Processing Incoming Segments

RFC 793 Section 3.9 SEGMENT ARRIVES Page 65 Onwards

1981

Abstract

A re-write of RFC 793 Section 3.9 into a structure that is easier to follow and more amenable to interpretation, with numbered sections. Use this document in its PDF form maximum ease-of-understanding. Includes content from later RFCs including RFC 1122, RFC 2873, RFC 5961 and RFC 7323 as well as RFC errata.

Contents

1 CLOSED **4**
- 1.1 If the ACK bit is off . . . 4
- 1.2 If the ACK bit is on . . . 4

2 LISTEN **5**
- 2.1 Check for an RST . . . 5
- 2.2 Check for an ACK . . . 5
- 2.3 Check for a SYN . . . 5
  - 2.3.1 Check the security. . . . 5
  - 2.3.2 RFC 7323 Appendix D . . . 5
  - 2.3.3 Continue . . . 5
  - 2.3.4 RFC 7323 Appendix D . . . 6
  - 2.3.5 Continue . . . 6
- 2.4 Other text or control . . . 6

3 SYN-SENT **7**
- 3.1 Check the ACK bit . . . 7
  - 3.1.1 If the ACK bit is set . . . 7
- 3.2 Check the RST bit . . . 7
  - 3.2.1 If the RST bit is set . . . 7
- 3.3 Check the security and precedence . . . 7
- 3.4 Check the SYN bit . . . 8
  - 3.4.1 If the SYN bit is on . . . 8
- 3.5 If neither of the SYN or RST bits is set . . . 9

4 SYN-RECEIVED ESTABLISHED FIN-WAIT-1 FIN-WAIT-2 CLOSE-WAIT CLOSING LAST-ACK TIME-WAIT **10**
- 4.1 Check sequence number . . . 10
  - 4.1.1 RFC7323 Appendix D: Rescale the received window field . . . 10
  - 4.1.2 R1: RFC 7323, Section 5.3 Point R1 . . . 10
  - 4.1.3 R2: Segment is acceptable because it occupies a portion of valid receive sequence space . . . 10
  - 4.1.4 R3: RFC 7323, Section 5.3 Point R3 . . . 12
- 4.2 Check the RST bit . . . 12
  - 4.2.1 If the RST bit is not set . . . 12
  - 4.2.2 If the RST bit is set . . . 12
- 4.3 Check Security & Precedence . . . 13
- 4.4 Check the SYN bit . . . 13
  - 4.4.1 SYN-RECEIVED and connection was initiated with a passive OPEN (RFC 1122 Section 4.2.2.20 (e)) 13
  - 4.4.2 ESTABLISHED FIN-WAIT-1 FIN-WAIT-2 CLOSE-WAIT CLOSING LAST-ACK TIME-WAIT . . . 13
- 4.5 Check the ACK field . . . 13
  - 4.5.1 If the ACK bit is off . . . 13
  - 4.5.2 If the ACK bit is on . . . 13
- 4.6 Check the URG bit . . . 15
  - 4.6.1 ESTABLISHED FIN-WAIT-1 FIN-WAIT-2 . . . 15

4.6.2 CLOSE-WAIT CLOSING LAST-ACK TIME-WAIT 15
4.7 Process the Segment Text 16
4.7.1 ESTABLISHED FIN-WAIT-1 FIN-WAIT-2 16
4.7.2 CLOSE-WAIT CLOSING LAST-ACK TIME-WAIT 16
4.8 Check the FIN bit 16
4.8.1 CLOSED LISTEN SYN-SENT 16
4.8.2 If the FIN bit is set 16

List of Tables

4.1 Tests for Acceptability of an Incoming Segment 10

Chapter 1

CLOSED

Modern practice is to simply ignore (blackhole) instead of replying with RST. The following is historic practice.

1.1 If the ACK bit is off

If the state is CLOSED (i.e., TCB does not exist) then all data in the incoming segment is discarded. An incoming segment containing a RST is discarded. An incoming segment not containing a RST causes a RST to be sent in response. The acknowledgment and sequence field values are selected to make the reset sequence acceptable to the TCP that sent the offending segment. See below for specifics of this.

Sequence number zero is used <SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>.

Return.

1.2 If the ACK bit is on

If the state is CLOSED (i.e., TCB does not exist) then all data in the incoming segment is discarded. An incoming segment containing a RST is discarded. An incoming segment not containing a RST causes a RST to be sent in response. The acknowledgment and sequence field values are selected to make the reset sequence acceptable to the TCP that sent the offending segment. See below for specifics of this.

<SEQ=SEG.ACK><CTL=RST>

Return.

Chapter 2

LISTEN

2.1 Check for an RST

An incoming RST should be ignored. Return.

2.2 Check for an ACK

Modern practice is to simply ignore (blackhole) instead of replying with RST. The following is historic practice.

Any acknowledgment is bad if it arrives on a connection still in the LISTEN state. An acceptable reset segment should be formed for any arriving ACK-bearing segment. The RST should be formatted as <SEQ=SEG.ACK><CTL=RST>. Return.

2.3 Check for a SYN

2.3.1 Check the security.

Ignored as per RFC 2873.

2.3.2 RFC 7323 Appendix D

Check for a Window Scale option (WSopt); if one is found, save SEG.WSopt in Snd.Wind.Shift and set Snd.WS.OK flag on. Otherwise, set both Snd.Wind.Shift and Rcv.Wind.Shift to zero and clear Snd.WS.OK flag.

Check for a TSopt option; if one is found, save SEG.TSval in the variable TS.Recent and turn on the Snd.TS.OK bit.

2.3.3 Continue

Set RCV.NXT to SEG.SEQ+1, IRS is set to SEG.SEQ and any other control or text should be queued for processing later. ISS should be selected and a SYN segment sent of the form <SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>.

2.3.4 RFC 7323 Appendix D

If the Snd.WS.OK bit is on, include a WSopt <WSopt=Rcv.Wind.Shift> in this segment. If the Snd.TS.OK bit is on, include a TSopt <TSval=Snd.TSclock,TSecr=TS.Recent> in this segment. Last.ACK.sent is set to RCV.NXT.

2.3.5 Continue

SND.NXT is set to ISS+1 and SND.UNA to ISS. The connection state should be changed to SYN-RECEIVED. Note that any other incoming control or data (combined with SYN) will be processed in the SYN-RECEIVED state, but processing of SYN and ACK should not be repeated. If the listen was not fully specified (i.e., the foreign socket was not fully specified), then the unspecified fields should be filled in now.

2.4 Other text or control

Any other control or text-bearing segment (not containing SYN) must have an ACK and thus would be discarded by the ACK processing. An incoming RST segment could not be valid, since it could not have been sent in response to anything sent by this incarnation of the connection. So you are unlikely to get here, but if you do, drop the segment, and return.

Chapter 3

SYN-SENT

3.1 Check the ACK bit

3.1.1 If the ACK bit is set

If SEG.ACK <= ISS, or SEG.ACK > SND.NXT, send a reset (unless the RST bit is set, if so drop the segment and return) <SEQ=SEG.ACK><CTL=RST> and discard the segment.

Return.

If SND.UNA < SEG.ACK <= SND.NXT[1] then the ACK is acceptable.

Commentary: With the Errata 3300 correction, the two inequalities, with SND.UNA substituted for ISS (rewritten for intelligibility as SND.UNA >= SEG.ACK || SEG.ACK > SND.NXT and SND.UNA < SEG.ACK && SEG.ACK <= SND.NXT) cover the entirely of all possible values and so the statement in 3.4 is now valid.

Commentary: It is probably not wise to respond to an ACK without either a SYN or RST bit set.

3.2 Check the RST bit

This behaviour has been augmented by RFC 5961 Section 3.2 Page 8: "In the SYN-SENT state (a RST received in response to an initial SYN), the RST is acceptable if the ACK field acknowledges the SYN. In all other cases the receiver **MUST** silently discard the segment.

3.2.1 If the RST bit is set

If the ACK was acceptable then signal the user "error: connection reset", drop the segment, enter CLOSED state, delete the TCB, and return. Otherwise (no ACK) drop the segment and return.

3.3 Check the security and precedence

Ignored as per RFC 2873.

[1] Errata 3300: "In SYN-SENT, SND.UNA == ISS, so the first line is contradictory to the last. … In practice today, much code seems to be simplifying further and checking that SEG.ACK == SND.NXT, for stacks that are not sending data on the SYN"

3.4 Check the SYN bit

This step should be reached only if the ACK is acceptable[2], or there is no ACK, and it the segment did not contain a RST.

3.4.1 If the SYN bit is on

Security requirements ignored as per RFC 2873.

Then, RCV.NXT is set to SEG.SEQ+1, IRS is set to SEG.SEQ.

SND.UNA should be advanced to equal SEG.ACK (if there is an ACK), and any segments on the retransmission queue which are thereby acknowledged should be removed[3].

3.4.1.1 RFC 7323 Appendix D

Check for a Window Scale option (WSopt); if it is found, save SEG.WSopt in Snd.Wind.Shift; otherwise, set both Snd.Wind.Shift and Rcv.Wind.Shift to zero.

Check for a TSopt option; if one is found, save SEG.TSval in variable TS.Recent and turn on the Snd.TS.OK bit in the connection control block. If the ACK bit is set, use Snd.TSclock - SEG.TSecr as the initial RTT estimate.

3.4.1.2 If SND.UNA > ISS (our SYN has been ACKed)

Change the connection state to ESTABLISHED*, form an ACK segment <SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK> and send it.

RFC 7323 Appendix D

If the Snd.TS.OK bit is on, include a TSopt option <TSval=Snd.TSclock,TSecr=TS.Recent> in this <ACK> segment. Last.ACK.sent is set to RCV.NXT.

Continue

Data or controls which were queued for transmission may be included.

* RFC 1122 Section 4.2.2.20 (c)

When the connection enters the ESTABLISHED state, the following variables must be set:

- SND.WND <- SEG.WND
- SND.WL1 <- SEG.SEQ
- SND.WL2 <- SEG.ACK

If there are other controls or text in the segment then continue processing at 4.6.1, otherwise return.

[2] was 'ok'

[3] Since processing the SYN,ACK allows taking a measurement of round-trip-time (RTT) using a timestamp, and acknowledging segments uses timestamps to do so, this step should be deferred until after processing the TSopt below

3.4.1.3 Otherwise

Modern practice is to ignore (blackhole) this, which is known as a 'Simultaneous Open'. RFC 7323 Appendix D also changes the logic here, but this is not recorded herein.

Enter SYN-RECEIVED, form a SYN,ACK segment <SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK> and send it.

If there are other controls or text in the segment, queue them for processing after the ESTABLISHED state has been reached, return.

3.5 If neither of the SYN or RST bits is set

Drop the segment and return.

Chapter 4

SYN-RECEIVED ESTABLISHED FIN-WAIT-1 FIN-WAIT-2 CLOSE-WAIT CLOSING LAST-ACK TIME-WAIT

4.1 Check sequence number

4.1.1 RFC7323 Appendix D: Rescale the received window field

`TrueWindow = SEG.WND << Snd.Wind.Shift` and and use "TrueWindow" in place of SEG.WND in the following steps.

4.1.2 R1: RFC 7323, Section 5.3 Point R1

If the segment contains a Timestamps option and if bit `Snd.TS.OK` is on:-

If there is a Timestamps option in the arriving segment[1], `SEG.TSval < TS.Recent`, `TS.Recent` is valid (see later discussion), and if the `RST` bit is not set, then treat the arriving segment as not acceptable.

If the arriving segment is unacceptable then send an acknowledgment as for 4.1.3 below.

Note: it is necessary to send an <ACK> segment in order to retain TCP's mechanisms for detecting and recovering from half-open connections. For an example, see Figure 10 of RFC 793.

4.1.3 R2: Segment is acceptable because it occupies a portion of valid receive sequence space

Originally from RFC 793 Page 69:-

Segments are processed in sequence. Initial tests on arrival are used to discard old duplicates, but further processing is done in `SEG.SEQ` order. If a segment's contents straddle the boundary between old and new, only the new parts should be processed.

There are four cases for the acceptability test for an incoming segment:-

Table 4.1: Tests for Acceptability of an Incoming Segment

Segment Length (`SEG.LEN`)	Receive Window (`RCV.WND`)	Test
0	0	`SEG.SEQ = RCV.NXT`

[1] if timestamps have been negotiated, there will be for all segments bar `RST`, as we will validate that when parsing TCP options

Segment Length (SEG.LEN)	Receive Window (RCV.WND)	Test
0	>0	RCV.NXT <= SEG.SEQ < RCV.NXT+RCV.WND
>0	0	not acceptable
>0	>0	RCV.NXT <= SEG.SEQ < RCV.NXT+RCV.WND or RCV.NXT <= SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

If the RCV.WND is zero, no segments will be acceptable, but special allowance should be made to accept valid ACKs, URGs and RSTs.

If an incoming segment is not acceptable, an acknowledgment should be sent in reply of <SEQ=SND.NXT><ACK=RCV.NXT><CTL=A (unless the RST bit is set, if so drop the segment and return).

Last.ACK.sent is set to SEG.ACK of the acknowledgment. If the Snd.TS.OK bit is on, include the Timestamps option <TSval=Snd.TSclock,TSecr=TS.Recent> in this <ACK> segment (RFC7323 Appendix D).

After sending the acknowledgment, drop the unacceptable segment and return.

In the following it is assumed that the segment is the idealized segment that begins at RCV.NXT and does not exceed the window. One could tailor actual segments to fit this assumption by trimming off any portions that lie outside the window (including SYN and FIN), and only processing further if the segment then begins at RCV.NXT. Segments with higher begining sequence numbers SHOULD[2] be held for later processing.

Errata 3305

Not sure how to correct it systemmatically, so I just present the problem.

If in SYN-RECEIVED state, and received a SYN-ACK packet with no data as:

- SEG.SEQ = IRS
- SEG.ACK = ISS + 1

However in SYN-RECEIVED state, RCV.NXT = IRS + 1, which means the SYN-ACK packet will fail on the second test above. The SYN-ACK packet will be dropped and an ACK packet is sent in reply. As a result, we lost the SYN part, but it is fine because we've already received SYN packet once. However, we also lost the ACK part which is supposed to be the ACK of our SYN. Thus we will never reach the ESTABLISHED state.

4.1.3.1 RFC 1122 Section 4.2.2.20 Additional Changes

In general, the processing of received segments MUST be implemented to aggregate ACK segments whenever possible. For example, if the TCP is processing a series of queued segments, it MUST process them all before sending any ACK segments.

1. CLOSE Call, CLOSE-WAIT state, p. 61: enter LAST-ACK state, not CLOSING.
2. USER TIMEOUT p. 77: It would be better to notify the application of the timeout rather than letting TCP force the connection closed. However, see also RFC 1122 Section 4.2.3.5.

[2] Changed by RFC 1122 Section 4.2.2.20

4.1.4 R3: RFC 7323, Section 5.3 Point R3

R3) If an arriving segment satisfies `SEG.TSval >= TS.Recent` and `SEG.SEQ <= Last.ACK.sent` (see RFC 7323 Section 4.3), then record its timestamp in `TS.Recent`.

4.2 Check the RST bit

4.2.1 If the RST bit is not set

Continue processing at 4.3.

4.2.2 If the RST bit is set

This logic has been updated from RFC 5961 Section 3.

4.2.2.1 SYN-SENT

In the SYN-SENT state (a RST received in response to an initial SYN), the RST is acceptable if the ACK field acknowledges the SYN. In all other cases the receiver MUST silently discard the segment.

4.2.2.2 All other states

In all states except SYN-SENT, all reset (RST) segments are validated by checking their SEQ-fields [sequence numbers]. A reset is valid if its sequence number exactly matches the next expected sequence number. If the RST arrives and its sequence number field does NOT match the next expected sequence number but is within the window, then the receiver should generate an ACK. In all other cases, where the SEQ-field does not match and is outside the window, the receiver MUST silently discard the segment.

SYN-RECEIVED

If this connection was initiated with a passive OPEN (i.e., came from the LISTEN state), then return this connection to LISTEN state and return. The user need not be informed. If this connection was initiated with an active OPEN (i.e., came from SYN-SENT state) then the connection was refused, signal the user “connection refused”. In either case, all segments on the retransmission queue should be removed. And in the active OPEN case, enter the CLOSED state and delete the TCB, and return.

ESTABLISHED FIN-WAIT-1 FIN-WAIT-2 CLOSE-WAIT

Any outstanding RECEIVEs and SEND should receive “reset” responses. All segment queues should be flushed. Users should also receive an unsolicited general “connection reset” signal. Enter the CLOSED state, delete the TCB, and return.

CLOSING LAST-ACK TIME-WAIT

Enter the CLOSED state, delete the TCB, and return.

4.3 Check Security & Precedence

Ignored as per RFC 2873.

Continue Processing at 4.4.

4.4 Check the SYN bit

4.4.1 SYN-RECEIVED and connection was initiated with a passive OPEN (RFC 1122 Section 4.2.2.20 (e))

In SYN-RECEIVED state and if the connection was initiated with a passive OPEN, then return this connection to the LISTEN state and return.

It is not clear from RFC 1122 whether checks for the SYN being in the window are appropriate.

4.4.2 ESTABLISHED FIN-WAIT-1 FIN-WAIT-2 CLOSE-WAIT CLOSING LAST-ACK TIME-WAIT

This has been updated by RFC 5961 Section 4.2.

If the SYN bit is set, irrespective of the sequence number, TCP MUST send an ACK (also referred to as challenge ACK) to the remote peer <SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>. After sending the acknowledgment, TCP MUST drop the unacceptable segment and stop processing further.

Originally:-

- If the SYN is in the window: It is an error, send a reset, any outstanding RECEIVEs and SEND should receive "reset" responses, all segment queues should be flushed, the user should also receive an unsolicited general "connection reset" signal, enter the CLOSED state, delete the TCB, and return.
- if the SYN is not in the window: This step would not be reached and an ack would have been sent in the first step (sequence number check).

4.5 Check the ACK field

4.5.1 If the ACK bit is off

Drop the segment and return

4.5.2 If the ACK bit is on

4.5.2.1 SYN-RECEIVED

- If [the segment acknowledgment is acceptable] (SND.UNA < SEG.ACK <= SND.NXT)[3] then enter ESTABLISHED state and continue processing (?fallthrough?).
- If the segment acknowledgment is not acceptable, form a reset segment <SEQ=SEG.ACK><CTL=RST> and send it.

[3] The check of SNA.UNA was changed by Errata 3301 from SND.UNA <= SEG.ACK to SND.UNA < SEG.ACK.

When the connection enters ESTABLISHED state[4], the variables listed in RFC 1122 Section 4.2.2.20 (c) must be set, viz:-

- SND.WND <- SEG.WND
- SND.WL1 <- SEG.SEQ
- SND.WL2 <- SEG.ACK

4.5.2.2 ESTABLISHED

The ACK value is considered acceptable only if it is in the range of ((SND.UNA - MAX.SND.WND) <= SEG.ACK <= SND.NXT). All incoming segments whose ACK value doesn't satisfy the above condition MUST be discarded and an 'ACK" sent back.[5]

- If SND.UNA < SEG.ACK <= SND.NXT (the segment acknowledgment is acceptable) then, set SND.UNA <- SEG.ACK. Also compute a new estimate of round-trip time. If Snd.TS.OK bit is on, use Snd.TSclock - SEG.TSecr; otherwise, use the elapsed time since the first segment in the retransmission queue was sent. Any segments on the retransmission queue which are thereby entirely acknowledged are removed. Users should receive positive acknowledgments for buffers which have been SENT and fully acknowledged (i.e., SEND buffer should be returned with "ok" response).
- If the ACK is a duplicate (SEG.ACK <= SND.UNA[6]), it can be ignored[7].
- If the ACK acks something not yet sent (SEG.ACK > SND.NXT) then send an ACK, drop the segment, and return.

If (SND.UNA <= SEG.ACK <= SND.NXT)[8], the send window should be updated. If (SND.WL1 < SEG.SEQ or (SND.WL1 = SEG.SEQ and SND.WL2 <= SEG.ACK)), set SND.WND <- SEG.WND, set SND.WL1 <- SEG.SEQ, and set SND.WL2 <- SEG.ACK.

Note that SND.WND is an offset from SND.UNA, that SND.WL1 records the sequence number of the last segment used to update SND.WND, and that SND.WL2 records the acknowledgment number of the last segment used to update SND.WND. The check here prevents using old segments to update the window.

Errata 4785

If the ACK is a duplicate (SEG.ACK <= SND.UNA), it can be ignored except when equality is met (SEG.ACK = SND.UNA). This can occur when:-

- there are segments in flight;
- the receiver shrinks RCV.BUF such that all of these segments would drop;
- the reciever sends an ACK with a WND of zero (a window update ACK)

If the window is zero then the senderr starts a persist timer for sending zero-window probes [9].

See also the definition of "DUPLICATE ACKNOWLEDGMENT" in RFC 5681 Section 2 Page 4.

4.5.2.3 FIN-WAIT-1

In addition to the processing for the ESTABLISHED state, if our FIN is now acknowledged then enter FIN-WAIT-2 and continue processing in that state.

[4] RFC 1122 Section 4.2.2.220 (f)
[5] RFC 5961 Section 5.
[6] RFC1122 Section 4.2.2.20 (g) (the = was omitted)
[7] But see Errata 4785
[8] RFC1122 Section 4.2.2.20 (g)
[9] RFC 1122 Section 4.2.2.17, page 92

4.5.2.4 FIN-WAIT-2

In addition to the processing for the ESTABLISHED state, if the retransmission queue is empty, the user's CLOSE can be acknowledged ("ok") but do not delete the TCB.

4.5.2.5 CLOSE-WAIT

Do the same processing as for the ESTABLISHED state.

4.5.2.6 CLOSING

In addition to the processing for the ESTABLISHED state, if the ACK acknowledges our FIN then enter the TIME-WAIT state, otherwise ignore the segment.

4.5.2.7 LAST-ACK

The only thing that can arrive in this state is an acknowledgment of our FIN. If our FIN is now acknowledged, delete the TCB, enter the CLOSED state, and return.

4.5.2.8 TIME-WAIT

The only thing that can arrive in this state is a retransmission of the remote FIN. Acknowledge it, and restart the 2 MSL timeout.

4.6 Check the URG bit[10]

Ignored

4.6.1 ESTABLISHED FIN-WAIT-1 FIN-WAIT-2

If the URG bit is set, `RCV.UP <- max(RCV.UP,SEG.UP)`, and signal the user that the remote side has urgent data if the urgent pointer (RCV.UP) is in advance of the data consumed. If the user has already been signaled (or is still in the "urgent mode") for this continuous sequence of urgent data, do not signal the user again.

4.6.2 CLOSE-WAIT CLOSING LAST-ACK TIME-WAIT

This should not occur, since a FIN has been received from the remote side. Ignore the URG.

[10] The URG bit is valid on a received SYNACK in the SYN-SENT stage; it is handled as if ESTABLISHED. In theory it is probably not invalid on a SYN either

4.7 Process the Segment Text

4.7.1 ESTABLISHED FIN-WAIT-1 FIN-WAIT-2

Once in the ESTABLISHED state, it is possible to deliver segment text to user RECEIVE buffers. Text from segments can be moved into buffers until either the buffer is full or the segment is empty. If the segment empties and carries an PUSH flag, then the user is informed, when the buffer is returned, that a PUSH has been received.

When the TCP takes responsibility for delivering the data to the user it must also acknowledge the receipt of the data.

Once the TCP takes responsibility for the data it advances RCV.NXT over the data accepted, and adjusts RCV.WND as apporopriate to the current buffer availability. The total of RCV.NXT and RCV.WND should not be reduced.

Please note the window management suggestions in section 3.7.

Send an acknowledgment of the form: <SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>. If the Snd.TS.OK bit is on, include the Timestamps option <TSval=Snd.TSclock,TSecr=TS.Recent> in this <ACK> segment. Set Last.ACK.sent to SEG.ACK of the acknowledgment, and send it[11]. This acknowledgment should be piggybacked on a segment being transmitted if possible without incurring undue delay.

4.7.2 CLOSE-WAIT CLOSING LAST-ACK TIME-WAIT

This should not occur, since a FIN has been received from the remote side. Ignore the segment text.

4.8 Check the FIN bit

4.8.1 CLOSED LISTEN SYN-SENT

Do not process the FIN if the state is CLOSED, LISTEN or SYN-SENT since the SEG.SEQ cannot be validated; drop the segment and return.

4.8.2 If the FIN bit is set

Signal the user “connection closing” and return any pending RECEIVEs with same message, advance RCV.NXT over the FIN, and send an acknowledgment for the FIN.

Note that FIN implies PUSH for any segment text not yet delivered to the user.

4.8.2.1 SYN-RECEIVED ESTABLISHED

Enter the CLOSE-WAIT state.

4.8.2.2 FIN-WAIT-1

If our FIN has been ACKed (perhaps in this segment), then enter TIME-WAIT, start the time-wait timer, turn off the other timers; otherwise enter the CLOSING state.

[11] RFC 7323 Appendix D

4.8.2.3 FIN-WAIT-2

Enter the TIME-WAIT state. Start the time-wait timer, turn off the other timers.

4.8.2.4 CLOSE-WAIT

Remain in the CLOSE-WAIT state.

4.8.2.5 CLOSING

Remain in the CLOSING state.

4.8.2.6 LAST-ACK

Remain in the LAST-ACK state.

4.8.2.7 TIME-WAIT

Remain in the TIME-WAIT state. Restart the 2 MSL time-wait timeout.