16-TCP-data-transfer 实验报告

作者：苗屹松

时间：2018年1月14日

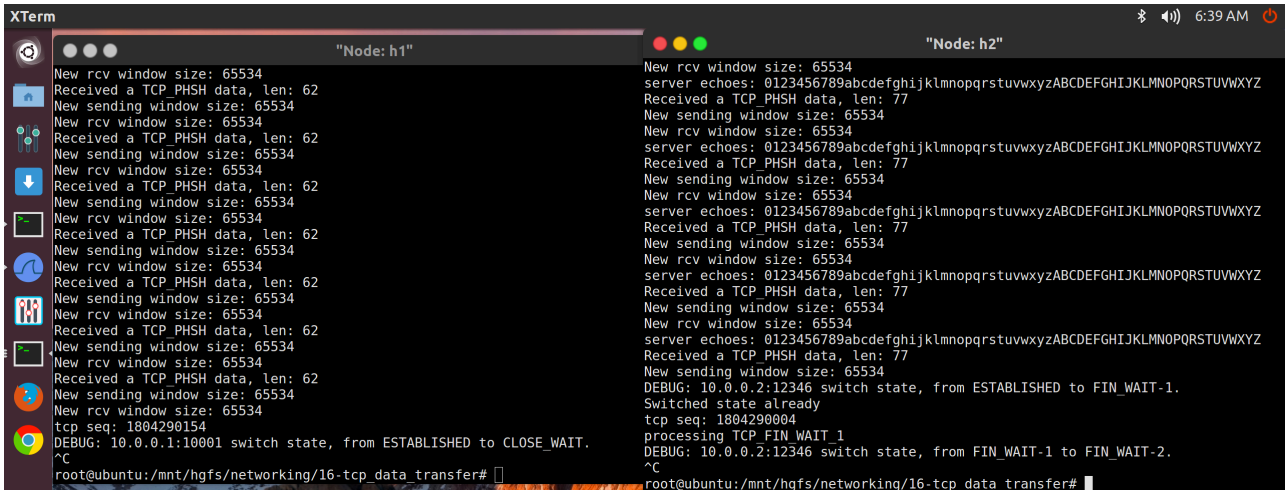
1. Overview

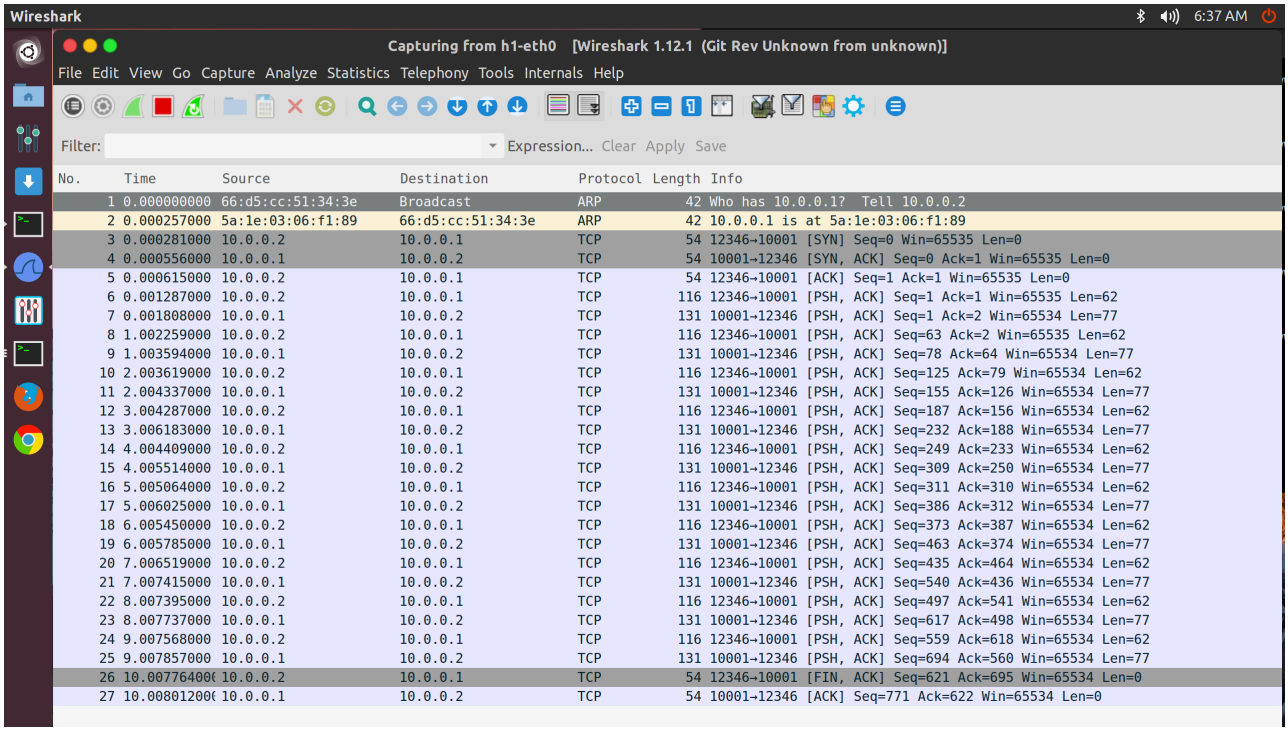
本次实验，我完成了TCP数据传输。

代码同时存储在：https://gitlab.com/miaoyisong/network_lab/tree/master/16-tcp_data_transfer

2.实验结果

我圆满了完成了实验要求，和老师的reference结果一致～如下图：

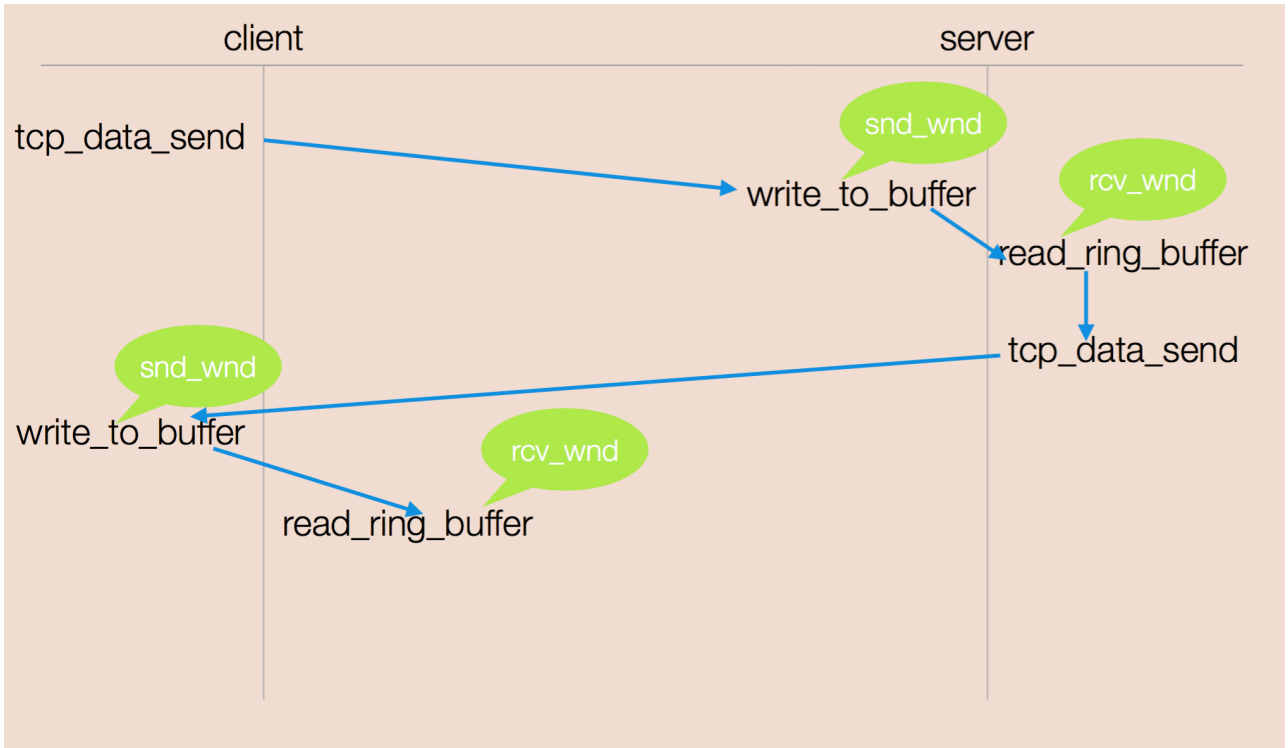




3. 实验思路

我把h2向h1发送数据定义为一个循环，当我把这一个循环的思路理清之后，实验就相当清晰了！

如下图：



在上图中，client发送数据包后，

server写入到ring_buffer中，之后从ring_buffer中读出，再echo回client

client收到server的echo后，先写入到ring_buffer中，之后从ring_buffer中读出。

我们可以看到，snd_wnd都是在write_to_buffer调整的（通过对端的rcv_wnd）

rcv_wnd都是在read_ring_buffer处调整的(这时自己的ring_buffer状态有了更新)

4. 关键步骤

本次实验的核心是这两个变量：`snd_wnd`和`rcv_wnd`

rev_wnd是由tsk自己决定的，tsk通过调整自己的rcv_wnd来表达自己的接收能力。

snd_wnd却是由对端tsk决定的，对端告诉己端它的接收能力，己端就把自己的snd_wnd设置为对端的rcv_wnd的大小。

那么tsk如何调整自己的接收能力呢?

我使用了老师写好的函数：`static inline int ring_buffer_free(struct ring_buffer *rbuf)`

在每一次read_ring_buffer后更新自己的`rev_wnd`

```
int tcp_sock_read(struct tcp_sock *tsk, char *buf, int len){
    if(ring_buffer_empty(tsk->rcv_buf)){
        return 0;
    }
    int read_len = read_ring_buffer(tsk->rcv_buf, buf, len);
    tsk->rcv_wnd = ring_buffer_free(tsk->rcv_buf);
    printf("New rcv window size: %d\n", tsk->rcv_wnd);

    return read_len;
}
```

那么tsk如何调整`snd_wnd`呢?

就是在Established状态下，如果接收到一个TCP_ACK&TCP_PSH的包，读出其中的`rcv_wnd`，让`snd_wnd`等于`rcv_wnd`

```
case TCP_ESTABLISHED:
            ...
            if(cb->flags & TCP_PSH){
                tsk->rcv_nxt = cb->seq + 1;
                printf("Received a TCP_PHSH data, len: %d\n", cb->pl_len);
                write_ring_buffer(tsk->rcv_buf, cb->payload, cb->pl_len);
                tcp_update_window(tsk, cb);
            }
```

不过在本次实验中，发送数据包的大小分别是62和77，每次循环里都能把ring_buffer里的所有内容读完，所以window_size一直都是满的状态～

5.实验总结

本次实验的核心还是要读懂`tcp_app.c`里蕴含的逻辑，后面就比较好做了～

另外要读懂ring_buffer部分的源码，通过`%`符号，我知道了这个确实是一个ring😂这部分的API也非常清晰，很方便我们在程序中调用

这应该是我在网络课程的最后一次实验报告，谢谢武老师一个学期的辛苦付出！