计算机网络实验报告

警示

1. 实验心得体会如有雷同，雷同各方当次实验心得体会成绩均以 0 分计。
2. 在规定时间内未上交实验报告的，不得以其他方式补交，当次心得体会成绩按 0 分计。
3. 报告文件以 PDF 文件格式提交。

本报告主要描述学生在实验中承担的工作、遇到的困难以及解决的方法、体会与总结等。

院系	计算机学院	班 级	计科一班
学号	22336057		实验名称：实验 2：网络编程
学生	丁晓琪		

一. 本人承担的工作

- 编写 UDP 实验的 C 语言版本，并且在无线网络下实验(客户端收到来自服务器的 100 个数据包)

```c
int main(){
    char *client="172.26.105.168";
    SOCKET sockfd;
    struct sockaddr_in client_addr;
    char message[] = "Hello, UDP client!";
    int num_sent;
    WSADATA wsadata;

    //
        if (WSAStartup(MAKEWORD(2, 2), &wsadata) != 0) {
       // errexit("WSAStartup failed\n");
    }
    //创建套接字
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd == INVALID_SOCKET) {
        fprintf(stderr, "Error creating socket\n");
        WSACleanup();
        exit(1);
    }
    //客户信息
        memset(&client_addr, 0, sizeof(client_addr));
    client_addr.sin_family = AF_INET;
    client_addr.sin_port = htons(9999);
    client_addr.sin_addr.s_addr = inet_addr(client);//发送给本机就是特殊的addr

    int i=100;
    while (i>0) {
        // 更新消息内容
        i--;

        // 发送消息到客户端
        num_sent = sendto(sockfd, message, strlen(message), 0, (const struct sockaddr *)&client_addr, s
        if (num_sent < 0) {
            perror("sendto failed");
        } else {
            printf("Message sent: %s\n", message);
        }

        // 等待一段时间再发送下一条消息
        sleep(0.1);
    }

      close(sockfd);
      return 0;
}
```

```c
int main(int argc,char *argv[]){
    //char *host="127.0.0.1";
    WSADATA wsadata;
    SOCKET sockfd;
    struct sockaddr_in client;
    struct sockaddr_in server_addr;
    char buffer[4096];
    if (WSAStartup(MAKEWORD(2, 2), &wsadata) != 0) {
       // errexit("WSAStartup failed\n");
    }
    //创造套接字：UDP
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
     if (sockfd == INVALID_SOCKET) {
        fprintf(stderr, "Error creating socket\n");
        WSACleanup();
        exit(1);
    }
    //绑定端口
    memset(&client, 0, sizeof(client));
    client.sin_family=AF_INET;
    client.sin_addr.s_addr=INADDR_ANY;//本地任意地址 ??
    client.sin_port=htons(9999);
    if(bind(sockfd,(struct sockaddr *)& client, sizeof(client))<0){
        perror("bind failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }
   //设置服务器相关信息//好像不需要指定，这个调用recvform会获取的
    // memset(&server_addr, 0, sizeof(server_addr));
    // server_addr.sin_family = AF_INET;
    // server_addr.sin_port = htons(12346);  // 假设服务器在12346端口上发送数据
    // server_addr.sin_addr.s_addr = inet_addr(host);//发送给本机就是特殊的addr

    ssize_t num_bytes; //如果正常接收是非负数，否则是-1

  //发送方地址长度
  int addr_len=sizeof(server_addr);
  //循环接收来自服务器的数据
  int count=0;//计算丢包

  while(1){
    memset(buffer,0,4096);//清空缓冲区
    num_bytes=recvfrom(sockfd,buffer,4096,0, (struct sockaddr *)&server_addr, &addr_len);
    printf("the number of lost packets: %d\n",count);
    if (num_bytes < 0) {
            count++;

        }
        //打印
        buffer[num_bytes] = '\0';  // 确保字符串以null结尾（尽管UDP数据包不保证是字符串）
        printf("Data: %.*s\n", (int)num_bytes, buffer);  // 使用%.*s来限制打印的字节数

  }
      close(sockfd);
      return 0;
}
```

- 编写实验例程 3-1 TCP 实验的 C 语言版（但是有做出修改），并且在无线网络下实验（实验报告中有截图）

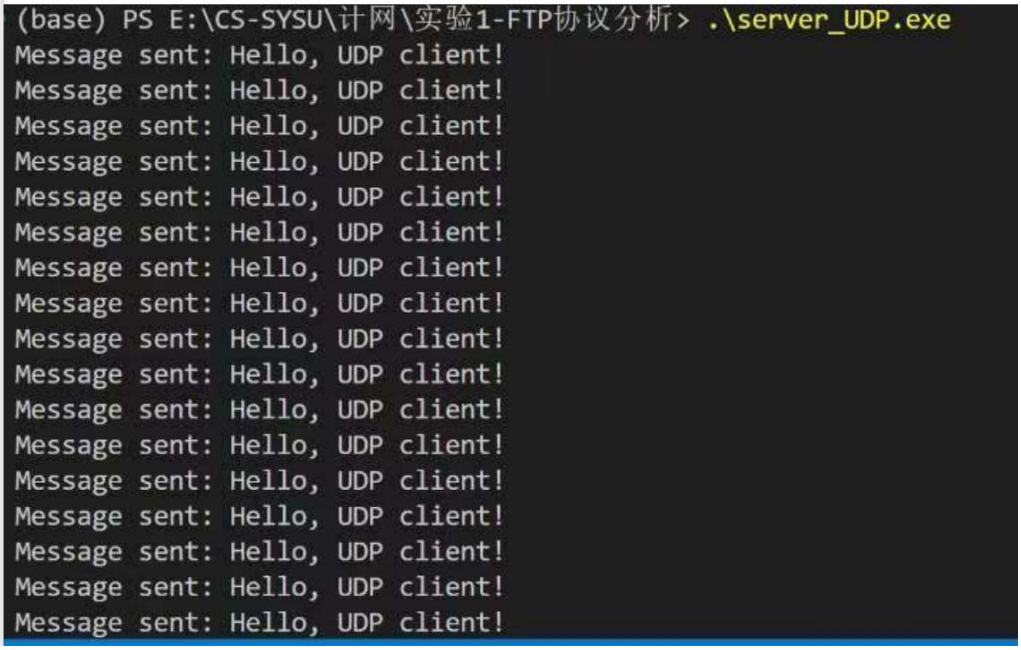
二. 遇到的困难及解决方法

- 问题：在 UDP 实验中，发现客户端可以成功发送消息，但是服务器端无法接收（但是在同一台机器的两个端口运作正常）

  解决：客户端每次发送一个数据包后有一个 0.1 秒的延迟，关闭掉服务器端的防火墙

  修复后的结果：

```
PS E:\计网\实验2-网络编程> gcc client_UDP.c -o client_UDP.exe -lws2_32
PS E:\计网\实验2-网络编程> gcc client_UDP.c -o client_UDP.exe -lws2_32
PS E:\计网\实验2-网络编程> .\client_UDP.exe
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
Data: Hello, UDP client!
the number of lost packets: 0
```



- 问题：在 TCP 实验中，发现客户端发送一个数据包时，服务器端可以接收。但是当客户端在一个 TCP 连接上连续发送多个数据包时，服务器端只能接收到第一个数据包

  解决：问题在于在一个 TCP 连接上接收数据包时只需要接收（accepct）一次客户端的连接请求(如果客户端只发送一个连接请求（connect）)。但是错误版本中，服务器端每次接收完一个数据包，调用 accepct 一次并且就用同一个 ssock（接收数据包的套接字）接收 accepct 函数产生的新的套接字。导致上一次 ssock 和客户端的连接断开，而这一次客户端没有重新发送连接请求（connect）。

  错误版本：

```
while(1){
    alen=sizeof(struct sockaddr); //accept 函数会接受这个连接，创建一个新的套接字 ssock 用于
    ssock=accept(msock,(struct sockaddr*)&fsin,&alen);
    if (ssock == INVALID_SOCKET) {
       // errexit("accept failed\n");
        continue; // 或者选择退出循环
    }
  char buffer[1024]; // 用于存储接收的数据
int bytes_received = recv(ssock, buffer, sizeof(buffer), 0); // 接收数据包

if (bytes_received == SOCKET_ERROR) {
   // errexit("recv failed\n");
    close(ssock); // 关闭套接字
}

if (bytes_received == 0) {
    // 客户端关闭了连接
    close(ssock); // 关闭套接字
}
```

修改版本：

```
while(1){ //外层循环监听链接
    alen=sizeof(struct sockaddr); //accept 函数会接受这个连接，创建一个新的套接字 ssock 用于与客户端通信，
    ssock=accept(msock,(struct sockaddr*)&fsin,&alen);
    if (ssock == INVALID_SOCKET) {
       // errexit("accept failed\n");
        continue; // 或者选择退出循环
    }

while(1){ //内层循环接连接的数据
int bytes_received = recv(ssock, buffer, sizeof(buffer), 0); // 接收数据包
if (bytes_received == SOCKET_ERROR) {
    break;
}
if (bytes_received == 0) {
    break;
}
buffer[bytes_received]='\0';
// 打印接收到的数据
printf("Received data from client: %s\n", buffer);

// 清空接收缓冲区
memset(buffer, 0, sizeof(buffer));
}
```

三. 体会与总结

- UDP 实验：客户端发送数据时不需要用 connect 函数和服务器请求连接，直接使用 sendto 函数发送。而服务器接收数据时不需要使用 accept 函数接收连接，直接使用 recvfrom 函数接收数据包。服务器端没有监听端口
- TCP 实验：客户端在使用 send 函数发送消息前，需要用 connect 函数向服务器申请连接。服务器端要先创

建一个监听端口，用 listen 函数开启监听。在使用 recv 函数接收数据前要用 accept 函数从排队的连接申请中取出一个接收，同时建立接收数据的新套接字。需要用新的套接字接收数据

- 编程记录和课堂 ppt 记录：
  - 使用的地址结构：

```
struct sockaddr_in {
    short sin_family;          // AF_INET 地址族，表示这是一个 IPv4 地址
    unsigned short sin_port; // 使用的端口，2 字节
    struct in_addr sin_addr; // IP 地址，4 字节
    char sin_zero[8];          // 预留，8 字节，通常用于对齐或填充
};

struct sockaddr {
    unsigned short sa_family; // 地址族，例如 AF_INET 表示 IPv4
    char sa_data[14];             // 地址数据，具体格式取决于地址族
};
```

TCP流程：	具体：
初始化/加载Winsock库	调用WSAStartup()
创建套接字	SOCKET socket(int domain,int type,int protocol) • domain 协议簇：PF_INET为因特网，PF_UNIX UNIX的管道功能 • type套接字类型：SOCK_STREAM 基于连接的字节流方式；SOCK_DGRAM表示无连接的数据报方式（UDP） • Protocol 协议号 UNSPEC没有特别的 • TCP(domain=PF_INET, type=SOCK_STREAM) UDP (domain=PF_INET, type=SOCK_DGRAM)
将本地IP地址和端口号绑定在所创建的套接字上	Int bind（SOCKET socket，struct sockaddr* address，int addr_len) 套接字，本地地址，地址长度。
服务器端：套接字的监听	监听端口，建立客户请求连接等待队列 Int listen(SOCKET socket,int backlog) 监听套接字，指定正在等待连接的最大队列长度
客户端：发出连接请求	（2）发出连接请求 int connect(SOCKET socket, struct sockaddr * address, int addr_len) 参数: 套接字,地址, 地址长度 返回: 0(无错), 或错误码 * 调用前, 参数 '地址' 需要给出服务器的IP地址和端口号 * 系统自动获得客户端IP地址, 并产生一个客户端当前未使用的端口号.
服务器端：套接字等待连接，接收来自客户端的请求	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 络地址(若接受一个连接请求，该地址中将包括客户的IP地 地址和端口号 址和端口号),网络 网络 地址长度 • 返回: 一个新的套接字(以下成为连接字), 或 或 INVALID_SOCKET

计算机网络实验报告

发送数据	注意：参数SOCKET为连接SOCKET不是监听那个，成功发送的字节数要看返回值
	 
	■ 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
	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
	■ 返回: 实际发送的字节数(无错时), 或SOCKET_ERROR
	■ 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。
	简例:
	#define BUFSIZE 4096
	char buf[BUFSIZE];
	int left = BUFSIZE;
	char* p = buf;
	// 给buf填充4096字节的待发数据
	// 假设s是已经连接的流式Socket
	while (left > 0)
	{
	ret = send(s, p, left, 0);
	if (ret == SOCKET_ERROR)
	{ // 错误处理
	}
	left -= ret;
	p += ret;
	}

接收数据	注意flag的含义和返回值的含义  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 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。 返回: 实际接收的字节数(无错时), 或SOCKET_ERROR
服务器端：关闭连接套接字（如果有需要转到accpect处）	int closesocket (SOCKET socket); 接收新的请求要新的连接套接字：旧的关闭
释放Winsock使用	WSAcleanup()

【交报告】

上传报告：助教

说明:上传文件名：小组号_学号_姓名_XX 实验.pdf