Assignment 2: CMPT 371 **Alan Ou**
301320642

Textbook: Chapter 3

P4

a) Suppose you have the following 2 bytes: 0101 1100 and 0110 0101. What is the 1s complement of the sum of these 2 bytes?

$$\begin{array}{r} 1111\ 1 \\ 0101\ 1100 \\ +\ 0110\ 0101 \\ \hline 1100\ 0001 \end{array}$$

therefore 1's complement of 1100 0001 is **0011 1110.**

b) Suppose you have the following 2 bytes: 1101 1010 and 0110 0101. What is the 1s complement of the sum of these 2 bytes?

$$\begin{array}{r} 1 \\ 1101\ 1010 \\ +\ 0110\ 0101 \\ \hline 1\ 0011\ 1111 \end{array}$$ wraparound $$\begin{array}{r} 1111\ 1111 \\ 1101\ 1010 \\ +\ 0110\ 0101 \\ \hline 1\ 0100\ 0000 \end{array}$$

therefore 1's complement of 0100 0000 is **1011 1111.**

c) For the bytes in part (a), give an example where one bit is flipped in each of the 2 bytes and yet the 1s complement doesn't change.

Suppose that the 2 bytes were: 0101 1101 and 0110 0100. The last bit in the first byte changed from a 0 to 1 and the last bit in the second byte changed from a 1 to a 0. The sum of the two bytes would be as follows:

$$\begin{array}{r} 1111\ 1 \\ 0101\ 1101 \\ +\ 0110\ 0100 \\ \hline 1100\ 0001 \end{array}$$

therefore 1's complement of 1100 0001 is **0011 1110.**

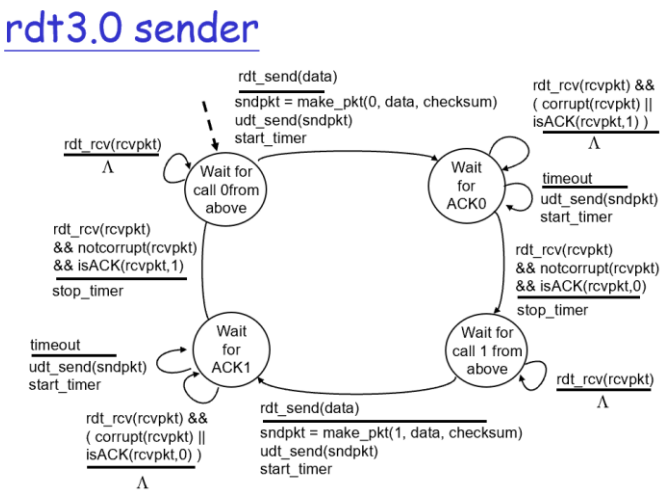
As shown, the 1s complement does not change.

P5. Suppose that the UDP receiver computes the Internet checksum for the received UDP segment and finds that it matches the value carried in the checksum field. Can the receiver be absolutely certain no bit errors have occurred? Explain.

No, the receiver cannot be absolutely certain that no bit errors have occurred because as noted in class, even if the "computed checksum equals checksum field value" there can still be errors. The explanation uses something similar to what is shown in part C of question 1.

For example, since the checksum value consists of two or more 16-bit word values encoded in binary, which only have 1's and 0's, if there are one or more pairs of bits that has one bit change from 1 to 0 and one bit change from 0 to 1, then the overall checksum value will remain the same. This would fool the receiver into believing that there were no bit errors.

P10. Consider a channel that can lose packets but has a maximum delay that is known. Modify protocol rdt2.1 to include sender timeout and retransmit. Informally argue why your protocol can communicate correctly over this channel.



We would add a countdown timer to "wait for ACK0" and "wait for ACK1". This would make the protocol similar to rdt3.0 as shown in the diagram to the left, from the course notes. Once a particular timeout is reached, that packet gets retransmitted.

The protocol can communicate correctly over this channel because the receiver sees no change; rdt2.1 can handle all the situations that this new protocol can.

Consider the scenario that an ACK is lost. When an ACK is lost, once the sender reaches its timeout, it retransmits the packet, but the receiver will see that it is a duplicate and handles the duplicate in the way it knows how to deal with duplicates in rdt2.1; it discards the duplicate packet and retransmits the ACK.

Consider the case where there is a loss packet, the receiver will notice nothing because the packet never reached the receiver. Once the timeout is reached by the sender, the sender will transmit the packet and the receiver receives the packet.

P24. Answer true or false

a) With the SR protocol, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

True, consider a premature timeout situation.

For example, if we had a window size of 2.

At t = 0, packet 0 is sent by the sender.
At t = 1, packet 1 is sent by the sender and ACK 0 is sent by the receiver.
At t = 2, ACK 1 is sent by the receiver.
At t = 3, the sender timeouts and packet 0 is resent.
At t = 4, the sender timeouts and packet 1 is resent.

At t = 5, the receiver sees the duplicate and sends ACK 0
At t = 6, the sender receives ACK 0 sent at t = 1 and moves window to 1,2
At t = 7, the receiver sees the duplicate and sends ACK 1
At t = 8, the sender receives ACK 1 sent at t = 2 and moves window to 2,3
At t = 9, the sender receives ACK 0 sent at t = 3
At t = 10, the sender receives ACK 1 sent at t = 4

As seen, the ACKs received at t = 9 and t = 10 are outside of the sender's window.

b) With GBN, it is possible for the sender to receive an ACK for a packet that falls outside of its current window.

True, the explanation is the same as in part A.

c) The alternating-bit protocol is the same as the SR protocol with a sender and receiver window size of 1.

True. The difference between the SR protocol, GBN protocol, and alternating-bit protocol is the sender and receiver window size; however, if we set the window sizes to 1, then all protocols will be the same.

d) The alternating-bit protocol is the same as the GBN protocol with a sender and receiver window size of 1.

True, same explanation as part C.

P25

We have said that an application may choose UDP for a transport protocol because UDP offers finer application control (than TCP) of what data is sent in a segment and when.

a) Why does an application have more control of what data is sent in a segment?

An application has more control of what data is sent in segment when UDP is chosen as the transport protocol because with UDP, all the data that is sent to UDP will be packaged inside a UDP segment. On the contrary, TCP grabs chunks of data up to the maximum segment size. As a result, data may be separated into different segments.

b) Why does an application have more control on when the segment is sent?

An application that uses UDP will have more control on when the segment is sent as UDP does not require connection establishment, has no congestion control or flow control, all of which can cause significant delay. TCP on the other hand, requires all those, as a result, the delay takes control away from the application.

Assignment 2: CMPT 371 **Alan Ou**
301320642

P31

Suppose that the five measured SampleRTT values (see Section 3.5.3) are 106 ms, 120 ms, 140 ms, 90 ms, and 115 ms. Compute the EstimatedRTT after each of these SampleRTT values is obtained, using a value of $\alpha = 0.125$ and assuming that the value of EstimatedRTT was 100 ms just before the first of these five samples were obtained. Compute also the DevRTT after each sample is obtained, assuming a value of $\beta = 0.25$ and assuming thevalue of DevRTT was 5 ms just before the first of these five samples was obtained. Last, compute the TCP TimeoutInterval after each of these samples is obtained.

EstimatedRTT = (1- α)* EstimatedRTT + α*SampleRTT

DevRTT = (1- β) * DevRTT + β * | SampleRTT – EstimatedRTT |

TimeoutInterval = EstimatedRTT + 4*DevRTT

Sample RTT 106 ms

EstimatedRTT = (1- α)* EstimatedRTT + α*SampleRTT

= (1- 0.125)* 100 + 0.125*106

= **100.75 ms**

DevRTT = (1- β) * DevRTT + β * | SampleRTT – EstimatedRTT |

= (1-0.25) * 5 + 0.25 * |106 – 100.75|

= 5.0625 ms

TimeoutInterval = EstimatedRTT + 4*DevRTT

= 100.75 + 4 * 5.0625

= 121 ms

Sample RTT 120 ms

EstimatedRTT = (1- α)* EstimatedRTT + α*SampleRTT

= (1- 0.125)* 100.75 + 0.125 * 120

= **103.15625 ms**

DevRTT = (1- β) * DevRTT + β * | SampleRTT – EstimatedRTT |

= (1-0.25) * 5.0625 + 0.25 * |120 – 103.15625|

= 8.0 ms

TimeoutInterval = EstimatedRTT + 4*DevRTT

= 103.15625 + 4 * 8

= 135.15625 ms

Sample RTT 140 ms

EstimatedRTT = (1- α)* EstimatedRTT + α*SampleRTT

= (1- 0.125)* 103.15625 + 0.125*140

= **107.76 ms**

DevRTT = (1- β) * DevRTT + β * | SampleRTT – EstimatedRTT |

= (1-0.25) * 8 + 0.25 * |140 – 107.76|

= 14.06 ms

TimeoutInterval = EstimatedRTT + 4*DevRTT

= 107.76 + 4 * 14.06

= 164 ms

Sample RTT 90 ms

EstimatedRTT = (1- α)* EstimatedRTT + α*SampleRTT

= (1- 0.125)* 107.76 + 0.125*90

= **105.54 ms**

DevRTT = (1- β) * DevRTT + β * | SampleRTT – EstimatedRTT |

= (1-0.25) * 14.06 + 0.25 * |90 – 105.54|

= 14.43 ms

TimeoutInterval = EstimatedRTT + 4*DevRTT

= 105.54 + 4 * 14.43

= 163.26 ms

Sample RTT 115 ms

EstimatedRTT = (1- α)* EstimatedRTT + α*SampleRTT

= (1- 0.125)* 105.54 + 0.125*115

= **106.7225 ms**

DevRTT = (1- β*)* * DevRTT + β * | SampleRTT – EstimatedRTT |

= (1-0.25) * 14.43 + 0.25 * |115 – 106.7225|

= 12.8919 ms

TimeoutInterval = EstimatedRTT + 4*DevRTT

= 106.7225 + 4 * 12.8919

= 158.29

P32(a)(b)

Consider the TCP procedure for estimating RTT. Suppose that α = 0.1(see slides 61 of Chapter 3 for more details on α). Let SampleRTT1 be the most recent sample RTT, let SampleRTT2 be the next most recent sample RTT and so on.

a. For a given TCP connection, suppose four acknowledgements have been returned with corresponding sample RTTs:

EstimatedRTT$_1$ = $SampleRTT_1$

EstimatedRTT$_2$ = α x $SampleRTT_1$ + (1 − α) x $SampleRTT_2$

EstimatedRTT$_3$ = (1 − α) x $SampleRTT_3$ + α x $SampleRTT_2$

= α x $SampleRTT_1$ + (1 − α) x α x $SampleRTT_2$ + (1 − α)^2 x $SampleRTT_3$

EstimatedRTT$_4$ = (1 − α) x $SampleRTT_4$ + α x $SampleRTT_3$

= α x $SampleRTT_1$ + (1 − α) x α x $SampleRTT_2$ + (1 − α)^2 x α x $SampleRTT_3$ + (1 − α)^3 x $SampleRTT_4$

b. Generalize your formula for n sample RTTs.

$$\alpha \sum_{i=1}^{n-1} ((1-\alpha)^i \, x \, SampleRTT_i) + \, (1-\alpha)^n \, x \, n$$

P40

Consider figure 3.58. Assuming TCP Reno is the protocol experiencing the behaviour shown above, answer

a) Identify the intervals of time when TCP slow start is operating

The intervals are **[1,6] and [23, 26].**

b) Identify the intervals of time when TCP congestion avoidance is operating

The intervals are **[6,16] and [17,22].**

c) After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout

After the 16th transmission round, segment loss is detected by a triple duplicate ACK. It is a triple duplicate ACK and not a timeout because a timeout would cause a slow start and cause the congestion window to drop down 1 MSS.

d) After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?

After the 22nd transmission round, the segment loss is detected by a timeout because this time, the congestion window dropped down to 1 MSS which does not happen if the segment loss was detected by a triple duplicate ACK.

e) What is the initial value of ssthresh at the first transmission round?

The initial value of ssthresh at the first transmission round is approximately at 32 as this is where slow start stops.

f) What is the value of ssthresh at the 18th transmission round?

The sender reacts to a packet loss by a multiplicative decrease, in other words, cuts the congestion window size by half. As a result, as a packet is lost at round 16, the threshold was cut from 42 to 21. Therefore, the value of ssthresh is **21**.

g) What is the value of ssthresh at the 24th transmission round?

Similar to the above explaination, when a packet loss was detected at round 22, the threshold got cut in half, going from 29 to 14.5, which we either take the ceiling or floor of, which turns to 15 or 14 respectively.

h) During what transmission round is the 70th segment sent?

By some estimations, at round 1, 1 segment is sent; at round 2, 2 segments, totaling 3 segments; at round 3, 4 segments, totaling 7; at round 4, 8 segments, totaling 15 segments; at round 5, 16 segments, totaling 31 segments; at round 6, 32 segments, totaling 63 segments; at round 7, 33 segments, totaling 96 segments.

Therefore, the 70th segment is sent in the 7th transmission round.

i) Assuming a packet loss is detected after the 26th round by the receipt of a triple duplicate ACK, what will be the values of the congestion window size and of ssthresh?

The congestion window size and ssthresh will both be set to **4** as the original congestion window size was around 8, when cut in half equals 4.

j) Suppose TCP Tahoe is used (instead of TCP Reno), and assume that triple duplicate ACKs are received at the 16th round. What are the ssthresh and the congestion window size at the 19th round?

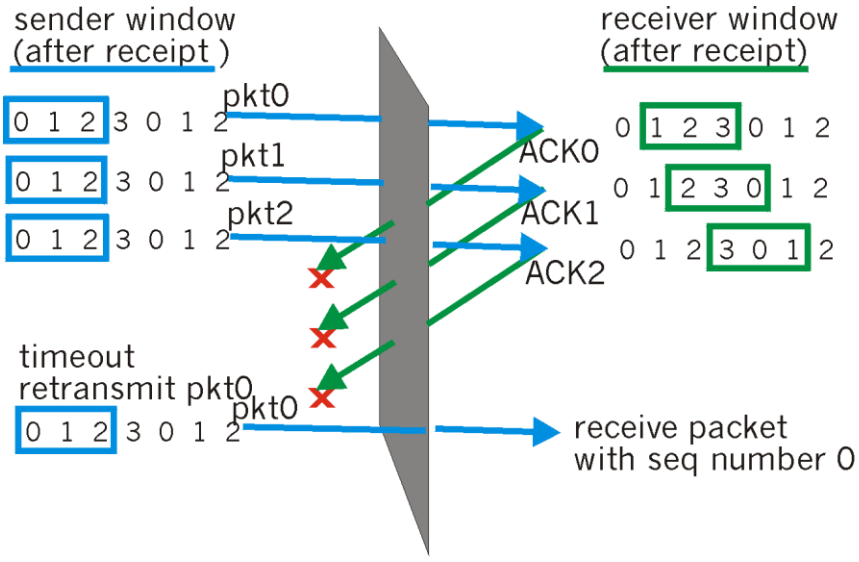
The ssthresh would be 42/2 = **21** and congestion window size is set to 1 MSS at round 16. At round 19, the ssthresh would remain at 21 but the congestion window size would be set to 4 MSS.

k) Again suppose TCP Tahoe is used, and there is a timeout event at 22nd round. How many packets have been sent out from 17th round till 22nd round, inclusive?

At round 17, 1 segment is sent; at round 18, 2 segments, totaling 3 segments; at round 19, 4 segments, totaling 7; at round 20, 8 segments, totaling 15 segments; at round 21, 16 segments, totaling 31 segments; at round 22, 21 segments, totaling **52** packets.

Additional questions

(1) Show that if the sequence number space is identical to the window size, then neither GBN nor selective repeat will work.



Using the example in the class notes 3-72. We are provided that the sequence number space is 0, 1, 2, 3 and window size is 3.

The difference between GBN and selective repeat that we would have considered is the window size, however since the sequence number space and window size are identical in both GBN and selective repeat, then there is no difference, therefore the explanation below can be used to explain why GBN and selective repeat will not work.

Consider the diagram from the class notes. The sequence number space is 0, 1, 2, 3 then it repeats 0, 1, 2, 3 and so on. This is where the problem occurs. No problem occurs during the first round, or in other words, when the first packets numbered 0, 1, 2, 3 are sent and received. When the sender sends the second packet 0 and every packet after, even though the packet may possibly contain different data, when the receiver receives this packet, it will think that it is the packet 0 it received before, as a result, the receiver would consider the second packet 0 and later packets as duplicates and discard it.

(2) Explain in what scenario is the destination IP address necessary for TCP demultiplexing.

As mentioned in the course textbook, when TCP is used as the transport protocol, the destination IP is required for all scenarios as each TCP socket requires all four fields (source IP address, source port, destination IP address, destination port) in order to demultiplex the segment to the correct socket.

As mentioned in the class notes, this is useful as the “server host may support many simultaneous TCP sockets [and] Web servers have different sockets for each connecting client”.

(3) For bursty loss, which protocol is better, GBN or selective repeat?

For bursty loss, GBN would be better because after one packet timeouts, all the packets that are numbered after that packet are resent; there is no need to wait for individual timeouts. Selective repeat only resends the packet when the packet timeout is reached, which is unnecessary time wasted.