ECE/CS 438: Communication Networks **Spring 2023**

Homework 2

Handed Out: Febuary 12^{th}*,* 2023 *Due: 11:59pm, February* 19^{th}*,* 2023

TA: Pradhyumna Padmanabha

- Homework assignments must be submitted online through GradeScope. Hard copies are not accepted. Please submit a pdf file to GradeScope (`https://www.gradescope.com`). You can either type your solution or scan a **legible** hand-written copy. We will not correct anything we do not understand. Contact the TAs via Campuswire if you face technical difficulties in submitting the assignment. Note: Please select your page(s) for each question during submission on GradeScope; Otherwise, your submission will not be graded.
- Homework assignments can be done in groups of two, but only one person needs to submit on GradeScope. Remember to include your partners name in the solution and on GradeScope by editing "Group Members". It is your responsibility that your partner's name is included. For detailed instruction please refer to (`https://youtu.be/rue7p_kATLA`).
- You can use Campuswire to find a partner. We highly recommend working in groups. You will not get extra credit for working alone.
- Please use Campuswire and come to office hours if you have questions about the homework. Failure to understand the solutions will be the student's fault.
- While we encourage discussion within and outside of the class, cheating and copying is strictly prohibited. Copied solutions will result in the entire assignment being discarded from grading at the very least and a report filed in the FAIR system. It is also your responsibility to ensure that your partner obeys the academic integrity rules as well.

1 DNS - 14 points

The task requires using the `dig` command to provide answers. To ensure accurate results, it is recommended to perform these steps from a computer located on a campus network. The user can refer to the `dig` documentation to understand how to utilize it.

1. Starting from one of the root servers *a*–*m*`.root-servers.net`, perform an *iterative* lookup for the host `www.eecs.mit.edu`. For instance, you can initiate the search by using the following command:

   ```
   dig @h.root-servers.net www.eecs.mit.edu
   ```

   Please provide a list of the following information for each name server you visit during the lookup process:

   (a) Can you specify the domain name of the name server being visited?

   (b) Can you provide the IP address of the name server that is currently being used?

   (c) How long did the query take?

(d) What is the round-trip time (RTT) to the server, which can be determined by using the "ping" command to connect to the server?

(e) For how long can you store the results in cache?

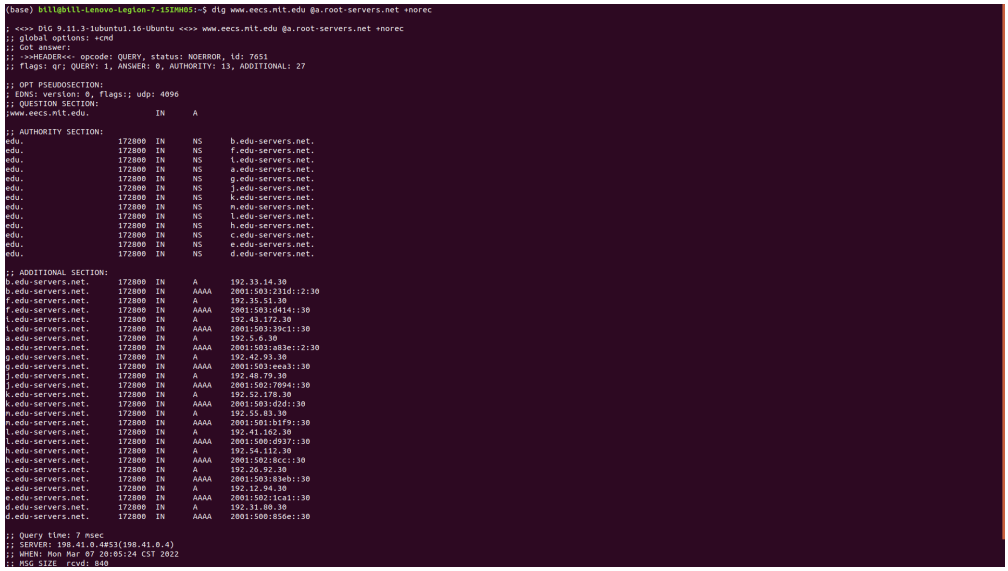
2. Perform a *recursive* query using `resolver.illinois.edu` of the name `www.eecs.mit.edu`. Was this query faster or slower than the sum of the iterative steps? Why do you think that is?

3. Perform an iterative reverse-mapping query for the address of `www.eecs.mit.edu` you found in the previous steps, using `dig -x`. List again the information asked in part 1.

4. Can you explain why the DNS protocol tends to utilize UDP rather than TCP, considering what has been previously discussed in this inquiry?

Solution: (5+3+3+3 points)

1. Note that the commands and responses might differ.



```
f.edu-servers.net.      172800  IN      A       192.35.51.30
f.edu-servers.net.      172800  IN      AAAA    2001:503:d414::30
l.edu-servers.net.      172800  IN      A       192.43.172.30
l.edu-servers.net.      172800  IN      AAAA    2001:503:39c1::30
a.edu-servers.net.      172800  IN      A       192.5.6.30
a.edu-servers.net.      172800  IN      AAAA    2001:503:a83e::2:30
g.edu-servers.net.      172800  IN      A       192.42.93.30
g.edu-servers.net.      172800  IN      AAAA    2001:503:eea3::30
j.edu-servers.net.      172800  IN      A       192.48.79.30
j.edu-servers.net.      172800  IN      AAAA    2001:502:7094::30
k.edu-servers.net.      172800  IN      A       192.52.178.30
k.edu-servers.net.      172800  IN      AAAA    2001:503:d2d::30
m.edu-servers.net.      172800  IN      A       192.55.83.30
m.edu-servers.net.      172800  IN      AAAA    2001:501:b1f9::30
i.edu-servers.net.      172800  IN      A       192.41.162.30
i.edu-servers.net.      172800  IN      AAAA    2001:500:d937::30
h.edu-servers.net.      172800  IN      A       192.54.112.30
h.edu-servers.net.      172800  IN      AAAA    2001:502:8cc::30
c.edu-servers.net.      172800  IN      A       192.26.92.30
c.edu-servers.net.      172800  IN      AAAA    2001:503:83eb::30
e.edu-servers.net.      172800  IN      A       192.12.94.30
e.edu-servers.net.      172800  IN      AAAA    2001:502:1ca1::30
d.edu-servers.net.      172800  IN      A       192.31.80.30
d.edu-servers.net.      172800  IN      AAAA    2001:500:856e::30

;; Query time: 7 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Mon Mar 07 20:05:24 CST 2022
;; MSG SIZE  rcvd: 840

(base) bill@bill-Lenovo-Legion-7-15IMH05:~$ dig www.eecs.mit.edu @b.edu-servers.net +norec

; <<>> DiG 9.11.3-1ubuntu1.16-Ubuntu <<>> www.eecs.mit.edu @b.edu-servers.net +norec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 929
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 8, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.eecs.mit.edu.              IN      A

;; AUTHORITY SECTION:
mit.edu.                172800  IN      NS      usw2.akam.net.
mit.edu.                172800  IN      NS      asia1.akam.net.
mit.edu.                172800  IN      NS      asia2.akam.net.
mit.edu.                172800  IN      NS      use2.akam.net.
mit.edu.                172800  IN      NS      ns1-37.akam.net.
mit.edu.                172800  IN      NS      ns1-173.akam.net.
mit.edu.                172800  IN      NS      eur5.akam.net.
mit.edu.                172800  IN      NS      use5.akam.net.

;; Query time: 21 msec
;; SERVER: 192.33.14.30#53(192.33.14.30)
;; WHEN: Mon Mar 07 20:05:59 CST 2022
;; MSG SIZE  rcvd: 212

(base) bill@bill-Lenovo-Legion-7-15IMH05:~$
```

```
(base) bill@bill-Lenovo-Legion-7-15IMH05:~$ dig mlteecs.wpengine.com @a.root-servers.net +norec

; <<>> DiG 9.11.3-1ubuntu1.16-Ubuntu <<>> mlteecs.wpengine.com @a.root-servers.net +norec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33826
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;mlteecs.wpengine.com.          IN      A

;; AUTHORITY SECTION:
com.                    172800  IN      NS      e.gtld-servers.net.
com.                    172800  IN      NS      b.gtld-servers.net.
com.                    172800  IN      NS      j.gtld-servers.net.
com.                    172800  IN      NS      m.gtld-servers.net.
com.                    172800  IN      NS      i.gtld-servers.net.
com.                    172800  IN      NS      f.gtld-servers.net.
com.                    172800  IN      NS      a.gtld-servers.net.
com.                    172800  IN      NS      g.gtld-servers.net.
com.                    172800  IN      NS      h.gtld-servers.net.
com.                    172800  IN      NS      l.gtld-servers.net.
com.                    172800  IN      NS      k.gtld-servers.net.
com.                    172800  IN      NS      c.gtld-servers.net.
com.                    172800  IN      NS      d.gtld-servers.net.

;; ADDITIONAL SECTION:
e.gtld-servers.net.     172800  IN      A       192.12.94.30
e.gtld-servers.net.     172800  IN      AAAA    2001:502:1ca1::30
b.gtld-servers.net.     172800  IN      A       192.33.14.30
b.gtld-servers.net.     172800  IN      AAAA    2001:503:231d::2:30
j.gtld-servers.net.     172800  IN      A       192.48.79.30
j.gtld-servers.net.     172800  IN      AAAA    2001:502:7094::30
m.gtld-servers.net.     172800  IN      A       192.55.83.30
m.gtld-servers.net.     172800  IN      AAAA    2001:501:b1f9::30
i.gtld-servers.net.     172800  IN      A       192.43.172.30
i.gtld-servers.net.     172800  IN      AAAA    2001:503:39c1::30
f.gtld-servers.net.     172800  IN      A       192.35.51.30
f.gtld-servers.net.     172800  IN      AAAA    2001:503:d414::30
a.gtld-servers.net.     172800  IN      A       192.5.6.30
a.gtld-servers.net.     172800  IN      AAAA    2001:503:a83e::2:30
g.gtld-servers.net.     172800  IN      A       192.42.93.30
g.gtld-servers.net.     172800  IN      AAAA    2001:503:eea3::30
h.gtld-servers.net.     172800  IN      A       192.54.112.30
h.gtld-servers.net.     172800  IN      AAAA    2001:502:8cc::30
l.gtld-servers.net.     172800  IN      A       192.41.162.30
l.gtld-servers.net.     172800  IN      AAAA    2001:500:d937::30
k.gtld-servers.net.     172800  IN      A       192.52.178.30
k.gtld-servers.net.     172800  IN      AAAA    2001:503:d2d::30
c.gtld-servers.net.     172800  IN      A       192.26.92.30
c.gtld-servers.net.     172800  IN      AAAA    2001:503:83eb::30
d.gtld-servers.net.     172800  IN      A       192.31.80.30
d.gtld-servers.net.     172800  IN      AAAA    2001:500:856e::30

;; Query time: 6 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Mon Mar 07 20:06:47 CST 2022
;; MSG SIZE  rcvd: 845
```

```
f.gtld-servers.net.     172800  IN      A       192.35.51.30
f.gtld-servers.net.     172800  IN      AAAA    2001:503:d414::30
a.gtld-servers.net.     172800  IN      A       192.5.6.30
a.gtld-servers.net.     172800  IN      AAAA    2001:503:a83e::2:30
g.gtld-servers.net.     172800  IN      A       192.42.93.30
g.gtld-servers.net.     172800  IN      AAAA    2001:503:eea3::30
h.gtld-servers.net.     172800  IN      A       192.54.112.30
h.gtld-servers.net.     172800  IN      AAAA    2001:502:8cc::30
l.gtld-servers.net.     172800  IN      A       192.41.162.30
l.gtld-servers.net.     172800  IN      AAAA    2001:500:d937::30
k.gtld-servers.net.     172800  IN      A       192.52.178.30
k.gtld-servers.net.     172800  IN      AAAA    2001:503:d2d::30
c.gtld-servers.net.     172800  IN      A       192.26.92.30
c.gtld-servers.net.     172800  IN      AAAA    2001:503:83eb::30
d.gtld-servers.net.     172800  IN      A       192.31.80.30
d.gtld-servers.net.     172800  IN      AAAA    2001:500:856e::30

;; Query time: 6 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Mon Mar 07 20:06:47 CST 2022
;; MSG SIZE  rcvd: 845

(base) bill@bill-Lenovo-Legion-7-15IMH05:~$ dig mlteecs.wpengine.com @e.gtld-servers.net +norec

; <<>> DiG 9.11.3-1ubuntu1.16-Ubuntu <<>> mlteecs.wpengine.com @e.gtld-servers.net +norec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61606
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 13

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;mlteecs.wpengine.com.          IN      A

;; AUTHORITY SECTION:
wpengine.com.           172800  IN      NS      jim.ns.cloudflare.com.
wpengine.com.           172800  IN      NS      kara.ns.cloudflare.com.

;; ADDITIONAL SECTION:
jim.ns.cloudflare.com.  172800  IN      A       108.162.193.125
jim.ns.cloudflare.com.  172800  IN      A       172.64.33.125
jim.ns.cloudflare.com.  172800  IN      A       173.245.59.125
jim.ns.cloudflare.com.  172800  IN      AAAA    2606:4700:58::adf5:3b7d
jim.ns.cloudflare.com.  172800  IN      AAAA    2803:f800:50::6ca2:c17d
jim.ns.cloudflare.com.  172800  IN      AAAA    2a06:98c1:58::ac40:217d
kara.ns.cloudflare.com. 172800  IN      A       108.162.192.123
kara.ns.cloudflare.com. 172800  IN      A       172.64.32.123
kara.ns.cloudflare.com. 172800  IN      A       173.245.58.123
kara.ns.cloudflare.com. 172800  IN      AAAA    2606:4700:50::adf5:3a7b
kara.ns.cloudflare.com. 172800  IN      AAAA    2803:f800:50::6ca2:c07b
kara.ns.cloudflare.com. 172800  IN      AAAA    2a06:98c1:50::ac40:207b

;; Query time: 26 msec
;; SERVER: 192.12.94.30#53(192.12.94.30)
;; WHEN: Mon Mar 07 20:07:02 CST 2022
;; MSG SIZE  rcvd: 364

(base) bill@bill-Lenovo-Legion-7-15IMH05:~$
```

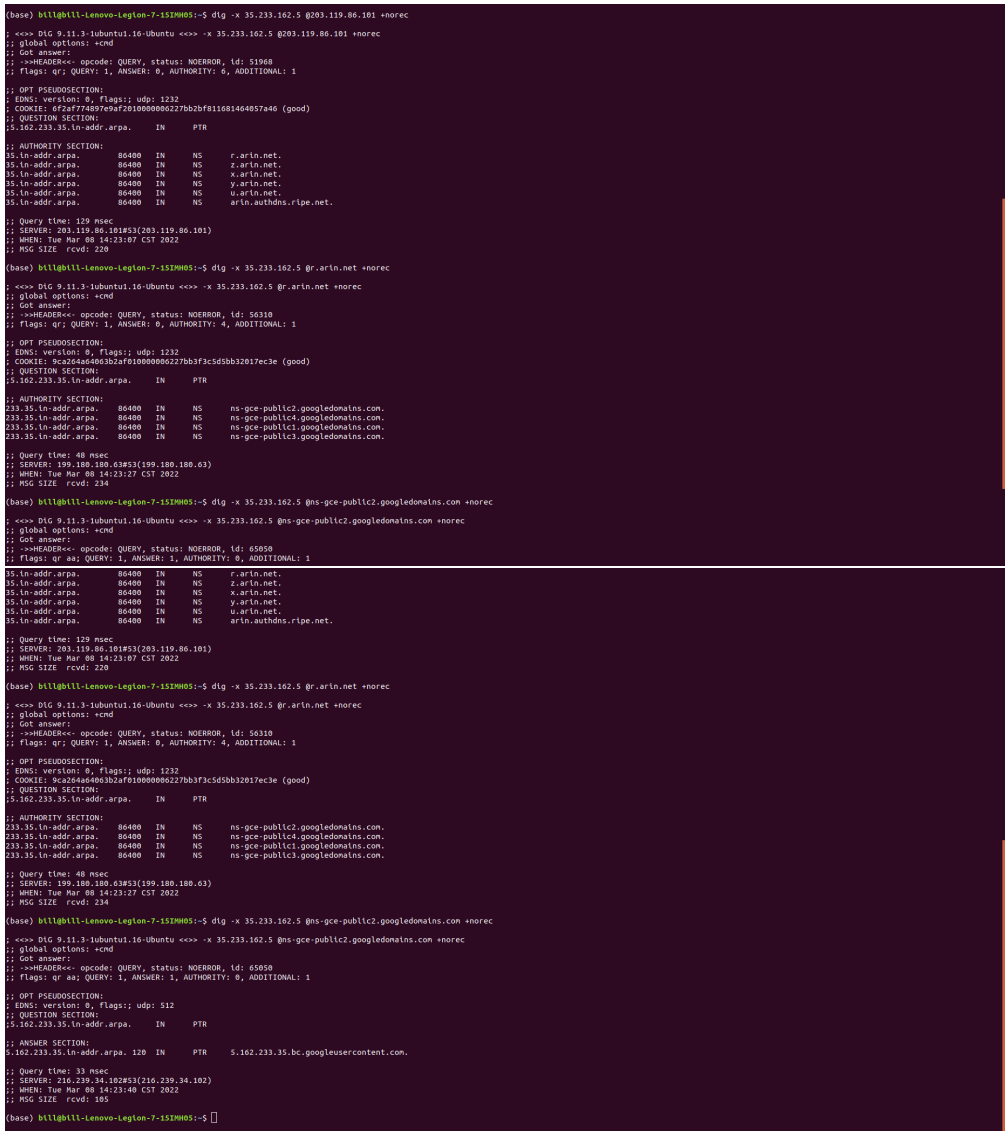
```
(base) bill@bill-Lenovo-Legion-7-15IMH05:~$ dig mlteecs.wpengine.com @e.gtld-servers.net +norec

; <<>> DiG 9.11.3-1ubuntu1.16-Ubuntu <<>> mlteecs.wpengine.com @e.gtld-servers.net +norec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 61606
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 13

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;mlteecs.wpengine.com.          IN      A

;; AUTHORITY SECTION:
wpengine.com.           172800  IN      NS      jin.ns.cloudflare.com.
wpengine.com.           172800  IN      NS      kara.ns.cloudflare.com.

;; ADDITIONAL SECTION:
jin.ns.cloudflare.com.  172800  IN      A       108.162.193.125
jin.ns.cloudflare.com.  172800  IN      A       172.64.33.125
jin.ns.cloudflare.com.  172800  IN      A       173.245.59.125
jin.ns.cloudflare.com.  172800  IN      AAAA    2606:4700:58::adf5:3b7d
jin.ns.cloudflare.com.  172800  IN      AAAA    2803:f800:50::6ca2:c17d
jin.ns.cloudflare.com.  172800  IN      AAAA    2a06:98c1:50::ac40:217d
kara.ns.cloudflare.com. 172800  IN      A       108.162.192.123
kara.ns.cloudflare.com. 172800  IN      A       172.64.32.123
kara.ns.cloudflare.com. 172800  IN      A       173.245.58.123
kara.ns.cloudflare.com. 172800  IN      AAAA    2606:4700:58::adf5:3a7b
kara.ns.cloudflare.com. 172800  IN      AAAA    2803:f800:50::6ca2:c07b
kara.ns.cloudflare.com. 172800  IN      AAAA    2a06:98c1:50::ac40:207b

;; Query time: 26 msec
;; SERVER: 192.12.94.30#53(192.12.94.30)
;; WHEN: Mon Mar 07 20:07:02 CST 2022
;; MSG SIZE  rcvd: 364

(base) bill@bill-Lenovo-Legion-7-15IMH05:~$ dig mlteecs.wpengine.com @jin.ns.cloudflare.com +norec

; <<>> DiG 9.11.3-1ubuntu1.16-Ubuntu <<>> mlteecs.wpengine.com @jin.ns.cloudflare.com +norec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30520
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;mlteecs.wpengine.com.          IN      A

;; ANSWER SECTION:
mlteecs.wpengine.com.   120     IN      A       35.231.163.5

;; Query time: 10 msec
;; SERVER: 108.162.193.125#53(108.162.193.125)
;; WHEN: Mon Mar 07 20:07:23 CST 2022
;; MSG SIZE  rcvd: 65

(base) bill@bill-Lenovo-Legion-7-15IMH05:~$
```

(a) is the domain name after "@" in the command, (b) is provided in the "SERVER:" line in the end of the response, (c) is provided in the "Query time" line, (e) is provided in the second column of both "ANSWER" and "AUTHORITY" section.

2. If it's slower it's likely because going to the resolver takes more time. If it's faster it's likely because cached data are being used.

3. The information is at the same location as in question 1.

```
(base) bill@bill-Lenovo-Legion-7-15IMH05:~$ dig -x 35.233.162.5 @a.root-servers.net +norec

; <<>> DiG 9.11.3-1ubuntu1.16-Ubuntu <<>> -x 35.233.162.5 @a.root-servers.net +norec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18125
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 13

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;5.162.233.35.in-addr.arpa.     IN      PTR

;; AUTHORITY SECTION:
in-addr.arpa.           172800  IN      NS      e.in-addr-servers.arpa.
in-addr.arpa.           172800  IN      NS      f.in-addr-servers.arpa.
in-addr.arpa.           172800  IN      NS      d.in-addr-servers.arpa.
in-addr.arpa.           172800  IN      NS      c.in-addr-servers.arpa.
in-addr.arpa.           172800  IN      NS      b.in-addr-servers.arpa.
in-addr.arpa.           172800  IN      NS      a.in-addr-servers.arpa.

;; ADDITIONAL SECTION:
e.in-addr-servers.arpa. 172800  IN      A       203.119.86.101
e.in-addr-servers.arpa. 172800  IN      AAAA    2001:dd8:6::101
f.in-addr-servers.arpa. 172800  IN      A       193.0.9.1
f.in-addr-servers.arpa. 172800  IN      AAAA    2001:67c:e0::1
d.in-addr-servers.arpa. 172800  IN      A       200.10.60.53
d.in-addr-servers.arpa. 172800  IN      AAAA    2001:13c7:7010::53
c.in-addr-servers.arpa. 172800  IN      A       196.216.169.10
c.in-addr-servers.arpa. 172800  IN      AAAA    2001:43f8:110::10
b.in-addr-servers.arpa. 172800  IN      A       199.253.183.183
b.in-addr-servers.arpa. 172800  IN      AAAA    2001:500:87::87
a.in-addr-servers.arpa. 172800  IN      A       199.180.182.53
a.in-addr-servers.arpa. 172800  IN      AAAA    2620:37:e000::53

;; Query time: 7 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Tue Mar 08 14:22:50 CST 2022
;; MSG SIZE  rcvd: 430

(base) bill@bill-Lenovo-Legion-7-15IMH05:~$ dig -x 35.233.162.5 @203.119.86.101 +norec

; <<>> DiG 9.11.3-1ubuntu1.16-Ubuntu <<>> -x 35.233.162.5 @203.119.86.101 +norec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51968
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 6f2af774897e9af201000086227bb2bf811681464857a46 (good)
;; QUESTION SECTION:
;5.162.233.35.in-addr.arpa.     IN      PTR

;; AUTHORITY SECTION:
35.in-addr.arpa.        86400   IN      NS      r.arin.net.
35.in-addr.arpa.        86400   IN      NS      z.arin.net.
35.in-addr.arpa.        86400   IN      NS      x.arin.net.
35.in-addr.arpa.        86400   IN      NS      y.arin.net.
35.in-addr.arpa.        86400   IN      NS      u.arin.net.
```



4. TCP would require a connection establishment part for each query, so you would have to add an RTT to each of the steps in the lookup. OR There are too much overhead and delay of establishing TCP connection (3-wary handshake, which will be covered in later lectures) for small packets of DNS. Also, UDP does not have to keep connection states, which allows the server to support many more active clients than using TCP.

2 Client-Server - 14 points

Think about spreading a F-bit file among N peers using a client-server structure. Let the server have an maximum upload capacity μ_s, and each client c has a download capacity d_c. Let $d_{min} = \min_c d_c$ be the minimum download rate. Assume that the server can serve multiple clients simultaneously and fluidly set the rate for each client, r_c, as long as $\forall_c r_c \leq d_c$ and $\sum_c r_c \leq \mu_s$.

1. Suppose that $\mu_s/N \leq d_{min}$. How would you set the rates r_c so that the file is fully

distributed to all clients in a minimum time? (i.e., you are minimizing the time that the slowest client receives the file.) What would the distribution time be?

2. Suppose now that $\mu_s/N > d_{min}$. How would you set the rates r_c now to fully distribute the file to the clients in a minimum time? And what would this time be?

3. Consider a concrete example with 5 clients with $d_c = \{06, 12, 18, 24, 30\}$ and a server upload capacity of $\mu_s = 30$. How would you set the rates to get the smallest average download time *without* increasing the total distribution time from the previous part?

Solution: (4+4+6 points)

1. The time required to distribute a file of F bits to N peers is given by

$$max\big(NF/\mu_s, F/d_{min}\big)$$

Since here we have $\mu_s/N \leq d_{min}$, this implies $NF/\mu_s \geq F/d_{min}$. Hence the distribution time will be NF/μ_s, with each client being set a rate of μ_s/N, i.e. $r_c = \mu_s/N$ for all c.

2. In this case we have $\mu_s/N > d_{min}$. Hence $NF/\mu_s < F/d_{min}$, and the minimum time required to distribute the file will be F/d_{min}. Hence the distribution time here is being defined by the bottleneck client and we need to ensure that the bottleneck client is set to maximum capacity, i.e. d_{min}. All other clients, as long as they are faster than d_{min}, will not affect the distribution time. Hence the valid client rate setting would be: $\forall_c r_c \geq r_{min} = d_{min}$ (where $\forall_c r_c \leq d_c$ and $\sum_c r_c \leq \mu_s$).

3. In this situation, we have $\mu_s/N \leq d_{min}$ and therefore, $r_c = \{r_0 = 06, r_1 = 06, r_2 = 06, r_3 = 06, r_4 = 06\}$.

3 BitTorrent - 16 points

BitTorrent employs a choking mechanism for distributing bandwidth to peers. It selects the four peers that have provided the best download performance using a tit-for-tat strategy and one additional peer chosen randomly, referred to as an "optimistic unchoking". The selection of the four best peers and the random choice are updated every second.

Suppose Bob joins a BitTorrent swarm with 30 other peers. Let each peer, including Bob, have an upload speed of 80 $Mbps$ and unlimited download speed. Consider what happens in the phase of the protocol where for each pair of peers A and B, A has some blocks that B wants and vice versa; i.e, any peer can productively download data from any other peer.

1. If Bob intends to not upload any data and become a free rider, what would his average download speed be?

2. What would be the average download rate for the remaining peers?

3. Suppose that Bob runs a second client that pretends to be a separate peer, who also becomes a free rider. How fast can Bob download data now?

4. Suppose Bob switches his two clients to the regular BitTorrent code and they both start uploading as well. Will Bob's performance see a noticeable improvement compared to when he was not contributing?

Solution: 4+4+4+4

1. Each peer will unchoke Bob with probability 1/26 as he will never come in top 4 as he is a free rider. When Bob is unchoked, he will receive 1/5 of the bandwidth for that peer, and as there are 30 other peers his average download speed will be

$$1/26 \times 80\,\text{Mbps}/5 \times 30 \approx 18.46\,\text{Mbps}$$

2. Note that the peers are uploading an aggregate 2400 Mbps, of which 18.46 goes to Bob, so the remainder gets downloaded by the honest peers, for an average of

$$(2400 - 18.46)/30 \approx 79.38\,\text{Mbps}$$

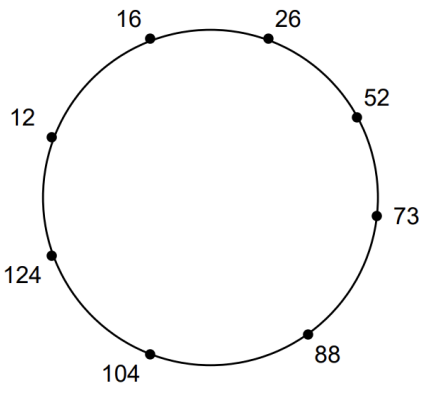
3. We replace 1/26 by 2/27 in the calculation:

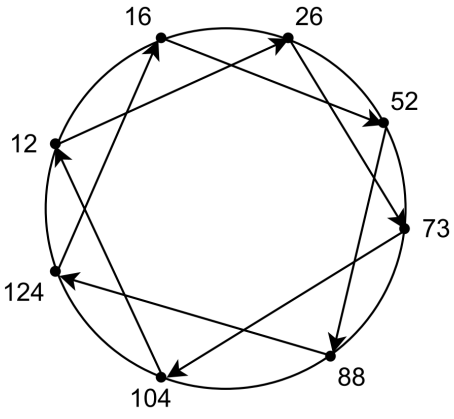
$$2/27 \times 80\,\text{Mbps}/5 \times 30 \approx 35.55\,\text{Mbps}$$

4. He won't do much better because his two clients will have to share the 80Mbps bandwidth that Bob has. Thus, each client will have only half the uploading capacity that a normal client has. His clients will rarely be chosen in the tit-for-tat mechanism.

4 DHT - 16 points

Consider a distributed hash table (DHT) with 8 peers and an 8-bit hash ID that operates in a circular fashion. The peers are represented by the values $\{12, 16, 26, 52, 73, 88, 104, 124\}$. The key-value pairs are assigned to the peer that immediately follows the key in the circular DHT.





1. What is hashing ID space?
2. In which peer is the key-value pair stored in the following circumstances:
   (a) The key has a value of 19?
   (b) The key has a value of 125?
3. If peer 88 initiates a search for the value associated with key 23, as shown in the illustration on the left, how many messages will be necessary to complete the query? This includes the message that initiates the query and the message that returns the value. Provide an explanation for the answer?
4. Assume each peer is given one cord and is now aware of the next two successors as shown in the right figure above. If peer 12 starts a query for the value associated with key 111, then how many messages (including the query message and return value message) are required to resolve this query? Explain why?
5. What is the maximum number of messages to resolve any query if
   (a) each peer is aware of its immediate successor as shown in the left figure above?
   (b) each peer is given one cord and is now aware of the next two successors?
6. Given that each peer is provided with one cord, can you propose a new arrangement of the cords that would minimize the maximum number of messages needed to resolve any query? Draw a representation of the circular distributed hash table and provide an argument to support why your solution is effective.

Solution: 2+2+3+3+2+4

1. $\{0, 1, 2, ..., 255\}$
2. (a) 26
   (b) 12
3. 6, $88 \rightarrow 104 \rightarrow 124 \rightarrow 12 \rightarrow 16 \rightarrow 26 \rightarrow 88$
4. 5, $12 \rightarrow 26 \rightarrow 73 \rightarrow 104 \rightarrow 124 \rightarrow 12$
5. (a) 8
   (b) 5
6. It is possible for multiple solutions to meet the criteria of being correct.

5 HTTP Connections - 8 points

1. How does a web server handle multiple HTTP connections at the same time, even though all of the data packets received from different sessions are all directed to its

TCP port 80? In other words, how does the server ensure that each packet is correctly delivered to its corresponding socket?

2. Is there a restriction on the number of simultaneous connections a single host can make with a web server, such as "a host can only maintain one HTTP connection with the server at any given time"? If such limitations exist, please describe them. If not, please explain the reasons why there are no limitations.

Solution: (4+4 points)

1. In TCP, the sockets are uniquely identified by the following tuple - (*Source IP Address*, *Source Port*, *Destination IP Address*, *Destination Port*). Although multiple HTTP connections on the web server all have the same *Destination IP Address* and *Destination Port*, all the connections can be distinguished because they have different *Source IP Address* and *Source Port*. Hence, although packets from different clients may arrive at the same port on the web server, they will be directed to the correct socket since there is only one socket that corresponsds to the source IP address and source port of the packet.

2. No, there is no such limitation. A single host can have multiple HTTP connections to a server. This is because each connection will be associated with a different socket on the web server. These sockets can be distinguished by their *Source Port* fields, and hence again the packets will be correctly delivered to the corresponding socket.

6 Reliable Data Transfer – 12 points

1. How can the maximum utilization of the network be calculated, when Alice is sending data to Bob using the stop-and-wait method, given that the bottleneck bandwidth is 40 Mbps and the end-to-end delay is 25 ms, and each packet being sent is 1200 bytes with 32 bits reserved for the checksum and sequence number?

2. How should Alice and Bob set their timeout value?

3. What changes would you make to the protocols if the connection between Alice and Bob is ensured to not cause any packet damage?

4. What would be the impact on performance of this change?

Solution: 2+2+2+2

1. Time to send a packet is: $t_1 = \frac{1200bytes}{40Mbps} + 25ms$ Time to send an ack is: $t_2 = \frac{4byte}{40Mbps} + 25ms$ Overall sending rate: $r = \frac{1196bytes}{t_1+t_2}$ Utilization: $u = \frac{r}{40Mbps}$

2. They should set it to the RTT approx (50 ms) plus some extra buffer value. The RTT can be measured (in which case it will come out to closer to $t_1 + t_2$ from above), the

buffer value needs to be big enough to deal with any (typical) variation in processing and queueing delays.

3. There is no need to use checksums. A number of people offered other suggestions, like using selective ACK or go- back-N. We marked this wrong because whether you upgrade from stop-and-wait to these other protocols isn't really affected by whether the channel corrupts data. Some said we could get rid of sequence numbers, but this is still necessary in case of packet loss.

4. Less processing delay and less packet header overhead that was previously used for the checksum.