网络传输机制实验2报告

张翔雨 2018K8009929035

一、实验题目：网络传输机制实验2

二、实验内容

- 执行create_randfile.sh，生成待传输数据文件client-input.dat
- 运行给定网络拓扑(tcp_topo_loss.py)
- 在节点h1上执行TCP程序
    - 执行脚本(disable_offloading.sh , disable_tcp_rst.sh)
    - 在h1上运行TCP协议栈的服务器模式 (./tcp_stack server 10001)
- 在节点h2上执行TCP程序
    - 执行脚本(disable_offloading.sh, disable_tcp_rst.sh)
    - 在h2上运行TCP协议栈的客户端模式 (./tcp_stack client 10.0.0.1 10001)
        - Client发送文件client-input.dat给server，server将收到的数据存储到文件server-output.dat
- 使用md5sum比较两个文件是否完全相同
- 使用tcp_stack.py替换其中任意一端，对端都能正确收发数据

三、实验过程

1.引入send_buffer和ofo_buffer

每一次发包后将发送包对应消息的内容以及相关flag等信息存入发送队列中。

在接受到新包后，抛弃掉seq小于当前下一个要接受序号的包，并将相等的包写入ring_buffer，大于的包写入ofo队列。如果有新的包写入ringbuffer，需要扫描接受队列并将其中与刚刚写入的包序号连续的包写入ringbuffer并更新rcv_nxt信息。

```
struct send_buffer {
    struct list_head list;
    char *packet;
    int len;
    u32 seq_end;
    int times;
    int timeout;
};

struct ofo_buffer {
    struct list_head list;
    struct tcp_sock *tsk;
    u32 seq;
    u32 seq_end;
    char *payload;
    int pl_len;
};
```

需要注意的是这里写入ofo队列需要按序写入，实现代码如下：

```
void write_ofo_buffer(struct tcp_sock *tsk, struct tcp_cb *cb)
{
    struct ofo_buffer *buf = (struct ofo_buffer*)malloc(sizeof(struct ofo_buffer));
    buf->tsk = tsk;
    buf->seq = cb->seq;
    buf->seq_end = cb->seq_end;
    buf->pl_len = cb->pl_len;
    buf->payload = (char*)malloc(buf->pl_len);
    memcpy(buf->payload, cb->payload, buf->pl_len);
    struct ofo_buffer head_ext;
    head_ext.list = tsk->rcv_ofo_buf;
    int insert = 0;
    struct ofo_buffer *pos, *last = &head_ext;
    list_for_each_entry(pos, &tsk->rcv_ofo_buf, list)
    {
        if (cb->seq > pos->seq)
        {
            last = pos;
            continue;
        }
        else if (cb->seq == pos->seq) return;
        list_insert(&buf->list, &last->list, &pos->list);
        insert = 1;
        break;
    }
    if (!insert)
        list_add_tail(&buf->list, &tsk->rcv_ofo_buf);
}
```

2.添加状态处理逻辑

为了实现对不符合seq的包进行丢弃的逻辑，对之前的状态机处理逻辑进行重构，由根据当前的状态来对传入的包进行判断改为由传入的包的类型再根据当前状态来进行状态转移的判断。

在established状态中，将符合条件seq的包加入ringbuffer中，并缓存将要接受的包。

```
            u32 seq_end = tsk->rcv_nxt;
             if (seq_end == cb->seq)
             {
                 write_ring_buffer(tsk->rcv_buf, cb->payload, cb->pl_len);
                 seq_end = cb->seq_end;
                 struct ofo_buffer *entry, *q;
                 list_for_each_entry_safe(entry, q, &tsk->rcv_ofo_buf, list)
                 {
                     if (seq_end < entry->seq)
                         break;
                     else
                     {
                         seq_end = entry->seq_end;
                         write_ring_buffer(entry->tsk->rcv_buf, entry->payload, entry-
>pl_len);
                         list_delete_entry(&entry->list);
                         free(entry->payload);
                         free(entry);
                     }
                 }
                 tsk->rcv_nxt = seq_end;
             }
            else if (seq_end < cb->seq)
                 write_ofo_buffer(tsk, cb);
             if (tsk->wait_recv->sleep)
                 wake_up(tsk->wait_recv);
             tcp_send_control_packet(tsk, TCP_ACK);
             if (tsk->wait_send->sleep)
                 wake_up(tsk->wait_send);
```

3.写函数改进

实现`int tcp_sock_write(struct tcp_sock *tsk, char *buf, int len)`

主要改进为根据当前发送队列大小判断是否需要睡眠，以实现拥塞控制。

```
        pthread_mutex_lock(&tsk->count_lock);
         while (tsk->send_buf_count >= 5)
         {
             pthread_mutex_unlock(&tsk->count_lock);
             sleep_on(tsk->wait_send);
             pthread_mutex_lock(&tsk->count_lock);
         }
         pthread_mutex_unlock(&tsk->count_lock);
```

4.TCP重传计时器实现

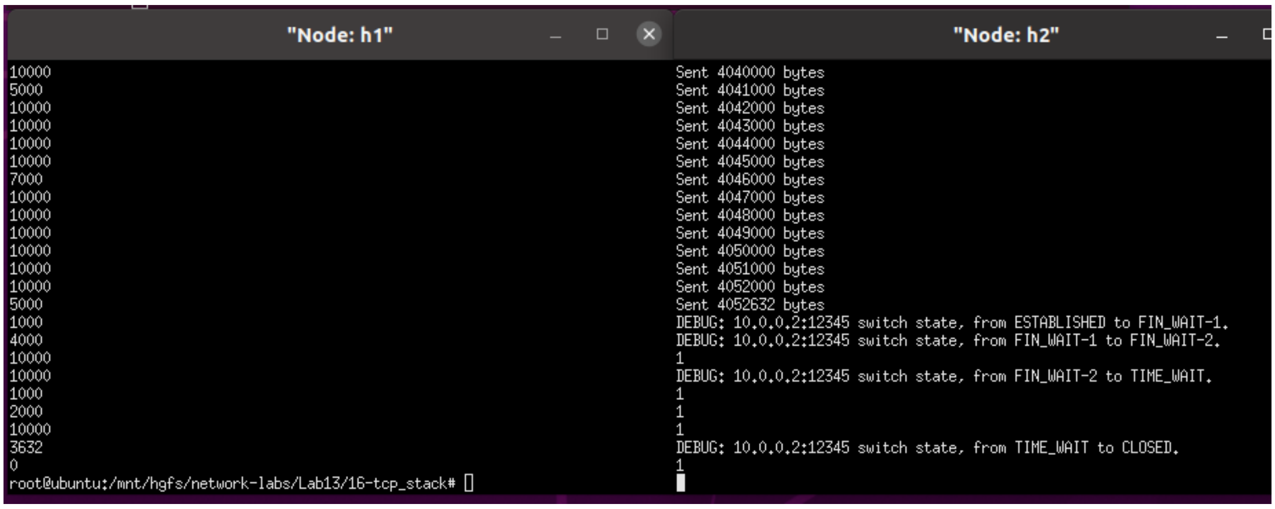
将重传计时器添加到timerlist中，在扫描线程进行扫描时，根据计时器类型判断，对重传计时器进行不同的处理逻辑, 如下。

```
             tsk = retranstimer_to_tcp_sock(timer);
             struct send_buffer *entry;
             list_for_each_entry(entry, &tsk->send_buf, list)
             {
                 entry->timeout -= TCP_TIMER_SCAN_INTERVAL;
                 if (!entry->timeout)
                 {
                     if (entry->times++ == 3)
                     {
                         entry->times = 1;
                         entry->timeout = TCP_RETRANS_INTERVAL_INITIAL;
                     }
                     else
                     {
                         char *temp = (char*)malloc(entry->len * sizeof(char));
                         memcpy(temp, entry->packet, entry->len);
                         ip_send_packet(temp, entry->len);
                         if (entry->times == 2)
                             entry->timeout = entry->times * TCP_RETRANS_INTERVAL_INITIAL;
                         else
                             entry->timeout = 4 * TCP_RETRANS_INTERVAL_INITIAL;
                     }
                 }
             }
```

四、实验结果

收发文件测试

（1）server脚本与自编写client交互



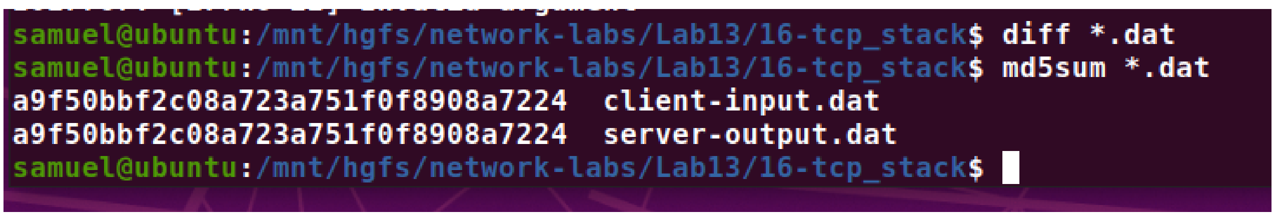
"Node: h1"

```
10000
5000
10000
10000
10000
10000
7000
10000
10000
10000
10000
10000
10000
5000
1000
4000
10000
10000
1000
2000
10000
3632
0
root@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack# 
```

"Node: h2"

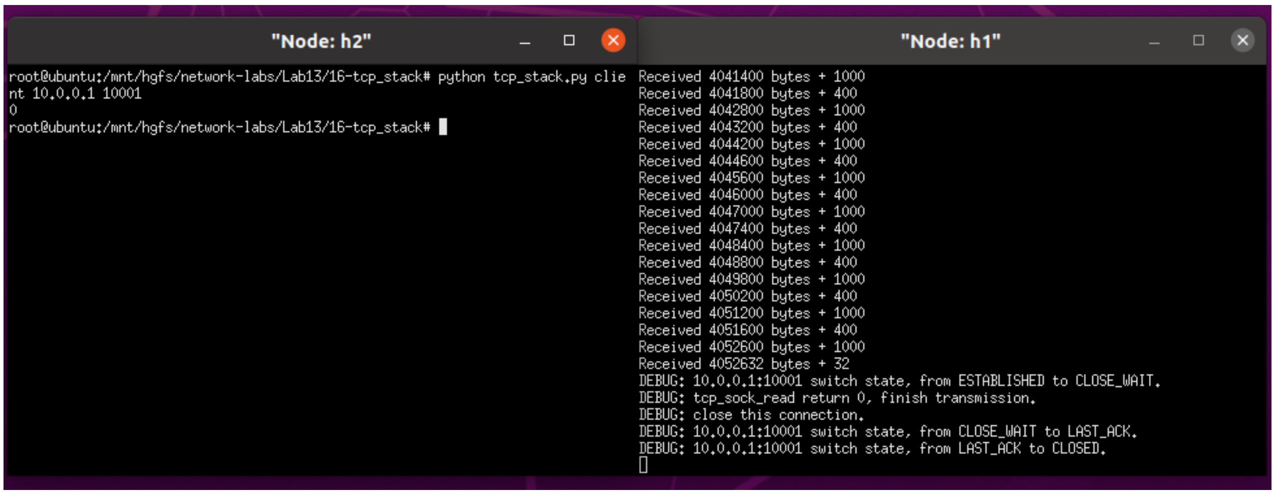
```
Sent 4040000 bytes
Sent 4041000 bytes
Sent 4042000 bytes
Sent 4043000 bytes
Sent 4044000 bytes
Sent 4045000 bytes
Sent 4046000 bytes
Sent 4047000 bytes
Sent 4048000 bytes
Sent 4049000 bytes
Sent 4050000 bytes
Sent 4051000 bytes
Sent 4052000 bytes
Sent 4052632 bytes
DEBUG: 10.0.0.2:12345 switch state, from ESTABLISHED to FIN_WAIT-1.
DEBUG: 10.0.0.2:12345 switch state, from FIN_WAIT-1 to FIN_WAIT-2.
1
DEBUG: 10.0.0.2:12345 switch state, from FIN_WAIT-2 to TIME_WAIT.
1
1
1
DEBUG: 10.0.0.2:12345 switch state, from TIME_WAIT to CLOSED.
1
```

发送完成后，在服务器命令行（左）执行md5sum与diff命令，发现md5sum相同，diff未返回不同的地方，证明实验成功。



```
samuel@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack$ diff *.dat
samuel@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack$ md5sum *.dat
a9f50bbf2c08a723a751f0f8908a7224  client-input.dat
a9f50bbf2c08a723a751f0f8908a7224  server-output.dat
samuel@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack$ 
```

（2）client脚本与自编写server交互



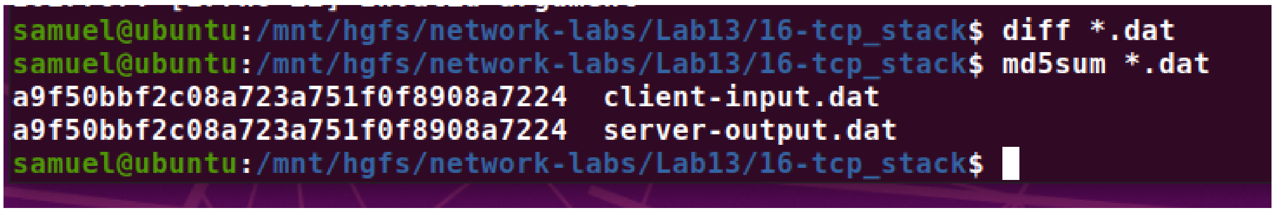
"Node: h2"

```
root@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack# python tcp_stack.py clie
nt 10.0.0.1 10001
0
root@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack# 
```

"Node: h1"

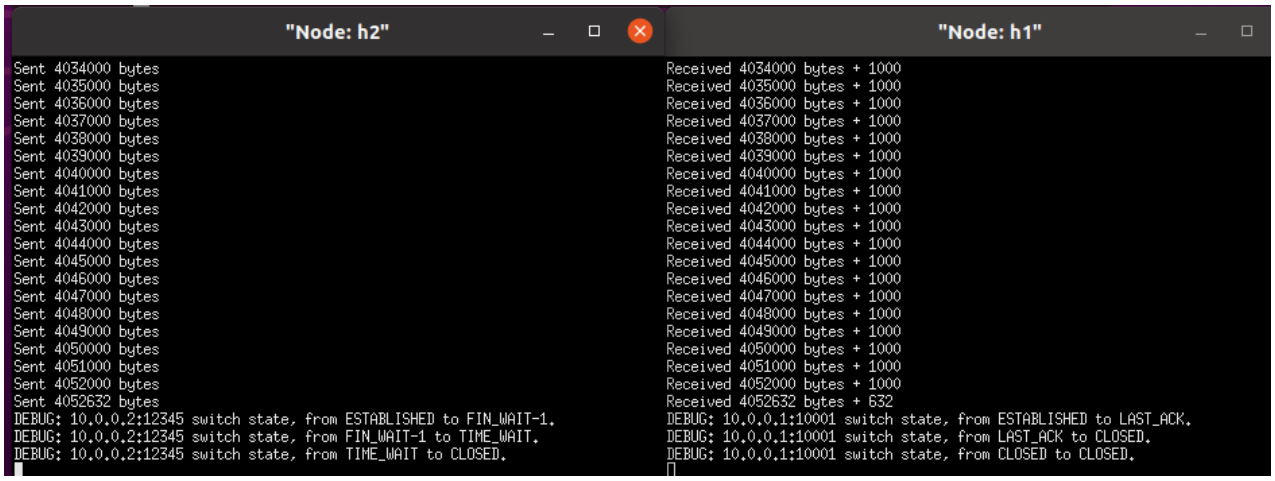
```
Received 4041400 bytes + 1000
Received 4041800 bytes + 400
Received 4042800 bytes + 1000
Received 4043200 bytes + 400
Received 4044200 bytes + 1000
Received 4044600 bytes + 400
Received 4045600 bytes + 1000
Received 4046000 bytes + 400
Received 4047000 bytes + 1000
Received 4047400 bytes + 400
Received 4048400 bytes + 1000
Received 4048800 bytes + 400
Received 4049800 bytes + 1000
Received 4050200 bytes + 400
Received 4051200 bytes + 1000
Received 4051600 bytes + 400
Received 4052600 bytes + 1000
Received 4052632 bytes + 32
DEBUG: 10.0.0.1:10001 switch state, from ESTABLISHED to CLOSE_WAIT.
DEBUG: tcp_sock_read return 0, finish transmission.
DEBUG: close this connection.
DEBUG: 10.0.0.1:10001 switch state, from CLOSE_WAIT to LAST_ACK.
DEBUG: 10.0.0.1:10001 switch state, from LAST_ACK to CLOSED.
```

发送完成后，在客户端命令后（右）执行md5sum和diff命令，发现md5sum相同，diff未返回不同的地方，证明实验成功。



```
samuel@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack$ diff *.dat
samuel@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack$ md5sum *.dat
a9f50bbf2c08a723a751f0f8908a7224  client-input.dat
a9f50bbf2c08a723a751f0f8908a7224  server-output.dat
samuel@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack$ 
```

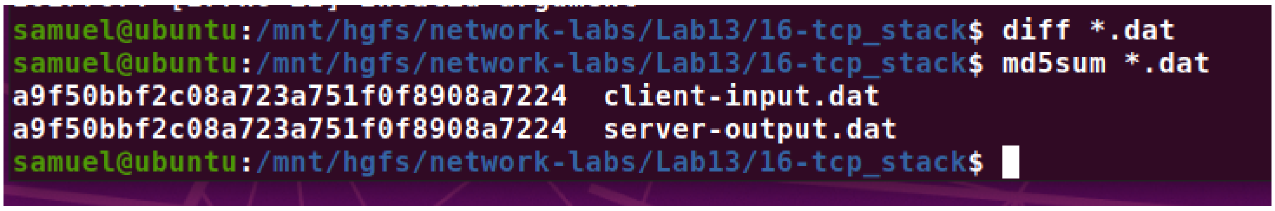
（3）自编写server与client交互



```
Sent 4034000 bytes
Sent 4035000 bytes
Sent 4036000 bytes
Sent 4037000 bytes
Sent 4038000 bytes
Sent 4039000 bytes
Sent 4040000 bytes
Sent 4041000 bytes
Sent 4042000 bytes
Sent 4043000 bytes
Sent 4044000 bytes
Sent 4045000 bytes
Sent 4046000 bytes
Sent 4047000 bytes
Sent 4048000 bytes
Sent 4049000 bytes
Sent 4050000 bytes
Sent 4051000 bytes
Sent 4052000 bytes
Sent 4052632 bytes
DEBUG: 10.0.0.2:12345 switch state, from ESTABLISHED to FIN_WAIT-1.
DEBUG: 10.0.0.2:12345 switch state, from FIN_WAIT-1 to TIME_WAIT.
DEBUG: 10.0.0.2:12345 switch state, from TIME_WAIT to CLOSED.
```

```
Received 4034000 bytes + 1000
Received 4035000 bytes + 1000
Received 4036000 bytes + 1000
Received 4037000 bytes + 1000
Received 4038000 bytes + 1000
Received 4039000 bytes + 1000
Received 4040000 bytes + 1000
Received 4041000 bytes + 1000
Received 4042000 bytes + 1000
Received 4043000 bytes + 1000
Received 4044000 bytes + 1000
Received 4045000 bytes + 1000
Received 4046000 bytes + 1000
Received 4047000 bytes + 1000
Received 4048000 bytes + 1000
Received 4049000 bytes + 1000
Received 4050000 bytes + 1000
Received 4051000 bytes + 1000
Received 4052000 bytes + 1000
Received 4052632 bytes + 632
DEBUG: 10.0.0.1:10001 switch state, from ESTABLISHED to LAST_ACK.
DEBUG: 10.0.0.1:10001 switch state, from LAST_ACK to CLOSED.
DEBUG: 10.0.0.1:10001 switch state, from CLOSED to CLOSED.
```

先看到发送和接受字节数正确，且状态转移正确。退出后在当前目录下执行md5sum和diff命令，发现md5sum相同，diff未返回不同的地方，证明实验成功。



```
samuel@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack$ diff *.dat
samuel@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack$ md5sum *.dat
a9f50bbf2c08a723a751f0f8908a7224  client-input.dat
a9f50bbf2c08a723a751f0f8908a7224  server-output.dat
samuel@ubuntu:/mnt/hgfs/network-labs/Lab13/16-tcp_stack$ 
```

六、实验总结

本次实验较为复杂，由于存在loss的情况，有时实验出现的bug难以复现，同时暴露了过去代码的一些问题。本次实验过程对状态机处理代码进行了重构，为了解决丢弃不符合seq要求的包的问题，可能不是最优解。