模拟总线型以太网数据帧发送过程

1、 目的

1）掌握 TCP、UDP 协议通信流程，以及 TCP 协议可靠性通信应用系统设计与实现原理；2）掌握 TCP、UDP 协议通信时序关系，并分析采用这种时序关系的原因；掌握利用 TCP、UDP 协议实现两台计算机之间的多媒体文件传输。

2、 要求

1）两名同学一组,分别完成发送端和接收端的程序设计。

2） 每名同学独立完成所有实验内容。

3）分析在 UDP 通信中，发送端和接收端分别在什么时刻知道通信的五元组信息:分析为什么在 UDP 通信中，第一次通信必须是客户端向服务器发送数据，而不能颠倒顺序。

4）分析在 TCP 通信中，发送端和接收端分别在什么时刻知道通信的五元组信息;分析为什么在 TCP 通信中，第一次通信必须是客户端向服务器发送连接请求，而一旦 TCP 连接建立完成后，第一次发送数据可能是从客户端到服务器，也可能从服务器到客户端;分析在 TCP 通信中，客户端和服务器分别在什么时刻知道 TCP 连接建立完成。

5）分析为什么在 TCP 通信中需要建立连接，连接含义是什么，并且通信结束后释放连接的原因。

3、 相关知识

Socket：

socket 是在应用层和传输层之间的一个抽象层，它把 TCP/IP 层复杂的操作抽象为 杂的操作抽象为几个简单的接口，供应用层调用，以实现进程在网络中通信。

UDP 的数据报发送特性：

使用 UDP 协议进行信息的传输之前不需要建议连接。换句话说就是客户端向服务器发送信息，客户端只需要给出服务器的 ip 地址和端口号，然后将信息封装到一个待发送的报文中并且发送出去。至于服务器端是否存在，或者能否收到该报文，客户端根本不用管。

TCP 通讯过程：

1、服务器初始化——LISTEN

1）调用 socket 函数创建文件描述符。

2）调用 bind 函数将当前的文件描述符和 ip/port 绑定在一起。如果这个端口已经被其他进程占用了，就会 bind 失败。

3）调用 listen 函数声明当前这个文件描述符作为一个服务器的文件描述符，为 accept 做好准

备。

4）调用 accept 函数阻塞等待客户端连接起来。

2、建立连接的过程——三次握手

第一次：调用 connect 函数发出 SYN 段向服务器发起连接请求，并阻塞等待服务器应答。

第二次：服务器收到客户端的 SYN 段后，会应答一个 SYN-ACK 段表示“同一建立连接”。

第三次：客户器端收到 SYN-ACK 后会从 connect 函数中返回，同时应答一个 ACK 段。

3、数据传输的过程

建立连接后，TCP 协议提供全双工的通信服务。所谓全双工，意思是：在同一条链路中的同一时刻，通信双方可以同时写数据。相对的概念叫做半双工，即：在同一条链路中的同一时刻，只能由一方来写数据。

1）服务器从 accept 函数返回后立刻调用 read 函数读 socket 里的数据。读 socket 就像读管道一样，如果没有数据到达就阻塞等待。

2）客户端调用 write 函数发送请求给服务器，服务器收到后就向客户端回复 ACK，并从 read 函数中返回，对客户端的请求进行处理。在此期间客户端调用 read 函数阻塞等待服务器的应答。

3）服务器调用 write 函数将处理结果发回客户端，客户端收到后就回复 ACK。服务器再次调用 read 函数阻塞等待下一条请求，。

4）客户端从 read 函数中返回，并发送下一条请求，如此循环下去。

4、断开连接的过程——四次挥手

第一次：如果客户端没有更多的请求就调用 close 函数关闭连接，客户端会向服务器端发送 FIN 端。

第二次：服务器收到 FIN 后会回应一个 ACK，同时 read 返回 0。

第三次：客户端收到 FIN 后，再返回一个 ACK 给服务器。

DNS（Domain Name System）：域名系统。
IIS （Internet Information Server）：Internet 信息服务。
FTP（File Transfer Protocol）：文件传输协议。
SMTP（Simple Mail Transfer Protocol，简单邮件传输协议）和 POP3（Post Office Protocol，邮局协议）

4、 实现原理及流程图

TCP 通信流程：

1）创建套接字，返回值 sockid 是一个整数(句柄)，即 socket 号。

2）server 端使用 bind()绑定本机地址。

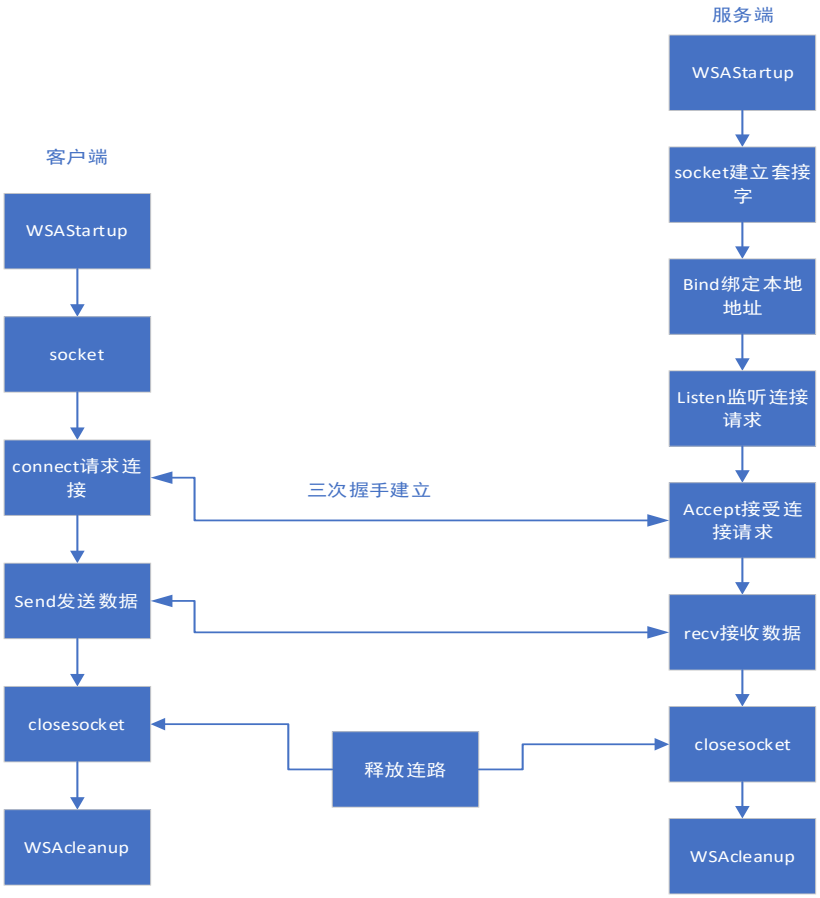
3）使用 connect 请求连接服务器，其中传入五元组参数。

4）使用 send 与 recv 来进行传输数据，此时因为已经有了通信五元组，不再需要传入五元组信息参数。

5）使用 close 释放掉连接。

6）Udp 协议原理大致与 tcp 相同，只是不需要 connect 来建立连接，而需要第一次发送数据时五元组会作为数据传输到服务端。

7）文件发送只需要将文件内容读到字符串中分组发送即可，服务器端只需要将接收到的字符串写入文件即可。



UDP 通信流程：

1）创建套接字，返回值 sockid 是一个整数(句柄)，即 socket 号。

2）server 端使用 bind()绑定本机地址。

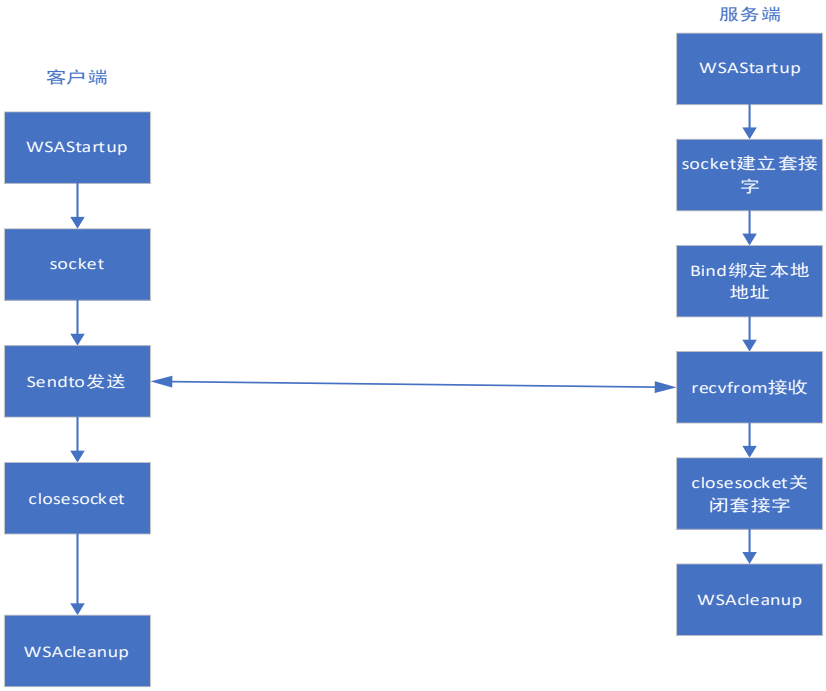
3）使用 sendto 与 recvfrom 来进行传输数据，此时向 sendto 传入通信五元组参数。

4）使用 close 释放掉连接。

5）udp 协议原理大致与 tcp 相同，只是不需要 connect 来建立连接，而需要第一次发送数据时五元组会作为数据传输到服务端。

6）文件发送只需要将文件内容读到字符串中分组发送即可，服务器端只需要将接收到的字符

串写入文件即可。



5、程序代码(以附件形式,编程环境:VC++6.0)

TCP 单向客户端

```
#include <winsock2.h>
#include <stdio.h>
#define BUFFER_SIZE 512
int main()
{
    WSADATA wsaData;
    SOCKET sClient;
    char *Serip = "192.168.9.35";
    int SeriPort = 5050;
    char send_buf[BUFFER_SIZE];
    struct sockaddr_in seraddr;
    memset(send_buf,0, BUFFER_SIZE);
    if(WSAStartup(MAKEWORD(2,2),&wsaData)!=0)
    {
        printf("failed to load winsock\n");
        return -1;
    }
    seraddr.sin_family = AF_INET;
    seraddr.sin_port = htons(SeriPort);
```

```
    seraddr.sin_addr.s_addr = inet_addr(Serip);
    sClient = socket(AF_INET,SOCK_STREAM,0);
    if(sClient == INVALID_SOCKET)
    {
        printf("creat socket failed :%d\n",WSAGetLastError());
        return -1;
    }
    if(connect(sClient,(struct sockaddr *)&seraddr,sizeof(seraddr))== INVALID_SOCKET)
    {
        printf("connect failed:%d",WSAGetLastError());
        return -1;
    }
    printf("start send data to server:\n");
    while(1)
    {
        scanf("%s",send_buf);
        int isnd = send(sClient,send_buf,sizeof(send_buf),0);
        if(isnd == 0) break;
        else if(isnd == SOCKET_ERROR)
        {
            printf("send data error:%d",WSAGetLastError());
            return -1;
        }
    }
    closesocket(sClient);
    WSACleanup();
    return 0;
}
```

TCP 单向服务器端

```
#include<winsock2.h>
#include<stdio.h>
#include<stdlib.h>
int main(){
    WSADATA wsaData;
    SOCKET oldSocket,newSocket;

    int ilen=0;
    int isend=0;
    int ircv=0;
    char recv_buffer[512];
    struct sockaddr_in seraddr,cliaddr;

    if(WSAStartup(MAKEWORD(2,2),&wsaData)!=0){
        printf("failed to load winsocket\n");
```

```
        return -1;
    }
    oldSocket=socket(AF_INET,SOCK_STREAM,0);
    if(oldSocket==INVALID_SOCKET){
        printf("create socket failed:%d",WSAGetLastError());
        return -1;
    }
    seraddr.sin_family=AF_INET;
    seraddr.sin_port=htons(5050);
    seraddr.sin_addr.s_addr=inet_addr("192.168.9.35");
    if(bind(oldSocket,(LPSOCKADDR)&seraddr,sizeof(seraddr))==SOCKET_ERROR){
        printf("bind() failed:%d\n",WSAGetLastError());
        return -1;
    }
    printf("Server start to recive data:\n");

    if(listen(oldSocket,5)==SOCKET_ERROR){
        printf("listen() failed:%d\n",WSAGetLastError());
        return -1;
    }
    ilen=sizeof(cliaddr);
    newSocket=accept(oldSocket,(struct sockaddr*)&cliaddr,&ilen);
    if(newSocket==INVALID_SOCKET){
        printf("accept() failed:%d\n",WSAGetLastError());
        return -1;
    }
    while(1){
        memset(recv_buffer,0,512);
        ircv=recv(newSocket,recv_buffer,sizeof(recv_buffer),0);
        if(ircv==SOCKET_ERROR){
            printf("rcv() failed:%d\n",WSAGetLastError());
            break;
        }
        else if(ircv==0)    break;
        else{
            printf("Server receive data from Client: %s\n",recv_buffer);
        }
    }
    closesocket(newSocket);
    closesocket(oldSocket);
    WSACleanup();

}
```

TCP 双向客户端

```
#include <winsock2.h>
#include <stdio.h>
#define BUFFER_SIZE 512

int main()
{
	WSADATA wsaData;
	SOCKET sClient;
	char *Serip = "192.168.9.35";
	int SeriPort = 5050;
	char send_buf[BUFFER_SIZE],recv_buff[BUFFER_SIZE];
	struct sockaddr_in seraddr;
	memset(send_buf,0, BUFFER_SIZE);
	if(WSAStartup(MAKEWORD(2,2),&wsaData)!=0)
	{
		printf("failed to load winsock\n");
		return -1;
	}

	seraddr.sin_family = AF_INET;
	seraddr.sin_port = htons(SeriPort);
	seraddr.sin_addr.s_addr = inet_addr(Serip);
	sClient = socket(AF_INET,SOCK_STREAM,0);

	if(sClient == INVALID_SOCKET)
	{
		printf("creat socket failed :%d\n",WSAGetLastError());
		return -1;
	}

	if(connect(sClient,(struct sockaddr *)&seraddr,sizeof(seraddr))== INVALID_SOCKET)
	{
		printf("connect failed:%d",WSAGetLastError());
		return -1;
	}
	printf("start send data to server:\n");
	int ilen = sizeof(seraddr);
	while(1)
	{
		scanf("%s",send_buf);
		int isnd = sendto(sClient,send_buf,sizeof(send_buf),0,(struct sockaddr*)&seraddr,sizeof(seraddr));
		if(isnd == 0) break;
		else if(isnd == SOCKET_ERROR)
		{
```

```
            printf("send data error:%d",WSAGetLastError());
            return -1;
        }
        int ircv= recvfrom(sClient,recv_buff,sizeof(recv_buff),0,(struct sockaddr*)&seraddr,&ilen);
        printf("client receive from server:%s\n",recv_buff);
        printf("server ip:[%s],port:[%d]\n",inet_ntoa(seraddr.sin_addr),ntohs(seraddr.sin_port));
    }
    closesocket(sClient);
    WSACleanup();
    return 0;
}
```

TCP 双向服务器端

```
#include<winsock2.h>
#include<stdio.h>
#include<stdlib.h>

int main(){
    WSADATA wsaData;
    SOCKET oldSocket,newSocket;

    int ilen=0;
    int isend=0;
    int ircv=0;
    char recv_buffer[512];
    struct sockaddr_in seraddr,cliaddr;

    if(WSAStartup(MAKEWORD(2,2),&wsaData)!=0){
        printf("failed to load winsocket\n");
        return -1;
    }
    oldSocket=socket(AF_INET,SOCK_STREAM,0);
    if(oldSocket==INVALID_SOCKET){
        printf("create socket failed:%d",WSAGetLastError());
        return -1;
    }
    seraddr.sin_family=AF_INET;
    seraddr.sin_port=htons(5050);
    seraddr.sin_addr.s_addr=inet_addr("192.168.9.35");
    if(bind(oldSocket,(LPSOCKADDR)&seraddr,sizeof(seraddr))==SOCKET_ERROR){
        printf("bind() failed:%d\n",WSAGetLastError());
        return -1;
    }
    printf("Server start to recive data:\n");
```

```
    if(listen(oldSocket,5)==SOCKET_ERROR){
        printf("listen() failed:%d\n",WSAGetLastError());
        return -1;
    }
    ilen=sizeof(cliaddr);
    newSocket=accept(oldSocket,(struct sockaddr*)&cliaddr,&ilen);
    if(newSocket==INVALID_SOCKET){
        printf("accept() failed:%d\n",WSAGetLastError());
        return -1;
    }
    while(1){
        memset(recv_buffer,0,512);
        ircv=recvfrom(newSocket,recv_buffer,sizeof(recv_buffer),0,(struct sockaddr*)&cliaddr,&ilen);
        if(ircv==SOCKET_ERROR){
            printf("rcv() failed:%d\n",WSAGetLastError());
            break;
        }
        else if(ircv==0)      break;
        else{
            printf("Server receive data from Client: %s\n",recv_buffer);
            printf("Client ip:[%s],port:[%d]\n",inet_ntoa(cliaddr.sin_addr),ntohs (cliaddr.sin_port));
        }
        int isnd=sendto(newSocket,recv_buffer,sizeof(recv_buffer),0,(struct sockaddr*)&cliaddr,sizeof(cliaddr));
    }
    closesocket(newSocket);
    closesocket(oldSocket);
    WSACleanup();
}
```

TCP 传输文件客户端

```
#include <winsock2.h>
#include <iostream>
#include <stdio.h>
#pragma comment(lib,"ws2_32.lib")
#define DATA_BUFFER 512

int main()
{
    WSADATA wsaData;
    SOCKET sClient;
    char *Serip = "192.168.9.35";
    int SeriPort = 5050;
```

```
struct sockaddr_in seraddr;
if(WSAStartup(MAKEWORD(2,2),&wsaData)!=0)
{
	printf("failed to load winsock\n");
	return -1;
}

seraddr.sin_family = AF_INET;
seraddr.sin_port = htons(SeriPort);
seraddr.sin_addr.s_addr = inet_addr(Serip);

sClient = socket(AF_INET,SOCK_STREAM,0);

if(sClient == INVALID_SOCKET)
{
	printf("creat socket failed :%d\n",WSAGetLastError());
	return -1;
}

FILE *fileptr = NULL;
fileptr = fopen("C:/Documents and Settings/Administrator/桌面/多媒体测试文件.avi","rb");
if(fileptr == NULL)
{
	printf("file can not open!\n");
	return -1;
}
if(connect(sClient,(struct sockaddr*)&seraddr,sizeof(seraddr))==INVALID_SOCKET)
{
	printf("connect() failed:%d\n",WSAGetLastError());
	return -1;
}
printf("cilent start to send file!\n");
char data_buffer[512];
memset(data_buffer,0,512);
while(1)
{

	if(!feof(fileptr))
	{
		int iread = fread(data_buffer,1,512,fileptr);
		int isend = send(sClient,data_buffer,iread,0);

	}
	else break;
```

```
    }
    printf("file send end successfully!\n");
    char str[100] = "quit";

    send(sClient,str,sizeof(str),0);

    fclose(fileptr);
    closesocket(sClient);
    WSACleanup();
    return 0;

}
```

TCP 文件服务器端

```
#include<winsock2.h>
#include<stdio.h>
#include<stdlib.h>
#pragma comment(lib,"ws2_32.lib")
int main(){
    WSADATA wsaData;
    SOCKET oldSocket,newSocket;
    struct sockaddr_in seraddr,cliaddr;

    if(WSAStartup(MAKEWORD(2,2),&wsaData)!=0){
        printf("failed to load winsocket\n");
        return -1;
    }
    oldSocket=socket(AF_INET,SOCK_STREAM,0);
    if(oldSocket==INVALID_SOCKET){
        printf("create socket failed:%d",WSAGetLastError());
        return -1;
    }

    seraddr.sin_family=AF_INET;
    seraddr.sin_port=htons(5050);
    seraddr.sin_addr.s_addr=htonl(INADDR_ANY);
    if(bind(oldSocket,(LPSOCKADDR)&seraddr,sizeof(seraddr))==SOCKET_ERROR){
        printf("bind() failed:%d\n",WSAGetLastError());
        return -1;
    }
    printf("Server start to recive data:\n");

    if(listen(oldSocket,5)==SOCKET_ERROR){
        printf("listen() failed:%d\n",WSAGetLastError());
        return -1;
```

```
    }
    int ilen=sizeof(cliaddr);
    newSocket=accept(oldSocket,(struct sockaddr*)&cliaddr,&ilen);
    if(newSocket==INVALID_SOCKET){
        printf("accept() failed:%d\n",WSAGetLastError());
        return -1;
    }
    FILE *ptrfile=fopen("c:/多媒体测试文件.avi","wb");
    if(ptrfile==NULL){
        printf("file can not open!\n");
        return -1;
    }
    char data_buffer[512];
    memset(data_buffer,0,512);

    while(1){
        int irecv=recv(newSocket,data_buffer,512,0);
        int iwrite=fwrite(data_buffer,1,irecv,ptrfile);
        if(strcmp(data_buffer,"quit")==0) break;
    }
    printf("Server recive data end successfuly\n");
    fclose(ptrfile);
    closesocket(newSocket);
    closesocket(oldSocket);
    WSACleanup();
}
```

UDP 单向通信客户端

```
#include <winsock2.h>
#include <stdio.h>
#define BUFFER_SIZE 1024
void main()
{
    char * serip = "192.168.9.35";
    int Seriport = 5050;
    char send_buf[BUFFER_SIZE];
    memset(send_buf,0,sizeof(send_buf));

    WSADATA wsadata;
    if(WSAStartup(MAKEWORD(2,2),&wsadata)!=0)
    {
        printf("failed to load winsock\n");
        return ;
    }
```

```
    struct sockaddr_in seradd;
    seradd.sin_family = AF_INET;
    seradd.sin_port = htons(Seriport);
    seradd.sin_addr.s_addr = inet_addr(serip);

    SOCKET sclient;
    sclient = socket(AF_INET,SOCK_DGRAM,0);
    if(sclient == INVALID_SOCKET)
    {
        printf("build socket failed!\n");
        return -1;
    }
    while(1)
    {
        scanf("%s",send_buf);
        sendto(sclient,send_buf,sizeof(send_buf),0,(struct sockaddr*)&seradd,sizeof(seradd));
    }
    closesocket(sclient);
    WSACleanup();
}
```

UDP 单向通信服务器端

```
#include<winsock2.h>
#include<stdio.h>
#include<stdlib.h>
#define BUFFER_SIZE 1024

int main(){
    SOCKET sSocket;
    char   recv_buf[BUFFER_SIZE];

    struct sockaddr_in seradd, cliadd;
    WSADATA wsadata;

    if (WSAStartup(MAKEWORD(2,2),&wsadata)!=0){
            printf("failed to load winsocket\n");
            return -1;
        }
    sSocket=socket(AF_INET,SOCK_DGRAM,0);
    if(sSocket==INVALID_SOCKET){
        printf("socket() failed:%d/n",WSAGetLastError());
        return -1;
    }
    seradd.sin_family=AF_INET;
    seradd.sin_port=htons(5050);
```

```
    seradd.sin_addr.s_addr= inet_addr("192.168.9.35");
    if(bind(sSocket,(LPSOCKADDR)&seradd,sizeof(seradd))==SOCKET_ERROR){
        printf("地址绑定时出错:%d\n",WSAGetLastError());
        return -1;
    }
    int ilen=sizeof(cliadd);

    memset(recv_buf,0,sizeof(recv_buf));

    while(1){
        int irecv=recvfrom(sSocket,recv_buf,sizeof(recv_buf),0,(struct sockaddr*)&cliadd,&ilen);
        if(irecv==SOCKET_ERROR){
            printf("接收时出错%d\n",WSAGetLastError());
            return-1;
        }
        else if(irecv==0)
            break;
        else{
            printf("\n%s--",recv_buf);
            printf("Server received from Client ip:[%s],port:[%d]\n",inet_ntoa(cliadd.sin_addr),ntohs (cliadd.sin_port));
        }
    }
    closesocket(sSocket);
    WSACleanup();
}
```

UDP 双向通信客户端

```
//#include <stdafx.h>
#include <winsock2.h>
#include <stdio.h>
#define BUFFER_SIZE 1024
void main()
{
    //初始化服务器 IP 地址和端口号
    char * serip = "192.168.9.35";
    int Seriport = 5050;
    //初始化发送缓存
    char send_buf[BUFFER_SIZE],recv_buf[BUFFER_SIZE];
    memset(send_buf,0,sizeof(send_buf));
    //memset(recv_buf,0,sizeof(recv_buf));
    WSADATA wsadata;
    if(WSAStartup(MAKEWORD(2,2),&wsadata)!=0)
    {
        printf("failed to load winsock\n");
        return -1;
```

```
    }
    //初始化服务器....
    struct sockaddr_in seradd;
    seradd.sin_family = AF_INET;
    seradd.sin_port = htons(Seriport);
    seradd.sin_addr.s_addr = inet_addr(serip);

    SOCKET sclient;
    sclient = socket(AF_INET,SOCK_DGRAM,0);
    if(sclient == INVALID_SOCKET)
    {
        printf("build socket failed!\n");
        return -1;
    }
    printf("Client start to send data:\n");
    int ilen = sizeof(seradd);
    while(1)
    {
        scanf("%s",send_buf);
        int isend = sendto(sclient,send_buf,sizeof(send_buf),0,(struct sockaddr*)&seradd,sizeof(seradd));
        memset(recv_buf,0,sizeof(recv_buf));
        int irecv = recvfrom(sclient,recv_buf,sizeof(recv_buf),0,(struct sockaddr*)&seradd,&ilen);
        printf("client receive data from server:%s\n",recv_buf);
        printf("server ip:[%s],port:[%d]\n",inet_ntoa(seradd.sin_addr),ntohs(seradd.sin_port));
    }
    closesocket(sclient);
    WSACleanup();
}
```

UDP 双向通信服务器端

```
//#include"stdafx.h"
#include<winsock2.h>
#include<stdio.h>
#include<stdlib.h>
#define BUFFER_SIZE 1024
int main(){
    SOCKET sSocket;
    char    recv_buf[BUFFER_SIZE],send_buf[BUFFER_SIZE];

    struct sockaddr_in seradd, cliadd;
    WSADATA wsadata;

    if (WSAStartup(MAKEWORD(2,2),&wsadata)!=0){
            printf("failed to load winsocket\n");
            return -1;
```

```
        }
    sSocket=socket(AF_INET,SOCK_DGRAM,0);
    if(sSocket==INVALID_SOCKET){
        printf("socket() failed:%d/n",WSAGetLastError());
        return -1;
    }
    seradd.sin_family=AF_INET;
    seradd.sin_port=htons(5050);
    seradd.sin_addr.s_addr= inet_addr("192.168.9.35");
    if(bind(sSocket,(LPSOCKADDR)&seradd,sizeof(seradd))==SOCKET_ERROR){
        printf("地址绑定时出错:%d\n",WSAGetLastError());
        return -1;
    }
    int ilen=sizeof(cliadd);

    memset(recv_buf,0,sizeof(recv_buf));
    printf("Server start to recive data:\n");
    while(1){
        int irecv=recvfrom(sSocket,recv_buf,sizeof(recv_buf),0,(struct sockaddr*)&cliadd,&ilen);
        if(irecv==SOCKET_ERROR){
            printf("接收时出错%d\n",WSAGetLastError());
            return-1;
        }
        else if(irecv==0)
            break;
        else{
            printf("Server received from Client %s\n",recv_buf);
            printf("Client ip:[%s],port:[%d]\n",inet_ntoa(cliadd.sin_addr),ntohs (cliadd.sin_port));
        }
        int isend = sendto(sSocket, recv_buf,sizeof(recv_buf),0,(struct sockaddr*)&cliadd,sizeof(cliadd));
    }
    closesocket(sSocket);
    WSACleanup();
}
```

UDP 多媒体文件传输客户端

```
//#include <stdafx.h>
#include <winsock2.h>
#include <stdio.h>
#define BUFFER_SIZE 1024
void main()
{
    //初始化服务器 IP 地址和端口号
    char * serip = "192.168.9.35";
```

```
int Seriport = 5050;
WSADATA wsadata;
if(WSAStartup(MAKEWORD(2,2),&wsadata)!=0)
{
	printf("failed to load winsock\n");
	return -1;
}
//初始化服务器....
struct sockaddr_in seradd;
seradd.sin_family = AF_INET;
seradd.sin_port = htons(Seriport);
seradd.sin_addr.s_addr = inet_addr(serip);

SOCKET sclient;
sclient = socket(AF_INET,SOCK_DGRAM,0);
if(sclient == INVALID_SOCKET)
{
	printf("build socket failed!\n");
	return -1;
}
FILE *fileptr = NULL;
fileptr = fopen("C:/Documents and Settings/Administrator/桌面/多媒体测试文件.avi","rb");
if(fileptr == NULL)
{
	printf("file can not open!\n");
	return -1;
}
printf("cilent start to send file!\n");
char data_buffer[512];
memset(data_buffer,0,512);
while(1)
{

	if(!feof(fileptr))
	{
		int iread = fread(data_buffer,1,512,fileptr);
		int isend = sendto(sclient,data_buffer,iread,0,(struct sockaddr*)&seradd,sizeof(seradd));

	}
	else break;
}
printf("file send end successfully!\n");
char str[100] = "quit";
sendto(sclient,str,sizeof(str),0,(struct sockaddr*)&seradd,sizeof(seradd));
```

```
    fclose(fileptr);
    closesocket(sclient);
    WSACleanup();
}
```

UDP 多媒体文件传输服务器端

```
//#include"stdafx.h"
#include<winsock2.h>
#include<stdio.h>

int main(){
    SOCKET sSocket;

    struct sockaddr_in seradd, cliadd;
    WSADATA wsadata;

    if (WSAStartup(MAKEWORD(2,2),&wsadata)!=0){
            printf("failed to load winsocket\n");
            return -1;
        }
    sSocket=socket(AF_INET,SOCK_DGRAM,0);
    if(sSocket==INVALID_SOCKET){
        printf("socket() failed:%d/n",WSAGetLastError());
        return -1;
    }
    seradd.sin_family=AF_INET;
    seradd.sin_port=htons(5050);
    seradd.sin_addr.s_addr= inet_addr("192.168.9.35");
    if(bind(sSocket,(LPSOCKADDR)&seradd,sizeof(seradd))==SOCKET_ERROR){
        printf("地址绑定时出错:%d\n",WSAGetLastError());
        return -1;
    }
    int ilen=sizeof(cliadd);
    FILE *ptrfile=fopen("c:/多媒体测试文件.avi","wb");
    if(ptrfile==NULL){
        printf("file can not open!\n");
        return -1;
    }
    char data_buffer[512];
    memset(data_buffer,0,512);
    printf("Server start to recive file:\n");
    while(1){
        int irecv=recvfrom(sSocket,data_buffer,512,0,(struct sockaddr*)&cliadd,&ilen);
        int iwrite=fwrite(data_buffer,1,irecv,ptrfile);
```

```
        if(strcmp(data_buffer,"quit")==0) break;
    }
    printf("Server recive data end successfuly\n");
    fclose(ptrfile);
    closesocket(sSocket);
    WSACleanup();
}
```

6、运行结果与分析

基于 UDP 的单向通信：

客户端向服务器端发送信息，服务器端输出信息，并输出客户端的 ip 地址和端口

基于 UDP 的双向通信：

客户端向服务器端发送信息，服务器端输出信息，并输出客户端的 ip 地址和端口，再把收到的信息发送给客户端，客户端输出打印的信息，并输出服务器端的 ip 地址和端口。

基于 UDP 的文件传输：

基于 TCP 的单向通信：

客户端向服务器端发送信息，服务器端输出信息。

基于 TCP 双向通信：

客户端向服务器端发送信息，服务器端输出信息，并把收到的消息发给客户端，客户端再把收到的信息输出

基于 TCP 的文件传输：

7、参考文献

《计算机网络原理实验分析与实践》 姚烨、朱怡安

8、附加题目

UDP 相关

1）UDP 通信为什么必须是客户端首先向服务器发送数据？

因为服务端绑定了自己的 ip 与 port，并不知道客户端 ip 与 port，而客户端知道自己的 ip 与 port 与 服务器的 ip 与 port，所以知道了通信五元组，而服务端只有在接收到数据后才知道通信五元组。

2）客户端何时知道通信五元组？

客户端在调用 sendto 时知道通信五元组。

3）服务器何时知道通信五元组？

服务器端在接受到数据时知道通信五元组。

TCP 相关

1）TCP 建立连接的相关函数有那些？

connect，listen，accept。

2）客户端何时知道通信五元组

客户端在 connect 时知道通信五元组。

3）服务端何时知道通信五元组

服务器端在 listen 后知道通信五元组。

4）TCP 连接一旦建立，第一次通信哪一方发送数据为什么没有限制

因为此时五元组双方都已经知道，所以第一次通信无论哪一方发数据都是没有限制的。

5）TCP 建立连接时，为什么是客户端必须向服务器发送连接请求，而不是相反方向。

因为只有客户端知道五元组，所以只能单方向由客户端向服务器端发送请求。

6）TCP 服务端 close（）函数需要调用几次，前后两次有什么区别？

需要调用两次，首先第一次关闭侦听套接字，第二次时关闭传输数据套接字。

127.0.0.1、localhost、本机 IP 的区别

127.0.0.1 是一个私有 IP，代表的是本机环回地址，其本质上是绑定在虚拟网卡（loopback）上的 IP，经过网卡传输，依赖网卡协议，并受到网卡相关协议的限制。使用 IP 访问的时候，等于本机通过网络再去访问本机，会涉及到网络用户的权限；而 localhost 是域名，默认是指

向 127.0.0.1 的。不经网卡传输，不会受到网卡协议的限制。设置程序时本地服务用 localhost，不会解析成 IP，也不会占用网卡、网络资源。localhost 和 127.0.0.1 并不需要联网访问，即使在无网络环境下访问这两者都能找到本机。

本机 IP 中的有线网 IP 和无线网 IP 都是需要联网后才能正常分配和访问的，它们是本机对外开放的 IP 地址。

作业 1：

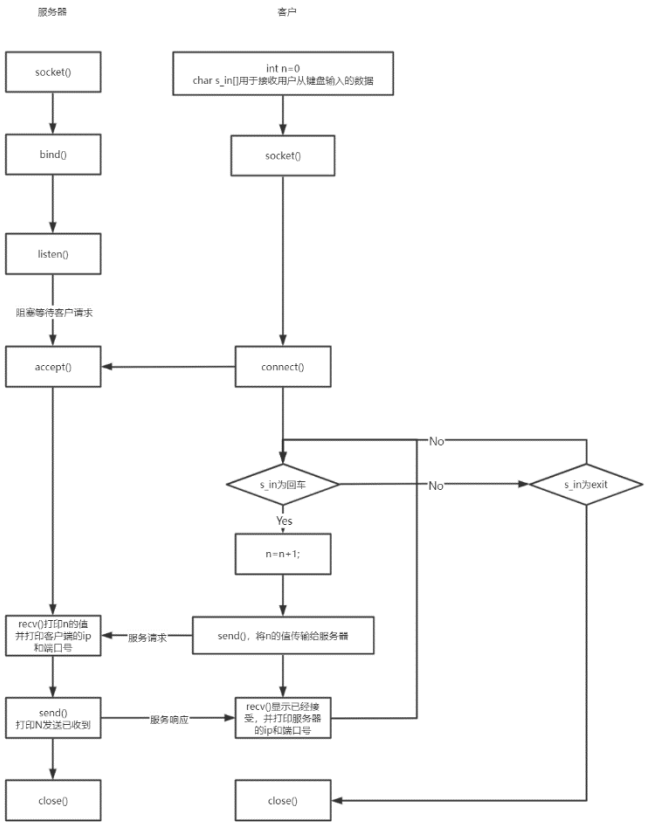
在window/linux下，使用socket编程接口实现client/server之间通信，分别采用TCP与UDP协议实现.

实验结果要求如下：

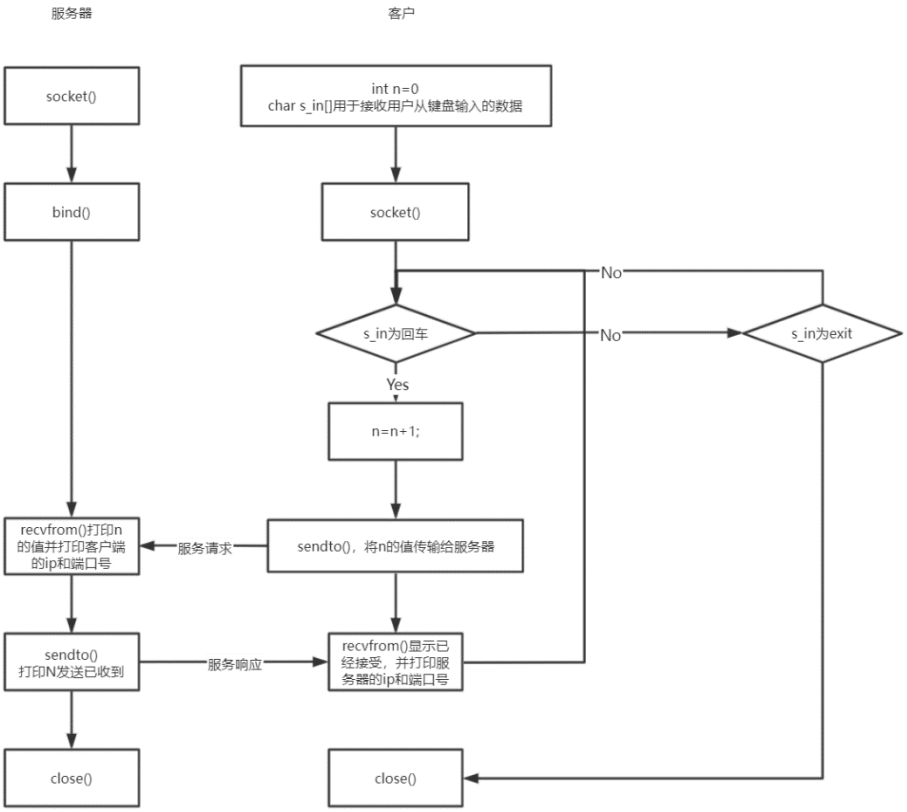
- Int N = 0;
- 1)CLIENT每次按一下“RETURN”键，向SERVER发送N;
- 2)SERVER界面显示:收到IP地址为**,端口号为**发送来的内容:N;
- 3)SERVER向CLIENT发送应答:N已经收到;
- 4)CLIENT界面显示:收到IP地址为**,端口号为**发送来的应答:N已经收到;
- N++;
- GOTO 1)，直到用输入exit(or EXIT)，程序结束。

流程图：

1）TCP 实现：



2）UDP 实现：



作业 2：

在window/linux下，使用socket编程接口实现
client/server 之间通信，分别采用TCP与UDP协议实现.
实验要求如下：

- 将一个10M大小文件通过客户端发送给服务器；
- 每次发送，传输层“数据”字段大小为512字节.

解答：思路和多媒体文件传输的方式类似，其中客户端需要向服务器传送 10M 大小的文件，而每次传输的数据大小为 512 字节，此时只需要在客户端以 512 字节为单位，将文件通过 fread 函数每次读取 512 字节，放入 send 或者 sendto 函数中，在服务器端使用 recv 或者 recvfrom 函数读取接收到的数据，并通过 fwrite 写入即可实现。