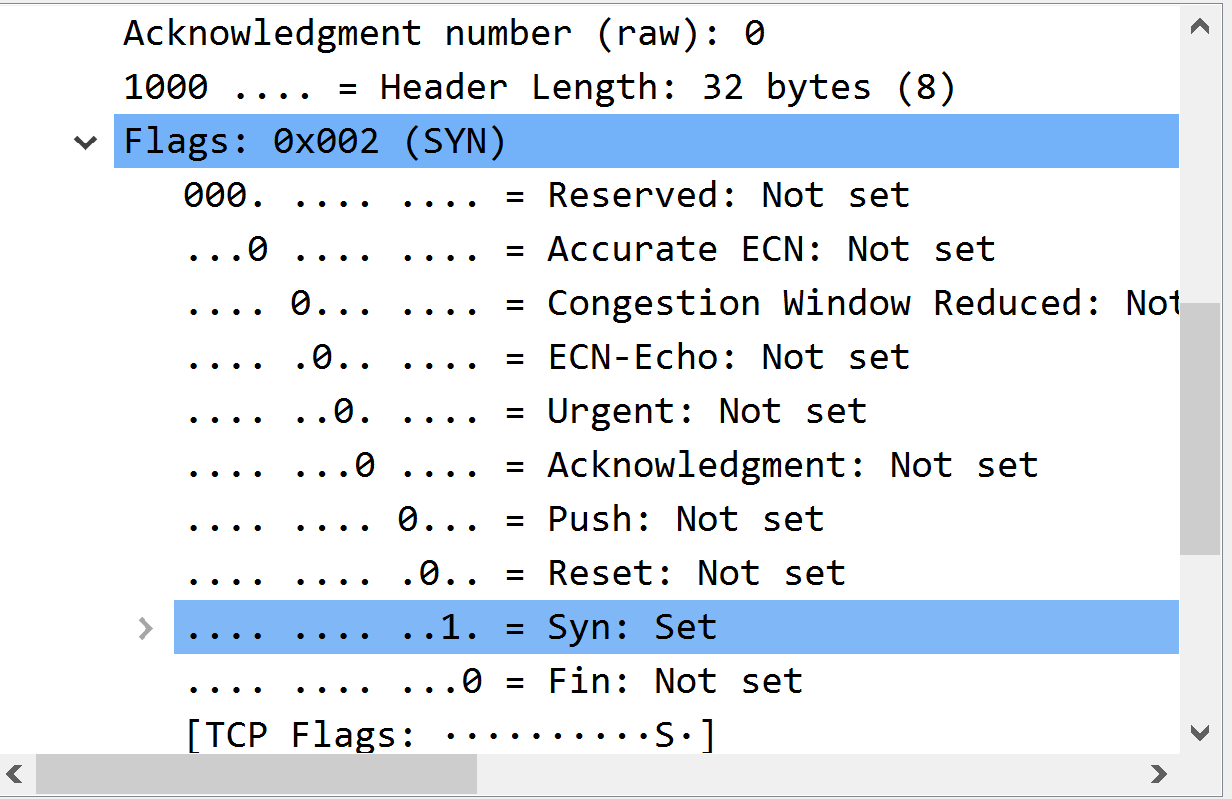
KTHYUS001 – Yusuf Kathrada

Assignment 2

Manual Marked Section

**3(b). What is it in this TCP segment that identifies the segment as a SYN segment?**

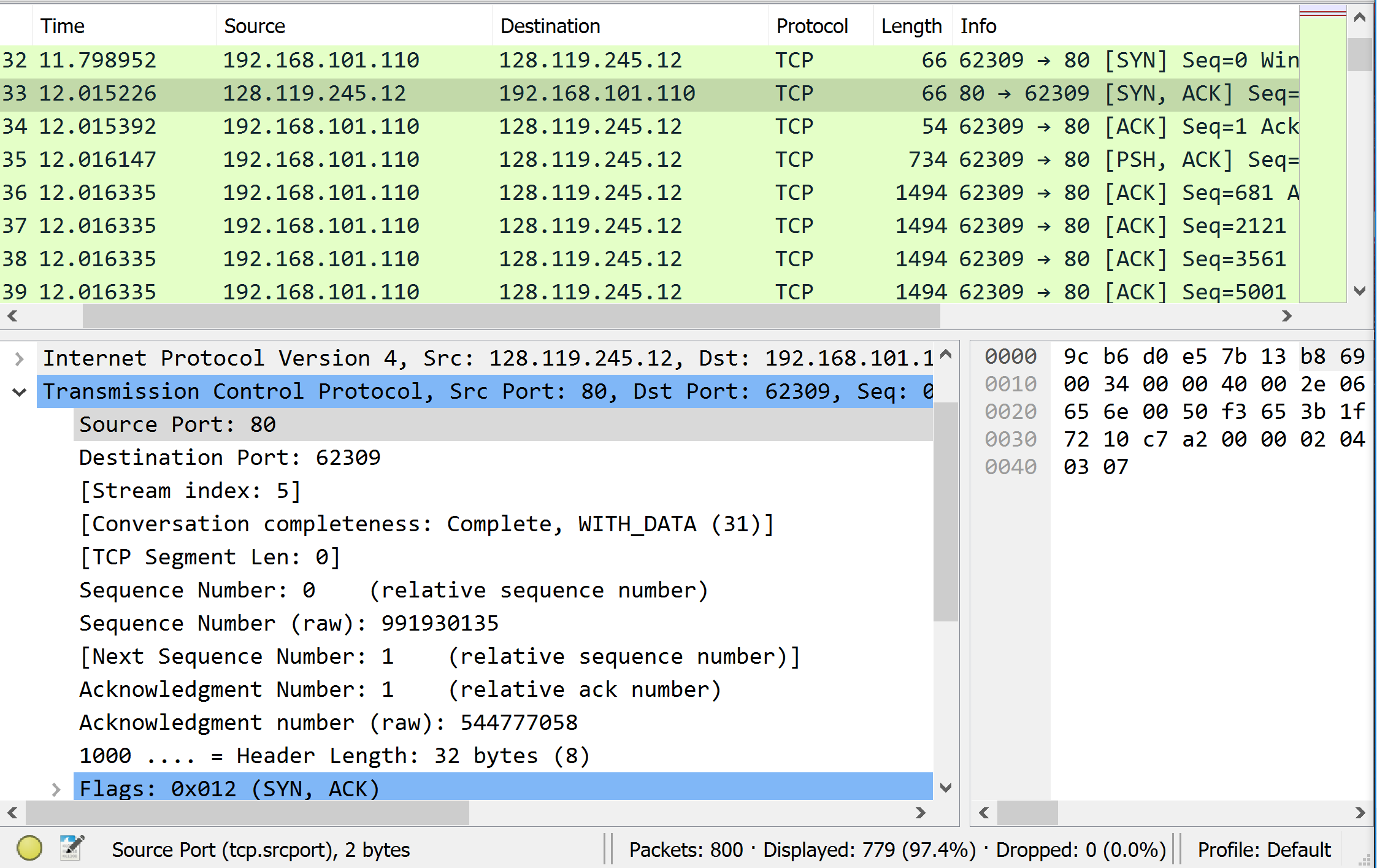
Graphical user interface, application

Description automatically generated

The arrows in the images above indicate that in this TCP segment there is a flag which identifies this TCP segment as a TCP SYN segment. This is further confirmed by the Syn: Set to 1 (…. …. ..1. = Syn: Set).

**4(b). What is it in the segment that identifies the segment as a SYNACK segment?**

Text, table

Description automatically generated

The arrows in the images above indicate that in this TCP segment there is flag which identifies this TCP segment as a TCP SYNACK segment. This is further confirmed by the Syn: Set to 1 (…. …. ..1. = Syn: Set) and the Acknowledgement: Set to 1 (…. …1 …. = Acknowledgement: Set).

**4(d). What is it in the segment that identifies the segment as a SYNACK segment?**

The acknowledgement number (544777058) is 1 more than the sequence number of the TCP SYN segment (544777057) which suggests that the packet was successfully delivered and that the acknowledgment number represents the next byte it is expecting.

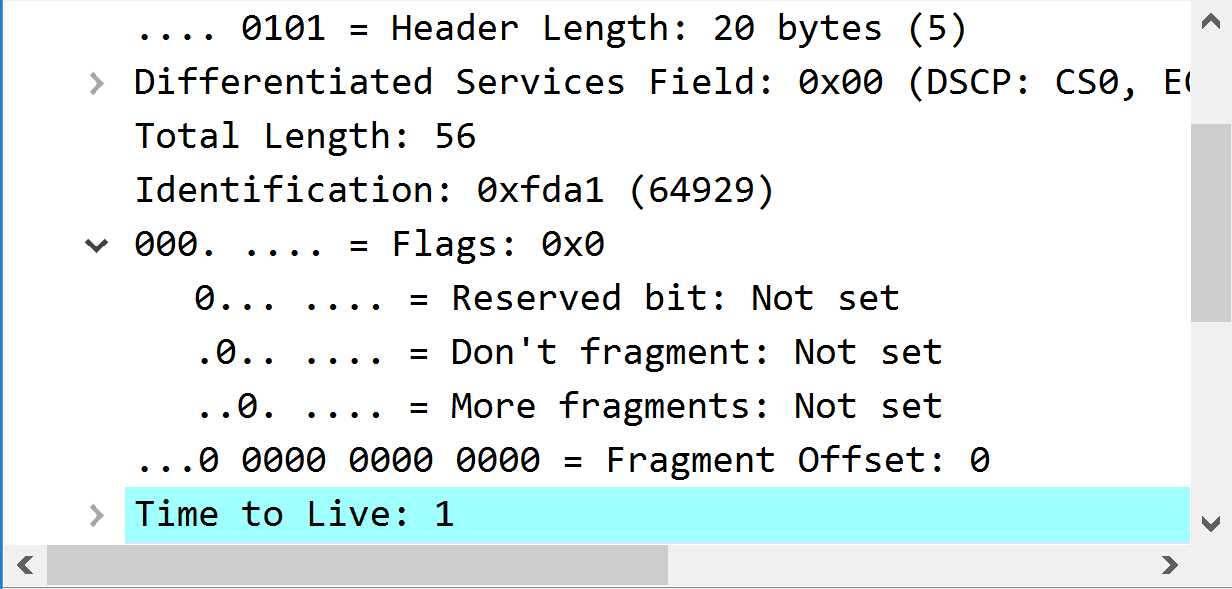
**8(b). Does the lack of receiver buffer space ever throttle the sender for these first four data-carrying segments?**

No, the sender is not throttled as the first four data-carrying segments all fall within the available buffer space.

**9. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?**

No, this was checked by looking for any repeating sequence numbers and none were found.

**15. Explain how you determined whether or not the datagram has been fragmented.**



The More fragments is equal to 0 (..0. … = More fragments: Not set).

**16. Which fields in the IP datagram always change from one datagram to the next within this series of UDP segments sent by your computer destined to 128.119.245.12, via traceroute?**

The fields which always change from one datagram to the next is the Identification, Time to Live and Header Checksum.

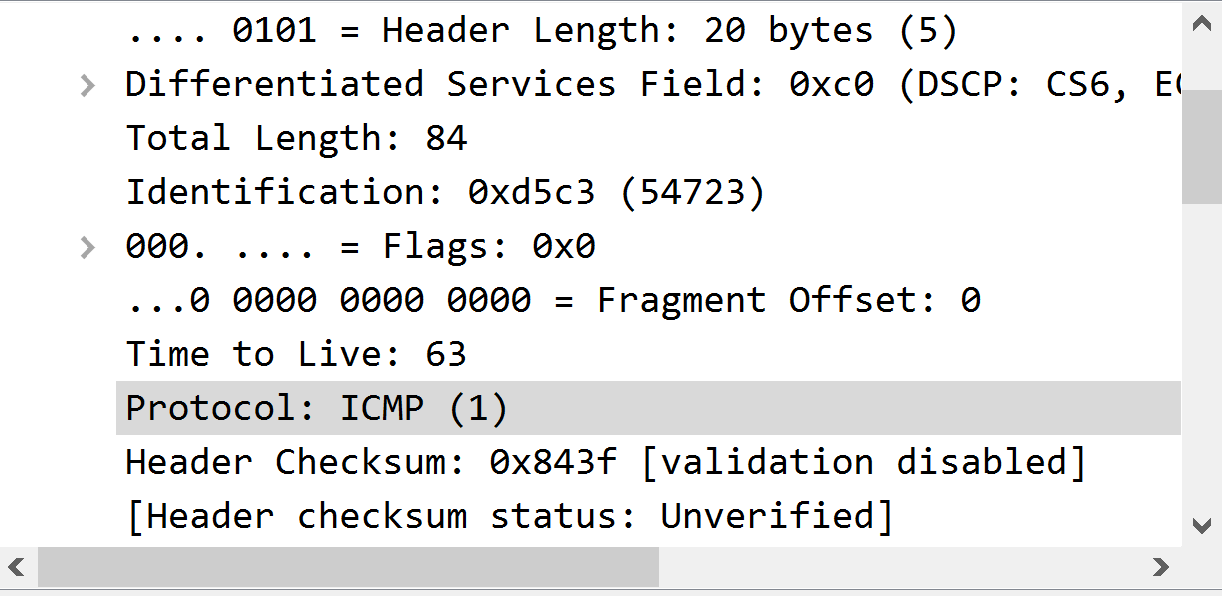
**17. Which fields in this sequence of IP datagrams (containing UDP segments) stay constant? Why?**

* Version, all versions are equal to 4 as we are using IPv4 across all datagrams.
* Source IP, the source from which we send does not change and thus it stays the same.
* Destination IP, we are sending to same source.
* Header Length, this is due to the fact that across all datagrams they are all ICMP packets.
* Differentiated Services, all the packets are the same (ICMP) and thus they all require the same services.
* Upper Layer Protocol, this is also due to the fact that all the packets are ICMP.

**18. Describe the pattern you see in the values in the Identification field of the IP datagrams being sent by your computer.**

The identification field whose value is in hexadecimal (and decimal in brackets) increments each time there is a ICMP request.

**19. What is the upper layer protocol specified in the IP datagrams returned from the routers?**



The upper layer protocol is ICMP.

**20. Are the values in the Identification fields (across the sequence of all of ICMP packets from all of the routers) similar in behaviour to your answer to question 18 above?**

No, they are not incrementing in constant amounts with the identification of one packet to the next varying substantially.

**21. Are the values of the TTL fields similar, across all of ICMP packets from all of the routers?**

Yes, the TTL fields across all ICMP packets are all 1.

**22. Devise an experiment that will falsify, or validate the hypothesis “The network is slower during higher stages of loadshedding (stage > 4) than lower ones ( stage < 4)”. That is, write down a procedure about what needs to be done to investigate this to obtain a conclusion ‘yes this does happen’ or ‘no, the performance is the same’. This procedure should be written in such a way that it can be given to a classmate and they’d be able to carry out the experiment you devised.**

Hypothesis:

The network is slower during higher stages of loadshedding (stage > 4) than the lower ones (stage < 4)

Experiment:

1. Evaluate when and what stage loadshedding your area receives on the day. Loadshedding can be unpredictable at times so by using an app such as ESP (Eskom Se Push) you are able to view the stages and schedule for the day.
2. Pick a time in which you have no loadshedding (This will be your control test), a time you have stage 5 or higher and a time you have between stage 1 and 4. You will run the next steps for each of the before mentioned periods.
3. Start your web browser and visit: <http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html>.
4. Select the browse button on the form and select a file to upload. This same file must be used for all tests and should preferably be larger than 150k bytes.
5. Start Wireshark and start packet capture.
6. Return to the browser and select the “Upload <filename>” button. You should receive a congratulations message once it has successfully been uploaded.
7. Stop the Wireshark packet capture.
8. Calculate the RTT and tabulate your findings. Repeat steps 3-8 for each period 5 times (i.e., a total of 5x3 = 15 tests) in order to get more conclusive evidence.
9. Calculate the Estimated RTT (using α = 0.125) for each period.
10. From these calculations, draw your conclusions based on whether there are noteworthy differences in the RTT found in the three periods.

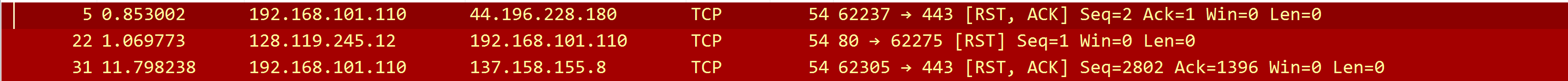
**23. Did your network have problems when uploading the Gutenberg file? Try the filters from the table below and report whether there was anything out of the ordinary. For instance, applying the tcp.analysis.flags may show a “TCP Dup ACK”, a duplicate ACK received, tcp.analysis.ack\_rtt > 0.1 may show packets as well (look at the bottom-right for number of packets and percentage of slow RTT packets).**

Yes there were problems when uploading the files.

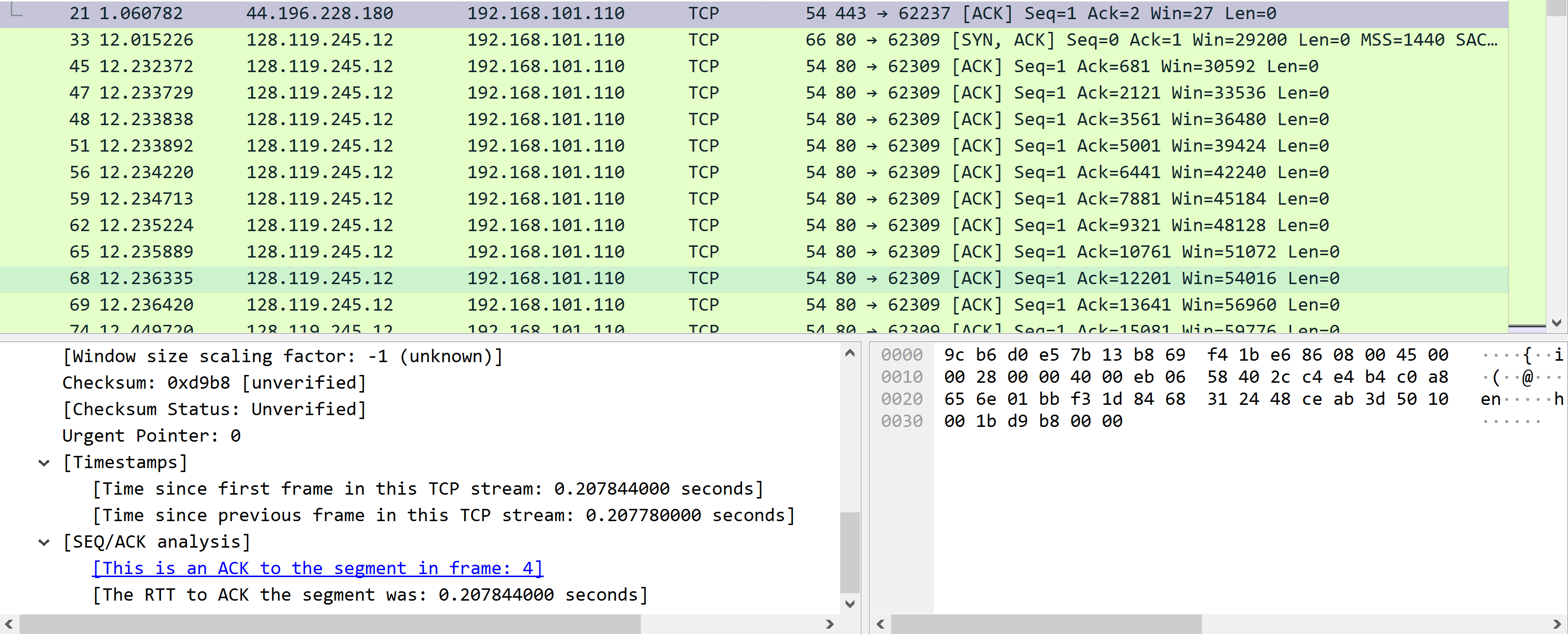
Graphical user interface, text, application

Description automatically generated

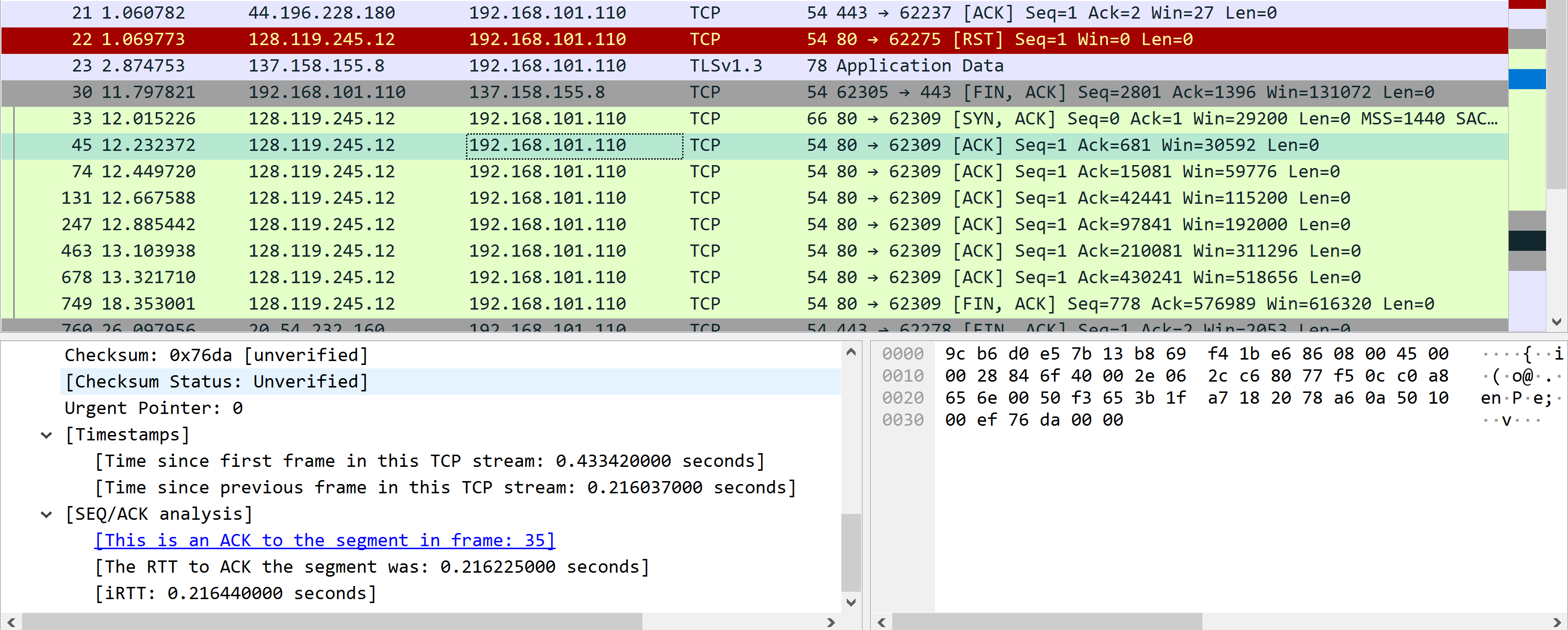
The above image shows that there was a duplicate ACK received. Filter: tcp.analysis.flags && !tcp.analysis.window\_update



The above image shows a network failure or timeout that happened mid conversation that caused connection reset. Filter: tcp.flags.reset==1



The above image shows slow RTT’s which there was many. This may be a sigtn of congestion as it climbs through the trace. Filter: tcp.analysis.ack\_rtt > 0.1



The above image shows that there were instances were the server was too busy to respond. Filter: tcp.time\_delta > 0.1

There were no evidence that suggests that there was slow DNS times present. Filter: dns.time > 1