

# 计算机网络实验

## 实验3：基于UDP服务设计可靠传输协议并编程实现

### 实验3-2

姓名：蒋浩南      学号：2012948

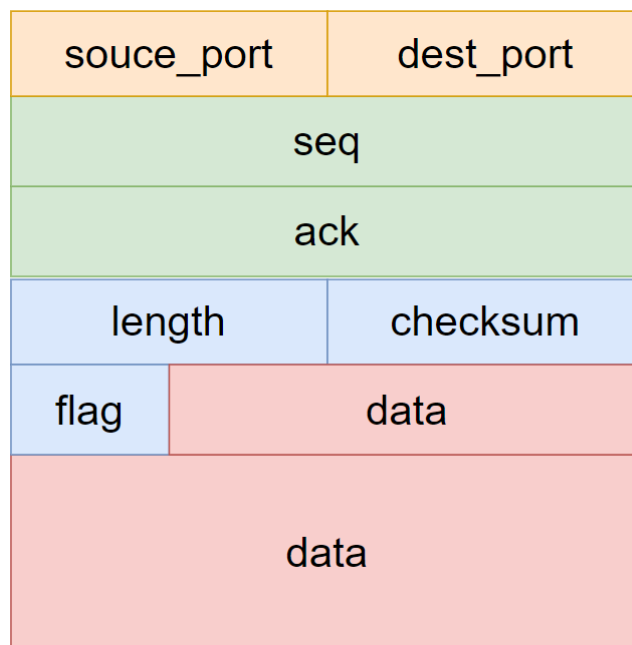
- 一、实验要求
- 二、协议设计
  - (一) 报文结构
  - (二) 校验和计算和验证
    - (1)伪首部
    - (2)计算校验和
    - (3)验证校验和
  - (三) 三次握手
  - (四) 四次挥手
  - (五) Go Back N和累计确认
- 三、GBN代码实现
  - (一) 发送端
  - (二) 接收端
- 四、程序演示
- 五、代码库

## 一、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。
- 多个序列号；
- 发送缓冲区、接受缓冲区；
- 滑动窗口：Go Back N；
- 有必要日志输出（须显示传输过程中发送端、接收端的窗口具体情况）。

## 二、协议设计

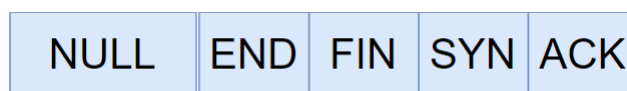
### (一) 报文结构



1. 对于报文的设计，含有2字节的源端口，2字节的目标端口，4字节的seq，4字节的ack，2字节的长度，2字节的校验和，1字节的标志，若干字节数据。

长度为传输报文时，记录当前报文所传输数据的有效字节数。

2. 对于标志位：

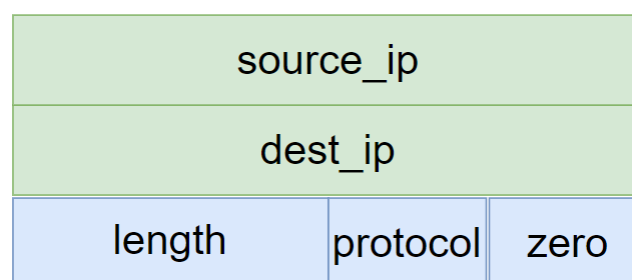


从低到高分别为ACK，SYN，FIN和END。其中END为在传输文件结束后发送报文的标志位。

## (二) 校验和计算和验证

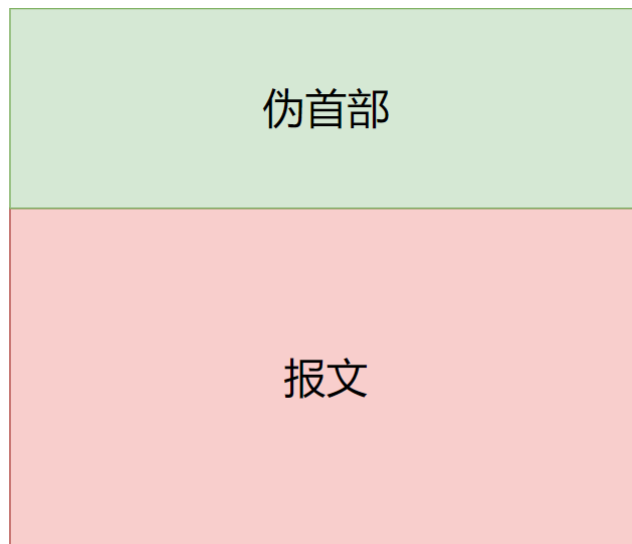
### (1) 伪首部

首先生成伪首部，伪首部结构如下：



分别为源ip，目标ip，长度，版本号和填充0。

### (2) 计算校验和



```
void setChecksum(msg* message, pseudoHead* ph) {  
    //设为0  
    message->checksum = 0;  
    int sum = 0;  
    int len_pseudo = sizeof(pseudoHead);  
    int len_msg = sizeof(msg);  
    for (int i = 0; i < len_pseudo / 2; i++) {  
        sum += ((WORD*)ph)[i];  
    }  
    for (int i = 0; i < len_msg / 2; i++) {  
        sum += ((WORD*)message)[i];  
    }  
    while (sum >> 16) {  
        sum = (sum & 0xffff) + (sum >> 16);  
    }  
    message->checksum = ~sum;  
};
```

设置校验和的时候，计算伪首部和报文的16位和，取反。

### (3)验证校验和

```
bool verfiyChecksum(msg* message, pseudoHead* ph) {  
  
    int sum = 0;  
    int len_pseudo = sizeof(pseudoHead);  
    int len_msg = sizeof(msg);  
    for (int i = 0; i < len_pseudo / 2; i++) {  
        sum += ((WORD*)ph)[i];  
    }  
    for (int i = 0; i < len_msg / 2; i++) {  
        sum += ((WORD*)message)[i];  
    }  
    while (sum >> 16) {  
        sum = (sum & 0xffff) + (sum >> 16);  
    }  
    return sum == 0xffff;  
};
```

验证校验和，将生成伪首部，计算伪首部和接收到的报文的16位和，如结果为0xffff，则验证正确。

### (三) 三次握手

对于三次握手

(1) 客户端:

1. 发送同步报文，标记位为SYN,seq=0,ack=0;
2. 开始计时，接收服务器报文，若超时则重传同步报文。
3. 判断接收的报文是否为：标志位（SYN，ACK），seq=0，ack=1。若是发送报文：标志位（ACK），seq=1，ack=1。否则返回退出。

(2) 服务器:

1. 阻塞，接收客户端报文，如果是SYN,seq=0,ack=0。发送报文:标志位（SYN，ACK），seq=0，ack=1；否则循环继续等待接收报文。
2. 非阻塞，开始计时，接收客户端报文，若超时则重传同步报文。
3. 判断接收的报文是否为：标志位（ACK），seq=1，ack=1。若是，建联成功。否则返回退出。

### (四) 四次挥手

对于四次挥手

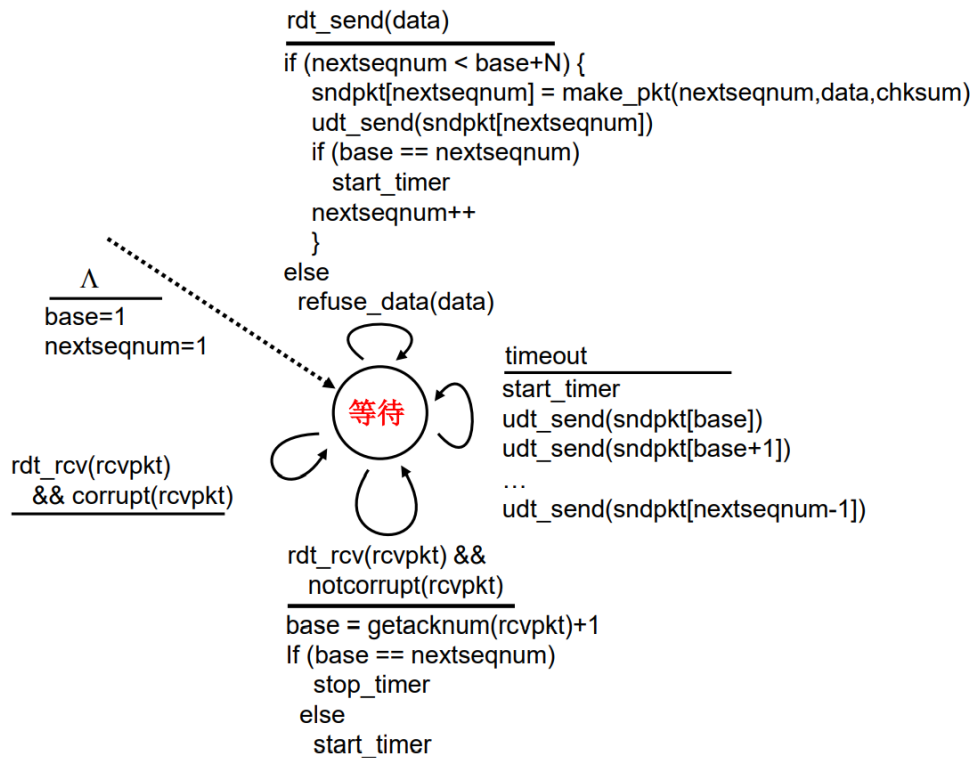
(1) 客户端:

1. 发送结束报文，标记位为FIN。
2. 开始计时，接收服务器报文，若超时则重传报文。
3. 判断接收的报文是否为：标志位（FIN，ACK）。否则继续接收。
4. 阻塞，接收报文，判断是否为：标志位（FIN）。若是，发送报文，标志位（FIN，ACK）。否则继续接收。
5. 等待2msl，如果收到服务器的FIN报文，重传确定报文。
6. 返回退出。

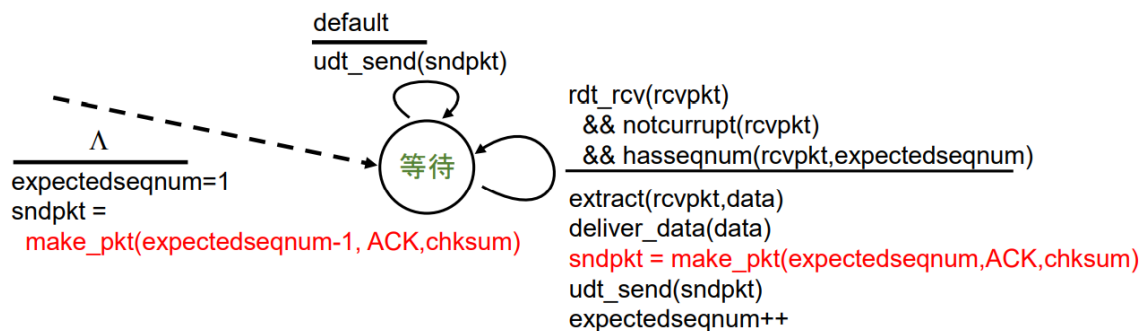
(2) 服务器:

1. 阻塞，接收客户端报文，如果是FIN。发送报文:标志位（FIN，ACK）；否则循环继续等待接收报文。
2. 无要传输的数据，发送FIN报文。
3. 非阻塞，超时重传Fin报文。判断接收的报文是否为(FIN,ACK)，如是断开连接。

### (五) Go Back N和累计确认



1. 对于发送端，维持了固定的窗口大小N，在运行过程中维持base和nextseqnum。
2. 其中base为已发送且已确认的序列号。nextseqnum为已发送的最新序列号。
3. 当nextseqnum < base+N时，才可以发送报文。当nextseqnum == base时，认为是该窗口的开始，开始计时器。发送序号为nextseqnum后，nextseqnum++。
4. 当收到确认报文时，如果base <= recpkt.ack则设置base = recpkt.ack+1。
5. 如果超时，则重传序列号从base到nextseqnum-1的报文，重设计时器。



1. 对于接收端，维持expectedseqnum为接收报文序号的累计序号。
2. 只有收到的报文的seq == expectedseqnum且校验通过，才将该报文的数据读取，发送确认报文，同时expectedseqnum++；
3. 否则重传ack为expectedseqnum-1报文，即重传已经确认的最大序列号。

### 三、GBN代码实现

## (一) 发送端

1. 初始化base=1, nextseqnum=1。
2. 分为主线程用来发送数据, 和接收线程用来接收确认报文。
3. 发送: `(nextseqnum<base+N)&&(nextseqnum<=packetNUM)` 时可发送报文, 第一次发送某报文, 其发送的seq为nextseqnum, 发送后nextseqnum++。若 `base == nextseqnum` 认为是该窗口的开始, 开始计时器。
4. 接收: `isAck(rec) && verfiyChecksum(rec, &ph) && (rec->ack_num >= base)` 当收到确认报文, 且 `rec->ack_num >= base` 时, `base = rec->ack_num + 1`。如果 `base == nextseqnum` 认为该窗口结束, 关闭计时器。
5. 超时, 重传seq为base到nextseqnum-1的报文, 重设计时器。
6. 发送数据报文结束, 发送标志位END的报文, 其数据为发送文件的名称。等待确认, 超时重传, 关闭。

接收线程:

```
//接收线程
DWORD WINAPI RecHandle(LPVOID param) {

    int len = sizeof(SOCKADDR_IN);
    char* recpktBuffer = new char[sizeof(msg)];
    msg* rec = (msg*)recpktBuffer;
    u_long imode = 0;
    ioctlsocket(sockClient, FIONBIO, &imode); //阻塞

    //rec&&notcorrupt(recpkt)
    while (rec_stage) {
        recvfrom(sockClient, recpktBuffer, sizeof(msg), 0, (sockaddr*)&addr_server,
        &len);

        if (isAck(rec) && verfiyChecksum(rec, &ph) && (rec->ack_num >=
        base)) {

            base = rec->ack_num + 1;
            printf("接收
            \tack:\t%d\tACK:\t%d\tlength:\t%d\tchecksum:\t%d\tbase: \t%d\tnextseqnum:
            \t%d\n", rec->ack_num, isAck(rec), rec->length, rec->checksum, base,
            nextseqnum);

            if (base == nextseqnum) {
                start_t = 0;
            }
            else {
                start_t = 1;
                start_timer = clock();
            }

        }
    }

    return 1;
}
```

主线程:

```
void GBN_send_FSM(unsigned long length_file ,char * file,char * filename) {

    int packetNUM = int(length_file / Max_Size) + (length_file % Max_Size ? 1 : 0);
    cout << "packetNUM: " << packetNUM << endl;

    int index = 0;

    int len = sizeof(SOCKADDR_IN);
    int packetDataLen = min(Max_Size, length_file - index * Max_Size);
    char *dataBuffer=new char[Max_Size];
    char *pktBuffer = new char[sizeof(msg)];
    char* recpktBuffer = new char[sizeof(msg)];
    msg* rec = (msg*)recpktBuffer;
    msg sndpkt;
    base=1;
    nextseqnum=1;
    start_t = 0;

    bool* first_send_pkt = new bool[packetNUM+1];
    memset(first_send_pkt, 1, packetNUM + 1);
    cout << "本次文件数据长度为      " << length_file << "Bytes,      需要传输" << packetNUM << "个数据包" << endl;
    rec_stage = 1;
    HANDLE rechandler = CreateThread(nullptr, 0, Rechandle, nullptr ,0, nullptr);

    while(1){

        //此分支为数据包发送结束，发送数据为文件名和标志位END，并等待服务器确认。
        if (base == packetNUM+1) {
            CloseHandle(rechandler); //关闭线程
            rec_stage = 0;
            u_long imode = 1;
            ioctlsocket(sockClient, FIONBIO, &imode); //非阻塞
            char* sendBuffer = new char[sizeof(msg)];
            memset(sendBuffer, 0, sizeof(msg));
            msg* sed = (msg*)sendBuffer;
            setEnd(sed);
            sed->source_port = port_client;
            sed->dest_port = port_server;
            string fn = filename;
            int filename_len = sizeof(fn);

            memcpy(sed->msg, filename, filename_len);
            sed->length = filename_len;
            setChecksum(sed, &ph); //设置校验和

            //发送
```

```

        sendto(sockClient, sendBuffer, sizeof(msg), 0,
(sockaddr*)&addr_server, len);
        cout << "客户端: 发送报文 (END) " << endl;

        clock_t start_timer = clock(); //开始计时

        while (recvfrom(sockClient, recpktBuffer, sizeof(msg), 0,
(sockaddr*)&addr_server, &len) <= 0 || !(isEnd(rec) && isAck(rec))) {
            // over time
            if (clock() - start_timer >= MAX_TIME) {

                //超时重传
                sendto(sockClient, sendBuffer, sizeof(msg), 0,
(sockaddr*)&addr_server, len);
                cout << "客户端: 发送报文 (END),重传" << endl;
                start_timer = clock();
            }
        }
        if (isEnd(rec) && isAck(rec) && verfiyChecksum(rec, &ph)) {
            cout << "客户端: 接收服务器报文 (END, ACK), 文件传输完成" << endl;
            return;
        }
        else
            continue;
    }

    //send(data)
    if((nextseqnum<base+N)&&(nextseqnum<=packetNUM)){

        for (int i = nextseqnum; (i < base + N)&&
(i<=packetNUM)&&first_send_pkt[i] ; i++) {
            index = i - 1;
            packetDataLen = min(Max_Size, length_file - index * Max_Size);
            memcpy(dataBuffer, file + index * Max_Size, packetDataLen);
            sndpkt = make_pkt(i, dataBuffer, packetDataLen);
            memcpy(pktBuffer, &sndpkt, sizeof(msg));
            sendto(sockClient, pktBuffer, sizeof(msg), 0,
(sockaddr*)&addr_server, len);
            printf("发送
\\tseq:\\t%d\\tindex:\\t%d\\tlength:\\t%d\\tchecksum:\\t%d\\tbase: \\t%d\\tnextseqnum:
\\t%d\\n", i, index, packetDataLen, sndpkt.checksum, base, nextseqnum);
            first_send_pkt[i] = 0; //用来标志当前包是否是第一次发送, 若非第一次发送
则由超时重传再次发送
            if (base == nextseqnum ) {
                start_t = 1;
                start_timer = clock();
            }
            nextseqnum++;
        }
    }

    //timeout
    if((clock() - start_timer >= MAX_TIME)&&start_t==1){

```



```

        start_timer = clock();
        for(int i=base;i <nextseqnum;i++){
            index = i - 1;
            packetDataLen = min(Max_Size, length_file - index * Max_Size);
            memcpy(dataBuffer, file+index* Max_Size, packetDataLen);
            sndpkt = make_pkt(i, dataBuffer, packetDataLen);
            memcpy(pktBuffer, &sndpkt, sizeof(msg));
            sendto(sockClient, pktBuffer, sizeof(msg), 0,
(sockaddr*)&addr_server, len);
            printf("发送\tseq:\t%d\tindex:\t%d\tlength:\t%d\tchecksum:\t%d\tbase:
\t%d\t\nextseqnum: \t%d\t(重传)\n", i, (i-1), packetDataLen, sndpkt.checksum,
base, nextseqnum);

        }
    }
}
}

```

## (二) 接收端

1. 初始化expectedseqnum=1。
2. 当收到报文的seq! =expectedseqnum时，重传ack为expectedseqnum-1报文,即已经确认的最大序列号。
3. 当收到报文的seq==expectedseqnum，且校验通过，将报文数据复制。发送确定报文，ack=expectedseqnum，expectedseqnum++。
4. 当收到标志位为END的报文，读取其数据为文件名称，发送确定报文，结束。

```

DWORD GBN_receive_FSM(char* file,char *filename) {

    int len = sizeof(SOCKADDR_IN);
    char* pktBuffer = new char[sizeof(msg)];
    char* recpktBuffer = new char[sizeof(msg)];
    char* sendBuffer = new char[sizeof(msg)];
    DWORD rec_data_len = 0;
    msg* rec = (msg*)recpktBuffer;
    u_long imode = 0;
    if(ioctlsocket(sockServer, FIONBIO, &imode)==SOCKET_ERROR)
        cout << "error" << endl;
    int expectedseqnum=1;

    while(1){

        recvfrom(sockServer, recpktBuffer, sizeof(msg), 0,
(sockaddr*)&addr_client, &len);

        //当收到标志位为END的报文，读取其数据为文件名称，发送确定报文，结束。
        if (isEnd(rec)) {

            memcpy(filename, rec->msg, rec->length);
            cout << "传输完毕" << endl;
            memset(sendBuffer, 0, sizeof(msg));

```

```

        msg sed = make_pkt(0);
        setEnd(&sed);
        sed.checksum = 0;
        setChecksum(&sed, &ph);

        memcpy(sendBuffer, &sed, sizeof(msg));

        sendto(sockServer, sendBuffer, sizeof(msg), 0,
(sockaddr*)&addr_client, len);
        cout << "服务器：发送报文 (END, ACK) " << endl;
        return rec_data_len;
    }

    //当收到报文的seq!=expectedseqnum时，重传ack为expectedseqnum-1报文,即已经
    确认的最大序列号。
    if (rec->seq_num != expectedseqnum) {
        msg sedpkt = make_pkt(expectedseqnum-1);
        memcpy(sendBuffer, &sedpkt, sizeof(msg));
        sendto(sockServer, sendBuffer, sizeof(msg), 0,
(sockaddr*)&addr_client, len);
        cout << "发送\tack:\t" << expectedseqnum-1 << "\tACK:\t " <<
isAck(&sedpkt) << "\tlength:\t" << sedpkt.length << "\tchecksum:\t" <<
sedpkt.checksum << "\texpectedseqnum: \t" << expectedseqnum << "失序" << endl;
    }

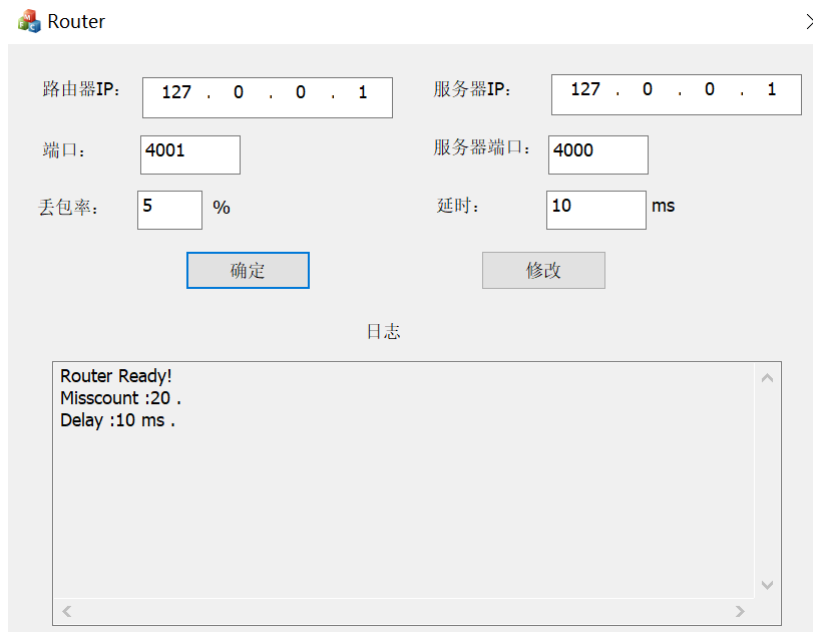
    //当收到报文的seq==expectedseqnum，且校验通过，将报文数据复制。发送确定报文，
    ack=expectedseqnum, expectedseqnum++.
    else if (rec->seq_num ==expectedseqnum && (verfiyChecksum(rec,
&ph))) {

        msg sedpkt = make_pkt(expectedseqnum);
        memcpy(sendBuffer, &sedpkt, sizeof(msg));
        sendto(sockServer, sendBuffer, sizeof(msg), 0,
(sockaddr*)&addr_client, len);
        expectedseqnum++;
        cout << "发送\tack:\t"<<expectedseqnum<<"\tACK:\t " <<
isAck(&sedpkt) << "\tlength:\t" << sedpkt.length << "\tchecksum:\t" <<
sedpkt.checksum << "\texpectedseqnum: \t" << expectedseqnum << endl;
        memcpy(file + rec_data_len, rec->msg, rec->length);
        rec_data_len += rec->length;
    }
}
}
}

```

## 四、程序演示

- (1)



路由器:

ip: 127.0.0.1 端口: 4001;

server:

ip: 127.0.0.1 端口: 4000;

- (2)建立连接



- (3) 传输

客户端:

```
C:\Users\nan\Desktop\网络实验_lab3-2_2012948_蒋浩南\程序\Client.exe
发送 seq: 224 index: 223 length: 8192 checksum: 11497 base: 217 nextseqnum: 224
接收 ack: 217 ACK: 1 length: 0 checksum: 56552 base: 218 nextseqnum: 224
发送 seq: 225 index: 224 length: 8192 checksum: 18316 base: 218 nextseqnum: 225
接收 ack: 218 ACK: 1 length: 0 checksum: 56551 base: 219 nextseqnum: 225
发送 seq: 226 index: 225 length: 8192 checksum: 33305 base: 219 nextseqnum: 226
接收 ack: 219 ACK: 1 length: 0 checksum: 56550 base: 220 nextseqnum: 226
发送 seq: 227 index: 226 length: 5961 checksum: 13282 base: 220 nextseqnum: 227
接收 ack: 220 ACK: 1 length: 0 checksum: 56549 base: 221 nextseqnum: 227
接收 ack: 221 ACK: 1 length: 0 checksum: 56548 base: 222 nextseqnum: 228
接收 ack: 222 ACK: 1 length: 0 checksum: 56547 base: 223 nextseqnum: 228
接收 ack: 223 ACK: 1 length: 0 checksum: 56546 base: 224 nextseqnum: 228
接收 ack: 224 ACK: 1 length: 0 checksum: 56545 base: 225 nextseqnum: 228
接收 ack: 225 ACK: 1 length: 0 checksum: 56544 base: 226 nextseqnum: 228
接收 ack: 226 ACK: 1 length: 0 checksum: 56543 base: 227 nextseqnum: 228
发送 seq: 227 index: 226 length: 5961 checksum: 13282 base: 227 nextseqnum: 228 (重传)
接收 ack: 227 ACK: 1 length: 0 checksum: 56542 base: 228 nextseqnum: 228
客户端: 发送报文 (END)
客户端: 接收服务器报文 (END, ACK), 文件传输完成
Total time: 23.833 s
吞吐率: 0.623456Mbps
/*****
```

服务器:

```
C:\Users\nan\Desktop\网络实验_lab3-2_2012948_蒋浩南\程序\Server.exe
发送 ack: 210 ACK: 1 length: 0 checksum: 56559 expectedseqnum: 211
发送 ack: 211 ACK: 1 length: 0 checksum: 56558 expectedseqnum: 212
发送 ack: 212 ACK: 1 length: 0 checksum: 56557 expectedseqnum: 213
发送 ack: 212 ACK: 1 length: 0 checksum: 56557 expectedseqnum: 213失序
发送 ack: 213 ACK: 1 length: 0 checksum: 56556 expectedseqnum: 214
发送 ack: 213 ACK: 1 length: 0 checksum: 56556 expectedseqnum: 214失序
发送 ack: 213 ACK: 1 length: 0 checksum: 56556 expectedseqnum: 214失序
发送 ack: 213 ACK: 1 length: 0 checksum: 56556 expectedseqnum: 214失序
发送 ack: 213 ACK: 1 length: 0 checksum: 56556 expectedseqnum: 214失序
发送 ack: 213 ACK: 1 length: 0 checksum: 56556 expectedseqnum: 214失序
发送 ack: 214 ACK: 1 length: 0 checksum: 56555 expectedseqnum: 215
发送 ack: 215 ACK: 1 length: 0 checksum: 56554 expectedseqnum: 216
发送 ack: 216 ACK: 1 length: 0 checksum: 56553 expectedseqnum: 217
发送 ack: 217 ACK: 1 length: 0 checksum: 56552 expectedseqnum: 218
发送 ack: 218 ACK: 1 length: 0 checksum: 56551 expectedseqnum: 219
发送 ack: 219 ACK: 1 length: 0 checksum: 56550 expectedseqnum: 220
发送 ack: 220 ACK: 1 length: 0 checksum: 56549 expectedseqnum: 221
发送 ack: 221 ACK: 1 length: 0 checksum: 56548 expectedseqnum: 222
发送 ack: 222 ACK: 1 length: 0 checksum: 56547 expectedseqnum: 223
发送 ack: 223 ACK: 1 length: 0 checksum: 56546 expectedseqnum: 224
发送 ack: 224 ACK: 1 length: 0 checksum: 56545 expectedseqnum: 225
发送 ack: 225 ACK: 1 length: 0 checksum: 56544 expectedseqnum: 226
发送 ack: 226 ACK: 1 length: 0 checksum: 56543 expectedseqnum: 227
发送 ack: 227 ACK: 1 length: 0 checksum: 56542 expectedseqnum: 228
传输完毕
服务器: 发送报文 (END, ACK)
/*****
```

传输结果:

» 网络实验\_lab3-2\_2012948\_蒋浩南 » 程序 » rec file



1.jpg

- (4)断开连接

```
是否继续接受传输 (Y/N) : n
/*****
服务器: 收到客户端Fin请求, 验证正确
服务器: 接收到报文 (FIN, ACK), 验证正确
连接关闭
请按任意键继续. . .
传输结束
客户端: 发送报文 (FIN)
客户端: 接收报文 (FIN, ACK) 验证正确
客户端: 接收到服务器报文 (FIN), 验证正
客户端: 发送报文 (FIN, ACK)
客户端: 连接关闭
请按任意键继续. . .
```

## 五、代码库

[lab3-2 · nan/computer network](#)