

## 一、实验题目：网络传输机制实验2

## 二、实验内容

### 内容1:

- 运行给定网络拓扑(tcp\_topo.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，连接h1并正确收发数据 (./tcp\_stack client 10.0.0.1 10001)
    - client向server发送数据，server将数据echo给client
- 使用tcp\_stack.py替换其中任意一端，对端都能正确收发数据

### 内容2:

- 修改tcp\_apps.c(以及tcp\_stack.py)，使之能够收发文件
- 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.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.添加状态处理逻辑

在 `established` 状态的处理逻辑中添加处理非 `FIN` 包的处理逻辑，即PSH包和ACK包，若收到数据长度是0则为ACK包，反之为数据包。基本处理逻辑是当收到数据包时，先判断 `ring_buffer` 是否已满，若已经满了则进入睡眠，等待读缓冲区函数唤醒，然后获取锁并写入缓冲区，之后唤醒因缓冲区为空而睡眠的进程并发送ACK包。

在处理 `FIN` 包时，标志此次传输已经结束，需要退出所有等待的进程，因此调用 `wait_exit` 函数设置dead位为1，并向所有等待的进程广播。

```
case TCP_ESTABLISHED:
    if(cb->flags & TCP_FIN)
    {
        tsk->rcv_nxt = cb->seq+1;
        wait_exit(tsk->wait_rcv);
        tcp_set_state(tsk, TCP_CLOSE_WAIT);
        tcp_send_control_packet(tsk, TCP_ACK);
    }
    else if(cb->flags == (TCP_PSH | TCP_ACK))
    {
        if(cb->pl_len == 0)
        {
            tsk->snd_una = cb->ack;
            tsk->rcv_nxt = cb->seq + 1;
            tcp_update_window_safe(tsk, cb);
        }
        else
        {
            while (ring_buffer_full(tsk->rcv_buf))
                sleep_on(tsk->wait_rcv);
            pthread_mutex_lock(&tsk->rcv_buf->lock);
            write_ring_buffer(tsk->rcv_buf, cb->payload, cb->pl_len);
            pthread_mutex_unlock(&tsk->rcv_buf->lock);
        }
    }
}
```

```

        tsk->rcv_nxt = cb->seq + cb->pl_len;
        tsk->snd_una = cb->ack;
        wake_up(tsk->wait_rcv);
        tcp_send_control_packet(tsk, TCP_ACK);
    }
}
break;

```

## 2.读写函数设计

### (1) 实现 `int tcp_sock_read(struct tcp_sock *tsk, char *buf, int len)`

主要功能是读取缓冲区数据，并根据读取的状态返回不同的值。当缓冲区为空时进入等待状态，等待收到数据包时唤醒，读过程结束后唤醒因缓冲区满而等待的进程。若被 `wait_exit` 的广播唤醒，则再次进行等待判断时，会因为dead位为1而等待失败，此时read函数返回0代表连接结束。

```

int tcp_sock_read(struct tcp_sock *tsk, char *buf, int len)
{
    while( ring_buffer_empty(tsk->rcv_buf) )
    {
        if(sleep_on(tsk->wait_rcv)<0)
            return 0;
    }
    int plen = read_ring_buffer(tsk->rcv_buf, buf, len);
    wake_up(tsk->wait_rcv);
    return plen;
}

```

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

主要功能是根据buf内容发送对应的数据包，首先要判断代发送数据的大小，若超过一个包的大小，则需要分多个包发送。返回发送的长度。

```

int tcp_sock_write(struct tcp_sock *tsk, char *buf, int len)
{
    int seq = tsk->snd_nxt;
    int send_len = len;
    int data_len = 1514 - ETHER_HDR_SIZE - IP_BASE_HDR_SIZE - TCP_BASE_HDR_SIZE;
    while (len > data_len)
    {
        char * packet = (char *)malloc(1514);
        memcpy(packet + ETHER_HDR_SIZE + IP_BASE_HDR_SIZE + TCP_BASE_HDR_SIZE,
               buf + (tsk->snd_nxt - seq), data_len);
    }
}

```

```

        tcp_send_packet(tsk, packet, 1514);
        usleep(10000);
        len -= data_len;
    }
    int pkt_size = ETHER_HDR_SIZE + IP_BASE_HDR_SIZE + TCP_BASE_HDR_SIZE + len;
    char *packet = (char *)malloc(pkt_size);
    memset(packet, 0, ETHER_HDR_SIZE + IP_BASE_HDR_SIZE + TCP_BASE_HDR_SIZE);
    memcpy(packet + ETHER_HDR_SIZE + IP_BASE_HDR_SIZE + TCP_BASE_HDR_SIZE,
           buf + (tsk->snd_nxt - seq), len);
    usleep(100000);
    tcp_send_packet(tsk, packet, pkt_size);
    return send_len;
}

```

## 四、实验结果

### 1. 内容1：收发数据测试

#### (1) server脚本与自编写client交互

```

"Node: h1"
root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack01# python tcp_stack.py se
rver 10001
('10.0.0.2', 12345)
<type 'str'>
<type 'str'>
<type 'str'>
<type 'str'>
<type 'str'>
<type 'str'>
<type 'str'>
<type 'str'>
<type 'str'>
<type 'str'>
root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack01#

"Node: h2"
root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack01# ./tcp_stack client 10.
0.0.1 10001
DEBUG: find the following interfaces: h2-eth0.
Routing table of 1 entries has been loaded.
DEBUG: 10.0.0.2:12345 switch state, from CLOSED to SYN_SENT.
DEBUG: 10.0.0.2:12345 switch state, from SYN_SENT to ESTABLISHED.
server echoes: 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ
server echoes: 123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ0
server echoes: 23456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ01
server echoes: 3456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ012
server echoes: 456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ0123
server echoes: 56789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ01234
server echoes: 6789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ012345
server echoes: 789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ0123456
server echoes: 89abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ01234567
server echoes: 9abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
OPQRSTUVWXYZ012345678
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.
DEBUG: 10.0.0.2:12345 switch state, from FIN_WAIT-2 to TIME_WAIT.
DEBUG: 10.0.0.2:12345 switch state, from TIME_WAIT to CLOSED.

```

可以看到客户端收到了正确的服务器返回的数据，并按照正常的窗台转移退出，服务器收到包的次数也相同，测试正确。

#### (2) client脚本与自编写server交互



```
"Node: h1"                                "Node: h2"
1460                                         Sent 3880000 bytes
1460                                         Sent 3890000 bytes
1460                                         Sent 3900000 bytes
1460                                         Sent 3910000 bytes
1240                                         Sent 3920000 bytes
1460                                         Sent 3930000 bytes
1460                                         Sent 3940000 bytes
1460                                         Sent 3950000 bytes
1460                                         Sent 3960000 bytes
1460                                         Sent 3970000 bytes
1460                                         Sent 3980000 bytes
1240                                         Sent 3990000 bytes
1460                                         Sent 4000000 bytes
1172                                         Sent 4010000 bytes
0                                             Sent 4020000 bytes
root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02# md5sum client-input.dat Sent 4030000 bytes
t                                             Sent 4040000 bytes
0b2230eb07edc7a2788be0808356cbcc client-input.dat Sent 4050000 bytes
root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02# md5sum server-output.dat Sent 4052632 bytes
at                                         DEBUG: 10.0.0.2:12345 switch state, from ESTABLISHED to FIN_WAIT-1.
root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02# diff client-input.dat DEBUG: 10.0.0.2:12345 switch state, from FIN_WAIT-1 to FIN_WAIT-2.
server-output.dat                      DEBUG: 10.0.0.2:12345 switch state, from FIN_WAIT-2 to TIME_WAIT.
root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02# |         DEBUG: 10.0.0.2:12345 switch state, from TIME_WAIT to CLOSED.
```

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。

## （2）client脚本与自编写server交互

```
"Node: h1"                                "Node: h2"
Received 4046000 bytes + 400               root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02# python tcp_stack.py cl
Received 4046400 bytes + 400               ient 10.0.0.1 10001
Received 4046800 bytes + 400               0
Received 4047200 bytes + 400               root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02# diff client-input.dat
Received 4047600 bytes + 400               server-output.dat
Received 4048000 bytes + 400               root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02# md5sum server-output,d
Received 4048400 bytes + 400               at
Received 4048800 bytes + 400               0b2230eb07edc7a2788be0808356cbcc server-output.dat
Received 4049200 bytes + 400               root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02# md5sum client-input,d
Received 4049600 bytes + 400               t
Received 4050000 bytes + 400               0b2230eb07edc7a2788be0808356cbcc client-input.dat
Received 4050400 bytes + 400               root@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02# |
Received 4050800 bytes + 400
Received 4051200 bytes + 400
Received 4051600 bytes + 400
Received 4052000 bytes + 400
Received 4052400 bytes + 400
Received 4052632 bytes + 232
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。

## （3）自编写server与client交互

```
"Node: h1"                                "Node: h2"
Received 4028760 bytes + 1460              Sent 3880000 bytes
Received 4030000 bytes + 1240              Sent 3890000 bytes
Received 4031460 bytes + 1460              Sent 3900000 bytes
Received 4032920 bytes + 1460              Sent 3910000 bytes
Received 4034380 bytes + 1460              Sent 3920000 bytes
Received 4035840 bytes + 1460              Sent 3930000 bytes
Received 4037300 bytes + 1460              Sent 3940000 bytes
Received 4038760 bytes + 1460              Sent 3950000 bytes
Received 4040000 bytes + 1240              Sent 3960000 bytes
Received 4041460 bytes + 1460              Sent 3970000 bytes
Received 4042920 bytes + 1460              Sent 3980000 bytes
Received 4044380 bytes + 1460              Sent 3990000 bytes
Received 4045840 bytes + 1460              Sent 4000000 bytes
Received 4047300 bytes + 1460              Sent 4010000 bytes
Received 4048760 bytes + 1460              Sent 4020000 bytes
Received 4050000 bytes + 1240              Sent 4030000 bytes
Received 4051460 bytes + 1460              Sent 4040000 bytes
Received 4052632 bytes + 1172              Sent 4050000 bytes
DEBUG: 10.0.0.1:10001 switch state, from ESTABLISHED to CLOSE_WAIT.          Sent 4052632 bytes
DEBUG: tcp_sock_read return 0, finish transmission.                          DEBUG: 10.0.0.2:12345 switch state, from ESTABLISHED to FIN_WAIT-1.
DEBUG: close this connection.                                                  DEBUG: 10.0.0.2:12345 switch state, from FIN_WAIT-1 to FIN_WAIT-2.
DEBUG: 10.0.0.1:10001 switch state, from CLOSE_WAIT to LAST_ACK.              DEBUG: 10.0.0.2:12345 switch state, from FIN_WAIT-2 to TIME_WAIT.
DEBUG: 10.0.0.1:10001 switch state, from LAST_ACK to CLOSED.                 DEBUG: 10.0.0.2:12345 switch state, from TIME_WAIT 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/Lab12/13-tcp_stack02$ md5sum server-output.dat
0b2230eb07edc7a2788be0808356cbcc  server-output.dat
samuel@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02$ md5sum client-input.dat
0b2230eb07edc7a2788be0808356cbcc  client-input.dat
samuel@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02$ diff client-input.dat server-output.dat
samuel@ubuntu:/mnt/hgfs/network-labs/Lab12/13-tcp_stack02$
```

## 六、实验总结

本次实验代码量较少，需要注意的地方却有一些。由于本次实现的读写函数不包含tcp重传与检测机制，导致在与脚本交互的过程中出现问题，通过wireshark抓包测试找到了重传发生的位置，发现对ACK包的序列号处理不当导致发生重传。