网络层实验总结

苗屹松 2018年1月14日

TCP实验一 Hash的思想

实验里最重要的数据结构就是 tcp_sock, 里面包含了这个socket所有的控制信息。

过程中，要收发各种的包，那么是谁来收发这些包呢？对，就是tcp_sock

那么我们如何找到我们需要的tcp_sock呢？就需要用hash

```
struct tcp_sock *tsk = tcp_sock_lookup(&cb);
```

老师的这段源码就告诉我：通过收到的包里的关键信息，就可以找到对应的socket。

我们分别在`tcp_sock_connect` 和`tcp_sock_listen`把我们socket放入`tcp_established_sock_table`和`tcp_listen_sock_table`中。

然后每次在tcp_process时，通过lookup函数找到需要的socket:

TCP实验一 sleep_on&wake_up

- 当一个资源(比如socket)还没到位时，我们先让这个函数暂停(sleep_on)在此处，等到资源到位时，再继续这个函数(wake_up)

```
struct tcp_sock *tcp_sock_accept(struct tcp_sock *tsk)
{
    if(list_empty(&(tsk->accept_queue))){
        sleep_on(tsk->wait_accept);
        struct tcp_sock * the_tcp_socket = tcp_sock_accept_dequeue(tsk);
        return the_tcp_socket;
    }
    ...
```

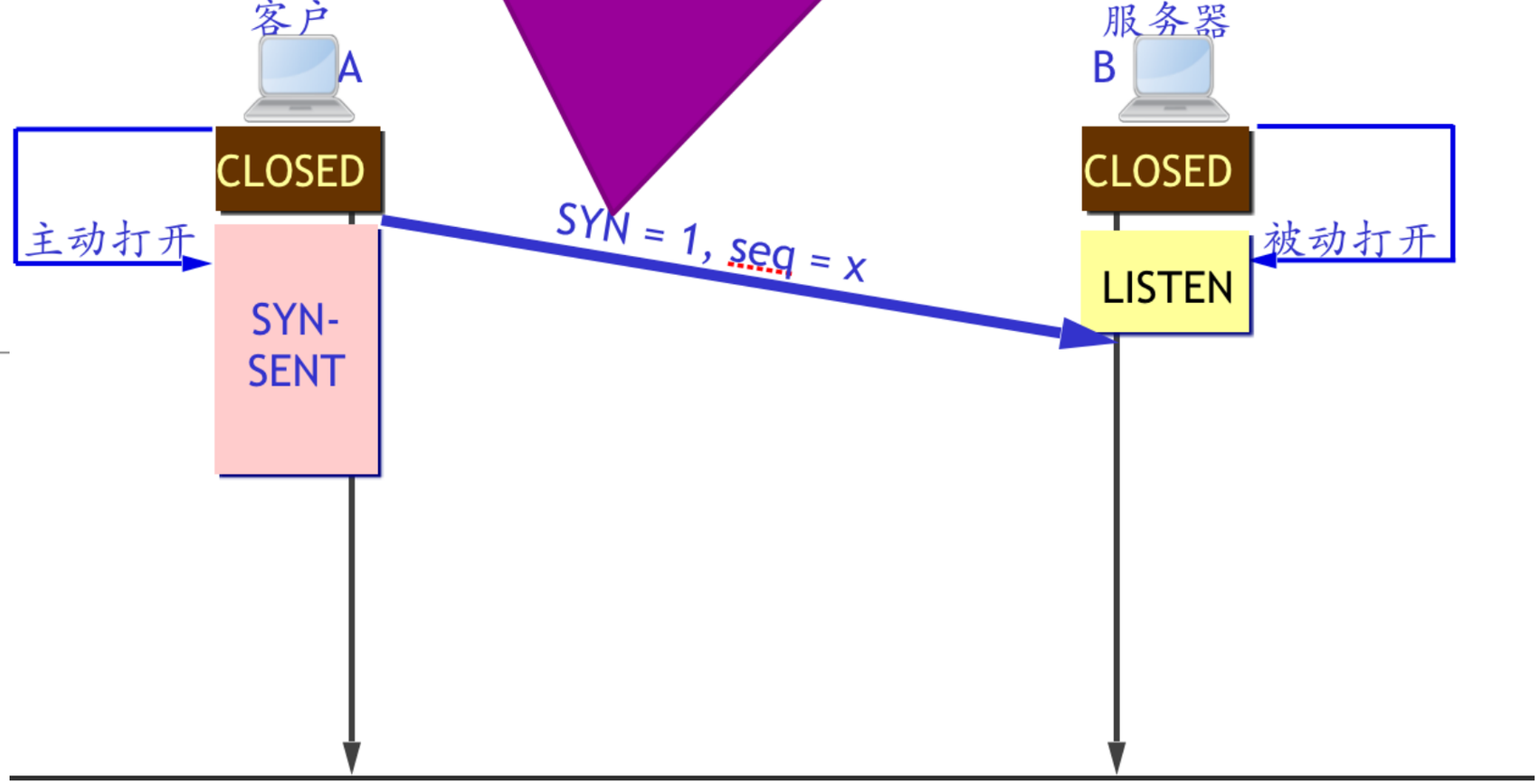
我们要accept一个tsk，但是此时资源还没到位，就先sleep_on

```
void tcp_state_syn_recv(struct tcp_sock *tsk, struct tcp_cb *cb, char *packet)
{
    ...
    tcp_sock_accept_enqueue(tsk);
    wake_up(tsk->parent->wait_accept);
}
```

在tcp_state_syn_recv函数中，当资源到位了，就wake_up，等待资源的函数就可以dequeue，得到需要的资源。

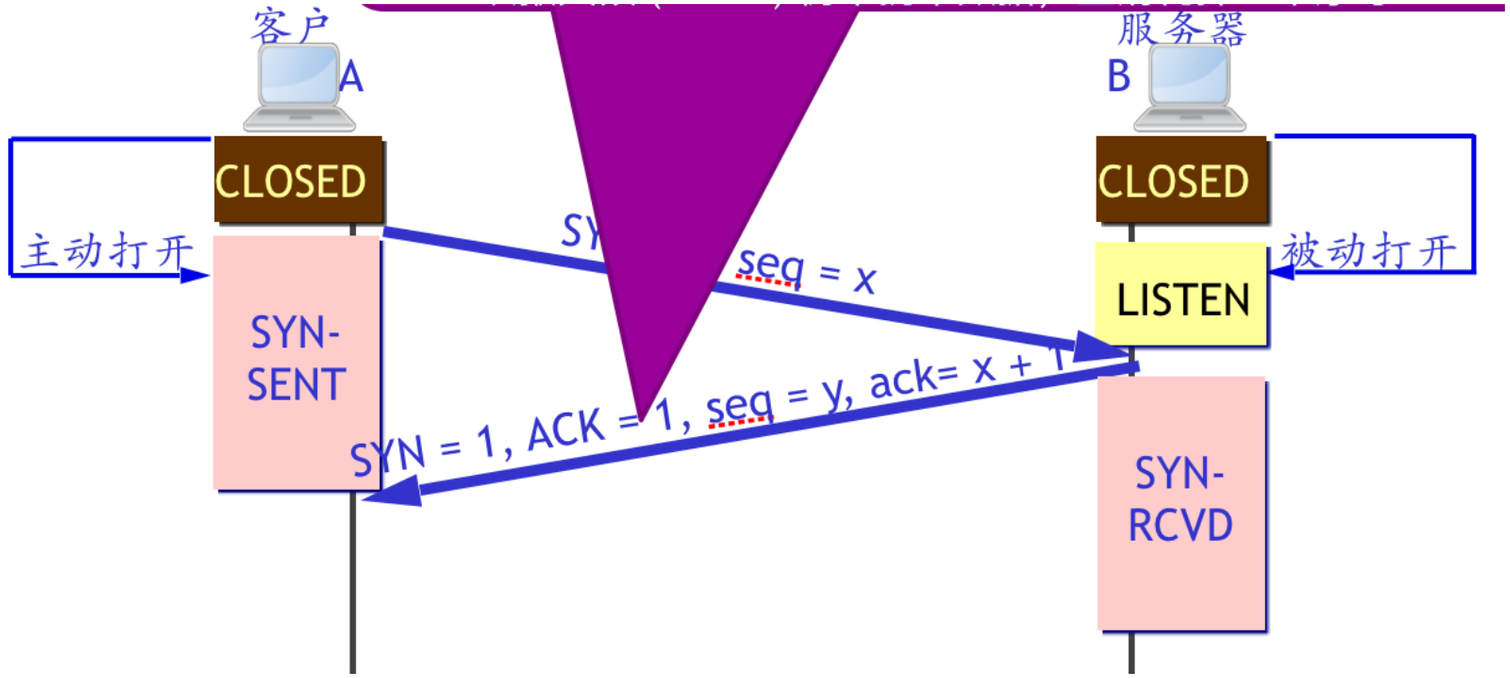
TCP实验一 复盘

- 在接下来的几页PPT中，我会把整个实验的流程复盘一下



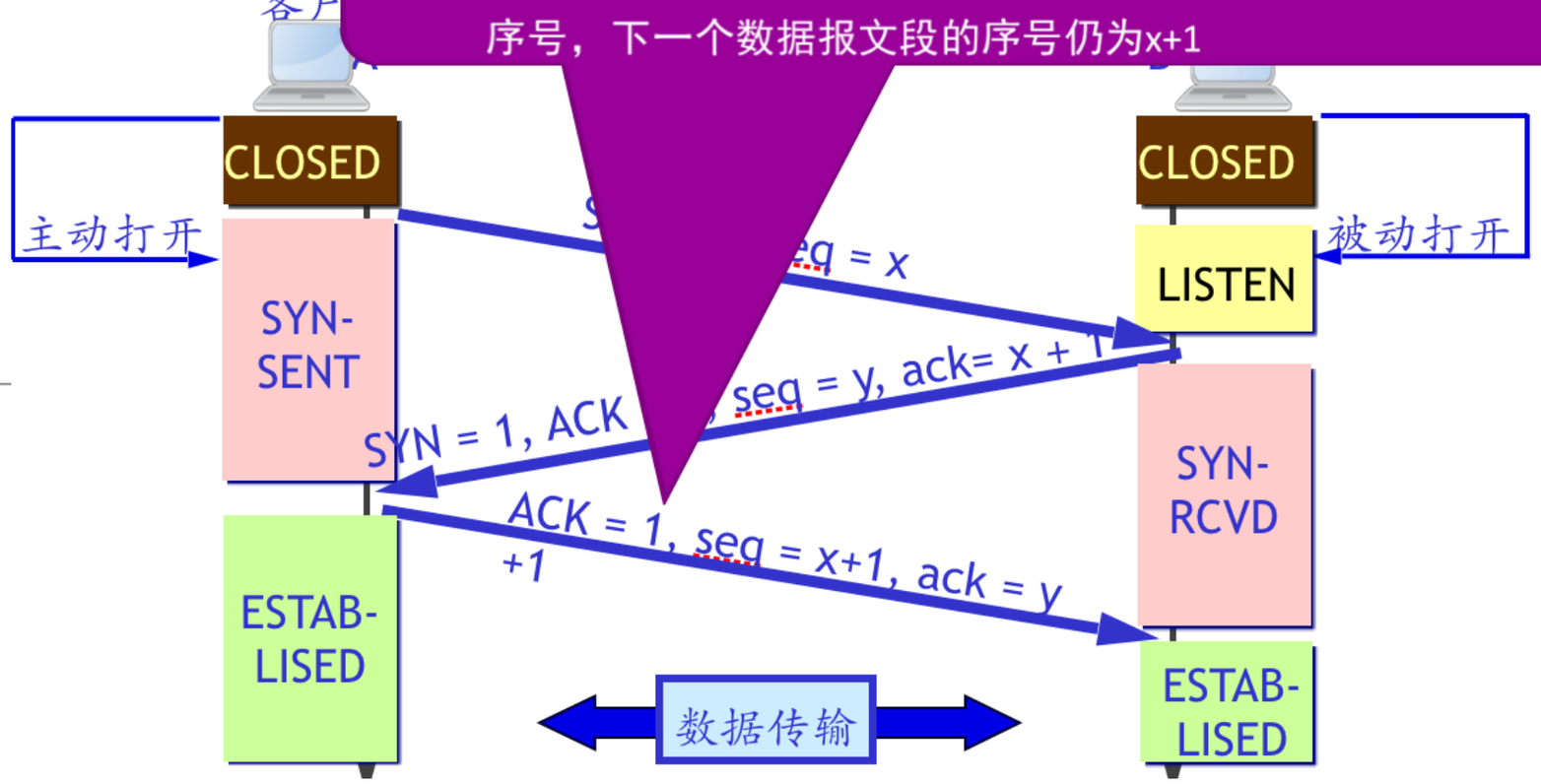
- h2:

```
tsk->rcv_nxt = 0;  //means ack eqauls to 0
tcp_set_state(tsk, TCP_SYN_SENT);
tcp_hash(tsk);
tcp_send_control_packet(tsk, TCP_SYN);
if(!sleep_on(tsk->wait_connect)){
    printf("The Force Awaken\n");
    return 1;
}
```



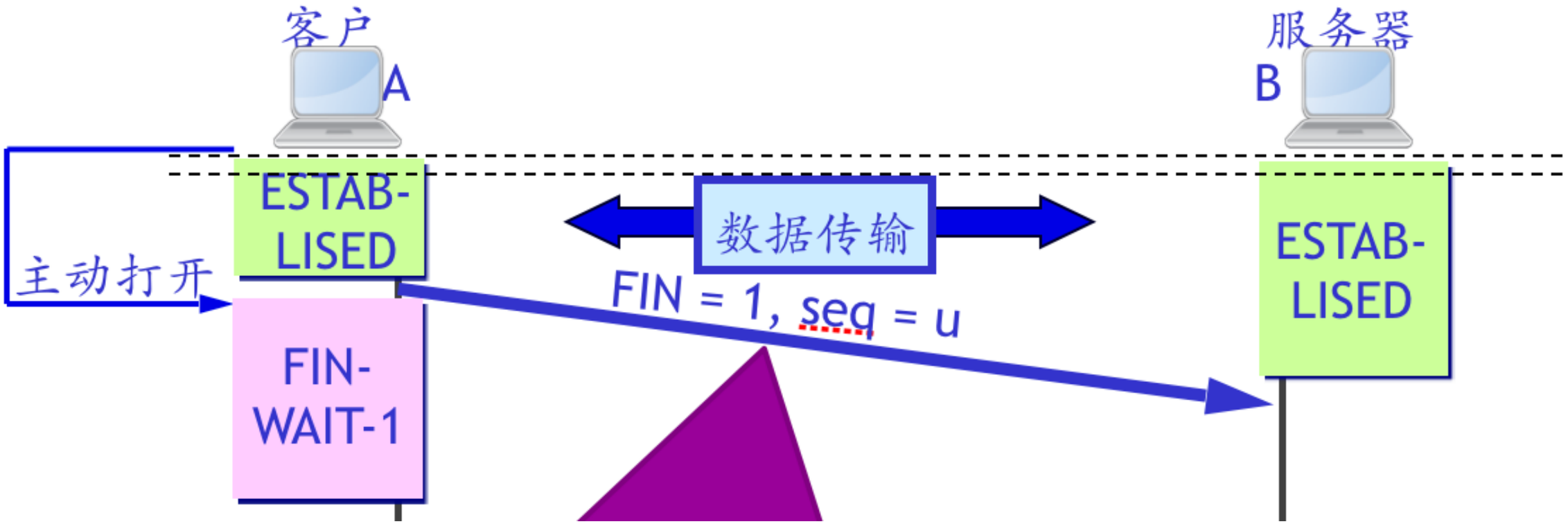
- h1:

```
void tcp_state_listen(struct tcp_sock *tsk, struct tcp_cb *cb, char *packet)
{
    //fprintf(stdout, "TODO: implement this function please listen.\n");
    printf("-----Under tcp_state_listen\n");

    tcp_set_state(tsk, TCP_SYN_RECV);

    struct tcp_sock *new_tsk = malloc(sizeof(struct tcp_sock));
    memcpy(new_tsk, tsk, (sizeof(struct tcp_sock)));
    new_tsk->parent = tsk;
    new_tsk->sk_dip = cb->saddr;
    new_tsk->sk_sip = cb->daddr;
    new_tsk->sk_dport = cb->sport;
    new_tsk->sk_sport = cb->dport;

    new_tsk->rcv_nxt = cb->seq + 1;  //ack = x + 1, see in PPT Chapter5-2, Page41

    //put into its parent's listen_queue
    list_add_tail(&new_tsk->list, &new_tsk->parent->listen_queue);

    tcp_hash(new_tsk);
    printf("Hashed child tsk!\n");

    tcp_send_control_packet(new_tsk, TCP_SYN|TCP_ACK);
    printf("*****Sent a TCP packet: TCP_SYN|TCP_ACK\n");
}
```



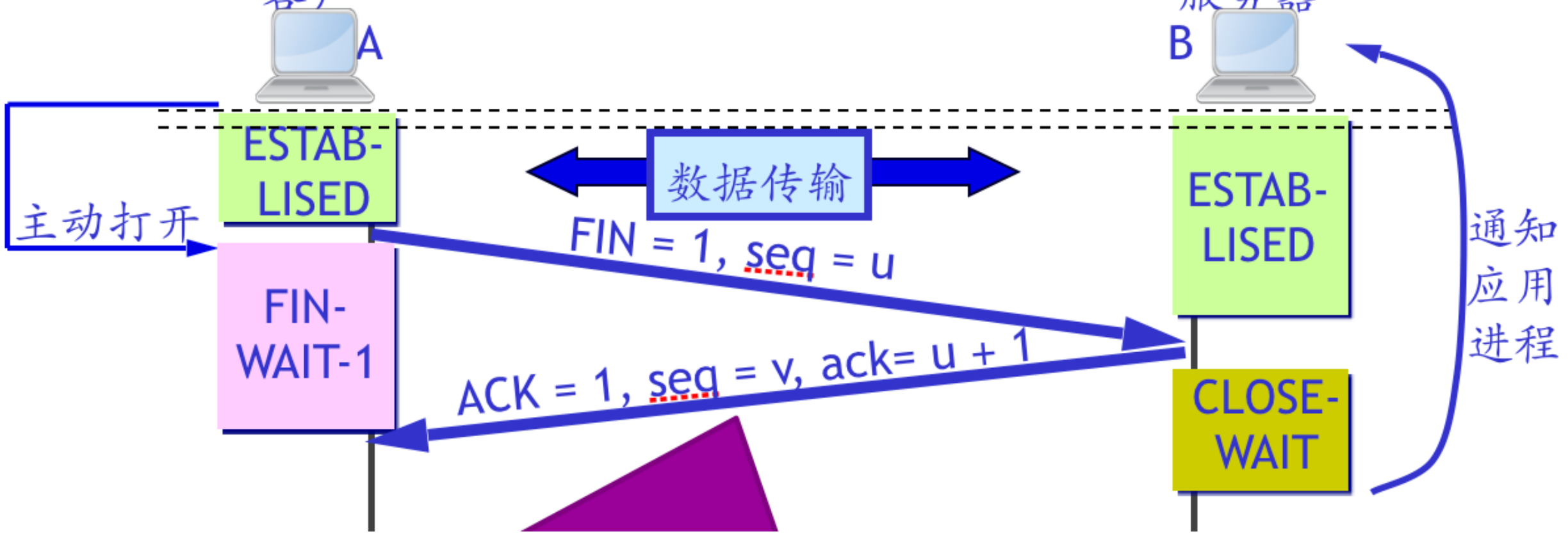
- h2:

```
void tcp_state_syn_sent(struct tcp_sock *tsk, struct tcp_cb *cb, char *packet)
{
    //fprintf(stdout, "TODO: implement this function please.syn_sent\n");
    printf("-----Under tcp_state_syn_sent\n");

    struct tcphdr *the_tcphdr = packet_to_tcp_hdr(packet);
    if(the_tcphdr->flags == (TCP_SYN + TCP_ACK)){
        printf("Receive TCP_SYN|TCP_ACK\n");
        tsk->rcv_nxt = cb->seq + 1;  //ack = y + 1
        tcp_send_control_packet(tsk, TCP_ACK);
        printf("*****Sent a TCP packet: TCP_ACK\n");
        tcp_set_state(tsk, TCP_ESTABLISHED);
        wake_up(tsk->wait_connect);
    }
}
```



- h2:

```
        break;
    case TCP_ESTABLISHED:
        tcp_set_state(tsk, TCP_FIN_WAIT_1);
        tsk->state = TCP_FIN_WAIT_1;
        printf("Switched state already\n");
        tcp_send_control_packet(tsk, TCP_FIN|TCP_ACK);
```

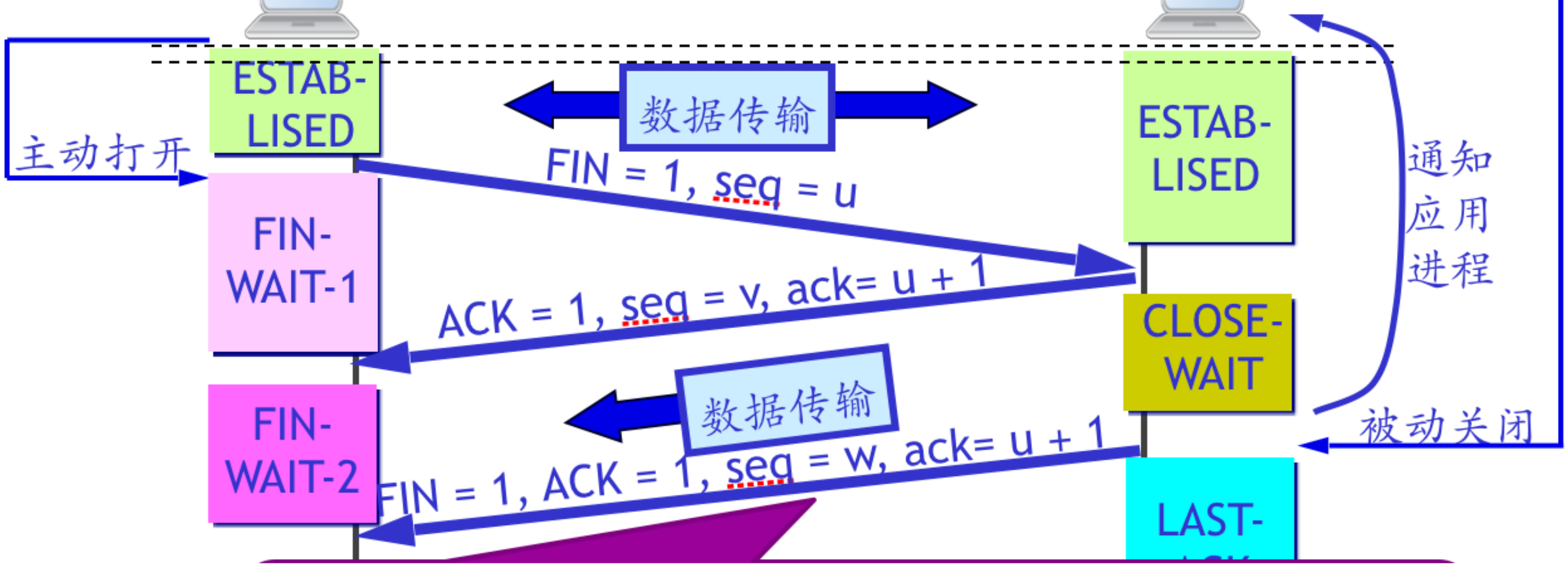


- h1:

```
case TCP_ESTABLISHED:
    //it is only for the server side.
    if(cb->flags & TCP_FIN){
        tsk->rcv_nxt = cb->seq + 1;
        tcp_send_control_packet(tsk, TCP_ACK);
        tcp_set_state(tsk, TCP_CLOSE_WAIT);
    }
    break;
```

- h2:

```
case TCP_FIN_WAIT_1:
    if(cb->flags & TCP_ACK){
        tcp_set_state(tsk, TCP_FIN_WAIT_2);
    }
    break;
```

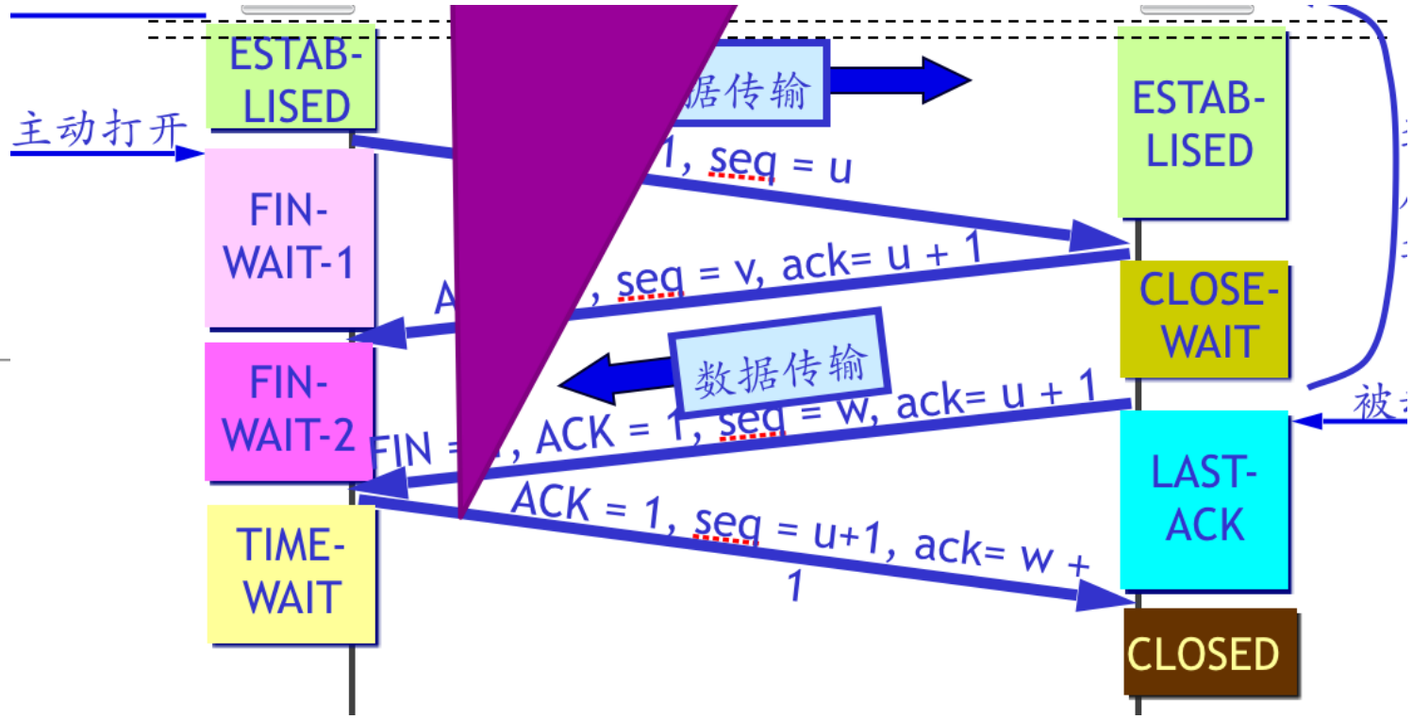


- h1关闭:

```
case TCP_CLOSE_WAIT:
    tcp_set_state(tsk, TCP_LAST_ACK);
    tcp_send_control_packet(tsk, TCP_FIN|TCP_ACK);
    break;
```

- h2:

```
case TCP_FIN_WAIT_2:
    if(cb->flags & TCP_FIN){
        tsk->rcv_nxt = cb->seq + 1;
        tcp_send_control_packet(tsk, TCP_ACK);
        tcp_set_state(tsk, TCP_TIME_WAIT);
        tcp_set_timewait_timer(tsk);
    }
```



- h1:

```
case TCP_LAST_ACK:
    if(cb->flags & TCP_ACK){
        tcp_set_state(tsk, TCP_CLOSED);
    }
    break;
```

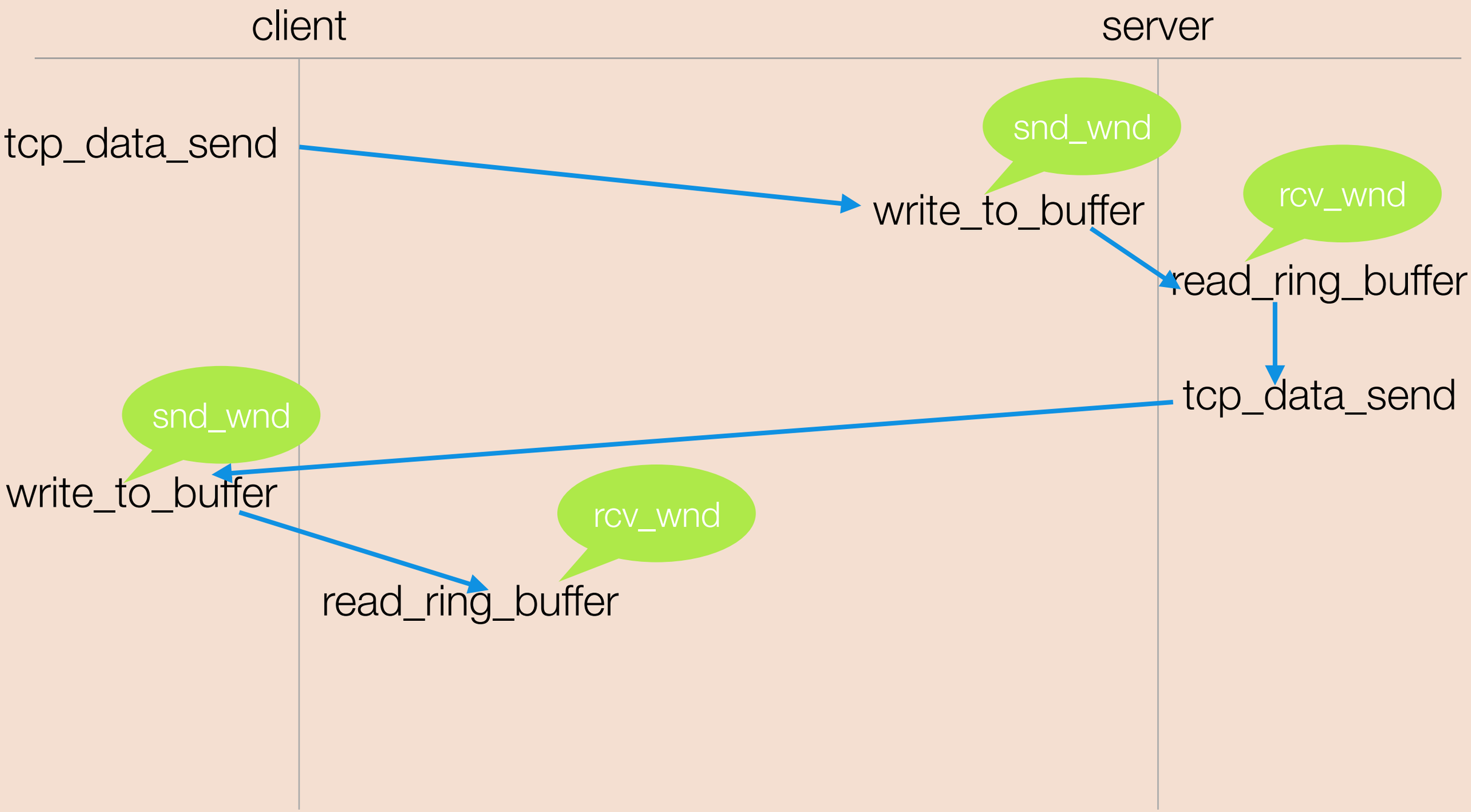
- h2在time_wait线程中被关闭

TCP实验二 复盘

- 在接下来的几页PPT中，我会把整个实验二的流程复盘一下

- 本次实验的核心是：通过调整rcv_wnd来表达自己的接收能力。对端通过rcv_wnd调整它的snd_wnd

TCP实验二 复盘


client
server
tcp_data_send
snd_wnd
write_to_buffer
rcv_wnd
read_ring_buffer
tcp_data_send
snd_wnd
write_to_buffer
rcv_wnd
read_ring_buffer

Thanks!

-Yisong