



HACETTEPE UNIVERSITY

DEPARTMENT OF  
COMPUTER ENGINEERING

---

## BBM453: Computer Networks Laboratory Lab 6: TCP

---

*Author*

Kayla AKYÜZ  
21726914  
Batuhan ÖZTÜRK  
21827742

*Advisors*

T.A. Tuğba ERDOĞAN  
[tugba.gurgen@hacettepe.edu.tr](mailto:tugba.gurgen@hacettepe.edu.tr)  
Assoc. Prof. Sevil ŞEN  
[ssen@cs.hacettepe.edu.tr](mailto:ssen@cs.hacettepe.edu.tr)

Group 14  
Source IP : 192.168.0.27

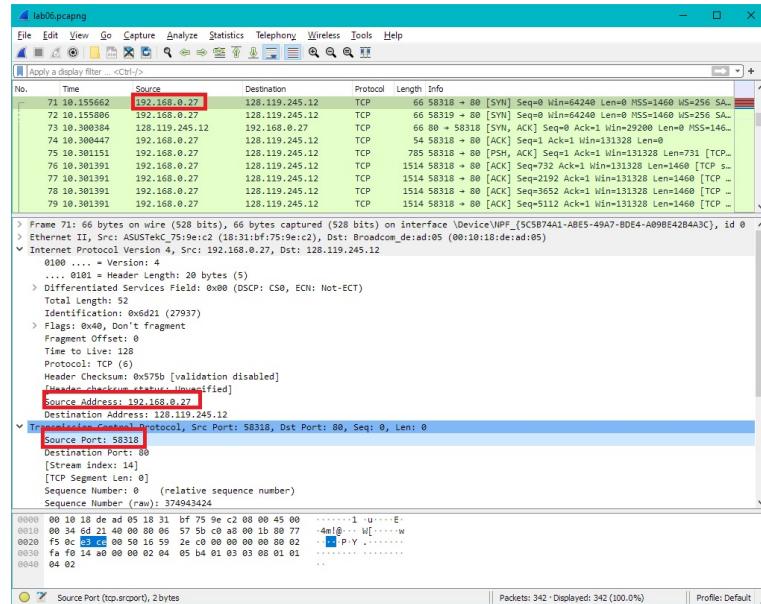
Nov 3,2021

# SOLUTIONS

## A first look at the captured trace

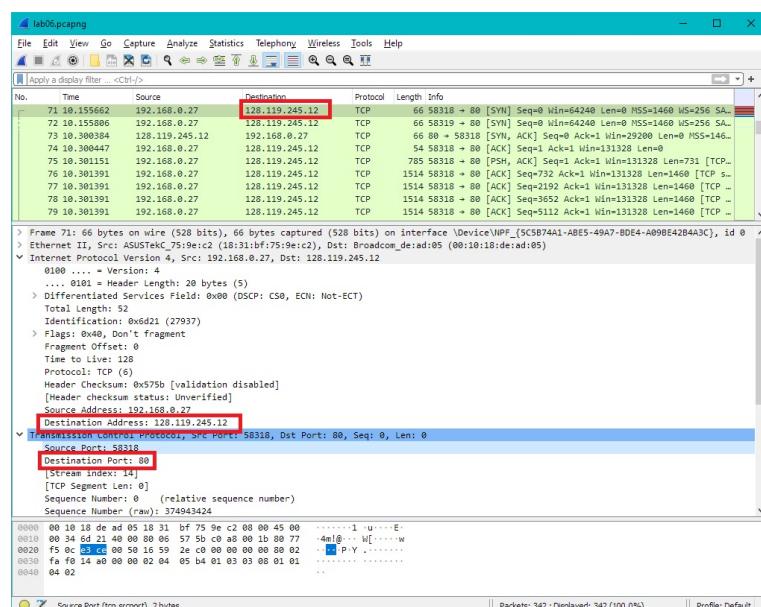
### 1.What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?

Client computer IP address is: 192.168.0.27 , Port 58318



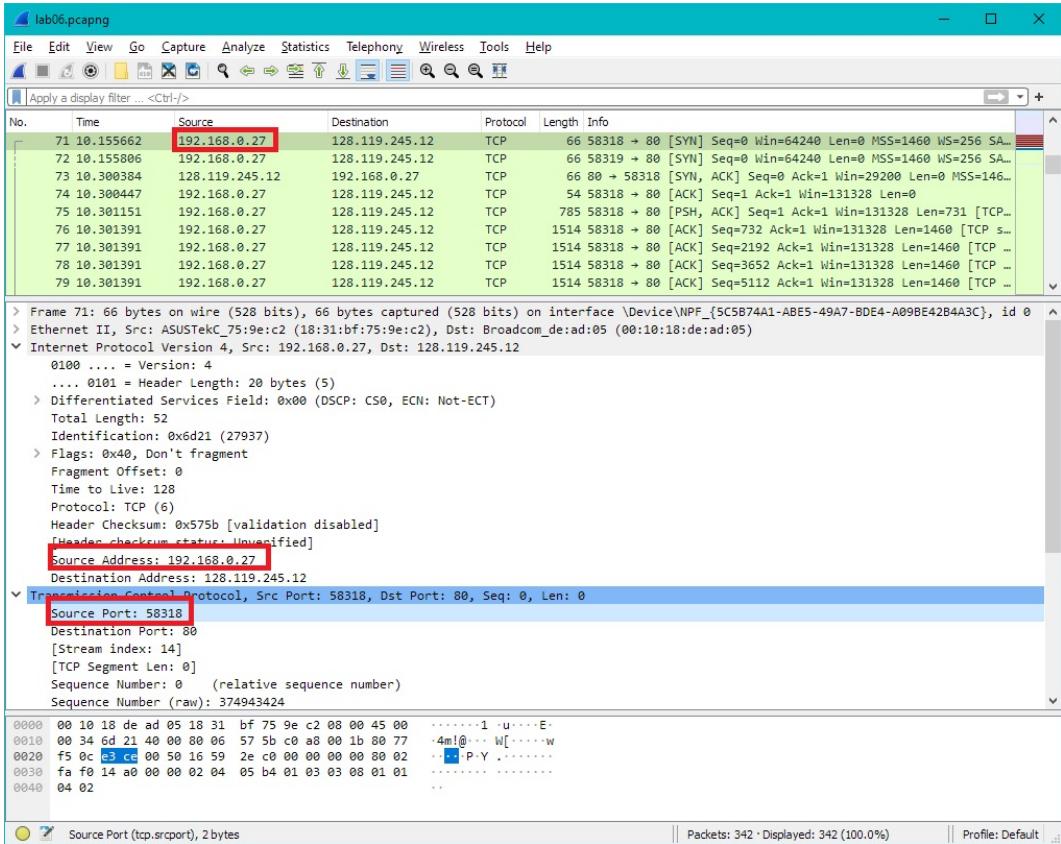
### 2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

IP address of gaia.cs.umass.edu is: 128.119.245.12 , used port is: 80



### 3. What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.edu?

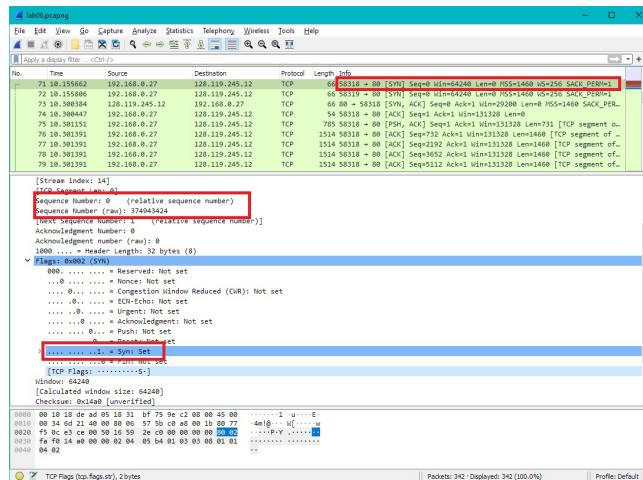
Client computer IP address is: 192.168.0.27 , Port 58318.



# TCP Basics

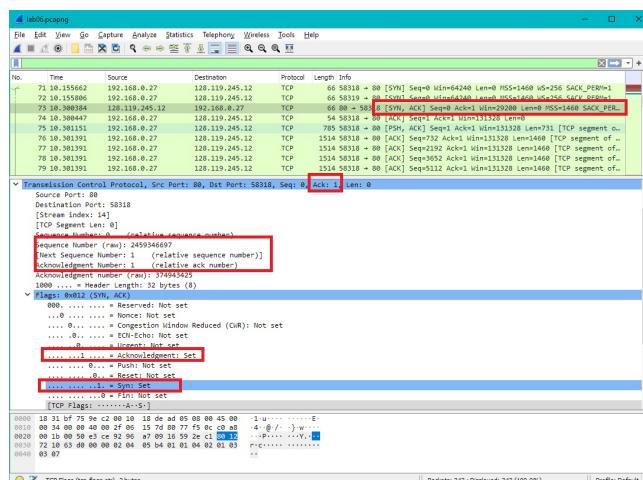
4. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

Sequence number is 0. We can see the SYN flag is set to 1 inside the flags.



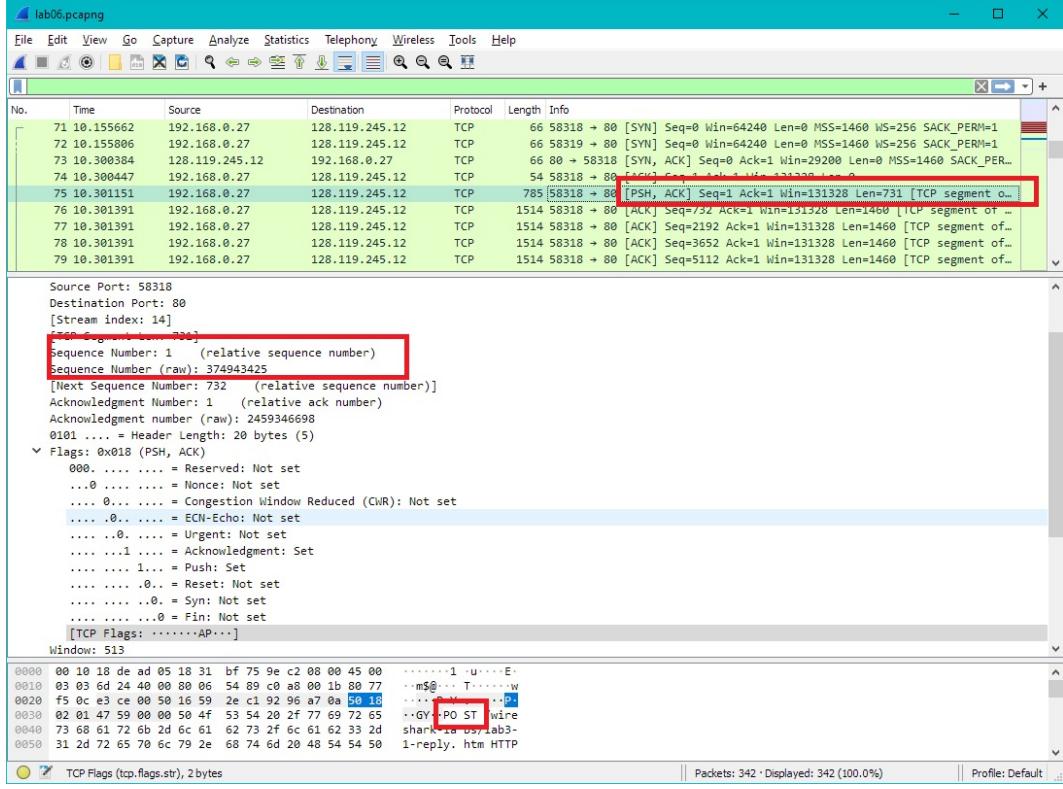
5. What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

Sequence number is 0, Acknowledgement number is 1. It is determined by incrementing sequence number send by client by one. We can see Acknowledgement flag and syn flag are set to 1.



## 6. What is the sequence number of the TCP segment containing the HTTP POST command?

Sequence number of the TCP segment containing the HTTP POST command is 1.

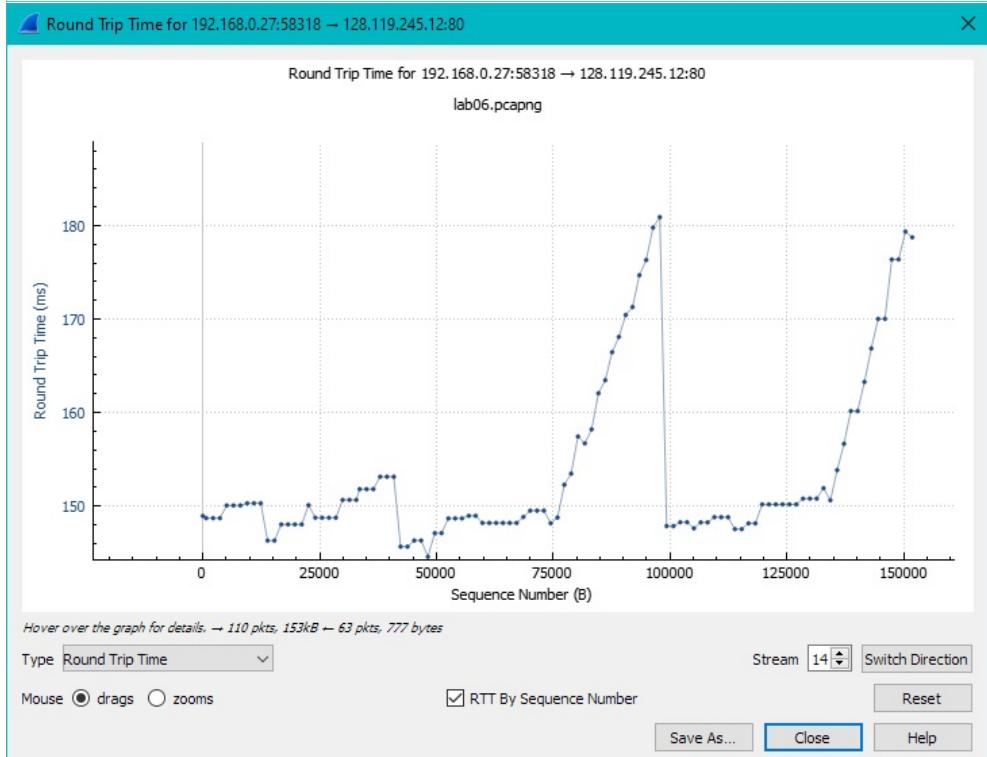
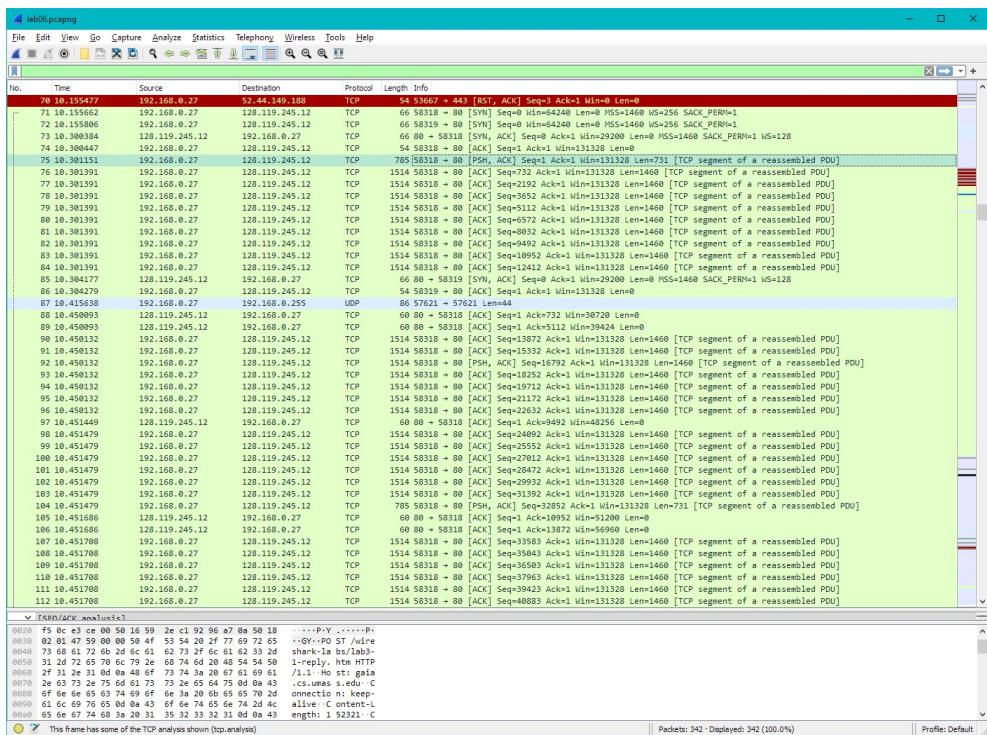


## 7. What are the sequence numbers of the first six segments in the TCP connection? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the EstimatedRTT value after the receipt of each ACK?

In the below tables you can see our calculated values. One weird situation we encountered is that our send TCP's are send at the same time as well as most ACK TCP's are received at the same time. Also the whole process is done way quicker than the shared example documents. We chose to provide calculations and data on our test even tho it is weird numbers however we also examined the TA's test results for better understanding of the subject.

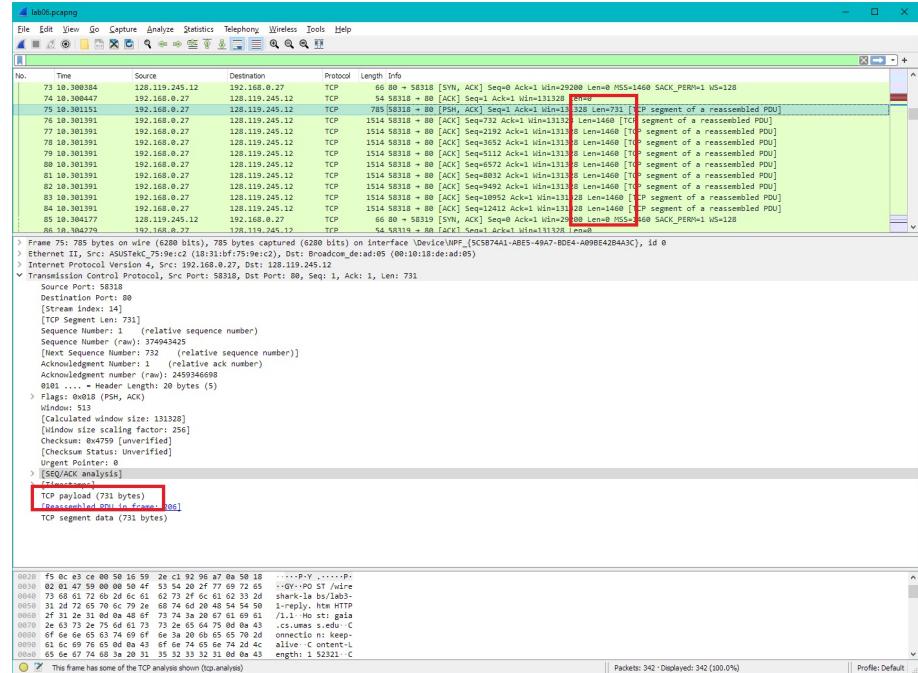
Segment	Sequence Number	Sent Time	ACK Received Time	RTT (seconds)
1	1	10.301151	10.450093	0.148942
2	732	10.301391	10.450093	0.148702
3	2192	10.301391	10.450093	0.148702
4	3652	10.301391	10.450093	0.148702
5	5112	10.301391	10.450093	0.148702
6	6572	10.301391	10.450132	0.148741

Segment	Sequence Number	Estimated RTT
1	1	First EstimatedRTT = Segment 1 RTT = 0.148942
2	732	$0.875 * 0.148942 + 0.125 * 0.148702 = 0.148912$
3	2192	$0.875 * 0.148912 + 0.125 * 0.148702 = 0.14888575$
4	3652	$0.875 * 0.14888575 + 0.125 * 0.148702 = 0.14886278125$
5	5112	$0.875 * 0.14886278125 + 0.125 * 0.148702 = 0.148842683594$
6	6572	$0.875 * 0.148842683594 + 0.125 * 0.148741 = 0.148829973145$



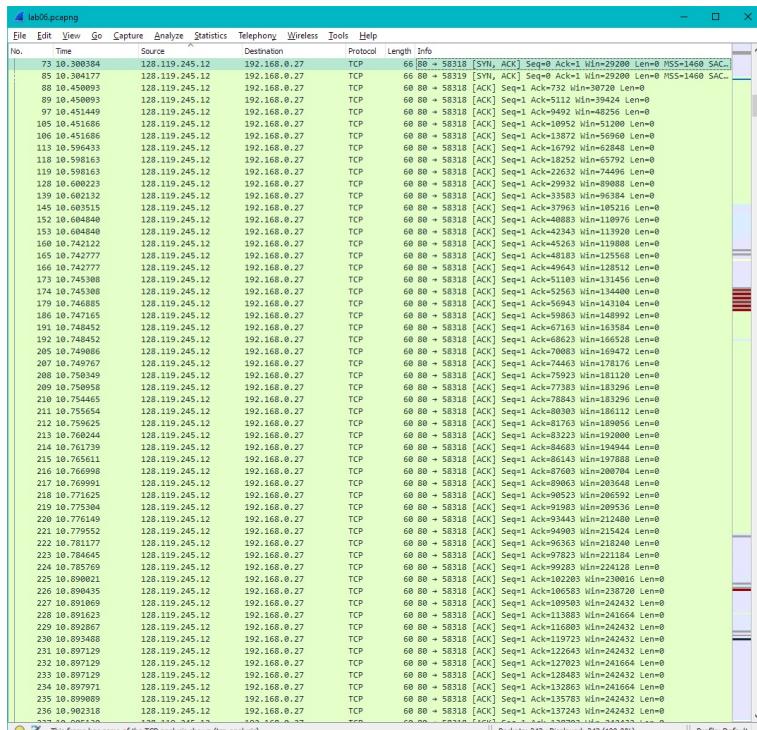
## 8. What is the length of each of the first six TCP segments?

As we can see in the screenshot, the first one which contains POST message is 731 bytes, other segments has 1460 bytes lengths.



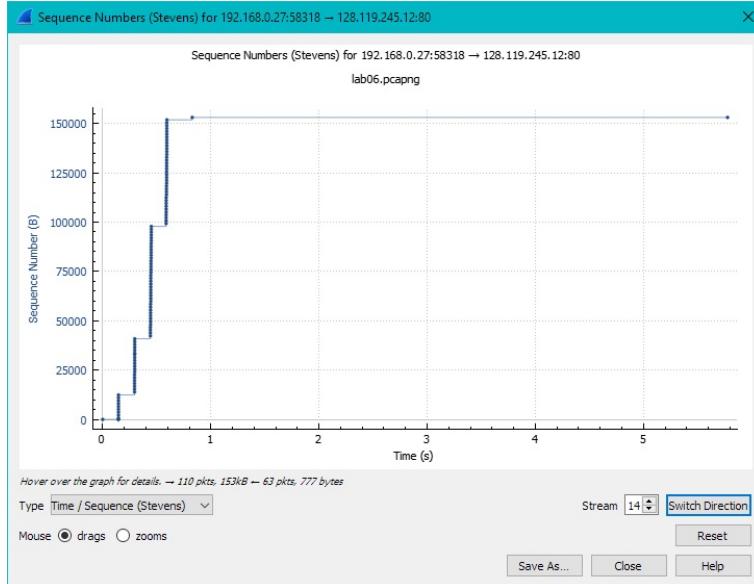
## 9. What is the minimum amount of available buffer space (receiver window) advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender? If any, justify your reason

Our buffer space starts from 29200 and grows until 242432 which seems to be the max. There is not a lack of receiver buffer ever.



## 10. Are there any re-transmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?

Check below screen shot for our time sequence graph (stevens). Since all sequence numbers are monotonically increasing there is no re-transmitted segments in the trace file.



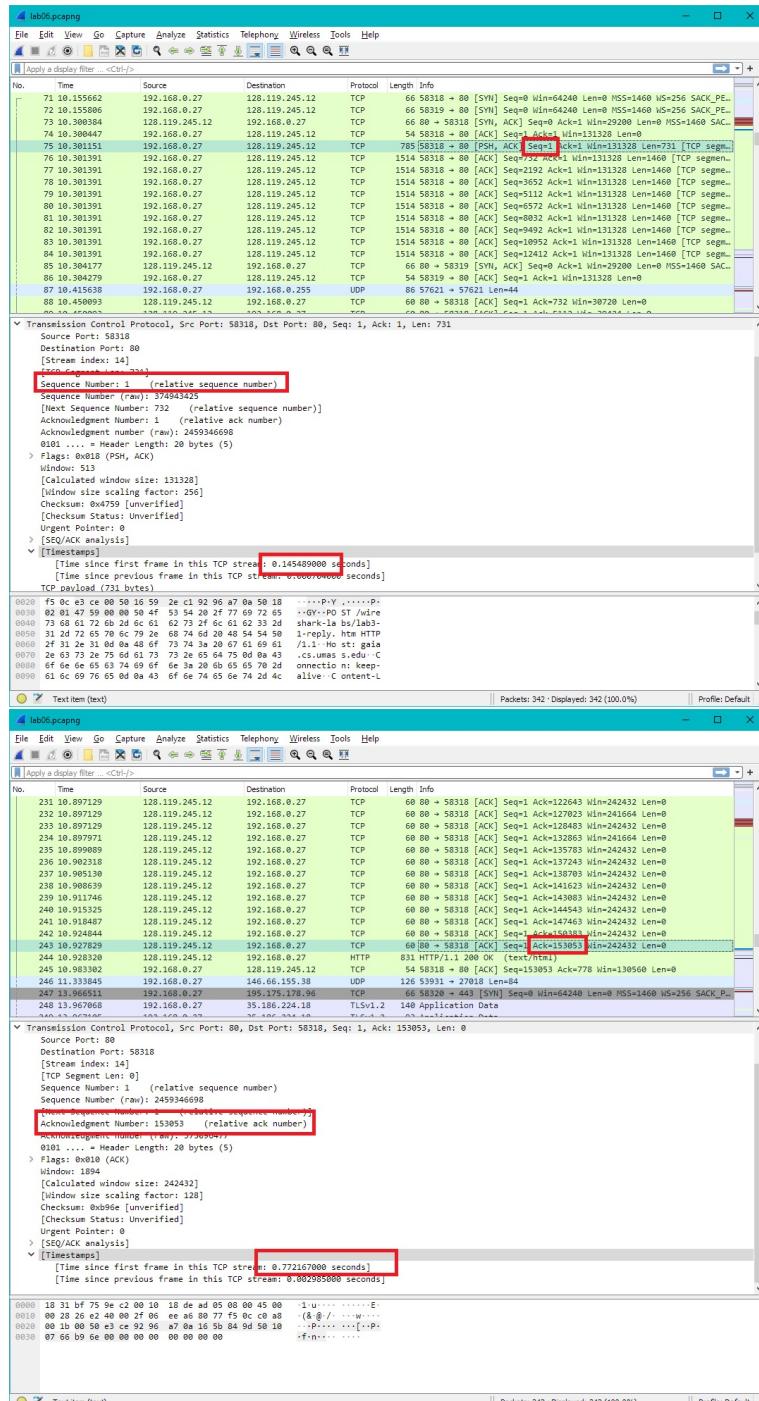
## 11. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 247 in the text)?

Typically it acknowledges 4380 which is triple send data. Meaning it acknowledges in triplets. However there are rare cases where it acknowledges more than that or sometimes less.

No.	Time	Source	Destination	Protocol	Length	Info
73	10.308384	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [SYN, ACK] Seq=0 Ack=1 Win=29280 Len=0 MSS=1468 SAC=1
85	10.304177	128.119.245.12	192.168.0.27	TCP	66	[89 → 58319] [SYN, ACK] Seq=0 Ack=1 Win=29290 Len=0 MSS=1468 SAC=1
88	10.450093	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=732 Win=39729 Len=0
89	10.450099	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=5112 Win=39424 Len=0
97	10.451444	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=4992 Win=48256 Len=0
105	10.451686	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=4992 Win=51200 Len=0
113	10.595456	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=4992 Win=51200 Len=0
118	10.598163	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=16792 Win=62348 Len=0
119	10.598163	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=18292 Win=65792 Len=0
128	10.600224	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=29932 Win=89088 Len=0
139	10.600232	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=30032 Win=89384 Len=0
140	10.753915	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=29883 Win=134408 Len=0
152	10.600480	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=40883 Win=118976 Len=0
153	10.600480	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=42343 Win=113920 Len=0
169	10.742122	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=45262 Win=119880 Len=0
165	10.742777	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=48183 Win=125568 Len=0
166	10.742777	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=49643 Win=128512 Len=0
173	10.742778	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=49643 Win=128512 Len=0
174	10.745908	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=52353 Win=134408 Len=0
179	10.746805	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=56943 Win=143104 Len=0
186	10.747165	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=59863 Win=148992 Len=0
191	10.748452	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=67163 Win=163584 Len=0
192	10.748452	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=68623 Win=165528 Len=0
205	10.749088	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=70883 Win=169472 Len=0
206	10.749089	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=70883 Win=169472 Len=0
208	10.750849	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=75023 Win=161128 Len=0
209	10.750958	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=77326 Win=183206 Len=0
210	10.750958	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=77843 Win=183296 Len=0
211	10.755625	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=80834 Win=186112 Len=0
212	10.759625	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=91764 Win=189956 Len=0
213	10.760244	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=93223 Win=192098 Len=0
214	10.760245	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=93223 Win=192098 Len=0
215	10.765611	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=81643 Win=197888 Len=0
216	10.765991	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=76903 Win=208704 Len=0
217	10.769991	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=90963 Win=203648 Len=0
218	10.771625	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=90523 Win=206592 Len=0
219	10.775386	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=91964 Win=209536 Len=0
220	10.776149	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=92148 Win=212488 Len=0
221	10.776150	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=94089 Win=212488 Len=0
222	10.781177	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=916893 Win=212348 Len=0
224	10.784645	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=97823 Win=221184 Len=0
225	10.809021	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=102203 Win=230016 Len=0
226	10.809435	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=106583 Win=238720 Len=0
227	10.809436	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=106583 Win=238720 Len=0
229	10.893233	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=1124564 Win=244564 Len=0
229	10.893267	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=116893 Win=242432 Len=0
229	10.893498	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=119723 Win=242432 Len=0
231	10.897129	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=122643 Win=242432 Len=0
232	10.897129	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=127923 Win=241664 Len=0
233	10.897129	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=128483 Win=242432 Len=0
234	10.897972	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=132883 Win=241664 Len=0
234	10.897972	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=132883 Win=241664 Len=0
236	10.982218	128.119.245.12	192.168.0.27	TCP	66	[89 → 58318] [ACK] Seq=1 Ack=137243 Win=242432 Len=0

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

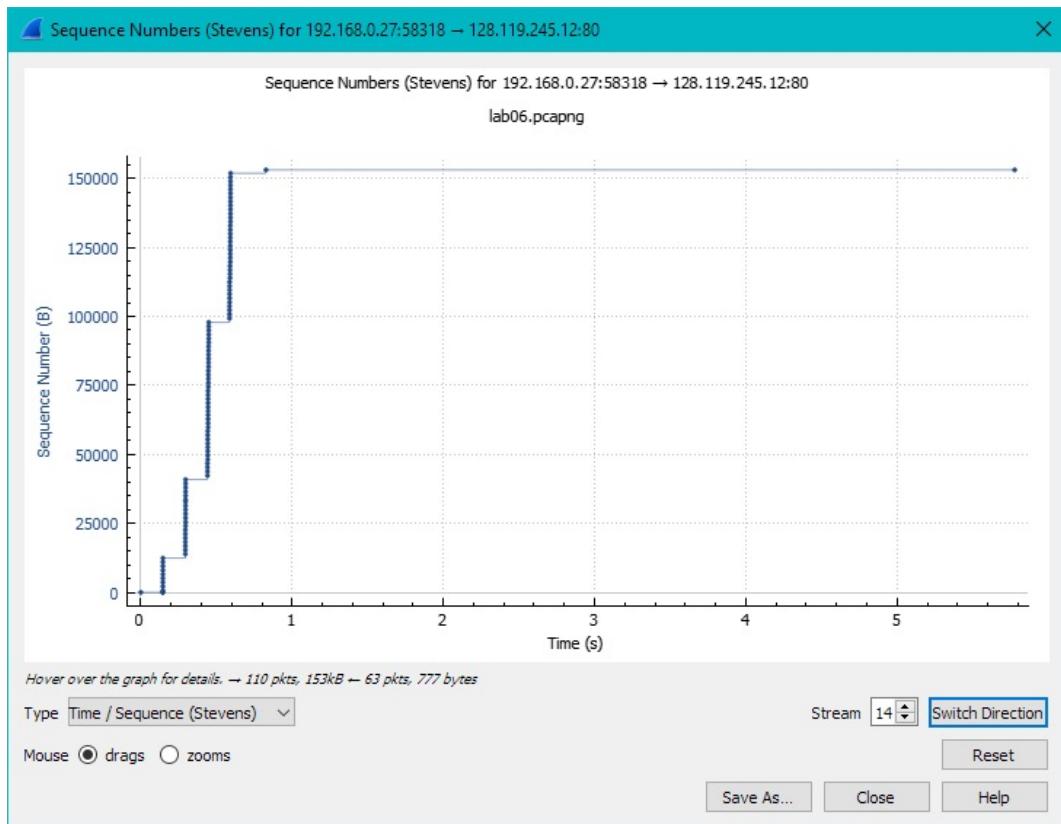
We get the size of the data of TCP connection by subtracting first segment's sequence from the last ACK's number. Last ACK's number is 153053 and first segment's sequence number is 1. So our total data throughput is  $153053 - 1 = 153052$ . We get the total time of the connection similarly looking at the time instant of the first segment and last ACK and subtracting them. Last ACK's time instant is 0.772167 second and first segment's time instant is 0.145489 seconds. So total time of connection is  $0.772167 - 0.145489 = 0.626678$ . Dividing the total amount of data to total transmission time  $153052 / 0.626678 = 244227.498013$ , we get 244.23 KByte/sec.



## TCP congestion control in action

13. Use the Time-Sequence-Graph(Stevens) plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. Can you identify where TCP's slowstart phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text.

We identify slow start by looking at the screenshot below as starting at 0.14 and lasting until 0.3 which takes around 0.15 seconds. After that rate increases greatly until congestion avoidance which seems to be around 0.58 because we observe that the data sent is cut down. This is similar to idealized behaviour we studied.



14. Answer each of two questions above for the trace that you have gathered when you transferred a file from your computer to gaia.cs.umass.edu.

We put screenshots for all the questions and all of them are from our test results.

## **REFERENCES**

- LaTex Tutorials
- Assignment Paper
- Wireshark FAQ
- Wireshark User's Guide
- Finding Throughput With Wireshark