Name_______________ Student ID____________________ Department/Year___________

# 2nd Examination

Introduction to Computer Networks (Online)
Class#: EE 4020, Class-ID: 901E31110
Spring 2022

10:20-12:10 Wednesday
April 27, 2022

### Cautions

1. There are in total 100 points to earn. You have 90 minutes to answer the questions. Skim through all questions and start from the questions you are more confident with.
2. Use only English to answer the questions. Misspelling and grammar errors will be tolerated, but you want to make sure with these errors your answers will still make sense.

1. (ch26, 3pt) Which of the following are jobs of a DASH video client?
   (a) Measuring delay
   (b) Measuring bandwidth
   (c) Encoding videos into different resolutions
   (d) Dividing videos into chunks
   (e) Reading URLs for videos chunks encoded in different resolutions
   (f) Buffering video

Sample Solution:

(b)(e)(f) or (a)(b)(e)(f)

(Grading policy: -1pt per wrong pick till 0pt)

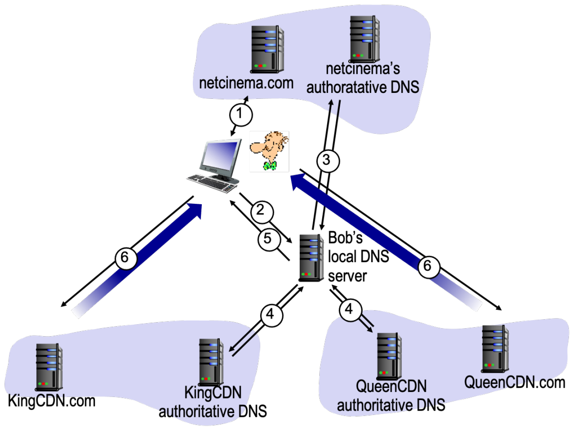
2. (ch26, 6pt) Recall these two terms buffer starvation and buffer overflow.
   (1) What is buffer starvation? (1pt)
   (2) Is buffer starvation good or bad? How to enforce buffer starvation if it's good? How to prevent buffer starvation if it's bad? (2pt)
   (3) What is buffer overflow? (1pt)
   (4) Is buffer overflow good or bad? How to enforce buffer overflow if it's good? How to prevent buffer overflow if it's bad? (2pt)

Sample Solution:

In ch26, these terms mean the client-side buffer starving and overflowing.

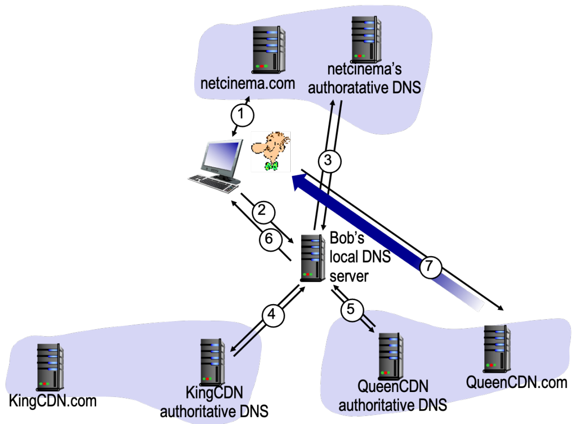
Describe the terms in your own words and justify for your view.

3. (ch26, 6pt) The benefit of implementing CDN via DNS redirection is that a content provider (netcinema.com) can rotate and redirect to two CDN providers (KingCDN and QueenCDN) as illustrated in Model I below .

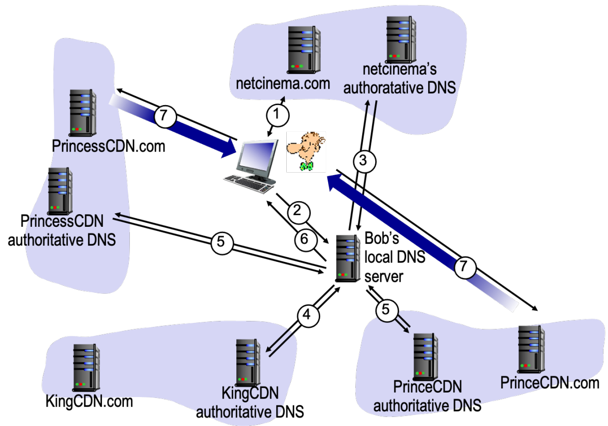



(1) Model I

Consider 2 more models of DNS redirection, i.e., Model II and Model III. In Model II, netcinema only deals with KingCDN and KingCDN can offload the work to another CDN, i.e., QueenCDN. In Model III, KingCDN can further offload the work to two other CDN providers, i.e., PrinceCDN and PrincessCDN. Discuss the benefits of implementing (1) Model I, (2) Model II, and (3) Model III.




(2) Model II




(3) Model III

Sample Solution:

Whatever makes sense. Below are some possibilities.

(1) More tolerant to faults/inaccessibility to any of the CDN providers. Lighter load to each of the 2 CDNs.

(2) Communicating/negotiating with only 1 CDN provider.

(3) Communicating/negotiating with only 1 CDN provider while maintaining fault tolerance, having the sole CDN provider offloading redirections flexibly to multiple sub-CDN providers.

4. (ch34, 4pt) Recall rdt 1.0, rdt 2.2, and rdt 3.0. The assumption of the underlying channel

is relaxed from being error and loss free, only loss free, to error or loss possible.

(1) How does rdt 2.2 detect bit errors in a data packet? (1pt)

(2) What does rdt 2.2 do after detecting bit errors in a data packet? (1pt)

(3) How does rdt 3.0 detect a packet loss? (1pt)

(4) What does rdt 3.0 do after detecting a packet loss? (1pt)

Sample Solution:

Whatever makes sense. Below are some possibilities.

(1) The sender computes and includes the Internet checksum in the packet

(2) The receiver sends ACK when no bit error is detected and sends NAK when a bit error is detected.

(3) The sender waits for a timeout interval.

(4) The sender retransmit the packet when a retransmission timeout is fired.

5. (ch34, 4pt) Recall rdt 2.0, rdt 2.1, and rdt 2.2.
   (1) What does rdt 2.1 do more than rdt 2.0? (1pt)
   (2) What is the purpose of the additional mechanism in rdt 2.1 (over rdt 2.0)? (1pt)
   (3) How is rdt 2.2 different from rdt 2.1? (1pt)
   (4) What is the purpose of the additional mechanism in rdt 2.2 (over rdt 2.1)? (1pt)

Sample Solution:

Whatever makes sense. Below are some possibilities.

(1) In rdt 2.1, the receiver adds checksum to the ACK/NAK packets. The sender retransmits data packets when ACK/NACK packets are corrupted. Packets are given sequence # 0 and 1 to tell duplicates apart.

(2) rdt 2.1 considers possibility of bit errors in ACK/NAK packets, because bit errors do not discriminate ACK/NACK against data packets.

(3) In rdt 2.2, there is no NAK packets and ACK packets carry a sequency number indicating the sequence number of the next data packet the receiver is expecting. The sender determines whether the ACK packet is positive or negative (NAK) by checking the sequence number in the ACK packet. If it’s a duplicate/old sequence number, it’s a NAK. Otherwise, an ACK.

(4) Only 1 packet type to define. Simpler FSM and potentially cleaner implementation.

6. (ch34, 3pt) Let’s consider a variation of rdt 3.0 and name it rdt 3.1. In that, the sender retransmits the data packet when the ACK packet sequence number is not expected or

the ACK packet is corrupted. This will avoid waiting for a timeout interval and retransmit faster.

(1) Think of a disadvantage of using rdt 3.1. (1pt)

(2) Give a case that illustrates the disadvantage. (2pt)

Sample Solution:

(1) higher bandwidth consumption
(2) A case where there is an unnecessarily retransmission due to premature timeout

7. (ch34, 4pt) Compare and contrast the 3 pipelined rdt protocols, GBN, SR and TCP (i.e., TCP's rdt.
   (1) Which method(s) tend to use more network bandwidth to retransmit for losses and why? (1pt)
   (2) Which method(s) tend to require more memory space at the receiver end and why? (1pt)
   (3) Which method(s) tend to require more timers at the sender end and why? (1pt)
   (4) If you are to program a pipelined rdt, which method would you choose to implement and why? (1pt)

Sample Solution:

(1) GBN or GBN&TCP. GBN/TCP(simplified) retransmits a whole flight of packets.

(2) SR or SR&TCP. Both store out of order arrivals.

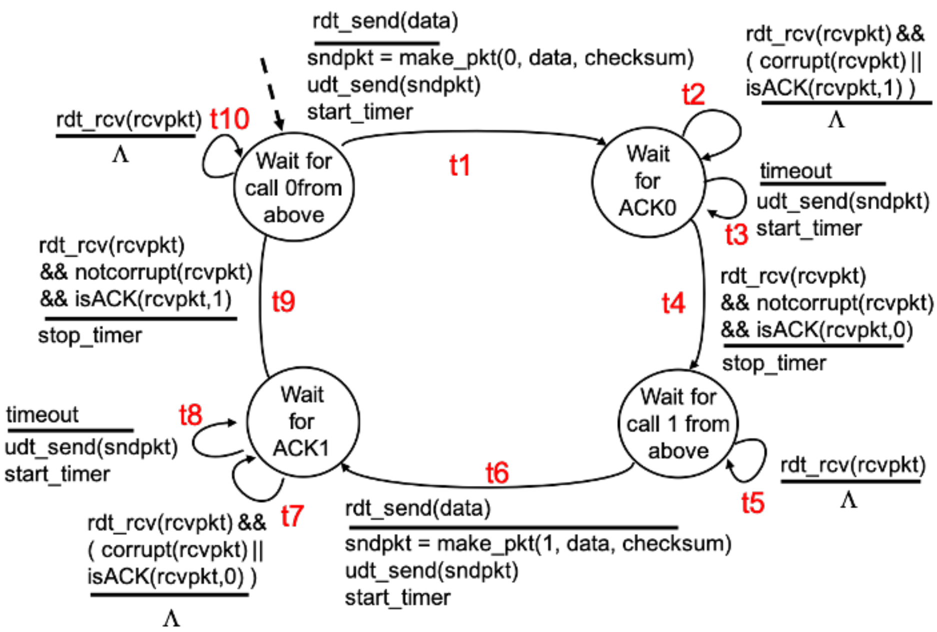
(3) SR. One timer per packet in flight.

(4) Take your pick and justify.

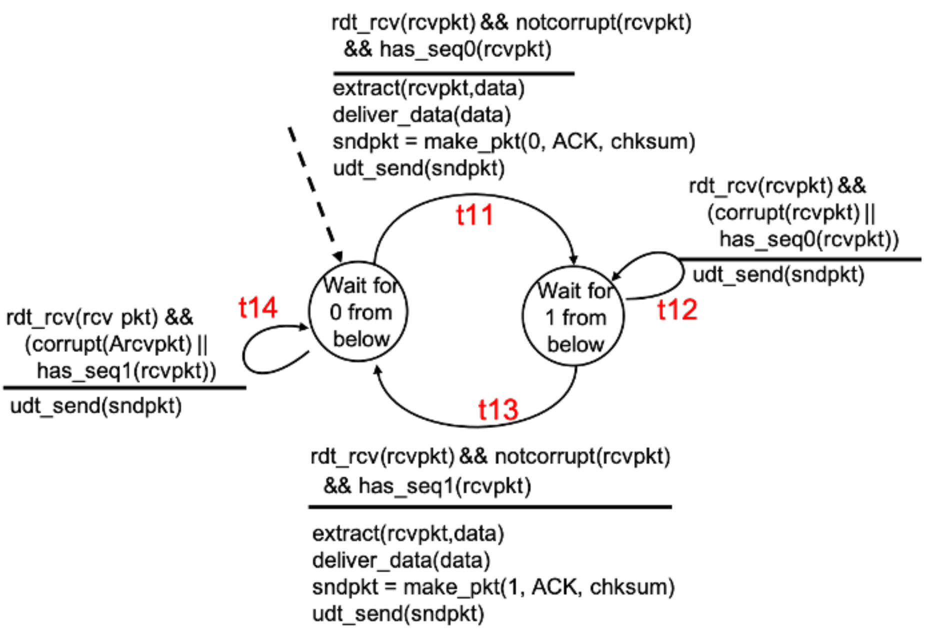
8. (ch34, 2pt) Below is the FSM of rdt 3.0. Tell the sequence of transitions for the following

two scenarios.

<u>rdt 3.0 sender</u>




<u>rdt 3.0 receiver</u>




(1) Start from the initial state. The sender sends two data packets. No bit errors or packet losses except that the first data packet is lost in the first trial. (1pt)

(2) Start from the initial state. The sender sends two data packets. No bit errors or packet losses except that the second data packet is lost in the first trial. (1pt)

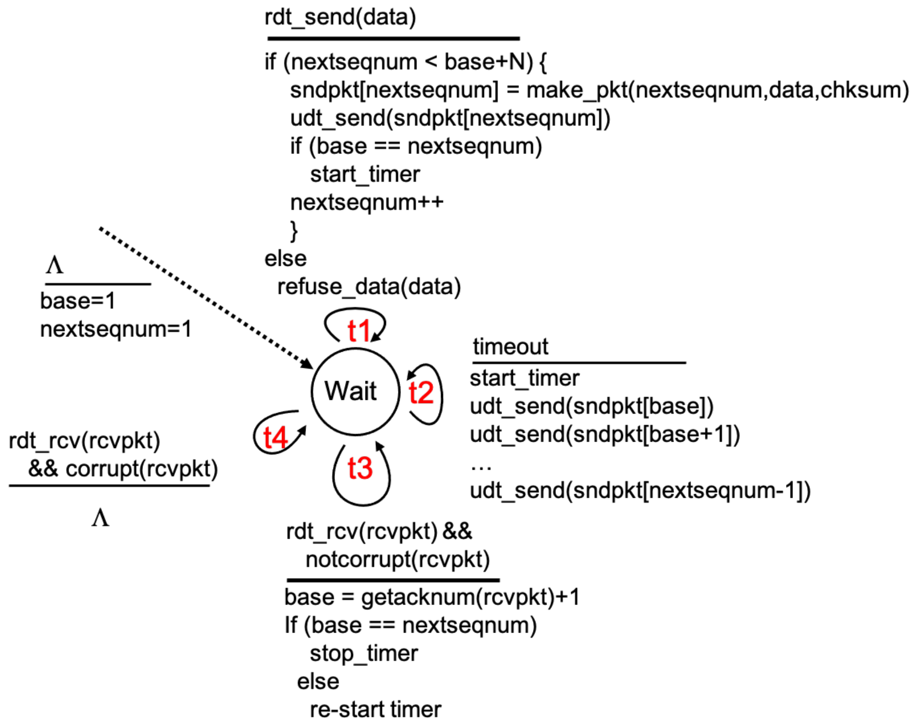
Sample Solution:

(1) t1, t3, t11, t4, t6, t13, t9

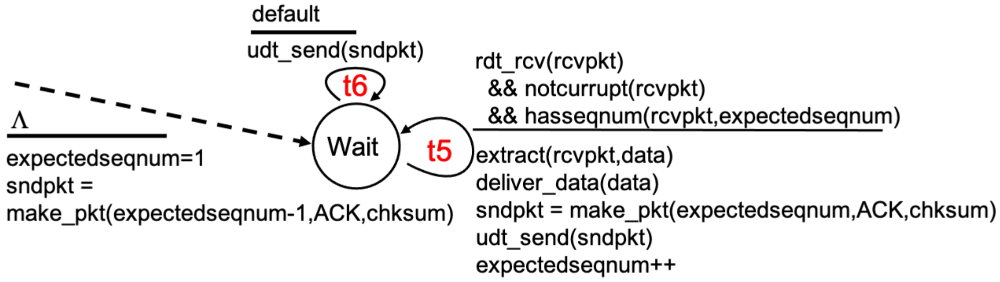
(2) t1, t11, t4, t6, t8, t13, t9

9. (ch34, 2pt) Below is the FSM of GBN. Let N = 10. Tell the sequence of transitions for the following two scenarios.

GBN sender




GBN receiver




(1) The sender sends two data packets. No bit errors or packet losses except that the first data packet is lost in the first trial. (1pt)

(2) The sender sends two data packets. No bit errors or packet losses except that the second data packet is lost in the first trial. (1pt)

Sample Solution:

(1) t1, t1, t6, t3, t2, t5, t5, t3, t3

(2) t1, t1, t5, t3, t2, t5, t3

10. (ch34, 2pt) Below is the FSM of SR. Let N = 10. Tell the sequence of transitions for the following two scenarios.

<u>SR sender</u>

**t1: data from above:**

- if next available seq # in window, send pkt

**t2: timeout(n):**

- resend pkt n, restart timer

**t3: ACK(n)** in [sendbase,sendbase+N-1]:

- mark pkt n as received
- if n is the smallest unACKed pkt, advance window base to next unACKed seq #

<u>SR receiver</u>

**t4: pkt n in** [rcvbase, rcvbase+N-1]

- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

**t5: pkt n in** [rcvbase-N,rcvbase-1]

- ACK(n)

**t6: otherwise:**

- ignore

(1) The sender sends two data packets. No bit errors or packet losses except that the first data packet is lost in the first trial. (1pt)

(2) The sender sends two data packets. No bit errors or packet losses except that the second data packet is lost in the first trial. (1pt)

Sample Solution:

(1) t1, t1, t4, t3, t2, t4, t3

(2) t1, t1, t4, t3, t2, t4, t3

11. (ch35, 6pt) Recall the algorithm estimating the retransmission timeout – TimeoutInterval.

(1) What is the disadvantage when the TimeoutInterval is set too short? (1pt)

(2) What is the disadvantage when the TimeoutInterval is set too long? (1pt)

(3) Why does the algorithm omit calculating the RTT of a data and ACK packet pair, i.e., getting a sampleRTT, for a data packet that has been retransmitted? (1pt)

(4) Why does the algorithm calculate a smoothed average, i.e., estimatedRTT, instead of using the most recent sampleRTT? (1pt)

(5) What does DevRTT estimate? (1pt)

(6) The algorithm sets the retransmission TimeoutInterval to estimatedRTT+4*DevRTT.

Why 4 times DevRTT, not 3 times or 5 times? (1pt)

Sample Solution:

(1) Premature and unnecessary timeout
(2) Slow in recovering from packet losses
(3) Not sure which data packet the ACK is acknowledging for.
(4) sampleRTT is the instantaneous RTT and it varies significantly from time to time. A smoothed average gives a longer-term, stable estimate.
(5) Variation of instantaneous RTTs to the estimatedRTT
(6) Empirically determined, heuristic, or something more elaborated -- if RTTs over a long time converges to a normal distribution, estimatedRTT (the avg) + 4*DevRTT (the variation) covers a very high % of RTTs (68–95–99.7 rule), therefore reducing the chance of premature timeout to a super low %.

12. (ch35, 6pt) Despite similarities between TCP's rdt and GBN, there are quite a few optimizations in TCP's rdt over GBN. Tell the benefit(s) of implementing these optimizations.
    (1) Sequence number in byte count. (1pt)
    (2) Only new acks refresh the timer. (1pt)
    (3) Retransmit only 1 packet after a timeout event. (1pt)
    (4) Timeout interval is doubled in case of back-to-back timeouts. (1pt)
    (5) Delayed ack. (1pt)
    (6) Fast retransmission. (1pt)

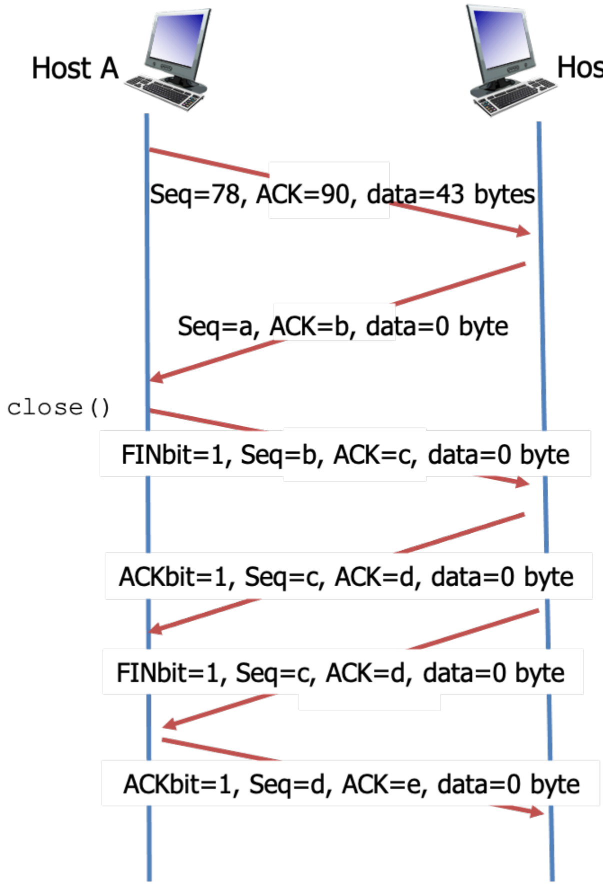
Sample Solution:

(1) flexible packet size and better utilization of the window space
(2) timeout faster in case of many duplicate acks
(3) conservative in transmitting data after a timeout event (severe network congestion)
(4) conservative in transmitting data after multiple timeout events (even more severe)
(5) less acks (acks are cumulative anyway)
(6) retransmit faster in case of having duplicate acks

In general, shorter transmission time, lower bandwidth consumption, conservative in case of congestion, flexibility

13. (ch35, 5pt) Depicted below is a sequence of packet exchange in a TCP connection between Host A and B. Host A sends the last data packet and receives the ACK packet

from Host B. Then, Host A sends a FIN packet to initiate closing of the connection. Host B sends a FIN packet as well. Tell the value of a, b, c, d, e.




Sample Solution:

(a, b, c, d, e) = (90, 121, 90, 122, 91)

14. (ch35, 6pt) Listed below are fields in a TCP's transport layer packet header.

(1) Tell the use of port number
(2) Tell the use of checksum
(3) Tell the use of acknowledgement number
(4) Tell the use of F bit
(5) Tell the use of receive window
(6) Tell the use of E bits

Sample Solution:

(1) to demultiplex
(2) to detect bit error
(3) to inform the data sender to which byte the data are received correctly

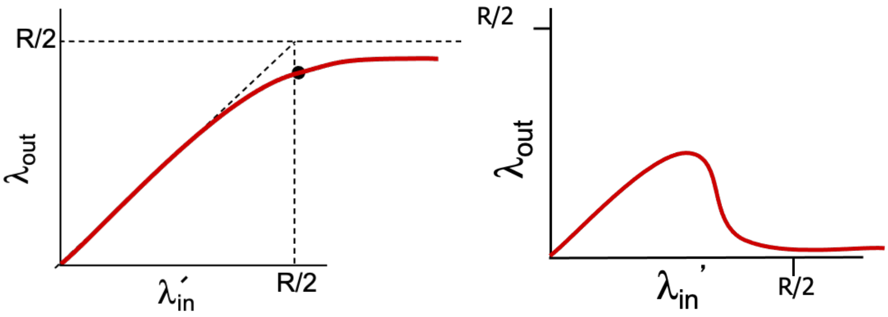
(4) to close the connection

(5) to inform the data sender the remaining amount of buffer space at the receiver

(6) to explicitly inform the data sender that some middle router is running low in buffer space

15. (ch36, 4pt) Recall the lecture about the causes of network congestion in 3 scenarios. The $\lambda_{in}'$ and $\lambda_{out}$ relationship in scenario 2 and 3 are depicted as follows:

Scenario 2 Scenario 3




(1) Tell the cause(s) of $\lambda_{out}$ being lower than R/2 when $\lambda_{in}'$ approaches R/2 in scenario 2. (2pt)

(2) Explain why $\lambda_{out}$ drops lower after $\lambda_{in}'$ increases beyond the tipping point in scenario 3. (2pt)

Sample Solution:

Whatever makes sense. Below are some possibilities.

(1) Retransmission due to packet drops at the router
Unnecessary retransmission due to premature timeout

(2) When $\lambda_{in}'$ is close to R/2 the buffer space builds up. The packets from a stream might survive the 1st router ($\lambda_{out}$ no more than R/2). The surviving packets are likely dropped or queued at the end of the 2nd router as the link utilization is approaching 100%. The resulting $\lambda_{out}$ can be significantly lower than R/2 depending on the state of the queues.

16. (ch37, 5pt) A TCP connection starts with cwnd = 1MSS and transmits packets of constant size 1MSS. Let's also assume the delayed ack is disabled for simplicity. With the initial cwnd=1MSS, the sender sends the 1st data packet (sequence number 0) to the receiver. After receiving the 1st ACK packet (ack sequence number 1MSS) from the receiver, the sender increases the cwnd to 2MSS. The sender can therefore send the 2nd and 3rd data

packet back-to-back. The 2nd and 3rd ACK packet will come back and trigger changes to cwnd subsequently. This will go on and all data and ACK packets will come through smoothly except for the 9th data packet. It is dropped. Listed below are the values of cwnd receiving 12 ACK packets respectively.

(2MSS, 3MSS, 4MSS, 4.25MSS, a, b, c, d, d, d, d, d, …)

(1) What is the value of ssthresh at the beginning of the connection?
(2) Tell the value of a and show your derivation.
(3) Tell the value of b and show your derivation.
(4) Tell the value of c and show your derivation.
(5) Tell the value of d and show your derivation.

Sample Solution:

(1) 4MSS
(2) 4.49MSS (4.25 + 1/4.25)
(3) 4.71MSS (a+1/a = 4.49 + 1/4.49)
(4) 4.92MSS (b+1/b = 4.71 + 1/4.71)
(5) 5.12MSS (c+1/c = 4.92 + 1/4.92)

17. (ch37, 4pt) Continue from Problem Set 16. Answer the questions below assuming the fast recovery mechanism is disabled.

(1) What is the ssthresh after receiving the 11th ACK?
(2) What is the cwnd after receiving the 11th ACK?
(3) What is the sequence number of the data packet(s) transmitted after getting the 11th ACK? Why?
(4) What is the ack sequence number of the 13th ACK packet? Why?

Sample Solution:

(1) 2.56 (d/2)
(2) 2.56 (d/2)
(3) 8MSS. Retransmitting the 9th data packet. The sequence number of its 1st byte is 8MSS.
(4) 13MSS. Acknowledging 9th-13th data packet altogether. The ack sequence number of the ACK packet is 13MSS.

18. (ch37, 3pt) Continue from Problem Set 16. Answer the questions below assuming the fast recovery mechanism is enabled.
    (1) What is the cwnd after receiving the $11^{th}$ ACK?
    (2) What is the cwnd after receiving the $12^{th}$ ACK?
    (3) What is the sequence number of the data packet(s) transmitted after getting the $12^{th}$ ACK? Why?

Sample Solution:

(1) 5.56 (c/2+3MSS)
(2) 6.56 (cwnd+1MSS)
(3) 13MSS. The cwnd allows 6 packets in flight and therefore the sender can send 1 more packet, i.e., the $14^{th}$ data packet.

19. (PA, 13pt) Please go on the PA workstation, create the exam2 subdirectory, and work under the exam2 directory for this problem set.
   (1) Develop exam2-p19-1.go such that it connects and then closes the connection to the servers running on port 11991-11993 sequentially. (2pt)
   (2) Develop exam2-p19-2.go such that it connects, sends “delay\n”, and then closes the connection to the servers running on port 11991-11993 sequentially. (2pt)
   (3) Develop exam2-p19-3.go such that it connects, sends “delay\n”, reads a line of message, prints the message, and then closes the connection to the servers running on port 11991-11993 sequentially. (2pt)
   (4) Develop exam2-p19-4.go such that it extends exam2-p19-3.go, finds the server that gives the lowest delay and prints the port number of the server. (2pt)
   (5) Develop exam2-p19-5.go such that it connects only to the server with the lowest delay (from running exam2-p19-4.go), sends “bandwidth\n”, reads a line of message, prints the message, and then closes the connection to the lowest delay server. (1pt)
   (6) Develop exam2-p19-6.go such that it connects only to the server with the lowest delay, sends “resolutions\n”, reads multiple lines until a line that reads exactly “.\n”, prints all the lines, and then closes the connection to the server. (2pt)
   (7) Develop exam2-p19-7.go such that it connects only to the server with the lowest delay, requests a video (e.g., “1040p\n”) whose bandwidth requirement is just below the bandwidth to the server (known by running your exam2-p19-5.go). (2pt)

Sample Solution:

Whatever that works.

20. (PA, 12pt) Please go on the PA workstation and work under the exam2 directory for this problem set.
    (1) Develop exam2-p20-1.go running on a port that is not occupied such that it allows multiple clients such as your exam2-p19-1.go to access concurrently. (2pt)
    (2) Develop exam2-p20-2.go such that it allows multiple clients such as your exam2-p19-2.go to connect, send “delay\n”, and close concurrently. (2pt)
    (3) Develop exam2-p20-3.go such that it allows multiple clients such as your exam2-p19-3.go to connect, send “delay\n”, receive a short response saying “delay = 10ms\n” and close concurrently. (2pt)
    (4) Develop exam2-p20-4.go such that it allows multiple clients such as your exam2-p19-5.go to connect, send “bandwidth\n”, receive a short response saying “bandwidth = 850kbps\n” and close concurrently. (2pt)
    (5) Develop exam2-p20-5.go such that it allows multiple clients such as your exam2-p19-6.go to connect, send “resolutions\n”, receive multiple lines saying “1080p in 1.4Mbps\n”, “720p in 800kbps\n“, “480p in 300kbps\n”, “.\n” and close concurrently. (2pt)
    (6) Develop exam2-p20-6.go such that it allows multiple clients such as your exam2-p19-7.go to connect, request video chunks of certain resolution by sending “1040p\n” for example, receive a short message saying “incoming!\n” if the video resolution requires a bandwidth less than 850kbps or “try a lower resolution video…\n” if the resolution requires a bandwidth higher than 850kbps. (2pt)

Sample Solution:

Whatever that works.