

# 计算机网络 3-1

## 个人信息

- 学号：2013750
- 姓名：管昀孜
- 专业：计算机科学与技术

## 实验要求

实验3-1：利用数据报套接字在用户空间实现面向连接的可靠数据传输，功能包括：建立连接、差错检测、确认重传等。流量控制采用停等机制，完成给定测试文件的传输。

## 协议设计

本次实验为实现单向传输，并且顺序较为固定。

- 对于客户端：建立链接（主动）-> 选择文件发送-> 关闭连接（主动）
- 对于服务器：建立连接（被动）-> 接收文件-> 关闭连接（被动）

本次基于的协议是rdt3.0，并根据个人用法稍微做出了一点改变：为了减少数据的传输量与方便编程，本次协议设计时在建立/关闭连接和发送文件内容时使用了两套协议（即两套不同的结构）

### 1. 建立与关闭连接

#### 格式设计

在建立和关闭连接时，最主要的标志位是SYN和FIN，这两个标志位就能告诉对方本次的数据包的作用。

- SYN同步标志：该标志仅在三次握手建立TCP连接时有效。在三次握手期间，SYN被置为1。
- FIN断开标志：带有该标志置位的数据包用来结束一个TCP回话，在挥手期间被置为1。

TCP协议除了SYN和FIN标志位以外还有另外4个标志位；但本次实验并不会用到这么多功能，因此进行简化处理，去掉另外4个标志位，而增加了另外一个标志位FLAG。

- FLAG标志位：为了避免因为翻转导致歧义，除去SYN、FIN和校验和之外，数据包中还增加了一个FLAG位用来标志本次数据包的作用。此标志位标明是握手还是挥手；且标明是第几次握手/挥手。

简单起见，建立连接是保持三次握手，但关闭连接时只进行两次挥手。

FLAG标志位设计如下：

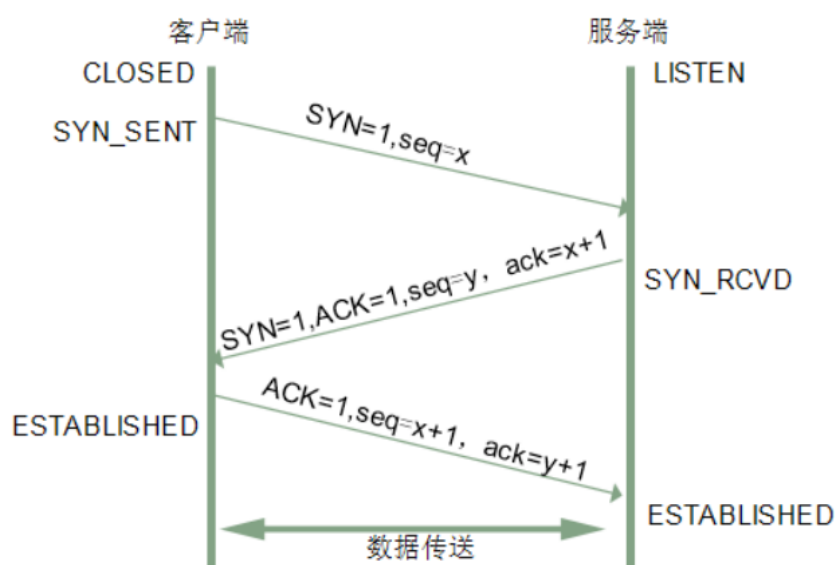
```
// 三次握手的时候对应的flag
const unsigned char SHAKE1 = 0x01;
const unsigned char SHAKE2 = 0x02;
const unsigned char SHAKE3 = 0x03;
// 四次挥手的时候的flag
const unsigned char WAVE1 = 0x04;
const unsigned char WAVE2 = 0x05;
```

数据包格式：`char package[4]`，具体为：

```
package[0]-----校验和
package[1]-----SYN
package[2]-----FIN
package[4]-----FLAG
```

## 建立连接 - 三次握手

1. 由client发起建立连接请求，发送时需要将**SYN置1**，**FIN置0**；第一次握手时**FLAG**需要设置为 **SHAKE1**。
2. server接收到第一次握手内容后，若不正确则不做任何相应，让client自行超时重传；若数据包正确，则发送**SYN为1**，**FIN为0**，**FLAG为SHAKE2**的数据包。
3. client收到来自server的第二次握手内容后，检查数据包，如果不正确，不做回复，等待server的超时重传；如果正确，向server发送**SYN为1**，**FIN为0**，**FLAG为SHAKE3**的数据包。
4. server收到来自client的第三次握手内容，检查数据包，如果不正确，回到最开始的状态重新等待连接；如果正确，回显连接建立。



如图，模仿的TCP三次握手，但没有设计ACK/seq/ack位

## 关闭连接 - 两次挥手

1. 由client发起关闭连接请求，发送时需要将**SYN置0**，**FIN置1**；第一次挥手时FLAG需要设置为 **WAVE1**。
2. server接收到第一次挥手内容后，若不正确则不做任何相应，让client自行超时重传；若数据包正确，则发送**SYN为0**，**FIN为1**，FLAG为**WAVE2**的数据包，并从服务器这边关闭连接。
3. client收到来自server的第二次挥手内容后，检查数据包，如果不正确，重新发送挥手数据包（重发的原因是猜测可能服务器那边没有收到挥手数据包，而在这里因为只有两次挥手，在这里可以简单当作收到第二次挥手的时候无论是否正确直接关闭连接），等待server的超时重传；如果正确，关闭连接。

## 2. 数据传输

### 格式设计

由于本次是单项传输：

- 对于client端来说，要发送校验和、seq位、数据长度、判断是否为最后一个包的标志位以及数据内容，因此比较复杂；
- 对于server端来说，只需要回复是ACK0还是ACK1即可（client作为发送端，server作为接收端），另外需要一个校验和，因此数据包格式比较简单

由于发送端和接收端复杂程度不同，为了简化，设计成两种不同的数据格式。

发送端：每次传送的数据包大小均为1024个字节，具体内容为：

```
data[1024];
data[0] ---- 校验和
data[1] ---- 序号seq位（仅为0,1翻转）
data[2] ---- 数据长度高8位
data[3] ---- 数据长度低8位
data[4] ---- isLast位，用于当前数据包是否为最后一个数据包
data[5~1023] ---- 要传输的数据内容，长度与data[2]、data[3]中算出来的数相同
```

接收端：传输内容仅为2个字节，具体内容为：

```
answer[2];
answer[0] ---- 校验和
answer[1] ---- ACK
```

## 传输流程

以下数据包在封装时默认都计算过校验和，并默认在收到数据包时都会检查校验和，校验和出错和下面其他位的出错处理方式相同。

1. 在发送一个文件时，发送方需要先发一个包，包中内容为需要发送的文件的名称，**seq置为0**；
2. 接收端收到的第一个数据包必须为刚刚发送的关于文件名称的数据包，解析数据包获得名称以后返回**ACK为0**的数据包，并将ACK设置为1；如果收到的包**出错**（校验和或者seq为1），向发送端发送**ACK为0x02**的数据包。
3. 发送端收到**收到ACK和自身seq状态相同**的数据包则封装下一个数据包，将自身状态的**seq翻转**，设置在数据包的seq位中；如果**状态不同则重新发送上一个数据包**。
4. 接收端收到**和自身ACK状态相同**的数据包则封装下一个回复的数据包（ACK位设置为自身ACK的状态），将自身状态的ACK翻转，并将接受的数据写入文件中；如果状态不同则重新发送上一个数据包
5. 重复步骤2、3，直到发送端在发送最后一个数据包时，将**isLast位设置为1**（其他操作同前）
6. 接收端收到**isLast为1**的数据包，在回复对应数据包后停止接收数据的过程，进入到**等待关闭连接**的状态。

其中，在每一次等待接收数据包时，双方都会记录本次发送的时间，以此来计算时间、进行超时重传。



```

handle = _findfirst(path.c_str(), &fileinfo);
// 判断句柄状态
if (handle == -1) {
    cout << "文件名称出现错误\n";
    return;
}
else {
    cout << "测试文件夹下目录列表为: \n";
    int tempNumOfFile = 0;
    do
    {
        //找到的文件的文件名
        if (fileinfo.name[0] == '.')
            continue;
        cout << ++tempNumOfFile << ": " << fileinfo.name << endl;

    } while (!_findnext(handle, &fileinfo));
    _findclose(handle);
    return;
}
}

```

### 3. 获取IP

```

void getIP(char* p) {
    cout << "Please use 'ipconfig' to get your ip and input\n";
    char ip[16];
    cin.getline(ip, sizeof(ip));
    int index = 0;
    while (ip[index] != '\0') {
        p[index] = ip[index];
        index++;
    }
}

```

## Client(发送端)相关函数

### 4. 三次握手 - 发送端

```

// 三次握手，建立连接
void connet2Server() {
    // 创建一个要发送的数据包，并初始化
    // 数据包: check, SYN, FIN, SHAKE/WAVE
    char shakeChar[4];
    shakeChar[1] = 0x01;
    shakeChar[2] = 0x00;
    shakeChar[3] = SHAKE1;
    shakeChar[0] = newGetChecksum(shakeChar+1, 3);

    // 发送shake1
    sendto(sendSocket, shakeChar, 4, 0, (SOCKADDR*)&serverAddr, socketAddrLen);
    cout << "开始建立连接，请求建立连接的数据包为（第一次握手）: \n";
    cout << "SYN: " << int(shakeChar[1]) << "\tFIN: " << int(shakeChar[2]) <<
    "\tSHAKE: " << int(shakeChar[3]) << "\tchecksum: " << int(shakeChar[0]) << endl;

    // 开始计时
}

```

```

int beginTime = clock();

// 接收缓冲区
char recvShakeBuf[4];
memset(recvShakeBuf, 0, 4);

//超时重传
while((recvfrom(sendSocket, recvShakeBuf, 4, 0, (SOCKADDR*)&serverAddr,
&socketAddrLen))==SOCKET_ERROR){
    if (clock() - beginTime > MAX_WAIT_TIME) {
        cout << "超过最长等待时间, 重传:";
        sendto(sendSocket, shakeChar, 4, 0, (SOCKADDR*)&serverAddr,
socketAddrLen);
        beginTime = clock();
        cout << "SYN: " << int(shakeChar[1]) << "\tFIN: " <<
int(shakeChar[2]) << "\tSHAKE: " << int(shakeChar[3]) << "\tchecksum: " <<
int(shakeChar[0]) << endl;
    }
}

// 收到数据包
cout << "收到数据包为: \n";
cout << "SYN: " << int(recvShakeBuf[1]) << "\tFIN: " << int(recvShakeBuf[2])
<< "\tSHAKE: " << int(recvShakeBuf[3]) << "\tchecksum: " << int(recvShakeBuf[0])
<< endl;

// 收到shake2正确, 则发送shake3
if (newGetChecksum(recvShakeBuf, 4)==0 && recvShakeBuf[1] == 0x01 &&
recvShakeBuf[2] == 0x00 && recvShakeBuf[3] == SHAKE2) {
    memset(shakeChar, 0, 4);
    shakeChar[1] = 0x01;
    shakeChar[2] = 0x00;
    shakeChar[3] = SHAKE3;
    shakeChar[0] = newGetChecksum(shakeChar+1, 3);
    sendto(sendSocket, shakeChar, 4, 0, (SOCKADDR*)&serverAddr,
socketAddrLen);
    cout << "发送第三次握手数据包: \n";
    cout << "SYN: " << int(shakeChar[1]) << "\tFIN: " << int(shakeChar[2])
<< "\tSHAKE: " << int(shakeChar[3]) << "\tchecksum: " << int(shakeChar[0]) <<
endl;
    cout << "\n连接建立成功\n\n";
}
}

```

## 5. 发送文件

```

static void sendFile2Server(char* fileName, unsigned short int len) {
    // 对输入的名字进行处理
    char filePath[] = "./test/";
    char* fileNameN = new char[7 + len + 1];
    for (int i = 0; i < 7 + len + 1; i++) {
        if (i < 7)
            fileNameN[i] = filePath[i];
        else
            fileNameN[i] = fileName[i - 7];
    }
    fileNameN[7 + len + 1] = '\0';
}

```

```

// 读取文件（以二进制形式读入）
ifstream toSendFile(fileName, ifstream::in | ios::binary);

// 计算文件的长度
toSendFile.seekg(0, toSendFile.end); // seekg() 是对输入文件定位，它有两个参数：第一个参数是偏移量，第二个参数是基地址。
// tellg() 函数不需要带参数，它返回当前定位指针的位置，也代表着输入流的大小。
unsigned int fileLen = toSendFile.tellg(); // 文件长度
toSendFile.seekg(0, toSendFile.beg);
cout << "计算得到要发送的文件长度为：" << fileLen << endl;

// 计算需要封装成多少个包
unsigned int totalNum = fileLen / 1019;
if (fileLen % 1019 > 0)
    totalNum += 1;
// 再加一个表示文件名称的包
totalNum += 1;
cout << "一共需要封装成" << totalNum << "个包\n";

// 创建临时缓冲区
char* tempBuf = new char[fileLen];
// 将图片装载到缓冲区
toSendFile.read(tempBuf, fileLen);

char seq = 0x00; // 从0号数据包开始，每次发送之后进行翻转
unsigned int currentNum = 0; // 记录已经成功发送了几个数据包，从0开始

///// 开始封装数据包
// 首先发送的是表示文件名称的数据包
char sendPackage[1024];
memset(sendPackage, 0, 1024);
// 设置包体的内容
sendPackage[1] = seq;
for (unsigned short int i = 0; i < len; i++) {
    sendPackage[i + 5] = fileName[i];
}
sendPackage[3] = unsigned char(len & 0x00FF);
sendPackage[2] = unsigned char((len >> 8) & 0x00FF);
sendPackage[0] = newGetChecksum(sendPackage+1, 1024-1);

// 从开始发送第一个数据包这里开始计算时间
sendFileTime = clock();

// 发送第一个数据包
sendto(sendSocket, sendPackage, 1024, 0, (SOCKADDR*)&serverAddr,
socketAddrLen);
cout << "当前发送数据包为：\nseq: " << int(sendPackage[1]) << "\tlen高: " <<
int(sendPackage[2]) << "\tlen低: " << int(sendPackage[3])
    << "\tchecksum: " << int(sendPackage[0]) << endl;

int beginTime = clock();
// 创建一个临时的包，用来存放内容，可以考虑改成2
char recvShakeBuf[2];
// 停等接收
while (true) {
    while (recvfrom(sendSocket, recvShakeBuf, 2, 0, (SOCKADDR*)&serverAddr,
&socketAddrLen) == SOCKET_ERROR) {
        if (clock() - beginTime > MAX_WAIT_TIME) {

```

```

        cout << "超过最长等待时间，重传:";
        sendto(sendSocket, sendPackage, 1024, 0, (SOCKADDR*)&serverAddr,
socketAddrLen);
        beginTime = clock();
        cout << "当前发送数据包为: \nseq: " << int(sendPackage[1]) << "\tlen
高: " << int(sendPackage[2]) << "\tlen低: " << int(sendPackage[3])
        << "\tchecksum: " << int(sendPackage[0]) << endl;
        // cout << "SYN: " << int(sendPackage[1]) << "\tFIN: " <<
int(sendPackage[2]) << "\tSHAKE: " << int(sendPackage[3]) << "\tchecksum: " <<
int(sendPackage[0]) << endl;
    }
}
    cout << "FROM S:\nack: " << int(recvShakeBuf[1]) << "\tchecksum: " <<
int(recvShakeBuf[0]) << endl;
    // 检查收到的包是否正确
    if (newGetChecksum(recvShakeBuf, 2)==0 && recvShakeBuf[1] == seq) {
        // 正确，打包下一部分发送
        // 记录的数量+1
        currentNum += 1;
        if (currentNum == totalNum) {
            cout << "文件传输完成\n";
            break;
        }
        // seq翻转
        if (seq == 0x00)
            seq = 0x01;
        else
            seq = 0x00;

        // 清空发送缓存
        memset(sendPackage, 0, 1024);
        // 设置标志位
        sendPackage[1] = seq;
        // 装载下一部分文件
        unsigned short int actualLen = 0;
        for (int tempPoint = 0; (tempPoint < 1019) && ((currentNum - 1) *
1019 + tempPoint)<fileLen; tempPoint++) {
            sendPackage[5 + tempPoint] = tempBuf[(currentNum - 1) * 1019 +
tempPoint];
            actualLen += 1;
        }
        // 设置长度
        sendPackage[3] = unsigned char(actualLen & 0x00FF);
        sendPackage[2] = unsigned char((actualLen >> 8) & 0x00FF);

        // 设置isLast
        if (currentNum == totalNum - 1)
            sendPackage[4] = 0x01;
        sendPackage[0] = newGetChecksum(sendPackage + 1, 1024 - 1);
        sendto(sendSocket, sendPackage, 1024, 0, (SOCKADDR*)&serverAddr,
socketAddrLen);
        beginTime = clock();
        cout << "当前发送数据包为: \nseq: " << int(sendPackage[1]) << "\tlen高:
" << int(sendPackage[2]) << "\tlen低: " << int(sendPackage[3])
        << "\tchecksum: " << int(sendPackage[0]) << endl;
    }
    else {
        // 错误，重传

```



```

        cout << "收到错误的ack, 重新发送\n";
        sendto(sendSocket, sendPackage, 1024, 0, (SOCKADDR*)&serverAddr,
socketAddrLen);
        beginTime = clock();
        cout << "当前发送数据包为: \nseq: " << int(sendPackage[1]) << "\tlen高:
" << int(sendPackage[2]) << "\tlen低: " << int(sendPackage[3])
        << "\tchecksum: " << int(sendPackage[0]) << endl;
    }
}

sendFileTime = clock() - sendFileTime;
// 计算成秒
sendFileTime = sendFileTime / CLOCKS_PER_SEC;
cout << "本次传输所用时间为: " << sendFileTime << " s." << endl;
// 计算吞吐率
Throughput = fileLen / sendFileTime;
}

```

## 6. 两次挥手 - 发送端

```

static void sayGoodBye2Server() {
    // 两次挥手, 断开连接

    // 数据包: check, SYN, FIN, SHAKE/WAVE
    char shakeChar[4];
    shakeChar[1] = 0x00;
    shakeChar[2] = 0x01;
    shakeChar[3] = WAVE1;
    shakeChar[0] = newGetChecksum(shakeChar+1, 4-1);
    // 发送WAVE1
    sendto(sendSocket, shakeChar, 4, 0, (SOCKADDR*)&serverAddr, socketAddrLen);
    cout << "断开连接, 发送数据包为 (第一次挥手): \n";
    cout << "SYN: " << int(shakeChar[1]) << "\tFIN: " << int(shakeChar[2]) <<
"\tSHAKE: " << int(shakeChar[3]) << "\tcheckSum: " << int(shakeChar[0]) << endl;

    // 开始计时
    int beginTime = clock();

    // 停等, 等服务器的挥手
    char rcvwaveBuf[4];
    while (true) {
        while ((recvfrom(sendSocket, rcvwaveBuf, 4, 0, (SOCKADDR*)&serverAddr,
&socketAddrLen))==SOCKET_ERROR) {
            if (clock() - beginTime > MAX_WAIT_TIME) {
                cout << "超过最长等待时间, 重传:";
                sendto(sendSocket, shakeChar, 4, 0, (SOCKADDR*)&serverAddr,
socketAddrLen);
                beginTime = clock();
                cout << "SYN: " << int(shakeChar[1]) << "\tFIN: " <<
int(shakeChar[2]) << "\tSHAKE: " << int(shakeChar[3]) << "\tcheckSum: " <<
int(shakeChar[0]) << endl;
            }
        }
        if (newGetChecksum(rcvwaveBuf, 4)==0 && rcvwaveBuf[1] == 0x00 &&
rcvwaveBuf[2] == 0x01 && rcvwaveBuf[3] == WAVE2) {
            // 收到握手包正确
            cout << "收到来自服务器的第二次挥手: \n";

```

```

        cout << "SYN: " << int(recvWaveBuf[1]) << "\tFIN: " <<
int(recvWaveBuf[2]) << "\tSHAKE: " << int(recvWaveBuf[3]) << "\tchecksum: " <<
int(recvWaveBuf[0]) << endl;
        memset(recvWaveBuf, 0, 4);
        memset(shakeChar, 0, 4);
        break;
    }
    else {
        cout << "服务器挥手失败，重传挥手数据包\n";
        sendto(sendSocket, shakeChar, 4, 0, (SOCKADDR*)&serverAddr,
socketAddrLen);
        cout << "断开连接，发送数据包为（第一次挥手）：\n";
        cout << "SYN: " << int(shakeChar[1]) << "\tFIN: " <<
int(shakeChar[2]) << "\tSHAKE: " << int(shakeChar[3]) << "\tchecksum: " <<
int(shakeChar[0]) << endl;
    }
}
}
}

```

## Server(接收端)相关函数

### 7. 三次握手 - 接收端

```

static bool shakeHand() {
    // 握手
    //Package p;
    //Package sendP;
    char recvShakeBuf[4];
    while (1) {
        memset(recvShakeBuf, 0, 4);
        // 停等机制接收数据包
        while (recvfrom(serverSocket, recvShakeBuf, 4, 0,
(sockaddr*)&clientAddr, &sockaddrLen) == SOCKET_ERROR) {

        }

        // 检查接收到的数据包校验和是否正确
        // 先对收到的数据进行转换
        //p = *((Package*)recvBuf);
        // 输出信息
        cout << "收到建立连接的请求，数据包为：\n";
        // packageOutput(p);
        cout << "SYN: " << int(recvShakeBuf[1]) << "\tFIN: " <<
int(recvShakeBuf[2]) << "\tSHAKE: " << int(recvShakeBuf[3]) << "\tchecksum: " <<
int(recvShakeBuf[0]) << endl;
        // 第一次
        if (newGetChecksum(recvShakeBuf,4)==0 && recvShakeBuf[1] == 0x01 &&
recvShakeBuf[2] == 0x00 && recvShakeBuf[3] == SHAKE1) {
            // 收到握手包正确
            cout << "握手包正确\n";
            // 创建第二次握手包
            char secondShake[4];
            secondShake[1] = 0x01;
            secondShake[2] = 0x00;
            secondShake[3] = SHAKE2;
            secondShake[0] = newGetChecksum(secondShake+1, 4-1);
            // 发送第二次握手包

```

```

        sendto(serverSocket, secondShake, 4, 0, (SOCKADDR*)&clientAddr,
sockaddrLen);
        // 输出内容
        cout << "SYN: " << int(secondShake[1]) << "\tFIN: " <<
int(secondShake[2]) << "\tSHAKE: " << int(secondShake[3]) << "\tchecksum: " <<
int(secondShake[0]) << endl;

        int beginTime = clock();

        // 清空刚刚接收的包，等待第三次握手
        memset(recvShakeBuf, 0, 4);

        while (recvfrom(serverSocket, recvShakeBuf, 4, 0,
(sockaddr*)&clientAddr, &sockaddrLen) == SOCKET_ERROR) {
            if (clock() - beginTime > MAX_WAIT_TIME) {
                cout << "超过最长等待时间，重传:";
                sendto(serverSocket, secondShake, 4, 0,
(SOCKADDR*)&clientAddr, sockaddrLen);
                beginTime = clock();
                cout << "SYN: " << int(secondShake[1]) << "\tFIN: " <<
int(secondShake[2]) << "\tSHAKE: " << int(secondShake[3]) << "\tchecksum: " <<
int(secondShake[0]) << endl;
            }
        }

        cout << "收到数据包为: \n";
        cout << "SYN: " << int(recvShakeBuf[1]) << "\tFIN: " <<
int(recvShakeBuf[2]) << "\tSHAKE: " << int(recvShakeBuf[3]) << "\tchecksum: " <<
int(recvShakeBuf[0]) << endl;

        if (newGetChecksum(recvShakeBuf, 4)==0 && recvShakeBuf[1] == 0x01 &&
recvShakeBuf[2] == 0x00 && recvShakeBuf[3] == SHAKE3) {
            // 收到握手包正确
            cout << "\n连接建立成功\n\n";
            return true;
        }
    }
    else {
        cout << "收到的数据包有误，重新等待连接\n";
        continue;
    }
}
}

```

## 8. 接收文件

```

static void recvFile() {
    // 接收文件
    // 初始ack
    char ack = 0x00;
    // 创建缓冲区
    char recvBuf[1024];
    memset(recvBuf, 0, 1024);
    // 创建发送使用的缓冲区
    char sendBuf[2];
    memset(sendBuf, 0, 2);
}

```

```

// 记录名字和名字长度
unsigned short int nameLen = 0;
char* fileName;
// 停等，等名字
while (true) {
    // 接收到的第一个包是名字
    while (recvfrom(serverSocket, recvBuf, 1024, 0, (sockaddr*)&clientAddr,
&sockaddrLen) == SOCKET_ERROR) {

    }
    cout << "From C\t收到的数据包为;\n";
    cout << "seq: " << int(recvBuf[1]) << "\tlen高: " << int(recvBuf[2]) <<
"\tlen低: " << int(recvBuf[3]) << "\tisLast: " << int(recvBuf[4]) << "\tchecksum:
" << int(recvBuf[0]) << endl;

    if (newGetChecksum(recvBuf, 1024)==0 && recvBuf[1] == ack) {
        // 记录名称
        // 计算名字的长度
        unsigned short int temp = unsigned short int(recvBuf[2]);
        nameLen = (temp << 8) + unsigned short int(recvBuf[3]);
        fileName = new char[nameLen];
        for (int i = 0; i < nameLen; i++)
            fileName[i] = recvBuf[i + 5];

        // 没有损坏，返回对应ack，然后ack翻转
        sendBuf[1] = ack;
        sendBuf[0] = newGetChecksum(sendBuf+1, 1);
        sendto(serverSocket, sendBuf, 2, 0, (SOCKADDR*)&clientAddr,
sockaddrLen);
        cout << "TO C\t正确返回数据包\n";
        cout << "checksum: " << int(sendBuf[0]) << "\tack: " <<
int(sendBuf[1]) << endl;

        // ack翻转
        if (ack == 0x00)
            ack = 0x01;
        else
            ack = 0x00;
        break;
    }
    else {
        // 有损坏，返回错误的ack
        // 返回ack1
        sendBuf[1] = 0x02;
        sendBuf[0] = newGetChecksum(sendBuf+1, 2-1);
        sendto(serverSocket, sendBuf, 2, 0, (SOCKADDR*)&clientAddr,
sockaddrLen);
        cout << "TO C\t接收到的数据包有损坏，返回错误ack\n";
        cout << "checksum: " << int(sendBuf[0]) << "\tack: " <<
int(sendBuf[1]) << endl;
        continue;
    }
}

// 对名字进行处理
char filePath[] = "C:\\Users\\11955\\Desktop\\test\\";
char* fileNameN = new char[28 + nameLen + 1];
for (int i = 0; i < 28 + nameLen; i++) {

```

```

        if (i < 28)
            fileNameN[i] = filePath[i];
        else {
            fileNameN[i] = fileName[i - 28];
        }
    }
    fileNameN[28 + nameLen] = '\0';

    //cout << fileNameN << endl;
    ofstream toWriteFile;
    toWriteFile.open(fileNameN, ios::binary);
    if (!toWriteFile) {
        cout << "文件创建失败\n";
    }

    int beginTime = clock();

    // 接下来停等接收正式的数据包
    int currentNum = 0;
    while (true) {
        while (recvfrom(serverSocket, recvBuf, 1024, 0, (sockaddr*)&clientAddr,
            &sockaddrLen) == SOCKET_ERROR) {
            if (clock() - beginTime > MAX_WAIT_TIME) {
                cout << "超过最大等待时间，重传：";
                sendBuf[1] = (~ack) & 0x01;
                sendBuf[0] = newGetChecksum(sendBuf + 1, 1);
                sendto(serverSocket, sendBuf, 2, 0, (SOCKADDR*)&clientAddr,
sockaddrLen);
                cout << "checksum: " << int(sendBuf[0]) << "\tack: " <<
int(sendBuf[1]) << endl;
                beginTime = clock();
            }
        }
        cout << "From C\t收到的数据包为;\n";
        cout << "seq: " << int(recvBuf[1]) << "\tlen高: " << int(recvBuf[2]) <<
"\tlen低: " << int(recvBuf[3]) << "\tisLast: " << int(recvBuf[4])<< "\tchecksum: "
<< int(recvBuf[0]) << endl;

        if (newGetChecksum(recvBuf, 1024)==0 && recvBuf[1] == ack) {
            // 将内容输入到文件中
            unsigned short int temp = unsigned short int(recvBuf[2]);
            //cout << "temp高位: " << temp << endl;
            temp = (temp << 8) + (unsigned short int(recvBuf[3]) & 0x00FF);
            //cout << "temp<<8" << (unsigned short int(recvBuf[2]) << 8) <<
endl;

            //cout << "temp低位: " << unsigned short int(recvBuf[3]) << endl;
            cout << "temp长度: " << temp << endl << endl;
            for (int i = 0; i < temp; i++) {
                toWriteFile << recvBuf[5 + i];
                //cout<< recvBuf[5 + i];
            }

            currentNum += 1;

            // 没有损坏，返回对应ack，然后ack翻转
            sendBuf[1] = ack;
            sendBuf[0] = newGetChecksum(sendBuf+1, 1);

```

```

        sendto(serverSocket, sendBuf, 2, 0, (SOCKADDR*)&clientAddr,
sockaddrLen);
        beginTime = clock();
        cout << "TO C\t正确返回数据包\n";
        cout << "checksum: " << int(sendBuf[0]) << "\tack: " <<
int(sendBuf[1]) << endl;

        // ack翻转
        if (ack == 0x00)
            ack = 0x01;
        else
            ack = 0x00;

        /* cout << "\n判断是不是最后一个\n";
        cout << (int(recvBuf[4]) == 1) << endl << endl;*/
        // 确认当前收到的是最后一个
        if (int(recvBuf[4]) == 1)
            break;
    }
    else {
        // 有损坏, 返回错误的ack
        // 返回ack1
        sendBuf[1] = 0x02;
        sendBuf[0] = newGetChecksum(sendBuf+1, 2-1);
        sendto(serverSocket, sendBuf, 2, 0, (SOCKADDR*)&clientAddr,
sockaddrLen);
        beginTime = clock();
        cout << "TO C\t接收到的数据包有损坏, 返回错误ack\n";
        cout << "check: " << int(sendBuf[0]) << "\tack: " << int(sendBuf[1])
<< endl;
        continue;
    }
}

// 文件接收完毕, 关闭文件
toWriteFile.close();

cout << "文件保存成功\n";
}

```

## 9. 两次挥手 - 接收端

```

static void waveHand() {
    // 挥手, 挥两次
    char recvWaveBuf[4];
    while (true) {
        while (recvfrom(serverSocket, recvWaveBuf, 4, 0, (sockaddr*)&clientAddr,
&sockaddrLen) == SOCKET_ERROR) {

        }

        cout << "收到断开连接的请求, 数据包为: \n";
        cout << "SYN: " << int(recvWaveBuf[1]) << "\tFIN: " <<
int(recvWaveBuf[2]) << "\tSHAKE: " << int(recvWaveBuf[3]) << "\tcheckSum: " <<
int(recvWaveBuf[0]) << endl;

        // 第一次

```

```

        if (newGetChecksum(recvWaveBuf, 4)==0 && recvWaveBuf[1] == 0x00 &&
recvWaveBuf[2] == 0x01 && recvWaveBuf[3] == WAVE1) {
            // 收到挥手包正确
            cout << "挥手包正确\n";
            // 创建第二次挥手包
            char secondShake[4];
            secondShake[1] = 0x00;
            secondShake[2] = 0x01;
            secondShake[3] = WAVE2;
            secondShake[0] = newGetChecksum(secondShake+1, 4-1);
            // 发送第二次挥手包
            sendto(serverSocket, secondShake, 4, 0, (SOCKADDR*)&clientAddr,
sockaddrLen);
            // 输出内容
            cout << "发送给客户端的第二次挥手: \n";
            cout << "SYN: " << int(secondShake[1]) << "\tFIN: " <<
int(secondShake[2]) << "\tSHAKE: " << int(secondShake[3]) << "\tchecksum: " <<
int(secondShake[0]) << endl;

            memset(secondShake, 0, 4);
            memset(recvWaveBuf, 0, 4);
            break;
        }
    }
}

```

## 吞吐率计算

使用C++中的clock函数，从开始传输文件时计时，文件传输完成结束。

吞吐率计算公式为：传输的文件大小/总时长；其中文件大小按字节计算，总时长按秒计算

## 使用方法

### Client - 发送端

1. 运行程序
2. 输入要发送的IP地址
3. 输入本地的IP地址
4. 选择文件发送（需要输入文件的全名）
5. 文件发送完成自动结束程序

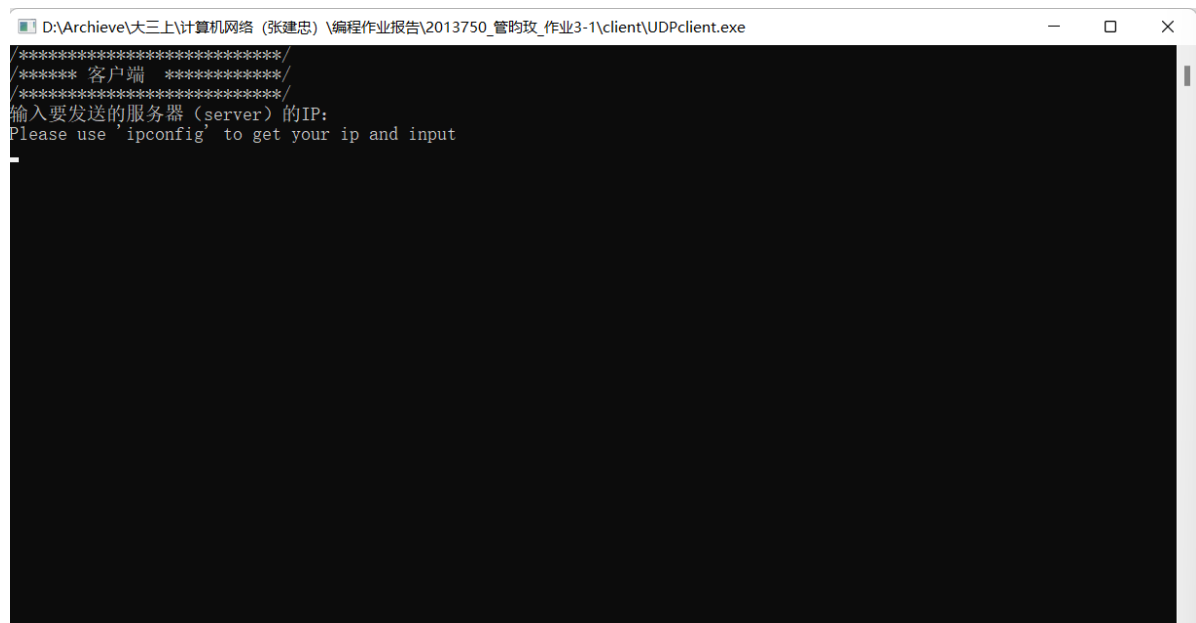
### Server - 接收端

1. 运行程序
2. 输入要接收的发送端的IP
3. 等待连接
4. 连接成功后等待发送端发送文件
5. 文件接收完成后自动结束程序

## 实验成果展示

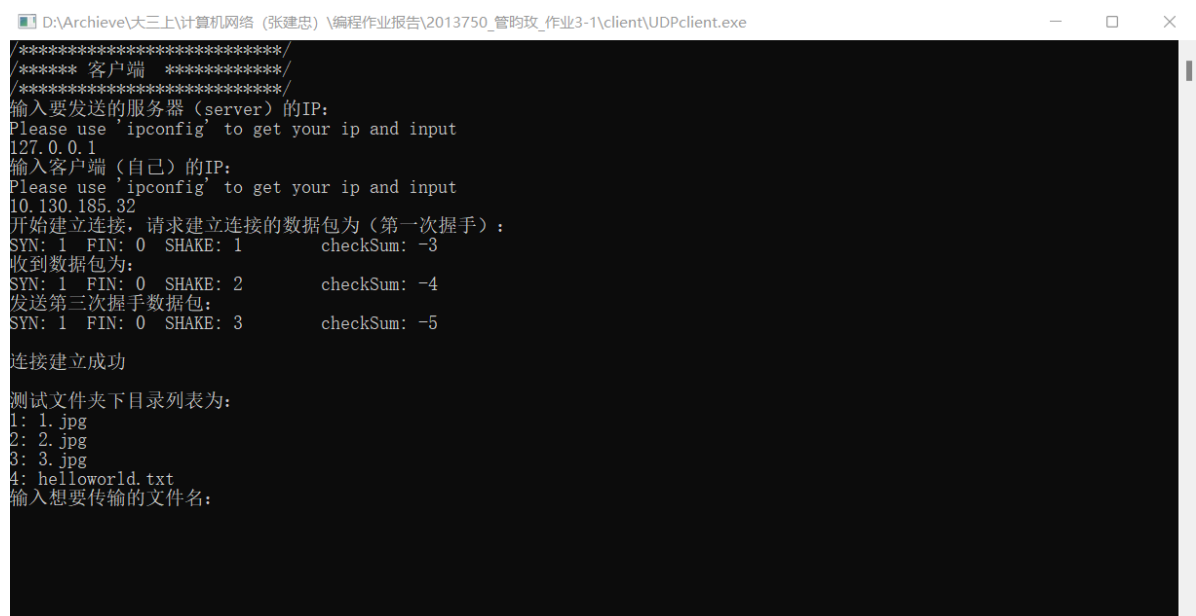
## Client发送端

等待连接页面



```
D:\Archive\大三上\计算机网络 (张建忠) \编程作业报告\2013750_管昀孜_作业3-1\client\UDPclient.exe
*****
/***** 客户端 *****/
*****
输入要发送的服务器 (server) 的IP:
Please use 'ipconfig' to get your ip and input
```

输入服务器IP和客户端IP，三次握手成功，自动弹出文件列表并等待输入



```
D:\Archive\大三上\计算机网络 (张建忠) \编程作业报告\2013750_管昀孜_作业3-1\client\UDPclient.exe
*****
/***** 客户端 *****/
*****
输入要发送的服务器 (server) 的IP:
Please use 'ipconfig' to get your ip and input
127.0.0.1
输入客户端 (自己) 的IP:
Please use 'ipconfig' to get your ip and input
10.130.185.32
开始建立连接，请求建立连接的数据包为 (第一次握手):
SYN: 1 FIN: 0 SHAKE: 1 checksum: -3
收到数据包为:
SYN: 1 FIN: 0 SHAKE: 2 checksum: -4
发送第三次握手数据包:
SYN: 1 FIN: 0 SHAKE: 3 checksum: -5
连接建立成功
测试文件夹下目录列表为:
1: 1.jpg
2: 2.jpg
3: 3.jpg
4: helloworld.txt
输入想要传输的文件名:
```

输入要发送的文件名，进行发送



```
输入想要传输的文件名: helloworld.txt
计算得到要发送的文件长度为: 1655808
一共需要封装成1626个包
```

发送过程中



```

当前发送数据包为:
seq: 1 len高: 3 len低: -5 checksum: 19
FROM S:
ack: 1 checksum: -2
当前发送数据包为:
seq: 0 len高: 3 len低: -5 checksum: -70
FROM S:
ack: 0 checksum: -1
当前发送数据包为:
seq: 1 len高: 3 len低: -5 checksum: 77
FROM S:
ack: 1 checksum: -2
当前发送数据包为:
seq: 0 len高: 3 len低: -5 checksum: -89
FROM S:
ack: 0 checksum: -1
当前发送数据包为:
seq: 1 len高: 3 len低: -5 checksum: -22
FROM S:
ack: 1 checksum: -2
当前发送数据包为:
seq: 0 len高: 3 len低: -5 checksum: -30
FROM S:
ack: 0 checksum: -1
当前发送数据包为:
seq: 1 len高: 3 len低: -5 checksum: 93
FROM S:
ack: 1 checksum: -2
当前发送数据包为:
seq: 0 len高: 3 len低: -5 checksum: -91

```

完成发送后，告知用户文件传输完成，并输出传输时间和吞吐率

最后进行挥手，结束连接

```

文件传输完成
本次传输所用时间为: 38.535 s.
断开连接，发送数据包为（第一次挥手）：
SYN: 0 FIN: 1 SHAKE: 4 checkSum: -6
收到来自服务器的第二次挥手：
SYN: 0 FIN: 1 SHAKE: 5 checkSum: -7
本次传输吞吐率为: 310601
请按任意键继续. . .

```

## Server接收端

输入接收端IP并等待连接(本该输入本机IP，但此处为了方便直接固定为回环地址127.0.0.1)

```

D:\Archive\大三上\计算机网络 (张建忠) \编程作业报告\2013750_管昀玫_作业3-1\server\UDPserver.exe
*****
/***** 服务器 *****/
*****
获取client的IP
Please use 'ipconfig' to get your ip and input
10.130.185.32
服务器启动成功，等待连接.....

```

三次握手成功页面

```
D:\Archive\大三上\计算机网络 (张建忠) \编程作业报告\2013750_管昀孜_作业3-1\server\UDPServer.exe
*****
***** 服务器 *****
*****
获取client的IP
Please use 'ipconfig' to get your ip and input
10.130.185.32
服务器启动成功，等待连接.....
收到建立连接的请求，数据包为：
SYN: 1 FIN: 0 SHAKE: 1      checkSum: -3
握手包正确
SYN: 1 FIN: 0 SHAKE: 2      checkSum: -4
收到数据包为：
SYN: 1 FIN: 0 SHAKE: 3      checkSum: -5
连接建立成功
```

### 接收文件页面

```
T0 C 正确返回数据包
checksum: -2 ack: 1
From C 收到的数据包为:
seq: 0 len高: 3 len低: -5 isLast: 0 checksum: -61
temp长度: 1019

T0 C 正确返回数据包
checksum: -1 ack: 0
From C 收到的数据包为:
seq: 1 len高: 3 len低: -5 isLast: 0 checksum: 18
temp长度: 1019

T0 C 正确返回数据包
checksum: -2 ack: 1
From C 收到的数据包为:
seq: 0 len高: 3 len低: -5 isLast: 0 checksum: 93
temp长度: 1019

T0 C 正确返回数据包
checksum: -1 ack: 0
From C 收到的数据包为:
seq: 1 len高: 3 len低: -5 isLast: 0 checksum: -85
temp长度: 1019

T0 C 正确返回数据包
checksum: -2 ack: 1
From C 收到的数据包为:
seq: 0 len高: 3 len低: -5 isLast: 0 checksum: -100
temp长度: 1019
```

### 文件传输完成，并自动进行挥手，结束连接

```
T0 C 正确返回数据包
checksum: -1 ack: 0
From C 收到的数据包为:
seq: 1 len高: 3 len低: -5 isLast: 0 checksum: 116
temp长度: 1019

T0 C 正确返回数据包
checksum: -2 ack: 1
From C 收到的数据包为:
seq: 0 len高: 3 len低: -5 isLast: 0 checksum: -102
temp长度: 1019

T0 C 正确返回数据包
checksum: -1 ack: 0
From C 收到的数据包为:
seq: 1 len高: 3 len低: -72 isLast: 1 checksum: 113
temp长度: 952

T0 C 正确返回数据包
checksum: -2 ack: 1
文件保存成功
数据传输完成，等待客户端退出.....
收到断开连接的请求，数据包为：
SYN: 0 FIN: 1 SHAKE: 4      checkSum: -6
挥手包正确
发送给客户端的第二次挥手：
SYN: 0 FIN: 1 SHAKE: 5      checkSum: -7
挥手结束，BYE
服务正常结束
请按任意键继续. . .
```

接收文件夹

> test

▼ ↺



1.jpg



2.jpg



3.jpg



helloworld.txt