计算机网络LAB1——滑动窗口协议实验
1400012849 李岩昊

1. 实验概述

在一个数据链路层的模拟实现环境中，用 C 语言实现下面三个数据链路层协议。

1) 比特滑动窗口协议
2) 回退 N 帧滑动窗口协议
3) 选择性重传协议

充分理解滑动窗口协议，根据滑动窗口协议，模拟滑动窗口协议中发送端的功能，对系统发送的帧进行缓存并加入窗口等待确认，并在超时或者错 误时对部分帧进行重传。

编写停等及退回 N 滑动窗口协议函数，响应系统的发送请求、接收帧消息以及超时消息，并根据滑动窗口协议进行相应处理。

编写选择性重传协议函数，响应系统的发送请求、接受帧消息以及错误消息，并根据滑动窗口协议进行相应处理。

2. 实验实现思路

在我的理解里，整个滑动窗口协议所基于的数据结构是不会改变的。比特滑动窗口协议与回退 N 帧滑动窗口协议、选择性重传协议的区别仅仅在于窗口的大小不同。而选择性重传协议和比特滑动窗口协议、回退 N 帧滑动窗口协议的区别仅在于重新发送损坏的Frame时选择Frame的策略不同。因此在设计程序底层数据结构的时候都是一样的，处理MSG_TYPE_SEND和ACK Frame的做法也是一样的。

1. 底层数据结构：一个输入缓冲区用deque实现，一个unsigned int用于记录窗口大小。

我们注意到输入缓冲区和窗口的特点都是一个队列，而且两个队列的头部是可以重复利用的——即输入缓冲区的队头部分可以直接作为发送窗口。其中队列储存的数据结构定义如下，len是Frame的数据大小：

```
typedef struct store{
    frame* p;
    unsigned int len;
};
```

2.MSG_TYPE_SEND的处理方法：

```
if (messageType == MSG_TYPE_SEND){
    // save new frame in queue
    ns.p = new frame;
    *ns.p = *((frame*)pBuffer);
    ns.len = bufferSize;
    qbuff1.push_back(ns);
    // window not full
    if (windowSize1 < WINDOW_SIZE_STOP_WAIT){
        ns = qbuff1[windowSize1++];
        SendFRAMEPacket((unsigned char*)(ns.p), ns.len);
    }
}
```

制造好全新的Frame并推入队尾，判断窗口大小和窗口大小上限的关系并将队头不在窗口中的Frame放入窗口中，窗口大小+1。

3.处理ACK的做法：

```
else if (messageType == MSG_TYPE_RECEIVE){
    unsigned int tmpack = ntohl(((frame*)pBuffer)->head.ack);
    for (int i = 0; i < windowSize2; ++i){
        if (tmpack == ntohl(qbuff2[i].p->head.seq)){
            for (int j = 0; j <= i; ++j){
                qbuff2.pop_front();
                windowSize2--;
                if (qbuff2.size() > windowSize2 && windowSize2 < WINDOW_SIZE_BACK_N_FRAME){
                    ns = qbuff2[windowSize2++];
                    SendFRAMEPacket((unsigned char*)(ns.p), ns.len);
                }
            }
        }
    }
}
```

判断队列里是否有Frame符合ACK，将其及其之前的Frame全部出队。判断缓存区等待进入窗口的Frame数量是否足够，若足够则将其发送并纳入发送窗口。

4.处理NAK时的做法：

```
else if (messageType == MSG_TYPE_RECEIVE && ntohl(((frame*)pBuffer)->head.kind) == nak){
    unsigned int naknum = ntohl(((frame*)pBuffer)->head.ack);
    for (unsigned int i = 0; i < windowSize3; ++i){
        if (naknum == ntohl(qbuff3[i].p->head.seq)){
            SendFRAMEPacket((unsigned char*)(qbuff3[i].p), qbuff3[i].len);
            break;
        }
    }
}
```

遍历窗口中的所有Frame判断是否有符合NAK的编号，将其发送即可。

5.处理超时的做法：直接将窗口中的Frame全部发送一遍。

3. 实验中遇到的困难

1. 测试样例中Frame的编号从1开始而并非0，对一开始的编写造成了不少困扰，经过Debug之后发现了这个小问题。
2. 对于回退N协议中的MSG_TYPE_RECEIVE，我一开始的理解有误。认为ACK带回来的编号一定是发送窗口中最早发送的Frame——例如发送窗口中停留着1、2、3、4四个Frame，那么ACK一定不会是2、3、4。但在实际测试样例中会直接对2进行ACK，经过Debug之后这个问题加深了我对滑动窗口协议的理解。
3. 对于超时状态将全部未确认的Frame发送，在实现协议的时候更加深刻地理解了这个做法的好处和对带宽的浪费。之前错误的理解是重复发送不会再次发送已经超时的Frame，具体在测试样例中就是将顺序123456 3456 3456理解为123456 3456 456。

4. 附录代码

见提交tar包中附的代码以及在线测试系统上的代码