设计思路

要实现RENO，就是在3-2的基础上加上一些状态，并根据所处状态和收到的ack/超时情况改变滑动窗口大小。

注：SS = 慢启动阶段，QR = 快速恢复阶段，CA = 拥塞避免阶段

1. 开始阶段：进入SS, threshold = 30, cwnd = 1，收到一个ack则cwnd = cwnd + 1 MSS
2. 遇到超时：threshold = cwnd / 2, cwnd = 1 MSS, 进入SS，准备重传
3. 遇到三次重复确认ack：进入QR, threshold = cwnd / 2, cwnd = threshold + 3 MSS，准备重传
4. 收到新、正确的ack
    - 如果是SS状态下，cwnd = cwnd + 1 MSS; 如果cwnd > threshold, 进入CA
    - 如果是CA状态下，cwnd = cwnd + MSS * (MSS / cwnd)
    - 如果是QR状态下，cwnd = threshold, 进入CA
5. 收到旧ack（也就是重复的ack）
    - 如果是非QR状态下，等待三次，然后进入QR
    - 如果是QR状态下，cwnd + 1 MSS

代码实现

首先是开始阶段，在每次执行sendfile()函数发送单个文件前额外初始化一些与拥塞控制相关的变量即可：

```
void sendfile(const char* filename) {
    // 其他准备工作...
    status = SS_STATUS; // 开始时是SS状态
    threshold = INIT_SW_SIZE; // INIT_SW_SIZE扩展到30
    cwnd = 1;
    dup_ack_times = 0;
    // 发送文件...（首先会检查是否需要重传）
}
```

接下来是遇到超时，按照思路写代码即可：

```
void timerThread() {
    while(!THREAD_END) {
        last = clock();
        if (last - start >= TIMEOUT) {
            // 如果超时，调整threshold和cwnd，进入SS
            threshold = cwnd / 2;
            cwnd = 1;
            status = SS_STATUS;
            dup_ack_times = 0;
            if (!finishSend) // 如果文件没发完，就输出一下Timeout的日志
                cout << "Timeout! Now threshold = " << threshold << ", cwnd = "
<< cwnd << " * MSS." << endl;

            start = clock();
            resend = true; // 准备重传window中的包
```

```
        }
    }
}
```

接下来是在recvRespondThread()中收到ack后根据当前所处状态和重复ack次数做出不同的反应：

```
void recvRespondThread() {
    // 接受响应报文的线程
    // 接受并处理receive端发过来的ack...
    if (recvBuf[FLAG_BIT_POSITION] == 0b100 && ack_opp >= base) {
        // 收到了正确的响应报文，则根据status改变cwnd，并移动滑动窗口的base到下一个需要确认
的包
        base = ack_opp + 1; // 移动滑动窗口
        resend = false;
        index = 0;

        // 根据status改变cwnd
        if (status == SS_STATUS) {
            cwnd++;
            // 判断cwnd是否超过threshold，要不要进入CA
            if (cwnd > threshold) {
                status = CA_STATUS;
            }
        } else if (status == CA_STATUS) {
            cwnd += 1 / ((cwnd > 1) ? floor(cwnd) : 1); // 这里这么写是为了避免除以0
        } else if (status == QR_STATUS) {
            status = CA_STATUS;
            cwnd = threshold;
        }

        dup_ack_times = 0;
        cout << "Having received the correct ack = " << ack_opp << ", now
threshold = " << threshold << ", cwnd = " << cwnd << " * MSS." << endl;

        // 启动计时（实际上是重置计时器）
        start = clock();
    } else {
        // 如果接受到的ack < base，即错误的ack，则等待收到三次重复的ack
        dup_ack_times++;
        cout << "dup_ack_times = " << dup_ack_times << endl;

        // 如果已经到了三次，则进入QR状态，并调整threshold, cwnd，并准备进行重传
        if (status != QR_STATUS && dup_ack_times == 3) {
            status = QR_STATUS;
            threshold = cwnd / 2;
            cwnd = threshold + 3;

            resend = true; // 准备重传
            std:cout << "Enter the QR status!" << endl;
        } else if (status == QR_STATUS) {
            // 如果是在快速恢复阶段收到了重复ack，说明网络状况良好
            cwnd++;
        }
```

```
        cout << "Received the wrong ack = " << ack_opp << ", now threshold = "
<< threshold << ", cwnd = " << cwnd << " * MSS." << endl;
    }
}
```