Conceptual Homework 2

Michael Brodskiy
Professor: E. Bernal Mor

October 23, 2023

1. Consider a TCP connection between Host A and Host B. Suppose that the TCP segments traveling from Host A to Host B have source port number x and destination port number y. What are the source and destination port numbers for the segments traveling from Host B to Host A?

   Since traveling from A to B results in source port x and destination port y, traveling in the opposite direction, from B to A, would reverse this. As such, the source port would become y and the destination would be x.

2. Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 6789. Will both of these segments be directed to the same socket at host C? If so, how will the process at Host C know that these two segments originated from two different hosts?

   Yes, both of these segments would be directed to the same socket. The UDP header would contain the IP addresses of Hosts A and B, which would allow Host C to identify the origin.

3. Suppose that a Web server runs in Host C on port 80. Suppose this Web server uses persistent connections, and is currently receiving requests from two different Hosts, A and B. Are all of the requests being sent through the same socket at Host C? If they are being passed through different sockets, do both of the sockets have port 80? Discuss and explain.

   Given that the connection is persistent, each connection is directed to a separate socket depending on the contents of the four-tuple header (source address, source port, destination address, and destination port). In this identifier, the destination fields, would, however, be the same. Thus, the two would have different sockets but the same port (in this case, the typical port for HTTP, 80).

4. In our rdt protocols, why did we need to introduce sequence numbers? And why did we need to introduce timers? Briefly explain your answers.

   In the rdt protocols, we first introduced sequence numbers to essentially differentiate between the last two packets. That is, the sender attaches a sequence number to the packet so that the receiver can "remember" the previous packet and check whether the received packet was duplicate.

Timers were introduced to be able to use timeout. If a timer expires, it is assumed that the ACK/NACK for a sent packet was lost, and the packet was retransmitted. The receiver would then check the sequence number of the retransmitted packet to check whether it is a duplicate. In creating these timers, we can speed up transmission and forego any connection freezes, as the sender would not just wait until something is received.

5. Consider a scenario where packet losses and bit errors are possible. Consider a reliable data transfer protocol that uses only negative acknowledgments (NACKS). When the sender sends a packet, if a NACK is not received at the sender after a given time, the sender considers that the packet was delivered correctly. The loss of a packet is detected in the receiver when a packet with higher sequence number than the expected is received, and then the receiver sends a NACK. Discuss the following scenarios:

   (a) Suppose the sender sends data only infrequently. Would this NACK-only protocol be preferable to a protocol that uses ACKs? Why?

   In the case that data is sent infrequently, the NACK-only protocol would not be preferable. This is due to several negative consequences: first and foremost, in the event that a NACK is sent to signify a lost packet, both the lost packet and next one would have to be retransmitted, which results in inefficiencies. Furthermore, with such a protocol, lost packets can be detected only upon receiving the next packet, thus it may take a long time for this loss to be realized.

   (b) Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NACK-only protocol be preferable to a protocol that uses ACKs? Why?

   In the case of frequently-sent data with low loss, such a protocol may be preferred, as less NACKS would be sent than ACKs to confirm reception, which can significantly reduce the time necessary to transmit something. Furthermore, a loss may be realized more quickly than a system with ACKs since data is sent more frequently.

6. Consider two hosts, one located on the West Coast of United States and the other located on the East Coast. One host is sending packets to the other host. The round-trip-time (RTT) between these two end systems is approximately 30 ms. Suppose the bitrate in the access networks of the sender is 1 Gbps and the packet size is $L = 1500$ bytes including header fields and data. How big would the window size (number of packets) of the sender host must be for the channel utilization to be greater than 98 percent?

   We know the utility may be defined as:

$$U = N\left(\frac{L/R}{RTT + L/R}\right)$$

We can first calculate the transmission rate:

$$\frac{L}{R} = \frac{8 \cdot 1500}{10^9} = 1.2 \cdot 10^{-5}[\text{s}] = 12[\mu\text{s}]$$

We can then set up the equation as follows:

$$\begin{gathered} .98 = N\left(\frac{12}{30 \cdot 10^3 + 12}\right) \\ .98 = N\left(4 \cdot 10^{-4}\right) \\ N = 2451 \end{gathered}$$

Thus, for 98% utilization, the window should be approximately 2451 packets in length.

7. UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit words: 01010011, 01100110, 01110100. (Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums). Compute the value that would be included in an 8-bit checksum field following the same approach used in UDP and TCP. Show all work.

   We can first begin by summing all of the bits:

$$\begin{array}{rr} & 01010011 \\ + & 01100110 \\ \hline & 10111001 \end{array}$$

$$\begin{array}{rr} & 10111001 \\ + & 01110100 \\ \hline & 100101101 \end{array}$$

   We then send the overflow to the last bit:

$$\begin{array}{rr} & 00101101 \\ + & 00000001 \\ \hline & 00101110 \end{array}$$

   From this, we can compute the checksum by inverting the ones and zeros of the above sum (finding the one's compliment):

$$00101110 \rightarrow \boxed{11010001}$$

   Thus, we can see that the boxed value is a checksum, and may be used to verify whether the aggregate of sent data is accurate. This would be done by summing the checksum and bit sum from above. If this sum contains all ones, there is no error. If it contains any zeros, then there has been some error.

8. Suppose that three measured SampleRTT values are 106 ms, 120 ms and 90ms. Assume that the value of EstimatedRTT was 100 ms just before the first of these samples were obtained and α=0.125. Assume the value of DevRTT was 5 ms just before the first of these samples was obtained and β=0.25. Compute the TCP TimeoutInterval after each of these samples is obtained. Show all your work.

   First, we can write out the formulas for the values we need to calculate:

$$EstimatedRTT = (1-\alpha)\cdot EstimatedRTT + \alpha \cdot SampleRTT$$
$$DevRTT = (1-\beta)\cdot DevRTT + \beta|SampleRTT - EstimatedRTT|$$
$$TimeoutInterval = EstimatedRTT + 4\cdot DevRTT$$

   We begin by finding all EstimatedRTT values:

$$E_{106} = .875\cdot 100 + .125\cdot 106$$
$$\boxed{E_{106} = 100.75[\text{ms}]}$$
$$E_{120} = .875\cdot 100.75 + .125\cdot 120$$
$$\boxed{E_{120} = 103.16[\text{ms}]}$$
$$E_{90} = .875\cdot 103.16 + .125\cdot 90$$
$$\boxed{E_{90} = 101.51[\text{ms}]}$$

   We can then find all of the DevRTT times:

$$D_{106} = .75\cdot 5 + .25\cdot [106 - 100.75]$$
$$\boxed{D_{106} = 5.0625[\text{ms}]}$$
$$D_{120} = .75\cdot 5.0625 + .25\cdot [120 - 103.16]$$
$$\boxed{D_{120} = 8[\text{ms}]}$$
$$D_{90} = .75\cdot 8 + .25\cdot [90 - 101.51]$$
$$\boxed{D_{90} = 8.8775[\text{ms}]}$$

   And finally, we calculate the TimeoutInterval values:

$$T_{106} = 100.75 + 4\cdot 5.0625$$
$$\boxed{T_{106} = 121[\text{ms}]}$$
$$T_{120} = 103.16 + 4\cdot 8$$
$$\boxed{T_{120} = 135.16[\text{ms}]}$$
$$T_{90} = 101.51 + 4\cdot 8.8775$$

$$\boxed{T_{90} = 137.02[\text{ms}]}$$

Thus, the timeout intervals calculated after each of the samples can be written as $\langle 121, 135.16, 137.02\rangle[\text{ms}]$.