CS118 Midterm

Sriram Balachandran

TOTAL POINTS

82.5 / 100

QUESTION 1

20 pts

1.1 5 / 5

✓ **- 0 pts** Correct (4 tuples: SrcIP, SrcPort, DstIP, DstPort)

1.2 5 / 5

✓ **- 0 pts** Correct

1.3 5 / 5

✓ **- 0 pts** correct

1.4 2 / 5

✓ **- 3 pts** Incorrect reason (No adaptive ReTX timer)

QUESTION 2

20 pts

2.1 6 / 6

✓ **- 0 pts** Correct: 184

2.2 5 / 7

✓ **- 2 pts** Wrong calculation of tx/prop delay

2.3 5 / 7

✓ **- 2 pts** Wrong calculation of tx delay

QUESTION 3

15 pts

3.1 2 / 5

✓ **- 3 pts** Wrong RTT

3.2 3 / 5

✓ **- 2 pts** Wrong aggregation/Wrong tx delay

3.3 5 / 5

✓ **- 0 pts** Correct

QUESTION 4

25 pts

4.1 1.5 / 2

✓ **- 0.5 pts** didn't answer how the caching resolver learns the IP address of the root server

4.2 2 / 2

✓ **- 0 pts** Correct

4.3 4 / 4

✓ **- 0 pts** Correct

4.4 0 / 4

✓ **- 4 pts** Incorrect answer. Correct answer is
Total time = 4ms + 80ms = 84ms
A record for www.united.com expired (TTL = 20),
A record for united.com nameserver is still in caching resolver,
Query united.com nameserver (80ms)

4.5 4 / 4

✓ **- 0 pts** Correct

4.6 4 / 4

✓ **- 0 pts** Correct

4.7 5 / 5

✓ **- 0 pts** correct

QUESTION 5

20 pts

5.1 0 / 0.5

✓ - 0.5 pts Incorrect

5.2 0.25 / 0.25

✓ - 0 pts Correct (FIN=0)

5.3 0.5 / 0.5

✓ - 0 pts Correct (ACK=1)

5.4 1 / 1

✓ - 0 pts Correct (Seq#=762)

5.5 1 / 1

✓ - 0 pts Correct (ACK#=8514)

5.6 0.5 / 0.5

✓ - 0 pts Correct (SYN=0)

5.7 0.25 / 0.25

✓ - 0 pts Correct (FIN=0)

5.8 0.5 / 0.5

✓ - 0 pts Correct (ACK=0/1)

5.9 1 / 1

✓ - 0 pts Correct (Seq#=2162)

5.10 1 / 1

✓ - 0 pts Correct (ACK#=8514)

5.11 0.5 / 0.5

✓ - 0 pts Correct (SYN=0)

5.12 0.5 / 0.5

✓ - 0 pts Correct (ACK=0)

5.13 1 / 1

✓ - 0 pts Correct (Seq#=3562)

5.14 1 / 1

✓ - 0 pts Correct (ACK#=8514)

5.15 0.5 / 0.5

✓ - 0 pts Correct (SYN=0)

5.16 0.5 / 0.5

✓ - 0 pts Correct (FIN=0)

5.17 0.5 / 0.5

✓ - 0 pts Correct (ACK=1)

5.18 1 / 1

✓ - 0 pts Correct (Seq#=8514)

5.19 1 / 1

✓ - 0 pts Correct (ACK#=3562)

5.20 0.5 / 0.5

✓ - 0 pts Correct (SYN=0)

5.21 1 / 1

✓ - 0 pts Correct (ACK=1)

5.22 1 / 1

✓ - 0 pts Correct (Seq#=8514)

5.23 1 / 1

✓ - 0 pts Correct (ACK#=3563)

5.24 0.5 / 0.5

✓ - 0 pts Correct (SYN=0)

5.25 0 / 0.5

✓ - 0.5 pts Incorrect

5.26 0.5 / 0.5

✓ - 0 pts Correct (ACK=1)

5.27 1 / 1

✓ - 0 pts Correct (Seq#=3563)

5.28 1 / 1

✓ - 0 pts Correct (ACK#=8515)

CS118
Winter 2020 Midterm Exam

This midterm exam is 1 hour 50 minutes.
Closed book and closed notes; NO use of any device except calculators.
Only 2 pages of cheat sheet is allowed; Write your answer legibly.

- This exam has 7 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 dumping anything and everything, relevant or irrelevant, on the paper. Irrelevant answers may lead to point-deduction as they show lack of understanding to the problem.

Your name: SRIRAM BALACHANDRAN

Student ID: 805167463

	Points	Your score
Problem 1	20	
Problem 2	20	
Problem 3	15	
Problem 4	25	
Problem 5	20	
Total	100	

Problem 1 (20 points) Short and quick questions

1.1 (5 points) Popular web servers, such as cnn.com, may handle web requests from millions of users at the same time. How does a web server distinguish individual TCP connections with different users?

Web server distinguishes individual connections by looking at incoming source IP & Port # in TCP packets it receives.

1.2 (5 points) HTTP is called a stateless protocol that is it handles each and every HTTP request independently from any previous ones. However when one looks up some big websites, e.g. www.amazon.com, that one has visited before, the websites can recognize the user instantly. How do you explain this seemingly conflict between HTTP being a stateless protocol and the "web server can remember you" fact?

This is achieved through site-specific cookies that are assigned by the server, which has a database capable of matching cookie-ids & relevant user-info. When user sends subsequent requests, these cookies are also appended to the user requests, allowing servers to look up info based on cookie id achieving the "web server can remember you" effect.

1.3 (5 points) If a TCP connection operates in the congestion avoidance phase all the time, and has an average congestion window size of 1000 bytes, and an average RTT of 5 msec, what is the average throughput of this connection? Assuming the receiver window size is 8000 bytes (therefore has no impact).

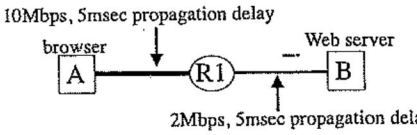
$$\text{Throughput} = \frac{\#\text{ bytes sent}}{RTT} = \frac{1000\text{ Bytes}}{.005\text{ sec}} = \frac{10^3\text{ Bytes}}{5\times10^{-3}\text{ sec}} = \frac{1}{5}\text{ B/s}$$

$$0.2\text{ B/s} = 1.6\text{ bps}$$

1.4 (5 points) Because DNS uses UDP for data delivery, packets may get lost, so DNS caching resolvers must detect lost requests and retransmit. We learned in class that TCP uses a fine-tuned adaptive retransmission timer. Does DNS also use an adaptive retransmission timer? Why or why not?

No it does not. TCP RTOs can be fine tuned since a TCP connection is simply between 2 agents. However a DNS caching resolver will communicate with a number of root & authoritative servers. The overhead of simultaneously maintaining timers for each server the resolver hits outweighs the benefit of using UDP in the first place. Therefore DNS will opt for a "dumber" timer.

Problem 2 (20 points) A web browser on host A is connected to a web server on host B through router R1, the link A—R1 has 10Mbps (10^7 bits/sec) bandwidth, link R1—B 2Mbps, and both links have 5msec propagation delay in each direction, as shown in the figure. A sets up a TCP connection with B by sending SYN packet. Upon receiving SYN-ACK from B, A sends exactly 4 HTTP requests to B, each of these 4 HTTP requests retrieves 1250 bytes of data. Assuming that the size of HTTP requests and TCP SYN/SYN-ACK packets are small enough so that their transmission delays are ignored, and there is no packet loss. Also assume that TCP flow and congestion control window sizes are big enough so that they do not slow down packet transmission. Note that all the questions below ask about the time needed to receive all the data; one does not need to consider the time needed to close the TCP connection.



2.1 (6 points) If the browser uses HTTP/1.0 to retrieve the data and uses a single TCP connection at any given time: starting from sending the first TCP connection setup (SYN) packet, how long will it take for the browser to receive all the 4 pieces of data?

1250 × 8 = 10,000

4 request, non persistent

$D_{req} = \underbrace{2*(5+5)}_{RTT-TCP} + \underbrace{2*(5+5)}_{RTT-req} + \underbrace{\frac{10^4 \cdot 10^3}{2\times10^6}}_{D_{trans-res}} + \frac{10^4 \cdot 10^3}{10^7} = 40 + 5 + 1 = 46$ msec

$D_{tot} = 4 * D_{req} = 4 \times 46 = \boxed{184 \text{ msec}}$

RTT = 20msec

2.2 (7 points) To speed up the retrieval, the browser (still using HTTP/1.0)opens 4 TCP connections in parallel. Starting from sending the first TCP connection setup (SYN) packet, how long will it take for the browser to receive all 4 pieces of data?

Each packet D_{trans} ↑ 4

$D_{req} = RTT + RTT + D_{trans}$

$= 20 + 20 + \left(\frac{10^4 \times 10^3}{(2\times10^6)/4} + \frac{10^4 \cdot 10^3}{10^7/4}\right)$

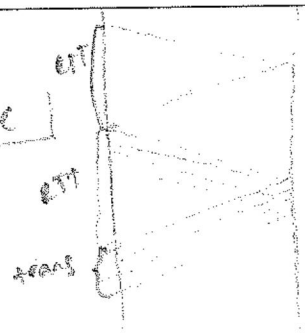
$= 20 + 20 + (20 + 4)$

$D_{tot} = \frac{4 * D_{req}}{4} = \boxed{64 \text{ msec}}$

2.3 (7 points) If the browser uses HTTP/1.1 with pipelining to retrieve the data over a single TCP connection. How long will it take for the browser to receive all 4 pieces of data in this case?

$D_{Tot} = \underbrace{RTT}_{TCP} + \underbrace{RTT}_{req} + 4(D_{trans})$

$= 20 + 20 + 4(5+1) = \boxed{64 \text{ msec}}$



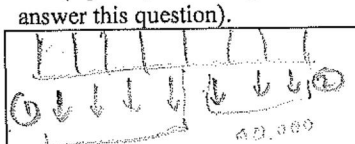
A ——— 10 msec ——— B

10 Mbps

Problem 3 (15 points) Two hosts A and B are connected by a link with bandwidth of 10 Mbps (10^7 bits-per-second) and propagation delay of 10 msec in each direction. Host A has a 2,000,000-bit (2M) file to send to host B. A uses GoBackN reliable transport protocol and divides the file into 10,000-bit packets. The GoBackN protocol uses a fixed window size of 4 packets. You may assume that a) there is no connection setup step, b) the *transmission time* of ACK packets is negligible, and c) no data or ACK packet gets lost.

3.1 (5 points) How long will it take for the 7th packet to be completely received by Host B? (drawing a diagram may help answer this question).

A CK



$\frac{2\times10^6}{10^4} = 200$ packets RTT = 20

$D_7 = D_{1-4} + D_{5-8} = 2D_{window}$

$= 20 + 4\times\frac{10^4\times10^3}{10^7} + 20 + 4\times\frac{10^4\times10^3}{10^7} = 48$ msec

$D_{window} = 48/2 = 24$ msec

ACK lost

3.2 (5 points) How long will it take before the entire file is received by Host B?

$\#win = \frac{\#\,Packet}{win\_size} = \frac{200}{4} = 50$

$D_{tot} = \#win \times D_{window} = 50 \times 24 = 1200$ msec

3.3 (5 points) Can the file be delivered to host B faster by adjusting the window size? If so what is the minimal window size, in terms of number of packets, that would allow the file to be received by B in shortest possible time (assume no other setting is changed)?

Think back to packet switching HW

Current Utilisation $= \frac{4\times\frac{L}{R}}{RTT+\frac{L}{R}} = \frac{4}{21}$ The file can be delivered faster

Max Utilisation $= 1 = \frac{N\times\frac{L}{R}}{RTT+\frac{L}{R}} = \frac{N}{21}$

$N = 21$

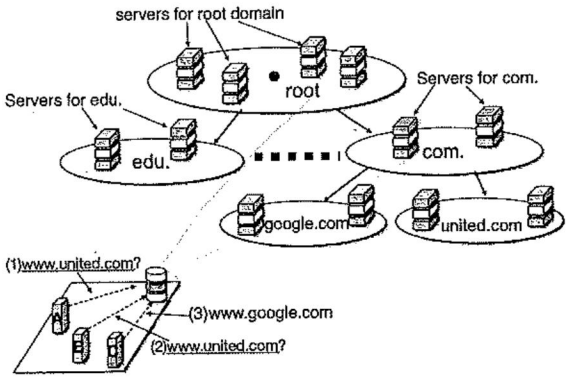
21 is the minimal window size needed for max utilisation

Problem 4 (25 points) Consider the following DNS resolution process: at time T=0, the caching resolver in the figure has an empty cache; the one-way delay between all local hosts and the caching resolver is 2 msec (4 msec RTT), and it takes 80 msec for the caching resolver to get a reply from each of any of the authoritative DNS servers (i.e. RTT=80msec). All authoritative servers support iterative queries only. There is no packet loss unless stated explicitly.

a) Host-A sends a query to resolve the DNS name of *www.united.com* to get its IP address.

b) 30 seconds 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*).

c) Host-C sends a query asking the IP address of *www.google.com* at the same time with Host-B's query.



We also know that the caching resolver received the following information before it sends the reply to Host-A; note that the second column below is the TTL value (in the number of seconds) of each received resource record.

```
;; ANSWER SECTION:
www.united.com.         20      IN      A       184.25.233.142

;; AUTHORITY SECTION:
united.com.             144632  IN      NS      a26-64.akam.net.
united.com.             144632  IN      NS      a5-65.akam.net.
united.com.             144632  IN      NS      a1-66.akam.net.
united.com.             144632  IN      NS      a18-67.akam.net.
united.com.             144632  IN      NS      a11-66.akam.net.
united.com.             144632  IN      NS      a10-65.akam.net.

;; ADDITIONAL SECTION:
a5-65.akam.net.         86222   IN      A       95.100.168.65
a5-65.akam.net.         79398   IN      AAAA    2600:1480:b000::41
a10-65.akam.net.        134354  IN      A       96.7.50.65
a11-66.akam.net.        44211   IN      A       84.53.139.66
a11-66.akam.net.        43610   IN      AAAA    2600:1480:1::42
a18-67.akam.net.        30855   IN      A       95.101.36.67
a26-64.akam.net.        73520   IN      A       23.74.25.64
```

4.1 (2 points) Where does the caching resolver send a DNS query when its cache is empty? How does it learn the IP address of the entity it needs to send the query to?

Sends to root server. It iterates based on the response from root, then will query again based on root response until the query is resolved

4.2 (2 points) How many name servers does the *united.com* domain have?

6.

4.3 (4 points) How long does it take for Host-A to get the answer back for the IP address of *www.united.com*?

$RTT_{cache} + 3RTT_{DNS} = 4 + 3(80) = 244$ msec

4.4 (4 points) How long does it take for Host-B to get the answer back for the IP address of *www.united.com*?

It is cached - so only RTT_{cache}, 9 msec

4.5 (4 points) How long does it take for Host-C to get the answer back for the IP address of *www.google.com*?

$RTT_{cache} + 2RTT_{DNS} = 4 + 2(80) = 164$ msec

.com is in cache, so Cache resolver hits .com, receives google.com and then hits google.com

4.6 (4 points) Who determines TTL values that determine how long DNS servers cache name-to-address mapping records? What are the pros and cons of using a small TTL value?

By using a small TTL value, one can quickly recover if perhaps the server at the cache-address goes down. However using a small TTL can result in higher avg. DNS lookup latency, since the cache would expire frequently, requiring the cache to request data from servers more frequently. The authoritative servers determine the TTL.

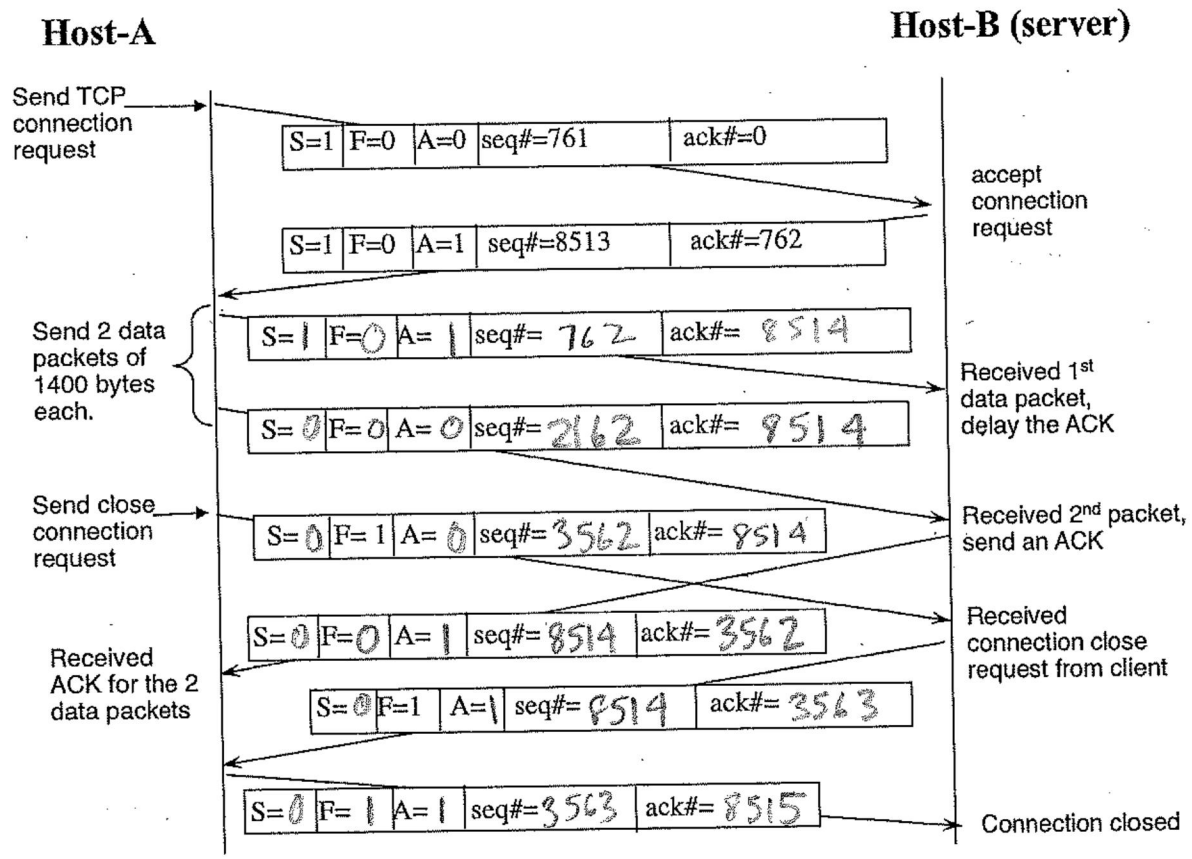
4.7 (5 points) By default DNS uses UDP, instead of TCP, as its transport protocol. Can you identify any advantages and/or disadvantages from DNS's use of UDP versus TCP? Please explain your answer.

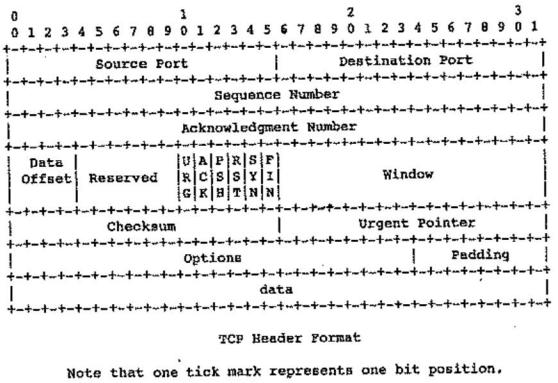
UDP is faster than TCP. However UDP is unreliable compared to TCP. This reliability is the key factor to consider when comparing uses. TCP maintains reliability by retransmitting as effectively as possible using a smart-adaptive timer. However, in the case of DNS and DNS cache-resolvers specifically, the overhead that would result in attempting to intelligently maintain RTOs for every request could slow down the cache speed. By using a "dumb" retransmission protocol with low computation & UDP, we can achieve faster performance.

Problem 5 (20 points) Assuming a student developed a Simple-File-Push application (SFP) that uses port number 1234 at the server end. Host-A opens a TCP connection (its TCP source port is 5678), uses SFP to send a file to Host-B, then close the connection immediately. The figure below shows the packet exchanges between the two nodes. B's TCP receiver window size is 5000 bytes, and we do not consider congestion control in this problem.

- Host-A sends the first packet to set up a TCP connection, and chooses 761 as its initial sequence number.
- The SFP server accepts the connection request and chooses an initial sequence number of 8513.
- The 3rd and 4th packets are from Host-A to B, each carrying 1400 bytes of data.
- The 5th packet is also from Host-A to B, to close the connection (i.e. this packet has the FIN flag set to 1), before receiving the delayed-ACK from B (which is the 6th packet below)
- Two more packets are exchanged to close the TCP connection, SFP successfully pushes a 2800-byte file from A to B.

Please fill in *all the blank fields* in the last 6 packets below (S: SIN flag; F: FIN; A: ACK flag in TCP header). FYI the TCP header format is shown at the bottom of the page.





TCP Header Format

Note that one tick mark represents one bit position.

Data offset: indicates where the data begins starting from the beginning of the TCP segment
The meaning of the 5 flag bits:
URG: urgent flag; PSH: push flag (both no longer in use)
RST: reset flag;
SYN: i.e. connection setup request
ACK: 1 indicates that Acknowledgment Number field being used
FIN: connection close request

TCP delayed-ACK: to reduce the number of pure ACK packets, in the absence of packet losses, TCP receiver may send one ACK for every 2 packets received if the 2nd one arrives within 500msec after the first.