Vicky Tu

CPSC 433 Computer Networks

Assignment 2

2/16/2016

[2a; 8pt] Please discuss how you tested and validated your program in Part 1. In particular, please describe:

How did you handle encoding/decoding to make sure that your code works across platforms?

I made my code portable by using the same charset whenever I used getBytes on the server and client. Instead of using putInt or putLong, I turn the numbers into bytes by hand, and then put the bytes into my buffer.

How may you extend the program (client and server) to measure the bandwidth (from client to server, and reverse)? A reference is to use packet pair to estimate bandwidth: http://www.usenix.org/event/usits01/full_papers/lai/lai_html/

Assume that we send two equally sized packets such that the difference between the times they are sent is t01- t00. Then suppose we have a bottleneck link that slows down the packets such that when they arrive, the different between the times they are received is tn1- tn0. Then bandwidth = s/ tn1- tn0, where s is the size of one packet. Of course, this assumes that s/b > t01- t00, that no other packets were being sent, that the two packets were sent close enough that they queued at the bottleneck, etc. But if s, tn1, and tn0 are recorded then the receiver can calculate the bandwidth. We can do this multiple times both ways in order to average and improve accuracy.

Is there any way you can extend the program to synchronize the clocks of the client and the server? What is the accuracy that you may get?

You can use Network Time Protocol (NTP) to synchronize the clocks of the client and server. NTP works by having the client send its time stamp, then the server reply with the receive time stamp and its current time stamp, then when the client receives both it can calculate how long the packet took to travel. The accuracy can be a few milliseconds if it's over the Web, or less than one millisecond if it's over a local area network. One could also use SNTP, a simplified NTP, but it's not as reliable.

Another way you can synchronize the clocks is use 10 pings to find the RTT, then the client can send a lot of messages to the server, each message being its current time + .5(RTT). The server would calculate the difference between its time and the time it received for every

message received. It could average these differences then set its clock to its time + the average of the differences. So effectively the server is setting its clock to the client's clock, taking into account things like the connection reliability, travel time, and the number of messages the client sends, which also affect accuracy.

But this is not a nice way to synchronize clocks if the server has many clients since it would have to synchronize its clock to every new client. So another way we could synchronize clocks is the get the client to synchronize its clock to the server. The PINGECHO message should be extended to include server_send_time and server_receive_time. Since the client can calculate RTT, it can subtract the server processing time (server_send_time - server_receive_time), divide by two, add the result the server_send_time, and set that as its own time. The accuracy of this synchronization is dependent on the reliability of the connection and the amount of traffic the server is experiencing.

[2b; 7pt] We discussed in class briefly on the implementation of DNS using UDP. Please answer the following questions:

- We mentioned that each DNS packet includes an identification field to handle reply forwarding. One implementation is the following. Assume that a DNS server receives a DNS request R in UDP with src IP being h, src port being p, and identification being id. The server needs to generate a query to another server to answer R. Then the server (1) stores in a hash map so that id maps to (h, p), and (2) uses the identification field id specified in R for the new query. Does this work? Please justify your answer.

  This does work because the server to which the new query is sent should not need to know the src IP or src port to satisfy the query, and then when query returns to the server with the stored id (since DNS is recursive), it would be able to match the id of the response to the src IP and port. But this does assume that ids will not collide.

- Assume that a DNS server needs to wait for, on average, 300 ms for the reply of its query that it forwarded. One concern is that the identification numbers may become exhausted when a new query that needs to be forwarded arrives. What is an expression to compute the exhausting likelihood?

  The DNS table is like a circuit-switched network because there are multiple things in the table at one time and anyone of them can leave the table to bring it back to the previous state. Thus using the equation for circuit-switched network queuing theory, we get $p_N = \frac{\frac{1}{N!}\left(\frac{\lambda}{\mu}\right)^N}{\sum_{k=0}^{N}\frac{1}{k!}\left(\frac{\lambda}{\mu}\right)^k}$, where $1/\mu$ = 300ms and N = 2^{16} since ids are 16 bits.

- DNS servers depend heavily on caching to improve scalability. Given a DNS name a.b.c.d (e.g., cicada.zoo.cs.yale.edu) that should be resolved. In what order should a DNS server search its cache before querying other servers?

  DNS resolves names to IP address via a recursive query. That means when a.b.c.d is queried and cached, it is cached as a.b.c.d. Thus, when given a.b.c.d, the DNS server should search for a.b.c.d, in its cache. If it doesn't find a.b.c.d, then it should search fro b.c.d, since entries start from the root, which is d. Once it finds a match or doesn't, DNS should search outside, starting from the thing that was matched or the root if nothing was matched.

- **[2c; 5pt]** DNS is a major security target. Please describe, briefly, a scheme, to take down a country's Internet access through an attack on DNS.

  I would exploit DNS caching by creating a server that looks authoritative and putting filling it with fake entries. That is, I would have it be able to match real queries with fake or incorrect IPs. Since DNS cached entries have time-to-live (TTL) of days, this would poison the country's DNS cache (the cache will fill up with fake entries, real entries will get pushed out) and people would to be able to get the Internet content they want until the TTL expires, or some one flushes the cache. But even then if they start requesting again and don't figure out that my server is returning bad records, their cache would get poisoned again.

- **[3a]** DNS allows delegation of the name space hierarchy. Please use the dig tool to manually find out one sequence of name servers to resolve cicada.cs.yale.edu to its IP address. Please capture the sequence of commands and outputs of using dig. In particular, please start with

  dig +norecurse @a.root-servers.net cicada.cs.yale.edu A

  If multiple name servers are returned, please pick one and continue the manual process. One claims that Yale uses a server from another school as a backup of Yale's name servers. Is this true?

  -bash-4.2$ dig +norecurse @a.root-servers.net cicada.cs.yale.edu A

  ; <<>> DiG 9.9.4-RedHat-9.9.4-29.el7_2.2 <<>> +norecurse @a.root-servers.net cicada.cs.yale.edu A

  ; (2 servers found)

  ;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56716

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

;; OPT PSEUDOSECTION:

; EDNS: version: 0, flags:; udp: 1232

;; QUESTION SECTION:

;cicada.cs.yale.edu. IN A

;; AUTHORITY SECTION:

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

;; ADDITIONAL SECTION:

a.edu-servers.net.	172800	IN	A	192.5.6.30
c.edu-servers.net.	172800	IN	A	192.26.92.30
d.edu-servers.net.	172800	IN	A	192.31.80.30
f.edu-servers.net.	172800	IN	A	192.35.51.30
g.edu-servers.net.	172800	IN	A	192.42.93.30
l.edu-servers.net.	172800	IN	A	192.41.162.30
g.edu-servers.net.	172800	IN	AAAA	2001:503:cc2c::2:36

;; Query time: 12 msec

;; SERVER: 2001:503:ba3e::2:30#53(2001:503:ba3e::2:30)

;; WHEN: Sun Feb 14 16:32:19 EST 2016

;; MSG SIZE rcvd: 282

-bash-4.2$ dig +norecurse @a.edu-servers.net cicada.cs.yale.edu A

; <<>> DiG 9.9.4-RedHat-9.9.4-29.el7_2.2 <<>> +norecurse @a.edu-servers.net cicada.cs.yale.edu A

; (1 server found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25488

;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 5, ADDITIONAL: 6

;; 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.

yale.edu.		172800	IN	NS	yale-server.uchicago.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

yale-server.uchicago.edu. 172800 IN A 128.135.249.140

;; Query time: 34 msec

;; SERVER: 192.5.6.30#53(192.5.6.30)

;; WHEN: Sun Feb 14 16:35:46 EST 2016

;; MSG SIZE rcvd: 246

-bash-4.2$ dig +norecurse @serv1.net.yale.edu cicada.cs.yale.edu A

; <<>> DiG 9.9.4-RedHat-9.9.4-29.el7_2.2 <<>> +norecurse @serv1.net.yale.edu cicada.cs.yale.edu A

; (1 server found)

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52141

;; 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 serv2.net.yale.edu.

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

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

zoo.cs.yale.edu. 10800 IN NS serv1.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: 0 msec

;; SERVER: 130.132.1.9#53(130.132.1.9)

;; WHEN: Sun Feb 14 16:36:05 EST 2016

;; MSG SIZE rcvd: 236

While ssh'ed in the Zoo, I digged for cicada.cs.yale.edu and got 128.36.232.5. It does seem true that Yale uses a server form another school as a backup of Yale's name servers because I did see UChicago listed in the authority sections.

- **[3b]** Please find 3 gmail servers (IP addresses). Please show the command outputs to support your answer.

MX records specify which mail servers are responsible for accepting email messages on behalf of a recipient's domain, so the by digging the mx records of gmail and digging for the IP addresses of those returned servers, I've found 3 gmail servers. Their IPs are 74.125.24.26, 74.125.71.26, and 74.125.136.26.

-bash-4.2$ dig mx gmail.com

; <<>> DiG 9.9.4-RedHat-9.9.4-29.el7_2.2 <<>> mx gmail.com

;; global options: +cmd

;; Got answer:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47954

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

;; OPT PSEUDOSECTION:

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

;; QUESTION SECTION:

;gmail.com. IN MX

;; ANSWER SECTION:

gmail.com. 3600 IN MX 20 alt2.gmail-smtp-in.l.google.com.

gmail.com. 3600 IN MX 30 alt3.gmail-smtp-in.l.google.com.

gmail.com. 3600 IN MX 40 alt4.gmail-smtp-in.l.google.com.

gmail.com. 3600 IN MX 5 gmail-smtp-in.l.google.com.

gmail.com. 3600 IN MX 10 alt1.gmail-smtp-in.l.google.com.

;; AUTHORITY SECTION:

gmail.com. 58057 IN NS ns2.google.com.

gmail.com. 58057 IN NS ns3.google.com.

gmail.com. 58057 IN NS ns1.google.com.

gmail.com. 58057 IN NS ns4.google.com.

;; ADDITIONAL SECTION:

ns1.google.com. 253888 IN A 216.239.32.10

ns2.google.com. 236353 IN A 216.239.34.10

ns3.google.com. 236353 IN A 216.239.36.10

ns4.google.com. 236353 IN A 216.239.38.10

;; Query time: 22 msec

;; SERVER: 130.132.1.11#53(130.132.1.11)

;; WHEN: Sun Feb 14 16:55:17 EST 2016

;; MSG SIZE rcvd: 297

-bash-4.2$ dig +noall +answer alt2.gmail-smtp-in.l.google.com.

alt2.gmail-smtp-in.l.google.com. 56 IN A 74.125.24.26

-bash-4.2$ dig +noall +answer alt3.gmail-smtp-in.l.google.com.

alt3.gmail-smtp-in.l.google.com. 33 IN A 74.125.71.26

-bash-4.2$ dig +noall +answer alt4.gmail-smtp-in.l.google.com.

alt4.gmail-smtp-in.l.google.com. 300 IN A 74.125.136.26

- **[3c]** Is a server with IP address 173.194.1.1 an authorized mail transfer agent for gmail? Please show the command outputs to support your answer.

  Yes, it is. I kept following gmail's redirects and found a list of its authorized IPs. One of them is ip4:173.194.0.0/16, which means the last 2 bytes can very. Thus. 173.194.1.1 is an authorized mail transfer agent for gmail.

Chloe:~ vickytu$ dig TXT gmail.com

; <<>> DiG 9.8.3-P1 <<>> TXT gmail.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42564
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;gmail.com. IN TXT

;; ANSWER SECTION:
gmail.com. 300 IN TXT "v=spf1 redirect=_spf.google.com"

```
;; AUTHORITY SECTION:
gmail.com.        42139    IN    NS    ns3.google.com.
gmail.com.        42139    IN    NS    ns4.google.com.
gmail.com.        42139    IN    NS    ns1.google.com.
gmail.com.        42139    IN    NS    ns2.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.        42073    IN    A    216.239.32.10
ns2.google.com.        42073    IN    A    216.239.34.10
ns3.google.com.        42073    IN    A    216.239.36.10
ns4.google.com.        42073    IN    A    216.239.38.10

;; Query time: 24 msec
;; SERVER: 130.132.1.9#53(130.132.1.9)
;; WHEN: Tue Feb 16 21:41:38 2016
;; MSG SIZE  rcvd: 214

Chloe:~ vickytu$ dig TXT _spf.google.com

; <<>> DiG 9.8.3-P1 <<>> TXT _spf.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57247
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;_spf.google.com.              IN    TXT

;; ANSWER SECTION:
_spf.google.com.300    IN    TXT    "v=spf1 include:_netblocks.google.com
include:_netblocks2.google.com include:_netblocks3.google.com ~all"

;; AUTHORITY SECTION:
google.com.            41790    IN    NS    ns4.google.com.
google.com.            41790    IN    NS    ns3.google.com.
google.com.            41790    IN    NS    ns1.google.com.
google.com.            41790    IN    NS    ns2.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.        42026    IN    A    216.239.32.10
ns2.google.com.        42026    IN    A    216.239.34.10
ns3.google.com.        42026    IN    A    216.239.36.10
ns4.google.com.        42026    IN    A    216.239.38.10
```

```
;; Query time: 18 msec
;; SERVER: 130.132.1.9#53(130.132.1.9)
;; WHEN: Tue Feb 16 21:42:25 2016
;; MSG SIZE  rcvd: 285

Chloe:~ vickytu$ dig TXT _netblocks.google.com

; <<>> DiG 9.8.3-P1 <<>> TXT _netblocks.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6066
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;_netblocks.google.com.          IN      TXT

;; ANSWER SECTION:
_netblocks.google.com.   3031 IN      TXT   "v=spf1 ip4:64.18.0.0/20 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 ip4:207.126.144.0/20 ip4:209.85.128.0/17 ip4:216.58.192.0/19
ip4:216.239.32.0/19 ~all"

;; AUTHORITY SECTION:
google.com.             41770   IN      NS      ns4.google.com.
google.com.             41770   IN      NS      ns3.google.com.
google.com.             41770   IN      NS      ns2.google.com.
google.com.             41770   IN      NS      ns1.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         42006   IN      A       216.239.32.10
ns2.google.com.         42006   IN      A       216.239.34.10
ns3.google.com.         42006   IN      A       216.239.36.10
ns4.google.com.         42006   IN      A       216.239.38.10

;; Query time: 4 msec
;; SERVER: 130.132.1.9#53(130.132.1.9)
;; WHEN: Tue Feb 16 21:42:45 2016
;; MSG SIZE  rcvd: 429
```