计算机网络课程实验报告

郭裕彬 2114052 物联网工程

实验3-2：基于UDP服务设计可靠传输协议并编程实现

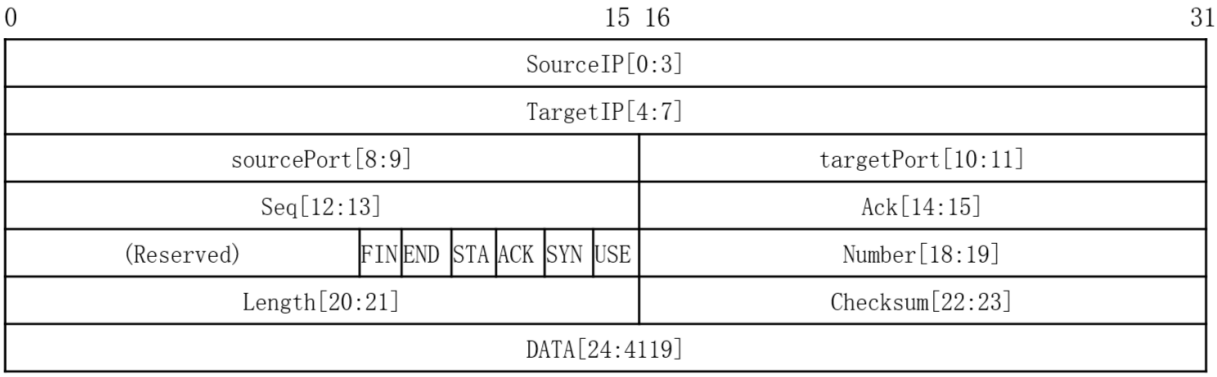
发送端使用地址：127.0.0.1:9995
接收端使用地址：127.0.0.1:9997
路由器使用地址：127.0.0.1:9996

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，发送窗口和接收窗口采用相同大小，支持累积确认，完成给定测试文件的传输。
- 协议设计：数据包格式，发送端和接收端交互，详细完整
- 流水线协议：多个序列号
- 发送缓冲区、接收缓冲区
- 累积确认：Go Back N
- 日志输出：收到/发送数据包的序号、ACK、校验和等，发送端和接收端的窗口大小等情况，传输时间与吞吐率
- 测试文件：必须使用助教发的测试文件（1.jpg、2.jpg、3.jpg、helloworld.txt）

协议设计

整体结构



协议报文头部共24字节：

- SourceIP：32位源IP地址
- TargetIP：32位目的IP地址
- SourcePort：16位源端口号
- TargetPort：16位目的端口号
- Seq：16位序列号，取值为0或1
- Ack：16位确认序列号，取值为0或1。（本实验握手部分的ACK包的Ack值设置为对应包的Seq+1；其他部分的ACK包的Ack值都为对应包的Seq）
- FLAGS
    - USE：表示该报文有效
    - SYN：表示为连接请求建立报文
    - ACK：表示为确认报文
    - STA：表示为文件传输开始报文
    - END：表示为文件传输结束报文
    - FIN：表示为连接请求断开报文
- Number：16位数据，用于在STA报文中记录文件分包总数
- Length：16位数据，用于标识该包的数据段长度
- Checksum：16位校验和

具体代码实现同实验3-1，此处不再叙述。

双端交互

- 发送端发起连接请求，与接收端实现类似TCP的三次握手，发送端进入等待发送状态，接收端进入等待接收状态
- 发送端对文件进行处理，封装STA包放入缓冲区包队列第一个位置，之后依次放入读取并封装好的数据包
- 发送端的发送线程从队列中依次取出包并发送直至发送窗口满
- 接收端一直监听，收到第一个STA数据包时进入接收状态，循环接收数据包：收到有效且是期望的数据包时放入接收缓冲区并返回对该包的ACK确认报文；收到不完整或序列号大于期望序列号的包则重发对前一个包的ACK；收到序列号小于期望序列号的包则忽略
- 发送端的接收线程不断监听接收端返回的ACK包，当收到的ACK包的确认序列号在窗口之间时更新标识窗口左侧的变量，使得窗口右滑；发送线程感知到窗口右滑后，取出新的数据包并发送，填满发送窗口；当定时器时间内窗口没有更新时依次重发窗口内所有数据包
- 循环上述两个步骤直至发送端发送队列中的最后一个数据包END包
- 接收端收到END包后返回ACK，进入等待接收状态，等待继续接收文件或断开连接；发送端收到END包的ACK后，进入等待发送状态，等待用户选择继续发送文件或断开连接
- 用户选择断开连接，发送端发起断开连接请求，与接收端实现类似TCP的四次挥手，完成后双端关闭

握手挥手

- 类似TCP的三次握手：发送端发起一次握手，接收二次握手，发送三次握手；接收端接收一次握手，发送二次握手，接收三次握手建立连接，使用超时重传机制。
- 类似TCP的四次挥手：发送端发起一次挥手，接收二次挥手，接收三次挥手，发送四次挥手；接收端接收一次挥手，发送二次、三次挥手，接收四次挥手，使用超时重传机制。
- 具体代码实现除了将序列号从0/1更新为逐步增大的唯一序列号外与实验3-1的代码一致，此处不再叙述。

差错校验

与实验3-1的实现一致，不再叙述

滑动窗口&超时重传

定义四个全局变量`windowSize`、`base`、`lastWritten`、`nextSeqNum`分别记录窗口大小、窗口左侧最大的已收到ACK的包的位置、缓冲区队列中最后一个包的位置和下一个待发送的包的位置。

发送线程负责初始化发送满窗口的数据包、在窗口滑动时发送新的数据包以及检测超时时重发窗口内所有数据包。

```
int windowSize = 1;//窗口大小
int base = -1;//窗口左侧最大已确认的包位置
int lastWritten = 0;//最后的包位置
int nextSeqNum = 0;//下一个待发送的包位置

//发送线程
DWORD WINAPI sendHandle(LPVOID lparam){
    time_t start = clock();
    time_t end;
    int p;//用于之后判断是否有收到新的ACK
    while(base<=lastWritten){
        //发送窗口满了，等待接收方返回ACK告知可以更新窗口
        if(base + windowSize == nextSeqNum){
            Sleep(200);
        }
        p = base;//获得窗口最左侧的位置
        for (int i =
0;nextSeqNum<=base+windowSize&&nextSeqNum<=lastWritten;nextSeqNum++
,i++){
            if(i == 0){//每次发送第一个包的时候重置计时器
                start = clock();
            }
            //依次发送窗口内的所有数据包
```

```
            sendto(client,
(char*)&sendList[nextSeqNum],sizeof(sendList[nextSeqNum]),0,
(SOCKADDR*)&targetAd,sizeof(targetAd));
            cout<<"[Send]发送第"<<nextSeqNum<<"个数据包 ";
            sendList[nextSeqNum].printInfo();
        }
        //如果收到最后一个包的ACK，退出发送
        if(base == lastWritten)return 1;
        //如果base没有更新
        if(p == base){
            //说明一直未收到新的ACK
            //检查是否超时，超时则重发
            end = clock();
            if((double)(end - start) / CLOCKS_PER_SEC >= 1){
                cout<<"[Info]尝试重传......"<<endl;
                nextSeqNum = p+1;
            }
        }
    }
    return 1;
}
```

具体实现为，每一次循环开始时线程都发送[nextSeqNum,(base+windowSize)||(lastWritten)]范围内的所有数据包，有以下几种情况

- 窗口为空，即最初发送开始或一次性收到原窗口所有ACK或超时使得nextSeqNum退回到窗口左侧时，nextSeqNum=base+1，因此会发送windowSize个新的顺位数据包；
- 窗口非空，接收线程收到ACK包更新了base，使其右滑了n个位置(n<windowSize)，那么base+windowSize也右移了n，会发送新的n个数据包，nextSeqNum增大n；
- 窗口右侧接触到lastWritten，即发送窗口触碰到了发送队列末尾，窗口右侧不再滑动，只会根据接收线程更新base，不发送新的包

该过程完成后，判断窗口左侧base在发送前后是否发生了变化，未变化则检测是否超时，是则将nextSeqNum设置为base+1，意味着在下一次循环中线程将重发窗口内所有数据包。值得一提的是，在发送过程中设置了一个判断语句，每次经过这个发送过程时，如果有发送新的包，就在第一个新的包发送的同时更新计时器，这是因为发送新的包就代表窗口发生了滑

动（或开始了一次重发），应当重新开始计时以准备超时重传。

累积确认

- 接收端

  接收端循环接收数据包，分别对接收到的包大于、等于和小于期望序列号的三种情况进行处理

```
while(1){
    UDP_packet receive,send;
    send.setFlags(FLAG_USE);
    send.setFlags(FLAG_ACK);
    send.setSourceIP(sourceAd);
    send.setTargetIP(targetAd);
    int len = sizeof(SOCKADDR);
    recvfrom(client,(char*)&receive,sizeof(receive),0,
(SOCKADDR*)&t,&len);
    if(receive.checkChecksum()&&receive.flags&FLAG_USE){
        if(receive.Seq ==
exceptedSeqNum&&receive.checkChecksum()){
            //收到的是期望的包，接收并返回ACK
            RecvList[exceptedSeqNum]=receive;
            exceptedSeqNum++;
            send.setAck(receive.Seq);
            send.calChecksum();
            sendto(client,(char*)&send,sizeof(send),0,
(SOCKADDR*)&t,sizeof(SOCKADDR));
            cout<<"[Recv] ";
            receive.printInfo();
            cout<<"[Info] ";
            printWindowInfo();
            cout<<"[Send] ";
            send.printInfo();
        }
        else
if(receive.Seq>exceptedSeqNum||!receive.checkChecksum()){
            //收到的是大于期望值的包，说明收到乱序包/校验和不通过，包
有丢失，重发之前的ACK
```

```
            send.setAck(exceptedSeqNum-1);
            send.calChecksum();
            sendto(client,(char*)&send,sizeof(send),0,
(SOCKADDR*)&t,sizeof(SOCKADDR));
            cout<<"[Info]收到乱序的数据包"<<receive.Seq<<endl;
            cout<<"[Resend] ";
            send.printInfo();
            continue;
        }
        else if (receive.Seq<exceptedSeqNum){
            cout<<"[Info]收到重复的数据包"<<receive.Seq<<endl;
        }
        if(receive.flags&FLAG_END){
            length = receive.length;
            cout<<"[Info]该数据包为最后一个包"<<endl;
            cout<<"[Info]--------文件接受完成--------"<<endl;
            break;
        }
    }
}
```

- 发送端

  发送端使用接收线程循环接收从接收端发回的ACK包，只要返回的ACK的确认序列号大于当前发送窗口左端的base，就更新base为ACK中的确认序列号值，同时打印一些信息

```
//接收线程
DWORD WINAPI receiveHandle(LPVOID lparam){
    while(base < lastWritten){
        UDP_packet receive;
        int len=sizeof(targetAd);
        recvfrom(client,(char*)&receive,sizeof(receive),0,
(SOCKADDR*)&targetAd,&len);

 if(receive.checkChecksum()&&receive.flags&FLAG_USE&&receive
.flags&FLAG_ACK){
            if(receive.Ack>=base){
                //收到确认ACK，窗口右移
                base = receive.Ack;
```

```
                cout<<"[Recv] ";
                receive.printInfo();
                cout<<"[Info] ";
                printWindowInfo();
            }
        }
    }
    return 1;
}
```

单向传输

在实验3-1`sendFile`的基础上进行修改，使得读取文件内容到缓冲区之后将其打包封装成能够发出的数据包放入缓冲区包队列，而不是直接发送；之后再创建发送和接收线程进行数据包传输，线程返回后判断是否传输完成，同时打印相关信息。

```
void sendFile(SOCKADDR_IN s,SOCKADDR_IN t,char* filePath,SOCKET
&client){
    //读取并发送文件的部分
    //···some code···
    //读取文件到缓冲区并记录总共的包数和最后一个包的数据长度
    //···some code···
    lastWritten = amount;
    UDP_packet send,receive;
    //发送STA包，传递文件路径、包总数
    send.setNumber(amount);
    send.setSourceIP(s);send.setTargetIP(t);
    send.setFlags(FLAG_USE);send.setFlags(FLAG_STA);
    send.setLength(strlen(filePath));
    memcpy(send.data,filePath,strlen(filePath));
    send.calChecksum();
    sendList[0]=send;
    //把带有数据的包存入发送缓冲队列
    for(int i=1;i<=amount+1;i++){
        UDP_packet pk;
        pk.setSourceIP(s);pk.setTargetIP(t);
        pk.setNumber(amount);
        pk.setFlags(FLAG_USE);
```

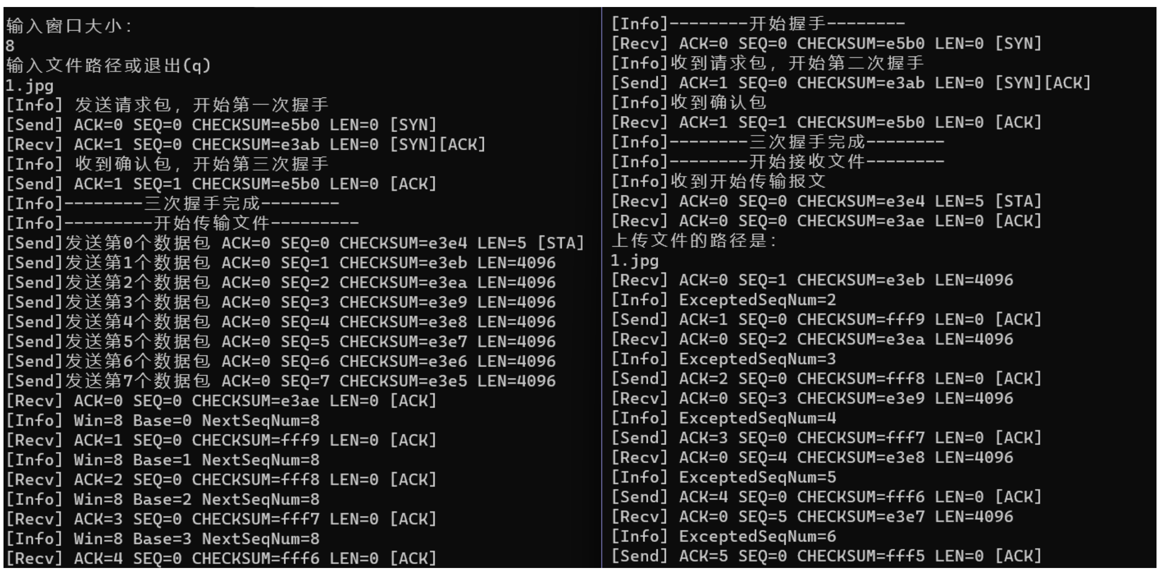
```
        pk.setSeq(i);
        if(i==amount){
            //最后一个数据包
            pk.setFlags(FLAG_END);
            pk.setLength(p);
            for(int j=0;j<p;j++)
            {
                pk.data[j]=sendbuf[i][j];
            }
        }
        else{
            pk.setLength(MAX_SIZE);
            for(int j=0;j<MAX_SIZE;j++)
            {
                pk.data[j]=sendbuf[i][j];
            }
        }
        pk.calChecksum();
        sendList[i]=pk;
    }

    time_t startsend=clock();
    time_t endsend;
    cout<<"[Info]---------开始传输文件---------"<<endl;
    if(base<lastWritten){
        HANDLE recvThread =
CreateThread(NULL,NULL,receiveHandle,LPVOID(),0,NULL);
        HANDLE sendThread =
CreateThread(NULL,NULL,sendHandle,LPVOID(),0,NULL);
        WaitForSingleObject(recvThread,INFINITE);
        WaitForSingleObject(sendThread,INFINITE);
        if(base == lastWritten){
            cout<<"[Info]---------传输完成---------"<<endl;
            endsend = clock();
        }
    }
    //输出本次传输的相关数据
    long fileSize=(long)(amount*MAX_SIZE+p);
    double time=(double)(endsend-startsend)/CLOCKS_PER_SEC;
```

```
    double rate=(double)(fileSize*8.0)/(time*1000);
    printf("[Info]发送时间：%lfs\n总大小：%dBytes\n吞吐率：%lfKbps\n",time,fileSize,rate);
    cout<<"[Info]---------文件发送成功---------"<<endl;
    return ;
}
```

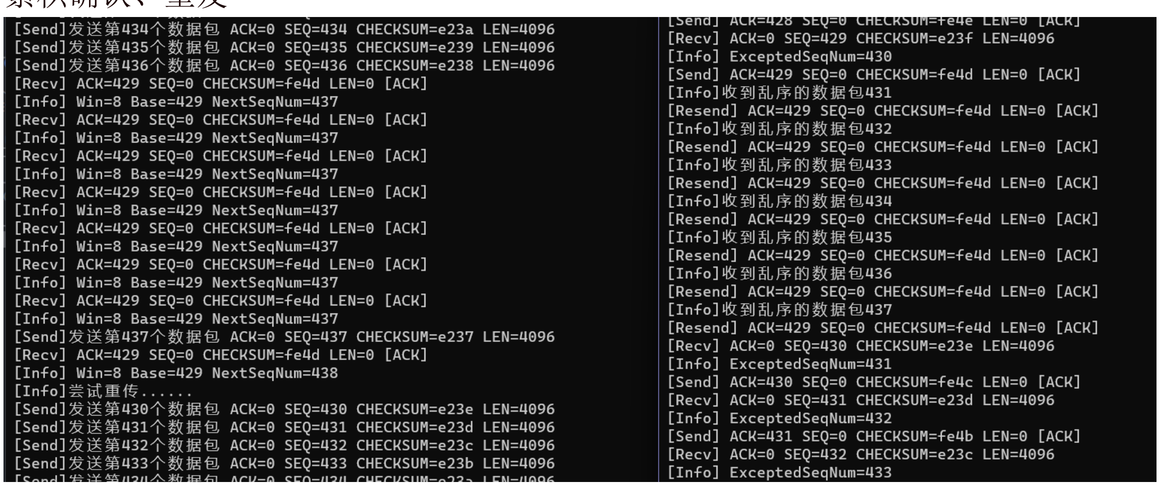
实验结果

运行截图

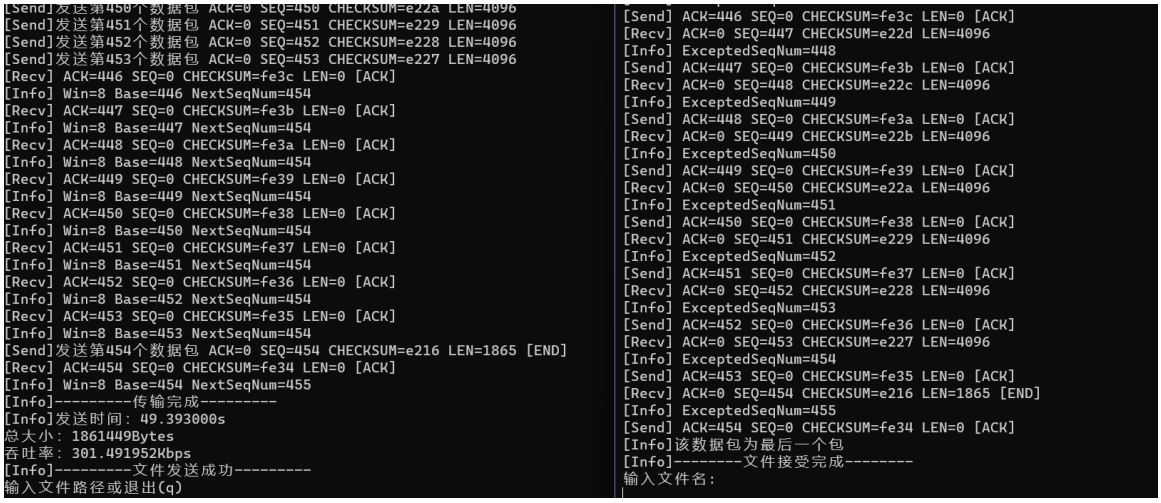
- 握手



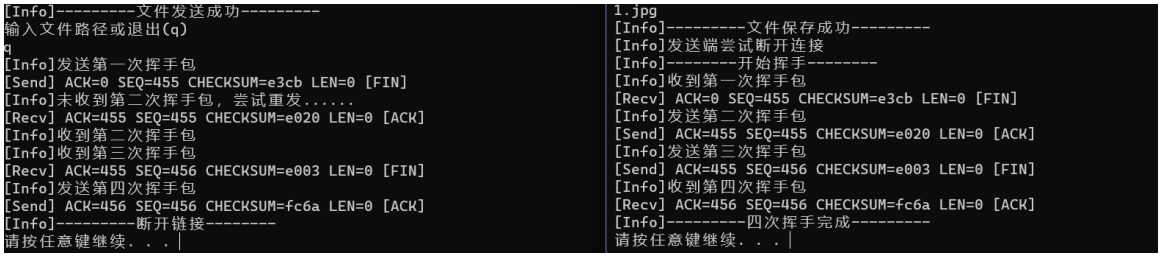
- 累积确认、重发



- 传输结束



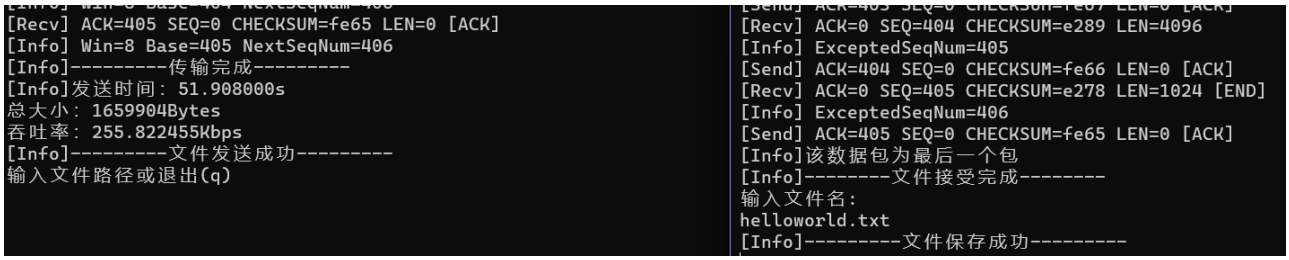
- 挥手



测试结果

与讲解实验时设置一致，路由器设置为丢包率3%，延迟3ms，窗口大小设置为8：

	文件大小/BYTES	发送时间/S	吞吐率
helloworld.txt	1659904	51.908	255.822kbps
1.jpg	1861449	60.186	247.426kbps
2.jpg	5902601	186.428	253.292kbps
3.jpg	11973090	377.299	253.870kbps



```
[Recv] ACK=451 SEQ=0 CHECKSUM=fe37 LEN=0 [ACK]
[Info] Win=8 Base=451 NextSeqNum=453
[Recv] ACK=452 SEQ=0 CHECKSUM=fe36 LEN=0 [ACK]
[Info] Win=8 Base=452 NextSeqNum=453
[Send]发送第453个数据包 ACK=0 SEQ=453 CHECKSUM=e227
[Send]发送第454个数据包 ACK=0 SEQ=454 CHECKSUM=e216
[Recv] ACK=453 SEQ=0 CHECKSUM=fe35 LEN=0 [ACK]
[Info] Win=8 Base=453 NextSeqNum=455
[Info]尝试重传......
[Send]发送第454个数据包 ACK=0 SEQ=454 CHECKSUM=e216
[Recv] ACK=454 SEQ=0 CHECKSUM=fe34 LEN=0 [ACK]
[Info] Win=8 Base=454 NextSeqNum=455
[Info]---------传输完成---------
[Info]发送时间: 60.186000s
总大小: 1861449Bytes
吞吐率: 247.426179Kbps
[Info]---------文件发送成功---------
输入文件路径或退出(q)
```

```
[Recv] ACK=0 SEQ=451 CHECKSUM=e229 LEN=4096
[Info] ExceptedSeqNum=452
[Send] ACK=451 SEQ=0 CHECKSUM=fe37 LEN=0 [ACK]
[Recv] ACK=0 SEQ=452 CHECKSUM=e228 LEN=4096
[Info] ExceptedSeqNum=453
[Send] ACK=452 SEQ=0 CHECKSUM=fe36 LEN=0 [ACK]
[Recv] ACK=0 SEQ=453 CHECKSUM=e227 LEN=4096
[Info] ExceptedSeqNum=454
[Send] ACK=453 SEQ=0 CHECKSUM=fe35 LEN=0 [ACK]
[Recv] ACK=0 SEQ=454 CHECKSUM=e216 LEN=1865 [END]
[Info] ExceptedSeqNum=455
[Send] ACK=454 SEQ=0 CHECKSUM=fe34 LEN=0 [ACK]
[Info]该数据包为最后一个包
[Info]--------文件接受完成--------
输入文件名:
1.jpg
[Info]---------文件保存成功---------
```

```
[Recv] ACK=1439 SEQ=0 CHECKSUM=fa5b LEN=0 [ACK]
[Info] Win=8 Base=1439 NextSeqNum=1442
[Recv] ACK=1440 SEQ=0 CHECKSUM=fa5a LEN=0 [ACK]
[Info] Win=8 Base=1440 NextSeqNum=1442
[Recv] ACK=1441 SEQ=0 CHECKSUM=fa59 LEN=0 [ACK]
[Info] Win=8 Base=1441 NextSeqNum=1442
[Info]---------传输完成---------
[Info]发送时间: 186.428000s
总大小: 5902601Bytes
吞吐率: 253.292467Kbps
[Info]---------文件发送成功---------
输入文件路径或退出(q)
```

```
[Recv] ACK=0 SEQ=1440 CHECKSUM=da71 LEN=4096
[Info] ExceptedSeqNum=1441
[Send] ACK=1440 SEQ=0 CHECKSUM=fa5a LEN=0 [ACK]
[Recv] ACK=0 SEQ=1441 CHECKSUM=da60 LEN=265 [END]
[Info] ExceptedSeqNum=1442
[Send] ACK=1441 SEQ=0 CHECKSUM=fa59 LEN=0 [ACK]
[Info]该数据包为最后一个包
[Info]--------文件接受完成--------
输入文件名:
2.JPG
[Info]---------文件保存成功---------
```

```
[Recv] ACK=2921 SEQ=0 CHECKSUM=f491 LEN=0 [ACK]
[Info] Win=8 Base=2921 NextSeqNum=2924
[Recv] ACK=2922 SEQ=0 CHECKSUM=f490 LEN=0 [ACK]
[Info] Win=8 Base=2922 NextSeqNum=2924
[Recv] ACK=2923 SEQ=0 CHECKSUM=f48f LEN=0 [ACK]
[Info] Win=8 Base=2923 NextSeqNum=2924
[Info]---------传输完成---------
[Info]发送时间: 377.299000s
总大小: 11973090Bytes
吞吐率: 253.869531Kbps
[Info]---------文件发送成功---------
输入文件路径或退出(q)
```

```
[Recv] ACK=0 SEQ=2922 CHECKSUM=cedd LEN=4096
[Info] ExceptedSeqNum=2923
[Send] ACK=2922 SEQ=0 CHECKSUM=f490 LEN=0 [ACK]
[Recv] ACK=0 SEQ=2923 CHECKSUM=cecc LEN=482 [END]
[Info] ExceptedSeqNum=2924
[Send] ACK=2923 SEQ=0 CHECKSUM=f48f LEN=0 [ACK]
[Info]该数据包为最后一个包
[Info]--------文件接受完成--------
输入文件名:
3.JPG
[Info]---------文件保存成功---------
```