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	计算机学院	班 级	软件工程	组长	梁冠轩
学号	19335118	19335258			
学生	梁冠轩	余世龙			

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）。

（2）注意实验时简述设计思路。

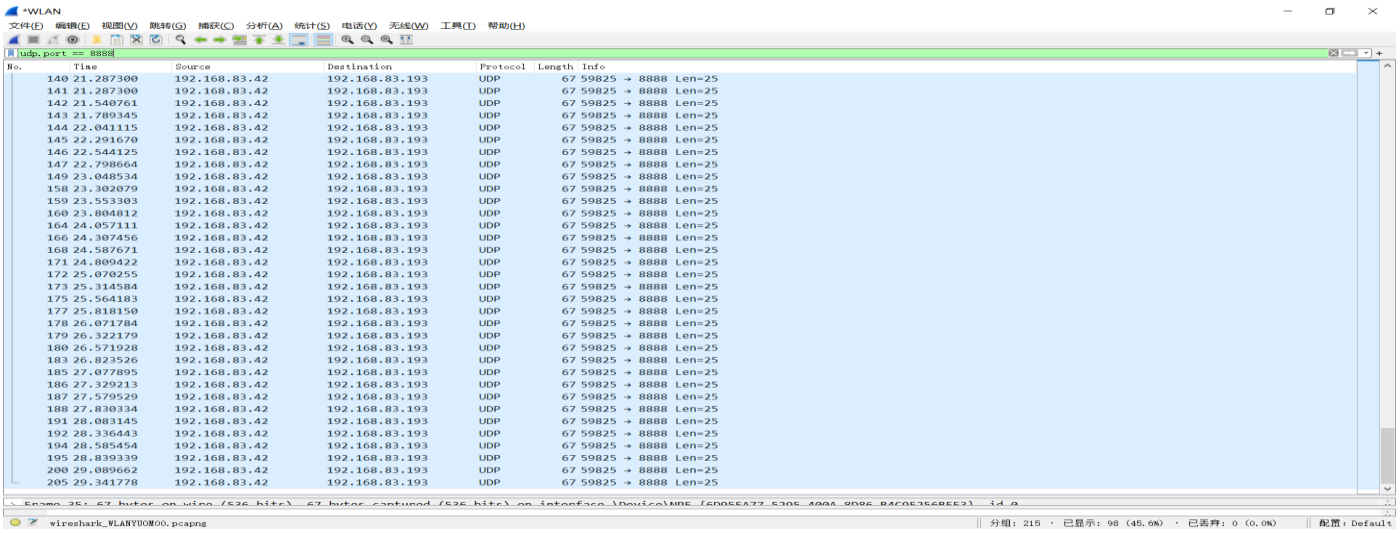
（3）引起 UDP 丢包的可能原因是什么？

本次实验完成后，请根据组员在实验中的贡献，请实事求是，自评在实验中应得的分数。（按百分制）

（1）局域网：两台电脑连上同一个 wifi 热点，热点 ip 地址为：

```
无线局域网适配器 WLAN:

   连接特定的 DNS 后缀 . . . . . . . :
   IPv6 地址 . . . . . . . . . . . . : 2408:8456:c26:43e6:4f5:7fa5:cd:aa69
   临时 IPv6 地址. . . . . . . . . . : 2408:8456:c26:43e6:a1c3:ae31:e753:70d1
   本地链接 IPv6 地址. . . . . . . . : fe80::4f5:7fa5:cd:aa69%12
   IPv4 地址 . . . . . . . . . . . . : 192.168.83.193
   子网掩码 . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : fe80::d0c0:80ff:fe45:cc5f%12
                                       192.168.83.41
```

服务器与客户端使用连上同一个 ip，客户端给服务器发送 100 个 UDP 数据包，服务器接收到了 98 个数据包，丢失个数为 2 个，丢包率为 2%。同时使用 wireshark 对数据包进行追踪分析，发现从 8888 端口接收到的 UDP 数据包也为 98 个。



已显示：98 (45.6%) 98 个数据包

```
send the 1 packets
send the 2 packets
send the 3 packets
send the 4 packets
send the 5 packets
send the 6 packets
send the 7 packets
send the 8 packets
send the 9 packets
send the 10 packets
send the 11 packets
send the 12 packets
send the 13 packets
send the 14 packets
send the 15 packets
send the 16 packets
send the 17 packets
send the 18 packets
send the 19 packets
send the 20 packets
send the 21 packets
send the 22 packets
send the 23 packets
send the 24 packets
send the 25 packets
send the 26 packets
send the 27 packets
send the 28 packets
send the 29 packets
send the 30 packets
send the 31 packets
send the 32 packets
send the 33 packets
send the 34 packets
send the 35 packets
send the 36 packets
send the 37 packets
send the 38 packets
send the 39 packets
send the 40 packets
send the 41 packets
send the 42 packets
send the 43 packets
send the 44 packets
send the 45 packets
send the 46 packets
send the 47 packets
send the 48 packets
send the 49 packets
send the 50 packets
send the 51 packets
send the 52 packets
send the 53 packets
send the 54 packets
send the 55 packets
send the 56 packets
send the 57 packets
send the 58 packets
send the 59 packets
send the 60 packets
send the 61 packets
send the 62 packets
send the 63 packets
send the 64 packets
send the 65 packets
send the 66 packets
send the 67 packets
send the 68 packets
send the 69 packets
send the 70 packets
send the 71 packets
send the 72 packets
send the 73 packets
send the 74 packets
send the 75 packets
send the 76 packets
send the 77 packets
send the 78 packets
send the 79 packets
send the 80 packets
send the 81 packets
send the 82 packets
send the 83 packets
send the 84 packets
send the 85 packets
send the 86 packets
send the 87 packets
send the 88 packets
send the 89 packets
send the 90 packets
send the 91 packets
send the 92 packets
send the 93 packets
send the 94 packets
send the 95 packets
send the 96 packets
send the 97 packets
send the 98 packets
send the 99 packets
send the 100 packets
```

```
NO75: Packets from the client.
NO76: Packets from the client.
NO77: Packets from the client.
NO78: Packets from the client.
NO79: Packets from the client.
NO80: Packets from the client.
NO81: Packets from the client.
NO82: Packets from the client.
NO83: Packets from the client.
NO84: Packets from the client.
NO85: Packets from the client.
NO86: Packets from the client.
NO87: Packets from the client.
NO88: Packets from the client.
NO89: Packets from the client.
NO90: Packets from the client.
NO91: Packets from the client.
NO92: Packets from the client.
NO93: Packets from the client.
NO94: Packets from the client.
NO95: Packets from the client.
NO96: Packets from the client.
NO97: Packets from the client.
NO98: Packets from the client.
```

互联网：两台电脑连不同的热点，但由于无法连通，无法进行实验

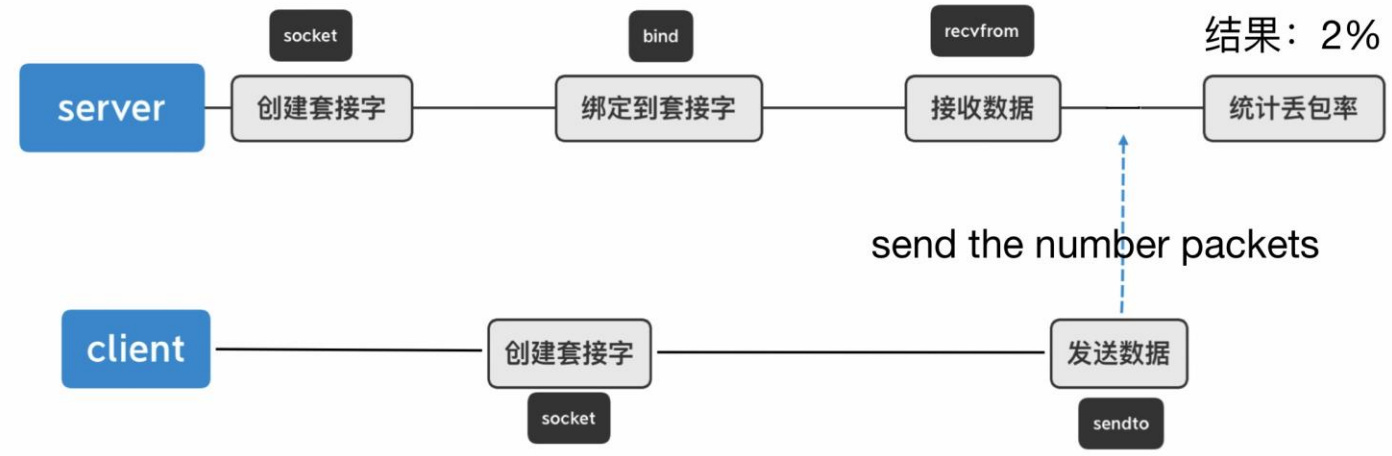
```
C:\Users\余世龙>ping 192.168.83.193

正在 Ping 192.168.83.193 具有 32 字节的数据:
请求超时。
请求超时。
请求超时。
请求超时。

192.168.83.193 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 0，丢失 = 4 (100% 丢失)，
```

实验思考：

1.

刚开始实验时，发现服务器和客户端编译时无法连接到 wsock32.lib 库，需要编译时加上-lwsock32 手动连接到库中。代码编译成功后，两台电脑在同一个热点的情况下无法连通。研究过后发现是用错了 ip 地址，用的是手机的 ip 地址，改为本机地址后，可以成功连通，客户端发送了 100 个 UDP 数据包，服务器接收到了 98 个数据包。

2.



服务器：

```
int main(){
    WSADATA wsaData;
    //将两个byte型的"2"合并成一个word，分别在高8位和低8位
    WORD sockVersion = MAKEWORD(2, 2);
    if (WSAStartup(sockVersion, &wsaData) != 0){
        return 0;
    }
    //创建套接字，指明协议族为AF_INET，socket类型为SOCK_DGRAM，type为IPPROTO_UDP
    SOCKET serSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    if (serSocket == INVALID_SOCKET){
        printf("socket error!");
        return 0;
    }
    SOCKADDR_IN serAddr;
    serAddr.sin_family = AF_INET;
    serAddr.sin_port = htons(8888);
    serAddr.sin_addr.S_un.S_addr = inet_addr("192.168.83.193");
    //将ip地址和端口号绑定到套接字
    if (bind(serSocket, (SOCKADDR *)&serAddr, sizeof(serAddr)) == SOCKET_ERROR){
        printf("bind error!");
        //关闭连接套接字
        closesocket(serSocket);
        return 0;
    }
    SOCKADDR_IN remoteAddr;
    int nAddrLen = sizeof(remoteAddr);
    int i = 1;
    while (1){
        char recData[255];
        //接受数据，指明要读取的套接口文件描述符，保存信息的缓冲区起始地址，缓冲区最大长度，标志和SOCKADDR_IN和其长度
        int ret = recvfrom(serSocket, recData, 255, 0, (SOCKADDR *)&remoteAddr, &nAddrLen);
        if (ret > 0){
            recData[ret] = 0x00;
            printf("NO%d: ", i++);
            printf(recData);
        }
    }
    //关闭连接套接字
    closesocket(serSocket);
    //终止Winsock 2 DLL的使用
    WSACleanup();
    return 0;
```

客户端：

```c
#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
int main(){
    WORD socketVersion = MAKEWORD(2, 2);
    WSADATA wsaData;
    if (WSAStartup(socketVersion, &wsaData) != 0){
        return 0;
    }
    //创建套接字，指明协议族为AF_INET，socket类型为SOCK_DGRAM，type为IPPROTO_UDP
    SOCKET cliSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    if (cliSocket == INVALID_SOCKET){
        printf("socket error!");
        return 0;
    }
    SOCKADDR_IN cliAddr;
    cliAddr.sin_family = AF_INET;
    cliAddr.sin_port = htons(8888);
    cliAddr.sin_addr.S_un.S_addr = inet_addr("192.168.83.193");
    int len = sizeof(cliAddr);
    for (int i = 1; i <= 100; i++){
        char *sendData = "Packets from the client.\n";
        printf("send the %d packets\n", i);
        //发送数据，指明套接口文件描述符，发送的数据的指针，字节长度，标志位，SOCKADDR_IN和其长度
        sendto(cliSocket, sendData, strlen(sendData), 0, (SOCKADDR *)&cliAddr, len);
        Sleep(250);
    }
    //关闭连接套接字
    closesocket(cliSocket);
    //终止Winsock 2 DLL的使用
    WSACleanup();
    return 0;
}
```

3.

socket api 开发通信过程中服务器程序需要以下函数：

```c
SOCKET socket(int af, int type, int protocol)
```

```c
int bind(SOCKET s, const struct sockaddr *name, int namelen)
```

```c
int recvfrom(SOCKET s, char *buf, int len, int flags, struct sockaddr *from, int *fromlen)
```

客户端需要以下函数：

```c
SOCKET socket(int af, int type, int protocol)
```

```c
int sendto(SOCKET s, const char *buf, int len, int flags, const struct sockaddr *to, int tolen)
```

4.

connect 函数: 指向存放服务器地址信息的结构体

```
int connect(SOCKET s, const struct sockaddr *name, int namelen)
```

bind 函数：指向特定协议的地址结构的指针

```
int bind(SOCKET s, const struct sockaddr *name, int namelen)
```

recvfrom 函数：指向装有源地址的缓冲区

```
int recvfrom(SOCKET s, char *buf, int len, int flags, struct sockaddr *from, int *fromlen)
```

sendto 函数：指向特定协议的地址结构的指针

```
int sendto(SOCKET s, const char *buf, int len, int flags, const struct sockaddr *to, int tolen)
```

5.

TCP 属于可靠的传输协议：使用 SOCK_STREAM,需要三次握手，因为传输前双方建立好了连接，相当于买卖双方建立好了交易合同，传输中一般不会出现意外，直到连接终止；

UDP 属于不可靠的传输协议：使用 SOCK_DGRAM ，不需要三次握手，UDP 的所谓连接相当于一种映射，UDP 单方面的认 为目标地址（端口）是可用的，从而进行收发数据，而实际上目标地址（端口）未必可用，所以传输数据不可靠

6.

TCP 使用流式套接字（SOCK_STREAM）：提供了一个面向连接、可靠的数据传输服务，数据无差错、无重复地发送，且按发送顺序接收。内设流量控制，避免数据流超限；数据被看作是字节流，无长度限制。

UDP 使用数据报式套接字（SOCK_DGRAM）：提供了一个无连接服务。数据包以独立包形式被发送，不提供无错保证，数据可能丢失或重复，并且接收顺序混乱。

UDP 不保证收取顺序，所以无法用最后一个包进行接收结束的判定，而是应该对每个包都进行确认，以保证数据接受完毕且验证完整性

7.

TCP 和 UDP 的优缺点：

TCP 面向连接，UDP 是无连接的，即发送数据之前不需要建立连接。TCP 提供可靠的服务，通过 TCP 连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP 尽最大努力交付，即不保证可靠交付。UDP 具有较好的实时性，工作效率比 TCP 高。TCP 对系统资源要求较多，UDP 对系统资源要求较少。TCP 连接只能是点到点的，UDP 支持一对一，一对多，多对一和多对多的交互通信。

使用场合：

TCP：当对网络通信质量有要求时，比如：整个数据要准确无误的传递给对方，这往往对于一些要求可靠的应用，比如 HTTP，HTTPS，FTP 等传输文件的协议，POP，SMTP 等邮件的传输协议。

UDP：对当前网络通讯质量要求不高的时候，要求网络通讯速度尽量的快，这时就使用 UDP，日常生活中常见使用 UDP 协议：微信语音，腾讯视频等。

8.

实验过程中使用 socket 时是工作在阻塞方式。

非阻塞模式可以理解为，执行此套接字的网络调用时，不管是否执行成功，都会立即返回。如调用 recv( )函数读取网络缓冲区中的数据时，不管是否读到数据都立即返回，而不会一直挂在此函数的调用上。而阻塞模式为只有接收到数据后才会返回，套接字默认为阻塞模式。

（2）实验设计思路：

客户端（发送方）与服务器（接收方）均创建套接字，服务器将 ip 地址和端口绑定到

套接字上，由于是 UDP，所以无需在客户端与服务器间建立连接，而是由服务器端执行 recvfrom 函数等待客户数据，而客户端用 sendto 执行发送数据（请求），接着，服务器端处理请求（显示并统计接收的数据包的数目），最后计算丢包率。

（3）UDP 丢包的可能原因：

1.接收端处理时间过长 2.发送的数据过大，无法 send 3.发送的数据超过缓冲区大小 4.发送频率过快 5.局域网的问题 6.UDP 协议根据实际情况，代码处理所接受的数据的步骤简单，而客户端发送的数据也仅为大约 24 个字节的数据，且每发送完一次都会进行一次睡眠，所以能排除接收端处理时间过长，发送的数据过大和发送频率过快等原因，而经过实验，丢包率仅为 2%，属于网络问题和 UDP 协议的问题。

【交实验报告】

上传实验报告：ftp://172.18.178.1/ 截止日期（不迟于）：1 周之内

上传包括两个文件：

（1）小组实验报告。上传文件名格式：小组号_ Ftp 协议分析实验.pdf （由组长负责上传）

例如: 文件名“10_ Ftp 协议分析实验.pdf”表示第 10 组的 Ftp 协议分析实验报告

（2）小组成员实验体会。每个同学单独交一份只填写了实验体会的实验报告。只需填写自己的学号和姓名。

文件名格式：小组号_学号_姓名_ Ftp 协议分析实验.pdf （由组员自行上传）

例如: 文件名“10_05373092_张三_ Ftp 协议分析实验.pdf”表示第 10 组的 Ftp 协议分析实验报告。

注意：不要打包上传！

学号	学生	自评分
19335118	梁冠轩	100
19335258	余世龙	100