Rob Johansen
u0531837
CS 4480 - Homework Assignment 3

P3

At the application layer, DNS is needed in order to look up the IP address of the HTTP server. At the transport layer, UDP is needed in order to support the DNS lookup. TCP is also needed at the transport layer to facilitate the HTTP request.

P7

The total elapsed time can be calculated by summing all the RTT for DNS ($RTT_1...RTT_n$), along with RTT for receiving the object (RTT_0). This calculation can be expressed as follows:

$$\sum_{i=1}^{n} RTTi + RTTo$$

P8

If the client is a Windows host, then the RTT for DNS only occurs once because the result is cached by the client host. Thus, the first part of the equation from P7 appears only once in each equation of this problem (as the first operand).

a. With non-persistent HTTP and no parallel TCP connections, the client host must wait for all the DNS RTT ($RTT_1...RTT_n$), then wait for each object RTT separately (9 * RTT_0). This calculation can be expressed as follows:

$$\sum_{i=1}^{n} RTTi + (9 * RTTo)$$

b. With non-persistent HTTP and 5 parallel TCP connections, the client host must wait for all the DNS RTT ($RTT_1...RTT_n$), then wait for only two object RTT separately (2 * RTT_0). This calculation can be expressed as

follows:

$$\sum_{i=1}^{n} RTTi + (2 * RTTo)$$

c. With persistent HTTP and the stated assumption of zero transmission time, the client can effectively request all 9 objects at once. Thus the client host only waits for the DNS RTT ($RTT_1 ... RTT_n$), plus the time of one object RTT (RTT_0). This calculation can be expressed as follows (the same as P7):

$$\sum_{i=1}^{n} RTTi + RTTo$$

P18

a. A database that stores information about the registered user(s) of a domain name, block of IP address, or some other such Internet resource.

b. I used the whois.net database to look up the DNS servers for the utahrealestate.com domain:

NS1.WFRMLS.COM
NS2.WFRMLS.COM

I also used the networksolutions.com database to look up the DNS servers for the fidelity.com domain:

FIDNS2.FIDELITY.COM
FIDNS3.FIDELITY.COM

c. Unfortunately the remote DNS servers from part (b) refused all my nslookup queries. Thus I used my local DNS server, Google's DNS server, and the OpenDNS server. The following summarizes my findings for the google.com domain:

Type A (local DNS server):
74.125.239.134
74.125.239.136
74.125.239.129

74.125.239.142
74.125.239.137
74.125.239.130
74.125.239.132
74.125.239.133
74.125.239.128
74.125.239.131

Type NS (Google DNS server):

ns1.google.com internet address = 216.239.32.10
ns4.google.com internet address = 216.239.38.10
ns3.google.com internet address = 216.239.36.10
ns2.google.com internet address = 216.239.34.10

Type MX (OpenDNS server):

google.com MX preference = 50, mail exchanger = alt4.aspmx.l.google.com
google.com MX preference = 30, mail exchanger = alt2.aspmx.l.google.com
google.com MX preference = 10, mail exchanger = aspmx.l.google.com
google.com MX preference = 20, mail exchanger = alt1.aspmx.l.google.com
google.com MX preference = 40, mail exchanger = alt3.aspmx.l.google.com

d. I used nslookup to find the IP addresses for www.utah.edu. It does have two IP addresses (one IPv6 and the other IPv4):

    2001:1948:414:6::2
    155.97.137.55

e. I searched whois.arin.net/ui for "University of Utah" and found the following IP ranges:

    198.60.30.0 - 198.60.30.255
    198.60.31.0 - 198.60.31.255
    204.99.144.0 - 204.99.147.255
    204.99.159.0 - 204.99.177.255
    204.99.139.0 - 204.99.139.255
    199.104.93.0 - 199.104.93.255
    198.60.23.0 - 198.60.23.255
    205.247.208.128 - 205.247.208.255
    204.99.128.0 - 204.99.128.255
    192.5.12.0 - 192.5.12.255
    192.31.39.0 - 192.31.39.255
    192.12.56.0 - 192.12.56.255
    155.99.0.0 - 155.99.255.255
    155.100.0.0 - 155.100.255.255
    128.110.0.0 - 128.110.255.255

155.101.0.0 - 155.101.255.255
155.98.0.0 - 155.98.255.255
155.97.0.0 - 155.97.255.255

f. An attacker could use nslookup to determine the IP addresses of specific targets to attack remotely (such as web servers, email servers, and DNS server). An attacker could also use whois databases to gain administrator contact information (such as phone number or email address) for a social engineering attack.

g. I think whois databases should be publicly available so that the appropriate registrants and administrators can be contacted regarding their domain (for security reasons, for commerce reasons, and so on).

P19

a. I performed the query "dig @j.root-servers.net www.utah.edu +trace" and got the following list of DNS server names in the delegation chain (along with the actual A record):

```
; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.17.rc1.el6.3 <<>> @j.root-
servers.net www.utah.edu +trace
; (2 servers found)
;; global options: +cmd
.                       518400  IN      NS      h.root-servers.net.
.                       518400  IN      NS      j.root-servers.net.
.                       518400  IN      NS      d.root-servers.net.
.                       518400  IN      NS      e.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
.                       518400  IN      NS      m.root-servers.net.
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      k.root-servers.net.
.                       518400  IN      NS      b.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      f.root-servers.net.
.                       518400  IN      NS      a.root-servers.net.
;; Received 512 bytes from 192.58.128.30#53(192.58.128.30) in 199 ms

edu.                    172800  IN      NS      g.edu-servers.net.
edu.                    172800  IN      NS      f.edu-servers.net.
edu.                    172800  IN      NS      d.edu-servers.net.
edu.                    172800  IN      NS      l.edu-servers.net.
edu.                    172800  IN      NS      a.edu-servers.net.
edu.                    172800  IN      NS      c.edu-servers.net.
```

```
;; Received 265 bytes from 192.112.36.4#53(192.112.36.4) in 53 ms

utah.edu.               172800  IN      NS      fiber.utah.edu.
utah.edu.               172800  IN      NS      ns.utah.edu.
utah.edu.               172800  IN      NS      ns1.utah.edu.
utah.edu.               172800  IN      NS      fiber1.utah.edu.
;; Received 170 bytes from 192.42.93.30#53(192.42.93.30) in 53 ms

www.utah.edu.           300     IN      A       155.97.137.55
utah.edu.               1800    IN      NS      ns.utah.edu.
utah.edu.               1800    IN      NS      fiber.utah.edu.
;; Received 115 bytes from 128.110.124.120#53(128.110.124.120) in 0 ms
```

b. Following are the results I got from digging google.com and amazon.com:

google.com:

```
; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.17.rc1.el6.3 <<>> @j.root-
servers.net www.google.com +trace
; (2 servers found)
;; global options: +cmd
.                       518400  IN      NS      d.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      NS      h.root-servers.net.
.                       518400  IN      NS      b.root-servers.net.
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      j.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      f.root-servers.net.
.                       518400  IN      NS      m.root-servers.net.
.                       518400  IN      NS      e.root-servers.net.
.                       518400  IN      NS      k.root-servers.net.
.                       518400  IN      NS      a.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
;; Received 512 bytes from 192.58.128.30#53(192.58.128.30) in 188 ms

com.                    172800  IN      NS      b.gtld-servers.net.
com.                    172800  IN      NS      g.gtld-servers.net.
com.                    172800  IN      NS      c.gtld-servers.net.
com.                    172800  IN      NS      l.gtld-servers.net.
com.                    172800  IN      NS      i.gtld-servers.net.
com.                    172800  IN      NS      j.gtld-servers.net.
com.                    172800  IN      NS      d.gtld-servers.net.
com.                    172800  IN      NS      h.gtld-servers.net.
com.                    172800  IN      NS      a.gtld-servers.net.
```

```
com.                    172800  IN      NS      k.gtld-servers.net.
com.                    172800  IN      NS      e.gtld-servers.net.
com.                    172800  IN      NS      m.gtld-servers.net.
com.                    172800  IN      NS      f.gtld-servers.net.
;; Received 492 bytes from 199.7.91.13#53(199.7.91.13) in 59 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 168 bytes from 192.41.162.30#53(192.41.162.30) in 252 ms

www.google.com.         300     IN      A       74.125.225.211
www.google.com.         300     IN      A       74.125.225.209
www.google.com.         300     IN      A       74.125.225.210
www.google.com.         300     IN      A       74.125.225.212
www.google.com.         300     IN      A       74.125.225.208
;; Received 112 bytes from 216.239.38.10#53(216.239.38.10) in 50 ms
```

amazon.com:

```
; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.17.rc1.el6.3 <<>> @j.root-
servers.net www.amazon.com +trace
; (2 servers found)
;; global options: +cmd
.                       518400  IN      NS      k.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      m.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      a.root-servers.net.
.                       518400  IN      NS      h.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
.                       518400  IN      NS      d.root-servers.net.
.                       518400  IN      NS      j.root-servers.net.
.                       518400  IN      NS      e.root-servers.net.
.                       518400  IN      NS      f.root-servers.net.
.                       518400  IN      NS      b.root-servers.net.
;; Received 512 bytes from 192.58.128.30#53(192.58.128.30) in 171 ms

com.                    172800  IN      NS      a.gtld-servers.net.
com.                    172800  IN      NS      b.gtld-servers.net.
com.                    172800  IN      NS      c.gtld-servers.net.
com.                    172800  IN      NS      d.gtld-servers.net.
com.                    172800  IN      NS      e.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      i.gtld-servers.net.
com.                    172800  IN      NS      j.gtld-servers.net.
com.                    172800  IN      NS      k.gtld-servers.net.
com.                    172800  IN      NS      l.gtld-servers.net.
com.                    172800  IN      NS      m.gtld-servers.net.
;; Received 492 bytes from 199.7.83.42#53(199.7.83.42) in 263 ms

amazon.com.             172800  IN      NS      pdns3.ultradns.org.
amazon.com.             172800  IN      NS      pdns4.ultradns.org.
amazon.com.             172800  IN      NS      pdns1.ultradns.net.
amazon.com.             172800  IN      NS      pdns2.ultradns.net.
amazon.com.             172800  IN      NS      pdns5.ultradns.info.
amazon.com.             172800  IN      NS      pdns6.ultradns.co.uk.
amazon.com.             172800  IN      NS      ns1.p31.dynect.net.
amazon.com.             172800  IN      NS      ns3.p31.dynect.net.
amazon.com.             172800  IN      NS      ns2.p31.dynect.net.
amazon.com.             172800  IN      NS      ns4.p31.dynect.net.
;; Received 438 bytes from 192.12.94.30#53(192.12.94.30) in 183 ms

www.amazon.com.         907     IN      NS      ns-941.amazon.com.
www.amazon.com.         907     IN      NS      ns-912.amazon.com.
www.amazon.com.         907     IN      NS      ns-922.amazon.com.
www.amazon.com.         907     IN      NS      ns-921.amazon.com.
www.amazon.com.         907     IN      NS      ns-942.amazon.com.
www.amazon.com.         907     IN      NS      ns-924.amazon.com.
www.amazon.com.         907     IN      NS      ns-911.amazon.com.
www.amazon.com.         907     IN      NS      ns-923.amazon.com.
;; Received 328 bytes from 204.13.250.31#53(204.13.250.31) in 161 ms

www.amazon.com.         60      IN      A       72.21.215.232
;; Received 48 bytes from 72.21.208.215#53(72.21.208.215) in 52 ms
```

P26

a. The description of the BitTorrent protocol makes it *sound* like Bob should not be able to free ride. However, I don't think any protocol or system is perfectly bullet-proof. In fact, a quick Google search turned up a number of research papers describing a so-called "Large View Exploit" that enables free riding, as well as modified BitTorrent clients that claim to be able to free ride.

b. If Bob used modified BitTorrent clients that only uploaded chunks to other peers on the same LAN, his claim is plausible.

P28

Peer 6 will begin by sending peer 15 a message:

"What will be peer 6's predecessor and successor?"

This message gets forwarded through the DHT until it reaches peer 5, who realizes that it will be peer 6's predecessor and that its current successor, peer 8 will become peer 6's successor. Next, peer 5 sends this predecessor and successor information to peer 6. Peer 6 can now join the DHT by making peer 8 its successor and by notifying peer 5 that it should change its immediate successor to 6.

P32

It will *not* be necessary to change UDPServer.py.

The port numbers for the sockets in UDPClient and UPDServer are 5432 and 12000, respectively.

Before making this change the UDPClient port was randomly assigned by the operating system, and the UPDServer port was 12000.

Calling the bind() method in this way simply tells the client's socket to use port 5432 instead of the randomly assigned port. When the client subsequently sends its message to the server, this newly bound port (5432) will be attached to the packet automatically by the sendto() method (along with the client's IP address). Thus, the UDPServer will know to send the response back to the client's IP address and port 5432.