# 计算机网络实验3-3报告

学号：2213034

姓名：辛杰

专业：物联网工程



## 一、实验目的

在实验3-2的基础上，选择实现一种拥塞控制算法，也可以是改进的算法，完成给定测试文件的传输

## 二、协议设计

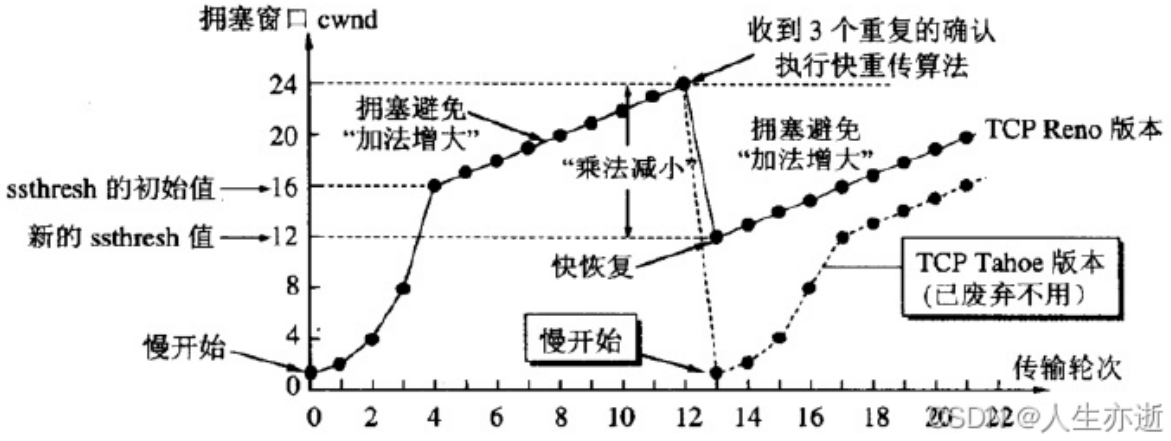
### 1、报文格式


| 标识 | 含义 |
| --- | --- |
| 源端口号 | 发送此报文的终端所对应的端口号 |
| 目的端口号 | 接收此报文的终端所对应的端口号 |
| 源IP | 发送此报文的终端IP地址 |
| 目的IP | 接收此报文的终端IP地址 |
| 初始序号seq | 初始序号值，代表当前报文状态，也用于去除冗余 |
| 确认序号ack | 与seq相应，用以进行回复 |
| 数据段长度Length | 代表本次报文中数据段有效信息的长度 |
| 接收窗口大小 | 可以接受或缓存的数据包的数量 |
| ACK | 标志位，接收到上一次报文 |
| SYN | 标志位，申请建立连接 |
| FIN | 标志位，申请断开连接 |
| ST | 标志位，本次报文为一个文件的首个报文 |

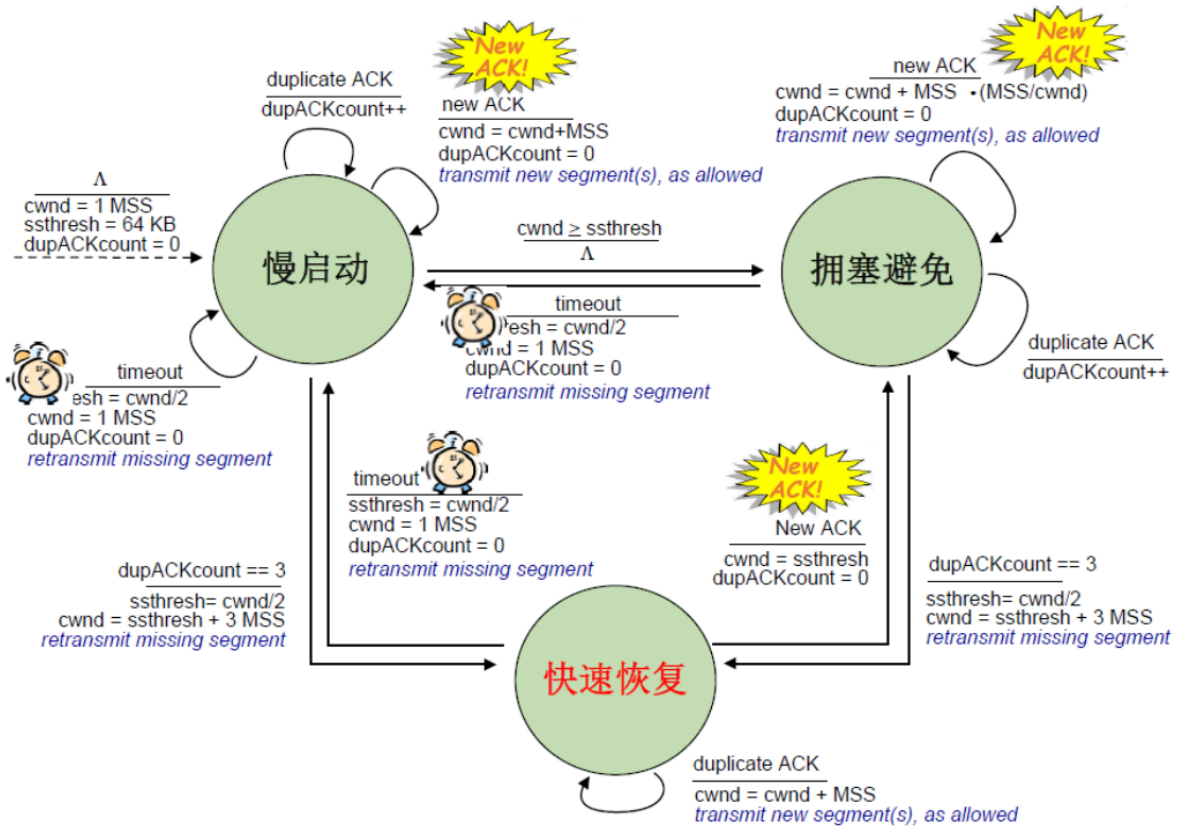
| 标识 | 含义 |
| --- | --- |
| OV | 标志位，本次报文为当前文件的最后一个报文 |
| 校验和 | 存放校验和。 |

## 2、RENO算法

**Slow Start（慢启动）**
slow start 慢启动算法
初始用塞窗口设置为1，慢启动阀值设置为ssthresh，此时可以传1个MSS大小的数据，每收到一个ACK，用塞窗口cwnd加一，这样每过一个RTT，cwnd翻倍，指数上升。达到ssthresh进入用塞避免阶段
**拥塞避免算法**
每经过一个RTT（往返时间）拥塞窗口cwnd加1，相当于进行了一个可发送窗口的线性探测（AI）
**快速重传算法**
sender 连续收到3个相同的ACK，立即重传，设置新的慢启动门限为ssthresh = cwnd / 2
**快速恢复算法**
快速重传后立即执行快速恢复算法，设置cwnd = ssthresh = cwnd / 2 ，然后进行拥塞避免算法（MD）
没有慢启动的过程。其中cwnd窗口呈现周期性变化。




(注：图片取自网上)




# 三、核心代码分析

## 1、慢启动

```
if (state == SS)
{
    windowLength++;
    ConfirmCount = 0;
    if (windowLength > ssthresh)
    {
        state = CA;
    }
}
```

每收到一个ack，接收窗口大小加1，实现指数增长

窗口大于慢启动阈值，进入拥塞避免阶段

## 2、拥塞避免

```
else if (state == CA)
{
    //拥塞避免阶段
    ConfirmCount = 0;
    newAckCount++;
    if (newAckCount >= windowLength)
    {
        newAckCount = 0;
        windowLength++;
    }
}
```

每接收到完整窗口的确认 (ACK)，将发送窗口大小增加 1，窗口的增长速度从指数变为线性

## 3、快速恢复

```
else if (state == QR)
{
    windowLength = ssthresh;
    ConfirmCount = 0;
    state = CA;
}
```

窗口大小变为慢启动阈值，进入拥塞避免

## 4、快速重传

```
if (recvack == lastConfirmAck)
{
        ConfirmCount++;
        if (ConfirmCount == 3)
        {
            //三次ack重复，重传窗口内第一个包
            cout << "快速恢复" << endl;
            ssthresh = windowLength / 2;
            if (ssthresh == 0)
                ssthresh++;
            windowLength = ssthresh + 3;
            CubicRepetition();
            state = QR;
        }
 }

 void CubicRepetition()
{
    // 遍历发送窗口中的所有数据包
    slidingWindow[0].clock = clock();
    for (size_t i = 0; i < slidingWindow.size(); ++i)
    {
        sendto(sockSrv, slidingWindow[i].buffer,
sizeof(slidingWindow[i].buffer), 0, (SOCKADDR*)&Server, len);
```

```
            sendlog(slidingWindow[i].buffer);
        }
    }
```

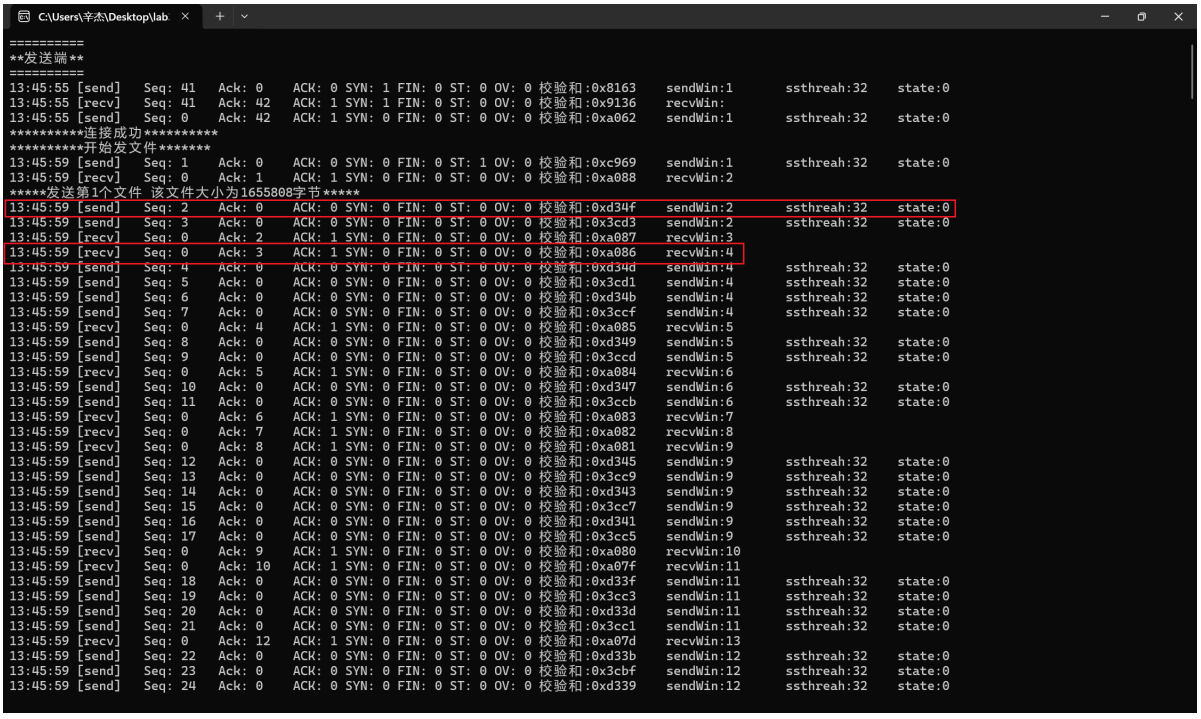
如果重复收到3次最后确认的ack，进入快速重传，将滑动窗口里的数据包统统发送

# 四、程序展示

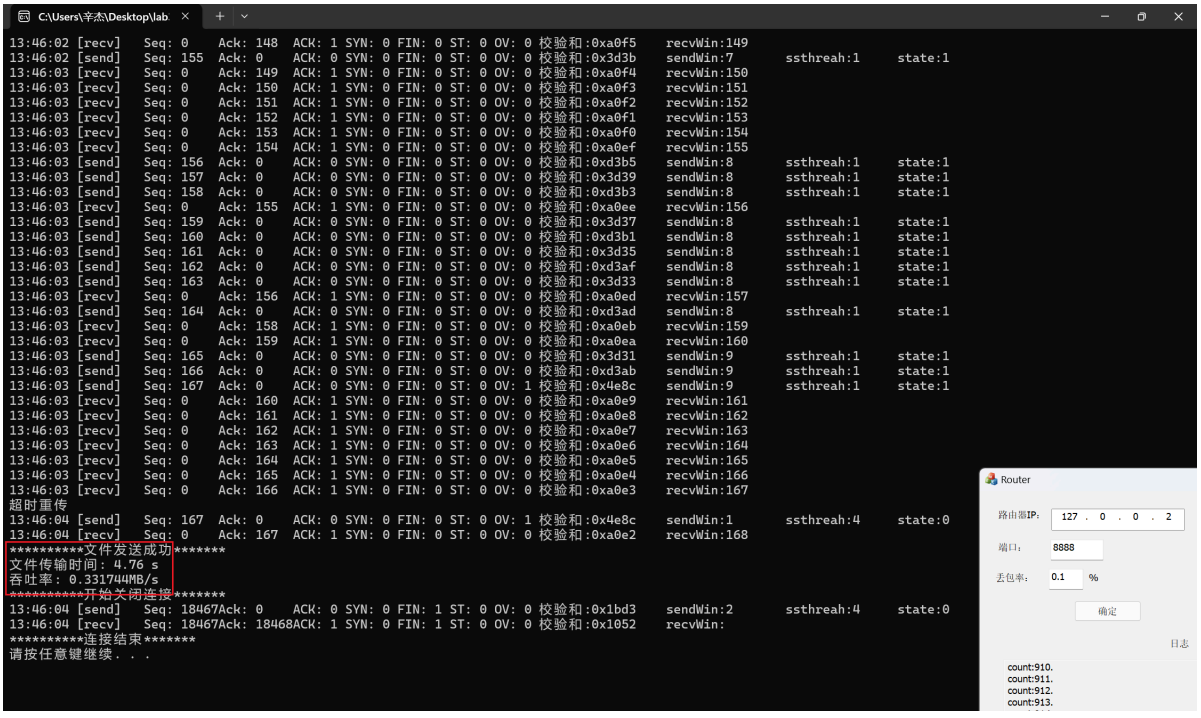
## 1、日志输出

如图所示，可以看到Seq、ACK、标志位、校验和、发送窗口大小、接收窗口、慢启动阈值、当前状态

> 接收窗口大小为1，这里的接收窗口的值是希望接收下一个的ack，即接收窗口的位置
>
> 当前状态有0、1、2，分别代表SS（慢启动）、CA（拥塞避免）、QR（快速恢复）




可以看到**文件传输速率和吞吐率**

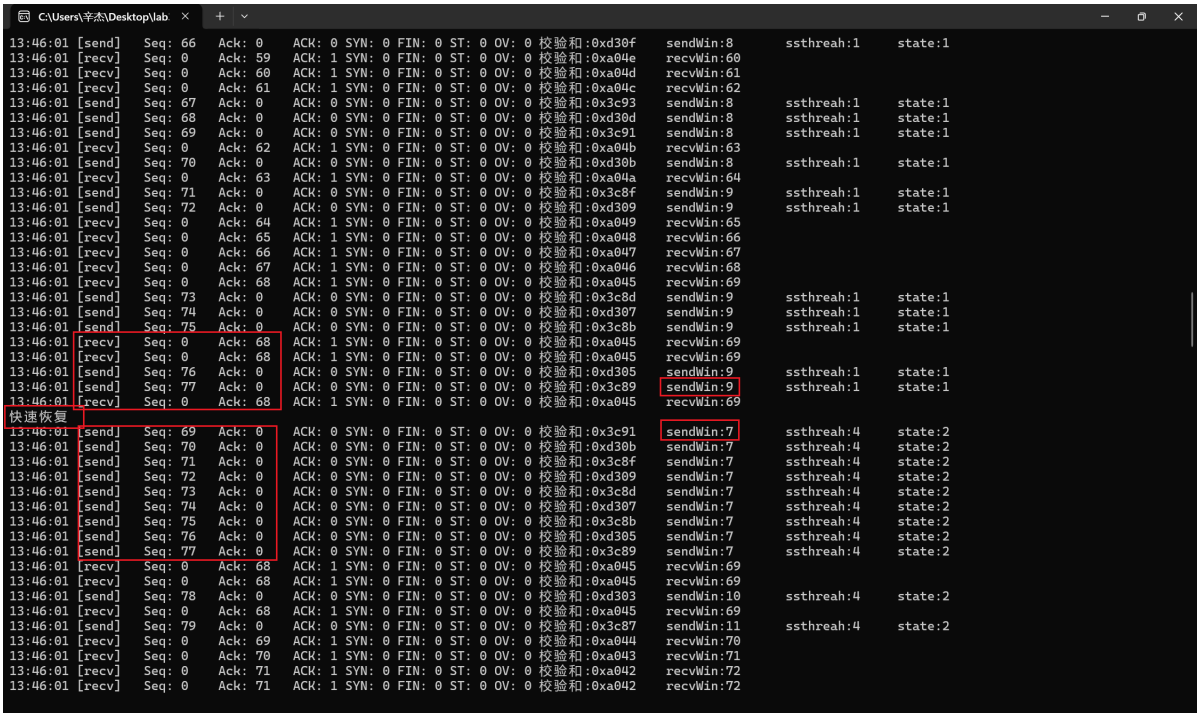


## 2、快速恢复

如图所示，重复收到68的ack，说明seq为69的包发生丢失，快速重传发送窗口的所有数据包

发送窗口大小也从9变为7（9/2+3=7）

状态也从1到2 （拥塞避免到快速恢复）




## 3、超时重传

但快速恢复重发发送窗口的所有数据包的时候，再次发送丢包就会发生超时重传

如图所示快速恢复的时候Seq为21的包丢失，发生超时重传，重传当前发送窗口的所有数据包

发送窗口变为1，慢启动阈值为原来的一半