计算机网络实验报告

警示

1.实验报告如有雷同，雷同各方当次实验成绩均以 0 分计。
2.当次小组成员成绩只计学号、姓名登录在下表中的。
3.在规定时间内未上交实验报告的，不得以其他方式补交，当次成绩按 0 分计。
4.实验报告文件以 PDF 格式提交。

专业	计算机科学与技术（超算）	班 级	2019 级行政四班	组长	
学号	19335112				
学生	李钰				

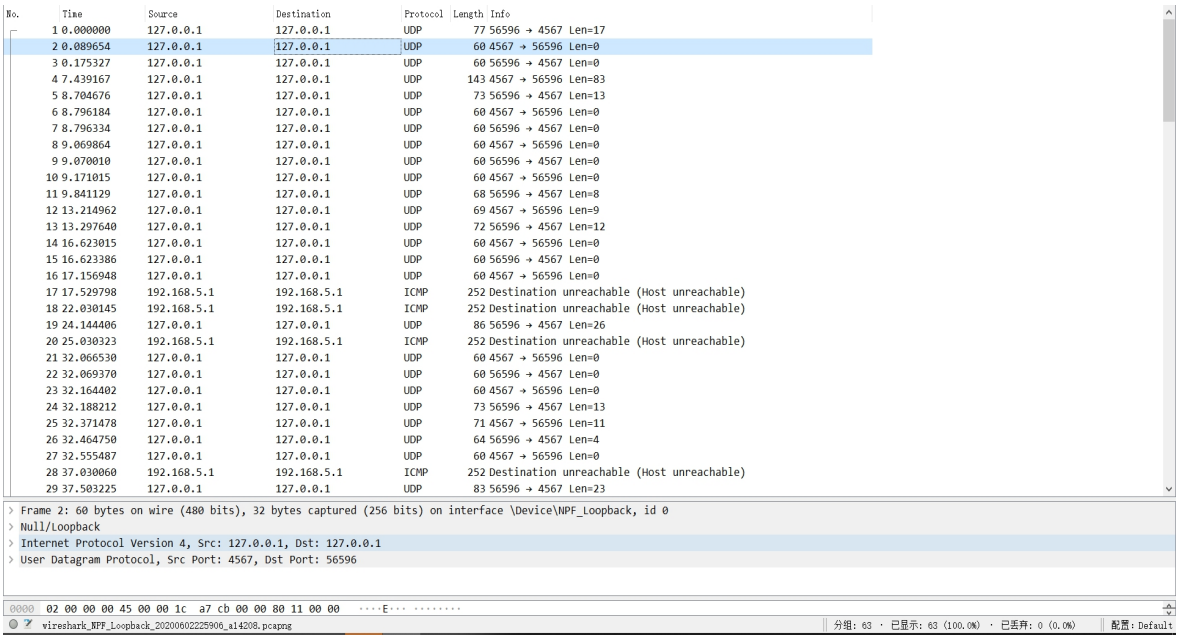
编程实验

【实验内容】

（1） 完成实验教程实例 3-2 的实验（考虑局域网、互联网两种实验环境），回答实验提出的问题及实验思考。（P103）。

【局域网】

在本机使用 ip 地址为 127.0.0.1，抓包情况如下



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	UDP	77	56596 → 4567 Len=17
2	0.089654	127.0.0.1	127.0.0.1	UDP	60	4567 → 56596 Len=0
3	0.175327	127.0.0.1	127.0.0.1	UDP	60	56596 → 4567 Len=0
4	7.439167	127.0.0.1	127.0.0.1	UDP	143	4567 → 56596 Len=83
5	8.704676	127.0.0.1	127.0.0.1	UDP	73	56596 → 4567 Len=13
6	8.796184	127.0.0.1	127.0.0.1	UDP	60	4567 → 56596 Len=0
7	8.796334	127.0.0.1	127.0.0.1	UDP	60	56596 → 4567 Len=0
8	9.069864	127.0.0.1	127.0.0.1	UDP	60	4567 → 56596 Len=0
9	9.070010	127.0.0.1	127.0.0.1	UDP	60	56596 → 4567 Len=0
10	9.171015	127.0.0.1	127.0.0.1	UDP	60	4567 → 56596 Len=0
11	9.841129	127.0.0.1	127.0.0.1	UDP	68	56596 → 4567 Len=8
12	13.214962	127.0.0.1	127.0.0.1	UDP	69	4567 → 56596 Len=9
13	13.297640	127.0.0.1	127.0.0.1	UDP	72	56596 → 4567 Len=12
14	16.623015	127.0.0.1	127.0.0.1	UDP	60	4567 → 56596 Len=0
15	16.623386	127.0.0.1	127.0.0.1	UDP	60	56596 → 4567 Len=0
16	17.156948	127.0.0.1	127.0.0.1	UDP	60	4567 → 56596 Len=0
17	17.529798	192.168.5.1	192.168.5.1	ICMP	252	Destination unreachable (Host unreachable)
18	22.030145	192.168.5.1	192.168.5.1	ICMP	252	Destination unreachable (Host unreachable)
19	24.144406	127.0.0.1	127.0.0.1	UDP	86	56596 → 4567 Len=26
20	25.030323	192.168.5.1	192.168.5.1	ICMP	252	Destination unreachable (Host unreachable)
21	32.066530	127.0.0.1	127.0.0.1	UDP	60	4567 → 56596 Len=0
22	32.069370	127.0.0.1	127.0.0.1	UDP	60	56596 → 4567 Len=0
23	32.164402	127.0.0.1	127.0.0.1	UDP	60	4567 → 56596 Len=0
24	32.188212	127.0.0.1	127.0.0.1	UDP	73	56596 → 4567 Len=13
25	32.371478	127.0.0.1	127.0.0.1	UDP	71	4567 → 56596 Len=11
26	32.464750	127.0.0.1	127.0.0.1	UDP	64	56596 → 4567 Len=4
27	32.555487	127.0.0.1	127.0.0.1	UDP	60	4567 → 56596 Len=0
28	37.030060	192.168.5.1	192.168.5.1	ICMP	252	Destination unreachable (Host unreachable)
29	37.503225	127.0.0.1	127.0.0.1	UDP	83	56596 → 4567 Len=23

从图中可以发现，局域网中的 UDP 协议几乎是不丢包的这是因为局域网网络环境比较好，并且局域网测试时的数据包大多比较小。但是如果调整数据包的的大小以及缓冲区的大小，也可以出现丢包。

【互联网】

利用助教提供的 echo 服务，向 ip 地址为 8.129.101.161 的服务器发送数据，wireshark 捕捉情况如下

ip.addr == 8.129.101.161

No.	Time	Source	Destination	Protocol	Length	Info
169	27.621524	172.19.12.179	8.129.101.161	UDP	49	59225 → 6789 Len=7
170	27.633733	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=7
180	30.685860	172.19.12.179	8.129.101.161	UDP	47	59225 → 6789 Len=5
181	30.699202	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=5
202	32.918592	172.19.12.179	8.129.101.161	UDP	45	59225 → 6789 Len=3
205	32.931226	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=3
220	38.619186	172.19.12.179	8.129.101.161	UDP	49	59225 → 6789 Len=7
221	38.641833	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=7
236	41.687110	172.19.12.179	8.129.101.161	UDP	49	59225 → 6789 Len=7
237	41.707376	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=7
290	55.563936	172.19.12.179	8.129.101.161	UDP	48	59225 → 6789 Len=6
292	55.586084	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=6
301	57.605435	172.19.12.179	8.129.101.161	UDP	50	59225 → 6789 Len=8
302	57.617198	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=8
320	62.051532	172.19.12.179	8.129.101.161	UDP	49	59225 → 6789 Len=7
321	62.065011	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=7
330	63.483469	172.19.12.179	8.129.101.161	UDP	46	59225 → 6789 Len=4
331	63.498191	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=4
386	79.638811	172.19.12.179	8.129.101.161	UDP	49	59225 → 6789 Len=7
387	79.668045	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=7
396	81.429364	172.19.12.179	8.129.101.161	UDP	46	59225 → 6789 Len=4
397	81.513170	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=4
402	82.757331	172.19.12.179	8.129.101.161	UDP	46	59225 → 6789 Len=4
403	82.777547	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=4
414	84.585358	172.19.12.179	8.129.101.161	UDP	47	59225 → 6789 Len=5
415	84.663219	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=5
427	86.200691	172.19.12.179	8.129.101.161	UDP	47	59225 → 6789 Len=5
428	86.218090	8.129.101.161	172.19.12.179	UDP	60	6789 → 59225 Len=5

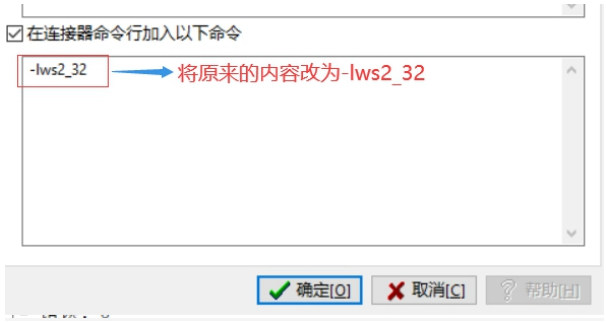
① 说明在实验中遇到的问题和解决方法

1) 编写代码之后，报错

编译器 (12) 资源 编译日志 调试 搜索结果 关闭

行	列	单元	信息
		C:\Users\16435\AppData\Local\Temp\cc391Jnt.o	In function `main':
14		C:\Users\16435\Desktop\study\计网\yanglingling\实验二 网...	undefined reference to `_imp__WSAStartup@8'
16		C:\Users\16435\Desktop\study\计网\yanglingling\实验二 网...	undefined reference to `_imp__socket@12'
19		C:\Users\16435\Desktop\study\计网\yanglingling\实验二 网...	undefined reference to `_imp__WSAGetLastError@0'
26		C:\Users\16435\Desktop\study\计网\yanglingling\实验二 网...	undefined reference to `_imp__htons@4'

上网查询之后发现是编译器环境配置的问题，按照网上的指引在，Dev C++的工具栏中选择编译选项，进行如下修改



即可编译成功，正常运行

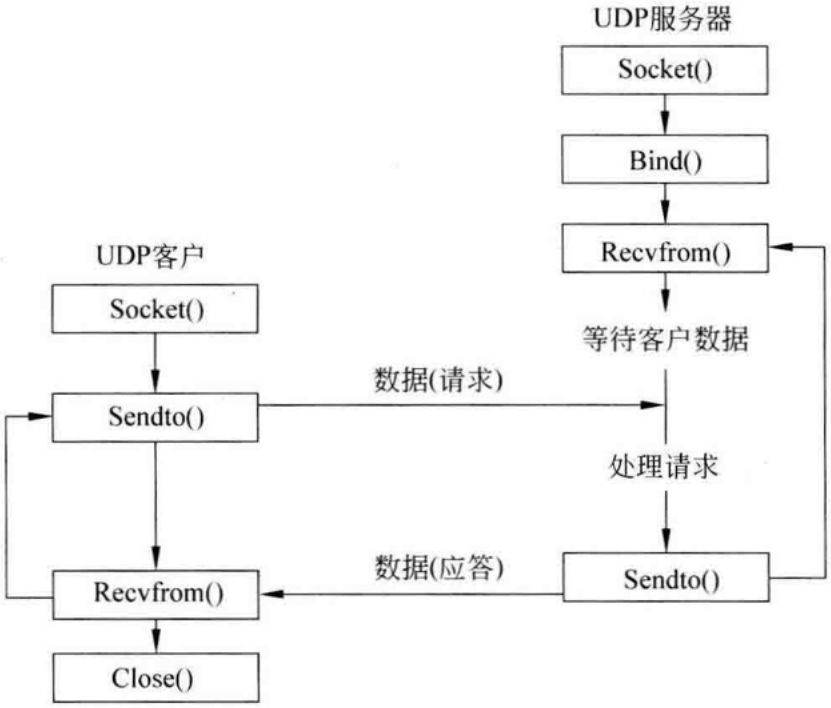
2) 不同环境下的头文件不同

a. 在 Linux 下，头文件使用的是<sys/socket.h>,在 Windows 环境下，使用头文件<winsock.h>或<winsock2.h>代替;

b. 在 Linux/Ubuntu 环境下，使用头文件<arpa/inet.h>,在 Windows 环境下，使用头文件<windows.h>代替;

② 给出程序详细的流程图和对程序关键函数的详细说明

1) 流程图



2) 关键函数说明

a. **Socket（）函数：**调用建立 Socket。为了执行网络输入输出，一个进程必须做的第一件事就是调用 socket 函数获得一个文件描述符。

b. **Bind（）函数：**连接 Socket。为套接口分配一个本地 IP 和协议端口，对于网际协议，协议地址是 32 位 IPv4 地址或 128 位 IPv6 地址与 16 位的 TCP 或 UDP 端口号的组合;如指定端口为 0，调用 bind 时内核将选择一个临时端口，如果指定一个通配 IP 地址，则要等到建立连接后内核才选择一个本地 IP 地址。

c. **Recvfrom（）函数：**从 Socket 接受数据。因为是无连接的协议，所以 recvfrom（）函数不需要建立连接，它对到达相连协议端口的任何数据都做出响应。当 recvfrom（）函数从 Socket 收到一个数据报时，它将保存发送此数据包的进程的网络地址以及数据包本身。

d. **Sendto（）函数：**通过 Socket 发送数据。UDP 使用 sendto()函数发送数据，他类似于标准的 write()，但是在 sendto()函数中要指明目的地址。

e. **Close（）函数：**关闭连接套接字，释放所占有的资源。

③ 使用 Socket APT 开发通信程序中的客户端按程序和服务器时，各需要哪些不同的函数？

1) TCP 协议：

服务器程序：WSAStartup（），socket（），bind（）,listen（），accept（），recv（），send（），closesocket（），WSACleanup（）。

客户端程序：WSAStartup（），socket（），connect（），send（），recv（），closesocket（），WSACleanup（）。

2) UDP 协议：

服务器程序：WSAStartup（），socket（），bind（），recvfrom（），sendto（），closesocket（），WSACleanup（）。

客户端程序：WSAStartup（），socket（），sendto（），recvfrom（），closesocket（），WSACleanup（）。

④ 解释 connect()、bind()等函数中 struct sockaddr *addr 参数各个部分含义，并用具体的数据说明举例。

1) **SOCKET socket(int domain, int type, int protocol)**

参数：协议簇，套接字类型,协议簇中的协议号

返回：新创建的套接字句柄(以下称此套接字为监听套接字)

domain: 协议簇

PF_INET 表示因特网

PF_UNIX 表示 Unix 管道功能

type: 套接字类型

SOCK_STREAM 表示基于连接的字节流方式(如 TCP)

SOCK_DGRAM 表示无连接的数据报方式(如 UDP)

Protocol: 协议簇中的协议号

可以说明为 UNSPEC(unspectified).

domain 和 type 已经可以指定一个传输层协议，如，

TCP(domain=PF_INET, type=SOCK_STREAM)

UDP (domain=PF_INET, type=SOCK_ DGRAM)

2) **int bind(SOCKET socket, struct sockaddr * address, int addr_len)**

参数：套接字，本地地址，地址长度

返回：0（无错时），或错误码

* 本地地址包括服务器端口号

* 有一些端口号已成为标准端口号，如： 80 一般作为 Web 服务器的端口号

* 端口号也可以自己定义，一般使用 2000 以上的端口号

建立半连接，需要明确 address 中关于主机的部分

3) **int listen(SOCKET socket, int backlog)**

参数：监听套接字(已绑定但未联接的套接字)，指定正在等待联接的最大队列长度

返回：0（无错时），或错误码

例如：listen(s,1)表示连接请求队列长度为 1,即只允许有一个请求,若有多个请求（因为服务器一般可以提供多个连接),则出现错误,给出错误代码 WSAECONNREFUSED

4) **SOCKET accept(SOCKET socket, struct sockaddr * address, int *addr_len)**

参数:处于监听模式的套接字,接收成功后返回客户端的网络地址,网络地址长度

返回：一个新的套接字(以下成为连接字)，或 INVALID_SOCKET

accept()阻塞(缺省)等待请求队列中的请求

client 也是一个 sockaddr_in 结构,连接建立时填入请求连接的套接口的半相关信息

5) **int send(SOCKET socket, char *message, int msg_len,int flags)**

参数：连接套接字,缓冲区起始地址，要发送字节数

返回：实际发送的字节数(无错时)，或 SOCKET_ERROR

socket:服务器端监听已经连接的套接字。
Message:指向待发送数据缓冲区的指针。
msg_len:发送数据缓冲区的长度。
Flags:数据发送标记,可为 0、MSG_DONTROUTE 或 MSG_OOB
注意：send()并不保证发送所有请求的数据。它实际发送的字节数由返回值指示。也许需要循环调用 send()来得到需要的结果。

6） **int recv(SOCKET socket, char *message, int msg_len,int flags)**
参数:连接套接字,缓冲区起始地址,缓冲区长度
返回: 实际接收的字节数(无错时), 或 SOCKET_ERROR
socket:准备接收数据的套接字;
message:准备接收数据的缓冲区(用来存储所接收数据的字符串);
msg_len:准备接收数据缓冲区的大小(字符串的长度减 1，留下一个字节用于存放结束符);
flags:数据接收标记,可为 0、MSG_PEEK 或 MSG_OOB;
0:最常用的参数值,它将信息移到指定的字符串，并从缓冲区清除。
MSG_PEEK:只查看数据而不将数据从缓冲区清除。
MSG_OOB:用于 DECnet 协议。

7） int connect(int sockfd, struct sockaddr * serv_addr, int addrlen);
参数：客户端的套接字，服务端的套接字所在的“地方”，该“地方”的大小;
返回：成功则返回 0，失败返回-1，错误原因存于 errno 中;
sockfd：标识一个套接字;
serv_addr：套接字想要连接的主机地址和端口号;
addrlen：name 缓冲区的长度;
头文件：<winsock2.h>;
函数说明：connect()用来将参数 sockfd 的 socket 连至参数 serv_addr 指定的网络地址.参数 addrlen 为 sockaddr 的结构长度;

⑤ 说明面向连接的客户端和面向非连接的客户端在建立 Socket 时有什么区别？

对于无连接的 Socket，每个数据包可以选择不同的路径，每个数据包之间都是独立的，互相不影响，除了迷路的或者发生意外的数据包，最后都能到达,但是，到达的顺序是不确定的.

面向连接的 Socket 在正式通信之前要先确定一条路径，没有特殊情况的话，以后就固定地使用这条路径来传递数据包了。当然，路径被破坏的话，比如某个路由器断电了，那么会重新建立路径。为了保证数据包准确、顺序地到达，发送端在发送数据包以后，必须得到接收端的确认才发送下一个数据包；如果数据包发出去了，一段时间以后仍然没有得到接收端的回应，那么发送端会重新再发送一次，直到得到接收端的回应。这样一来，发送端发送的所有数据包都能到达接收端，并且是按照顺序到达的。

综上所述：
无连接 Socket 传输效率高，但是不可靠，有丢失数据包、捣乱数据的风险；
有连接 Socket 非常可靠，万无一失，但是传输效率低，耗费资源多。

⑥ 说明面向连接的客户端和面向非连接的客户端在收发数据时有什么区别，面向非链接的客户端又是如何判断数据发送结束的？

面向非连接的客户端是一起发送数据包，陆续接收数据包，并且不确定接收顺序。

面向连接的客户端是先发送一个数据包在得到接收端确认之后才会再次发送数据包。在服务器端结束传输操作后加上一返回值 0，以此来判断数据发送结束。

⑦ 比较面向连接的通信和无连接通信，它们各有什么优点和缺点？适合在哪种场合下使用？

面向连接的协议比面向无连接的协议在可靠性上有着显著的优势，但建立连接前必须等待接收方响应，传输信息过程中必须确认信息是否传到，断开连接时需要发出响应信号等，无形中加大了面向连接协议的资源开销。具体到 TCP 和 UDP 协议来说，除了源端口和目的端口，TCP 还包括序号、确认信号、数据偏移、控制标志(通常说的 URG、ACK、PSH、RST、SYN、FIN)、窗口、校验和、紧急指针、选项等信息，UDP 则只包含长度和校验和信息。UDP 数据报比 TCP 小许多，这意味着更小的负载和更有效的使用带宽。

两种协议的特点决定了它们的应用场景，在网络中，有些服务，如 HTTP、FTP 等，对数据的可靠性要求较高，在使用这些服务时，必须保证数据包能够完整无误的送达，因此这种场景下会选择面向连接的通信;而另外一些服务，如 DNS、即时聊天工具等，并不需要这么高的可靠性，高效率和实时性才是关键所在，因此会更愿意选择无连接通信。

⑧ 实验过程中使用 Socket 时是工作在阻塞方式还是非阻塞方式？通过网络探索阐述这两种操作方式的不同。

阻塞模式。

（2） 注意实验时简述设计思路。

① 首先学习变成需要的函数，了解其中参数定义
② 设计实验流程图
③ 根据流程图进行编程
④ 运行 wireshark，输入过滤条件
   1) 局域网下：ip.addr == 127.0.0.1 && udp
   2) 互联网下：ip.addr == 8.129.101.161 && udp
⑤ 编译运行服务器端代码，编译运行客户端代码
⑥ 发送数据
⑦ 抓包
⑧ 分析

（3） 引起 UDP 丢包的可能原因是什么？

① 接收端处理时间过长导致丢包。
② 发送的包巨大丢包。
③ 发送的包较大，超过接受者缓存导致丢包。
④ 发送的包频率太快。

【心得体会】

通过本次实验，我掌握了网络编程的基本方法，通过套接字达到进程间通信的目的。学习了若干函数的使用、有连接（TCP）和无连接（UDP）之间的差别和联系，以及客户端与服务器之间利用套接字建立逻辑信道的进本流程，巩固了利用 wireshark 来抓捕数据包并进行分析的技能。

在实验过程中，我通过阅读书上的例子、在网上查找资料、询问同学等方法，完成了本次编程实验，受益颇多。