# 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) RTT from computer to local DNS server: 10ms
RTT from local DNS server to root DNS server: 100ms
RTT from local DNS server to .edu TLD server: 100ms
RTT from local DNS server to ucla.edu authoritative name server: 100ms
Total time: 310ms

b) local DNS server contains IP-hostname translation for ucla.edu
RTT from computer to local DNS server: 10ms
RTT from local DNS server to ucla.edu authoritative name server: 100ms
RTT from local DNS server to ccle.ucla.edu authoritative name server: 100ms
Total time: 210ms

# 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) The IPv4 address for google.com is 172.217.4.142

b) The Answer section would look the same as it does now, except with the numerical value as 179 instead of 239. This value is known as the TTL of a packet in the network and it is the maximum amount of time that the packet should travel through the network before it is discarded. Thus, a minute later, the TTL of the packet is 60 seconds less.

c) The earliest time the local DNS server would contact one of google.com's name servers again would be 239 seconds.

d) The earliest time the local DNS server would contact one of the .com name servers again would be 12412 seconds.

## 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) Free-riding is possible if the members of the swarm have different parts of the file that make up the whole file. If any part of the whole file is missing, then no one can finish downloading the file, unless a seeder connects to the swarm. A seeder is someone who has the complete file and is uploading to other peers in the swarm. Therefore, as long as different parts of the file are present in the peers connected to the swarm, Bob can download the complete file.

b) Bob can make his free-riding more efficient by using multiple systems. He can configure each system to download a different part of the file. When all systems are finished downloading, he will have the complete file.

# 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?

If the sender sends data infrequently, then an ACK protocol would be preferred over an NAK-only protocol. Because data is sent infrequently, a loss would take a long time to recover. For example, if we lose one packet, the receiver would not know that a packet was lost, until the receiver receives the next packet, which could take a long time since data is sent infrequently. Only after receiving the next packet will the receiver send an NAK to the sender. If the packets must be received in order, then the sender must retransmit the lost packet, as well as the packet that follows. This creates a high delay. If we use the ACK-only protocol, the sender would realize sooner if a packet was lost and wouldn't have to use resources to retransmit the following packet.

If the sender sends data frequently and has a lot of data to send, then a NAK-only protocol would be preferred over an ACK-only protocol. This is because the ACK-only protocol would cause a high amount of overhead and delay, as it must send an ACK for every successfully transmitted packet. Since there are many packets, the number of ACKs sent would be very high. Furthermore, even if a packet is lost, the sender would immediately send the following packet and the receiver would send an NAK for the lost packet. Since the packets are sent so frequently, the delay from a lost packet would be short.

# 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 contains two parts:
1. the first part is made of packets that have been sent, but not yet ACK'd by the receiver
2. the second part is made of packets that are ready to be transmitted by have not yet been sent
Since k represents the start of the second part of the window, and the window is of size 6, there are two extreme cases:
1. the second part consists only of the next-order packet k, and the first part consists of 5 already sent packets - window ranges from (k - 5, k)
2. the second part makes up the entire window - window ranges from (k, k + 5)
Thus, the possible sets of sequence numbers must lie within the range (k - 5, k + 5), with a window size of 6.

b) If the first part contains 5 values and the second part contains only 1, then the range of possible values of the ACK field is (k - 5, k - 1). If the first part were any smaller, it would still lie in that range, with the highest value being k - 1, since k has not been transmitted yet.