1. In both parts of Fig. 6-6, there is a comment that the value of SERVERPORT must be the same in both client and server. Why is this so important?

   Solution: If the client sends a packet to SERVER PORT and the server is not listening to that port, the packet will not be delivered to the upper application of server.

2. Imagine that a two-way handshake rather than a three-way handshake were used to set up connections. In other words, the third message was not required. Are deadlocks now possible? Give an example or show that none exist.

   Solution: Deadlocks are possible. For example, a packet arrives at A out of the blue, and A acknowledges it. The acknowledgement gets lost, but A is now open while B knows nothing at all about what has happened. Now the same thing happens to B, and both are open, but expecting different sequence numbers. Timeouts have to be introduced to avoid the deadlocks.

3. Why does UDP exist? Would it not have been enough to just let user processes send raw IP packets?

   Solution: No. IP packets contain IP addresses, which specify a destination machine. Once such a packet arrived, how would the network handler know which process to give it to? UDP packets contain a destination port. This information is essential so they can be delivered to the correct process.

4. A client sends a 128-byte request to a server located 100 km away over a 1-gigabit optical fiber. What is the efficiency of the line during the remote procedure call?

   Solution: Sending 1000 bits over a 1 Gbps line takes 1μsec. The speed of light in fiber optics is 200 km/msec, so it takes 0.5 msec for the request to arrive and another 0.5 msec for the reply to get back. In all, 1000 bits have been transmitted in 1 msec. This is equivalent to 1 megabit/sec, or 1/10 of 1% efficiency.

5. Datagram fragmentation and reassembly are handled by IP and are invisible to TCP. Does this mean that TCP does not have to worry about data arriving in the wrong order?

   Solution: Even though each datagram arrives intact, it is possible that datagrams arrive in the wrong order, so TCP has to be prepared to reassemble the parts of a message properly.

6. The maximum payload of a TCP segment is 65,495 bytes. Why was such a strange number

chosen?

Solution: The entire TCP segment must fit in the 65,515-byte payload field of an IP packet. Since the TCP header is a minimum of 20 bytes, only 65,495 bytes are left for TCP data.

7. If the TCP round-trip time, RTT, is currently 30 msec and the following acknowledgements come in after 26, 32, and 24 msec, respectively, what is the new RTT estimate using the Jacobson algorithm? Use α=0.9.

   Solution:
   29.6=0.9*30+（1-0.9）*26
   29.84=0.9*29.6+（1-0.9）*32
   29.256=0.9*29.84+（1-0.9）*24
   The successive estimates are 29.6, 29.84, 29.256.

8. To get around the problem of sequence numbers wrapping around while old packets still exist, one could use 64-bit sequence numbers. However, theoretically, an optical fiber can run at 75 Tbps. What maximum packet lifetime is required to make sure that future 75-Tbps networks do not have wrap around problems even with 64-bit sequence numbers? Assume that each byte has its own sequence number, as TCP does.

   Solution: The size of the sequence space is2^64 bytes, which is about 2*10^19 bytes. A 75-Tbpstransmitter uses up sequence space at a rate of 9.375×10^12 sequence numbers per second. It takes 2 million seconds to wrap around. Since there are 86,400 seconds in a day, it will take over 3 weeks to wrap around, even at 75 Tbps. A maximum packet lifetime of less than 3 weeks will prevent the problem.

9. Consider that only a single TCP (Reno) connection uses one 10Mbps link which does not buffer any data. Suppose that this link is the only congested link between the sending and receiving hosts. Assume that the TCP sender has a huge file to send to the receiver, and the receiver's receive buffer is much larger than the congestion window. We also make the following assumptions: each TCP segment size is 1,500 bytes; the two way propagation delay of this connection is 150 msec; and this TCP connection is always in congestion avoidance phase, that is, ignore slow start.
   – a. What is the maximum window size (in segments) that this TCP connection can achieve?
   – b. What is the average window size (in segments) and average throughput (in bps) of this TCP connection?
   – c. How long would it take for this TCP connection to reach its maximum window again after recovering from a packet loss?

Solution:

a. Assume the window size is W,

$$\mathrm{MSS} * \mathrm{W}/\mathrm{RTT} = 10\mathrm{M}$$

$$1500 * 8 * W / 0.15 = 10 * 10 \wedge 6$$

$$W = 125$$

b. As congestion window size varies from W/2 to W, then the average window size is 0.75W = 93.75 segments. Average throughput is 93.75*1500*8/0.15=7.5Mbps

c. (125/2) *0.15= 20 seconds, as the number of RTTs (that this TCP connections needs in order to increase its window size from W/2 to W) is given by W/2. Recall the window size increases by one in each RTT..