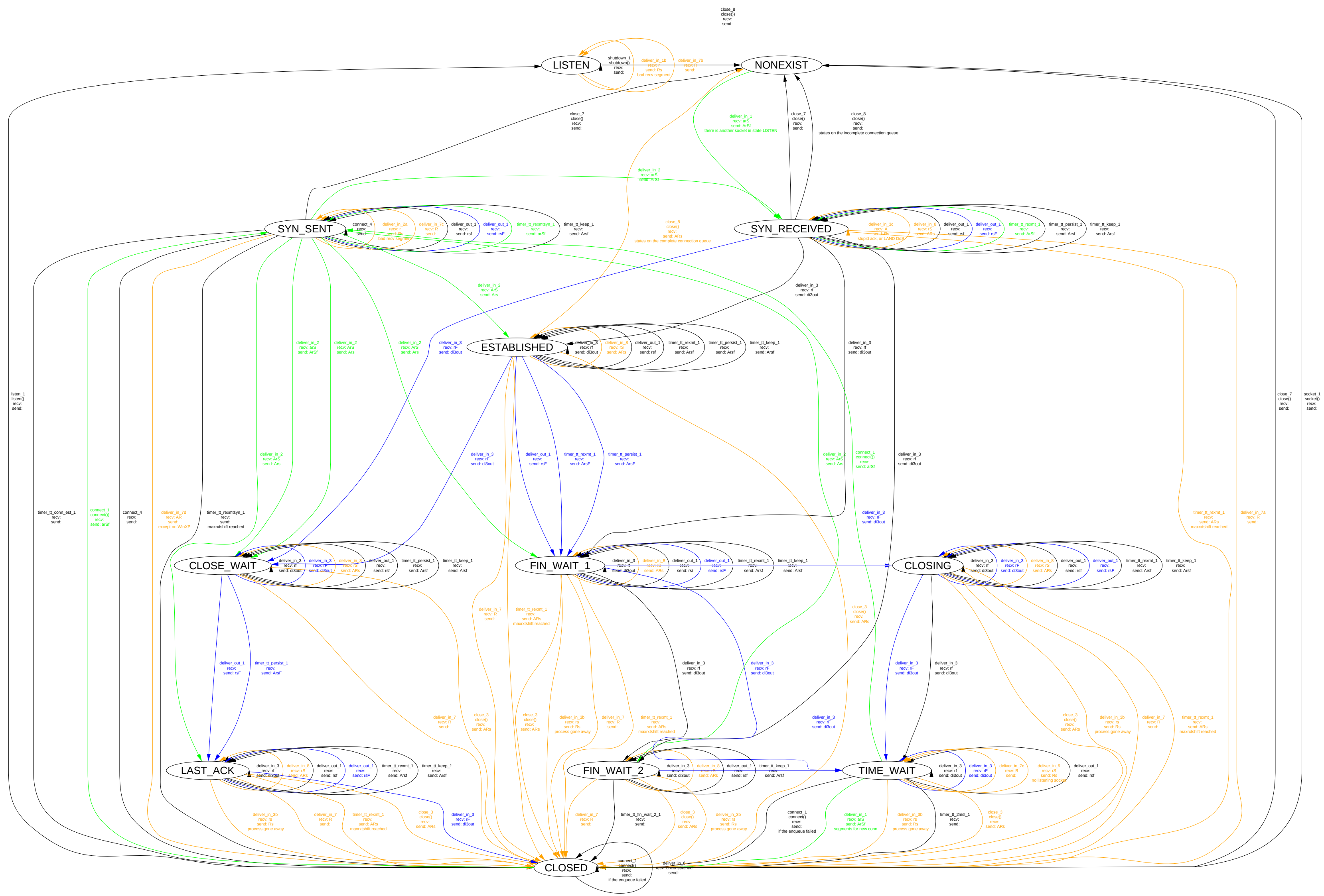
TCP: an approximation to the real state diagram



http://www.cl.cam.ac.uk/users/pes20/Netsem
March 18, 2005

What Is This?

This graph shows an approximation to the Host Transition System of the TCP specification

TCP, UDP, and Sockets: rigorous and experimentally-validated behavioural specification. Volume 1: Overview. Volume 2: The Specification. Steven Bishop, Matthew Fairbairn, Michael Norrish, Peter Sewell, Michael Smith, and Keith Wansbrough. 2005.

The states are the classic 'TCP states', though note that these are only a tiny part of the protocol endpoint state, in the specification or in implementations. The transitions are an over-approximation to the set of all the transitions in the model which (1) affect the TCP state of a socket, and/or (2) involve processing segments from the host's input queue or adding them to its output queue, except that transitions involving ICMPs are omitted, as are transitions arising from the pathological BSD behaviour in which arbitrary sockets can be moved to LISTEN states. Transitions are labelled by their Host LTS rule name (e.g. *socket_1*, *deliver_in_3*, etc.), any socket call involved (e.g. close()), and constraints on the flags of any TCP segment received and sent, with e.g. R indicating that RST is set and r indicating RST is clear. Transitions involving segments (either inbound or outbound) with RST set are coloured orange; others that have SYN set are coloured green; others that have FIN set are coloured blue; others are coloured black. The FIN indication includes the case of FINs that are constructed by reassembly rather than appearing in a literal segment.

The graph is based on data extracted manually from the HOL specification. The data does not capture all the invariants of the model, so some depicted transitions may not be reachable in the model (or in practice). Similarly, the constraints on flags shown may be overly weak.

Transition Rules

close_3	Successful abortive close of a synchronised socket
close_7	Successfully close the last file descriptor for a socket in the CLOSED, SYN_SENT or SYN_RECEIVED states.
close_8	Successfully close the last file descriptor for a listening TCP socket
connect_1	Begin connection establishment by creating a SYN and trying to enqueue it on host's outqueue
connect_4	Fail: socket has pending error
deliver_in_1	Passive open: receive SYN, send SYN,ACK
deliver_in_1b	For a listening socket, receive and drop a bad datagram and either generate a RST segment or ignore it. Drop the incoming segment if the socket's queue of incomplete connections is full.
deliver_in_2	Completion of active open (in SYN_SENT receive SYN,ACK and send ACK) or simultaneous open (in SYN_SENT receive SYN and send SYN,ACK)
deliver_in_2a	Receive bad or boring datagram and RST or ignore for SYN_SENT socket
deliver_in_3	Receive data, FINs, and ACKs in a connected state
deliver_in_3b	Receive data after process has gone away
deliver_in_3c	Receive stupid ACK or LAND DoS in SYN_RECEIVED state
deliver_in_6	Receive and drop (silently) a sane segment that matches a CLOSED socket
deliver_in_7	Receive RST and zap non-{CLOSED; LISTEN; SYN_SENT; SYN_RECEIVED; TIME_WAIT} socket
deliver_in_7a	Receive RST and zap SYN_RECEIVED socket
deliver_in_7b	Receive RST and ignore for LISTEN socket
deliver_in_7c	Receive RST and ignore for SYN_SENT(unacceptable ack) or TIME_WAIT socket
deliver_in_7d	Receive RST and zap SYN_SENT(acceptable ack) socket
deliver_in_8	Receive SYN in non-{CLOSED; LISTEN; SYN_SENT; TIME_WAIT} state
deliver_in_9	Receive SYN in TIME_WAIT state if there is no matching LISTEN socket or sequence number has not increased
deliver_out_1	Common case TCP output
listen_1	Successfully put socket in LISTEN state
listen_1c	Successfully put socket in the LISTEN state from any non-{CLOSED; LISTEN} state on FreeBSD
shutdown_1	Shut down read or write half of TCP connection
socket_1	Successfully return a new file descriptor for a fresh socket
timer_tt_2msl_1	2*MSL timer expires
timer_tt_conn_est_1	connection establishment timer expires
timer_tt_fin_wait_2_1	FIN_WAIT_2 timer expires
timer_tt_keep_1	keepalive timer expires
timer_tt_persist_1	persist timer expires
timer_tt_rexmt_1	retransmit timer expires
timer_tt_rexmtsyn_1	SYN retransmit timer expires

The RFC793 Original

```
                              Transmission Control Protocol
                                Functional Specification

                              +---------+ ---------\      active OPEN
                              |  CLOSED |            \    -----------
                              +---------+<---------\   \   create TCB
                                |     ^              \   \  snd SYN
                   passive OPEN |     |   CLOSE        \   \
                   ------------ |     | ----------       \   \
                    create TCB  |     | delete TCB         \   \
                                V     |                      \   \
                              +---------+            CLOSE    |    \
                              |  LISTEN |          ---------- |     |
                              +---------+          delete TCB |     |
                   rcv SYN      |     |     SEND              |     |
                  -----------   |     |    -------            |     V
 +---------+      snd SYN,ACK  /       \   snd SYN          +---------+
 |         |<-----------------           ------------------>|         |
 |   SYN   |                    rcv SYN                     |   SYN   |
 |   RCVD  |<-----------------------------------------------|   SENT  |
 |         |                    snd ACK                     |         |
 |         |------------------           -------------------|         |
 +---------+   rcv ACK of SYN  \       /  rcv SYN,ACK       +---------+
   |           --------------   |     |   -----------
   |                  x         |     |     snd ACK
   |                            V     V
   |  CLOSE                   +---------+
   | -------                  |  ESTAB  |
   | snd FIN                  +---------+
   |                   CLOSE    |     |    rcv FIN
   V                  -------   |     |    -------
 +---------+          snd FIN  /       \   snd ACK          +---------+
 |  FIN    |<-----------------           ------------------>|  CLOSE  |
 | WAIT-1  |------------------                              |   WAIT  |
 +---------+          rcv FIN  \                            +---------+
   | rcv ACK of FIN   -------   |                            CLOSE  |
   | --------------   snd ACK   |                           ------- |
   V        x                   V                           snd FIN V
 +---------+                  +---------+                   +---------+
 |FINWAIT-2|                  | CLOSING |                   | LAST-ACK|
 +---------+                  +---------+                   +---------+
   |                rcv ACK of FIN |                 rcv ACK of FIN |
   |  rcv FIN       -------------- |    Timeout=2MSL -------------- |
   |  -------              x       V    ------------        x       V
    \ snd ACK                 +---------+delete TCB         +---------+
     ------------------------>|TIME WAIT|------------------>| CLOSED  |
                              +---------+                   +---------+

                      TCP Connection State Diagram
                               Figure 6.                September 1981
```