实验 6 用户数据报协议（UDP）

【实验目的】

1. 掌握 UDP 协议的报文格式
2. 掌握 UDP 协议校验和的计算方法
3. 理解 UDP 协议的优缺点
4. 理解协议栈对 UDP 协议的处理方法
5. 理解 UDP 上层接口应满足的条件

【学时分配】

4 学时

【实验环境】

该实验采用网络结构一

【实验原理】

一. 进程到进程的通信

在学习 UDP 协议之前，首先应该了解主机到主机的通信和进程到进程的通信，以及这两种通信之间的区别。

IP 协议负责主机到主机的通信。作为一个网络层协议，IP 协议只能把报文交付给目的主机。这是一种不完整的交付，因为这个报文还没有送交到正确的进程。像 UDP 这样的传输层协议负责进程到进程的通信。UDP 协议负责把报文交付到正确的进程。下图描绘了 IP 协议和 UDP 协议的作用范围。

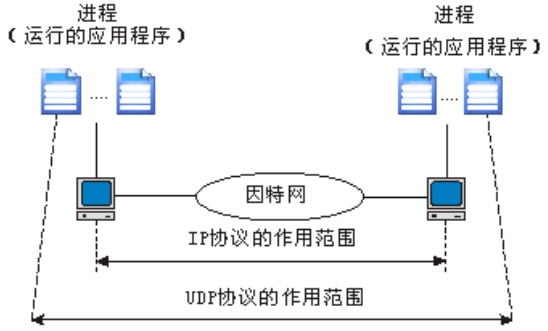

图 6-1UDP 与 IP 的区别

1.端口号

在网络中，主机是用 IP 地址来标识的。而要标识主机中的进程，就需要第二个标识符，这就是端口号。在 TCP/IP 协议族中，端口号是在 0～65535 之间的整数。

在客户/服务器模型中，客户程序使用端口号标识自己，这种端口号叫做短暂端口号，短暂的意思是生存时间比较短。一般把短暂端口取为大于 1023 的数，这样可以保证客户程序工作得比较正常。

服务器进程也必须用一个端口号标识自己。但是这个端口号不能随机选取。如果服务器随机选取端口号，那么客户端在想连接到这个服务器并使用其服务的时候就会因为不知道这个端口号而无法连接。TCP/IP 协议族采用熟知端口号的办法解决这个问题。每一个客户进程都必须知道相应的服务器进程熟知端口号。

UDP 的熟知端口号如下表所示：

表 6-1UDP 的熟知端口号

端口	协议	说明
7	Echo	把收到的数据报回送到发送端
9	Discard	丢弃收到的任何数据报
11	Users	活跃的用户
13	Daytime	返回日期和时间
17	Quote	返回日期和引用（译者注：可参阅 RFC856）
19	Chargen	返回字符串
53	Nameserver	域名服务
67	Bootps	下载引导程序信息的服务器端口
68	Bootpc	下载引导程序信息的客户端口
69	TFTP	简单文件传送协议
111	RPC	远程过程调用
123	NTP	网络时间协议
161	SNMP	简单网络管理协议
162	SNMP	简单网络管理协议（陷阱）
520	RIP	路由信息协议

在一个 IP 数据包中，目的 IP 地址和端口号起着不同的寻址作用。目的 IP 地址定义了在世界范围内惟一的一台主机。当主机被选定后，端口号定义了在这台主机上运行的多个进程中的一个。

2.套接字地址

一个 IP 地址与一个端口号结合起来就叫做一个套接字地址。客户套接字地址惟一地定义了客户进程，而服务器套接字地址惟一地定义了服务器进程。

要使用 UDP 的服务，就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。客户套接字地址指定了客户端的 IP 地址和客户进程，服务器套接字地址指定了服务器的

IP 地址和服务器进程。

二.面向连接的服务与面向无连接的服务

从通信的角度来看，在 OSI 参考模型中，下层能向上层提供两种不同形式的服务：面向连接的服务和面向无连接的服务。

1.面向连接的服务

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。面向连接服务在进行数据交换前，先建立连接。当数据传输结束后，应释放这个连接。因此，采用面向连接的服务进行数据传送要经历三个阶段：

(1)建立连接阶段：在有关的服务原语以及协议数据单元中，必须给出源用户和目的用户的完整地址。同时可以协商服务质量和其它一些选项。

(2)数据交换阶段：在这个阶段，每个报文中不必包含完整的源用户和目的用户的完整地址，而是使用一个连接标识符来代替。由于连接标识符相对于地址信息要短得多，因此使控制信息在报文中所占的比重相对减小，从而可减小系统的额外开销，提高信道的有效利用率。另外，报文的发送和接收都是按固定顺序的，即发送方先发送的报文，在接受方先收到。

(3)释放连接阶段：通过相应的服务原语完成释放操作。

从面向连接服务的三个阶段来看，连接就像一个管道，发送端在其一端依次发送报文，接收者依次在其另一端按同样的顺序接收报文。这种连接又称虚拟电路。它可以避免报文的丢失、重复和乱序。

若两个用户经常需要通信，则可以建立永久虚电路。这样可以免除每次通信时建立连接和释放连接这两个阶段。这点与电话网中的专线很相似。

2.面向无连接的服务

在面向无连接服务的情况下，两个实体之间的通信不必事先建立一个连接。相对于面向连接的服务，面向无连接服务灵活方便且快速。但它不能防止报文的丢失、重复和乱序。由于它的每个报文必须包括完整的源地址的目的地址，因此开销较大。

面向无连接服务主要有三种类型：

(1)数据报：它的特点是发完报文就结束，而对方不做任何响应。数据报的服务简单，额外开销少，但可靠性差，它比较适合于数据具有很大的冗余度以及要求有较高的实时性的通信场合。

(2)证实交付：又称可靠的数据报。这种服务对每一个报文产生一个证实给发送方，不过这种证实不是来自对应方用户，而是来自提供服务的层。这种证实只能保证报文已经发给目的站了，而不能保证对应方用户正确地接收到报文。

(3)请求回答：这种类型服务是接收端用户每收到一个报文，即向发送端用户发送一个应答报文。但是双方发送的报文都有可能丢失。如果接收端发现报文有错误，则回送一个表示有错误的报文。

三.UDP 协议简介

UDP（用户数据报协议），主要用来支持那些需要在计算机之间传输数据的网络应用。包括网络视频会议系统在内的众多的客户/服务器模式的网络应用都需要使用 UDP 协议。

UDP 协议从问世至今已经被使用了很多年，虽然其最初的光彩已经被一些类似的协议所掩盖，但是即使是在今天，UDP 仍然不失为一项非常实用和可行的网络传输层协议。

UDP 协议直接位于 IP 协议的上层。根据 OSI 参考模型,UDP 和 TCP 都属于传输层协议。UDP 协议不提供端到端的确认和重传功能，它不保证数据包一定能到达目的地，因此是不可靠协议。

UDP 协议有以下特点：

●UDP 是面向事务的协议，它用最少的传输量为应用程序向其它程序发送报文提供了一个途径。

●UDP 是无连接的、不可靠的传输机制。在发送数据报前，UDP 在发送和接收两者之间不建立连接。

●UDP 让应用程序能直接访问网络层的数据报服务，例如分段和重组等网络层所提供的数据报服务。

●UDP 使用 IP 协议作为数据传输机制的底层协议。

●UDP 报头和数据都以与最初传输时相同的形式被传送到最终目的地。

●UDP 不提供确认，也不对数据的到达顺序加以控制。因此 UDP 报文可能会丢失。

●不实现数据包的传送和重复检测。

●当数据包在传送过程中发生错误时，UDP 不能报告错误。

●吞吐量不受拥塞控制算法的调节，只受应用程序生成数据的速率、传输带宽、发送端和接收端主机性能的限制。

四.UDP 报文格式

下图显示了 UDP 报文格式。每个 UDP 报文称为一个用户数据报（UserDatagram），用户数据报分为两个部分：UDP 首部和 UDP 数据。首部被分为四个 16 位的字段，分别代表源端口号、目的端口号、报文的长度以及 UDP 校验和。

源端口（16 位）	目的端口（16 位）
有效负载长度（16 位）	校验和（16 位）
数据	
……	

图 6-2UDP 报文格式

●源端口：该字段表示发送端的端口号。如果源端口没有使用，那么此字段的值就被指定为 0。这是一个可选的字段。不同的应用程序使用不同的端口号，UDP 协议使用端口号为不同的应用程序保留其各自的数据传输通道，从而实现了同一时间段内多个应用程序可以一起使用网络进行数据的发送和接收。

●目的端口：该字段表示数据包被发往的目的端的端口号。

●有效负载长度:该字段表示包括 UDP 首部和 UDP 数据在内的整个用户数据报的长度。该字段的最小值是 8。数据报的最大尺寸随操作系统的不同而不同。在两字节字段中，理论上数据报最多可达 65535 字节。然而，一些 UDP 实现将数据报的大小限制到了 8192 字节。

●校验和：UDP 的校验的校验范围包括伪首部（IP 首部一部分字段）、UDP 首部和 UDP

数据，该字段是可选的。如果该字段值为零就说明不进行校验。

五.UDP 封装

当进程有报文要通过 UDP 发送时，它就把这个报文连同一对套接字地址以及数据的长度传递给 UDP。UDP 收到数据后就加上 UDP 首部。然后 UDP 就把这用户数据报连同套接字地址一起传递给 IP。IP 加上自己的首部，在高层协议类型字段使用值 17，指出该数据是从 UDP 协议来的。这个 IP 数据报再传递给数据链路层。数据链路层接收到 IP 数据报后，加上自己的首部（可能还有尾部），再传给物理层。物理层把这些位编码为电信号或光信号，把它发送到远程的主机。如下图所示：

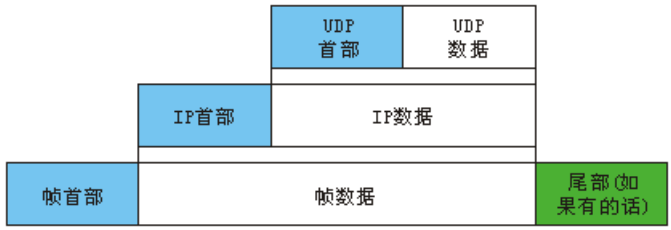

图 6-3UDP 封装

六.UDP 校验和

UDP 校验和的计算与 IP 和 ICMP 校验和的计算不同。UDP 校验和校验的范围包括三部分：伪首部、UDP 首部以及从应用层来的数据。

伪首部是 IP 首部的一部分，其中有些字段要填入 0。用户数据报封装在 IP 数据包中。如下图所示：

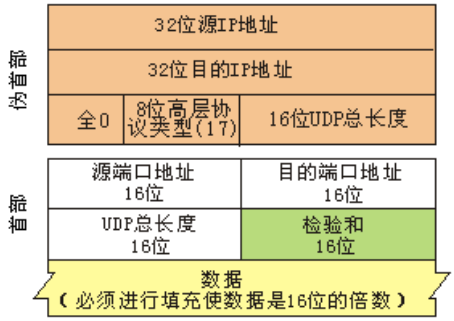

图 6-4 伪首部添加在 UDP 数据报上

若校验和不包括伪首部，用户数据报也可能是安全的和正确的。但是，若 IP 首部受到损伤，则它可能被交付到错误的主机。

伪首部中包含高层协议类型字段是为了确保这个数据包是属于 UDP 而不是属于 TCP（参见实验七）的。使用 UDP 的进程和使用 TCP 的进程可以使用同一个端口号。UDP 的高层协议类型字段是 17。若在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 就可丢弃这个数据包。这样就不会交付给错误的协议。

1.在发送端的校验和计算

在发送端按以下步骤计算校验和：

(1)把伪首部填加到 UDP 用户数据报上。

(2)把校验和字段填入零。

(3)按 16 位长度将数据报分段。

(4)若分段总数不是偶数，则增加一个分段的填充（全 0）。填充只是为了计算校验和，计算完毕后就把它丢弃。

(5)把所有 16 位的分段使用反码算术运算相加。

(6)把得到的结果取反码，它是一个 16 位的数，把这个数插入到校验和字段。

(7)把伪首部和填充丢掉。

(8)把 UDP 用户数据报交付给 IP 进行封装。

在伪首部中的各行的顺序对校验和的计算没有任何影响。此外，增加 0 也不影响计算的结果。

下图给出了一个计算 UDP 校验和的例子。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。

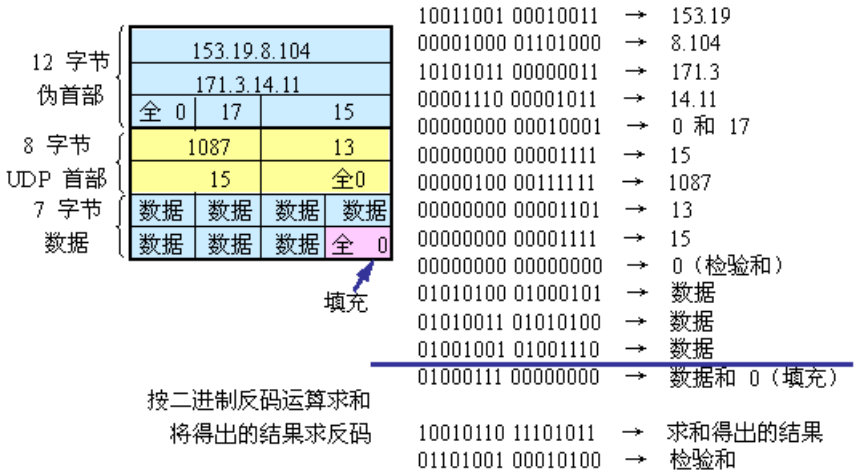

图 6-5UDP 校验和的计算过程

2.在接收端的校验和计算

接收端按以下 6 个步骤计算校验和是否正确：

(1)把伪首部加到 UDP 用户数据报上。

(2)若需要，就增加填充。

(3)把数据报按 16 位长度分段。

(4)把所有 16 位的分段使用反码算术运算相加。

(5)把得到的结果取反码。

(6)若得到的结果是全零，则丢弃首部和填充，并接受这个用户数据报。若结果是非零，就丢弃这个用户数据报。

校验和是可选使用的，若不计算校验和，则校验和字段就填入 0。

七.UDP 应用

下面列出了 UDP 协议的一些用途：

●UDP 适用于这样的进程，它需要简单的请求——响应通信，而较少考虑流量控制和差错控制。对于需要传送成块数据的进程，如 FTP，则通常不使用 UDP；

●UDP 适用于具有内部流量控制和差错控制机制的进程；

●对多播和广播来说，UDP 是个比较合适的传输层协议；

●UDP 可用于管理进程，如 SNMP 协议；

●UDP 可用于某些路由选择更新协议，如路由信息协议（RIP 协议，参考实验 17）。

八.协议栈实现代码解析

本实验将通过对安装目录 JLCSS\ExpCNC\Work\NPL\ExpNPL_studentnet\netproto_udp_student\netproto_udp_student 下的 proto_udp_student.h、netproto_udp_shudent.c 和安装目录 JLCSS\ExpCNC\Work\NPL\ExpNPL_studentnet\netproto_udp_student\netproto_udpif_student 下的 netproto_udpif_student.h、netproto_udpif_student.c 四个文件进行编码，完成协议栈中 UDP 协议的实现。

netproto_udp_student.h 和 netproto_udp_shudent.c 文件用于实现 UDP 数据包发送和接收。其中，netproto_udp_student.h 文件中定义了 UDP 协议实现相关数值以及 UDP 的负载内容、负载长度，关键代码如下所示：

```
#defineUDP_SOUR_PORT5893/**<源端口号*/
#defineUDP_DEST_PORT5893/**<目的端口号*/
#defineIP_DEST_ADDR"0.0.0.0"/**<目的 IP 地址*/
#definePAYLOAD_DATA"Hello,World!"/**<有效负载*/
#definePAYLOAD_LENsizeof(PAYLOAD_DATA)/**<有效负载长度*/
```

这段代码定义了 5 个宏，他们代表的含义如下表所示：

表 6-2netproto_udp_student.h 中定义的宏

宏	值	描述
UDP_SOUR_PORT	5893	定义 UDP 包头中“源端口”字段的值
UDP_DEST_PORT	5893	定义 UDP 包头中“目的端口”字段值
IP_DEST_ADDR	"0.0.0.0"	指定目的 IP 地址
PAYLOAD_DATA	"Hello,World!"	自定义 UDP 负载内容
PAYLOAD_LEN	sizeof(PAYLOAD_DATA)	自定义 UDP 负载的长度

在实验的编码过程中，应该使用这些宏对相应的变量进行赋值。

netproto_udp_shudent.c 文件是协议栈中 UDP 数据包发送和接收的实现部分，其中定义了 2 个函数。下面介绍这些协议栈的实现部分。

函数 netp_udp_output_student 的功能是构造并发送一个 UDP 数据包，其高层协议为自定义协议类型，负载内容为自定义负载。这个函数的编码工作需要由学生完成。

当有数据到达本机网络接口时，函数 netp_udp_input_student 将被调用，并传递给这个函数原始数据。在该函数中，需要判断一些条件值来确定接收到的数据包为自定义 UDP 数据，如果是自定义 UDP 数据包，则输出负载内容，如果不是，则返回 NETP_PUSH_TO_LWIP 交给协议栈继续处理。

netproto_udpif_student.h 和 netproto_udpif_shudent.c 文件用于实现 UDP 上层投递的功能，即为高层使用 UDP 协议提供了接口。其中，netproto_udpif_student.h 文件中并没有定义太多内容。netproto_udpif_shudent.c 文件是协议栈中 UDP 上层投递的功能的实现部分，其中定义了一个全局变量 recv_port 和 2 个函数。

全局变量 recv_port 的作用很简单，它记录了发送 UDP 数据报时的源端口号作为接收 UDP 数据报的过滤条件。

函数 netp_send_udp 通过 IP 层接口发送 UDP 数据报，该函数功能需要学生完成。

函数 netp_udp_input_student 处理输入数据包，如果输入的数据报满足过滤条件，则投递给上层协议使用。该函数功能需要学生完成。

在编码过程中可能会设计到一些结构体、宏和函数，下表是对他们进行介绍：

表 6-3 实验涉及的结构体、宏和函数

结构体/宏/函数	声明或定义	描述

struct netp_eth_header	structnetp_eth_header{
	u8_tdest_address[ETH_ADDRESS_LEN];
	u8_tsour_address[ETH_ADDRESS_LEN];
	u16_ttype;
	};
		以太网帧头结构

structnetp_ip_header	structnetp_ip_header{
	u8_theaderlen:4;
	u8_tversion:4;
	u8_tdiff_services;
	u16_ttotal_length;
	u16_tidentification;
	u16_tflags_offset;
	u8_ttime_to_live;
	u8_tprotocol;
	u16_theader_checksum;
	structip_addrsource_address;
	structip_addrdestination_address;
	};
		ipv4 包头结构

structnetp_udp_header	structnetp_udp_header{
	u16_tsour_port;
	u16_tdest_port;
	u16_tlength;
	u16_tchecksum;
	};
		UDP 报头结构

structnetp_udp_pseudo	structnetp_udp_pseudo{ u32_tsour_addr; u32_tdest_addr; u8_tzero; u8_tprotocol; u16_tlength;	UDP 伪首部结构

结构体/宏/函数	声明或定义	描述
	};	

structin_addr	structin_addr{
	u32_ts_addr;
	};
		32 位地址

UDP_HEADER_LEN	#defineUDP_HEADER_LEN8	UDP 包头长度
PAYLOAD_DATA	#definePAYLOAD_DATA"Hello,EXPcns!"	UDP 负载内容
PAYLOAD_LEN	#definePAYLOAD_LENsizeof(PAYLOAD_DATA)	UDP 负载长度
NETP_UDP_PSEUDO_LEN	#defineNETP_UDP_PSEUDO_LEN sizeof(structnetp_udp_pseudo)	UDP 伪首部长度
ETH_HEADER_LEN	#defineETH_HEADER_LEN14	以太网帧头长度
IP_HEADER_LEN	#defineIP_HEADER_LEN20	IP 包头长度
netp_current_udp_addr	u32_tnetp_current_udp_addr();	获取当前正在使用的网络接口的 UDP 地址
htons	u16_thtons(u16_tn);	将 16 位数值由主机字节序转换为网络字节序
inet_addr	u32_tinet_addr(constchar*cp);	将 ASCII 编码的 Internet 地址转换成为网络字节序地址
inet_chksum	u16_tinet_chksum(void*dataptr,u16_tlen)	计算校验和

九.各模块推荐流程

1.UDP 数据包发送流程

编码实现 UDP 数据包发送推荐使用如下流程：

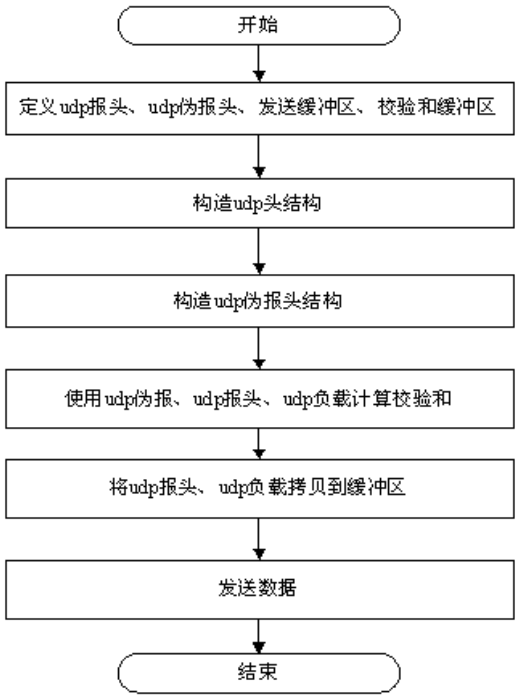

图 6-6UDP 数据包发送推荐流程

2.输入 UDP 数据包处理流程

编码实现处理 UDP 输入数据包推荐使用如下流程：

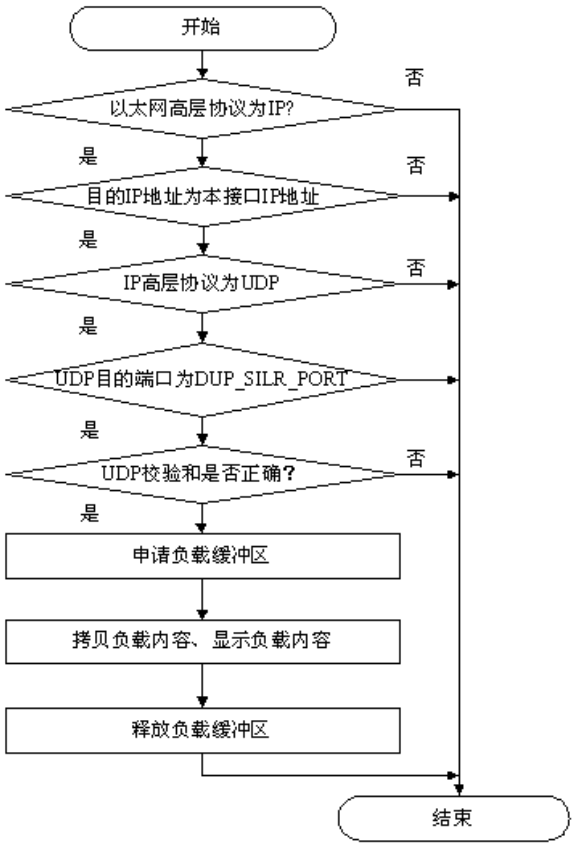

图 6-7 处理 UDP 输入数据包推荐流程

3.UDP 发送接口实现流程

编码实现 UDP 发送接口推荐使用如下流程：

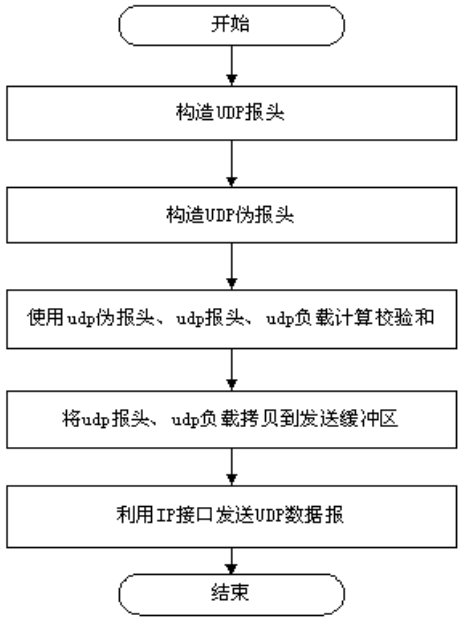

图 6-8UDP 发送接口实现推荐流程

4.UDP 接收接口实现流程

编码实现 UDP 接收接口推荐使用如下流程：

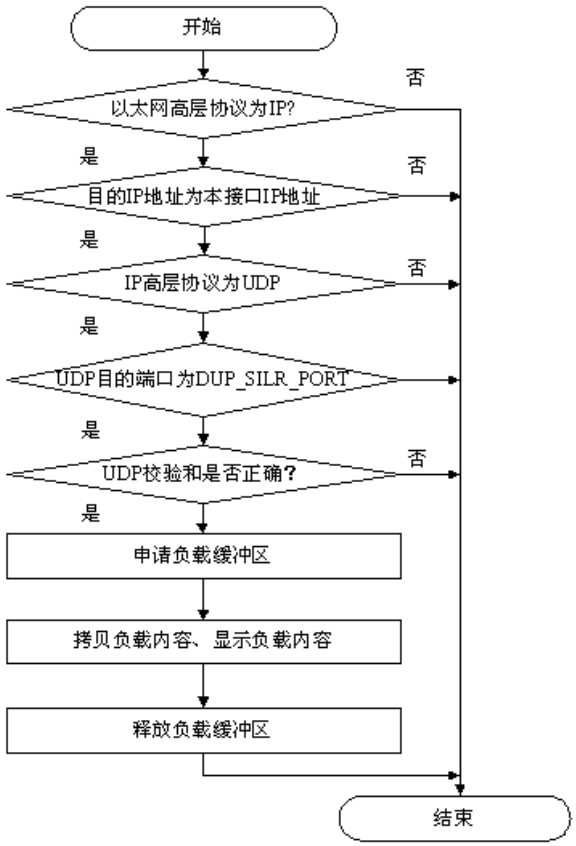

图 6-9UDP 接收接口实现推荐流程

【实验步骤】

练习 1 编辑并发送 UDP 数据报

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机 A 和 B 作为一组，主机 C 和 D 作为一组，主机 E 和 F 作为一组。现仅以主机 A、B 所在组为例，其它组的操作参考主机 A、B 所在组的操作。

1.主机 A 打开协议编辑器，编辑发送给主机 B 的 UDP 数据报。

MAC 层：

目的 MAC 地址：接收方 MAC 地址

源 MAC 地址：发送方 MAC 地址

协议类型或数据长度：0800，即 IP 协议

IP 层：

总长度：包括 IP 层、UDP 层和数据长度

高层协议类型：17，即 UDP 协议

首部校验和：其它所有字段填充完毕后填充此字段

源 IP 地址：发送方 IP 地址

目的 IP 地址：接收方 IP 地址

UDP 层：

源端口：1030

目的端口：大于 1024 的端口号

有效负载长度：UDP 层及其上层协议长度

其它字段默认，计算校验和。

●UDP 在计算校验和时包括哪些内容？

2.在主机 B 上启动协议分析器捕获数据，并设置过滤条件（提取 UDP 协议）。

3.主机 A 发送已编辑好的数据报。

4.主机 B 停止捕获数据，在捕获到的数据中查找主机 A 所发送的数据报。

思考问题：

1.为什么 UDP 协议的“校验和”要包含伪首部？

2.比较 UDP 和 IP 的不可靠程度？

练习 2UDP 单播通信

本练习将主机 A、B、C、D、E、F 作为一组进行实验。

1.主机 B、C、D、E、F 上启动实验平台工具栏中的 UDP 工具”，作为服务器端，监听端口设置为 2483，“创建”成功。

2.主机 C、E 上启动协议分析器开始捕获数据，并设置过滤条件（提取 UDP 协议）。

3.主机 A 上启动实验平台工具栏中的“UDP 工具”，作为客户端，以主机 C 的 IP 为目的 IP 地址，以 2483 为端口，填写数据并发送。

4.察看主机 B、C、D、E、F 上的“UDP 工具”接收的信息。

●哪台主机上的“UDP 工具”能够接收到主机 A 发送的 UDP 报文？

5.察看主机 C 协议分析器上的 UDP 报文，并回答以下问题：

●UDP 是基于连接的协议吗？阐述此特性的优缺点。

●UDP 报文交互中含有确认报文吗？阐述此特性的优缺点。

6.主机 A 上使用协议编辑器向主机 E 发送 UDP 报文，其中：

目的 MAC 地址：E 的 MAC 地址

目的 IP 地址：主机 E 的 IP 地址

目的端口：2483

校验和：0

有效负载长度：UDP 层及其上层协议长度

首部校验和：其它所有字段填充完毕后填充此字段

总长度：包括 IP 层、UDP 层和数据长度

发送此报文，并回答以下问题：

●主机 E 上的 UDP 通信程序是否接收到此数据包？UDP 是否可以使用 0 作为校验和进行通信？

7. 主机 B、C、D、E、F 关闭服务端，主机 A 关闭客户端。

思考问题：

1. 思考 UDP 的差错处理能力。

练习 3UDP 广播通信

本练习将主机 A、B、C、D、E、F 作为一组进行实验。

1. 主机 B、C、D、E、F 上启动实验平台工具栏中的“UDP 工具”，作为服务器端，监听端口设为 2483。
2. 主机 B、C、D、E、F 启动协议分析器捕获数据，并设置过滤条件（提取 UDP 协议）。
3. 主机 A 上启动 UDP 工具，作为客户端，以 255.255.255.255 为目的地址，以 2483 为端口，填写数据并发送。
4. 察看主机 B、C、D、E、F 上的“UDP 工具”接收的信息。

●哪台主机能够接收到主机 A 发送的 UDP 报文？

5. 察看协议分析器上捕获的 UDP 报文，并回答以下问题：

●主机 A 发送的报文的目的 MAC 地址和目的 IP 地址的含义是什么？

思考问题：

1. 如果将目的 MAC 地址换成某一个主机的 MAC 地址，是否所有主机还会收到这种报文？

2. 如果将目的 MAC 地址设成广播地址，目的 IP 设成某一主机的 IP 地址，结果怎样？

3. 在可靠性不是最重要的情况下，UDP 可能是一个好的传输协议。试给出这种特定情况的一些示例。

4. UDP 协议本身是否能确保数据报的发送和接收顺序？

实验 7 传输控制协议（TCP）

【实验目的】

1. 掌握 TCP 协议的报文格式
2. 掌握 TCP 连接的建立和释放过程
3. 掌握 TCP 数据传输中编号与确认的过程
4. 掌握 TCP 协议校验和的计算方法
5. 理解 TCP 重传机制

【学时分配】

4 学时

【实验环境】

该实验采用网络结构一

【实验原理】

一. TCP 协议简介

TCP（传输控制协议）协议是 TCP/IP 协议族中的面向连接的、可靠的传输层协议。TCP 与 UDP 不同，它允许发送和接收字节流形式的数据。为了使服务器和客户端以不同的速度发送和接收数据，TCP 提供了发送和接收两个缓冲区。TCP 提供全双工服务，数据同时能双向流动。通信的每一方都有发送和接收两个缓冲区，可以双向发送数据。TCP 在报文中加上一个递增的确认序列号来告诉发送端，接收端期望收到的下一个报文，如果在规定时间内，没有收到关于这个包的确认响应，则重新发送此包，这保证了 TCP 是一种可靠的传输层协议。

TCP 的常用熟知端口如下表所示：

表 7-1TCP 常用熟知端口

端口	协议	端口	协议
20	FTPData	80	HTTP
21	FTPControl	110	POP3
23	TELNET	143	IMAP
25	SMTP		

二. TCP 报文格式

TCP 报文的格式如下图所示：

源端口（16 位）	目的端口（16 位）
序列号（32 位）	
确认号（32 位）	

首部长度（4 位）	保留（4 位）	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	窗口大小（16 位）
校验和（16 位）										紧急指针（16 位）
选项和填充										

图 7-1TCP 报文格式

TCP 报文包括 20～60 字节的首部，接着是应用程序的数据部分。首部在没有选项时是 20 字节，而当有选项时长度会增加，但是最大不会超过 60 字节。

●源端口：该字段定义了在主机中发送这个报文的应用程序端口号。

●目的端口：该字段定义了数据报发往的主机中接收这个报文的应用程序的端口号。

●序列号：该字段定义了指派给本报文第一个数据字节的一个序号。TCP 是流式传输协议，为了保证连通性，要在发送的每一个字节上编号。序号指定了这个序列中的哪一个字节是报文的第一个字节。在连接建立时，双方使用随机数产生器产生初始序号，通常每一方的初始序号都是不同的。

●确认号：该字段定义了报文的接收端期望从对方接收的序号。如果报文的接收端成功地接收了对方发来的序号为 x 的报文，它就把确认号定义为 x+1。确认可以和数据一起发送。

●首部长度：该字段指定 TCP 首部的长度，以 4 字节为单位。首部长度可以在 20～60 字节之间。因此，这个字段的值可以在 5 至 15 之间。

●保留：这是 6 位字段，保留为今后使用。

●控制：这个字段定义了 8 种不同的标志。如下图所示。在同一时间可设置一位或多位标志。

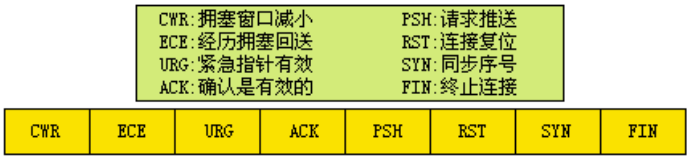

图 7-2 控制字段

这些标志用在 TCP 的流量控制、连接建立和终止以及数据传送的方式等方面。下表给出了每一位的简要说明。

表 7-2TCP 标志位

标志	说明
CWR	拥塞窗口减小（用来表明发送主机接收到了设置 ECE 标志的 TCP 包。拥塞窗口是被 TCP 维护的一个内部变量，用来管理发送窗口大小）
ECE	经历拥塞回送（用来在 TCP3 次握手时表明一个 TCP 端是具备 ECN 功能的，并且表明接收到的 TCP 包的 IP 头部的 ECN 被设置为 11）
URG	紧急指针字段值有效

标志	说明
ACK	确认字段值有效
PSH	推送数据
RST	连接必须复位
SYN	在连接建立时对序号进行同步
FIN	终止连接

●窗口大小：该字段定义对方必须维持的窗口值（以字节为单位）。这个字段的长度是 16 位，因此窗口值的最大长度是 65535 字节。这个值通常是作为接收窗口，并由接收端来确定。这时，发送端必须服从接收端的决定。

●校验和：该字段的校验范围包括伪首部、TCP 首部和 TCP 数据部分。

●紧急指针：只有当紧急标志置位时，这个 16 位字段才有效，这时的报文中包括紧急数据。

●选项：在 TCP 首部中可以有多达 40 字节的可选信息。

三. TCP 封装

TCP 报文封装在 IP 数据报中，然后再封装成数据链路层中的帧，如下图所示：

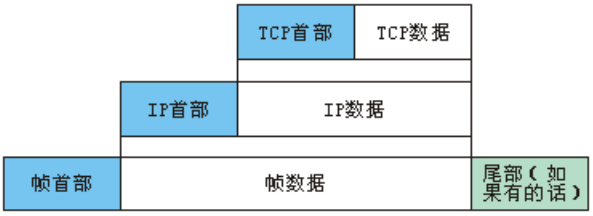

图 7-3TCP 封装

四. TCP 校验和

TCP 的校验和与 UDP 的校验和计算过程是一样的。但是，UDP 是否使用校验和是可选的，而 TCP 是否使用校验和则是强制性的。在计算 TCP 校验和时也要在报文上添加伪首部。对于 TCP 的伪首部，高层协议类型字段的值是 6。如下图所示：

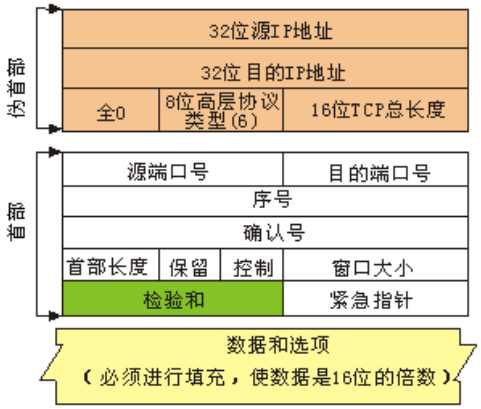

图 7-4 伪首部加到 TCP 报文上

五. TCP 连接建立与释放

1. 连接建立

TCP 以全双工方式传送数据。当两个进程建立了 TCP 连接后，它们能够同时向对方发送数据。在传送数据之前，双方都要对通信进行初始化，得到对方的认可。

2. 三次握手

TCP 的连接建立过程叫做三次握手。服务器程序首先准备好接受 TCP 连接，这个过程叫做被动打开请求。这时，服务器的 TCP 就已准备好接受任何一台主机的 TCP 连接了。

客户程序发出 TCP 连接请求的过程叫做主动打开。然后服务器与客户端就开始三次握手过程，如下图所示（在图中客户端与服务器端各使用一条时间线，并给出每个阶段的几个重要字段，包括序号、确认号、控制标志以及非零的窗口值）。这个过程有以下 3 个步骤。

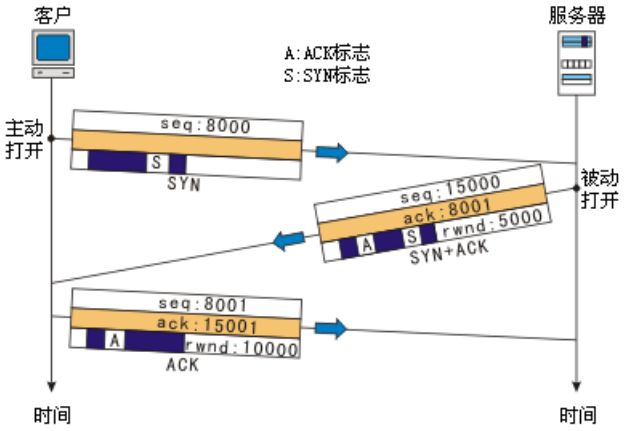

图 7-5 使用三次握手的连接建立

(1)客户发送第一个报文，这是一个 SYN 报文，在这个报文中只有 SYN 标志置为 1。这个报文的作用是使序号同步。

(2)服务器发送第二个报文，即 SYN+ACK 报文，其中 SYN 和 ACK 标志被置为 1。这个报文有两个目的。首先，它是一个用来和对方进行通信的 SYN 报文。服务器使用这个报文同步初始序号，以便从服务器向客户发送字节。服务器还使用 ACK 标志确认已从客户端收到了 SYN 报文，同时给出期望从客户端收到的下一个序号。另外，服务器还定义了客户端要使用的接收窗口的大小。

(3)客户发送第三个报文。这仅仅是一个 ACK 报文。它使用 ACK 标志和确认号字段来确认收到了第二个报文。

3. 连接终止

通信双方中的任何一方都可以关闭连接。当一方的连接被终止时，另一方还可继续向对方发送数据。TCP 的连接终止有两种方式：三次握手和具有半关闭的四次握手。

4. 三次握手方式终止连接

使用三次握手的 TCP 终止过程如下图所示：

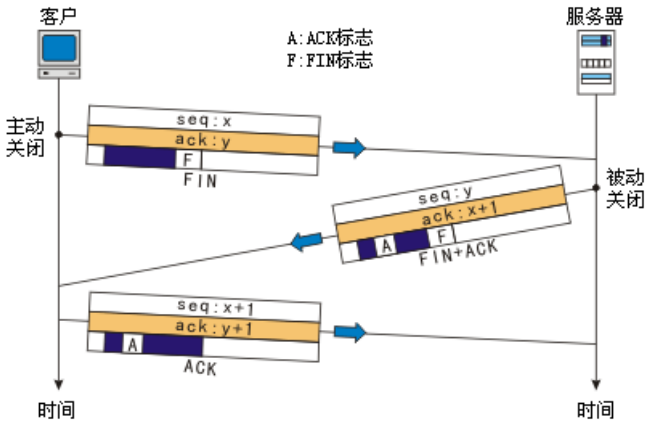

图 7-6 使用三次握手的连接终止

(1)当客户端想关闭 TCP 连接时，它发送一个 TCP 报文，把 FIN 标志位设置为 1。

(2)服务器端在收到这个 TCP 报文后，把 TCP 连接即将关闭的消息发送给相应的进程，并发送第二个报文——FIN+ACK 报文，以证实从客户端收到了 FIN 报文，同时也说明，另一个方向的连接也关闭了。

(3)客户端发送最后一个报文以证实从 TCP 服务器收到了 FIN 报文。这个报文包括确认号，它等于从服务器收到的 FIN 报文的序号加 1。

5.半关闭的四次握手方式终止连接

在 TCP 连接中，一方可以终止发送数据，但仍然保持接收数据，这就叫做半关闭。半关闭通常是由客户端发起的。图 7-7 描绘了半关闭的过程。客户发送 FIN 报文，半关闭了这个连接。服务器发送 ACK 报文接受这个半关闭。但是，服务器仍然可以发送数据。当服务器已经把所有处理的数据都发送完毕时，就发送 FIN 报文，客户端发送 ACK 报文给予确认。

在半关闭一条连接后，客户端仍然可以接收服务器发送的数据，而服务器也可以接收客户端发送的确认。但是，客户端不能传送数据给服务器。

六.流量控制

在发送端收到接收端的确认报文之前，流量控制可以对发送端发送的数据量进行管理。

在不考虑流量控制的情况下，传输层协议可以每次只发送一个字节的数据，然后在发送下一个字节数据之前等待接收端的确认报文。这是一个非常缓慢的过程，如果数据要走很长的距离，发送端就要在等待确认报文时一直处在空闲状态。还有一种情况是传输层协议一次就将全部数据发送出去，而不理会确认报文。这样虽然加速了发送过程，但可能会使接收端来不及接收而瘫痪。此外，若有一部分数据丢失、重复、失序或损坏，发送端就要一直等到接收端将全部数据都检查完毕后才能知道。

TCP 的流量控制采用一种折中的方法。它在缓存上定义一个窗口。缓存是用来暂时存放将要发送的数据的。TCP 发送数据的多少由这个窗口决定。

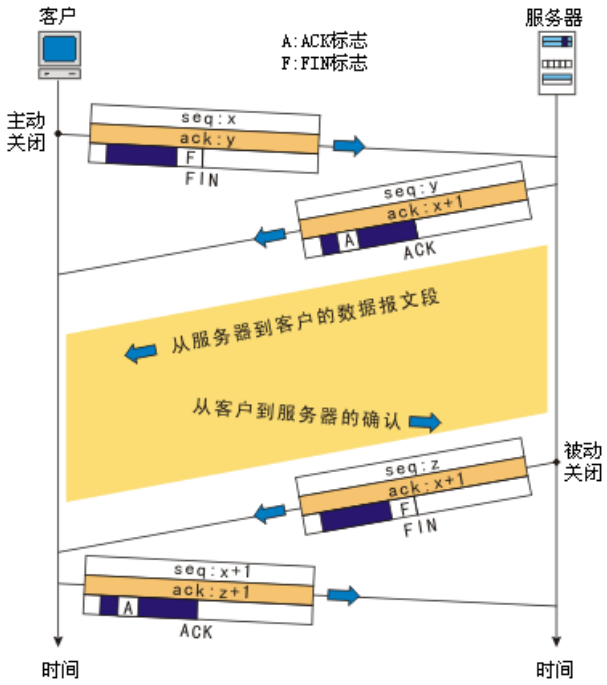

图 7-7 半关闭

1. 滑动窗口协议

为了完成流量控制，TCP 使用滑动窗口协议。窗口覆盖了缓存的一部分，在这个窗口中的数据是可以发送而不必考虑确认的。窗口有两个沿：一个在左边，另一个在右边。因为左沿和右沿都是可以滑动的，所以这个窗口叫做滑动窗口。如下图所示：

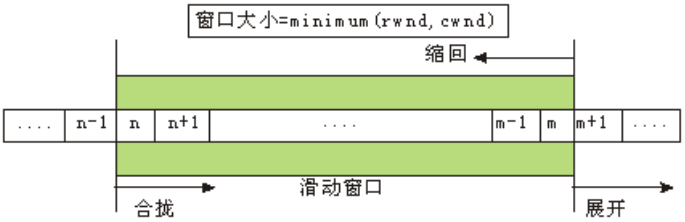

图 7-8 滑动窗口

窗口有三种动作：展开、合拢或缩回。这三种动作受接收端的控制而不是发送端的控制。

展开窗口表示窗口的右沿向右移动，这样就可以从缓存中发送更多的数据。合拢窗口表示窗口的左沿向右移动，这表示某些数据已经被确认了，发送端可以不再担心它们。缩回窗口表示窗口的右沿向左移动，这在某些实现中是不允许的，因为这会使某些可以发送的数据变成不能发送的。如果发送端已经发送了这些字节，就会产生错误。窗口的左沿不能向左移动，因为这表示已经发送出去的并且经过确认的数据现在又要收回了。

窗口大小由接收窗口和拥塞窗口两者中的较小者决定。接收窗口大小由接收方发送的确认报文中的窗口大小字段值所确定。这是接收端在缓存溢出导致数据被丢弃之前所能接受的最大字节数。拥塞窗口大小是由网络根据拥塞情况而确定的。

七. 差错控制

TCP 是可靠的传输层协议。应用程序把数据流交付给 TCP 后，就依靠 TCP 把整个数据流交付给接收端的应用程序，并且保证数据流是按序的、没有差错的、也没有任何一部分是丢

失的或重复的。

TCP 使用差错控制提供可靠性。差错控制包括以下的一些机制：检测受到损伤的报文、丢失的报文、失序的报文和重复的报文。差错控制还包括检测出差错后纠正差错的机制。TCP 的差错检测和差错纠正是通过校验和、确认以及超时重传三种机制实现的。

1.校验和

每一个 TCP 报文都包括校验和字段，用来检查报文是否损坏。若报文损坏，接收端就将报文丢弃，并认为这个报文丢失了。

2.确认

TCP 采用确认报文的方法来证实收到了数据报文。确认报文不携带数据，但消耗一个序号。除了 ACK 报文之外，确认报文也需要被确认。

3.重传

差错控制的核心是报文的重传机制。当一个报文损坏、丢失或延迟时，就需要重传这个报文。有两种情况需要对报文进行重传：当重传超时计时器时间到期时，或当发送端收到了 3 个重复的确认报文时。

(1)重传超时计时器到期之后的重传

发送端为每一个 TCP 报文都设置一个重传超时计时器。若计时器时间到期时还没有收到对这个报文的确认报文，就认为这个报文丢失了，于是重传这个报文，即使可能由于报文延迟到达，或确认报文延迟到达，或确认报文丢失等原因。重传超时计时器的值是动态的，它根据报文的往返时间而更新。报文的往返时间是报文离开发送端到发送端收到此报文的确认报文所需的时间。

(2)三个重复的确认报文之后的重传

一个报文的丢失会导致接收端收到的报文失序，这时接收端会发送对丢失报文的确认报文，当发送端收到 3 个重复的确认报文之后，发送端立即重传这个报文，这叫做快重传。

对不消耗序号的报文不进行重传。对所有 ACK 报文都不进行重传。

【实验步骤】

练习 1 察看 TCP 连接的建立和释放

各主机打开工具区的“拓扑验证工具”，选择相应的网络结构，配置网卡后，进行拓扑验证，如果通过拓扑验证，关闭工具继续进行实验，如果没有通过，请检查网络连接。

本练习将主机 A 和 B 作为一组，主机 C 和 D 作为一组，主机 E 和 F 作为一组。现仅以主机 A、B 为例，其它组的操作参考主机 A、B 的操作。

1.主机 B 启动协议分析器捕获数据，并设置过滤条件（提取 TCP 协议）。

主机 B 在命令行下输入：netstat -a -n 命令来查看主机 B 的 TCP 端口号。

2.主机 A 启动 TCP 工具连接主机 B。

主机 A 启动实验平台工具栏中的“TCP 工具”。选中“客户端”单选框，在“地址”文本框中填入主机 B 的 IP 地址，在“端口”文本框中填入主机 B 的一个 TCP 端口，点击[连接]按钮进行连接。

3.察看主机 B 捕获的数据，填写下表。

表 7-3 实验结果

字段名称	报文 1	报文 2	报文 3
序列号			

确认号			
ACK			
SYN			

●TCP 连接建立时，前两个报文的首部都有一个“最大字段长度”字段，它的值是多少？作用是什么？结合 IEEE802.3 协议规定的以太网最大帧长度分析此数据是怎样得出的。

4.主机 A 断开与主机 B 的 TCP 连接。

5.察看主机 B 捕获的数据，填写下表。

表 7-4 实验结果

字段名称	报文 4	报文 5	报文 6	报文 7
序列号				
确认号				
ACK				
FIN				

●结合步骤 3、5 所填的表，理解 TCP 的三次握手建立连接和四次握手的释放连接过程，理解序号、确认号等字段在 TCP 可靠连接中所起的作用。

思考问题：

1.为什么在 TCP 连接过程中要使用三次握手？如不这样做可能会出现什么情况。

2.解释 TCP 协议的释放过程？