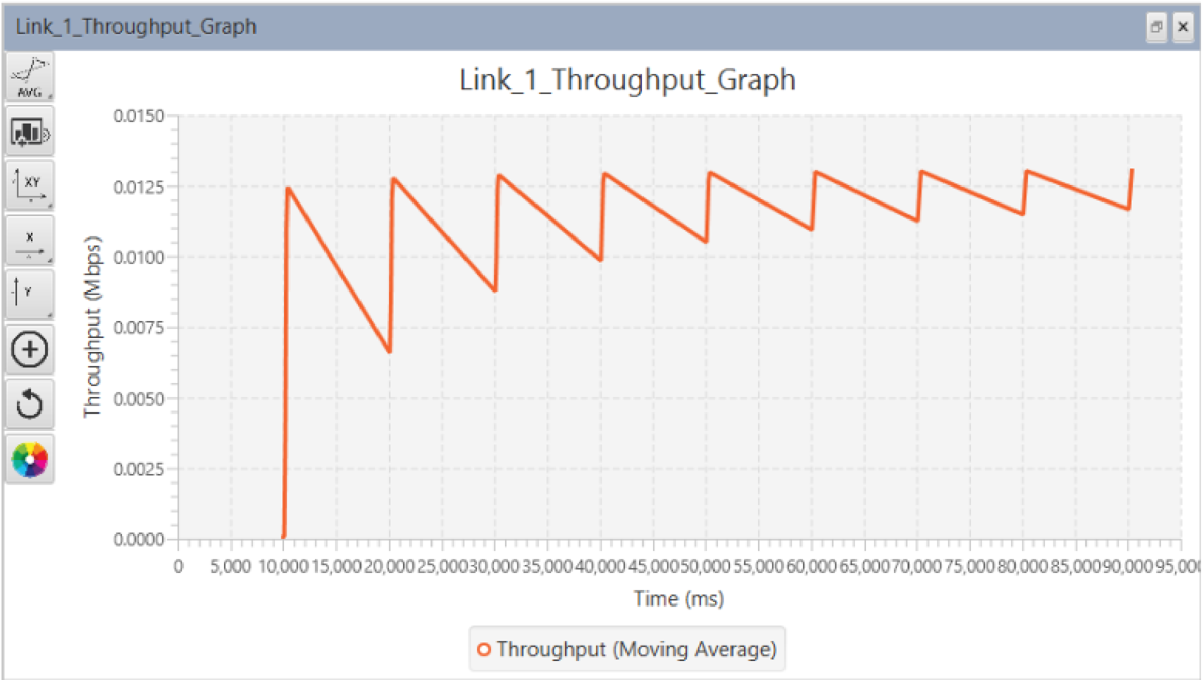
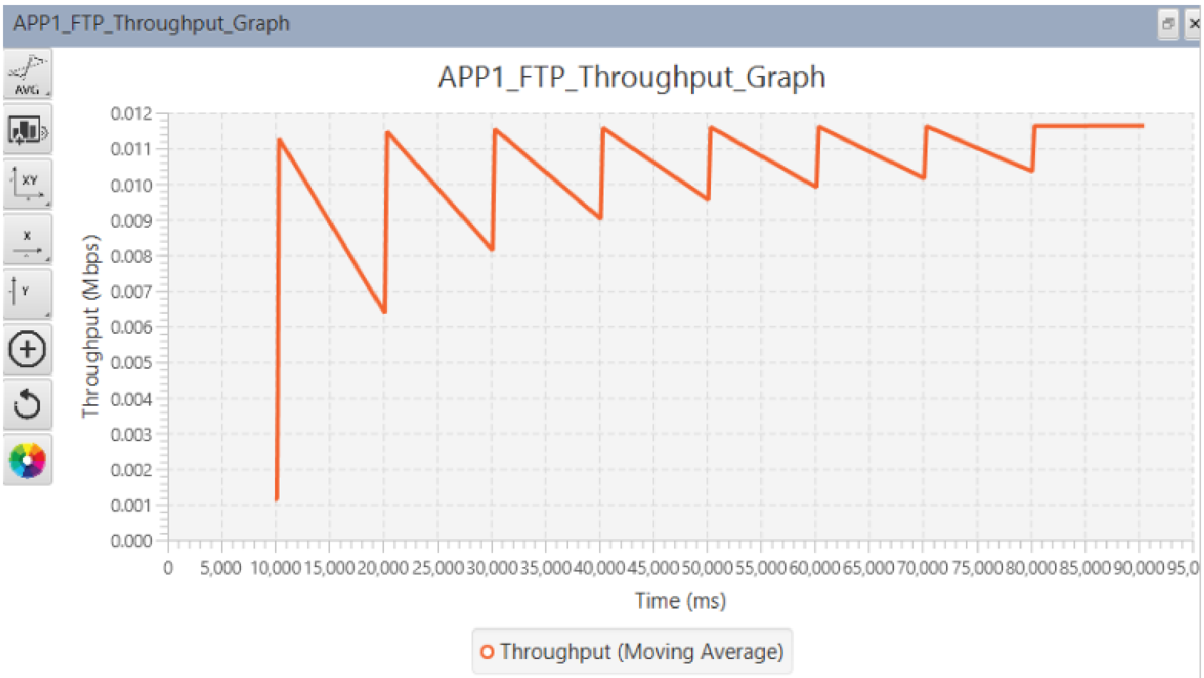
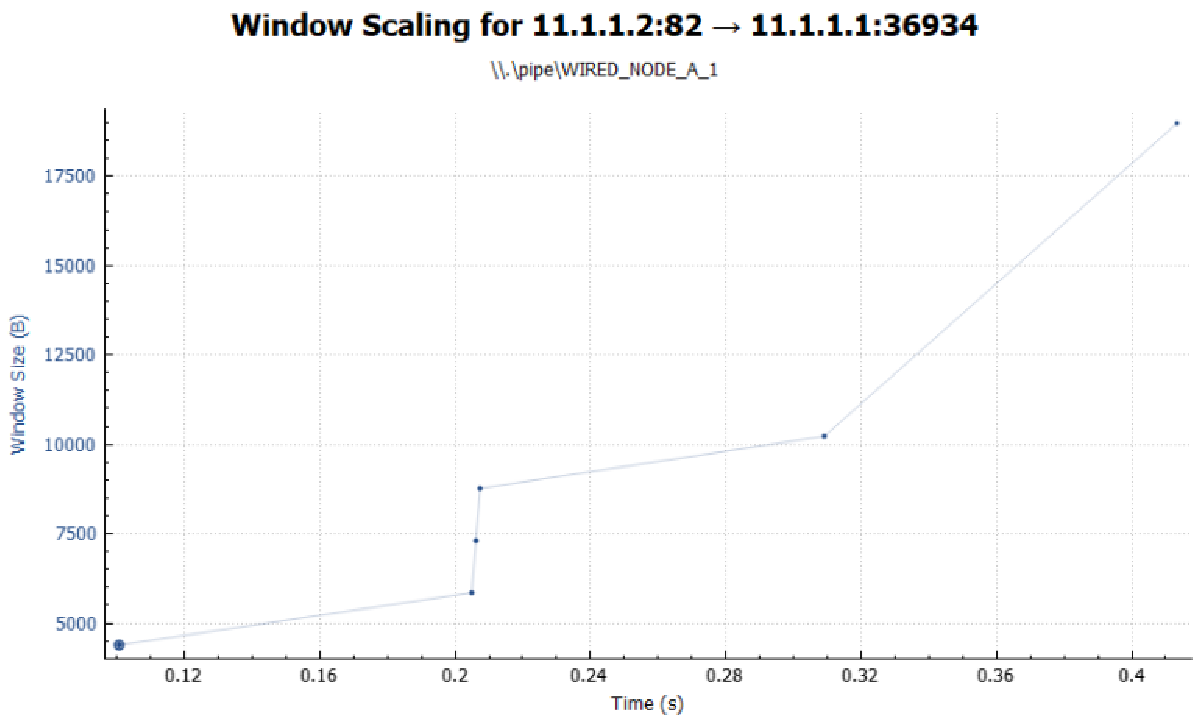
IT – 304 | CN | LAB-5

(Harsh Gajjar – 202201140)

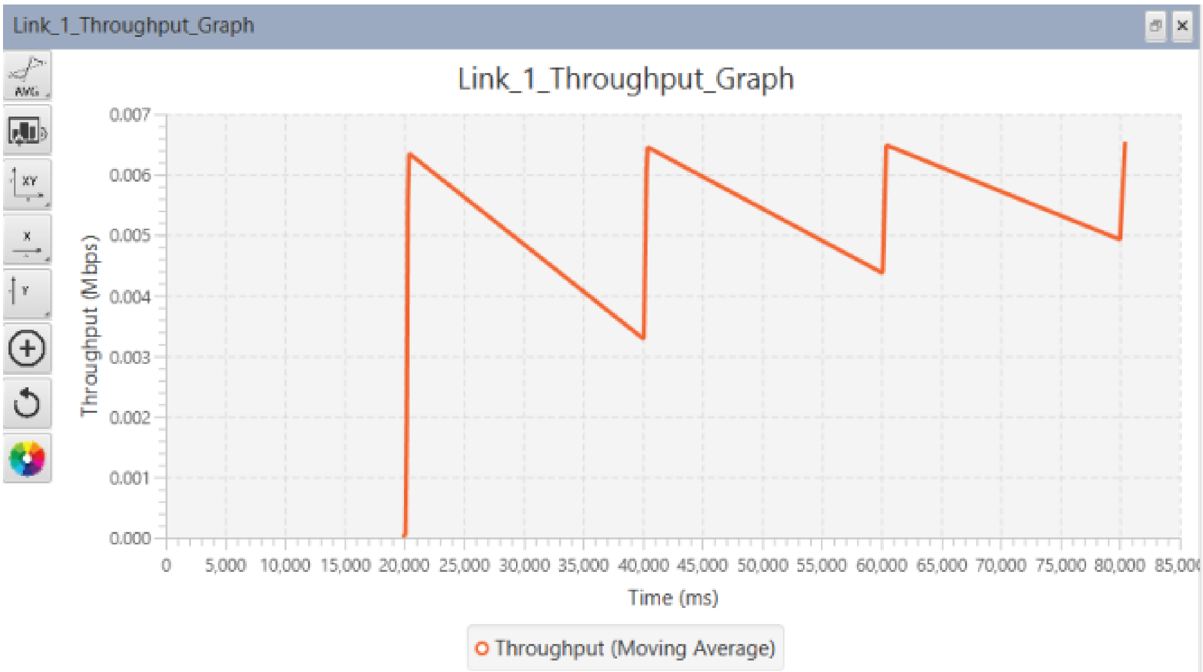
1. Introduction to TCP Connection Management

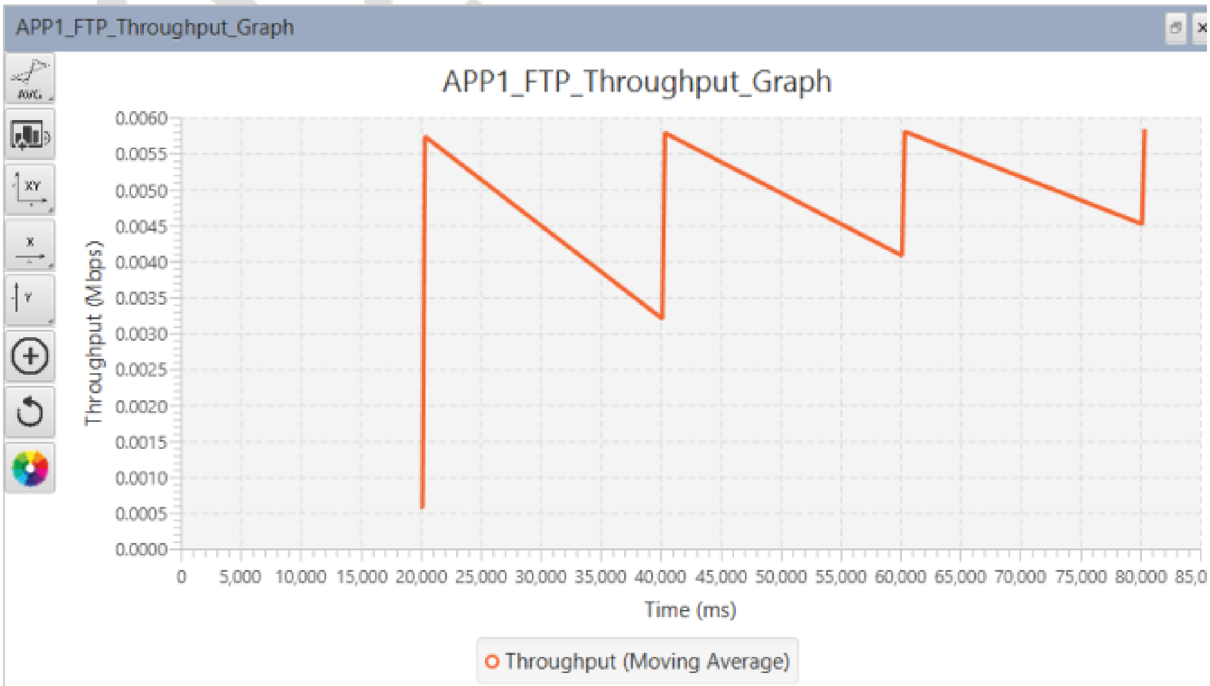


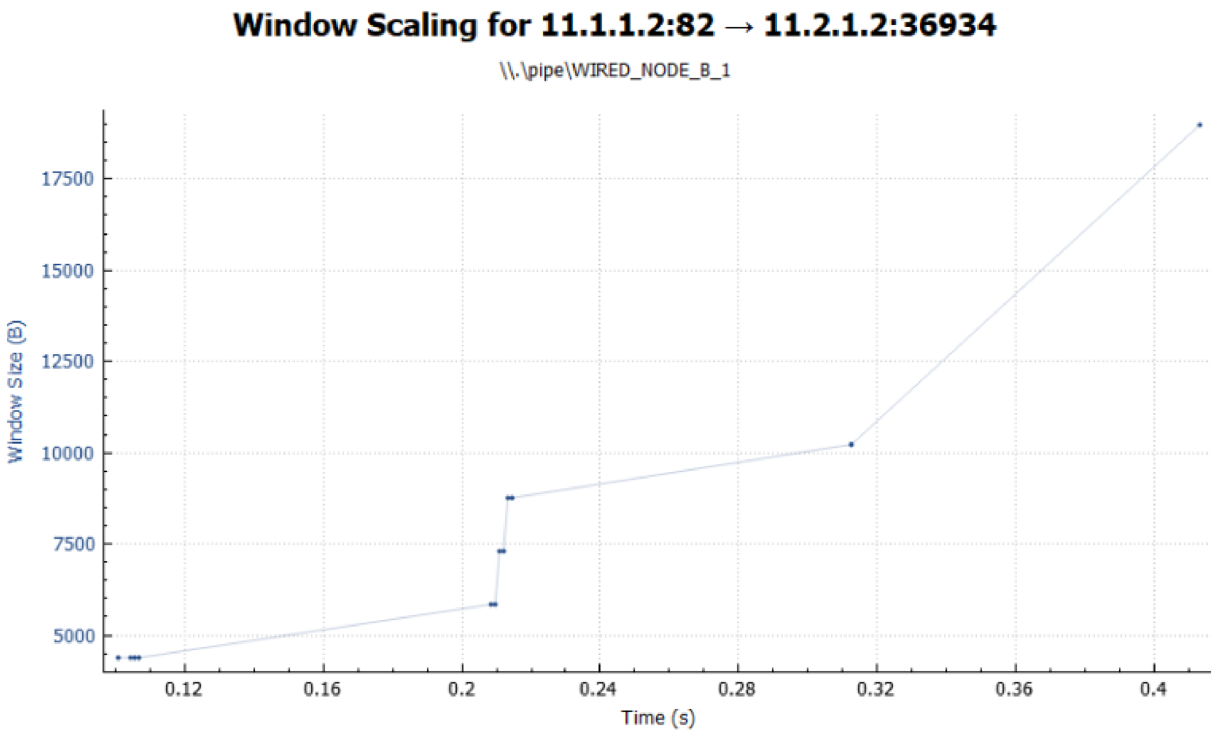




2. Reliable Data Transfer with TCP

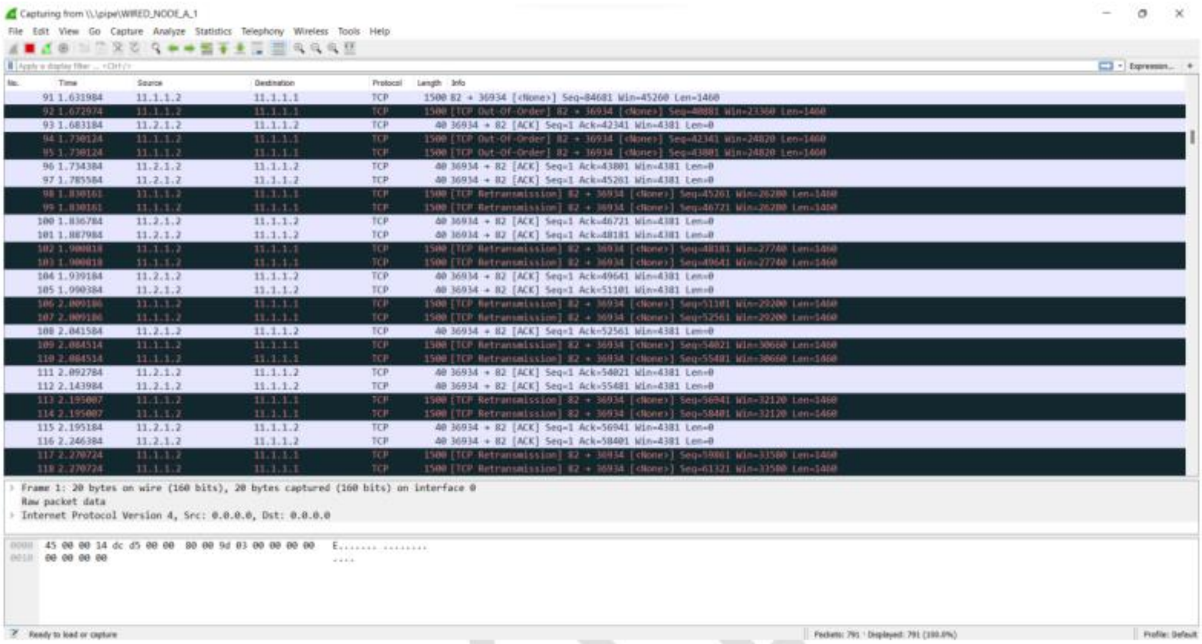


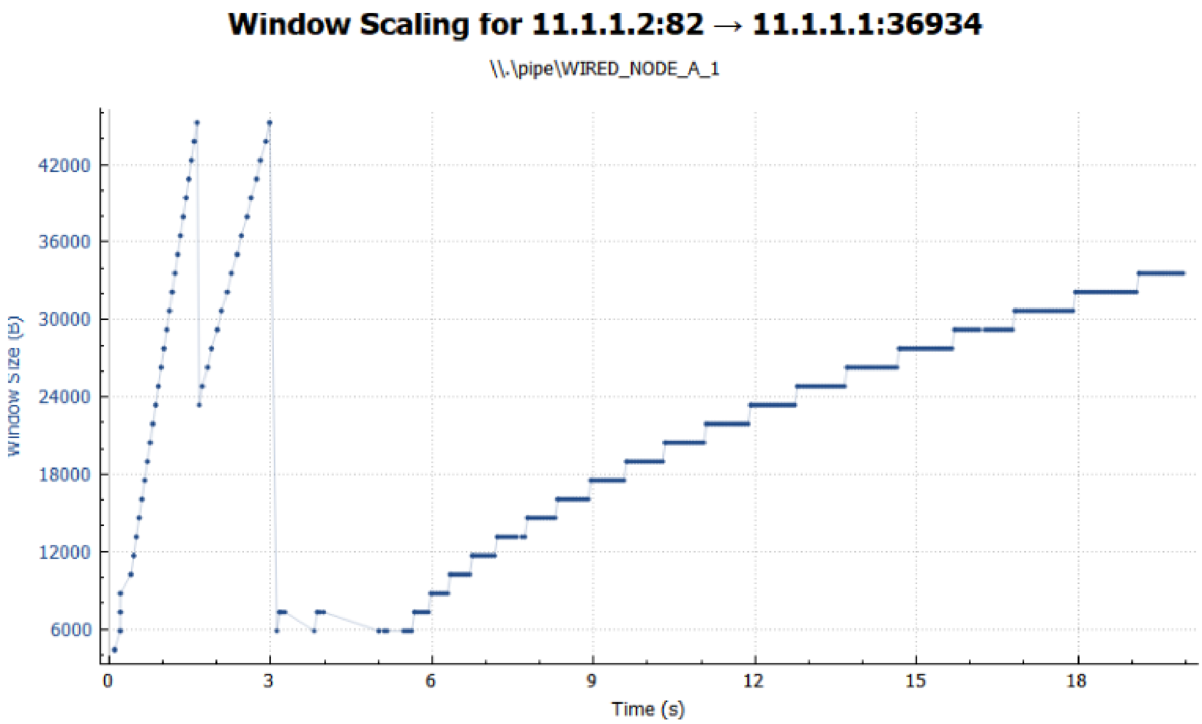




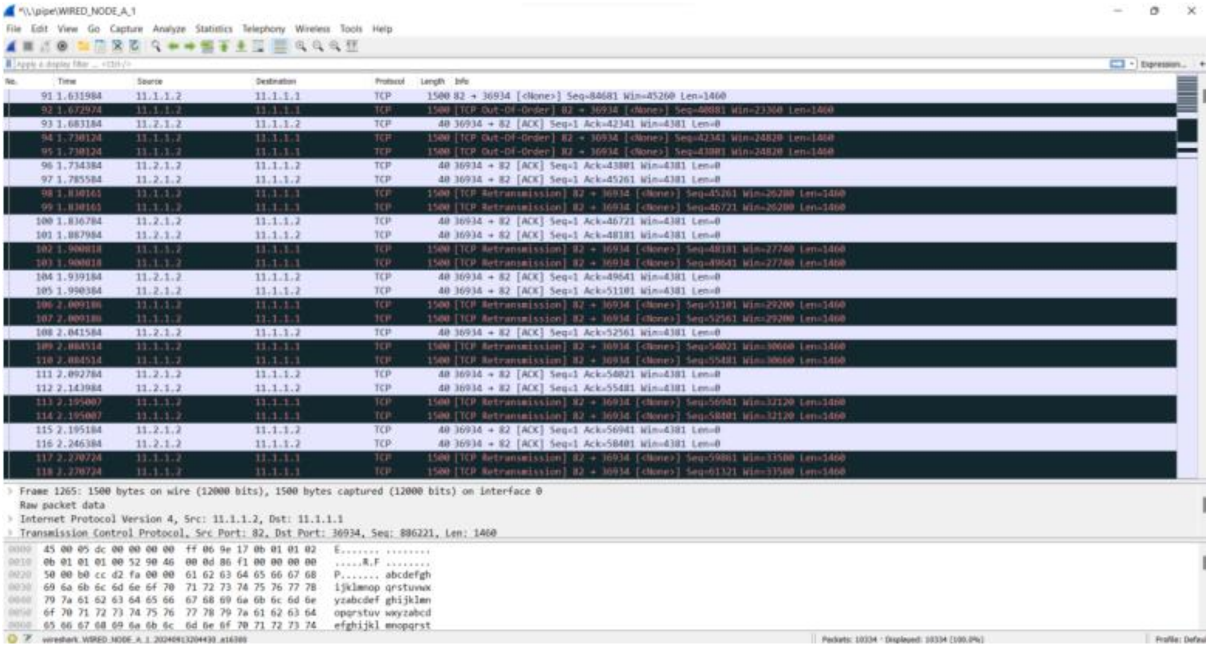
3. TCP Congestion Control Algorithms

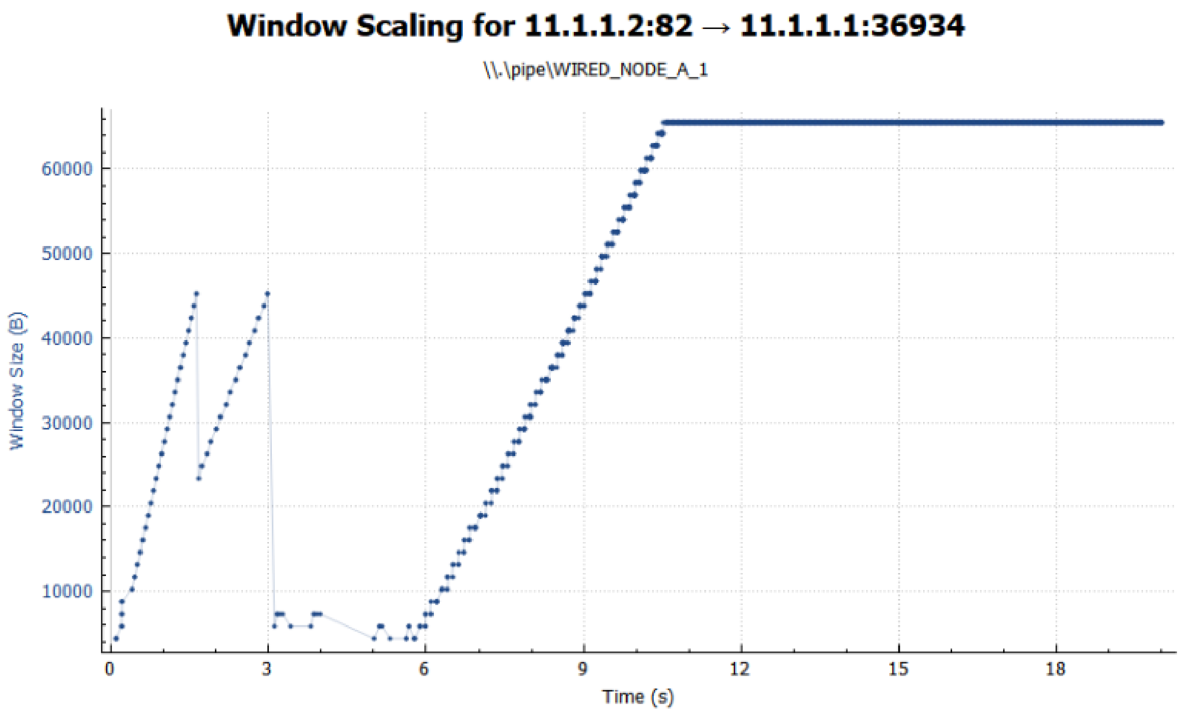
1) OLD TAHOE



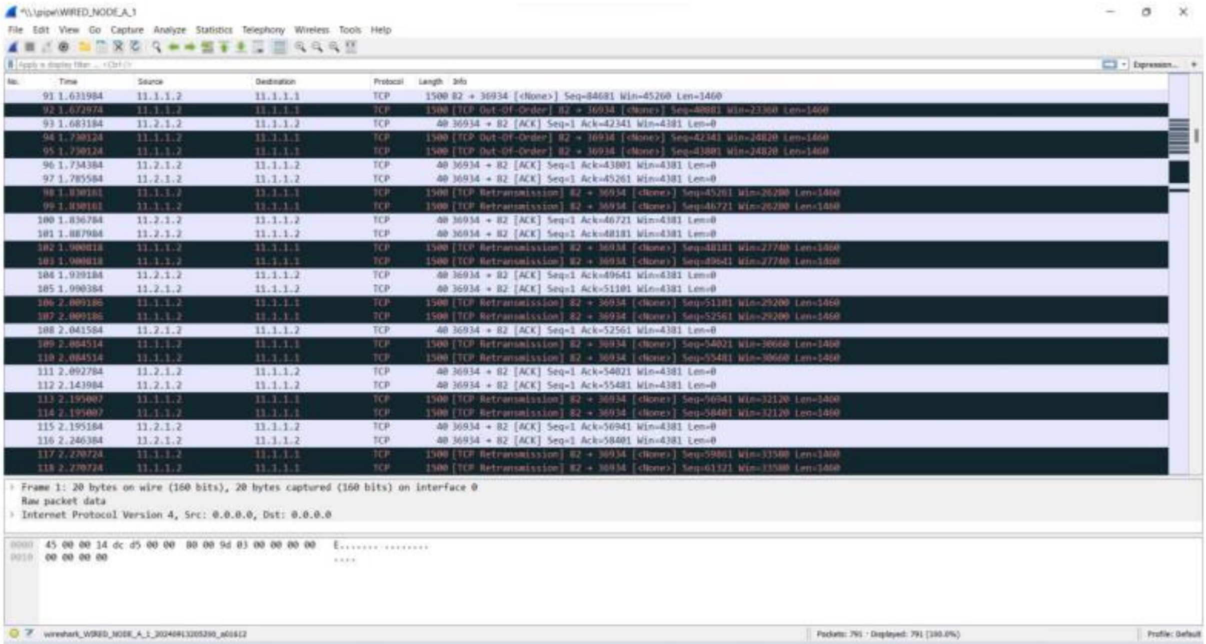

Window Scaling for 11.1.1.2:82 → 11.1.1.1:36934
\\.\pipe\WIRED_NODE_A_1
Window size (B)
42000
36000
30000
24000
18000
12000
6000
0
3
6
9
12
15
18
Time (s)

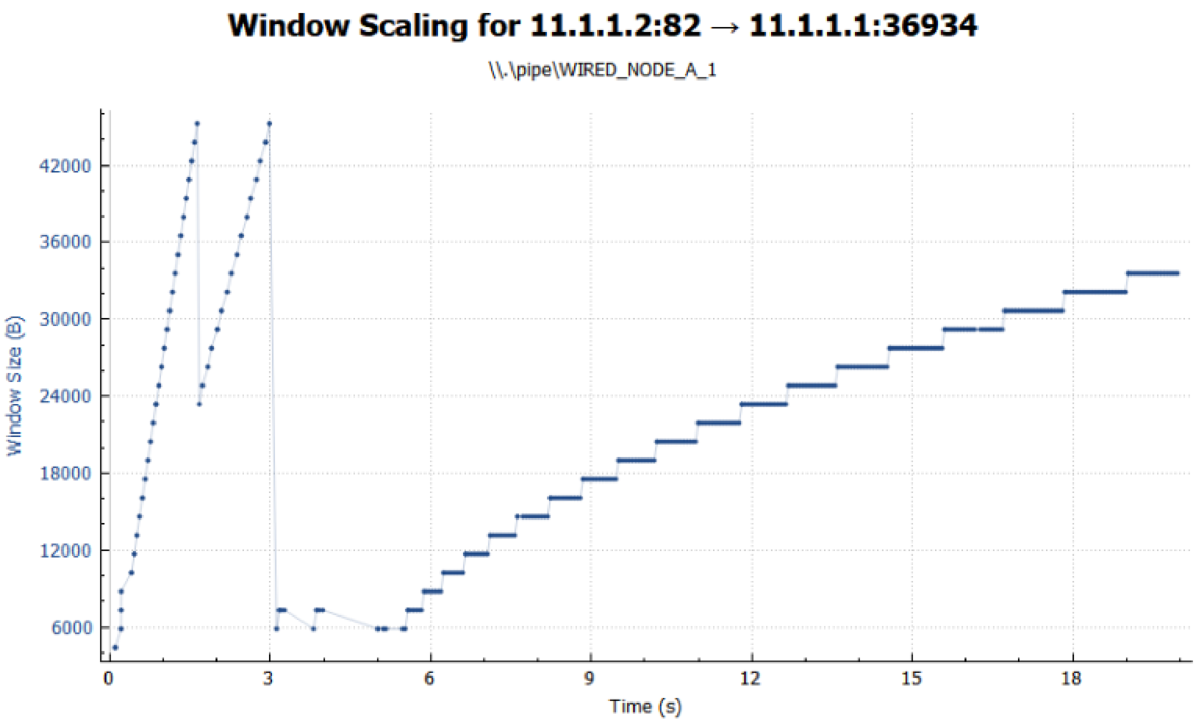
2) TAHOE


*\\.\pipe\WIRED_NODE_A_1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help


Window Scaling for 11.1.1.2:82 → 11.1.1.1:36934
\\.\pipe\WIRED_NODE_A_1
Window Size (B)
60000
50000
40000
30000
20000
10000
0
3
6
9
12
15
18
Time (s)

3) NEW RENO




Window Scaling for 11.1.1.2:82 → 11.1.1.1:36934
\\.\pipe\WIRED_NODE_A_1
Window Size (B)
42000
36000
30000
24000
18000
12000
6000
0
3
6
9
12
15
18
Time (s)

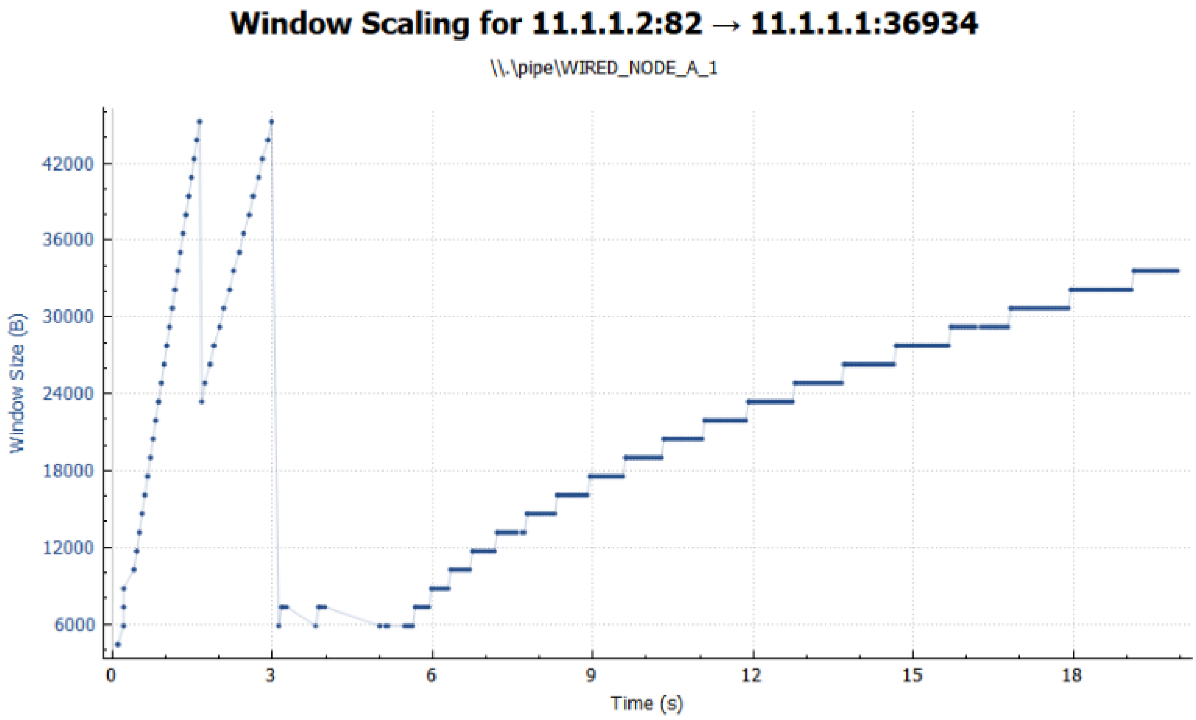
Exercise:

1. Understanding the communication between server and client via TCP. Plotting the congestion plot. Write up a full explanation describing why the plot has certain movements at certain points in all the exercises.

→ Reliable connection between the server and client via TCP (Transmission Control Protocol) guarantees that data is transferred without loss and in the right order.
TCP employs techniques like acknowledgements (ACKs), sequence numbers for data tracking, and window sizes to regulate the amount of data transferred at any one time in order to do this. TCP uses strategies like Slow Start, Congestion Avoidance, and Fast Retransmit to modify the data flow in the event of network congestion, ensuring uninterrupted communication without overloading the connection.

Explanation of Congestion Plots for Each TCP Type:

1) OLD TAHOE

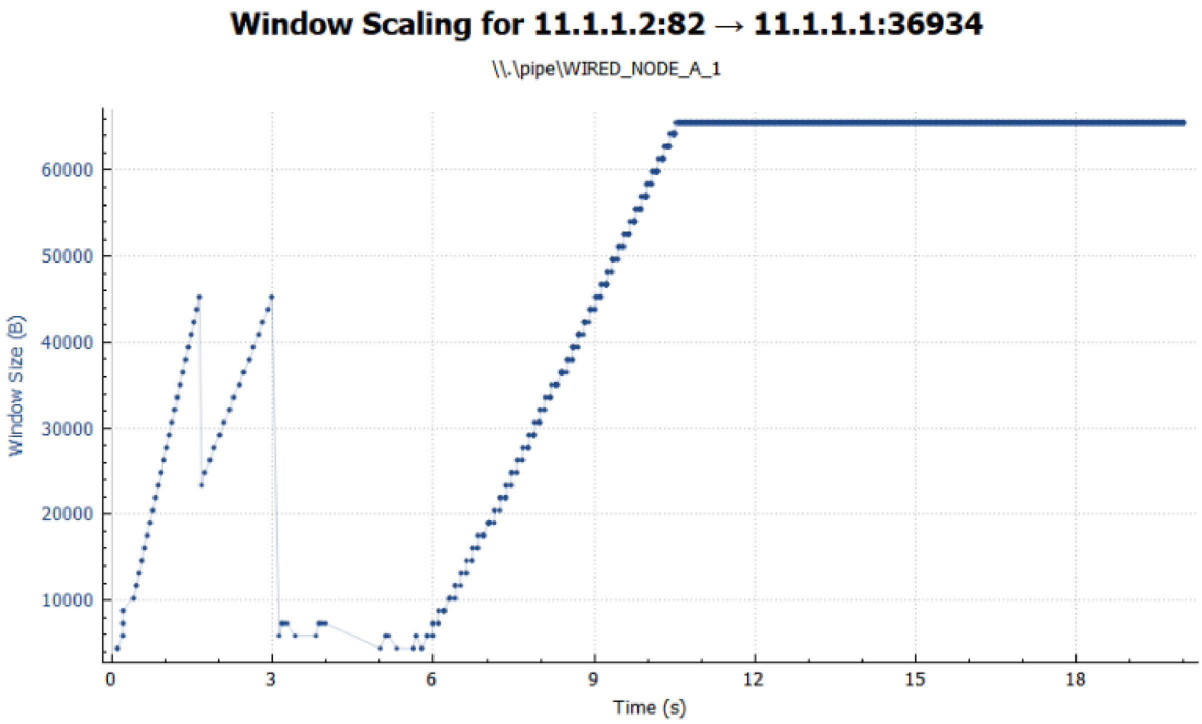


→ In order to prevent congestion and preserve stability, linear growth takes place after the first Slow Start phase, which is represented by exponential growth in the congestion window on the graph.

→ Unlike contemporary techniques like New Reno, the graph shows that if a single packet is lost, all data must be retransmitted from the beginning, resulting in a delayed recovery.

→ After congestion is detected, abrupt steep dips to 1 in the congestion window are noted.

2) TAHOE



→ The congestion plot for New Reno starts out slowly but expands rapidly.

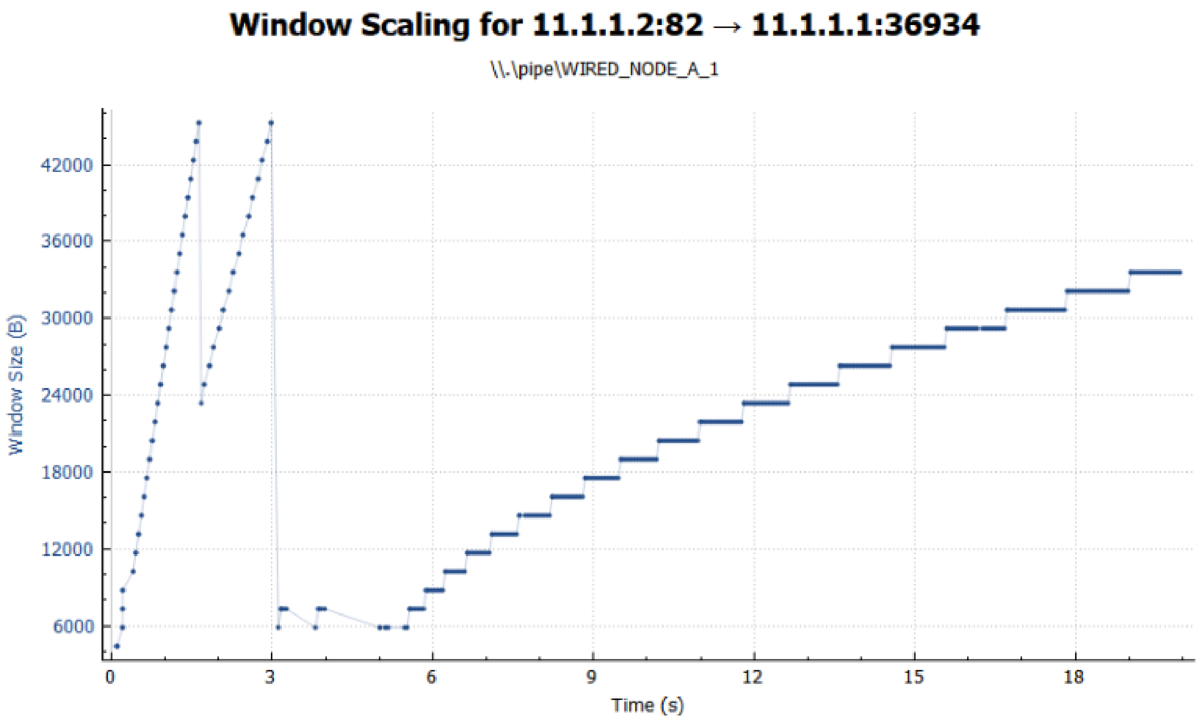
→ To avoid congestion and preserve network reliability, it transitions to linear growth after hitting the window size barrier.

→ The congestion window abruptly drops to 1 when packet loss is detected, which slows down the recovery process.

→ Lost packets are fully retransmitted either after a timeout or after getting several duplicate ACKs.

→ As it adjusts to network conditions, the algorithm alternates between exponential growth, abrupt dips, and linear increments.

3) NEW RENO



→ The congestion plot of New Reno rises dramatically at first.

→ To avoid congestion and guarantee network dependability, it switches to linear development after hitting the window size threshold.

→ The graph demonstrates how this method, which relies on partial ACKs to retransmit only the lost data, provides for a smoother and speedier recovery by not resetting the window size to 1 following packet loss.

→ After a timeout, the graph shows a rapid drop, which causes the congestion window to reset and develop slowly once more.

2. Briefly discuss the TCP Tahoe, Reno and New Reno Algorithms. In what characteristics they differ from each other?

→ Tahoe begins with a slow start, where it gradually increases the sending rate until a specific threshold is reached. On the other hand, Reno and New Reno also initiate with a slow start but recover faster when congestion is identified.

→ In Tahoe, congestion avoidance follows a linear growth pattern, where the window size slowly increases after the threshold is crossed. While Reno and New Reno also manage congestion similarly, New Reno allows for smoother growth even in the event of packet loss.

→ After a packet loss occurs, Tahoe fully resets the congestion window, restarting the transmission from the beginning. Reno, however, reduces the window size by half for quicker recovery. New Reno enhances this by only retransmitting the lost packets and preserving the congestion window.

→ Tahoe waits for a timeout to retransmit lost packets, while Reno employs fast retransmit by resending lost data upon detecting three duplicate acknowledgments. New Reno also retransmits promptly but manages partial acknowledgments more efficiently, facilitating smoother recovery.

→ Tahoe treats every packet loss uniformly, causing the sender to significantly slow down. Conversely, Reno and New Reno handle packet losses more effectively, allowing parts of the data transfer to continue during recovery. New Reno, in particular, excels at managing partial packet losses, enabling smoother and more continuous transmission.

3. Understand the plots with lossy network and lossless network and explain their behaviour throughout the plot.

→ Lossless Network: The window size increases steadily without any disruptions or packet losses. The network makes full use of the available bandwidth, leading to efficient and uninterrupted throughput.

→ Lossy Network: While the window size increases, it experiences sudden decreases due to packet losses, triggering frequent retransmissions and recovery processes. Constant packet loss and retransmissions hinder the network's ability to effectively utilize the available bandwidth, resulting in reduced overall throughput.

Groups Specific Questions:

1) For Group 1-2: Based on the Netsim Manual Experiment 7, Write your observations of the below packet capture file.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	0.0.0.0	IPv4	20	
2	0.000000	11.1.1.2	11.1.1.1	TCP	44	82 → 36934 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
3	0.100335	11.3.1.2	11.1.1.2	TCP	44	36934 → 82 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460
4	0.100335	11.1.1.2	11.1.1.1	TCP	40	82 → 36934 [ACK] Seq=1 Ack=1 Win=4380 Len=0
5	0.100335	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=1 Win=4380 Len=1460
6	0.100335	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=1461 Win=4380 Len=1460
7	0.100335	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=2921 Win=4380 Len=1460
8	0.204208	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=1461 Win=4381 Len=0
9	0.204208	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=4381 Win=5840 Len=1460
10	0.204208	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=5841 Win=5840 Len=1460
11	0.205430	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=2921 Win=4381 Len=0
12	0.205430	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=7301 Win=7300 Len=1460
13	0.205430	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=8761 Win=7300 Len=1460
14	0.206651	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=4381 Win=4381 Len=0
15	0.206651	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=10221 Win=8760 Len=1460
16	0.206651	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=11681 Win=8760 Len=1460
17	0.308027	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=5841 Win=4381 Len=0
18	0.308027	11.1.1.2	11.1.1.1	TCP	1500	82 → 36934 [<None>] Seq=13141 Win=10220 Len=1460
19	0.309249	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=7301 Win=4381 Len=0
20	0.310471	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=8761 Win=4381 Len=0
21	0.311692	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=10221 Win=4381 Len=0
22	0.312914	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=11681 Win=4381 Len=0
23	0.314136	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=13141 Win=4381 Len=0
24	0.411846	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [ACK] Seq=1 Ack=14601 Win=4381 Len=0
25	0.411846	11.1.1.2	11.1.1.1	TCP	40	82 → 36934 [FIN] Seq=14601 Win=18980 Len=0
26	0.512161	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [FIN, ACK] Seq=1 Ack=14601 Win=4381 Len=0
27	0.512215	11.3.1.2	11.1.1.2	TCP	40	36934 → 82 [FIN] Seq=2 Win=4381 Len=0
28	0.512215	11.1.1.2	11.1.1.1	TCP	40	82 → 36934 [ACK] Seq=14602 Ack=3 Win=18980 Len=0

1. TCP Handshake Initialization:

- In packet 3, the connection setup begins with a SYN message sent from source 11.1.1.2 to destination 11.1.1.1 (from port 82 to port 36934).
- Packet 4 contains the SYN-ACK response from the destination, 11.1.1.1.
- Finally, packet 5 completes the three-way handshake with an ACK from the source, establishing the connection.

2. Data Transmission:

- Packets 6 through 22 involve the exchange of TCP segments with different sequence numbers and acknowledgments.
- The large packet sizes (around 1500 bytes) suggest the transfer of significant amounts of data.

3. Acknowledgments:

- TCP acknowledgments can be seen in packets 8, 11, 14, 17, and 20, sent from 11.1.1.1 back to 11.1.1.2, confirming the receipt of data.
- These ACK packets reference different sequence numbers, ensuring successful data transmission.

4. Multiple Connections:

- Traffic from another source, 11.3.1.2, is visible in packets 9 and 24, indicating that multiple clients are communicating with the same server (11.1.1.1), possibly conducting simultaneous data transfers.

5. Session Termination:

- Packets 26 and 27 show the session being terminated. Packet 26, sent by the destination 11.1.1.1, includes a FIN flag, indicating the intention to close the connection.
- In packet 27, the source 11.1.1.2 acknowledges the FIN, thereby concluding the TCP session.