implementation of a trade-off between memory and messages to provide information for this checking.

The simplest three-way handshake is shown in figure 7 below. The figures should be interpreted in the following way. Each line is numbered for reference purposes. Right arrows (-->) indicate departure of a TCP segment from TCP A to TCP B, or arrival of a segment at B from A. Left arrows (<--), indicate the reverse. Ellipsis (...) indicates a segment which is still in the network (delayed). An "XXX" indicates a segment which is lost or rejected. Comments appear in parentheses. TCP states represent the state AFTER the departure or arrival of the segment (whose contents are shown in the center of each line). Segment contents are shown in abbreviated form, with sequence number, control flags, and ACK field. Other fields such as window, addresses, lengths, and text have been left out in the interest of clarity.

```
    TCP A                                                TCP B

1.  CLOSED                                               LISTEN

2.  SYN-SENT    --> <SEQ=100><CTL=SYN>               --> SYN-RECEIVED

3.  ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK>  <-- SYN-RECEIVED

4.  ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK>       --> ESTABLISHED

5.  ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK><DATA> --> ESTABLISHED
```

Basic 3-Way Handshake for Connection Synchronization

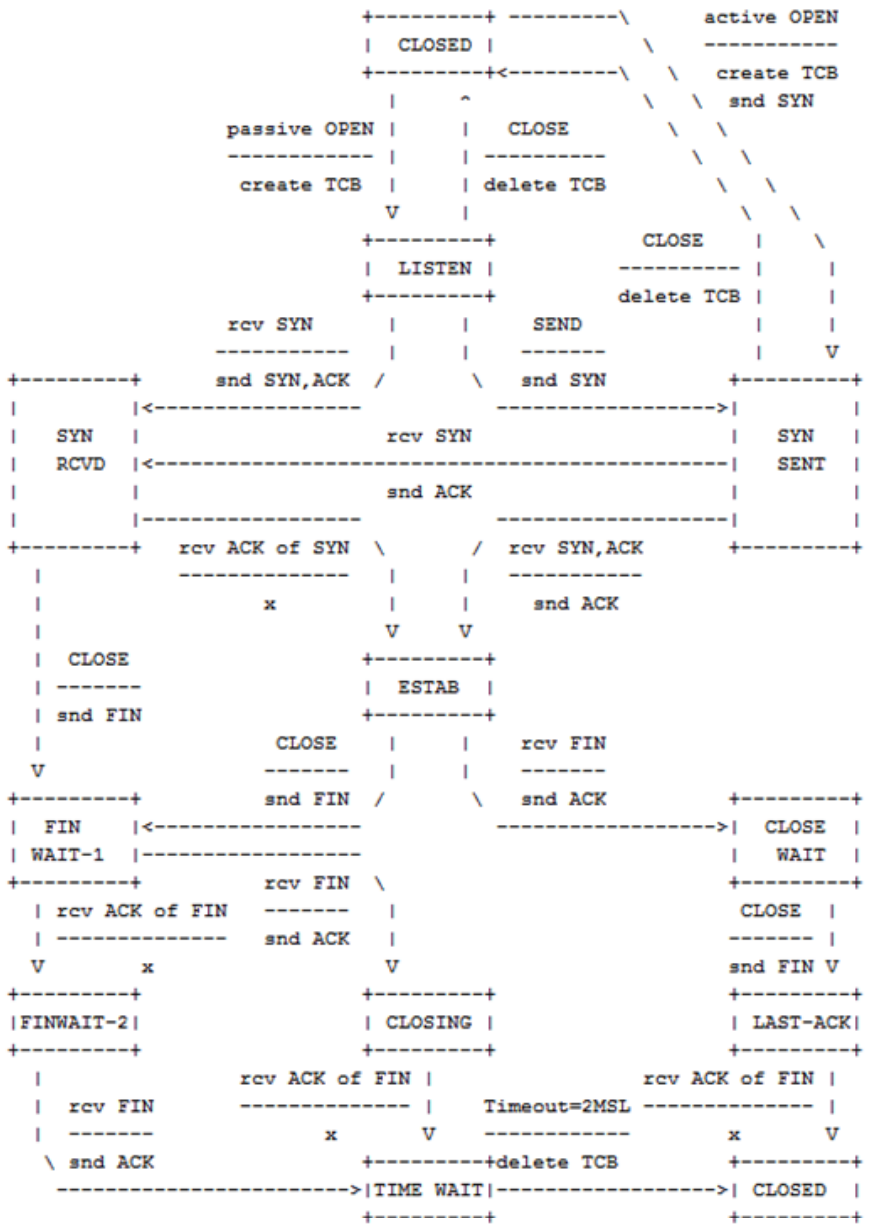
Figure 7.

In line 2 of figure 7, TCP A begins by sending a SYN segment indicating that it will use sequence numbers starting with sequence number 100. In line 3, TCP B sends a SYN and acknowledges the SYN it received from TCP A. Note that the acknowledgment field indicates TCP B is now expecting to hear sequence 101, acknowledging the SYN which occupied sequence 100.

At line 4, TCP A responds with an empty segment containing an ACK for TCP B's SYN; and in line 5, TCP A sends some data. Note that the sequence number of the segment in line 5 is the same as in line 4 because the ACK does not occupy sequence number space (if it did, we would wind up ACKing ACK's!).

IETF RFC 793

Example of Sequence Diagram



TCP Connection State Diagram
Figure 6.

Finite State Diagramm for TCP

(IETF RFC 793)

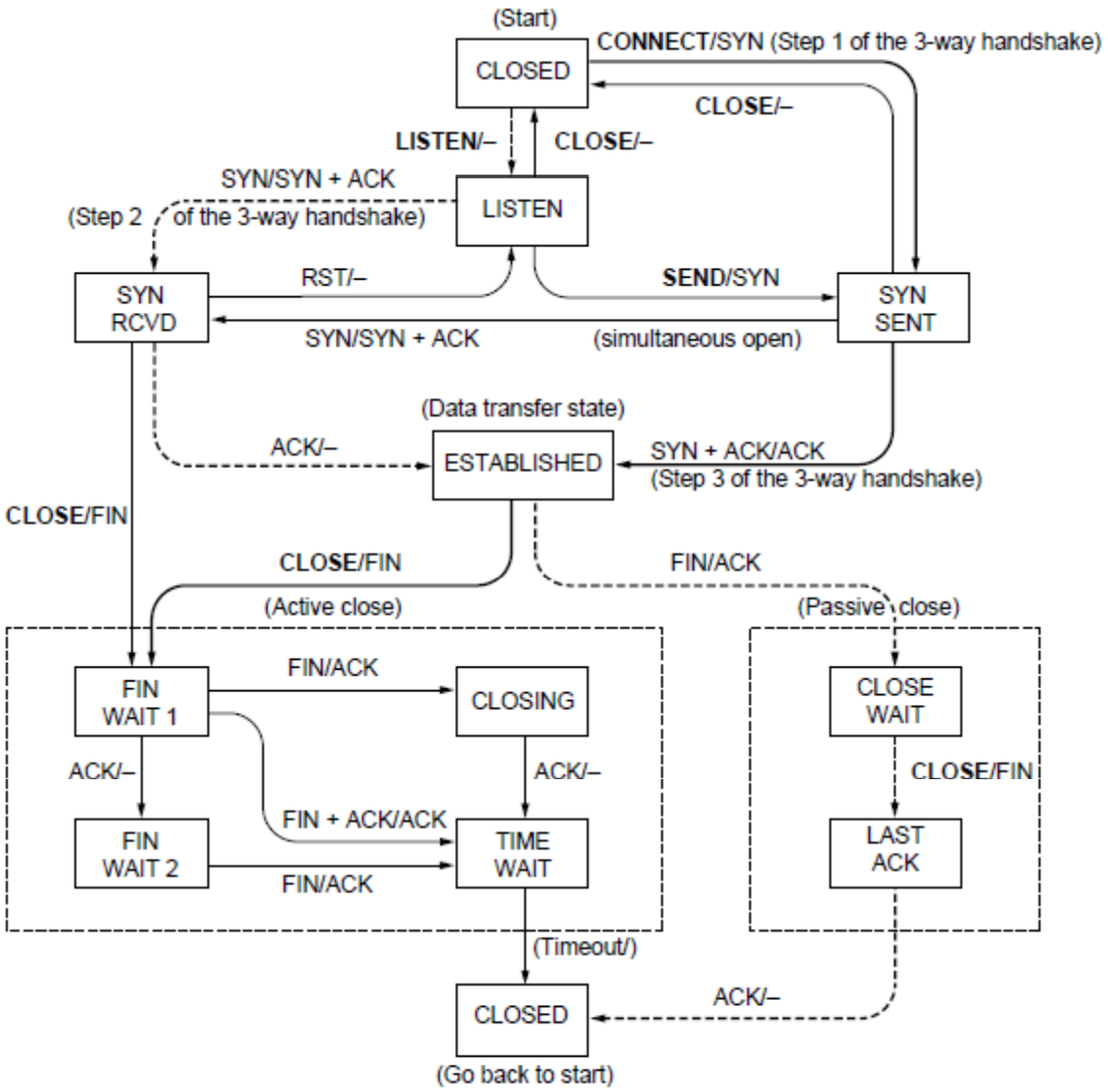
TCP Connection State Modeling

Solid line is the normal path for a client.

Dashed line is the normal path for a server.

Light lines are unusual events.

Transitions are labeled by the cause and action, separated by a slash.



```
struct tcp_pcb {
  struct tcp_pcb *next;
  enum tcp_state state;      /* TCP state */
  void (* accept)(void *arg, struct tcp_pcb *newpcb);
  void *accept_arg;
  struct ip_addr local_ip;
  u16_t local_port;
  struct ip_addr dest_ip;
  u16_t dest_port;
  u32_t rcv_nxt, rcv_wnd;    /* receiver variables */
  u16_t tmr;
  u32_t mss;                 /* maximum segment size */
  u8_t flags;
  u16_t rttest;              /* rtt estimation */
  u32_t rtseq;               /* sequence no for rtt estimation */
  s32_t sa, sv;              /* rtt average and variance */
  u32_t rto;                 /* retransmission time-out */
  u32_t lastack;             /* last ACK received */
  u8_t dupacks;              /* number of duplicate ACKs */
  u32_t cwnd, u32_t ssthresh;  /* congestion control variables */
  u32_t snd_ack, snd_nxt,    /* sender variables */
    snd_wnd, snd_wl1, snd_wl2, snd_lbb;
  void (* recv)(void *arg, struct tcp_pcb *pcb, struct pbuf *p);
  void *recv_arg;
  struct tcp_seg *unsent, *unacked, /* queues */
        *ooseq;
};
```

Implementation of TCP State Machine in C:

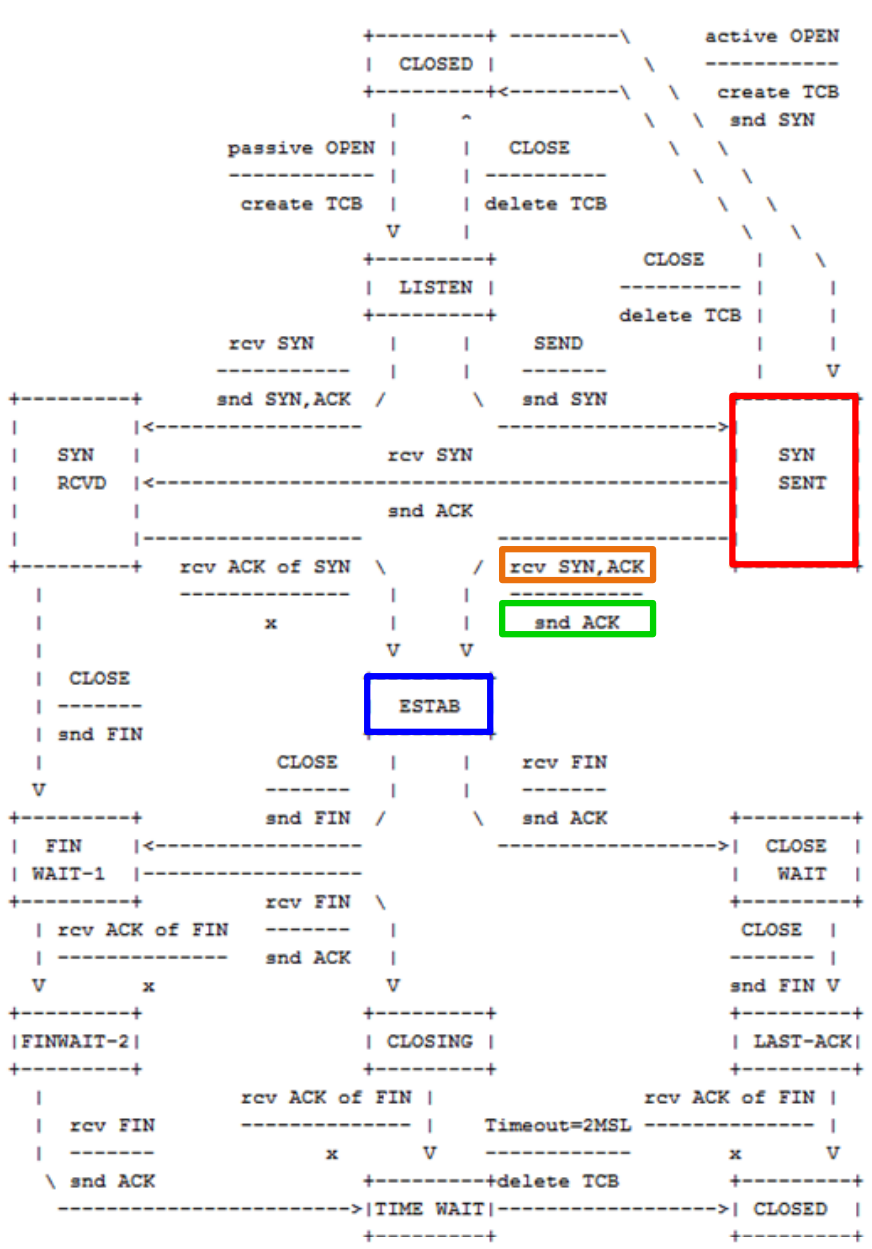
Declaration of TCP Protocol Control Block (tcp_pcb)

```
/**
 * @file
 * Transmission Control Protocol, incoming traffic
 *
 * The input processing functions of the TCP layer.
 *
 * These functions are generally called in the order (ip_input() ->)
 * tcp_input() -> * tcp_process() -> tcp_receive() (-> application).
 *
 */

/*
 * Copyright (c) 2001-2004 Swedish Institute of Computer Science.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 *    this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 * 3. The name of the author may not be used to endorse or promote products
 *    derived from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
 * SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
 * OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
 * OF SUCH DAMAGE.
 *
 * This file is part of the lwIP TCP/IP stack.
 *
 * Author: Adam Dunkels <adam@sics.se>
 *
 */
```

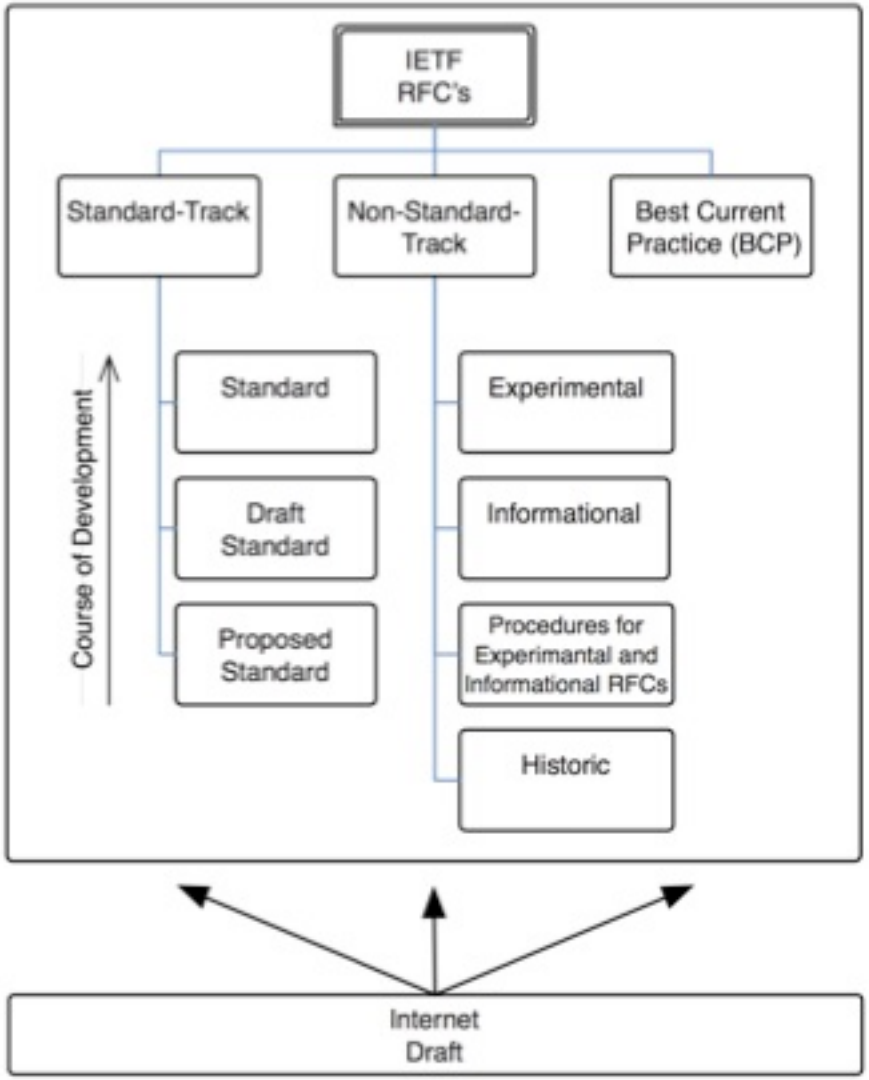
Header of OpenSource Implementation of TCP Protocol (C)

```c
/* Do different things depending on the TCP state. */
switch (pcb->state) {
case SYN_SENT:
  LWIP_DEBUGF(TCP_INPUT_DEBUG, ("SYN-SENT: ackno %"U32_F" pcb->snd_nxt %"U32_F" unacked %"U
   pcb->snd_nxt, ntohl(pcb->unacked->tcphdr->seqno)));
  /* received SYN ACK with expected sequence number? */
  if ((flags & TCP_ACK) && (flags & TCP_SYN)
      && ackno == ntohl(pcb->unacked->tcphdr->seqno) + 1) {
    pcb->snd_buf++;
    pcb->rcv_nxt = seqno + 1;
    pcb->rcv_ann_right_edge = pcb->rcv_nxt;
    pcb->lastack = ackno;
    pcb->snd_wnd = tcphdr->wnd;
    pcb->snd_wl1 = seqno - 1; /* initialise to seqno - 1 to force window update */
    pcb->state = ESTABLISHED;
    [...]
    tcp_ack_now(pcb);
  }
  [...]
  }
  break;
case SYN_RCVD:
  if (flags & TCP_ACK) {
    /* expected ACK number? */
    if (TCP_SEQ_BETWEEN(ackno, pcb->lastack+1, pcb->snd_nxt)) {
      u16_t old_cwnd;
      pcb->state = ESTABLISHED;
      [...]
```



Implementation of TCP State Machine in C:

Example of Coding State "SYN SENT" and one State Transition



Standard:
- Significant experiences in implementations
- Successful operation

Draft Standard:
- At least two independently developed implementations
- Implementations have no interoperability problems

Proposed Standard:
- Specification is stable
- No known problems

"Course of Development" for Standards in IETF