CHAPTER 12

TCP

Exercises

1.

Common fields in UDP and TCP header:

source port number

destination port number

checksum

Fields present in TCP header that are absent from UDP header:

sequence number	(for flow and error control)
acknowledgment number	(for flow and error control)
header length	(TCP has variable header length)
control bits	(flow and error control and connection)
urgent pointer	(needed for flow and error control)
options and padding	(for better performance)

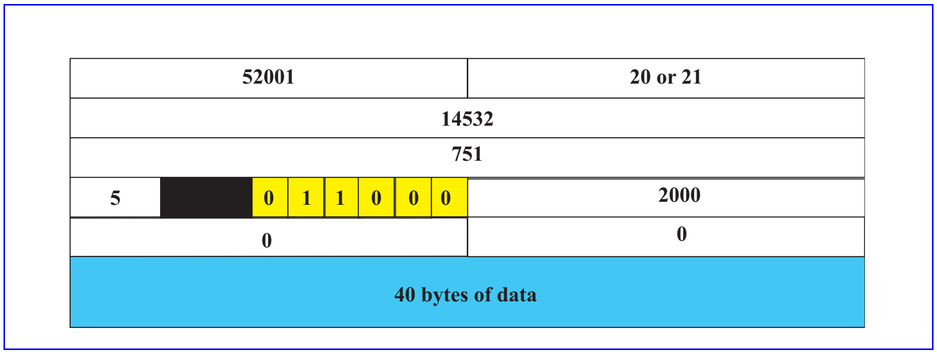
Fields present in UDP header that are absent from TCP header:

total length	(actually not needed)

2. It looks as if both the destination IP address and the destination port number are corrupted. TCP calculates the checksum and drops the segment.
3. The port is not listed in the transmission control block.
4. UDP is preferred because each user datagram can be used for each chunk of data. However, a better solution is the new transport protocol, SCTP, discussed in Chapter 13.
5. The maximum size of the TCP header is 60 bytes. The minimum size of the TCP header is 20 bytes.
6. 0111 in decimal is 7. The total length of the header is 7×4 or 28. The base header is 20 bytes. The segment has 8 bytes of options.

7. See Figure 12.1.

Figure 12.1 *Exercise 7*



8.

   a. None of the control bits are set. The segment is part of a data transmission without piggybacked acknowledgment.
   b. The FIN bit is set. This is a FIN segment request to terminate the connection.
   c. The ACK and the FIN bits are set. This is a FIN + ACK in response to a received FIN segment.
   d. The RST bit is set. This a request for resetting.
   e. The SYN bit is set. This is a SYN segment. The client wishes to establish a connection with the server.
   f. The ACK and the SYN bits are set. This is a SYN + ACK segment is sent in response to a SYN segment.

9.

   a. The source port number is 0532 in hex and 1330 in decimal.
   b. The destination port number is 0017 in hex and 23 in decimal.
   c. The sequence number is 00000001 in hex and 1 in decimal.
   d. The acknowledgment number is 00000000 in hex and 0 in decimal.
   e. The header length is 5 in hex and 5 in decimal. 5×4 is 20 bytes of header.
   f. The control field is 002 in hex. This indicates a SYN segment used for connection establishment.
   g. The window size field is 07FF in hex and 2047 in decimal. The window is 2047 bytes.

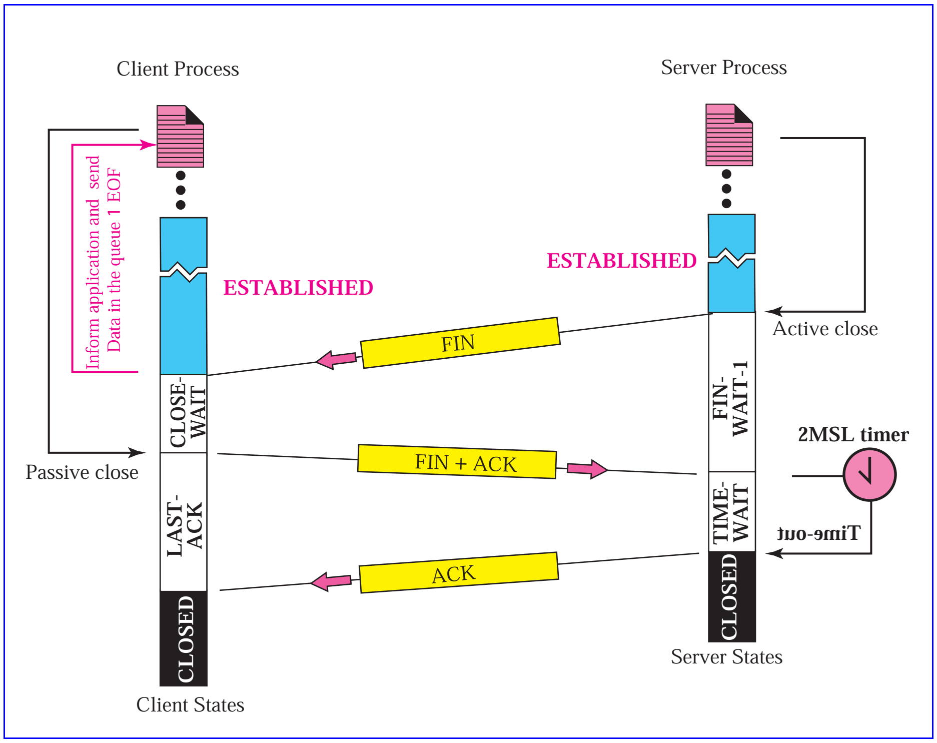
10.

000000	(a data segment with not acknowledgment)
110000	(a data segment with urgent data and acknowledgment)
010000	(a data segment with acknowledgment)
000010	(a SYN segment)
011000	(a data segment with push data and acknowledgment)

11. Every second the counter is incremented by 64,000 × 2 = 128,000. The sequence number field is 32 bits long and can hold only 2^{32}–1. So it takes (2^{32}–1)/(128,000) seconds or 33554 seconds.

12. See Figure 12.2.

Figure 12.2 *Exercise 12*

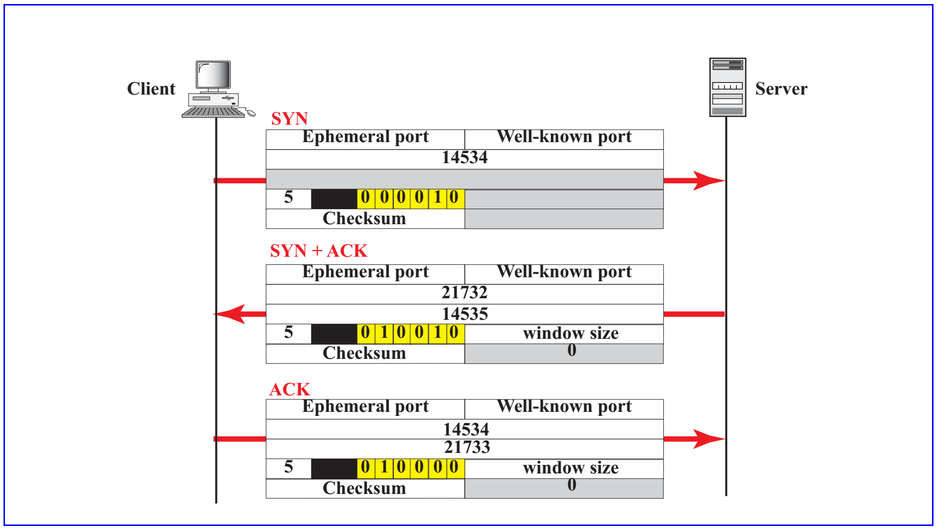


13.
a. 2171
b. 2172
c. 3172

14. 3000 – 2000 = 1000 bytes

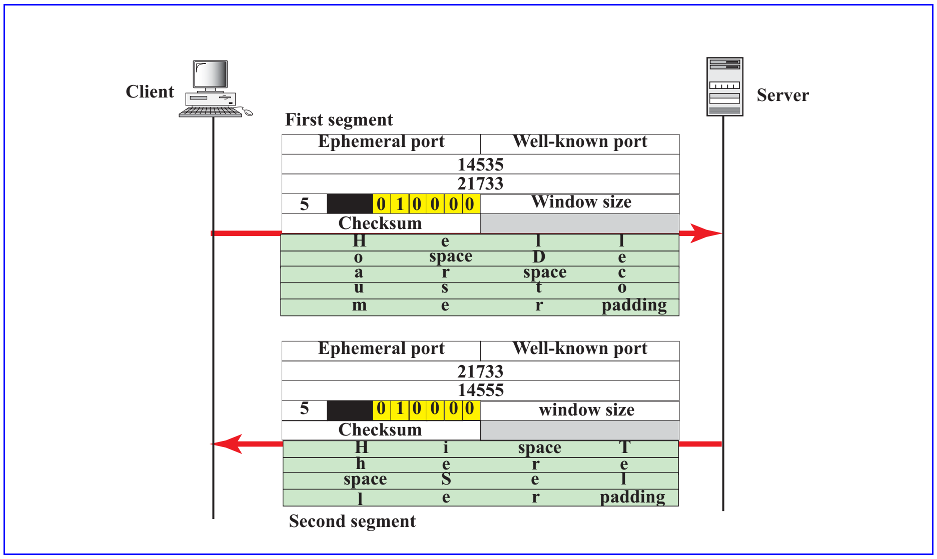
15. See Figure 12.3.

Figure 12.3 *Exercise 15*



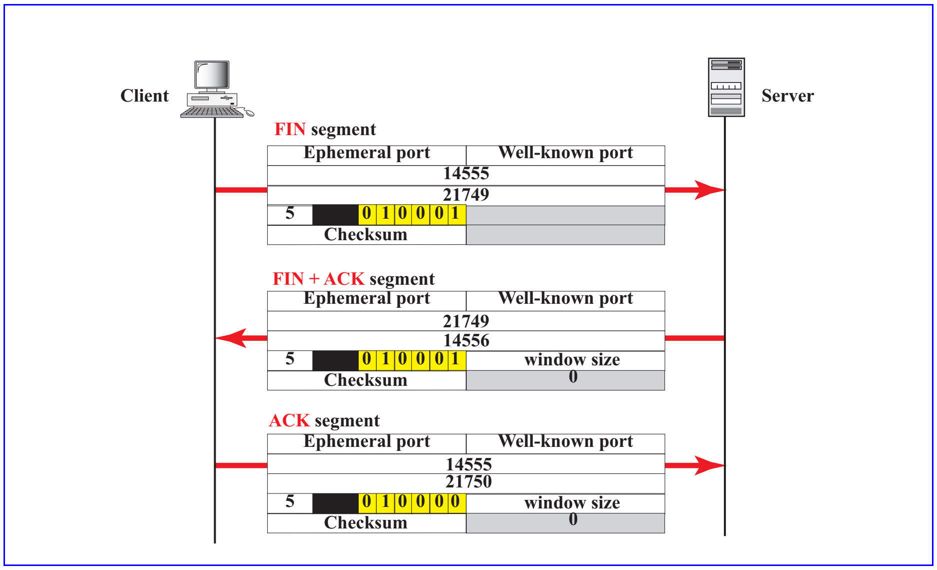
16. See Figure 12.4.

Figure 12.4 *Exercise 16*



17. See Figure 12.5.

Figure 12.5 *Exercise 17*



18.

At TCP level:

16 bytes of data / 36 bytes of header and data = 0.444

At IP level:

16 bytes of data / [(20 bytes of IP header) + (36 bytes TCP segment)] = 0.286

At data link level:

16 bytes of data / [(20 bytes of IP header) + (36 bytes TCP segment) + (19 bytes Ethernet header and trailer)] = 0.213, assuming no preamble

19. The largest number in the sequence number field is $2^{32} - 1$. If we start at 7000, it takes $[(2^{32} - 1) - 7000] / 1{,}000{,}000$ = **4295** seconds.

20. See Figure 12.6.

21. See Figure 12.7.

22.

a. The server can receive a FIN segment while it is in the ESTABLISHED state. When the FIN segment is received, the server sends an ACK segment to the client and moves to the CLOSE-WAIT state.

b. When the “close” message is received from the application, the client TCP sends a FIN segment; the client goes to the FIN-WAIT-1 state and waits for an ACK.

Figure 12.6 *Exercise 20*

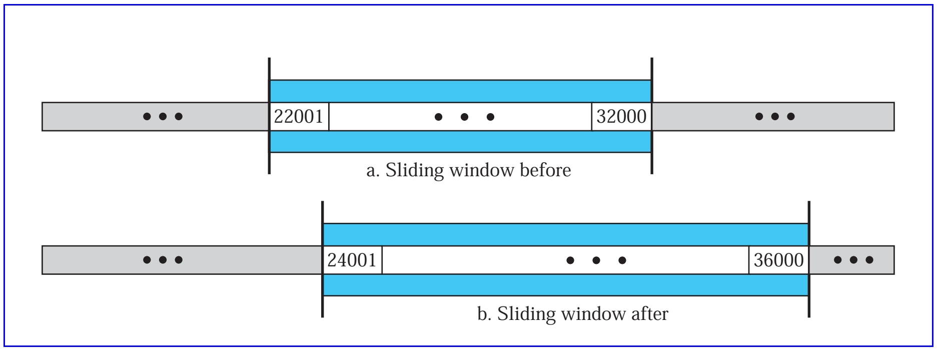
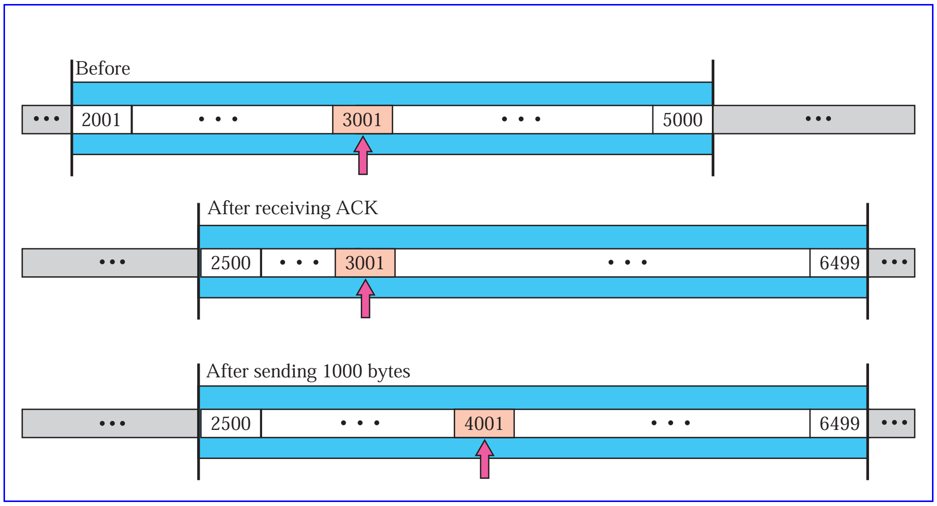


Figure 12.7 *Exercise 21*



23.

a. When the "close" message is received from the application, the client TCP sends a FIN segment; the client goes to the FIN-WAIT-1 state and waits for an ACK.

b. When the client receives an ACK segment from the server, it moves to the FIN-WAIT-2 state, and waits for the server to send a FIN segment.

24.

a. No ACK needed at 0:0:0:0:000, according to Rule 2

b. No ACK needed at 0:0:0:0:027, according to Rule 2

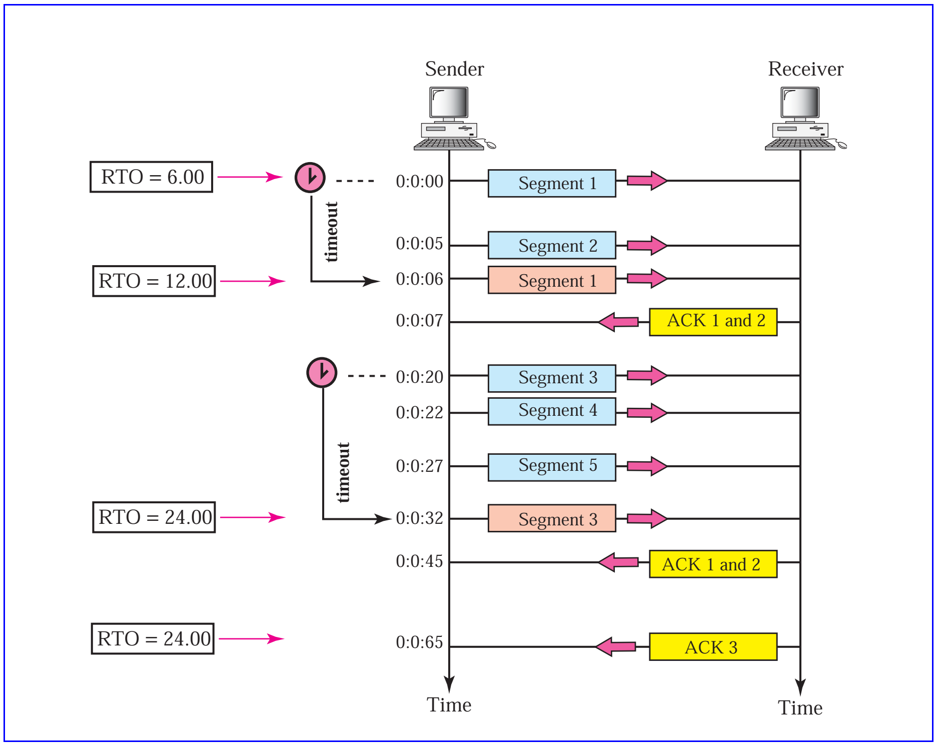
c. ACK: 4 can be sent at 0:0:0:0:500

d. ACK: 5 can be sent at 0:0:0:1:200

e. No ACK needed at 0:0:0:1:208, according to Rule 2

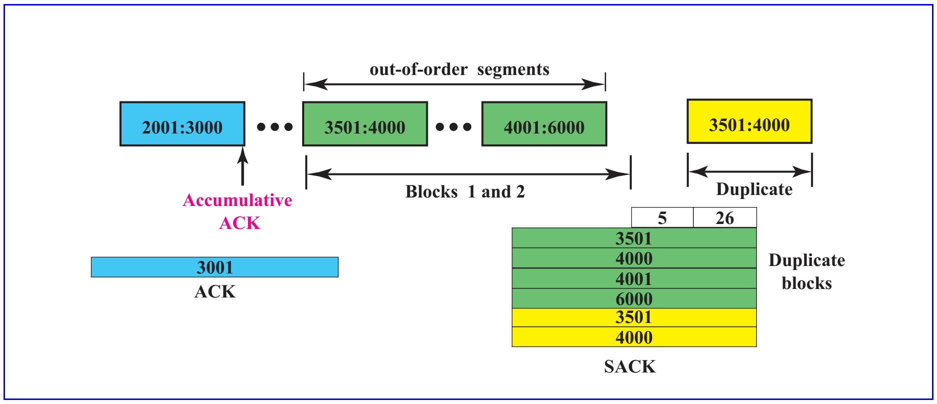
25. See Figure 12.8.

Figure 12.8 *Exercise 25*



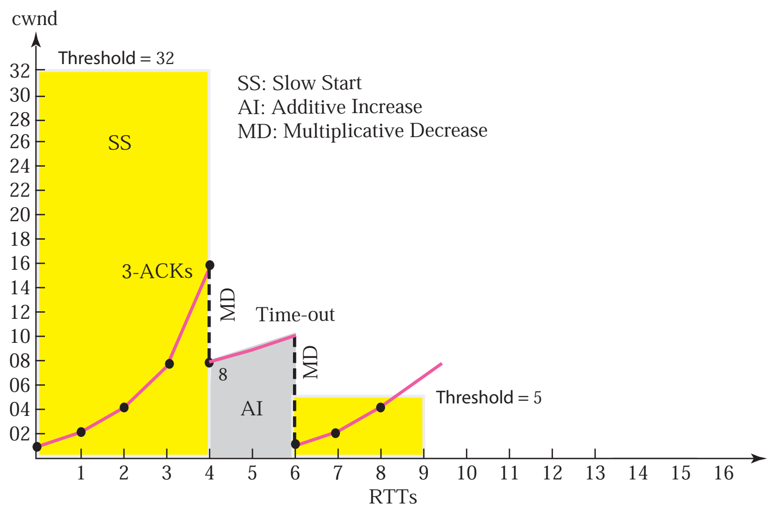
26. See Figure 12.9.

Figure 12.9 *Exercise 26*



27. Although some implementations reacts to the congestion detection (3ACKs or timeout) only during the congestion avoidance, we assume that the system reacts to the 3-ACK events even during the slows tart as shown in See Figure 12.10.

Figure 12.10 *Exercise 27*


cwnd
Threshold = 32
SS
3-ACKs
SS: Slow Start
AI: Additive Increase
MD: Multiplicative Decrease
MD
Time-out
MD
AI
Threshold = 5
RTTs