**Lab 1 - Reliable Data Transport Protocol**

517021911052

周佳懿

[chris98122@sjtu.edu.cn](mailto:chris98122@sjtu.edu.cn)

**Design**

**帧定界**

因为是模拟link layer 到physical layer，所以一个frame和一个packet能够对应起来，所以帧定界这一个步骤就被省略了。

**frame header**

包含payload size，kind，isend，seq, ack。

Kind分DATA和ACK，占1bit。

isend d表示该packet是不是message的最后一个包，占1bit。

Seq和ack占6bit,这使得最大的seq可以支持2^6-1= 63，不过我设置MAX\_SEQ为7。由于目前的协议不需要支持piggyback ack，所以我根据frame的kind来决定这6个bit存储的是seq还是ack，从而实现最大的空间利用率。

**error detection**

使用CRC32 ,生成多项式为

receiver发现corruption就直接扔掉该包，然后等待sender超时重传。

**packet的数据分布**

| **kind,isend,seq,ack** | **payload size** | **payload** | **CRC8** |
| --- | --- | --- | --- |
| **1byte** | **1 byte** | **最多122byte。最少1byte。** | **4 byte** |

**Retransmission**

选择性重传（selective repeat）

选择性重传接收方需要较大的缓冲区（以用来排序），适合出错率高的信道。我基于降低packet的数量的考虑，在本次lab中使用的是选择性重传机制。

重传的包需要满足两个条件：

1. 当前时间减去发送时间大于TIMEOUT的值
2. 未收到ACK

**Receiver的接收机制**

接收方对于无差错的数据包进行正常接收，对于有差错数据包进行丢弃，并使用缓冲区来保存乱序的packet，在收到所有packet时候进行重组，把完整的一条message发给upper layer。

**Implementation**

**timer**

使用double数组保存每个packet的发送时间。

**Timeout 机制**

发送一个包时同时检查Sender\_isTimerSet()。

如果目前timer没有设置的话，说明没有等待ACK的packet, 于是为新发的packet设置一个计时器，调用starttimer(TIMEOUT)。

如果当前Timer已经设置了，说明还有等待ACK的packet，则不再设置计时器，而是等待计时器timeout再处理。 因为先发的packet会先超时，如果先发的packet正常接收到ACK了，那么等到计时器到时，回调timeout()函数，调用starttimer(time\_interval)来重新计时。

计时器的停止：

如果所有的packet都接收到了ACK，那么在timeout()中不再调用starttimer()，从而计时器停止。

**sender window**

滑动窗口长度为MAX\_SEQ+1。

通过window\_delete(seq\_nr seq)和window\_add(frame \*frm, seq\_nr seq)两个接口操作。

**Buffer for message**

使用STL的queue保存还未发送的Message, 使得upper layer的throughput 大于链路层throughput的时候message不会丢失。

通过msgbuffer\_push\_back()和msgbuffer\_pop\_front()两个接口操作。

**Buffer in receiver**

receive\_content保存乱序的packet

**Protocol的问题与解决**

在某些packet到达receiver的时候已经属于过期的packet（即对应的message已经组装好发送给upper layer）的情况下，receiver会将该过期的packet视作新的message的packet,并加入buffer。这就会导致message的错误。

我的解决方法是receiver需要判断一下packet的序号与last\_end的关系，如果seq不属于应该有的序号范围，则视作是一个过期的packet,直接丢弃，并不发送ACK。

**Performance**

rdt\_sim 1000 0.1 100 0.02 0.02 0.02 0 - there could be packet loss, corruption,

or reordering in the underlying link medium.

rdt\_sim 1000 0.1 100 0.3 0.3 0 0