Comp4621 Wireshark labs: TCP and UDP

TA: Waheed Ghali

E-mail: wggghali@connect.ust.hk

Outline

- TCP
  - Review of TCP
  - Capturing a bulk TCP transfer from your computer to a remote server
  - A first look at the captured trace
  - TCP Basics: 3-way handshake, RTT
  - Plot graphs
- UDP
  - Review of UDP
  - UDP basics

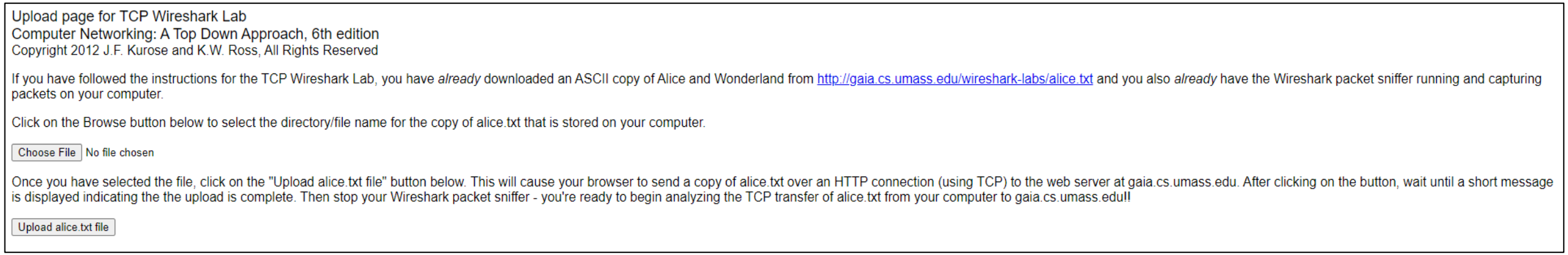
TCP: Review of TCP

- TCP: reliable, in-order delivery
- Sequence numbers: byte stream “number” of first byte in segment’s data
- Acknowledgements: sequence number of next byte expected from other side (cumulative ACK)
- Setting TCP timeout value: based on estimation of RTT by exponential moving average of sampled RTTs
- Establish connection via 3-way handshake: client sends SYN message, server sends SYNACK message, client sends ACK message
- Flow Control: receiver notifies free buffer space through ‘rwnd’ value in TCP header, for which sender limits ‘in-flight’ data

TCP: Capturing a bulk TCP transfer from your computer to a remote server

1. Start up your web browser. Go to 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.

2. Go to http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html


Upload page for TCP Wireshark Lab
Computer Networking: A Top Down Approach, 6th edition
Copyright 2012 J.F. Kurose and K.W. Ross, All Rights Reserved

If you have followed the instructions for the TCP Wireshark Lab, you have *already* downloaded an ASCII copy of Alice and Wonderland from http://gaia.cs.umass.edu/wireshark-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.

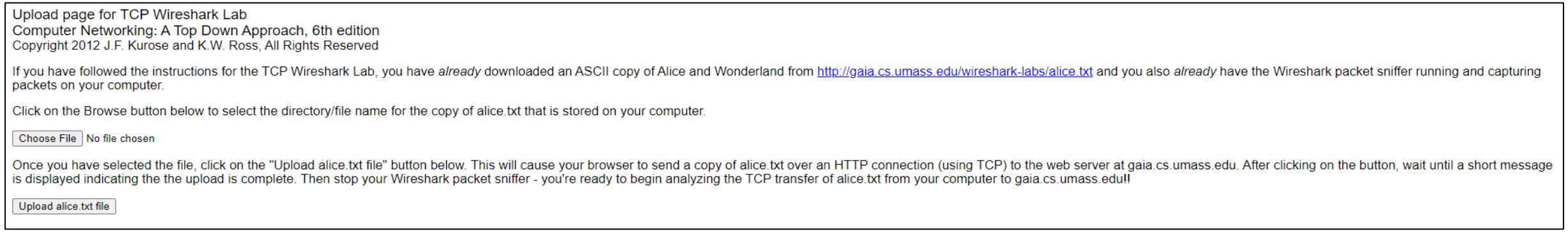
Choose File No file chosen

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

Web page

TCP: Capturing a bulk TCP transfer from your computer to a remote server


Upload page for TCP Wireshark Lab
Computer Networking: A Top Down Approach, 6th edition
Copyright 2012 J.F. Kurose and K.W. Ross, All Rights Reserved

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

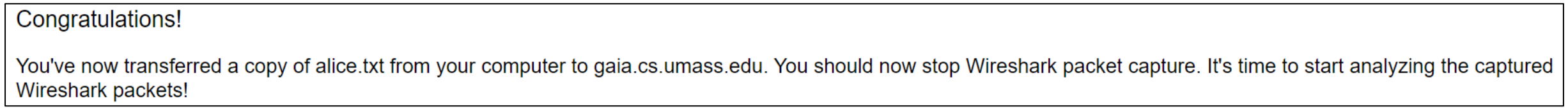
Click on the Browse button below to select the directory/file name for the copy of alice.txt that is stored on your computer.

Choose File No file chosen

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!!

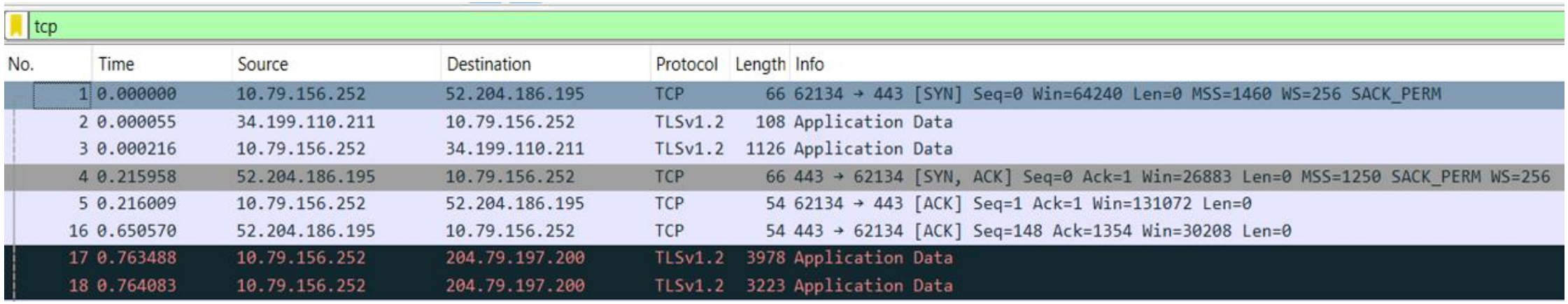
Upload alice.txt file

3. Use the “Choose File” button to choose the file on your computer containing Alice in Wonderland.
4. Start up Wireshark and begin packet capture.
5. Returning to your browser, press the “Upload alice.txt file” button to upload the file to the gaia.cs.umass.edu server.
6. Stop Wireshark packet capture.


Congratulations!

You've now transferred a copy of alice.txt from your computer to gaia.cs.umass.edu. You should now stop Wireshark packet capture. It's time to start analyzing the captured Wireshark packets!

TCP: A first look at the captured trace



tcp

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.79.156.252	52.204.186.195	TCP	66	62134 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000055	34.199.110.211	10.79.156.252	TLSv1.2	108	Application Data
3	0.000216	10.79.156.252	34.199.110.211	TLSv1.2	1126	Application Data
4	0.215958	52.204.186.195	10.79.156.252	TCP	66	443 → 62134 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1250 SACK_PERM WS=256
5	0.216009	10.79.156.252	52.204.186.195	TCP	54	62134 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
16	0.650570	52.204.186.195	10.79.156.252	TCP	54	443 → 62134 [ACK] Seq=148 Ack=1354 Win=30208 Len=0
17	0.763488	10.79.156.252	204.79.197.200	TLSv1.2	3978	Application Data
18	0.764083	10.79.156.252	204.79.197.200	TLSv1.2	3223	Application Data

TCP: captured trace

- The initial three-way handshake

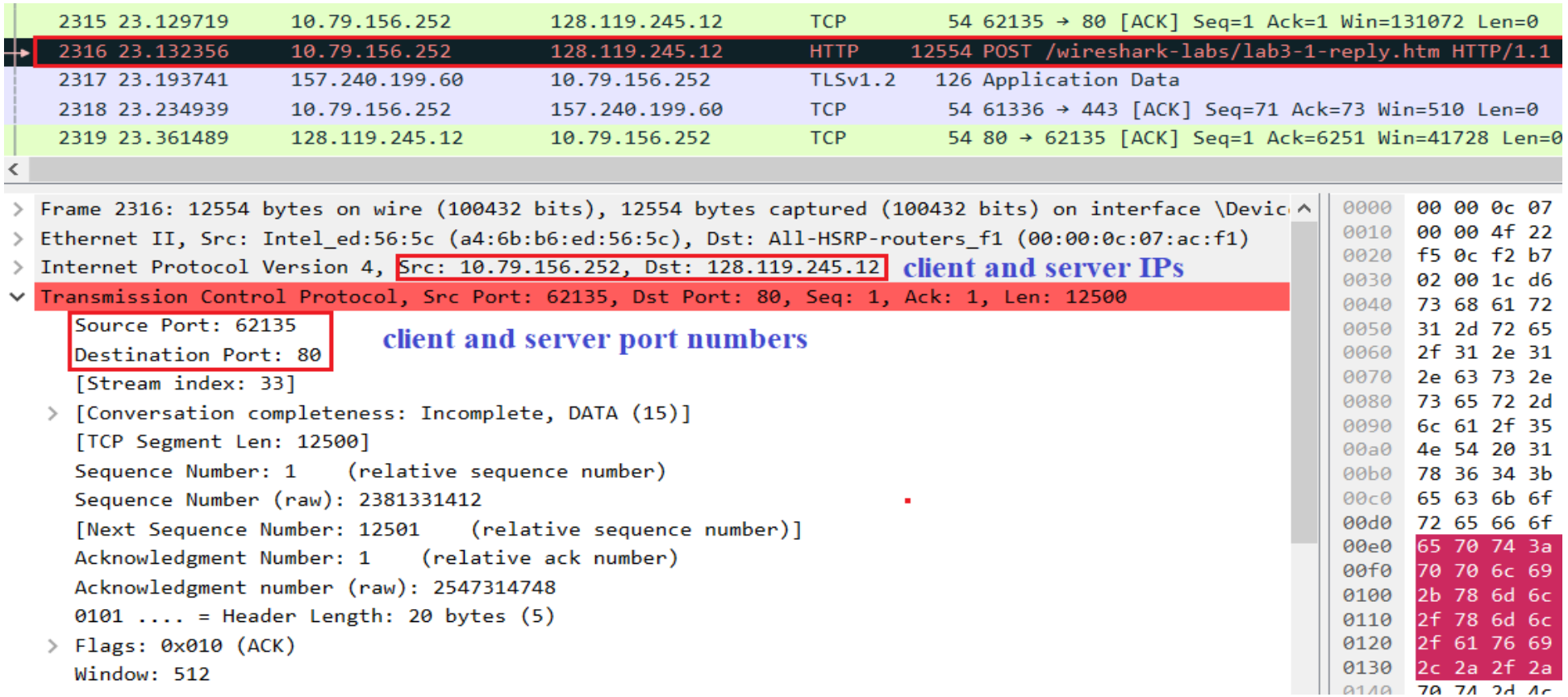
1	0.000000	10.79.156.252	52.204.186.195	TCP	66	62134 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000055	34.199.110.211	10.79.156.252	TLSv1.2	108	Application Data
3	0.000216	10.79.156.252	34.199.110.211	TLSv1.2	1126	Application Data
4	0.215958	52.204.186.195	10.79.156.252	TCP	66	443 → 62134 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1250 SACK_PERM WS=256
5	0.216009	10.79.156.252	52.204.186.195	TCP	54	62134 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0

- Multiple TCP segments used to carry a single HTTP message

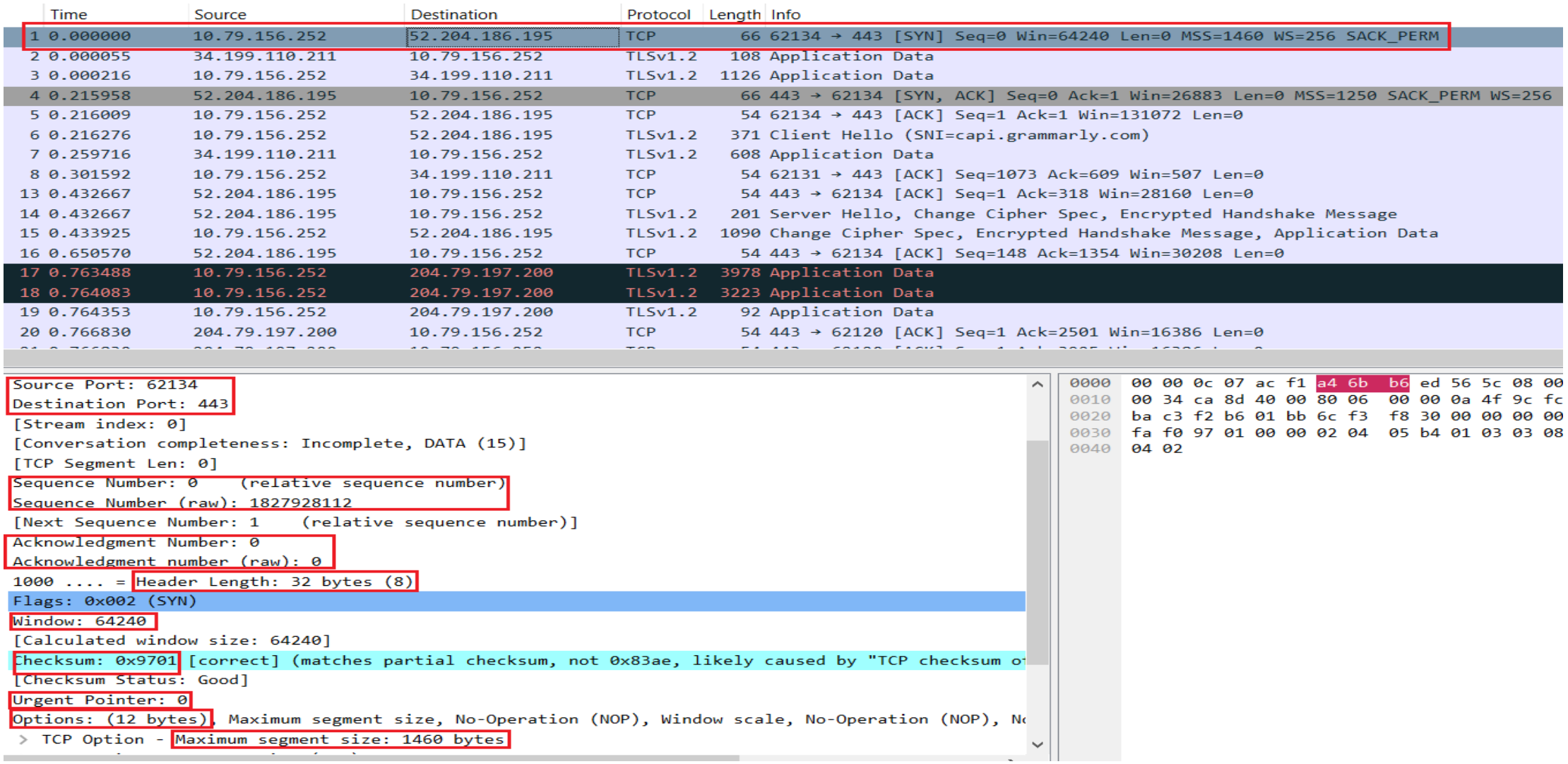
No.	Time	Source	Destination	Protocol	Length	Info
1813	14.490980	92.223.85.120	10.79.156.252	TCP	66	[TCP Keep-Alive ACK] 443 → 55583 [ACK] Seq=1 Ack=2 Win=501 Len=0 SLE=1 SRE=2
1814	15.403369	10.79.156.252	172.64.151.4	TCP	55	62049 → 443 [ACK] Seq=1 Ack=1 Win=510 Len=1 [TCP segment of a reassembled PDU]
1815	15.407472	172.64.151.4	10.79.156.252	TCP	66	443 → 62049 [ACK] Seq=1 Ack=2 Win=8 Len=0 SLE=1 SRE=2
1818	15.574390	10.79.156.252	13.224.163.73	TCP	55	62115 → 443 [ACK] Seq=1 Ack=1 Win=509 Len=1 [TCP segment of a reassembled PDU]
1819	15.578807	13.224.163.73	10.79.156.252	TCP	66	443 → 62115 [ACK] Seq=1 Ack=2 Win=142 Len=0 SLE=1 SRE=2
2242	15.995505	10.79.156.252	157.240.199.1	TLSv1.2	83	Application Data
2243	16.000083	157.240.199.1	10.79.156.252	TCP	54	443 → 56110 [ACK] Seq=80 Ack=3464 Win=3160 Len=0
2244	16.164945	157.240.199.1	10.79.156.252	TLSv1.2	79	Application Data
2245	16.211861	10.79.156.252	157.240.199.1	TCP	54	56110 → 443 [ACK] Seq=3464 Ack=105 Win=510 Len=0
2246	16.396873	10.79.156.252	52.1.247.29	TCP	55	62113 → 443 [ACK] Seq=1 Ack=1 Win=509 Len=1 [TCP segment of a reassembled PDU]
2247	16.411704	10.79.156.252	44.218.89.127	TCP	55	62114 → 443 [ACK] Seq=1 Ack=1 Win=510 Len=1 [TCP segment of a reassembled PDU]
2262	16.616800	52.1.247.29	10.79.156.252	TCP	66	443 → 62113 [ACK] Seq=1 Ack=2 Win=265 Len=0 SLE=1 SRE=2
2263	16.628537	44.218.89.127	10.79.156.252	TCP	66	443 → 62114 [ACK] Seq=1 Ack=2 Win=122 Len=0 SLE=1 SRE=2
2271	17.047320	10.79.156.252	52.111.234.0	TCP	55	61704 → 443 [ACK] Seq=1 Ack=1 Win=508 Len=1 [TCP segment of a reassembled PDU]
2272	17.080711	52.111.234.0	10.79.156.252	TCP	66	443 → 61704 [ACK] Seq=1 Ack=2 Win=2047 Len=0 SLE=1 SRE=2
2273	17.203475	10.79.156.252	52.111.240.8	TLSv1.2	89	Application Data
2274	17.304533	52.111.240.8	10.79.156.252	TCP	54	443 → 61955 [ACK] Seq=1 Ack=36 Win=16386 Len=0

TCP: captured trace (Cont.)

- HTTP POST message



TCP: TCP Basics, 3-way handshake

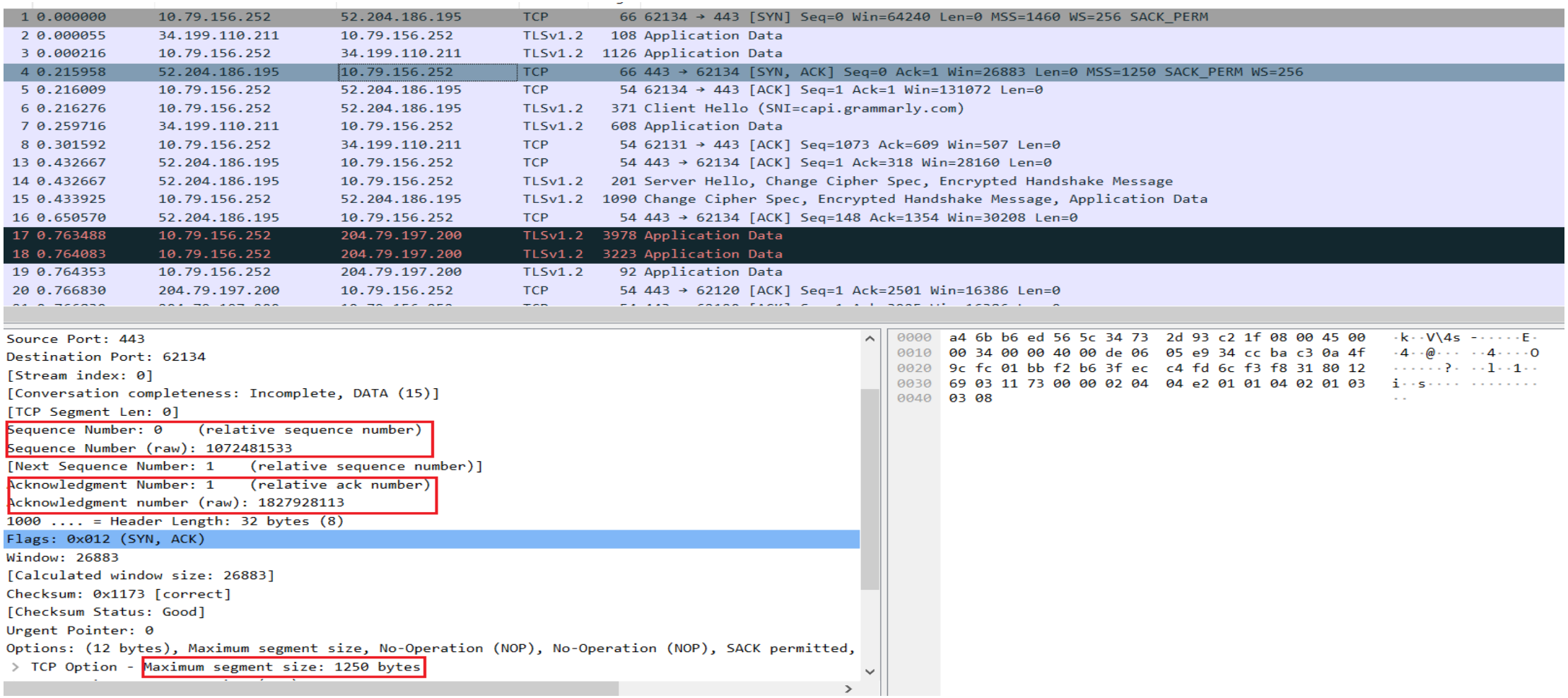


	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.79.156.252	52.204.186.195	TCP	66	62134 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000055	34.199.110.211	10.79.156.252	TLSv1.2	108	Application Data
3	0.000216	10.79.156.252	34.199.110.211	TLSv1.2	1126	Application Data
4	0.215958	52.204.186.195	10.79.156.252	TCP	66	443 → 62134 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1250 SACK_PERM WS=256
5	0.216009	10.79.156.252	52.204.186.195	TCP	54	62134 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
6	0.216276	10.79.156.252	52.204.186.195	TLSv1.2	371	Client Hello (SNI=capi.grammarly.com)
7	0.259716	34.199.110.211	10.79.156.252	TLSv1.2	608	Application Data
8	0.301592	10.79.156.252	34.199.110.211	TCP	54	62131 → 443 [ACK] Seq=1073 Ack=609 Win=507 Len=0
13	0.432667	52.204.186.195	10.79.156.252	TCP	54	443 → 62134 [ACK] Seq=1 Ack=318 Win=28160 Len=0
14	0.432667	52.204.186.195	10.79.156.252	TLSv1.2	201	Server Hello, Change Cipher Spec, Encrypted Handshake Message
15	0.433925	10.79.156.252	52.204.186.195	TLSv1.2	1090	Change Cipher Spec, Encrypted Handshake Message, Application Data
16	0.650570	52.204.186.195	10.79.156.252	TCP	54	443 → 62134 [ACK] Seq=148 Ack=1354 Win=30208 Len=0
17	0.763488	10.79.156.252	204.79.197.200	TLSv1.2	3978	Application Data
18	0.764083	10.79.156.252	204.79.197.200	TLSv1.2	3223	Application Data
19	0.764353	10.79.156.252	204.79.197.200	TLSv1.2	92	Application Data
20	0.766830	204.79.197.200	10.79.156.252	TCP	54	443 → 62120 [ACK] Seq=1 Ack=2501 Win=16386 Len=0

```
Source Port: 62134
Destination Port: 443
[Stream index: 0]
[Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 0]
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 1827928112
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
1000 .... = Header Length: 32 bytes (8)
Flags: 0x002 (SYN)
Window: 64240
[Calculated window size: 64240]
Checksum: 0x9701 [correct] (matches partial checksum, not 0x83ae, likely caused by "TCP checksum o
[Checksum Status: Good]
Urgent Pointer: 0
Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), N
 > TCP Option - Maximum segment size: 1460 bytes
```

```
0000  00 00 0c 07 ac f1 a4 6b  b6 ed 56 5c 08 00
0010  00 34 ca 8d 40 00 80 06  00 00 0a 4f 9c fc
0020  ba c3 f2 b6 01 bb 6c f3  f8 30 00 00 00 00
0030  fa f0 97 01 00 00 02 04  05 b4 01 03 03 08
0040  04 02
```

TCP: TCP Basics, 3-way handshake (Cont.)



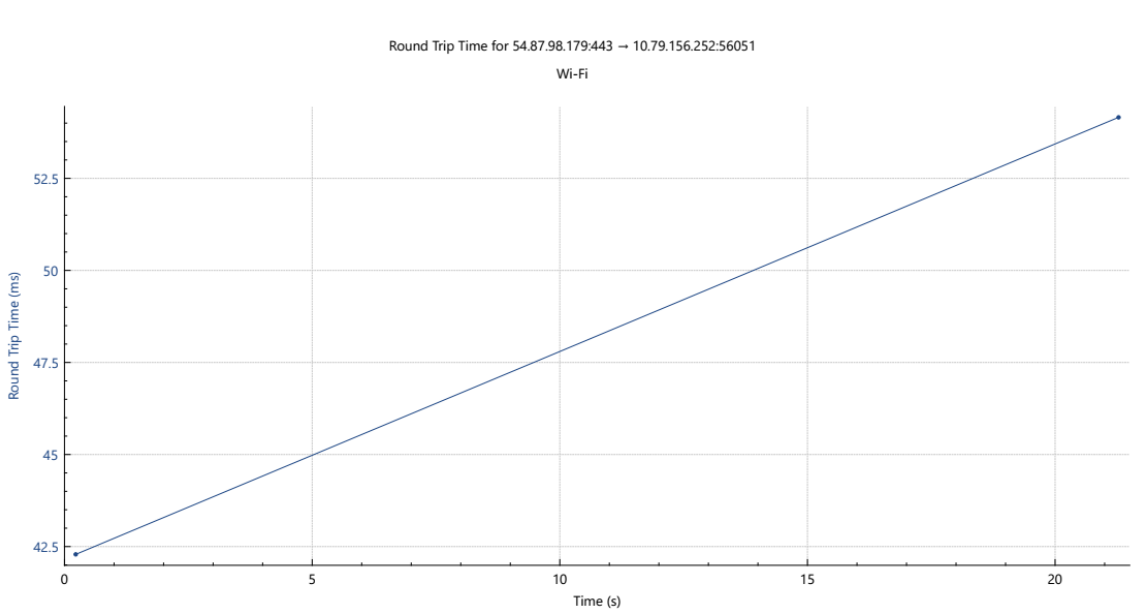
```
 1 0.000000    10.79.156.252    52.204.186.195   TCP       66 62134 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
 2 0.000055    34.199.110.211   10.79.156.252    TLSv1.2  108 Application Data
 3 0.000216    10.79.156.252    34.199.110.211   TLSv1.2 1126 Application Data
 4 0.215958    52.204.186.195   10.79.156.252    TCP       66 443 → 62134 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1250 SACK_PERM WS=256
 5 0.216009    10.79.156.252    52.204.186.195   TCP       54 62134 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
 6 0.216276    10.79.156.252    52.204.186.195   TLSv1.2  371 Client Hello (SNI=capi.grammarly.com)
 7 0.259716    34.199.110.211   10.79.156.252    TLSv1.2  608 Application Data
 8 0.301592    10.79.156.252    34.199.110.211   TCP       54 62131 → 443 [ACK] Seq=1073 Ack=609 Win=507 Len=0
13 0.432667    52.204.186.195   10.79.156.252    TCP       54 443 → 62134 [ACK] Seq=1 Ack=318 Win=28160 Len=0
14 0.432667    52.204.186.195   10.79.156.252    TLSv1.2  201 Server Hello, Change Cipher Spec, Encrypted Handshake Message
15 0.433925    10.79.156.252    52.204.186.195   TLSv1.2 1090 Change Cipher Spec, Encrypted Handshake Message, Application Data
16 0.650570    52.204.186.195   10.79.156.252    TCP       54 443 → 62134 [ACK] Seq=148 Ack=1354 Win=30208 Len=0
17 0.763488    10.79.156.252    204.79.197.200   TLSv1.2 3978 Application Data
18 0.764083    10.79.156.252    204.79.197.200   TLSv1.2 3223 Application Data
19 0.764353    10.79.156.252    204.79.197.200   TLSv1.2   92 Application Data
20 0.766830    204.79.197.200   10.79.156.252    TCP       54 443 → 62120 [ACK] Seq=1 Ack=2501 Win=16386 Len=0
```

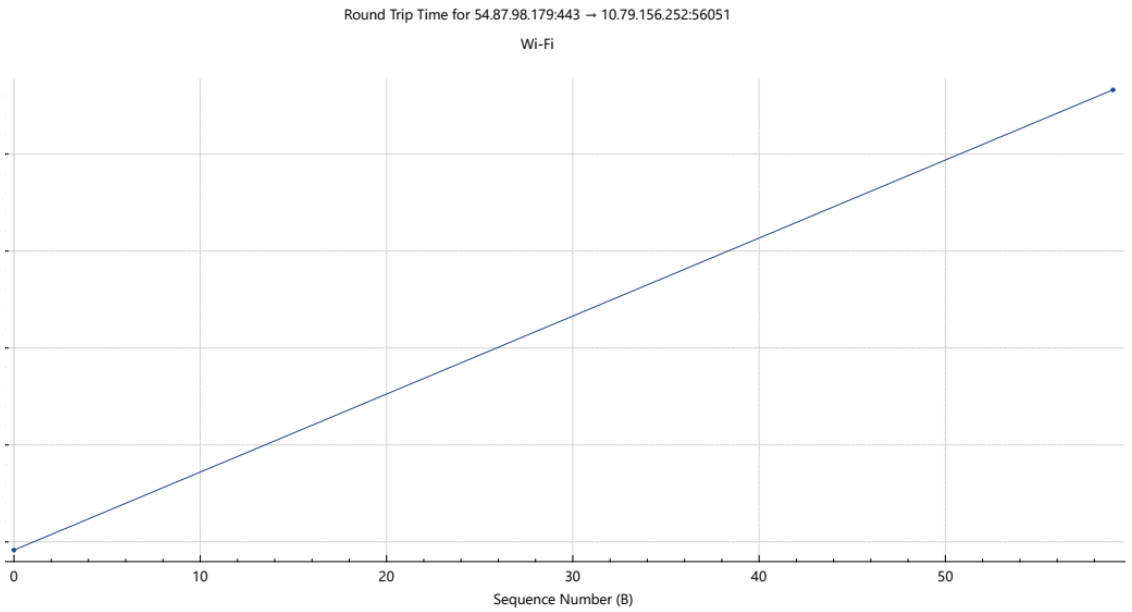
```
Source Port: 443
Destination Port: 62134
[Stream index: 0]
[Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 0]
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 1072481533
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 1827928113
1000 .... = Header Length: 32 bytes (8)
Flags: 0x012 (SYN, ACK)
Window: 26883
[Calculated window size: 26883]
Checksum: 0x1173 [correct]
[Checksum Status: Good]
Urgent Pointer: 0
Options: (12 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted,
 > TCP Option - Maximum segment size: 1250 bytes
```

```
0000  a4 6b b6 ed 56 5c 34 73  2d 93 c2 1f 08 00 45 00   ·k··V\4s -·····E·
0010  00 34 00 00 40 00 de 06  05 e9 34 cc ba c3 0a 4f   ·4··@··· ··4····O
0020  9c fc 01 bb f2 b6 3f ec  c4 fd 6c f3 f8 31 80 12   ·····?· ··l··1··
0030  69 03 11 73 00 00 02 04  04 e2 01 01 04 02 01 03   i··s···· ········
0040  03 08                                              ··
```

TCP: Plot graphs

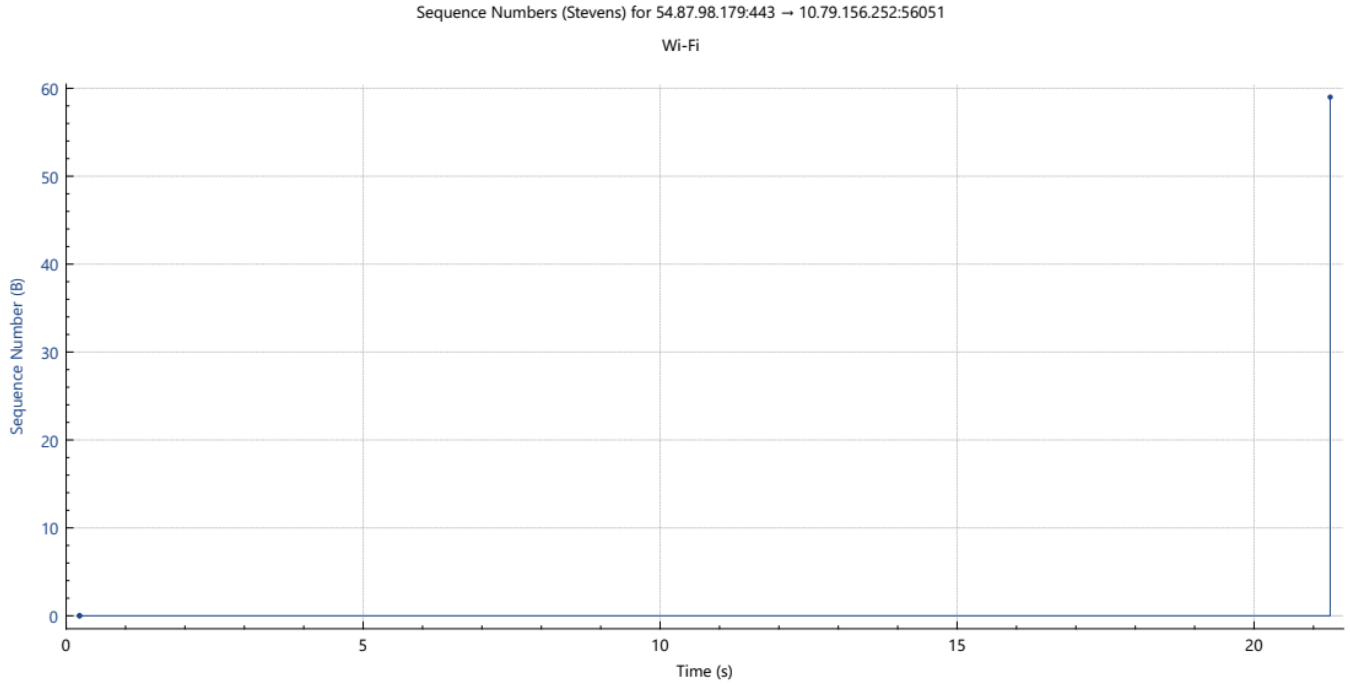
- Plot **the RTT for each of the TCP segments** sent in Wireshark: Select a TCP segment in the "listing of captured packets" window that is being sent from the client to the gaia.cs.umass.edu server. Then select Statistics->TCP Stream Graphs->Round Trip Time.





TCP: Plot graphs

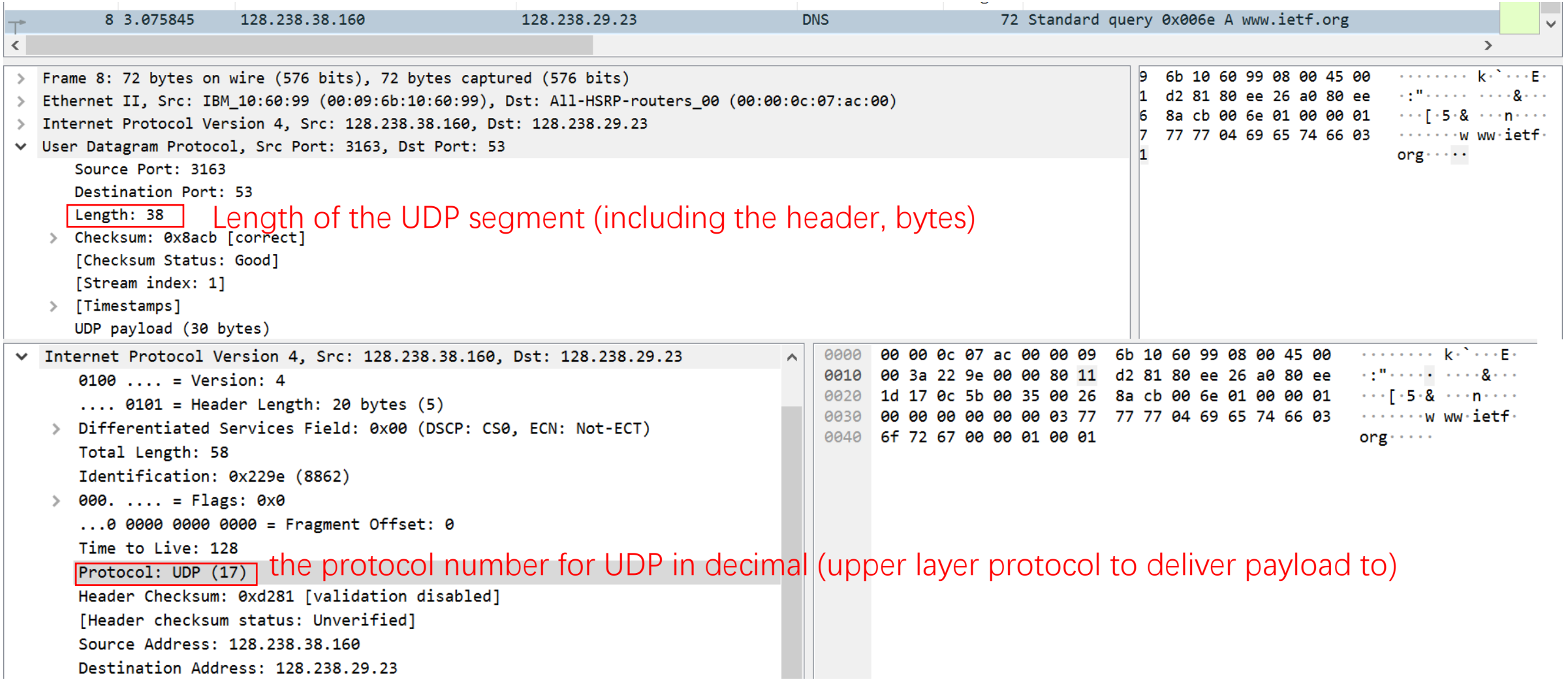
- Plot the **Time-sequence graph** in Wireshark: Select a TCP segment in the Wireshark's "listing of captured-packets" window. Then select the menu : Statistics->TCP Stream Graphs-> Time-Sequence (Stevens).



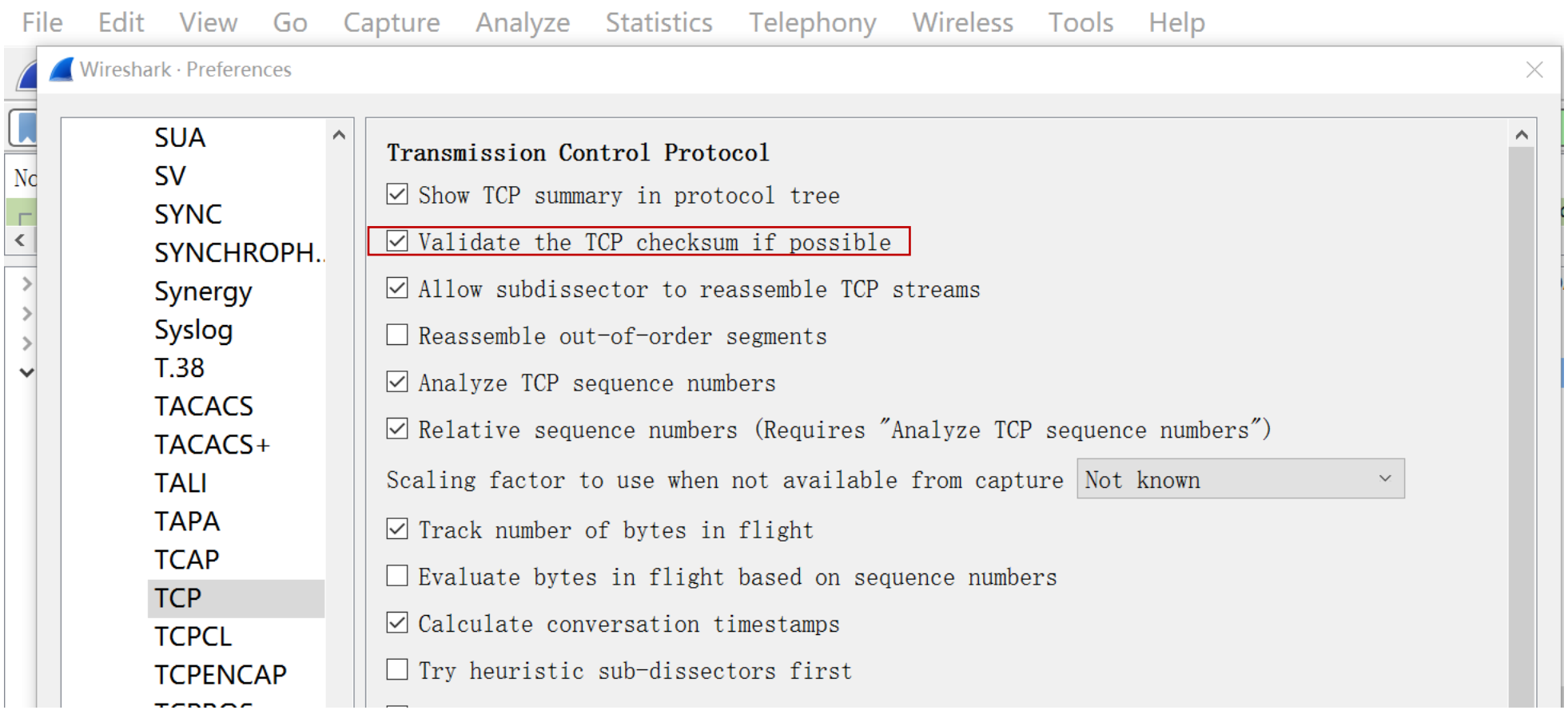
UDP: Review of UDP

- UDP: unreliable, unordered delivery
- Connectionless: No handshaking, each segment handled independently of others
- Checksum: detect errors in transmitted segment

UDP: UDP Basics



Wireshark: validate checksum if possible



Edit → Preferences → Protocols → TCP

Conclusion

- TCP
  - Review of TCP
  - Capturing a bulk TCP transfer from your computer to a remote server
  - A first look at the captured trace
  - TCP Basics: 3-way handshake, RTT
  - Plot graphs: RTT, Time-sequence Graph
- UDP
  - Review of UDP
  - UDP basics

Lab assignment: TCP and UDP

- The exercises will be posted on Canvas.
- For TCP, follow the instructions in ‘Wireshark_TCP_v7.0’ in section 2 (A first look at the captured trace) and answer the following questions: 1, 2, 3.
- For UDP, follow the instructions in ‘Wireshark_UDP_v7.0’ and answer the following questions: 3, 6.
- Submit a typed response with cropped screen captures to the above questions through Canvas.