EE122–Fall 2013 — Solutions to Homework 1

Alexander Tom (ID: 20964861), ee122-ki

December 2, 2013

Q1.

(a.)
```
; <<>> DiG 9.8.1-P1 <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23171
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 4, ADDITIONAL: 4

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

;; ANSWER SECTION:
www.google.com. 268 IN A 74.125.239.112
www.google.com. 268 IN A 74.125.239.116
www.google.com. 268 IN A 74.125.239.113
www.google.com. 268 IN A 74.125.239.115
www.google.com. 268 IN A 74.125.239.114

;; AUTHORITY SECTION:
google.com. 14484 IN NS ns1.google.com.
google.com. 14484 IN NS ns3.google.com.
google.com. 14484 IN NS ns2.google.com.
google.com. 14484 IN NS ns4.google.com.

;; ADDITIONAL SECTION:
ns2.google.com. 263358 IN A 216.239.34.10
ns3.google.com. 264248 IN A 216.239.36.10
ns1.google.com. 264248 IN A 216.239.32.10
ns4.google.com. 263358 IN A 216.239.38.10
```

```
;; Query time: 1 msec
;; SERVER: 128.32.112.21#53(128.32.112.21)
;; WHEN: Sun Dec  1 13:4
```

The first field is the NAME which is the name of the domain in question being returned. The second field is the TTL. The third field is the CLASS of the records being requested which in our case is IN for internet. The forth field is the TYPE which determines the record type which in our case is A which stands for a mapping a domain name to an IPv4 address. The last field is the IP address.

(b.)

```
; <<>> DiG 9.8.3-P1 <<>> google.com +trace
;; global options: +cmd
. 413209 IN NS l.root-servers.net.
. 413209 IN NS j.root-servers.net.
. 413209 IN NS e.root-servers.net.
. 413209 IN NS g.root-servers.net.
. 413209 IN NS k.root-servers.net.
. 413209 IN NS d.root-servers.net.
. 413209 IN NS b.root-servers.net.
. 413209 IN NS i.root-servers.net.
. 413209 IN NS a.root-servers.net.
. 413209 IN NS h.root-servers.net.
. 413209 IN NS c.root-servers.net.
. 413209 IN NS f.root-servers.net.
. 413209 IN NS m.root-servers.net.
;; Received 344 bytes from 128.32.136.9#53(128.32.136.9) in 780 ms

com. 172800 IN NS k.gtld-servers.net.
com. 172800 IN NS f.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 a.gtld-servers.net.
com. 172800 IN NS d.gtld-servers.net.
com. 172800 IN NS c.gtld-servers.net.
com. 172800 IN NS m.gtld-servers.net.
com. 172800 IN NS i.gtld-servers.net.
com. 172800 IN NS j.gtld-servers.net.
com. 172800 IN NS b.gtld-servers.net.
com. 172800 IN NS e.gtld-servers.net.
```

```
;; Received 488 bytes from 192.112.36.4#53(192.112.36.4) in 488 ms

google.com. 172800 IN NS ns2.google.com.
google.com. 172800 IN NS ns1.google.com.
google.com. 172800 IN NS ns3.google.com.
google.com. 172800 IN NS ns4.google.com.
;; Received 164 bytes from 192.41.162.30#53(192.41.162.30) in 69 ms

google.com. 300 IN A 74.125.239.100
google.com. 300 IN A 74.125.239.104
google.com. 300 IN A 74.125.239.99
google.com. 300 IN A 74.125.239.110
google.com. 300 IN A 74.125.239.105
google.com. 300 IN A 74.125.239.102
google.com. 300 IN A 74.125.239.103
google.com. 300 IN A 74.125.239.96
google.com. 300 IN A 74.125.239.97
google.com. 300 IN A 74.125.239.101
google.com. 300 IN A 74.125.239.98
;; Received 204 bytes from 216.239.34.10#53(216.239.34.10) in 64 ms
```

Sequence of the name servers queried and domain each server was responsible for:

```
Name server, Domain Server resonsible for
*.root-servers.net, .
*.gtld-servers.net, com.
ns*.google.com, google.com.
74.125.239.*, google.com.
```

(c.) DNS Query using ns1.ii.kgp.ac.in:

```
; <<>> DiG 9.8.1-P1 <<>> www.google.com @ns1.iitkgp.ac.in
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20675
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 4, ADDITIONAL: 1

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

```
;; ANSWER SECTION:
www.google.com. 100 IN A 74.125.236.210
www.google.com. 100 IN A 74.125.236.211
www.google.com. 100 IN A 74.125.236.208
www.google.com. 100 IN A 74.125.236.209
www.google.com. 100 IN A 74.125.236.212

;; AUTHORITY SECTION:
google.com. 156744 IN NS ns4.google.com.
google.com. 156744 IN NS ns2.google.com.
oogle.com. 156744 IN NS ns1.google.com.
google.com. 156744 IN NS ns3.google.com.

;; ADDITIONAL SECTION:
ns1.google.com. 6138 IN A 216.239.32.10

;; Query time: 292 msec
;; SERVER: 203.110.245.241#53(203.110.245.241)
;; WHEN: Sun Dec  1 14:28:20 2013
;; MSG SIZE  rcvd: 200
```

DNS Query using nsl.fujitsu.fr:

```
; <<>> DiG 9.8.1-P1 <<>> www.google.com @nsl.fujitsu.fr
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59463
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 4, ADDITIONAL: 4

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

;; ANSWER SECTION:
www.google.com. 258 IN A 173.194.40.180
www.google.com. 258 IN A 173.194.40.176
www.google.com. 258 IN A 173.194.40.177
www.google.com. 258 IN A 173.194.40.178
www.google.com. 258 IN A 173.194.40.179

;; AUTHORITY SECTION:
```

```
google.com. 308514 IN NS ns1.google.com.
google.com. 308514 IN NS ns2.google.com.
google.com. 308514 IN NS ns3.google.com.
google.com. 308514 IN NS ns4.google.com.

;; ADDITIONAL SECTION:
ns1.google.com. 62117 IN A 216.239.32.10
ns2.google.com. 62117 IN A 216.239.34.10
ns3.google.com. 62117 IN A 216.239.36.10
ns4.google.com. 62117 IN A 216.239.38.10

;; Query time: 165 msec
;; SERVER: 62.244.109.18#53(62.244.109.18)
;; WHEN: Sun Dec  1 14:29:27 2013
;; MSG SIZE  rcvd: 248
```

We see a difference in latency because the latancy is directly propotional to the distance to the DNS server. nsl.fujitsu.fr is closer than ns1.iitkgp.ac.in. We can prove this by looking at the prefix of the ip address of the name servers. nsl.fujitsu.fr goes to a DNS server whose IP address is 62.244.109.18 which is in France and ns1.iitkgp.ac.in maps to a DNS server whose IP address is 203.110.245.241 which is in India. When we dont specify a DNS server, the DNS server is 128.32.112.21 which is right in Berkeley.

(d.) Garry's DNS server must return the following query to redirect traffic to his eavilsearch.com:

```
For Type = A, the DNS server should return:
name = google.com
value = <IP address of evilsearch.com>
ttl = <ttl value less than the ttl to google.com>

For Type = NS, the DNS server should return:
name = google.com
value = <dns server of evilsearch.com>
ttl = <ttl value less than the ttl to google.com>
```

To prevent this attack, the DNS server can look at the IP address of the DNS servers and verify they all match since google has multple dns servers. You can also ping multiple DNS servers and verify that there is overlap between the results.

Q2.

(a.) **10R**

(b.) **6R**

(c.) **2R + 2 * 4/4R = 4R**

(d.) **2R + 2 * 4 * 0.5R = 6R**

(e.) **2R + 0.5R + 4 * 0.5R = 4.5R**

(f.) **2R + 2 * 4/4R * 0.5 = 3R**

(g.) **2R + 2 * 3/4R + 2 * 1/2R + 2 * 1/3R + 2 * 1/4R = 5.666R**

(h.) **2R + 2 * 3/4R + 2 * 1/2R + 2 * 1/3R + 2 * 1/4R = 5.666R**

(i.) **2R + max{2 * 3/4R, 2 * 1/2R, 2 * 1/3R, 2 * 1/4R} = 3.5R**

Q3.

(a.)
- **Case 1:**
  - (i.) Yes the transmission is successful and the original transmission is NOT impacted. B can listen to more than one node and B is not sending anything.
  - (ii.) No, A cannot transmit to B because B is already receiving from E. MACA will not send a CTS until the transmission between E and B is done.
- **Case 2:**
  - (i.) Yes the transmission is successful and the original transmission is NOT impacted.
  - (ii.) Yes the transmission is successful and the original transmission is NOT impacted.
- **Case 3:**
  - (i.) Yes the transmission is successful and the original transmission is NOT impacted.
  - (ii.) No, the transmission is not successful because A overheard a CTS from B.

(b.)
- **Case 1:**
  - (i.) No, A cannot transmit to B because B is already transmitting to E.
  - (ii.) No, A cannot transmit to B because B is already sending to E. MACA will not send a CTS until the transmission between E and B is done.
- **Case 2:**
  - (i.) Yes the transmission is successful and the original transmission is NOT impacted.
  - (ii.) No, F doesn't hear an CTS back from C because C heard an CTS from B.
- **Case 3:**
  - (i.) No, C needs to wait for B to finish transmitting.
  - (ii.) No, C and A don't hear a CTS. B is transmitting data and A won't hear an RTS from C.

(c.)
- (i.) Nothing can successfully communicate. Every node is waiting for A to be done transmitting to B.
- (ii.) Nothing can successfully communicate. Every node is waiting for an RTS from A.

(d.) Using CS, yes, the nodes can successfully transmit because D, E, and F never hear someone else transmitting. A, B, and C hear transmitting but that information doesn't propogate to D, E, F.

Using MACA, no, the nodes cannot transmit all at once because if any one of A, B, or C begins transmitting, the other two nodes overheard a CTS and have to wait for the transmission to end.

(e.) Using CS, (D, A), (E, B), and (F, C) can simultaneously communicate. Each edge won't see each other transmitting, so both sides on the edge can begin transmitting simultaneously.
Using MACA, (D, A), (E, B), and (F, C) can simultaneously communicate. If both sides of the edge send a RTS at the same time, they will both send back a CTS and begin transmitting.

Q4.

(a.) (i.) **Root: 0; Edges: (4, 1), (1, 0), (0, 2), (2, 3), (3, 5)**

(ii.) **Root: 1; Edges: (1, 2), (2, 3), (1, 4), (4, 5)**

(b.) (1.) **Switches: 0, 1, 2, 3, 4, 5; End-hosts: a, c, d, e, f, g**

(2.) **Switches: 2, 0, 1; End-hosts: b**

(3.) **Switchea: 2, 3, 5; End-hosts: c, f**

(4.) **Switches: 0, 1; End-hosts: b**

(5.) **Switches: 0, 1, 2, 3, 4, 5; End-hosts: b, c, d, e, f, g**

(c.) (1.) **Flood**

(2.) **Unicast**

(3.) **Flood**

(4.) **Unicast**

(5.) **Unicast**

(6.) **Flood**

(7.) **Unicast**

(8.) **Unicast**

(9.) **Flood**

(10.) **Unicast**

(11.) **Unicast**

(12.) **Flood**

(i.) (a to b): 0/4 flooded, 4/4 Unicasted; (b to c): 1/4 flooded, 3/4 Unicasted; (c to b): 4/4 Flooded, 0/4 Unicasted

(ii.) Swap 8 and 9 and swap 11 and 12