22W-COM SCI-118-LEC-1 Midterm

[redacted]

TOTAL POINTS

99 / 100

QUESTION 1

25 pts

1.1 10 / 10

✓ **- 0 pts** **Correct**

- 3 pts incorrect ack number

1.2 5 / 5

✓ **- 0 pts** **Correct**

- 5 pts incorrect

- 3 pts 126 or 128 (but basic concept is correct)

1.3 10 / 10

✓ **- 0 pts** **Correct**

- 4 pts part 1: incorrect

- 4 pts part 2: 2pts partial credits **(only applies on reasoning based on fast transmits triggered)**

- 4 pts part 2: still the duplicate ack

- 3 pts part 2: not cumulative ack

- 2 pts part 2: miscalculation

- 6 pts part 2: incorrect

QUESTION 2

25 pts

2.1 4 / 4

✓ **- 0 pts** **Correct**

- 2 pts Incorrect ssthresh (correct answer is 16 or (8,16]). Excluding 23/2.

- 2 pts Improper explanation

- 2 pts Compute 13th round (23/2) with correct explanation.

Note: Second transmission round != second phase.

2.2 8 / 8

✓ **- 0 pts** **Correct.**

[1,5]u[19,22] (+-1 ar the margin is allowed)

- 6 pts Only part of the answer

- 8 pts Incorrect

2.3 8 / 8

✓ **- 0 pts** **Correct.**

[5-12]u[13,18] (+-1 ar the margin is allowed)

- 4 pts Only part of the answer

- 6 pts Only part of the answer

- 8 pts Incorrect

2.4 5 / 5

✓ **- 0 pts** **Correct**

3 duplicated ACK; cwnd is cut by half, not to 1.

- 2 pts Missing or improper explanation

- 5 pts Incorrect

QUESTION 3

25 pts

3.1 5 / 5

✓ **- 0 pts** **Correct**

- 1 pts Incomplete explanation

- 3 pts Incorrect explanation

- 5 pts Incorrect

3.2 5 / 6

✓ **- 0 pts** **Correct**

✓ **- 1 pts** **Incorrect or missing fragment sizes in explanation**

- 2 pts Incomplete explanation

- 3 pts Incorrect explanation

- 6 pts Incorrect

3.3 8 / 8

✓ **- 0 pts** **Correct**

- **2 pts** Incorrect or missing fragment offset
- **1 pts** Incorrect or missing fragment sizes
- **3 pts** Incorrect or missing total number of fragments
- **3 pts** Incorrect explanation
- **8 pts** Incorrect

3.4 6 / 6

✓ - **0 pts** **Correct**
- **6 pts** Incorrect

QUESTION 4

25 pts

4.1 5 / 5

✓ - **0 pts** **Correct**
- **2 pts** Caching Resolver RTT Error.
- **4 pts** Procedure Error.

4.2 4 / 4

✓ - **0 pts** **Correct**
- **2 pts** Caching Resolver RTT Error.
- **4 pts** Error for the checking procedure

4.3 4 / 4

✓ - **0 pts** **Correct**
- **3 pts** Incorrect number

4.4 4 / 4

✓ - **0 pts** **Correct**
- **2 pts** Unclear explanation

4.5 4 / 4

✓ - **0 pts** **Correct**
- **1 pts** Domain owner/operator determine the TTL
- **1 pts** Pros Unclear
- **1 pts** Cons Unclear

4.6 4 / 4

✓ - **0 pts** **Correct**
- **1 pts** Do not mention fast, without connection setup delay
- **3 pts** Do not compare TCP/UDP

CS118
Winter 2022 Midterm Exam

1 hour 50 minutes
Close book and closed notes; NO use of any device except calculators.
Two pages of cheat sheet allowed; write your answer legibly.

- This exam has 5 pages including this cover page. Do all your work on these exam sheets, use the back side if needed.
- Cross out all the scratch work that you do not want to be counted as part of your answer before you submit the exam.
- Show ***all*** your work, including unfinished problems that you wish to be considered for partial credit.
- Be *specific* and *clear* in your answers, and *explain all your answers*.
- When the answer to a problem is not immediately clear, do not simply dump anything and everything that comes to your mind on the paper, without knowing whether it is relevant or irrelevant. Irrelevant answers can lead to point-deduction as they show a lack of understanding to the problem.

Your name: [redacted]

Student ID: [redacted]

	Points	Your score
Problem 1	25	
Problem 2	25	
Problem 3	25	
Problem 4	25	
Total	100	

Problem 1 (25 points) Host A and B are communicating over a TCP connection, and Host B has already received from A all the bytes up through byte 126. Suppose Host A then sends three data segments to Host B continuously. The first, second, and third segment contains 80, 40, and 20 bytes of data, respectively. The first segment header contains sequence number 127, the source port number 302, and the destination port number 80. Host B sends an acknowledgment whenever it receives a data segment from Host A; if a segment arrives out of order, B sends a duplicate acknowledgment.

1.1 (10 points) Assume that all segments and acknowledgments arrive in order, what are the values for acknowledgment number, source port number, and destination port number in the second acknowledgment sent from Host B to A, ?

ACK = 127+80+40 = 247 since received first and second data already

Source port = 80 since that was where the packet was sent to

destination port = 302 since that was where the data packet came from

1.2 (5 points) If the second data segment arrives at B before the first one, in the acknowledgment of this first arrived segment, what is the acknowledgment number? Assume there is no additional out-of-order delivery.

ACK = 127 since it is out of order, thus B returns a duplicate ACK for 127.

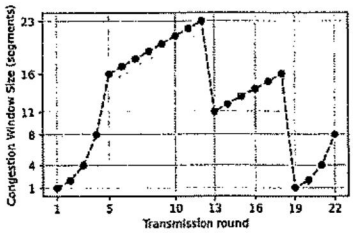
1.3 (10 points) Suppose the first data segment to B is lost, but there is no additional packet loss or out-of-order delivery. The second and third acknowledgment from B to A arrive successfully, but they arrive *after* A's retransmission timer on the first segment expires (thus A retransmitted the first data segment).

- Assuming TCP fast retransmit is enabled, will it be triggered by duplicate acknowledgements in this process?
- What is the acknowledgment number in the last acknowledgement sent from B to A?

No, since TCP fast retransmit needs 3 duplicate ACKs, this would only send two duplicate ACKs for 127.

ACK = 127+80+40+20
= 267 since the "retransmitted first segment will have arrived filling in the gap with the already arrived second and third segments. Everything was eventually received, so acknowledge it all.

Problem 2 (25 points) Assuming the value of a TCP connection's congestion control window, cwnd, changes over time as shown in the figure, where X-axis indicates the number of round trips, and Y-axis shows the value of cwnd as measured by the number of segments. To help refresh your mind about TCP congestion control, the description from the last lecture is attached below: learn from observations



- when cwnd < ssthresh, increase cwnd exponentially
- when cwnd ≥ ssthresh, increase cwnd linearly (to the round trip time)
  - if packet loss is detected: have gone too far: ssthresh = cwnd / 2
  - If the loss detected by 3 dup. ACKs: network capable of delivering some packets, cwnd = cwnd / 2
  - If loss detected by timeout: slow-start again (cwnd = 1 segment)

2.1 (4 points) What is the value of ssthresh at the ~~send~~ second transmission round? Please explain your answer.

ssthresh = 16 since there is no packet loss between 1 and 12 thus ssthresh does not change between them, and after round 5, the cwnd only increases by 1 per round thus cwnd = ssthresh = 16, which thus is the same for the second round.

2.2 (8 points) Identify all the time intervals (measured by RTTs) when TCP operates in slow-start mode.

1 - 5
19 - 22
since the cwnd grows exponentially not linearly, doubling every round instead of incrementing by 1.

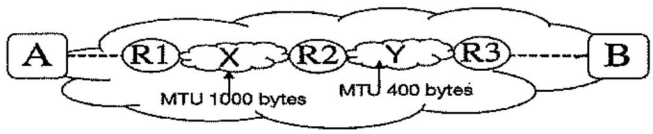
2.3 (8 points) Identify the time intervals when TCP operates in congestion avoidance mode

5 - 18
since during this interval, cwnd increases linearly, incrementing by 1 per round.

2.4 (5 points) There is a segment loss after the 12th round trip, therefore the cwnd value drops. Is this segment loss detected by 3 duplicate ACKs, or by TCP retransmission timeout? Explain your answer.

3 duplicate ACKs because the congestion window was not reset to 1 but was halved. This could only happen if the loss was detected by 3 duplicate ACKs. If instead it was by timeout, the cwnd would be 1 not 11.

Problem 3 (25 points) Assume that IPv4 packets are being transmitted across a large network from A to B. To get from A to B, a packet must first pass through a subnet *X* with an MTU size of 1000 bytes, and then through another subnetwork *Y* with an MTU size of 400 bytes. The MTU size outside *X* and *Y* are 1500 bytes.



3.1 (5 points) Without packet loss, if an IPv4 packet carrying a payload (i.e. the data portion) of 384 bytes is sent from A to B, how many fragment(s) would arrive at B? Explain your answer.

2 because the smallest IP header is 20 bytes thus the total size is 404 bytes which is greater than Y's MTU. Thus R2 will fragment the packet into 2 fragments to fit under the MTU.

3.2 (6 points) Without packet loss, if an IPv4 packet carrying a payload of 900 bytes is sent from A to B, how many fragments would arrive at B? Explain your answer.

3, because the total size (including header) is 920 which is greater than Y's MTU. But 2 fragment would only carry 800 bytes, thus we need another fragment.

3.3 (8 points) Without packet loss, and assume that an IPv4 packet carrying a payload 1100 bytes (which is a TCP segment) is sent from A to B, how many fragments would arrive at B? What is the value in the fragment offset field of the last fragment arrived at B? Explain your answer.

Total size = 1120 bytes > 1000 > 400

At X → 996 byte + 144 byte (20+976) (20+124) ← except for last fixed payload must be divisible by 8 bytes since frag. offset uses units of 8 bytes

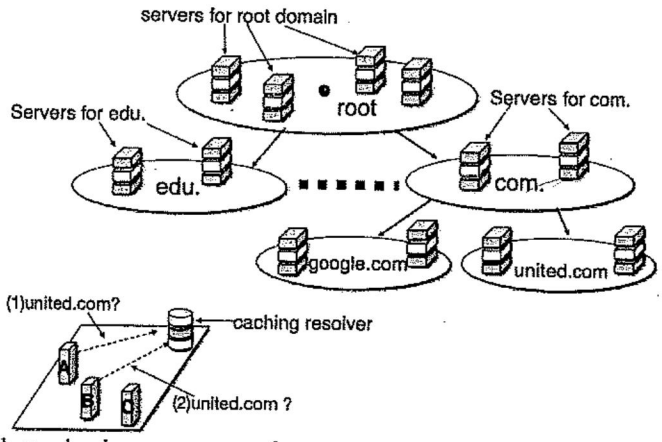
At Y → 396 + 396 + 244 + 144 ⇒ 4 fragments
(20+376) (20+376) (20+224) (20+124)

offset = 396+396+224 / 8 = 976/8 = 122

3.4 (6 points) Assuming the same setting as question 3.3, except that the first fragment after Router R3 is dropped and all the other fragments arrive at B, what will B do with these received fragments? Can it pass to the transport module? Explain your answer.

B cannot pass this to the transport layer since B is missing a fragment. B can only drop the received fragments since IP will not (by itself) retransmit fragments. Since the packet was a TCP segment, TCP will eventually retransmit the entire segment and B will receive all the fragments again. Only then can B reassemble and pass it to the transport module.

Problem 4 (25 points) Consider the following DNS resolution process: at time T=0, the caching resolver in the figure has an empty cache, and Host-A sends a query to resolve the DNS name of *www.united.com* to get the IP address. 2 minutes after Host-A received the answer from the caching resolver, Host-B sends the same query (i.e. asking the IP address of *www.united.com*). Assuming that the one-way delay between all local hosts and the caching resolver is 20 msec (40 msec RTT), and it takes 150 msec for the caching resolver to get a reply from each of all the authoritative DNS servers. All authoritative servers support iterative queries only. There is no packet loss. We also know that the caching resolver finally receives the following information and then sends the reply to Host-A; note that the second column is the TTL value of each received resource record.



www.united.com.	200	IN	A	142.250.68.142
united.com.	4023	IN	NS	ns1.united.com.
united.com.	4023	IN	NS	ns2.united.com.
united.com.	4023	IN	NS	ns3.united.com.

4.1 (5 points) How long does it take for Host-A to get the answer back for the IP address of www.united.com?

40 + 150 + 150 + 150 = 490 ms
cache root .com united.com

4.2 (4 points) How long does it take for Host-B to get the answer back for the IP address of united.com?

40ms, since the record has not expired in the cache yet thus B only needs to query the cache.

4.3 (4 points) How many name servers are in the united.com domain?

3, shown by the cache response there are 3 entries with NS (name server) type

4.4 (4 points) How does the caching resolver know the identity of the root servers when its cache is empty?

The root servers addresses are manually configured into the caching resolver

4.5 (4 points) Who determines TTL values that determine how long the caching resolvers can cache the received resource records? What are the pros and cons of using a small TTL value?

The domain owner determines the TTL values for their domain. A small TTL value (pro) can ensure that the records are up-to-date and can also be used for load-balancing. However (con), a short TTL means the caching resolver needs to query more often.

4.6 (4 points) Why does DNS run on UDP by default instead of TCP?

UDP is faster than TCP as it does not need a 3-way handshake to set up the connection. UDP can immediately send the query and get the response ASAP. UDP can even do reliable data transfer if implemented by the application layer.