CS118 Quiz 2, Spring 2020

Name: ____________________________

Student ID: Solution

Notes:

1. This is an open-book, open-notes quiz. You have two hours to work on your quiz, scan or photo your paper copy, and upload to the Gradescope.
2. You need to upload your scanned copy or the photoed picture of your answer sheet to the Gradescope before the deadline.
3. You are allowed to use your calculator. You are advised **not** to use the Internet to search for hints during the quiz. By submitting your quiz, you declare that **your work is solely done by yourself and you have not interacted with anyone else other than the instructor and proctors during the test**.
4. If you have any issues with the quiz, you can tune in the regular Lecture Zoom link and use the Chatroom there during the quiz time. We will provide clarifications there during the quiz.
5. Be **brief** and **concise** in your answers. Answer only within the space provided. If you need additional work sheets, use them but do NOT submit these sheets.
6. If you wish to be considered for partial credit, show all your work.
7. **Show your steps to receive partial credit**. *For the TCP congestion control problem, you can type or write your answer in whatever format. Moreover, you may further draw your diagram on your answer sheet, take a picture of your diagram and upload it to show your steps.*
8. You have 6 problems in 7 pages plus this page.

PROBLEM	MAX SCORE	YOUR SCORE
1	20	
2	15	
3	12	
4	14	
5	9	
6	20	
TOTAL	90	

Problem 1 (20 points): Multiple choices (2 points each). Select <u>all</u> correct answers.

1. B Which field(s) in the UDP header are used in connectionless demultiplexing?
   - Your answer ____ (A) Source port number; (B) Destination port number; (C) Length; (D) Sequence number.

2. C When can the first data segment start its transmission during the TCP connection lifetime?
   - Your answer ____ (A) together with the first SYN message; (B) before the second SYN+ACK message is received; (C) together with the third ACK message; (D) After the third ACK message has arrived at the receiver.

3. B,C Which statement(s) on UDP would be **incorrect**?
   - Your answer ____ (A) UDP provides the Internet's connectionless service. (B) UDP also implements congestion control. (C) UDP uses the 4-tuple of (source-IP-address, destination-IP-address, source-port-number, destination-port-number) as its identifier. (D) UDP does not implement reliable data transfer.

4. D Which header field(s) will appear in **one but not both** UDP and TCP segment headers?
   - Your answer ____ (A) Source port number; (B) Destination port number; (C) Checksum; (D) Sequence number.

5. D Which mechanism is **not** absolutely required to ensure reliable transfer upon corrupted packets (but without packet loss)?
   - Your answer ____ (A) error detection via checksum; (B) sequence numbers for transmitted packets; (C) retransmissions; (D) Pipelined transmission of data packets.

6. C Assume Selective-Repeat protocol uses a window size of 10 packets at the sender side. How many unique sequence numbers do we need *at least* if the receiver's buffer can be arbitrarily large?
   - Your answer ____ (A) 2; (B) 10; (C) 20; (D) 11.

7. B Which mechanism is **not** absolutely required to ensure reliable transfer upon packet loss (but without packet corruption)?
   - Your answer ____ (A) sequence numbers for transmitted packets; (B) checksum; (C) retransmission; (D) acknowledgment numbers for received packets.

8. D In the Go-back-N Protocol, given the sender's window [201,202] and N =2, select which is **impossible** to be the receiver's next expected sequence number.
   - Your answer ____ (A) 201; (B) 202; (C) 203; (D) 204.

9. A Suppose the TCP sender's congestion window size is 12 segments, and its receiver's advertised window is 10 segments. What is the maximum number of back-to-back segments TCP can send?
   - Your answer ____ (A) 10; (B) 12; (C) 20; (D) 24.

10. C Assume a TCP sender transmits multiple segments back to back, with only the first one being lost and all others arriving at the receiver in order. Upon receiving three duplicate ACKs, the sender runs fast retransmit (FR) with the new parameters being ssthresh=5 segments and cwnd=8 segments. After running FR, what is the maximum number of TCP segments in the current sender window that can still be in flight without reaching the receiver according to FR?

    - Your answer ____ (A) $5/2 \approx 2$; (B) $8/2=4$; (C) 5; (D) 8.

Problem 2 (15 points; 3 points each): Answer the following questions. Be brief and concise.

1. A UDP receiver computes the Internet checksum for the received UDP segment and finds a *mismatch* with the checksum field. Can the receiver be absolutely sure that bit errors have occurred with **the received UDP data payload**? Assume it matches the value carried in the checksum field. Can the receiver be absolutely sure that no bit errors have occurred? Briefly justify your answer. (hint: a UDP segment includes the UDP header and the UDP data payload)

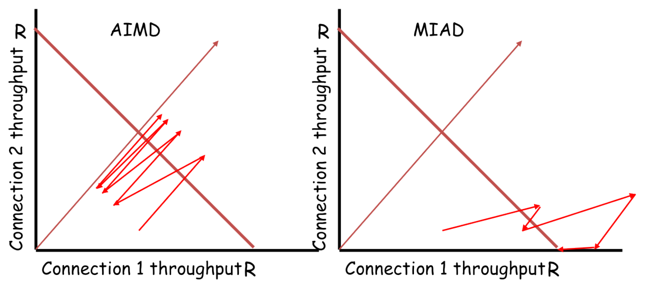
   (1)No. The error might happen in the UDP header. (1.5 points deducted if yes).
   (2)No, the receiver cannot be absolutely certain that no bit errors have occurred.
   If bit errors transform the packet to another one with identical checksum value, then such errors will go undetected.

2. Can the four-way handshake (i.e., 4 message exchanges) in closing a TCP connection be reduced similarly to the three-way handshake of establishing a TCP connection in *any arbitrary* TCP connection setting? Justify your answer.

   No. Different from the TCP connection setup, closing the TCP connection needs to tear down two one-way data communications. Each side can tear down its one-way connection after data sending is done from its side. However, it cannot tell whether the one-way connection in the reverse direction has completed its data transfer. Therefore, it has to go through two one-way connection teardown, i.e., 4-way handshake in the general case.

3. Consider two TCP connections sharing a single link, with identical round-trip-times and segment size. Additive-increase, multiplicative-decrease (AIMD) can ensure fairness for both TCP connections eventually. Now some one claims that multiplicative-increase, additive-decrease (MIAD) can also ensure fairness eventually for these two connections, starting from any arbitrary initial window size. Show why this is NOT true. You can draw a figure to help your explanation.

MIAD cannot ensure fairness as illustrated in the figure:



4. TCP congestion control is designed with the premise that fast retransmit/fast recovery (FR/FR) is used more frequently for common scenarios, rather than retransmission timeout. Identify *two* (not-so-common) scenarios where FR/FR is not triggered but timeout is activated.

   1. when cwnd is very small, no 3rd duplicate ACK can be sent by the receiver; 2. when multiple back-to-back losses are seen in the same window, timeout may be used; 3. TCP experiences segment losses at its early stage of running the slow start algorithm; 4. RTO value is very small (e.g., in data center networks).

5. For the Go-Back-N protocol with the sender window size of 16, what is the minimum number of bits needed for the Sequence number field?

   The number of sequence numbers needed is: 16+1=17. Therefore, 5-bit sequence number field is needed.

Problem 3 (12 points): Reliable Data Transfer Protocol In the following scenario, discuss how each reliable transfer protocol reacts:

1. (3 points) In the Stop-and-Wait protocol (i.e., rdt3.0), how does the sender react when a corrupted acknowledgment is received?

Take no action upon receiving corrupted acknowledgement based on FSM for rdt3.0.

2. (3 points) Assume cumulative acknowledgment (ACK) is used and each data segment is 100B. In the Selective-Repeat protocol (assuming the sender window size remains unchanged here), how does the sender react when receiving an ACK segment (with acknowledgment number 900B) right after receiving a duplicate ACK (with acknowledgment number 800B)?

   (1) Leap the window forward for 2 segments; this is indicated by ACK 900 (as this new ACK covers the previous duplicate ACK); (2) send two new data packets allowed by the window; (3) associate the timer with each of two sent segments (per-segment timer) or the 1st segment in the window (per-window timer).

3. (3 points) Someone designs a new reliable transfer protocol by combining the Go-back-N protocol with TCP fast retransmit. In this new protocol, if timeout is triggered at the sender before the sender receives the third duplicate ACK, how does the sender react? Justify your answer.

   1. retransmit all the packets in the current window; 2. restart the timer with the 1st packet in the window.

4. (3 points) Will premature timeout (i.e., the retransmission timer has a value smaller than RTT) affect the correctness of reliable transfer protocol? Briefly justify your answer.

   No. Premature timeouts only trigger unnecessary retransmissions, but the reliable data transfer is still ensured correctly.

Problem 4 (14 points): Figure 1 shows two clients (A and C) and one Web server B using TCP.

TCP segments sent from the clients to the server are shown in the figure. The IP addresses of Hosts A, B and C are given as 1.1.1.1, 2.2.2.2, and 3.3.3.3, respectively. You are asked to answer the following questions:

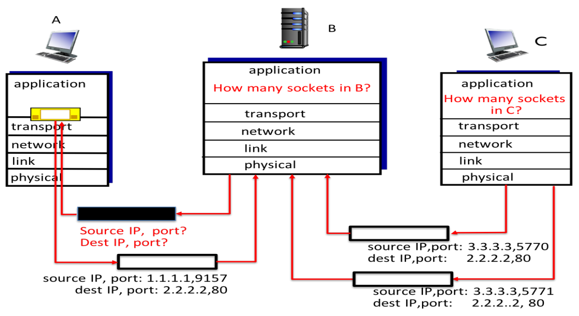


Figure 1: Clients A and B connect with Server C.

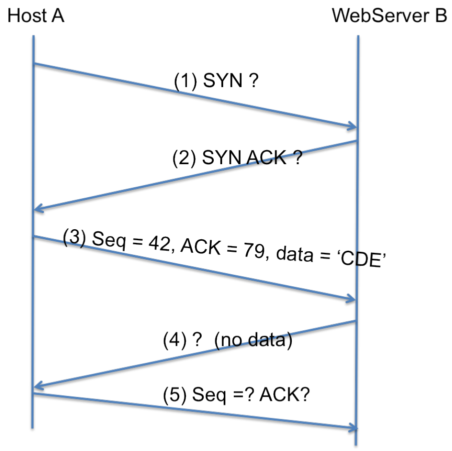
1. (2 points) What is the source IP, source port, dest IP dest port in the packet from the server B to the client A?

   Source IP: 2.2.2.2; dest IP: 1.1.1.1; source port: 80; dest port: 9157

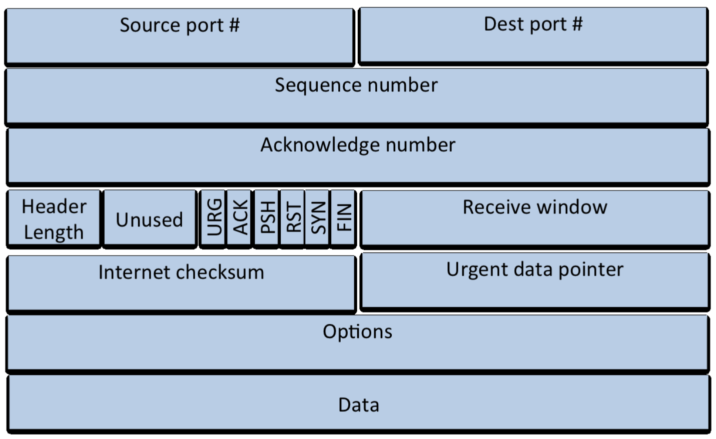
2. (2 points) How many sockets are running at client C side and how many sockets are at server B? You only need to count those network sockets (with external endpoints over the Internet).

   C has 2 sockets and B has 3 TCP sockets.

3. The below left figure shows the message exchanges between A and B. The third message from A to B contains payload *"CDE"* (3 bytes) with its sequence and ACK numbers being shown in the figure. The server replies with an ACK (in the fourth message), but carries no data.



(a) TCP Message Exchanges



(b) TCP Header

   (a) (2 points) What is the procedure performed over the first three messages?

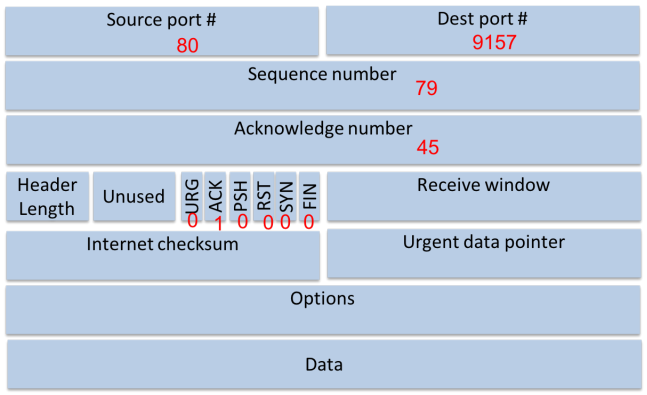
TCP connection establishment.

(b) (4 points) Please list the sequence number and ack number for the first, second and fifth message.

the first packet is SYN, Seq number = 41. No ACK number.
the 2nd segment is SYN ACK. Seq number = 78. ACK number =42
the 5th segment: Seq number = 45, ACK number = 79.

(c) (4 points) Given this information, please fill in for the fourth message in the above right figure (Hint: Fill in as many fields as possible).

The forth packet is given as:



Problem 5 (9 points): TCP Retransmission Timeout

1. (6 points) You are asked to compute the retransmission timeout (RTO) for TCP. The initial estimated round-trip time (RTT) is set as 80ms, and initial RTT variation is 40ms. The RTT samples for 3 TCP segments are 160ms, 400ms, 200ms. In these 3 segments, the second TCP segment has been retransmitted once, and the third TCP segment has been retransmitted twice. Compute *all* RTO values upon receiving *each* of three TCP segments. Show the intermediate steps in your calculation. Please use $\alpha = \frac{7}{8}$ and $\beta = \frac{3}{4}$ in your computation.

For $\alpha = \frac{1}{8}$ and $\beta = \frac{1}{4}$ RTO for the 1st segment: RTO=280ms.
The calculation is as follows:
$estRTT = 80 \times \frac{7}{8} + 160 \times \frac{1}{8} = 90;$
$estDev = 40 \times \frac{3}{4} + \frac{1}{4} \times |160 - 90|;$
$RTO = estRTT + 4 \times estDev = 280$
RTO for the 2nd and 3rd segments is still 280ms. This is Karn's algorithm, since 2nd and 3rd segments will be ignored in RTO computation due to retransmissions.
For $\alpha = \frac{7}{8}$ and $\beta = \frac{3}{4}$ RTO for the 1st segment: RTO=400ms.
The calculation is as follows:
$estRTT = 80 \times \frac{1}{8} + 160 \times \frac{7}{8} = 150;$
$estDev = 40 \times \frac{1}{4} + \frac{3}{4} \times |160 - 90|;$
$RTO = estRTT + 4 \times estDev = 400$
RTO for the 2nd and 3rd segments is still 400ms. This is Karn's algorithm, since 2nd and 3rd segments will be ignored in RTO computation due to retransmissions.

2. (3 points) Now assume that the RTO timer (based on the above calculation) for the 4th segment expires three times. TCP retransmits the 4th segment whenever the timer expires during the process. What is the final RTO value afterwards? Show your reasoning.

Exponential growth of timeout value. $RTO = 280ms \times 2^3 = 8 \times 280ms = 2240ms.$
Exponential growth of timeout value. $RTO = 400ms \times 2^3 = 8 \times 400ms = 3200ms.$

Problem 6 (20 points): TCP Congestion Control You will work on two scenarios on TCP congestion control. Note Part 1 and Part 2 are for different TCP connections; they are not correlated.

1. (8 points) Consider a scenario that a timeout event has been observed at the TCP sender. When the timeout occurs, the congestion window size *cwnd* at the sender is 9 segments. In this scenario, we assume that the receiver's advertised window size is always larger than 16 segments.

   (a) (2 points) How does TCP congestion control update its *cwnd* upon timeout?

   cwnd=1 segment.

   (b) (3 points) How does TCP congestion control update its *ssthresh* (i.e., slow start threshold value) upon timeout? Show your steps.

   apply timeout-based congestion control as follows:
   $ssthresh = 9/2 \approx 4$.

   (c) (3 points) Can the TCP sender transmit a new segment in addition to retransmitting the segment that experiences timeout? Briefly justify your answer.

   No. Since cwnd=1, TCP cannot transmit another new segment except the retransmitted one.

2. (12 points, 3 points each) Consider another new TCP connection. Assume that all algorithms are implemented in TCP congestion control: slow start, congestions avoidance, fast retransmit and fast recovery, and retransmission upon timeout. For fast recovery phase, if $ssthresh = cwnd$, use the slow start algorithm upon the non-duplicate new ACK.

   You can draw your diagram on a separate sheet, scan or photo it, and upload it for submission to receive maximum partial credit.

   - TCP uses reliable transfer. The used ACK for each segment is based on cumulative ACK and the acknowledgment number indicates the next expected segment. The receiver acknowledges every segment (which is NOT corrupted), and the sender always has data available for transmission.
   - Initially *ssthresh* at the sender is set to 7, and *cwnd* as 1. Assume *cwnd* and *ssthresh* are counted in segments, and the transmission time for each segment is negligible (equivalently, you can assume that each segment size is conceptually 1B). Retransmission timeout (RTO)

is initially set to 500ms and remains unchanged during the connection lifetime. The RTT is 100ms for all transmissions.

- The connection starts from the initial sequence number of 1 at t=0. Segment with sequence number 6 is lost once. Segments with sequence number 9 and 10 arrived at the receiver out of order (i.e., segment 10 arrives before segment 9). No other misbehavior is observed.

(a) The receiver sends an ACK upon receiving segment with sequence number 5. What algorithm should the sender use when receiving this ACK? What is the updated value for *cwnd* and *ssthresh* after running this algorithm?

slow start when receiving ACK number 6; cwnd=6; ssthresh=7.

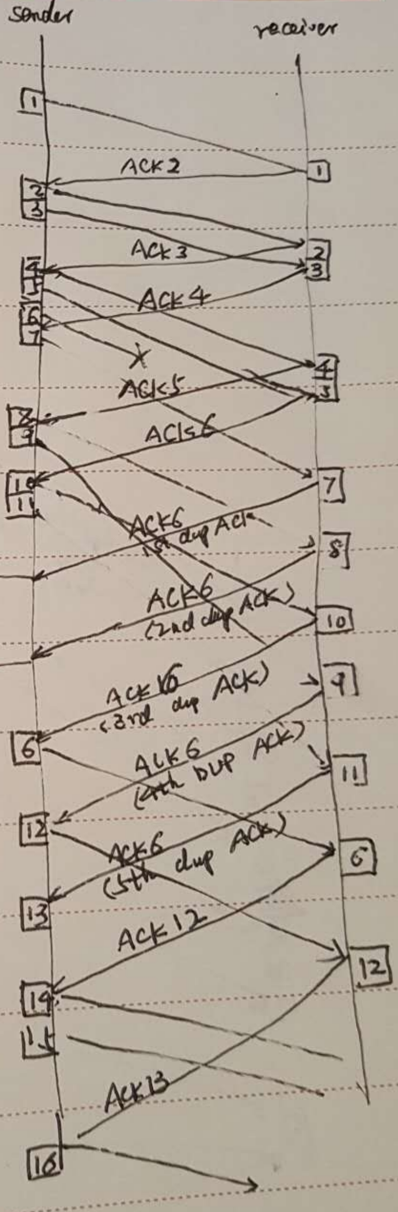
(b) The receiver sends an ACK upon receiving segment with sequence number 10. What algorithm should the sender use when receiving this ACK? What is the updated value for *cwnd* and *ssthresh* after running this algorithm?

Fast retransmit algorithm up receiving this 3rd duplicate ACK.
$ssthresh = \frac{6}{2} = 3$; $cwnd = 3 + 3 = 6$.

(c) The receiver sends an ACK upon receiving the segment with sequence number 9. What is the sequence of actions the sender takes upon receiving this ACK? What is the updated value for *cwnd* and *ssthresh*?
Fast recovery algorithm.
1. update $cwnd = 6 + 1 = 7$; 2. send segment with sequence number 12.
cwnd=7; ssthresh=3.

(d) What is the acknowledgment number when the receiver receives the retransmitted segment 6? What is the updated value for *cwnd* and *ssthresh* at the sender after receiving this ACK and the corresponding invoked congestion control algorithm(s)?
The ACK number is 12 when receiving the ACK for retransmitted segment 6.
both fast recovery (for receiving a non-duplicate new ACK) and slow start algorithms are invoked here.
cwnd=4; ssthresh=3.

algorith	ssth	cwnd	SR
SS	7	1	$\boxed{1}$
SS		1+1=2	$\boxed{2,3}$
SS		2+1=3	$\boxed{3.4.5}$
SS		3+1=4	$\boxed{4.5.6.7}$
SS		4+1=5	$\boxed{5.6.7.8.9}$
SS		5+1=6	$\boxed{6.7.8.9.10.11}$
nothing	7	6	$\boxed{6,7,8,9,10,11}$
nothing	7	6	$\boxed{6,7,8,9,10,11}$
fast rexmit	SS= $\frac{6}{2}=3$	cwnd= 3+3=6	$\boxed{6,7,8,9,10,11}$
fast recovery	3	cwnd= 6+1=7	$\boxed{6,7,8,9,10,11,12}$
fast recovery	3	cwnd= 7+1=8	$\boxed{6,7,8,9,10,11,12,13}$
fast recovery	3	cwnd=3	$\boxed{12,13,14}$
slow start	3	cwnd=3+1	$\boxed{12,13,14,15}$
congestion avoidance	3	cwnd ~~4~~ $4+\frac{1}{4}$ ≈ 4	$\boxed{13,14,15,16}$


sender
receiver
ACK2
ACK3
ACK4
ACK5
ACK6
ACK6 1st dup ACK
ACK6 (2nd dup ACK)
ACK 6 (3rd dup ACK)
ACK6 (4th DUP ACK)
ACK6 (5th dup ACK)
ACK12
ACK13