第 5 章 传 输 层

本章从网络环境中分布式进程通信的概念出发，系统地讨论传输层的基本功能、向应用层提供的服务以及传输层的 UDP 与 TCP 这两个协议，为读者进一步研究应用层与应用层协议奠定基础。

本章学习要求

- 理解网络环境中的分布式进程通信的概念。
- 掌握进程通信中的客户/服务器模式的概念。
- 掌握传输层的基本功能与服务质量的概念。
- 掌握 UDP 的基本内容。
- 掌握 TCP 的基本内容。

5.1 传输层与传输层协议

5.1.1 传输层的基本功能

网络层、数据链路层与物理层实现了网络中主机之间的数据通信，但是数据通信并不是组建计算机网络的最终目的。计算机网络的本质活动是实现分布在不同地理位置的主机之间的进程通信，以实现应用层的各种网络服务。传输层的主要功能是要实现分布式进程通信。因此，传输层是实现各种网络应用的基础。图 5-1 给出了传输层的基本功能。

理解传输层的基本功能时需要注意以下 3 个问题：

- 网络层的 IP 地址标识主机、路由器的位置信息。路由选择算法在互联网中选择一条源主机-路由器、路由器-路由器、路由器-目的主机的多段点-点链路组成的传输路径；IP 协议通过这条传输路径完成 IP 分组的传输。传输层协议利用网络层提供的服务，在源主机与目的主机的应用进程之间建立端-端连接，以实现分布式进程通信。
- 互联网中的路由器与通信线路构成传输网（或承载网）。传输网一般是由电信公司运营与管理的。传输网提供的服务有可能不可靠（例如丢失分组），而用户又无法对传输网加以控制。解决这个问题需要从两方面入手：一是电信公司提高传输网的

服务质量；二是传输层对分组丢失、线路故障进行检测，并采取相应的差错控制措施，以满足分布式进程通信的服务质量（QoS）要求。因此，在传输层要讨论如何改善 QoS，以达到计算机进程通信要求的问题。

- 传输层可以屏蔽传输网实现技术上的差异性，弥补网络层所提供服务的不足，使应用层在设计各种网络应用系统时，仅需考虑选择怎样的传输层协议以满足应用进程的通信要求的问题，而不需要考虑数据传输的细节问题。

因此，从点-点通信到端-端通信是一次质的飞跃，为此传输层需要引入很多新的概念和机制。

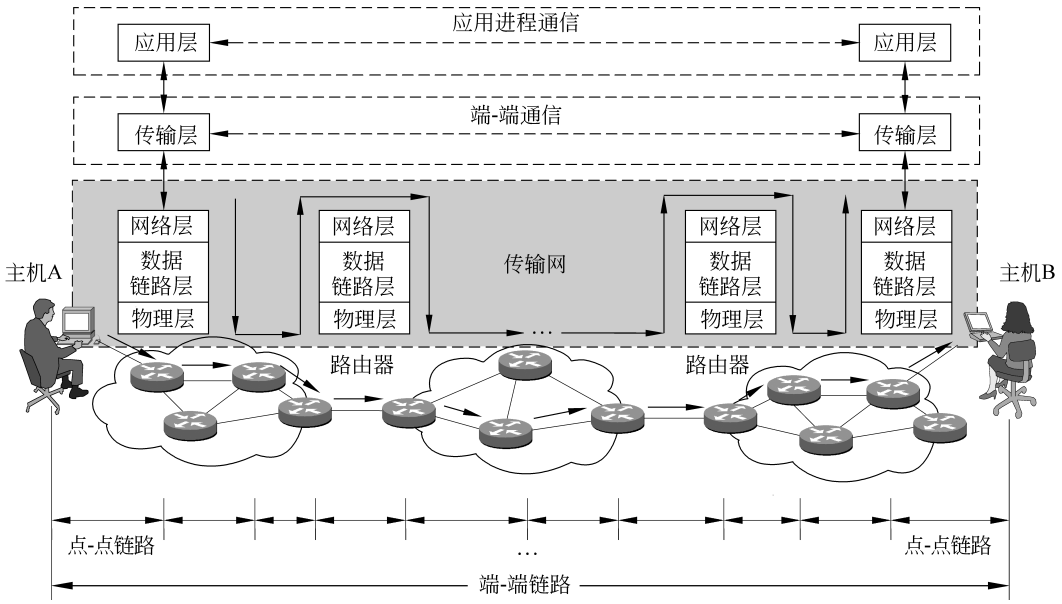

图 5-1　传输层的基本功能

5.1.2　传输协议数据单元的概念

传输层中实现传输层协议的软件称为传输实体（transport entity）。传输实体可以在操作系统内核中，也可以在用户程序中。图 5-2 给出了传输实体的概念示意图。从中可以看出传输层与应用层、网络层之间的关系。

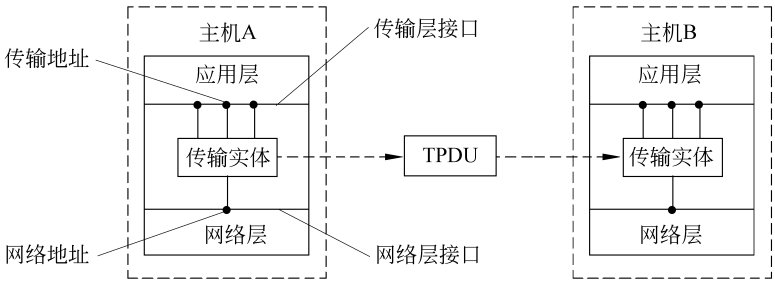

图 5-2　传输实体的概念示意图

传输层之间传输的报文称为传输协议数据单元(Transport Protocol Data Unit,TPDU)。TPDU的有效载荷是应用层数据,有效载荷之前加上TPDU头形成TPDU。当TPDU传送到网络层时,加上IP分组头形成IP分组;当IP分组传送到数据链路层时,加上帧头、帧尾形成帧。当帧通过传输介质到达目的主机时,经过数据链路层与网络层处理之后,交给传输层的数据就是TPDU,接下来读取TPDU头并按要求执行。TPDU头用于传达传输层协议的命令和响应。图5-3给出了TPDU与IP分组、帧的关系。

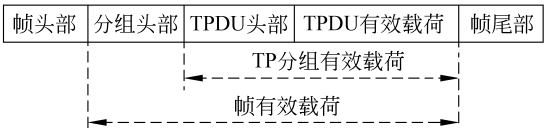

图5-3 TPDU与IP分组、帧的关系

5.1.3 应用进程、传输层接口与套接字

传输层接口与套接字是传输层的两个重要概念。图5-4给出了应用进程、套接字与IP地址的关系示意图。

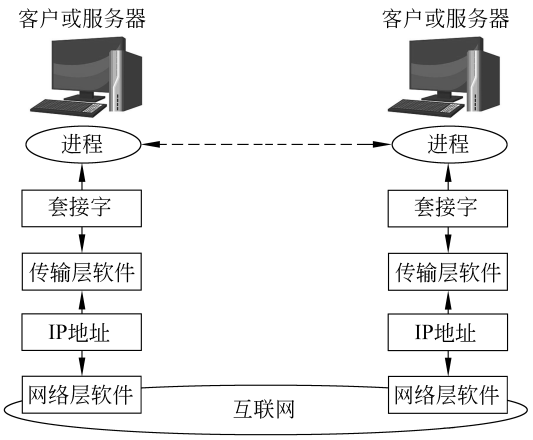

图5-4 应用进程、套接字与IP地址的关系

1. 应用程序、传输层软件与主机操作系统的关系

应用程序与TCP、UDP在主机操作系统的控制下工作。应用程序开发者只能根据需要选择TCP或UDP,设定相应的缓存、最大报文长度等参数。在传输层协议类型与参数选定之后,传输层协议软件在主机操作系统的控制下,为应用程序提供进程通信服务。

2. 进程通信与传输层端口号、网络层IP地址的关系

下面举一个例子来形象地说明进程、传输层端口号(port number)与网络层IP地址的关系。如果一位同学到南开大学计算机学院网络教研室找作者讨论问题,可以先找到计算机学院办公室进行查询,得知网络教研室位于伯苓楼501。这里,伯苓楼相当于IP地址,501相当于端口号。IP地址仅能告诉这位同学要找的教研室位于哪座楼。这位同学还必须知道是哪座楼的哪个房间,才能顺利地找到要去的地方。在这位同学找到作者之后,讨论问题的过程相当于两台主机进程之间通信的过程。在计算机网络中,只有知道IP地址与端口

号，才能唯一地找到准备通信的进程。

3. 套接字的概念

传输层需要解决的一个重要问题是进程标识。在一台计算机中，不同进程可以用进程号(process ID)来标识。进程号又称为端口号。在网络环境中，标识一个进程必须同时使用IP地址与端口号。RFC 793定义的套接字(socket)由IP地址与端口号(形式为“IP地址：端口号”)组成。例如，一个IP地址为202.1.2.5、端口号为30022的客户端与一个IP地址为151.8.22.51、端口号为80的Web服务器建立TCP连接，那么标识客户端的套接字为202.1.2.5:30022，标识服务器端的套接字为151.8.22.51:80。

术语socket有多种不同的含义：

- 在网络原理讨论中，RFC 793中的socket为“IP地址:端口号”。
- 在网络软件编程中，网络应用程序的编程接口(Application Programming Interface，API)又称为socket。
- 在API中，有一个函数名也是socket。
- 在操作系统讨论中，也会出现术语socket。

5.1.4 网络环境中的分布式进程标识方法

为了实现网络环境中的分布式进程通信，首先需要解决两个基本问题：进程标识与多重协议的识别。

1. 进程标识的基本方法

传输层寻址是通过TCP与UDP端口号来实现的。互联网应用程序的类型很多，例如基于客户/服务器(Client/Server，C/S)模式的FTP、E-mail、Web、DNS与SNMP应用，以及基于对等(Peer-to-Peer，P2P)模式的文件共享、即时通信类应用。这些应用程序在传输层分别选择TCP或UDP。为了区别不同的网络应用程序，TCP与UDP用不同的端口号来表示不同的应用程序。

2. 端口号的分配方法

1) 端口号的数值范围

在TCP/IP中，端口号的数值是0～65 535的整数。

2) 端口号的类型

IANA定义的端口号有3种类型：熟知端口号、注册端口号和临时端口号。图5-5给出了IANA对于端口号数值范围的划分。

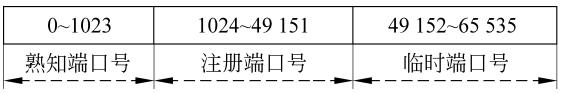

图5-5 IANA对于端口号数值范围的划分

TCP/UDP给每种标准的互联网服务器进程分配一个确定的全局端口号，称为熟知端口号(well-known port number)或公认端口号。每个客户进程都知道相应的服务器进程的熟知端口号。熟知端口号数值范围为0～1023，它是由IANA统一分配的。熟知端口号列表可以在http://www.iana.org中查询。

注册端口号数值范围为1024～49 151。当用户开发一种新的网络应用时,为了防止这种网络应用在互联网中使用时出现冲突,应为这种网络应用的服务器程序向IANA登记一个注册端口号。

临时端口号数值范围为49 152～65 535。客户进程使用临时端口号,它可以由TCP/UDP软件随机选取。临时端口号仅对一次进程通信有效。

图5-6给出了进程标识方法示意图。

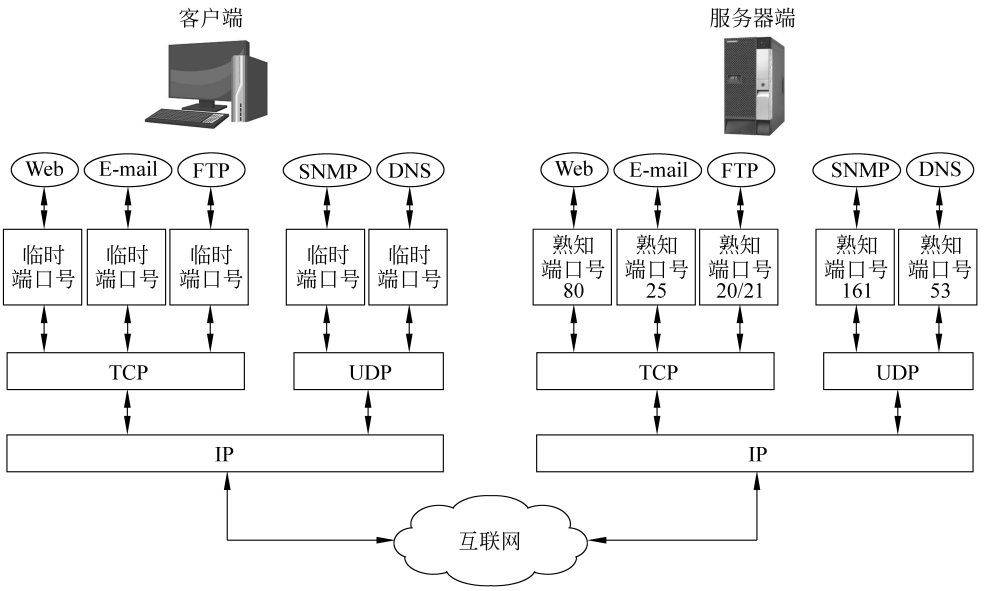

图5-6　进程标识方法示意图

3. 熟知端口号的分配方法

1) UDP的熟知端口号

表5-1给出了UDP常用的熟知端口号。UDP服务与端口号的映射表定期在RFC 768等文档中公布,并可以在多数UNIX主机的/etc/services文件中找到。

表5-1　UDP常用的熟知端口号

端口号	服务进程	说　明	端口号	服务进程	说　明
53	DNS	域名系统	161/162	SNMP	简单网络管理协议
67/68	DHCP	动态主机配置协议	520	RIP	路由信息协议
69	TFTP	简单文件传输协议			

需要注意的是,DHCP和SNMP的熟知端口号的使用与DNS不同。DHCP和SNMP的客户端和服务器端在通信时都使用熟知端口号。

2) TCP的熟知端口号

表5-2给出了TCP常用的熟知端口号。

表 5-2　TCP 常用的熟知端口号

端口号	服务进程	说　明	端口号	服务进程	说　明
20/21	FTP	文件传输协议	80	HTTP	超文本传输协议
23	TELNET	远程登录协议	110	POP	邮局协议
25	SMTP	简单邮件传输协议	179	BGP	边界路由协议

4. 多重协议的识别

实现分布式进程通信要解决的另一个问题是多重协议的识别。例如，UNIX 操作系统在传输层采用 TCP 与 UDP。Xerox 网络系统(Xerox Network System，XNS)在传输层使用自己的顺序分组协议(Sequential Packet Protocol，SPP)与网间数据报协议(Internetwork Datagram Protocol，IDP)。其中，SPP 相当于 TCP，IDP 相当于 UDP。在实际的应用中，还有其他类似的传输层协议。

网络中的两台主机要实现进程通信，必须事先约定好传输层协议类型。如果一台主机的传输层使用 TCP，另一台主机的传输层使用 UDP，两种协议的报文格式、端口号分配及协议执行过程不同，使得两个进程无法正常交换数据。因此，两台主机必须在通信之前确定都采用 TCP 或 UDP。

如果考虑到进程标识和多重协议的识别，网络中一个进程的全网唯一标识应该用三元组来表示：协议、本地 IP 地址与本地端口号。在 UNIX 操作系统中，这个三元组又称为半相关(half-association)。图 5-7 给出了三元组的结构。

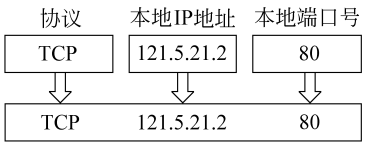

图 5-7　三元组的结构

由于分布式进程通信涉及两个主机的进程，因此一个完整的进程通信标识需要一个五元组表示。这个五元组是：协议、本地地址、本地端口号、远程地址与远程端口号。在 UNIX 操作系统中，这个五元组又称为相关(association)。例如，客户端的套接字为 202.1.2.5:30022，服务器端的套接字为 121.5.21.2:80，则客户端标识与服务器 TCP 连接的五元组为“TCP，202.1.2.5:30022，121.5.21.2:80”。

5.1.5　传输层的多路复用与分解

一台 TCP/IP 主机可能同时运行不同应用程序。如果客户端和服务器端同时运行 4 个应用程序：域名服务(DNS)、Web 服务(HTTP)、电子邮件(SMTP)与网络管理(SNMP)。其中，HTTP、SMTP 使用 TCP，DNS、SNMP 使用 UDP。TCP/IP 允许多个应用程序同时使用一个 IP 地址和物理链路来发送和接收数据。在发送端，IP 协议将 TCP 或 UDP 的 TPDU 都封装成 IP 分组来发送；在接收端，IP 协议将从 IP 分组中拆分出来的 TPDU 传送到传输层，由传输层根据 TPDU 端口号加以区分，分别交给相应的 4 个应用进程。这个过程称为传输层的多路复用(multiplexing)与多路分解(demultiplexing)。图 5-8 给出了传输层的多路复用与多路分解过程。

5.1.6　TCP、UDP 与应用层协议的关系

应用层协议依赖于某种传输层协议，这种依赖关系分为 3 类：应用层协议仅依赖于

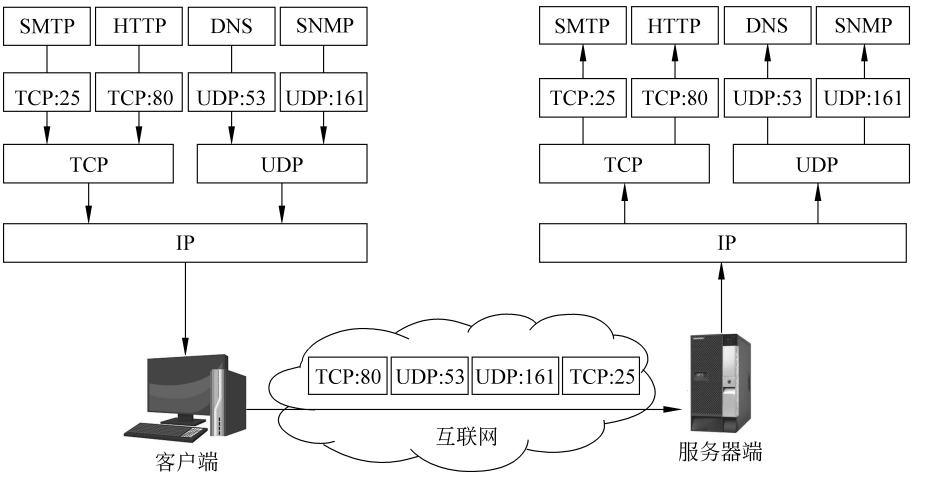

图 5-8　传输层的多路复用与多路分解过程

TCP;应用层协议仅依赖于 UDP;应用层协议可依赖于 TCP 或 UDP。图 5-9 给出了传输层协议与应用层协议的关系。

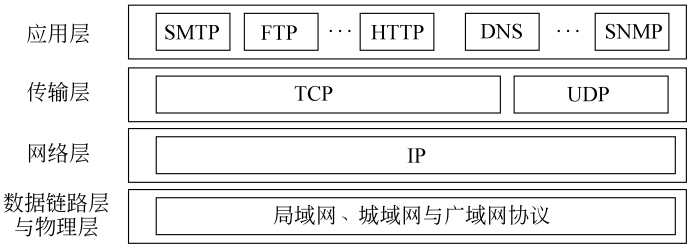

图 5-9　传输层协议与应用层协议的关系

仅依赖于 TCP 的应用层协议通常一次传输大量数据,例如 TELNET、SMTP、FTP、HTTP 等。仅依赖于 UDP 的应用层协议通常频繁交换少量数据,典型协议是 SNMP。有些应用可依赖于 TCP 或 UDP,例如 DNS。UDP 的优点是简洁、效率高、处理速度快,这在 P2P 类应用中显得更加突出。

5.2　UDP

5.2.1　UDP 的主要特点

UDP 的设计原则是协议简洁、运行快捷。1980 年,RFC 768 文档定义了 UDP 的内容,整个文档仅有 3 页。RFC 1122 文档对 UDP 进行了修订。

UDP 的特点主要表现在以下几个方面。

1. 无连接的传输层协议

理解 UDP 的无连接传输特点时需要注意以下几个问题:

- UDP 传输报文之前无须在通信双方之间建立连接,这样做有效减少了协议开销与传输延时。

- 除了为报文提供一种可选的校验和之外，UDP 几乎没有提供保证数据传输可靠性的措施。
- 如果 UDP 软件发现接收到的分组出错，它就会丢弃这个分组，既不确认又不通知发送端重传。

因此，UDP 提供的是尽力而为的传输服务。

2. 面向报文的传输层协议

图 5-10 给出了 UDP 对应用程序数据的处理方式。

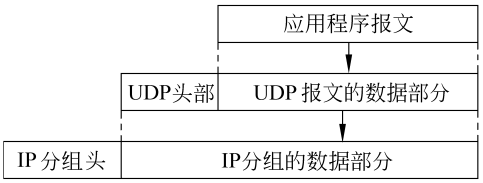

图 5-10　UDP 对应用程序数据的处理方式

理解 UDP 面向报文的传输特点时需要注意以下几个问题：

- 对于应用程序提交的报文，在添加 UDP 报头形成 TPDU 之后，就向下提交给网络层的 IP 协议。
- 对于应用程序提交的报文，UDP 既不合并也不拆分，而是保留报文的长度与格式。接收端将接收的报文原封不动地提交给应用程序。因此，应用程序必须选择好长度合适的报文。
- 如果应用程序提交的报文过短，则处理开销较大；如果应用程序提交的报文过长，则 IP 协议可能要对 TPDU 进行分片，这样也会降低处理效率。

5.2.2　UDP 报文格式

图 5-11 给出了 UDP 报文的格式。UDP 报文有长度固定为 8B 的报头。

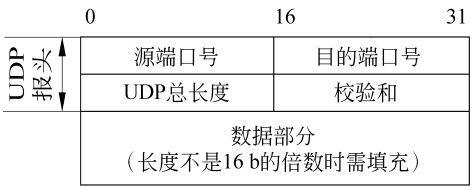

图 5-11　UDP 报文的格式

UDP 报头主要包括以下几个字段：

(1) 端口号字段。包括源端口号与目的端口号，每个字段长度均为 16b。源端口号表示发送端进程使用的端口号，目的端口号表示接收端进程使用的端口号。如果发送端进程是客户端，源端口号是 UDP 软件分配的临时端口号，目的端口号是服务器的熟知端口号。

(2) UDP 总长度字段。表示 UDP 报文的总长度，字段长度是 16b。因此，UDP 报文长度最小为 8B，最大为 65 535B。

(3) 校验和字段。用来检测整个 UDP 报文(包括伪报头)在传输中是否出错，字段长度

为16b。校验和字段在UDP中是可选的字段,这反映了效率优先的思想。如果应用进程对通信效率的要求高于可靠性,应用进程可选择不使用校验和。

5.2.3 UDP校验和的概念

1. 伪报头的概念

理解在校验时增加伪报头的目的时需要注意以下几个问题:

- 伪报头不是UDP报文的真正头部,只是在计算校验和时临时增加的。
- 伪报头仅在计算时起作用,它既不向低层也不向高层传输。
- 伪报头包括IP分组头的源IP地址(32b)、目的IP地址(32b)、协议字段(8b)与UDP长度(16b),以及全0的填充字段(8b)。
- 如果没有伪报头,校验对象仅是UDP报文,也能够判断UDP报文传输是否出错。但是,设计者考虑到以下情况:如果IP分组头出错,那么IP分组可能传送到错误的主机,因此在计算UDP校验和时增加了伪报头。

2. 伪报头结构

UDP校验和主要包括3部分:伪报头(pseudo header)、UDP报头与数据。伪报头的长度为12B。图5-12给出了伪报头的结构。伪报头取自IP分组头的一部分,填充字段需要全部填0,使伪报头长度为16位的整数倍。IP分组头的协议号为17,表示一个UDP报文。UDP总长度不包括伪报头的长度。

图5-12 UDP校验和校验的伪报头与报头的结构

5.2.4 UDP的适用范围

确定应用程序在传输层是否采用UDP,应主要考虑以下3类应用。

1. 视频播放应用

用户在互联网环境中播放视频时,最关注的是视频流尽快、不间断播放,丢失个别报文对视频节目的播放效果不会产生很大影响。如果采用TCP,可能因重传丢失的报文而增大传输延迟,反而对视频播放造成不利影响。视频播放应用对数据交付实时性要求较高,而对数据交付可靠性要求相对较低,UDP更为适用。

2. 简短的交互式应用

有一类应用仅需进行频繁、简短的请求与应答报文交互,客户端发送一个简短的请求报文,服务器回复一个简短的应答报文,这时应用程序应该选择UDP。应用程序可通过定时器/重传机制来处理IP分组丢失问题,而无须选择有确认/重传机制的TCP,以提高这类网络应用的工作效率。

3. 多播与广播应用

UDP 支持一对多与多对多的交互式通信，这一点是 TCP 不支持的。UDP 报头长度只有 8B，比 TCP 报头长度短。同时，UDP 没有拥塞控制机制，在网络拥塞时不会要求源主机降低发送速率，而是丢弃个别报文。这个特点适用于 IP 电话、视频会议应用。

当然，任何事情都有两面性。UDP 的优点是简洁、快速、高效，但是没有提供必要的差错控制机制，在拥塞严重时缺乏控制与调节手段。对于使用 UDP 的应用程序来说，设计者需要在应用层设置必要的机制对上述问题加以解决。总之，UDP 是一种适用于实时语音与视频传输的传输层协议。

5.3 TCP

5.3.1 TCP 的主要特点

RFC793 文档最早描述了 TCP，此后出现了几十个 RFC 文档对 TCP 功能进行扩充与调整。例如，RFC 2415 文档补充了 TCP 的滑动窗口与确认策略，RFC 2581 文档补充了 TCP 的拥塞控制机制，RFC 2988 文档补充了 TCP 的重传定时器。

TCP 的特点主要表现在以下几个方面。

1. 支持面向连接的服务

如果将 UDP 提供的服务比作一封平信，那么 TCP 提供的服务相当于电话。UDP 是一种可满足最低要求的传输层协议，而 TCP 是一种功能完善的传输层协议。

面向连接对提高数据传输的可靠性很重要。应用程序使用 TCP 传送数据之前，必须在源进程与目的进程之间建立一条 TCP 连接。每个连接用通信双方的端口号来标识，并为双方的一次进程通信提供服务。TCP 建立在不可靠的 IP 协议之上，由于 IP 不提供任何可靠性保障机制，因此 TCP 的可靠性需要靠自己来解决。

2. 支持字节流传输

图 5-13 给出了 TCP 支持字节流传输的过程。流(stream)相当于一个管道，从一端放入什么内容，从另一端原样取出什么内容。流描述了一个没有出现丢失、重复和乱序的数据传输过程。

如果数据是通过键盘输入的，那么应用程序逐个将字符提交给发送端。如果数据是从文件中获取的，那么数据可能会逐行或逐块交付给发送端。应用程序与 TCP 每次交互的数据长度可能不同，但是 TCP 将应用程序提交的数据看成一串无结构的字节流。为了能够支持字节流传输，发送端和接收端都需要使用缓存。发送端使用发送缓存保存来自应用程序的数据。发送端不可能为每个写操作创建一个报文段，而是将几个写操作组合成一个报文段，由 IP 协议封装成 IP 分组之后发送到接收端。在接收端，IP 协议将接收的 IP 分组拆封之后，将数据部分提交给 TCP 并按字节流存储在接收缓存中，最后由应用程序通过读操作从接收缓存中读出数据。

由于 TCP 在传输过程中将应用程序提交的数据看成一串无结构的字节流，因此接收端的数据字节起始与终结位置必须由应用程序自己确定。

3. 支持全双工通信

TCP 允许通信双方的应用程序在任何时候都可以发送数据。由于通信双方都设置了