Homework 6

Name: Wang Haoyuan
Number: 3220105114

Problem 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:

Because whether UDP or TCP is used, when the client sends a message to the server, the server needs to know which port to listen to. If the port number is not the same in both client and server, the server will not be able to receive the message.

Problem 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.

Answer:

The deadlock is possible to happen. When the SYN packet from the client is retransferred because of the time-out, and the retransferred packet arrives at the server after the connection is closed.

In this scene, server gets the SYN packet and sends back a SYN/ACK packet. After that, the server will wait for the data sent from the client. However, the client doesn't actually connect to the server, so it causes the server is in the deadlock state.

Also, the same situation can happen on the client side. Once the ACKno and SEQno is ex

Problem 3

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

Answer:

The purpose of UDP is to provide a connectionless service. By UDP, the sender and receiver can transfer data with less time delay.

If just send raw IP packets, the information of source port and destination port will be lost, which will cause the receiver cannot identify the sender. Also, UDP is able to provide the service of checksum, if

just send raw IP packets, the receiver cannot check the data integrity.

Problem 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:

The time spent on transfering the data is:

$$\frac{128 \times 8}{10^9} = 1.024 \times 10^{-6} s \approx 10^{-6} s$$

The time spent on the distance is:

$$\frac{100 \times 10^3}{200000 \times 10^3} = 500 \mu s$$

So the efficiency of the line during the remote procedure call is:

$$\frac{10^{-6} s}{500 \mu s \times 2 + 10^{-6} s} \approx 0.1\%$$

Problem 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. Even though the datagram fragmentation and reassembly are handled by IP, The order of datagrams is still not guaranteed. So TCP must cope with the order of datagrams.

Problem 6

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

Answer:

This number is actually calculated by the TCP/IP protocol.

In IP segment, the maximum payload is $2^{16} = 65535$ bytes.

For IP header and TCP header, the minimum size is 20 bytes respectively.

So, the maximum payload of a TCP segment is:

$$65535 - 20 - 20 = 65495bytes$$

So this is why 65495 is the maximum payload of the TCP segment.

Problem 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:

calculate every EstimatedRTT respectively:

after 26 msec:

$$t_1 = 30 \times 0.9 + 26 \times 0.1 = 29.6msec$$

after 32 msec:

$$t_2 = 29.6 \times 0.9 + 32 \times 0.1 = 29.84msec$$

after 24 msec:

$$t_3 = 29.84 \times 0.9 + 24 \times 0.1 = 29.256msec$$

So the RTT estimate using the Jacobson algorithm is 29.6 msec, 29.84 msec, and 29.256 msec.

Problem 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:

if every byte has its own sequence number, then the whole 2^{64}seq number can represent $2^{64} * 8 = 2^{67}$ bits. so the maximum packet lifetime is:

$$\frac{2^{67}}{75 \times 10^{12}} =\approx 1967652.70s$$

So the maximum lifetime is about 22.7 days, or about 3 weeks.

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 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?
- 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?

Answer:

- a:

For each TCP segment, its size is:

$$1500 \times 8 = 12,000 bits$$

If the link is 10Mbps, then the maximum throughput is:

$$10^7 \times 150 \times 10^{-3} = 1.5 \times 10^6 bits$$

So the maximum window size is:

$$1.5 \times 10^6 / 12,000 = 125 segments$$

- b:

if the window reaches the maximum size and leads to a packet loss, the window size will decrease by half. So the average window size and throughput will both be $\frac{3}{4}$ of the maximum.

the average window size is:

$$\frac{3}{4} \times 125 = 93.75 segments$$

the average throughput is:

$$\frac{3}{4} \times 10^7 = 7.5 \times 10^6 bps$$

- c:

if the window size is halved, the cwnd is 62 segments.

so the time spent on recovering from a packet loss is:

$$(125 - 62) \times 150 \times 10^{-3} = 9.45s$$