果加上协议，比如 http 协议，将是常见的在浏览器输入的 Web 请求（http：// 202.4.25.21：80），请求远程 Web 服务器提供服务。

为了使用 Internet 中的传输服务，我们需要一对套接字地址：客户套接字地址和服务器套接字地址。这四条信息是网络层分组首部和传输层分组首部的组成部分。第一个

首部包含 IP 地址，而第二个首部包含端口号。每一条 TCP 连接唯一地被通信两端的两个端点（即一对套接字）所确定，即：

$$\text{TCP 连接}::=\{\text{Socket1},\text{Socket2}\}=\{(\text{IP1}:\ \text{port1}),(\text{IP2}:\ \text{port2})\} \tag{5-2}$$

5.1.3 传输层的多路复用与多路分解

每当一个实体从一个以上的源接收到数据项时，称为多路复用（multiplexing，多对一）；每当一个实体将数据项传递到一个以上的源时，称为多路分解（demultiplexing，一对多）。源端的传输层执行复用；目的端的传输层执行多路分解，如图 5.7 所示。

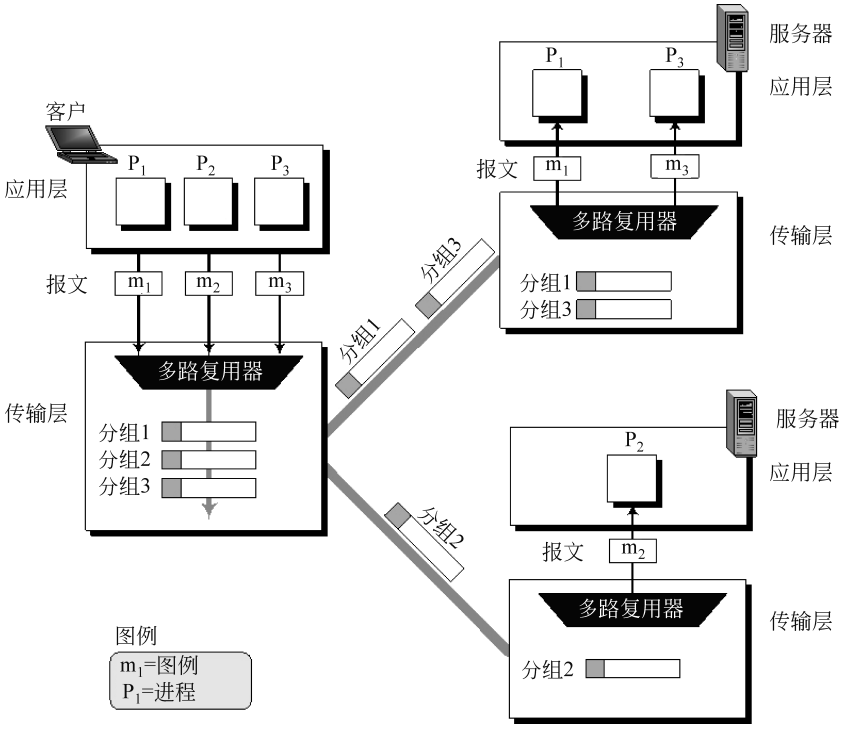

图 5.7 多路复用和多路分解

图 5.7 给出了一个客户和两个服务器之间的通信。客户端运行三个进程：P_1、P_2 和 P_3。进程 P_1 和 P_3 需要将请求发送到对应的服务器进程。客户进程 P_2 需要将请求发送到位于另外一个服务器的服务器进程。客户端的传输层接收到来自三个进程的三个报文并创建三个分组。它起到了多路复用器的作用，分组 1 和 3 使用相同的逻辑信道到达第一个服务器的传输层。当它们到达服务器时，传输层起到多路分解器的作用并将报文分发到两个不同的进程。第二个服务器的传输层接收分组 2 并将它传递到相应的进程。注意，尽管只有一个报文，我们仍然用到多路分解。

5.1.4 无连接服务与面向连接服务

从通信的角度上看，网络中各层所提供的服务可以分成两大类：无连接的服务和面

向连接的服务。无连接的服务和邮政系统的工作模式相似，两个系统之间的通信不需要先建立好连接，是一种不可靠的传输。面向连接的服务和电话系统的工作模式相似，具有连接建立、数据传输和连接释放三个阶段。

由于 TCP/IP 的网络层提供的是面向无连接的数据报服务（见第 4 章的介绍），也就是说 IP 数据报传送会出现丢失、重复和乱序的情况，因此在 TCP/IP 网络结构中传输层就变得极为重要。TCP/IP 的传输层提供了两个主要的协议：用户数据报协议（User Datagram Protocol，UDP）和传输控制协议（Transmission Control Protocol，TCP），如图 5.8 所示，并给出了这两种协议在协议栈中的位置。

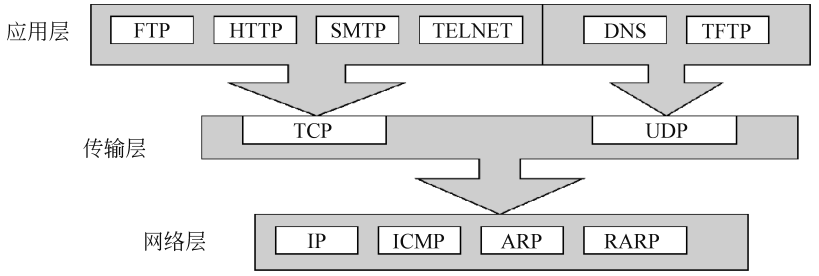

图 5.8　TCP/IP 的传输层协议与上下层中的协议

TCP 是面向连接的、可靠的传输协议。在传送数据之前必须先建立连接，数据传送结束后释放连接。它能把报文分解成数段，在目的端再重新装配这些段，重新发送没有被收到的段。在用户通信之间 TCP 提供了一个虚电路。

UDP 是无连接的，而且“不可靠”，远端主机的传输层在收到 UDP 报文后，不需要给出任何确认，也没有对发送段进行软件校验。因此，被称为“不可靠”。

无连接的服务和面向连接的服务在网络体结构的各层中都有实现。各功能层具体实现协议举例如表 5.2 所示。

表 5.2　两大类服务的具体实现协议

协议层次	无连接的服务	面向连接的服务
传输层	UDP	TCP
网络层	IP	X.25 分组级
数据链路层	CSMA/CD	PPP、HDLC

当传输层采用面向连接的协议（如 TCP）时，它为应用进程在传输实体间建立一条全双工的可靠逻辑信道，尽管下面的网络可能是不可靠的（如 IP 交换网络）。

5.2　UDP

用户数据报协议（User Datagram Protocol，UDP）是无连接不可靠的传输层协议，它除了提供进程到进程之间的通信之外，只是在 IP 的数据报服务的基础上增加了端口复用

分用和差错检测的功能。UDP 协议具有如下特点。

(1) UDP 是无连接的,在传送数据前不需要与对方建立连接,因此减少了开销和发送数据之前的时延。

(2) UDP 提供不可靠的服务,没有拥塞控制,数据传送到对方可能没有按顺序、重复甚至丢失,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 同时支持点到点和多点之间的通信,对网络实时应用(如视频电话等)是很重要的。网络出现拥塞不会使源主机的发送速率降低,而这些实时应用要求源主机以恒定的速率发送数据,允许在网络发生拥塞时丢失一些数据,但不允许数据有太大的时延。UDP 正好适合这种要求。

(4) UDP 的首部只有 8 字节,传输开销小。

(5) UDP 是面向报文的。使用 UDP 发送一个很短的报文,在发送方和接收方之间的交互要比使用 TCP 时少得多。

5.2.1 用户数据报概述

UDP 分组称为用户数据报(user datagram),有 8 字节的固定首部,这个首部由 4 个字段组成,每个字段 2 字节(16 位)。图 5.9 说明了用户数据报的格式。

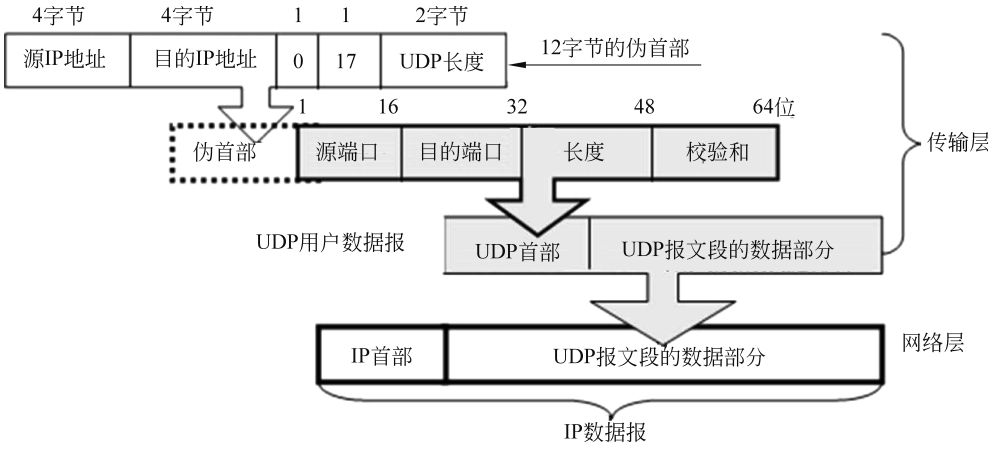

图 5.9 用户数据报格式

(1) 源端口字段:包含 16 位长度的发送端 UDP 协议端口号。

(2) 目的端口字段:包含 16 位长度的接收端 UDP 协议端口号。

(3) 长度字段:UDP 用户数据报的长度,记录该数据报的长度,即首部加数据的长度,16 位可以定义的总长度范围是 0～65 535。

(4) 校验和字段:防止 UDP 用户数据报在传输中出错。校验和字段是可选择的,如该字段值为 0 则表明不进行校验。一般说来,使用校验和字段是必要的。

UDP 校验和包含三部分:伪首部、UDP 首部和从应用层来的数据。伪首部是 IP 分组的首部的一部分,其中有些字段要填入 0,如图 5.9 所示。伪首部总共 12 字节,分 5 个字段,第 1 个字段为源 IP 地址占 4 字节(32 位);第 2 个字段为目的 IP 地址占 4 字节(32

位);第 3 字段是全 0;第 4 个字段是 IP 首部中的协议字段的值,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。

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。只是在计算校验和时,临时和 UDP 用户数据报连接在一起,得到一个过渡的临时的 UDP 用户数据报。校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。如果校验和不包括伪首部,用户数据报也可能是安全完整地到达。但是,如果 IP 首部受到损坏,那么它可能被提交到错误的主机。

增加协议字段是确保这个分组是属于 UDP,而不是属于其他传输层协议。我们在后面将会看到,如果一个进程既可用 UDP 又可用 TCP,则端口号可以是相同的。UDP 的协议字段值是 17。如果传输过程中这个值改变了,在接收端计算校验和时就可检测出来,UDP 就可丢弃这个分组。这样就不会传递给错误的协议。

UDP 计算校验和的方法和计算 IP 数据报首部校验和的方法相似(在第 4 章介绍)。但不同的是:IP 数据报的校验和只检验 IP 数据报的首部,但 UDP 的校验和是将首部和数据部分一起都检验。在发送端,首先是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 位的字串接起来。若 UDP 用户数据报的数据部分不是偶数字节,则要填入一个全零字节(即:最后一个基数字节应是 16 位数的高字节,而低字节填 0)。然后按二进制反码计算出这些 16 位字的和(两个数进行二进制反码求和的运算的规则是: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)。将此和的二进制反码写入校验和字段后,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 位字的和。当无差错时其结果应全为 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)。

【例 5-1】 假设从源端 A 要发送下列 3 个 16 位的二进制数:word1、word2 和 word3 到终端 B,校验和计算如下:

word1:0110011001100110

word2:0101010101010101

word3:0000111100001111

【解】 先将校验和字段全填 0;即 word4 为 0000000000000000。

将 4 个 16 位二进制反码求和 sum=word1+word2+word3+word4=1100101011001010。

校验和(sum 的反码)为 0011010100110101。

从发送端发出的 4 个(word1、word2、word3 以及校验和)16 位二进制数之和为 1111111111111111,如果接收端收到的这 4 个 16 位二进制数之和也是全"1",就认为传输过程中没有出差错。

【例 5-2】 以下是十六进制格式的 UDP 首部内容:DDAB 0035 002A 8E7A,请问:

(1) 源端口号是多少?

(2) 目的端口号是多少?

(3) 用户数据报总长度是多少?

(4) 数据长度是多少?

(5) 分组是从客户端发往服务器端的还是相反方向的?

(6) 客户进程是什么?

【解】 (1) 源端口号是头 4 位十六进制数(DDAB)16,这意味着源端口号是 56747(一般的端口号)。

(2) 目的端口号是第二组 4 位十六进制数(0035) 16,即目的端口号为 53(是 DNS 的查询默认端口)。

(3) 第三组 4 位十六进制数(002A)16,定义了整个 UDP 分组的长度,即长度为 42 字节。

(4) 数据的长度是整个分组长度减去首部长度,即 42－8＝34 字节。

(5) 由于目的端口号是 53(为专用端口号),分组是从客户端发送到服务器端。

(6) 客户进程是 DNS(见表 5.1)。

5.2.2 UDP 应用

尽管 UDP 不满足我们之前讨论的可靠传输层协议标准,但是,UDP 更适合某些应用。

如前所述,UDP 是无连接协议。同一个应用程序发送的 UDP 分组之间是独立的。这个特征可以看作是优势也可以看作是劣势,这要取决于应用要求。例如,如果一个客户应用需要向服务器发送一个短的请求并接收一个短的响应,那么这就是优势。如果请求和响应各自可以填充进一个数据报,那么无连接服务可能就更可取。在这种情况下,建立和关闭连接的开销可能很可观。在面向连接服务中,要达到以上目标,至少需要在客户和服务器之间交换 9 个分组;在无连接服务中只需要交换 2 个分组。无连接提供了更小的延迟;面向连接服务造成了更多的延迟。例如,一种客户－服务器应用如 DNS(第 6 章讨论),它使用 UDP 服务,因为客户需要向服务器发送一个短的请求,并从服务器接收快速响应。请求和响应可以填充进一个用户数据报。由于在每个方向上只交换一个报文,因此无连接特性不是问题;客户或服务器不担心报文会时序传递。

UDP 不提供差错控制;它提供的是不可靠服务。绝大多数应用期待从传输层协议中得到可靠服务。尽管可靠服务是人们想要的,但是它可能有一些副作用,这些副作用对某些应用来说是不可接受的。当一个传输层提供可靠服务时,如果报文的一部分丢失或者被破坏,它就需要被重传。这意味着接收方传输层不能向应用立即传送那一部分信息;在传向应用层的不同报文部分会有不一致的延迟。对于某些应用根本注意不到这些不一致的延迟,但是对于有些应用这些延迟却是致命的。假设我们正在使用一个实时交互应用,例如 Skype。音频和视频被分割成帧并且一个接一个发送。如果传输层应该重传某些被破坏或丢失的帧,那么整个传输的同步性就会丧失。观众会突然看到空白屏幕并且需要等待,直到第二个传输到达。这是不可容忍的。然而,如果屏幕的每个小部分都使用一个用户数据报传送,那么接收 UDP 可以轻易地忽略被破坏或丢失的分组,并将其余分组传递到应用程序。屏幕的那部分会空白很短时间,而绝大多数观众都不会注意到。

UDP 不提供拥塞控制。然而在倾向于出错的网络中 UDP 没有创建额外的通信量。