CS118 Midterm

Alexander Chen

TOTAL POINTS

72 / 100

QUESTION 1

1 Problem 1 - 1,2,3 6 / 6

✓ **- 0 pts Correct**

- 2 pts 1 wrong

- 2 pts 2 wrong

- 2 pts 3 wrong

- 1 pts 1 partial

- 1 pts 2 partial

- 1 pts 3 partial

QUESTION 2

2 Problem 1 - 4,5,6 6 / 6

✓ **- 0 pts Correct**

- 2 pts 4 B

- 2 pts 5 C

- 2 pts 6 A

- 1 pts 4 partial correct

- 1 pts 5 partial correct

- 1 pts 6 partial correct

QUESTION 3

3 Problem 1 - 7,8,9 6 / 6

✓ **- 0 pts Correct**

- 2 pts 7 A

- 2 pts 8 D

- 2 pts 9 B

- 1 pts 7 partial correct

- 1 pts 8 partial correct

- 1 pts 9 partial correct

QUESTION 4

4 Problem 2 13 / 15

- 2 pts Answer other than caching or disable recursive query

- 1.5 pts Did not mention either Cookie OR Privacy/change of browser

- 1.5 pts Missing figure OR wrong explanation

- 1.5 pts Missing about chunks OR Matching streams

- 3 pts wrong answer

✓ **- 2 pts wrong explanation**

- 1 pts partial correct

- 3 pts wrong answer

- 0 pts all correct

- 15 pts All wrong

- 3 pts wrong answer

QUESTION 5

5 Problem 3 6 / 8

✓ **- 2 pts Answer: gethostname(hostn, sizeof(hostn)) == 0**

- 1 pts Answer: gethostbyname(hostn)

- 2 pts Answer: sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1

- 2 pts Answer: connect(sockfd, (struct sockaddr *) &their_addr, sizeof(struct sockaddr)) == -1

- 1 pts Answer: recv(sockfd, buf, MAXDATASIZE-1, 0)) == -1

- 0 pts all correct

- 1 pts Partial credit

- 1 pts partial credit

- 1 pts partial credit

- 8 pts all wrong

QUESTION 6

6 Problem 4 - 1 4 / 6

+ 2 pts Partially Correct

✓ **+ 4 pts Partially Correct**

+ 6 pts Correct

+ 0 pts Incorrect

QUESTION 7

7 Problem 4 - 2 2 / 8

✓ + 2 pts **Partially Correct**

+ 6 pts Partially Correct

+ 8 pts Correct

+ 0 pts Incorrect

QUESTION 8

8 Problem 5 12 / 14

- 1 pts 5.1a: student answers No

- 1 pts 5.1b: student answers No

- 1 pts 5.1c: partial credit

- 1 pts 5.1d: partial credit

- 2 pts 5.2a: wrong answer

✓ - 2 pts **5.2b: wrong answer**

- 1 pts 5.3a: wrong answer

- 1 pts 5.3b: wrong answer

- 1 pts 5.3c: After sending FIN, the sender cannot send. No need to wait for receiving ACK for FIN.

- 1 pts 5.3d: wrong answer

- 0 pts all correct

- 1 pts partia credit of one of question

- 1 pts partial credit

- 1 pts partial correct

- 2 pts wrong answer of one of sub-question, e.g. 5.1c or 5.1d

- 2 pts wrong

- 14 pts all wrong

QUESTION 9

9 Problem 6 - 1 3 / 4

+ 1 pts Partially Correct

+ 2 pts Partially Correct

✓ + 3 pts **Partially Correct**

+ 4 pts Correct

+ 0 pts Incorrect

QUESTION 10

10 Problem 6 - 2 4 / 6

+ 2 pts Partially Correct

✓ + 4 pts **Partially Correct**

+ 6 pts Correct

+ 0 pts incorrect

QUESTION 11

11 Problem 7 - 1 (abc) 3 / 7

- 1 pts Partially correct

- 2 pts Partially correct

- 3 pts Partially correct

✓ - 4 pts **Partially correct**

- 5 pts Partially correct

- 6 pts Partially correct

- 7 pts Incorrect

- 0 pts Correct (CA, cwnd 4, ss 4, seg 9, seg 10, CA, FR/FR)

QUESTION 12

12 Problem 7 - 1 (cd) 4 / 8

- 1 pts Partially Correct

- 2 pts Partially Correct

- 3 pts Partially Correct

✓ - 4 pts **Partially Correct**

- 5 pts Partially Correct

- 6 pts Partially Correct

- 7 pts Partially Correct

- 8 pts Incorrect

- 0 pts Correct (Seg 7, 11; cwnd 5, sst 2; seg 12, 13; SS, cwnd 3)

QUESTION 13

13 Problem 7 - 2 3 / 6

+ 1 pts Partially Correct

+ 2 pts Partially Correct

✓ + 3 pts **Partially Correct**

+ 4 pts Partially Correct

+ 5 pts Partially Correct

+ 6 pts Correct (cwnd 1, sst 2; timeout/SS; cwnd 1, sst 1; SS -> CA/timeout)

+ 0 pts Incorrect

CS118 Midterm Exam, Spring 2018

Name: Alexander Chen

Student ID: 404 837 697

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 computer.
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.
5. If you wish to be considered for partial credit, show all your work.
6. Make sure that you have 11 pages (including this cover page, a blank page for drawing diagram, and one-page Appendix) before you begin.

PROBLEM	MAX SCORE	YOUR SCORE
1	18	
2	15	
3	8	
4	14	
5	14	
6	10	
7	21	
TOTAL	100	

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

Problem 1: Multiple choices (18 points; 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 name 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. CDN: Content distribution network. TLD: Top level domain.

1. You are asked to implement reliable data transfer using UDP. Which of the following mechanisms are NOT necessary for your implementation?

   - Your answer D (A) Retransmissions upon timeout; (B) Error detection; (C) Sequence number and Acknowledgment number; (D) Negative acknowledgment for corrupted data.

2. Which of the following protocol is NOT using the client-server model?

   - Your answer B (A) HTTP; (B) TCP; (C) DNS; (D) SMTP.

3. Which protocol is NOT used when Bob uses his laptop to access emails at his Gmail account?

   - Your answer A (A) SMTP; (B) HTTP; (C) TCP; (D) DNS.

4. What is the value for the "Header Length" field in the TCP header if this TCP segment includes a 16-byte Options field?

   - Your answer B (A) 4; (B) 9; (C) 16; (D) 36.

5. Which is a shared feature by both packet switching and circuit switching?

   - Your answer C (A) statistical multiplexing; (B) resource reservation is needed; (C) used for sharing a network infrastructure; (D) congestion control is needed.

6. If iterative query mode is used among remote DNS servers, which of the following servers can be visited twice or more in a single DNS query?

   - Your answer A (A) local DNS server; (B) root DNS server; (C) top-level domain DNS server; (D) authoritative DNS server.

7. Which layers in the protocol stack are implemented at the SMTP server?

   - Your answer A (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.

8. Which mechanisms do NOT help BitTorrent to better scale to a large number of users?

   - Your answer D (A) Each file is divided into multiple chunks; (B) A peer can upload chunks while downloading other chunks; (C) the optimistically unchoked mechanism; (D) A peer completing file download can leave at any time.

9. What changes are needed to redirect the user's request to a CDN cache server, rather than to the original content server?

   - Your answer B (A) The client's local DNS server; (B) the record at the authoritative DNS server for the content server domain; (C) the DNS server for the CDN cache server; (D) The root DNS server.

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

1. Identify at least one main mechanism that helps the global DNS system to reduce the traffic volume (i.e., the number of DNS queries) at its root DNS servers.

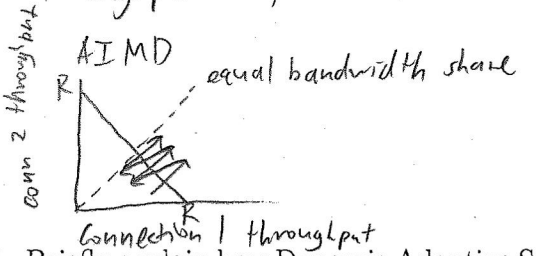
One mechanism is local DNS servers that cache IP addresses so fewer queries need to be made to root.

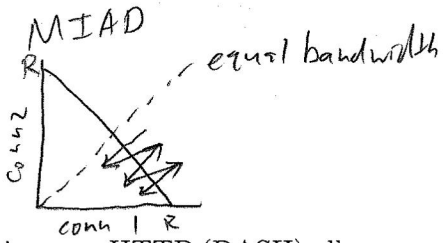
2. What mechanism is leveraged to simplify the Internet shopping experience for a user? Can you also identify a limitation of this mechanism for the user?

One mechanism is cookies which allows shopping websites to remember where you left off. One limitation is that cookies can sometimes give these websites your personal information such as name or email.

3. Consider two TCP connections sharing a single link, with identical round-trip-times and segment size. Someone claims that the multiplicative-increase, additive-decrease (MIAD) mode can outperform the well-known AIMD rule used by TCP. Assume the multiplicative factor is 80% (i.e., it uses 0.8 × window size as the new window size upon loss). Explain whether this change will result in unfair share or not. You can use a figure in your justification.

MIAD will result in an unfair share. As it leads the throughput away from the equal bandwidth line.





4. Briefly explain how Dynamic Adaptive Streaming over HTTP (DASH) allows users with different Internet access speed to stream in video with different qualifies.

The server divides the video into chunks and stores these chunks at different quality levels. Based off of the users bandwidth, the server will send different quality chunks. This allows people with slower internet to receive lower-quality but faster chunks.

5. Identify one scenario where the client-server model with CDN-based caching might outperform the peer-to-peer model, even for a large number of users.

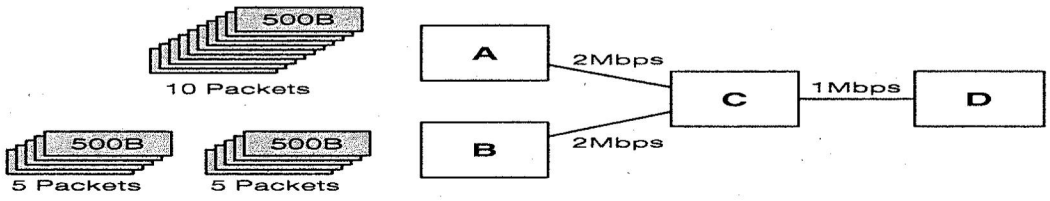
If a user's upload speed is very slow, the client-server model with CDN would be better. This is because in P2P, you match with peers with similar upload speed, so download from those peers could be slow. With CDN download speed is fast and independent of upload speed.

Problem 3 (8 points): Joe writes programs with a client and a server using stream sockets. Can you help Joe to fill in his missing code at the client side? You can use the Appendix for references.

```
#define PORT 1025 /*This is the client port for the connection*/
#define MAXDATASIZE 500 // max number of bytes we can get at once

int main(int argc, char *argv[])
{
int sockfd, numbytes;
char buf[MAXDATASIZE];
struct hostent *he;
struct sockaddr_in their_addr; // connector.s address information

char hostn[400]; //placeholder for the hostname
char ipadd[400]; //placeholder for my IP address

struct hostent *hostIP; //placeholder for the IP address

/*Assign IP address*/
if(hostn[0] != '\0')
          { hostIP = gethostbyname(hostn); }
else
          { printf("ERROR:FC4539 - IP Address not found."); }
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
    {   perror("socket");
exit(1);   }

their_addr.sin_family = AF_INET; // host byte order
their_addr.sin_port = htons(PORT); // short, network byte order
their_addr.sin_addr = *((struct in_addr *)he->h_addr);
memset(&(their_addr.sin_zero), 0, 8); // zero the rest
if (connect(sockfd, (struct sockaddr *) &their_addr, sizeof(their_addr)) < 0)
    { perror("connect");
exit(1); }
numbytes = recvfrom(sockfd, buf, MAXDATASIZE, 0, (struct sockaddr*)hostIP,
                                                    sizeof(hostIP));
if (numbytes < 0)
    { perror("recv");
exit(1);   }

close(sockfd);
return 0;
}
```

Problem 4 (14 points): Consider the topology shown below. Node A has 10 back-to-back packets to send to Node D via Node C, and starts its transmissions at time t=0. Node B has two batches of packets to send to Node D via Node C, with each batch having 5 back-to-back packets. Node B starts its first batch transmission at time t=1 milliseconds (0.001seconds), and starts its second batch at t=2 millisecond.

Assume each packet is 500 bytes. The propagation delay of Link A-C, Link B-C, link C-D is 10 milliseconds each. The bandwidth for Link A-C and Link B-C is 2Mbps ($2x10^6$ bits per second) each, and link B-C's bandwidth is 1Mbps. Node C and Node D can each buffer 20 packets.



1. When does the last packet arrive at Node D?

$t=0$ send 10 packets from A

$$5000 \text{ bytes} \cdot \frac{1 \text{ MB}}{1{,}000{,}000 \text{ bytes}} \cdot \frac{8 \text{Mb}}{1 \text{MB}} \cdot \frac{1 \text{s}}{2 \text{Mb}} = 0.02 \text{s}$$

$t=0.001$ send 5 packets from B

$$2500 \text{ bytes} \cdot \frac{1 \text{MB}}{1{,}000{,}000 \text{ bytes}} \cdot \frac{8 \text{Mb}}{1 \text{MB}} \cdot \frac{1 \text{s}}{2 \text{Mb}} = 0.01 \text{s}$$

1st packet arrives at Node C at 0.002 s

last packet from Node B arrives at C at $t = 0.021$

C–D is bottleneck $20 \cdot 500 \text{ bytes} \cdot \frac{8 \text{Mb}}{1{,}000{,}000} \cdot \frac{1 \text{s}}{1 \text{Mb}} = 0.08$ ← time to send 20 packets

$0.08 + 0.002 = \boxed{0.082 \text{ s}}$

2. What is the *average* queueing delay for all packets (coming from both A and B) at Node C?

$$\text{queue} = \frac{\text{time}}{\#\text{packets}} = \frac{0.08}{20} = \boxed{0.004 \text{s}}$$

↓

real answer greater than this

$$\text{queue} = \frac{\text{total wait time}}{\#\text{packets}} = \frac{\sum \text{send time} - \text{arrival time}}{20}$$

Problem 5 (14 points): Transport-layer Protocols: UDP and TCP

1. (6 points) (Multiplexing and demultiplexing) Two clients from Host A and Host B are sending requests to the server Host S at port 8080, which is hosting video streaming services for clients. The client from Host A is using port 10000, and the client from Host B is using port 20000.

   (a) (1 point) If the client uses UDP, will the server S differentiate whether the request is from client A or B? Justify your answer.

   Yes, the UDP header has a source port # field

   (b) (1 point) If the client uses TCP, will the server S differentiate whether a request is from client A or B? Justify your answer.

   Yes, the TCP header has a source port # field

   (c) (2 points) Consider the response from server S. If UDP is used, can server S respond to the corresponding request that originally came from client A or B? If so, how? If not so, why?

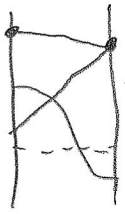
   Yes, server S would send a packet to the IP address and port # found in the IP header and UDP header respectively.

   (d) (2 points) Consider the response from server S. If TCP is used, can server S respond to the corresponding request that originally came from client A or B? If so, how? If not so, why?

   Yes, server S would send a packet to the IP address and port # found in the IP header and TCP header respectively.

2. (4 points) Consider the case of opening the UDP/TCP session for data transfer.

   (a) (2 points) Someone claims that (s)he can use two-way (request SYN from the client, followed by ACK from the server), rather than 3-way handshake, to open the session for video transfer. Can you identify a problem with this choice?

   2-way handshakes are unreliable they can very easily lead to a failed connection if the SYN and ACK packets do not arrive at the perfect time.

   (b) (2 points) If TCP is used, can its sequence number and acknowledgment number be the same after the connection setup phase?

   It cannot be as the same sequence and acknowledgment numbers at the beginning and after the setup phase implies that no data was sent over that period.

3. (4 points) Assume that the server and the client are using TCP. Consider the case of closing the TCP connection.

   (a) After the client receives ACK for its FIN, can the client still send new requests to server S? Briefly explain why.

   It cannot because sending FIN signifies no more requests will be sent. The server can still send responses, but the client cannot send requests.

(b) After the client receives ACK for its FIN, can the client continue to receive data segments from the server S? Briefly explain why.

Yes, it is possible that some requests sent before FIN arrived later, so the server must respond to these requests.

(c) When will server S be unable to send data segments any more?

Server S will no longer be able to send data segments after it sends FIN to the client.

(d) Will the client immediately terminate all its operations after sending ACK upon receiving FIN from the server?

No, it will wait for some time in case a data segment sent by the server before FIN arrives late.

Problem 6 (10 points): TCP Reliable Data Transfer In the following scenario, discuss how TCP reliable data transfer reacts:

- (2 points) Assume the TCP sender window size is 8 segments, when the new received ACK has the "Receive window" field being 1 segment. If retransmission timeout is triggered, how does the TCP sender react?

  the TCP sender will send the packet that was timed out

- (2 points) Identify one advantage if TCP uses cumulative ACK.

  One advantage is that it allows packets to be received out of order.

- (3 points) What is the negative impact of premature timeout? Explain how TCP avoids premature timeout in its design.

  Premature timeout can cause greater congestion because packets are unnecessarily sent. TCP avoids premature timeout by having much longer timeouts and relying more on the receipt of a triple duplicate ACK.

- (3 points) If TCP receives three corrupted ACKs consecutively (i.e., back to back), how does the TCP react in its reliable data transfer operations?

  It resends the packet that has the sequence number equal to that ACK number.

Problem 7 (21 points): TCP Congestion Control Consider a 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. Right after fast retransmit/fast recovery phase, if $ssthresh = cwnd$, use the slow start algorithm. Show your diagram to receive partial credit; you can draw your diagram on **Page 10** only.

- TCP uses reliable transfer. The receiver acknowledges every segment, and the sender always has data available for transmissions. The used ACK number for each segment is based on cumulative ACK and indicates the next expected segment.
- Initially, *ssthresh* is set to 4, and *cwnd* to 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 timer is set on a per window basis, and retransmission timeout (RTO) value is initially set to 500ms and remains unchanged. The RTT is 100ms for all transmissions. We assume that the receiver's advertised window size is always larger than 10 segments.
- TCP starts from the initial sequence number of 1 at t=0. Segments with sequence number 5 and 6 arrived at the receiver out of order (i.e., segment 6 arrives before segment 5).
- The sender uses the following equation to update cwnd during congestion avoidance (where MSS is the TCP segment size and $\lfloor \cdot \rfloor$ is the round-down function):

$$cwnd+ = MSS * MSS / \lfloor cwnd \rfloor$$

1. (15 points) Segment with sequence number 7 is lost once. No other loss is observed.

   (a) (3 points) The receiver sends an ACK upon receiving segment with sequence number 4.

   i. What congestion control algorithm(s) should the sender use when receiving this ACK?

   slow start

   ii. What are the updated values for *cwnd* and *ssthresh* upon this?

   cwnd = 5

   ssthresh = 4

   (b) (3 points) The receiver sends an ACK upon receiving segment with sequence number 5.

   i. What segments should the sender send out when receiving this ACK?

   11

   ii. What congestion control algorithm(s) will be used by the sender?

   Congestion avoidance

   (c) (4 points) The receiver sends ACK upon receiving the segment with sequence number 10.

   i. What congestion control algorithm(s) does the sender invoke upon receiving this ACK?

   fast retransmit

ii. What segment(s) will be transmitted after receiving this ACK?

7

iii. What are the updated values for *cwnd* and *ssthresh*?

cwnd = 5

ssthresh = 2

(d) (5 points) Assume the sender now receives the ACK for the retransmitted segment 7.

i. What segment(s) will the sender send out upon receiving this ACK?

13

ii. What congestion control algorithm(s) will be invoked by the sender?

fast recovery

iii. What is the updated value for *cwnd* after receiving this ACK?

2

2. (6 points) Now assume that two segments with sequence number 7 and 8 are lost once each.

(a) (2 points) How are the updated values for *cwnd* and *ssthresh* when the sender retransmits the lost segment 7?

cwnd = 5

ssthresh = 2

(b) (1 point) What congestion control algorithm(s) will be used when the sender retransmits the lost segment 7?

fast retransmit

(c) (2 points) How are the updated values for *cwnd* and *ssthresh* when the sender retransmits the lost segment 8?

cwnd = 5

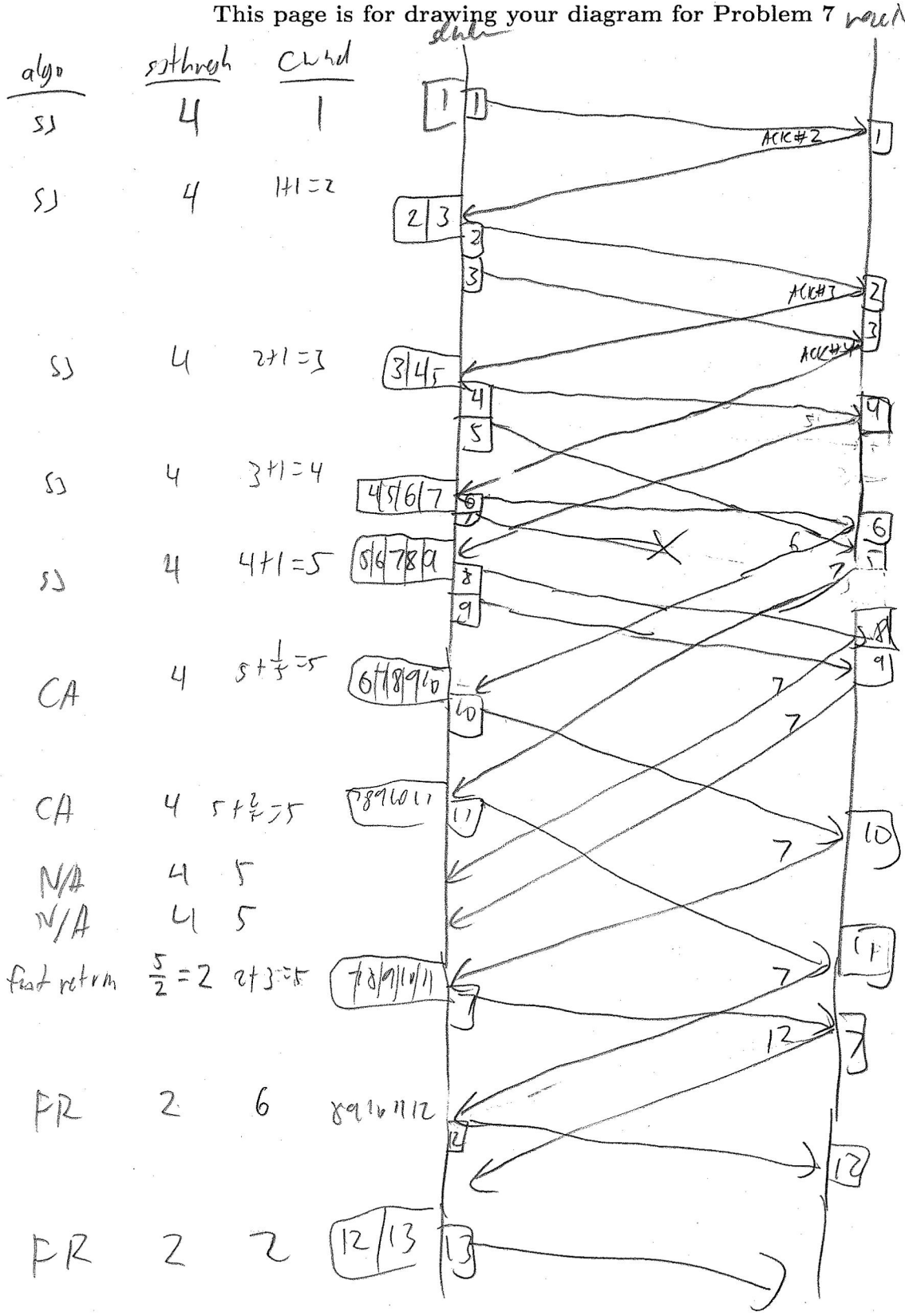
ssthresh = 2

(d) (1 point) What congestion control algorithm(s) will be used when the sender retransmits the lost segment 8?

fast retransmit

This page is for drawing your diagram for Problem 7

algo	ssthresh	cwnd
SS	4	1
SS	4	1+1=2
SS	4	2+1=3
SS	4	3+1=4
SS	4	4+1=5
CA	4	$5+\frac{1}{5}=5$
CA	4	$5+\frac{2}{5}=5$
N/A	4	5
N/A	4	5
fast retrm	$\frac{5}{2}=2$	2+3=5
FR	2	6
FR	2	2



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 socketaddr* 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 address 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 address 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 address 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);*
  *FD_ZERO (fd_set *fdset);*
  *FD_SET (int fd, fd_set *fdset);*
  *FD_ISSET (int fd, fd_set *fdset);*
  *FD_CLR (int fd, fd_set *fdset);*