计算机网络研讨课实验报告

冯吕 2015K8009929049

2018 年 6 月 29 日

1 实验题目

网络传输机制实验二

2 实验内容

- 在本次实验中，需要实现 TCP 数据收发功能，使得节点之间能够在无丢包网络环境中传输数据。
- 运行给定拓扑网络，在 h_1 和 h_2 上分别运行 tcp_stack，验证数据收发功能是否正确。

3 实验流程

本次实验中，需要实现数据收发功能，实现底层的 tcp_sock_write 和 tcp_sock_read

3.1 tcp_sock_write

tcp_sock_write 函数将上层应用 $buffer$ 中存储的数据封装成数据包发送出去,然后需要等待 $wait\_send$:

```
1  int tcp_sock_write(struct tcp_sock *tsk, char *buf, int len){
2
3          int data_len = min(len, 1500 - ETHER_HDR_SIZE
4          - IP_BASE_HDR_SIZE - TCP_BASE_HDR_SIZE);
5          int pkt_size = ETHER_HDR_SIZE + IP_BASE_HDR_SIZE
6          + TCP_BASE_HDR_SIZE + data_len;
7
8          char *packet = (char *) malloc (pkt_size);
9          memset(packet, 0, pkt_size);
10         memcpy(packet + ETHER_HDR_SIZE+IP_BASE_HDR_SIZE+
11         TCP_BASE_HDR_SIZE, buf, data_len);
12
13         tcp_send_packet(tsk, packet, pkt_size);
14
15         sleep_on(tsk->wait_send);
16
17         return data_len;
18 }
```

对端收到发送过去的数据包后，将数据包中的数据写入 *rcv_buf* 中，然后回复 *ACK*，同时 *wake_up wait_recv*。由于数据读写可能会发生冲突，因此，需要通过锁对 *rcv_buf* 进行互斥访问。

当收到对端发送回来的 *ACK* 后，再 *wake_up wait_send*。

3.2 *tcp_sock_read*

tcp_sock_read 函数则是从 *rcv_buf* 中读取数据到上层应用 *buffer* 中，如果 *rcv_buf* 为空，则 *wait_rcv*，等到收到数据后 *wake_up*：

```
1  int tcp_sock_read(struct tcp_sock *tsk, char *buf, int len){
2          pthread_mutex_lock(&tsk->rcv_buf->rw_lock);
3          int not_sleep = 1;
4
5          if (ring_buffer_empty(tsk->rcv_buf)){
6                  pthread_mutex_unlock(&tsk->rcv_buf->rw_lock);
7                  not_sleep = 0;
8                  sleep_on(tsk->wait_recv);
9          }
10         if (!not_sleep){
11                 pthread_mutex_lock(&tsk->rcv_buf->rw_lock);
12         }
13         int read_len = read_ring_buffer(tsk->rcv_buf, buf, len);
14         pthread_mutex_unlock(&tsk->rcv_buf->rw_lock);
15         return read_len;
16 }
```

3.3 运行实验

- 运行给定网络拓扑;
- 在节点 h_1 上运行 *tcp* 程序，并运行协议栈的服务器模式;
- 在节点 h_2 上运行 *tcp* 程序，并运行协议栈的客户端模式;
- *client* 向 *server* 发送数据，*server* 将数据 *echo* 给 *client*;

4 实验结果

运行 *tcp* 程序之后，能够成功建立连接，然后收发数据，最后关闭连接：

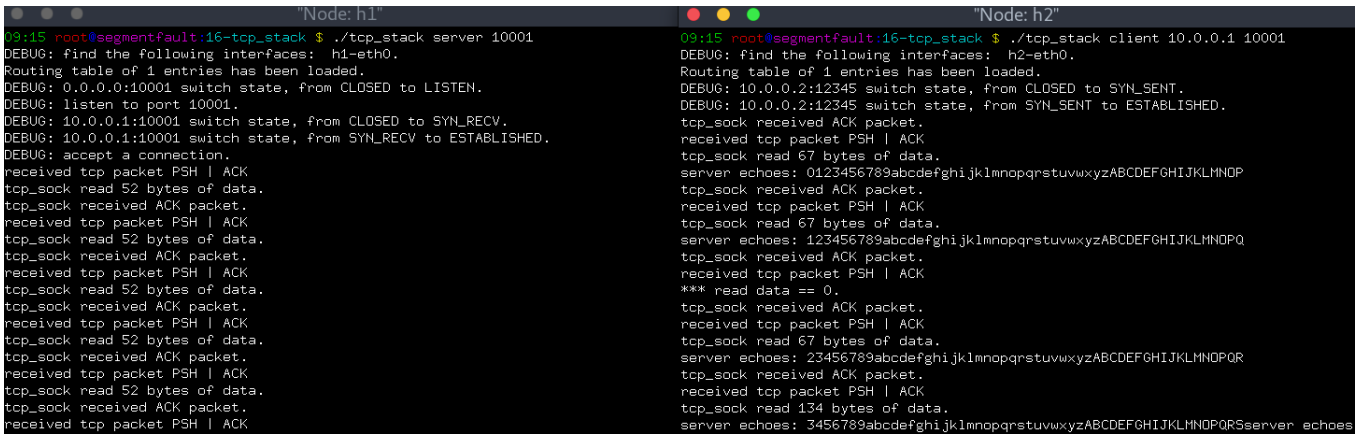

图 1: 建立连接 → 收发数据

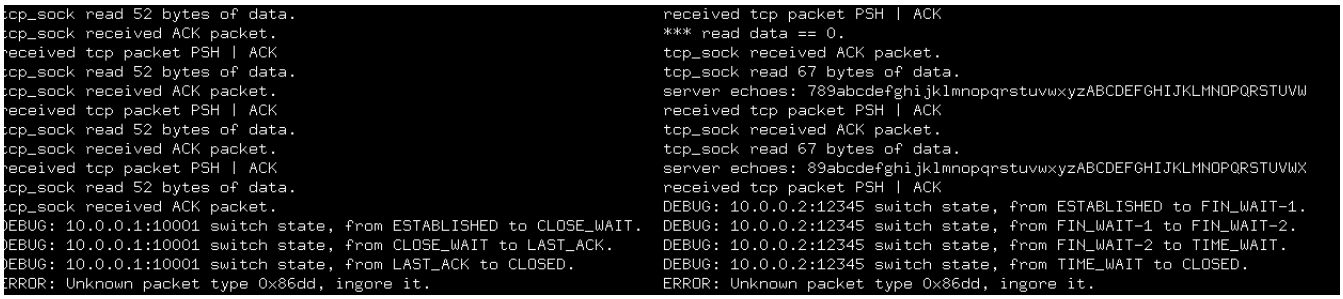

图 2: 收发数据 → 关闭连接

5 结果分析

节点建立连接之后能够在不丢包的情况下收发数据。