Lab 1 - Reliable Data Transport Protocol

519021910913 黄喆敏

xtommy@sjtu.edu.cn

1. 网络包设计

- 对于sender，包的结构如下：

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

- 对于receiver，需要发送ack包，包的结构如下：

```
|<-  2 bytes  ->|<-  4 bytes  ->|<-                  the rest              ->|
|   checksum    |   seq number  |                    payload                 |
```

- 此外，对于sender的第一个包，我们额外用4个byte表示message的大小，以便该message所有packet都收到后，可以进行合并，一起发送。

2. sender 收包 & 发包机制

我们主要采用**Go-Back-N的机制**，并且进行了一定的优化。

- sender收上层的包时，会将message全部放入buffer中，并且将message处理生成packet。同时维护一个滑动窗口，当滑动窗口的数量没有超过上限时，则将buffer中的数据取出，放入滑动窗口内。
- sender收下层发来的ack包时，会将ack与当前的seq进行对比。若ack<seq，代表之前有包没有发送成功，因此进行resend操作；若ack=seq，我们将当前的包发出，并移动滑动窗口。
- sender发包时，会将message进行切割操作，储存在队列中，每次取滑动窗口大小的包发送。若计时器超时，则会将ack与滑动窗口中的seq进行比较，重新发包。

3.receiver 收包 & 发包机制

- 当收到的包seq>ack时，我们会将该数据包存到buffer中。以便当后续收到seq=ack的包时，可以迅速移动滑动窗口，减少发包数量。
- 当seq=ack时，代表这是我们想要的包，滑动窗口右移。此时我们需要查看已经存下的buffer是否有满足条件的后续的包，若有则一起发送给上层。最后一起发送ack包。

4. 参数设置

与lab instruction中类似，滑动窗口大小设为10，timeout时间设为0.3 s。

5. checksum计算方法

Checksum采用了tcp checksum的计算方法。首先将checksum位置0，将整个包划分成一个个16进制数，然后将这些数逐个相加，最后将得到的结果取反。由于我们的包长度为偶数，因此不需要考虑补0的问题。

对于一个packet，先将其他部分填入，最后再计算checksum。对于checksum不准确的包，我们直接丢弃即可。

代码如下所示：

```
static short calc_checksum(struct packet *pkt) {
    long sum = 0;
    for (int i = 2; i < RDT_PKTSIZE; i += 2) {
        sum += *(unsigned short *) (&(pkt->data[i]));
    }
    while (sum >> 16) {
        sum = (sum & 0xffff) + (sum >> 16);
    }
    return ~sum;
}
```

6. 测试结果

测试结果如下所示。**除了`rdt_receiver.cc`和`rdt_sender.cc`，没有对其他文件进行改动。**对于文档中给的所有样例，均进行了20次以上的测试，没有发现错误。此外，采用了`rdt_sim 1000 0.1 100 0.7 0.7 0.7 0`，来测试错误概率较高的情况，也没有发现错误。

对于文档中给出参考吞吐量的两个样例，我们进行了如下测试，吞吐量在合理范围内。

- 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 1028.06s: sender finalizing ...
At 1028.06s: receiver finalizing ...

## Simulation completed at time 1028.06s with
  1008481 characters sent
  1008481 characters delivered
  50409 packets passed between the sender and the receiver
## Congratulations! This session is error-free, loss-free, and in order.
```

- 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 1840.56s: sender finalizing ...
At 1840.56s: receiver finalizing ...

## Simulation completed at time 1840.56s with
  1004946 characters sent
  1004946 characters delivered
  61311 packets passed between the sender and the receiver
## Congratulations! This session is error-free, loss-free, and in order.
```

- rdt_sim 1000 0.1 100 0.7 0.7 0.7 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 70.00%
  average loss rate is 70.00%
  average corrupt rate is 70.00%
  tracing level is 0
Please review these inputs and press <enter> to proceed.

At 0.00s: sender initializing ...
At 0.00s: receiver initializing ...
At 14788.13s: sender finalizing ...
At 14788.13s: receiver finalizing ...

## Simulation completed at time 14788.13s with
  992365 characters sent
  992365 characters delivered
  161676 packets passed between the sender and the receiver
## Congratulations! This session is error-free, loss-free, and in order.
```

Reference

[1] TCP checksum的计算 https://blog.csdn.net/breakout_alex/article/details/102515371