EE450 Introduction to Computer Networks - Fall 2019 - HW 7

Junzhe Liu, 2270250947

October 31, 2019

1 Problems to be solved:

1.1 Chapter 3, Page 287: R3 (15 points) 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?

source port number y and destination port number x.

1.2 Chapter 3, Page 287: R4 (15 points) Describe why an application developer might choose to run an application over UDP rather than TCP.

Because TCP takes time to build up a connection while UDP does not have such need and thus saves this time. And TCP provides realiable transmission and congestion control, therefore for application which can tolerate certain extent of packet loss and requires a minimum sending rate such as real-time applications, TCP's service is not particularly suitable to their needs. In comparison, UDP does not maintain the connection state and has a relatively smaller packet header, its transmission is considered to be faster than TCP's.

1.3 Chapter 3, Page 287: R5 (15 points) Why is it that voice and video traffic is often sent over TCP rather than UDP in today' s Internet? (Hint: The answer we are looking for has nothing to do with TCP' s congestion- control mechanism.)

Because some firewalls blocks UDP's transmission. TCP is needed to pass through these firewalls.

1.4 Chapter 3, Page 289: P3 (15 points): UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit bytes: 01010011, 01100110, 01110100. What is the 1s complement of the sum of these 8-bit bytes? (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.) Show all work. Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum? With the 1s complement scheme, how does the receiver detect errors? Is it possible that a 1-bit error will go undetected? How about a 2-bit error?)

first segment plus second segment:

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

sum plus third segment:

$$\begin{array}{rr} & 10111001 \\ + & 01110100 \\ \hline = & 00101101 \end{array}$$

takes 1s complement:

$$\begin{array}{rr} & 00101101 \\ \hline = & 11010010 \end{array}$$

This is the checksum of 3 segments. When the receiver adds up the 4 segments, it should get an all-1 byte:

$$01010011 + 01100110 + 01110100 + 11010010 = 11111111 \tag{1}$$

If there is a 1-bit error takes place at some certain bit, then the receiver will find the sum contains a zero at the corresponding bit. However 2-bit error might be ignored because when they happens to occur at the same bit, the effects will be countered.

1.5 Chapter 3, Page 291 : P14 (20 points) Consider a reliable data transfer protocol that uses only negative acknowledgments. Suppose the sender sends data only infrequently. Would a NAK-only protocol be preferable to a protocol that uses ACKs? Why? 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 NAK-only protocol be preferable to a protocol that uses ACKs? Why?

First of all, if the protocol is NAK-only, then the sender might get confused between successfully sent and packet lost. The only situation where the protocol can sense a loss of a certain packet x is that it receives the NAK of $x-1$ and $x+1$, then it realizes that x has never reached the receiver side.

When the sender sends the data infrequently, then the recovery of the lost packet will be very slow. But when the sender has a lot of data to send, then a packet loss can be quickly recovered; moreover, this protocol will reduce the quantity of feedback significantly compared to the ACK-protocol.

1.6 Chapter 3, Page 292: P15 (20 points) Consider the cross-country example shown in Figure 3.17. How big would the window size have to be for the channel utilization to be greater than 98 percent? Suppose that the size of a packet is 1,500 bytes, including both header fields and data.

It takes $1500 * 8/10^9 = 12$ ms to send a packet. In order to reach a utilization percentage of 98:

$$\frac{0.012 \cdot n}{30.012} = 98\% \Rightarrow n \geq 2451 \tag{2}$$