计算机网络实验报告3-2

姓名：高祎珂　　学号：2011743

实验要求

在实验3-1的基础上，将**停等机制**改成基于**滑动窗口**的流量控制机制，采用**固定窗口大小**，支持**累积确认**，完成给定测试文件的传输。

实验原理

1、数据报套接字UDP:

UDP是User Datagram Protocol的简称,中文名是用户数据报协议,是OSI参考模型中的传输层协议,它是一种无连接的传输层协议,提供面向事务的简单不可靠信息传送服务.

2、建立连接:

根据TCP的三次握手和四次挥手原则，进行发送端与接收端的连接和断连。

3、流水线协议

在确认未返回之前允许发送多个分组。

4、滑动窗口

滑动窗口(Sliding window)是一种流量控制技术。如果网络通信中，通信双方不会考虑网络的拥挤情况直接发送数据，由于大家不知道网络拥塞状况，同时发送数据，则会导致中间节点阻塞掉包，谁也发不了数据， 所以就有了滑动窗又机制来解决此问题。 TCP中采用滑动窗口来进行传输控制，滑动窗又的大小意味着接收方还有多大的缓冲区可以用于接收数据。发送方可以通过滑动窗口的大小来确定应该发送多少字节的数据。当滑动窗口余量为0时，发送方一般不能再发送数据报，但有两种情况除外，一种情况是可以发送紧急数据，例如，允许用户终止在远端机上的运行进程。另一种情况是发送方可以发送一个1字节的数据报来通知接收方重新声明，新声明它希望接收的下一字节及发送方的滑动窗口大小。

5、累积确认

Go-Back-N协议采用累积确认的方式，GBN的关键是发 送方能够在收到确认之前发送多个分组,但接收方只能缓存一个分组。发送方为发送出去的分组保留副本，直到来自接收方确认达到。

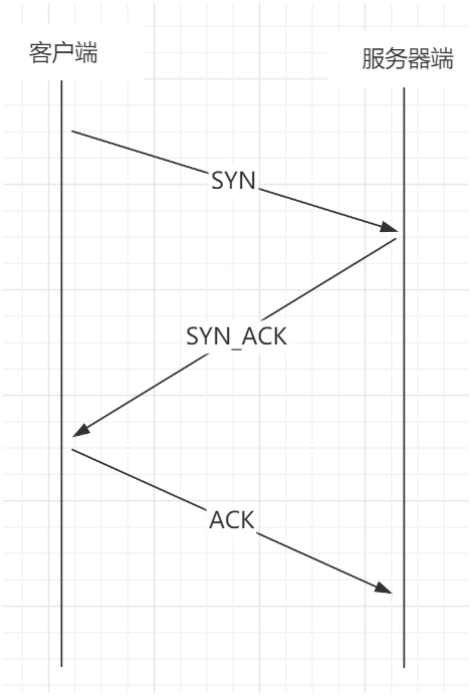
协议设计

- **报文格式**

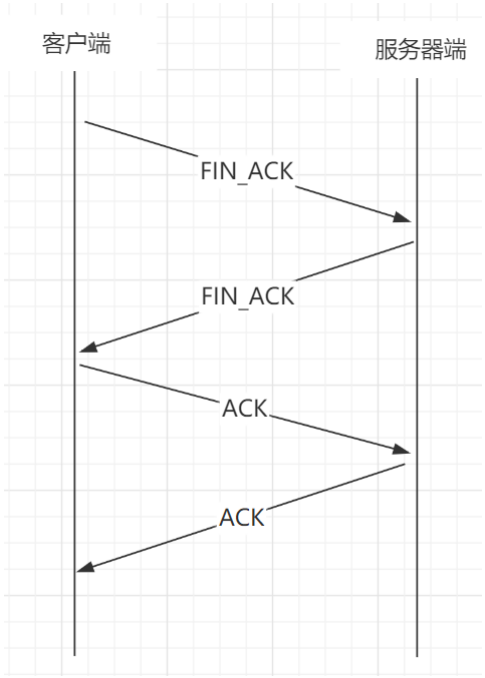
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
序列号															
数据长度															
					NAK							EOF	FIN	ACK	SYN
校验和															
设定大小的数据															

前16位为数据长度，用于记录数据区大小，17-32位位校验和，用于检验传输的正确性，33-40为标志位，40-48位为传输的数据包的序列号。

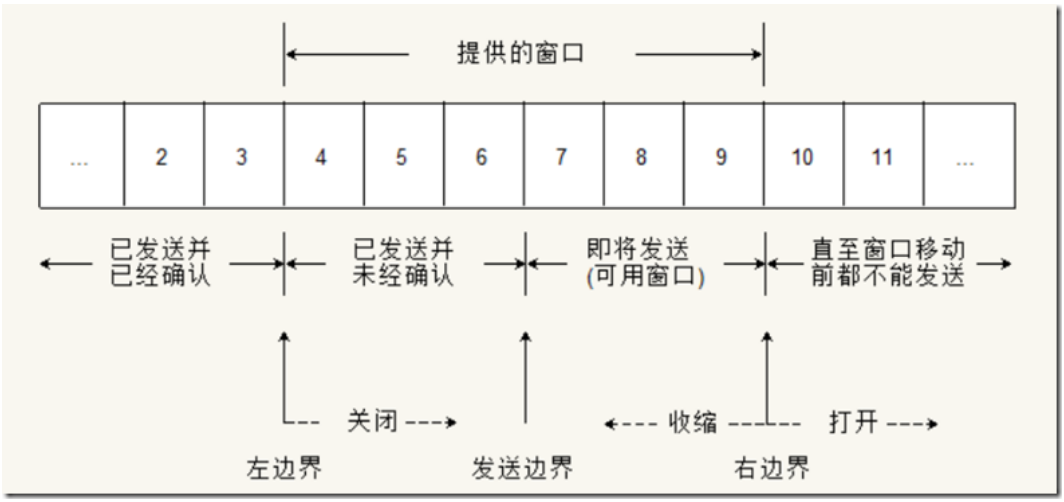
- **连接与断开**
  - **三次握手**



  - **四次挥手**



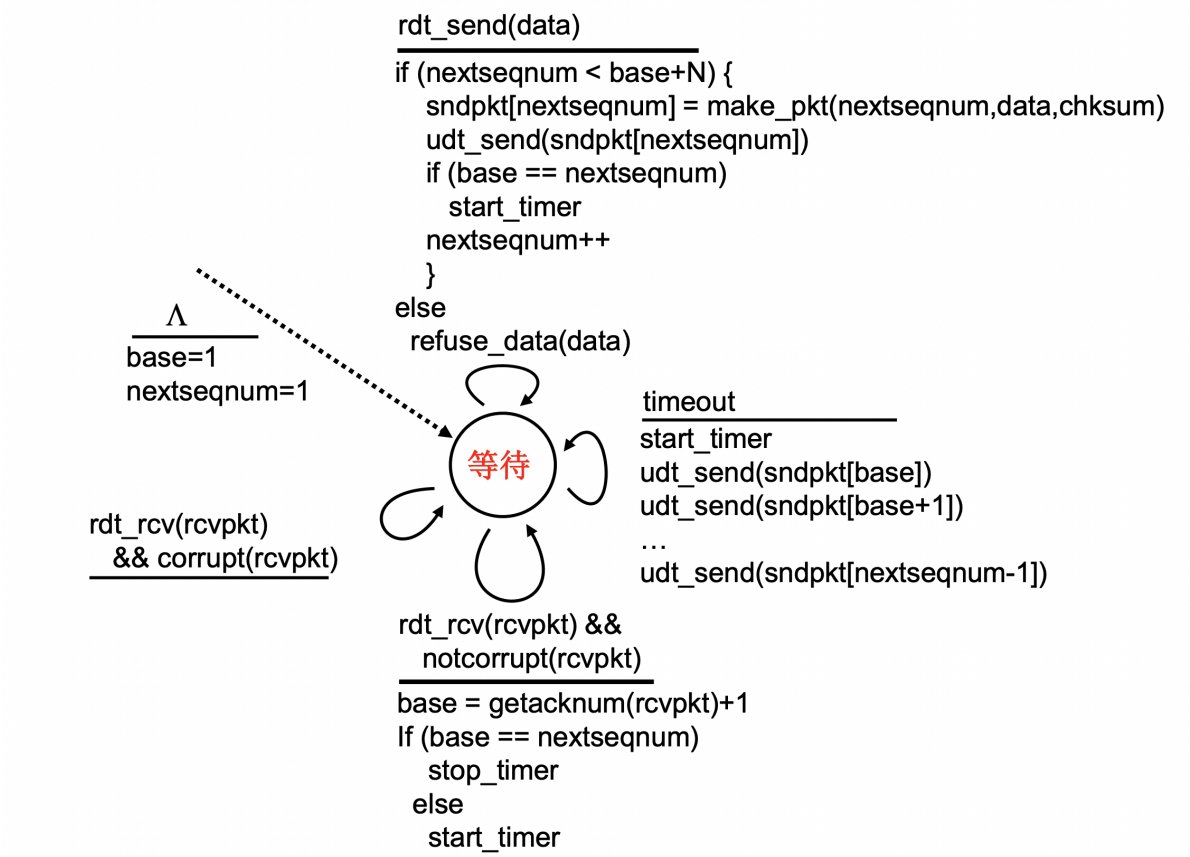
- **滑动窗口**



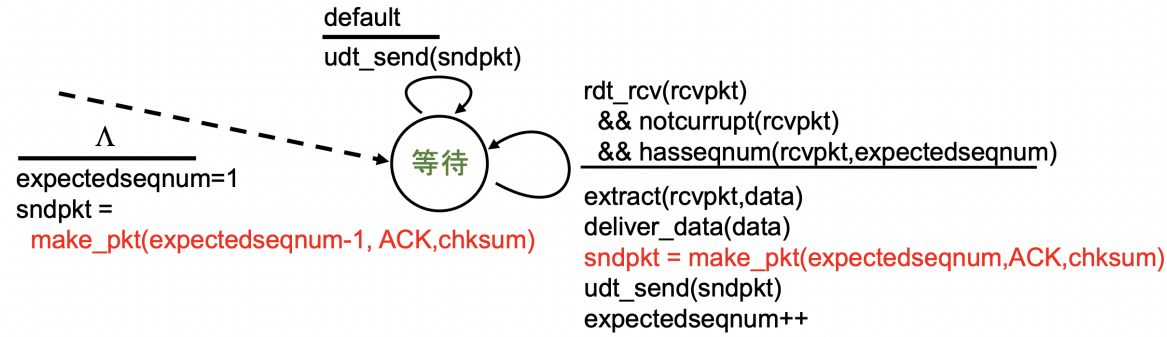
窗口分为左边界、发送边界和右边界，窗口大小固定。窗口左边界左侧为已经发送并得到确认的数据，左边界到发送边界的数据为已发送但未得到确认的数据，发送边界到右边界为等待发送的数据，右边界右侧为不可发送的数据。

发送端状态机

使用rdt3.0实现可靠数据传输



接收端状态机



代码实现(握手，挥手与上一个实验基本相同)

一些相同的代码不再展示

数据包

```
struct HEADER {
    unsigned int Seq;
    unsigned short datasize;
    unsigned short Flags;
    unsigned short Checksum;
    char content[MAXSIZE];
    HEADER() {
        this->Seq = 0;
        this->datasize = 0;
        this->Flags = 0;
        this->Checksum = 0;
        for (int i = 0; i < MAXSIZE; i++)
```

```
            this->content[i] = 0;
    }
}
```

发送端发包

```
void SEND()
{
    queue<pair<int, int>> time_list;//存入timer记录发送出去的时间点, order
    HEADER msg, result;

    cout << "请输入文件名: " << endl;
    string filename = "";
    cin >> filename;
    // 发送文件名
    sendto(ClientSocket,(char*)(filename.c_str()), filename.length(),0,
(SOCKADDR*)&ServerAddr,l);

    //读文件
    char** sendbuf = new char* [80000];
    for (int i = 0; i < 80000; i++)
        sendbuf[i] = new char[MAXSIZE];

    ifstream filein;
    filein.open(filename, ifstream::binary);
    cout << "文件是否打开: " << filein.is_open() << endl;
    while (!filein.is_open())
    {
        cout << "找不到该文件，请重新输入，或确认文件是否存在" << endl;
        cin >> filename;
        filein.open(filename, ifstream::binary);
        if (filein.is_open())
            cout << "文件已打开" << endl;
    }

    filein.seekg(0, filein.end);  //将文件流指针定位到流的末尾
    length = filein.tellg();
    cout << "文件长度: " << length << endl;

    filein.seekg(0, filein.beg);  //将文件流指针重新定位到流的开始

    int pagenum = 0;
    int len = length;
    while (len > 0)
    {
        filein.read(sendbuf[pagenum], min(len, MAXSIZE));
        len -= MAXSIZE;
        pagenum++;
    }


    filein.close();
    int save_pagenum = pagenum;
    clock_t start = clock();
```

```
    //未得到确认的第一个包
    int base = 0;
    //发送包的最后一个
    int tail = 0;
    while (pagenum > 0)
    {
        if (time_list.size() < NUM_WINDOW)
        {
            if (pagenum == 1)
                msg.set_EOF();
            msg.Seq = tail;
            msg.datasize = min(MAXSIZE, length);
            msg.clearcontent();
            msg.setcontent(sendbuf[tail], min(MAXSIZE, length));

            msg.Checksum = 0;
            msg.set_Checksum();
            //cout << msg.Checksum << endl;
            setsockopt(ClientSocket, SOL_SOCKET, SO_SNDBUF, (const
char*)&sendbuf, sizeof(char) * MAXSIZE);
            sendto(ClientSocket, (char*)&msg, sizeof(HEADER), 0,
(SOCKADDR*)&ServerAddr, sizeof(SOCKADDR));
            cout << "[Send]  " << msg.datasize << " bytes!" << " SEQ:" <<
int(msg.Seq) << " SUM:" << int(msg.Checksum) << endl;
            time_list.push(make_pair(clock(), msg.Seq));//开始计时，在末尾存入一个新
元素
            //cout << clock() << endl;
            tail++;
        }
        cout << "[WINDOW_AREA] :" << base << "to" << base + NUM_WINDOW - 1 <<
endl;
        int res = recvfrom(ClientSocket, (char*)&result, sizeof(HEADER), 0,
(SOCKADDR*)&ServerAddr, &l);
        //cout << " result'Ack: " << result.Ack << " " << endl;

        if (result.get_ACK() && !result.get_NAK() && result.Seq - 1 ==
time_list.front().second)
        {
            while (time_list.size() > 0 && result.Seq ==
time_list.front().second + 1)
            {
                cout << "[RECV]  SEQ: " << result.Seq << "   FLAG: ACK"   <<
endl;
                pagenum--;
                length -= MAXSIZE;

                base++;
                time_list.pop();
            }
        }
        else
        {
            if (clock() - time_list.front().first > MAX_TIME)
            {
                cout << "TIME OUT!" << endl;
```

```
                tail = base;
                while (time_list.size() != 0)
                    time_list.pop();
            }
        }
    }


    clock_t end = clock();

    stime = (end-start)/ CLOCKS_PER_SEC;
    cout << "文件发送完毕" << endl;
    cout << "共计用时" << stime << "ms" << endl;
    double throupt_rate = save_pagenum * sizeof(HEADER) / 1024 / stime * 1000;
    cout << "吞吐率为" << throupt_rate << "Mb/s" << endl;
    cout << "传输完毕" << endl;
    for (int i = 0; i < 65536; i++)
        delete[]sendbuf[i];
    delete[]sendbuf;
}
```

接收端收包

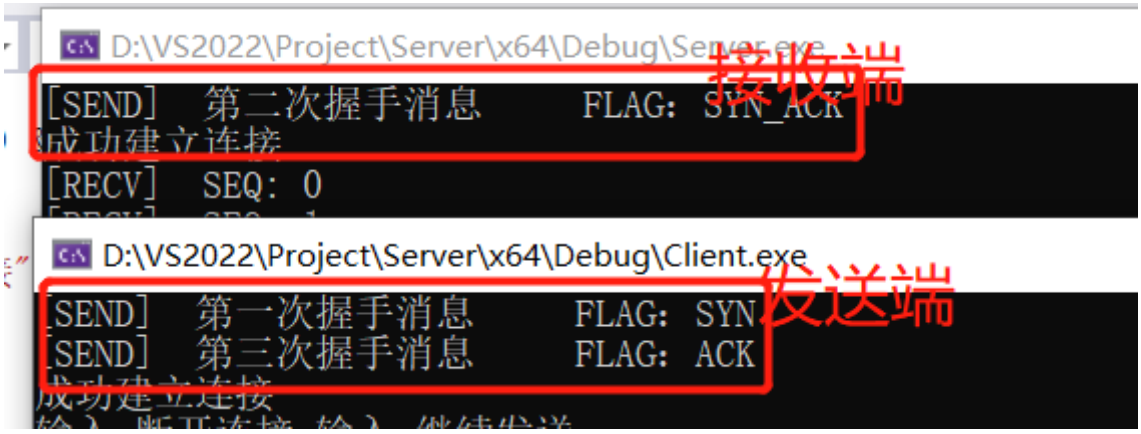
```
void RECV_FILE()
{
    //想要得到的包的序列号
    ack = 0;
    HEADER recv_msg, answer;
    //收文件名
    char* mes = new char[20];
    int length=recvfrom(ListenSocket, mes, 20, 0, (SOCKADDR*)&ServerAddr, &l);

    string filename;
    for (int i = 0; i < length; i++)
    {
        filename = filename + mes[i];
    }
    ofstream fileout;
    fileout.open(filename, ofstream::binary);

    //cvfrom(ListenSocket, mes, 4, 0, (SOCKADDR*)&ServerAddr, &l);
    //cout << mes << length << endl;
    int resize = recvfrom(ListenSocket, (char*)&recv_msg, sizeof(HEADER), 0,
(SOCKADDR*)&ServerAddr, &l);


    if (recv_msg.get_FIN() && recv_msg.get_ACK())
    {
        DisConnect();
        return;
    }
    if (recv_msg.check_Checksum())
    {
        fileout.write(recv_msg.content, recv_msg.datasize);
        ack = recv_msg.Seq + 1;
```

```
        cout << "[RECV]  SEQ: " << recv_msg.Seq << endl;

    }
    else
        answer.set_NAK();
    answer.set_ACK();
    answer.Seq = ack;

    sendto(ListenSocket, (char*)&answer, sizeof(HEADER), 0,
(SOCKADDR*)&ServerAddr, sizeof(SOCKADDR));

    while (!recv_msg.get_EOF())
    {

        recv_msg.clearcontent();

        int res = recvfrom(ListenSocket, (char*)&recv_msg, sizeof(HEADER), 0,
(SOCKADDR*)&ServerAddr, &l);
        while (res < 0)
        {
            sendto(ListenSocket, (char*)&answer, sizeof(HEADER), 0,
(SOCKADDR*)&ServerAddr, sizeof(SOCKADDR));
            res = recvfrom(ListenSocket, (char*)&recv_msg, sizeof(HEADER), 0,
(SOCKADDR*)&ServerAddr, &l);
        }
        if (recv_msg.Seq != ack)
        {
            sendto(ListenSocket, (char*)&answer, sizeof(HEADER), 0,
(SOCKADDR*)&ServerAddr, sizeof(SOCKADDR));
            cout << "[RECV] not want ,need to resend" << endl;
            continue;
        }

        if (recv_msg.get_FIN() && recv_msg.get_ACK())
        {
            DisConnect();
            return;
        }

        if (recv_msg.check_Checksum())
        {

            fileout.write(recv_msg.content, recv_msg.datasize);
            cout << "[RECV]  SEQ: " << recv_msg.Seq << endl;
            ack = recv_msg.Seq + 1;
        }
        else
        {
            answer.set_NAK();
            cout << "收到的包数据有误，WRONG CHECKNUM: " << recv_msg.Checksum <<
endl;
        }

        answer.set_ACK();
        answer.Seq = ack;
```

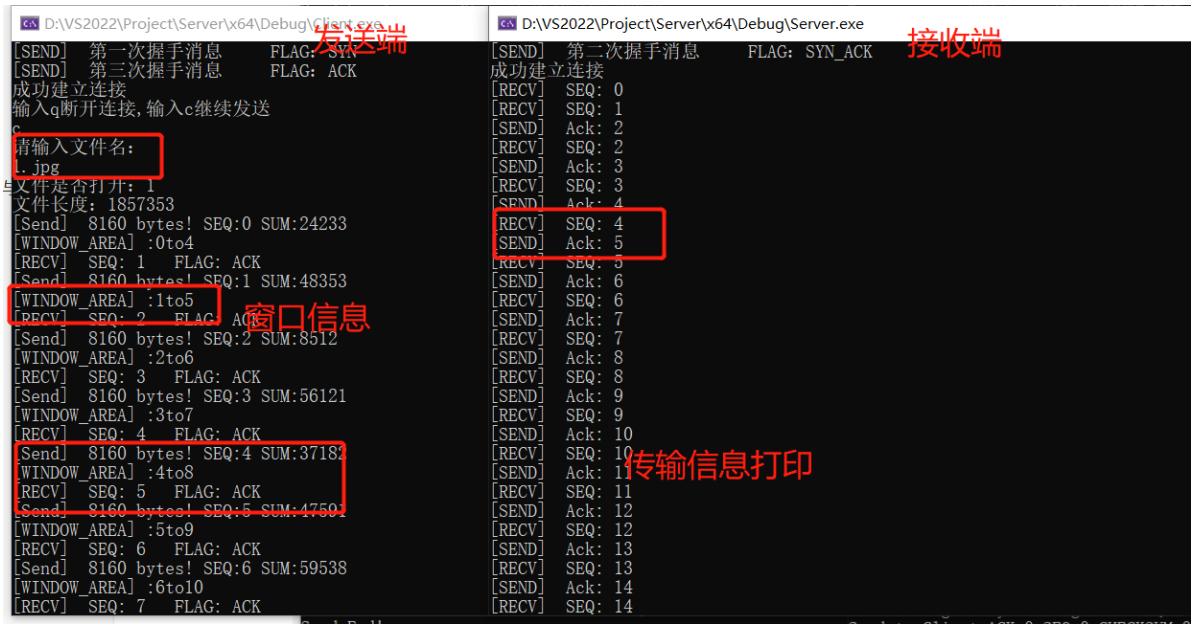
```
        sendto(ListenSocket, (char*)&answer, sizeof(HEADER), 0,
(SOCKADDR*)&ServerAddr, sizeof(SOCKADDR));
        cout << "[SEND]  Ack: " << answer.Seq << endl;
    }


    cout << "接收完毕" << endl;
    fileout.close();
}
```

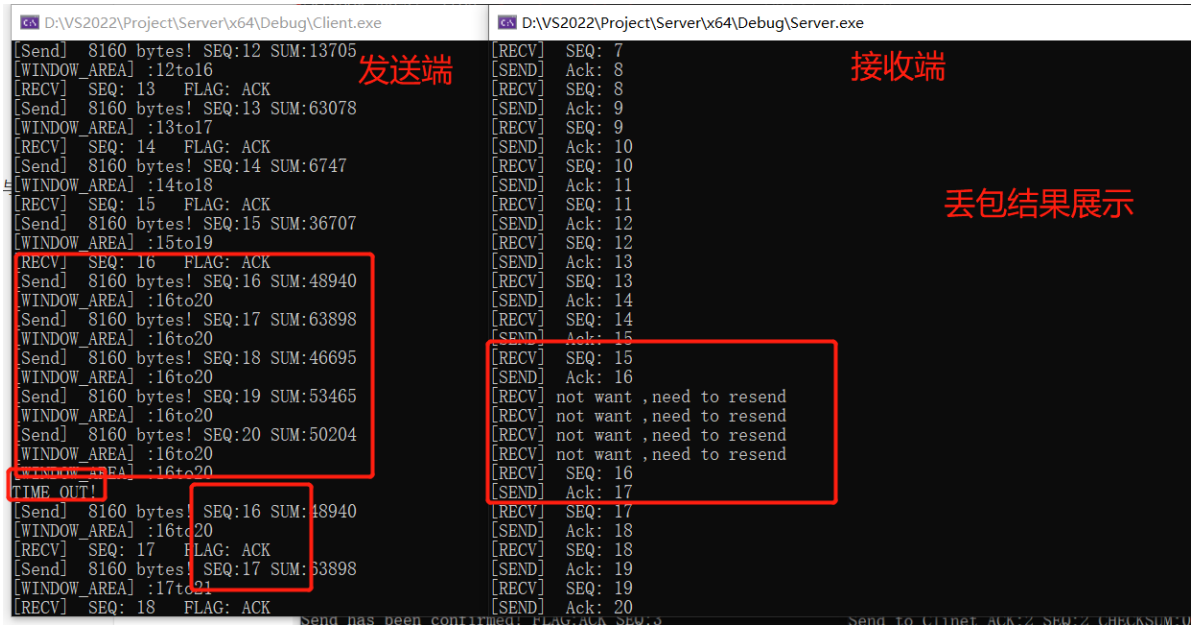
结果展示

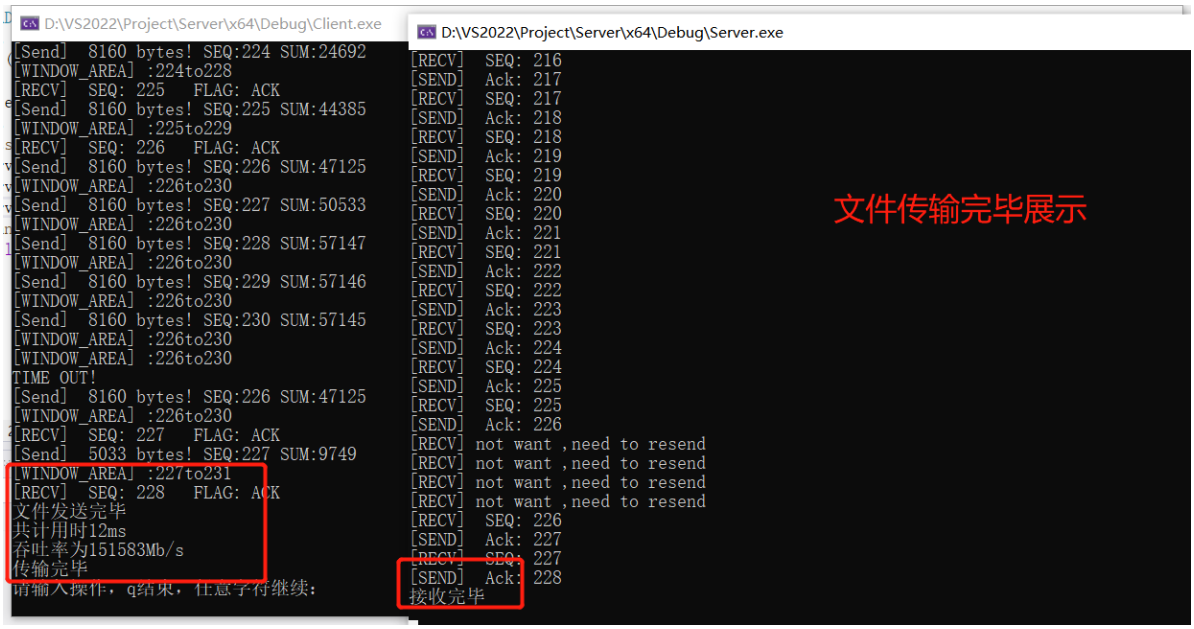
握手展示

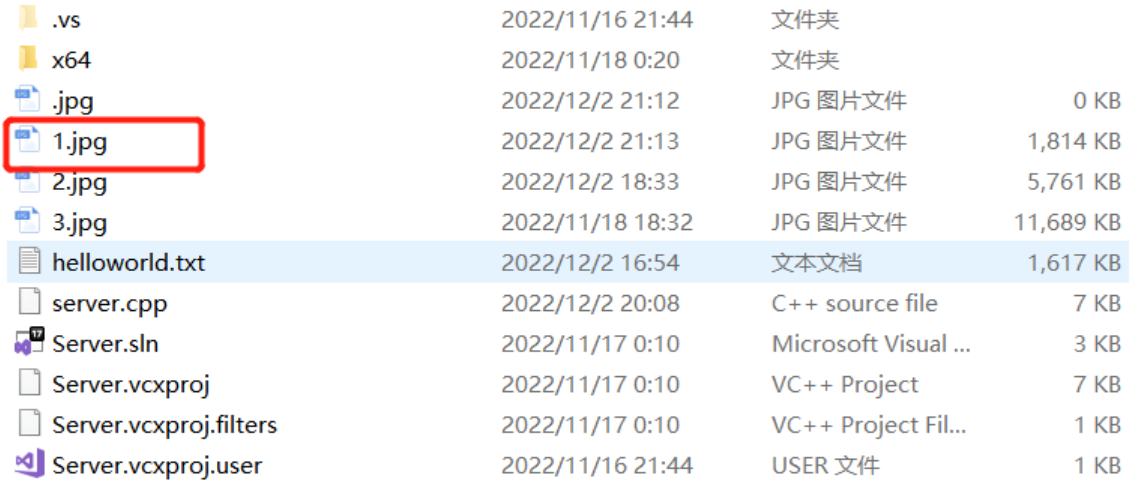


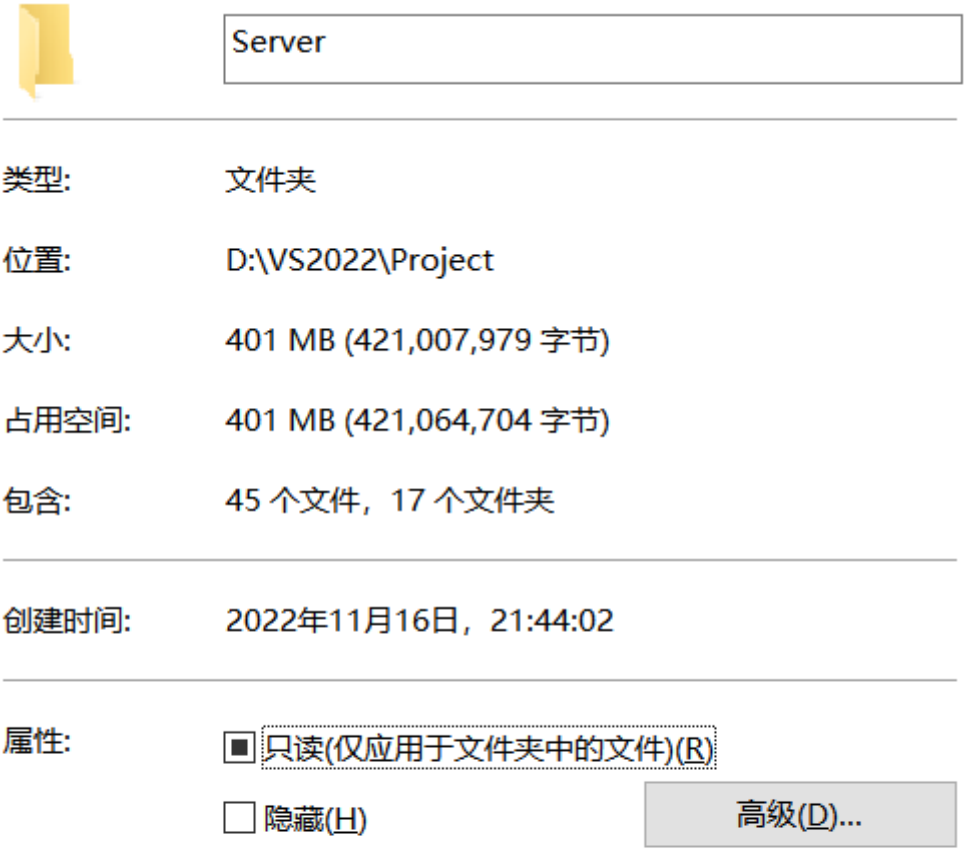
数据传输展示

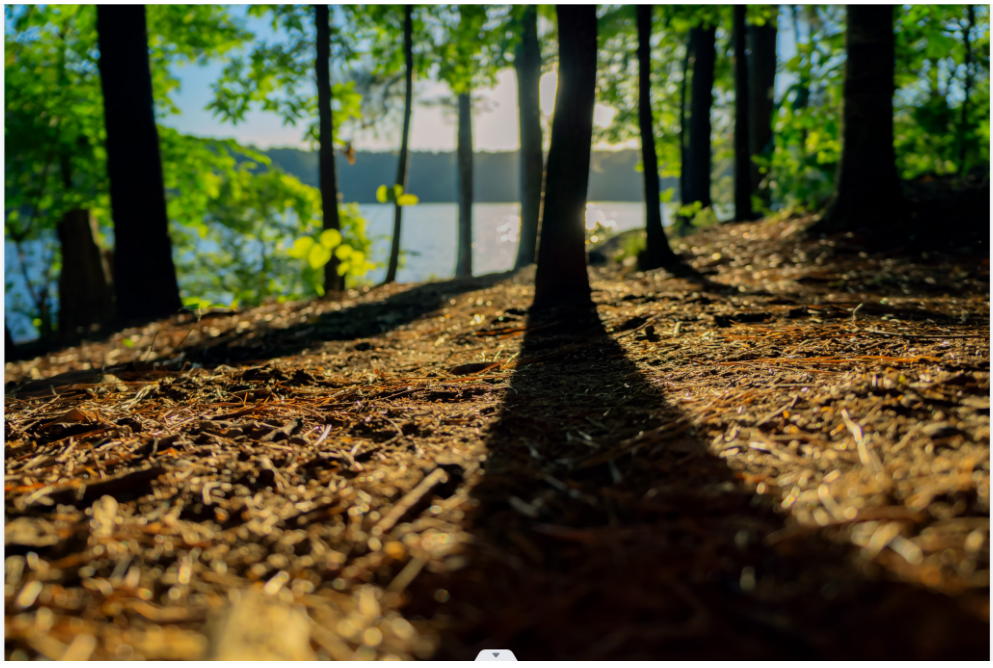



D:\VS2022\Project\Server\x64\Debug\Client.exe
[Send] 8160 bytes! SEQ:12 SUM:13705
[WINDOW_AREA] :12to16
[RECV] SEQ: 13 FLAG: ACK
[Send] 8160 bytes! SEQ:13 SUM:63078
[WINDOW_AREA] :13to17
[RECV] SEQ: 14 FLAG: ACK
[Send] 8160 bytes! SEQ:14 SUM:6747
[WINDOW_AREA] :14to18
[RECV] SEQ: 15 FLAG: ACK
[Send] 8160 bytes! SEQ:15 SUM:36707
[WINDOW_AREA] :15to19
[RECV] SEQ: 16 FLAG: ACK
[Send] 8160 bytes! SEQ:16 SUM:48940
[WINDOW_AREA] :16to20
[Send] 8160 bytes! SEQ:17 SUM:63898
[WINDOW_AREA] :16to20
[Send] 8160 bytes! SEQ:18 SUM:46695
[WINDOW_AREA] :16to20
[Send] 8160 bytes! SEQ:19 SUM:53465
[WINDOW_AREA] :16to20
[Send] 8160 bytes! SEQ:20 SUM:50204
[WINDOW_AREA] :16to20
TIME OUT!
[Send] 8160 bytes! SEQ:16 SUM:48940
[WINDOW_AREA] :16to20
[RECV] SEQ: 17 FLAG: ACK
[Send] 8160 bytes! SEQ:17 SUM:63898
[WINDOW_AREA] :17to21
[RECV] SEQ: 18 FLAG: ACK
发送端
D:\VS2022\Project\Server\x64\Debug\Server.exe
[RECV] SEQ: 7
[SEND] Ack: 8
[RECV] SEQ: 8
[SEND] Ack: 9
[RECV] SEQ: 9
[SEND] Ack: 10
[RECV] SEQ: 10
[SEND] Ack: 11
[RECV] SEQ: 11
[SEND] Ack: 12
[RECV] SEQ: 12
[SEND] Ack: 13
[RECV] SEQ: 13
[SEND] Ack: 14
[RECV] SEQ: 14
[SEND] Ack: 15
[RECV] SEQ: 15
[SEND] Ack: 16
[RECV] not want ,need to resend
[RECV] not want ,need to resend
[RECV] not want ,need to resend
[RECV] not want ,need to resend
[RECV] SEQ: 16
[SEND] Ack: 17
[RECV] SEQ: 17
[SEND] Ack: 18
[RECV] SEQ: 18
[SEND] Ack: 19
[RECV] SEQ: 19
[SEND] Ack: 20
接收端
丢包结果展示


D:\VS2022\Project\Server\x64\Debug\Client.exe
[Send] 8160 bytes! SEQ:224 SUM:24692
[WINDOW_AREA] :224to228
[RECV] SEQ: 225 FLAG: ACK
[Send] 8160 bytes! SEQ:225 SUM:44385
[WINDOW_AREA] :225to229
[RECV] SEQ: 226 FLAG: ACK
[Send] 8160 bytes! SEQ:226 SUM:47125
[WINDOW_AREA] :226to230
[Send] 8160 bytes! SEQ:227 SUM:50533
[WINDOW_AREA] :226to230
[Send] 8160 bytes! SEQ:228 SUM:57147
[WINDOW_AREA] :226to230
[Send] 8160 bytes! SEQ:229 SUM:57146
[WINDOW_AREA] :226to230
[Send] 8160 bytes! SEQ:230 SUM:57145
[WINDOW_AREA] :226to230
[WINDOW_AREA] :226to230
TIME OUT!
[Send] 8160 bytes! SEQ:226 SUM:47125
[WINDOW_AREA] :226to230
[RECV] SEQ: 227 FLAG: ACK
[Send] 5033 bytes! SEQ:227 SUM:9749
[WINDOW_AREA] :227to231
[RECV] SEQ: 228 FLAG: ACK
文件发送完毕
共计用时12ms
吞吐率为151583Mb/s
传输完毕
请输入操作，q结束，任意字符继续：
D:\VS2022\Project\Server\x64\Debug\Server.exe
[RECV] SEQ: 216
[SEND] Ack: 217
[RECV] SEQ: 217
[SEND] Ack: 218
[RECV] SEQ: 218
[SEND] Ack: 219
[RECV] SEQ: 219
[SEND] Ack: 220
[RECV] SEQ: 220
[SEND] Ack: 221
[RECV] SEQ: 221
[SEND] Ack: 222
[RECV] SEQ: 222
[SEND] Ack: 223
[RECV] SEQ: 223
[SEND] Ack: 224
[RECV] SEQ: 224
[SEND] Ack: 225
[RECV] SEQ: 225
[SEND] Ack: 226
[RECV] not want ,need to resend
[RECV] not want ,need to resend
[RECV] not want ,need to resend
[RECV] not want ,need to resend
[RECV] SEQ: 226
[SEND] Ack: 227
[RECV] SEQ: 227
[SEND] Ack: 228
接收完毕
文件传输完毕展示

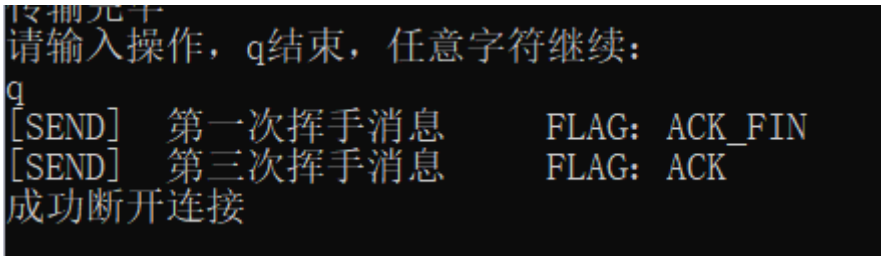

.vs 2022/11/16 21:44 文件夹
x64 2022/11/18 0:20 文件夹
.jpg 2022/12/2 21:12 JPG 图片文件 0 KB
1.jpg 2022/12/2 21:13 JPG 图片文件 1,814 KB
2.jpg 2022/12/2 18:33 JPG 图片文件 5,761 KB
3.jpg 2022/11/18 18:32 JPG 图片文件 11,689 KB
helloworld.txt 2022/12/2 16:54 文本文档 1,617 KB
server.cpp 2022/12/2 20:08 C++ source file 7 KB
Server.sln 2022/11/17 0:10 Microsoft Visual ... 3 KB
Server.vcxproj 2022/11/17 0:10 VC++ Project 7 KB
Server.vcxproj.filters 2022/11/17 0:10 VC++ Project Fil... 1 KB
Server.vcxproj.user 2022/11/16 21:44 USER 文件 1 KB





最终可以得到正常数据，文件大小和原来的都一样。

挥手展示



可以看到实现了GBN滑动窗口的传输