Problem 1

Suppose that you walked into Boelter Hall and get connected to CSD WiFi network, which automatically gave you IP address of the local DNS server. Suppose the local DNS server has just rebooted and its cache is completely empty; RTT between your computer and the local DNS serverr is 10ms and RTT between the caching resolver and any authoritative name server is 100ms; all responses have TTL 12 hours. Suppose the DNS lookup follows iterative searching.

(a) If you try to go to `ucla.edu`, what would be minimum amount of time you will need to wait before your web browser will be able to initiate connect to the UCLAs web server? Suppose the location of UCLAs web server is delegated to ucla.edu name server.

(b) What would be the time, if a minute later you will decide to go to `ccle.ucla.edu`? Suppose `ccle.ucla.edu` is delegated to ucla.edu name server.

a)
It would take 10 ms from client to local DNS server
100 ms from local DNS to root DNS
100ms from root DNS to TLD .edu server
100 ms from local DNS to the .edu

total time: 310 ms

b)
It would take 10ms from client to local DNS server
and since responses stored in cache for 12 hours, the only other time would be from client to ucla.edu server is 100 ms

total time: 110 ms

Problem 2

Suppose you have a new computer just set up. `dig` is one of the most useful DNS lookup tool. You can check out the manual of `dig` at `http://linux.die.net/man/1/dig`. A typical invocation of `dig` looks like: `dig @server name type`.

Suppose that on April 17, 2019 at 15:35:21, you have issued "`dig google.com A`" to get an IPv4 address for `google.com` domain from your caching resolver and got the following result:

```
; <<>> DiG 9.8.3-P1 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17779
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;google.com.                IN     A

;; ANSWER SECTION:
google.com.           239     IN     A      172.217.4.142

;; AUTHORITY SECTION:
google.com.           12412   IN     NS     ns4.google.com.
google.com.           12412   IN     NS     ns2.google.com.
google.com.           12412   IN     NS     ns1.google.com.
google.com.           12412   IN     NS     ns3.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.       13462   IN     A      216.239.32.10
ns2.google.com.       13462   IN     A      216.239.34.10
ns3.google.com.       13462   IN     A      216.239.36.10
ns4.google.com.       13462   IN     A      216.239.38.10

;; Query time: 81 msec
;; SERVER: 128.97.128.1#53(128.97.128.1)
;; WHEN: Wed Apr 17 15:35:21 2019
;; MSG SIZE rcvd: 180
```

(a) What is the discovered IPv4 address of `google.com` domain?

(b) If you issue the same command 1 minute later, how would "ANSWER SECTION" look like?

(c) If the client keeps issuing `dig google.com A` every second, when would be the earliest (absolute) time the local DNS server would contact one of the `google.com` name servers again?

(d) If the client keeps issuing `dig google.com A` every second, when would be the earliest (absolute) time the local DNS server would contact one of the `.com` name servers?

Problem 2 continued on next page...

a) 172.217.4.142

b) The TTL would decrease. the 239 number would change to 179 sec.

c) It would contact it once the time of the TTL is gone. So 239 seconds later it will contact google server.

d) It would contact a .com server 13462 seconds after since that is when the TLL for the .com connection will expire.

Problem 3

Suppose Bob joins a BitTorrent torrent, but he does not want to upload any data to any other peers (so called free-riding).

(a) Bob claims that he can receive a complete copy of the file that is shared by the swarm. Is Bob's claim possible? Why or why not?

(b) Bob further claims that he can further make his "free-riding" more efficient by using a collection of multiple computers (with distinct IP addresses) in the computer lab in his department. How can he do that?

a) This statement is true. This can only be true if there are enough peers on the same network so that Bob can always receive data without uploading anything.

b) This is also true because bob could write a simple program that allows the different computers in his department to only ask for files that the other computers do not have.

Problem 4

Consider a reliable data transfer protocol that uses only negative acknowledgments. Suppose the sender sends data only infrequently. Would a NAK-only protocol be preferable to a protocol that uses ACKs? Why? Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

The first case is if the user is sending small amounts of data. In NAK, a packet loss is only detected after the next packet is recieved. If there is a long delay between x and x+1, then it will be a long time until x can be recovered in NAK since the data sent will be infrequent.

If the data is sent more often, then NAK can recover that loss packet more quick. NAK only sent where there is few errors and ACK are not even sent which reduces feedback in NAK over ACK.

Problem 5

Consider the GBN protocol with a sender window size of 6 and a sequence number range of 1,024. Suppose that at time t, the next in-order packet that the receiver is expecting has a sequence number of k. Assume that the medium does not reorder messages. Answer the following questions:

(a) What are the possible sets of sequence numbers inside the senders window at time t? Justify your answer.

(b) What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time t? Justify your answer.

a) The window is size 6. If the sender used the ACK protocol, then the sender window would be [k,k+N-1]. But if none of the ACK was received at the sender, then the window would be k-1 and the N packets including k-1. So the window would be [k-N, k-1].

Since the window is size 6, then the range is [k-6, k].

b) If the reciever is waiting for packet k, then it has received k-1 and all the N-1 packets before that. If none of the packets were recieved, ACK in the windows [k-N, k-1] are still propagating back.
But since the sender has sent the packets in the window [k-N, k-1], sender must have received ACK for k-N-1, which means cannot send an ACK that is less than that. So the range becomes k-N-1 to k-1.