8 Reliable data transfer with TCP

8.1 Introduction

TCP provides reliable data transfer service to the application processes even when the underlying network service (IP service) is unreliable (loses, corrupts, garbles or duplicates packets). TCP uses checksum, sequence numbers, acknowledgements, timers, and retransmission to ensure correct and in order delivery of data to the application processes.

TCP views the data stream from the client application process as an ordered stream of bytes. TCP will grab chunks of this data (stored temporarily in the TCP send buffer), add its own header and pass it on to the network layer. A key field of the TCP header is the sequence number which indicates the position of the first byte of the TCP data segment in the data stream. The sequence number will allow the TCP receiver to identify segment losses, duplicate packets and to ensure correct delivery of the data stream to the server application process.

When a server receives a TCP segment, it acknowledges the same with an ACK segment (the segment carrying the acknowledgement has the ACK bit set to 1) and also conveys the sequence number of the first missing byte in the application data stream, in the acknowledgement number field of the TCP header. All acknowledgements are cumulative hence, all missing and out-of-order TCP segments will result in duplicate acknowledgements for the corresponding TCP segments.

TCP sender relies on sequence numbering and acknowledgements to ensure reliable transfer of the data stream. In the event of a timeout (no acknowledgement is received before the timer expires) or triple duplicate acknowledgements (multiple ACK segments indicate a lost or missing TCP segment) for a TCP segment, the TCP sender will retransmit the segment until the TCP segment is acknowledged (at least cumulatively). In **Figure 8-1**, we illustrate retransmission by the TCP sender after a timeout for acknowledgement.

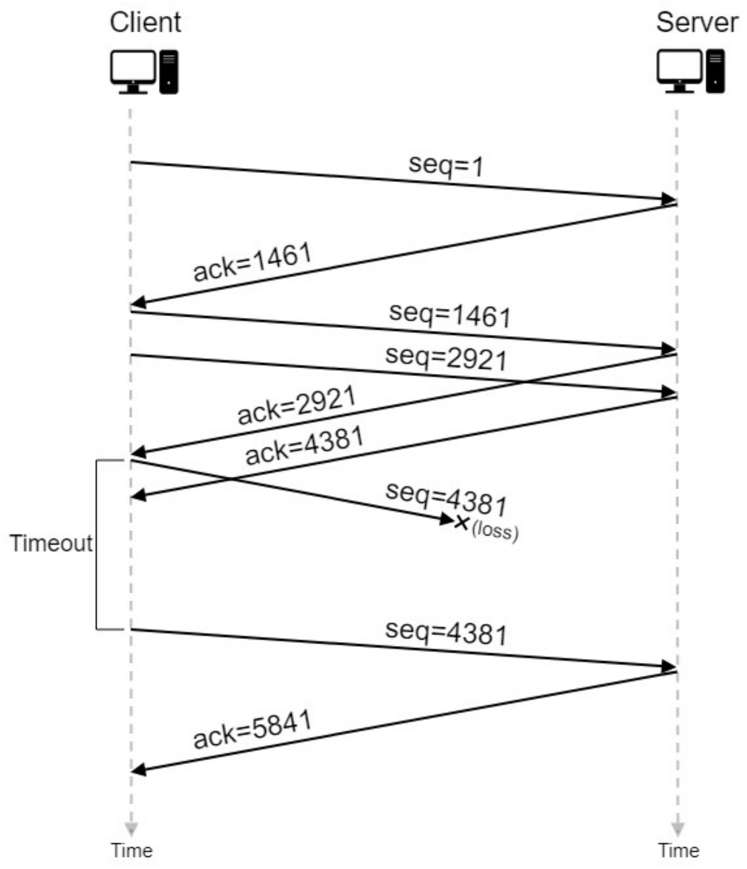
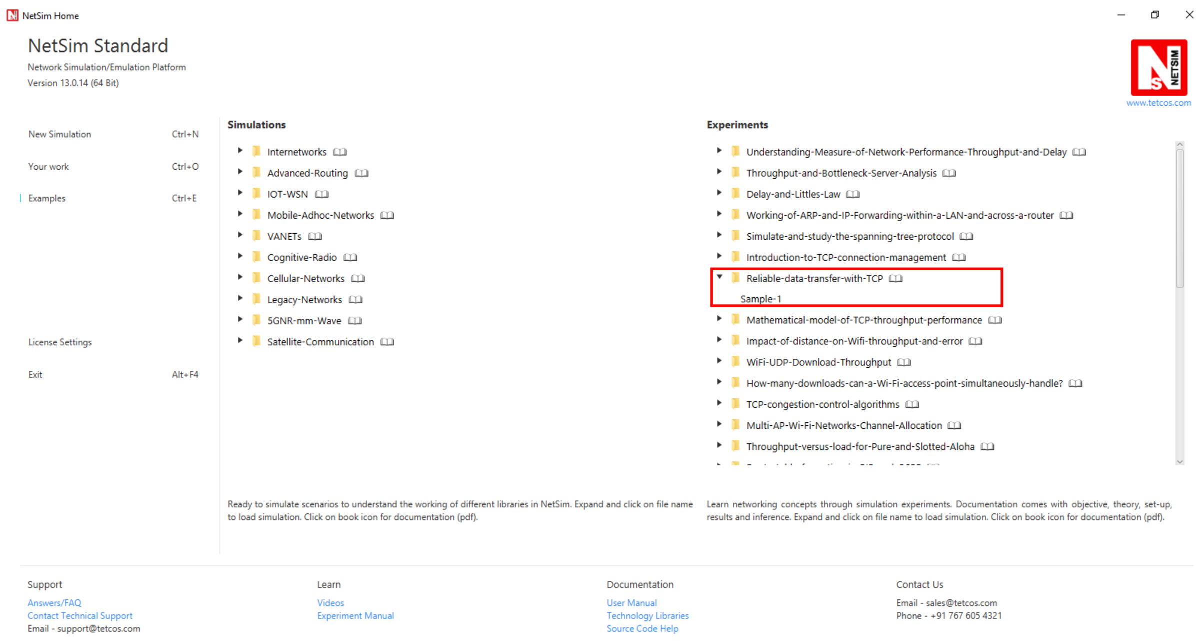


Figure 8-1: An illustration of TCP retransmission with timeout. The segment with sequence number 4381 is lost in the network. The TCP client retransmits the segment after a timeout event

8.2 Network Setup

We will seek a simple file transfer with TCP over a lossy link to study reliable data transfer with TCP. We will simulate the network setup illustrated in **Figure 8-3** with the configuration parameters listed in detail to study reliable data transfer with TCP connection.

Open NetSim and click **Examples > Experiments > Reliable-data-transfer-with-TCP > Sample-1** as shown below **Figure 8-2.**



NetSim Home

NetSim Standard

Network Simulation/Emulation Platform

Version 13.0.14 (64 Bit)

New Simulation	Ctrl+N
Your work	Ctrl+O
Examples	Ctrl+E
License Settings	
Exit	Alt+F4

Simulations

- Internetworks
- Advanced-Routing
- IOT-WSN
- Mobile-Adhoc-Networks
- VANETs
- Cognitive-Radio
- Cellular-Networks
- Legacy-Networks
- 5GNR-mm-Wave
- Satellite-Communication

Ready to simulate scenarios to understand the working of different libraries in NetSim. Expand and click on file name to load simulation. Click on book icon for documentation (pdf).

Experiments

- Understanding-Measure-of-Network-Performance-Throughput-and-Delay
- Throughput-and-Bottleneck-Server-Analysis
- Delay-and-Littles-Law
- Working-of-ARP-and-IP-Forwarding-within-a-LAN-and-across-a-router
- Simulate-and-study-the-spanning-tree-protocol
- Introduction-to-TCP-connection-management
- Reliable-data-transfer-with-TCP
  - Sample-1
- Mathematical-model-of-TCP-throughput-performance
- Impact-of-distance-on-Wifi-throughput-and-error
- WiFi-UDP-Download-Throughput
- How-many-downloads-can-a-Wi-Fi-access-point-simultaneously-handle?
- TCP-congestion-control-algorithms
- Multi-AP-Wi-Fi-Networks-Channel-Allocation
- Throughput-versus-load-for-Pure-and-Slotted-Aloha

Learn networking concepts through simulation experiments. Documentation comes with objective, theory, set-up, results and inference. Expand and click on file name to load simulation. Click on book icon for documentation (pdf).

www.tetcos.com

Support
Answers/FAQ
Contact Technical Support
Email - support@tetcos.com

Learn
Videos
Experiment Manual

Documentation
User Manual
Technology Libraries
Source Code Help

Contact Us
Email - sales@tetcos.com
Phone - +91 767 605 4321

Figure 8-2: Experiment List

NetSim UI displays the configuration file corresponding to this experiment as shown below **Figure 8-3**.

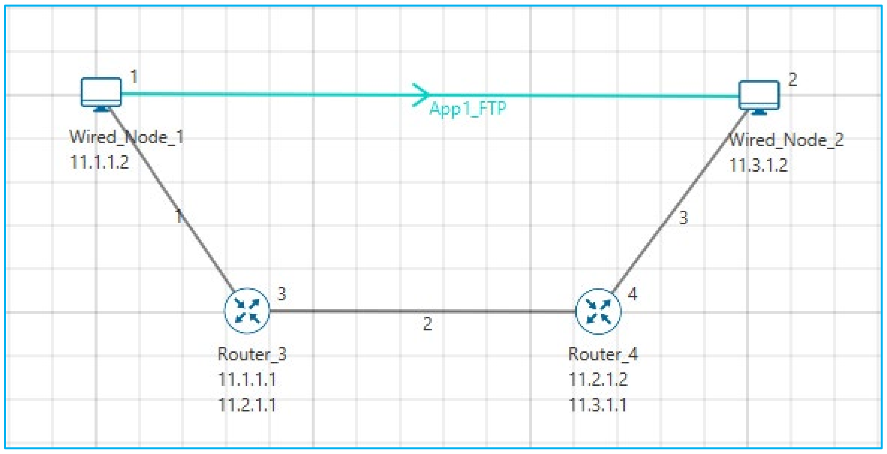


Figure 8-3: FTP application between a client and a server

8.3 Procedure

The following set of procedures were done to generate this sample.

Step 1: A network scenario is designed in NetSim GUI comprising of 2 Wired Nodes and 2 Routers in the **“Internetworks”** Network Library.

Step 2: In the General Properties of Wired Node 1 i.e., Source and Wired Node 2 i.e., Destination, Wireshark Capture is set to Online. Transport Layer properties Congestion plot is set to true.

Note: *Accept default properties for Routers as well as the Links.*

Step 3: Right-click the link ID (of a wired link) and select Properties to access the link's properties. Set Max Uplink Speed and Max Downlink Speed to **10** Mbps. Set Uplink BER and Downlink BER to **0**. Set Uplink Propagation Delay and Downlink Propagation Delay as **100** microseconds for the links 1 and 3 (between the Wired Node's and the routers). Set Uplink Propagation Delay and Downlink Propagation Delay as **50000** microseconds and Uplink BER and Downlink BER to **0.00001** for the backbone link connecting the routers, i.e., 2.

Step 4: Right click on the Application Flow **App1 FTP** and select Properties or click on the Application icon present in the top ribbon/toolbar.

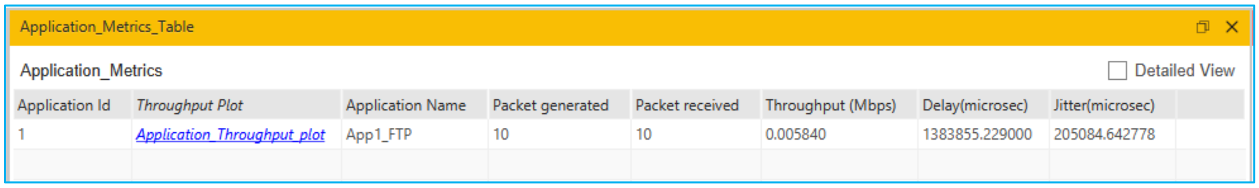
An FTP Application is generated from Wired Node 1 i.e. Source to Wired Node 2 i.e. Destination with File Size set to 14600 Bytes and File Inter Arrival Time set to 20 Seconds.

Step 5: Click on Display Settings > Device IP check box in the NetSim GUI to view the network topology along with the IP address.

Step 6: Enable the plots and click on Run simulation. The simulation time is set to 20 seconds.

8.4 Output

We aimed to transfer a file of size 14600 bytes (i.e., 10 packets, each of size 1460 bytes) with TCP over a lossy link. In **Figure 8-4**, we report the application metrics data for FTP which indicates that the complete file was transferred.


Application_Metrics_Table

Application_Metrics ☐ Detailed View

Application Id	Throughput Plot	Application Name	Packet generated	Packet received	Throughput (Mbps)	Delay(microsec)	Jitter(microsec)
1	Application_Throughput_plot	App1_FTP	10	10	0.005840	1383855.229000	205084.642778

Figure 8-4: Application metrics table for FTP

We have enabled Wireshark Capture in Wired _Node 1 and Wired Node _2. The PCAP files are generated at the end of the simulation and are shown in **Figure 8-5** and **Figure 8-6**.

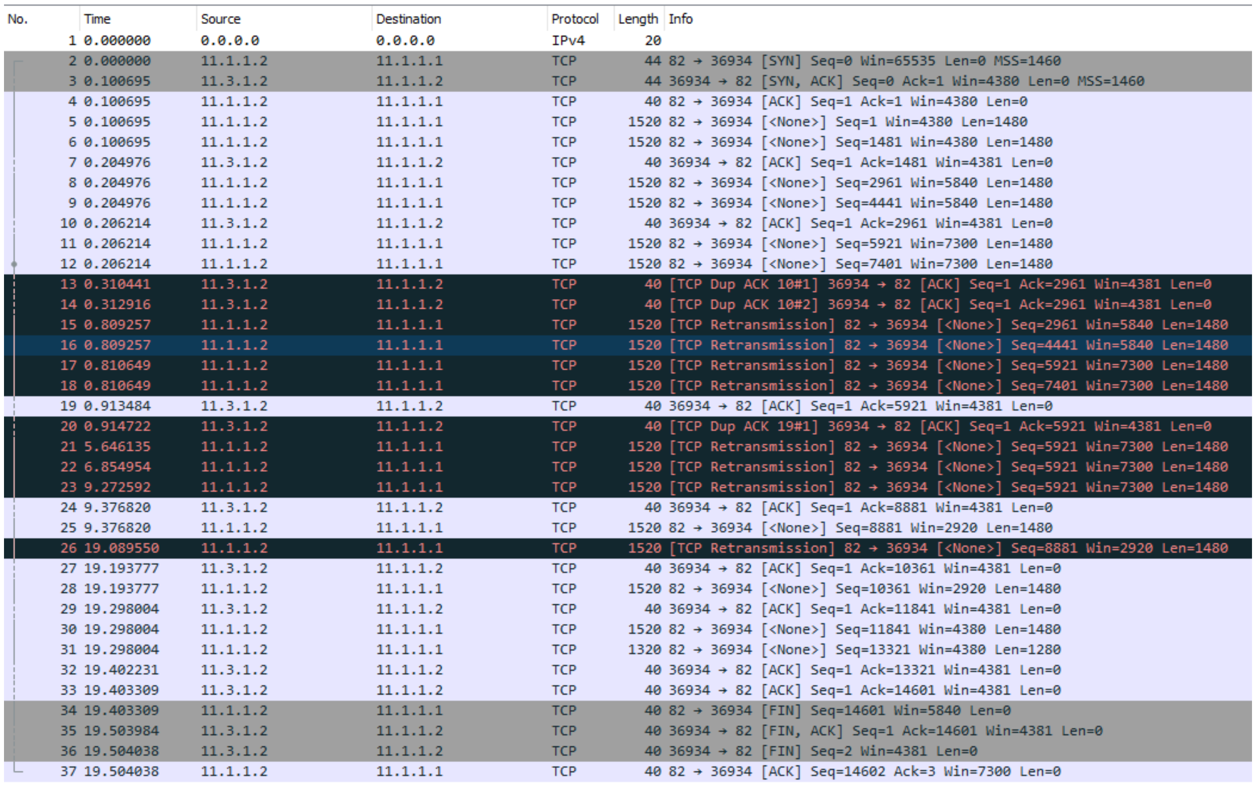

```
No. Time       Source    Destination Protocol Length Info
 1 0.000000    0.0.0.0   0.0.0.0     IPv4     20
 2 0.000000    11.1.1.2  11.1.1.1    TCP      44 82 → 36934 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
 3 0.100695    11.3.1.2  11.1.1.2    TCP      44 36934 → 82 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460
 4 0.100695    11.1.1.2  11.1.1.1    TCP      40 82 → 36934 [ACK] Seq=1 Ack=1 Win=4380 Len=0
 5 0.100695    11.1.1.2  11.1.1.1    TCP      1520 82 → 36934 [<None>] Seq=1 Win=4380 Len=1480
 6 0.100695    11.1.1.2  11.1.1.1    TCP      1520 82 → 36934 [<None>] Seq=1481 Win=4380 Len=1480
 7 0.204976    11.3.1.2  11.1.1.2    TCP      40 36934 → 82 [ACK] Seq=1 Ack=1481 Win=4381 Len=0
 8 0.204976    11.1.1.2  11.1.1.1    TCP      1520 82 → 36934 [<None>] Seq=2961 Win=5840 Len=1480
 9 0.204976    11.1.1.2  11.1.1.1    TCP      1520 82 → 36934 [<None>] Seq=4441 Win=5840 Len=1480
10 0.206214    11.3.1.2  11.1.1.2    TCP      40 36934 → 82 [ACK] Seq=1 Ack=2961 Win=4381 Len=0
11 0.206214    11.1.1.2  11.1.1.1    TCP      1520 82 → 36934 [<None>] Seq=5921 Win=7300 Len=1480
12 0.206214    11.1.1.2  11.1.1.1    TCP      1520 82 → 36934 [<None>] Seq=7401 Win=7300 Len=1480
13 0.310441    11.3.1.2  11.1.1.2    TCP      40 [TCP Dup ACK 10#1] 36934 → 82 [ACK] Seq=1 Ack=2961 Win=4381 Len=0
14 0.312916    11.3.1.2  11.1.1.2    TCP      40 [TCP Dup ACK 10#2] 36934 → 82 [ACK] Seq=1 Ack=2961 Win=4381 Len=0
15 0.809257    11.1.1.2  11.1.1.1    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=2961 Win=5840 Len=1480
16 0.809257    11.1.1.2  11.1.1.1    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=4441 Win=5840 Len=1480
17 0.810649    11.1.1.2  11.1.1.1    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=5921 Win=7300 Len=1480
18 0.810649    11.1.1.2  11.1.1.1    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=7401 Win=7300 Len=1480
19 0.913484    11.3.1.2  11.1.1.2    TCP      40 36934 → 82 [ACK] Seq=1 Ack=5921 Win=4381 Len=0
20 0.914722    11.3.1.2  11.1.1.2    TCP      40 [TCP Dup ACK 19#1] 36934 → 82 [ACK] Seq=1 Ack=5921 Win=4381 Len=0
21 5.646135    11.1.1.2  11.1.1.1    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=5921 Win=7300 Len=1480
22 6.854954    11.1.1.2  11.1.1.1    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=5921 Win=7300 Len=1480
23 9.272592    11.1.1.2  11.1.1.1    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=5921 Win=7300 Len=1480
24 9.376820    11.3.1.2  11.1.1.2    TCP      40 36934 → 82 [ACK] Seq=1 Ack=8881 Win=4381 Len=0
25 9.376820    11.1.1.2  11.1.1.1    TCP      1520 82 → 36934 [<None>] Seq=8881 Win=2920 Len=1480
26 19.089550   11.1.1.2  11.1.1.1    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=8881 Win=2920 Len=1480
27 19.193777   11.3.1.2  11.1.1.2    TCP      40 36934 → 82 [ACK] Seq=1 Ack=10361 Win=4381 Len=0
28 19.193777   11.1.1.2  11.1.1.1    TCP      1520 82 → 36934 [<None>] Seq=10361 Win=2920 Len=1480
29 19.298004   11.3.1.2  11.1.1.2    TCP      40 36934 → 82 [ACK] Seq=1 Ack=11841 Win=4381 Len=0
30 19.298004   11.1.1.2  11.1.1.1    TCP      1520 82 → 36934 [<None>] Seq=11841 Win=4380 Len=1480
31 19.298004   11.1.1.2  11.1.1.1    TCP      1320 82 → 36934 [<None>] Seq=13321 Win=4380 Len=1280
32 19.402231   11.3.1.2  11.1.1.2    TCP      40 36934 → 82 [ACK] Seq=1 Ack=13321 Win=4381 Len=0
33 19.403309   11.3.1.2  11.1.1.2    TCP      40 36934 → 82 [ACK] Seq=1 Ack=14601 Win=4381 Len=0
34 19.403309   11.1.1.2  11.1.1.1    TCP      40 82 → 36934 [FIN] Seq=14601 Win=5840 Len=0
35 19.503984   11.3.1.2  11.1.1.2    TCP      40 36934 → 82 [FIN, ACK] Seq=1 Ack=14601 Win=4381 Len=0
36 19.504038   11.3.1.2  11.1.1.2    TCP      40 36934 → 82 [FIN] Seq=2 Win=4381 Len=0
37 19.504038   11.1.1.2  11.1.1.1    TCP      40 82 → 36934 [ACK] Seq=14602 Ack=3 Win=7300 Len=0
```

Figure 8-5: PCAP file at Wired Node 1. TCP ensures reliable data transfer using timeout, duplicate ACKs and retransmissions.

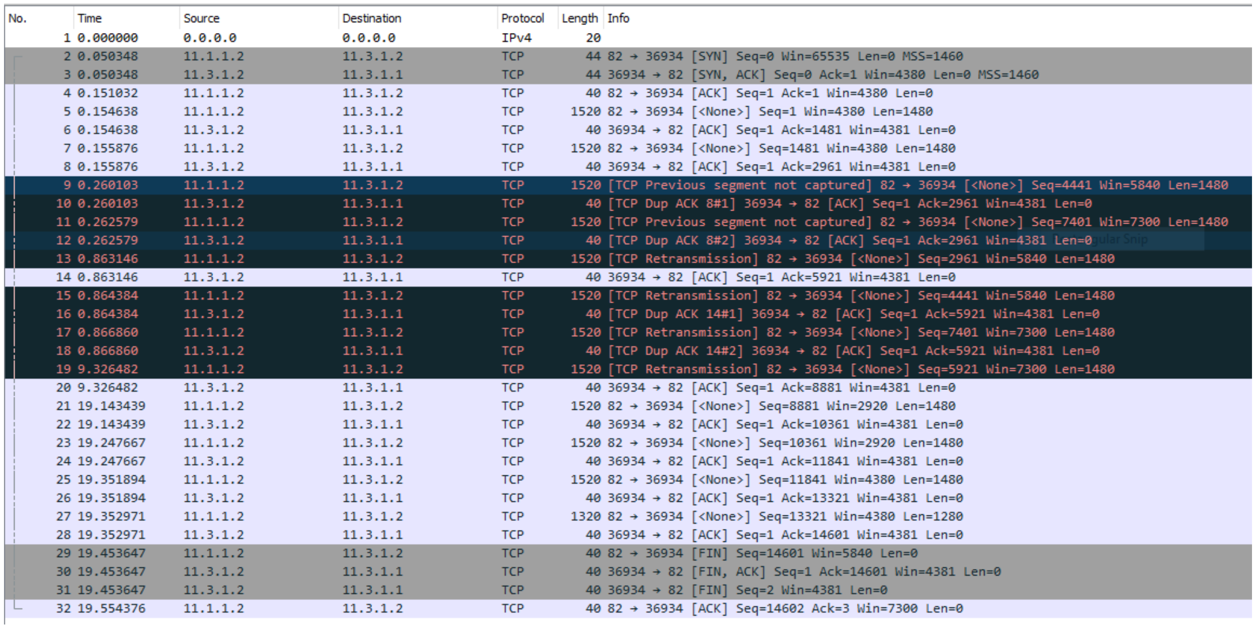

```
No. Time       Source    Destination Protocol Length Info
 1 0.000000    0.0.0.0   0.0.0.0     IPv4     20
 2 0.050348    11.1.1.2  11.3.1.2    TCP      44 82 → 36934 [SYN] Seq=0 Win=65535 Len=0 MSS=1460
 3 0.050348    11.3.1.2  11.3.1.1    TCP      44 36934 → 82 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460
 4 0.151032    11.1.1.2  11.3.1.2    TCP      40 82 → 36934 [ACK] Seq=1 Ack=1 Win=4380 Len=0
 5 0.154638    11.1.1.2  11.3.1.2    TCP      1520 82 → 36934 [<None>] Seq=1 Win=4380 Len=1480
 6 0.154638    11.3.1.2  11.3.1.1    TCP      40 36934 → 82 [ACK] Seq=1 Ack=1481 Win=4381 Len=0
 7 0.155876    11.1.1.2  11.3.1.2    TCP      1520 82 → 36934 [<None>] Seq=1481 Win=4380 Len=1480
 8 0.155876    11.3.1.2  11.3.1.1    TCP      40 36934 → 82 [ACK] Seq=1 Ack=2961 Win=4381 Len=0
 9 0.260103    11.1.1.2  11.3.1.2    TCP      1520 [TCP Previous segment not captured] 82 → 36934 [<None>] Seq=4441 Win=5840 Len=1480
10 0.260103    11.3.1.2  11.3.1.1    TCP      40 [TCP Dup ACK 8#1] 36934 → 82 [ACK] Seq=1 Ack=2961 Win=4381 Len=0
11 0.262579    11.1.1.2  11.3.1.2    TCP      1520 [TCP Previous segment not captured] 82 → 36934 [<None>] Seq=7401 Win=7300 Len=1480
12 0.262579    11.3.1.2  11.3.1.1    TCP      40 [TCP Dup ACK 8#2] 36934 → 82 [ACK] Seq=1 Ack=2961 Win=4381 Len=0
13 0.863146    11.1.1.2  11.3.1.2    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=2961 Win=5840 Len=1480
14 0.863146    11.3.1.2  11.3.1.1    TCP      40 36934 → 82 [ACK] Seq=1 Ack=5921 Win=4381 Len=0
15 0.864384    11.1.1.2  11.3.1.2    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=4441 Win=5840 Len=1480
16 0.864384    11.3.1.2  11.3.1.1    TCP      40 [TCP Dup ACK 14#1] 36934 → 82 [ACK] Seq=1 Ack=5921 Win=4381 Len=0
17 0.866860    11.1.1.2  11.3.1.2    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=7401 Win=7300 Len=1480
18 0.866860    11.3.1.2  11.3.1.1    TCP      40 [TCP Dup ACK 14#2] 36934 → 82 [ACK] Seq=1 Ack=5921 Win=4381 Len=0
19 9.326482    11.1.1.2  11.3.1.2    TCP      1520 [TCP Retransmission] 82 → 36934 [<None>] Seq=5921 Win=7300 Len=1480
20 9.326482    11.3.1.2  11.3.1.1    TCP      40 36934 → 82 [ACK] Seq=1 Ack=8881 Win=4381 Len=0
21 19.143439   11.1.1.2  11.3.1.2    TCP      1520 82 → 36934 [<None>] Seq=8881 Win=2920 Len=1480
22 19.143439   11.3.1.2  11.3.1.1    TCP      40 36934 → 82 [ACK] Seq=1 Ack=10361 Win=4381 Len=0
23 19.247667   11.1.1.2  11.3.1.2    TCP      1520 82 → 36934 [<None>] Seq=10361 Win=2920 Len=1480
24 19.247667   11.3.1.2  11.3.1.1    TCP      40 36934 → 82 [ACK] Seq=1 Ack=11841 Win=4381 Len=0
25 19.351894   11.1.1.2  11.3.1.2    TCP      1520 82 → 36934 [<None>] Seq=11841 Win=4380 Len=1480
26 19.351894   11.3.1.2  11.3.1.1    TCP      40 36934 → 82 [ACK] Seq=1 Ack=13321 Win=4381 Len=0
27 19.352971   11.1.1.2  11.3.1.2    TCP      1320 82 → 36934 [<None>] Seq=13321 Win=4380 Len=1280
28 19.352971   11.3.1.2  11.3.1.1    TCP      40 36934 → 82 [ACK] Seq=1 Ack=14601 Win=4381 Len=0
29 19.453647   11.1.1.2  11.3.1.2    TCP      40 82 → 36934 [FIN] Seq=14601 Win=5840 Len=0
30 19.453647   11.3.1.2  11.3.1.1    TCP      40 36934 → 82 [FIN, ACK] Seq=1 Ack=14601 Win=4381 Len=0
31 19.453647   11.3.1.2  11.3.1.1    TCP      40 36934 → 82 [FIN] Seq=2 Win=4381 Len=0
32 19.554376   11.1.1.2  11.3.1.2    TCP      40 82 → 36934 [ACK] Seq=14602 Ack=3 Win=7300 Len=0
```

Figure 8-6: PCAP file at Wired Node 2

8.5 Inference

1. From **Figure 8-5** and **Figure 8-6**, we note that the packets with sequence number 2961, 5921, and 8881 are lost in the network.

2. After receiving three duplicate ACKs (in lines 13, 14 of **Figure 8-5**), TCP retransmits the lost packet with sequence number 2691 (in line 15 of **Figure 8-5**). After a timeout (see lines 17, 21, 22 and 23, lines 25 and 26), TCP retransmits the lost packet with sequence numbers 5921 and 8881.
3. TCP connection is terminated only after the complete file transfer is acknowledged.