Michael Lau
CMPE 150
Lab 2
2/13/18

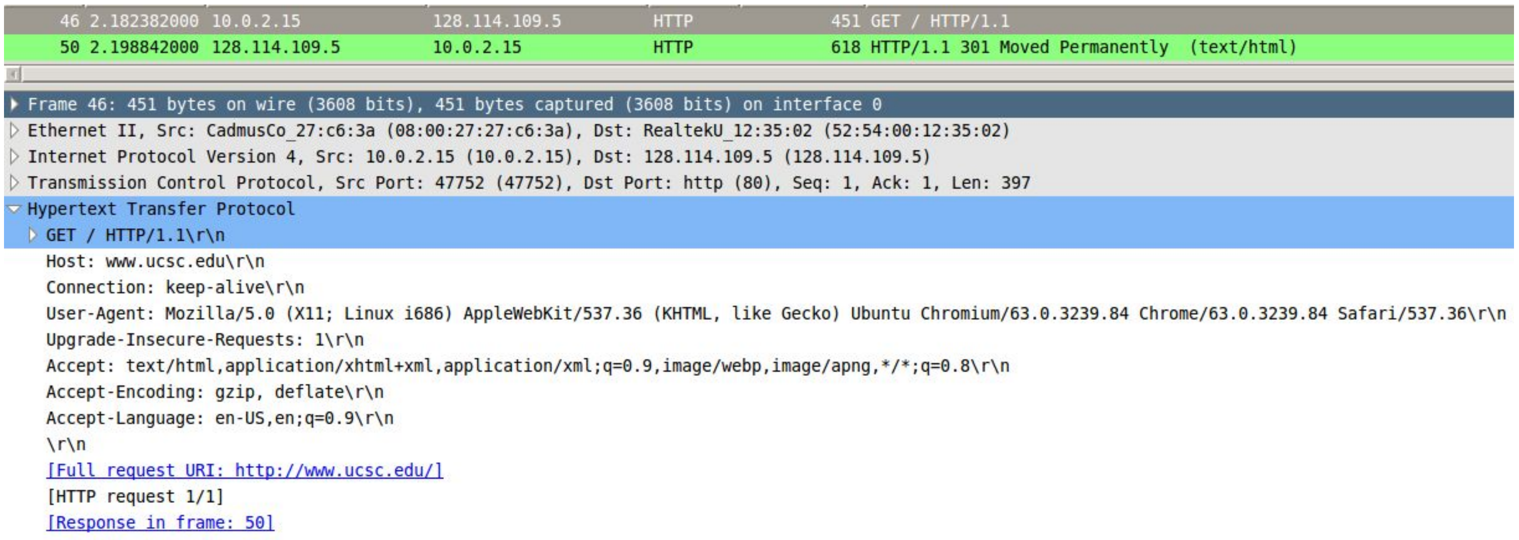
Pre-Lab

1. 1) 404 - Requested resource was not found
   2) 400 - Request can not be processed due to client error
   3) 100 - Request received, proceed to body as normal
   4) 500 - Generic error message when other codes aren't applicable
   5) 429 - User sent in too many requests in a given time

2. 1) Options - a request for information about the communication options available on the request/response chain identified by the Request-URI
   2) Get - retrieve whatever information is identified by the Request-URI
   3) Head - identical to GET except that the server MUST NOT return a message-body in the response
   4) 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
   5) Put - requests that the enclosed entity be stored under the supplied Request-URI
   6) Delete - requests that the origin server delete the resource identified by the Request-URI
   7) Trace - used to invoke a remote, application-layer loop- back of the request message
   8) Connect - reserved for use with a proxy that can dynamically switch to being a tunnel
3. wget -S -O- example.com returns HTTP/1.1 status code 200 and last modified Fri, 09 Aug 2013 23:54:35 GMT. -S prints the status code and last modified and -O- prevents it from saving.
4. This telnet server prints out Star Wars Episode IV: A New Hope's intro in ASCII.
5. DNS resource record is basically a mapping file that tells the DNS server what IP each domain is associated with. MX shows the mail exchanger for UCSC which in our case is google so that's where the incoming emails to UCSC go.
6. This output shows authoritative and non-authoritative name-servers for UCSC.
7. You can uniquely identify them by the TCP Ports. The Source and destination ports will be different for each application.
8. Window mechanism in TCP is a form of flow control because not everyone has the same bandwidth. It allows buffer and queueing of packets before more packets are transferred.
9. MTU (Maximum transmission unit) is the max data that can be communicated in a single network transaction. If a packet is bigger than MTU then it'll be fragmented and sent out in multiple transactions.

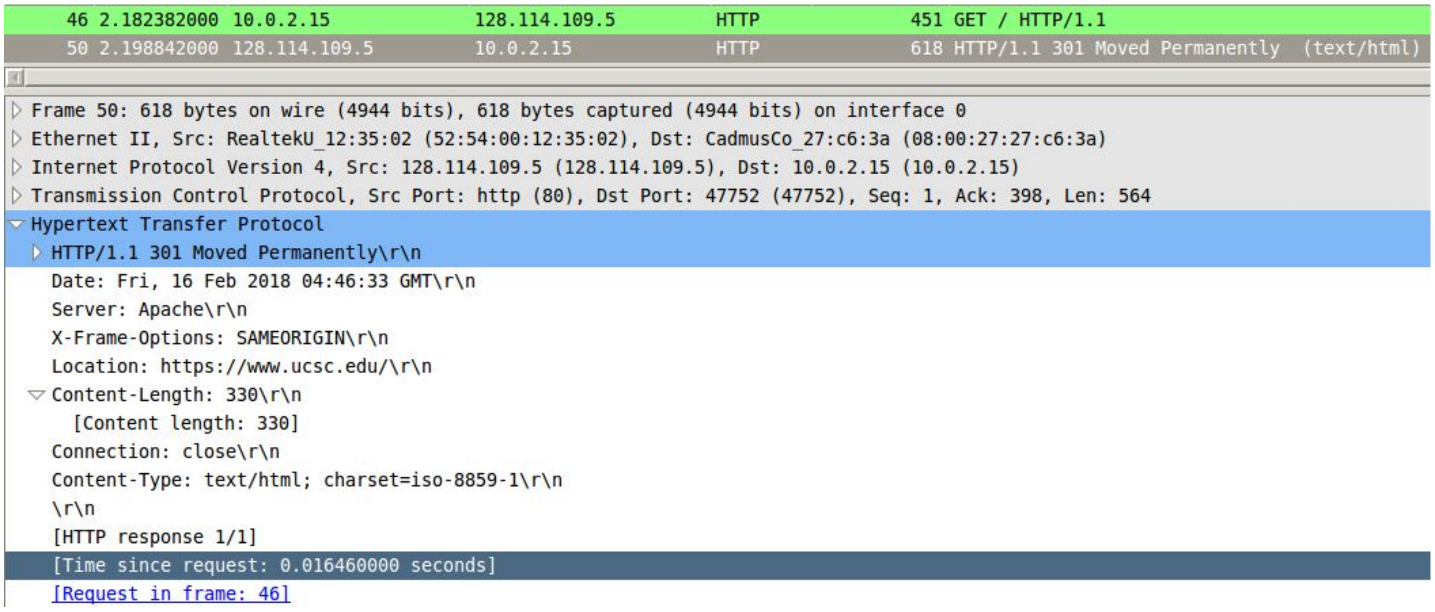
Lab

1. The computer used GET to make the request and the URI was http://www.ucsc.edu/



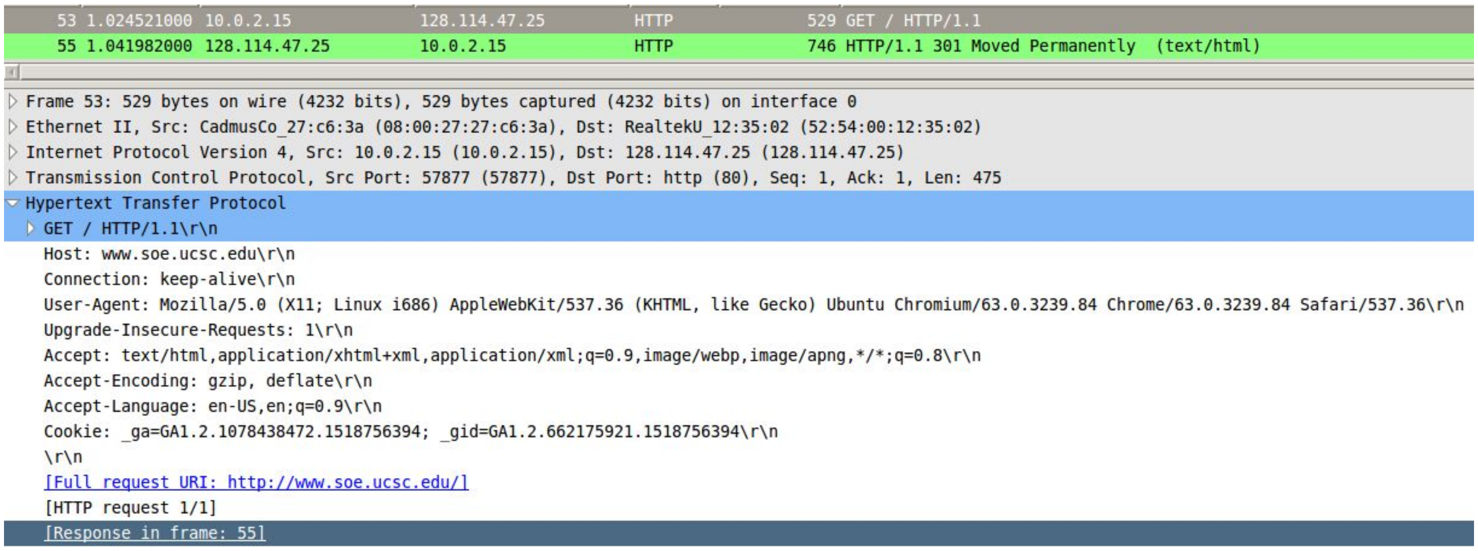
```
46 2.182382000 10.0.2.15        128.114.109.5     HTTP        451 GET / HTTP/1.1
50 2.198842000 128.114.109.5    10.0.2.15         HTTP        618 HTTP/1.1 301 Moved Permanently  (text/html)

Frame 46: 451 bytes on wire (3608 bits), 451 bytes captured (3608 bits) on interface 0
Ethernet II, Src: CadmusCo_27:c6:3a (08:00:27:27:c6:3a), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 128.114.109.5 (128.114.109.5)
Transmission Control Protocol, Src Port: 47752 (47752), Dst Port: http (80), Seq: 1, Ack: 1, Len: 397
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  Host: www.ucsc.edu\r\n
  Connection: keep-alive\r\n
  User-Agent: Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/63.0.3239.84 Chrome/63.0.3239.84 Safari/537.36\r\n
  Upgrade-Insecure-Requests: 1\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: en-US,en;q=0.9\r\n
  \r\n
  [Full request URI: http://www.ucsc.edu/]
  [HTTP request 1/1]
  [Response in frame: 50]
```

2. The response is 301 Moved permanently because ucsc.edu switched to https. The content type was text/html.



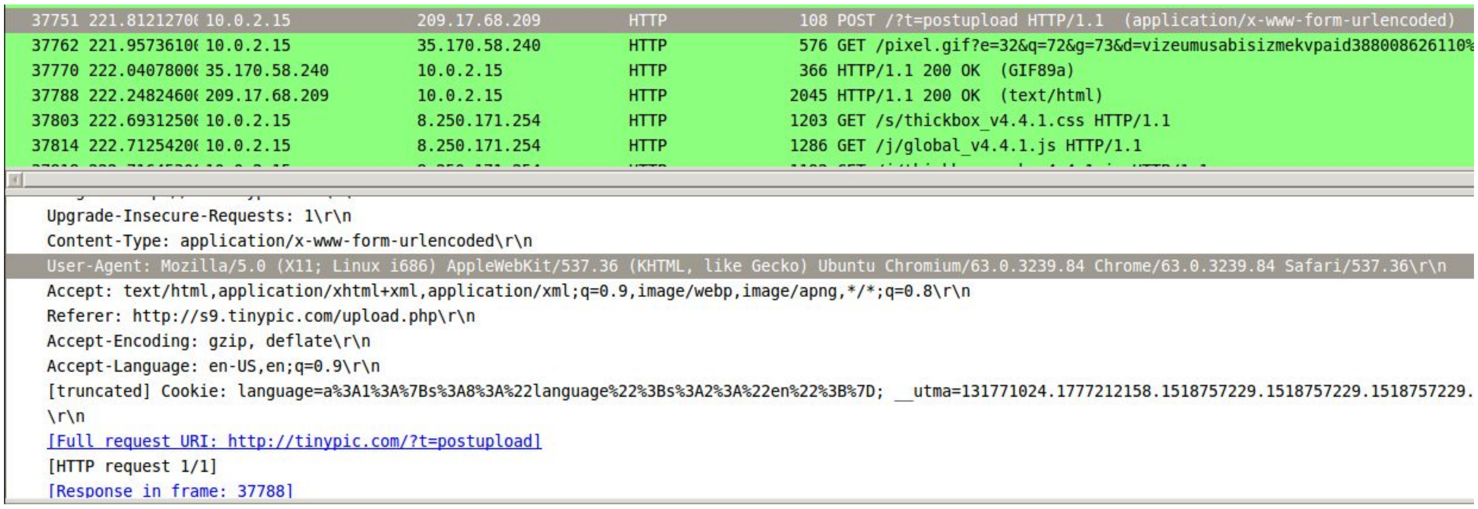
```
46 2.182382000 10.0.2.15        128.114.109.5     HTTP        451 GET / HTTP/1.1
50 2.198842000 128.114.109.5    10.0.2.15         HTTP        618 HTTP/1.1 301 Moved Permanently  (text/html)

Frame 50: 618 bytes on wire (4944 bits), 618 bytes captured (4944 bits) on interface 0
Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: CadmusCo_27:c6:3a (08:00:27:27:c6:3a)
Internet Protocol Version 4, Src: 128.114.109.5 (128.114.109.5), Dst: 10.0.2.15 (10.0.2.15)
Transmission Control Protocol, Src Port: http (80), Dst Port: 47752 (47752), Seq: 1, Ack: 398, Len: 564
Hypertext Transfer Protocol
  HTTP/1.1 301 Moved Permanently\r\n
  Date: Fri, 16 Feb 2018 04:46:33 GMT\r\n
  Server: Apache\r\n
  X-Frame-Options: SAMEORIGIN\r\n
  Location: https://www.ucsc.edu/\r\n
  Content-Length: 330\r\n
    [Content length: 330]
  Connection: close\r\n
  Content-Type: text/html; charset=iso-8859-1\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.016460000 seconds]
  [Request in frame: 46]
```

3. The only difference was the length. They both switched to https and gave http 301 moved permanently.



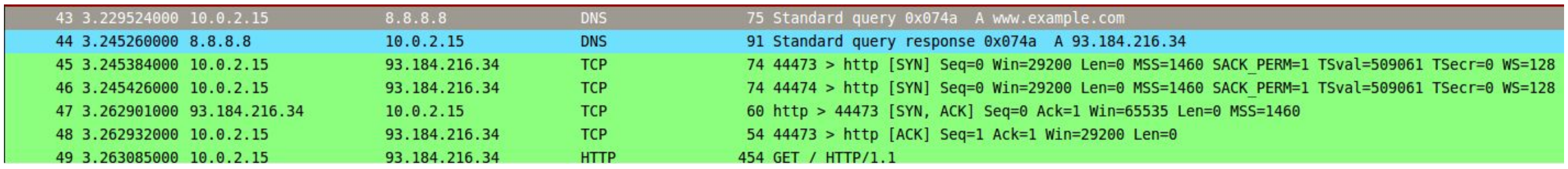
```
   53 1.024521000  10.0.2.15            128.114.47.25        HTTP          529 GET / HTTP/1.1
   55 1.041982000  128.114.47.25        10.0.2.15            HTTP          746 HTTP/1.1 301 Moved Permanently  (text/html)

▷ Frame 53: 529 bytes on wire (4232 bits), 529 bytes captured (4232 bits) on interface 0
▷ Ethernet II, Src: CadmusCo_27:c6:3a (08:00:27:27:c6:3a), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▷ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 128.114.47.25 (128.114.47.25)
▷ Transmission Control Protocol, Src Port: 57877 (57877), Dst Port: http (80), Seq: 1, Ack: 1, Len: 475
▽ Hypertext Transfer Protocol
  ▷ GET / HTTP/1.1\r\n
    Host: www.soe.ucsc.edu\r\n
    Connection: keep-alive\r\n
    User-Agent: Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/63.0.3239.84 Chrome/63.0.3239.84 Safari/537.36\r\n
    Upgrade-Insecure-Requests: 1\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9\r\n
    Cookie: _ga=GA1.2.1078438472.1518756394; _gid=GA1.2.662175921.1518756394\r\n
    \r\n
    [Full request URI: http://www.soe.ucsc.edu/]
    [HTTP request 1/1]
    [Response in frame: 55]
```

4. This one was a bit tricky since most page retrieval is GET and many sites moved to https and made things harder to record with wireshark. I managed to find a file uploading site www.tinypic.com that was unsecured and got status code POST when I uploaded a picture.



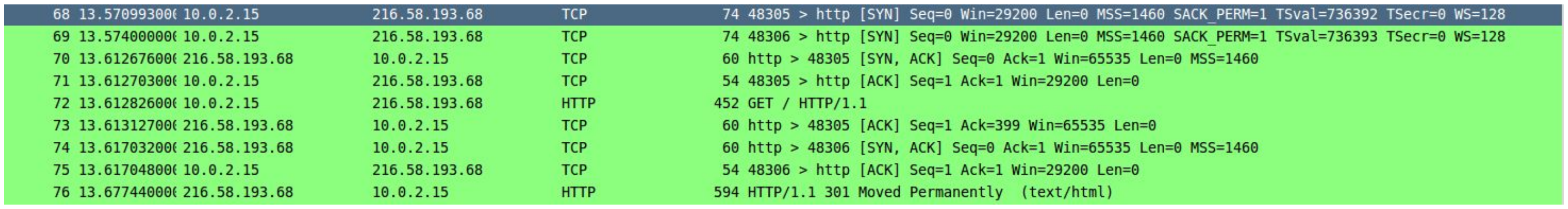
```
 37751 221.8121270( 10.0.2.15         209.17.68.209        HTTP          108 POST /?t=postupload HTTP/1.1  (application/x-www-form-urlencoded)
 37762 221.9573610( 10.0.2.15         35.170.58.240        HTTP          576 GET /pixel.gif?e=32&q=72&g=73&d=vizeumusabisizmekvpaid388008626110%
 37770 222.0407800( 35.170.58.240     10.0.2.15            HTTP          366 HTTP/1.1 200 OK  (GIF89a)
 37788 222.2482460( 209.17.68.209     10.0.2.15            HTTP         2045 HTTP/1.1 200 OK  (text/html)
 37803 222.6931250( 10.0.2.15         8.250.171.254        HTTP         1203 GET /s/thickbox_v4.4.1.css HTTP/1.1
 37814 222.7125420( 10.0.2.15         8.250.171.254        HTTP         1286 GET /j/global_v4.4.1.js HTTP/1.1

    Upgrade-Insecure-Requests: 1\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    User-Agent: Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/63.0.3239.84 Chrome/63.0.3239.84 Safari/537.36\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8\r\n
    Referer: http://s9.tinypic.com/upload.php\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9\r\n
    [truncated] Cookie: language=a%3A1%3A%7Bs%3A8%3A%22language%22%3Bs%3A2%3A%22en%22%3B%7D; __utma=131771024.1777212158.1518757229.1518757229.1518757229.
    \r\n
    [Full request URI: http://tinypic.com/?t=postupload]
    [HTTP request 1/1]
    [Response in frame: 37788]
```

5. Yes, the computer first send TCP packets and must contact the DNS server to get the IP of www.example.com I've set my DNS on my modem to Google's DNS (8.8.8.8) and it responded back with www.example.com's IP. The computer needs these steps because the computer doesn't know the IP of www.example.com but Google does, if not another DNS would.



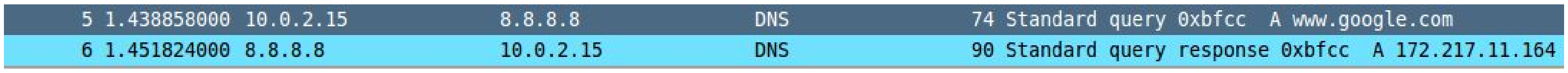
```
   43 3.229524000  10.0.2.15            8.8.8.8              DNS           75 Standard query 0x074a  A www.example.com
   44 3.245260000  8.8.8.8              10.0.2.15            DNS           91 Standard query response 0x074a  A 93.184.216.34
   45 3.245384000  10.0.2.15            93.184.216.34        TCP           74 44473 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=509061 TSecr=0 WS=128
   46 3.245426000  10.0.2.15            93.184.216.34        TCP           74 44474 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=509061 TSecr=0 WS=128
   47 3.262901000  93.184.216.34        10.0.2.15            TCP           60 http > 44473 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
   48 3.262932000  10.0.2.15            93.184.216.34        TCP           54 44473 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
   49 3.263085000  10.0.2.15            93.184.216.34        HTTP         454 GET / HTTP/1.1
```

6. No extra steps were necessary when going to http://216.58.193.68 (Google.com) because DNS server wasn't required as the IP was directly inputted. The only thing done was obviously send TCP packets directly to http://216.58.193.68.



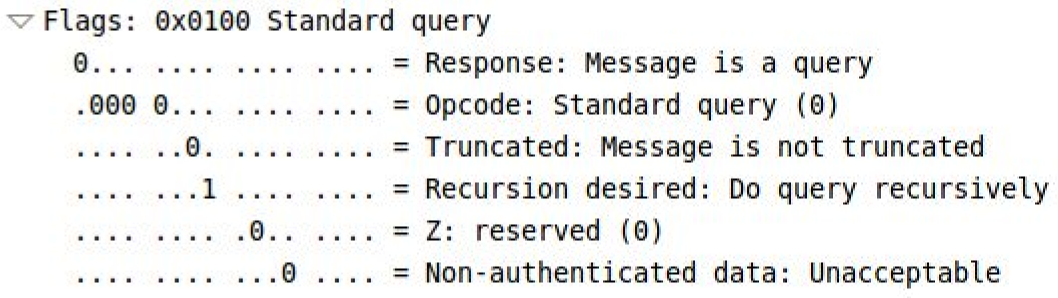
```
68 13.57099300( 10.0.2.15       216.58.193.68   TCP    74 48305 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=736392 TSecr=0 WS=128
69 13.57400000( 10.0.2.15       216.58.193.68   TCP    74 48306 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=736393 TSecr=0 WS=128
70 13.61267600( 216.58.193.68   10.0.2.15       TCP    60 http > 48305 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
71 13.61270300( 10.0.2.15       216.58.193.68   TCP    54 48305 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
72 13.61282600( 10.0.2.15       216.58.193.68   HTTP  452 GET / HTTP/1.1
73 13.61312700( 216.58.193.68   10.0.2.15       TCP    60 http > 48305 [ACK] Seq=1 Ack=399 Win=65535 Len=0
74 13.61703200( 216.58.193.68   10.0.2.15       TCP    60 http > 48306 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
75 13.61704800( 10.0.2.15       216.58.193.68   TCP    54 48306 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
76 13.67744000( 216.58.193.68   10.0.2.15       HTTP  594 HTTP/1.1 301 Moved Permanently  (text/html)
```

7. Nslookup Type A gave us the return response of 172.217.11.164



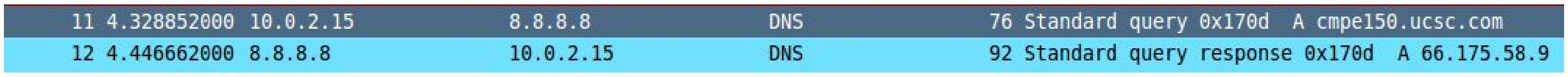
```
5 1.438858000  10.0.2.15   8.8.8.8     DNS   74 Standard query 0xbfcc  A www.google.com
6 1.451824000  8.8.8.8     10.0.2.15   DNS   90 Standard query response 0xbfcc  A 172.217.11.164
```

8. Yes, the computer wants to request recursively. It tells us this in the query flag.



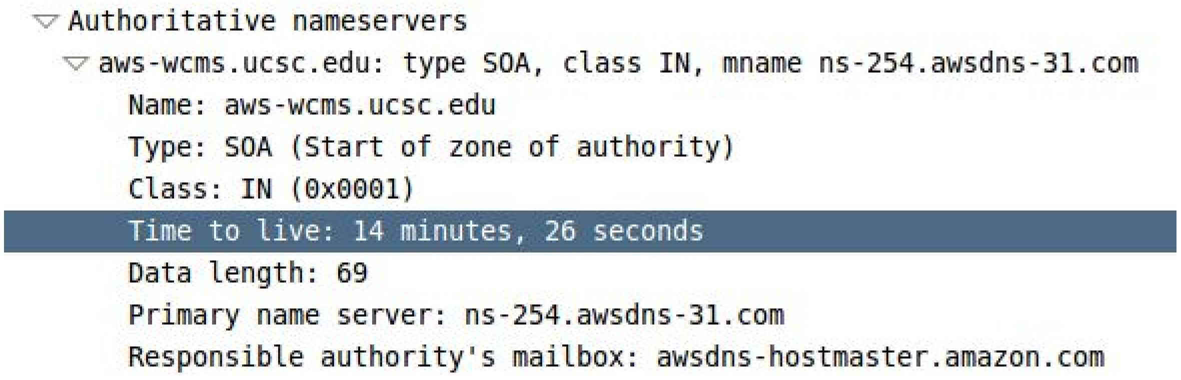
```
▽ Flags: 0x0100 Standard query
    0... .... .... .... = Response: Message is a query
    .000 0... .... .... = Opcode: Standard query (0)
    .... ..0. .... .... = Truncated: Message is not truncated
    .... ...1 .... .... = Recursion desired: Do query recursively
    .... .... .0.. .... = Z: reserved (0)
    .... .... ...0 .... = Non-authenticated data: Unacceptable
```

9. The request resolved and gave me the IP 66.175.58.9.



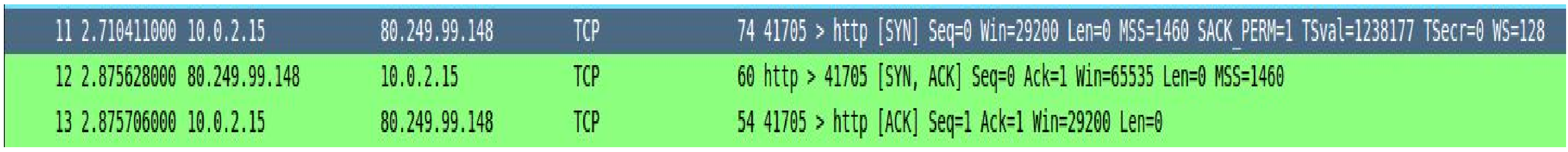
```
11 4.328852000  10.0.2.15   8.8.8.8     DNS   76 Standard query 0x170d  A cmpe150.ucsc.com
12 4.446662000  8.8.8.8     10.0.2.15   DNS   92 Standard query response 0x170d  A 66.175.58.9
```

10. The authoritative name server for ucsc.edu was not found but it was found with www.ucsc.edu. The authoritative name server can be found in terminal's output or wireshark's queries tab. It is obtained by nslookup -type=ns www.ucsc.edu. The name server is aws-wcsm.ucsc.edu



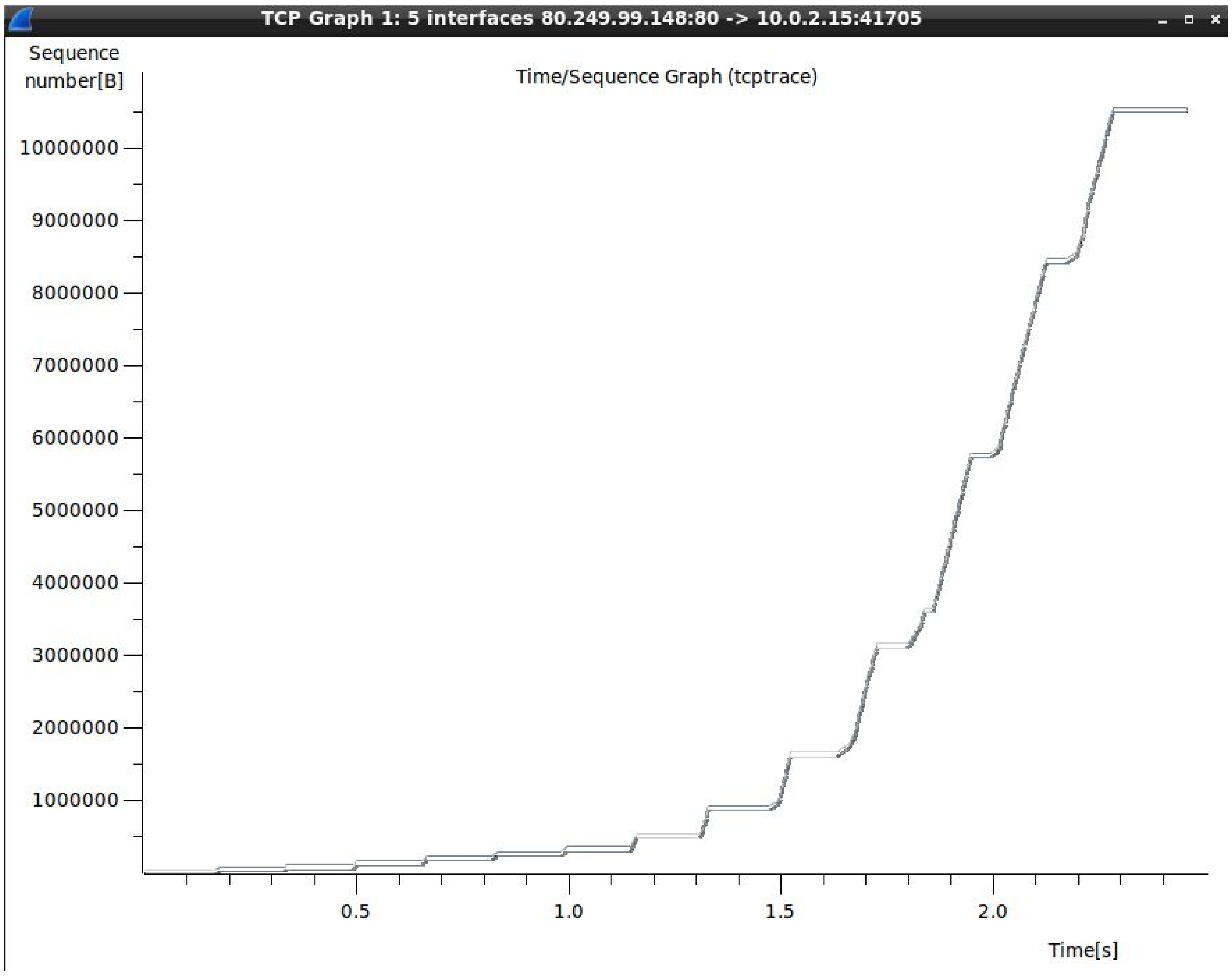
```
▽ Authoritative nameservers
  ▽ aws-wcms.ucsc.edu: type SOA, class IN, mname ns-254.awsdns-31.com
      Name: aws-wcms.ucsc.edu
      Type: SOA (Start of zone of authority)
      Class: IN (0x0001)
      Time to live: 14 minutes, 26 seconds
      Data length: 69
      Primary name server: ns-254.awsdns-31.com
      Responsible authority's mailbox: awsdns-hostmaster.amazon.com
```

11. The initial window size advertised by the computer was 29200 and the initial window size advertised by the server was 65535.

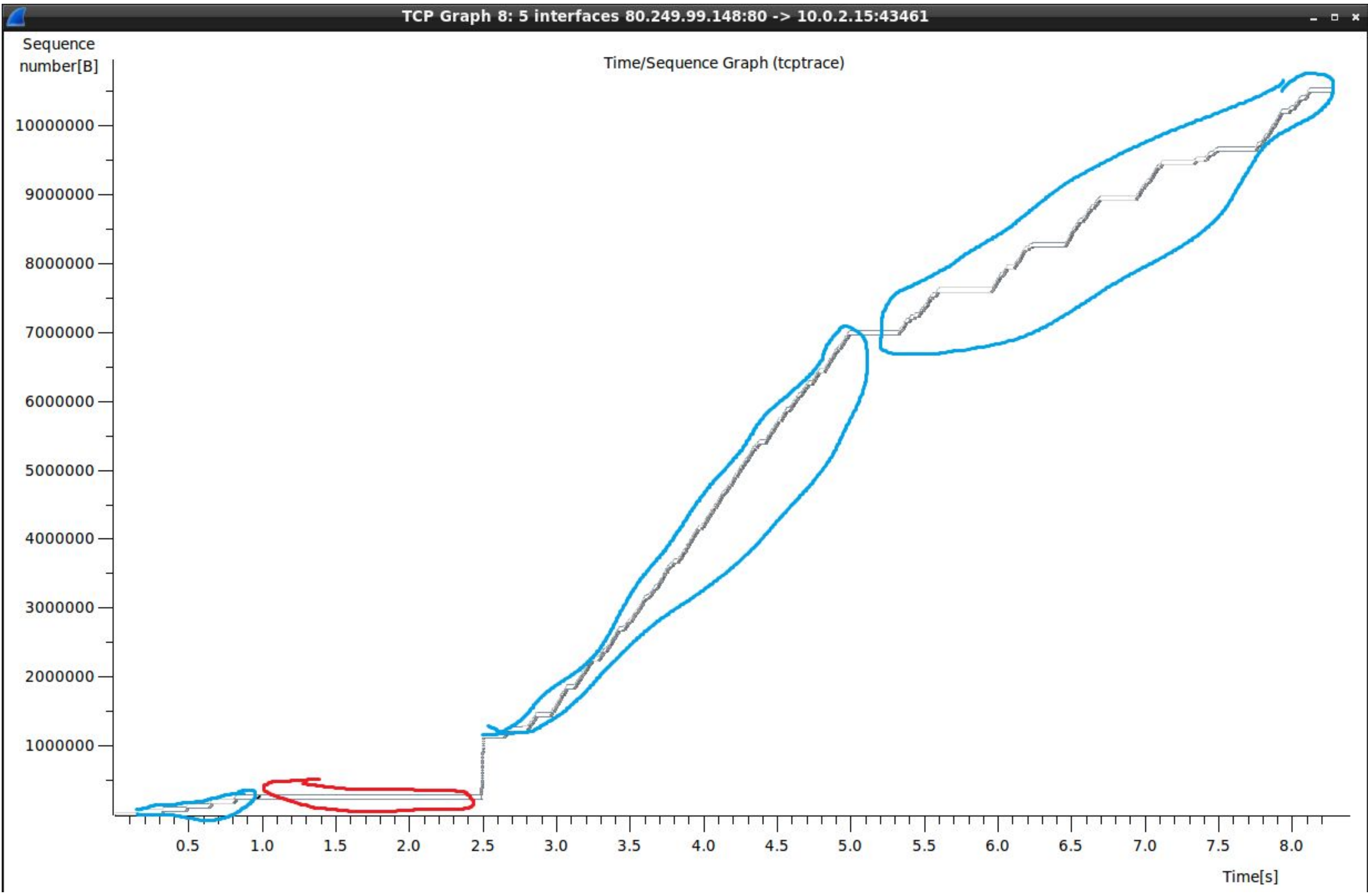


```
11 2.710411000 10.0.2.15      80.249.99.148   TCP   74 41705 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1238177 TSecr=0 WS=128
12 2.875628000 80.249.99.148  10.0.2.15       TCP   60 http > 41705 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
13 2.875706000 10.0.2.15      80.249.99.148   TCP   54 41705 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
```

12. The graph is showing transfer of packets in sequence numbers over time. From the graph, I could tell that it's a slow start style and doubles throughout with brief periods of pauses. The steeper the slope the faster it is transferring packets.



13. The red zone is 100% loss, blue zone is 0% loss and slow start/congestion control. The flat lines inside the blue zone is where congestion control occurs. This graph shows the transfer of packets in sequence numbers over time. The steeper the slope, the faster the transfer of packets.


TCP Graph 8: 5 interfaces 80.249.99.148:80 -> 10.0.2.15:43461
Time/Sequence Graph (tcptrace)
Sequence number[B]
Time[s]