

# UCLA-19S-CS118 Midterm

Galen Wong

TOTAL POINTS

**84 / 100**

QUESTION 1

Problem 1 18 pts

1.1.1.1 D 2 / 2

✓ - 0 pts Correct

1.2 1.2 C 2 / 2

✓ - 0 pts Correct

1.3 1.3 B 1 / 2

✓ - 1 pts More chosen

1.4 1.4 B 2 / 2

✓ - 0 pts Correct

1.5 1.5 D 2 / 2

✓ - 0 pts Correct

1.6 1.6 C 0 / 2

✓ - 2 pts Incorrect

1.7 1.7 BC 2 / 2

✓ - 0 pts Correct

1.8 1.8 A 2 / 2

✓ - 0 pts Correct

1.9 1.9 C 2 / 2

✓ - 0 pts Correct

QUESTION 2

Problem 2 18 pts

2.1 3 / 3

✓ - 0 pts Correct

2.2 3 / 3

✓ - 0 pts Correct

2.3 3 / 3

✓ - 0 pts Correct

2.4 3 / 3

✓ - 0 pts Correct

2.5 3 / 3

✓ - 0 pts Correct

2.6 2 / 3

✓ - 1 pts Got 17. But wrong bits

QUESTION 3

3 Problem 3 6 / 8

✓ + 2 pts One Item Correct

✓ + 2 pts One Item Correct

✓ + 2 pts One Item Correct

+ 2 pts One Item Correct

QUESTION 4

Problem 4 12 pts

4.1 3 / 4

✓ - 1 pts Others (additional RTT)

4.2 3 / 4

✓ - 1 pts [Click here to replace this description.](#)

4.3 2 / 4

✓ - 2 pts Partially Correct

QUESTION 5

5 Problem 5 10 / 12

✓ - 2 pts Hdr len wrong

8.5 Figure 0 / 0

✓ - 0 pts Correct

QUESTION 6

Problem 6 12 pts

6.1 3 / 3

✓ - 0 pts Correct

6.2 2 / 3

✓ - 1 pts No drop the packet

6.3 2 / 3

✓ - 1 pts Wrong advanced window (not 900)

6.4 3 / 3

✓ - 0 pts No restarting the timer

QUESTION 7

Problem 7.1 8 pts

7.1 2 / 2

✓ - 0 pts Correct

7.2 3 / 3

✓ - 0 pts Correct

7.3 3 / 3

✓ - 0 pts Correct

QUESTION 8

Problem 7.2 12 pts

8.1 3 / 3

✓ - 0 pts Correct

8.2 3 / 3

✓ - 0 pts Correct

8.3 3 / 3

✓ - 0 pts Correct

8.4 1 / 3

✓ - 2 pts 4

# CS118 Midterm Exam, Spring 2019

Name: Galen Wong

Student ID: \_\_\_\_\_

## Notes:

1. This is a closed-book, closed-notes examination. But you can use the one-page cheat sheet.
2. You are not allowed to use your electronic devices.
3. 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 with the midterm examination.
4. Use the back pages for scratch paper. You should cross out your scratch work when you submit your exam paper. Any answers or work shown on the back of any page will NOT be considered or graded.
5. If you wish to be considered for partial credit, show all your work.
6. Make sure that you have 10 pages (including this page and one-page Appendix) before you begin.

PROBLEM	MAX SCORE	YOUR SCORE
1	18	
2	18	
3	8	
4	12	
5	12	
6	12	
7	20	
TOTAL	100	

**DO NOT TURN TO THE NEXT PAGE UNLESS YOU GET PERMISSION !!**

**Problem 1: Multiple choices (2 points each).** Select all the correct answers from the four choices. Note that there can be *multiple* correct answers.

For your reference, the following are the full names for the used Acronyms: HTTP: hypertext transfer protocol; DNS: Domain name system; SMTP: Simple mail transfer protocol; POP3: Post office protocol version 3; IMAP: Internet Mail access protocol; TCP: transmission control protocol; UDP: User datagram protocol.

1. What would be the correct feature(s) of a file-distribution application using the peer-to-peer (P2P) model rather than the client-server model?
  - Your answer \_\_\_\_ (A) All peers have to remain always on; (B) A peer cannot leave even after its file downloading completes; (C) P2P model scales worse than the client-server model; (D) A peer can act as a client by issuing requests while serving other peers.
2. Which protocol is not used when Bob uses his smartphone to browse the ESPN Web site?
  - Your answer \_\_\_\_ (A) TCP; (B) HTTP; (C) IMAP; (D) DNS.
3. 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.
4. What is true regarding Dynamic Adaptive Streaming over HTTP (DASH) for video streaming?
  - Your answer \_\_\_\_ (A) The video is encoded into several different versions, which have different qualities but have the same bit rate; (B) It allows clients with different Internet access rates to stream in video; (C) The client selects different chunks one at a time with HTTP PUT request messages; (D) It does not allow a client to dynamically adjust its streaming rate once the streaming session starts.
5. Which is a benefit of packet switching but not circuit switching?
  - Your answer \_\_\_\_ (A) congestion may occur inside the network; (B) reservation is always needed before data delivery; (C) providing delay guaranteed services; (D) statistical multiplexing
6. When can the first TCP data segment starts its transmission during the TCP connection?
  - 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.
7. Which of the following statement about DNS is true?
  - Your answer \_\_\_\_ (A) A local DNS server never queries the root DNS server; (B) DNS uses caching to improve performance; (C) Some of DNS queries can be iterative and others recursive, in the sequence of queries to translate a hostname; (D) Only authoritative DNS servers can respond to queries.
8. Which layers in the protocol stack are implemented at the end host?
  - Your answer \_\_\_\_ (A) application layer, transport layer, network layer, link layer, physical layer; (B) application layer, transport layer; (C) network layer, link layer, physical layer; (D) application layer only.

9. Which mechanism of HTTP is used to allow a cache to verify that its objects are up to date?

- Your answer \_\_\_\_ (A) Cookies; (B) stateless HTTP server; (C) conditional GET; (D) HTTP with persistent connections.

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

10 8  
10<sup>4</sup>/<sub>10</sub>  
10:1

1. In an ongoing TCP connection, the TCP sender receives a new Acknowledgment segment, which has its 'Receive Window' field set as 11000 Bytes. Before receiving this Acknowledgment, the TCP sender has not perceived any segment loss with its *cwnd* being 10000 Bytes and its *ssthresh* as 8000 Bytes. What is the window size updated by TCP in its reliable transfer right after receiving this Acknowledgment? Assume the maximum segment size (MSS) is 1000 Bytes. Show your steps.

With additive increase, new  $cwnd = 10000 + \frac{1000}{10000} \times 1000 = 110100$  Bytes

$\therefore cwnd < \text{Receive Window}$ .

$\therefore$  The window size =  $cwnd = 10100$  Bytes.

2. Consider the queuing delay in a router buffer (preceding an outbound link). Suppose all packets are  $L$  bits, the transmission rate is  $R$  bps, and that all  $N$  packets simultaneously arrive at the buffer at time  $t=0$ . Find the queuing delay of the packet that is transmitted last.

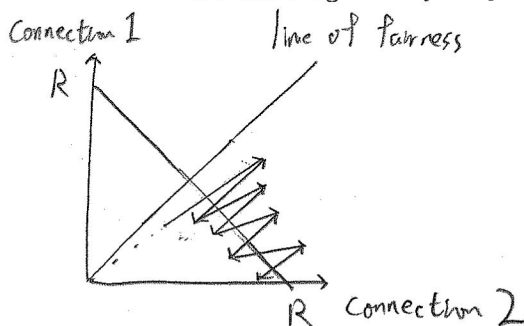
The last packet has to wait for  $N-1$  packets to be transmitted.

$$\therefore \text{delay}_{\text{last}} = \frac{(N-1)L}{R}$$

3. A new video streaming startup *v-start* decides to use the third-party CDN provider *limelight* to scale and reduce streaming latency. Explain how to leverage DNS to intercept and redirect the user clicks on <http://video.v-start.com/> to the CDN servers deployed by *limelight*.

- ① Client request manifest file
- ② Client resolves video URL with *v-start* DNS.
- ③ *v-start* DNS returns CDN's DNS that is close to client
- ④ Client resolves video URL with *limelight*'s DNS, which gives URL to CDN server.
- ⑤ Client connect to given CDN server.

4. Consider two TCP connections sharing a single link, with identical round-trip-times and segment sizes. The additive-increase, multiplicative-decrease (AIMD) mode is known to ensure fair share for both connections eventually. Now a programmer decides to use multiplicative-increase, additive-decrease (MIAD) mode in his implementation. Explain whether this change will result in starvation of a connection (i.e., if starting from an arbitrary window size, one connection will eventually be starved to the lowest speed while the other will get the highest speed) or not. You can use a figure in your justification.



With multiplicative increase, the vector grows away from the origin.  
With additive decrease, the vector decrease with slope 1.

This leads to convergence towards one connection and starvation of the other connection.

$\therefore$  MIAD will result in starvation.

5. Consider that amazon.com uses a session cookie to track the shopping cart on Susan's browser. Can ebay.com reuse the same cookie? Can amazon.com reuse the same cookie on Bob's browser?  
 ebay.com cannot reuse the same cookie. Cookies are specific to websites.  
 amazon.com cannot reuse Susan's cookie on Bob's browser.  
 Cookie has to be unique to each user.
6. 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?

$$\text{Max Seq \#} = N + 1 = 17$$

**Problem 3 (8 points):** Joe is writing programs with a client and a server using stream sockets. The following is the SERVER code that Joe wrote. Can you help Joe to find four errors (there can be more) in his code? You can mark your answers in his code, and label the errors in the code. You can use the Appendix for references.

```

#include <server.h> /* assume all headers are included correctly */
#define PORT 8080
int main(int argc, char const *argv[])
{
    int server_fd, new_fd; /* listen on server_fd, new connection on new_fd */
    struct sockaddr_in address;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char *hello = "Hello from server";

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) > 0) {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }
    address.sin_family = PF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = ntohs(PORT);
    if (bind(server_fd, (struct sockaddr *)&address, addrlen) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }
    if (accept(server_fd, 3) < 0) {
        exit(EXIT_FAILURE);
    }
    if ((new_fd = listen(server_fd, (struct sockaddr *)&address, (socklen_t*)&addrlen)) < 0) {
        exit(EXIT_FAILURE);
    }
    int valread = read(server_fd, buffer, 1024);
    printf("%s\n", buffer);
    sendto(new_fd, hello, strlen(hello), 0);
    printf("Hello message sent\n");
}

```

*should be less than (<)*  
*↓*  
*should be AF\_INET.*  
*← we are converting to network format, not from. ∴ we should use htons(PORT);*  
*error, we read from new\_fd.*  
*← dangerous, buffer might not be 0 terminated!*  
*← By specification, sendto takes in 2 more arguments.*  
*∴ sendto(new\_fd, hello, strlen(hello), 0, NULL, 0);*

**Problem 4 (12 points): Delay in Application Layer Protocols** One end host A visits store.google.com and www.amazon.com sequentially to compare the price of a smart watch.

- (4 points) (a) Assume Host A's local DNS server's cache is empty initially. Therefore, Host A needs to get the IP address of store.google.com via DNS query. Also assume that:
  - Only iterative DNS queries are used.
  - The RTT between Host A and the local DNS server is 20 ms.
  - The RTT between the local DNS server to any authoritative DNS server is 100 ms.
  - Ignore any DNS server's processing time.
  - TTL value for any record is 1 hour.
  - Any domain under google.com is hosted by ns.google.com; any domain under amazon.com is hosted by ns.amazon.com.

How many milliseconds would have elapsed when Host A gets the IP of store.google.com?

- ① visit Root DNS    ② visit com DNS    ③ visit google.com DNS  
 ④ visit ns.google.com DNS,

$$\therefore \text{total time} = 20 + 4 \times 100 = 420 \text{ ms}$$

$$\begin{array}{r} 404 \\ \times 14 \\ \hline 1616 \\ 498 \\ \hline 2024 \end{array}$$

- (4 points) (b) Host A next visits www.amazon.com to compare the price of the same product. Assume www.amazon.com points to a 1000-Byte HTML file which references 4 images. Each referenced image size is 500 Bytes. The one-way propagation delay between Host A and Amazon's Web server is 100 ms and the transmission rate is 1 Mbps. When calculating the transmission delay, assume only HTTP response payload counts for that. If Host A uses HTTP 1.0, how many milliseconds would have elapsed when Host A receives all referenced images of www.amazon.com since it inputs the URL www.amazon.com in a browser's address bar?

- ① visit com DNS    ② visit amazon.com DNS    ③ visit ns.amazon.com DNS  
 ④ time to resolve IP =  $20 + 3 \times 100 = 320 \text{ ms}$

$$RTT = 2 \times 100 = 200 \text{ ms}$$

$$\text{time to get HTML} = 2RTT + \frac{1000 \times 8 \text{ b}}{1 \text{ Mbps}} = 408 \text{ ms}$$

$$\text{time to get one image} = \underbrace{2RTT}_{400} + \frac{500 \times 8 \text{ b}}{1 \text{ Mbps}} = 404 \text{ ms}$$

$$\therefore \text{total time} = 408 + 404 \times 4 = 2024 \text{ ms}$$

- (4 points) (c) Repeat (b), assuming Host A uses HTTP 1.0 with maximum 4 parallel connections. All other assumptions are the same.

$$\text{time to resolve IP} = 320 \text{ ms}$$

$$\text{time to get HTML} = 408 \text{ ms}$$

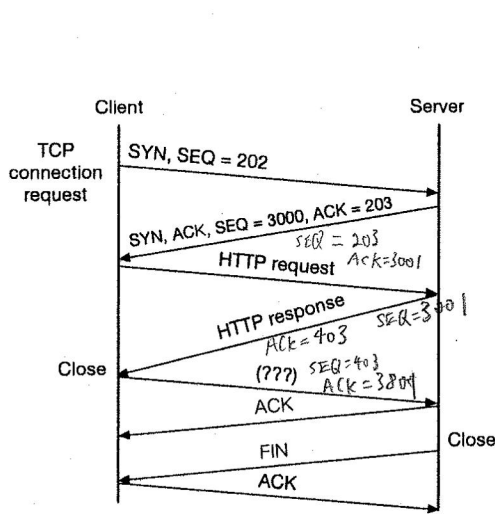
$$\text{time to get all images} = 404 \text{ ms}$$

$$\therefore \text{total time} = 812 \text{ ms}$$

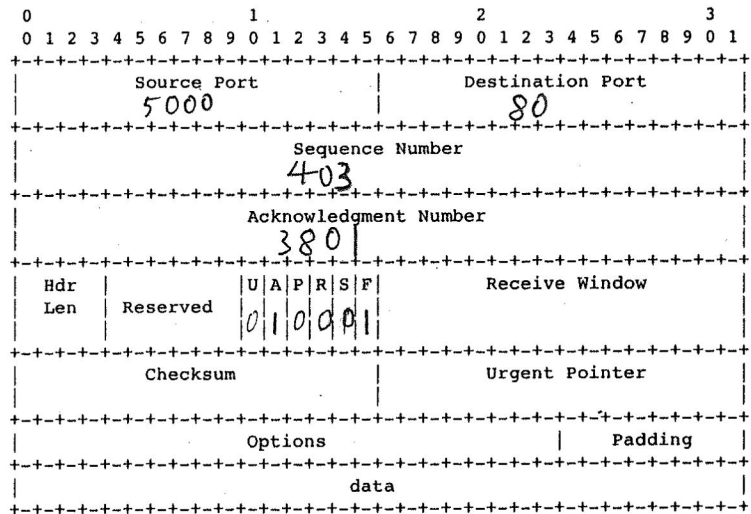
$$\begin{array}{r} 408 \\ + 04 \\ \hline 812 \end{array}$$

**Problem 5 (12 points): TCP Protocol** You are asked to fill in as many fields as possible in the TCP header (given in the right subfigure) for the fifth TCP message (marked with (???)) shown in the left figure. The left figure shows the sequence of messages being exchanged between the client and the Web server, starting from TCP connection establishment, HTTP request/response, to TCP connection close.

Write your answers in decimal format directly in the right figure. Assume that non-persistent HTTP is used. The HTTP request is 200 Bytes, while the HTTP response from the server is 800 Bytes. Client and Server's IP addresses are 111.111.111.111 and 222.222.222.222, respectively, and the client uses port 5000 for the TCP connection (hint: the TCP port number used on the Web server is 80).



(a) TCP Message Exchanges



(b) TCP Header

**Problem 6 (12 points): Reliable Transfer Protocol** In the following scenario, discuss how each reliable transfer protocol reacts:

- (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, it should not. Premature timeout triggers retransmission of segments. However, reliable transfer protocol handles duplicate segments correctly. The only impact of premature timeout is that we send unnecessary traffic.  
 (a good protocol, like TCP, should adjust timeout accordingly as well!)

- (3 points) In the Stop-and-Wait protocol, how does the receiver reacts if a duplicate data packet is received?

We should send an ACK for the last valid packet, namely the duplicated packet, since there is only the 'last' and 'current' packet in stop-and-wait.



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

We should mark the segment with sequence number 800 as received by receiver.

We advance the window if segment with SEQ# = 800 is the beginning of the window.

- (3 points) In the Go-back-N protocol, if timeout is triggered at the sender before the sender receives the third duplicate ACK, how does the sender react? Justify your answer.

Go-back-N should not care about duplicate ACKs.

It only retransmits all packets/segments within the window upon timeout.

Therefore, GoBN just retransmits all segments in the window upon timeout.

**Problem 7 (20 points): TCP Congestion Control** You will work on two scenarios on TCP congestion control. Note that Part 1 and Part 2 are considering 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 10 segments.

- (2 points) How does TCP congestion control update its  $cwnd$  upon timeout?

We set  $cwnd$  to 1 segment.

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

$$ssthresh = \max\left(\frac{cwnd}{2}, 2\right) = \max\left(\frac{9}{2}, 2\right) = \left\lfloor \frac{9}{2} \right\rfloor = 4$$

- (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 currently the  $cwnd$  size is 1. We can only retransmit the loss segment.  $cwnd$  does not allow us to transmit a new segment.

2. (12 points) Consider another new TCP connection. Assume that all algorithms are implemented in TCP congestion control: slow start, congestion avoidance, fast retransmit and fast recovery, and retransmission upon timeout. Right after fast retransmit/fast recovery phase, if  $ssthresh = cwnd$ , use the slow start algorithm. You must draw a diagram to show the intermediate steps on Page 9 to receive full 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, and the sender always has data available for transmission.
- Initially  $ssthresh$  at the sender is set to 8, 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 one unit). 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  $t=0$ . Segments with sequence number 4 and 5 arrived at the receiver out of order (i.e., segment 5 arrives before segment 4). Segment with sequence number 7 is lost once. No other misbehavior is observed.

(a) (3 points) 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  $ssthresh$  upon this?

since the ACK is not a new ACK, sender should not take any action.

$$cwnd = 4, ssthresh = 8.$$

(b) (3 points) The receiver sends an ACK upon receiving segment with sequence number 4. What algorithm should the sender use when receiving this ACK? What is the updated value for  $cwnd$  and  $ssthresh$  upon this?

It should use slow start.

$$cwnd = 5, ssthresh = 8.$$

(c) (3 points) The receiver sends an ACK upon receiving the segment with sequence number 10. What is the sequence of actions the sender takes upon receiving this ACK?

It is the 3rd duplicate ACK 7.

Sender should use fast retransmit.

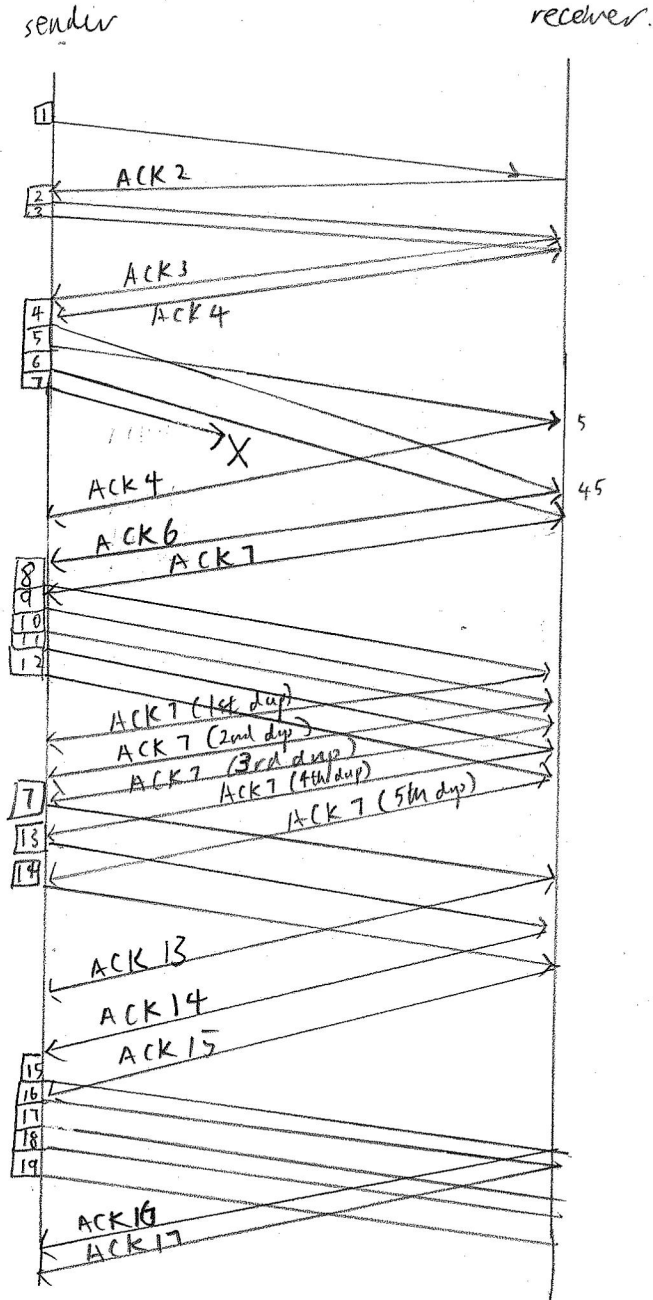
$$It \text{ set } ssthresh = \frac{cwnd}{2} = 3 \text{ and } cwnd = ssthresh + 3 = 6.$$

(d) (3 points) What is the congestion window size  $cwnd$  at the sender when the sender transmits segment with sequence number 16?

It is 5.

Draw your diagram for Problem 7.2 here.

cwnd	sssthresh	policy	window
1	8	ss	1
2	8	ss	2 3
4	8	ss	4 5 6 7
5	8	ss	6 7 8 9 10
6	8	ss	7 8 9 10 11 12
6	3	fr	7 8 9 10 11 12
7	3	fr	7 8 9 10 11 12 13
8	3	fr	7 8 9 10 11 12 13 14
5	3	fr	13 14 15 16 17
$5 + \frac{1}{5} = 5$	3	ca	13 14 15 16 17
$5 \frac{1}{5} + \frac{1}{5} = 5$	3	ca	14 15 16 17 18
$5 \frac{2}{5} + \frac{1}{5} = 5$	3	ca	15 16 17 18 19



## Appendix. Socket Programming Function Calls.

- *struct in\_addr* { *in\_addr\_t s\_addr*; /\* 32-bit IP addr \*/ }
- *struct sockaddr\_in* {  
  *short sin\_family*; /\* e.g., AF\_INET \*/  
  *ushort sin\_port*; /\* TCP/UDP port \*/  
  *struct in\_addr*; /\* IP address \*/ }
- *struct hostent\* gethostbyaddr* (*const char\* addr*, *size\_t len*, *int family*)  
  *struct hostent\* gethostbyname* (*const char\* hostname*);  
  *char\* inet\_ntoa* (*struct in\_addr inaddr*);  
  *int gethostname* (*char\* name*, *size\_t namelen*);
- *int socket* (*int family*, *int type*, *int protocol*);  
  [*family*: AF\_INET (IPv4), AF\_INET6 (IPv6), AF\_UNIX (Unix socket); *type*: SOCK\_STREAM (TCP), SOCK\_DGRAM (UDP); *protocol*: 0 (typically)]
- *int bind* (*int sockfd*, *struct sockaddr\* myaddr*, *int addrlen*);  
  [*sockfd*: socket file descriptor; *myaddr*: includes IP address and port number; *addrlen*: length of address structure == sizeof(struct sockaddr\_in)]  
  returns 0 on success, and sets *errno* on failure.
- *int sendto* (*int sockfd*, *char\* buf*, *size\_t nbytes*, *int flags*, *struct sockaddr\* destaddr*, *int addrlen*);  
  [*sockfd*: socket file descriptor; *buf*: data buffer; *nbytes*: number of bytes to try to read; *flags*: typically use 0; *destaddr*: IP addr and port of destination socket; *addrlen*: length of address structure == sizeof(struct sockaddr\_in)]  
  returns number of bytes written or -1. Also sets *errno* on failure.
- *int listen* (*int sockfd*, *int backlog*);  
  [*sockfd*: socket file descriptor; *backlog*: bound on length of accepted connection queue]  
  returns 0 on success, -1 and sets *errno* on failure.
- *int recvfrom* (*int sockfd*, *char\* buf*, *size\_t nbytes*, *int flags*, *struct sockaddr\* srcaddr*, *int\* addrlen*);  
  [*sockfd*: socket file descriptor; *buf*: data buffer; *nbytes*: number of bytes to try to read; *flags*: typically use 0; *destaddr*: IP addr and port of destination socket; *addrlen*: length of structure == sizeof(struct sockaddr\_in)]  
  returns number of bytes read or -1, also sets *errno* on failure.
- *int connect* (*int sockfd*, *struct sockaddr\* servaddr*, *int addrlen*);  
  [*sockfd*: socket file descriptor; *servaddr*: IP addr and port of the server; *addrlen*: length of structure == sizeof(struct sockaddr\_in)]  
  returns 0 on success, -1 and sets *errno* on failure.
- *int close* (*int sockfd*);  
  returns 0 on success, -1 and sets *errno* on failure.
- *int accept* (*int sockfd*, *struct sockaddr\* cliaddr*, *int\* addrlen*);  
  [*sockfd*: socket file descriptor; *cliaddr*: IP addr and port of the client; *addrlen*: length of structure == sizeof(struct sockaddr\_in)]  
  returns file descriptor or -1 sets *errno* on failure
- *int shutdown* (*int sockfd*, *int howto*);  
  returns 0 on success, -1 and sets *errno* on failure.
- *int write* (*int sockfd*, *char\* buf*, *size\_t nbytes*);  
  [*sockfd*: socket file descriptor; *buf*: data buffer; *nbytes*: number of bytes to try to write]  
  returns number of bytes written or -1.
- *int read* (*int sockfd*, *char\* buf*, *size\_t nbytes*);  
  [*sockfd*: socket file descriptor; *buf*: data buffer; *nbytes*: number of bytes to try to read]  
  returns number of bytes read or -1.
- *int select* (*int maxfdp1*, *fd\_set \*readfds*, *fd\_set \*writefds*, *fd\_set \*exceptfds*, *struct timeval \*tvptr*);
- Convert multi-byte integer types from host byte order to network byte order:
  - *uint32\_t htonl* (*uint32\_t hostlong*): host to network short
  - *uint16\_t htons* (*uint16\_t hostshort*): host to network long
  - *uint32\_t ntohl* (*uint32\_t netlong*): network to host short
  - *uint16\_t ntohs* (*uint16\_t netshort*): network to host long