# 哈尔滨工业大学

# <<计算机网络>>

# 实验报告

**(2018 年度春季学期)**

| | |
|---|---|
| 姓名： | |
| 学号： | |
| 学院： | 计算机学院 |
| 教师： | |

# 一、实验目的

理解滑动窗口协议的基本原理；<br>
掌握 GBN 的工作原理；<br>
掌握基于 UDP 设计并实现一个 GBN 协议的过程与技术。

# 二、实验内容

1) 基于 UDP 设计一个简单的 GBN 协议，实现单向可靠数据传输（服务器到客户的数据传输）；

2) 模拟引入数据包的丢失，验证所设计协议的有效性；

3) 改进所设计的 GBN 协议，支持双向数据传输；

4）将所设计的 GBN 协议改进为 SR 协议。

# 三、实验过程及结果

## 1.实验要点

1) 基于 UDP 实现的 GBN 协议，可以不进行差错检测，可以利用 UDP 协议差错检测；
2) 自行设计数据帧的格式，应至少包含序列号 Seq 和数据两部分；
3) 自行定义发送端序列号 Seq 比特数 L 以及发送窗口大小 W，应满 足条件 W+1<=2L。
4) 一种简单的服务器端计时器的实现办法：设置套接字为非阻塞方 式，则服务器端在 recvfrom 方法上不会阻塞，若正确接收到 ACK 消息， 则计时器清零，若从客户端接收数据长度为-1（表示没有接收到任何数 据），则计时器+1，对计时器进行判断，若其超过阈值，则判断为超时， 进行超时重传。（当然，如果服务器选择阻塞模式，可以用到 select 或 epoll 的阻塞选择函数，详情见 MSDN）
5) 为了模拟 ACK 丢失，一种简单的实现办法：客户端对接收的数 据帧进行计数，然后对总数进行模 N 运算，若规定求模运算结果为零则 返回 ACK，则每接收 N 个数据帧才返回 1 个 ACK。当 N 取值大于服务 器端的超时阀值时，则会出现服务器端超时现象。
6) 当设置服务器端发送窗口的大小为 1 时，GBN 协议就是停-等协议。

## 2.数据分组格式

|  |  |  |
|---|---|---|
| Seq | Data | end |

各个域作用

Seq 为 1 个字节，取值为 0~255，（故序列号最多为 256 个）；

Data≤1024 个字节，为传输的数据；

最后一个字节放入 EOF0，表示结尾

## 3.确认分组格式

|  |  |
|---|---|
| Ack | 0 |

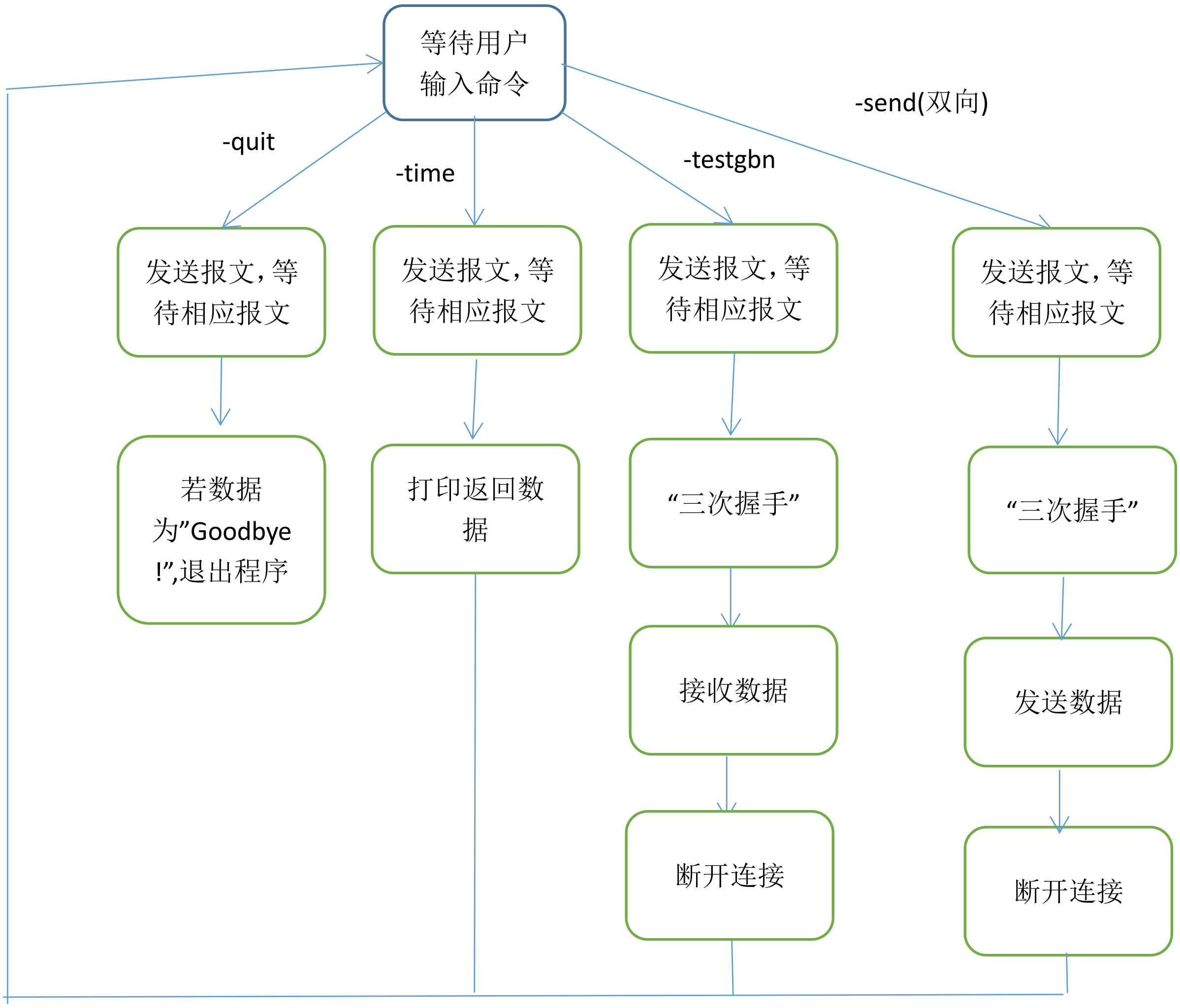
各个域作用:

ACK 字段为一个字节，表示序列号数值；

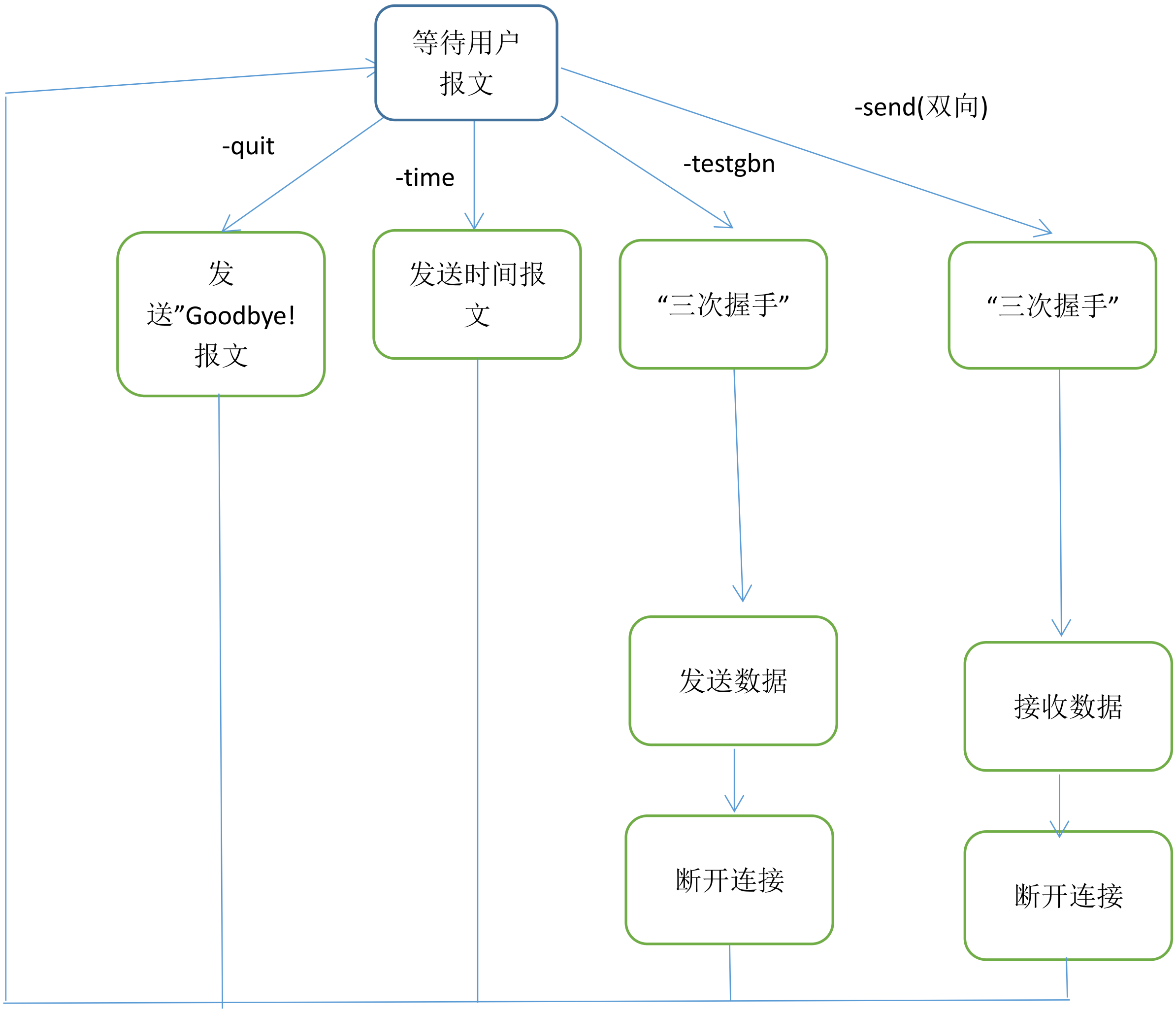
末尾放入 0，表示数据结束。

## 4.协议两端程序流程图

客户端流程图：

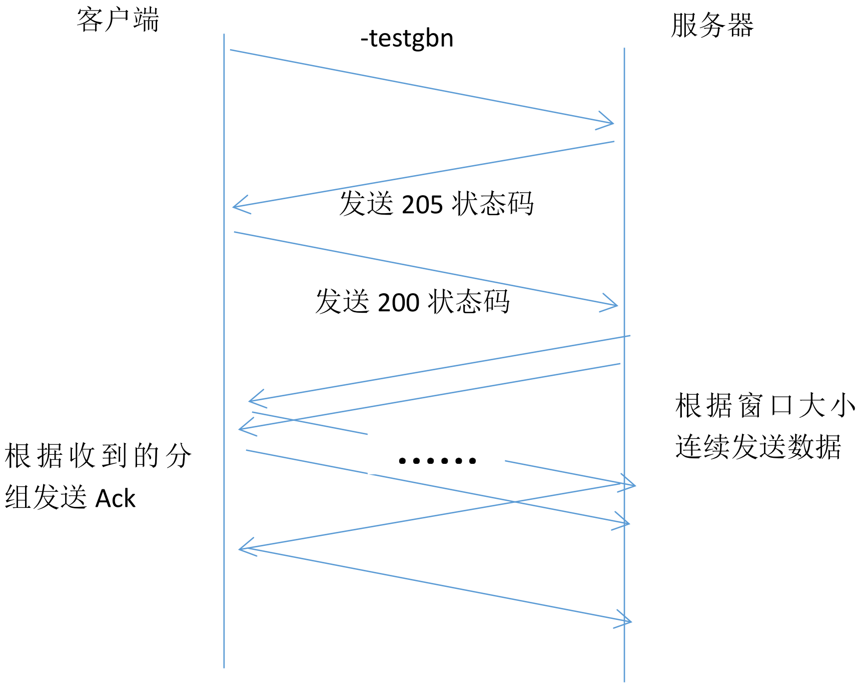


服务器端流程图：

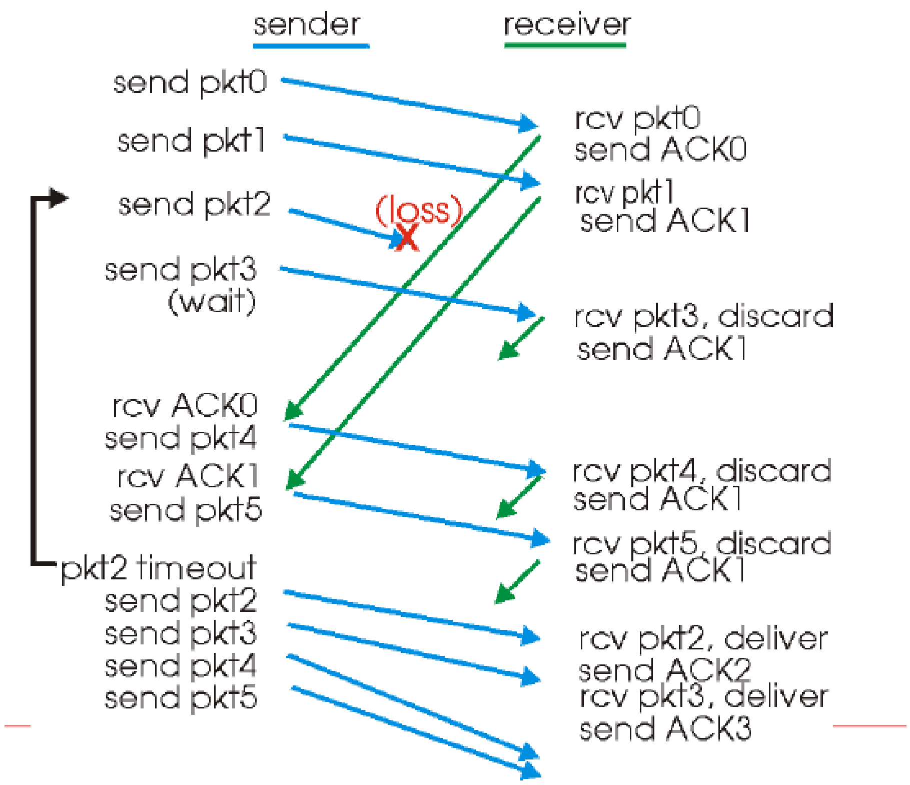


等待用户报文
-quit
-time
-testgbn
-send(双向)
发送"Goodbye!报文
发送时间报文
“三次握手”
“三次握手”
发送数据
接收数据
断开连接
断开连接

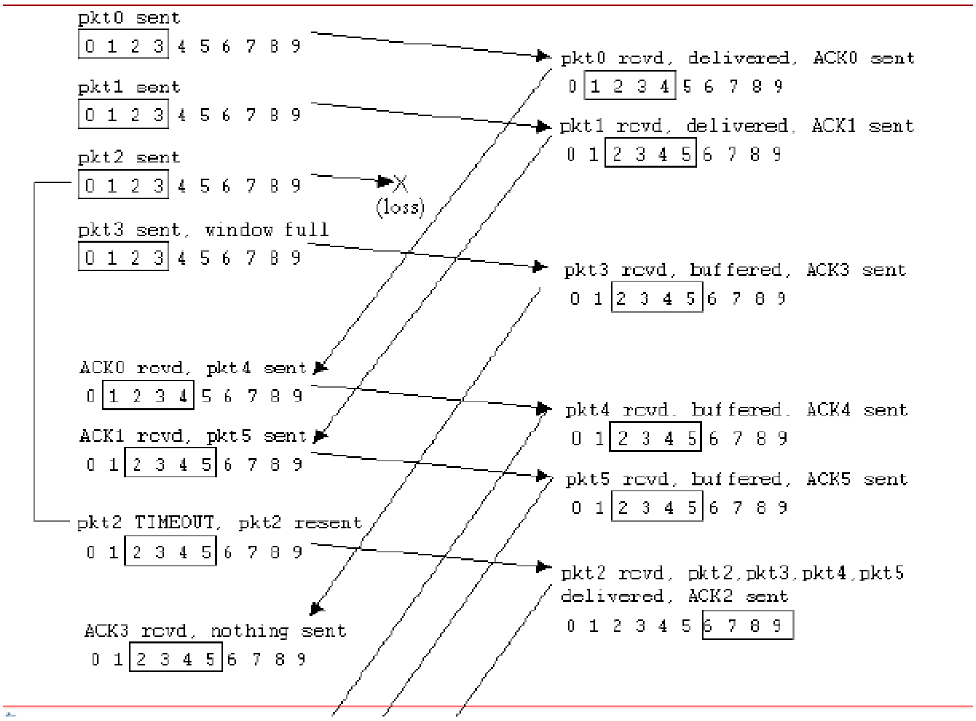
## 5.协议典型交互过程



### 1. GBN 示例


sender
receiver
send pkt0
rcv pkt0
send ACK0
send pkt1
rcv pkt1
send ACK1
send pkt2
(loss)
send pkt3
(wait)
rcv pkt3, discard
send ACK1
rcv ACK0
send pkt4
rcv ACK1
send pkt5
rcv pkt4, discard
send ACK1
rcv pkt5, discard
send ACK1
pkt2 timeout
send pkt2
send pkt3
send pkt4
send pkt5
rcv pkt2, deliver
send ACK2
rcv pkt3, deliver
send ACK3

## 2. SR 示例


pkt0 sent
0 1 2 3 4 5 6 7 8 9
pkt0 rcvd, delivered, ACK0 sent
0 1 2 3 4 5 6 7 8 9
pkt1 sent
0 1 2 3 4 5 6 7 8 9
pkt1 rcvd, delivered, ACK1 sent
0 1 2 3 4 5 6 7 8 9
pkt2 sent
0 1 2 3 4 5 6 7 8 9
(loss)
pkt3 sent, window full
0 1 2 3 4 5 6 7 8 9
pkt3 rcvd, buffered, ACK3 sent
0 1 2 3 4 5 6 7 8 9
ACK0 rcvd, pkt4 sent
0 1 2 3 4 5 6 7 8 9
pkt4 rcvd, buffered, ACK4 sent
0 1 2 3 4 5 6 7 8 9
ACK1 rcvd, pkt5 sent
0 1 2 3 4 5 6 7 8 9
pkt5 rcvd, buffered, ACK5 sent
0 1 2 3 4 5 6 7 8 9
pkt2 TIMEOUT, pkt2 resent
0 1 2 3 4 5 6 7 8 9
pkt2 rcvd, pkt2,pkt3,pkt4,pkt5
delivered, ACK2 sent
0 1 2 3 4 5 6 7 8 9
ACK3 rcvd, nothing sent
0 1 2 3 4 5 6 7 8 9

## 6.数据分组丢失验证模拟方法

客户端：收到数据后，以一定概率丢失该数据，不向服务器发送ACK，模拟数据丢失，打印该数据的序列号;

服务器：收到Ack后，服务器判断是否需要重传；

验证方式：在服务器端将test.txt文件作为要传输的数据，如果数据传输完成后在客户端收到的数据recv.txt是准确无误的，则认为是GBN/SR是可靠的。

## 7.程序实现的主要类（或函数）及其主要作用

服务器接收数据函数
void recv(SOCKET &, char [], SOCKADDR_IN &, float &, float &)

服务器发送数据函数，基于GBN协议
void testGBN(SOCKET &, char [], SOCKADDR_IN &)

服务器发送数据函数，基于SR协议
void testSR(SOCKET &, char [], SOCKADDR_IN &)

客户端发送数据函数
void sendData(SOCKET &, char [], SOCKADDR_IN &);
客户端接收数据函数
void recv(SOCKET &, char [], SOCKADDR_IN &, float &, float &)

模拟丢包函数
BOOL lossInLossRatio(float lossRatio)

累积确认Ack
void ackHandler(char c)

超时重传函数
void timeoutHandler()

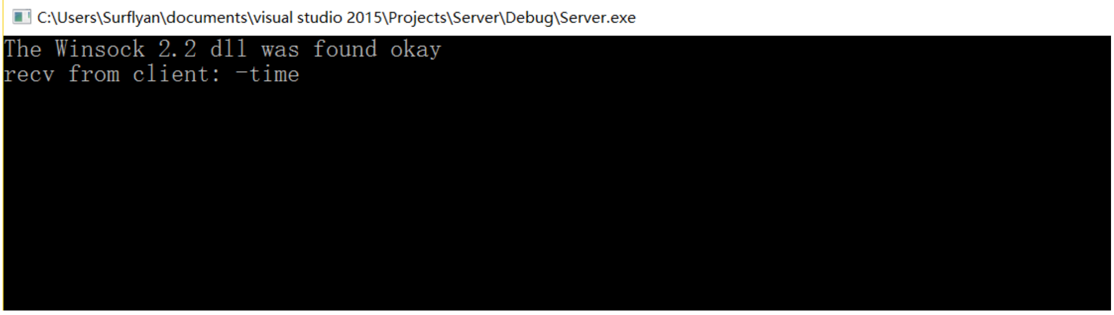
判断当前序列号 curSeq 是否可用
bool seqIsAvailable()

获取当前系统时间，结果存入 ptime 中
void getCurTime(char *ptime)

## 8.实验验证结果

### -time

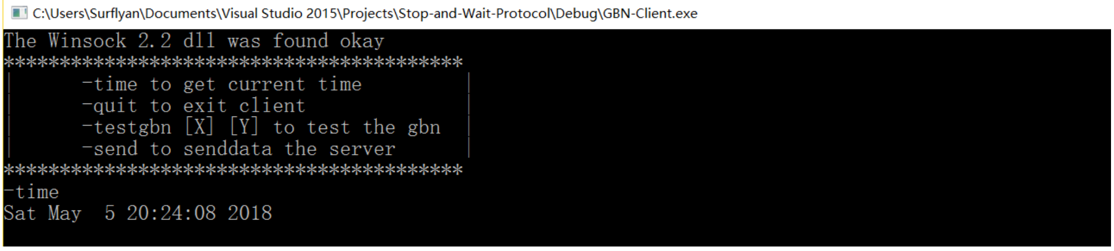
server：



C:\Users\Surflyan\documents\visual studio 2015\Projects\Server\Debug\Server.exe

```
The Winsock 2.2 dll was found okay
recv from client: -time
```

client:

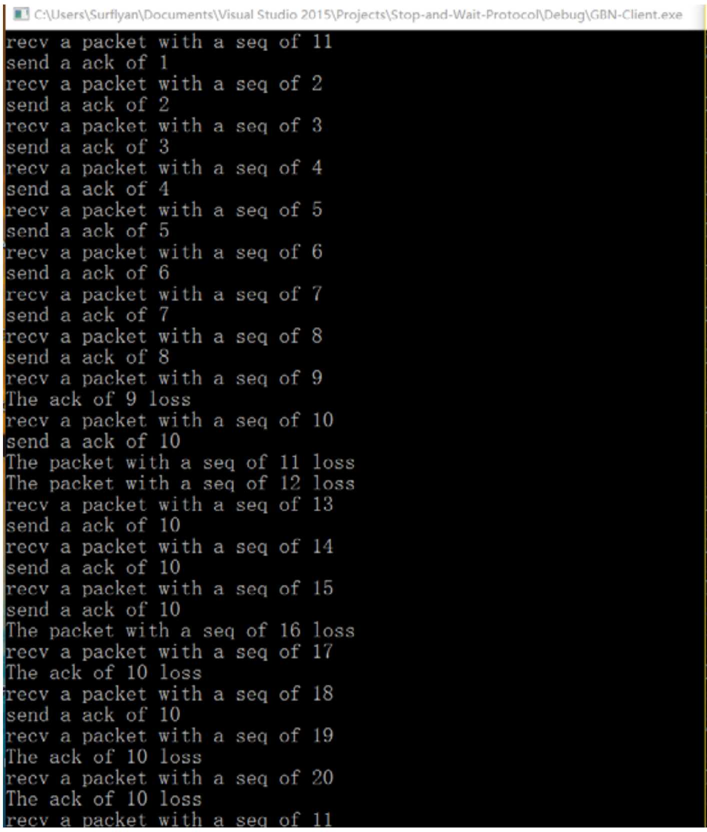


C:\Users\Surflyan\Documents\Visual Studio 2015\Projects\Stop-and-Wait-Protocol\Debug\GBN-Client.exe

```
The Winsock 2.2 dll was found okay
*****************************************
|       -time to get current time       |
|       -quit to exit client            |
|       -testgbn [X] [Y] to test the gbn|
|       -send to senddata the server    |
*****************************************
-time
Sat May  5 20:24:08 2018
```

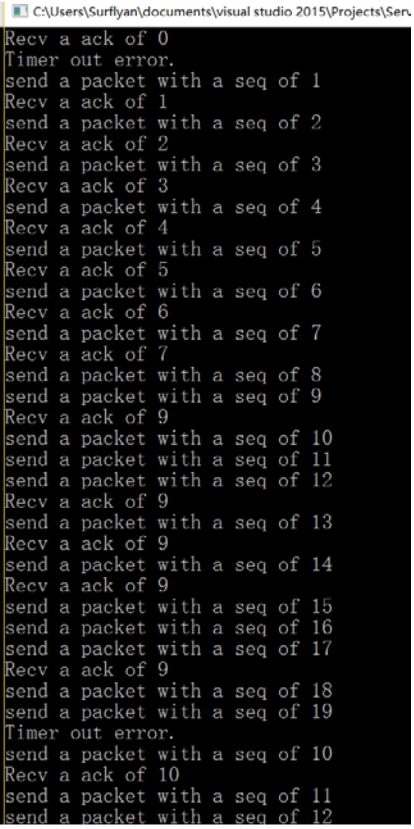
## -testgbn

Client:


C:\Users\Surflyan\Documents\Visual Studio 2015\Projects\Stop-and-Wait-Protocol\Debug\GBN-Client.exe

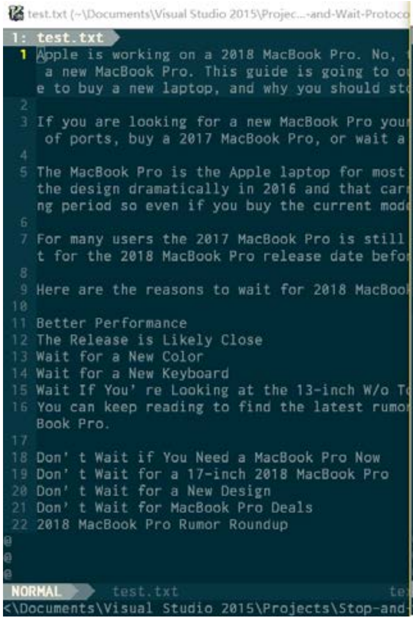
```
recv a packet with a seq of 11
send a ack of 1
recv a packet with a seq of 2
send a ack of 2
recv a packet with a seq of 3
send a ack of 3
recv a packet with a seq of 4
send a ack of 4
recv a packet with a seq of 5
send a ack of 5
recv a packet with a seq of 6
send a ack of 6
recv a packet with a seq of 7
send a ack of 7
recv a packet with a seq of 8
send a ack of 8
recv a packet with a seq of 9
The ack of 9 loss
recv a packet with a seq of 10
send a ack of 10
The packet with a seq of 11 loss
The packet with a seq of 12 loss
recv a packet with a seq of 13
send a ack of 10
recv a packet with a seq of 14
send a ack of 10
recv a packet with a seq of 15
send a ack of 10
The packet with a seq of 16 loss
recv a packet with a seq of 17
The ack of 10 loss
recv a packet with a seq of 18
send a ack of 10
recv a packet with a seq of 19
The ack of 10 loss
recv a packet with a seq of 20
The ack of 10 loss
recv a packet with a seq of 11
```

Server:

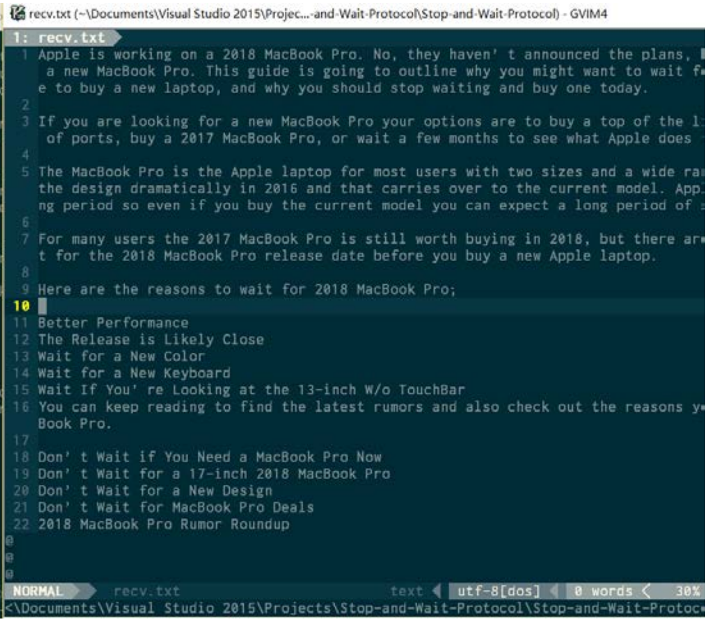

C:\Users\Surflyan\documents\visual studio 2015\Projects\Serv

```
Recv a ack of 0
Timer out error.
send a packet with a seq of 1
Recv a ack of 1
send a packet with a seq of 2
Recv a ack of 2
send a packet with a seq of 3
Recv a ack of 3
send a packet with a seq of 4
Recv a ack of 4
send a packet with a seq of 5
Recv a ack of 5
send a packet with a seq of 6
Recv a ack of 6
send a packet with a seq of 7
Recv a ack of 7
send a packet with a seq of 8
send a packet with a seq of 9
Recv a ack of 9
send a packet with a seq of 10
send a packet with a seq of 11
send a packet with a seq of 12
Recv a ack of 9
send a packet with a seq of 13
Recv a ack of 9
send a packet with a seq of 14
Recv a ack of 9
send a packet with a seq of 15
send a packet with a seq of 16
send a packet with a seq of 17
Recv a ack of 9
send a packet with a seq of 18
send a packet with a seq of 19
Timer out error.
send a packet with a seq of 10
Recv a ack of 10
send a packet with a seq of 11
send a packet with a seq of 12
```

发送的文件与收到的文件比较


test.txt (~\Documents\Visual Studio 2015\Projec...-and-Wait-Protoco
1: test.txt
NORMAL
test.txt
te

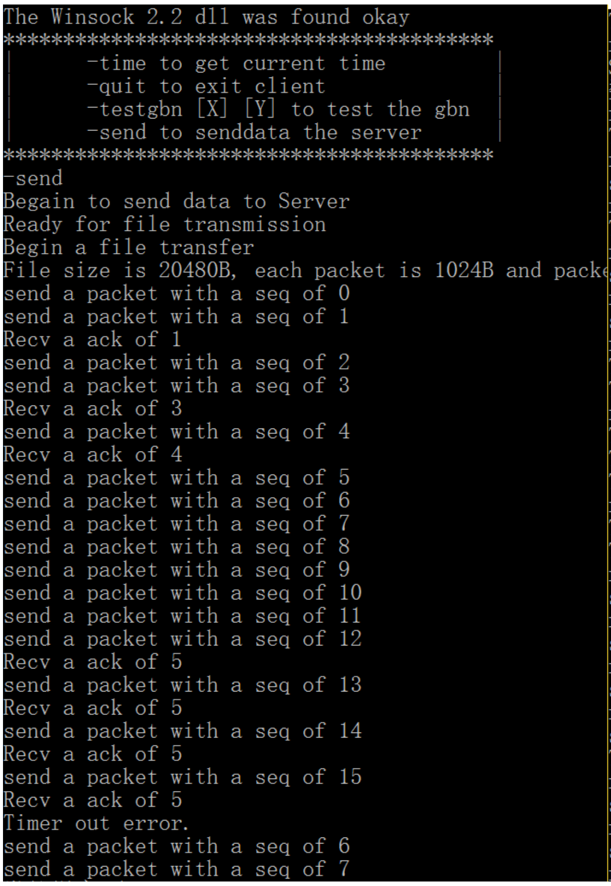
```
Apple is working on a 2018 MacBook Pro. No,
 a new MacBook Pro. This guide is going to o
e to buy a new laptop, and why you should st

If you are looking for a new MacBook Pro your
 of ports, buy a 2017 MacBook Pro, or wait a

The MacBook Pro is the Apple laptop for most
the design dramatically in 2016 and that car
ng period so even if you buy the current mod

For many users the 2017 MacBook Pro is still
t for the 2018 MacBook Pro release date befo

Here are the reasons to wait for 2018 MacBoo

Better Performance
The Release is Likely Close
Wait for a New Color
Wait for a New Keyboard
Wait If You' re Looking at the 13-inch W/o T
You can keep reading to find the latest rumo
Book Pro.

Don' t Wait if You Need a MacBook Pro Now
Don' t Wait for a 17-inch 2018 MacBook Pro
Don' t Wait for a New Design
Don' t Wait for MacBook Pro Deals
2018 MacBook Pro Rumor Roundup
@
@
@
<\Documents\Visual Studio 2015\Projects\Stop-and
```


recv.txt (~\Documents\Visual Studio 2015\Projec...-and-Wait-Protocol\Stop-and-Wait-Protocol) - GVIM4
1: recv.txt
NORMAL
recv.txt
text
utf-8[dos]
0 words
30%

```
Apple is working on a 2018 MacBook Pro. No, they haven' t announced the plans,
 a new MacBook Pro. This guide is going to outline why you might want to wait f
e to buy a new laptop, and why you should stop waiting and buy one today.

If you are looking for a new MacBook Pro your options are to buy a top of the l
 of ports, buy a 2017 MacBook Pro, or wait a few months to see what Apple does

The MacBook Pro is the Apple laptop for most users with two sizes and a wide ra
the design dramatically in 2016 and that carries over to the current model. App
ng period so even if you buy the current model you can expect a long period of

For many users the 2017 MacBook Pro is still worth buying in 2018, but there ar
t for the 2018 MacBook Pro release date before you buy a new Apple laptop.

Here are the reasons to wait for 2018 MacBook Pro;


Better Performance
The Release is Likely Close
Wait for a New Color
Wait for a New Keyboard
Wait If You' re Looking at the 13-inch W/o TouchBar
You can keep reading to find the latest rumors and also check out the reasons y
Book Pro.

Don' t Wait if You Need a MacBook Pro Now
Don' t Wait for a 17-inch 2018 MacBook Pro
Don' t Wait for a New Design
Don' t Wait for MacBook Pro Deals
2018 MacBook Pro Rumor Roundup
@
@
@
<\Documents\Visual Studio 2015\Projects\Stop-and-Wait-Protocol\Stop-and-Wait-Protoc
```

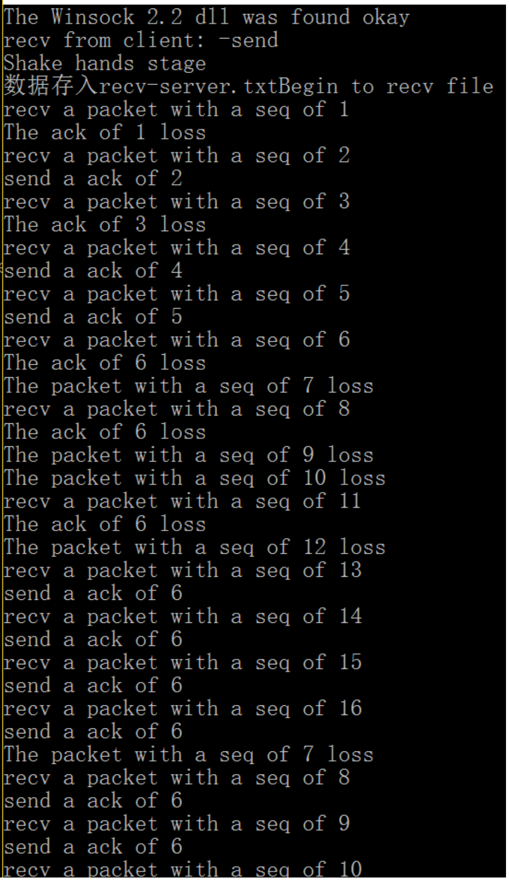
## -send

Client:



```
The Winsock 2.2 dll was found okay
*****************************************
|       -time to get current time       |
|       -quit to exit client            |
|       -testgbn [X] [Y] to test the gbn |
|       -send to senddata the server    |
*****************************************
-send
Begain to send data to Server
Ready for file transmission
Begin a file transfer
File size is 20480B, each packet is 1024B and packe
send a packet with a seq of 0
send a packet with a seq of 1
Recv a ack of 1
send a packet with a seq of 2
send a packet with a seq of 3
Recv a ack of 3
send a packet with a seq of 4
Recv a ack of 4
send a packet with a seq of 5
send a packet with a seq of 6
send a packet with a seq of 7
send a packet with a seq of 8
send a packet with a seq of 9
send a packet with a seq of 10
send a packet with a seq of 11
send a packet with a seq of 12
Recv a ack of 5
send a packet with a seq of 13
Recv a ack of 5
send a packet with a seq of 14
Recv a ack of 5
send a packet with a seq of 15
Recv a ack of 5
Timer out error.
send a packet with a seq of 6
send a packet with a seq of 7
```

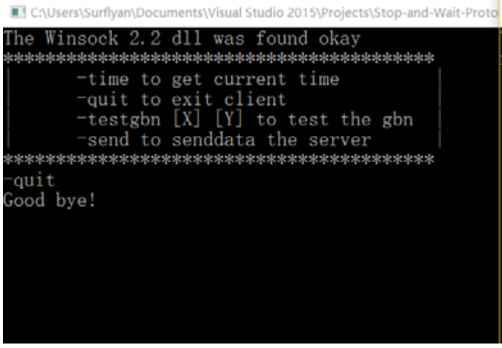
Server:



```
The Winsock 2.2 dll was found okay
recv from client: -send
Shake hands stage
数据存入recv-server.txtBegin to recv file
recv a packet with a seq of 1
The ack of 1 loss
recv a packet with a seq of 2
send a ack of 2
recv a packet with a seq of 3
The ack of 3 loss
recv a packet with a seq of 4
send a ack of 4
recv a packet with a seq of 5
send a ack of 5
recv a packet with a seq of 6
The ack of 6 loss
The packet with a seq of 7 loss
recv a packet with a seq of 8
The ack of 6 loss
The packet with a seq of 9 loss
The packet with a seq of 10 loss
recv a packet with a seq of 11
The ack of 6 loss
The packet with a seq of 12 loss
recv a packet with a seq of 13
send a ack of 6
recv a packet with a seq of 14
send a ack of 6
recv a packet with a seq of 15
send a ack of 6
recv a packet with a seq of 16
send a ack of 6
The packet with a seq of 7 loss
recv a packet with a seq of 8
send a ack of 6
recv a packet with a seq of 9
send a ack of 6
recv a packet with a seq of 10
```

## -quit

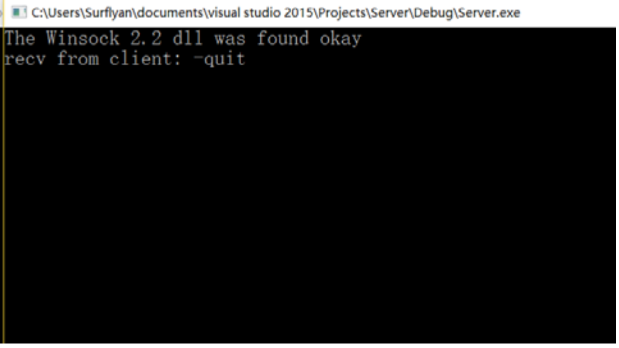
Client:



C:\Users\Surflyan\Documents\Visual Studio 2015\Projects\Stop-and-Wait-Proto

```
The Winsock 2.2 dll was found okay
*****************************************
|       -time to get current time       |
|       -quit to exit client            |
|       -testgbn [X] [Y] to test the gbn |
|       -send to senddata the server    |
*****************************************
-quit
Good bye!
```

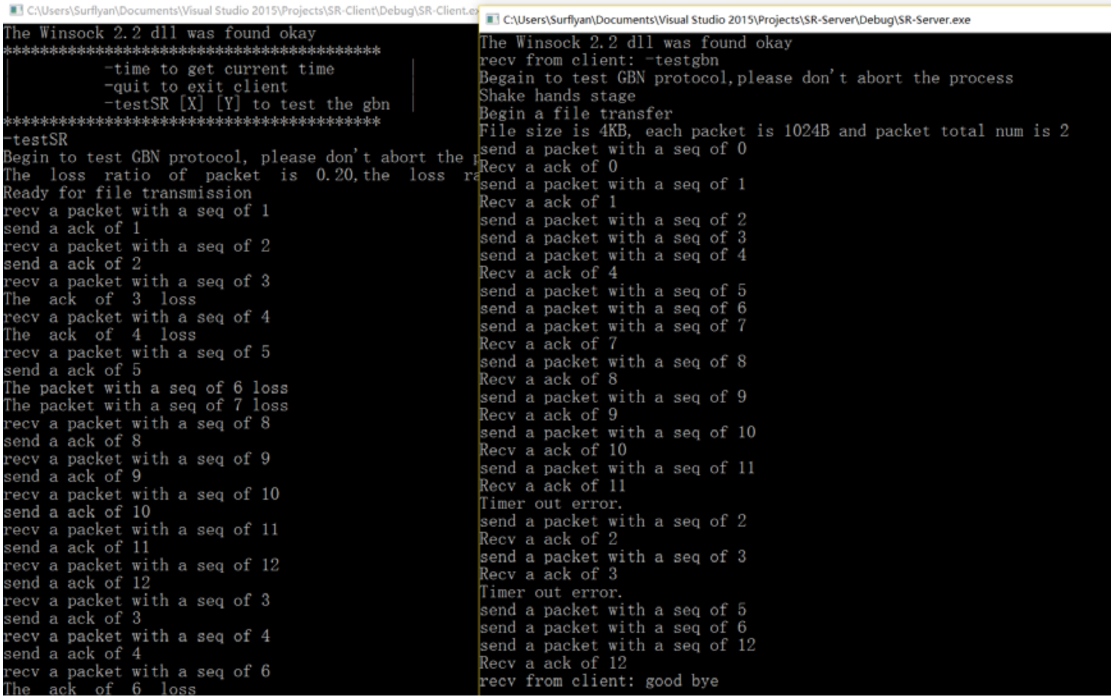
Server:



C:\Users\Surflyan\documents\visual studio 2015\Projects\Server\Debug\Server.exe

```
The Winsock 2.2 dll was found okay
recv from client: -quit
```

## -testSR


C:\Users\Surflyan\Documents\Visual Studio 2015\Projects\SR-Client\Debug\SR-Client.ex
C:\Users\Surflyan\Documents\Visual Studio 2015\Projects\SR-Server\Debug\SR-Server.exe

```
The Winsock 2.2 dll was found okay
****************************************
|          -time to get current time    |
|          -quit to exit client         |
|          -testSR [X] [Y] to test the gbn |
****************************************
-testSR
Begin to test GBN protocol, please don't abort the p
The  loss  ratio  of  packet  is  0.20,the  loss  ra
Ready for file transmission
recv a packet with a seq of 1
send a ack of 1
recv a packet with a seq of 2
send a ack of 2
recv a packet with a seq of 3
The  ack  of  3  loss
recv a packet with a seq of 4
The  ack  of  4  loss
recv a packet with a seq of 5
send a ack of 5
The packet with a seq of 6 loss
The packet with a seq of 7 loss
recv a packet with a seq of 8
send a ack of 8
recv a packet with a seq of 9
send a ack of 9
recv a packet with a seq of 10
send a ack of 10
recv a packet with a seq of 11
send a ack of 11
recv a packet with a seq of 12
send a ack of 12
recv a packet with a seq of 3
send a ack of 3
recv a packet with a seq of 4
send a ack of 4
recv a packet with a seq of 6
The  ack  of  6  loss
```

```
The Winsock 2.2 dll was found okay
recv from client: -testgbn
Begain to test GBN protocol,please don't abort the process
Shake hands stage
Begin a file transfer
File size is 4KB, each packet is 1024B and packet total num is 2
send a packet with a seq of 0
Recv a ack of 0
send a packet with a seq of 1
Recv a ack of 1
send a packet with a seq of 2
send a packet with a seq of 3
send a packet with a seq of 4
Recv a ack of 4
send a packet with a seq of 5
send a packet with a seq of 6
send a packet with a seq of 7
Recv a ack of 7
send a packet with a seq of 8
Recv a ack of 8
send a packet with a seq of 9
Recv a ack of 9
send a packet with a seq of 10
Recv a ack of 10
send a packet with a seq of 11
Recv a ack of 11
Timer out error.
send a packet with a seq of 2
Recv a ack of 2
send a packet with a seq of 3
Recv a ack of 3
Timer out error.
send a packet with a seq of 5
send a packet with a seq of 6
send a packet with a seq of 12
Recv a ack of 12
recv from client: good bye
```

# 四、实验心得

通过此次实验,我对 GBR 和 SR 协议的有了更加深刻的认识，SR 相比 GBR 在效率上有了很大的提高。也对滑动窗口这种机制有了更加切实的体会。在试验过程中，还是遇到了不少坑，主要由于对 SR 协议窗口滑动的细节之前理解有一点偏差，导致没有得到预期的结果。这次实验也让我对 C/S 的数据传输的原理有了一定的了解。

# 五、详细注释源程序

## GBN

服务器

```
 #include <stdlib.h>
#include <time.h>
#include <WinSock2.h>
#include <fstream>
#pragma comment(lib,"ws2_32.lib")
#define SERVER_PORT 12340//端口号
#define SERVER_IP "0.0.0.0" //IP 地址
const int BUFFER_LENGTH = 1026;//缓冲区大小，（以太网中 UDP 的数据帧中包长度应小于1480 字节）
const int SEND_WIND_SIZE = 10;//发送窗口大小为 10，GBN 中应满足 W + 1 <= N（W 为发送窗口大小，N 为序列号个数）
```

```
//本例取序列号0...19共20个
//如果将窗口大小设为1，则为停-等协议
const int SEQ_SIZE = 20; //序列号的个数，从0~19共计20个
//由于发送数据第一个字节如果值为0，则数据会发送失败
//因此接收端序列号为1~20，与发送端一一对应
BOOL ack[SEQ_SIZE];//收到ack情况，对应0~19的ack
int curSeq;//当前数据包的seq
int curAck;//当前等待确认的ack
int totalSeq;//收到的包的总数
int totalPacket;//需要发送的包总数


//************************************
// Method:       getCurTime
// FullName:     getCurTime
// Access:       public
// Returns:      void
// Qualifier: 获取当前系统时间，结果存入ptime中
// Parameter: char * ptime
//************************************
void getCurTime(char *ptime){
	time_t now;
	time(&now);

	// 定义两个变量，存储转换结果
	struct tm tmTmp;


	// 转换为tm结构
	localtime_s(&tmTmp, &now);

	// 转换为字符串并输出
	asctime_s(ptime, 64, &tmTmp);

}

//************************************
// Method:       seqIsAvailable
// FullName:     seqIsAvailable
// Access:       public
// Returns:      bool
// Qualifier: 当前序列号curSeq是否可用
//************************************
bool seqIsAvailable(){
    int step;
    step = curSeq - curAck;
    step = step >= 0 ? step : step + SEQ_SIZE;
    //序列号是否在当前发送窗口之内
    if (step >= SEND_WIND_SIZE){
        return false;
    }
    if (ack[curSeq]){
        return true;
    }
    return false;
}


//***********************************
// Method:       timeoutHandler
// FullName:    timeoutHandler
// Access:       public
// Returns:      void
// Qualifier:  超时重传处理函数，滑动窗口内的数据帧都要重传
//***********************************
void timeoutHandler(){
    printf("Timer out error.\n");
    int index;
    for (int i = 0; i< SEND_WIND_SIZE; ++i){
        index = (i + curAck) % SEQ_SIZE;
        ack[index] = TRUE;
    }
    totalSeq -= SEND_WIND_SIZE;
    curSeq = curAck;
}


//***********************************
// Method:       ackHandler
// FullName:    ackHandler
// Access:       public
// Returns:      void
// Qualifier:  收到 ack，累积确认，取数据帧的第一个字节
//由于发送数据时，第一个字节（序列号）为 0（ASCII）时发送失败，因此加一了，此处需要减一还原
// Parameter: char c
//***********************************
void ackHandler(char c){
    unsigned char index = (unsigned char)c - 1; //序列号减一
    printf("Recv a ack of %d\n", index);
    if (curAck <= index){
        for (int i = curAck; i <= index; ++i){
            ack[i] = TRUE;
        }
        curAck = (index + 1) % SEQ_SIZE;
    }
    else{
        //ack 超过了最大值，回到了 curAck 的左边
        for (int i = curAck; i< SEQ_SIZE; ++i){
            ack[i] = TRUE;

        }
        for (int i = 0; i <= index; ++i){

            ack[i] = TRUE;
        }
        curAck = index + 1;
    }
}

//************************************
// Method:       InitSocket
// FullName:    InitSocket
// Access:       public
// Returns:     BOOL
// Qualifier:  初始化套接字
//************************************
BOOL InitSocket(SOCKET &server){

    //加载套接字库（必须）
    WORD wVersionRequested;
    WSADATA wsaData;

    //套接字加载时错误提示
    int err;
    //版本 2.2
    wVersionRequested = MAKEWORD(2, 2);
    //加载 dll 文件 Scoket 库
    err = WSAStartup(wVersionRequested, &wsaData);
    if (err != 0){
        //找不到 winsock.dll
        printf("WSAStartup failed with error: %d\n", err);
```

```
		return FALSE;
	}
	if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 2)
	{
		printf("Could not find a usable version of Winsock.dll\n");
		WSACleanup();
		return FALSE;
	}

	printf("The Winsock 2.2 dll was found okay\n");
	server = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
	//设置套接字为非阻塞模式
	int iMode = 1; //1：非阻塞，0：阻塞
	ioctlsocket(server, FIONBIO, (u_long FAR*) &iMode);//非阻塞设置
	SOCKADDR_IN addrServer;
	//服务器地址
	//addrServer.sin_addr.S_un.S_addr = inet_addr(SERVER_IP);
	addrServer.sin_addr.S_un.S_addr = htonl(INADDR_ANY);//两者均可
	addrServer.sin_family = AF_INET;
	addrServer.sin_port = htons(SERVER_PORT);

	err = bind(server, (SOCKADDR*)&addrServer, sizeof(SOCKADDR));
	if (err){
		err = GetLastError();
		printf("Could not bind the port %d for socket.Error code is %d\n", SERVER_PORT, err);
		WSACleanup();
		return FALSE;
	}
	return TRUE;
}

BOOL lossInLossRatio(float lossRatio){

	int lossBound = (int)(lossRatio * 100);

	int r = rand() % 101;

	if (r <= lossBound){



		return TRUE;

	}
```

```
    return FALSE;
}

//初始化 ack
void initACK()
{
    for (int i = 0; i < SEQ_SIZE; ++i){
        ack[i] = TRUE;
    }
}


//接收数据
void recv(SOCKET &socketClient, char buffer[], SOCKADDR_IN &addrServer, float
&packetLossRatio, float &ackLossRatio)
{


    int waitCount = 0;
    int rec;
    int stage = 0;
    BOOL b;
    unsigned char u_code;//状态码
    unsigned short seq;//包的序列号
    unsigned short recvSeq = 1;//接收窗口大小为 1，已确认的序列号
    unsigned short waitSeq;//等待的序列号
    int len = sizeof(SOCKADDR);

    BOOL runflag = true;
    printf("Shake hands stage\n");

    while (runflag)
    {

        switch (stage){
        case 0://发送 205 阶段
            buffer[0] = 205;
            sendto(socketClient, buffer, strlen(buffer) + 1, 0, (SOCKADDR*)&addrServer,
sizeof(SOCKADDR));
            Sleep(100);
            stage = 1;
            break;
        case 1://等待接收 200 阶段，没有收到则计数器+1，超时则放弃此次“连接”，等
待从第一步开始
            rec = recvfrom(socketClient, buffer, BUFFER_LENGTH, 0,
((SOCKADDR*)&addrServer), &len);
            if (rec < 0){
                ++waitCount;
                if (waitCount > 20){
                    runflag = false;
                    printf("Timeout error\n");
                    break;
                }
                Sleep(500);
                continue;
            }
            else{

                if ((unsigned char)buffer[0] == 200){
                    printf("Begin to recv file \n");
                    curSeq = 0;
                    curAck = 0;
                    totalSeq = 0;
                    waitCount = 0;
                    waitSeq = 1;
                    stage = 2;
                }
            }
            break;

        case 2://等待接收数据阶段
            rec = recvfrom(socketClient, buffer, BUFFER_LENGTH, 0,
((SOCKADDR*)&addrServer), &len);

            seq = (unsigned short)buffer[0];

            if (rec < 0)
            {
                waitCount++;
            }
            if (waitCount > 10)
            {
                return;
            }

            if (seq == 65491)
```

```
{
    buffer[0] = 211;
    buffer[1] = '\0';
    sendto(socketClient, buffer, 2, 0, (SOCKADDR*)&addrServer, sizeof(SOCKADDR));
    printf("recv finish\n");
    return;

}
//随机法模拟包是否丢失

b = lossInLossRatio(packetLossRatio);


if (b){
    printf("The packet with a seq of %d loss\n", seq);
    continue;
}

printf("recv a packet with a seq of %d\n", seq);

//如果是期待的包，正确接收，正常确认即可
if (!(waitSeq - seq)){

    ++waitSeq;
    if (waitSeq == 21){
        waitSeq = 1;

    }

    //输出数据

    printf("%s\n", &buffer[1]);

    buffer[0] = seq;
    recvSeq = seq;
    buffer[1] = '\0';
}
else{
    //如果当前一个包都没有收到，则等待 Seq 为 1 的数据包，不是则不返回 ACK（因为并没有上一个正确的 ACK）

    if (!recvSeq){
        continue;
```

```
                    }
                    buffer[0] = recvSeq;
                    buffer[1] = '\0';
                }

                b = lossInLossRatio(ackLossRatio);
                if (b){
                    printf("The ack of %d loss\n", (unsigned char)buffer[0]);
                    continue;

                }

                sendto(socketClient, buffer, 2, 0, (SOCKADDR*)&addrServer, sizeof(SOCKADDR));
                printf("send a ack of %d\n", (unsigned char)buffer[0]);
                break;
            }

            Sleep(500);
        }
}


//建立连接，传输数据
void testGBN(SOCKET &sockServer, char buffer[], SOCKADDR_IN &addrClient)
{

    //读入测试数据
    char data[1024 * 20];
    std::ifstream icin;
    icin.open("test.txt");
    ZeroMemory(data, sizeof(data));
    icin.read(data, 1024 * 20);
    icin.close();
    totalPacket = sizeof(data) / 1024;
    int length = sizeof(SOCKADDR);
    //进入 gbn 测试阶段
    //首先 server（server 处于 0 状态）向 client 发送 205 状态码（server 进入 1 状态）
    //server 等待 client 回复 200 状态码， 如果收到 （server 进入 2 状态） ，则开始传
输文件，否则延时等待直至超时\
    //在文件传输阶段，server 发送窗口大小设为
    ZeroMemory(buffer, sizeof(buffer));
    int recvSize;
    int waitCount = 0;
    printf("Begain to test GBN protocol,please don't abort the process\n");
    //加入了一个握手阶段
    //首先服务器向客户端发送一个 205 大小的状态码（我自己定义的）表示服务器准备好了，可以发送数据
    //客户端收到 205 之后回复一个 200 大小的状态码，表示客户端准备好了，可以接收数据了

    //服务器收到 200 状态码之后，就开始使用 GBN 发送数据了
    printf("Shake hands stage\n");
    int stage = 0;
    bool runFlag = true;
    while (runFlag){
            switch (stage){
            case 0://发送 205 阶段
                buffer[0] = 205;
                sendto(sockServer, buffer, strlen(buffer) + 1, 0, (SOCKADDR*)&addrClient, sizeof(SOCKADDR));
                Sleep(100);
                stage = 1;
                break;
            case 1://等待接收 200 阶段，没有收到则计数器+1，超时则放弃此次“连接”，等待从第一步开始
                recvSize = recvfrom(sockServer, buffer, BUFFER_LENGTH, 0, ((SOCKADDR*)&addrClient), &length);
                if (recvSize < 0){
                    ++waitCount;
                    if (waitCount > 20){
                        runFlag = false;
                        printf("Timeout error\n");
                        break;
                    }
                    Sleep(500);
                    continue;
                }
                else{

                    if ((unsigned char)buffer[0] == 200){
                        printf("Begin a file transfer\n");
                        printf("File size is %dB, each packet is 1024B and packet total num is %d\n", sizeof(data), totalPacket);
                        curSeq = 0;
                        curAck = 0;
                        totalSeq = 0;
```

```
                    waitCount = 0;
                    stage = 2;
                }
            }
            break;
        case 2://数据传输阶段
            if (seqIsAvailable()){
                //发送给客户端的序列号从 1 开始
                buffer[0] = curSeq + 1;
                ack[curSeq] = FALSE;
                //数据发送的过程中应该判断是否传输完成

                //为简化过程此处并未实现
                memcpy(&buffer[1], data + 1024 * totalSeq, 1024);
                printf("send a packet with a seq of %d\n", curSeq);
                sendto(sockServer, buffer, BUFFER_LENGTH, 0, (SOCKADDR*)&addrClient,
sizeof(SOCKADDR));

                ++curSeq;
                curSeq %= SEQ_SIZE;

                ++totalSeq;

                Sleep(500);
            }
            //等待 Ack，若没有收到，则返回值为-1，计数器+1
            recvSize = recvfrom(sockServer, buffer, BUFFER_LENGTH, 0,
((SOCKADDR*)&addrClient), &length);
            if (recvSize < 0){
                waitCount++;
                //20 次等待 ack 则超时重传
                if (waitCount > 20)
                {
                    timeoutHandler();
                    waitCount = 0;
                }
            }
            else{
                //收到 ack
                ackHandler(buffer[0]);
                waitCount = 0;
                //断开连接
                if (totalSeq > totalPacket)
                {
                    buffer[0] = 211;
                    buffer[1] = '\0';
                    sendto(sockServer, buffer, BUFFER_LENGTH, 0,
(SOCKADDR*)&addrClient, sizeof(SOCKADDR));
                    printf("send over\n");
                    stage = 3;
                    break;
                }
            }
            Sleep(500);
            break;

        case 3://等待确认断开
            recvSize = recvfrom(sockServer, buffer, BUFFER_LENGTH, 0,
((SOCKADDR*)&addrClient), &length);

            if (recvSize < 0){
                waitCount++;
                //超时重新发送重传确认
                if (waitCount > 5)
                {
                    buffer[0] = 211;
                    buffer[1] = '\0';
                    sendto(sockServer, buffer, strlen(buffer) + 1, 0,
(SOCKADDR*)&addrClient, sizeof(SOCKADDR));


                }
                if (waitCount > 10)
                {
                    return;
                }

            }
            else if (buffer[0] == 211)
            {
                return;
            }

            Sleep(500);
            break;
        }
    }
}
```

```
int main(int argc, char* argv[])
{
    SOCKET sockServer;

    if (InitSocket(sockServer) ==   FALSE)
    {
        goto End;
    }

    SOCKADDR_IN addrClient;//客户端地址
    int length = sizeof(SOCKADDR);

    char buffer[BUFFER_LENGTH]; //数据发送接收缓冲区
    ZeroMemory(buffer, sizeof(buffer));

    float ackloss = 0.2;
    float packetloss = 0.2;

    int interval = 1;
    int recvSize;
    srand((unsigned)time(NULL));
    while (true){
        //非阻塞接收，若没有收到数据，返回值为-1
        recvSize=recvfrom(sockServer, buffer, BUFFER_LENGTH, 0, ((SOCKADDR*)&addrClient),
&length);

        if (recvSize < 0){
            Sleep(200);
            continue;
        }

        if (strcmp(buffer, "-time") == 0){
            printf("recv from client: %s\n", buffer);
            getCurTime(buffer);
            sendto(sockServer, buffer, strlen(buffer) + 1, 0, (SOCKADDR*)&addrClient,
sizeof(SOCKADDR));
        }
        else if (strcmp(buffer, "-quit") == 0){
            printf("recv from client: %s\n", buffer);
            strcpy_s(buffer, strlen("Good bye!") + 1, "Good bye!");
            sendto(sockServer, buffer, strlen(buffer) + 1, 0, (SOCKADDR*)&addrClient,
sizeof(SOCKADDR));
        }
        else if (strcmp(buffer, "-testgbn") == 0){
            printf("recv from client: %s\n", buffer);
            initACK();
            int iMode = 1;
            ioctlsocket(sockServer, FIONBIO, (u_long FAR*) &iMode);
            testGBN(sockServer, buffer, addrClient);
        }
        else if (strcmp(buffer, "-send") == 0){
            printf("recv from client: %s\n", buffer);
            int iMode = 0;
            ioctlsocket(sockServer, FIONBIO, (u_long FAR*) &iMode);
            recv(sockServer, buffer, addrClient, packetloss, ackloss);
        }


        Sleep(500);
    }
//关闭套接字，卸载库
    closesocket(sockServer);
    WSACleanup();
End:
    while (1);
    return 0;
}
```

客户端

```
#include "stdafx.h"
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#include <stdlib.h>
#include <WinSock2.h>
#include <time.h>
#include <stdio.h>
#include <fstream>
#pragma comment(lib,"ws2_32.lib")

#define SERVER_PORT 12340 //接收数据的端口号
#define SERVER_IP "127.0.0.1" // 服务器的 IP 地址
```

```
const int BUFFER_LENGTH = 1026;//缓冲区大小，（以太网中 UDP 的数据帧中包长度应小于
1480 字节）
const int SEND_WIND_SIZE = 10;//发送窗口大小为 10，GBN 中应满足 W + 1 <= N（W 为发
送窗口大小，N 为序列号个数）
//本例取序列号 0...19 共 20 个
//如果将窗口大小设为 1，则为停-等协议
const int SEQ_SIZE = 20; //序列号的个数，从 0~19 共计 20 个
//由于发送数据第一个字节如果值为 0， 则数据会发送失败
//因此接收端序列号为 1~20，与发送端一一对应
BOOL ack[SEQ_SIZE];//收到 ack 情况，对应 0~19 的 ack
int curSeq;//当前数据包的 seq
int curAck;//当前等待确认的 ack
int totalSeq;//收到的包的总数
int totalPacket;//需要发送的包总数


/****************************************************************/
/*          -time 从服务器端获取当前时间
-quit 退出客户端
-testgbn [X] 测试 GBN 协议实现可靠数据传输

[X] [0,1] 模拟数据包丢失的概率

[Y] [0,1] 模拟 ACK 丢失的概率
*/
/****************************************************************/
void printTips(){

    printf("******************************************\n");

    printf("|           -time to get current time                 | \n");

    printf("|           -quit to exit client                        | \n");

    printf("|           -testgbn [X] [Y] to test the gbn    | \n");
    printf("|           -send to senddata the server          | \n");

    printf("******************************************\n");
}
//************************************
// Method:        lossInLossRatio
// FullName:    lossInLossRatio
// Access:        public
// Returns:     BOOL
// Qualifier: 根据丢失率随机生成一个数字，判断是否丢失,丢失则返回 TRUE，否则返回
FALSE
// Parameter: float lossRatio [0,1]
//************************************
BOOL lossInLossRatio(float lossRatio){

    int lossBound = (int)(lossRatio * 100);

    int r = rand() % 101;

    if (r <= lossBound){


        return TRUE;

    }

    return FALSE;
}

bool seqIsAvailable(){
    int step;
    step = curSeq - curAck;
    step = step >= 0 ? step : step + SEQ_SIZE;
    //序列号是否在当前发送窗口之内
    if (step >= SEND_WIND_SIZE){
        return false;
    }
    if (ack[curSeq]){
        return true;
    }
    return false;
}

void timeoutHandler(){
    printf("Timer out error.\n");
    int index;
    for (int i = 0; i< SEND_WIND_SIZE; ++i){
        index = (i + curAck) % SEQ_SIZE;
        ack[index] = TRUE;
    }
    totalSeq -= SEND_WIND_SIZE;
    curSeq = curAck;
```

```
}

void ackHandler(char c){
    unsigned char index = (unsigned char)c - 1; //序列号减一
    printf("Recv a ack of %d\n", index);
    if (curAck <= index){
        for (int i = curAck; i <= index; ++i){
            ack[i] = TRUE;
        }
        curAck = (index + 1) % SEQ_SIZE;
    }
    else{
        //ack超过了最大值，回到了curAck的左边
        for (int i = curAck; i< SEQ_SIZE; ++i){
            ack[i] = TRUE;

        }
        for (int i = 0; i <= index; ++i){

            ack[i] = TRUE;
        }
        curAck = index + 1;
    }
}
//初始化
int init()
{
    //加载套接字库（必须）

    WORD wVersionRequested;

    WSADATA wsaData;

    //套接字加载时错误提示
    int err;
    //版本2.2
    wVersionRequested = MAKEWORD(2, 2);
    //加载dll文件Scoket库
    err = WSAStartup(wVersionRequested, &wsaData);
    if (err != 0){
        //找不到winsock.dll
        printf("WSAStartup failed with error: %d\n", err);
        return -1;
    }
    if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 2)
    {
        printf("Could not find a usable version of Winsock.dll\n");
        WSACleanup();
        return -1;

    }

    printf("The Winsock 2.2 dll was found okay\n");
    return 0;

}
//初始化 ACK
void initACK()
{
    for (int i = 0; i < SEQ_SIZE; ++i){
        ack[i] = TRUE;
    }
}


//接收数据
void recv(SOCKET &socketClient, char buffer[], SOCKADDR_IN &addrServer, float
&packetLossRatio, float &ackLossRatio)
{
    printf("%s\n", "Begin to test GBN protocol, please don't abort the      process");

    printf("The loss ratio of packet is %.2f,the loss ratio of ack is %.2f\n", packetLossRatio,
ackLossRatio);
    int waitCount = 0;
    int rec;
    int stage = 0;
    BOOL b;
    unsigned char u_code;//状态码
    unsigned short seq;//包的序列号
    unsigned short recvSeq;//接收窗口大小为 1，已确认的序列号
    unsigned short waitSeq;//等待的序列号
    int len = sizeof(SOCKADDR);
    sendto(socketClient, "-testgbn", strlen("-testgbn") + 1, 0, (SOCKADDR*)&addrServer,
sizeof(SOCKADDR));
    BOOL runflag = true;
    while (runflag)
    {
        //等待 server 回复设置 UDP 为阻塞模式
        rec = recvfrom(socketClient, buffer, BUFFER_LENGTH, 0, (SOCKADDR*)&addrServer, &len);
        switch (stage){
        case 0://等待握手阶段
            u_code = (unsigned char)buffer[0];
            if ((unsigned char)buffer[0] == 205)
            {
                printf("Ready for file transmission\n");
                buffer[0] = 200;
                buffer[1] = '\0';
                sendto(socketClient, buffer, 2, 0, (SOCKADDR*)&addrServer, sizeof(SOCKADDR));
                stage = 1;
                recvSeq = 0;
                waitSeq = 1;
            }
            break;

        case 1://等待接收数据阶段
            seq = (unsigned short)buffer[0];
            if (rec < 0)
            {
                waitCount++;
            }
            if (waitCount > 10)
            {
                return;
            }
            if (seq == 65491)
            {
                buffer[0] = 211;
                buffer[1] = '\0';
                sendto(socketClient, buffer, 2, 0, (SOCKADDR*)&addrServer, sizeof(SOCKADDR));

                printf("recv finish\n");
                return;

            }
            //随机法模拟包是否丢失

            b = lossInLossRatio(packetLossRatio);

            if (b){
```

```
            printf("The packet with a seq of %d loss\n", seq);
            continue;
        }
        printf("recv a packet with a seq of %d\n", seq);

        //如果是期待的包，正确接收，正常确认即可
        if (!(waitSeq - seq)){

            ++waitSeq;
            if (waitSeq == 21){
                waitSeq = 1;

            }

            //输出数据

            printf("%s\n", &buffer[1]);

            buffer[0] = seq;
            recvSeq = seq;
            buffer[1] = '\0';
        }
        else{
            //如果当前一个包都没有收到，则等待Seq为1的数据包，不是则不返回ACK（因为并没有上一个正确的ACK）

            if (!recvSeq){
                continue;



            }
            buffer[0] = recvSeq;
            buffer[1] = '\0';
        }

        b = lossInLossRatio(ackLossRatio);
        if (b){
            printf("The ack of %d loss\n", (unsigned char)buffer[0]);
            continue;

        }

        sendto(socketClient, buffer, 2, 0, (SOCKADDR*)&addrServer, sizeof(SOCKADDR));
        printf("send a ack of %d\n", (unsigned char)buffer[0]);
```

```
                        break;

                }

                Sleep(500);

        }
}


//发送数据
void sendData(SOCKET &sockServer, char buffer[], SOCKADDR_IN &addrClient)
{

        //读入测试数据
        char data[1024 * 20];
        std::ifstream icin;
        icin.open("test.txt");
        ZeroMemory(data, sizeof(data));
        icin.read(data, 1024 * 20);
        icin.close();
        totalPacket = sizeof(data) / 1024;
        int length = sizeof(SOCKADDR);

        ZeroMemory(buffer, sizeof(buffer));
        int recvSize = 0;
        int waitCount = 0;
        int len = sizeof(SOCKADDR);
        int rec = 0;

        printf("Begain to send data to Server\n");

        int stage = 0;

        sendto(sockServer, "-send", strlen("-send") + 1, 0, (SOCKADDR*)&addrClient,
sizeof(SOCKADDR));
        bool runFlag = true;
        while (runFlag){
                switch (stage){
                case 0://等待握手阶段
                        rec = recvfrom(sockServer, buffer, BUFFER_LENGTH, 0, (SOCKADDR*)&addrClient,
&len);
                        if ((unsigned char)buffer[0] == 205)
                        {
```

```
                printf("Ready for file transmission\n");
                buffer[0] = 200;
                buffer[1] = '\0';
                sendto(sockServer, buffer, 2, 0, (SOCKADDR*)&addrClient,
sizeof(SOCKADDR));
                stage = 1;
            }
            break;

        case 1://准备发送数据

            printf("Begin a file transfer\n");
            printf("File size is %dB, each packet is 1024B and packet total num is %d\n",
sizeof(data), totalPacket);
            curSeq = 0;
            curAck = 0;
            totalSeq = 0;
            waitCount = 0;
            stage = 2;
            break;

        case 2://数据传输阶段
            if (seqIsAvailable()){
                //发送给客户端的序列号从 1 开始
                buffer[0] = curSeq + 1;
                ack[curSeq] = FALSE;



                memcpy(&buffer[1], data + 1024 * totalSeq, 1024);
                printf("send a packet with a seq of %d\n", curSeq);

                sendto(sockServer, buffer, BUFFER_LENGTH, 0, (SOCKADDR*)&addrClient,
sizeof(SOCKADDR));

                ++curSeq;
                curSeq %= SEQ_SIZE;

                ++totalSeq;

                Sleep(500);
            }

            //等待 Ack，若没有收到，则返回值为-1，计数器+1
            recvSize = recvfrom(sockServer, buffer, BUFFER_LENGTH, 0,
((SOCKADDR*)&addrClient), &length);

            if (recvSize < 0){
                waitCount++;
                //20 次等待 ack 则超时重传
                if (waitCount > 20)
                {
                    timeoutHandler();
                    waitCount = 0;
                }
            }
            else{
                //收到 ack
                ackHandler(buffer[0]);
                waitCount = 0;
                //断开连接
                if (totalSeq > totalPacket)
                {

                    buffer[0] = 211;
                    buffer[1] = '\0';
                    sendto(sockServer, buffer, BUFFER_LENGTH, 0,
(SOCKADDR*)&addrClient, sizeof(SOCKADDR));
                    printf("send over\n");
                    stage = 3;
                    break;
                }
            }
            Sleep(500);
            break;

        case 3://等待确认断开
            recvSize = recvfrom(sockServer, buffer, BUFFER_LENGTH, 0,
((SOCKADDR*)&addrClient), &length);

            if (recvSize < 0){
                waitCount++;
                //超时重新发送重传确认
                if (waitCount > 2)
                {
                    buffer[0] = 211;
                    buffer[1] = '\0';
                    sendto(sockServer,      buffer,      strlen(buffer)      +      1,      0,
(SOCKADDR*)&addrClient, sizeof(SOCKADDR));

                }
                if (waitCount > 5)
                {
                    return;
                }

            }
            else if (buffer[0] == 211)
            {
                return;
            }
            Sleep(500);
            break;
        }
    }
}

int main(int argc, char* argv[])
{

    if (init() < 0)

    {
        goto End;
    }

    SOCKET socketClient = socket(AF_INET, SOCK_DGRAM, 0);
    //int iMode = 1; //1：非阻塞，0：阻塞
    //ioctlsocket(socketClient, FIONBIO, (u_long FAR*) &iMode);
    SOCKADDR_IN addrServer;

    addrServer.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");

    addrServer.sin_family = AF_INET;

    addrServer.sin_port = htons(SERVER_PORT);

    //接收缓冲区

    char buffer[BUFFER_LENGTH];
```

```
ZeroMemory(buffer, sizeof(buffer));

int len = sizeof(SOCKADDR);

//为了测试与服务器的连接，可以使用 -time 命令从服务器端获得当前时间
//使用 -testgbn [X] [Y] 测试 GBN 其中[X]表示数据包丢失概率

// [Y]表示 ACK 丢包概率
printTips();

int ret;

int interval = 1;//收到数据包之后返回 ack 的间隔，默认为 1 表示每个都返回 ack，0 或者负数均表示所有的都不返回 ack

char cmd[128];

float packetLossRatio = 0.2; //默认包丢失率 0.2

float ackLossRatio = 0.2;
//默认 ACK 丢失率 0.2

//用时间作为随机种子，放在循环的最外面

srand((unsigned)time(NULL));

while (true){
    gets_s(buffer);


    ret  =sscanf_s(buffer, "%s%f%f", cmd, 128,&packetLossRatio, &ackLossRatio);


    //开始 GBN 测试，使用 GBN 协议实现 UDP 可靠文件传输

    if (!strcmp(cmd, "-testgbn")){

        int iMode = 0; //1：非阻塞，0：阻塞
        ioctlsocket(socketClient, FIONBIO, (u_long FAR*) &iMode);
        recv(socketClient, buffer, addrServer, packetLossRatio, ackLossRatio);



    }
    else if (!strcmp(cmd, "-send"))//向服务器发送数据
```

```
        {
            initACK();
            int iMode = 1; //1：非阻塞，0：阻塞
            ioctlsocket(socketClient, FIONBIO, (u_long FAR*) &iMode);//非阻塞设置
            sendData(socketClient, buffer, addrServer);
        }


        else
        {
            sendto(socketClient, buffer, strlen(buffer) + 1, 0, (SOCKADDR*)&addrServer, sizeof(SOCKADDR));

            ret = recvfrom(socketClient, buffer, BUFFER_LENGTH, 0, (SOCKADDR*)&addrServer, &len);
            printf("%s\n", buffer);
        }

        if (!strcmp(buffer, "Good bye!")){
            break;
        }



        printTips();

    }

    //关闭套接字

    closesocket(socketClient);

    WSACleanup();
End:
    while (1);
    return 0;
}
```

## SR

### Server：

```
int main(int argc, char* argv[])
```

```
{
    //加载套接字库（必须）
    WORD wVersionRequested;
    WSADATA wsaData;
    //套接字加载时错误提示
    int err;
    //版本 2.2
    wVersionRequested = MAKEWORD(2, 2);
    //加载 dll 文件 Scoket 库
    err = WSAStartup(wVersionRequested, &wsaData);
    if (err != 0){
        //找不到 winsock.dll
        printf("WSAStartup failed with error: %d\n", err);
        return -1;
    }
    if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 2)
    {
        printf("Could not find a usable version of Winsock.dll\n");
        WSACleanup();
    }
    else{
        printf("The Winsock 2.2 dll was found okay\n");
    }
    SOCKET sockServer = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    //设置套接字为非阻塞模式
    int iMode = 1; //1：非阻塞，0：阻塞
    ioctlsocket(sockServer, FIONBIO, (u_long FAR*) &iMode);//非阻塞设置
    SOCKADDR_IN addrServer;     //服务器地址
    //addrServer.sin_addr.S_un.S_addr = inet_addr(SERVER_IP);
    addrServer.sin_addr.S_un.S_addr = htonl(INADDR_ANY);//两者均可
    addrServer.sin_family = AF_INET;
    addrServer.sin_port = htons(SERVER_PORT);
    err = bind(sockServer, (SOCKADDR*)&addrServer, sizeof(SOCKADDR));
    if (err){
        err = GetLastError();
        printf("Could not bind the port %d for socket.Error code
is %d\n",SERVER_PORT,err);
            WSACleanup();
        return -1;
    }

    SOCKADDR_IN addrClient;     //客户端地址
    int length = sizeof(SOCKADDR);
    char buffer[BUFFER_LENGTH]; //数据发送接收缓冲区
    ZeroMemory(buffer, sizeof(buffer));
    //将测试数据读入内存

    HANDLE fhadle = CreateFile("../test.txt",
        0, 0, NULL, OPEN_ALWAYS, 0, 0
        );
    int length_lvxiya = GetFileSize(fhadle, 0);
    totalPacket = length_lvxiya / 1024 + 1;
    char *data = new char[1024 * (totalPacket + SEND_WIND_SIZE*SEND_WIND_SIZE)];
    ZeroMemory(data, 1024 * (totalPacket + SEND_WIND_SIZE * SEND_WIND_SIZE));
    std::ifstream icin;
    icin.open("../test.txt");

    //char data[1024 * 113];
    //ZeroMemory(data, sizeof(data));
    icin.read(data, 1024 * (totalPacket + SEND_WIND_SIZE * SEND_WIND_SIZE));
    icin.close();



    int recvSize;
    for (int i = 0; i < SEQ_SIZE; ++i){
        ack[i] = 1;
    }
    while (true){
        //非阻塞接收，若没有收到数据，返回值为-1
        recvSize =
            recvfrom(sockServer, buffer, BUFFER_LENGTH, 0, ((SOCKADDR*)&addrClient), &length);
        if (recvSize < 0){
            Sleep(200);
            continue;
        }
        printf("recv from client: %s\n", buffer);
        if (strcmp(buffer, "-time") == 0){
            getCurTime(buffer);
        }
        else if (strcmp(buffer, "-quit") == 0){
            strcpy_s(buffer, strlen("Good bye!") + 1, "Good bye!");
        }
        else if (strcmp(buffer, "-testgbn") == 0){
            //进入 gbn 测试阶段
            //首先 server（server 处于 0 状态）向 client 发送 205 状态码（server 进入 1 状态）
            //server 等待 client 回复 200 状态码，如果收到（server 进入 2 状态），则开始传输文件，否则延时等待直至超时\
            //在文件传输阶段，server 发送窗口大小设为
            ZeroMemory(buffer, sizeof(buffer));
        int recvSize;
        int waitCount = 0;
        printf("Begain to test GBN protocol,please don't abort the process\n");
        //加入了一个握手阶段
        //首先服务器向客户端发送一个 205 大小的状态码（我自己定义的）表示服务器准备好了，可以发送数据
            //客户端收到 205 之后回复一个 200 大小的状态码，表示客户端准备好了，可以接收数据了
            //服务器收到 200 状态码之后，就开始使用 GBN 发送数据了
            printf("Shake hands stage\n");
        int stage = 0;
        bool runFlag = true;
        while (runFlag){
            switch (stage){
            case 0://发送 205 阶段
                buffer[0] = 205;
                sendto(sockServer, buffer, strlen(buffer) + 1, 0,
                    (SOCKADDR*)&addrClient, sizeof(SOCKADDR));
                Sleep(100);
                stage = 1;
                break;
            case 1://等待接收 200 阶段，没有收到则计数器+1，超时则放弃此次“连接”，等待从第一步开始
                    recvSize =
                    recvfrom(sockServer, buffer, BUFFER_LENGTH, 0, ((SOCKADDR*)&addrClient), &length);
                if (recvSize < 0){
                    ++waitCount;
                    if (waitCount > 20){
                        runFlag = false;
                        printf("Timeout error\n");
                        break;
                    }
                    Sleep(500);
                        continue;
                }
                else{
                    if ((unsigned char)buffer[0] == 200){
                        printf("Begin a file transfer\n");
                        printf("File size is %dKB, each packet is 1024B and packet total
num is %d\n",sizeof(data),totalPacket);
                                curSeq = 0;
                            curAck = 0;
                            totalSeq = 0;
                            waitCount = 0;
                            stage = 2;
                        }
                    }
                    break;
                case 2://数据传输阶段
                    if (seqIsAvailable()&&totalSeq-SEND_WIND_SIZE<=totalPacket){
                        //发送给客户端的序列号从 1 开始
                        buffer[0] = curSeq + 1;
                        ack[curSeq] = 0;
                        //数据发送的过程中应该判断是否传输完成
                        //为简化过程此处并未实现
                        memcpy(&buffer[1], data + 1024 * (curSeq + (totalSeq /
SEND_WIND_SIZE)*SEND_WIND_SIZE), 1024);
                        printf("send a packet with a seq of %d\n", curSeq);
                        sendto(sockServer, buffer, BUFFER_LENGTH, 0,
                            (SOCKADDR*)&addrClient, sizeof(SOCKADDR));
                        ++curSeq;
                        curSeq %= SEQ_SIZE;
                        ++totalSeq;
                        Sleep(500);
                    }

                    else if (curSeq - curAck >= 0 ? curSeq - curAck <= SEND_WIND_SIZE :
curSeq - curAck + SEQ_SIZE <= SEND_WIND_SIZE && totalSeq - SEND_WIND_SIZE <= totalPacket){
                        curSeq++;
                        curSeq %= SEQ_SIZE;
                    }
                    else if (totalSeq - SEND_WIND_SIZE > totalPacket){
                        memcpy(buffer,"good bye\0",9);
                        runFlag = false;
                        break;
                    }
                    //等待 Ack，若没有收到，则返回值为-1，计数器+1
                    recvSize =
                        recvfrom(sockServer, buffer, BUFFER_LENGTH, 0,
((SOCKADDR*)&addrClient), &length);
                    if (recvSize < 0){
                        waitCount++;
                        //20 次等待 ack 则超时重传
                        if (waitCount > 20)
                        {
                            timeoutHandler();
                            waitCount = 0;
                        }
                    }
                    else{
                        //收到 ack
                        ackHandler(buffer[0]);
                        waitCount = 0;
                    }
                    Sleep(500);
                    break;
                }
            }
        }
        sendto(sockServer, buffer, strlen(buffer) + 1, 0, (SOCKADDR*)&addrClient,
            sizeof(SOCKADDR));
        Sleep(500);
    }
    //关闭套接字，卸载库
    closesocket(sockServer);
    WSACleanup();
    return 0;
}
```

**Client**

```
int main(int argc, char* argv[])
{
    //加载套接字库（必须）
    WORD wVersionRequested;
    WSADATA wsaData;
    //套接字加载时错误提示
    int err;
    //版本 2.2
    wVersionRequested = MAKEWORD(2, 2);
    //加载 dll 文件 Scoket 库
    err = WSAStartup(wVersionRequested, &wsaData);
    if (err != 0){
        //找不到 winsock.dll
        printf("WSAStartup failed with error: %d\n", err);
        return 1;
    }
    if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 2)
    {
        printf("Could not find a usable version of Winsock.dll\n");
        WSACleanup();
    }
    else{
        printf("The Winsock 2.2 dll was found okay\n");
    }
    SOCKET socketClient = socket(AF_INET, SOCK_DGRAM, 0);
    SOCKADDR_IN addrServer;
    addrServer.sin_addr.S_un.S_addr = inet_addr(SERVER_IP);
    addrServer.sin_family = AF_INET;
    addrServer.sin_port = htons(SERVER_PORT);
    //接收缓冲区
    char buffer[BUFFER_LENGTH];
    ZeroMemory(buffer, sizeof(buffer));
    int len = sizeof(SOCKADDR);
    //为了测试与服务器的连接，可以使用  -time  命令从服务器端获得当前时间
    //使用  -testgbn [X] [Y]  测试 GBN  其中[X]表示数据包丢失概率
    //                        [Y]表示 ACK 丢包概率
    printTips();
    int ret;
    int interval = 1;//收到数据包之后返回 ack 的间隔，默认为 1 表示每个都返回 ack，0
或者负数均表示所有的都不返回 ack
    char cmd[128];
    float packetLossRatio = 0.2;  //默认包丢失率 0.2
    float ackLossRatio = 0.2;  //默认 ACK 丢失率 0.2
    //用时间作为随机种子，放在循环的最外面
    srand((unsigned)time(NULL));
    while (true){
        gets_s(buffer);
        ret = sscanf(buffer, "%s%f%f", &cmd, &packetLossRatio, &ackLossRatio);
        //开始 GBN 测试，使用 GBN 协议实现 UDP 可靠文件传输
        if (!strcmp(cmd, "-testSR")){
            printf("%s\n", "Begin to test GBN protocol, please don't abort the process");
            printf("The loss ratio of packet is %.2f,the loss ratio of ack
is %.2f\n", packetLossRatio, ackLossRatio);
            int waitCount = 0;
            int stage = 0;
            BOOL b;
            unsigned char u_code;//状态码
            unsigned short seq;//包的序列号
            unsigned short recvSeq;//已确认的最大序列号
            unsigned short waitSeq;//等待的序列号 ，窗口大小为 10，这个为最小的值
            char  buffer_1[RECV_WIND_SIZE][BUFFER_LENGTH];//接收到的缓冲区数据
```

```
----------------add bylvxiya
            int i_state = 0;
            for (i_state = 0; i_state < RECV_WIND_SIZE; i_state++){
                ZeroMemory(buffer_1[i_state],sizeof(buffer_1[i_state]));
            }

            BOOL ack_send[RECV_WIND_SIZE];//ack 发送情况的记录，对应 1-20 的 ack,刚开
始全为 false
            int success_number=0;// 窗口内成功接收的个数
            for (i_state = 0; i_state < RECV_WIND_SIZE; i_state++){//记录哪一个成功接收了
                ack_send[i_state] = false;
            }
            std::ofstream out_result;
            out_result.open("result.txt", std::ios::out | std::ios:: trunc);
            if (!out_result.is_open()){
                printf("文件打开失败！！！ \n");
                continue;
            }
            //---------------------------------
            sendto(socketClient, "-testgbn", strlen("-testgbn") + 1, 0,
                (SOCKADDR*)&addrServer, sizeof(SOCKADDR));
            while (true)
            {
                //等待 server 回复设置 UDP 为阻塞模式
               recvfrom(socketClient, buffer, BUFFER_LENGTH, 0, (SOCKADDR*)&addrServer,
&len);
                switch (stage){
                case 0://等待握手阶段
                    u_code = (unsigned char)buffer[0];
                    if ((unsigned char)buffer[0] == 205)
                    {
                        printf("Ready for file transmission\n");
                        buffer[0] = 200;
                        buffer[1] = '\0';
                        sendto(socketClient, buffer, 2, 0,
                            (SOCKADDR*)&addrServer, sizeof(SOCKADDR));
                        stage = 1;
                        recvSeq = 0;
                        waitSeq = 1;
                    }
                    break;
                case 1://等待接收数据阶段
                    if (!memcmp(buffer, "good bye\0", 9)){
                        printf("数据传输成功！！！ \n");
```

```
        goto success;
    }
    seq = (unsigned short)buffer[0];
    //随机法模拟包是否丢失
    b = lossInLossRatio(packetLossRatio);
    if (b){
        printf("The packet with a seq of %d loss\n", seq);
        continue;
    }
    printf("recv a packet with a seq of %d\n", seq);
    //如果是期待的包的范围，正确接收，正常确认即可，如果小于期待的范围，直接回应 ack
    if ((seq<waitSeq && (waitSeq + RECV_WIND_SIZE>SEQ_SIZE ? seq >= (waitSeq + RECV_WIND_SIZE) % SEQ_SIZE :true)))//在接收窗口范围内
    {
        buffer[0] = seq;
        buffer[1] = '\0';
    }
    else if (seq >= waitSeq && (waitSeq + RECV_WIND_SIZE>SEQ_SIZE ? true: seq < (waitSeq + RECV_WIND_SIZE))){//在接收窗口范围内
        /*if (!(seq - waitSeq))
        {
            ++waitSeq;
            if (waitSeq == 21){
                waitSeq = 1;
            }
            //在这里应该向上层交付数据
        }*/
        memcpy(buffer_1[seq - waitSeq], &buffer[1], sizeof(buffer));
        ack_send[seq - waitSeq] = true;
        int ack_s = 0;
        while (ack_send[ack_s] && ack_s<RECV_WIND_SIZE){
            //向上层传输数据
            out_result << buffer_1[ack_s];
            //printf("%s",buffer_1[ack_s - 1]);
            ZeroMemory(buffer_1[ack_s], sizeof(buffer_1[ack_s]));
            waitSeq++;
            if (waitSeq == 21){
                waitSeq = 1;
            }
            ack_s = ack_s + 1;
        }
        if (ack_s > 0){
            for (int i = 0; i < RECV_WIND_SIZE; i++){
                if (ack_s + i < RECV_WIND_SIZE)
                {
                    ack_send[i] = ack_send[i + ack_s];
                    memcpy(buffer_1[i],     buffer_1[i     +     ack_s], sizeof(buffer_1[i + ack_s]));
                    ZeroMemory(buffer_1[i + ack_s], sizeof(buffer_1[i + ack_s]));
                }
                else
                {
                    ack_send[i] = false;
                    ZeroMemory(buffer_1[i], sizeof(buffer_1[i]));
                }

            }
        }

        //输出数据
        //printf("%s\n",&buffer[1]);
        buffer[0] = seq;
        recvSeq = seq;
        buffer[1] = '\0';
    }
    else{
        //如果当前一个包都没有收到，则等待 Seq 为 1 的数据包，不是则不返回 ACK（因为并没有上一个正确的 ACK）
        if (!recvSeq){
            continue;
        }
        buffer[0] = recvSeq;
        buffer[1] = '\0';
    }
    b = lossInLossRatio(ackLossRatio);
    if (b){
        printf("The   ack   of   %d   loss\n", (unsigned
            char)buffer[0]);
        continue;
    }
    sendto(socketClient, buffer, 2, 0,
        (SOCKADDR*)&addrServer, sizeof(SOCKADDR));
    printf("send a ack of %d\n", (unsigned char)buffer[0]);
    break;
}
Sleep(500);
```

```
				}
			success:			out_result.close();
			}
			sendto(socketClient, buffer, strlen(buffer) + 1, 0,
				(SOCKADDR*)&addrServer, sizeof(SOCKADDR));
			ret =
				recvfrom(socketClient, buffer, BUFFER_LENGTH, 0, (SOCKADDR*)&addrServer,
				&len);
			printf("%s\n", buffer);
			if (!strcmp(buffer, "Good bye!")){
				break;
			}
			printTips();
		}
		//关闭套接字
		closesocket(socketClient);
		WSACleanup();
		return 0;
	}
```