CS118
Spring 2016 Midterm Exam

1 hour 50 minutes

Close book and closed notes; NO use of any device except calculators.

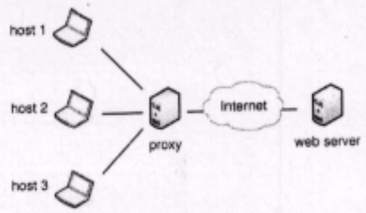
- This exam has 6 pages, including this cover page. Do all your work on these exam sheets.
- Cross out all the scratch work that you do not want to be counted as part of your answer before you submit the exam.
- Show ***all*** your work, including unfinished problems that you wish to be considered for partial credit.
- Be *specific, clear,* ***concise*** in your answers, and *explain* your answers.
- When the answer to a problem is not immediately clear, do not simply dump everything, relevant or irrelevant, on the paper. Irrelevant answers may lead to point-deduction as they show the lack of understanding of the problem.

Your name: Bo Yun (Andy) Shih

Student ID: 404 418 629

Problem 1 (20 points) Three hosts share the same web caching proxy whose cache is empty at the beginning. The browser *on host 1 sends the proxy a request for* `http://foo.com/info?uid=tom`. *This initial object contains three* referenced objects, which are then retrieved by the browser *on host 1:*

```
http://foo.com/logo.png
http://foo.com/profile?uid=tom
http://foo.com/footnote
```



10 seconds later, the browser on host 2 sends a request for `http://foo.com/info?uid=jerry`. This initial object also contains three referenced objects:

```
http://foo.com/logo.png
http://foo.com/profile?uid=jerry
http://foo.com/footnote
```

1.1 (4 points) Please circle one or more HTTP requests that were sent from the caching proxy in the first 10 seconds.

(a) `http://foo.com/info?uid=tom`
(b) `http://foo.com/logo.png`
(c) `http://foo.com/profile?uid=tom`
(d) `http://foo.com/footnote`
(e) `http://foo.com/info?uid=jerry`
(f) `http://foo.com/profile?uid=jerry`

1.2 (4 points) Please circle one or more the HTTP requests were sent from the caching proxy after the first 10 seconds.

(a) `http://foo.com/info?uid=tom`
(b) `http://foo.com/logo.png`
(c) `http://foo.com/profile?uid=tom`
(d) `http://foo.com/footnote`
(e) `http://foo.com/info?uid=jerry`
(f) `http://foo.com/profile?uid=jerry`

1.3 (12 points) Please circle True or False.

[F] (True or False) If another host, host 3, sends a request for `http://foo.com:8080/logo.png`, the proxy will not send any HTTP request.

[F] (True or False) If host 3 sends a request for `http://foo.com:80/logo.png`, the proxy will not send any HTTP request.

[F] (True or False) Host3 sends another request for `http://bar.com/logo.png`. When the caching proxy sends the DNS query for name `bar.com`, DNS returns the same IP address as the IP address for `foo.com`. This must be an error.

[T] (True or False) If `http://foo.com/logo.png` object is already cached in the proxy, the caching proxy will not send a separate HTTP request for `http://bar.com/logo.png`.

$RTT_0 = 2 \times d\_prop = 40ms$ $RTT_n =$ [illegible]

Total time = $8 \times (RTT_0) + 4 \times (RTT_n) = 380ms$

Need to do TCP connect for each object, since non-persistent.

2.2 (6 points) To speed up the retrieval, the browser opens 3 TCP connections in parallel. Again 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?

Total time = $4 \times (RTT_0) + 2 \times (RTT_n) = 190ms$

3 connections in parallel but 4 pieces of data, so one connection still has to retrieve two pieces of data. Same calculation as part A but with two pieces.

2.3 (4 points) Assuming the browser uses HTTP/1.1 ***without 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?

Total time = $5 \times (RTT_0) + 4 \times (RTT_n) = 260ms$

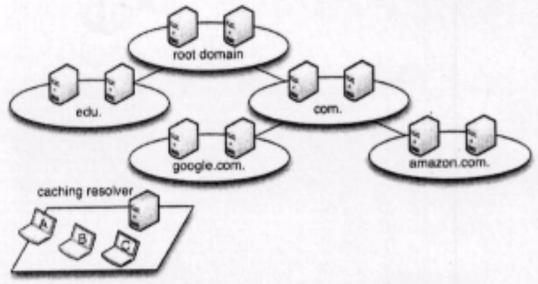
Only need to open TCP connection once, which is one RTT_0 for each object, send the request for one RTT_0 and RTT_n (d-trans)

2.4 (4 points) Assuming 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? Is the delay the same as the one of parallel connections? If so, why we still prefer HTTP/1.1 with pipelining?

Total time = $2 \times (RTT_0) + 4 \times (RTT_n) = 140ms$

delay is slightly faster than parallel connection with 3 channels. Pipelining is still preferred because less stress on the server to establish multiple TCP connections

Problem 3 (20 points) Consider the following DNS resolution process:
at time T=0: the caching resolver in the figure has an empty cache. Host-A sends a query to resolve the DNS name *www.google.com* and get the IP address.
T=30 minutes: Host-B sends a query for the IP address of *www.amazon.com* and gets the answer.
T=70 minutes: Host-C sends a query for DNS name *hangout.google.com* and another query for DNS name *video.amazon.com*.



Assuming that it takes 10 msec for packet resolver (10 msec is the round trip delay), and it takes 100 msec for the caching resolver to get a reply from any of the authoritative DNS servers. All authoritative servers support iterative queries only. All the DNS data has a TTL value of 1 hour. There is no packet loss.

3.1 (4 points) How long does it take for Host-A to get the answer back for the IP address of *www.google.com*?

10ms + 100ms + 100ms + 100ms = 310ms
(root→com, com→google, google→www.)

Nothing in caching resolver, so send request for everything

3.2 (4 points) How long does it take for Host-B to get the answer back for the IP address of *www.amazon.com*?

10ms + 100ms + 100 ms = 210ms
(com→amazon, amazon→www)

root→com already stored in caching resolver

3.3 (3 points) How long does it take for Host-C to get the answer back for the IP address of *hangout.google.com*?

10ms + 100ms + 100 ms + 100 = 310ms
(root→com, com→google, google→hangout)

root→com
com→google stored, but timed out since TTL = 1 hour

3.4 (3 points) How long does it take for Host-C to get the answer back for the IP address of *video.amazon.com*?

10 ms + 100 ms (amazon→video) = 110 ms

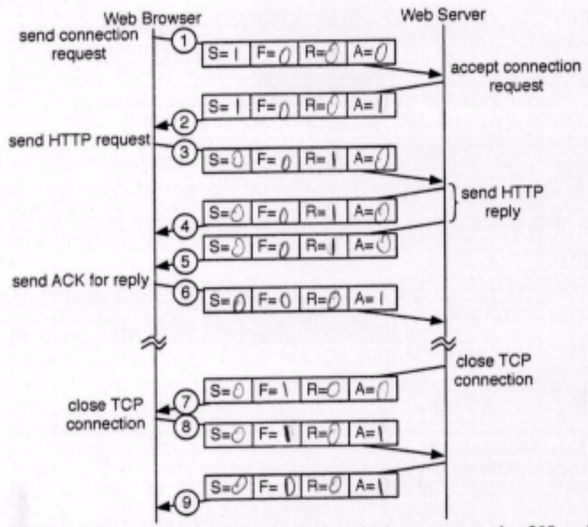
root→com and com→amazon stored

3.5 (6 points) At T=100 minutes, all the authoritative servers of .com go offline. Which domain names below can be resolved by Host-A? Circle those domain names:

(a) www.google.com (b) hangout.google.com (c) doc.google.com
(d) www.amazon.com (e) video.amazon.com (f) aws.amazon.com

Problem 4 (20 points) The following diagram shows a sequence of TCP packets for a session between a web browser and a web server. The HTTP in use is version 1.0 (non-persistent HTTP).

4.1 (6 points) Fill in all the missing flag values for the SYN, FIN, RST, and ACK flags in the TCP headers (when the flag is set, the value is 1, otherwise is 0).



4.2 (8 points) If the web browser starts its TCP connection with the initial sequence number 308, and web server picks 1110 as its initial sequence number, the HTTP request size is 150 bytes, and the HTTP reply is made of 2 packets with 1500 byte data each. What is the sequence number and acknowledge number on the ***numbered*** packets?

No	Sequence No.	Ack No.
1	308	--
2	1110	309
3	458	1111
4	2610	459

No	Sequence No.	Ack No.
5	4110	459
6	459	4111
7	4111	460
8	460	4112
9	4112	461

4.3 (3 points) Why the sequence number at each end of a TCP connection starts from a random number, instead of zero?

This is to prevent attacks from malicious users. If the sequence number always started at zeros malicious users can guess port numbers and shut down the TCP connection unwantedly. Randomizing this number prevents them from guessing as easily.

4.4 (3 points) How does the web server know that the browser has received the last packet (packet 9)?

The web server will wait one RTT for the browser. If the browser didn't receive the last packet it would resend another close. Otherwise if no further response then browser has received it. But you can never be sure because of two army paradox.

Problem 5 (20 points)

5.1 (4 points) You have learned four application layer protocols: HTTP, FTP, SMTP, and DNS. Only one of them can run over UDP. Which protocol is it? Why it is preferred to run over UDP than TCP? **(in one sentence, otherwise you will not get any credit)**

DNS uses UDP, because faster and can easily resend if transmission failed since low overhead.

5.2 (4 points) If you are asked to develop a real-time online conferencing application, will you choose TCP as the transport layer protocol? Justify your answer.

No, I would choose UDP as the transport layer protocol because TCP will be slow due to its reliability guarantees. Video conferences don't mind a few dropped packets, just need overall speed to be fast. (not waste time sending ACKs)

5.3 (4 points) You went to amazon.com website and Chrome shows you the above state in the address bar. In which case you can safely send your Amazon login and password information and why? If in some cases it is not safe, list those and explain why is it not safe and/or what could have gone wrong.

(a)

Safely send password and login ✓

(b)

- didn't register SSL no certificate, could be hacker imitating amazon and recording my password

(c)

- address doesn't match that of registered certificate. This is an imitation server so should not send password

5.4 (4 points) Some major email service providers recently announced that they have adopted HTTPS-like approach to secure the email communication (each connection between client and SMTP server and between SMTP servers is secured using HTTPS-like connection). Do you think their solution can secure email communication and eliminate all spam? Justify your answer.

No it will be relatively difficult because SMTP is fundamentally a push protocol. You still want to accept emails from new addresses because they can still be important, even if they're not certified. HTTPS works well because HTTP is pull protocol, so users actively find pages they want, so we just have to verify what users directly want.

5.5 (4 points) HTTP 1.1 already allows a client to send multiple requests in a single connection. Why we still need multiple streams in HTTP 2.0?

HTTP 1.1's multiple requests still need to arrive in order, so we are not getting maximum efficiency. Multiple streams in 2.0 mean packets can arrive out of order. As well, lower header costs since in 1.1 multiple req mean multiple headers so high overhead. (unless we use chunked encoding but more overhead)