姚熙源 3190300677 计算机网络作业 6

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?

Answer : The port number in client will be sent to the port of the server. If the port of client and server are not same, the server will not forward the data packets to the corresponding client port.

2. Imagine that a two-way handshake rather than a three-wayhandshake 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.

Answer : Deadlocks are possible. Assume that A sent a data packet to B with Seq=x, when B receive the packet, B reply a response to A with Seq=y, Ack=x, but the packet lost in halfway. At this time, A thinks that the connection has been established, so A waits for B to send the next data packet, while B is also waiting for the response from A, then two of them are in deadlock.

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

Answer : UDP exists to allow you to send IP-encapsulated data without the connection overhead of TCP. The advantage of UDP over simply sending raw IP packets is that UDP gives you the capability to multiplex/demultiplex on different source/destination ports.

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?

Answer : sending 128 * 8 = 1024 bits over a 1Gbps line takes 1024/1Gb = 1μs.

The speed of light in fiber optics is 200 km/milliseconds, so it takes 100/200 = 0.5milliseconds for the request to arrive and another 0.5milliseconds for the response to get back. 1024bits have been transmitted in 1millisecond. The efficiency of the line is 1μs / 1millisec = 0.1%

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?

Answer : No. Although IP handles the complete arrival of all the fragments of a large packet, it does not take care of the assembling of the packet in the correct order. So, TCP still needs to reassemble the datagrams to keep them in order.

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

Answer : IP header occupy with 20 bytes and TCP header is a minimum of 20 bytes, so 65535-20-20 = 65495 bytes are left for maximum payload of a TCP segment.

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.

Answer :

a. 0.9*30 + 0.1*26 = 29.6msec

b. 0.9*29.6 + 0.1*32 = 29.84msec

c. 0.9*29.84 + 0.1*24 = 29.256msec

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.

Answer : The maximum time to get sequence number wraps around T = 2^{64}B / (75*10^{12}/8 Bps) = 1967652secs = 22.77 Days.

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 twoway 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?

Answer : MSS * W / RTT = 10Mbps (bandwidth = (W/RTT)*MSS)

W = 10 * 10^6 * 0.15s / (1500 * 8) = 125(Maximum window size)

b. What is the average window size (in segments) and average throughput (in bps) of this TCP connection?

Answer : Average window size is (W + W/2) / 2 = 0.75W

Average Throughput is 0.75W * MSS / RTT = 0.75W * 1500 * 8 / 0.15 = 7.5Mbps

c. How long would it take for this TCP connection to reach its maximum window again after recovering from a packet loss?

Answer : It would take T = W/2 * RTT = (125/2) * 0.15s = 9.375sec to reach its maximum window again.