Chapter 3 Review Questions

1.
2.
3. Source port number 61 and destination port number 37.

4. TCP's congestion control can throttle an application's sending rate at times of congestion. Designers of applications like IP telephony and IP videoconference prefer to avoid TCP's congestion control. Moreover, reliability of telephony and videoconference is not so important as compared to other critical applications. If the receiver does not receive a data or does not understand the data, he or she can always request the sender to retransmit.

5.

6. Application layer.

7. Yes, both segments will be directed to the same socket. For each received segment, at the socket interface, the operating system will provide the process with the IP addresses to determine the origins of the individual segments.

8. For each persistent connection, the Web server creates a separate "connection socket". Each connection socket is identified with a four-tuple: (source IP address, source port number, destination IP address, destination port number). When host C receives and IP datagram, it examines these four fields in the datagram/segment to determine to which socket it should pass the payload of the TCP segment. Thus, the requests from A and B pass through different sockets. The identifier for both of these sockets has 80 for the destination port; however, the identifiers for these sockets have different values for source IP addresses. Unlike UDP, when the transport layer passes a TCP segment's payload to the application process, it does not specify the source IP address, as this is implicitly specified by the socket identifier.

9.

10.

11.

12.

13.

14. a) false b) false c) true d) false e) true f) false g) false

15. a) 27 bytes b) 65 (assume that the packet before the first one mentioned in the problem is not lost).

16.

17. R/2

18. This is because TCP's congestion control consists of linear (additive) increase in cwnd of 1 MSS per RTT and then a halving (multiplicative decrease) of cwnd on a triple duplicate-ACK event.

Chapter 3 Problems

Problem 3

```
      00100011
+     01001110
   ---------------------------
      01110001

      01110001
+     01010100
   ---------------------------
      11000101
```

One's complement = 00111010

To detect errors, the receiver adds the four words (the three original words and the checksum). If the sum contains a zero, the receiver knows there has been an error. All one-bit errors will be detected, but two-bit errors can be undetected (e.g., if the last digit

of the first word is converted to a 0 and the last digit of the second word is converted to a 1).

Problem 15

It takes 9.6 microseconds (or 0.0096 milliseconds) to send a packet, as $1200*8/10^9=9.6$ microseconds. In order for the sender to be busy 95 percent of the time, we must have

$$util = 0.97 = (0.0096n)/30.0096$$

or n approximately 3032 packets.

Problem 23

In order to avoid the scenario of Figure 3.27, we want to avoid having the leading edge of the receiver's window (i.e., the one with the “highest” sequence number) wrap around in the sequence number space and overlap with the trailing edge (the one with the "lowest" sequence number in the sender's window). That is, the sequence number space must be large enough to fit the entire receiver window and the entire sender window without this overlap condition. So - we need to determine how large a range of sequence numbers can be covered at any given time by the receiver and sender windows.

Suppose that the lowest-sequence number that the receiver is waiting for is packet m. In this case, it's window is [m,m+w-1] and it has received (and ACKed) packet m-1 and the w-1 packets before that, where w is the size of the window. If none of those w ACKs have been yet received by the sender, then ACK messages with values of [m-w,m-1] may
still be propagating back. If no ACKs with these ACK numbers have been received by the sender, then the sender's window would be [m-w,m-1].

Thus, the lower edge of the sender's window is m-w, and the leading edge of the receivers window is m+w-1. In order for the leading edge of the receiver's window to not overlap with the trailing edge of the sender's window, the sequence number space must thus be big enough to accommodate 2w sequence numbers. That is, the sequence number space must be at least twice as large as the window size, $k \geq 2w$.

Problem 26

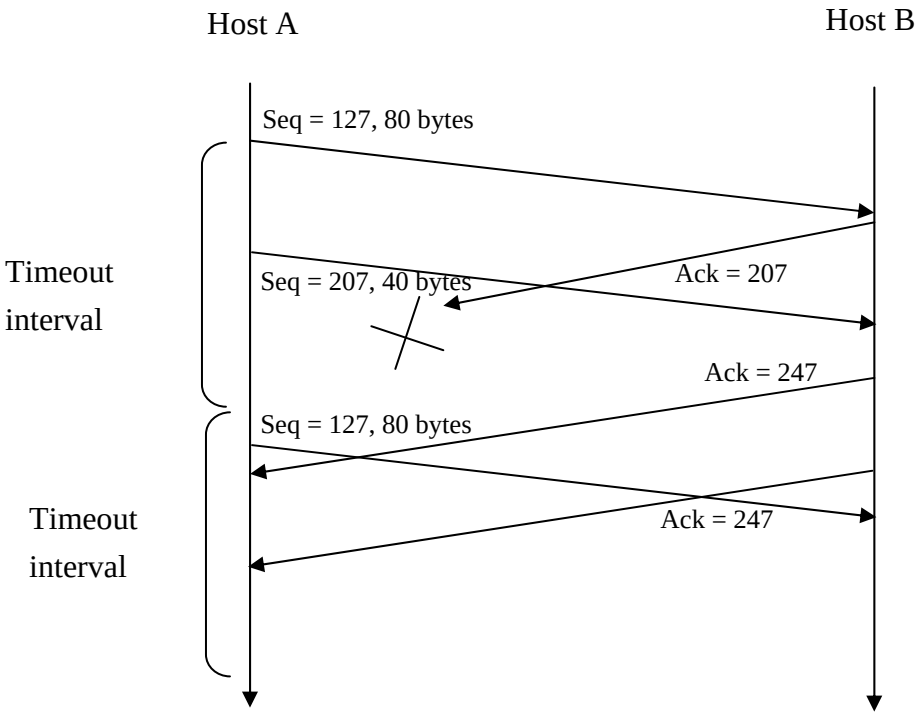
There are $2^{32} = 4{,}294{,}967{,}296$ possible sequence numbers.

a) The sequence number does not increment by one with each segment. Rather, it increments by the number of bytes of data sent. So the size of the MSS is irrelevant -- the maximum size file that can be sent from A to B is simply the number of bytes representable by $2^{32} \approx 4.19\,\text{Gbytes}$.

b) The number of segments is $\left\lceil \frac{2^{32}}{512} \right\rceil = 8{,}388{,}608$. 64 bytes of header get added to each segment giving a total of 536,870,912 bytes of header. The total number of bytes transmitted is $2^{32} + 536{,}870{,}912 = 4.832 \times 10^{9}$ bytes.
Thus it would take 249 seconds to transmit the file over a 155 Mbps link.

Problem 27

a) In the second segment from Host A to B, the sequence number is 207, source port number is 302 and destination port number is 80.
b) If the first segment arrives before the second, in the acknowledgement of the first arriving segment, the acknowledgement number is 207, the source port number is 80 and the destination port number is 302.
c) If the second segment arrives before the first segment, in the acknowledgement of the first arriving segment, the acknowledgement number is 127, indicating that it is still waiting for bytes 127 and onwards.
d)



Problem 30

Either (1) the receiver immediately discards out-of-order segments (which, as we discussed earlier, can simplify receiver design), or (2) the receiver keeps the out-of-order bytes and waits for the missing bytes to fill in the gaps. Clearly, the latter choice is more efficient in terms of network bandwidth, and is the approach taken in practice.

Problem 32

a)

Denote $EstimatedRTT^{(n)}$ for the estimate after the *n*th sample.

$$EstimatedRTT^{(4)} = xSampleRTT_1 + (1-x)[xSampleRTT_2 + (1-x)[xSampleRTT_3 + (1-x)SampleRTT_4]]$$

$$= xSampleRTT_1 + (1-x)xSampleRTT_2$$

$$+ (1-x)^2 xSampleRTT_3 + (1-x)^3 SampleRTT_4$$

b)

$$EstimatedRTT^{(n)} = x\sum_{j=1}^{n-1}(1-x)^{j-1}SampleRTT_j$$

$$+ (1-x)^{n-1} SampleRTT_n$$

c)

$$EstimatedRTT^{(\infty)} = \frac{x}{1-x}\sum_{j=1}^{\infty}(1-x)^j SampleRTT_j$$

$$= \frac{1}{9}\sum_{j=1}^{\infty}.9^j SampleRTT_j$$

The weight given to past samples decays exponentially.

Problem 33

It is desirable to set the timeout equal to the `EstimatedRTT` plus some margin. The margin should be large when there is a lot of fluctuation in the `SampleRTT` values; it should be small when there is little fluctuation. All of these considerations are taken into account in TCP's method for determining the retransmission timeout interval:

```
TimeoutInterval = EstimatedRTT + 4 • DevRTT
```

Problem 40

a) TCP slowstart is operating in the intervals [1,6] and [23,26]
b) TCP congestion advoidance is operating in the intervals [6,16] and [17,22]
c) After the 16th transmission round, packet loss is recognized by a triple duplicate ACK. If there was a timeout, the congestion window size would have dropped to 1.
d) After the 22nd transmission round, segment loss is detected due to timeout, and hence the congestion window size is set to 1.
e) The threshold is initially 32, since it is at this window size that slow start stops and congestion avoidance begins.
f) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 16, the congestion windows size is 42. Hence the threshold is 21 during the 18th transmission round.
g) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 22, the congestion windows size is 26. Hence the threshold is 13 during the 24th transmission round.
h) During the 1st transmission round, packet 1 is sent; packet 2-3 are sent in the 2nd transmission round; packets 4-7 are sent in the 3rd transmission round; packets 8-15 are sent in the 4th transmission round; packets 16-31 are sent in the 5th transmission round; packets 32-63 are sent in the 6th transmission round; packets 64 – 96 are sent in the 7th transmission round. Thus packet 70 is sent in the 7th transmission round.
i) The threshold will be set to half the current value of the congestion window (8) when the loss occurred and congestion window will be set to the new threshold value + 3 MSS . Thus the new values of the threshold and window will be 4 and 7 respectively. ~~Threshold is 21, and congestion window size is 1.~~

j) round 17, 1 packet; round 18, 2 packets; round 19, 4 packets; round 20, 8 packets; round 21, 16 packets; round 22, 21 packets. So, the total number is 52.

Problem 41

TCP provides a flow-control service to its applications to eliminate the possibility of the sender overflowing the receiver's buffer. Flow control is thus a speed-matching service —matching the rate at which the sender is sending against the rate at which the receiving application is reading. On the other hand, a TCP sender can also be throttled due to congestion within the IP network; this form of sender control is referred to as congestion control.