**Name: Fan Feng**

**Assignment: 2**

**Course: CPSC 433/533**

**Part 1 (Coding)**

See Java code.

Here is a design note:

I created a class called PingMessage.java that contains all components of the message. PingMessage.java provides 2 APIs, on one that parses a byte array to set class attributes, the other generates byte array according to class attributes. Both take into consideration of white space between components.

This makes code more concise by avoiding duplicate code because otherwise both PingClient.java and PingServer.java need to have the same set of APIs. Even though the two sets of APIs do slightly different things (e.g., PingServer.java needs to convert header from PING to PINGECHO, while PingClient doesn’t need to), there is a significant amount of duplicate code.

I believe even with this design, my program should still work compatibly with the other’s program. Below is what happened when I tried to ping another student’s server.

[ff242@hawk ps2]$ java PingClient hawk.zoo.cs.yale.edu 7000 password

MIN RTT = 2, MAX RTT = 158, AVG RTT = 85.0

LOSS RATE = 0.4

[ff242@hawk ps2]$ java PingClient hawk.zoo.cs.yale.edu 7000 password

MIN RTT = 17, MAX RTT = 187, AVG RTT = 94.88888888888889

LOSS RATE = 0.1

[ff242@hawk ps2]$ java PingClient hawk.zoo.cs.yale.edu 7000 password

MIN RTT = 5, MAX RTT = 132, AVG RTT = 77.375

LOSS RATE = 0.2

**Part 2 (Design Questions)**

**a).**

I tested the program by printing out the ping message on both the client side and the server side. Note below that packet 5 and 8 are lost.

I also tested that if server and client use different port number, no packet is received by the server. In this situation, client reports a loss rate of 100%.

Last but not least, when passwords don’t match, client throws an exception without replying. In this situation, client reports a loss rate of 100% too.

Note that I didn’t use **printData(DatagramPacket packet)** provided in PingServer.java, because not every byte value corresponds to an ASCII character.

[ff242@scorpion ps2]$ java PingClient localhost 2000 123456

PING 0 1538186952119 123456

PINGECHO 0 1538186952119 123456

PING 1 1538186952185 123456

PINGECHO 1 1538186952185 123456

PING 2 1538186952340 123456

PINGECHO 2 1538186952340 123456

PING 3 1538186952530 123456

PINGECHO 3 1538186952530 123456

PING 4 1538186952583 123456

PINGECHO 4 1538186952583 123456

PING 5 1538186952771 123456

PING 6 1538186953772 123456

PINGECHO 6 1538186953772 123456

PING 7 1538186953968 123456

PINGECHO 7 1538186953968 123456

PING 8 1538186954030 123456

PING 9 1538186955031 123456

PINGECHO 9 1538186955031 123456

MIN RTT = 53, MAX RTT = 196, AVG RTT = 120.75

LOSS RATE = 0.2

[ff242@scorpion ps2]$ java PingServer 2000 123456

PING 0 1538186952119 123456

PINGECHO 0 1538186952119 123456

Reply sent.

PING 1 1538186952185 123456

PINGECHO 1 1538186952185 123456

Reply sent.

PING 2 1538186952340 123456

PINGECHO 2 1538186952340 123456

Reply sent.

PING 3 1538186952530 123456

PINGECHO 3 1538186952530 123456

Reply sent.

PING 4 1538186952583 123456

PINGECHO 4 1538186952583 123456

Reply sent.

Reply not sent.

PING 6 1538186953772 123456

PINGECHO 6 1538186953772 123456

Reply sent.

PING 7 1538186953968 123456

PINGECHO 7 1538186953968 123456

Reply sent.

Reply not sent.

PING 9 1538186955031 123456

PINGECHO 9 1538186955031 123456

Reply sent.

* Whenever a PingMessage object parses a byte array to set class attributes, or converts itself to a byte array, it first sets its buffer order to BIG\_ENDIAN. Additionally, all strings read from or written to the buffer ASCII as the charset.
* Based on the paper,

Where and are the arrival times of the first and second packets respectively at the destination, and are the transmission times of the first and second packets respectively. is the size of the second packet, and is the bandwidth of the bottleneck link.

If all variables except are known, then the bandwidth can be calculated based on the above formula. Particularly, if is extremely small, then .

* Yes.

i). Client records the client time when the packet was sent.

ii). Server receives packet, records the current server time .

iii). Server replies. Client receives packet at client time .

iv). The current server time is . Subtract from it gives us the clock difference between server and client.

v). Repeat steps i – iv, compute the average of all obtained clock differences, and use it to adjust the clocks on server or client.

The accuracy depends on the propagation delay. In other words, accuracy is +/- propagation delay, whatever it is.

**b).**

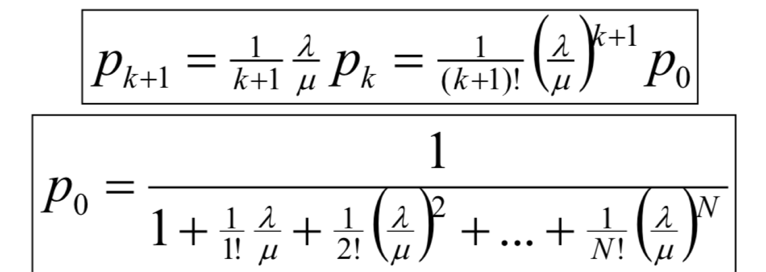
* No, because DNS uses UDP, which doesn’t guarantee reliability. In other words, there is a chance that the query to another server will be lost. Thus, further requests with the same id will not get an answer.

Also, when the DNS requests are extremely frequent, the chance that multiple requests have the same id is high.

* Let λ be the query arrival rate per second.

Since ID in DNS protocol take 2 bytes = 16 bits, the total number of available IDs is .

In Circuit switching, we know the following:



Substitute k+1 with , μ with 1/0.3, and with λ, can be calculated.

* In local DNS cache, search a.b.c.d first. If not found, search b.c.d. If not found, search c.d. If not found, search d.

During each search, if a DNS server is found in the local DNS cache, query a.b.c.d on that DNS server. Otherwise, recursively querying a.b.c.d is needed.

**c).**

Overwhelm the DNS service of the country so that normal traffic can no longer get through.

**Part 3 (Playing with Tools and Observe)**

**a).**

**1).** One gmail server is alt2.gmail-smtp-in.l.google.com

Fans-MacBook-Pro:~ fanfeng$ dig gmail.com MX +short

20 alt2.gmail-smtp-in.l.google.com.

10 alt1.gmail-smtp-in.l.google.com.

30 alt3.gmail-smtp-in.l.google.com.

40 alt4.gmail-smtp-in.l.google.com.

5 gmail-smtp-in.l.google.com.

**2).** The range of IP 173.194.0.0/16 is from 173.193.0.1 to 173.194.255.254. Since 173.194.1.1 is within the range, it is an authorized mail transfer agent for gmail.

Fans-MacBook-Pro:~ fanfeng$ dig gmail.com txt +short

"v=spf1 redirect=\_spf.google.com"

Fans-MacBook-Pro:~ fanfeng$ dig \_spf.google.com txt +short

"v=spf1 include:\_netblocks.google.com include:\_netblocks2.google.com include:\_netblocks3.google.com ~all"

Fans-MacBook-Pro:~ fanfeng$ dig \_netblocks.google.com txt +short

"v=spf1 ip4:35.190.247.0/24 ip4:64.233.160.0/19 ip4:66.102.0.0/20 ip4:66.249.80.0/20 ip4:72.14.192.0/18 ip4:74.125.0.0/16 ip4:108.177.8.0/21 ip4:173.194.0.0/16 ip4:209.85.128.0/17 ip4:216.58.192.0/19 ip4:216.239.32.0/19 ~all"

**3).** Googled to find out how to get the public key. The following command was found.

Fans-MacBook-Pro:PS2 fanfeng$ openssl s\_client -connect pop.gmail.com:995 | openssl x509 -pubkey -noout

depth=1 C = US, O = Google Trust Services, CN = Google Internet Authority G3

verify error:num=20:unable to get local issuer certificate

verify return:0

-----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwaOf95/73E2uNewPHr/F

fPX+KBypwK41PgxZuvU0n19lhH9C4ctzyjsQonQM4E5MQPjhBHupDbrLVK1eWE7r

8PRPPAKaq+xM75mQ1QcCc+2D2kUonwVYWUFB/irJE3U8Ff9RFw7CtUvBeO8cSBlo

ruarIi4LIS7Rgl11jPAfrccjD8HKjJbRlRFgFh2l77Nr2dnmaXECMsgvrGAp4aUB

36AZM27Hm9Rt40F6QdmChQyy3/DN5SDWDUoXZEUFUiqgRgzva0+Z0QoAYI1QiJbn

Ab35mKBOfdl7SCenq1CWklk9q8rApStB2O7c3gEhjleXgc6lnfSWvozTXmlNbAmK

1QIDAQAB

-----END PUBLIC KEY-----

**b).** 128.36.232.5

Fans-MacBook-Pro:~ fanfeng$ dig +norecurse @a.root-servers.net cicada.cs.yale.edu A

; <<>> DiG 9.10.6 <<>> +norecurse @a.root-servers.net cicada.cs.yale.edu A

; (1 server found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51084

;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 8

;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags:; udp: 4096

;; QUESTION SECTION:

;cicada.cs.yale.edu. IN A

;; AUTHORITY SECTION:

edu. 172800 IN NS f.edu-servers.net.

edu. 172800 IN NS a.edu-servers.net.

edu. 172800 IN NS g.edu-servers.net.

edu. 172800 IN NS l.edu-servers.net.

edu. 172800 IN NS c.edu-servers.net.

edu. 172800 IN NS d.edu-servers.net.

;; ADDITIONAL SECTION:

f.edu-servers.net. 172800 IN A 192.35.51.30

a.edu-servers.net. 172800 IN A 192.5.6.30

g.edu-servers.net. 172800 IN A 192.42.93.30

g.edu-servers.net. 172800 IN AAAA 2001:503:cc2c::2:36

l.edu-servers.net. 172800 IN A 192.41.162.30

c.edu-servers.net. 172800 IN A 192.26.92.30

d.edu-servers.net. 172800 IN A 192.31.80.30

;; Query time: 29 msec

;; SERVER: 198.41.0.4#53(198.41.0.4)

;; WHEN: Sat Sep 29 21:40:02 EDT 2018

;; MSG SIZE rcvd: 282

Fans-MacBook-Pro:~ fanfeng$ dig +norecurse @f.edu-servers.net cicada.cs.yale.edu A

; <<>> DiG 9.10.6 <<>> +norecurse @f.edu-servers.net cicada.cs.yale.edu A

; (1 server found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18311

;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 5

;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags:; udp: 4096

;; QUESTION SECTION:

;cicada.cs.yale.edu. IN A

;; AUTHORITY SECTION:

yale.edu. 172800 IN NS serv1.net.yale.edu.

yale.edu. 172800 IN NS serv2.net.yale.edu.

yale.edu. 172800 IN NS serv4.net.yale.edu.

yale.edu. 172800 IN NS serv3.net.yale.edu.

;; ADDITIONAL SECTION:

serv1.net.yale.edu. 172800 IN A 130.132.1.9

serv2.net.yale.edu. 172800 IN A 130.132.1.10

serv4.net.yale.edu. 172800 IN A 130.132.89.9

serv3.net.yale.edu. 172800 IN A 130.132.1.11

;; Query time: 76 msec

;; SERVER: 192.35.51.30#53(192.35.51.30)

;; WHEN: Sat Sep 29 21:42:12 EDT 2018

;; MSG SIZE rcvd: 195

Fans-MacBook-Pro:~ fanfeng$ dig +norecurse @serv1.net.yale.edu cicada.cs.yale.edu A

; <<>> DiG 9.10.6 <<>> +norecurse @serv1.net.yale.edu cicada.cs.yale.edu A

; (1 server found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35045

;; flags: qr aa ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 5

;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags:; udp: 4096

;; QUESTION SECTION:

;cicada.cs.yale.edu. IN A

;; ANSWER SECTION:

cicada.cs.yale.edu. 10800 IN CNAME cicada.zoo.cs.yale.edu.

cicada.zoo.cs.yale.edu. 10800 IN A 128.36.232.5

;; AUTHORITY SECTION:

zoo.cs.yale.edu. 10800 IN NS serv1.net.yale.edu.

zoo.cs.yale.edu. 10800 IN NS serv3.net.yale.edu.

zoo.cs.yale.edu. 10800 IN NS serv2.net.yale.edu.

zoo.cs.yale.edu. 10800 IN NS serv4.net.yale.edu.

;; ADDITIONAL SECTION:

serv1.net.yale.edu. 10800 IN A 130.132.1.9

serv2.net.yale.edu. 10800 IN A 130.132.1.10

serv3.net.yale.edu. 10800 IN A 130.132.1.11

serv4.net.yale.edu. 10800 IN A 130.132.89.9

;; Query time: 5 msec

;; SERVER: 130.132.1.9#53(130.132.1.9)

;; WHEN: Sat Sep 29 21:42:50 EDT 2018

;; MSG SIZE rcvd: 236