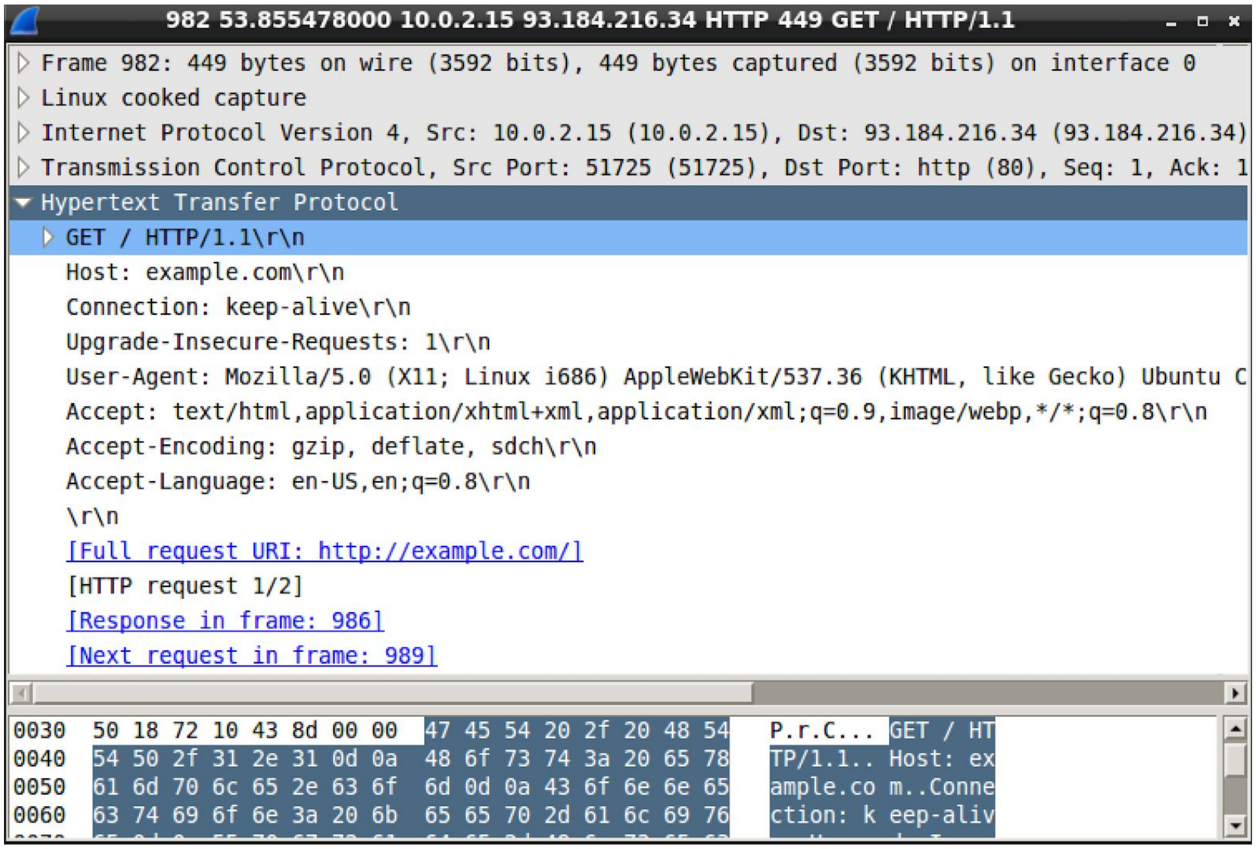
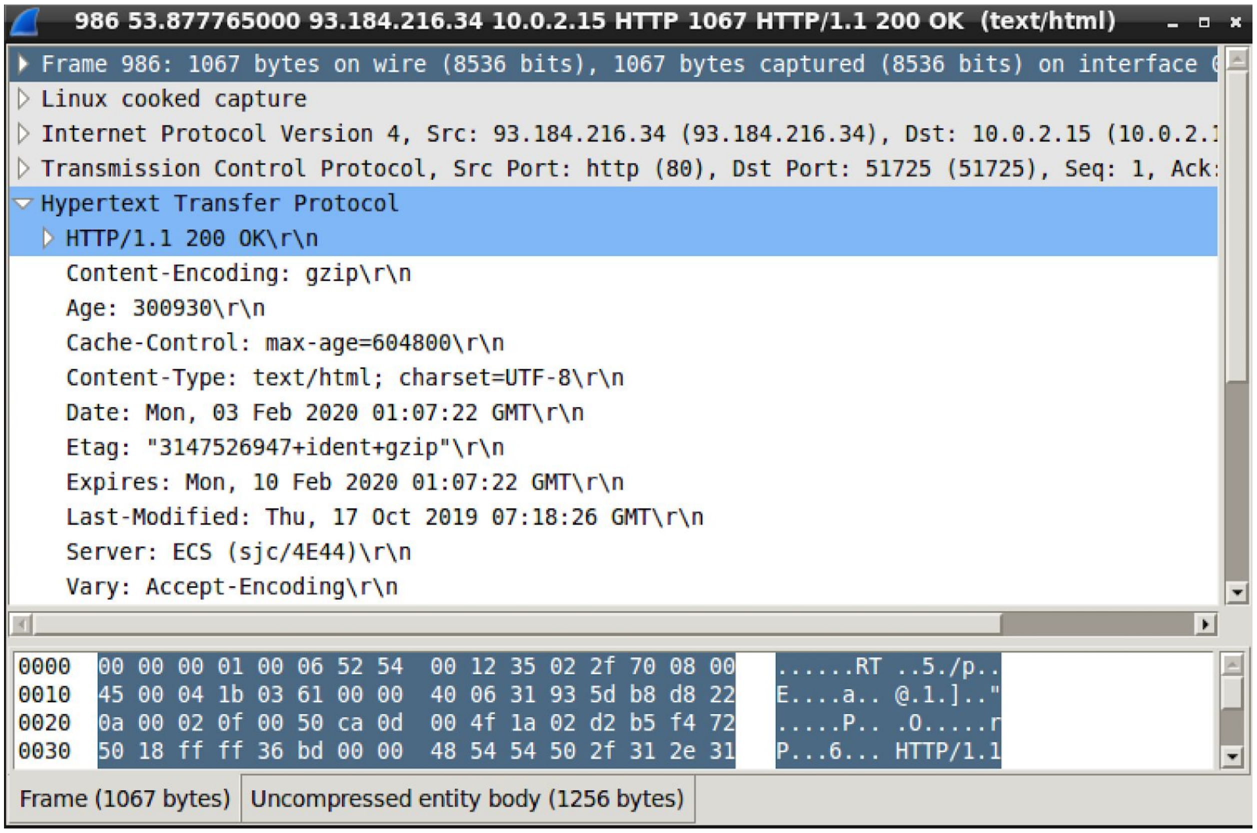
Lab 2

1. My computer had used the HTTP GET request in order to make the request to http://www.example.com. My computer requested the URI '/' or root as the request to http://www.example.com is also equivalent to the request of that server's home page, which is also known as the root page.



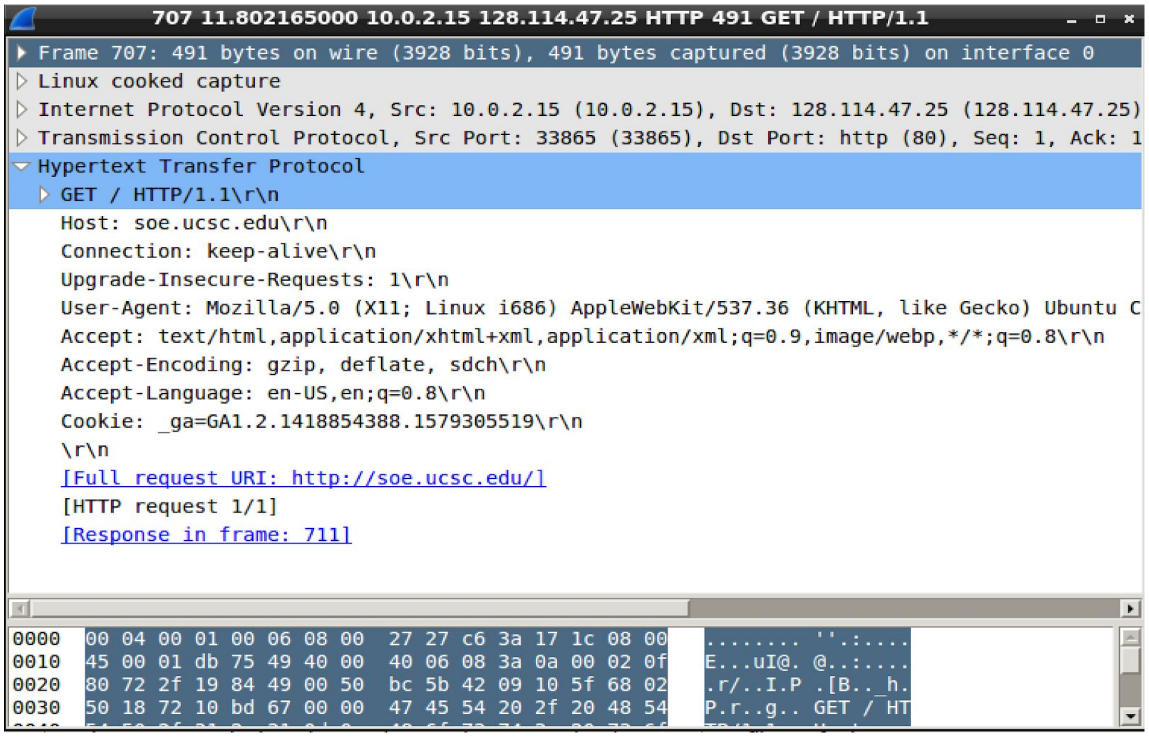
```
982 53.855478000 10.0.2.15 93.184.216.34 HTTP 449 GET / HTTP/1.1
▷ Frame 982: 449 bytes on wire (3592 bits), 449 bytes captured (3592 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 93.184.216.34 (93.184.216.34)
▷ Transmission Control Protocol, Src Port: 51725 (51725), Dst Port: http (80), Seq: 1, Ack: 1
▼ Hypertext Transfer Protocol
  ▷ GET / HTTP/1.1\r\n
    Host: example.com\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu C
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
    Accept-Encoding: gzip, deflate, sdch\r\n
    Accept-Language: en-US,en;q=0.8\r\n
    \r\n
    [Full request URI: http://example.com/]
    [HTTP request 1/2]
    [Response in frame: 986]
    [Next request in frame: 989]

0030  50 18 72 10 43 8d 00 00  47 45 54 20 2f 20 48 54    P.r.C... GET / HT
0040  54 50 2f 31 2e 31 0d 0a  48 6f 73 74 3a 20 65 78    TP/1.1.. Host: ex
0050  61 6d 70 6c 65 2e 63 6f  6d 0d 0a 43 6f 6e 6e 65    ample.co m..Conne
0060  63 74 69 6f 6e 3a 20 6b  65 65 70 2d 61 6c 69 76    ction: k eep-aliv
```

2. The HTTP response that the server issued in response to my request was a 200 OK. The content type of the response was text/html.

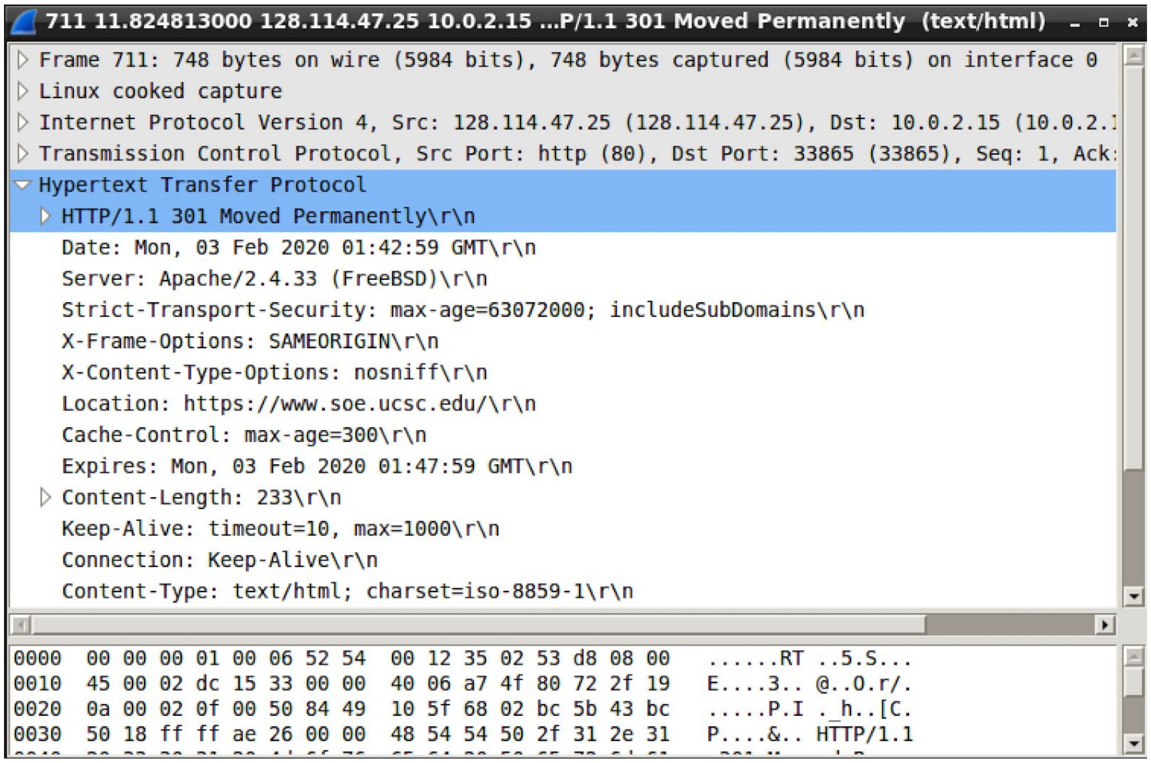


```
986 53.877765000 93.184.216.34 10.0.2.15 HTTP 1067 HTTP/1.1 200 OK  (text/html)
▸ Frame 986: 1067 bytes on wire (8536 bits), 1067 bytes captured (8536 bits) on interface (
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 93.184.216.34 (93.184.216.34), Dst: 10.0.2.15 (10.0.2.1
▷ Transmission Control Protocol, Src Port: http (80), Dst Port: 51725 (51725), Seq: 1, Ack:
▽ Hypertext Transfer Protocol
  ▷ HTTP/1.1 200 OK\r\n
    Content-Encoding: gzip\r\n
    Age: 300930\r\n
    Cache-Control: max-age=604800\r\n
    Content-Type: text/html; charset=UTF-8\r\n
    Date: Mon, 03 Feb 2020 01:07:22 GMT\r\n
    Etag: "3147526947+ident+gzip"\r\n
    Expires: Mon, 10 Feb 2020 01:07:22 GMT\r\n
    Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT\r\n
    Server: ECS (sjc/4E44)\r\n
    Vary: Accept-Encoding\r\n

0000  00 00 00 01 00 06 52 54  00 12 35 02 2f 70 08 00   ......RT ..5./p..
0010  45 00 04 1b 03 61 00 00  40 06 31 93 5d b8 d8 22   E....a.. @.1.].."
0020  0a 00 02 0f 00 50 ca 0d  00 4f 1a 02 d2 b5 f4 72   .....P.. .O.....r
0030  50 18 ff ff 36 bd 00 00  48 54 54 50 2f 31 2e 31   P...6... HTTP/1.1
```

3. The difference was the response that the server had sent, which was a 301 moved permanently response. My guess is that because I had tried to access the unsecure page of soe.ucsc.edu, the request was redirected to the secured page of the website and the response results in a 301 moved permanently.

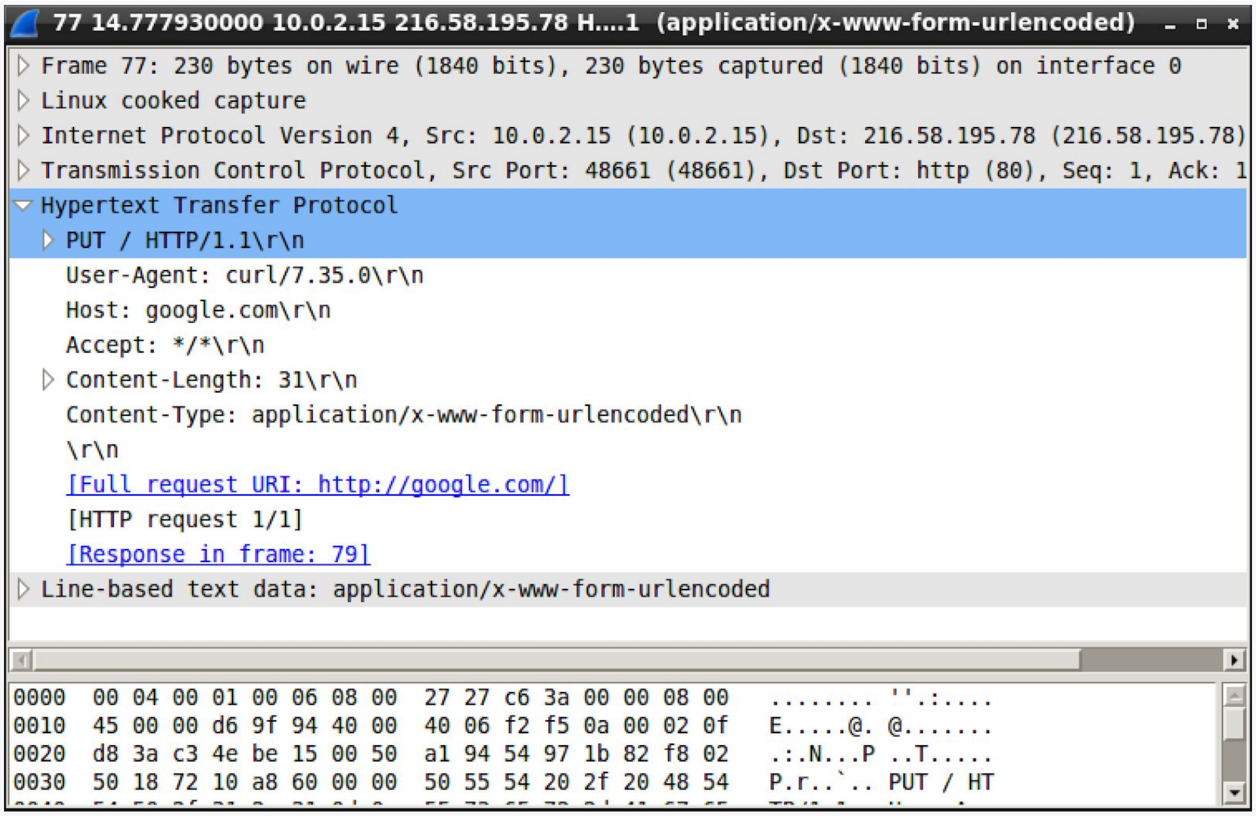


```
707 11.802165000 10.0.2.15 128.114.47.25 HTTP 491 GET / HTTP/1.1

▸ Frame 707: 491 bytes on wire (3928 bits), 491 bytes captured (3928 bits) on interface 0
▷ Linux cooked capture
▷ 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: 33865 (33865), Dst Port: http (80), Seq: 1, Ack: 1
▽ Hypertext Transfer Protocol
  ▷ GET / HTTP/1.1\r\n
    Host: soe.ucsc.edu\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu C
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
    Accept-Encoding: gzip, deflate, sdch\r\n
    Accept-Language: en-US,en;q=0.8\r\n
    Cookie: _ga=GA1.2.1418854388.1579305519\r\n
    \r\n
    [Full request URI: http://soe.ucsc.edu/]
    [HTTP request 1/1]
    [Response in frame: 711]

0000  00 04 00 01 00 06 08 00  27 27 c6 3a 17 1c 08 00   ........ ''.:....
0010  45 00 01 db 75 49 40 00  40 06 08 3a 0a 00 02 0f   E...uI@. @..:....
0020  80 72 2f 19 84 49 00 50  bc 5b 42 09 10 5f 68 02   .r/..I.P .[B.._h.
0030  50 18 72 10 bd 67 00 00  47 45 54 20 2f 20 48 54   P.r..g.. GET / HT
```



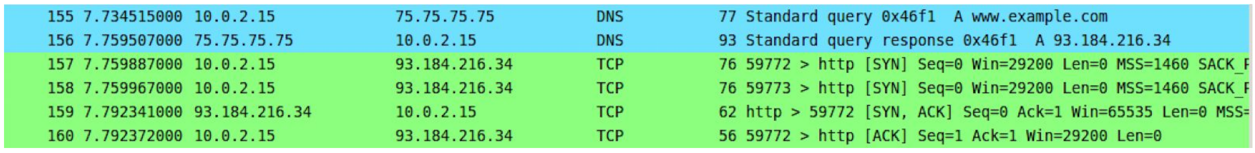
```
711 11.824813000 128.114.47.25 10.0.2.15 ...P/1.1 301 Moved Permanently  (text/html)

▷ Frame 711: 748 bytes on wire (5984 bits), 748 bytes captured (5984 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 128.114.47.25 (128.114.47.25), Dst: 10.0.2.15 (10.0.2.1
▷ Transmission Control Protocol, Src Port: http (80), Dst Port: 33865 (33865), Seq: 1, Ack:
▽ Hypertext Transfer Protocol
  ▷ HTTP/1.1 301 Moved Permanently\r\n
    Date: Mon, 03 Feb 2020 01:42:59 GMT\r\n
    Server: Apache/2.4.33 (FreeBSD)\r\n
    Strict-Transport-Security: max-age=63072000; includeSubDomains\r\n
    X-Frame-Options: SAMEORIGIN\r\n
    X-Content-Type-Options: nosniff\r\n
    Location: https://www.soe.ucsc.edu/\r\n
    Cache-Control: max-age=300\r\n
    Expires: Mon, 03 Feb 2020 01:47:59 GMT\r\n
  ▷ Content-Length: 233\r\n
    Keep-Alive: timeout=10, max=1000\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=iso-8859-1\r\n

0000  00 00 00 01 00 06 52 54  00 12 35 02 53 d8 08 00   ......RT ..5.S...
0010  45 00 02 dc 15 33 00 00  40 06 a7 4f 80 72 2f 19   E....3.. @..O.r/.
0020  0a 00 02 0f 00 50 84 49  10 5f 68 02 bc 5b 43 bc   .....P.I ._h..[C.
0030  50 18 ff ff ae 26 00 00  48 54 54 50 2f 31 2e 31   P....&.. HTTP/1.1
```

4. I had created an HTTP message of PUT to the server google.com by inputting a curl request of PUT to the destination google.com. The command looks like this:
   `curl -X PUT -d 'abc' google.com.`



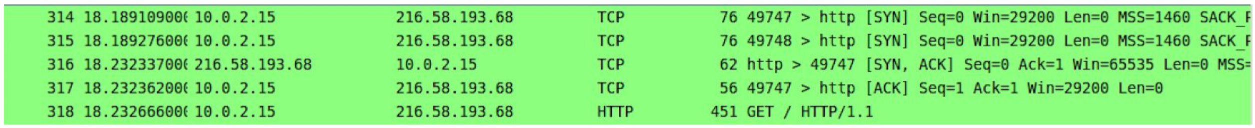
```
77 14.777930000 10.0.2.15 216.58.195.78 H....1  (application/x-www-form-urlencoded)
▷ Frame 77: 230 bytes on wire (1840 bits), 230 bytes captured (1840 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 216.58.195.78 (216.58.195.78)
▷ Transmission Control Protocol, Src Port: 48661 (48661), Dst Port: http (80), Seq: 1, Ack: 1
▽ Hypertext Transfer Protocol
  ▷ PUT / HTTP/1.1\r\n
    User-Agent: curl/7.35.0\r\n
    Host: google.com\r\n
    Accept: */*\r\n
  ▷ Content-Length: 31\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    \r\n
    [Full request URI: http://google.com/]
    [HTTP request 1/1]
    [Response in frame: 79]
▷ Line-based text data: application/x-www-form-urlencoded

0000   00 04 00 01 00 06 08 00  27 27 c6 3a 00 00 08 00   ........ ''.:....
0010   45 00 00 d6 9f 94 40 00  40 06 f2 f5 0a 00 02 0f   E.....@. @.......
0020   d8 3a c3 4e be 15 00 50  a1 94 54 97 1b 82 f8 02   .:.N...P ..T.....
0030   50 18 72 10 a8 60 00 00  50 55 54 20 2f 20 48 54   P.r..`.. PUT / HT
```

5. There were steps taken by my computer before the webpage was loaded. As shown in the screenshot below, there exists 6 packets before the GET request to the webpage was executed. The first two packets are the DNS packets which I believe query the server that I am trying to reach. The 4 other packets are TCP packets whose purpose is to hop to different servers until it reaches the server that I have requested before the GET request can be executed.



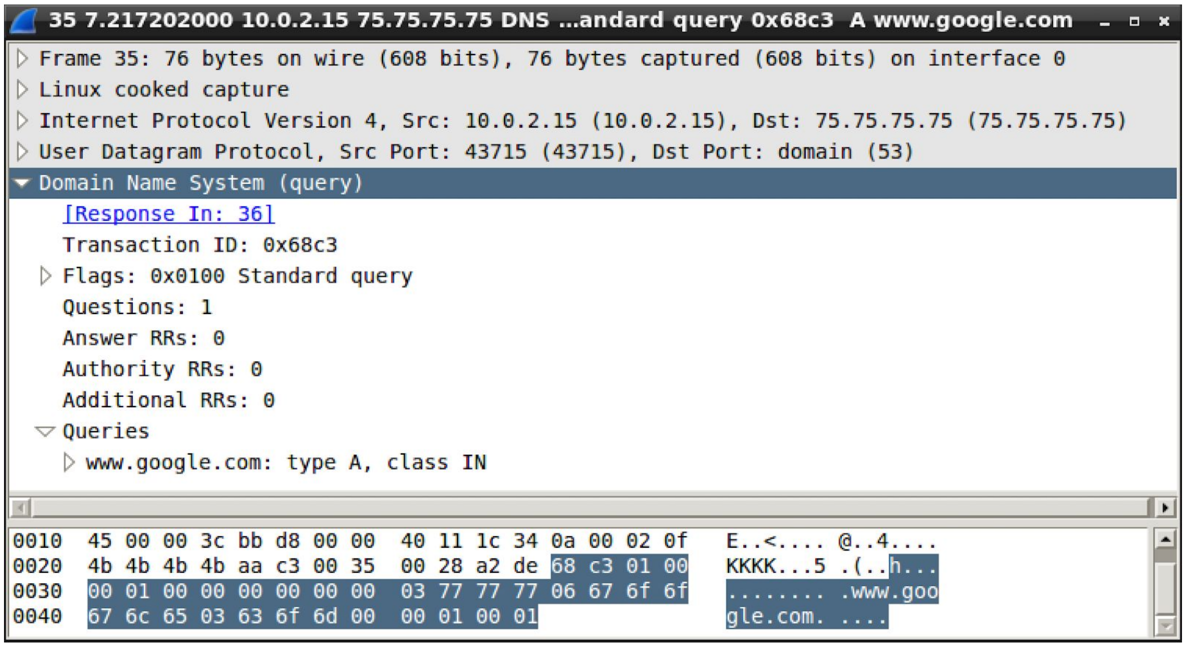
```
155 7.734515000 10.0.2.15       75.75.75.75     DNS    77 Standard query 0x46f1  A www.example.com
156 7.759507000 75.75.75.75     10.0.2.15       DNS    93 Standard query response 0x46f1  A 93.184.216.34
157 7.759887000 10.0.2.15       93.184.216.34   TCP    76 59772 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_P
158 7.759967000 10.0.2.15       93.184.216.34   TCP    76 59773 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_P
159 7.792341000 93.184.216.34   10.0.2.15       TCP    62 http > 59772 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=
160 7.792372000 10.0.2.15       93.184.216.34   TCP    56 59772 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
```

6. There were steps taken by my computer before the webpage was loaded, but the steps only consisted of hopping to the server that was requested. Unlike the previous question, there was no DNS query as I had only provided an IP address instead of a URL, so my computer had no reason to decode a non-provided URL into an already provided IP address.

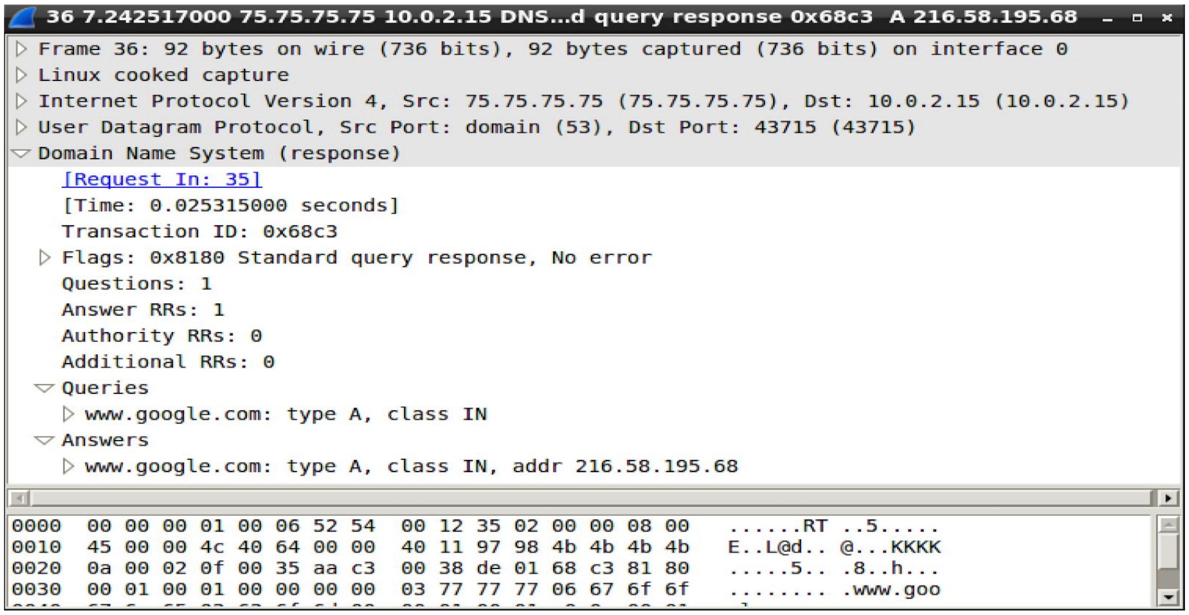


```
314 18.18910900( 10.0.2.15       216.58.193.68   TCP    76 49747 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_F
315 18.18927600( 10.0.2.15       216.58.193.68   TCP    76 49748 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_F
316 18.23233700( 216.58.193.68   10.0.2.15       TCP    62 http > 49747 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=
317 18.23236200( 10.0.2.15       216.58.193.68   TCP    56 49747 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
318 18.23266600( 10.0.2.15       216.58.193.68   HTTP  451 GET / HTTP/1.1
```

7. The request was resolved and the IP address that I was given for www.google.com is 216.58.195.68.

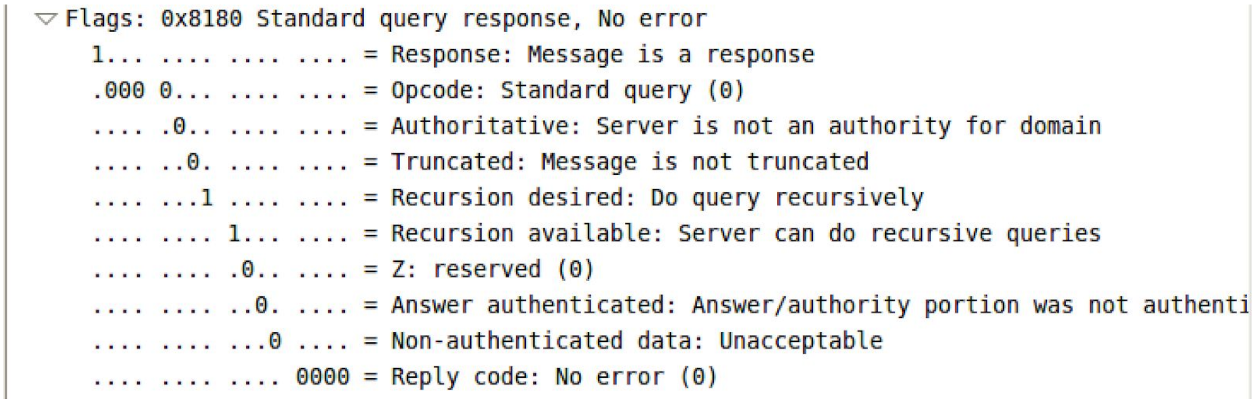


```
35 7.217202000 10.0.2.15 75.75.75.75 DNS ...andard query 0x68c3  A www.google.com
Frame 35: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 75.75.75.75 (75.75.75.75)
User Datagram Protocol, Src Port: 43715 (43715), Dst Port: domain (53)
Domain Name System (query)
    [Response In: 36]
    Transaction ID: 0x68c3
  Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  Queries
    www.google.com: type A, class IN

0010  45 00 00 3c bb d8 00 00  40 11 1c 34 0a 00 02 0f   E..<.... @..4....
0020  4b 4b 4b 4b aa c3 00 35  00 28 a2 de 68 c3 01 00   KKKK...5 .(..h...
0030  00 01 00 00 00 00 00 00  03 77 77 77 06 67 6f 6f   ........ .www.goo
0040  67 6c 65 03 63 6f 6d 00  00 01 00 01               gle.com. ....
```



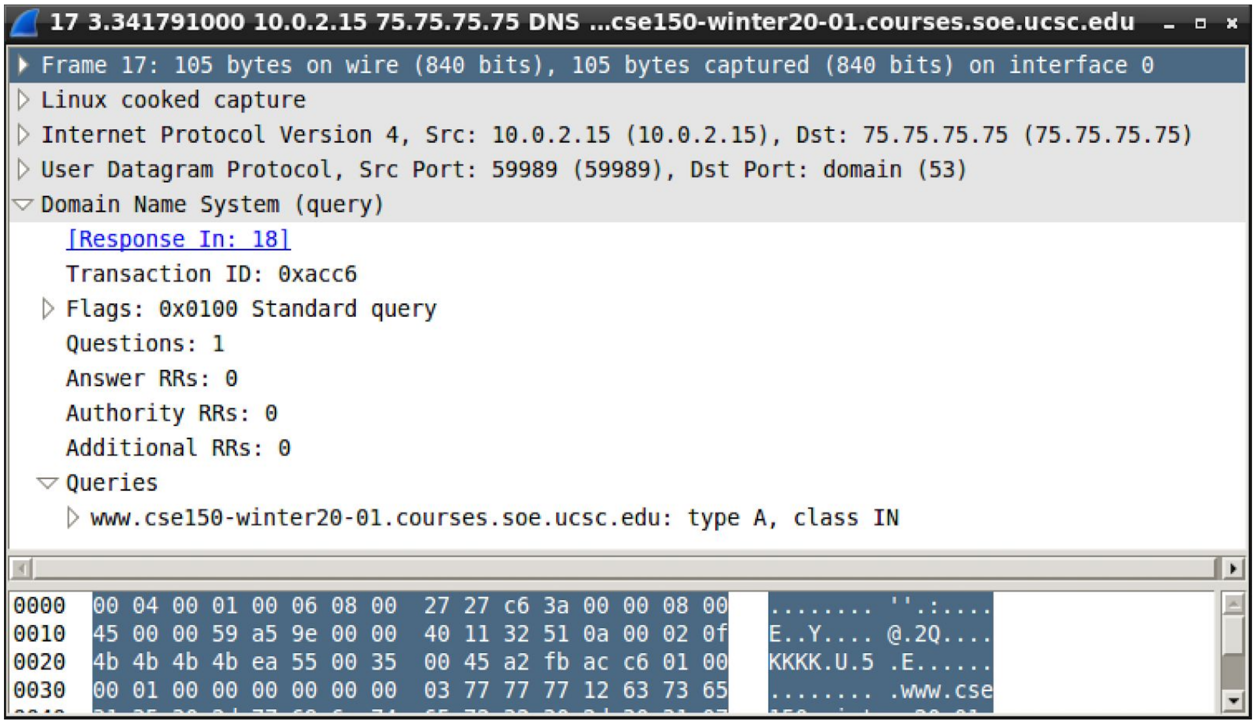
```
36 7.242517000 75.75.75.75 10.0.2.15 DNS...d query response 0x68c3  A 216.58.195.68
Frame 36: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 75.75.75.75 (75.75.75.75), Dst: 10.0.2.15 (10.0.2.15)
User Datagram Protocol, Src Port: domain (53), Dst Port: 43715 (43715)
Domain Name System (response)
    [Request In: 35]
    [Time: 0.025315000 seconds]
    Transaction ID: 0x68c3
  Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  Queries
    www.google.com: type A, class IN
  Answers
    www.google.com: type A, class IN, addr 216.58.195.68

0000  00 00 00 01 00 06 52 54  00 12 35 02 00 00 08 00   ......RT ..5.....
0010  45 00 00 4c 40 64 00 00  40 11 97 98 4b 4b 4b 4b   E..L@d.. @...KKKK
0020  0a 00 02 0f 00 35 aa c3  00 38 de 01 68 c3 81 80   .....5.. .8..h...
0030  00 01 00 01 00 00 00 00  03 77 77 77 06 67 6f 6f   ........ .www.goo
```

8. My computer had wanted to complete the request recursively. As shown in the screenshot below, under the Flags section, there exists an Authoritative section where it displays that the server that I am connected to is not an authority of domain, which means that the address I am connecting to is not getting resolved by the Authoritative DNS, but instead by the Recursive DNS. In addition, there is a field called Recursion desired, which is set to ‘Do query recursively’.

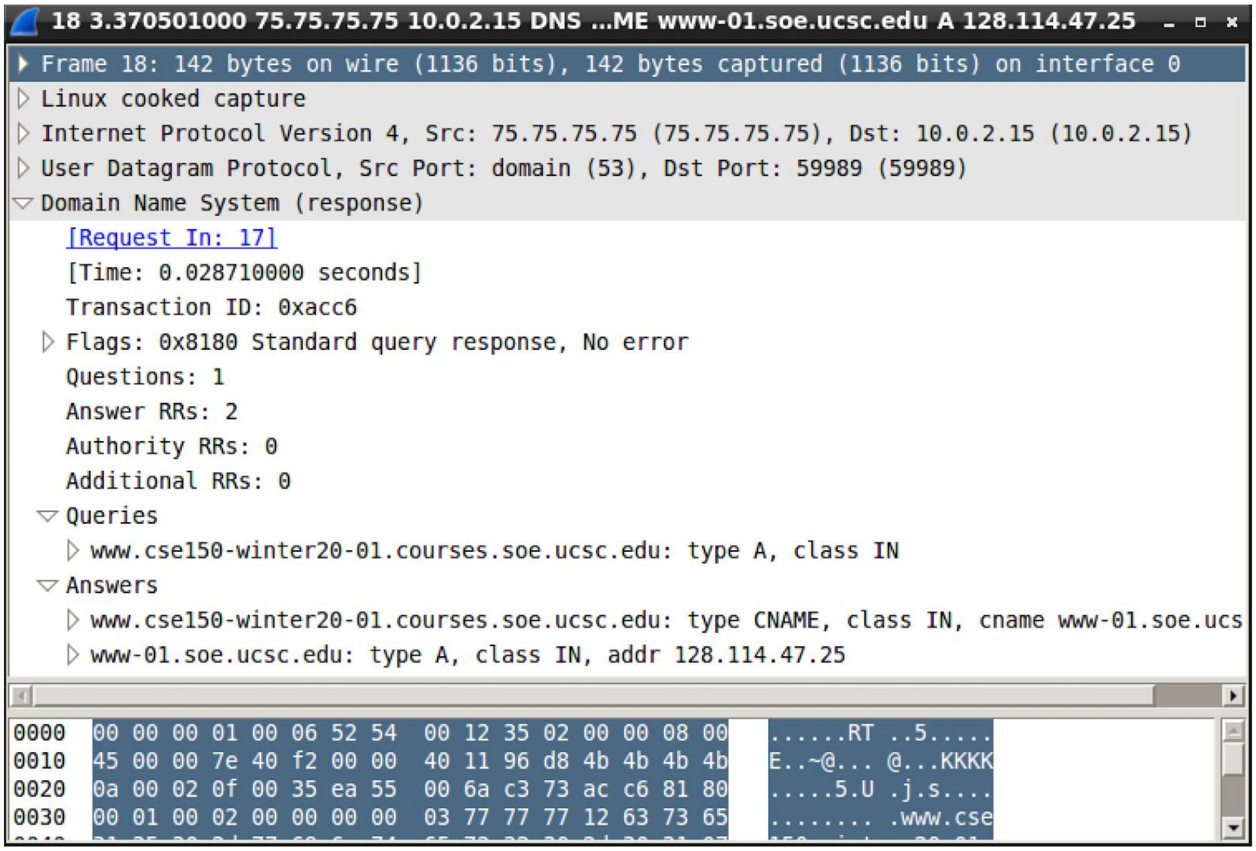


```
▽ Flags: 0x8180 Standard query response, No error
    1... .... .... .... = Response: Message is a response
    .000 0... .... .... = Opcode: Standard query (0)
    .... .0.. .... .... = Authoritative: Server is not an authority for domain
    .... ..0. .... .... = Truncated: Message is not truncated
    .... ...1 .... .... = Recursion desired: Do query recursively
    .... .... 1... .... = Recursion available: Server can do recursive queries
    .... .... .0.. .... = Z: reserved (0)
    .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenti
    .... .... ...0 .... = Non-authenticated data: Unacceptable
    .... .... .... 0000 = Reply code: No error (0)
```

9. The request was resolved and the IP address that I was given for www.cse150-winteer20-01.courses.soe.ucsc.edu is 128.114.47.25.

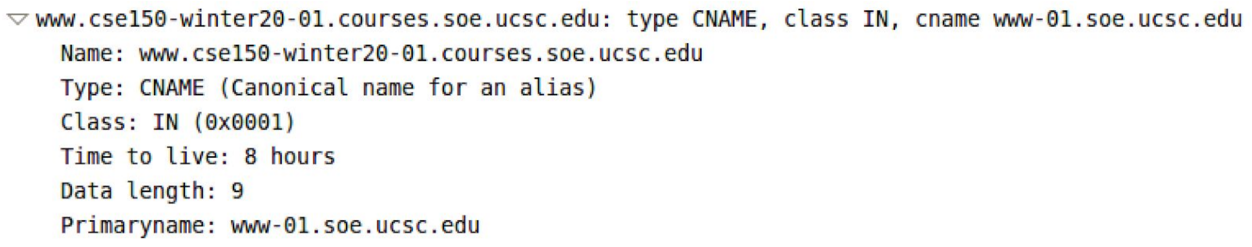


```
17 3.341791000 10.0.2.15 75.75.75.75 DNS ...cse150-winter20-01.courses.soe.ucsc.edu
▸ Frame 17: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 75.75.75.75 (75.75.75.75)
▷ User Datagram Protocol, Src Port: 59989 (59989), Dst Port: domain (53)
▽ Domain Name System (query)
    [Response In: 18]
    Transaction ID: 0xacc6
  ▷ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▽ Queries
    ▷ www.cse150-winter20-01.courses.soe.ucsc.edu: type A, class IN

0000  00 04 00 01 00 06 08 00  27 27 c6 3a 00 00 08 00   ........ '':.....
0010  45 00 00 59 a5 9e 00 00  40 11 32 51 0a 00 02 0f   E..Y.... @.2Q....
0020  4b 4b 4b 4b ea 55 00 35  00 45 a2 fb ac c6 01 00   KKKK.U.5 .E......
0030  00 01 00 00 00 00 00 00  03 77 77 77 12 63 73 65   ........ .www.cse
```



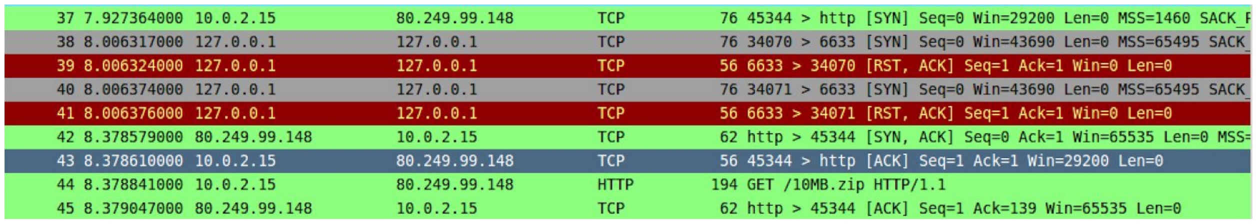
```
18 3.370501000 75.75.75.75 10.0.2.15 DNS ...ME www-01.soe.ucsc.edu A 128.114.47.25
▸ Frame 18: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 75.75.75.75 (75.75.75.75), Dst: 10.0.2.15 (10.0.2.15)
▷ User Datagram Protocol, Src Port: domain (53), Dst Port: 59989 (59989)
▽ Domain Name System (response)
    [Request In: 17]
    [Time: 0.028710000 seconds]
    Transaction ID: 0xacc6
  ▷ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 2
    Authority RRs: 0
    Additional RRs: 0
  ▽ Queries
    ▷ www.cse150-winter20-01.courses.soe.ucsc.edu: type A, class IN
  ▽ Answers
    ▷ www.cse150-winter20-01.courses.soe.ucsc.edu: type CNAME, class IN, cname www-01.soe.ucs
    ▷ www-01.soe.ucsc.edu: type A, class IN, addr 128.114.47.25

0000  00 00 00 01 00 06 52 54  00 12 35 02 00 00 08 00   ......RT ..5.....
0010  45 00 00 7e 40 f2 00 00  40 11 96 d8 4b 4b 4b 4b   E..~@... @...KKKK
0020  0a 00 02 0f 00 35 ea 55  00 6a c3 73 ac c6 81 80   .....5.U .j.s....
0030  00 01 00 02 00 00 00 00  03 77 77 77 12 63 73 65   ........ .www.cse
```

10. The authoritative name server for thee ucsc.edu domain is www-01.soe.ucsc.edu. As shown in the screenshot below, there is a field called Primaryname which implicates the name of the Primary Name Server, which is one of the two types of Authoritative Name servers.

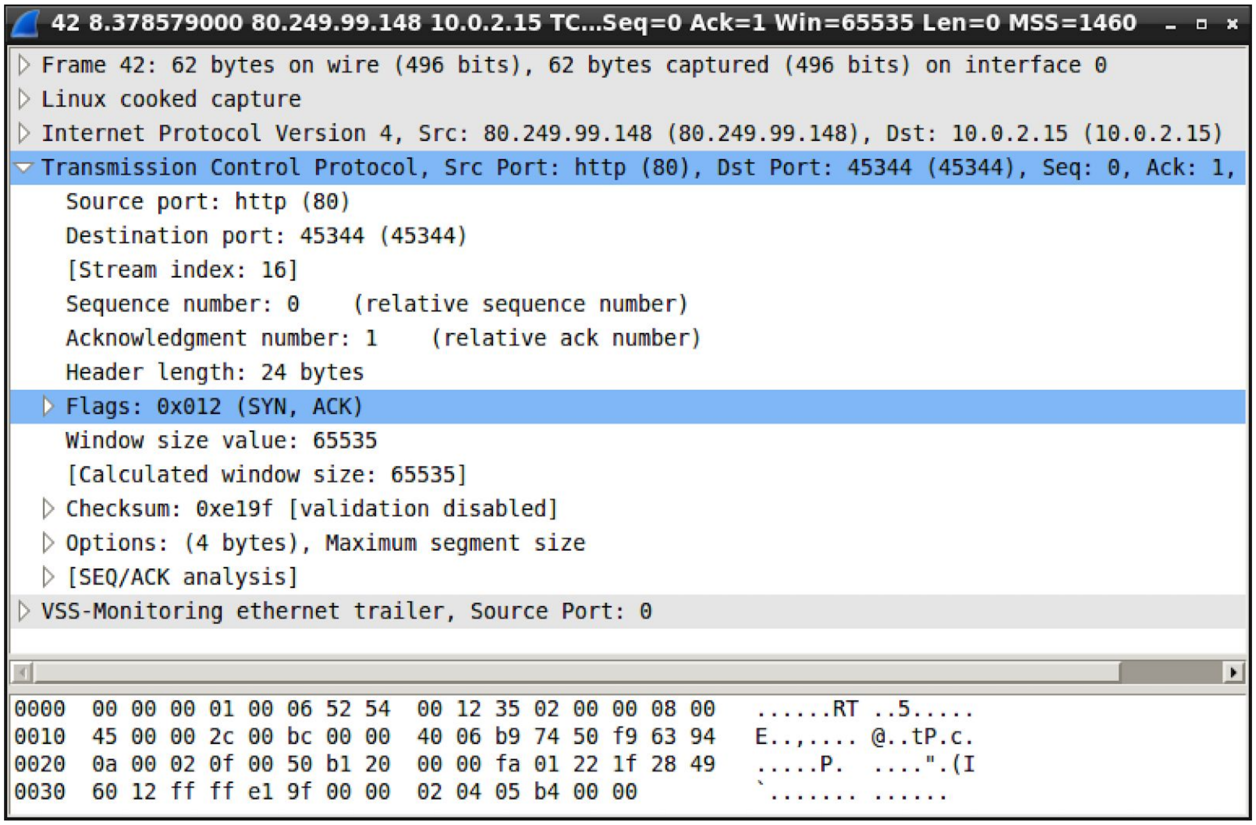


```
▽ www.cse150-winter20-01.courses.soe.ucsc.edu: type CNAME, class IN, cname www-01.soe.ucsc.edu
    Name: www.cse150-winter20-01.courses.soe.ucsc.edu
    Type: CNAME (Canonical name for an alias)
    Class: IN (0x0001)
    Time to live: 8 hours
    Data length: 9
    Primaryname: www-01.soe.ucsc.edu
```

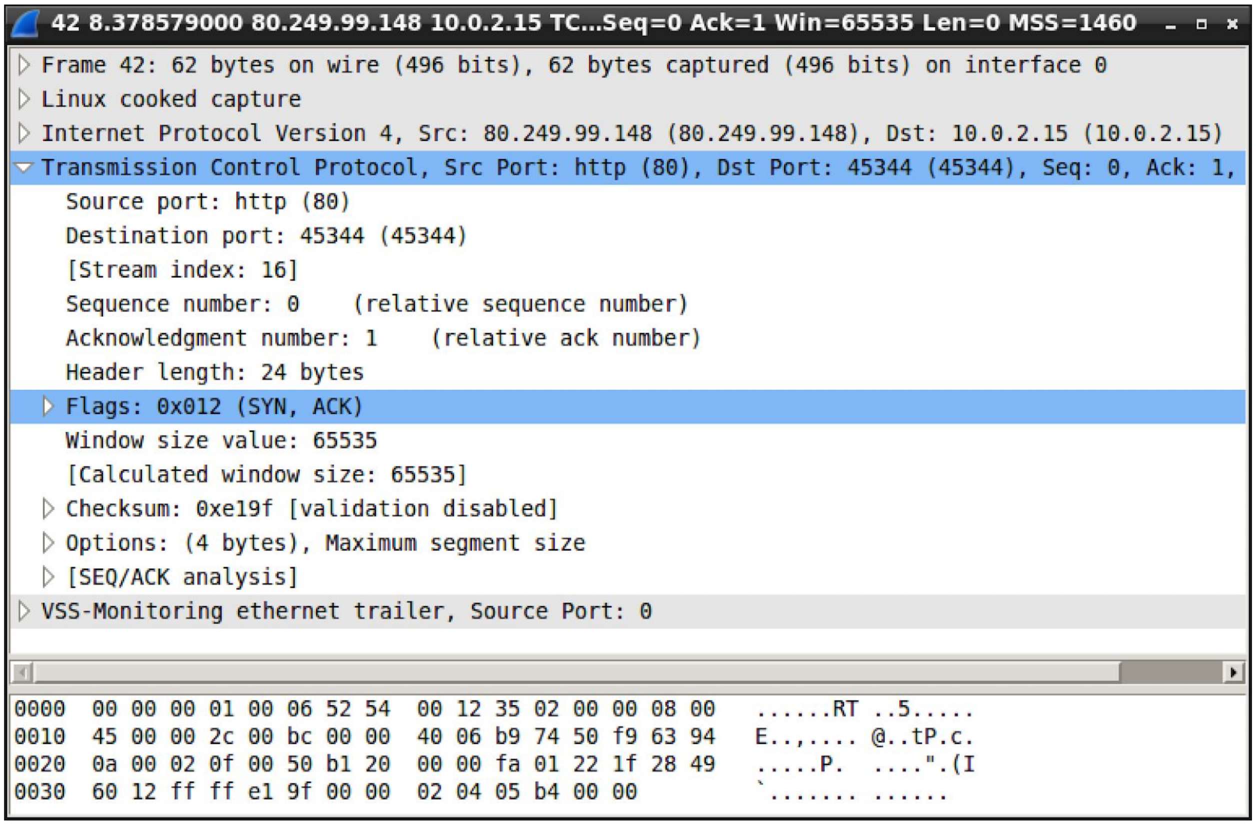
11. The initial window size that my computer advertised to the server is 29200 bytes and the initial window size that the server advertised to my computer is 65535 bytes.



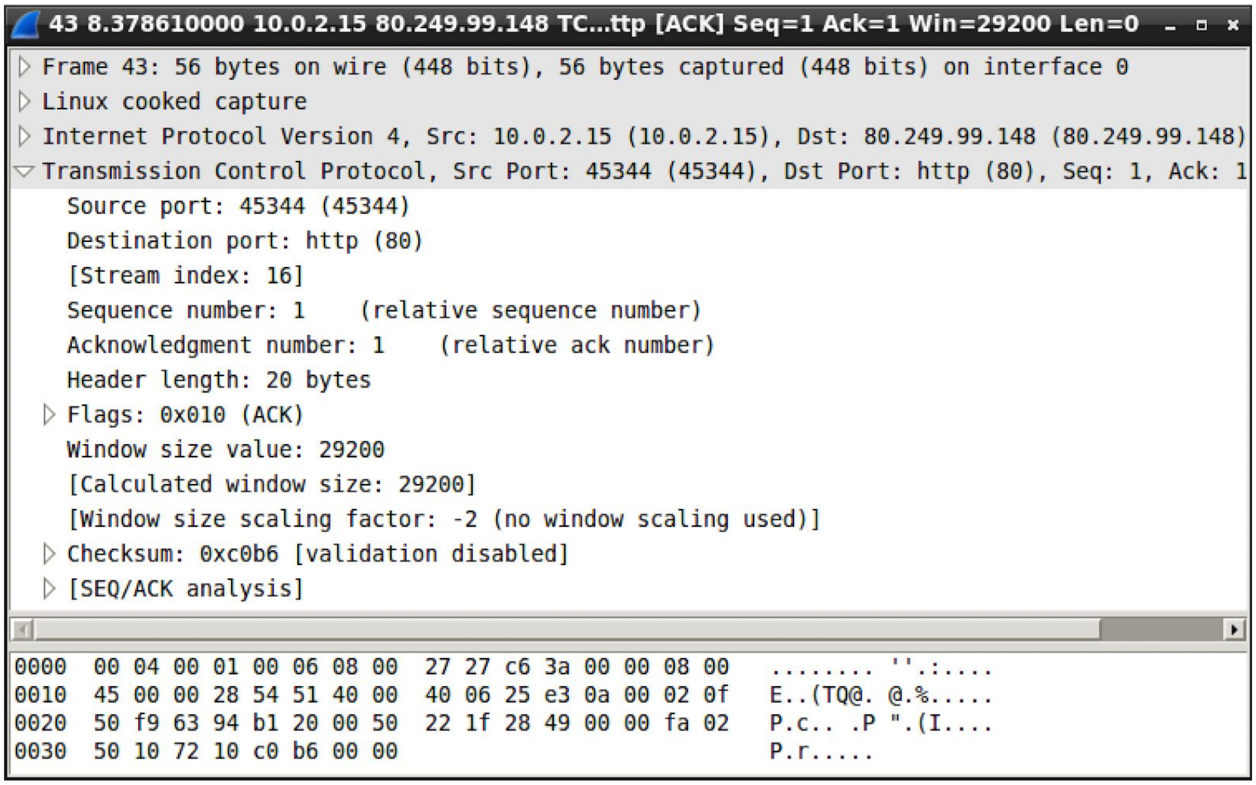
```
37 7.927364000 10.0.2.15      80.249.99.148  TCP   76 45344 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_
38 8.006317000 127.0.0.1      127.0.0.1      TCP   76 34070 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_
39 8.006324000 127.0.0.1      127.0.0.1      TCP   56 6633 > 34070 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40 8.006374000 127.0.0.1      127.0.0.1      TCP   76 34071 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_
41 8.006376000 127.0.0.1      127.0.0.1      TCP   56 6633 > 34071 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
42 8.378579000 80.249.99.148  10.0.2.15      TCP   62 http > 45344 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=
43 8.378610000 10.0.2.15      80.249.99.148  TCP   56 45344 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
44 8.378841000 10.0.2.15      80.249.99.148  HTTP 194 GET /10MB.zip HTTP/1.1
45 8.379047000 80.249.99.148  10.0.2.15      TCP   62 http > 45344 [ACK] Seq=1 Ack=139 Win=65535 Len=0
```



```
42 8.378579000 80.249.99.148 10.0.2.15 TC...Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
▷ Frame 42: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 80.249.99.148 (80.249.99.148), Dst: 10.0.2.15 (10.0.2.15)
▽ Transmission Control Protocol, Src Port: http (80), Dst Port: 45344 (45344), Seq: 0, Ack: 1,
    Source port: http (80)
    Destination port: 45344 (45344)
    [Stream index: 16]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    Header length: 24 bytes
  ▷ Flags: 0x012 (SYN, ACK)
    Window size value: 65535
    [Calculated window size: 65535]
  ▷ Checksum: 0xe19f [validation disabled]
  ▷ Options: (4 bytes), Maximum segment size
  ▷ [SEQ/ACK analysis]
▷ VSS-Monitoring ethernet trailer, Source Port: 0

0000  00 00 00 01 00 06 52 54  00 12 35 02 00 00 08 00   ......RT ..5.....
0010  45 00 00 2c 00 bc 00 00  40 06 b9 74 50 f9 63 94   E..,.... @..tP.c.
0020  0a 00 02 0f 00 50 b1 20  00 00 fa 01 22 1f 28 49   .....P.  ....".(I
0030  60 12 ff ff e1 9f 00 00  02 04 05 b4 00 00         `....... ......
```

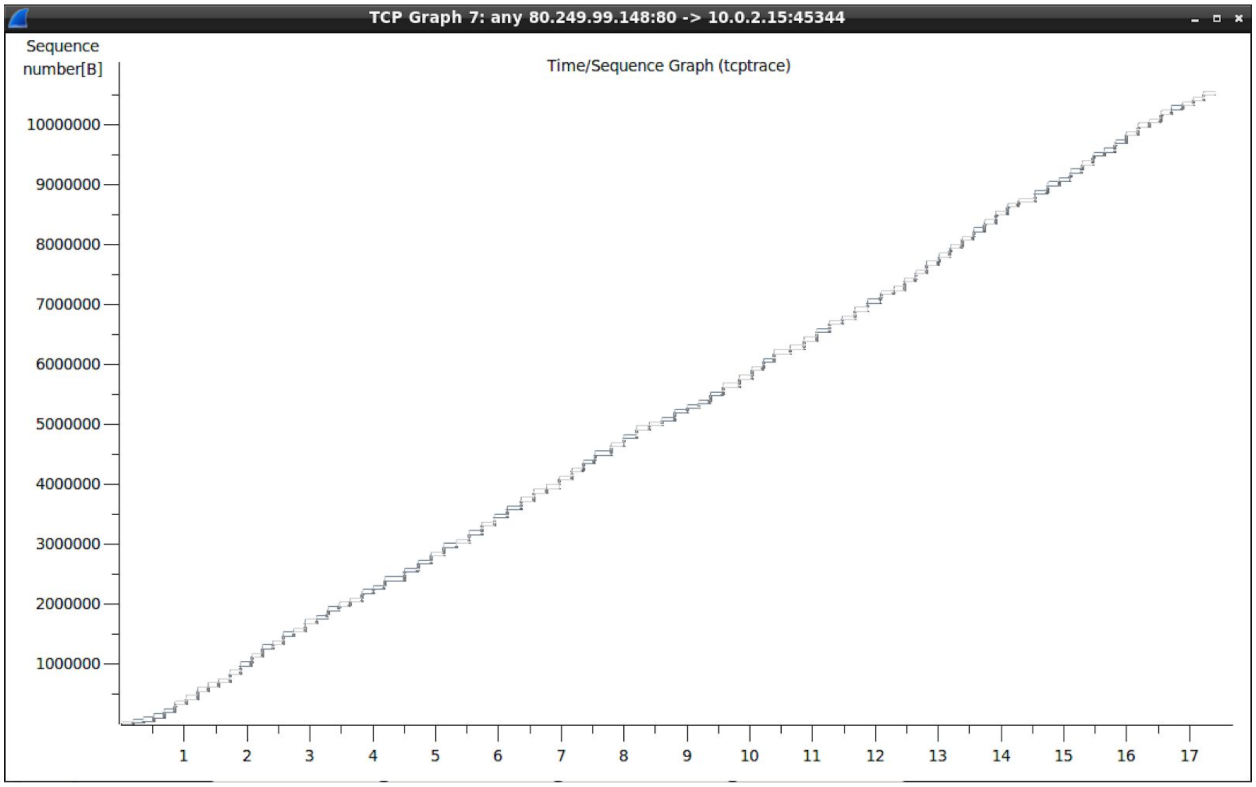


```
42 8.378579000 80.249.99.148 10.0.2.15 TC...Seq=0 Ack=1 Win=65535 Len=0 MSS=1460

▷ Frame 42: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 80.249.99.148 (80.249.99.148), Dst: 10.0.2.15 (10.0.2.15)
▽ Transmission Control Protocol, Src Port: http (80), Dst Port: 45344 (45344), Seq: 0, Ack: 1,
    Source port: http (80)
    Destination port: 45344 (45344)
    [Stream index: 16]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    Header length: 24 bytes
  ▷ Flags: 0x012 (SYN, ACK)
    Window size value: 65535
    [Calculated window size: 65535]
  ▷ Checksum: 0xe19f [validation disabled]
  ▷ Options: (4 bytes), Maximum segment size
  ▷ [SEQ/ACK analysis]
▷ VSS-Monitoring ethernet trailer, Source Port: 0

0000  00 00 00 01 00 06 52 54  00 12 35 02 00 00 08 00   ......RT ..5.....
0010  45 00 00 2c 00 bc 00 00  40 06 b9 74 50 f9 63 94   E..,.... @..tP.c.
0020  0a 00 02 0f 00 50 b1 20  00 00 fa 01 22 1f 28 49   .....P.  ....".(I
0030  60 12 ff ff e1 9f 00 00  02 04 05 b4 00 00         `....... ......
```

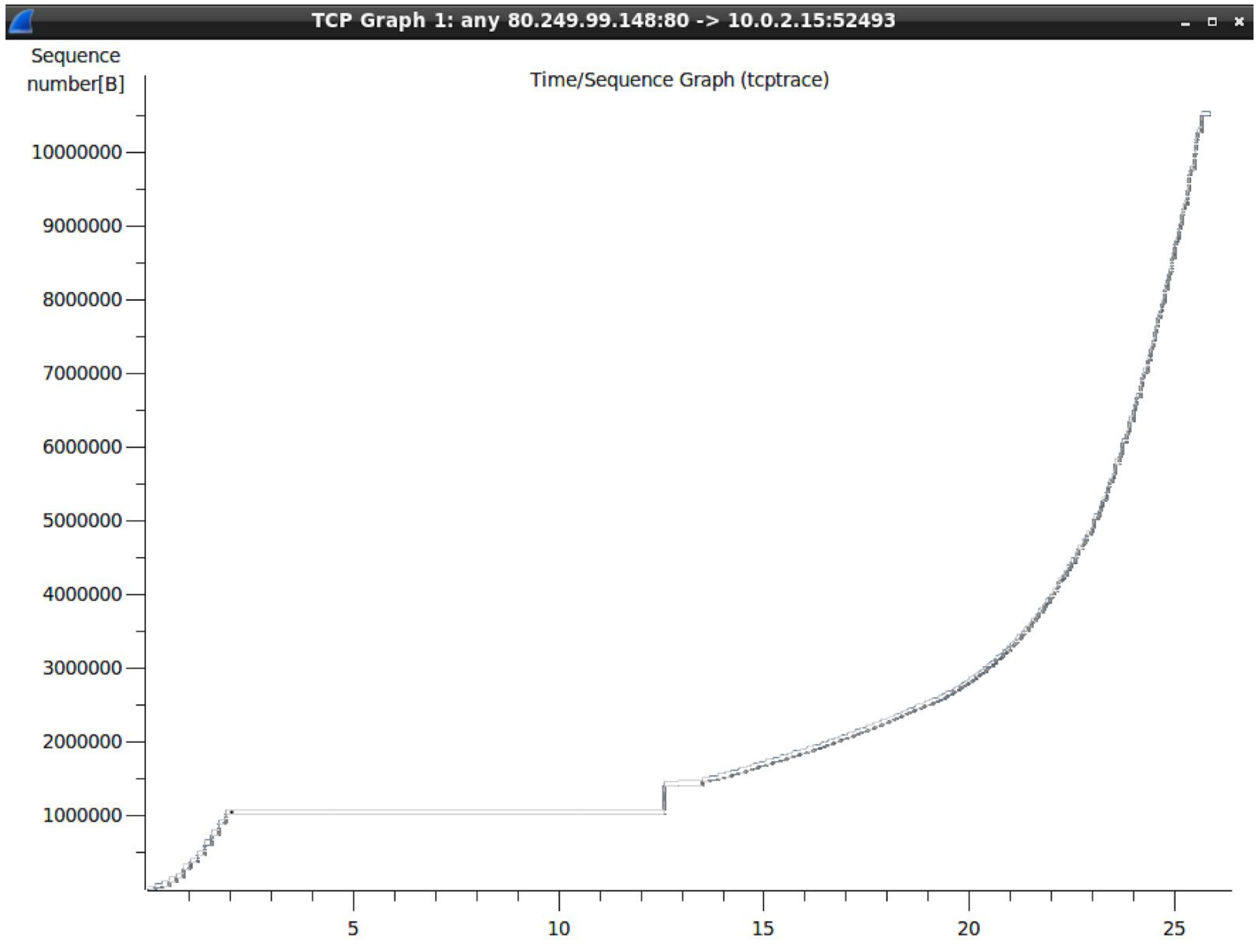


```
43 8.378610000 10.0.2.15 80.249.99.148 TC...ttp [ACK] Seq=1 Ack=1 Win=29200 Len=0

▷ Frame 43: 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: 80.249.99.148 (80.249.99.148)
▽ Transmission Control Protocol, Src Port: 45344 (45344), Dst Port: http (80), Seq: 1, Ack: 1
    Source port: 45344 (45344)
    Destination port: http (80)
    [Stream index: 16]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    Header length: 20 bytes
  ▷ Flags: 0x010 (ACK)
    Window size value: 29200
    [Calculated window size: 29200]
    [Window size scaling factor: -2 (no window scaling used)]
  ▷ Checksum: 0xc0b6 [validation disabled]
  ▷ [SEQ/ACK analysis]

0000  00 04 00 01 00 06 08 00  27 27 c6 3a 00 00 08 00   ........ ''.:....
0010  45 00 00 28 54 51 40 00  40 06 25 e3 0a 00 02 0f   E..(TQ@. @.%.....
0020  50 f9 63 94 b1 20 00 50  22 1f 28 49 00 00 fa 02   P.c.. .P ".(I....
0030  50 10 72 10 c0 b6 00 00                            P.r.....
```

12. The graph displays 2 increasing zigzag lines that intersect with one another at every few intervals. The top zig zag line represents the window while the bottom line represents the ACK. As time goes on, represented by the X-axis, the window increases its sequence number, represented by the Y-axis, every time it successfully sends packets of data to the destination, or in this case my computer. After the window increases its sequence number, it waits for an ACK, which is represented by the bottom line. After the server has received an ACK, which is represented by the two lines intersecting, the window then increases its sequence number and sends more data to my computer, and the cycle repeats until all data had been sent from the server to my computer.



13. The graph has the same purpose as the graph in the previous question where the top line represents the window and the bottom line represents the ACK. The only difference is that there is about a 10 second time period where the graph had plateaued. The only explanation is that after I had run the command *sudo tc qdisc change dev eth0 root netem loss 100%*, I was forcing my computer from accepting the packet that the window was trying to send me from the server, and as a result, no ACK message was sent to the window. As a result, the window was waiting indefinitely until my computer had sent an ACK and that was until I had run the command *sudo tc qdisc change dev eth0 root netem loss 0%* where my computer no longer lost 100% of the packets of data that was being sent by the window. As a result, my computer had successfully accepted the packets of data, and returned an ACK, which prompted the window to increment its sequence number and to continue to send data to my computer.


TCP Graph 1: any 80.249.99.148:80 -> 10.0.2.15:52493
Sequence number[B]
Time/Sequence Graph (tcptrace)