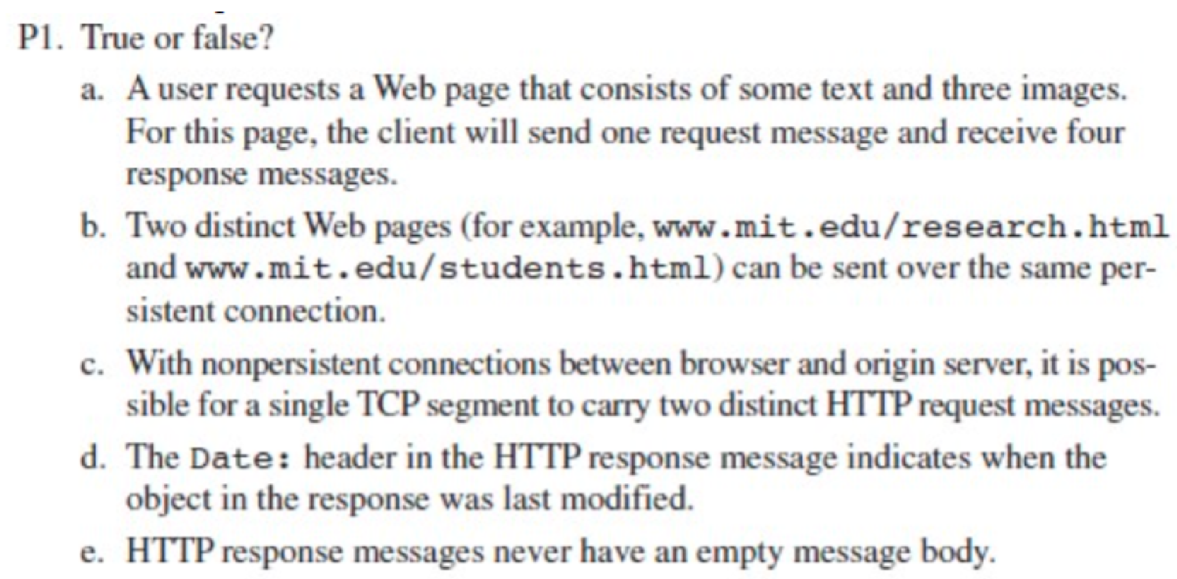
Julian Garcia
CSE 353: HW2



P1. True or false?

a. A user requests a Web page that consists of some text and three images. For this page, the client will send one request message and receive four response messages.

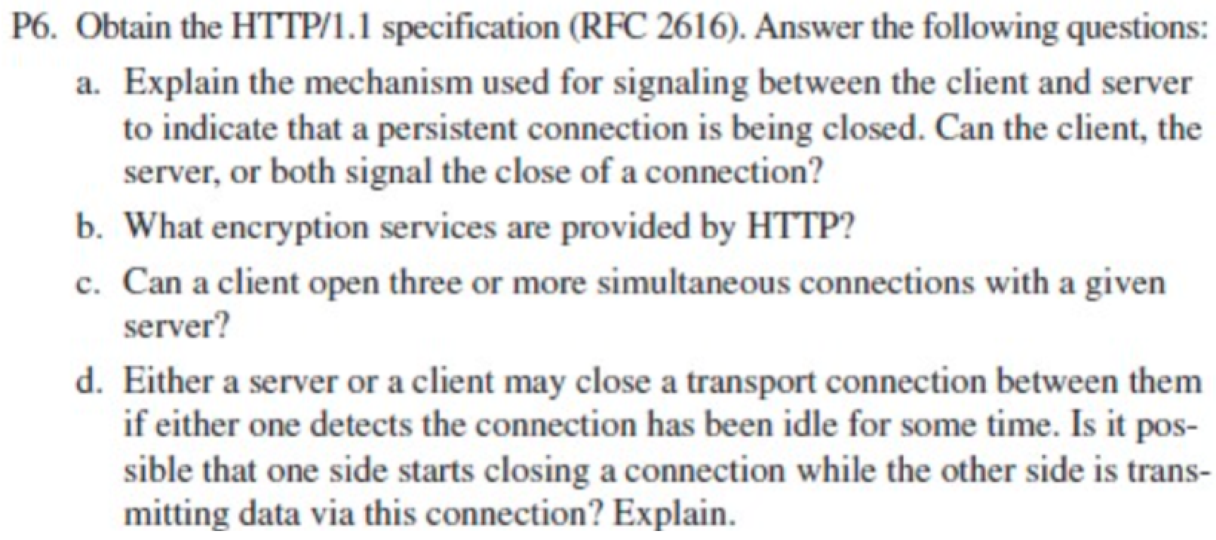
b. Two distinct Web pages (for example, `www.mit.edu/research.html` and `www.mit.edu/students.html`) can be sent over the same persistent connection.

c. With nonpersistent connections between browser and origin server, it is possible for a single TCP segment to carry two distinct HTTP request messages.

d. The `Date:` header in the HTTP response message indicates when the object in the response was last modified.

e. HTTP response messages never have an empty message body.

(a) False, there should be 4 requests and 4 responses
(b) True, the connection stays open for the next page
(c) False, there can only be one request per TCP segment
(d) False, the date is when the request was made
(e) False



P6. Obtain the HTTP/1.1 specification (RFC 2616). Answer the following questions:

a. Explain the mechanism used for signaling between the client and server to indicate that a persistent connection is being closed. Can the client, the server, or both signal the close of a connection?

b. What encryption services are provided by HTTP?

c. Can a client open three or more simultaneous connections with a given server?

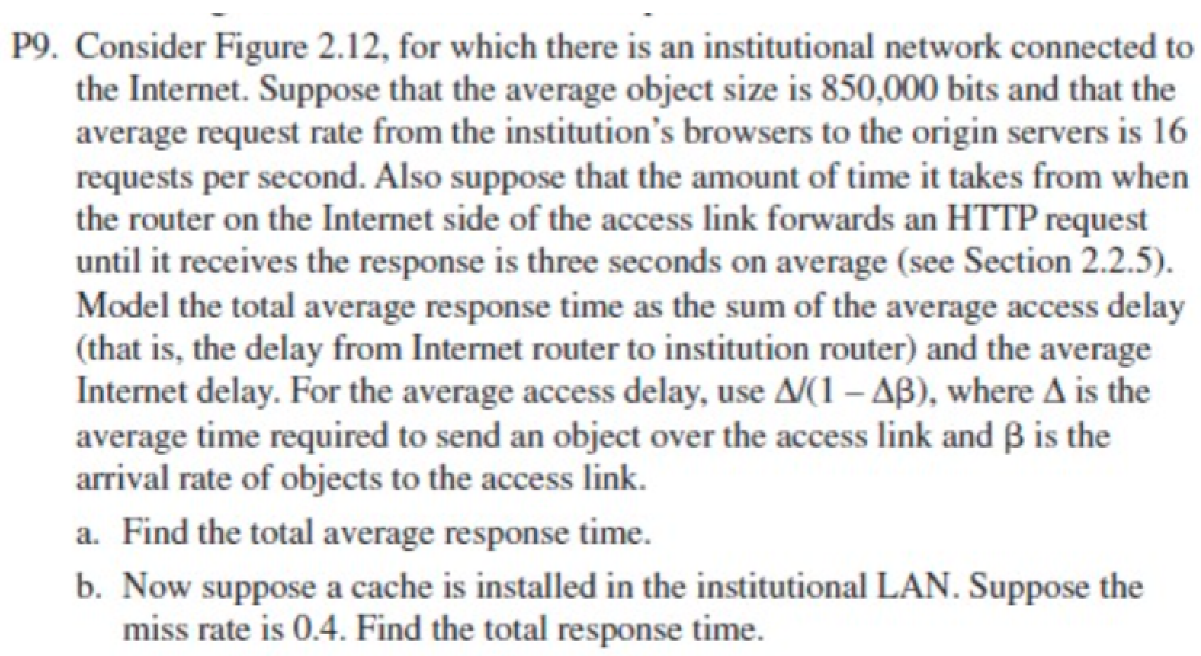
d. Either a server or a client may close a transport connection between them if either one detects the connection has been idle for some time. Is it possible that one side starts closing a connection while the other side is transmitting data via this connection? Explain.

(a) As explained in the specification, persistent connections are the default of the connection here. So the client assumes that the connection stays open until a signal is given otherwise. To close the connection, either the client or the server has to specify the connection is to be closed in the connection header. For the host to close the connection, the connection header has to have the “close” token in the request. The server basically does the same thing, it has to designate the same “close” token in the header along with its response.

(b) There aren't really any actual encryption services given by HTTP 1.1. Instead HTTP includes Content-Encoding. Content-Encoding allows you to store your content in coded form, which is then sent to and decoded only by the recipient. Content-Encodings basically allow you to transform your content as you see fit without having to worry about losing any information. So to actually use content-encoding, encoding values have to be placed in the header fields of 'Accept-Encoding' and 'Content-Encoding'. There's a risk involved here which can result in a response saying "Incorrect Media Type" after requesting a message, if any of the values you entered were wrong. To use actual encryption services, HTTPS is what is needed since it has a socket layer within the HTTP protocol that is secure which means it can encrypt sent messages.

(c) This is true, 3 or more simultaneous connections can be opened with a given server.

(d) It is possible for one side to close while the other side is transmitting. This is due to the fact that HTTP doesn't keep states (is stateless) so neither side can be aware of the other's state. Because of this problem, the client must be able to reopen the connection and resend the message.



P9. Consider Figure 2.12, for which there is an institutional network connected to the Internet. Suppose that the average object size is 850,000 bits and that the average request rate from the institution's browsers to the origin servers is 16 requests per second. Also suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is three seconds on average (see Section 2.2.5). Model the total average response time as the sum of the average access delay (that is, the delay from Internet router to institution router) and the average Internet delay. For the average access delay, use $\Delta/(1 - \Delta\beta)$, where Δ is the average time required to send an object over the access link and β is the arrival rate of objects to the access link.

a. Find the total average response time.

b. Now suppose a cache is installed in the institutional LAN. Suppose the miss rate is 0.4. Find the total response time.

Julian Garcia
CSE 353: HW2

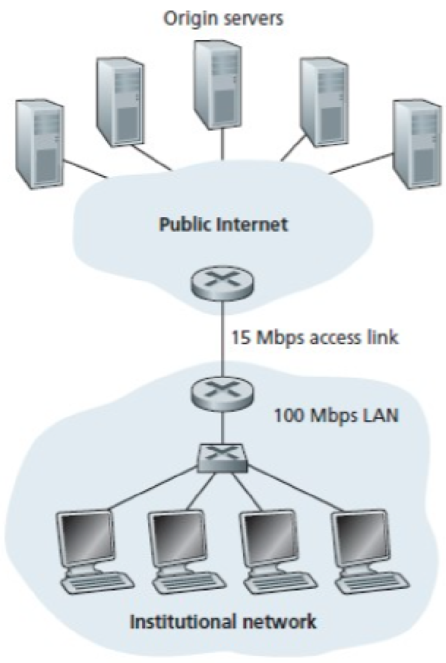
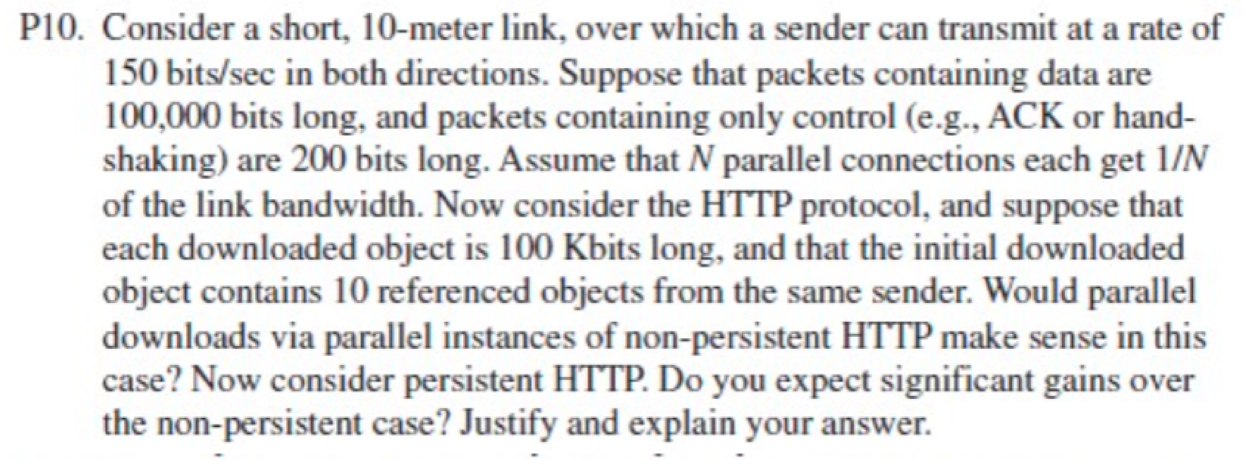


Figure 2.12

a) Size (L) = 850,000 bits
Rate (R) = 15,000,000 bits/sec
Δ = (850,000 bits)/(15,000,000 bits/sec) = .0567 sec
$\beta\Delta$=(16 requests/sec)(.0567 sec/request) = 0.907.
Average Access Delay = (.0567 sec)/(1 - .907) ≈ .6 seconds
The total average response time is 3sec + 0.6sec = 3.6 sec.

b)
Since the miss rate is 0.4, 60% of the requests are handled.
Avg access delay = 0.057sec/ (1-(0.4)(0.907)) = 0.089sec
Response time ≈ 0 if our 0.6 chance hits and the request is satisfied by cache
Avg response time = 0.89 + 3 = 3.089sec when our chance doesn't hit and our cache misses.
Therefore, Avg Response time = (0.6)(0) + (0.4)(3.089) = 1.24sec
So the average response time goes down from 3.6 seconds to 1.24 seconds.

Julian Garcia
CSE 353: HW2


P10. Consider a short, 10-meter link, over which a sender can transmit at a rate of 150 bits/sec in both directions. Suppose that packets containing data are 100,000 bits long, and packets containing only control (e.g., ACK or hand-shaking) are 200 bits long. Assume that *N* parallel connections each get 1/*N* of the link bandwidth. Now consider the HTTP protocol, and suppose that each downloaded object is 100 Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender. Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? Now consider persistent HTTP. Do you expect significant gains over the non-persistent case? Justify and explain your answer.

T_{Prop} is propagation delay in my work below:

10 Connections sharing 150 bits/sec
Each connection gets 15 bits/sec

Now calculate total time to receive every object:
$(200/150+T_{prop} + 200/150 +T_{prop} + 200/150+T_{prop} + 100,000/150+ T_{prop})+ (200/(150/10)+T_{prop} + 200/(150/10) +T_{prop} + 200/(150/10)+T_{prop} + 100,000/(150/10)+ T_{prop}) =$ $7377 + 8T_{prop}$ sec

On a persistent connection:
Total time calculation:
$(200/150+T_{prop} + 200/150 +T_{prop} + 200/150+T_{prop} + 100,000/150+ T_{prop})+ 10*(200/150+T_{prop} + 100,000/150+ T_{prop})$
$=7351 + 24*T_{prop}$ sec

So the difference lies in how big T_{prop} is, to find T_{prop} we have to take the length of the meter, and the speed of light (299 792 458 m / s) which I'll denote by the letter '*c*' in the equation below:
$T_{prop} = 10/c$
$T_{prop} = 3.335 \times 10^{-8}$ m/sec

Since T_{prop} is so ridiculously small, we can conclude that persistent HTTP isn't really significantly faster than the non-persistent version with parallel download.
Parallel downloads via parallel instances of non-persistent HTTP seems to make sense in this case.

Julian Garcia
CSE 353: HW2

P11. Consider the scenario introduced in the previous problem. Now suppose that the link is shared by Bob with four other users. Bob uses parallel instances of non-persistent HTTP, and the other four users use non-persistent HTTP without parallel downloads.

a. Do Bob's parallel connections help him get Web pages more quickly? Why or why not?

b. If all five users open five parallel instances of non-persistent HTTP, then would Bob's parallel connections still be beneficial? Why or why not?

a) Yes, the parallel connections would help him get web pages more quickly since the parallel instances would allow for requests to be made in parallel, meaning he won't have to wait in between requests unlike the users without parallel downloads.

b) Yes even with five open parallel instances, Bob's parallel connections will still be beneficial, this is true because every user is given a set of bandwidth to use, so multiple users having parallel instances open will not affect the benefit of Bob's parallel connections.

P14. How does SMTP mark the end of a message body? How about HTTP? Can HTTP use the same method as SMTP to mark the end of a message body? Explain.

SMTP uses a line that only has a period in it to mark the end of a message body.
HTTP uses a very different method, it designates a content-length in its header to give the total length of a message body, so the body is marked complete by when it hits its content-length.
They CANNOT use the same method's, this is because SMTP's message body is written in ASCII, while the HTTP message could just be data in binary.

P18. a. What is a *whois* database?

b. Use various whois databases on the Internet to obtain the names of two DNS servers. Indicate which whois databases you used.

c. Use nslookup on your local host to send DNS queries to three DNS servers: your local DNS server and the two DNS servers you found in part (b). Try querying for Type A, NS, and MX reports. Summarize your findings.

d. Use nslookup to find a Web server that has multiple IP addresses. Does the Web server of your institution (school or company) have multiple IP addresses?

e. Use the ARIN whois database to determine the IP address range used by your university.

f. Describe how an attacker can use whois databases and the nslookup tool to perform reconnaissance on an institution before launching an attack.

g. Discuss why whois databases should be publicly available.

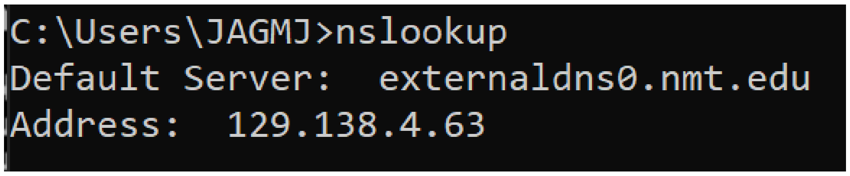
Julian Garcia
CSE 353: HW2

a) A whois database is a database that stores contact and registration information for domain names.

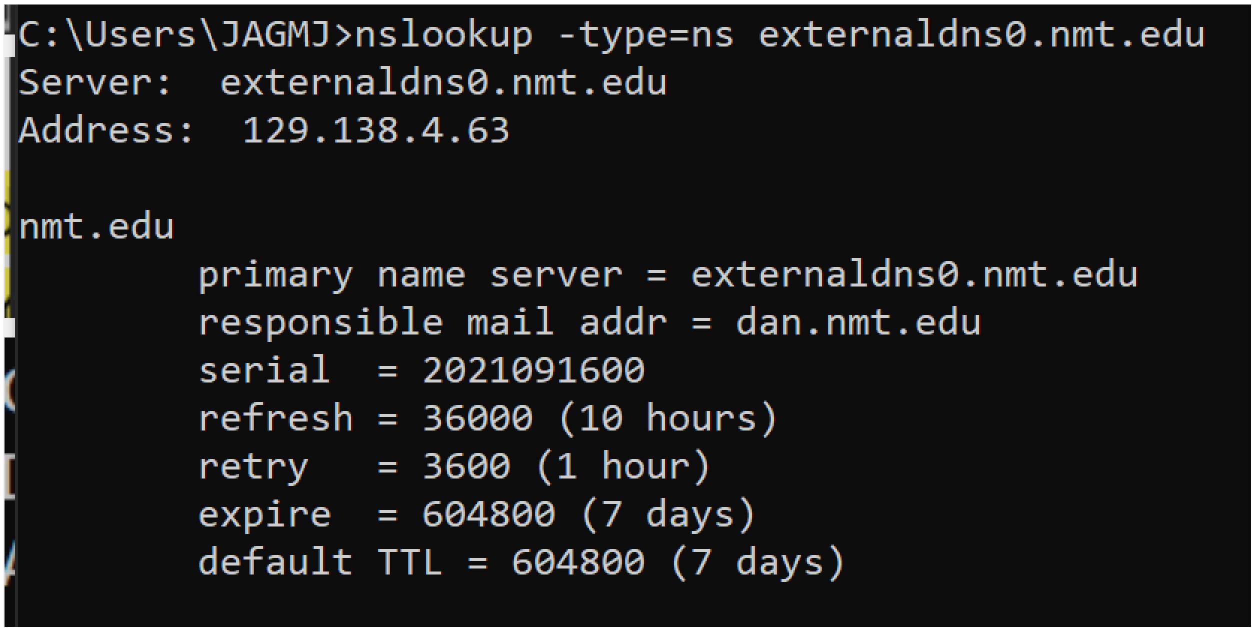
b) For the first lookup I used https://lookup.icann.org/lookup
I looked up kotaku.com on icann.org, the DNS server names that came up were:
NS-110.AWSDNS-13.COM
NS-1146.AWSDNS-15.ORG
NS-1835.AWSDNS-37.CO.UK
NS-624.AWSDNS-14.NET

For the second lookup I used https://www.networksolutions.com/whois/
I looked up carmax.com on networksolutions.com, the DNS server names that came up were:
Name Server: CMXNS01.CARMAX.NET
Name Server: CMXNS02.CARMAX.NET

c)
Local DNS server:

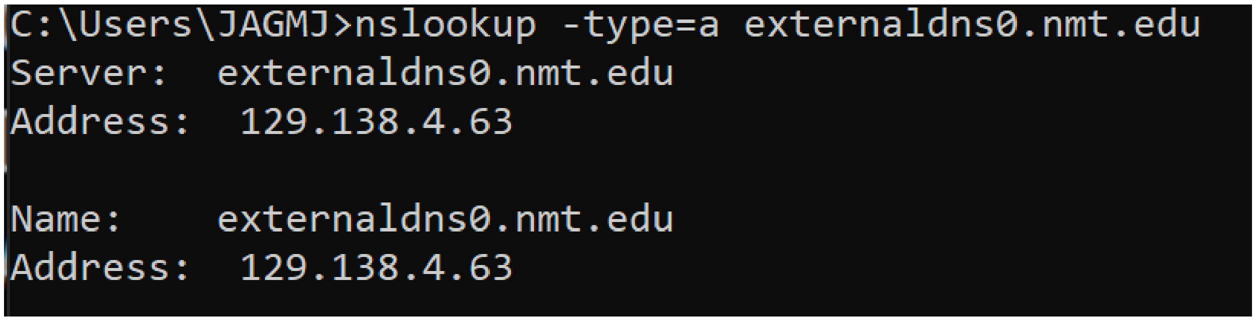


```
C:\Users\JAGMJ>nslookup
Default Server:  externaldns0.nmt.edu
Address:  129.138.4.63
```

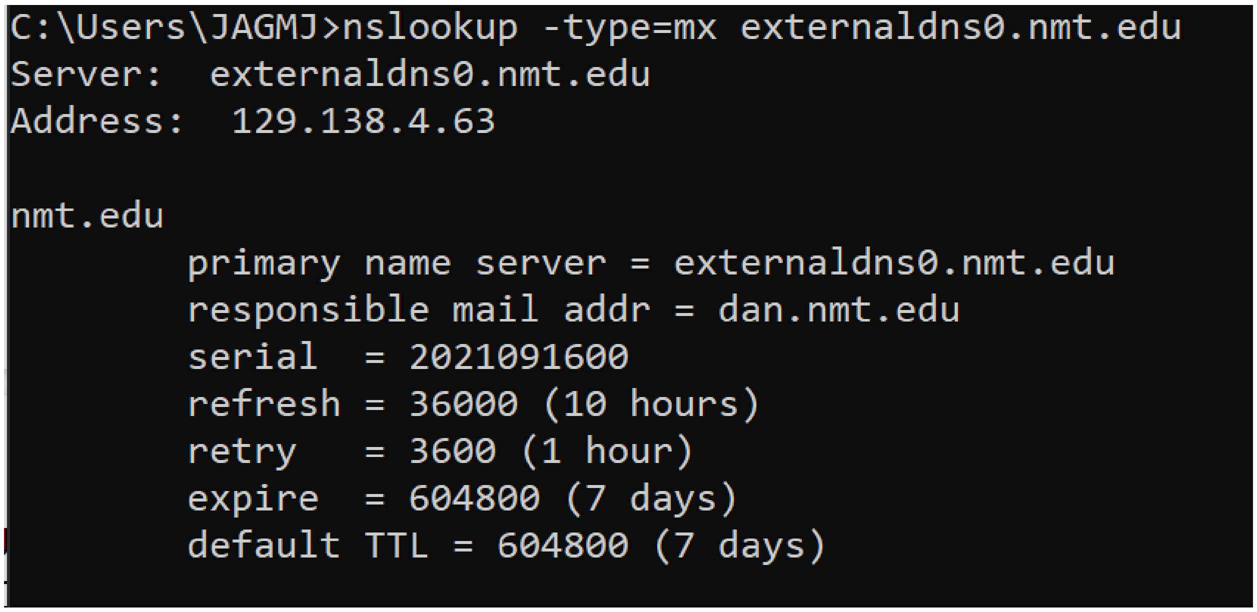


```
C:\Users\JAGMJ>nslookup -type=ns externaldns0.nmt.edu
Server:  externaldns0.nmt.edu
Address:  129.138.4.63

nmt.edu
        primary name server = externaldns0.nmt.edu
        responsible mail addr = dan.nmt.edu
        serial  = 2021091600
        refresh = 36000 (10 hours)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        default TTL = 604800 (7 days)
```

Julian Garcia
CSE 353: HW2

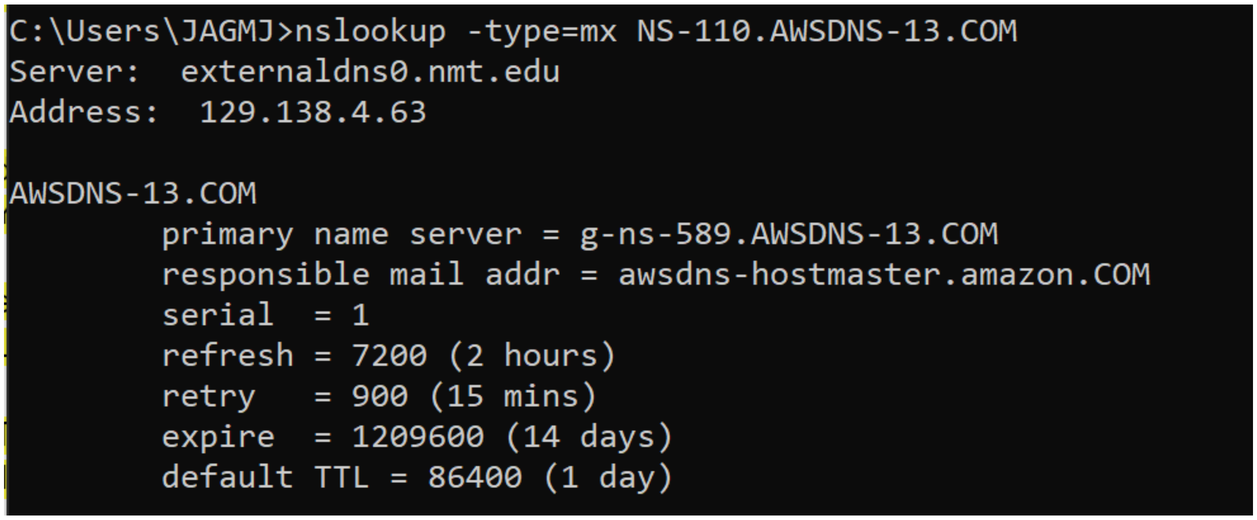


```
C:\Users\JAGMJ>nslookup -type=a externaldns0.nmt.edu
Server:  externaldns0.nmt.edu
Address:  129.138.4.63

Name:    externaldns0.nmt.edu
Address:  129.138.4.63
```



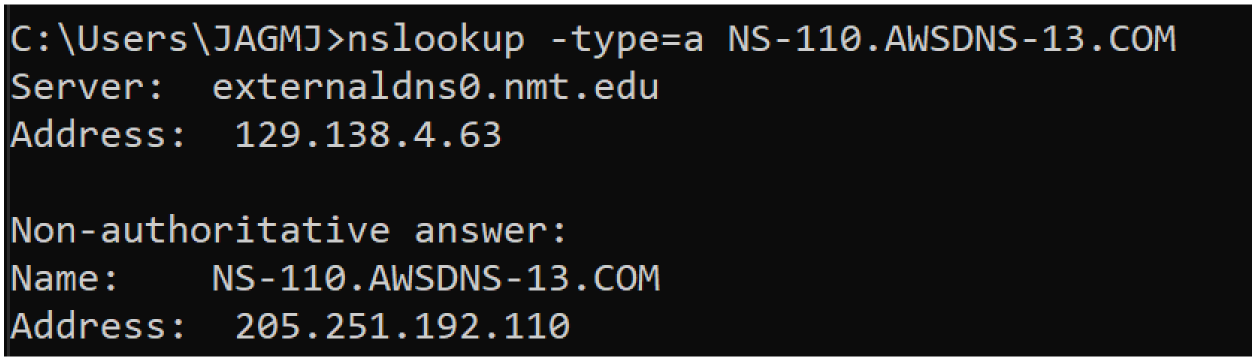
```
C:\Users\JAGMJ>nslookup -type=mx externaldns0.nmt.edu
Server:  externaldns0.nmt.edu
Address:  129.138.4.63

nmt.edu
        primary name server = externaldns0.nmt.edu
        responsible mail addr = dan.nmt.edu
        serial  = 2021091600
        refresh = 36000 (10 hours)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        default TTL = 604800 (7 days)
```

Now querying NS-110.AWSDNS-13.COM

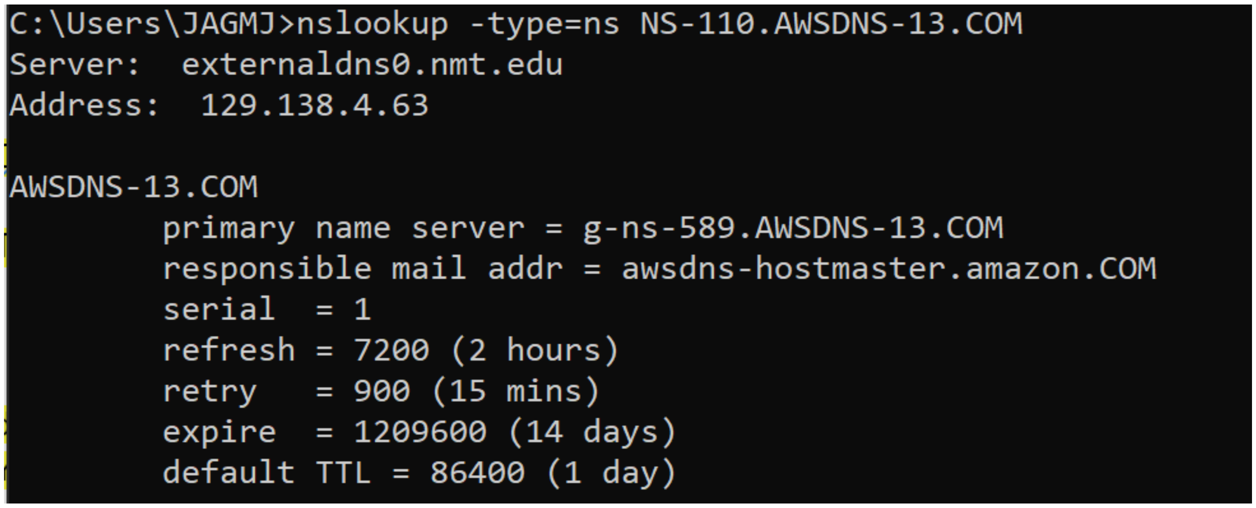


```
C:\Users\JAGMJ>nslookup -type=mx NS-110.AWSDNS-13.COM
Server:  externaldns0.nmt.edu
Address:  129.138.4.63

AWSDNS-13.COM
        primary name server = g-ns-589.AWSDNS-13.COM
        responsible mail addr = awsdns-hostmaster.amazon.COM
        serial  = 1
        refresh = 7200 (2 hours)
        retry   = 900 (15 mins)
        expire  = 1209600 (14 days)
        default TTL = 86400 (1 day)
```

Julian Garcia
CSE 353: HW2

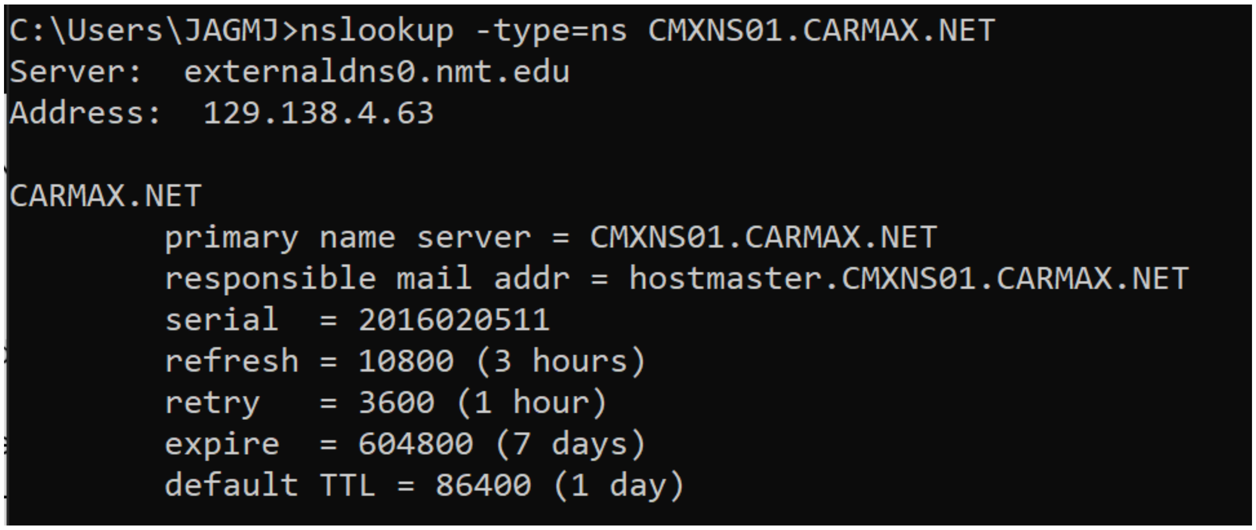


```
C:\Users\JAGMJ>nslookup -type=a NS-110.AWSDNS-13.COM
Server:  externaldns0.nmt.edu
Address:  129.138.4.63

Non-authoritative answer:
Name:    NS-110.AWSDNS-13.COM
Address:  205.251.192.110
```



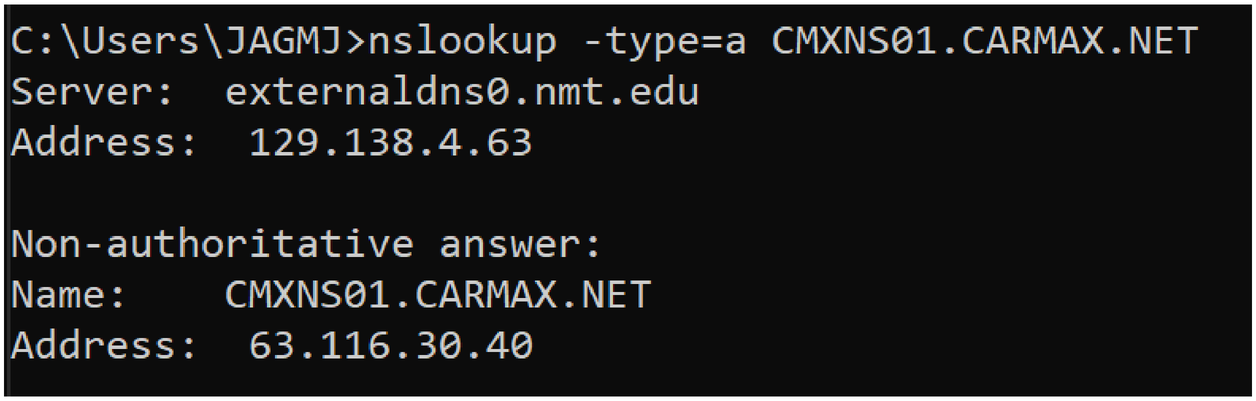
```
C:\Users\JAGMJ>nslookup -type=ns NS-110.AWSDNS-13.COM
Server:  externaldns0.nmt.edu
Address:  129.138.4.63

AWSDNS-13.COM
        primary name server = g-ns-589.AWSDNS-13.COM
        responsible mail addr = awsdns-hostmaster.amazon.COM
        serial  = 1
        refresh = 7200 (2 hours)
        retry   = 900 (15 mins)
        expire  = 1209600 (14 days)
        default TTL = 86400 (1 day)
```

Now querying CMXNS01.CARMAX.NET

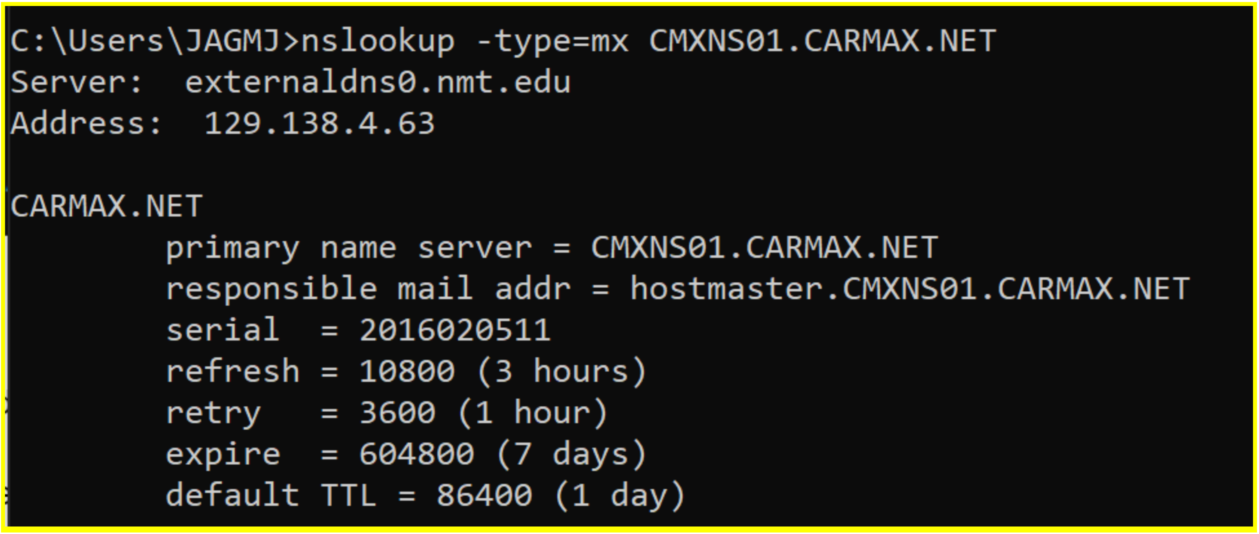


```
C:\Users\JAGMJ>nslookup -type=ns CMXNS01.CARMAX.NET
Server:  externaldns0.nmt.edu
Address:  129.138.4.63

CARMAX.NET
        primary name server = CMXNS01.CARMAX.NET
        responsible mail addr = hostmaster.CMXNS01.CARMAX.NET
        serial  = 2016020511
        refresh = 10800 (3 hours)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        default TTL = 86400 (1 day)
```

Julian Garcia
CSE 353: HW2

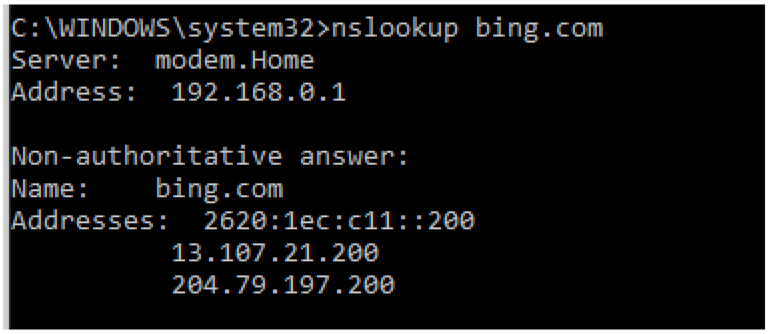


```
C:\Users\JAGMJ>nslookup -type=a CMXNS01.CARMAX.NET
Server:  externaldns0.nmt.edu
Address:  129.138.4.63

Non-authoritative answer:
Name:    CMXNS01.CARMAX.NET
Address:  63.116.30.40
```



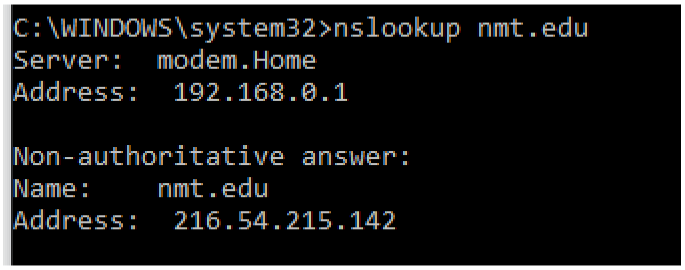
```
C:\Users\JAGMJ>nslookup -type=mx CMXNS01.CARMAX.NET
Server:  externaldns0.nmt.edu
Address:  129.138.4.63

CARMAX.NET
        primary name server = CMXNS01.CARMAX.NET
        responsible mail addr = hostmaster.CMXNS01.CARMAX.NET
        serial  = 2016020511
        refresh = 10800 (3 hours)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        default TTL = 86400 (1 day)
```

d) I looked up bing.com and found it had multiple addresses:



```
C:\WINDOWS\system32>nslookup bing.com
Server:  modem.Home
Address:  192.168.0.1

Non-authoritative answer:
Name:    bing.com
Addresses:  2620:1ec:c11::200
          13.107.21.200
          204.79.197.200
```

Looking up our school's web server, at NMT.edu, I got:

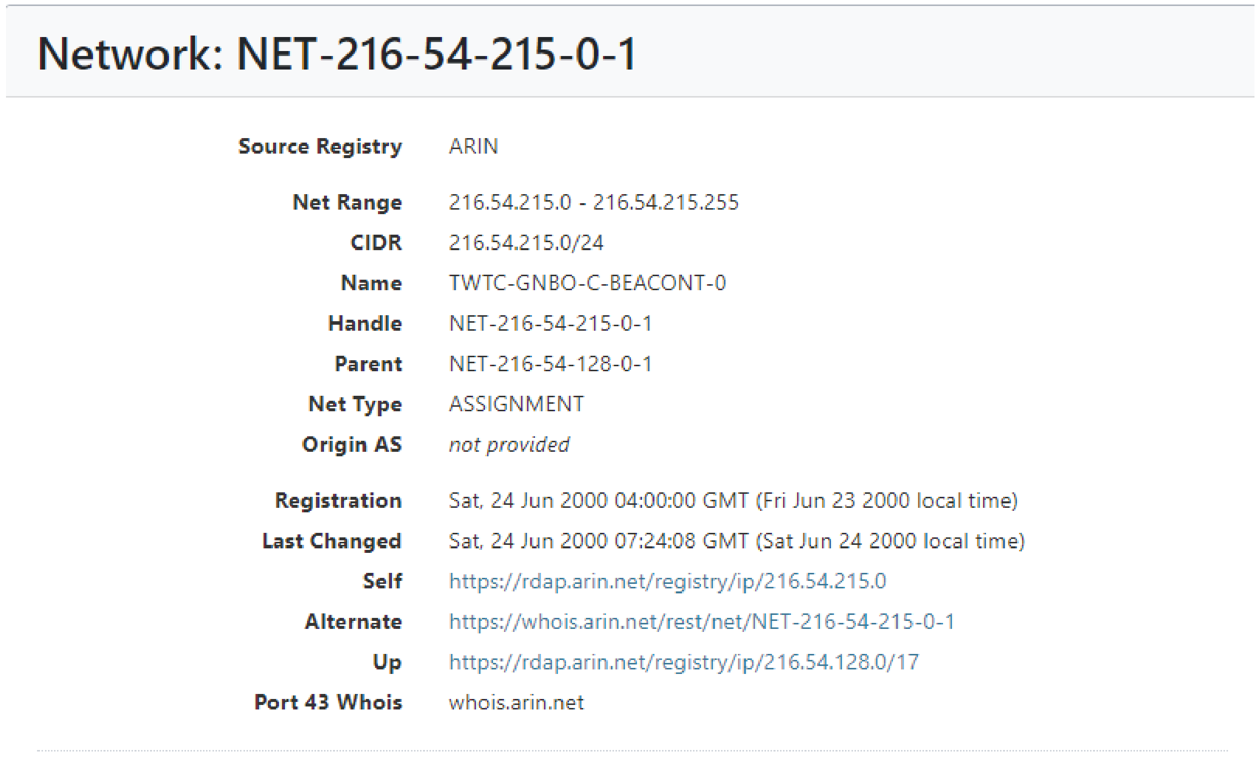


```
C:\WINDOWS\system32>nslookup nmt.edu
Server:  modem.Home
Address:  192.168.0.1

Non-authoritative answer:
Name:    nmt.edu
Address:  216.54.215.142
```

Julian Garcia
CSE 353: HW2

So, our school only has 1 IP address

e)
After running the IP in ARIN the results were as follows:



Network: NET-216-54-215-0-1

Source Registry	ARIN
Net Range	216.54.215.0 - 216.54.215.255
CIDR	216.54.215.0/24
Name	TWTC-GNBO-C-BEACONT-0
Handle	NET-216-54-215-0-1
Parent	NET-216-54-128-0-1
Net Type	ASSIGNMENT
Origin AS	*not provided*
Registration	Sat, 24 Jun 2000 04:00:00 GMT (Fri Jun 23 2000 local time)
Last Changed	Sat, 24 Jun 2000 07:24:08 GMT (Sat Jun 24 2000 local time)
Self	https://rdap.arin.net/registry/ip/216.54.215.0
Alternate	https://whois.arin.net/rest/net/NET-216-54-215-0-1
Up	https://rdap.arin.net/registry/ip/216.54.128.0/17
Port 43 Whois	whois.arin.net

So the range at nmt.edu is:
216.54.215.0 - 216.54.215.255

f) Whoever is looking into a target for an attack could theoretically just use a whois database and nslookup to find out every IP address the target for their attack is using, which would be valuable info for coordinating an attack.

g) Whois databases need to be publicly available because they're the main point of reference when anyone is trying to find out the registration details of a domain, as well as the IP info.