**CSCE416 Exam2 Review Anthony Frazier**

**R4: Describe why an application developer might choose to run an application over UDP rather than TCP.**

Developer may not need reliable data transfer, and may also not want to use the TCP congestion control.  
  
**R6: Is it possible for an application to enjoy reliable data transfer even when the application runs**

Yes, it is possible, the developer can put reliable data transfer into the application layer, but this requires a significant amount of work and debugging.

**R7: Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port 6789. Will both of these segments be directed to the same socket as Host C? If so, how will the process at Host C know that these two segments originated from two different hosts?** Yes, both segments can use the same socket. At the socket interface, the operating system will provide the process with the IP address for the origins.

**R8: Suppose that a Web server runs in Host C on port 80. Suppose this Web server uses persistent connections and is currently receiving requests from two different Hosts, A and B. Are all the requests being sent through the same socket at Host C? If they are being passed through different sockets, do both sockets have port 80? Discuss and explain.**  
  
For each persistent connection, the Web server creates a separate “connection socket”. Each is identified with a four-tuple: (source IP address, source port number, destination IP address, destination port number). When host C receives an IP datagram, it examines this tuple to determine to which socket it should pass the payload of the TCP segment. Thus, the requests from A and B pass through different sockets. The identifier for both sockets has 80 for the destination port; however, the identifiers for these sockets have different values for source IP addresses. Unlike UDP, when the transport layer passes a TCP segment’s payload to the application process, it does not specify the source IP address, as this is implicitly specified by the socket identifier.

**R14: True or false?**

1. **Host A is sending Host B a large file over a TCP connection. Assume Host B has no data to send Host A. Host B will not send acknowledgements to Host A because Host B cannot piggyback the acknowledgments on data.** False, TCP must have acknowledgement.
2. **The size of the TCP *rwnd* never changes throughout the duration of the connection?**

False, the rwnd represents the receiver’s free buffer space, which could change.

1. **Suppose Host A is sending Host B a large file over TCP connection. The number of unacknowledged bytes that A sends cannot exceed the size of the receiver buffer.**

True, the receiver buffer will limit the amount of data allowed at first, at least until an ack allows more file to be sent.

1. **Suppose Host A is sending a large file to Host B over a TCP connection. If the sequence number for a segment of this connection is m, then the sequence number for the subsequent segment will necessarily be m + 1.** False, the sequence number increments by the number of bytes of data sent.
2. **The TCP segment has a field in its header for *rwnd.***

True, this is how the rwnd can change from sequence to sequence.

1. **Suppose that the last *SampleRTT* in a TCP connection is equal to 1 sec. The current value of *TimeoutInterval* for the connection will necessarily be >= 1 sec.**

False.  
  
**TimeoutInterval = EstimatedRTT + 4\*DevRTT.**

**DevRTT = 0.75\*DevRTT + 0.75\*|SampleRTT-EstimatedRTT|  
EstimatedRTT = 0.875\*EstimatedRTT + 0.125\*SampleRTT**

1. **Suppose Host A sends one segment with sequence number 38 and 4 bytes of data over a TCP connection to Host B. In this same segment the acknowledgement number is necessarily 42.**

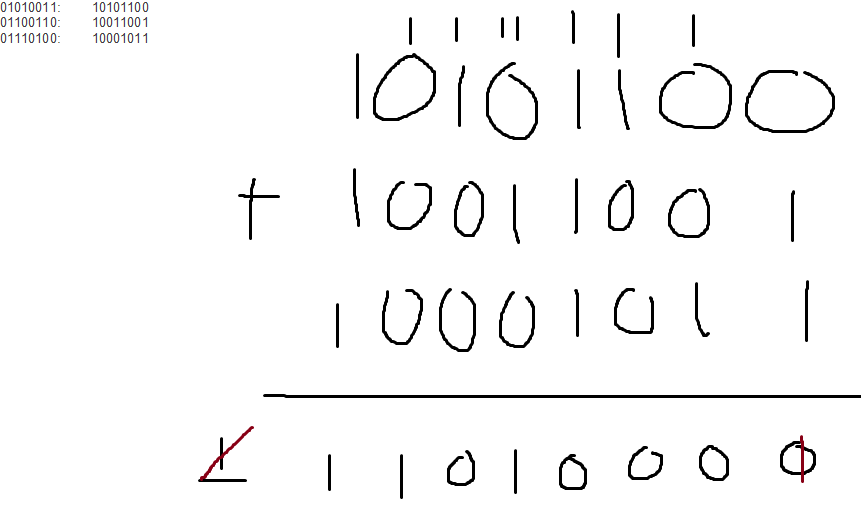
False, when sending a sequence, ack is always 1. The segment B replies to A will include ACK 42 and sequence 1.

**R15: Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has sequence number 90; the second has sequence number 110.**

1. **How much data is in the first segment?** 20 bytes.
2. **Suppose that the first segment is lost but the second segment arrives at B. In the acknowledgement that Host B sends to Host A, what will be the acknowledgement number?**

90. TCP acknowledgement is based on what the server has received up to but not including (<90)

**R18: True or false? Consider congestion control in TCP. When the timer expires at the sender, the value of *ssthresh* is set to one half of its previous value.** False, the threshold is sent to one half its current congestion size

**P3: 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? Show all work.**

**Why is it that UDP takes the 1s complement of the sum; that is, why not just use the sum?**

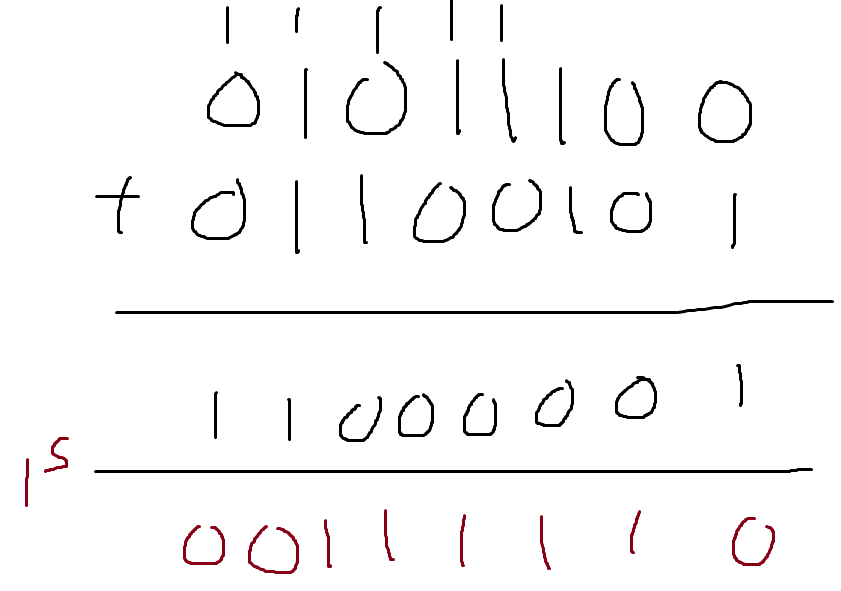
Use the 1s complement for error checking.

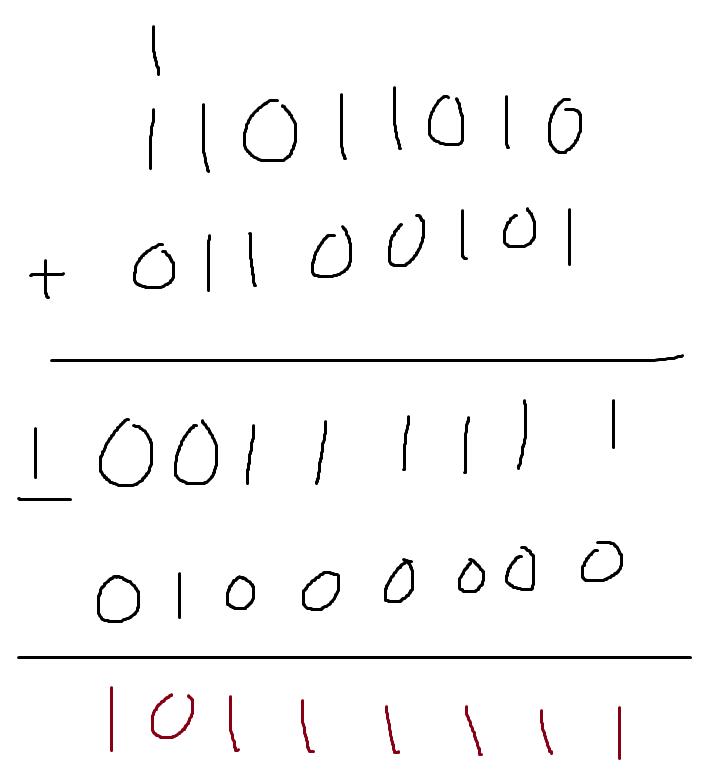
**With the 1s complement scheme, how does the receiver detect errors?**

The receiver adds the 3 original words and the 1s complement of the sum. If any digit is a 0, this is an error.

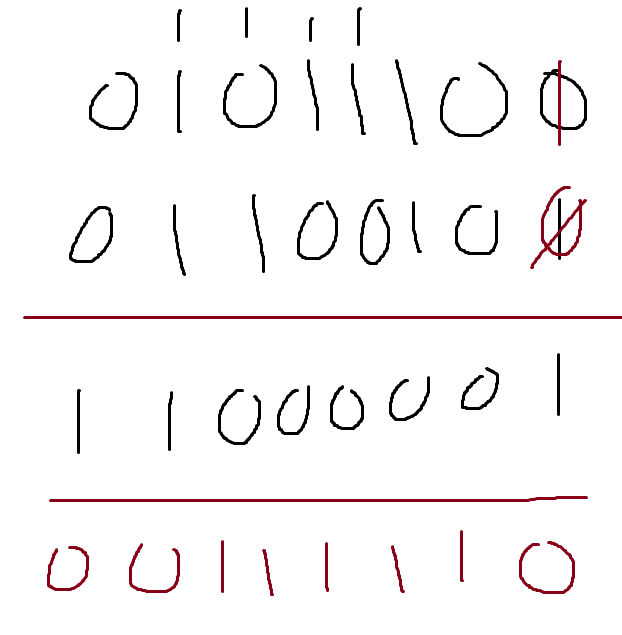
**Is it possible that a 1-bit error will go undetected?** It is not possible for a 1-bit error to go undetected.

**How about a 2-bit error?** Two bit errors can be undetected. If the last digit of the first word is converted to a 0 and the last digit of the second word is converted to a 1.  
 **P4: a) Suppose you have the following 2 bytes: 01011100 and 01100101. What is the 1s complement of the sum of these 2 bytes?**

  
  
 **b) Suppose you have the following 2 bytes: 11011010 and 01100101. What is the 1s complement of the sum of these 2 bytes?**



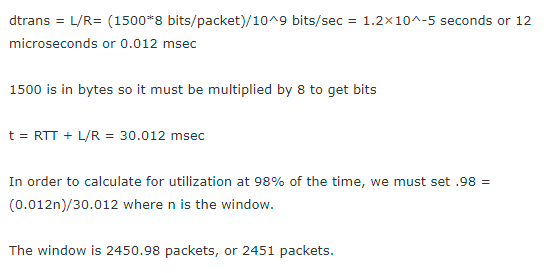
1. **For the bytes in part (a), give an example where one bit is flipped in each of the 2 bytes and yet the 1s complement doesn’t change.**



**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 that no bit errors have occurred? Explain.**

No, the receiver cannot be absolutely certain that no 2-bit errors have occurred.

**P15: 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.**

  
**P22: Consider the GBN protocol with a sender window size of 4 and a sequence number range of 1,024. Suppose that at time t, the next in-order packet that the receiver is expecting has a sequence number of k. Assume that the medium does not reorder messages. Answer the following questions:   
  
 a) What are the possible sets of sequence numbers inside the sender’s window at time t? Justify your answer.**We have not received up to k correctly: (k-4,k-3,k-2,k-1), (k-3,k-2,k-1,k), (k-2,k-1,k,k+1), (k-1,k,k+1,k+2), We have received up to but not including k: (k, k+1, k+2, k+3)

**b) What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time t? Justify your answer.** We have either received up to k-1, k-2, k-3, k-4, k-5. Due to the window size being 4, the ack must include one of these values.  
  
**P23: Consider the GBN and SR protocols. Suppose the sequence number space is of size k. What is the largest allowable sender window that will avoid the occurrence of problems such as that in Figure 3.27 for each of these protocols?** k must be at least twice the size of the window size to avoid this problem.

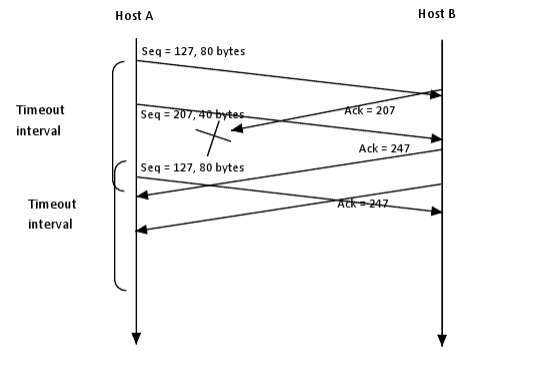
**P27: Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A.**

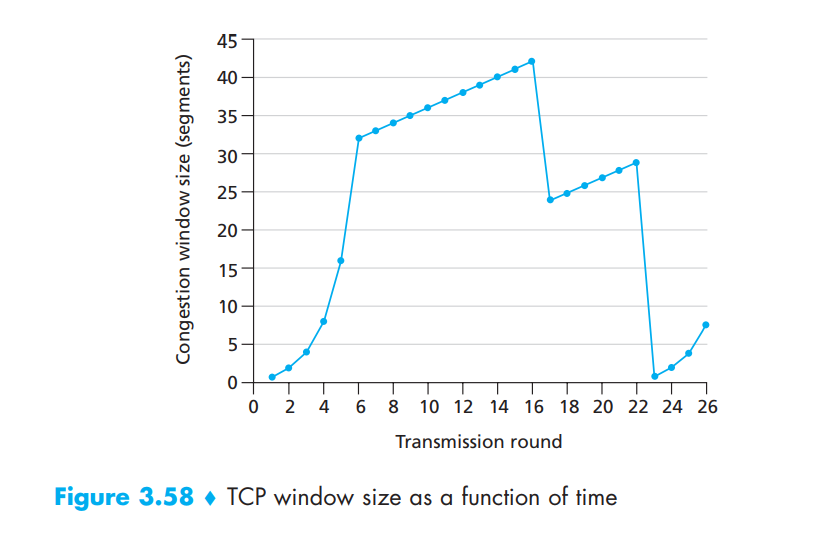
**a. In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?** Sequence number will be 207 source is 302 and destination is 80.

**b. If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number, the source port number, and the destination port number?**ack number will be 207, source is 80 and destination is 302.

**c. If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?** The ack number will be 127, because it is still waiting on the first segment.

**d. Suppose the two segments sent by A arrive in order at B. The first acknowledgment is lost and the second acknowledgment arrives after the first timeout interval. Draw a timing diagram, showing these segments and all other segments and acknowledgments sent. (Assume there is no additional packet loss.) For each segment in your figure, provide the sequence number and the number of bytes of data; for each acknowledgment that you add, provide the acknowledgment number.**



**P36: In Section 3.5.4, we saw that TCP waits until it has received three duplicate ACKs before performing a fast retransmit. Why do you think the TCP designers chose not to perform a fast retransmit after the first duplicate ACK for a segment is received?** If three duplicate acks are received, this is a strong indicator that the segment has been lost. Sending on one or two could end up with too many duplicate packets being transmitted.   


**P40: Consider Figure 3.58. Assuming TCP Reno is the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying your answer.**

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

1-6 and 23-26

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

6-16 and 17-22

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

Triple ack(cw is not set to 1 here)

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

Timeout, cw IS set to 1

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

32

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

42/2 = 25

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

29/2 = 14

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

SSTH 32

Round 1: cw = 1, segment = 1

Round 2: cw = 2, segment = 3

Round 3: cw = 4, segment = 7

Round 4: cw 8, segment = 15

Round 5: cw:16, segment = 31

Round 6: cw: 32, segment = 63

SSTH threshold reached, now in congestion control.

Round 7: cw: 33, segment = 96 will be sent in round7.

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?**

SSTH = 8/2 = 4, CW = SSTH + 3 = 7  
  
**--------------------------------------------------------**

TIMEOUT- SSTH = CW/2; CW = 1

TRIPLE ACK- SSTH = CW/2 + 3; CW = SSTH

Slow Start INC- CW < SSTH

Every RTT CW = cw\*2

Every ACK CW = CW+1

Congestion Avoidance- cw >= SSTH

Every RTT CW = cw+1  
Every Ack cw = cw + 1/CW

-Suppose congestion window was 16 segments when timeout occured. How many round trips does it take to deliver 48 segments? (Assume no more lost packets)

//Timout occured at 16 segments, thus we can say that at the time of timeout, SSTH = CW/2, so SSTH = 8. CW is set to 1 at this point.

We double CW each time until we hit SSTH.

Round 1: cw = 1

segments delivered: 1

Round 2: cw = 2

segments delivered: 3

Round 3: cw = 4

segments delivered: 7

Round 4: cw = 8

segments delivered: 15

we have hit **SSTH** here, so now we are only increasing by 1.

Round 5: cw = 9

segments delivered: 24

Round 6: cw = 10

segments delivered: 34

Round 7: cw = 11

segments delivered: 45

Round 8: cw = 12

segments delivered: 57 (segment delivered here).

So it will take 8 round trips to deliver 48 segments.