Nathan Tung
004-059-195

CS118 Homework 3

1. **Consider distributing a file of *F*=15 Gbits to *N* peers. The server has an upload rate of *us*=30 Mbps, and each peer has a download rate of *di*=2 Mbps and an upload rate of *u*. For N=10, 100, and 1,000 and u=300 Kbps, 700 Kbps, and 2 Mbps, prepare a chart giving the minimum distribution time for each of the combinations of N and u for both client-server distribution and P2P distribution.**
   →Let us first convert values, using a scale of 1024. If we choose to use 1000, the same principles apply with slightly varying results.

   F = 15 Gbits = 15360 Mbits.
   For upload speed u, 300 Kbps = 0.293 Mbps and 700 Kbps = 0.6836 Mbps.

   Client-Server Distribution
   The distribution time (D) is the maximum of either:
   Server Upload Time = F*N/*us* = (15360 Mbits)*N/(30 Mbps)
   = 512N
   Client Download Time = F/*di* = (15360 Mbits)/(2 Mbps)
   = 7680
   *Notice that the client's upload speed doesn't matter at all.

	u=300 Kbps	u=700 Kbps	u=2 Mbps
N=10	7680	7680	7680
N=100	51200	51200	51200
N=1000	512000	512000	512000

   P2P Distribution
   The distribution time (D) is the maximum of any of the following:
   Peer Upload Time = F/*us* = (15360 Mbits)/(30 Mbps)
   = 512
   Peer Download Time = F/*di* = (15360 Mbits)/(2 Mbps)
   = 7680
   Total Upload Time = F*N/(*us*+N*u) = (15360 Mbits)*N/(30 Mbps+N*u)

	u=300 Kbps	u=700 Kbps	u=2 Mbps
N=10	7680	7680	7680
N=100	25903.6	15616.2	7680
N=1000	47558.8	21524.9	7680

Nathan Tung
004-059-195

2. **UDP and TCP use 1s complement for their checksums. Suppose you have the following three 8-bit numbers: 01010101, 01110000, 01001100.**
   a. **What is the 1s complement of the sum of these 8-bit numbers? (Note that although UDP and TCP use 16-bit numbers in computing the checksum, for this problem, you are being asked to consider 8-bit summands.) Show all work.**

      → 01010101
        01110000
      \+ 01001100
      = 00010001 == 1s complemented ==> 11101110

   b. **UDP takes the 1s complement of the sum. In this scheme, how does the receiver detect errors?**

      →The receiver adds the three 8-bit numbers with the checksum (that is, the 1s complement of the sum). If the result is not all 1s (1111…1111), then the receiver detects that an error has occurred.

   c. **What if we just use the sum as the checksum? In this scheme, how does the receiver detect errors?**

      If we use the sum as the checksum (rather than 1s complementing it), then the receiver knows an error has occurred when the checksum and the sum of the three 8-bit numbers do not match.

      In other words, it can subtract the checksum from the sum of the first three numbers and return an error for anything other than all 0s (0000…0000).

   d. **Is it possible that a 1-bit error will go undetected? How about a two-bit error?**

      1-bit errors will never go undetected. For example, if we were to change a bit in one of the 8-bit numbers, then checksum will determine an error.

      However, it is possible that a 2-bit error can occur. For example, swapping a 0 and 1 in the same place value between two words will pass the checksum error detection.

Nathan Tung
004-059-195

3. **In protocol rdt3.0, the ACK packets flowing from the receiver to the sender do not have sequence numbers (although they do have an ACK field that contains the sequence number of the packet they are acknowledging). Why is it that our ACK packets do not require sequence numbers?**
   → As mentioned, the data packet from the sender to the receiver contains sequence numbers such that the receiver determines if that packet is a duplicate. In contrast, the sender doesn't require the ACK packets to bear sequence numbers.

   The sender transitions to the next state upon receiving the correct and expected ACK. Therefore, an incorrect (duplicate) ACK does nothing to alter or affect the sender, regardless of the sequence number – its state is already changed. It is the ACK itself and not the sequence number that causes the sender to change state.

4. **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?**
   → A protocol that uses negative acknowledgements comprises a receiver that issues a NAK only when it realizes a data packet is missing; this occurs when the desired packet is skipped and the receiver obtains the next packet.

   When data is used infrequently, a NAK-only protocol is not ideal. If a data packet is not received, but its subsequent packet is not sent soon after, it will take the receiver a long time to realize that something is erroneously missing.

   When data is used frequently, a NAK-only protocol may prove to be much better than using ACKs, because every time a packet is successfully received, an ACK is issued. On the other hand, a NAK is only issued when a packet is skipped or otherwise missing. In the long run, this has the potential to significantly reduce ACK-issuing overhead.

5. **Consider the cross-country example shown in Figure 3.17. The speed-of-light round-trip propagation delay between these two end systems, RTT, is approximately 30 milliseconds. Suppose that they are connected by a channel with a transmission rate, *R*, of 1 Gbps (109 bits per second). How big would the window size have to be for the channel utilization to be greater than 95 percent? Suppose that the size of a packet is 1,500 bytes, including both header fields and data.**
   → $U = (W*L/R)/(RTT + L/R) > 0.95$
   RTT = 30 ms = 0.03 s
   Transmission Rate, $R = 10^9$ bits/s
   Packet Size, L = 1500 bytes = 12000 bits
   $0.95 = (W*12000/10^9)/(0.03 + 12000/10^9)$
   $0.95 = W*0.000012/0.030012$
   $W = 0.95(0.030012/0.000012) = 2375.95$

   Therefore, window size must be at least 2376 packets to achieve 95% utilization.