学号：518021910269

姓名：胡昊源

邮箱：huhaoyuan@sjtu.edu.cn

一、对 rdt_sender.h , rdt_receiver.h , rdt_struct.h , rdt_sim.cc 做出的修改

没有做出什么修改。

二、我对 makefile 做出的修改

没有做出什么修改。

三、我的设计

1、策略选择和参数设置

我使用 GO-BACK-N 的实现方法；

我设置 MAX_SEQ 为 10，TIME_OUT 为 0.3；

我使用校验和 checksum 来实现检错功能；

```
short Internet_Checksum(struct packet *pkt) {
    unsigned long checksum = 0;
    // 前两个字节为 checksum
    for (int i = 2; i < RDT_PKTSIZE; i += 2) {
        checksum += *(short *)(&(pkt->data[i]));
    }
    while (checksum >> 16) {
        checksum = (checksum >> 16) + (checksum & 0xffff);
    }
    return ~checksum;
}
```

2、GO-BACK-N 的优化

receiver 会维护一个长为 MAX_SEQ 大小的 receiver_buffer，用来存储当前这个窗口下的对应的包。这是为了保证乱序情况下我们能够继续进行包的收发；

3、Sender 的收发包策略

- 收包机制
    1. 对于 upper 来的包，sender 维护一个 message_buffer，每次从上层接收到要发出去的消息后就先存入该缓存中，然后判断目前 sender 是否在发包（即检查 timer 是否正在计时），若没有则开始发包过程，若正在发包则结束收包工作。

2. 对于 lower 来的包，sender 会首先检测收到的 ack/seq 是否是自己目前想要的数据，如果是，go back n 中的窗口右移。之后重新计时，开始新一轮的消息切割和发送数据包的操作。

- 发包机制：
  1. sender 会首先调用 chunk_message 函数对当前 message_buffer 中所有等待发送的消息全部切割成小数据包，并按顺序存储在包的队列中。
  2. 之后 sender 每次从包的队列中取前 MAX_SEQ 个包（即当前窗户中的包）发给下层。
  3. 若计时器超时，说明仍然没有收到 ACK，因此重新计时，然后重发当前窗口中的包。

4、receiver 的收发包策略

- 收包机制
  1. 对于 lower 来的包，receiver 将恰好落在当前窗口的数据包放入 receiver_buffer 中，若接收到了想要的包，则会直接解析包的数据并写入相应的 message 位置，同时发回 ACK 包。
- 发包机制
  1. 直接向 lower 发送 ACK 包即可

5、包的结构设计

- sender 发出的数据包分为两种
  1. 如果这个数据包恰好是一个消息的第一个包，则 payload 的前四个 byte 表示一个整型，表示这个 message 的大小，方便receiver 合并包的数据重构消息。

     ```
     |<-  2 byte  ->|<-  4 byte  ->|<-  1 byte  ->|<-  4 byte  ->|<-  the
     rest  ->|
     |   checksum   |  packet seq  | payload size | message size |
     payload    |
     ```

  2. 如果这个数据包不是一个消息的第一个包，则 payload 正常表示，不包含消息大小信息。

     ```
     |<-  2 byte  ->|<-  4 byte  ->|<-  1 byte  ->|<-  the rest  ->|
     |   checksum   |  packet seq  | payload size |     payload    |
     ```

- receiver 会向下层发出ACK包。

  ```
  |<-  2 byte  ->|<-  4 byte  ->|<-  the rest  ->|
  |   checksum   |    ack seq   |   meaningless  |
  ```

四、测试结果

1、正确性测试

```
haoyuan@sjtu-huhaoyuan:~/lab_internet/rdt$ ./rdt_sim 1000 0.1 100 0 0 0 0
## Reliable data transfer simulation with:
        simulation time is 1000.000 seconds
        average message arrival interval is 0.100 seconds
        average message size is 100 bytes
        average out-of-order delivery rate is 0.00%
        average loss rate is 0.00%
        average corrupt rate is 0.00%
        tracing level is 0
Please review these inputs and press <enter> to proceed.

At 0.00s: sender initializing ...
At 0.00s: receiver initializing ...
At 1000.40s: sender finalizing ...
At 1000.40s: receiver finalizing ...

## Simulation completed at time 1000.40s with
        1003678 characters sent
        1003678 characters delivered
        28282 packets passed between the sender and the receiver
## Congratulations! This session is error-free, loss-free, and in order.
```

2、乱序测试

```
haoyuan@sjtu-huhaoyuan:~/lab_internet/rdt$ ./rdt_sim 1000 0.1 100 0.02 0 0 0
## Reliable data transfer simulation with:
        simulation time is 1000.000 seconds
        average message arrival interval is 0.100 seconds
        average message size is 100 bytes
        average out-of-order delivery rate is 2.00%
        average loss rate is 0.00%
        average corrupt rate is 0.00%
        tracing level is 0
Please review these inputs and press <enter> to proceed.

At 0.00s: sender initializing ...
At 0.00s: receiver initializing ...
At 1000.45s: sender finalizing ...
At 1000.45s: receiver finalizing ...

## Simulation completed at time 1000.45s with
        987488 characters sent
        987488 characters delivered
        27942 packets passed between the sender and the receiver
## Congratulations! This session is error-free, loss-free, and in order.
```

3、丢包测试

```
haoyuan@sjtu-huhaoyuan:~/lab_internet/rdt$ ./rdt_sim 1000 0.1 100 0 0.02 0 0
## Reliable data transfer simulation with:
        simulation time is 1000.000 seconds
        average message arrival interval is 0.100 seconds
        average message size is 100 bytes
        average out-of-order delivery rate is 0.00%
        average loss rate is 2.00%
        average corrupt rate is 0.00%
        tracing level is 0
Please review these inputs and press <enter> to proceed.

At 0.00s: sender initializing ...
At 0.00s: receiver initializing ...
At 1000.50s: sender finalizing ...
At 1000.50s: receiver finalizing ...

## Simulation completed at time 1000.50s with
        993899 characters sent
        993899 characters delivered
        29727 packets passed between the sender and the receiver
## Congratulations! This session is error-free, loss-free, and in order.
```

4、包损毁测试

```
haoyuan@sjtu-huhaoyuan:~/lab_internet/rdt$ ./rdt_sim 1000 0.1 100 0 0 0.02 0
## Reliable data transfer simulation with:
        simulation time is 1000.000 seconds
        average message arrival interval is 0.100 seconds
        average message size is 100 bytes
        average out-of-order delivery rate is 0.00%
        average loss rate is 0.00%
        average corrupt rate is 2.00%
        tracing level is 0
Please review these inputs and press <enter> to proceed.

At 0.00s: sender initializing ...
At 0.00s: receiver initializing ...
At 1000.70s: sender finalizing ...
At 1000.70s: receiver finalizing ...

## Simulation completed at time 1000.70s with
        987281 characters sent
        987281 characters delivered
        30262 packets passed between the sender and the receiver
## Congratulations! This session is error-free, loss-free, and in order.
```

5、综合测试

```
haoyuan@sjtu-huhaoyuan:~/lab_internet/rdt$ ./rdt_sim 1000 0.1 100 0.02 0.02 0.02
 0
## Reliable data transfer simulation with:
        simulation time is 1000.000 seconds
        average message arrival interval is 0.100 seconds
        average message size is 100 bytes
        average out-of-order delivery rate is 2.00%
        average loss rate is 2.00%
        average corrupt rate is 2.00%
        tracing level is 0
Please review these inputs and press <enter> to proceed.

At 0.00s: sender initializing ...
At 0.00s: receiver initializing ...
At 1000.65s: sender finalizing ...
At 1000.65s: receiver finalizing ...

## Simulation completed at time 1000.65s with
        996004 characters sent
        996004 characters delivered
        31758 packets passed between the sender and the receiver
## Congratulations! This session is error-free, loss-free, and in order.
```

```
haoyuan@sjtu-huhaoyuan:~/lab_internet/rdt$ ./rdt_sim 1000 0.1 100 0.15 0.15 0.15
 0
## Reliable data transfer simulation with:
        simulation time is 1000.000 seconds
        average message arrival interval is 0.100 seconds
        average message size is 100 bytes
        average out-of-order delivery rate is 15.00%
        average loss rate is 15.00%
        average corrupt rate is 15.00%
        tracing level is 0
Please review these inputs and press <enter> to proceed.

At 0.00s: sender initializing ...
At 0.00s: receiver initializing ...
At 1039.18s: sender finalizing ...
At 1039.18s: receiver finalizing ...

## Simulation completed at time 1039.18s with
        1001198 characters sent
        1001198 characters delivered
        50911 packets passed between the sender and the receiver
## Congratulations! This session is error-free, loss-free, and in order.
```

```
haoyuan@sjtu-huhaoyuan:~/lab_internet/rdt$ ./rdt_sim 1000 0.1 100 0.3 0.3 0.3 0
## Reliable data transfer simulation with:
        simulation time is 1000.000 seconds
        average message arrival interval is 0.100 seconds
        average message size is 100 bytes
        average out-of-order delivery rate is 30.00%
        average loss rate is 30.00%
        average corrupt rate is 30.00%
        tracing level is 0
Please review these inputs and press <enter> to proceed.

At 0.00s: sender initializing ...
At 0.00s: receiver initializing ...
At 1853.58s: sender finalizing ...
At 1853.58s: receiver finalizing ...

## Simulation completed at time 1853.58s with
        993261 characters sent
        993261 characters delivered
        62265 packets passed between the sender and the receiver
## Congratulations! This session is error-free, loss-free, and in order.
```

五、相关问题及困难：

1、 Go Back N 策略，sender 和 receiver 的 protocol 可以是同一个?

我们在 PPT 中以及上课时介绍过，Go Back N 策略是可以不区分 Sender 和 Receiver 的。现实情况更加复杂，RDT 会判断 upper 和lower 传进来的是什么来判断自己的身份。而在本次 Lab 中，Sender 和 Receiver 的身份是写死的，所以我们不能照着 PPT 上的代码 protocol 去写，那只能作为一个参考。

2、sender 有滑动窗口，receiver 有滑动窗口吗?

滑动窗口的作用是，滑动窗口最末一位如果收到 ACK，滑动窗口就会移动，sender 就知道滑动窗口之前的部分都没问题了，而且滑动窗口中的部分可以一起发，然后维护一个最小的还没收到 ACK 的 pkt 的 timeout，如果超时了就重发这个滑动窗口。

而 receiver 就没必要了，它维护一个 frame_expected，收到想要的就发 ACK，收到超前的话可以暴力丢弃，也可以 buffer 起来，这是一个可以优化的地方。

3、一个 Clock 如何实现多个 Timer?

这里一个实现起来比较简单的方式就是，我们只用一个 Timer。我们设置 Go Back N 滑动窗口的头部元素为 StartTimer，如果该 Timer 超时，我们重发整个滑动窗口。因为头部元素的 TIME_OUT 是最严格的。可能会有性能上的缺陷，但是对于程序的正确性来说是没有问题的。

4、Go Back N 的 Receiver 如果设置 buffer，如何处理一个 seq 对应多个 buffer 中的元素问题?

根据同学在群里所说，我们对 buffer设置一个上限，为 MAX_SEQ 即可。

根据同学搜到的资料，滑动窗口中 seq size 应该大于等于滑动串口大小的两倍，也可以规避这个问题。

六、感谢

太感谢王鑫伟助教啦，最近一直比较麻烦他 QAQ，问了他很多问题。学长都很耐心的给了答复，谢谢哇谢谢哇。

参考资料：

1、TCP中的糊涂窗口的产生及预防解释

2、理解TCP序列号（Sequence Number）和确认号（Acknowledgment Number）

3、运输层协议:(2)Go-Back-N 协议

4、校验和计算方法

5、教学PPT 《02_Mac_Error_SlidingWindow》

6、Lab1 指导文档 《Lab1 - Reliable Data Transport Protocol-1》