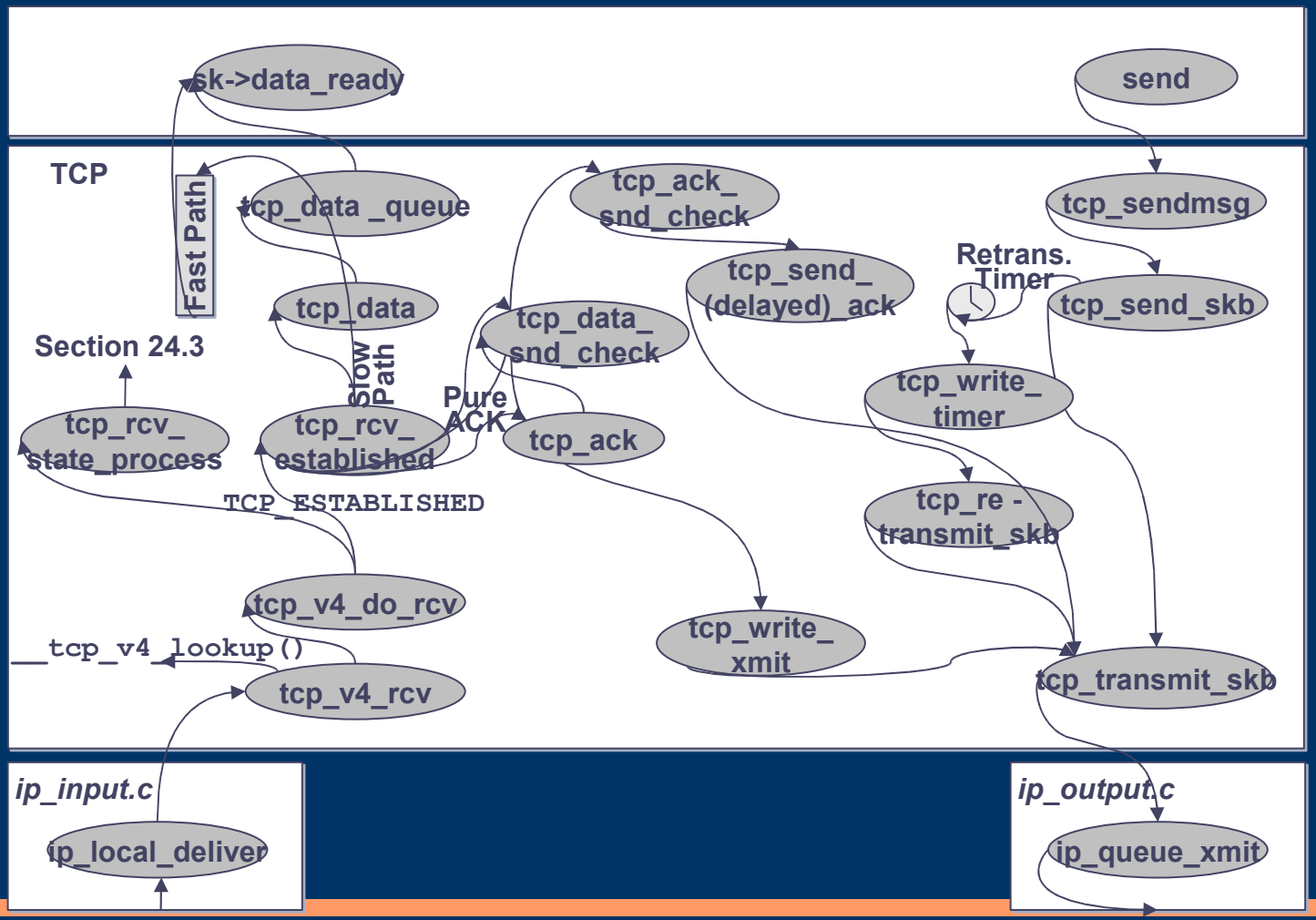
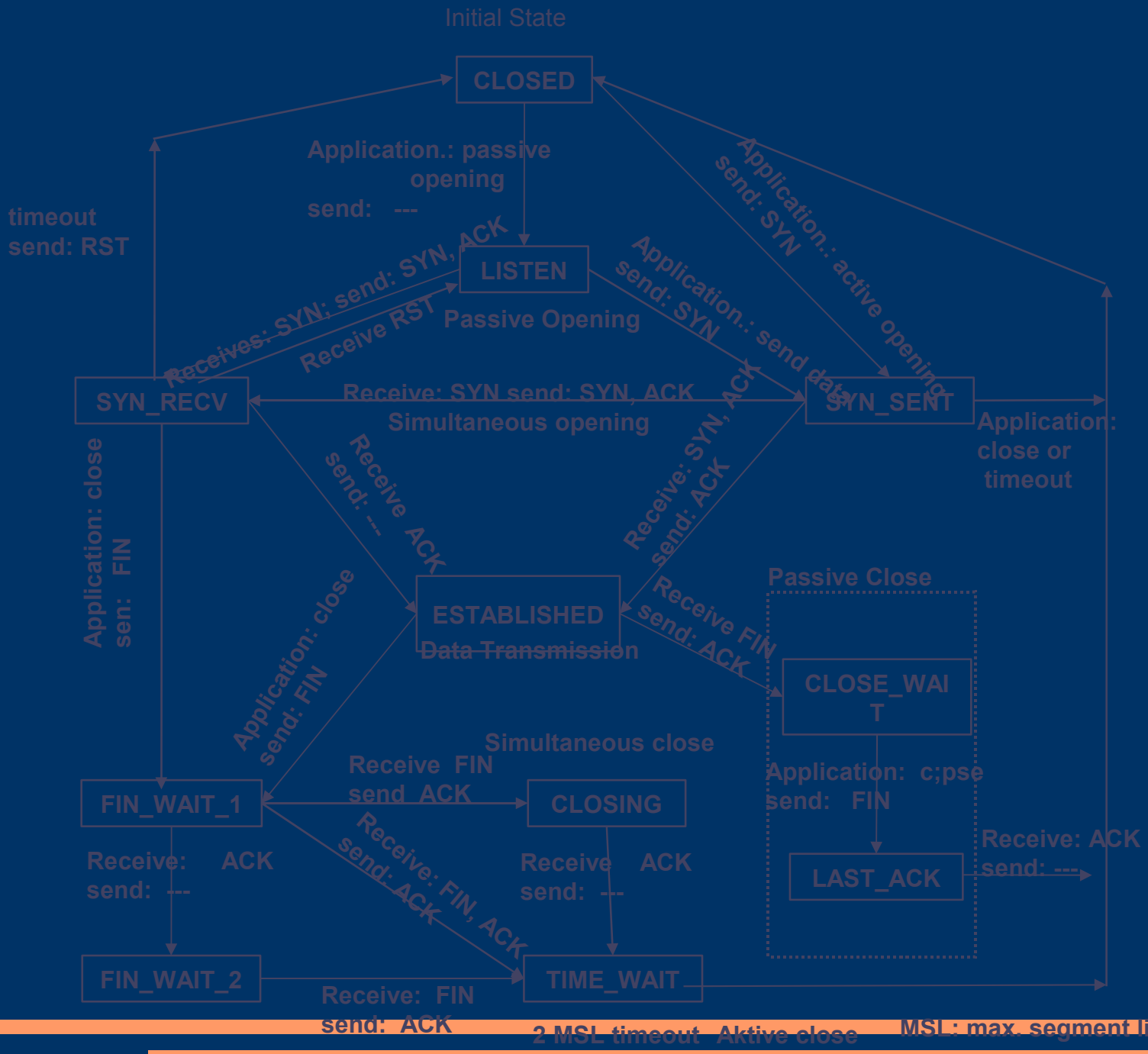
TCP Implementation in Linux




Initial State
CLOSED
Application.: passive opening
send: ---
timeout
send: RST
LISTEN
Receives: SYN; send: SYN, ACK
Receive RST
Passive Opening
Application.: send data
send: SYN
Application.: active opening
send: SYN
SYN_RECV
Receive: SYN send: SYN, ACK
Simultaneous opening
SYN_SENT
Application: close or timeout
Application: close
sen: FIN
Receive ACK
send: ---
Receive: SYN, ACK
send: ACK
Application: close
send: FIN
ESTABLISHED
Data Transmission
Passive Close
Receive FIN
send: ACK
CLOSE_WAI T
Simultaneous close
Receive FIN
send ACK
Application: c;pse
send: FIN
FIN_WAIT_1
CLOSING
Receive: ACK
send: ---
Receive: ACK
send: ---
Receive: FIN, ACK
send: ACK
LAST_ACK
Receive: ACK
send: ---
FIN_WAIT_2
TIME_WAIT
Receive: FIN
send: ACK
2 MSL timeout Aktive close
MSL: max. segment life

tcp_rcv_state_process()

- Handles mainly state transitions and connection management.
- For example, if the packet received contains an ACK flag,
  - If state=*SYN_RECV*, then state← *ESTABLISHED*, and the acknowledgement is processed.
  - If state=*FIN_WAIT_1*, then state← *FIN-WAIT2* and the *TIMEWAIT* timer is set.
  - If state=*CLOSING*, then state← *TIMEWAIT*
  - If state=*LAST_ACK*, state←*CLOSED* and the socket is reset.

Transition CLOSED→ SYN_SENT

- *connect()* → *tcp_v4_connect()* → *tcp_connect()*
- *tcp_connect()* changes the state to *SYN_SENT* by invoking *tcp_set_state(sk, TCP_SYN_SENT)*.

Transition LISTEN→SYN_RECV

- The LISTEN state is set when the server application invokes listen().
- When a SYN is received,
  - *tcp_rcv_state_process()→tcp_v4_hnd_req()→ tcp_check_req()→ tcp_v4_syn_recv_sock()→ tcp_create_openreq_child().*
  - In *tcp_create_openreq_child(),* the state is set to *TCP_SYN_RECV.*
  - *tcp→af_specific→conn_request()* (pointed to *tcp_v4_conn_request*()) is invoked to specify the initial SN.
  - *tcp_v4_send_synack()* sends a reply with the ACK and SYN flags set.

Transition SYN_SENT→ ESTABLISHED

- *tcp_rcv_state_process()→ tcp_rcv_synsent_state_process()*
- if *(th→ack) {*

  ……

  if (!th→syn) goto discard;

  ……..

  tcp_set_state(sk, TCP_ESTABLISHED);

  ……..

  tcp_schedule_ack(tp);

  ……..

  }

Transition SYN_SENT→ SYN_RECEIVED

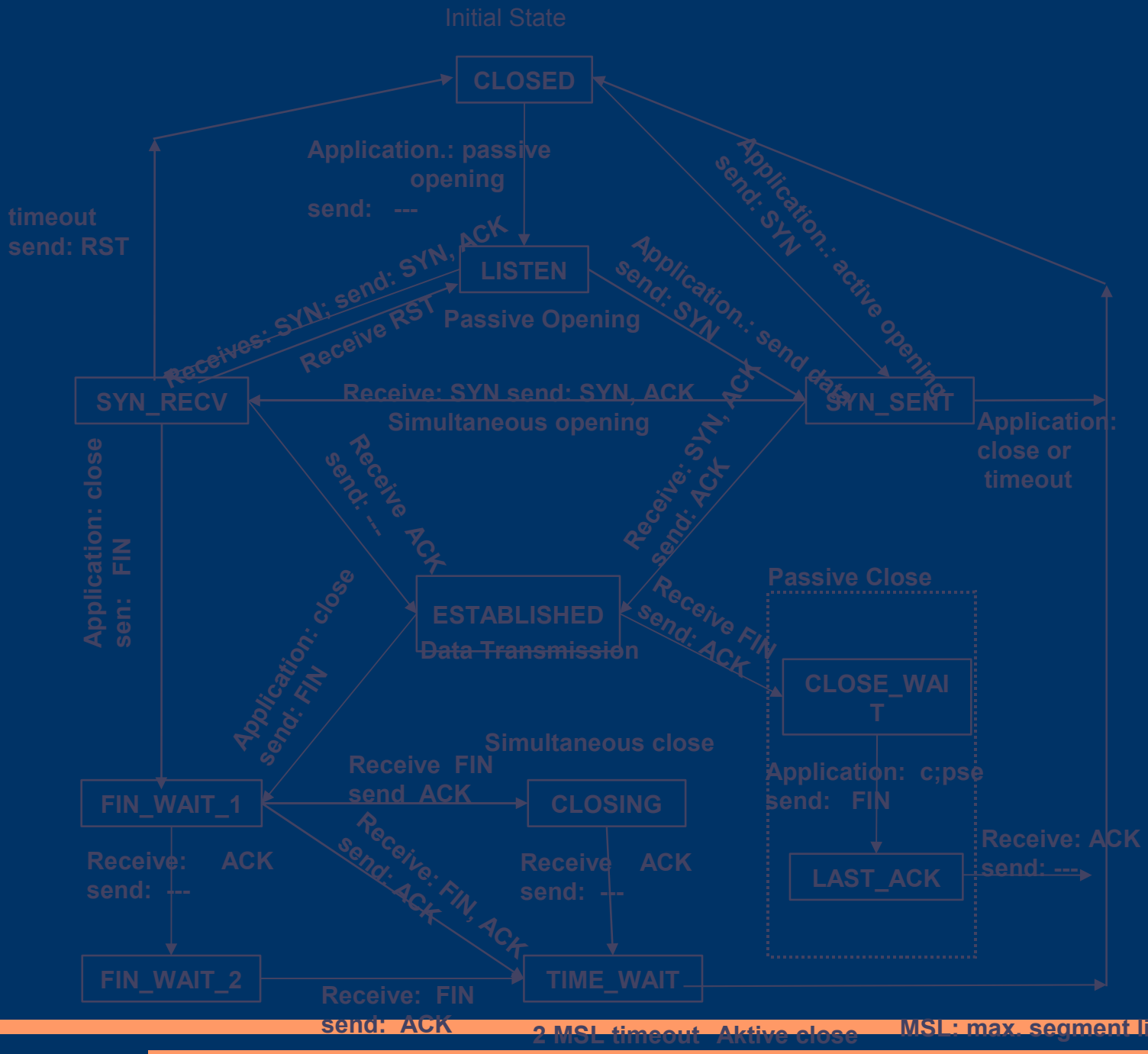
- This takes place in the case of simultaneous opening.
- *tcp_rcv_state_process()→ tcp_rcv_synsent_state_process()*
- *if (th→syn) {*

```
tcp_set_state(sk, TCP_SYN_RECV);
……
tcp_set_synack(sk);
……
}
```

Transition SYN_RECV→ ESTABLISHED

- tcp_rcv_state_process() processes this case.
- If (th→ack) {

```
switch(sk→state) {
        case TCP_SYN_RECV:
                ….
                tcp_set_state(sk, TCP_ESTALISHED);
}
}
```


Initial State
CLOSED
Application.: passive opening
send: ---
timeout
send: RST
LISTEN
Receives: SYN; send: SYN, ACK
Receive RST
Passive Opening
Application.: send data
send: SYN
Application.: active opening
send: SYN
SYN_RECV
Receive: SYN send: SYN, ACK
Simultaneous opening
SYN_SENT
Application: close or timeout
Application: close
sen: FIN
Receive ACK
send: ---
Receive: SYN, ACK
send: ACK
Application: close
send: FIN
ESTABLISHED
Data Transmission
Passive Close
Receive FIN
send: ACK
CLOSE_WAI T
Simultaneous close
Receive FIN
send ACK
Application: c;pse
send: FIN
FIN_WAIT_1
CLOSING
Receive: ACK
send: ---
Receive: ACK
send: ---
Receive: FIN, ACK
send: ACK
LAST_ACK
Receive: ACK
send: ---
FIN_WAIT_2
TIME_WAIT
Receive: FIN
send: ACK
2 MSL timeout Aktive close
MSL: max. segment life

Transition ESTABLISHED→ FIN_WAIT_1

- *close()→ tcp_close()→ tcp_close_state().*
- In *tcp_close_state(),* the state is changed from *ESTABLISHED* to *FIN_WAIT_1*.

Transition FIN_WAIT_1→ FIN_WAIT_2

- In *tcp_rcv_state_process()*
- *if (th→ack) {*

```
switch(sk→state) {
……
case TCP_FIN_WAIT1:
    ……
    tcp_set_state(sk,TCP_FIN_WAIT2);
    ……
  }
}
```

Transition FIN_WAIT2→ TIME_WAIT

- In tcp_fin(),
  switch(sk→state) {
  …..
  case TCP_FIN_WAIT2:
  tcp_send_ack(sk);
  tcp_time_wait(sk, TCP_TIME_WAIT, 0);
  break;
  …..
  }

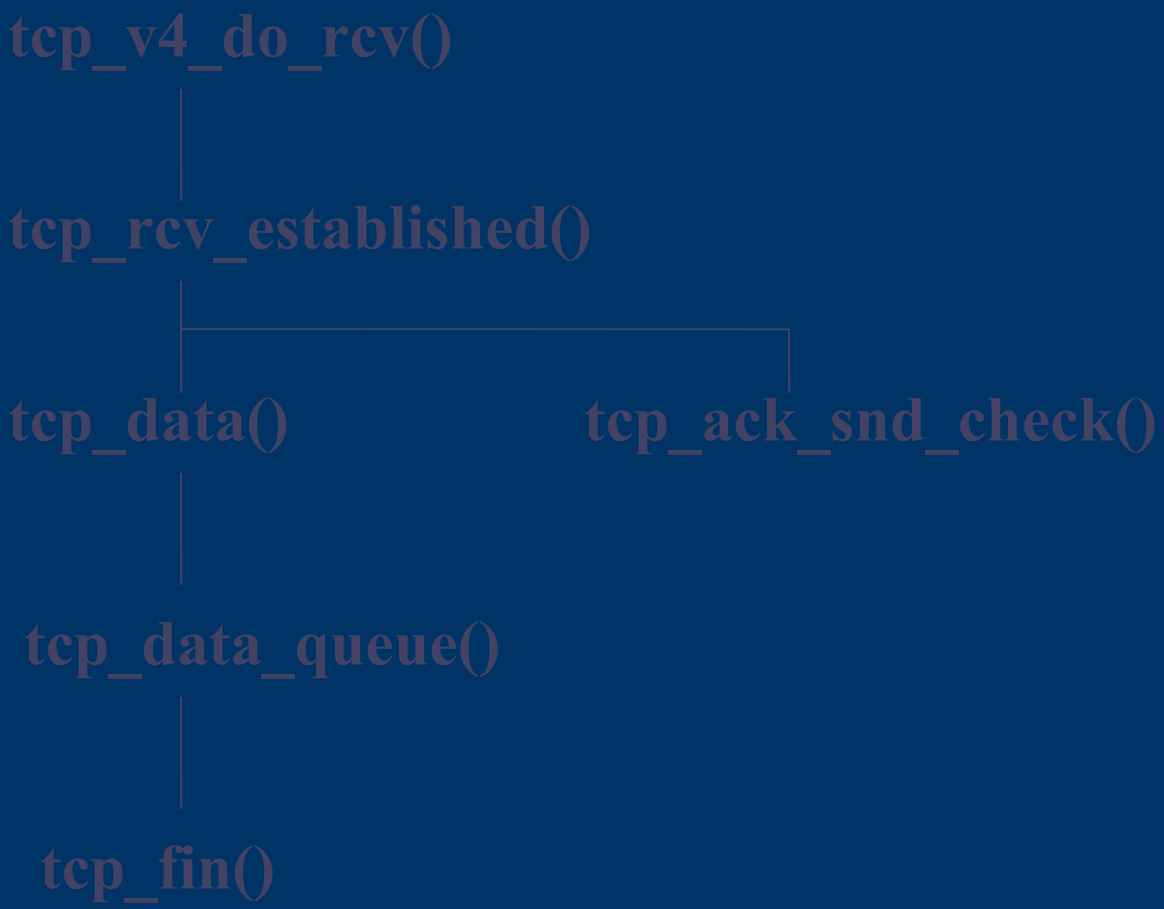
tcp_v4_do_rcv()

tcp_rcv_state_process()

tcp_data_queue()

tcp_fin()

Transition ESTABLISHED→ CLOSE_WAIT



Transition ESTABLISHED→ CLOSE_WAIT

- In tcp_fin(),

```
switch(sk→state) {
    case TCP_SYN_RECV:
    case TCP_ESTABLISHED:
        tcp_set_state(sk,TCP_CLOSE_WAIT);
        if (th→rst)
                sk→shutdown= SHUTDOWN_MASK;
        break;
```

- In tcp_ack_snd_check(), a packet is sent with the ACK flag set.

Transition CLOSE_WAIT→ LAST_ACK

- TCP on the other hand has closed the connection. Now we are waiting for our TCP instance to close the connection.
- *tcp_close()→ tcp_close_state()*
- In *tcp_close_state(), tcp_set_state(sk, LaST_ACK)* is invoked.