南開大學
Nankai University

南　　开　　大　　学

网 络 空 间 安 全 学 院

计算机网络实验报告

基于 UDP 服务设计可靠传输协议并编程实现

李潇逸 2111454

年级：2021 级

专业：信息安全、法学

指导教师：张建忠

2023 年 11 月 18 日

目录

一、 实验要求

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、接收确认、超时重传等。流量控制采用停等机制，完成给定测试文件的传输。

二、 原理

(一) client 端

建立连接：发送一条带有标识 SYN 的消息，表示希望建立连接，对方收到后回复对应的 ACK，连接建立成功。当收到 ACK 后进入状态 1

发送文件：

- 消息 1，文件开始标识位 SF=1，index= 发送文件需要的消息条数-1，filelength= 最后一条消息 msg 成员的有效位数。当收到对应的 ACK 后，进入状态 2 。每条消息都是收到上一条对应的 ACK 后再发送。
- 发送文件，每条 msg 的有效长度固定为 1024，最后一条 EF=1，收到 ACK 后退回状态 1

断开连接：发送一条标识为 FIN 的消息，对方收到后回复对应的 ACK，断开连接成功，退回状态 0

client 端由于是主动发送的一端，一般情况下不会设计蓝色线条的状态跳转，但是可能会有超过重传次数情况下做出的断网处理

(二) server 端

建立连接：收到一条 SYN，回复对应的 ACK，连接建立成功，进入状态 1

发送文件：收到包含 SF、index、filelength 的消息，开始接收文件，进入状态 2。对于每条消息都返回一条 ACK 。当收到包含标志位 EF 的消息，检查 index 与收到文件消息的条数，若一致，则返回 ACK，退回状态 1

断开连接：收到包含标识为 FIN 的消息，返回 ACK，进入状态 0

(三) 停等机制

简而言之就是发现一定时间不能收到返回的 ACK 就重发

三、 代码分析

包设计包括了各种包必须的属性，以及验证包是否损坏的函数

包设计

```
1 struct message
2 {
3 #pragma pack(1)
4     u_long flag{};
5     u_short seq{};//序列号
6     u_short ack{};//确认号
7     u_long len{};//数据部分长度
```

```
    u_long num{}; //发送的消息包含几个包
    u_short checksum{};//校验和
    char data[1024]{};//数据长度
#pragma pack()
    message() {
        memset(this, 0, sizeof(message));
    }
    bool isSYN() {
        return this->flag & 1;
    }
    bool isFIN() {
        return this->flag & 2;
    }
    bool isSTART() {
        return this->flag & 4;
    }
    bool isEND() {
        return this->flag & 8;
    }
    bool isACK() {
        return this->flag & 16;
    }
    bool isEXT() {
        return this->flag & 32;
    }
    void setSYN() {
        this->flag |= 1;
    }
    void setFIN() {
        this->flag |= 2;
    }
    void setFIR() {
        this->flag |= 4;
    }
    void setEND() {
        this->flag |= 8;
    }
    void setACK() {
        this->flag |= 16;
    }
    void setEXT() {
        this->flag |= 32;
    }
    void setchecksum() {
        int sum = 0;
        u_char* temp = (u_char*)this;
        for (int i = 0; i < 8; i++)
        {
```

```
            sum += (temp[i << 1] << 8) + temp[i << 1 | 1];
            while (sum >= 0x10000)
            {//溢出
                int t = sum >> 16;  // 将最高位回滚添加至最低位
                sum += t;
            }
        }
        this->checksum = ~(u_short)sum;  // 按位取反，方便校验计算
    }
    bool corrupt() {
        // 包是否损坏
        int sum = 0;
        u_char* temp = (u_char*)this;
        for (int i = 0; i < 8; i++)
        {
            sum += (temp[i << 1] << 8) + temp[i << 1 | 1];
            while (sum >= 0x10000)
            {//溢出
                int t = sum >> 16;//计算方法与设置校验和相同
                sum += t;
            }
        }
        //把计算出来的校验和和报文中该字段的值相加，如果等于0xffff，则校验成功
        if (checksum + (u_short)sum == 65535)
            return false;
        return true;
    }
    void output() {
        cout << "checksum=" << this->checksum << ",␣len=" << this->len << endl;
    }
};
```

三次握手

三次握手

```
int beginconnect()
{
    int iMode = 1; //1：非阻塞，0：阻塞
    ioctlsocket(Client, FIONBIO, (u_long FAR*) & iMode);//非阻塞设置
    SetColor(12, 0);
    cout << "开始连接！发送第一次握手！" << endl;
    message recvMsg, sendMsg;
    sendMsg.setSYN();
    sendMsg.seq = 88;

    sendmessage(sendMsg);

```

```
    int start = clock();
    int end;
    while (true)
    {
        recvMsg = recvmessage();
        if (!recvMsg.isEXT())
        {
            end = clock();
            if (end - start > 50) {
                SetColor(12, 0);
                cout << "连接超时,请确认网络通畅和服务端启动无误后再运行本程序! " << endl;
                sendmessage(sendMsg);
                break;
            }
            continue;
        }
        if (recvMsg.isACK() && recvMsg.isSYN() && recvMsg.ack == sendMsg.seq + 1) {
            SetColor(14, 0);
            cout << "收到第二次握手!" << endl;
            break;
        }

    }
    sendMsg.setACK();
    sendMsg.seq = 89;
    sendMsg.ack = recvMsg.seq + 1;
    SetColor(14, 0);
    cout << "发送第三次握手的数据包" << endl;
    sendmessage(sendMsg);
    return 0;
}
```

四次挥手

四次挥手

```
iint closeconnect() {  // 断开连接
    message recvMsg, sendMsg;
    sendMsg.setFIN();
    sendMsg.seq = 65534;//此处是u_short的表示范围的最大值-1, 而我们收到的将会再加一, 那么已经到了u_short的最大值了, 就自然结束了。
    sendmessage(sendMsg);
    cout << "发送出去第一次挥手! " << endl;
    int count = 0;
    while (true) {
        Sleep(100);
        if (count >= 50) {
            SetColor(12, 0);
```

```
            cout << "等待时间太长，退出连接" << endl;
            return closeconnect();
        }
        recvMsg = recvmessage();
        if (!recvMsg.isEXT()) {
            continue;
        }
        if (recvMsg.isACK() && recvMsg.ack == sendMsg.seq + 1) {
            break;
        }
        count++;
    }
    SetColor(12, 0);
    cout << "接收到确认连接，断开连接成功" << endl << endl;
    return 0;
}
```

四、 展示

图 1: Caption

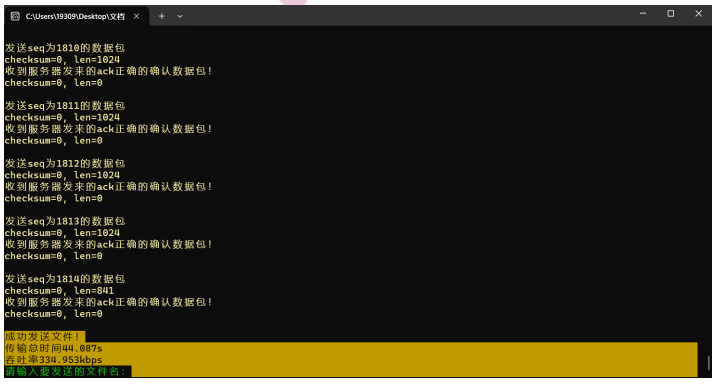

图 2: client 端

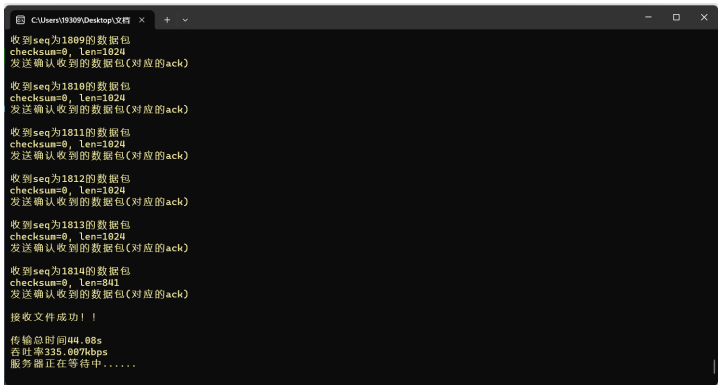

图 3: server 端

图 4: 路由器显示

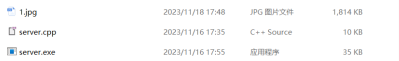

图 5: 结果

五、 总结

实现了 UDP 传输，将研究如何加快传输速度