LAB Manual

PART A

(PART A : TO BE REFFERED BY STUDENTS)

**Experiment No.4**

**A.1 Aim:**

To analyze the tcp packet and network statistics.

**A.2 Prerequisite:**

Role of packet sniffers in network security

**A.3 Outcome:**

**After successful completion of this experiment students will be able to** 1. Analyze tcp packet with different parameters.

2. Create network statistics using wireshark tool.

**A.4 Theory:**

View Packets You Have Captured



**TCP Analysis**

By default, Wireshark’s TCP dissector tracks the state of each TCP session and provides additional information when problems or potential problems are detected. Analysis is done once for each TCP packet when a capture file is first opened. Packets are processed in the order in which they appear in the packet list. You can enable or disable this feature via the “Analyze TCP sequence numbers” TCP dissector preference. For analysis of data or protocols layered on top of TCP (such as HTTP), see TCP Reassembly (in tutorial pdf).



Fig. TCP Analysis” packet detail items

TCP Analysis flags are added to the TCP protocol tree under “SEQ/ACK analysis”. Each flag is described below. Terms such as “next expected sequence number” and “next expected acknowledgement number” refer to the following”:

Next expected sequence number

The last-seen sequence number plus segment length. Set when there are no analysis flags and for zero window probes. This is initially zero and calculated based on the previous packet in the

same TCP flow. Note that this may not be the same as the tcp.nxtseq protocol field.

Next expected acknowledgement number

The last-seen sequence number for segments. Set when there are no analysis flags and for zero window probes.

Last-seen acknowledgment number

Always set. Note that this is not the same as the next expected acknowledgment number.

Last-seen acknowledgment number

Always updated for each packet. Note that this is not the same as the next expected

acknowledgment number.

TCP ACKed unseen segment

Set when the expected next acknowledgement number is set for the reverse direction and it’s less than the current acknowledgement number.

TCP Dup ACK <frame>#<acknowledgement number>

Set when all of the following are true:

• The segment size is zero.

• The window size is non-zero and hasn’t changed.

• The next expected sequence number and last-seen acknowledgment number are non-zero (i.e. the connection has been established).

• SYN, FIN, and RST are not set.

TCP Fast Retransmission

Set when all of the following are true:

• This is not a keepalive packet.

• In the forward direction, the segment size is greater than zero or the SYN or FIN is set.

• The next expected sequence number is greater than the current sequence number.

• We have more than two duplicate ACKs in the reverse direction.

• The current sequence number equals the next expected acknowledgement number.

• We saw the last acknowledgement less than 20ms ago.

TCP Keep-Alive

Set when the segment size is zero or one, the current sequence number is one byte less than the next expected sequence number, and any of SYN, FIN, or RST are set.

TCP Keep-Alive ACK

Set when all of the following are true:

• The segment size is zero.

• The window size is non-zero and hasn’t changed.

• The current sequence number is the same as the next expected sequence number.

• The current acknowledgement number is the same as the last-seen acknowledgement number.

• The most recently seen packet in the reverse direction was a keepalive.

• The packet is not a SYN, FIN, or RST.

TCP Out-Of-Order

Set when all of the following are true:

• This is not a keepalive packet.

• In the forward direction, the segment length is greater than zero or the SYN or FIN is set.

• The next expected sequence number is greater than the current sequence number.

• The next expected sequence number and the next sequence number differ.

• The last segment arrived within the Out-Of-Order RTT threshold. The threshold is either the value shown in the “iRTT” (tcp.analysis.initial\_rtt) field under “SEQ/ACK analysis” if it is present, or the default value of 3ms if it is not.

TCP Port numbers reused

Set when the SYN flag is set (not SYN+ACK), we have an existing conversation using the same addresses and ports, and the sequencue number is different than the existing conversation’s initial sequence number.

TCP Previous segment not captured

Set when the current sequence number is greater than the next expected sequence number.

TCP Spurious Retransmission

Checks for a retransmission based on analysis data in the reverse direction. Set when all of the following are true:

• The SYN or FIN flag is set.

• This is not a keepalive packet.

• The segment length is greater than zero.

• Data for this flow has been acknowledged. That is, the last-seen acknowledgement number has been set.

• The next sequence number is less than or equal to the last-seen acknowledgement number.

TCP Retransmission

Set when all of the following are true:

• This is not a keepalive packet.

• In the forward direction, the segment length is greater than zero or the SYN or FIN flag is set.

• The next expected sequence number is greater than the current sequence number.

TCP Window Full

Set when the segment size is non-zero, we know the window size in the reverse direction, and our segment size exceeds the window size in the reverse direction.

TCP Window Update

Set when the all of the following are true:

• The segment size is zero.

• The window size is non-zero and not equal to the last-seen window size.

• The sequence number is equal to the next expected sequence number.

• The acknowledgement number is equal to the last-seen acknowledgement number.

• None of SYN, FIN, or RST are set.

TCP ZeroWindow

Set when the receive window size is zero and none of SYN, FIN, or RST are set.

The window field in each TCP header advertises the amount of data a receiver can accept. If the receiver can’t accept any more data it will set the window value to zero, which tells the sender to pause its transmission. In some specific cases this is normal — for example, a printer might use a zero window to pause the transmission of a print job while it loads or reverses a sheet of paper. However, in most cases this indicates a performance or capacity problem on the receiving end. It might take a long time (sometimes several minutes) to resume a paused connection, even if the underlying condition that caused the zero window clears up quickly.

**Statistics**

Wireshark provides a wide range of network statistics which can be accessed via the Statistics menu. These statistics range from general information about the loaded capture file (like the number of captured packets), to statistics about specific protocols (e.g. statistics about the number of HTTP requests and responses captured).

General statistics

• **Capture File** Properties about the capture file.

• **Protocol Hierarchy** of the captured packets.

• **Conversations** e.g. traffic between specific IP addresses.

• **Endpoints** e.g. traffic to and from an IP addresses.

• **I/O Graphs** visualizing the number of packets (or similar) in time.

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the portal or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no portal access available)***

|  |  |
| --- | --- |
| Roll. No.: N049 | Name: Tarun Tanmay |
| Class MBATech CE, 3 Year | Batch: B3 |
| Date of Experiment: 29/07/2020 | Date of Submission: 29/7/2020 |
| Grade: | |

**B.1**

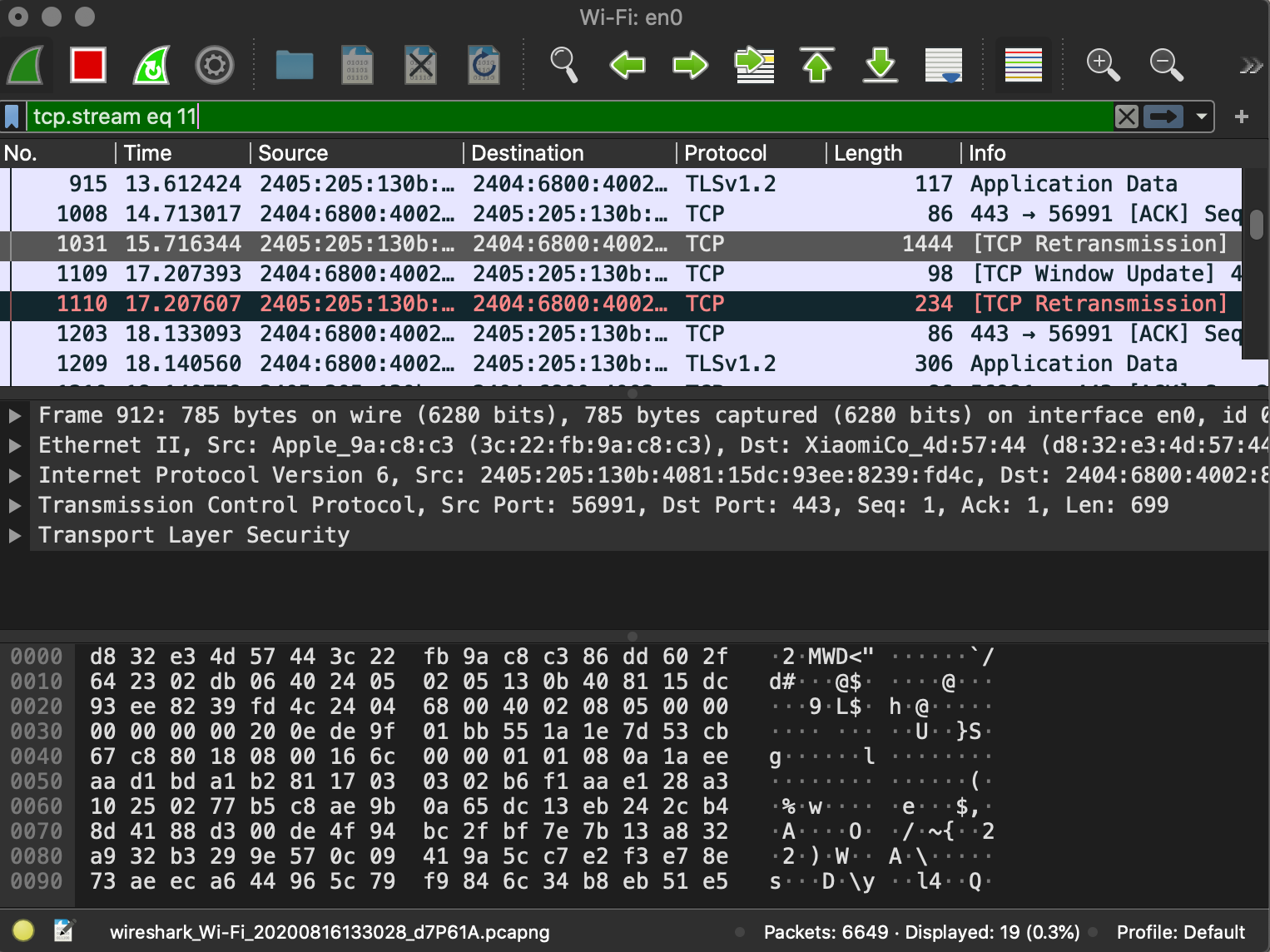
**Input and Output:**

**Input:**

Use wireshark tool for TCP packet analysis and network statistics

1. Identify different parameters with packet information and provide information.

Ex. – Source address, destination address, port, TTL, sequence number, sprotocol details etc.

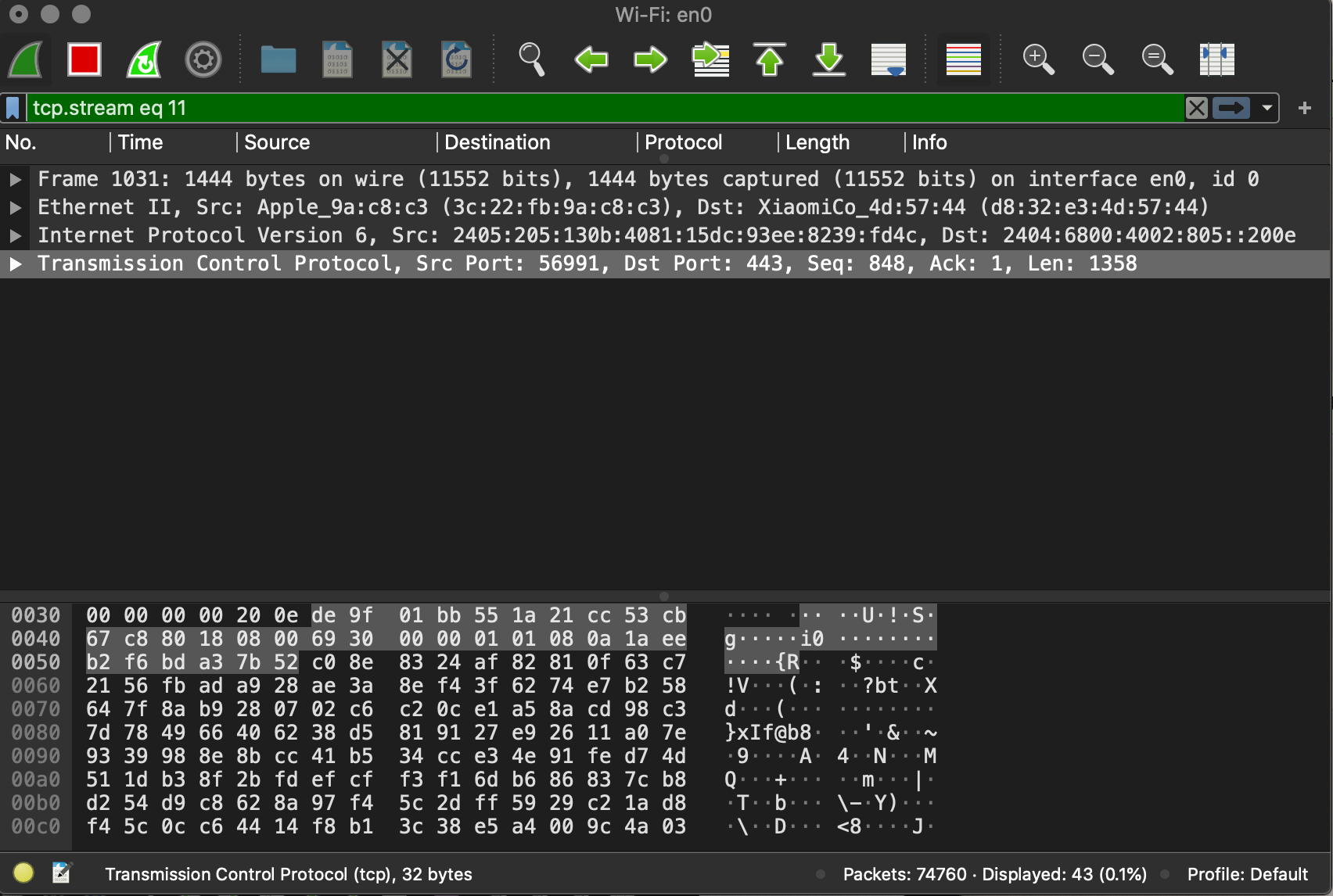


This image shows us the required details such as

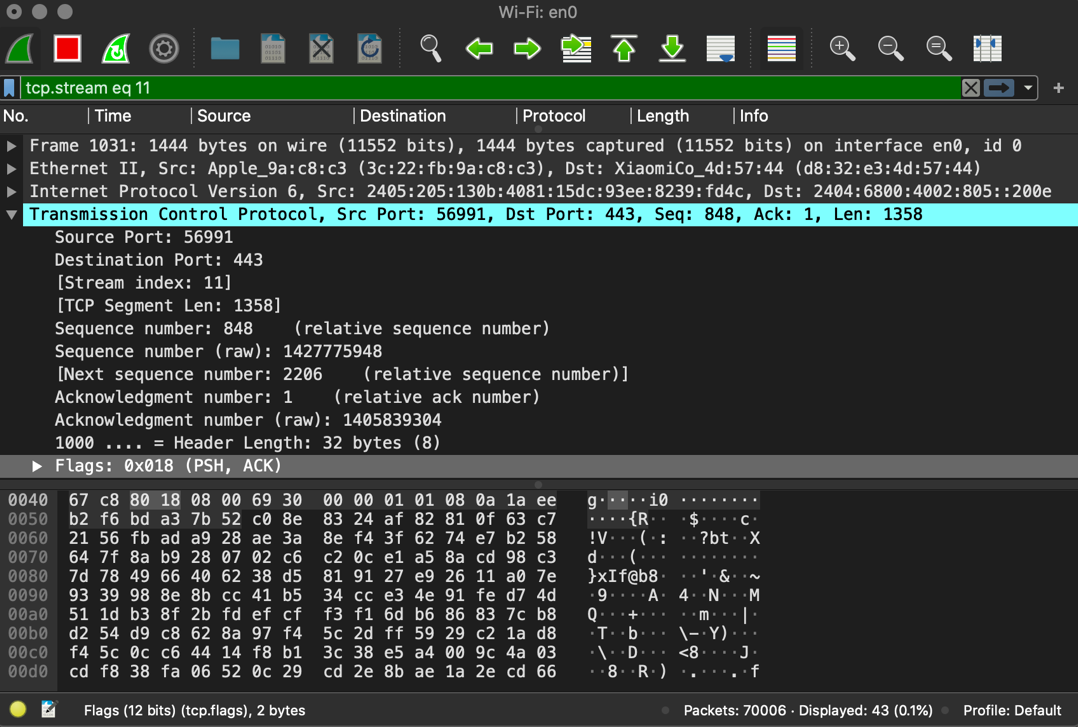
* Source address (**2405:w05:130b:..**)
* Destination address (**2404:6800:4002…**),
* Port: Source: **56991**
* Destination: **443**
* Sequence number (**848**) (**Relative Seq Number 0**)

1. Perform TCP Analysis of packet detail items and provide information about captured tcp packet flags.

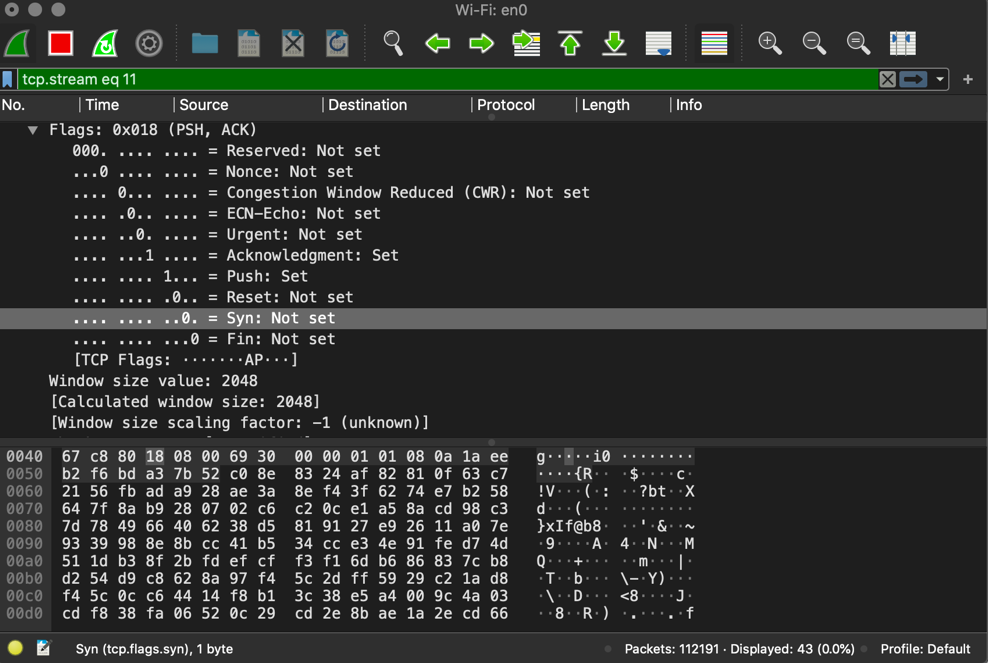
Carrying out the analysis on the following packet:



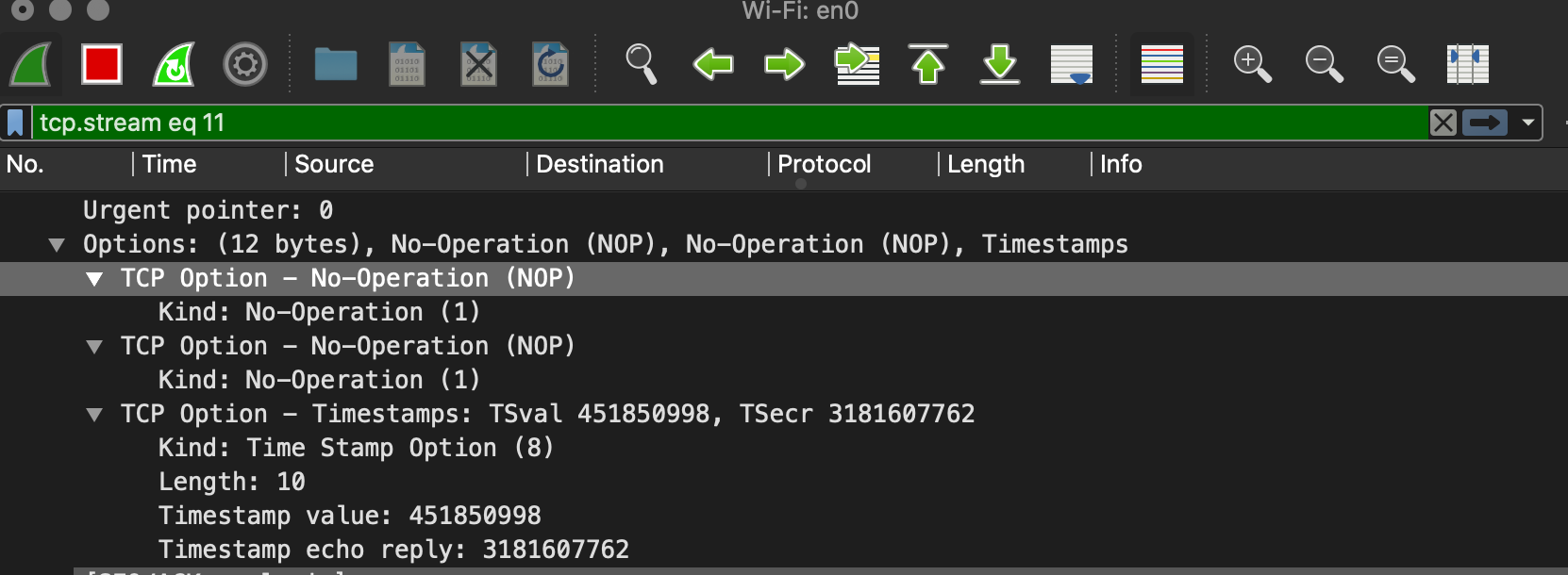
This window shows us the details of the required packet:



* The source and destination addresses are shown.
* The source and destination ports are shown.
* Relative sequence number is **0** indicating that it is the first of its stream. (This is done to make out analysis as humans easier.)
* As **ACK flag is set**, this means that the connection is set.
* Window Size = **2048** is the buffer size of my transmission.



We now view the options for further analysis:



1. Provide statistics (atleast two - Capture File, Protocol Hierarchy, Conversations, Endpoints, I/O Graphs) and observe the information.

**Endpoints:**

A screenshot of a computer screen

Description automatically generated

Packet analysis of the endpoints using the statistics tab provides information on Ethernet-6 packets connected and their addresses.

**I/O graph:**

**A screen shot of a person

Description automatically generated**

**B.2 Observations and learning:**

Via this practical, we carried out the analysis of packet details. WE understood how to do statistical analysis on the TCP packets. We also learnt about certain terminology and its interpretation when it comes to Packet Analysis.

**B.3 Conclusion:**

**After successful completion of this experiment I am able to**

1. Analyze tcp packet with different parameters.

2. Create network statistics using wireshark tool.

**B.3 Question of Curiosity:**

Q1: Give the uses of wireshark tool

Wireshark provides us with:

• Live capture and offline analysis.

• VoIP analysis.

• Decryption support for many protocols, including IPsec, ISAKMP, Kerberos.

• Deep inspection of multiple protocols like TCP, STUN.

Q2: List some other packet sniffing tools.

Ans2)

* Burp Suite
* DNSChef
* rtpbreak
* SniffJoke
* VoIPHopper
* SolarWinds
* PRTG
* Steel Central
* Fiddler
* Kismet
* NetworkMiner
* Tcpdump