TCP: Connection and Timer Management

Road to establish a connection

- LISTEN
  - Passive opening, waiting for connection request from others
- SYN_SENT
  - After sending SYN, but before receiving ACK
- SYN_RECV
  - Receiving SYN from others
- ESTABLISHED
  - Finished three-way handshaking

Transition from CLOSED to SYN_SENT

- User/application calls connect() socket API
- sys_socketcall maps it to tcp_v4_connect()
- tcp_v4_connect() invokes tcp_connect()
- tcp_connect() sends SYN packet
- Changes state from CLOSED to SYN_SENT
### Transition from LISTEN to SYN_RECV
- User/application calls `listen()` socket API
- Changing state from CLOSED to LISTEN
- Receiving SYN packet from another party
- Changing state from LISTEN to SYN_RECV
- Sending packet with SYN and ACK

### Transition from SYN_SENT to ESTABLISHED
- Currently in SYN_SENT state
- Receiving packet with SYN and ACK
- Sending ACK
- Changing from SYN_SENT to ESTABLISHED state

### Transition from SYN_SENT to SYN_RECV
- Currently in SYN_SENT state
- Receiving SYN packet (without ACK)
- Sending SYN and ACK
- Changing to SYN_RECV state
- Simultaneous connection establishment

### Transition from SYN_RECV to ESTABLISHED
- Currently in SYN_RECV state
- (we sent SYN/ACK)
- Receiving ACK packet
- Changing to ESTABLISHED state
tcp_rcv_state_process()

- Big function to handle TCP state transitions
- Specific behavior depending on current state and packet received

```c
if (th->ack) {
    switch (sk->state) {
        case TCP_SYN_RECV:
            ...
        tcp_set_state(sk, TCP_ESTABLISHED);
    }
}
```

Tearing down a connection

- Two ways to terminate a connection
  - Graceful close: all data transmitted
  - Abort: data can get lost
- Closing related state
  - FIN_WAIT_1: We close but not receive ACK
  - FIN_WAIT_2: We close and receive ACK
  - CLOSING: Both FINed, waiting for ACK
  - TIME_WAIT: Graceful close (wait some time)
  - CLOSE_WAIT: They close and we ACK
  - LAST_ACK: They close, then we close, waiting for ACK
  - CLOSED: connection is now closed

Transition from ESTABLISHED to FIN_WAIT_1

- User/application calls close() socket API
- sys_socketcall() maps it to sys_shutdown()
- Which calls tcp_close() (in TCP case)
- Sending FIN packet
- Changing state from ESTABLISHED to FIN_WAIT_1

Transition from ESTABLISHED to CLOSE_WAIT

- Currently in ESTABLISHED state
- Receiving FIN packet
- Sending ACK to FIN
- Changing state from ESTABLISHED to CLOSE_WAIT

- tcp_fin()
Transition from CLOSE_WAIT to LAST_ACK

- Currently in CLOSE_WAIT state
- (receiving FIN from another party)
- Finally we finish data transmission, we also close
- We send FIN packet
- Changing to LAST_ACK state to wait for the ACK packet

Transition from FIN_WAIT_1 to FIN_WAIT_2

- Currently we are in FIN_WAIT_1
- (We sent FIN)
- Receiving ACK (to FIN)
- Changing state from FIN_WAIT_1 to FIN_WAIT_2
- (we have not received FIN from another party)

Transition from FIN_WAIT_2 to TIME_WAIT

- We are in FIN_WAIT_2
- (we sent FIN and ACKed)
- Receiving FIN, sending ACK
- Changing state from FIN_WAIT_2 to TIME_WAIT

- For graceful close, wait for 2 MSL

Transition from FIN_WAIT_1 to TIME_WAIT

- Currently we are in FIN_WAIT_1
- (we sent FIN)
- Receiving ACK and FIN
- Changing state from FIN_WAIT_1 to TIME_WAIT
Transition from FIN_WAIT_1 to CLOSING

- Currently in FIN_WAIT_1 state
- (we sent FIN but not ACKed)
- Receiving FIN
- Sending ACK
- Changing from FIN_WAIT_1 to CLOSING state
- Waiting for ACK

Transition from CLOSING to TIME_WAIT

- Currently in CLOSING state
- Both sides FINed
- We Acked another party
- We waiting for being ACKed by another party
- Receiving ACK
- Changing state from CLOSING to TIME_WAIT

Timer management

- Seven different timers are maintained in TCP
  - SYNACK: waiting for ACK to our SYN
  - Retransmit: for data retransmission, exponential back-off
  - Delay ACK: hoping for piggy-back ACK
  - Keepalive: checking if a connection alive
  - Probe: testing if zero window size still applies
  - FIN_WAIT_2: switch to CLOSED if no FIN received
  - TWKill: how long to stay in TIME_WAIT

Timer data structure

```c
struct timer_list {
    struct list_head list;
    unsigned long expires;
    unsigned long data;
    void (*function)(unsigned long);
    volatile int running;
}
```