实验 3：基于 UDP 服务设计可靠传输协议并编程实现（任务 3-3）

姓　名: 杨科迪　　学　　号: 1813828
专　业: 计算机科学与技术　　指导老师: 徐敬东

实验时间：2020 年 12 月 18 日　　实验地点：实验楼 A 区 204

目录

1 实验目的

1. 实现拥塞控制算法
2. 完成给定测试文件的传输

2 实验原理

本实验实现的是 RENO 算法，RENO 算法的状态机如图1所示。

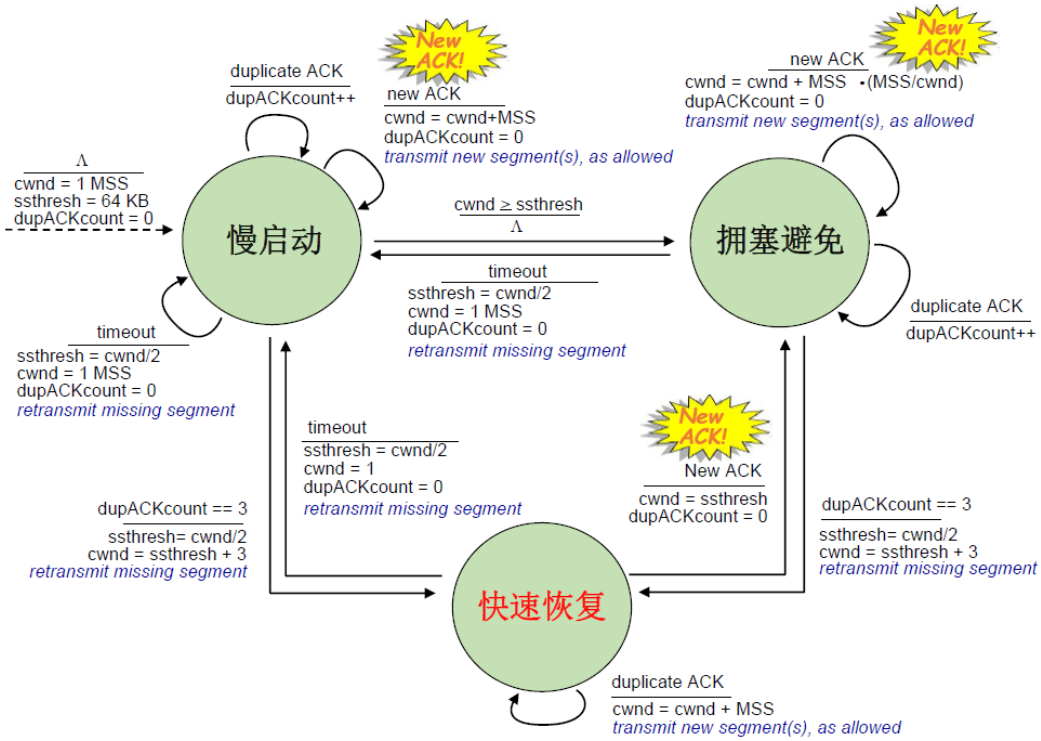

图 1: RENO 算法状态机

1. 起始阶段为慢启动阶段，每个 RTT，cwnd 翻倍，即每收到 1 个 ack，cwnd 增 1
2. 当 cwnd>=ssthresh 后，进入拥塞避免阶段，每个 RTT，cwnd 增 1
3. 收到 3 次重复 ack 后，进入快速恢复阶段，ssthresh = cwnd/2，cwnd = ssthresh+3
4. 在快速恢复阶段，每收到 1 个重复 ack，cwnd 增 1，收到新的 ack 后，进入拥塞避免阶段，cwnd=ssthresh
5. 超时后，进入慢启动状态，ssthresh=cwnd/2，cwnd=1

3 实验步骤

3.1 协议设计

本实验使用的协议同任务 3-2:

```
//传输层协议
struct packet
{
    unsigned int seq;                   //序列号
    unsigned int ack;                   //确认号
    unsigned short N;                   //滑动窗口长度
    unsigned char flag;                 //标志字段 (ACK,SYN,FIN) 0:fin 1:syn 2:ack
    unsigned short checkSum;            //检验和
    unsigned short dataLen;             //数据长度，以字节为单位
    unsigned char data[MAXBUFSIZE];     //应用层数据
};
```

各字段的含义如下：

seq 序列号

ack 确认号

N 滑动窗口大小，用于发送端通告接收端，接收端分配接收缓冲区大小

flag ACK、SYN、FIN 标志字段

checkSum 检验和字段

datalen 数据长度

data 应用层数据

3.2 拥塞控制算法

本实验在任务 3-2 的基础上增加了如下变量：

```
    int cwnd;                           //拥塞窗口
    int state;                          //慢启动阶段:1 拥塞避免阶段:2 快速恢复阶段:3
    int cnt;                            //计数
    int ssthresh;                       //拥塞阈值
```

cwnd 为拥塞窗口大小；state 为状态机所处的状态，1 为慢启动状态，2 为拥塞避免阶段，3 为快速恢复阶段；cnt 用于拥塞避免阶段 ack 计数，每当 cnt=cwnd 时，cwnd 增 1；ssthresh 为拥塞阈值。

```
//接受 ack 数据报
void SBuf::ack()
{
    /******/
    if (check(&p) && isAck(p.flag))
    {
        if(p.ack >= base && p.ack < nextSeqnum) //为一个新的 ack
```

```
8           {
9               /******/
10              dupAckCnt = 0;
11              int num = p.ack - base + 1;
12              if(state == 1)                          //慢启动状态
13              {
14                  cwnd += num;                        //收到 1 个 ack, 拥塞窗口加 1;
15                  if(cwnd >= ssthresh)                //cwnd>=ssthresh, 进入拥塞避免阶段
16                  {
17                      cnt = 0;
18                      state = 2;
19                  }
20              }
21              else if(state == 2)                     //拥塞避免阶段
22              {
23                  cnt += num;
24                  if (cnt >= cwnd)                    //每收到 cwnd 个 ack, cwnd 加 1
25                  {
26                      cwnd += 1;
27                      cnt = 0;
28                  }
29              }
30              else if(state == 3)                     //快速恢复阶段
31              {
32                                                      //进入拥塞避免阶段
33                  cwnd = max(ssthresh, 1);
34                  state = 2;
35              }
36              base = p.ack + 1;                       //base = ack + 1
37              ReleaseSemaphore(slots, num, NULL);
38          }
39          else                                        //为重复 ack
40          {
41              if(++dupAckCnt == 3)                   //三次重复 ack, 快速重传
42              {
43              /***sendpkt***/
44              dupAckCnt = 0;
45              if(state != 3)
46              {
47                  state = 3;                          //进入快速恢复阶段
```

```
                ssthresh = cwnd / 2;
                cwnd = ssthresh + 3;
            }
            }
            if(state == 3)
                cwnd += 1;
        }

    }
    ReleaseSemaphore(mutex, 1, NULL);
}
```

若收到的 ack 不是重复的：

1. state=1: 每收到 1 个 ack，cwnd 增 1，cwnd>=ssthresh 时进入拥塞避免阶段
2. state=2: 每收到 cwnd 个 ack，cwnd 增 1
3. state=3:cwnd=ssthresh, 进入拥塞避免阶段

若收到重复的 ack:

1. 处于慢启动阶段或拥塞避免阶段，收到 3 次重复 ack 后，重传数据包，ssthresh = cwnd/2,cwnd=ssthresh+3, 进入快速恢复阶段
2. 处于快速恢复阶段，收到重复 ack，cwnd 增 1

```
//超时重传
void SBuf::retrans()
{
    WaitForSingleObject(mutex, INFINITE);
    for (unsigned int i = base; i < nextSeqnum; i++)        //定时器超时，重传 [base, nextSeqnum) 中的数
    {
        if(timer[i % n].isStart && timer[i % n].testTimeOut())
        {
            if(i == base)
            {
                dupAckCnt = 0;                              //进入慢启动阶段
                ssthresh = cwnd / 2;
                cwnd = 1;
                state = 1;
                /***sendpkt***/
            }
            /******/
        }
    }
```

```
20      ReleaseSemaphore(mutex, 1, NULL);
21  }
```

超时后，不管处于哪个阶段，均进入慢启动阶段，cwnd=1，ssthresh = cwnd / 2。

4 实验结果

对给定测试文件进行了 20 次传输，滑动窗口大小为 32，测试结果如表1所示。

文件名称	平均传输时间 (ms)	平均吞吐率 (kbps)
1.jpg	847.85	48935.6
2.jpg	846.79	57514.1
3.jpg	1888.81	53881.8
helloworld.txt	324.97	50779.8
		52777.8

表 1: 测试结果