

Part 1: HTTP

1. Find the packet that corresponds to the initial HTTP request that your computer issued. Take a screenshot of this packet. What HTTP method did your computer use to make this request? What URI did your computer request from the server, as present in the HTTP request? (note: NOT the URL). Explain.

No.	Time	Source	Destination	Protocol	Length	Info
229	5.586547000	10.0.2.15	93.184.216.34	HTTP	453	GET / HTTP/1.1
230	5.586981000	93.184.216.34	10.0.2.15	TCP	62	http > 50993 [ACK] Seq=1 Ack=398 Win=65535 Len=0
▶ Frame 229: 453 bytes on wire (3624 bits), 453 bytes captured (3624 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: 50993 (50993), Dst Port: http (80), Seq: 1, Ack: 1, Len: 397						
▼ Hypertext Transfer Protocol						
▼ GET / HTTP/1.1\r\n						
▶ [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]						
Request Method: GET						
Request URI: /						
Request Version: HTTP/1.1						
Host: www.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 Chromium/53.0.2785.143 Chrome/53.0.2785.143 Safari/537.36\r\n						
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://www.example.com/]						
[HTTP request 1/2]						
[Response in frame: 231]						
[Next request in frame: 234]						

```
0000 00 04 00 01 00 06 08 00 27 27 c6 3a 00 00 08 00 ..... ''.....
0010 45 00 01 b5 4c a2 40 00 40 06 aa b7 0a 00 02 0f E...L@.@.....
0020 5d b8 d8 22 c7 31 00 50 bc a6 f2 19 00 b2 b6 02 ]..".l.P .....
0030 50 18 72 10 43 91 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 77 77 TP/1.1.. Host: ww
0050 77 2e 65 78 61 6d 70 6c 65 2e 63 6f 6d 0d 0a 43 w.exempl e.com..C
0060 6f 6e 6e 65 63 74 69 6f 6e 3a 20 6b 65 65 70 2d onnectio n: keep-
0070 61 6c 69 76 65 0d 0a 55 70 67 72 61 64 65 2d 49 alive..U pgrade-I
0080 6e 73 65 63 75 72 65 2d 52 65 71 75 65 73 74 73 nsecure- Requests
0090 3a 20 31 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a : 1..Use r-Agent:
```

The HTTP method that my computer used to make this request is GET, and the URI that the computer requested from the server is “http://www.example.com”

2. Find the packet that corresponds to the initial HTTP response the server issued in response to your request. Take a screenshot of this packet. What HTTP status code did the server return? What is the content type of the response the server is sending back? Explain.

The screenshot shows a Wireshark packet capture of an HTTP response. The packet list at the top shows packet 231 as an HTTP 200 OK response from 93.184.216.34 to 10.0.2.15. The packet details pane shows the following information:

- Hypertext Transfer Protocol**
 - HTTP/1.1 200 OK\r\n
 - [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
 - Request Version: HTTP/1.1
 - Status Code: 200
 - Response Phrase: OK
 - Content-Encoding: gzip\r\n
 - Cache-Control: max-age=604800\r\n
 - Content-Type: text/html; charset=UTF-8\r\n
 - Date: Tue, 29 Jan 2019 03:31:10 GMT\r\n
 - Etag: "1541025663+gzip"\r\n
 - Expires: Tue, 05 Feb 2019 03:31:10 GMT\r\n
 - Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT\r\n
 - Server: ECS (oxr/830F)\r\n
 - Vary: Accept-Encoding\r\n
 - X-Cache: HIT\r\n
 - Content-Length: 606\r\n
 - \r\n
 - [HTTP response 1/2]
 - [Time since request: 0.010573000 seconds]
 - [\[Request in frame: 229\]](#)
 - [\[Next request in frame: 234\]](#)
 - [\[Next response in frame: 236\]](#)
 - Content-encoded entity body (gzip): 606 bytes -> 1270 bytes

The packet bytes pane shows the raw data of the response, including the Content-Type header and the compressed HTML body.

The Status Code given for the response is 200, the Content Type is text/html

3. Find the packets that correspond to the initial HTTP request and response that your computer issued/received. Take a screenshot of these packets. What's different? Explain.

The screenshot shows a Wireshark packet capture of an HTTP request. The packet list at the top shows packet 34 as an HTTP GET request from 10.0.2.15 to 128.114.47.25. The packet details pane shows the following information:

- Hypertext Transfer Protocol**
 - GET / HTTP/1.1\r\n
 - [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
 - Request Method: GET
 - Request URI: /
 - Request Version: HTTP/1.1
 - User-Agent: Wget/1.15 (linux-gnu)\r\n
 - Accept: */*\r\n
 - Host: www.soe.ucsc.edu\r\n
 - Connection: Keep-Alive\r\n
 - \r\n
 - [\[Full request URI: http://www.soe.ucsc.edu/\]](#)
 - [HTTP request 1/1]
 - [\[Response in frame: 36\]](#)

The packet bytes pane shows the raw data of the request, including the GET method and the URI.

34	11.47170000	10.0.2.15	128.114.47.25	HTTP	170 GET / HTTP/1.1
35	11.47271700	128.114.47.25	10.0.2.15	TCP	62 http > 57063 [ACK] Seq=1 Ack=115 Win=65535 Len=0
36	11.47418200	128.114.47.25	10.0.2.15	HTTP	748 HTTP/1.1 301 Moved Permanently (text/html)
▶ Frame 36: 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 (10.0.2.15) ▶ Transmission Control Protocol, Src Port: http (80), Dst Port: 57063 (57063), Seq: 1, Ack: 115, Len: 692 ▼ Hypertext Transfer Protocol ▼ HTTP/1.1 301 Moved Permanently\r\n ▶ [Expert Info (Chat/Sequence): HTTP/1.1 301 Moved Permanently\r\n]					
Request Version: HTTP/1.1 Status Code: 301 Response Phrase: Moved Permanently Date: Tue, 29 Jan 2019 04:02:16 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: Tue, 29 Jan 2019 04:07:16 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 \r\n [HTTP response 1/1] [Time since request: 0.002482000 seconds] [Request in frame: 34]					
▼ Line-based text data: text/html <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">\n					

The request method was the same, which was “GET”, however the Response code was different, the Status Code is “301”

4. Take a screenshot of your packet and explain what you did to create it.

8	0.016836000	10.0.2.15	93.184.216.34	HTTP	170 HEAD / HTTP/1.1
▶ Frame 8: 170 bytes on wire (1360 bits), 170 bytes captured (1360 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: 51386 (51386), Dst Port: http (80), Seq: 1, Ack: 1, Len: 114 ▼ Hypertext Transfer Protocol ▼ HEAD / HTTP/1.1\r\n ▶ [Expert Info (Chat/Sequence): HEAD / HTTP/1.1\r\n]					
Request Method: HEAD Request URI: / Request Version: HTTP/1.1 User-Agent: Wget/1.15 (linux-gnu)\r\n Accept: */*\r\n Host: www.example.com\r\n Connection: Keep-Alive\r\n \r\n [Full request URI: http://www.example.com/] [HTTP request 1/1]					

```
mininet@mininet-vm:~$ wget -S --spider www.example.com
Spider mode enabled. Check if remote file exists.
--2019-01-28 20:19:59-- http://www.example.com/
Resolving www.example.com (www.example.com)... 93.184.216.34, 2606:2800:220:1:248:1893:25c8:1946
Connecting to www.example.com (www.example.com)|93.184.216.34|:80... connected.
HTTP request sent, awaiting response...
  HTTP/1.1 200 OK
  Content-Encoding: gzip
  Accept-Ranges: bytes
  Cache-Control: max-age=604800
  Content-Type: text/html; charset=UTF-8
  Date: Tue, 29 Jan 2019 04:19:59 GMT
  Etag: "1541025663+gzip"
  Expires: Tue, 05 Feb 2019 04:19:59 GMT
  Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
  Server: ECS (oxr/830C)
  X-Cache: HIT
  Content-Length: 606
Length: 606 [text/html]
Remote file exists and could contain further links,
but recursion is disabled -- not retrieving.

mininet@mininet-vm:~$
```

To create this packet, I went into Mininet and used the command “`wget -S --spider www.exmple.com`” which then gave me a Request Method of “Head” for the packet.

Part 2: DNS

5. Were any steps taken by your computer before the web page was loaded? If so, using your captured packets in Wireshark, find the packets that allowed your computer to successfully load `http://www.example.com`. Take a screenshot of these packets, and explain why you think these are the correct packets. If not, explain why your computer did not need to take these steps.

1	0.000000000	10.0.2.15	75.75.75.75	DNS	77	Standard query 0x118f A www.example.com
2	0.016875000	75.75.75.75	10.0.2.15	DNS	93	Standard query response 0x118f A 93.184.216.34

▶ Frame 1: 77 bytes on wire (616 bits), 77 bytes captured (616 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: 43694 (43694), Dst Port: domain (53)
- Domain Name System (query)
 - [\[Response In: 2\]](#)
 - Transaction ID: 0x118f
 - 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
 - Questions: 1
 - Answer RRs: 0
 - Authority RRs: 0
 - Additional RRs: 0
 - Queries
 - www.example.com: type A, class IN
 - Name: www.example.com
 - Type: A (Host address)
 - Class: IN (0x0001)

▶ Frame 2: 93 bytes on wire (744 bits), 93 bytes captured (744 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: 43694 (43694)
- Domain Name System (response)
 - [\[Request In: 1\]](#)
 - [Time: 0.016875000 seconds]
 - Transaction ID: 0x118f
 - Flags: 0x0180 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 authenticated by the server
 -0 = Non-authenticated data: Unacceptable
 - 0000 = Reply code: No error (0)
 - Questions: 1
 - Answer RRs: 1
 - Authority RRs: 0
 - Additional RRs: 0
 - Queries
 - www.example.com: type A, class IN
 - Answers
 - www.example.com: type A, class IN, addr 93.184.216.34
 - Name: www.example.com
 - Type: A (Host address)
 - Class: IN (0x0001)

6. Were any steps taken by your computer before the web page was loaded? If so, using your captured packets in Wireshark, find the packets that allowed your computer to successfully load <http://216.58.193.68>. Take a screenshot of these packets, and explain why you think these are the correct packets. If not, explain why your computer did not need to take these steps.

66	3.079019000	10.0.2.15	128.114.142.6	DNS	79	Standard query 0xe9f6 A fonts.gstatic.com
67	3.080769000	128.114.142.6	10.0.2.15	DNS	134	Standard query response 0xe9f6 CNAME.gstaticadssl.l.google.com A 172.217.5.99

7. Take a screenshot of the packets corresponding to your request, and the response from the server. If the request was resolved, what is the IP address you were given for www.google.com?

23	9.983497000	10.0.2.15	128.114.142.6	DNS	76	Standard query 0x530e A www.google.com
24	9.986096000	128.114.142.6	10.0.2.15	DNS	92	Standard query response 0x530e A 172.217.5.100

The IP address given for is 172.217.5.100

8. Did your computer want to complete the request recursively? How do you know? Take a screenshot proving your answer.

```

> Frame 23: 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: 128.114.142.6 (128.114.142.6)
> User Datagram Protocol, Src Port: 51297 (51297), Dst Port: domain (53)
< Domain Name System (query)
  [Response In: 24]
  Transaction ID: 0x530e
  < 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
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  < Queries
    > www.google.com: type A, class IN

```

In the packet file, the recursion flag is set on.

9. Take a screenshot of the packets corresponding to your request, and the response from the server. If the request was resolved, what is the IP address you were given for cmpe150.ucsc.edu?

7	2.427641000	10.0.2.15	128.114.142.6	DNS	89	Standard query 0x9aff	A cmpe150.ucsc.edu
8	2.430205000	128.114.142.6	10.0.2.15	DNS	142	Standard query response 0x9aff	No such name

10. What is the authoritative name server for the ucsc.edu domain? How do you know? Take a screenshot proving your answer.

```
▼ Authoritative nameservers
  ▼ ucsc.edu: type SOA, class IN, mname adns1.ucsc.edu
    Name: ucsc.edu
    Type: SOA (Start of zone of authority)
    Class: IN (0x0001)
    Time to live: 15 minutes
    Data length: 41
    Primary name server: adns1.ucsc.edu
    Responsible authority's mailbox: hostmaster.ucsc.edu
    Serial Number: 13537509
    Refresh Interval: 10800 (3 hours)
    Retry Interval: 900 (15 minutes)
    Expire limit: 2419200 (28 days)
    Minimum TTL: 900 (15 minutes)
```

Part 3: TCP

11. Find the packets corresponding to the SYN, SYN-ACK, and ACK that initiated the TCP connection for this file transfer. Take a screenshot of these packets. What was the initial window size that your computer advertised to the server? What was the initial window size that the server advertised to you?

The image displays two packets from a Wireshark capture. The first packet, Frame 15, is a SYN packet from 10.0.2.15 to 80.249.99.148. It has a source port of 52793 and a destination port of 80. The window size is 29200. The second packet, Frame 16, is a SYN-ACK packet from 80.249.99.148 to 10.0.2.15. It has a source port of 80 and a destination port of 52793. The window size is 65535. The acknowledgment number is 1, indicating it is acknowledging the SYN packet in frame 15. The round-trip time (RTT) to ACK the segment was 0.152538000 seconds.

```
▶ Frame 15: 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: 80.249.99.148 (80.249.99.148)
  ▼ Transmission Control Protocol, Src Port: 52793 (52793), Dst Port: http (80), Seq: 0, Len: 0
    Source port: 52793 (52793)
    Destination port: http (80)
    [Stream index: 5]
    Sequence number: 0 (relative sequence number)
    Header length: 40 bytes
  ▶ Flags: 0x002 (SYN)
    Window size value: 29200
    [Calculated window size: 29200]
  ▶ Checksum: 0xc0ca [validation disabled]
  ▼ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
    ▼ Maximum segment size: 1460 bytes
      Kind: MSS size (2)
      Length: 4
      MSS Value: 1460
    ▶ TCP SACK Permitted Option: True
    ▶ Timestamps: TSval 26050, TSecr 0
    ▶ No-Operation (NOP)
    ▼ Window scale: 7 (multiply by 128)
      Kind: Window Scale (3)
      Length: 3
      Shift count: 7
      [Multiplier: 128]

▶ Frame 16: 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: 52793 (52793), Seq: 0, Ack: 1, Len: 0
    Source port: http (80)
    Destination port: 52793 (52793)
    [Stream index: 5]
    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: 0xfcfl [validation disabled]
  ▼ Options: (4 bytes), Maximum segment size
    ▼ Maximum segment size: 1460 bytes
      Kind: MSS size (2)
      Length: 4
      MSS Value: 1460
    ▼ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 15]
      [The RTT to ACK the segment was: 0.152538000 seconds]
  ▼ VSS-Monitoring ethernet trailer, Source Port: 0
    Src Port: 0
```



```

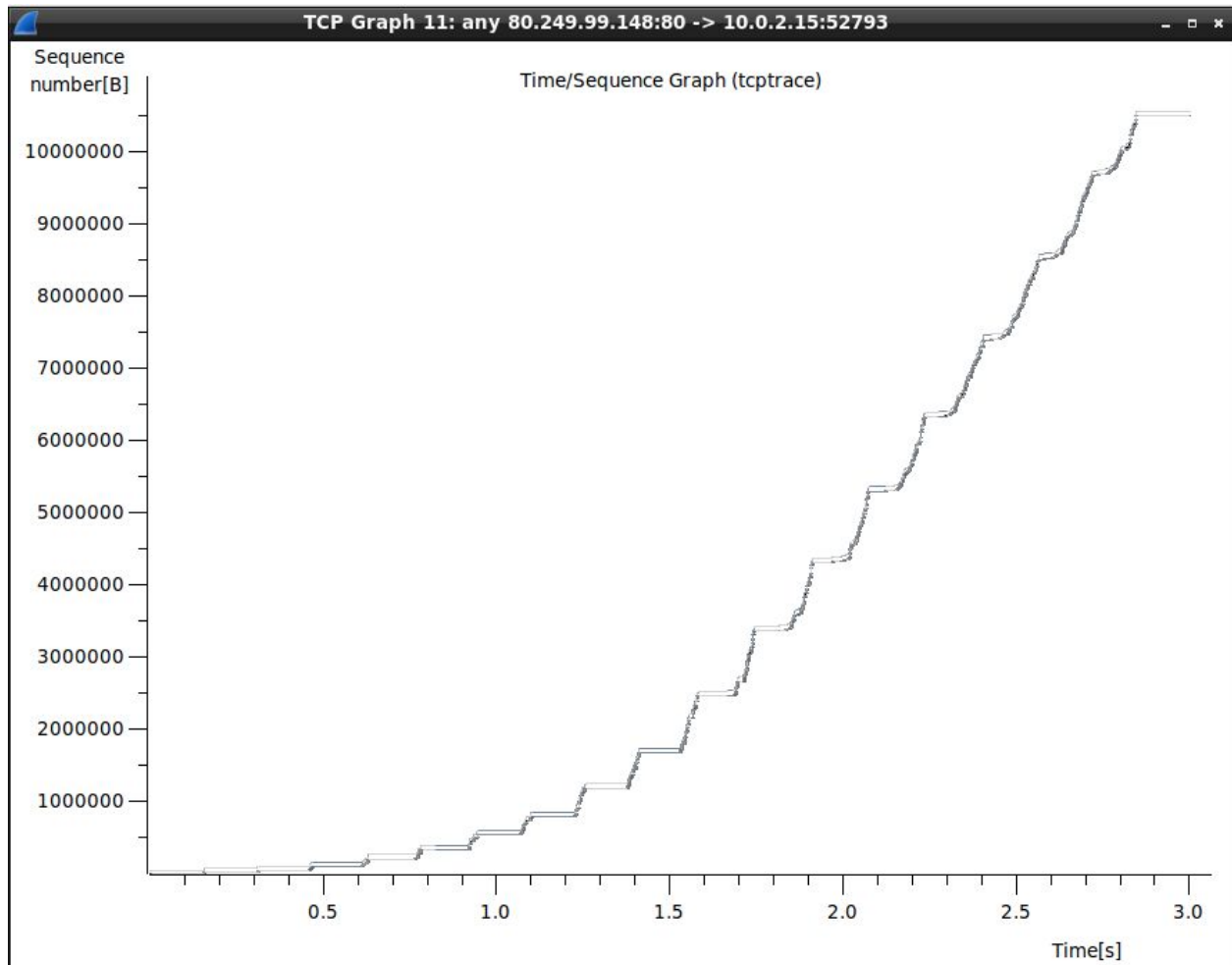
▶ Frame 17: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 0
▶ Linux cooked capture
▶ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 80.249.99.148 (80.249.99.148)
▼ Transmission Control Protocol, Src Port: 52793 (52793), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
  Source port: 52793 (52793)
  Destination port: http (80)
  [Stream index: 5]
  Sequence number: 1 (relative sequence number)
  Acknowledgment number: 1 (relative ack number)
  Header length: 20 bytes
  ▼ Flags: 0x010 (ACK)
    000. .... = Reserved: Not set
    ...0 .... = Nonce: Not set
    .... 0... = Congestion Window Reduced (CWR): Not set
    .... .0.. = ECN-Echo: Not set
    .... ..0. = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
  Window size value: 29200
  [Calculated window size: 29200]
  [Window size scaling factor: -2 (no window scaling used)]
  ▶ Checksum: 0xc0b6 [validation disabled]
  ▼ [SEQ/ACK analysis]
    [This is an ACK to the segment in frame: 16]
    [The RTT to ACK the segment was: 0.000042000 seconds]

```

13	4.196545000	128.114.142.6	10.0.2.15	DNS	157	Standard query response 0x317c	CNAME ipv4.download1.thinkbroadband.com A 80.249.99.148
14	4.196575000	128.114.142.6	10.0.2.15	DNS	218	Standard query response 0xaf09	CNAME ipv4.download1.thinkbroadband.com
15	4.197015000	10.0.2.15	80.249.99.148	TCP	76	52793 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=26050 TSecr=0 WS=128	
16	4.349553000	80.249.99.148	10.0.2.15	TCP	62	http > 52793 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460	
17	4.349595000	10.0.2.15	80.249.99.148	TCP	56	52793 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0	
18	4.352841000	10.0.2.15	80.249.99.148	HTTP	194	GET /10MB.zip HTTP/1.1	
19	4.353952000	80.249.99.148	10.0.2.15	TCP	62	http > 52793 [ACK] Seq=1 Ack=139 Win=65535 Len=0	
20	4.505244000	80.249.99.148	10.0.2.15	TCP	2694	[TCP segment of a reassembled PDU]	
21	4.505278000	10.0.2.15	80.249.99.148	TCP	56	52793 > http [ACK] Seq=139 Ack=2639 Win=34080 Len=0	
22	4.505812000	80.249.99.148	10.0.2.15	TCP	4316	[TCP segment of a reassembled PDU]	
23	4.505837000	10.0.2.15	80.249.99.148	TCP	56	52793 > http [ACK] Seq=139 Ack=6899 Win=42600 Len=0	
24	4.506278000	80.249.99.148	10.0.2.15	TCP	5736	[TCP segment of a reassembled PDU]	
25	4.506294000	10.0.2.15	80.249.99.148	TCP	56	52793 > http [ACK] Seq=139 Ack=12579 Win=53960 Len=0	

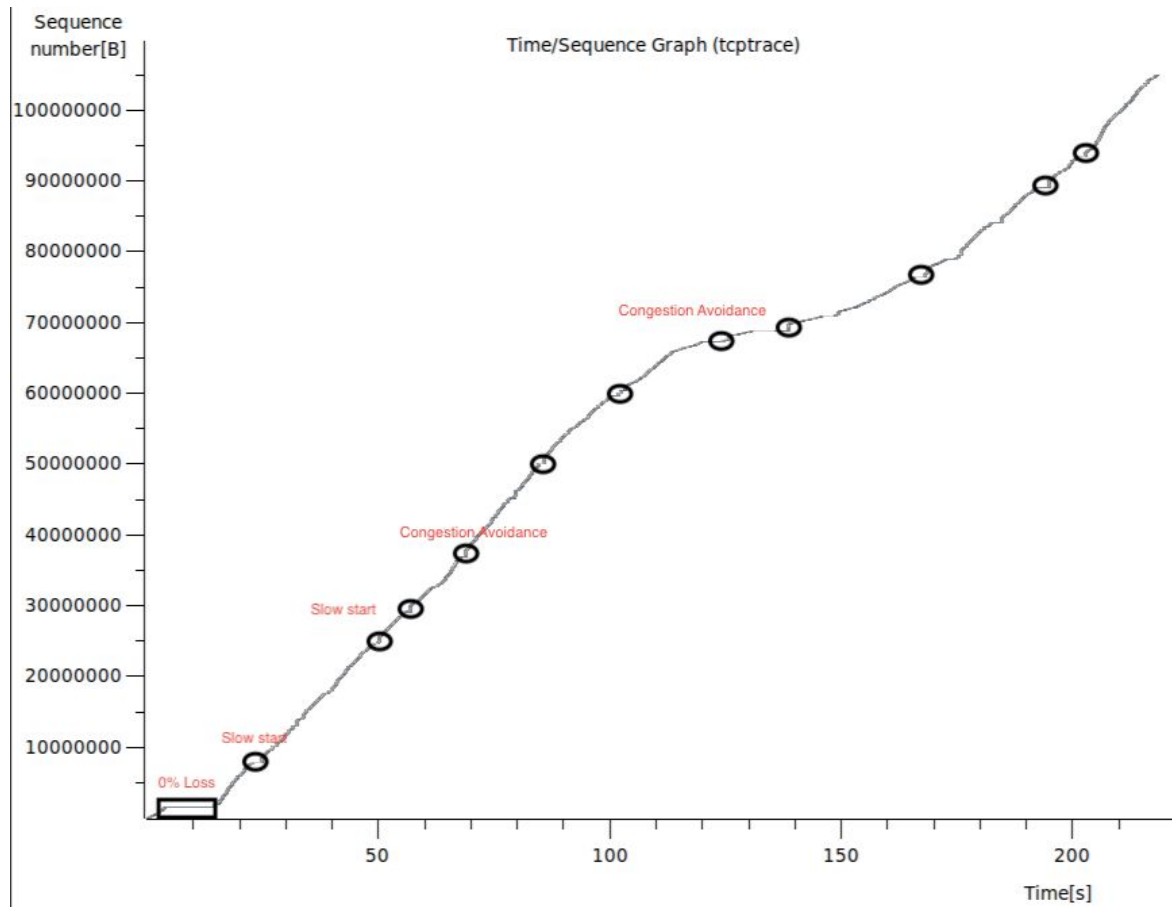
The initial window size the computer advertised to the server is 29200, the initial window size the server advertise to me is 65535.

12. Find a packet from the download whose source address is the server's address and the destination address is your computer's address. Using Wireshark, create a tcptrace graph with this packet selected. Take a screenshot of the graph and explain what it is showing. **Look into the Wireshark documentation if you need assistance making this graph.**



The graph is showing the receive window, the amount of bytes being sent, the TCP segments, and ACKs with the amount of bytes being sent over time.

13. Find a packet from the download whose source address is the address of the server and destination address is your computer's address. Create a tcptrace graph with this packet selected. Take a screenshot of the graph and explain what it is showing. Using an image editing program, circle the areas where the 0% loss is shown, as well as where TCP is in slow-start and congestion-avoidance.



The graph is showing the receive window, the amount of bytes being sent, the TCP segments, and ACKs with the amount of bytes being sent over time.