CPSC 471-06 Assignment #3

Required Software

Wireshark version 3.4.8 or newer must be used for this project.

Project Submission

You are responsible for the content of your submitted file. Double-check your submission in Canvas to ensure that the submitted file is not corrupted, and it is readable, and it contains all required files, and their contents are what you intend to submit. Submitting a wrong file (from another class for example) is not a valid excuse.

Submit **one zip file** using naming convention: **firstnamelastname_p3.zip** in Canvas.

The zip file shall contain the following files:

(1) PDF report file using naming convention **firstnamelastname_p3.pdf** (must be a pdf document, not Word).
(2) Wireshark capture file from Part 1 **xx1.pcapng**
(3) Wireshark capture file from Part 2 **xx2.pcapng**

where xx = initials (the two characters representing the first character of your first and last name).

Managing Submission File Size

Manage your submission File Size by controlling your Wireshark captures and check the zip file size before submitting. Do not exceed 10 Megabytes. A penalty of 10% will be deducted for each 10 MB violation in excess from the allowed 10 MB. For example, penalty for a 10.1 MB to 20 MB submission is 10%, and for a 20.1 MB to 30 MB submission the penalty is 20%. Limit your Wireshark capture file size by starting and stopping the capture process accordingly. Do not let your Wireshark run continuously without stopping. This will create a very large capture file and unnecessary penalty.

Part 1: UDP Transport Layer Protocol

In this project, we'll take a quick look at the UDP transport protocol. As we saw in Chapter 3 of the textbook, UDP is a streamlined, no-frills protocol. You may want to re-read section 3.3 in the textbook before doing this lab.

Start capturing packets in Wireshark and then do something that will cause your host to send and receive several UDP packets.

After stopping packet capture, set your packet filter so that Wireshark only displays the UDP packets sent and received at your host. Pick one of these UDP packets and expand the UDP fields in the details window.

What to Hand in

PDF file report

Create a section called **Part 1 – UDP Transport Layer Protocol**.

Note: You must include a screenshot from Wireshark capture with annotation and sufficient surrounding details when answering any question relating to the captured information. Annotate the screenshot to indicate which information you are referring to in the screenshot. The screenshots should be legible and include enough surrounding details clearly showing where it belonged to.

1. Select *one* UDP packet from your trace. From this packet, determine how many fields there are in the UDP header. Name these fields.
2. By consulting the displayed information in Wireshark's packet content field for this packet, determine the length (in bytes) of each of the UDP header fields.
3. The value in the Length field is the length of what? (Consult the textbook for this answer). Verify your claim with your captured UDP packet. Show your work details.
4. What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to 2. above)
5. What is the largest possible source port number?
6. What is the protocol number for UDP? Give your answer in both hexadecimal and decimal notation.
7. Examine a pair of UDP packets in which your host sends the first UDP packet and the second UDP packet is a reply to this first UDP packet. (Hint: for a second packet to be sent in response to a first packet, the sender of the first packet should be the destination of the second packet). Describe the relationship between the port numbers in the two packets.

Submission zip file

include your Wireshark capture file **xx1.pcapng** in the zip file.

where xx = initials (the two characters representing the first character of your first and last name).

Part 2: TCP Transport Layer Protocol

We'll investigate the behavior of the TCP protocol in detail. We'll do so by analyzing a trace of the TCP segments sent and received in transferring a 150KB file (containing the text of Lewis

Carrol’s *Alice’s Adventures in Wonderland*) from your computer to a remote server. We’ll study TCP’s use of sequence and acknowledgement numbers for providing reliable data transfer; we’ll see TCP’s congestion control algorithm – slow start and congestion avoidance – in action; and we’ll look at TCP’s receiver-advertised flow control mechanism. We’ll also briefly consider TCP connection setup, and we’ll investigate the performance (throughput and round-trip time) of the TCP connection between your computer and the server.

Before beginning, you’ll probably want to review sections 3.5 and 3.7 in the textbook.

Preparation

Perform the following:

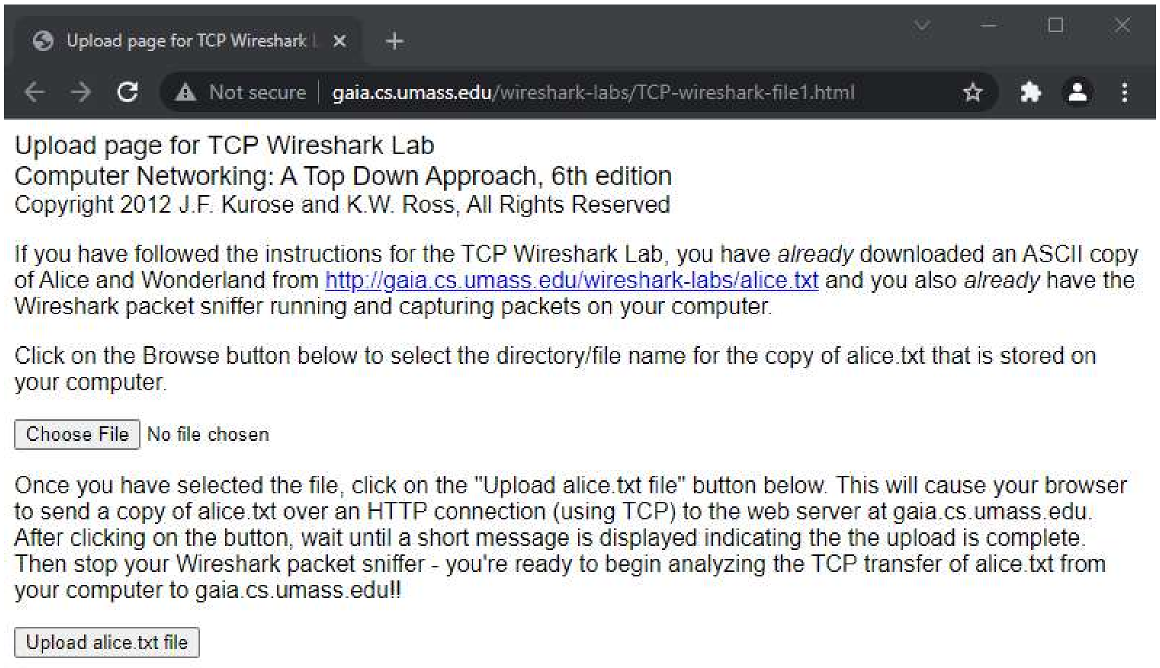
- Since this lab is about TCP rather than HTTP, let’s change Wireshark’s “listing of captured packets” window so that it shows information about the TCP segments containing the HTTP messages, rather than about the HTTP messages. To have Wireshark do this, select *Analyze->Enabled Protocols.* Then uncheck all the HTTP boxes (HTTP, HTTP2, HTTP3) and select *OK*.

- Filter the packets displayed in the Wireshark window by entering “tcp” (lowercase, no quotes, and don’t forget to press return after entering!) into the display filter specification window towards the top of the Wireshark window.

- 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.* Save 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 L
Not secure
gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html

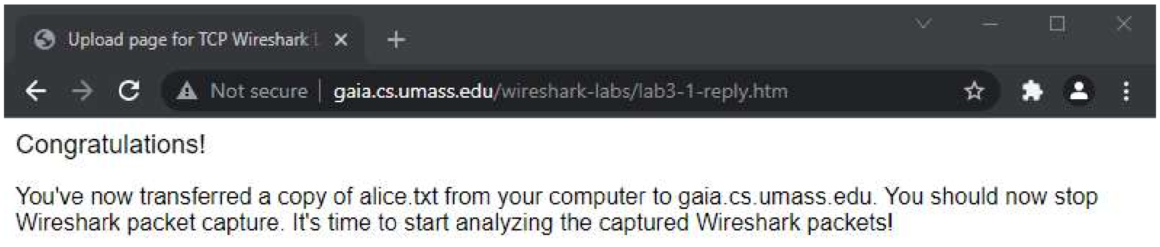
Upload page for TCP Wireshark Lab
Computer Networking: A Top Down Approach, 6th edition
Copyright 2012 J.F. Kurose and K.W. Ross, All Rights Reserved

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.

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!!

- Click on the *Choose File* button in this form to browse and select the name of the text file (full path name) on your computer containing *Alice in Wonderland*. **Do not 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.
- 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.


Upload page for TCP Wireshark L
Not secure
gaia.cs.umass.edu/wireshark-labs/lab3-1-reply.htm

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!

- Stop Wireshark packet capture.

You should see the initial three-way handshake containing a SYN message.

What to Hand in

PDF file report

Create a section called **Part 2 – TCP Transport Layer Protocol**.

Notes:

- Note: You must include a screenshot from Wireshark capture with annotation and sufficient surrounding details when answering any question relating to the captured information. Annotate the screenshot to indicate which information you are referring to in the screenshot. The screenshots should be legible and include enough surrounding details clearly showing where it belonged to.

1. From the packet that contains the initial three-way handshake SYN message, what is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it's probably easiest to enable HTTP protocol analysis, filter for only http, and select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details of the selected packet header window".
2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?
3. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?
4. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?
5. What is the sequence number of the TCP segment containing the HTTP POST command? Note that 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.
6. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection.

   Create a table to show the following data (from a to g below):

   a. What are the sequence numbers of the first six segments in the TCP connection (starting with the segment containing the HTTP POST)?

   b. At what time was each segment sent?

   c. When was the ACK for each segment received?

d. Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments?

e. What is the `EstimatedRTT` value (see Section 3.5.3, page 242 in textbook) after the receipt of each ACK? Assume that the value of the `EstimatedRTT` is equal to the measured RTT for the first segment, and then is computed using the `EstimatedRTT` equation on page 242 for all subsequent segments. Show your work details.

f. Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. 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 Graph->Round Trip Time Graph.* Include the round-trip Time Graph in your report.

g. What is the length of each of the first six TCP segments?

7. What is the minimum amount of available buffer space advertised at the received for the entire trace? Indicate which captured packet number in the trace. Does the lack of receiver buffer space ever throttle the sender? Explain.

8. Are there any retransmitted segments in the trace file? What did you check for (in the trace) to answer this question? Explain based on and include the Statistics/TCP Stream Graphs/Time Sequence (Stevens) graph.

9. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment?

10. What is the throughput (bytes transferred per unit time) for this TCP connection? Explain how you calculated this value.

11. From the *Time Sequence Graph (Stevens*) plotting showing the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Here, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents a series of packets that were sent back-to-back by the sender.

    a. Identify where TCP's Slow-Start phase begins and ends?

    b. Identify where Congestion Avoidance takes over?

    c. Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the textbook.

Submission zip file

include your Wireshark capture file **xx2.pcapng** in the zip file.

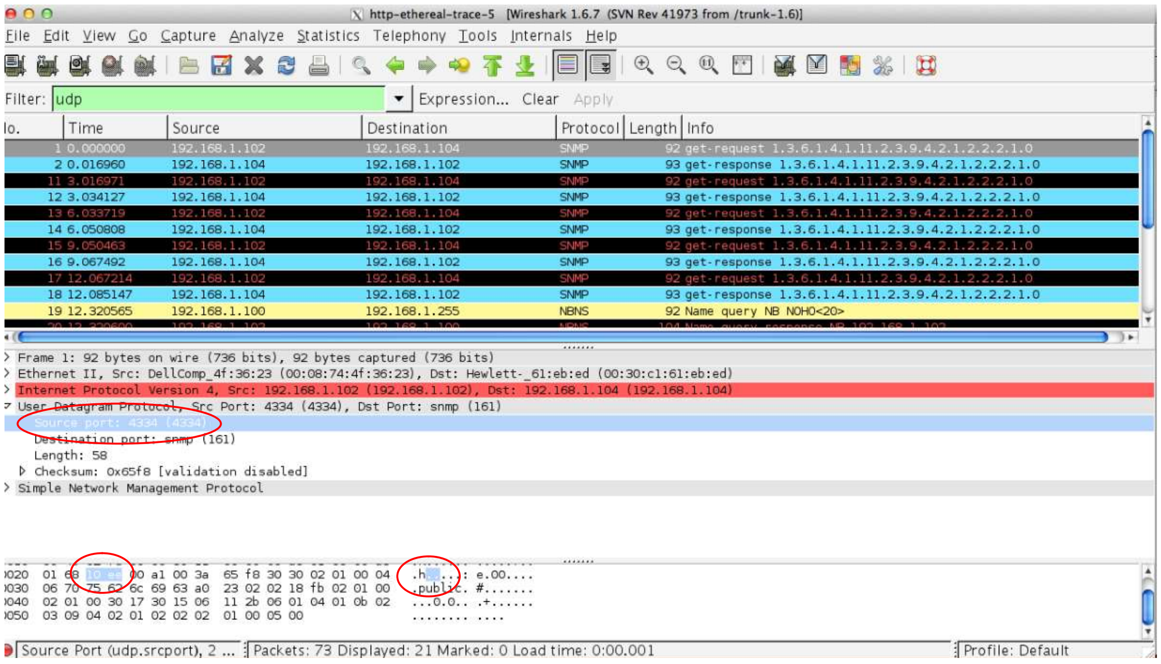
where xx = initials (the two characters representing the first character of your first and last name).

Appendix

Sample of Wireshark Screen Captures

For any answer containing information that was derived from a Wireshark captured you must justify it by providing a screenshot from that specific Wireshark capture with sufficient surrounding information, and with annotation/marking to indicate which portion of the capture your answer was based on.

An example of Wireshark screen capture with marking and annotation is shown below.


http-ethereal-trace-5 [Wireshark 1.6.7 (SVN Rev 41973 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: udp
Expression... Clear Apply

lo.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.102	192.168.1.104	SNMP	92	get-request 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.2.1.0
2	0.016960	192.168.1.104	192.168.1.102	SNMP	93	get-response 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.2.1.0
11	3.016971	192.168.1.102	192.168.1.104	SNMP	92	get-request 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.2.1.0
12	3.034127	192.168.1.104	192.168.1.102	SNMP	93	get-response 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.2.1.0
13	6.033719	192.168.1.102	192.168.1.104	SNMP	92	get-request 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.2.1.0
14	6.050808	192.168.1.104	192.168.1.102	SNMP	93	get-response 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.2.1.0
15	9.050463	192.168.1.102	192.168.1.104	SNMP	92	get-request 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.2.1.0
16	9.067492	192.168.1.104	192.168.1.102	SNMP	93	get-response 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.2.1.0
17	12.067214	192.168.1.102	192.168.1.104	SNMP	92	get-request 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.2.1.0
18	12.085147	192.168.1.104	192.168.1.102	SNMP	93	get-response 1.3.6.1.4.1.11.2.3.9.4.2.1.2.2.2.1.0
19	12.320565	192.168.1.100	192.168.1.255	NBNS	92	Name query NB NOHO<20>

```
> Frame 1: 92 bytes on wire (736 bits), 92 bytes captured (736 bits)
> Ethernet II, Src: DellComp_4f:36:23 (00:08:74:4f:36:23), Dst: Hewlett-_61:eb:ed (00:30:c1:61:eb:ed)
> Internet Protocol Version 4, Src: 192.168.1.102 (192.168.1.102), Dst: 192.168.1.104 (192.168.1.104)
▿ User Datagram Protocol, Src Port: 4334 (4334), Dst Port: snmp (161)
    Source port: 4334 (4334)
    Destination port: snmp (161)
    Length: 58
  ▷ Checksum: 0x65f8 [validation disabled]
> Simple Network Management Protocol
```

```
0020  01 68 10 ee 00 a1 00 3a  65 f8 30 30 02 01 00 04   .h.....: e.00....
0030  06 70 75 62 6c 69 63 a0  23 02 02 18 fb 02 01 00   .public. #.......
0040  02 01 00 30 17 30 15 06  11 2b 06 01 04 01 0b 02   ...0.0.. .+......
0050  03 09 04 02 01 02 02 02  01 00 05 00               ........ ....
```

```
Source Port (udp.srcport), 2 ...   Packets: 73 Displayed: 21 Marked: 0 Load time: 0:00.001   Profile: Default
```

In the above screenshot the port number is shown as a hexadecimal number (small lower left red circle) and in ASCII format (small lower right red circle) and is two bytes long.

TCP Round-Trip-Time Table

This table includes data to calculate TCP Estimated RTT.

Create and include this table in your report to show your Estimated RTT calculation work.

Example of the TCP RTT Table is shown below.

	Raw Seq#	Sent time (s)	ACK received time (s)	Sampled RTT (s)	Est. RTT (s)
Segment 1	3984255503	0.026477	0.053937	0.02746	0.02746
Segment 2	add raw seq#	0.041737	0.077294	0.035557	0.02847
Segment 3	add raw seq#	0.054026	0.124085	0.070059	
Segment 4	add raw seq#	0.05469	0.169118	0.11443	
Segment 5	add raw seq#	0.077405	0.217299	0.13989	
Segment 6	add raw seq#	0.078157	0.267802	0.18964	

For TCP we use the formula:

Estimated RTT = 0.875 * Estimated RTT + 0.125 * Sampled RTT

Described below is a calculation example for Estimated RTT values for the first two segments.

Segment 1:

- Sampled RTT = ACK received time – Sent time = 0.053937 sec – 0.026477 sec = 0.02746 sec
- Per instructions, for segment 1 we use Estimated RTT = sampled RTT = 0.02746 second.

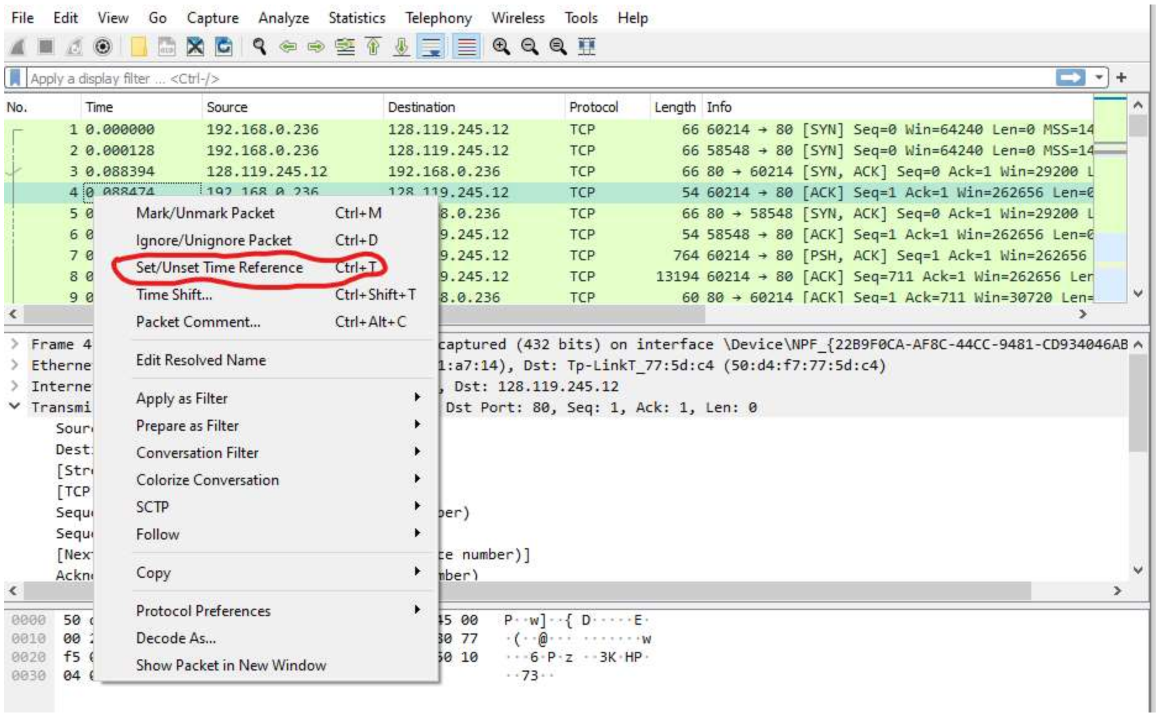
Segment 2:

- Sampled RTT = 0.077294 sec – 0.041737 sec = 0.035557 second
- Estimated RTT = 0.875 * 0.02746 + 0.125 * 0.035557 = 0.02847 second

Set/Unset Time Reference

Use Set/Unset Time Reference to let Wireshark automatically calculate time offset from any reference point. This is useful for extracting RTT data for TCP segments.

Select a frame of interest, then right click, and select Set/Unset Time Reference to mark that frame at time zero. Subsequent frame timing will be calculated using this zero-time referenced frame.

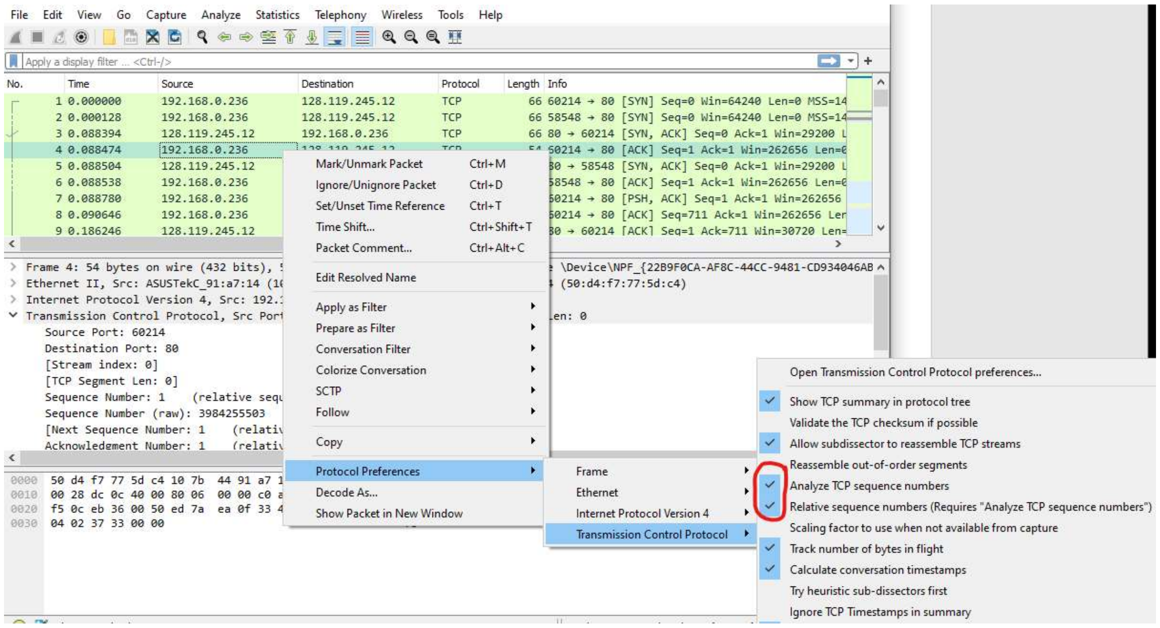


```
No.   Time       Source          Destination     Protocol  Length Info
    1 0.000000   192.168.0.236   128.119.245.12  TCP           66 60214 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=14
    2 0.000128   192.168.0.236   128.119.245.12  TCP           66 58548 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=14
    3 0.088394   128.119.245.12  192.168.0.236   TCP           66 80 → 60214 [SYN, ACK] Seq=0 Ack=1 Win=29200 L
    4 0.088474   192.168.0.236   128.119.245.12  TCP           54 60214 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
    5 0                          8.0.236         TCP           66 80 → 58548 [SYN, ACK] Seq=0 Ack=1 Win=29200 L
    6 0                          9.245.12        TCP           54 58548 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
    7 0                          9.245.12        TCP          764 60214 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262656
    8 0                          9.245.12        TCP        13194 60214 → 80 [ACK] Seq=711 Ack=1 Win=262656 Len
    9 0                          8.0.236         TCP           60 80 → 60214 [ACK] Seq=1 Ack=711 Win=30720 Len=

> Frame 4   captured (432 bits) on interface \Device\NPF_{22B9F0CA-AF8C-44CC-9481-CD934046AB
> Etherne   1:a7:14), Dst: Tp-LinkT_77:5d:c4 (50:d4:f7:77:5d:c4)
> Interne   , Dst: 128.119.245.12
v Transmi   Dst Port: 80, Seq: 1, Ack: 1, Len: 0
    Sour
    Dest
    [Str
    [TCP
    Sequ      ber)
    Sequ
    [Nex      e number)]
    Ackn      mber)

0000  50     45 00   P··w]··{ D·····E·
0010  00     30 77   ·(··@···  ·······w
0020  f5     50 10   ···6·P·z ··3K·HP·
0030  04             ··73··
```

TCP Relative Sequence Numbers

Wireshark has a nice feature to facilitate tracking of TCP segments by using relative sequence numbers instead of raw sequence numbers.

Select a frame of interest, then right click, and follow the submenu selections as illustrated below to select both “Analyze TCP sequence numbers” and “Relative sequence numbers” options.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time        Source           Destination      Protocol  Length Info
  1 0.000000    192.168.0.236    128.119.245.12   TCP        66 60214 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=14
  2 0.000128    192.168.0.236    128.119.245.12   TCP        66 58548 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=14
  3 0.088394    128.119.245.12   192.168.0.236    TCP        66 80 → 60214 [SYN, ACK] Seq=0 Ack=1 Win=29200 L
  4 0.088474    192.168.0.236                                 60214 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
  5 0.088504    128.119.245.12                                80 → 58548 [SYN, ACK] Seq=0 Ack=1 Win=29200 L
  6 0.088538    192.168.0.236                                 58548 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
  7 0.088780    192.168.0.236                                 60214 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262656
  8 0.090646    192.168.0.236                                 60214 → 80 [ACK] Seq=711 Ack=1 Win=262656 Len
  9 0.186246    128.119.245.12                                80 → 60214 [ACK] Seq=1 Ack=711 Win=30720 Len=
```

```
> Frame 4: 54 bytes on wire (432 bits),            \Device\NPF_{22B9F0CA-AF8C-44CC-9481-CD934046AB
> Ethernet II, Src: ASUSTekC_91:a7:14 (1           (50:d4:f7:77:5d:c4)
> Internet Protocol Version 4, Src: 192.
v Transmission Control Protocol, Src Port          Len: 0
    Source Port: 60214
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequ
    Sequence Number (raw): 3984255503
    [Next Sequence Number: 1    (relativ
    Acknowledgment Number: 1    (relativ
```

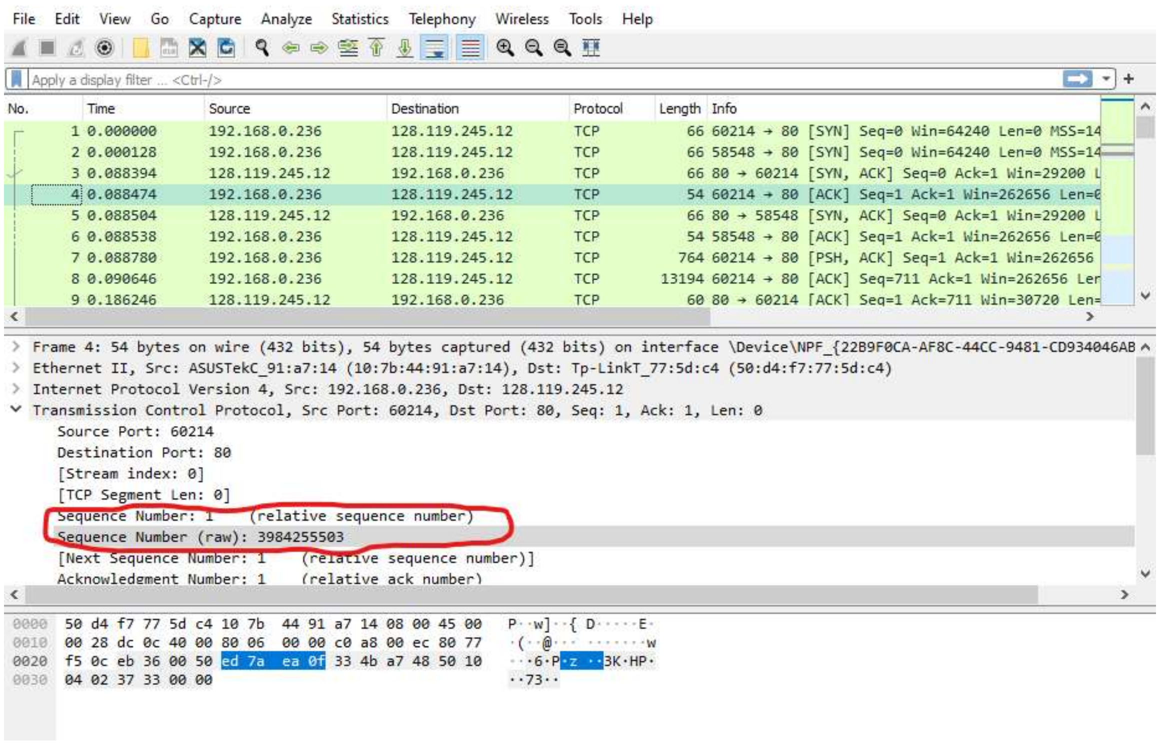
```
0000  50 d4 f7 77 5d c4 10 7b  44 91 a7 1
0010  00 28 dc 0c 40 00 80 06  00 00 c0 a
0020  f5 0c eb 36 00 50 ed 7a  ea 0f 33 4
0030  04 02 37 33 00 00
```

Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift... Ctrl+Shift+T
Packet Comment... Ctrl+Alt+C
Edit Resolved Name
Apply as Filter
Prepare as Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
Copy
Protocol Preferences
Decode As...
Show Packet in New Window
Frame
Ethernet
Internet Protocol Version 4
Transmission Control Protocol

Open Transmission Control Protocol preferences...

- ✓ Show TCP summary in protocol tree
- Validate the TCP checksum if possible
- ✓ Allow subdissector to reassemble TCP streams
- Reassemble out-of-order segments
- ✓ Analyze TCP sequence numbers
- ✓ Relative sequence numbers (Requires "Analyze TCP sequence numbers")
- Scaling factor to use when not available from capture
- ✓ Track number of bytes in flight
- ✓ Calculate conversation timestamps
- Try heuristic sub-dissectors first
- Ignore TCP Timestamps in summary

Wireshark TCP Relative and Raw Sequence numbers



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time       Source          Destination     Protocol  Length Info
   1 0.000000   192.168.0.236   128.119.245.12  TCP          66 60214 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=14
   2 0.000128   192.168.0.236   128.119.245.12  TCP          66 58548 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=14
   3 0.088394   128.119.245.12  192.168.0.236   TCP          66 80 → 60214 [SYN, ACK] Seq=0 Ack=1 Win=29200 L
   4 0.088474   192.168.0.236   128.119.245.12  TCP          54 60214 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
   5 0.088504   128.119.245.12  192.168.0.236   TCP          66 80 → 58548 [SYN, ACK] Seq=0 Ack=1 Win=29200 L
   6 0.088538   192.168.0.236   128.119.245.12  TCP          54 58548 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
   7 0.088780   192.168.0.236   128.119.245.12  TCP         764 60214 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262656
   8 0.090646   192.168.0.236   128.119.245.12  TCP       13194 60214 → 80 [ACK] Seq=711 Ack=1 Win=262656 Len
   9 0.186246   128.119.245.12  192.168.0.236   TCP          60 80 → 60214 [ACK] Seq=1 Ack=711 Win=30720 Len=
```

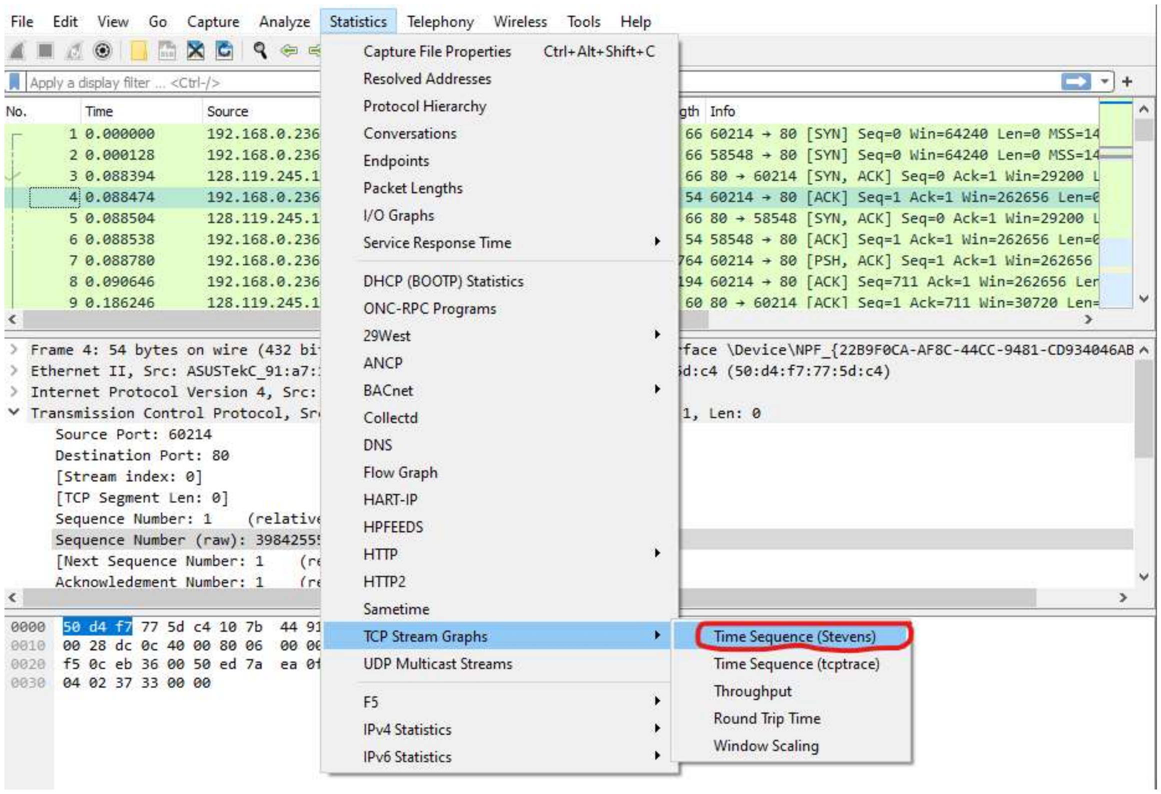
```
> Frame 4: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{22B9F0CA-AF8C-44CC-9481-CD934046AB
> Ethernet II, Src: ASUSTekC_91:a7:14 (10:7b:44:91:a7:14), Dst: Tp-LinkT_77:5d:c4 (50:d4:f7:77:5d:c4)
> Internet Protocol Version 4, Src: 192.168.0.236, Dst: 128.119.245.12
v Transmission Control Protocol, Src Port: 60214, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
     Source Port: 60214
     Destination Port: 80
     [Stream index: 0]
     [TCP Segment Len: 0]
     Sequence Number: 1    (relative sequence number)
     Sequence Number (raw): 3984255503
     [Next Sequence Number: 1    (relative sequence number)]
     Acknowledgement Number: 1    (relative ack number)
```

```
0000  50 d4 f7 77 5d c4 10 7b  44 91 a7 14 08 00 45 00   P··w]··{ D·····E·
0010  00 28 dc 0c 40 00 80 06  00 00 c0 a8 00 ec 80 77   ·(··@··· ·······w
0020  f5 0c eb 36 00 50 ed 7a  ea 0f 33 4b a7 48 50 10   ···6·P·z ··3K·HP·
0030  04 02 37 33 00 00                                  ··73··
```

Time Sequence Graph (Stevens)

Wireshark provides a TCP graphing utility called Time-Sequence-Graph(Stevens) which can be used to plot out data sent per unit time.

Refer to the illustration below to generate the TCP Time-Sequence plot.

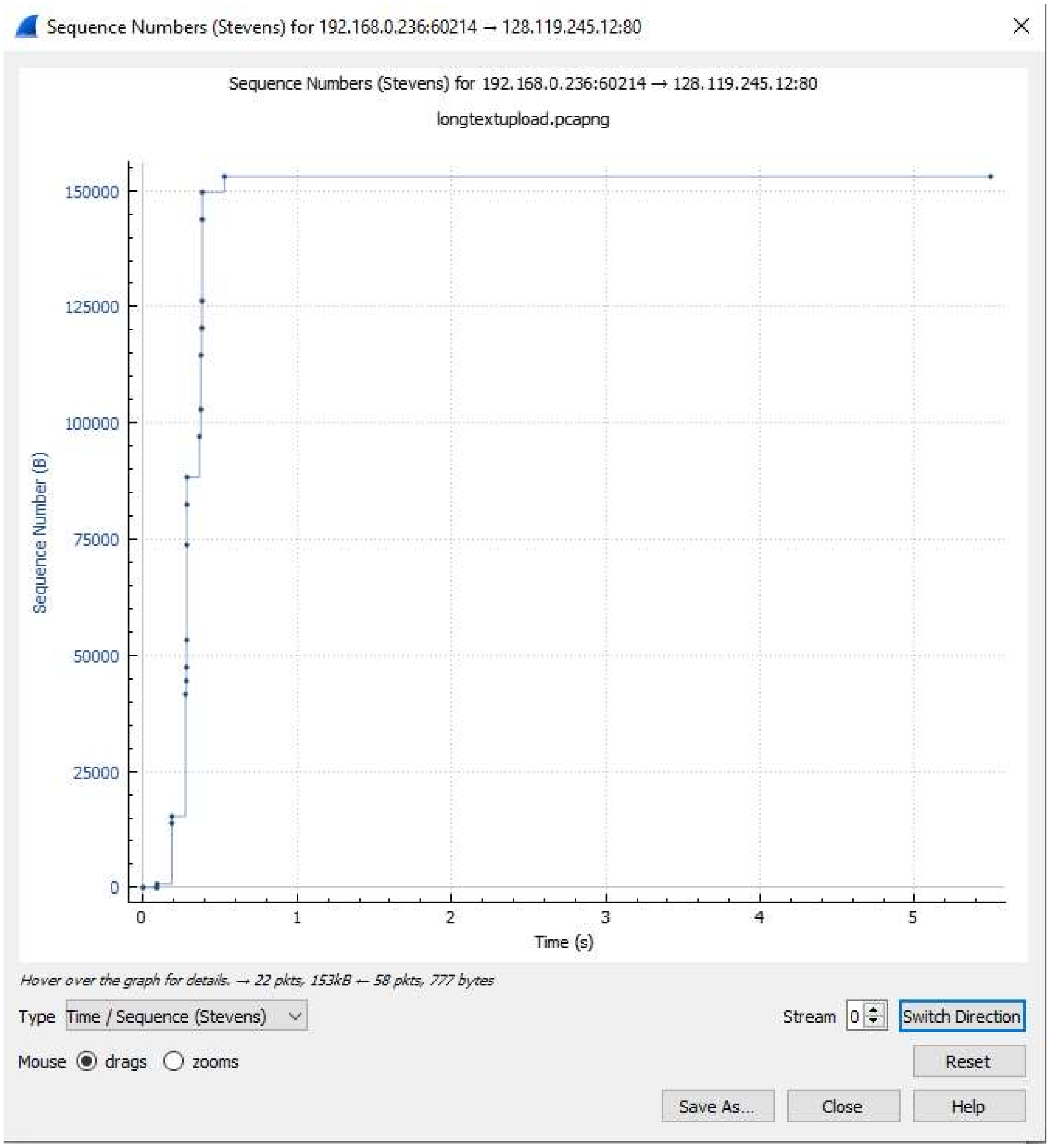


```
No.     Time        Source             gth  Info
  1 0.000000    192.168.0.236           66 60214 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=14
  2 0.000128    192.168.0.236           66 58548 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=14
  3 0.088394    128.119.245.1           66 80 → 60214 [SYN, ACK] Seq=0 Ack=1 Win=29200 L
  4 0.088474    192.168.0.236           54 60214 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
  5 0.088504    128.119.245.1           66 80 → 58548 [SYN, ACK] Seq=0 Ack=1 Win=29200 L
  6 0.088538    192.168.0.236           54 58548 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
  7 0.088780    192.168.0.236          764 60214 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262656
  8 0.090646    192.168.0.236          194 60214 → 80 [ACK] Seq=711 Ack=1 Win=262656 Len
  9 0.186246    128.119.245.1           60 80 → 60214 [ACK] Seq=1 Ack=711 Win=30720 Len=
```

```
> Frame 4: 54 bytes on wire (432 bi     face \Device\NPF_{22B9F0CA-AF8C-44CC-9481-CD934046AB
> Ethernet II, Src: ASUSTekC_91:a7::    d:c4 (50:d4:f7:77:5d:c4)
> Internet Protocol Version 4, Src:
v Transmission Control Protocol, Sr     1, Len: 0
    Source Port: 60214
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative
    Sequence Number (raw): 3984255
    [Next Sequence Number: 1    (re
    Acknowledgement Number: 1    (re
```

```
0000  50 d4 f7 77 5d c4 10 7b  44 91
0010  00 28 dc 0c 40 00 80 06  00 0e
0020  f5 0c eb 36 00 50 ed 7a  ea 01
0030  04 02 37 33 00 00
```

Example of Time Sequence Graph (Stevens)


Sequence Numbers (Stevens) for 192.168.0.236:60214 → 128.119.245.12:80

Sequence Numbers (Stevens) for 192.168.0.236:60214 → 128.119.245.12:80

longtextupload.pcapng

Hover over the graph for details. → 22 pkts, 153kB ← 58 pkts, 777 bytes

Type: Time / Sequence (Stevens)

Stream: 0

Mouse: [✓ drags] zooms