Solutions to Assignment Unit 7

1. Why does the maximum packet lifetime have to be large enough to ensure that not only the packet but also its acknowledgements have disappeared?

The time to live for a packet must be large enough for both the packet and the acknowledgment to ensure that if an ACK is sent, we receive it before the packet is discarded from memory and the next packet sent. Further, we need to ensure there is enough time that a delayed ACK is received before the packet is retransmitted, thus the maximum lifetime is large enough to support both the packet transmitted and the ACK message, reducing wasted bandwidth from retransmission and prevent sending duplicate packets to the receiver.

2. Give one potential disadvantage when Nagle's algorithm is used on a badly congested network.

Nagle's algorithm reduces the number of packets we need to send by buffering and holding data to send a smaller number of large packets instead. In a badly congested network, this can delay an application significantly because, rather than trying to send a series of smaller packets as they appear, we are now waiting. If the large packet becomes lost, or has to wait to transmit, we have increased the delay for the application. When there is high congestion, the smaller packets would have had a better chance to transmit more frequently, thus serving the application data faster.

3. Give two examples of cases where TCP sends data-less packets on an established connection (which is not being torn down).

One case is when one TCP end needs to ACK received data but it does not have any data to send. The other case is when one TCP end has data to send but it has received an advertised window = 0 from the other end. In the latter case, TCP sends data-less packets to “probe” the other end.

4. Exercise 5.0 from section 12.24 of the textbook:

a. Suppose you see multiple TCP connections on your workstation in state FIN_WAIT_1. What is likely going on? Whose fault is it?

As per the TCP state transition diagram, connections in FIN_WAIT_1 are waiting for either an ACK to the FIN that they sent or a FIN from the other end. So, it looks like the other end is not communicating at all (dead remote link or dead remote host?).

b. What might be going on if you see connections languishing in state FIN_WAIT_2?

As per the TCP state transition diagram, connections in FIN_WAIT_2 are waiting for FIN from the other end. So, no FIN from the other end most likely means that the remote application has not closed its side of the connection.