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） Since the local DNS server has just rebooted. It needs to query root DNS server, TLD DNS server (edu), and the ucla.edu name server. These will cost 3*100ms=300ms. We also need the 10ms to connect requesting host with the local DNS server. So in total we will need 310 ms.

(b) Since TTL is 12 hours. A minute later the address of ucla.edu server is cached. Thus we need only 100 ms to connect local host with ucla.edu name server to retrieve address of ccle.ucla.edu. We still need the 10 ms to connect requesting host with local DNS server. So in total we need 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) the IPV4 address is revealed in the answer section, it is 172.217.4.142

(b) The IPV4 address will not change, but the time to live will go down by 60 seconds. Thus it will look like:

```
;; ANSWER SECTION:
google.com.          179     IN      A       172.217.4.142
```

(c) the google.com server should be contacted when the TTL in the answer section expires. That would be 239 seconds after 15:35:21, which is 15:39:20.

(d) the .com server will be contacted when the authority section TTL runs out. That will be 12412 seconds after 15:35:21, which is 19:02:13.

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) Bob's claim is possible. Because every user participating in a BitTorrent will randomly pick one peer and send it data after a certain period of time. Thus it is possible that Bob receives all of the data of the file by being optimistically unchoked.

(b)The more computers participating in the BitTorrent, the more likely one of the computers will be optimistically unchoked. If each computer participates in the BitTorrent individually, it will be much faster to obtain all the data chunks of the file.

And Bob can combine all these file chunks obtained from all computers to get the entire 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?

In a NAK-only protocol,

If the package is corrupted NAK is sent back immediately. But if the package is lost or the

NAK is lost, the sender will only know it when the next package receives a NAK (generated by the receiver by comparing sequence number). Thus in a NAK-only protocol

which has little traffic (data is sent infrequently), this could take a lot of time for the sender to realize a package lost or NAK lost. While ACK would reduce this time cost. Thus NAK-only is not preferable when traffic is low.

If there is a lot of data to be sent, and the traffic is high. The time to realize bit error and package lost are both small. Also, a NAK-only protocol will reduce all the overhead generated by ACK traffic. Thus NAK-only is more preferable when traffic is high.

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) Since the receiver is expecting sequence number of k. If the receiver generates very fast replies, all the ACKs for sequence number up to k-1 can already arrive at sender. Thus the sender can have at most 6 messages in the sender windows, from sequence number k to k+5.

The other extreme is that the receiver generate very slow replies, thus all the ACKs for sequence number up to k-1 are on their way to be sent to the sender. Thus the sender can have at most 6 messages in the window from sequence number k-6 to k-1.

Thus all possible sets are size-6 sequence numbers in between the two extremes described above. They are [k-6, k-1],[k-5,k],[k-4,k+1],[k-3,k+2],[k-2,k+3],[k-1,k+4],and [k,k+5]. ([a,b] means larger or equal to a and smaller or equal to b, including both endpoints)

(b) Since the receiver is expecting sequence number k, the largest back propagating can only be k-1. We also know that for each smaller back propagating sequence numbers, all the sequence numbers larger than it must also be back propagating. Since there can be at most 6 such back propagating sequence numbers, the smallest back propagating sequence number is k-6. Thus all the possible values of the ACK field is in the range of [t-6, t-1].