Mokhtari (9831143) – Computer Networks 2 - HW 04

P13 from Chapter **3** of Kurose & Ross's Computer Networking, A Top-D own Approach, **8**th edition:

A video stream is typically fault-tolerant. So, if some packages are lost (for example, due to an overloaded router along the way), it will still be able to display the content, albeit with reduced quality.

If the live stream was using TCP/IP, it would have to wait for those dropped packages before continuing to process newer data.

What's worse is that:

Old data must be re-transmitted (most likely for a frame that has already been displayed and thus has no value), and new data cannot arrive until the old data is re-transmitted.

If the goal is to display as much up-to-date information as possible (which, for a live stream, we usually want to be up-to-date, even if our frames look a little worse), TCP will work against us.

The situation is slightly different for a recorded stream: we'll probably be buffering much more (possibly for several minutes!) and would rather have data re-transmitted than have some artifacts due to lost packages. TCP is a good match in this case (it could still be implemented in UDP, but TCP has fewer drawbacks than UDP in the live stream case).

Mokhtari (9831143) – Computer Networks 2 - HW 04

P15 from Chapter **3** of Kurose & Ross's Computer Networking, A Top-Down Approach, **8**th edition:

The round trip propogation delay between two systems (RTT) = 30 milli seconds

The transmission rate (R) = 1 Gbps

The packet size = 1500 Bytes

$$D_{trans} = L/R = \frac{1500 \times 8 \text{ bits/packet}}{10^8 \text{ bits/sec}}$$

$$= 12 \text{ microseconds}$$

$$= 0.012 \text{ milli seconds}$$

In order for the sender to be busy 95 percent of time, we must have

$$\text{Channel utilization} = \frac{(L/R) \cdot n}{RTT + (L/R)}$$

$$= \frac{(0.012) \cdot n}{30 + 0.012}$$

$$0.98 = \frac{(0.012) \cdot n}{30.012}$$

$$\text{Window size } (n) = \frac{0.98 \times 30.012}{0.012}$$

$$= 2450.98$$

$$\simeq 2451 \quad (\text{approx})$$

Mokhtari (9831143) – Computer Networks 2 - HW 04

P28 from Chapter **3** of Kurose & Ross's Computer Networking, A Top-Down Approach, **8**th edition:

The receive buffer will begin to fill up because Host A will be sending data into it faster than Host B can remove it. When the buffer is completely full, Host B will constantly send a message to Host A telling it to stop sending data until Host B can clear the buffer. Host B will then send Host A a TCP segment, basically telling it to continue sending data. The buffer will be filled again, and the process will be repeated until all of the data has been transferred from Host A to Host B.

P36 from Chapter **3** of Kurose & Ross's Computer Networking, A Top-Down Approach, **8**th edition:

The IP layer can cause packets to arrive out of order. So, whenever an out-of-order packet is received, it generates a duplicate ACK, and if we perform retransmission after the first duplicate ACK, the sender creates too many redundant packets in the network.

In other words, because TCP is unsure whether a duplicate ACK is the result of a lost segment or simply a reordering of segments, it waits for a small number of duplicate ACKs to be received. It is assumed that if only the segments are reordered, there will be one or two duplicate ACKs before the reordered segment is processed, which will generate a new ACK. If we receive three or more duplicate ACKs in a row, it is a strong indication that a segment has been lost. TCP then retransmits what appears to be the missing segment without waiting for the retransmission timer to run out.