Exercise 1:

1. IP address of gaia.cs.umass.edu: 128.119.245.12 , Port: 80

Source IP address: 192.168.1.102, Port: 1161

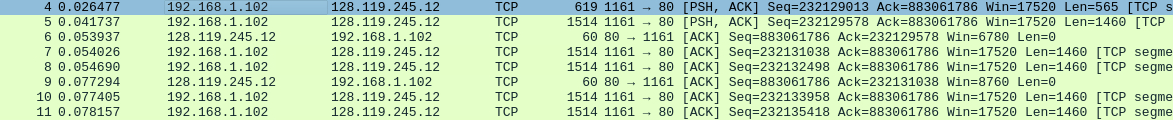


1. Seqno:232129013

3.

EstimatedRTT = (1-a)\*EstimatedRTT + a\*SampleRTT

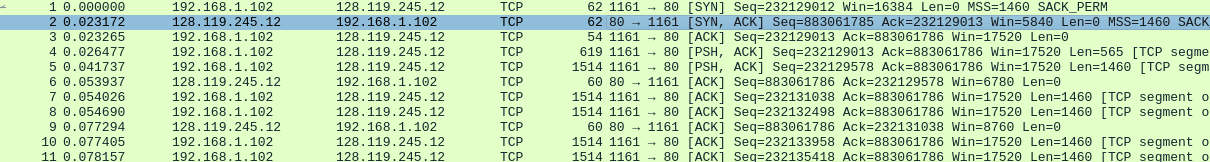
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| seqno | Time sent | Ack received | RTT | estimatedRTT | length |
| 232129013 | 0.026477 | 0.053937 | 0.02746 | 0.02746 | 565 |
| 232129578 | 0.041737 | 0.077294 | 0.035557 | 0.028472 | 1460 |
| 232131038 | 0.054026 | 0.124085 | 0.070059 | 0.033670 | 1460 |
| 232132498 | 0.054690 | 0.169118 | 0.114428 | 0.043765 | 1460 |
| 232133958 | 0.077405 | 0.217299 | 0.139894 | 0.055781 | 1460 |
| 232135418 | 0.078157 | 0.267802 | 0.189645 | 0.072514 | 1460 |



4.

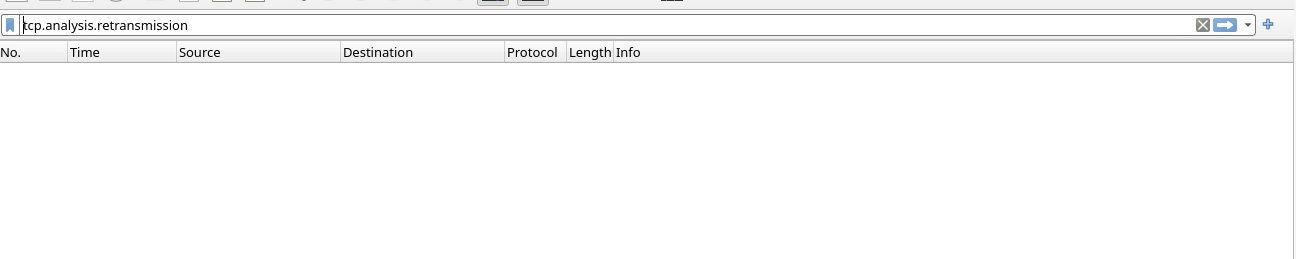
minimum amount of available buffer:5840

No, because it’s increased after 5840



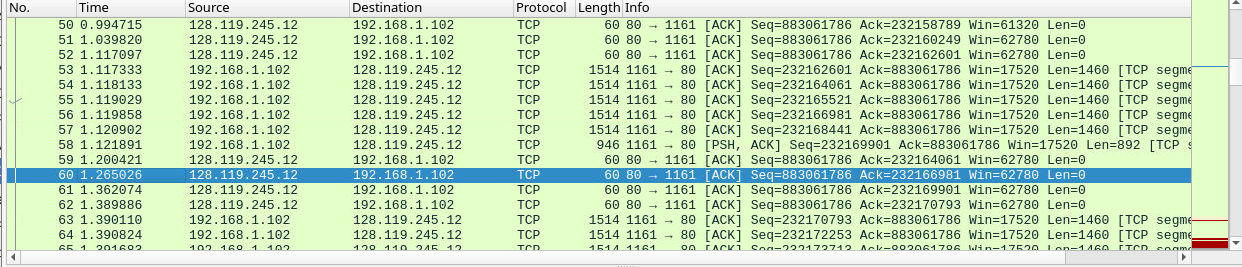
1. No retransmission segment

I check tcp.analysis.retransmission in filter

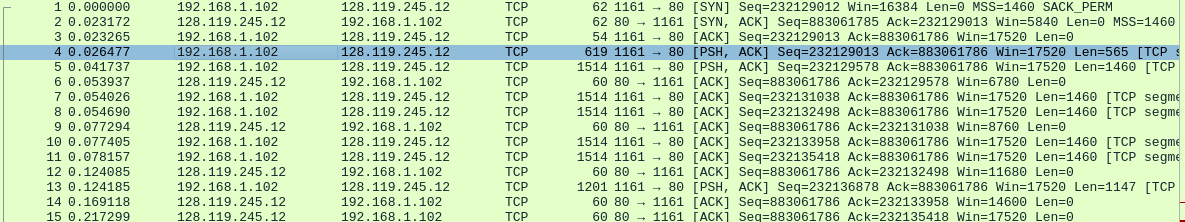


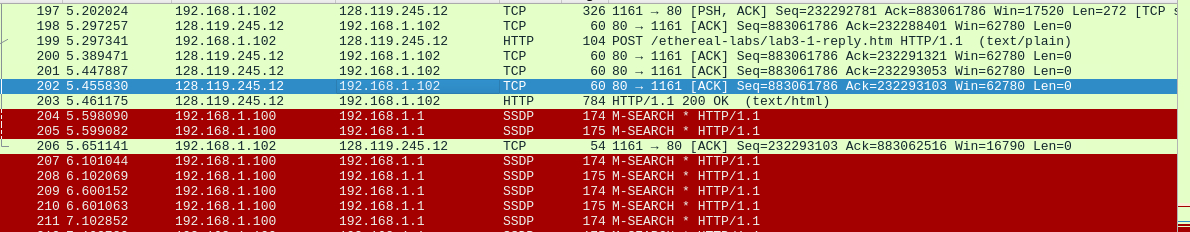
1. 1460

At first receiver will ack each packet and at #60 it ack two segments seq=232165521 and seq=232166981. and after #60 there are many cases it ack two segments.



1. Throughout = Total amount data / Total transmission time



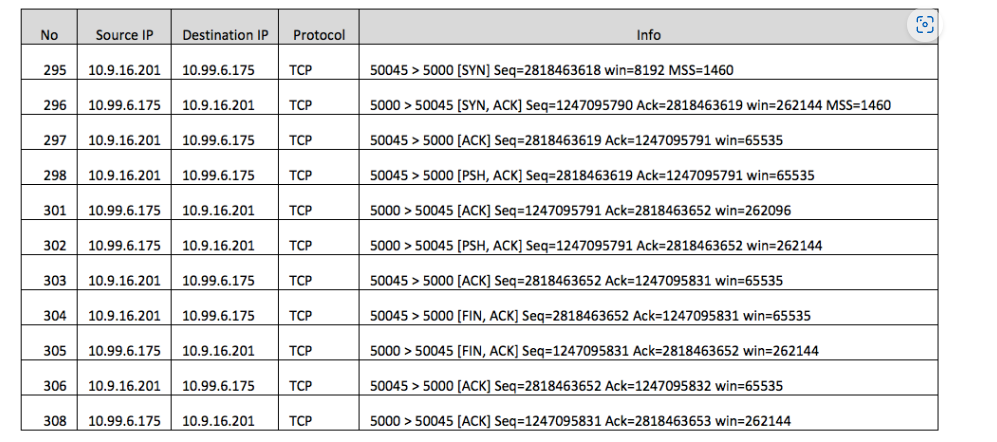


Total amount data = 232293103 - 232129013 = 164090 bytes

Total transmission time = 5.455830 - 0.026477 = 5.429353 seconds

Throughout = 164090 / 5.429353 = 30.223 KByte/s

Exercise 2:



1.2818463618

2.1247095790 2818463619 ensure the seqno from client is seqno + 1

3.2818463619 1247095791 No data contained

4. Both client and server do the acitve close. Because they both send FIN to each other and receive the ack which ACK is previous seqno +1. then close the connection. simultaneous close.

5. client->server = 2818463652 - 2818463619 = 33 bytes

server->client = 1247095831 - 1247095791 = 40 bytes

Final ack - initial seqno - 2 = total transferred data bytes

Like 2818463653 - 2818463618 - 2 = 33 bytes = client->server

This is because the SYN, FIN segments will +1 for ack. And for data segments the ack num is depends on the previous data amount, the ack num is always equal to the previous number + previous data amount.