Wireshark Lab: TCP

Submit online as a pdf in Canvas

General Instructions: What to hand in: Please answer the questions posed in this lab, please make it clear what questions you are answering, and please use screenshots to support your answers. Annotate the printout to explain your answer. To print a packet, use *File->Print*, choose *selected packet only*, choose *Packet summary line,* and select the minimum amount of packet detail that you need to answer the question. Marks will be awarded for correctness, completeness, and professionalism. You'll also be using this document to study for your tests and exams.

Learning Outcome: At the end of this lab you should:

- be familiar with the different TCP message types.
- understand the role of the TCP in any network.
- write a socket program over TCP.
- finally, be able to explain the operation of TCP.

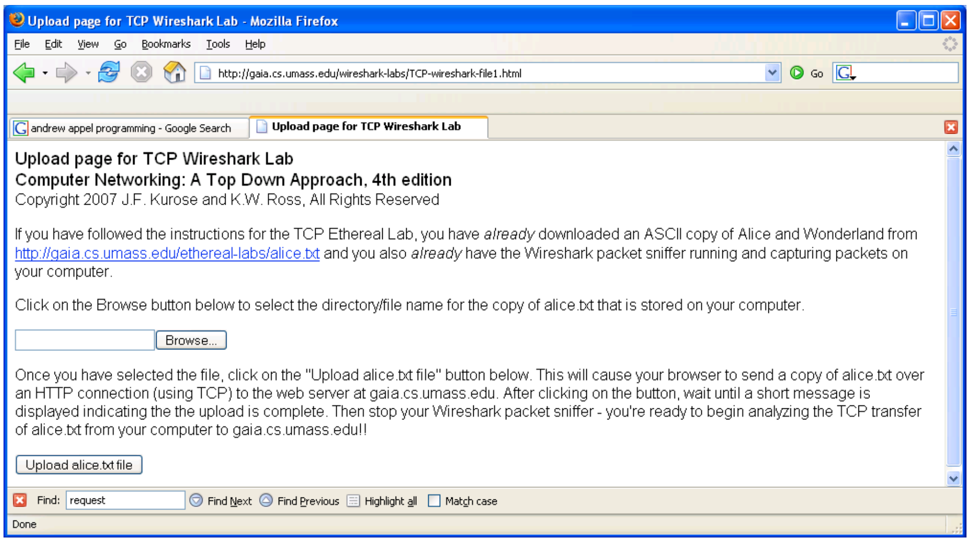
Exercise 1

Objectives for Exercise 1: To see how file can be uploaded to a remote server from your computer using the TCP protocol. To investigate the operation of TCP.

Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text of *Alice in Wonderland*), and then transfer the file to a Web server using the HTTP POST method (see section 2.2.3 in the text). We're using the POST method rather than the GET method as we'd like to transfer a large amount of data *from* your computer to another computer. Of course, we'll be running Wireshark during this time to obtain the trace of the TCP segments sent and received from your computer. (Make sure TCP relative sequence numbers box is checked in preference)

Do the following:

- Start up your web browser. Go the http://gaia.cs.umass.edu/wireshark-labs/alice.txt and retrieve an ASCII copy of *Alice in Wonderland.* Store this file somewhere on your computer.
- Next, go to http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html
- You should see a screen that looks like:


Upload page for TCP Wireshark Lab - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html
Go
andrew appel programming - Google Search
Upload page for TCP Wireshark Lab

Upload page for TCP Wireshark Lab
Computer Networking: A Top Down Approach, 4th edition
Copyright 2007 J.F. Kurose and K.W. Ross, All Rights Reserved

If you have followed the instructions for the TCP Ethereal Lab, you have *already* downloaded an ASCII copy of Alice and Wonderland from http://gaia.cs.umass.edu/ethereal-labs/alice.txt and you also *already* have the Wireshark packet sniffer running and capturing packets on your computer.

Click on the Browse button below to select the directory/file name for the copy of alice.txt that is stored on your computer.

Browse...

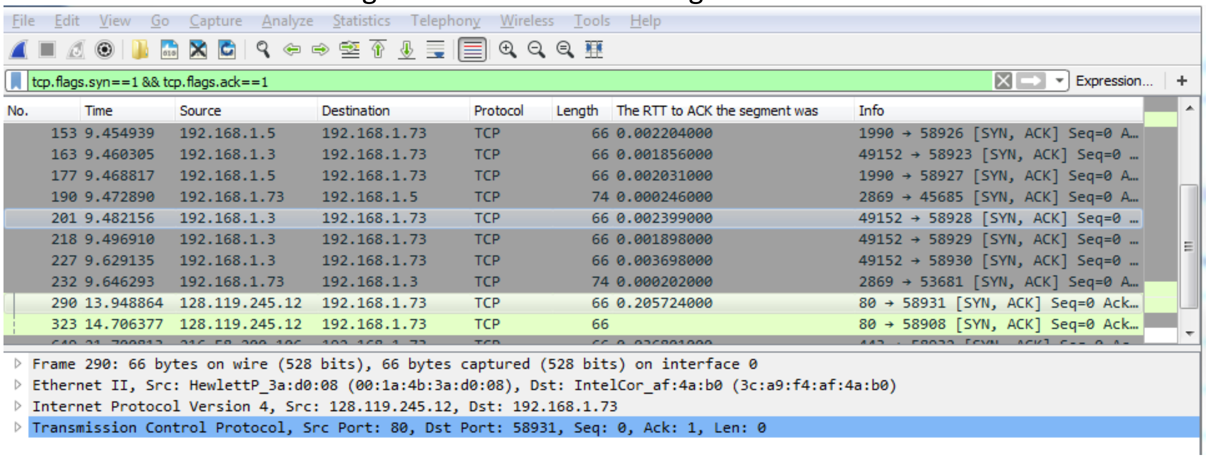
Once you have selected the file, click on the "Upload alice.txt file" button below. This will cause your browser to send a copy of alice.txt over an HTTP connection (using TCP) to the web server at gaia.cs.umass.edu. After clicking on the button, wait until a short message is displayed indicating the the upload is complete. Then stop your Wireshark packet sniffer - you're ready to begin analyzing the TCP transfer of alice.txt from your computer to gaia.cs.umass.edu!!

Upload alice.txt file

- Use the *Browse* button in this form to enter the name of the file (full path name) on your computer containing *Alice in Wonderland* (or do so manually). Don't yet press the "*Upload alice.txt file*" button.
- Now start up Wireshark and begin packet capture *(Capture->Start)* and then press *OK* on the Wireshark Packet Capture Options screen (we'll not need to select any options here).
- Returning to your browser, press the "*Upload alice.txt file*" button to upload the file to the gaia.cs.umass.edu server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.
- Stop Wireshark packet capture.

1.1 Filter the capture to only `TCP SYN` packets and `TCP ACK` packets you should see this. (Hint: `tcp.flags.syn==1 && tcp.flags.ack==1`). This finds the middle step in the three-way handshake. Find the middle step in the three-way handshake between your machine, and the machine that hosts http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html

It will look something like the screenshot in figure 1.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

tcp.flags.syn==1 && tcp.flags.ack==1

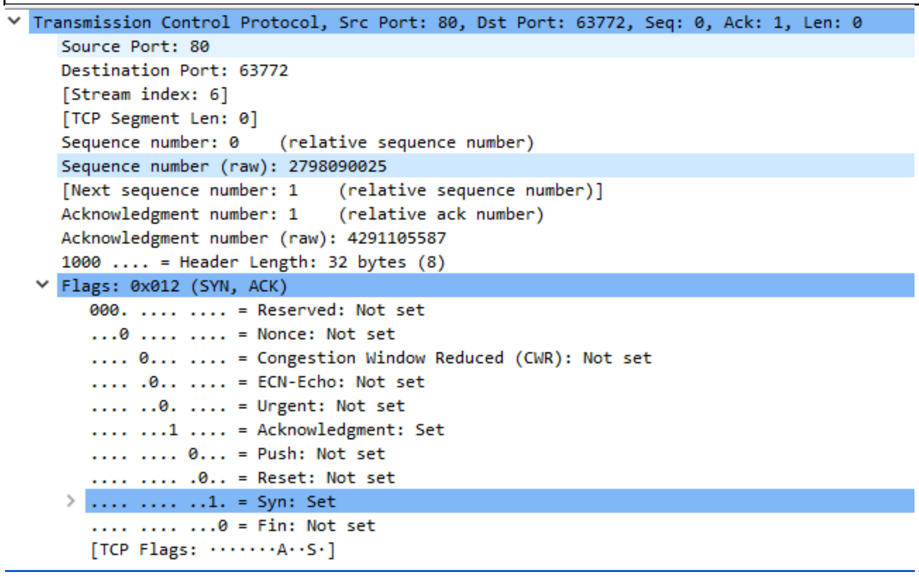
No.	Time	Source	Destination	Protocol	Length	The RTT to ACK the segment was	Info
153	9.454939	192.168.1.5	192.168.1.73	TCP	66	0.002204000	1990 → 58926 [SYN, ACK] Seq=0 A…
163	9.460305	192.168.1.3	192.168.1.73	TCP	66	0.001856000	49152 → 58923 [SYN, ACK] Seq=0 …
177	9.468817	192.168.1.5	192.168.1.73	TCP	66	0.002031000	1990 → 58927 [SYN, ACK] Seq=0 A…
190	9.472890	192.168.1.73	192.168.1.5	TCP	74	0.000246000	2869 → 45685 [SYN, ACK] Seq=0 A…
201	9.482156	192.168.1.3	192.168.1.73	TCP	66	0.002399000	49152 → 58928 [SYN, ACK] Seq=0 …
218	9.496910	192.168.1.3	192.168.1.73	TCP	66	0.001898000	49152 → 58929 [SYN, ACK] Seq=0 …
227	9.629135	192.168.1.3	192.168.1.73	TCP	66	0.003698000	49152 → 58930 [SYN, ACK] Seq=0 …
232	9.646293	192.168.1.73	192.168.1.3	TCP	74	0.000202000	2869 → 53681 [SYN, ACK] Seq=0 A…
290	13.948864	128.119.245.12	192.168.1.73	TCP	66	0.205724000	80 → 58931 [SYN, ACK] Seq=0 Ack…
323	14.706377	128.119.245.12	192.168.1.73	TCP	66		80 → 58908 [SYN, ACK] Seq=0 Ack…

```
▷ Frame 290: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▷ Ethernet II, Src: HewlettP_3a:d0:08 (00:1a:4b:3a:d0:08), Dst: IntelCor_af:4a:b0 (3c:a9:f4:af:4a:b0)
▷ Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.1.73
▷ Transmission Control Protocol, Src Port: 80, Dst Port: 58931, Seq: 0, Ack: 1, Len: 0
```

Figure 1.

1.1.1 Explore the TCP Header of this TCP packets in your capture and fill in the picture below.

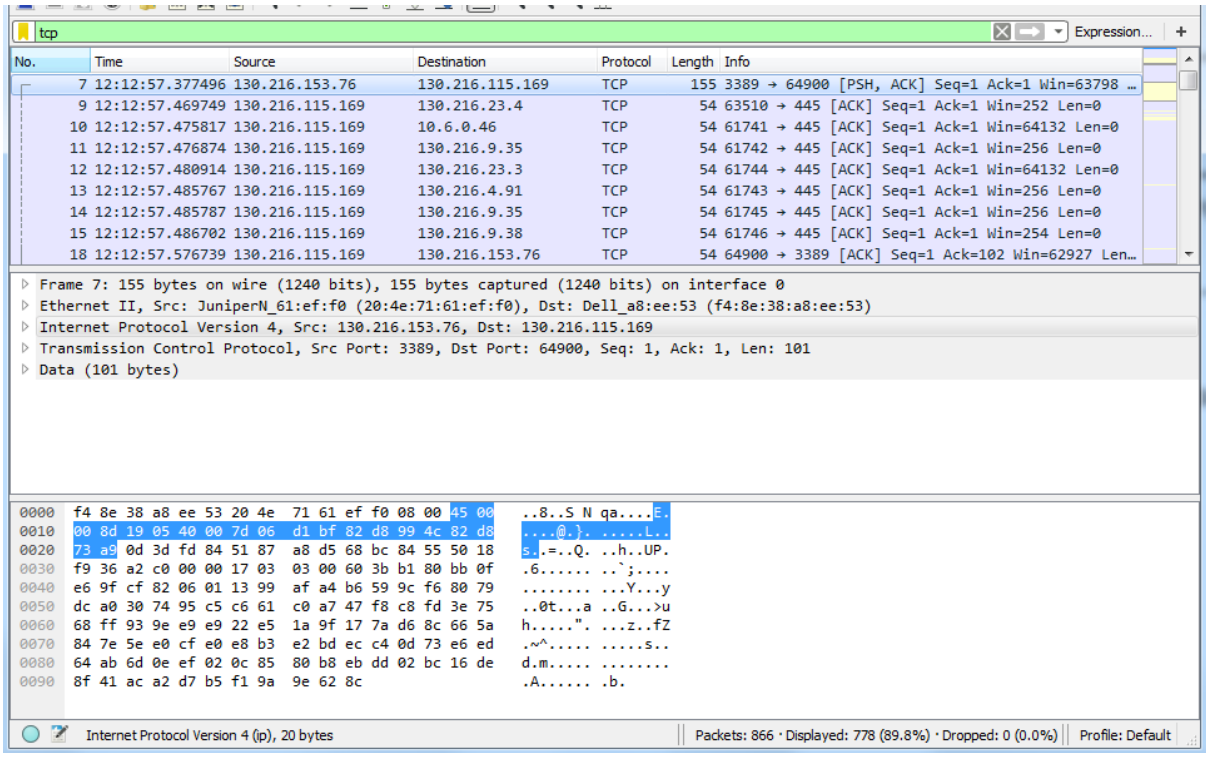
Source Port # 80								Destination Port # 57262
Sequence # 0								
Acknowledgement # 1								
Header Len 32 bytes	Not Used	U 0	A 1	P 0	R 0	S 1	F 0	Receive Window 29200
Checksum 0x3776								Urg Pointer 0
Options: Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted, No-Operation (NOP), Window scale								
Data								



```
Transmission Control Protocol, Src Port: 80, Dst Port: 63772, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 63772
    [Stream index: 6]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 2798090025
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 4291105587
    1000 .... = Header Length: 32 bytes (8)
    Flags: 0x012 (SYN, 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
        .... .... ..1. = Syn: Set
        .... .... ...0 = Fin: Not set
        [TCP Flags: ·······A··S·]
```

Figure 2: TPC Header

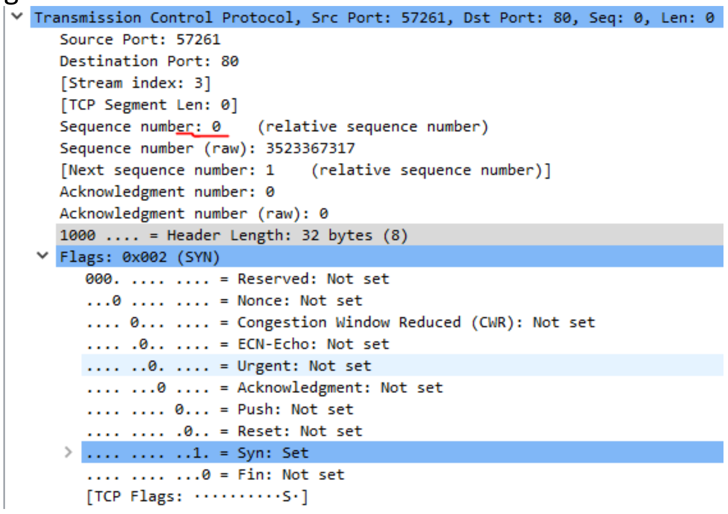
1.2 Clear out the previous filter and set your filter to "`tcp`", you should see a window similar to this:



```
tcp                                                                                    Expression...  +

No.   Time             Source           Destination      Protocol Length Info
    7 12:12:57.377496 130.216.153.76   130.216.115.169  TCP        155 3389 → 64900 [PSH, ACK] Seq=1 Ack=1 Win=63798 …
    9 12:12:57.469749 130.216.115.169  130.216.23.4     TCP         54 63510 → 445 [ACK] Seq=1 Ack=1 Win=252 Len=0
   10 12:12:57.475817 130.216.115.169  10.6.0.46        TCP         54 61741 → 445 [ACK] Seq=1 Ack=1 Win=64132 Len=0
   11 12:12:57.476874 130.216.115.169  130.216.9.35     TCP         54 61742 → 445 [ACK] Seq=1 Ack=1 Win=256 Len=0
   12 12:12:57.480914 130.216.115.169  130.216.23.3     TCP         54 61744 → 445 [ACK] Seq=1 Ack=1 Win=64132 Len=0
   13 12:12:57.485767 130.216.115.169  130.216.4.91     TCP         54 61743 → 445 [ACK] Seq=1 Ack=1 Win=256 Len=0
   14 12:12:57.485787 130.216.115.169  130.216.9.35     TCP         54 61745 → 445 [ACK] Seq=1 Ack=1 Win=256 Len=0
   15 12:12:57.486702 130.216.115.169  130.216.9.38     TCP         54 61746 → 445 [ACK] Seq=1 Ack=1 Win=254 Len=0
   18 12:12:57.576739 130.216.115.169  130.216.153.76   TCP         54 64900 → 3389 [ACK] Seq=1 Ack=102 Win=62927 Len…

▷ Frame 7: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits) on interface 0
▷ Ethernet II, Src: JuniperN_61:ef:f0 (20:4e:71:61:ef:f0), Dst: Dell_a8:ee:53 (f4:8e:38:a8:ee:53)
▷ Internet Protocol Version 4, Src: 130.216.153.76, Dst: 130.216.115.169
▷ Transmission Control Protocol, Src Port: 3389, Dst Port: 64900, Seq: 1, Ack: 1, Len: 101
▷ Data (101 bytes)

0000  f4 8e 38 a8 ee 53 20 4e  71 61 ef f0 08 00 45 00   ..8..S N qa....E.
0010  00 8d 19 05 40 00 7d 06  d1 bf 82 d8 99 4c 82 d8   ....@.}. .....L..
0020  73 a9 0d 3d fd 84 51 87  a8 d5 68 bc 84 55 50 18   s..=..Q. ..h..UP.
0030  f9 36 a2 c0 00 00 17 03  03 00 60 3b b1 80 bb 0f   .6...... ..`;....
0040  e6 9f cf 82 06 01 13 99  af a4 b6 59 9c f6 80 79   ........ ...Y...y
0050  dc a0 30 74 95 c5 c6 61  c0 a7 47 f8 c8 fd 3e 75   ..0t...a ..G...>u
0060  68 ff 93 9e e9 e9 22 e5  1a 9f 17 7a d6 8c 66 5a   h.....". ...z..fZ
0070  84 7e 5e e0 cf e0 e8 b3  e2 bd ec c4 0d 73 e6 ed   .~^..... .....s..
0080  64 ab 6d 0e ef 02 0c 85  80 b8 eb dd 02 bc 16 de   d.m..... ........
0090  8f 41 ac a2 d7 b5 f1 9a  9e 62 8c                  .A...... .b.

Internet Protocol Version 4 (ip), 20 bytes    Packets: 866 · Displayed: 778 (89.8%) · Dropped: 0 (0.0%)    Profile: Default
```

Questions

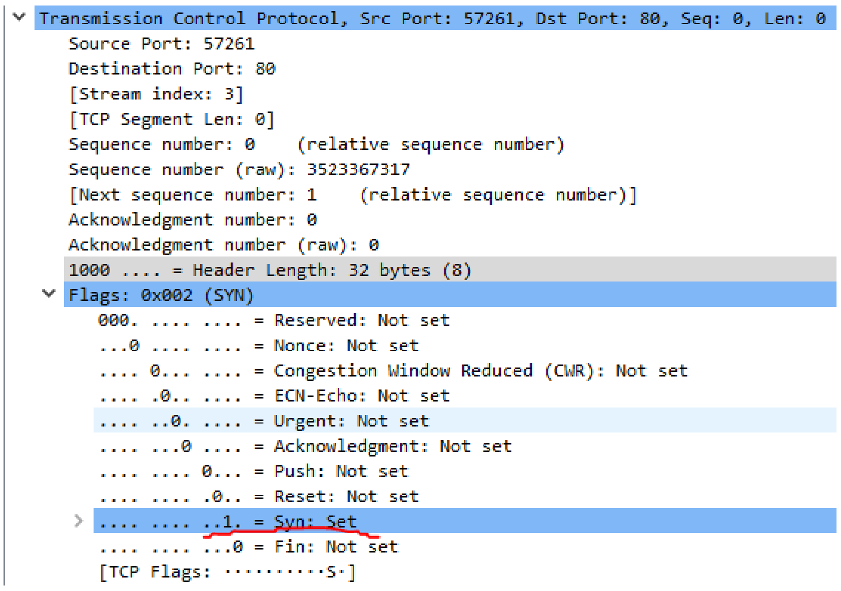
1.2.1 What is the relative sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu?



```
▾ Transmission Control Protocol, Src Port: 57261, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 57261
    Destination Port: 80
    [Stream index: 3]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 3523367317
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  ▾ Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    > .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ··········S·]
```

The relative sequence number of the segment is 0.

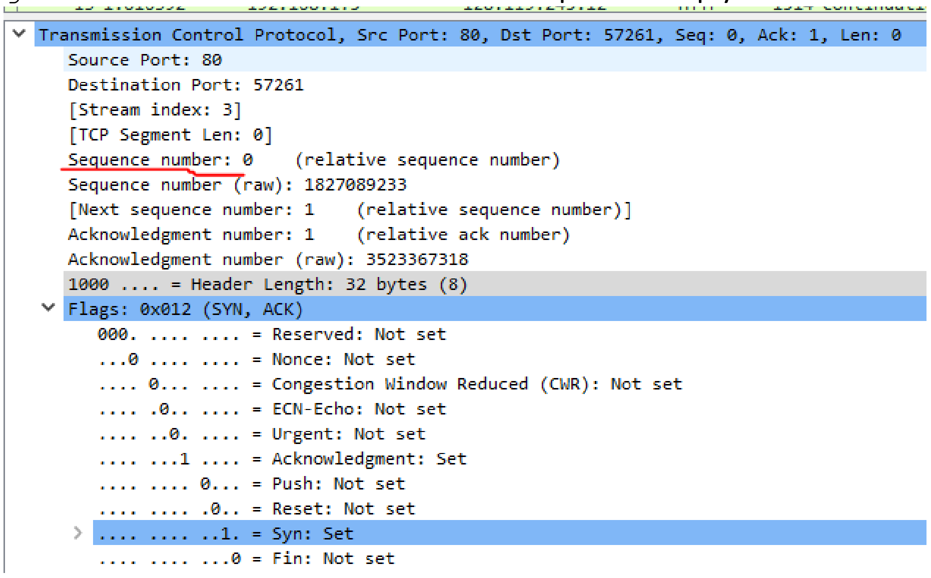
1.2.2 What is it in the segment that identifies the segment as a SYN segment?



```
Transmission Control Protocol, Src Port: 57261, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 57261
    Destination Port: 80
    [Stream index: 3]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 3523367317
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
    Flags: 0x002 (SYN)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Nonce: Not set
        .... 0... .... = Congestion Window Reduced (CWR): Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...0 .... = Acknowledgment: Not set
        .... .... 0... = Push: Not set
        .... .... .0.. = Reset: Not set
        .... .... ..1. = Syn: Set
        .... .... ...0 = Fin: Not set
        [TCP Flags: ··········S·]
```

The flag for Syn is set to 1 which indicates that the segment is a SYN segment.

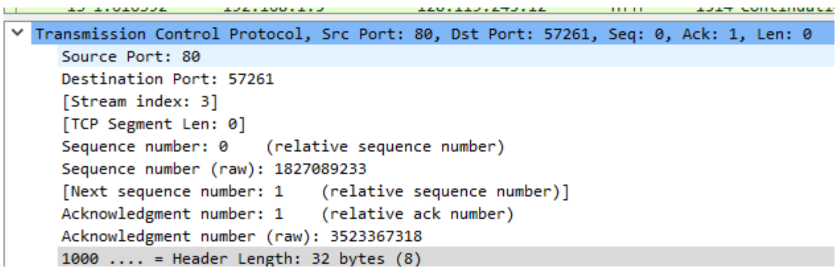
1.2.3 What is the relative sequence number of the SYNACK segment sent by `gaia.cs.umass.edu` to the client computer in reply to the SYN?



```
Transmission Control Protocol, Src Port: 80, Dst Port: 57261, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 57261
    [Stream index: 3]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 1827089233
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 3523367318
    1000 .... = Header Length: 32 bytes (8)
    Flags: 0x012 (SYN, 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
        .... .... ..1. = Syn: Set
        .... .... ...0 = Fin: Not set
```

The relative sequence number is 0.

1.2.4 What is the value of the Acknowledgement field in the SYNACK segment?



```
Transmission Control Protocol, Src Port: 80, Dst Port: 57261, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 57261
    [Stream index: 3]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Sequence number (raw): 1827089233
    [Next sequence number: 1    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 3523367318
    1000 .... = Header Length: 32 bytes (8)
```

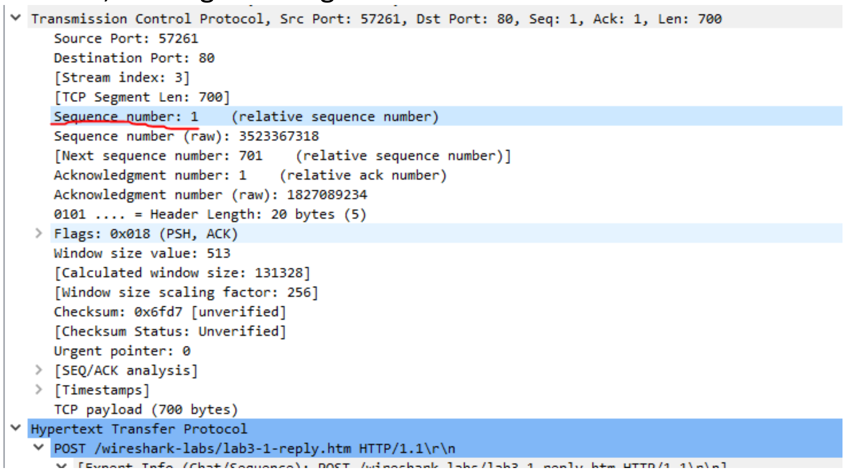
The value of the Acknowledgement field is 1.

1.2.5 How did `gaia.cs.umass.edu` determine the value in 1.2.4?

The value of the acknowledgement field corresponds to the sequence number + 1 for the segment it is acknowledging. The segment that is being responded to has a sequence number of 0, hence the 1 in the acknowledgement field.

1.2.6 What is it in the segment that identifies the segment as a SYNACK segment? Both the Acknowledgement flag and the Syn flag are set to 1.

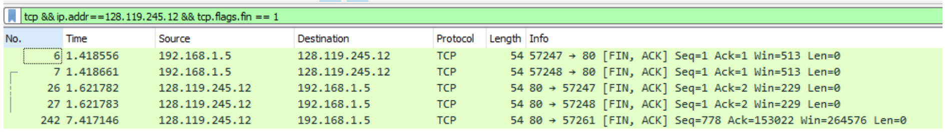
1.2.7 What is the relative sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.



```
v Transmission Control Protocol, Src Port: 57261, Dst Port: 80, Seq: 1, Ack: 1, Len: 700
    Source Port: 57261
    Destination Port: 80
    [Stream index: 3]
    [TCP Segment Len: 700]
    Sequence number: 1    (relative sequence number)
    Sequence number (raw): 3523367318
    [Next sequence number: 701    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 1827089234
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    Window size value: 513
    [Calculated window size: 131328]
    [Window size scaling factor: 256]
    Checksum: 0x6fd7 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (700 bytes)
v Hypertext Transfer Protocol
  v POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1\r\n
    v [Expert Info (Chat/Sequence): POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1\r\n]
```

The relative sequence number is 1.

1.2.8 Did you managed to capture the connection close down between yourself and `gaia.cs.umass?` (Hint what flag would be set in the TCP header?)



```
tcp && ip.addr==128.119.245.12 && tcp.flags.fin == 1

No.  Time      Source          Destination     Protocol Length Info
  6 1.418556   192.168.1.5     128.119.245.12  TCP      54 57247 → 80 [FIN, ACK] Seq=1 Ack=1 Win=513 Len=0
  7 1.418661   192.168.1.5     128.119.245.12  TCP      54 57248 → 80 [FIN, ACK] Seq=1 Ack=1 Win=513 Len=0
 26 1.621782   128.119.245.12  192.168.1.5     TCP      54 80 → 57247 [FIN, ACK] Seq=1 Ack=2 Win=229 Len=0
 27 1.621783   128.119.245.12  192.168.1.5     TCP      54 80 → 57248 [FIN, ACK] Seq=1 Ack=2 Win=229 Len=0
242 7.417146   128.119.245.12  192.168.1.5     TCP      54 80 → 57261 [FIN, ACK] Seq=778 Ack=153022 Win=264576 Len=0
```

Yes, the FIN flag is set in both the sender and receiver, so the connection is closed.

Exercise 2

Objectives for Exercise 2:

- To investigate the TCP stream.

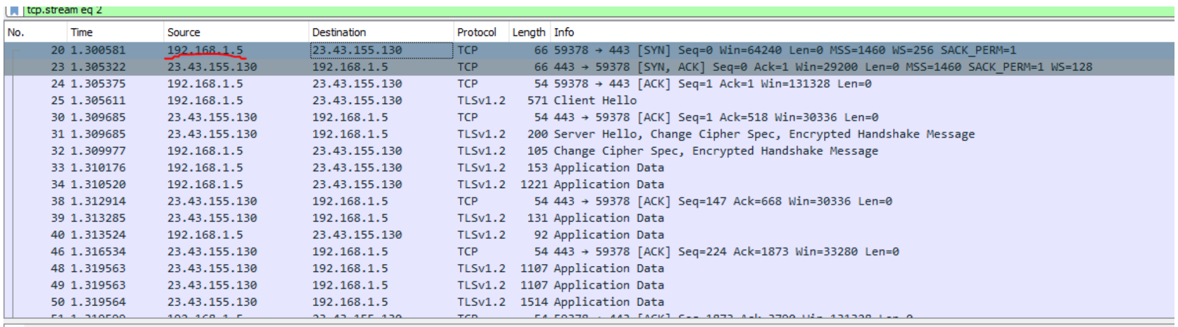
Following TCP streams make it possible to see the data from a TCP stream in the way that the application layer sees it. For example, trying to make sense of a data stream (the information been sent) or even the password. The stream content is displayed in the same sequence as it appeared on the network. Traffic from A to B is marked in red, while traffic from B to A is marked in blue by default. Looking through the trace packet-by-packet is time-consuming, tedious and prone to error, but through using the Follow TCP Stream feature in Wireshark we can work efficiently. By clicking on each of the messages in the TCP Stream Display window you can see the corresponding packet in the Packet List Pane

2.1 Start Wireshark, open www.cisco.com stop Wireshark.

- Identify which communication (packets) are as a result of visiting www.cisco.com. Right-click any of the TCP packets in the display window and choose “Follow TCP Stream”. The follow TCP Stream in Wireshark is the way in which the application layer sees the data transmission. Following TCP streams provides a different view on network traffic: instead of individual packets, one can see data flowing between client and server. In case you are trying to make sense of a data stream, or just for filtering purpose. TCP stream will make troubleshooting easier. Notice that the text displayed in the window is in two colours, red text (source to destination), and blue text (destination to source).

Questions

2.1.1 In the TCP Stream you got in 2.1 what is the IP address of the side that initiated the communication?

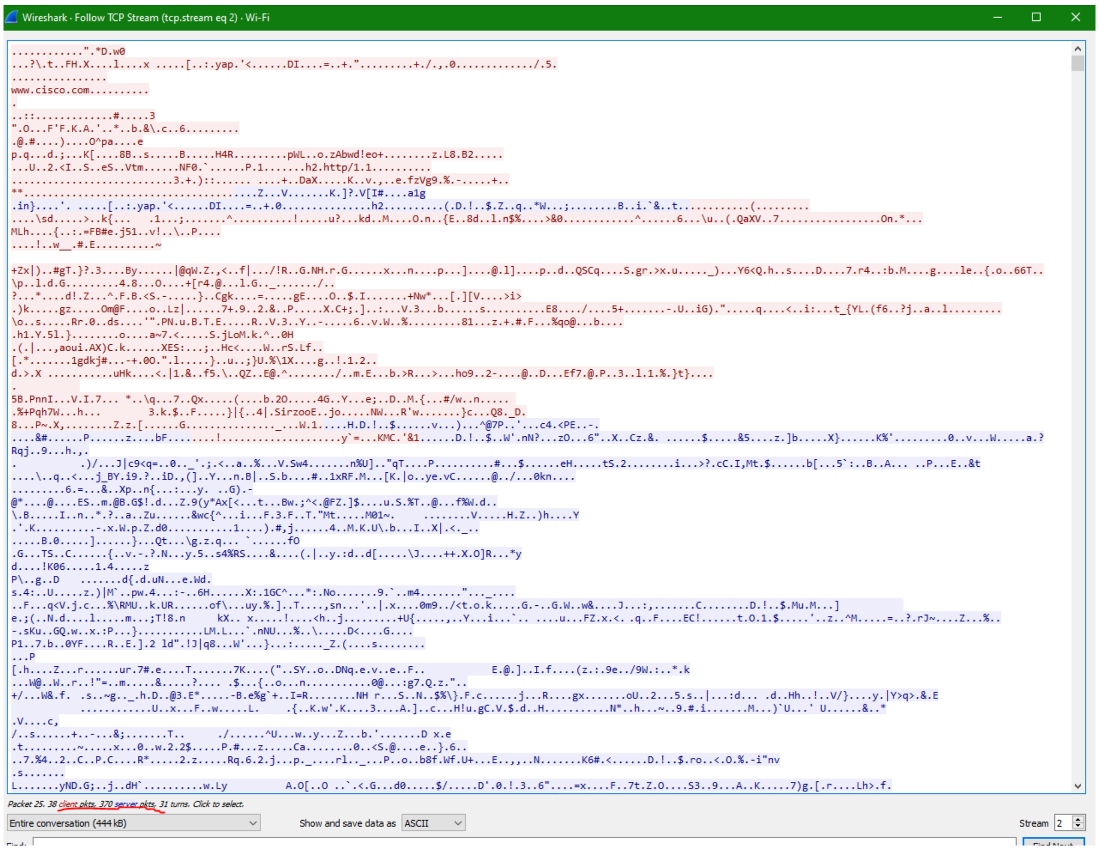


tcp.stream eq 2

No.	Time	Source	Destination	Protocol	Length	Info
20	1.300581	192.168.1.5	23.43.155.130	TCP	66	59378 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
23	1.305322	23.43.155.130	192.168.1.5	TCP	66	443 → 59378 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
24	1.305375	192.168.1.5	23.43.155.130	TCP	54	59378 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
25	1.305611	192.168.1.5	23.43.155.130	TLSv1.2	571	Client Hello
30	1.309685	23.43.155.130	192.168.1.5	TCP	54	443 → 59378 [ACK] Seq=1 Ack=518 Win=30336 Len=0
31	1.309685	23.43.155.130	192.168.1.5	TLSv1.2	200	Server Hello, Change Cipher Spec, Encrypted Handshake Message
32	1.309977	192.168.1.5	23.43.155.130	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
33	1.310176	192.168.1.5	23.43.155.130	TLSv1.2	153	Application Data
34	1.310520	192.168.1.5	23.43.155.130	TLSv1.2	1221	Application Data
38	1.312914	23.43.155.130	192.168.1.5	TCP	54	443 → 59378 [ACK] Seq=147 Ack=668 Win=30336 Len=0
39	1.313285	23.43.155.130	192.168.1.5	TLSv1.2	131	Application Data
40	1.313524	192.168.1.5	23.43.155.130	TLSv1.2	92	Application Data
46	1.316534	23.43.155.130	192.168.1.5	TCP	54	443 → 59378 [ACK] Seq=224 Ack=1873 Win=33280 Len=0
48	1.319563	23.43.155.130	192.168.1.5	TLSv1.2	1107	Application Data
49	1.319563	23.43.155.130	192.168.1.5	TLSv1.2	1107	Application Data
50	1.319564	23.43.155.130	192.168.1.5	TLSv1.2	1514	Application Data

The ip address that initiated the communication is 192.168.1.5

2.1.2 In the TCP Stream you got in 2.1 how many turns did the message go between client and server. Attach screenshot as evidence.



Wireshark · Follow TCP Stream (tcp.stream eq 2) · Wi-Fi

```
...........".*D.w0
...?\.t..FH.X....l....x .....[..:.yap.'<......DI....=..+."........+./.,.0............./.5.
................
www.cisco.com.........
.
..::.............#.....3
".O...F'F.K.A.'..*..b.&\.c..6.........
.@.#....)....O^pa....e
p.q...d.;...K[....8B..s.....B.....H4R.........pWL..o.zAbwd!eo+........z.L8.B2.....
...U..2.<I..S..eS..Vtm......NF0.`......P.1.......h2.http/1.1.........
...........................3.+.)::......  ....+..DaX.....K..v.,..e.fzVg9.%.-.....+..
**..................................Z...V.......K.]?.V[I#....a1g
.in}....'. .....[..:.yap.'<......DI....=..+.0...............h2..........(.D.!..$.Z..q..*W...;........B..i.`&..t...........(.........
....\sd.....>..k{...   .1...;.......^..........!.....u?...kd..M....O.n..{E..8d..l.n$%....>&0...........^......6...\u..(.QaXV..7.................On.*...
MLh....{..:.=FB#e.j51..v!..\..P....
....!..w__.#.E.........~

+Zx|)..#gT.}?.3....By......|@qW.Z.,<..f|.../!R..G.NH.r.G......x...n....p...]....@.l]....p..d..QSCq....S.gr.>x.u....._)...Y6<Q.h..s....D....7.r4..:b.M.....g....le..{.o..66T..
\p..l.d.G.........4.8...O....+[r4.@...l.G.._......./..
?...*....d!.Z...^.F.B.<S.-.....}..Cgk....=.....gE....O..$.I.......+Nw*...[.][V....>i>
.)k.....gz.....Om@F....o..Lz|......7+.9..2.&..P.....X.C+;.]..:...V.3...b......s..........E8..../....5+.......-.U..iG)."....q....<..i:...t_{YL.(f6..?j..a..l.........
\o..s.....Rr.0..ds....'".PN.u.B.T.E.....R..V.3..Y..-.....6..v.W..%.........81...z.+.#.F...%qo@...b....
.h1.Y.5l.}........o....a~7.<.....S.jLoM.k.^..0H
.(.|...,aoui.AX)C.k......XES:...;..Hc<....W..rS.Lf..
[.*.......1gdkj#...-+.0O.".l.....}..u..;}U.%\1X....g..!.1.2..
...
.V....c,
/..s......+..-...&;.......T..      ./......^U...w..y...Z...b.'.......D x.e
.t.........~.....x...0..w.2.2$.....P.#...z.....Ca........0..<S.@....e..}.6..
..7.%4..2..C..P.C....R*.....2.z.....Rq.6.2.j...p._....rl.._...P..o..b8f.Wf.U+...E..,,..N.......K6#.<......D.!..$.ro..<.O.%.-i"nv
.s.......
L.......yND.G;..j..dH`..........w.Ly          A.O[..O ..`.<.G...d0.....$/.....D'.0.!.3..6"....=x....F..7t.Z.O....S3..9...A..K.....7)g.[.r....Lh>.f.
```

Packet 25. 38 client pkts, 370 server pkts, 31 turns. Click to select.

Entire conversation (444 kB)

Show and save data as ASCII

Stream 2

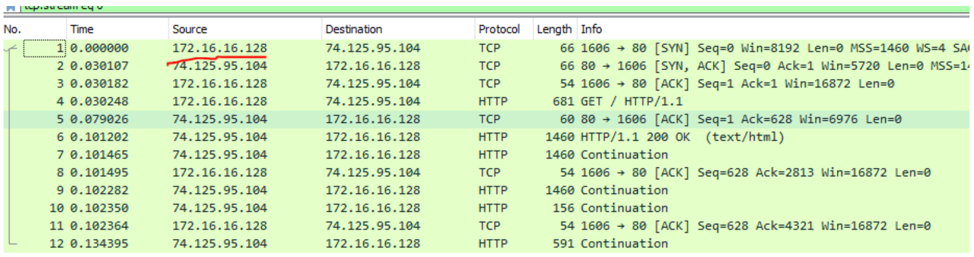
In total there were 31 turns of messages between the client and server.

2.2 Open the file named “`captureTCP1.pcap`”.

- Right-click any of the TCP or HTTP packets in the display window and choose “Follow TCP Stream”.

Questions

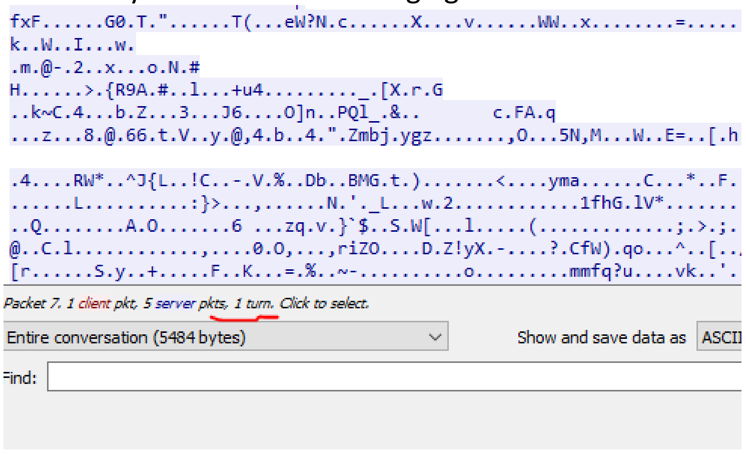
2.2.1 In the TCP Stream you got in 2.1 what is the IP address of the side that initiated the communication?



```
No.   Time       Source          Destination     Protocol  Length Info
    1 0.000000   172.16.16.128   74.125.95.104   TCP         66 1606 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SA
    2 0.030107   74.125.95.104   172.16.16.128   TCP         66 80 → 1606 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1
    3 0.030182   172.16.16.128   74.125.95.104   TCP         54 1606 → 80 [ACK] Seq=1 Ack=1 Win=16872 Len=0
    4 0.030248   172.16.16.128   74.125.95.104   HTTP       681 GET / HTTP/1.1
    5 0.079026   74.125.95.104   172.16.16.128   TCP         60 80 → 1606 [ACK] Seq=1 Ack=628 Win=6976 Len=0
    6 0.101202   74.125.95.104   172.16.16.128   HTTP      1460 HTTP/1.1 200 OK  (text/html)
    7 0.101465   74.125.95.104   172.16.16.128   HTTP      1460 Continuation
    8 0.101495   172.16.16.128   74.125.95.104   TCP         54 1606 → 80 [ACK] Seq=628 Ack=2813 Win=16872 Len=0
    9 0.102282   74.125.95.104   172.16.16.128   HTTP      1460 Continuation
   10 0.102350   74.125.95.104   172.16.16.128   HTTP       156 Continuation
   11 0.102364   172.16.16.128   74.125.95.104   TCP         54 1606 → 80 [ACK] Seq=628 Ack=4321 Win=16872 Len=0
   12 0.134395   74.125.95.104   172.16.16.128   HTTP       591 Continuation
```

The IP address that initiates the communication is 172.16.16.128 as it is sends the first request with the SYN flag set.

2.2.2 How many turns did the message go between client and server,



```
fxF......G0.T."......T(...eW?N.c......X....v......WW..x........=.....
k..W..I...w.
.m.@-.2..x...o.N.#
H......>.{R9A.#..l...+u4.........._.[X.r.G
..k~C.4...b.Z...3...J6....O]n..PQl_.&..       c.FA.q
...z...8.@.66.t.V..y.@,4.b..4.".Zmbj.ygz.......,O...5N,M...W..E=..[.h

.4....RW*..^J{L..!C..-.V.%..Db..BMG.t.).......<....yma......C...*..F.
......L..........:}>...,......N.'._L...w.2............1fhG.lV*.......
..Q........A.O.......6 ...zq.v.}`$..S.W[...l.....(.............;.>.;.
@..C.l..............,....0.O,...,riZO....D.Z!yX.-....?.CfW).qo...^..[..
[r......S.y..+.....F..K...=.%..~-..........o.........mmfq?u....vk..'.
```

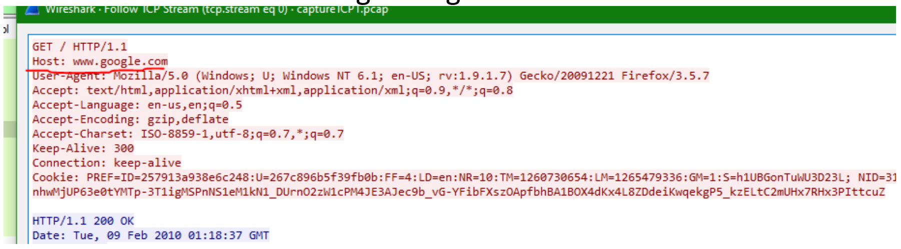
Packet 7. 1 client pkt, 5 server pkts, 1 turn. Click to select.

Entire conversation (5484 bytes)   Show and save data as ASCII

Find:

There was 1 turn between the client and server.

2.2.3 what is the name of the originating source host



Wireshark · Follow TCP Stream (tcp.stream eq 0) · captureTCP1.pcap

```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.7) Gecko/20091221 Firefox/3.5.7
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PREF=ID=257913a938e6c248:U=267c896b5f39fb0b:FF=4:LD=en:NR=10:TM=1260730654:LM=1265479336:GM=1:S=h1UBGonTuWU3D23L; NID=31
nhwMjUP63e0tYMTp-3T1igMSPnNS1eM1kN1_DUrnO2zW1cPM4JE3AJec9b_vG-YFibFXszOApfbhBA1BOX4dKx4L8ZDdeiKwqekgP5_kzELtC2mUHx7RHx3PIttcuZ

HTTP/1.1 200 OK
Date: Tue, 09 Feb 2010 01:18:37 GMT
```

The name is www.google.com

Exercise 3 Python Socket Network Programming (using python 3)

Objective to Exercise 3

- Gaining insight in to Socket programming
- Getting some skills that you’ll be using for your assignment

Sockets are the real backbones behind web browsing. There is always a server and a client in any application. In this exercise, we will explore the writing the client side of the client-server application program. Firstly import a socket library using

```
import socket
```

Make a simple TCP socket

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

The first parameter in the socket instance is `AF_INET` and it refers to the address family IPV4. The second parameter on the other hand `SOCK_STREAM` means connection oriented TCP protocol. For UDP protocol use `SOCK_DGRAM`

Using the socket created, we will connect to a server. We can only connect to a server by knowing its IP address. We want to connect google server. You can get the IP address of google.com in different ways, we will explore 2 ways in this lab. Open command prompt and type the following

```
$ ping www.google.com
```

On the other hand we can use python to get the IP address by using the following command

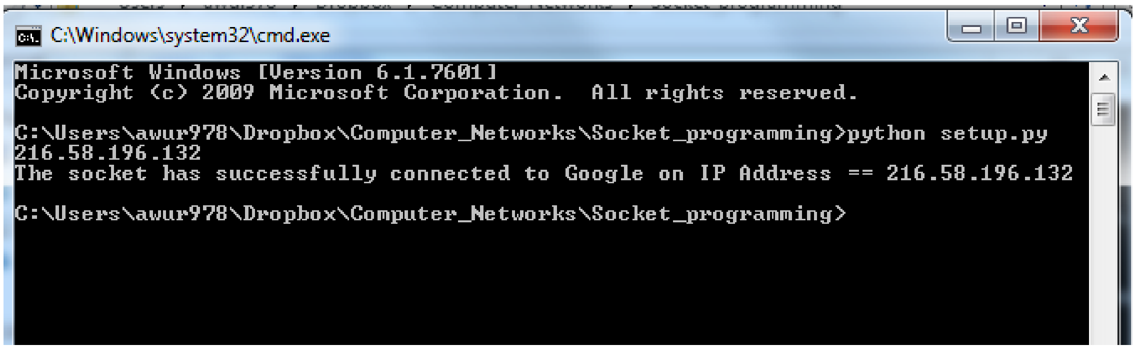
```
import socket

ip =
socket.gethostbyname('www.google.com')
print (ip)
```

Connecting to the server, you need the following the port number, here since we are connecting over the internet (http) we will use port 80. The created socket instance (s) is used to connect.

```
port = 80
s.connect((ip, port))
```

Finally print your result

```
print ("The socket has successfully connected to Google on
IP Address == %s" %(ip))
```



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\awur978\Dropbox\Computer_Networks\Socket_programming>python setup.py
216.58.196.132
The socket has successfully connected to Google on IP Address == 216.58.196.132

C:\Users\awur978\Dropbox\Computer_Networks\Socket_programming>
```

Here, google.com is our server and we do not need to write the server side of the application. We will try and send data (message) to the google server using the function *sendall.* Add the following to your script

```
message = "GET / HTTP/1.1\r\n\r\n"
s.sendall (message.encode())
print ("Message send successfully")
```

The message "`GET / HTTP/1.1\r\n\r\n`" is an http command to fetch the main page of a website

Finally, we will like to receive the message from the server (i.e. google.com main page) by using the function "`recv`". Include the following in your code

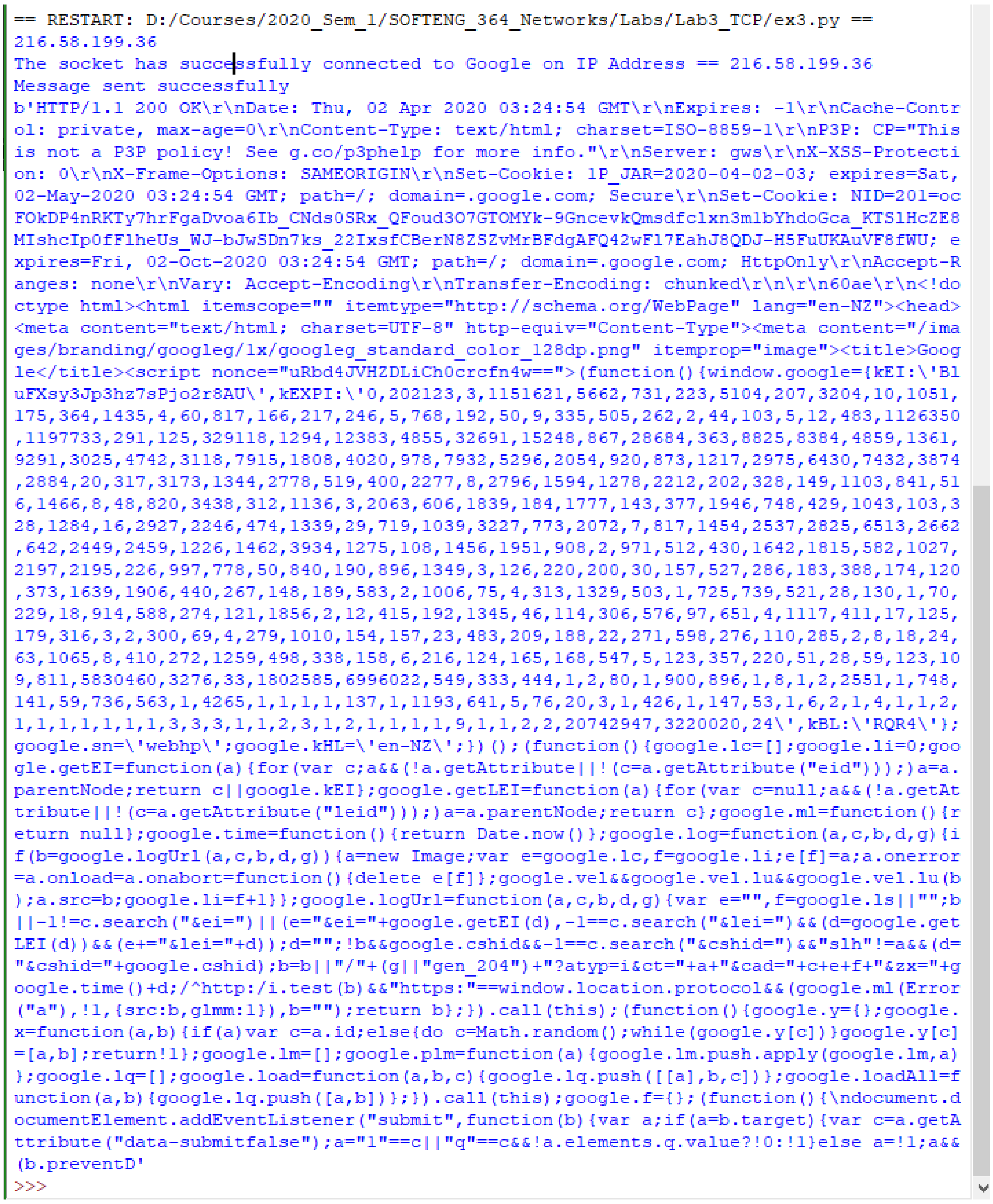
```
reply = s.recv(4096)
print (reply)
```

You should get something similar to the screen shot below

The 4096 is the size of the message buffer, the default is 1024. You can play around with this to see the effect.
Finally, you can close the created socket using

```
s.close()
```

Please include the screenshot of your capture as shown below.



```
== RESTART: D:/Courses/2020_Sem_1/SOFTENG_364_Networks/Labs/Lab3_TCP/ex3.py ==
216.58.199.36
The socket has successfully connected to Google on IP Address == 216.58.199.36
Message sent successfully
b'HTTP/1.1 200 OK\r\nDate: Thu, 02 Apr 2020 03:24:54 GMT\r\nExpires: -1\r\nCache-Contr
ol: private, max-age=0\r\nContent-Type: text/html; charset=ISO-8859-1\r\nP3P: CP="This
is not a P3P policy! See g.co/p3phelp for more info."\r\nServer: gws\r\nX-XSS-Protecti
on: 0\r\nX-Frame-Options: SAMEORIGIN\r\nSet-Cookie: 1P_JAR=2020-04-02-03; expires=Sat,
02-May-2020 03:24:54 GMT; path=/; domain=.google.com; Secure\r\nSet-Cookie: NID=201=oc
FOkDP4nRKTy7hrFgaDvoa6Ib_CNds0SRx_QFoud3O7GTOMYk-9GncevkQmsdfclxn3m1bYhdoGca_KTS1HcZE8
MIshcIp0fFlheUs_WJ-bJwSDn7ks_22IxsfCBerN8ZSZvMrBFdgAFQ42wF17EahJ8QDJ-H5FuUKAuVF8fWU; e
xpires=Fri, 02-Oct-2020 03:24:54 GMT; path=/; domain=.google.com; HttpOnly\r\nAccept-R
anges: none\r\nVary: Accept-Encoding\r\nTransfer-Encoding: chunked\r\n\r\n60ae\r\n<!do
ctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en-NZ"><head>
<meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/ima
ges/branding/googleg/1x/googleg_standard_color_128dp.png" itemprop="image"><title>Goog
le</title><script nonce="uRbd4JVHZDLiCh0crcfn4w==">(function(){window.google={kEI:\'Bl
uFXsy3Jp3hz7sPjo2r8AU\',kEXPI:\'0,202123,3,1151621,5662,731,223,5104,207,3204,10,1051,
175,364,1435,4,60,817,166,217,246,5,768,192,50,9,335,505,262,2,44,103,5,12,483,1126350
,1197733,291,125,329118,1294,12383,4855,32691,15248,867,28684,363,8825,8384,4859,1361,
9291,3025,4742,3118,7915,1808,4020,978,7932,5296,2054,920,873,1217,2975,6430,7432,3874
,2884,20,317,3173,1344,2778,519,400,2277,8,2796,1594,1278,2212,202,328,149,1103,841,51
6,1466,8,48,820,3438,312,1136,3,2063,606,1839,184,1777,143,377,1946,748,429,1043,103,3
28,1284,16,2927,2246,474,1339,29,719,1039,3227,773,2072,7,817,1454,2537,2825,6513,2662
,642,2449,2459,1226,1462,3934,1275,108,1456,1951,908,2,971,512,430,1642,1815,582,1027,
2197,2195,226,997,778,50,840,190,896,1349,3,126,220,200,30,157,527,286,183,388,174,120
,373,1639,1906,440,267,148,189,583,2,1006,75,4,313,1329,503,1,725,739,521,28,130,1,70,
229,18,914,588,274,121,1856,2,12,415,192,1345,46,114,306,576,97,651,4,1117,411,17,125,
179,316,3,2,300,69,4,279,1010,154,157,23,483,209,188,22,271,598,276,110,285,2,8,18,24,
63,1065,8,410,272,1259,498,338,158,6,216,124,165,168,547,5,123,357,220,51,28,59,123,10
9,811,5830460,3276,33,1802585,6996022,549,333,444,1,2,80,1,900,896,1,8,1,2,2551,1,748,
141,59,736,563,1,4265,1,1,1,1,137,1,1193,641,5,76,20,3,1,426,1,147,53,1,6,2,1,4,1,1,2,
1,1,1,1,1,1,1,3,3,3,1,1,2,3,1,2,1,1,1,1,9,1,1,2,2,20742947,3220020,24\',kBL:\'RQR4\'};
google.sn=\'webhp\';google.kHL=\'en-NZ\';})();(function(){google.lc=[];google.li=0;goo
gle.getEI=function(a){for(var c;a&&(!a.getAttribute||!(c=a.getAttribute("eid")));)a=a.
parentNode;return c||google.kEI};google.getLEI=function(a){for(var c=null;a&&(!a.getAt
tribute||!(c=a.getAttribute("leid")));)a=a.parentNode;return c};google.ml=function(){r
eturn null};google.time=function(){return Date.now()};google.log=function(a,c,b,d,g){i
f(b=google.logUrl(a,c,b,d,g)){a=new Image;var e=google.lc,f=google.li;e[f]=a;a.onerror
=a.onload=a.onabort=function(){delete e[f]};google.vel&&google.vel.lu&&google.vel.lu(b
);a.src=b;google.li=f+1}};google.logUrl=function(a,c,b,d,g){var e="",f=google.ls||"";b
||-1!=c.search("&ei=")||(e="&ei="+google.getEI(d),-1==c.search("&lei=")&&(d=google.get
LEI(d))&&(e+="&lei="+d));d="";!b&&google.cshid&&-1==c.search("&cshid=")&&"slh"!=a&&(d=
"&cshid="+google.cshid);b=b||"/"+(g||"gen_204")+"?atyp=i&ct="+a+"&cad="+c+e+f+"&zx="+g
oogle.time()+d;/^http:/i.test(b)&&"https:"==window.location.protocol&&(google.ml(Error
("a"),!1,{src:b,glmm:1}),b="");return b};}).call(this);(function(){google.y={};google.
x=function(a,b){if(a)var c=a.id;else{do c=Math.random();while(google.y[c])}google.y[c]
=[a,b];return!1};google.lm=[];google.plm=function(a){google.lm.push.apply(google.lm,a)
};google.lq=[];google.load=function(a,b,c){google.lq.push([[a],b,c])};google.loadAll=f
unction(a,b){google.lq.push([a,b])};}).call(this);google.f={};(function(){\ndocument.d
ocumentElement.addEventListener("submit",function(b){var a;if(a=b.target){var c=a.getA
ttribute("data-submitfalse");a="1"==c||"q"==c&&!a.elements.q.value?!0:!1}else a=!1;a&&
(b.preventD'
>>>
```