Pre-lab 2

1. HTTP 201 - This status code means that the action requested by the client was successfully received and accepted, and a new resource has been created.
   HTTP 204 - This status code means that the server has processed the request successfully and is not returning any content
   HTTP 404 - This status code means that the requested resource cannot be found
   HTTP 401 - This status code means that the server is refusing the request of the client as the user/client does not have the appropriate privilege
   HTTP 400 - This status code means that there was an error on the client side so the server cannot process the request
2. **OPTIONS** - allows the client to determine the options and/or requirements associated with a resource, or the capabilities of a server, without implying a resource action or initiating a resource retrieval
   GET - retrieves whatever information is identified by the Request-URI
   HEAD - retrieves whatever information is identified by the Request-URI, but the server must not return a message-body in the response
   POST - request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line
   PUT - requests that the enclosed entity be stored under the supplied Request-URI. Essentially the command tells the server to store an entity of the name that is provided in the Request-URI
   DELETE - requests that the origin server delete the resource identified by the Request-URI
   TRACE - invokes a remote, application-layer loopback of the request message
   CONNECT - reserves the method name CONNECT for use with a proxy that can dynamically switch to being a tunnel
3. The HTTP return status was 200 OK and the command ‘wget example.com’ was used to get the output.

4. This telnet server displays a Star Wars introduction, the introduction that one would see in every Star Wars movie. After the introduction, it shows a skit of the droid and R2D2 running from StormTroopers.

5. DNS resource record is essentially a database that maps names to IP addresses, like a phone book, which allows easy resolutions for DNS queries. After running the command ‘nslookup’ on ucsc.edu, a MX resource record is returned detailing 5 SMTP servers with their own IP address.

```
> set q=mx
> ucsc.edu
Server:         192.168.1.1
Address:        192.168.1.1#53

Non-authoritative answer:
ucsc.edu        mail exchanger = 10 alt4.aspmx.l.google.com.
ucsc.edu        mail exchanger = 10 alt3.aspmx.l.google.com.
ucsc.edu        mail exchanger = 5 alt2.aspmx.l.google.com.
ucsc.edu        mail exchanger = 5 alt1.aspmx.l.google.com.
ucsc.edu        mail exchanger = 1 aspmx.l.google.com.

Authoritative answers can be found from:
alt4.aspmx.l.google.com internet address = 64.233.177.26
alt4.aspmx.l.google.com has AAAA address 2607:f8b0:4002:c08::1a
alt2.aspmx.l.google.com internet address = 172.253.112.27
alt2.aspmx.l.google.com has AAAA address 2607:f8b0:4023::1b
alt1.aspmx.l.google.com internet address = 209.85.145.27
alt1.aspmx.l.google.com has AAAA address 2607:f8b0:4001:c1e::1b
aspmx.l.google.com      internet address = 74.125.142.27
aspmx.l.google.com      has AAAA address 2607:f8b0:400e:c08::1b
>
```

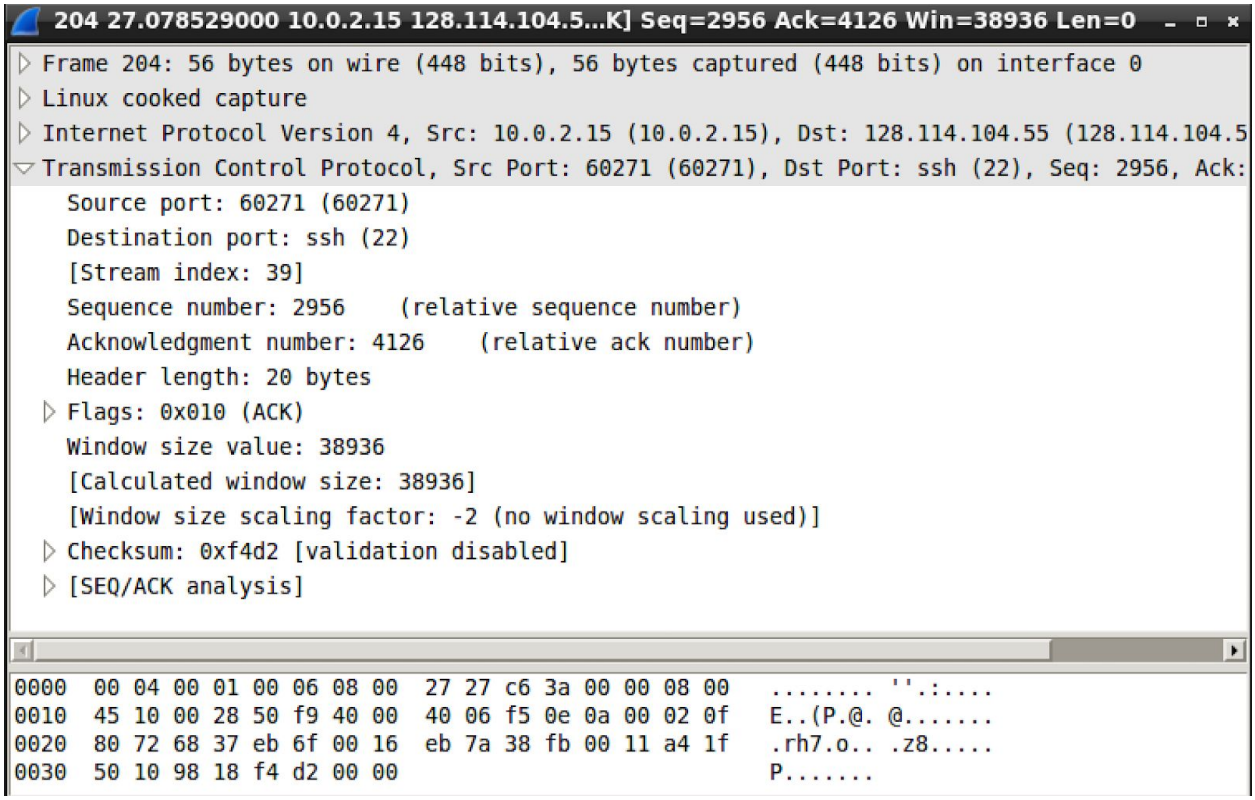
6. The command ‘nslookup -type=ns .’ calls to look up the name service of root with the option to change the type of the information query to ns. The output displays a Non-authoritative answer. This Non-authoritative answer displays a list of root servers that are authoritative nameservers. Additionally, it displays where the authoritative answer can be found as it lists the previous displayed nameservers followed by their IP address and AAAA address.

```
Non-authoritative answer:
.       nameserver = e.root-servers.net.
.       nameserver = f.root-servers.net.
.       nameserver = g.root-servers.net.
.       nameserver = h.root-servers.net.
.       nameserver = i.root-servers.net.
.       nameserver = j.root-servers.net.
.       nameserver = k.root-servers.net.
.       nameserver = l.root-servers.net.
.       nameserver = m.root-servers.net.
.       nameserver = a.root-servers.net.
.       nameserver = b.root-servers.net.
.       nameserver = c.root-servers.net.
.       nameserver = d.root-servers.net.
```

```
Authoritative answers can be found from:
a.root-servers.net      internet address = 198.41.0.4
a.root-servers.net      has AAAA address 2001:503:ba3e::2:30
b.root-servers.net      internet address = 199.9.14.201
b.root-servers.net      has AAAA address 2001:500:200::b
c.root-servers.net      internet address = 192.33.4.12
c.root-servers.net      has AAAA address 2001:500:2::c
d.root-servers.net      internet address = 199.7.91.13
d.root-servers.net      has AAAA address 2001:500:2d::d
e.root-servers.net      internet address = 192.203.230.10
e.root-servers.net      has AAAA address 2001:500:a8::e
f.root-servers.net      internet address = 192.5.5.241
f.root-servers.net      has AAAA address 2001:500:2f::f
g.root-servers.net      internet address = 192.112.36.4

mininet@mininet-vm:~$ a
```

7. Multiple application services running on a single machine with a single IP address can be uniquely identified through their port numbers. Each application service will have a unique port number trailing off of the IP address (i.e 127.0.0.1:5000) where 5000 is the unique port number that is assigned to one of the application services in order to be identified.
8. The purpose of the window mechanism in TCP is to control the flow of data between devices. As each device must keep track of the data that they are sending and receiving, the TCP sliding window mechanism helps keep track of those bytes of data by categorizing those bytes into different segments indicating whether the bytes of data are sent and acknowledged, not sent but the recipient is ready, not received but the recipient is ready, sent but not acknowledged, not sent but the recipient is not ready, and so forth.
9. An MTU is the Maximum Transmission Unit that is used to define the byte size that is allowed to pass through the data link. When a packet is larger than the MTU, then it may be necessary to break up the original message into fragments, then reassembled after each of the fragments have been individually sent through the data link.
10. 


204 27.078529000 10.0.2.15 128.114.104.5...K] Seq=2956 Ack=4126 Win=38936 Len=0
Frame 204: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 128.114.104.55 (128.114.104.5
Transmission Control Protocol, Src Port: 60271 (60271), Dst Port: ssh (22), Seq: 2956, Ack:
Source port: 60271 (60271)
Destination port: ssh (22)
[Stream index: 39]
Sequence number: 2956 (relative sequence number)
Acknowledgment number: 4126 (relative ack number)
Header length: 20 bytes
Flags: 0x010 (ACK)
Window size value: 38936
[Calculated window size: 38936]
[Window size scaling factor: -2 (no window scaling used)]
Checksum: 0xf4d2 [validation disabled]
[SEQ/ACK analysis]
0000 00 04 00 01 00 06 08 00 27 27 c6 3a 00 00 08 00 .... ''.:....
0010 45 10 00 28 50 f9 40 00 40 06 f5 0e 0a 00 02 0f E..(P.@. @.......
0020 80 72 68 37 eb 6f 00 16 eb 7a 38 fb 00 11 a4 1f .rh7.o.. .z8.....
0030 50 10 98 18 f4 d2 00 00 P.......