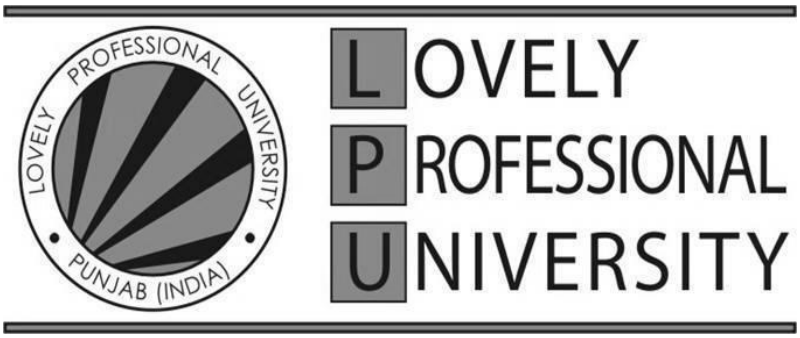


## ASSIGNMENT FOR CONTINUOUS ASSESSMENT-3

**Course Code:** INT301
**Course Name:** Open-Source Technologies

**Submitted by**

**Tanmoy Kumar Bera**

**Registration No: 11909768**
**Roll No: B50**
**Section: KE015**

**Program Name: B. Tech (CSE)**

# REPORT ON:

# Network Analysis for 10 Protocols and Data from Bluetooth and USB Using Wireshark

Under the Guidance of
Dr. Manjot Kaur
**Lovely Professional University, Phagwara**

# INDEX

# CHAPTER-1

## 1.1. INTRODUCTION:

In today's world, networks are essential for communication and information exchange between devices. As networks grow in complexity and size, it becomes increasingly important to analyse and understand the data being transmitted across them. Network analysis tools, such as Wireshark, provide a way to capture and analyse network traffic at a microscopic level, allowing for detailed investigation of the protocols and data being transmitted.

This project aims to use Wireshark to analyse a network at the microscopic level and investigate at least 10 protocols commonly used in networks. In addition, we will also read live data from Bluetooth and USB. The project will demonstrate the capabilities of Wireshark as a network analysis tool and provide insights into network traffic patterns, trends, and anomalies.

The report will begin with an overview of the project objectives and the methodology used to achieve them. We will then describe the target system being analysed and the assumptions and dependencies involved in the project. The main body of the report will consist of our findings from the analysis of the network traffic, including insights into the protocols being used and the data being transmitted. Finally, we will conclude with a summary of our findings, recommendations for future research, and reflections on the project overall.

Overall, this project will provide valuable insights into the use of network analysis tools for investigating network traffic and protocols. By gaining a deeper understanding of the data being transmitted across networks, we can optimize network performance, identify security threats, and improve the overall functioning of networked systems.

## 1.2. TYPES OF DIGITAL FORENSICS:

There are several types of digital forensics, including disk forensics, mobile forensics, network forensics, and memory forensics, among others.

Network forensics is a branch of digital forensics that focuses on the monitoring and analysis of network traffic in order to collect and preserve digital evidence. It involves the use of various tools and techniques to capture and analyze network data, such as packet captures, log files, and network flow data. Network forensics can be used to investigate various types of cybercrimes, including hacking, malware attacks, and data breaches.

Network forensics can be used to perform various tasks, such as identifying unauthorized access to a network, tracing the source of a cyber attack, and determining the extent of damage caused by an attack. It involves the use of specialized tools, such as network sniffers and protocol analyzers, to capture and analyze network traffic in real-time or from stored data. By analyzing the captured data, investigators can identify potential vulnerabilities, misconfigurations, or malicious activity on a network, and take appropriate actions to prevent further damage.

In summary, network forensics is a vital aspect of computer forensics, as it enables investigators to gather evidence and determine the scope of a cybercrime by analyzing network traffic.

## 1.3. OBJECTIVES OF THE PROJECT:

Listed below are the objectives of the project:

i) To gain practical experience with Wireshark as a network analysis tool.
ii) To analyse network traffic at the microscopic level and identify patterns, trends, and anomalies in the data.
iii) To investigate at least 10 protocols commonly used in networks and understand their purpose and significance.
iv) To understand how Bluetooth and USB data can be captured and analyzed using Wireshark.
v) To demonstrate the importance of network analysis and protocol investigation in maintaining network security and optimizing network performance.
vi) To provide recommendations for future research or improvements to the project.

## 1.4. METHODOLOGY:

To carry out this project, we followed a methodology that involved several steps. First, we set up Wireshark to capture network traffic, and then we identified the network that we wanted to analyse. We collected data from the network over a period, ensuring that we captured a sufficient amount of traffic for analysis. We then used Wireshark's filtering and display options to isolate specific protocols, and we investigated at least 10 protocols that were commonly used in the network. Additionally, we read live data from Bluetooth and USB using Wireshark to further investigate the data being transmitted across the network. We used a variety of techniques to analyse the data, including looking for patterns and trends, identifying anomalies and outliers, and comparing data across different protocols and devices. Finally, we drew conclusions based on our analysis and made recommendations for future research or improvements to the project.

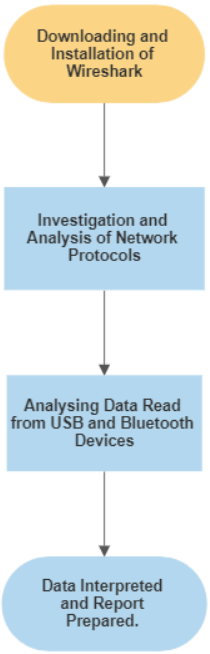


Fig 1.1. Block Diagram describing the methodology

## 1.5. SCOPE OF THE PROJECT:

The scope of this project is to use Wireshark to analyze a network at the microscopic level, investigate at least 10 protocols commonly used in networks, and read live data from Bluetooth and USB. Through this project, we aim to demonstrate the capabilities of Wireshark as a network analysis tool and to provide insights into network traffic patterns, trends, and anomalies. The project also aims to highlight the importance of network analysis and protocol investigation in maintaining network security and optimizing network performance. While this project focuses on a specific network and set of protocols, the methodologies and techniques used can be applied to other networks and protocols, making the project relevant to a wide range of network analysis scenarios.

# CHAPTER-2

## 2.1. TARGET SYSTEM DESCRIPTION:

The target system for this project is a network that uses a variety of protocols for communication between devices. The network may consist of a combination of wired and wireless devices, such as desktop computers, laptops, smartphones, tablets, printers, and servers. The protocols used in the network may include common ones like TCP/IP, HTTP, DNS, as well as others specific to the devices or applications used in the network. The network may also utilize Bluetooth and USB connections for data transfer between devices. The aim of the project is to analyse the traffic in this network using Wireshark and investigate at least 10 protocols commonly used in networks, as well

as read live data from Bluetooth and USB. Through this analysis, we aim to gain insights into the patterns, trends, and anomalies in the network traffic, and to understand the purpose and significance of the protocols used in the network.

## 2.2. ASSUMPTIONS AND DEPENDENCIES:

**Assumptions:**

i) The network being analysed is functioning correctly and is not experiencing any significant issues or errors.
ii) The devices in the network are all operating properly and are not experiencing any hardware or software issues that would affect their network communication.
iii) The protocols used in the network are all functioning as intended and are not experiencing any errors or issues.
iv) The devices in the network are authorized to communicate with each other, and there are no security restrictions that would prevent data capture and analysis.

**Functional Dependencies:**

i) Wireshark must be properly installed and configured to capture network traffic.
ii) The network being analysed must be accessible to the Wireshark software.
iii) The devices in the network must be properly configured to communicate with each other and with Wireshark.
iv) The protocols being investigated must be properly identified and isolated using Wireshark's filtering and display options.

**Non-functional Dependencies:**

i) The performance of the Wireshark software may be affected by the amount of network traffic being captured, the number of protocols being analysed, and the specifications of the computer running Wireshark.
ii) The accuracy and reliability of the data captured and analysed may be affected by external factors such as network interference or environmental conditions.
iii) The privacy and security of the network being analysed must be respected, and any sensitive data captured during the analysis must be handled appropriately to prevent unauthorized access or disclosure.

## 2.3. DATASET USED:

The dataset used includes the network traffic captured by Wireshark. This includes the packets transmitted and received by the devices in the network, as well as the metadata associated with each packet. Wireshark captures the data in a format that is specific to the software, but it can be exported and saved in various formats, including pcap, which is a standard format for packet capture files. The dataset will be used for analysis and investigation of at least 10 protocols commonly used in networks, as well as for reading live data from Bluetooth and USB. The dataset may be anonymized or modified as necessary to protect the privacy and security of the network being analysed.

# CHAPTER – 3

## 3.1. INSTALLATION AND UNDERSTANDING WIRESHARK:

To install Wireshark we can first download the .exe file from : https://www.wireshark.org/download.html
It is then installed into the system.

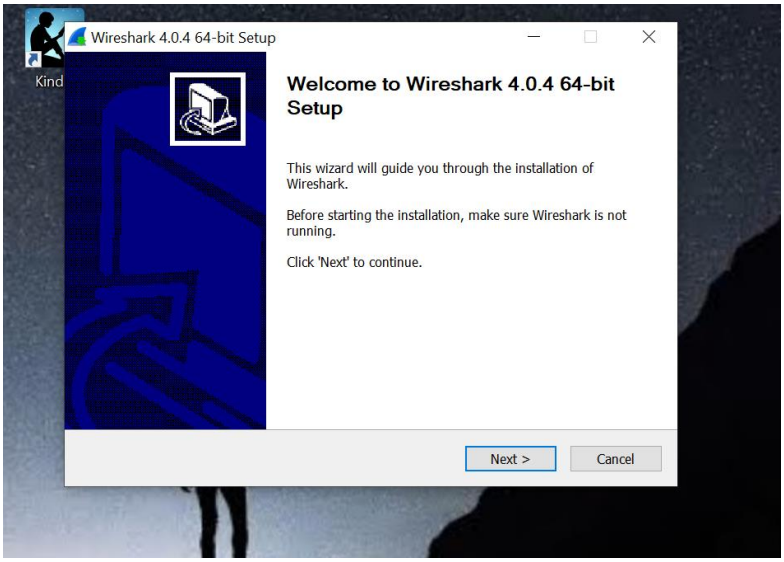


Fig. 2.1 First step in installation of wireshark

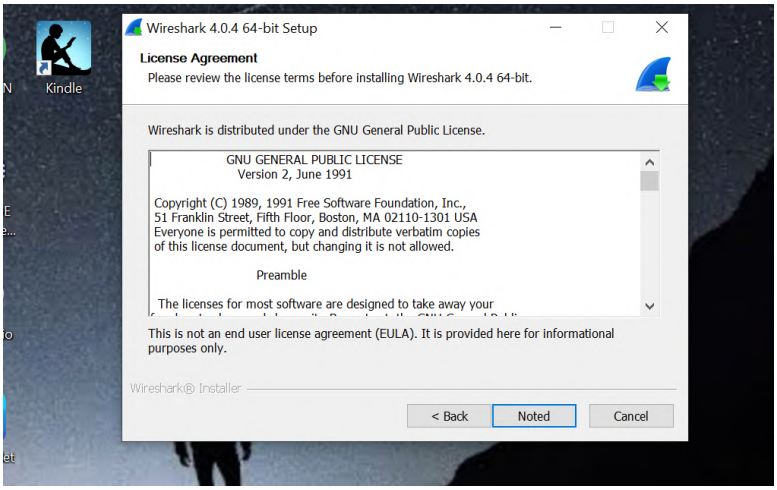


Fig. 2.2 Final step in installation of Wireshark

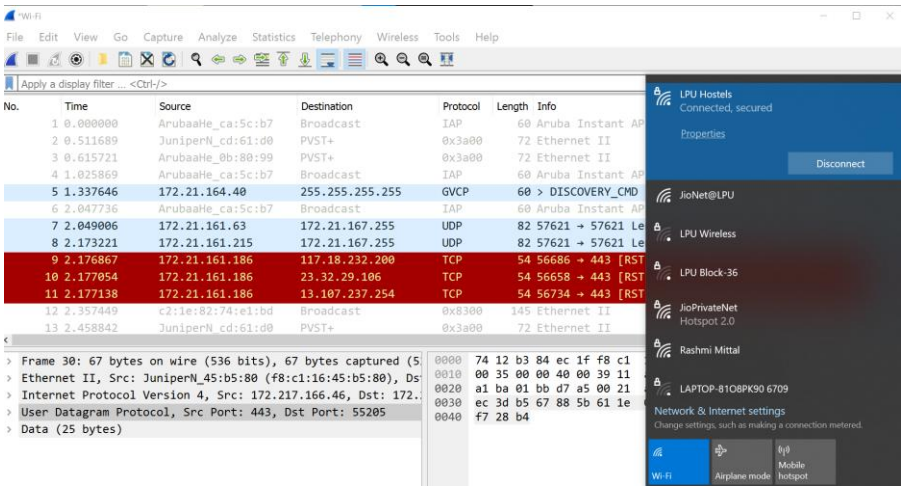


Fig. 2.3 “LPU Hostels” network connected for NETWORK ANALYSIS

## 3.2. USAGE OF WIRESHARK:

The Wireshark software window is shown below, and all the processes on the network are carried within this screen only.

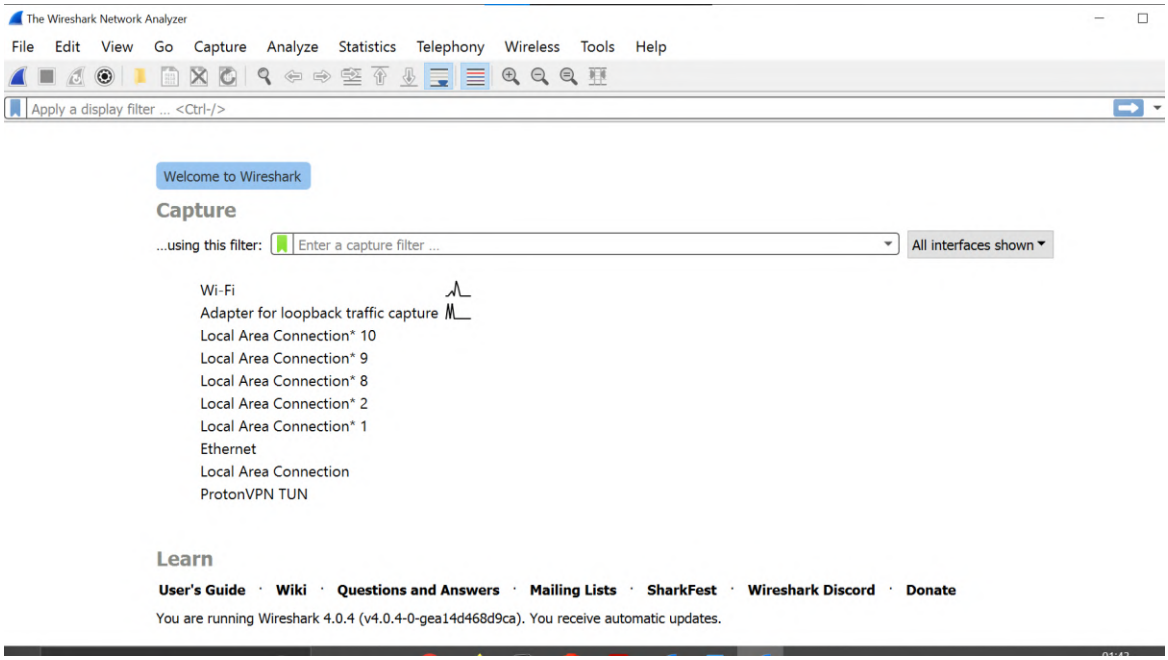


Fig . 3.1 . List of Interfaces displayed

The list presents various options for interfaces. The number of interface options available will be displayed. Once an option is selected, it will capture all traffic. For instance, when we select Wi-Fi option from the list shown above, a new window will open, displaying all current traffic on the network. The following image illustrates the live capture of packets, showing what Wireshark will look like:

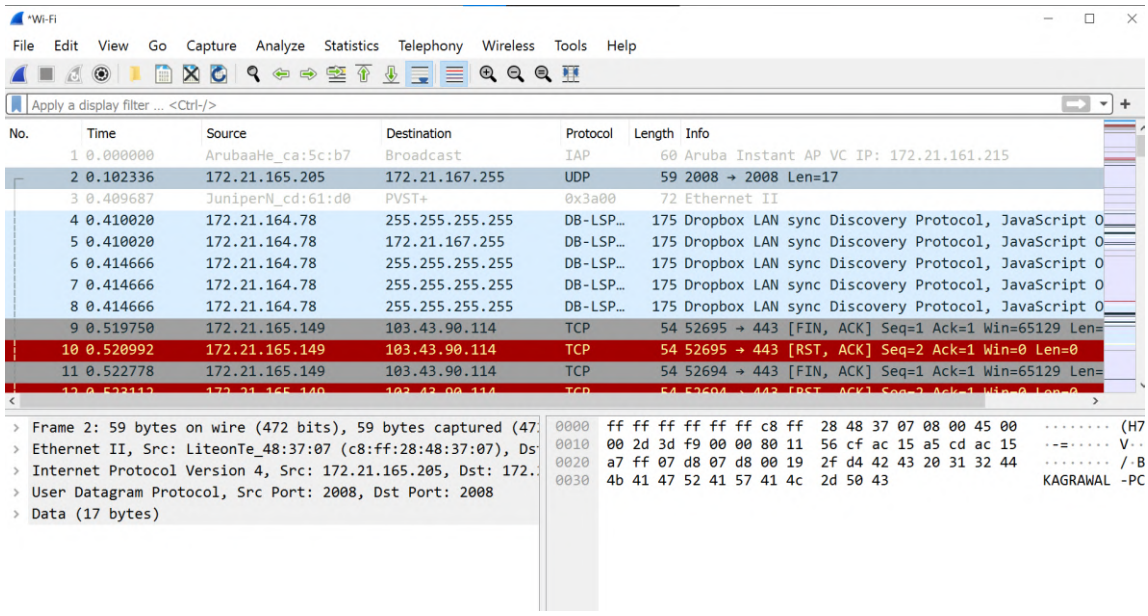


Fig. 3.2. Network Packet Analysis in wireshark

We will keep listening to all the data packets and gather a significant amount of data. If we wish to view a specific set of data, we can click on the red button. The traffic will come to a halt, allowing us to take note of the essential parameters such as time, source, destination, the protocol being used, length, and information. To view more in-depth details, we can click on the specific address, and an extensive range of information will be displayed below it. This way, we can access and examine the data we need without any hassle.

The Wireshark interface is divided into five main sections:

Menu bar: located at the top of the window, this section contains various options such as File and Capture. The Capture menu allows you to start the packet capturing process, while the File menu is used to open and save a capture file.

Packet listing window: this section displays the captured packets in a table format, showing packet number, time, source, destination, protocol, length, and info. The packet list can be sorted by clicking on the column headers.

Packet header detail window: this section provides detailed information about the components of each packet. The protocol information can be expanded or minimized as per the user's requirement.

Packet contents window: located at the bottom of the screen, this section displays the content of the captured packets in both ASCII and hexadecimal format.

Filter field: located at the top of the screen, this section allows users to filter the captured packets based on any component. For example, by applying a filter for HTTP protocol, only packets with HTTP as the protocol will be displayed on the screen.

In summary, Wireshark's interface consists of a menu bar, packet listing window, packet header detail window, packet contents window, and filter field. These sections enable users to capture and analyze packets efficiently.

To access more information on our network, the statistics drop-down menu proves to be an incredibly helpful tool. At the top of the screen, we can easily locate the statistics menu, which offers a range of metrics, including size and timing information, as well as plotted charts and graphs. By applying display filters, we can also narrow down the most relevant information.

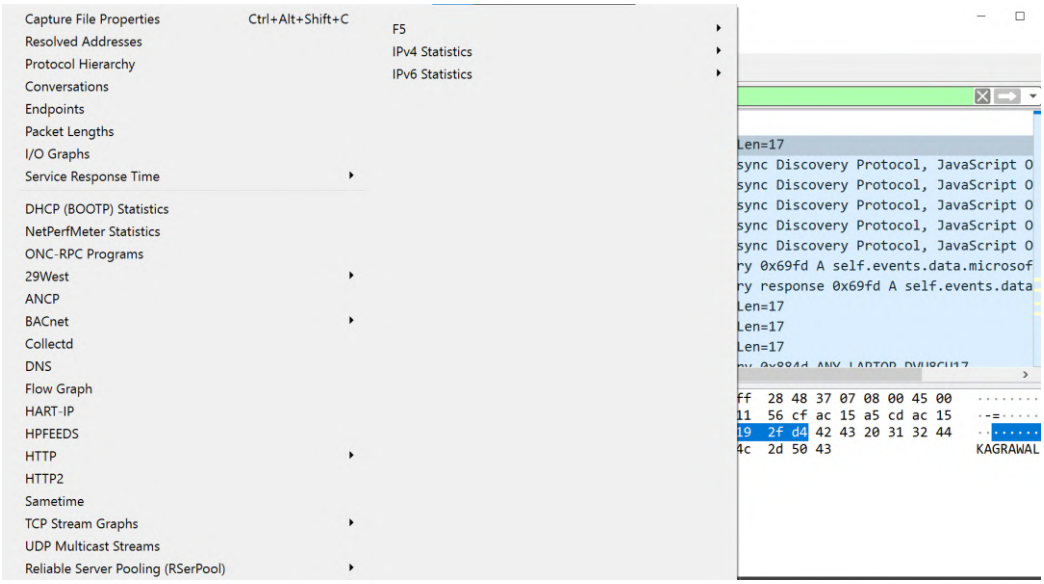


Fig. 3.3 The image displays Wireshark statistics menu:

The following are the main categories of statistics menu selections available:

Protocol Hierarchy: This option presents a complete table of all captured protocols, along with active display filters located at the bottom.

Conversations: This feature reveals the network conversation between two endpoints by displaying the exchange of traffic from one IP address to another.

Endpoints: This category displays a list of endpoints, which are locations where protocol traffic of a specific protocol layer comes to an end.

IO Graphs: This option displays user-specific graphs that visualize the number of packets throughout the data exchange.

RTP Statistics: This feature allows the user to save the content of an RTP audio stream directly to an Au-file.

Service Response Time: This section displays the response time between a request and the network's response.

TcpPduTime: This category displays the time taken to transfer data from a Protocol Data Unit, which can be used to find TCP retransmissions.

VoIP Calls: This option shows VoIP calls obtained from live captures.

Multicast Stream: This feature detects multicast streams and measures the size of bursts and the output buffers of certain speeds.

## 3.3. INVESTIGATING PROTOCOLS ON THE NETWORK AND ANALYSING THEM:

### 1) Transmission Control Protocol:

TCP (Transmission Control Protocol) is a widely used transport layer protocol that provides reliable, ordered, and error-checked delivery of data between applications running on different devices. Here we analysed the TCP in the network traffic captured by Wireshark. By analysing TCP traffic, we can gain insights into the reliability and performance of network connections, as well as identify any errors or congestion that may be affecting the flow of data. This information can be used to optimize network performance and improve the overall functioning of networked systems.

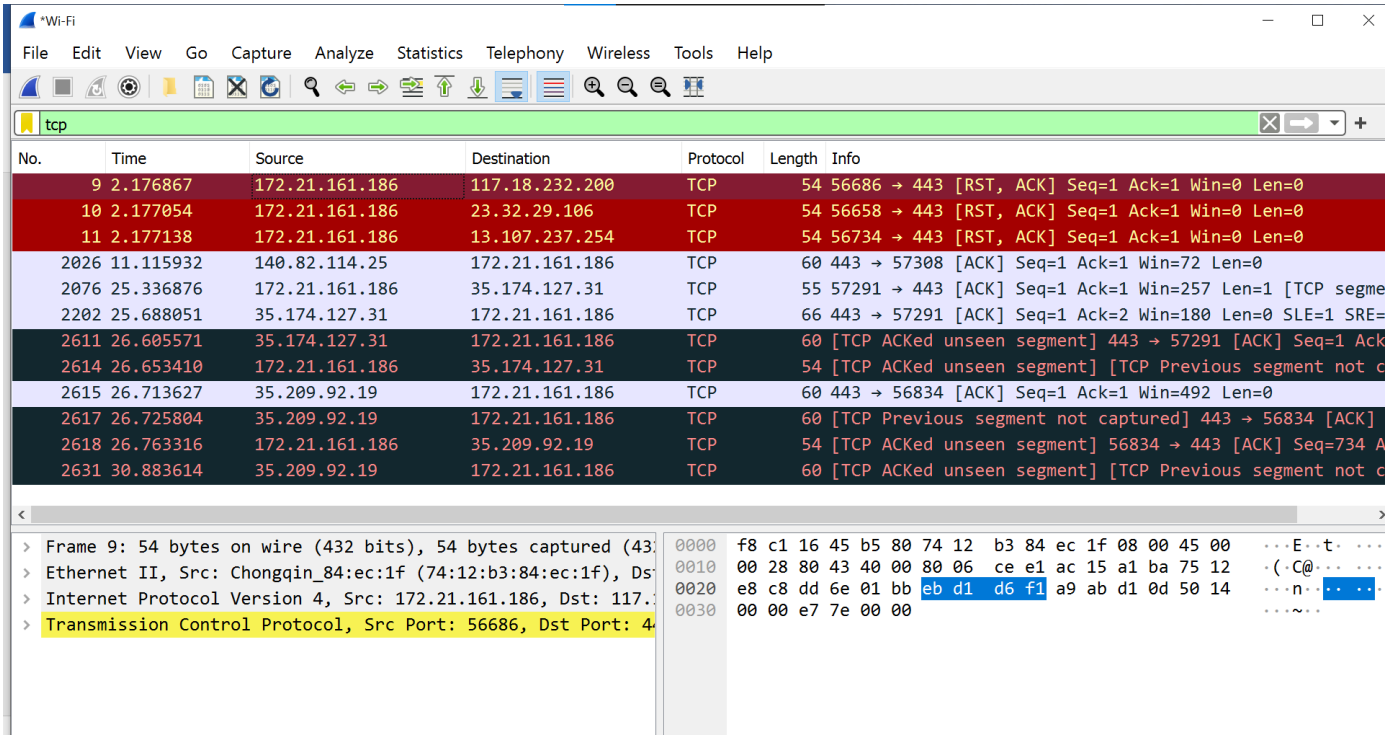


Fig.4.1 . Filtering TCP protocol through Wireshark

The I/O graph displays a graph of network traffic over time, allowing for a quick and easy visual analysis of packet flow, data transfer rates, and other network performance metrics. When analysing TCP traffic using Wireshark's I/O graph, the graph can be configured to show various aspects of TCP performance, such as the number of TCP segments sent and received, the round-trip time (RTT) for each segment, and the congestion window size. The I/O graph can also be used to identify any abnormalities in TCP traffic, such as retransmitted packets or sudden drops in throughput, which may indicate network issues or congestion.

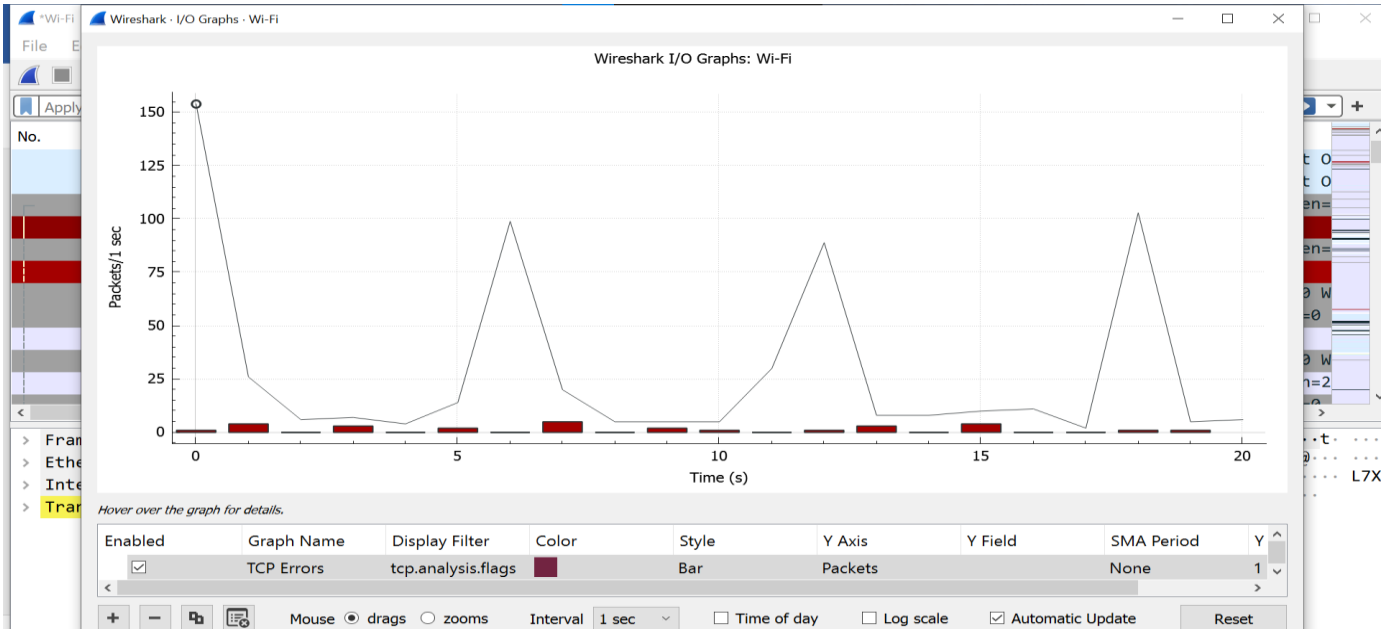


Fig.4.2. I/O Graph depicting TCP

## 2)User Diagram Protocol(UDP):

UDP (User Datagram Protocol) is a lightweight transport layer protocol that is commonly used in applications where low latency and high throughput are more important than reliable data transfer. One of the challenges of using UDP is that it does not provide the same reliability and error checking as TCP. Therefore, analyzing UDP traffic requires a different approach than TCP, as we must take into account the potential for lost or out-of-order packets. Wireshark can be used to analyze UDP traffic and identify any issues, such as packet loss, that may be affecting the performance of UDP-based applications.

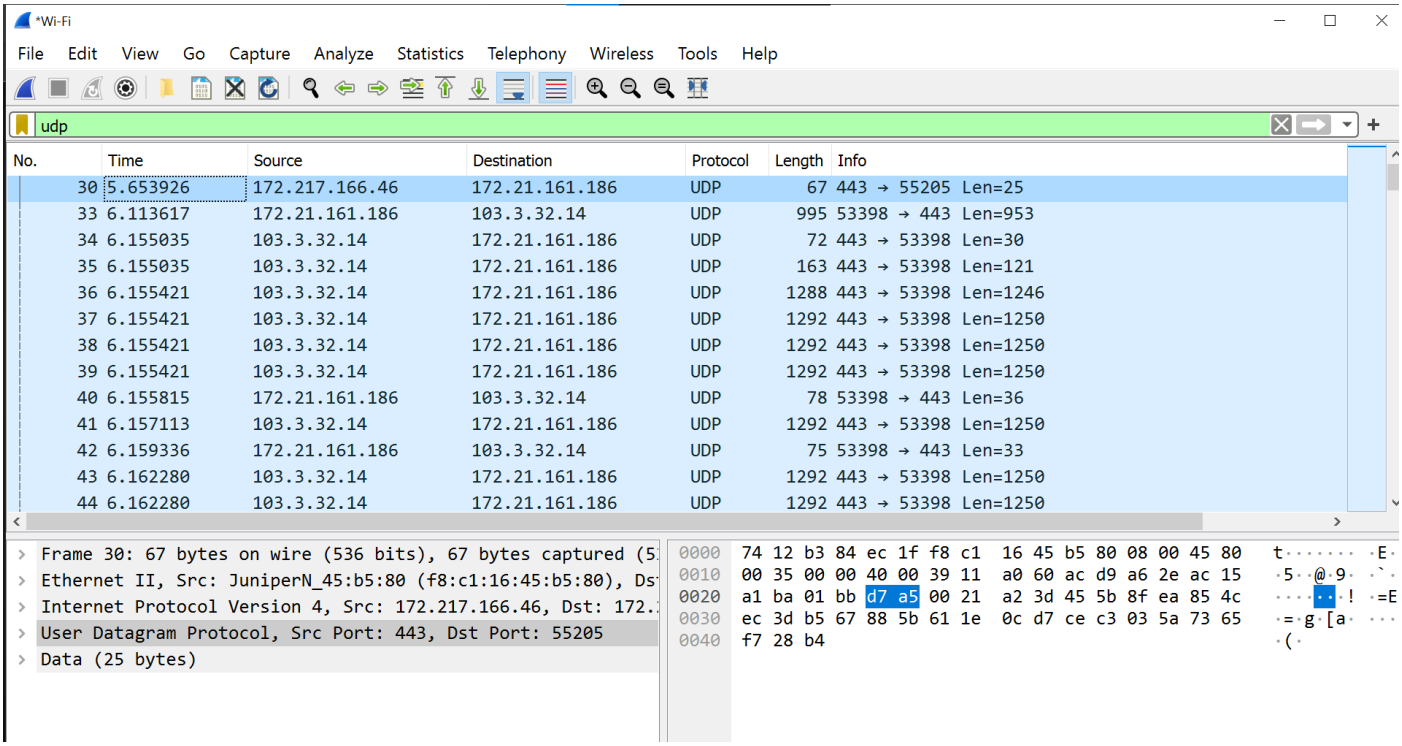


Fig.4.3 . Filtering UDP protocol through Wireshark

When analysing UDP traffic using the I/O graph, the graph can be configured to display the number of packets sent and received, as well as the number of bytes sent and received, over a given time period. Additionally, the I/O graph

can be used to identify any patterns in the data, such as spikes or drops in packet or byte counts, which may indicate issues or anomalies in the data being transmitted. One of the challenges of analysing UDP traffic using the I/O graph is that UDP packets can be lost or delivered out of order without detection, since there is no reliability mechanism built into the protocol. Therefore, when analysing UDP traffic using the I/O graph, it is important to take into account the potential for packet loss and reordering.

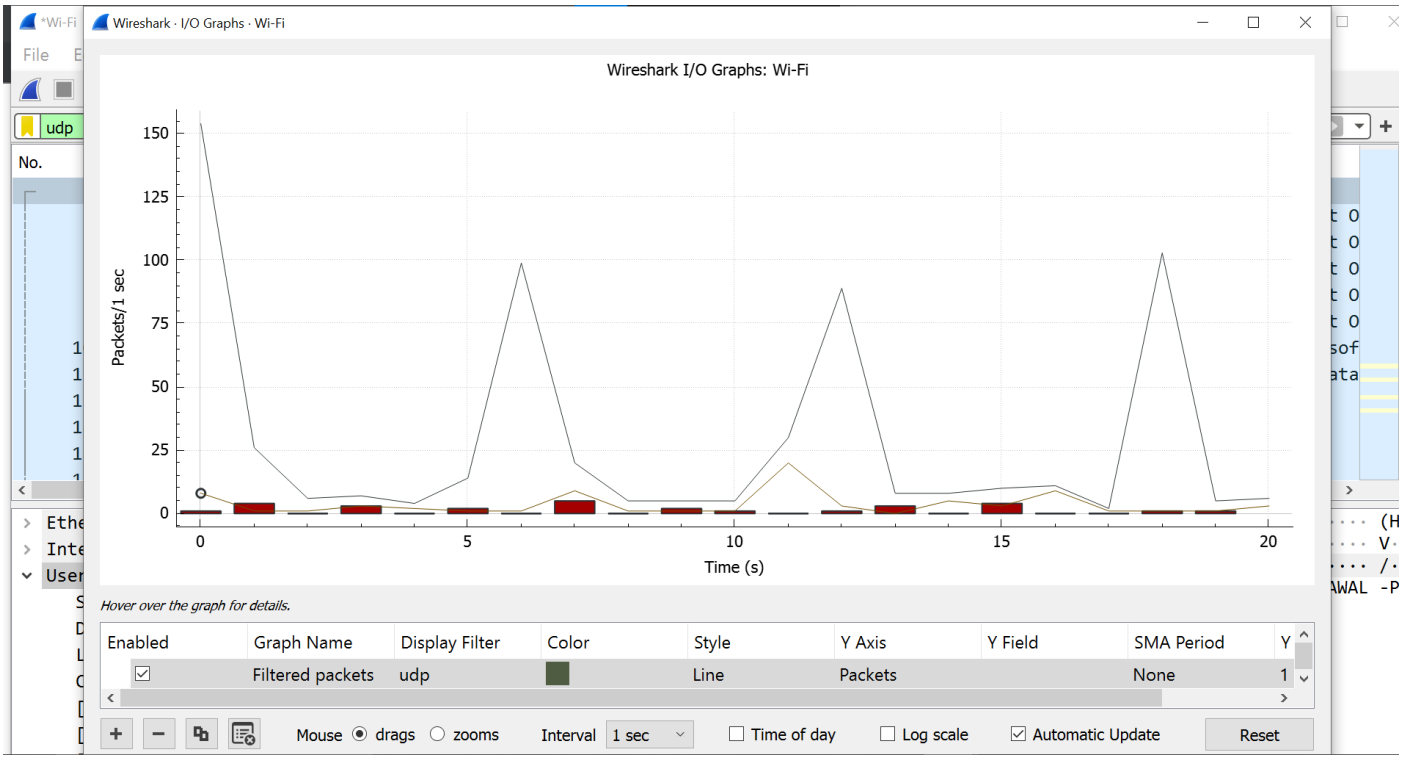


Fig.4.4. I/O Graph depicting UDP

### 3) Hypertext Transfer Protocol (HTTP):

HTTP (Hypertext Transfer Protocol) is a widely used application layer protocol for transmitting data over the internet. One of the key features of HTTP is that it uses a client-server architecture, where clients send requests to servers and servers respond with data. This request-response model is used extensively in web-based applications and forms the basis for the World Wide Web.Wireshark can be used to analyze HTTP traffic and provide details on the contents of the requests and responses being transmitted. This can include information such as the headers, message bodies, and other metadata associated with the request or response.

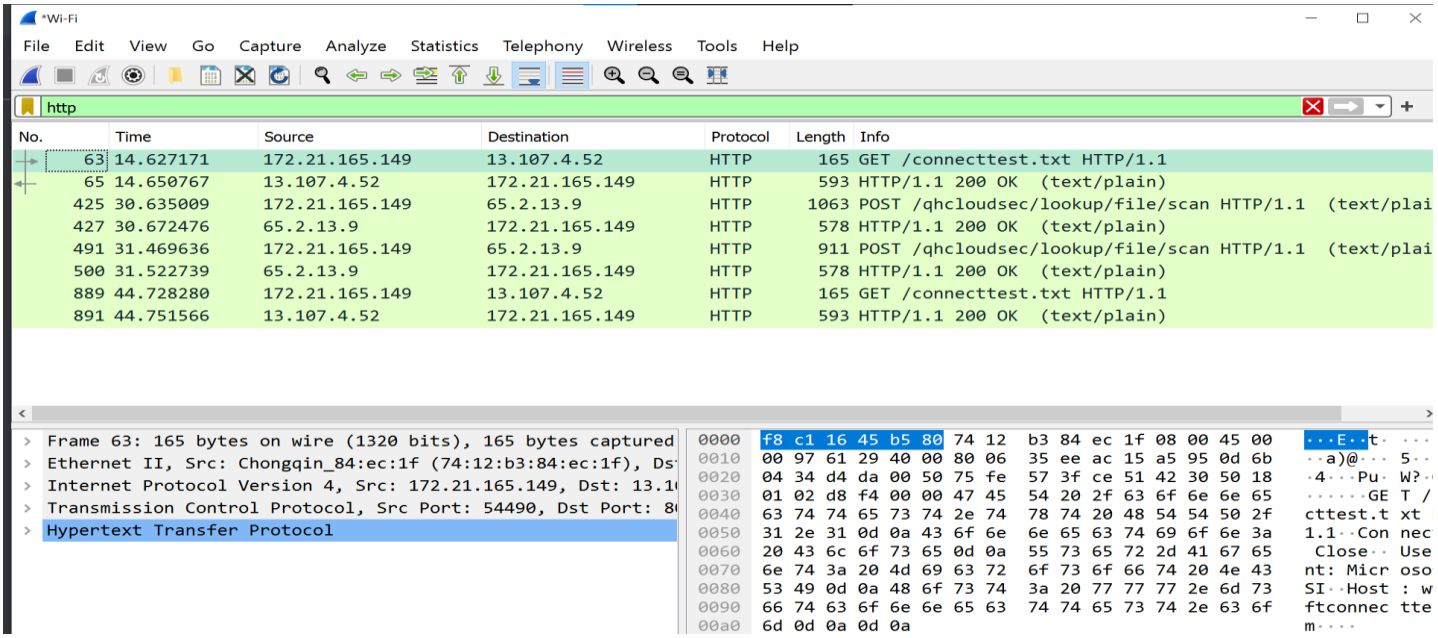


Fig.4.5 . Filtering HTTP protocol through Wireshark

When analyzing HTTP traffic using the I/O graph, the graph can be configured to display the number of packets sent and received, as well as the number of bytes sent and received, over a given time period. The I/O graph can also be used to analyze the response times of HTTP requests. By using the "Time delta from previous displayed packet" option in the I/O graph, we can see the time it takes for each HTTP request to receive a response. This information can be useful for identifying slow-loading pages or other performance issues in web-based applications.

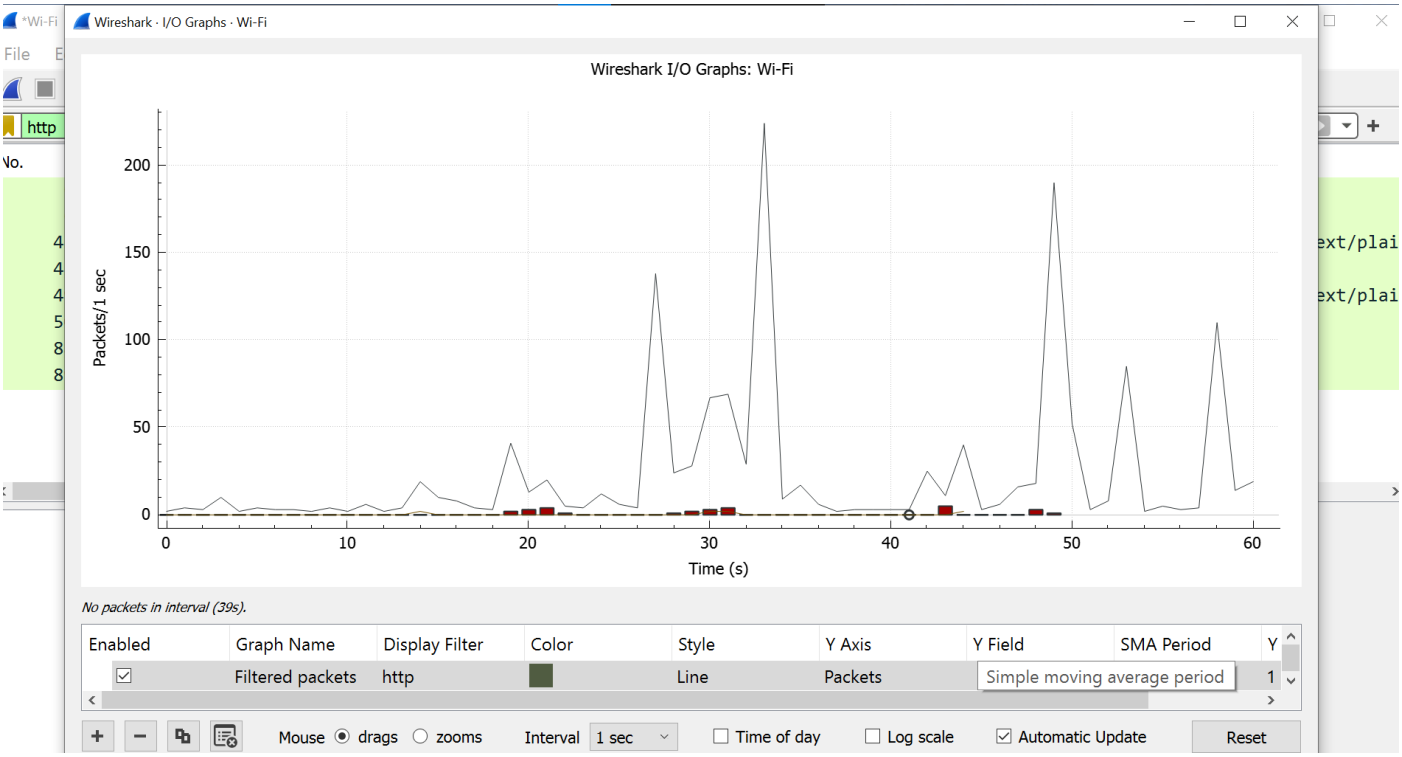


Fig.4.6. I/O Graph depicting HTTP

**4) DNS Protocol:**

DNS (Domain Name System) is a protocol used to resolve domain names into IP addresses. By analyzing DNS traffic, we can gain insights into the domain names being accessed and the corresponding IP addresses, as well as identify any issues or anomalies in the DNS resolution process. One of the key features of DNS is that it uses a hierarchical naming system, where domain names are organized into a tree-like structure. This naming system allows for efficient and scalable resolution of domain names to IP addresses. Wireshark can be used to analyze DNS traffic and provide details on the contents of the DNS queries and responses being transmitted. This can include information such as the domain names being queried, the IP addresses of the DNS servers, and other metadata associated with the DNS resolution process.

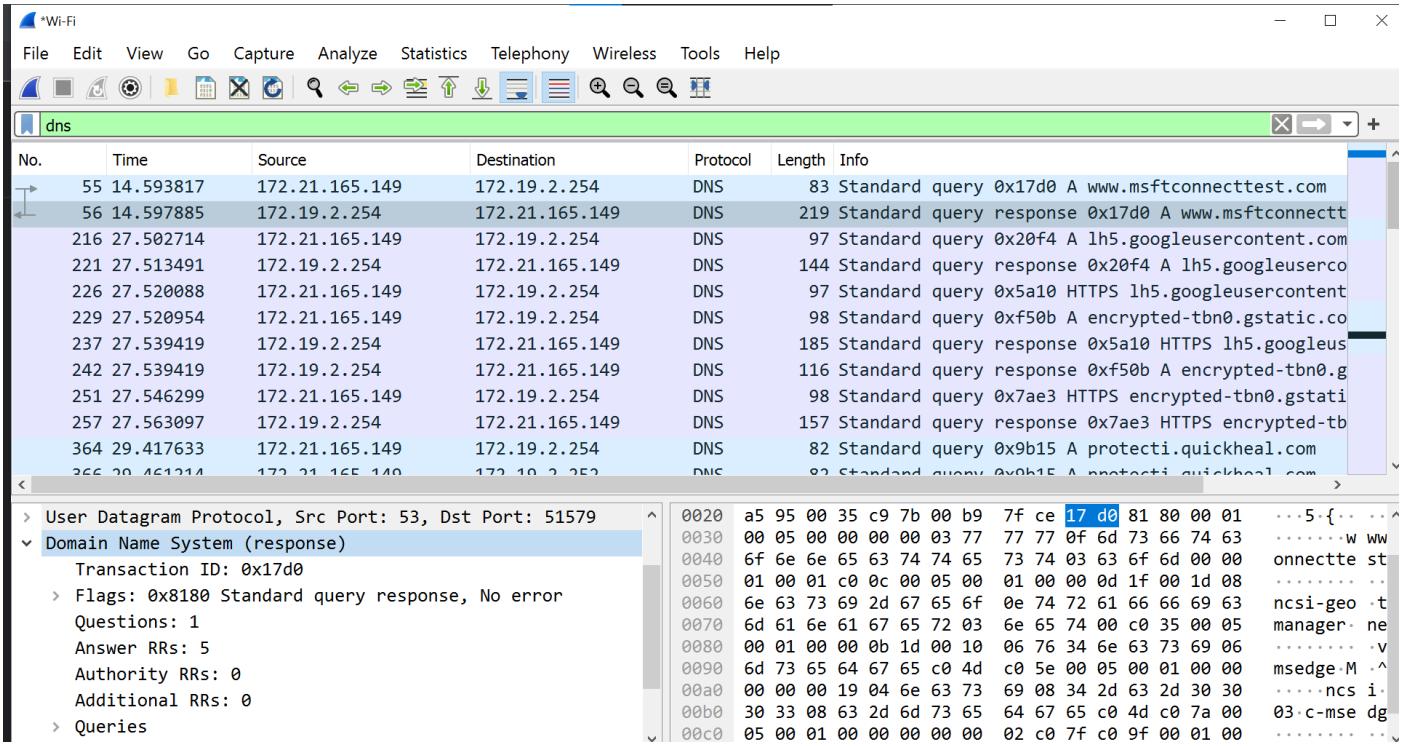


Fig.4.7 . Filtering DNS protocol through Wireshark

The I/O graph can be used to identify any patterns in the data, such as spikes or drops in packet or byte counts, which may indicate issues or anomalies in the DNS resolution process. For example, if there are sudden spikes in DNS request count, it could indicate that the DNS server is experiencing high traffic or that there is an issue with the DNS resolution process. It can also be used to analyze the response times of DNS requests. By using the "Time delta from previous displayed packet" option in the I/O graph, we can see the time it takes for each DNS request to receive a response. This information can be useful for identifying slow or unresponsive DNS servers, or for troubleshooting DNS resolution issues.

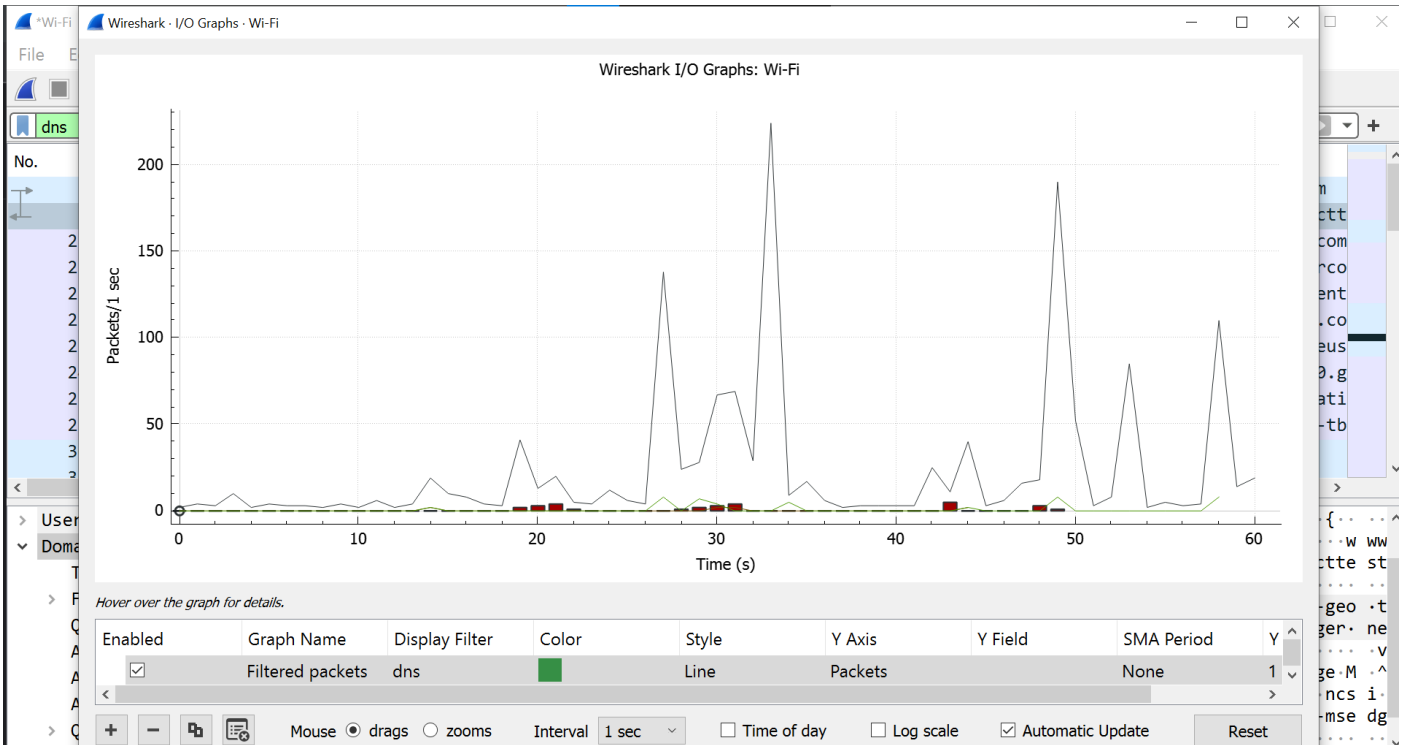


Fig.4.8. I/O Graph depicting DNS

## 5) NetBIOS Name Service(NBNS):

The NBNS (NetBIOS Name Service) protocol is a component of the NetBIOS protocol suite used in Microsoft Windows networks. The protocol is used to resolve NetBIOS names to IP addresses, and vice versa. Analyzing NBNS traffic can provide insights into the NetBIOS names being used in the network, and the corresponding IP addresses. By analyzing NBNS traffic using Wireshark, we can identify any issues or anomalies in the NetBIOS name resolution process. One of the challenges of using NBNS is that it uses a flat naming system, which can result in naming conflicts and make it difficult to manage a large number of systems. However, the protocol is still widely used in legacy Windows networks and can be useful for identifying and troubleshooting issues with NetBIOS name resolution.

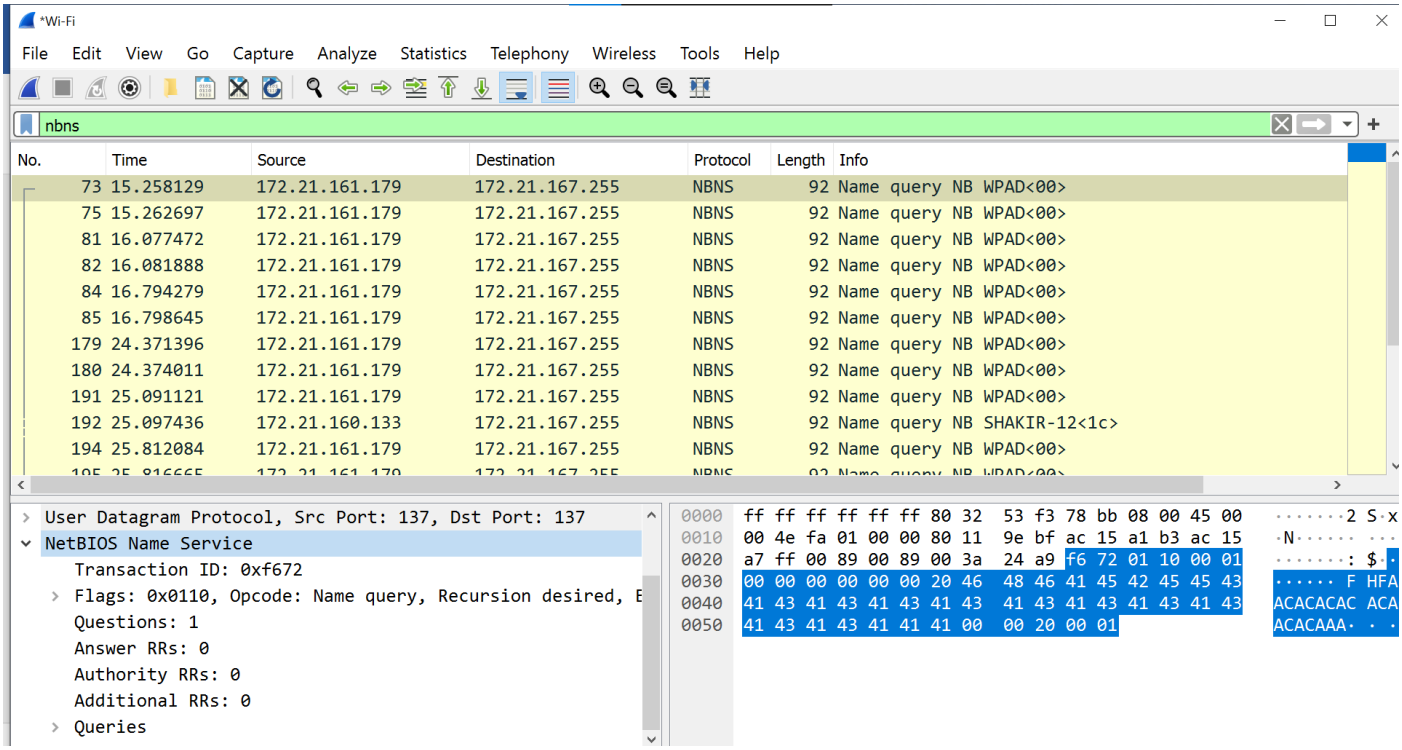


Fig.4.9 . Filtering NBNS protocol through Wireshark

The I/O graph can be used to identify any patterns in the data, such as spikes or drops in packet or byte counts, which may indicate issues or anomalies in the NBNS name resolution process. For example, if there are sudden spikes in NBNS request count, it could indicate that the network is experiencing high traffic or that there is an issue with the NetBIOS name resolution process. Additionally, the I/O graph can be used to analyse the response times of NBNS requests. By using the "Time delta from previous displayed packet" option in the I/O graph, we can see the time it takes for each NBNS request to receive a response. This information can be useful for identifying slow or unresponsive NBNS servers, or for troubleshooting NetBIOS name resolution issues.

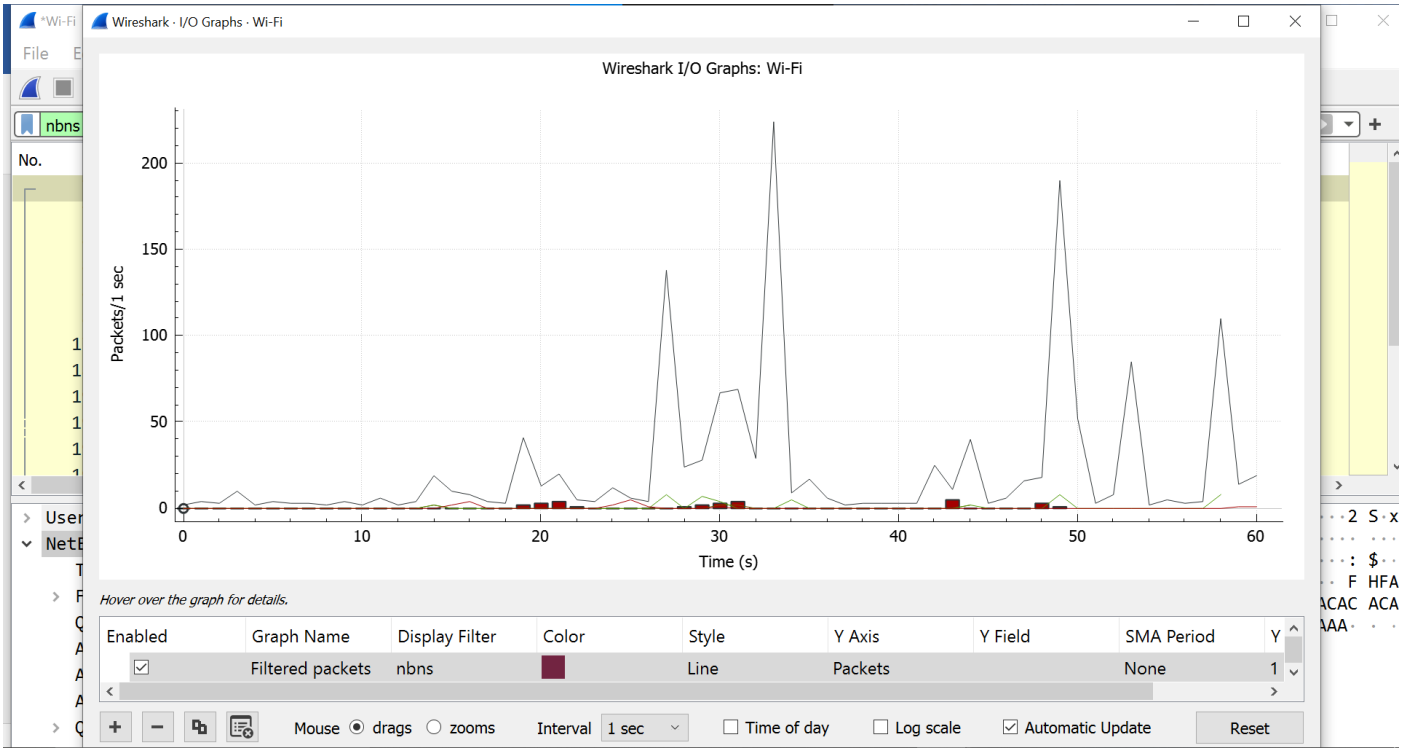


Fig.4.10. I/O Graph depicting NBNS

## 6) LLMNR (Link-Local Multicast Name Resolution):

The LLMNR (Link-Local Multicast Name Resolution) protocol is a Microsoft Windows protocol used for name resolution on local networks when DNS is not available. Analyzing LLMNR traffic can provide insights into the local name resolution process in Windows networks. LLMNR is often used in conjunction with other name resolution protocols, such as DNS and NBNS, and can be useful for identifying and troubleshooting issues with local name resolution. One of the advantages of using LLMNR is that it allows systems to resolve names of other systems on the same local network segment, without the need for a centralized DNS server. However, LLMNR does not provide the same level of security as DNS, and can be vulnerable to spoofing and other attacks.

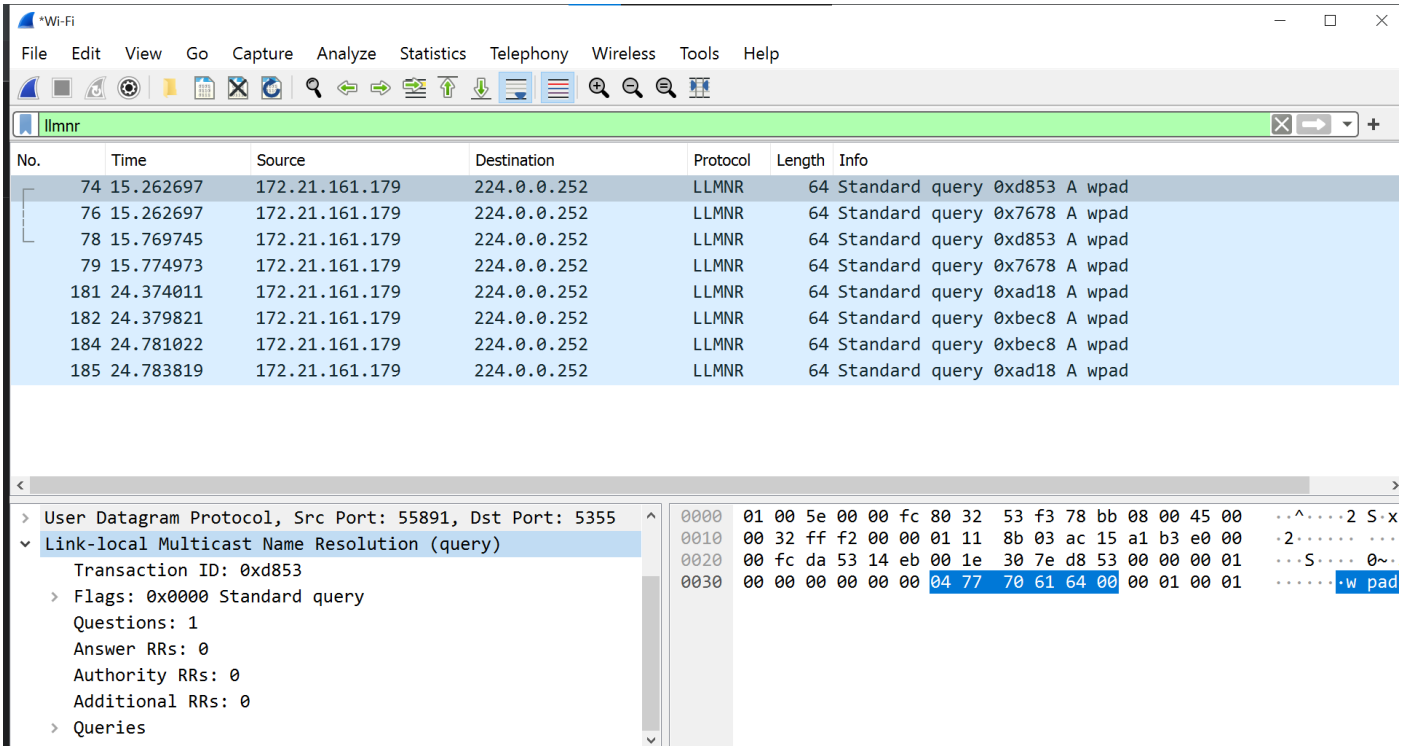


Fig.4.11 . Filtering LLMNR protocol through Wireshark

The I/O graph can also be used to identify any patterns in the data, such as spikes or drops in packet or byte counts, which may indicate issues or anomalies in the LLMNR name resolution process. For example, if there are sudden drops in LLMNR response count, it could indicate that the network is experiencing issues with name resolution. Additionally, the I/O graph can be used to analyze the response times of LLMNR requests. By using the "Time delta from previous displayed packet" option in the I/O graph, we can see the time it takes for each LLMNR request to receive a response. This information can be useful for identifying slow or unresponsive LLMNR servers, or for troubleshooting name resolution issues.

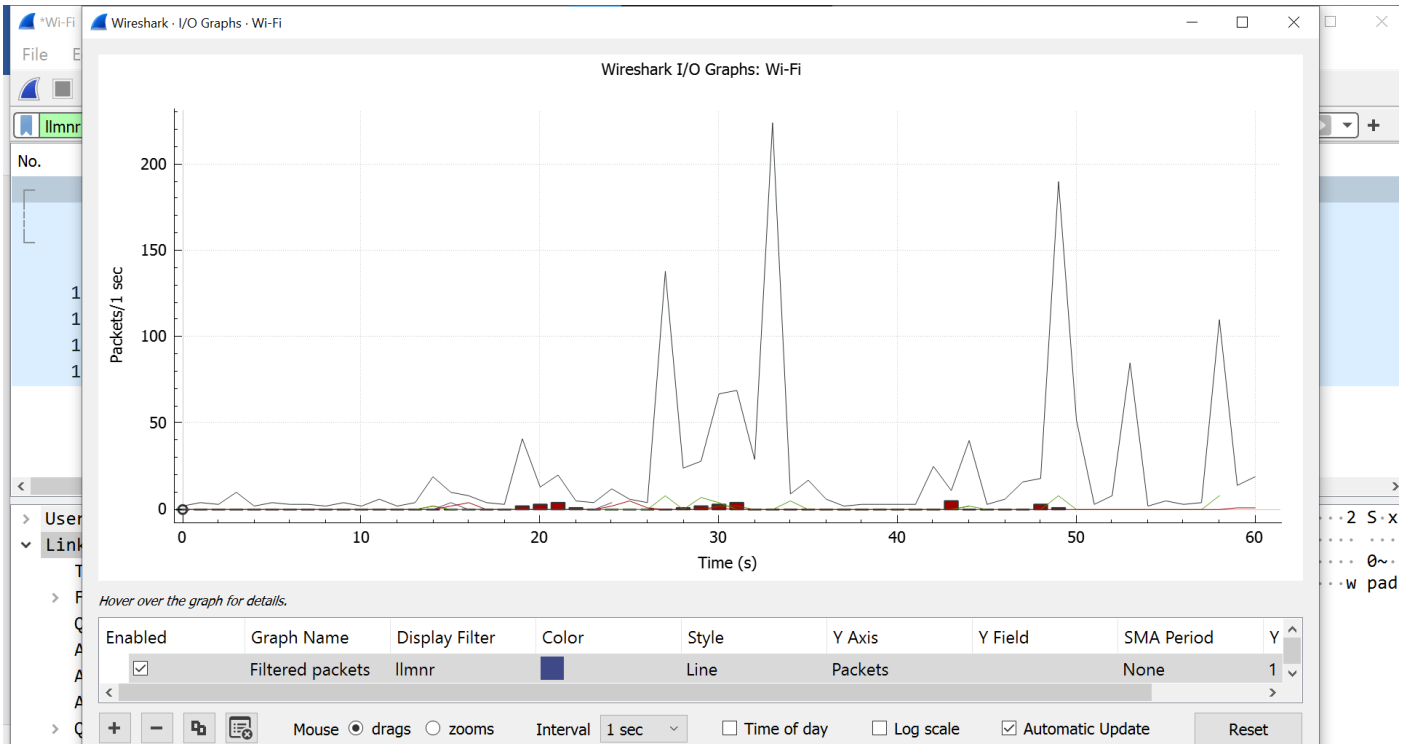


Fig.4.12. I/O Graph depicting LLMNR

## 7) Transport Layer Security (TLS):

TLS (Transport Layer Security) is a widely-used protocol for securing communications over the internet. In this project, we can use Wireshark to analyze TLS traffic and gain insights into how it is being used within our network. By analyzing the TLS handshake process, we can see how TLS connections are being established, authenticated, and secured between clients and servers.

Additionally, by inspecting the TLS application data, we can determine what type of data is being transmitted over the secured connection, and ensure that sensitive data is being properly encrypted and protected. The Wireshark TLS dissector provides detailed information on the various aspects of the TLS protocol, including the cipher suites and encryption methods used.

Overall, by analyzing TLS traffic using Wireshark, we can gain a deeper understanding of how TLS is being used to secure our network communications, and identify any potential security issues or vulnerabilities that need to be addressed.

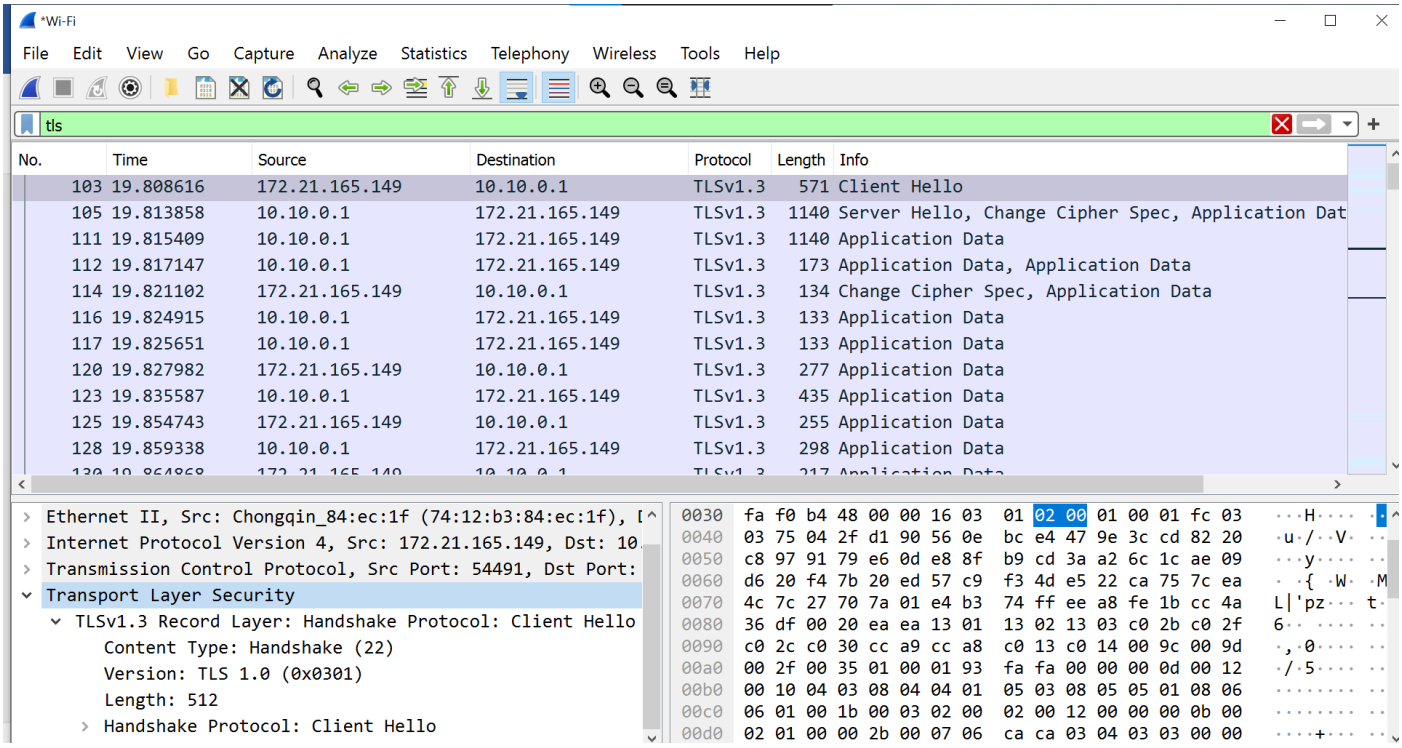


Fig.4.13 . Filtering TLS protocol through Wireshark

In the case of TLS, the I/O graph can show us the distribution of packet sizes, the rate of packet transmission, and the timing of TLS handshake messages. We can also use the I/O graph to identify any patterns of retransmission, indicating possible network congestion or packet loss. By applying filters to the I/O graph, we can isolate specific TLS connections or packets of interest, allowing us to focus our analysis on the most relevant data. For example, we can filter for specific cipher suites or TLS versions, or isolate packets containing application data.

Overall, the Wireshark I/O graph provides a powerful tool for visualizing and analyzing TLS traffic, helping us to better understand how this important protocol is being used within our network.

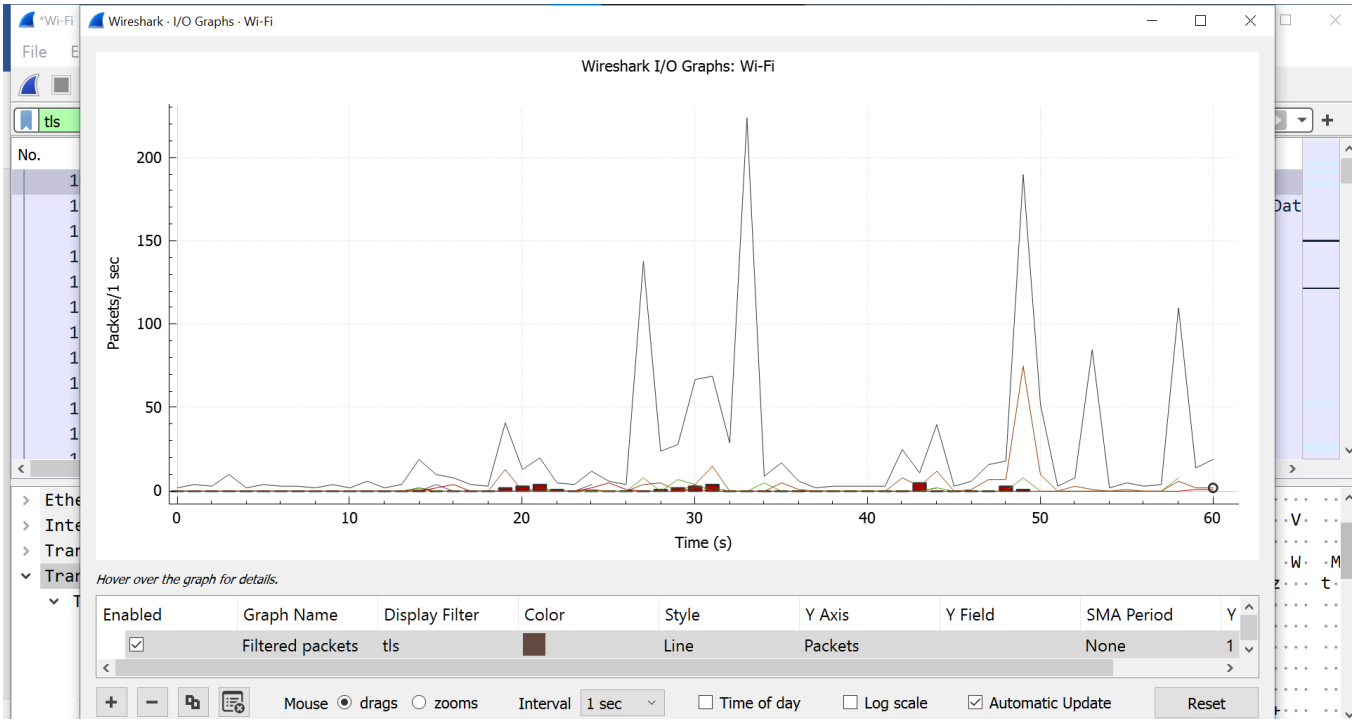


Fig.4.14. I/O Graph depicting TLS

## 8) QUIC (Quick UDP Internet Connections):

QUIC (Quick UDP Internet Connections) is a relatively new protocol developed by Google for optimizing web traffic over UDP (User Datagram Protocol). Unlike traditional TCP-based protocols, QUIC provides enhanced security and reliability features that can improve network performance and user experience. By examining the QUIC packet structure and analyzing the handshake process, we can better understand how QUIC connections are established and maintained, and identify any potential issues or vulnerabilities.

Additionally, by inspecting the QUIC application data, we can determine what type of data is being transmitted over the connection, and ensure that sensitive data is being properly encrypted and protected. The Wireshark QUIC dissector provides detailed information on the various aspects of the QUIC protocol, including the connection IDs, packet numbers, and encryption methods used.

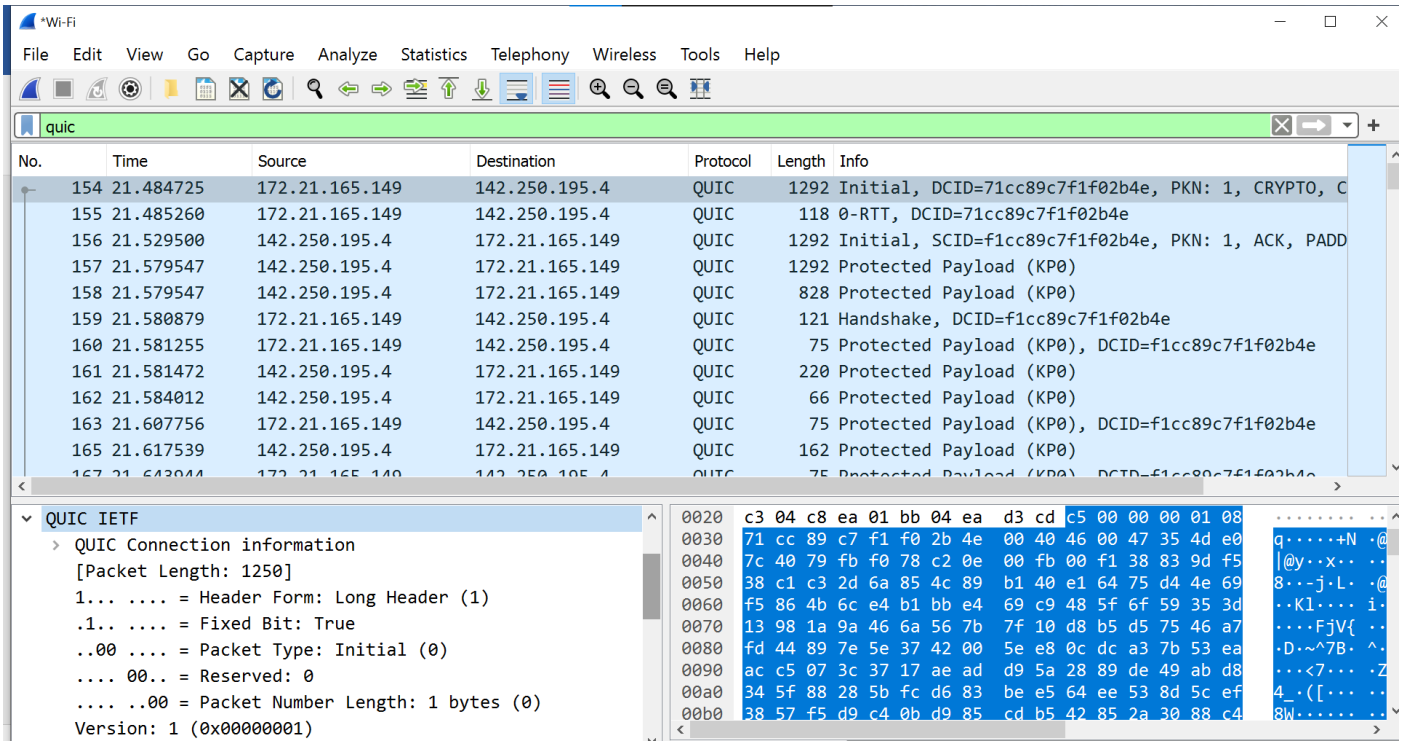


Fig.4.15 . Filtering QUIC protocol through Wireshark

The Wireshark I/O Graph for QUIC protocol can be useful in visualizing the network traffic and analyzing the performance of QUIC connections. The I/O Graph can display different metrics such as packet rate, packet size, round-trip time (RTT), and throughput.

For example, the packet rate graph can show the number of QUIC packets being transmitted over time, allowing us to identify any spikes or drops in traffic that may indicate issues with the network or the application. Similarly, the packet size graph can provide insights into the size of the QUIC packets being transmitted, and identify any unusually large or small packets that may be affecting performance.

Overall, the Wireshark I/O Graph for QUIC protocol can provide valuable insights into the performance and behavior of QUIC connections in our network, and help us identify and troubleshoot any issues affecting their performance.

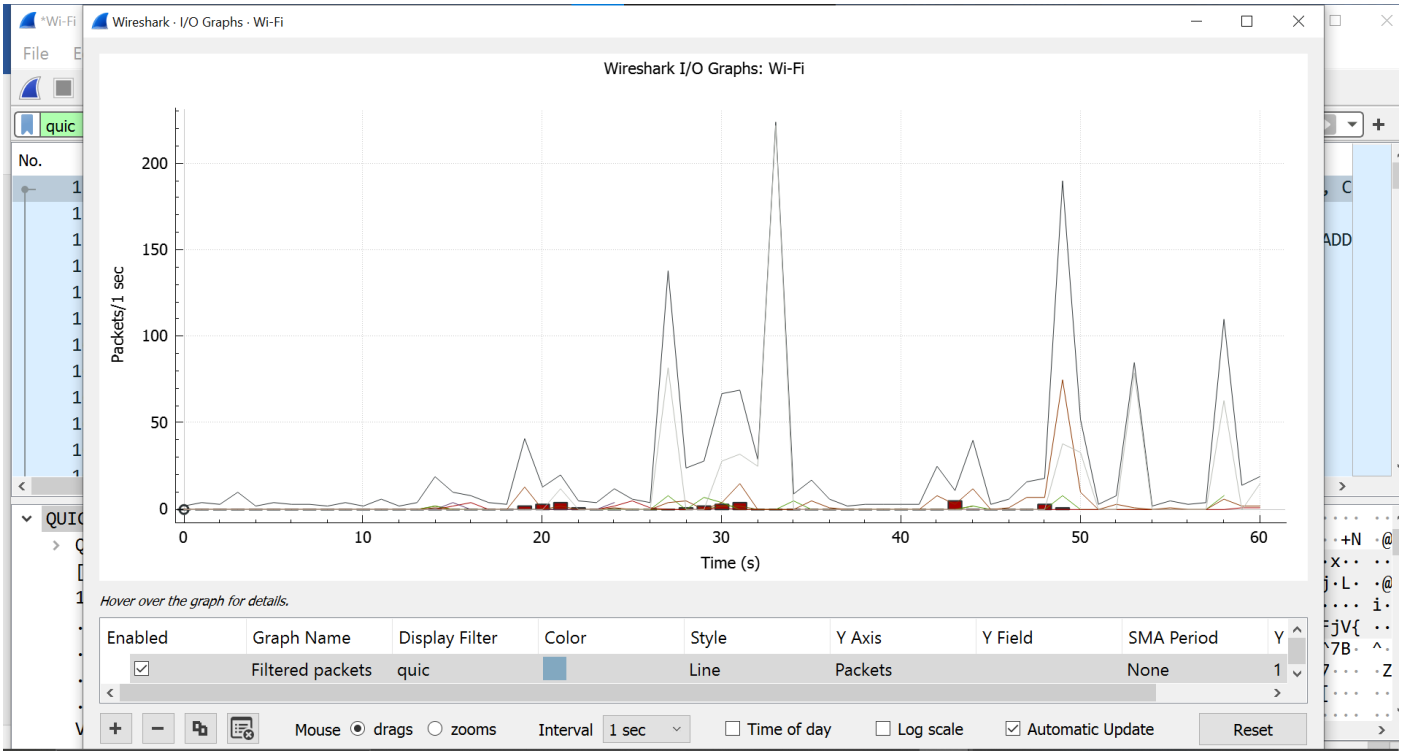


Fig.4.16. I/O Graph depicting QUIC

## 9) The Internet Group Management Protocol (IGMP):

The Internet Group Management Protocol (IGMP) is a communications protocol used by hosts and adjacent routers to establish multicast group memberships. Analyzing IGMP packets can provide insight into which multicast groups are being used, how many hosts are subscribed to each group, and whether there are any issues with multicast traffic delivery. Wireshark can also be used to capture and analyze IGMP queries and reports, which can help identify network congestion and other performance issues affecting multicast traffic.

Overall, analyzing IGMP protocol using Wireshark can help network administrators optimize multicast traffic delivery and ensure that critical applications and services are able to function as intended.

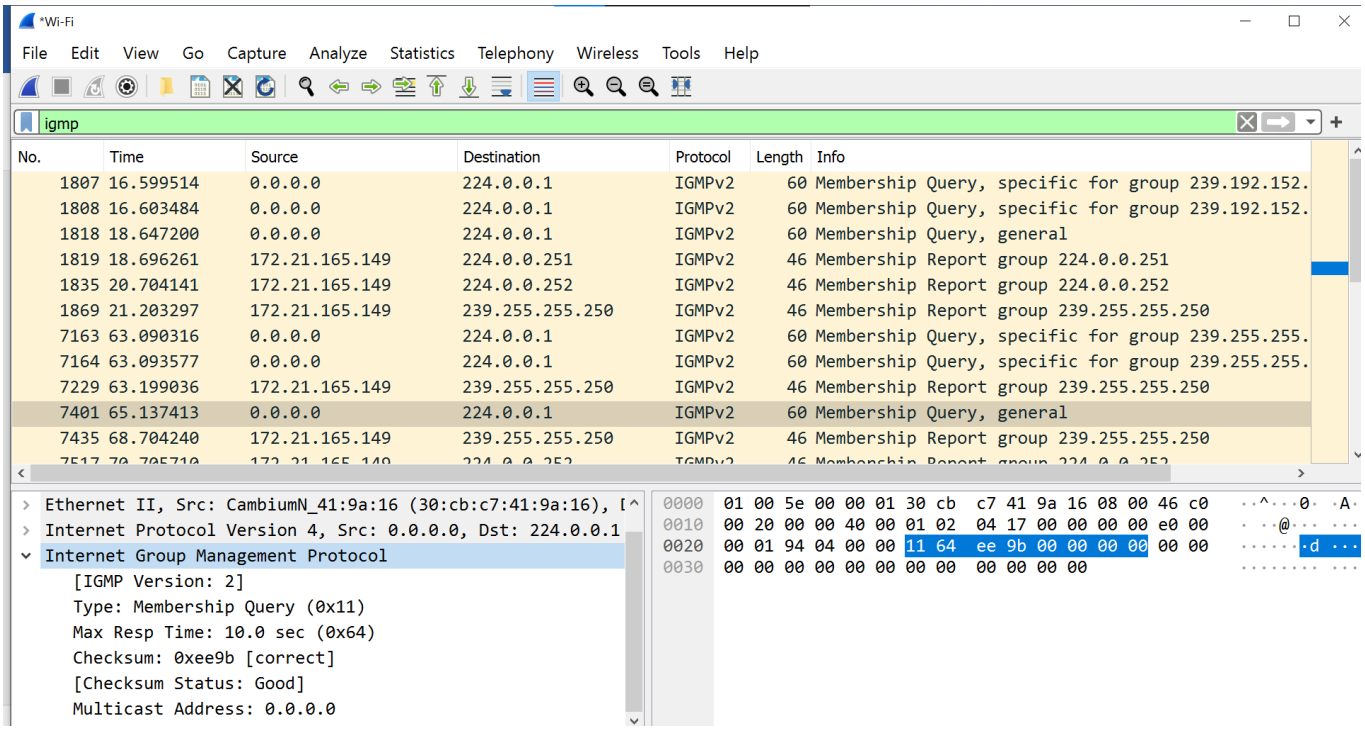


Fig.4.17 . Filtering IGMP protocol through Wireshark

The Wireshark I/O graph for IGMP protocol can show how multicast traffic is being sent and received on the network. The graph can display the number of IGMP queries and reports over time, providing insight into the number of multicast groups being used and how many hosts are subscribed to each group.

By analyzing the I/O graph, network administrators can identify patterns in IGMP traffic, such as whether there are spikes in activity at certain times of the day or during particular network events. This information can help administrators optimize multicast traffic delivery and ensure that critical applications and services are functioning as intended. The Wireshark I/O graph can also show the response times for IGMP queries and reports, providing information on how long it takes for hosts to join and leave multicast groups. By monitoring these response times, network administrators can identify any delays or performance issues that may be impacting multicast traffic delivery.

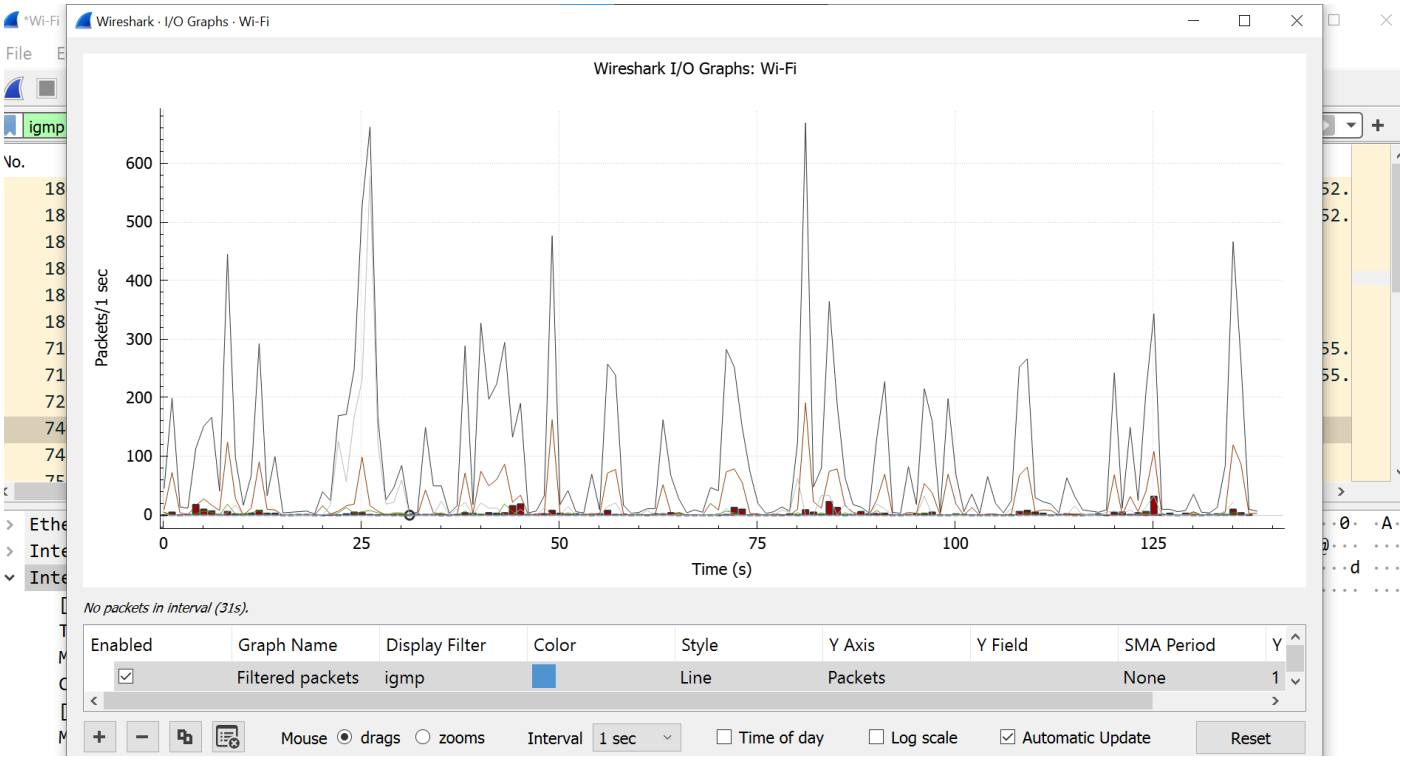


Fig.4.18. I/O Graph depicting IGMP

## 10) Internet Control Message Protocol (ICMP):

The ICMP protocol is an integral part of network communication as it is used to send error messages and operational information about network conditions. ICMP is used by network devices such as routers, switches and firewalls to communicate errors and status messages.

By analyzing ICMP packets using Wireshark, network administrators can identify network issues such as packet loss, high latency, or other errors. Additionally, ICMP can be used to detect network scans and other malicious activity, making it an important tool for network security.

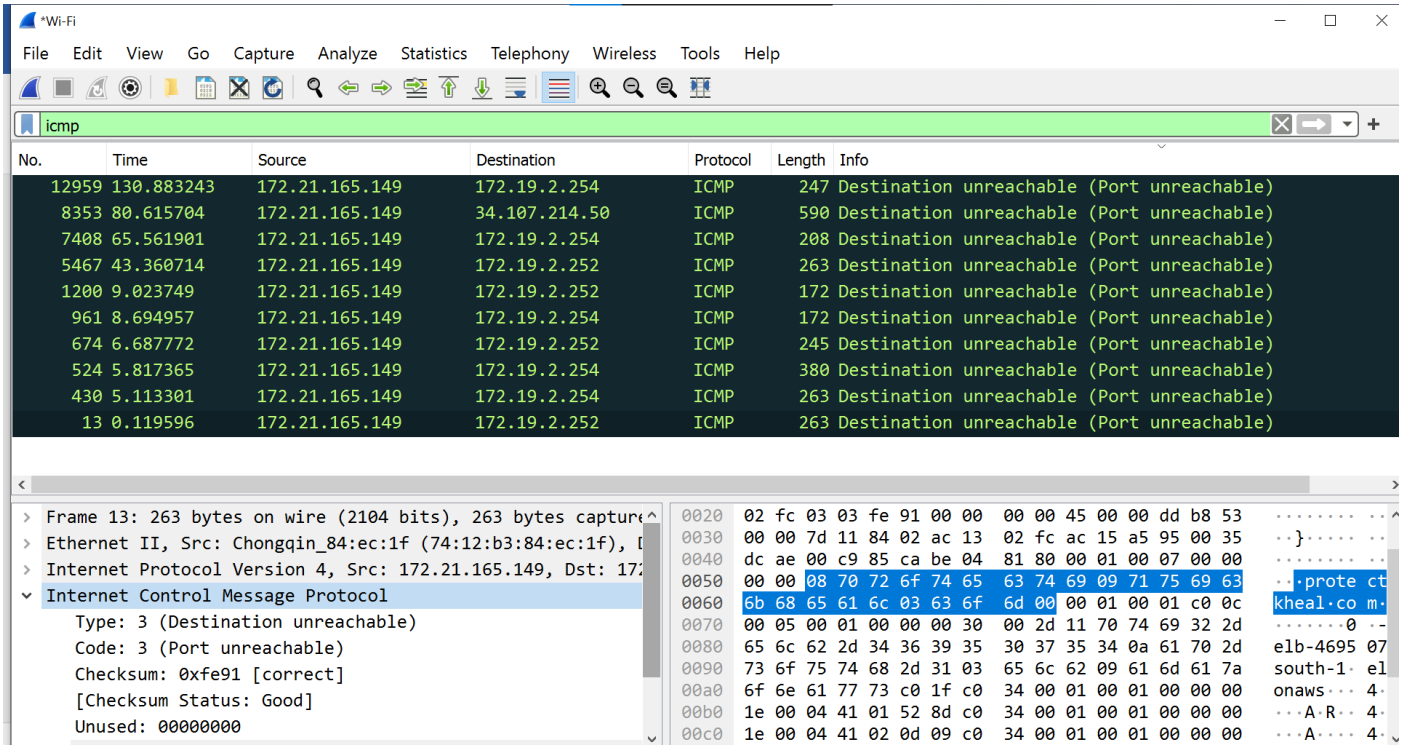


Fig.4.19 . Filtering ICMP protocol through Wireshark

The Wireshark I/O graph for ICMP can provide a visual representation of the frequency and duration of ICMP packets, allowing network administrators to identify patterns of network behavior. This information can be used to optimize network performance, troubleshoot network issues, and improve network security.

Overall, analyzing ICMP protocol using Wireshark can help network administrators to ensure that the network is operating smoothly, identify and address any issues that may arise, and enhance network security.

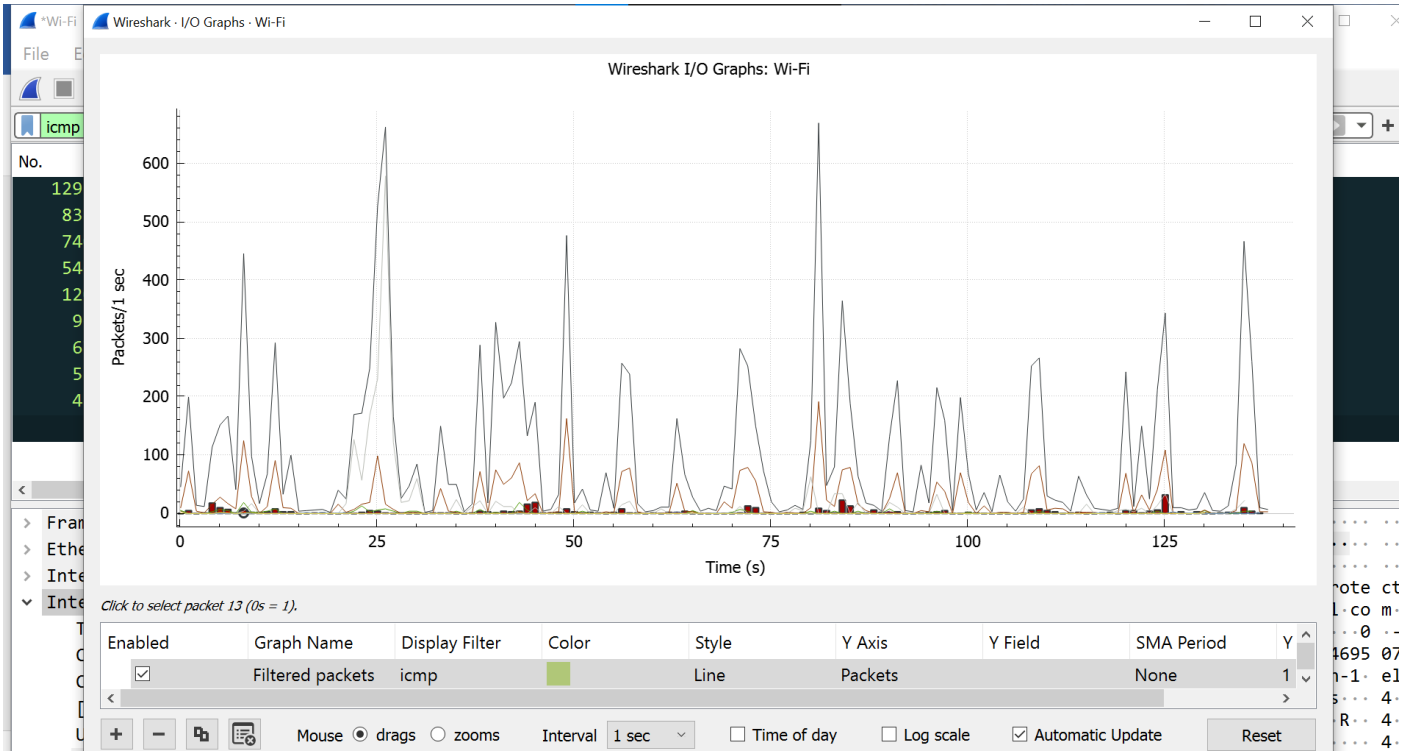


Fig.4.20. I/O Graph depicting ICMP

## 11) The Simple Service Discovery Protocol (SSDP):

The Simple Service Discovery Protocol (SSDP) is a network protocol used for discovering and configuring network devices, such as printers, media servers, and Internet of Things (IoT) devices. SSDP uses HTTP over UDP and operates on multicast address 239.255.255.250. By analyzing SSDP packets using Wireshark, network administrators can identify and track network devices and services on their network, as well as detect any issues with service discovery. Additionally, SSDP can be used by attackers to identify and exploit vulnerable devices on a network, making it important to monitor and secure.

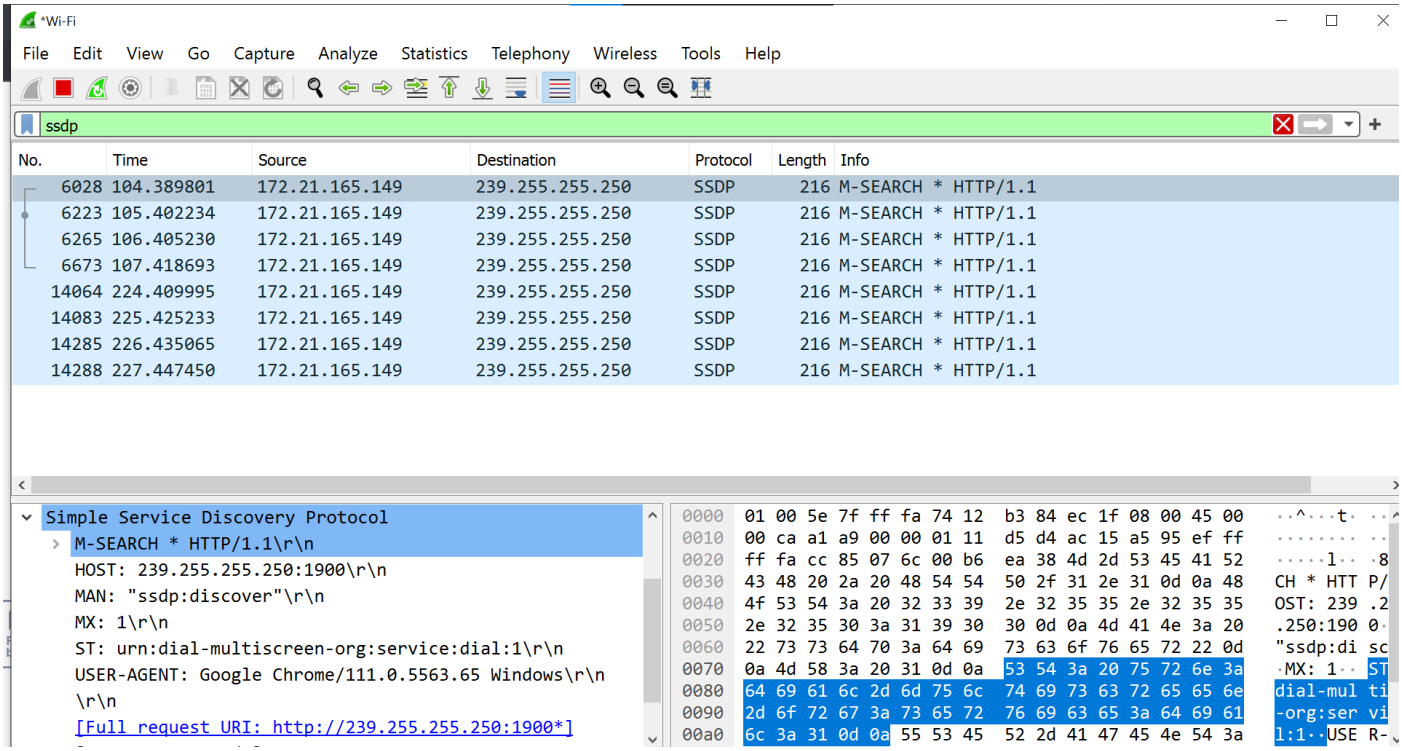


Fig.4.21 . Filtering SSDP protocol through Wireshark

The Wireshark I/O graph for SSDP can be useful for visualizing the frequency and volume of SSDP traffic on a network. In the I/O graph, the X-axis represents time, and the Y-axis represents the number of packets or bytes transmitted. The graph can reveal any patterns or spikes in SSDP traffic, which can be indicative of issues with service discovery or potential attacks.

For example, a sudden spike in SSDP traffic could suggest that a large number of new devices have been added to the network, or that an attacker is using SSDP to scan for vulnerable devices. The I/O graph can also be used to analyze the distribution of SSDP traffic across different ports and IP addresses. This information can help network administrators identify any devices or services that are generating an unusually high volume of SSDP traffic, which could be causing performance issues or pose a security risk.

Overall, the Wireshark I/O graph for SSDP can be a valuable tool for monitoring and troubleshooting issues related to service discovery on a network, as well as identifying and mitigating potential security threats.

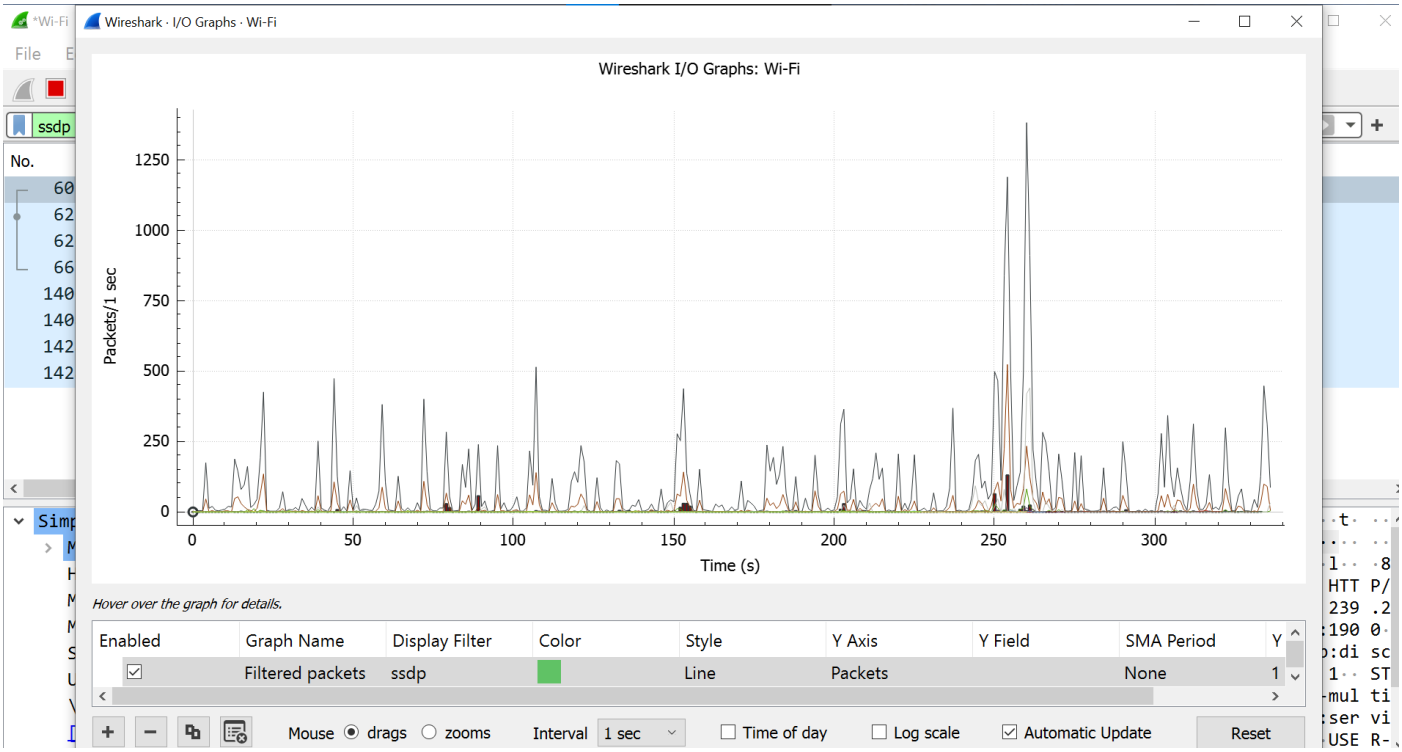


Fig.4.22. I/O Graph depicting SSDP

## 12) MS Windows Browser Protocol:

The Microsoft Windows Browser Protocol (also known as the "Browser" protocol) is used by computers running Windows operating systems to maintain a list of available resources on a local network. This protocol allows computers to locate and connect to other devices, including printers and shared folders. The Browser protocol operates on UDP port 137 and 138, and TCP port 139. It can be susceptible to security vulnerabilities such as man-in-the-middle attacks and denial-of-service attacks. When capturing packets with Wireshark, we can use filters to specifically capture traffic related to the Browser protocol. We can then use the Wireshark I/O graph to visualize the activity of this protocol, including the number of packets sent and received, as well as any spikes or anomalies in the traffic. This can help us to identify potential issues with the protocol and optimize network performance.

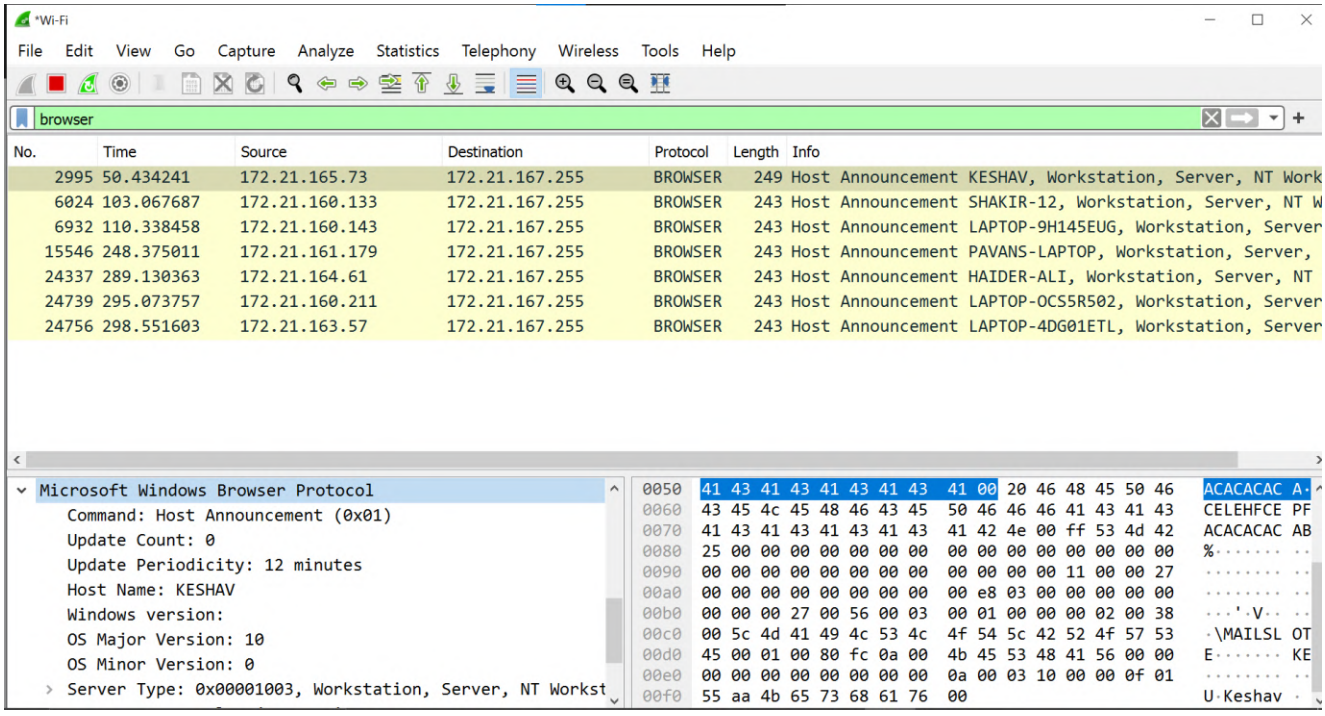


Fig.4.23 . Filtering Browser protocol through Wireshark

The I/O graph in Wireshark displays a graph of packet I/O rates over time, allowing us to visualize patterns in the traffic. By applying a filter to capture only packets related to the Browser protocol, we can see the rate of packets sent and received over UDP ports 137 and 138, and TCP port 139. We can also use the I/O graph to identify any spikes or anomalies in the traffic, which may indicate issues with the protocol or network performance. Furthermore, the I/O graph can also be used to compare the traffic patterns of the Browser protocol to other protocols on the network, allowing us to identify potential conflicts or performance bottlenecks. Overall, the Wireshark I/O graph is a valuable tool for analyzing the Microsoft Windows Browser Protocol and optimizing network performance.

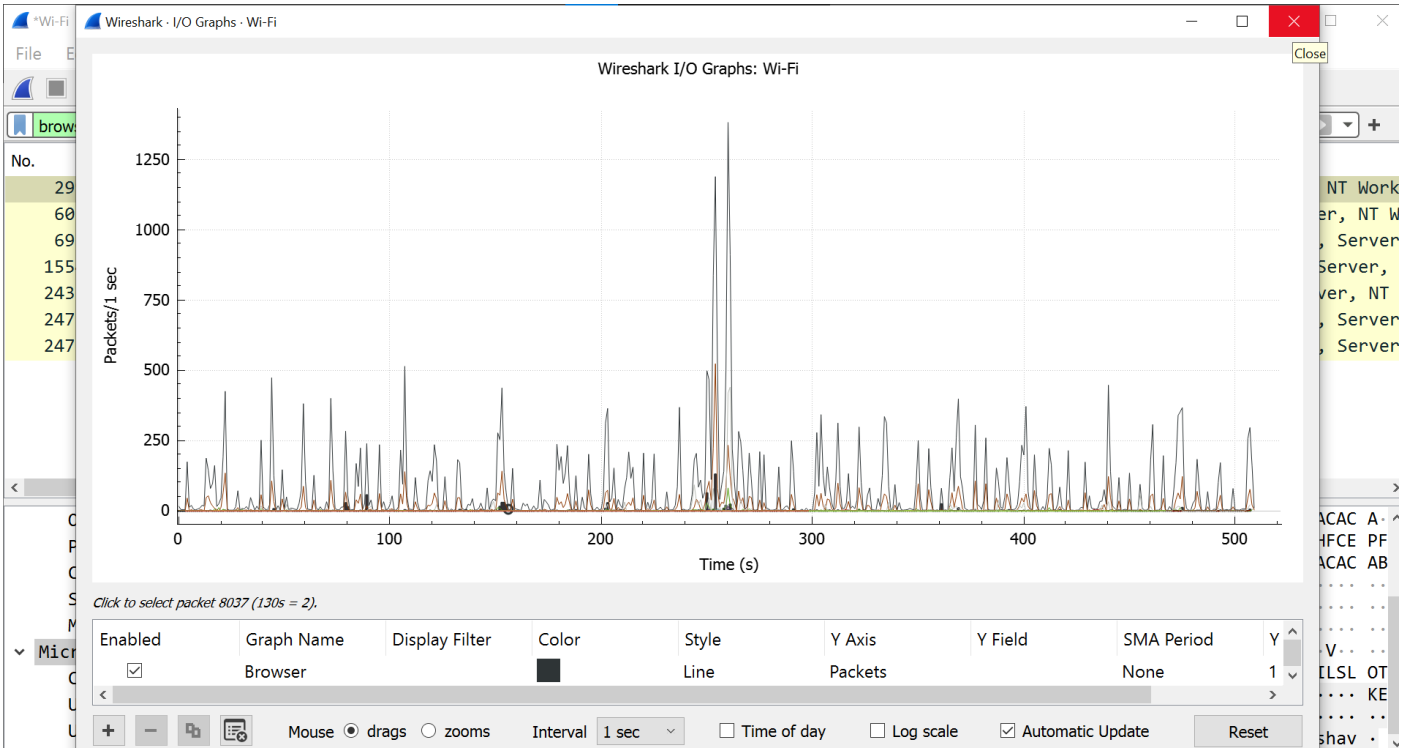


Fig.4.24. I/O Graph depicting Browser protocol

### 3.4. INVESTIGATING PROTOCOLS FROM LIVE DATA OF BLUETOOTH AND USB AND ANALYSING THEM:

**1) From USB:**

Wireshark can be used to capture and analyze USB traffic on a live system. To capture USB traffic, we connected USB device to the system and ran Wireshark as an administrator. The appropriate USB interface from the list of available interfaces in Wireshark was then selected.

Once the USB traffic is captured, we analyzed it in Wireshark using various filters and display options. Wireshark provides various dissectors for USB protocol analysis, allowing us to view the USB requests and responses in detail.

Reading live data from USB using Wireshark can be useful in diagnosing USB communication issues, analyzing USB device behavior, and reverse engineering USB protocols. However, it's important to note that capturing USB traffic on a live system can potentially disrupt the operation of the USB device or the system, so caution should be exercised when using this technique.

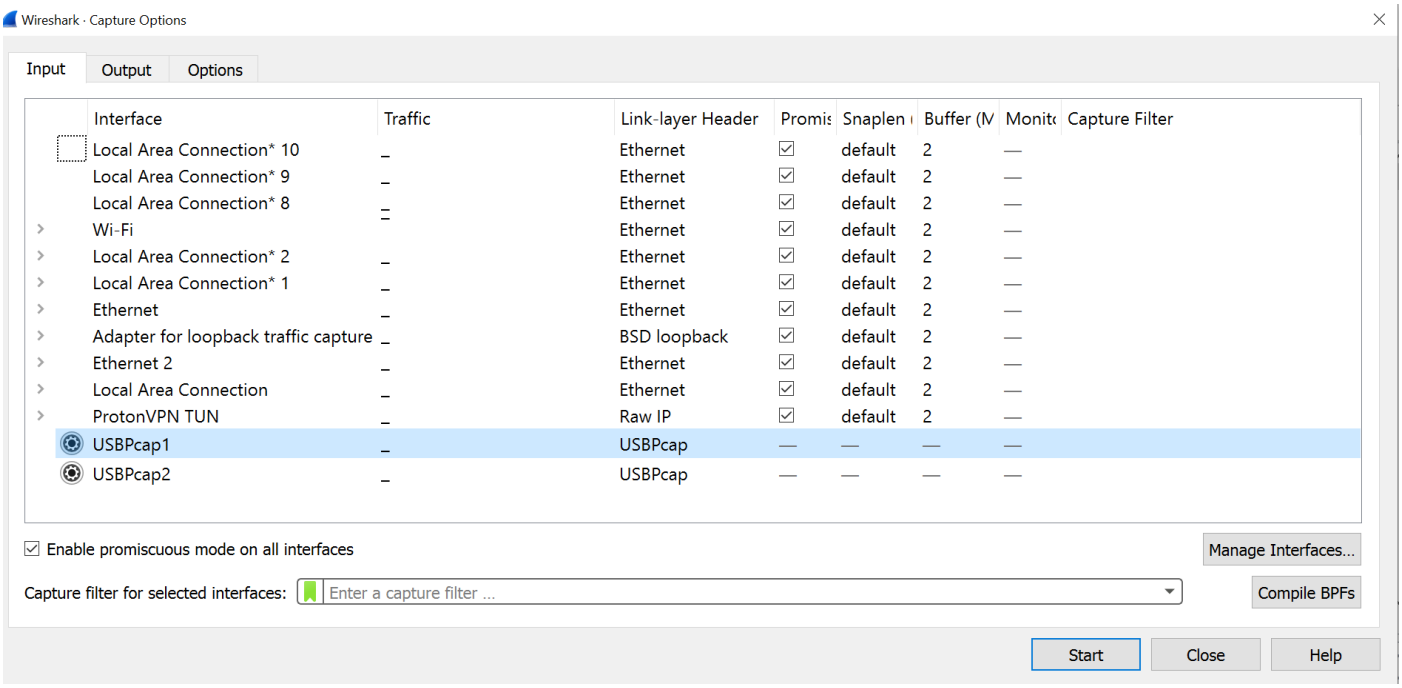


Fig.5.1. UBPcap1 denotes a wireless mouse USB connection.

- **USB Protocol:**

USB (Universal Serial Bus) is a widely used communication protocol for connecting devices to a computer. USB is a plug-and-play interface that allows devices to be connected and disconnected without requiring a restart of the computer. The USB protocol defines the electrical and functional specifications of the standard interface. It also defines how devices communicate with each other and with the computer. USB supports different data transfer speeds and multiple devices can be connected to a single USB port through the use of hubs. In this project, the Wireshark tool was used to analyze the USB protocol and capture live data from USB devices.

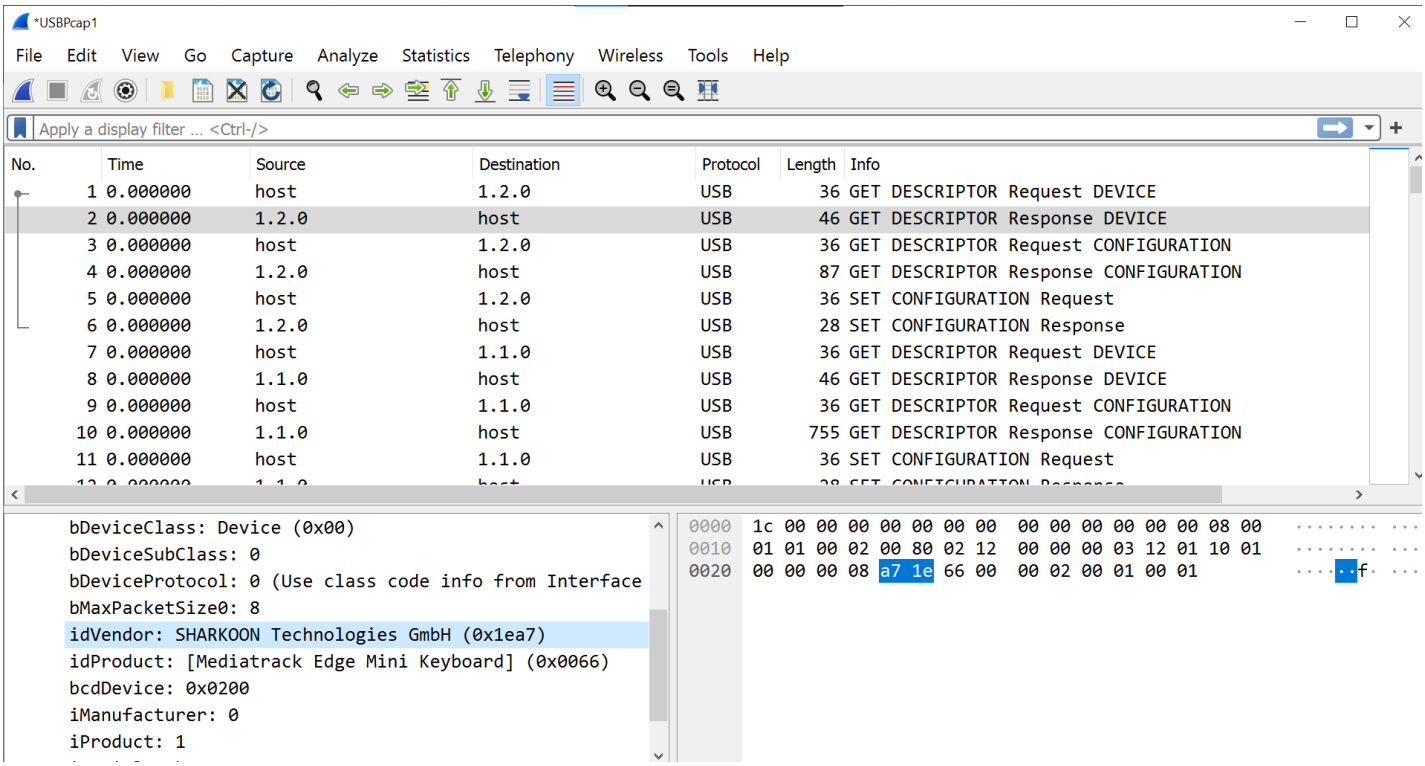


Fig.5.2. Filtering USB protocol through Wireshark

There is no I/O graph available for USB protocol in Wireshark. USB data is captured and displayed in Wireshark as a packet list. However, the packet list provides detailed information about each USB packet, such as the packet length, transfer type, endpoint, and data payload. Additionally, Wireshark allows filtering USB traffic based on various criteria, such as endpoint address or transfer type, to simplify analysis. With Wireshark, one can analyse USB traffic and investigate issues related to data transfer or device communication.

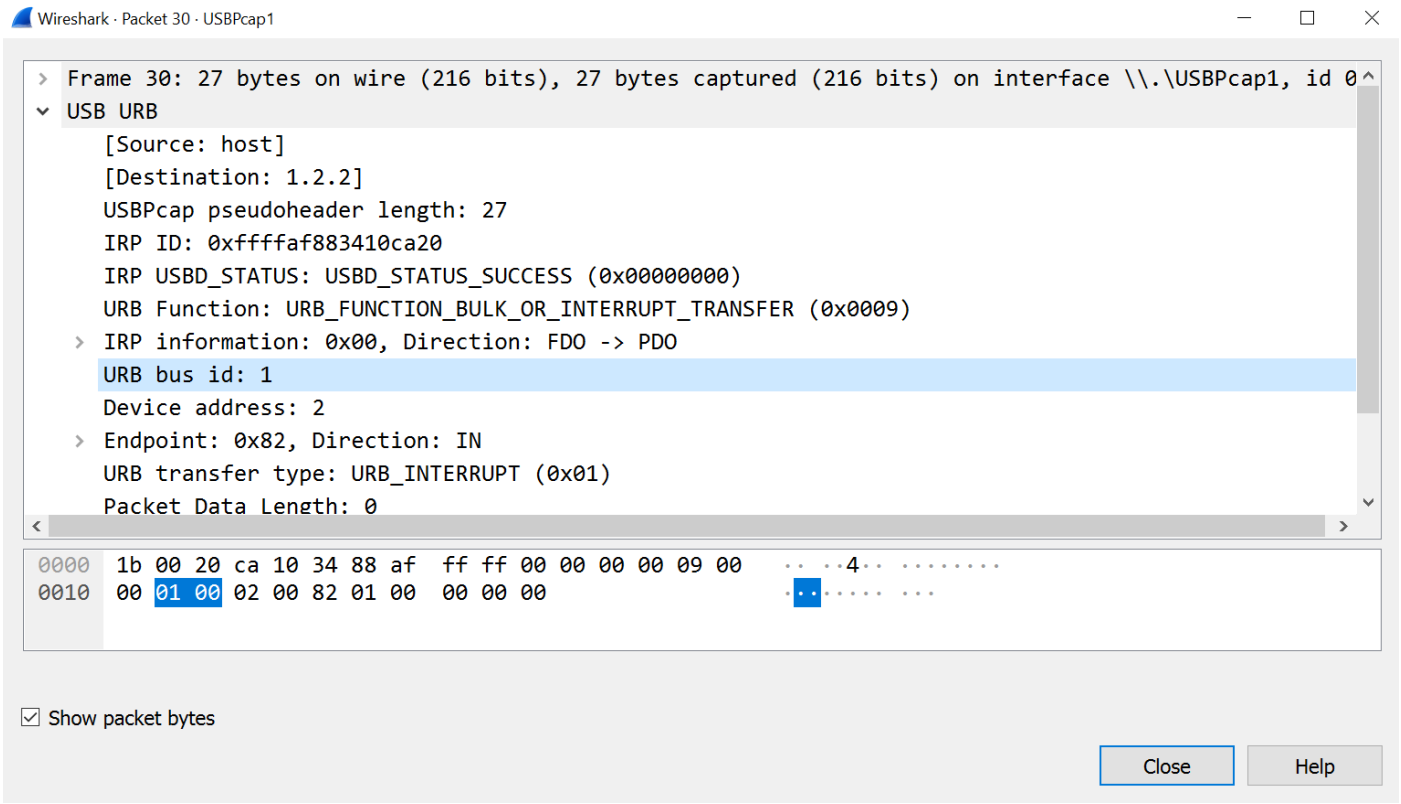


Fig. 5.3 . Detailed Packet analysis of USB Protocol

## 2) From Bluetooth devices:

We connected a Bluetooth enabled headset to our system using Bluetooth and analysed using Wireshark.

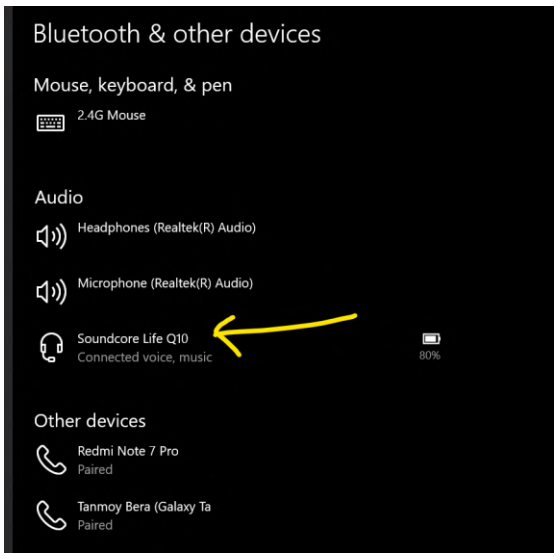


Fig.5.4. Bluetooth device connected

To capture Bluetooth traffic using Wireshark we need the BTP software package, we can get it at : https://learn.microsoft.com/en-us/windows-hardware/drivers/bluetooth/testing-btp-setup-package

Next we Install the package. The files are installed in C:\BTP[version] usually.

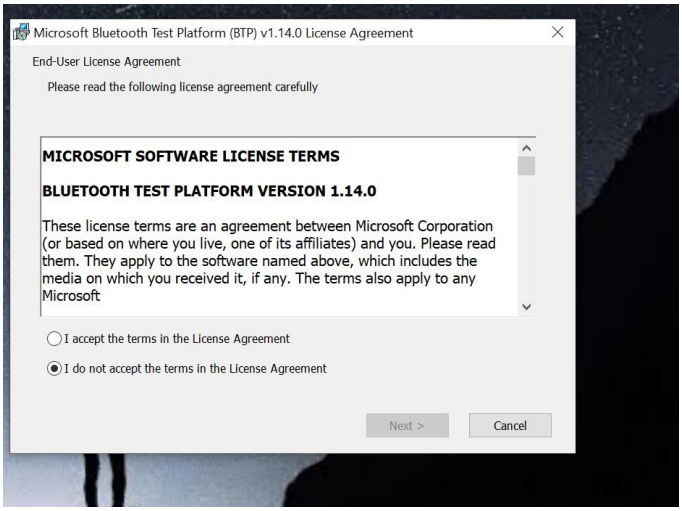


Fig. 5.5. 1st step in installation of BTP

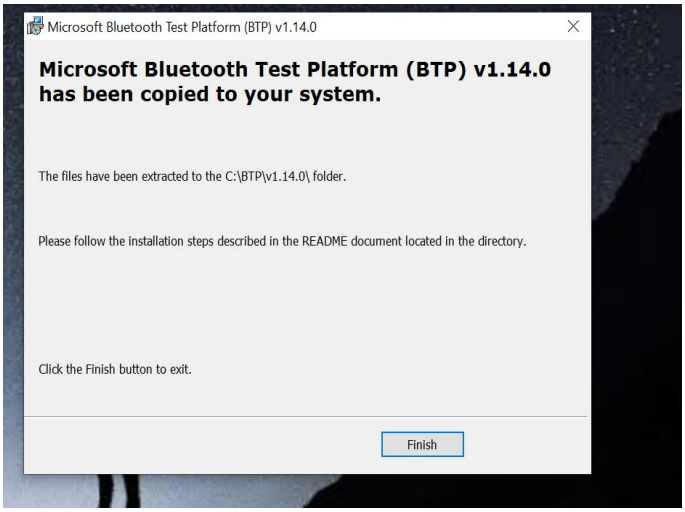


Fig.5.6. Final step in installation of BTP

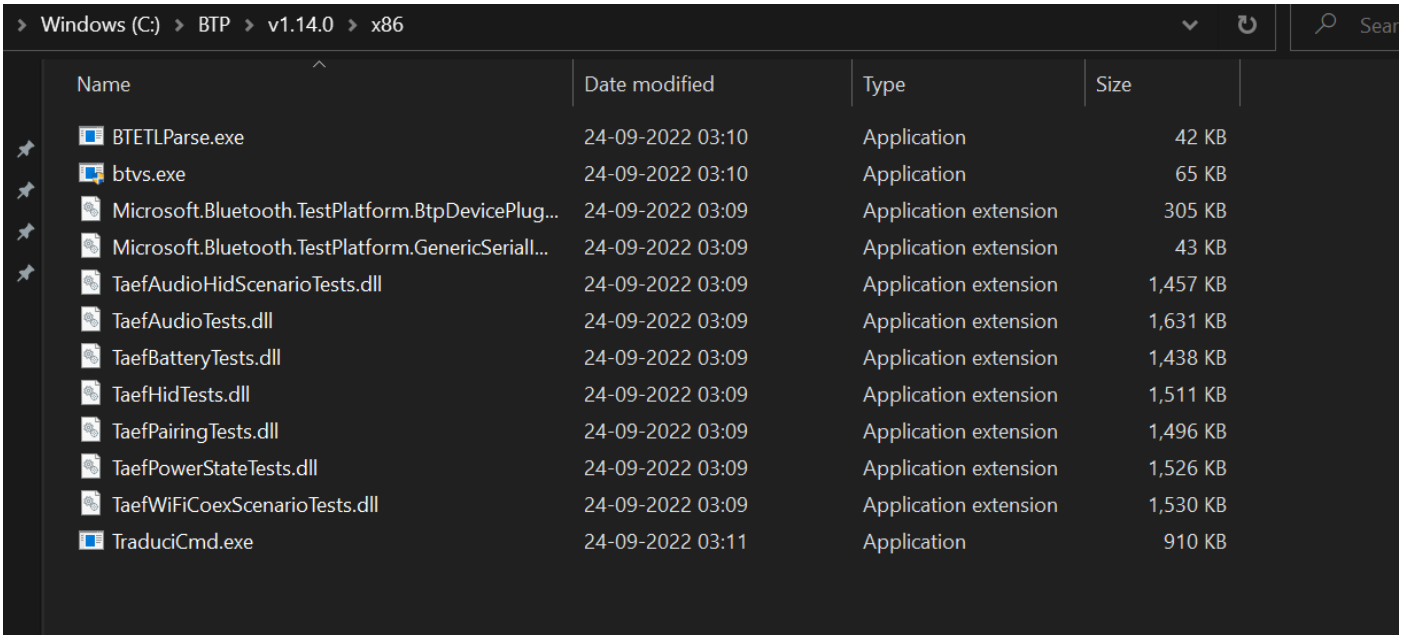


Fig.5.7. location of btvs.exe in x86 folder

Next we locate a file named btvs.exe in folder x86. Launched a console with the admin privileges and typed ./btvs.exe -Mode Wireshark. Wireshark windows will launch and start capturing traffic immediately.

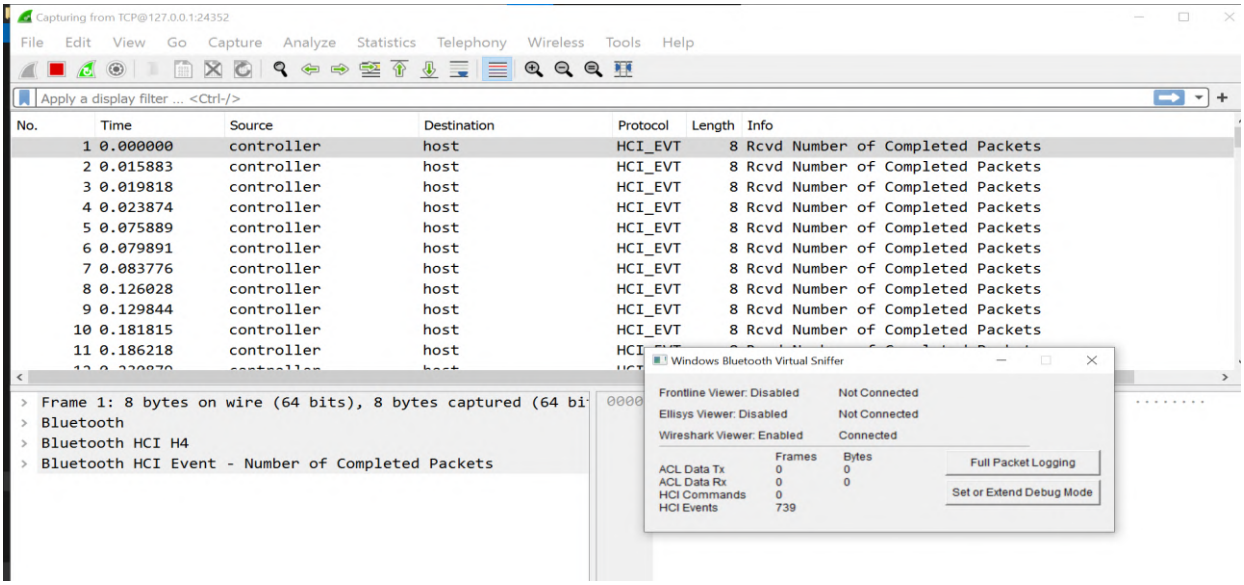


Fig. 5.8. Filtering HCI_EVT protocol through Wireshark

- **HCI_EVT (Host Controller Interface Event):**

HCI_EVT (Host Controller Interface Event) is a protocol used by Bluetooth devices to exchange information between the host controller and the Bluetooth device. It is used to communicate events that occur during the operation of the Bluetooth device, such as when a device is discovered, when a connection is established, or when data is transmitted. In this project, the analysis of HCI_EVT protocol can provide insights into the operation of Bluetooth devices and the communication between the host and the device. It can help in identifying potential issues or areas for improvement in Bluetooth device communication.

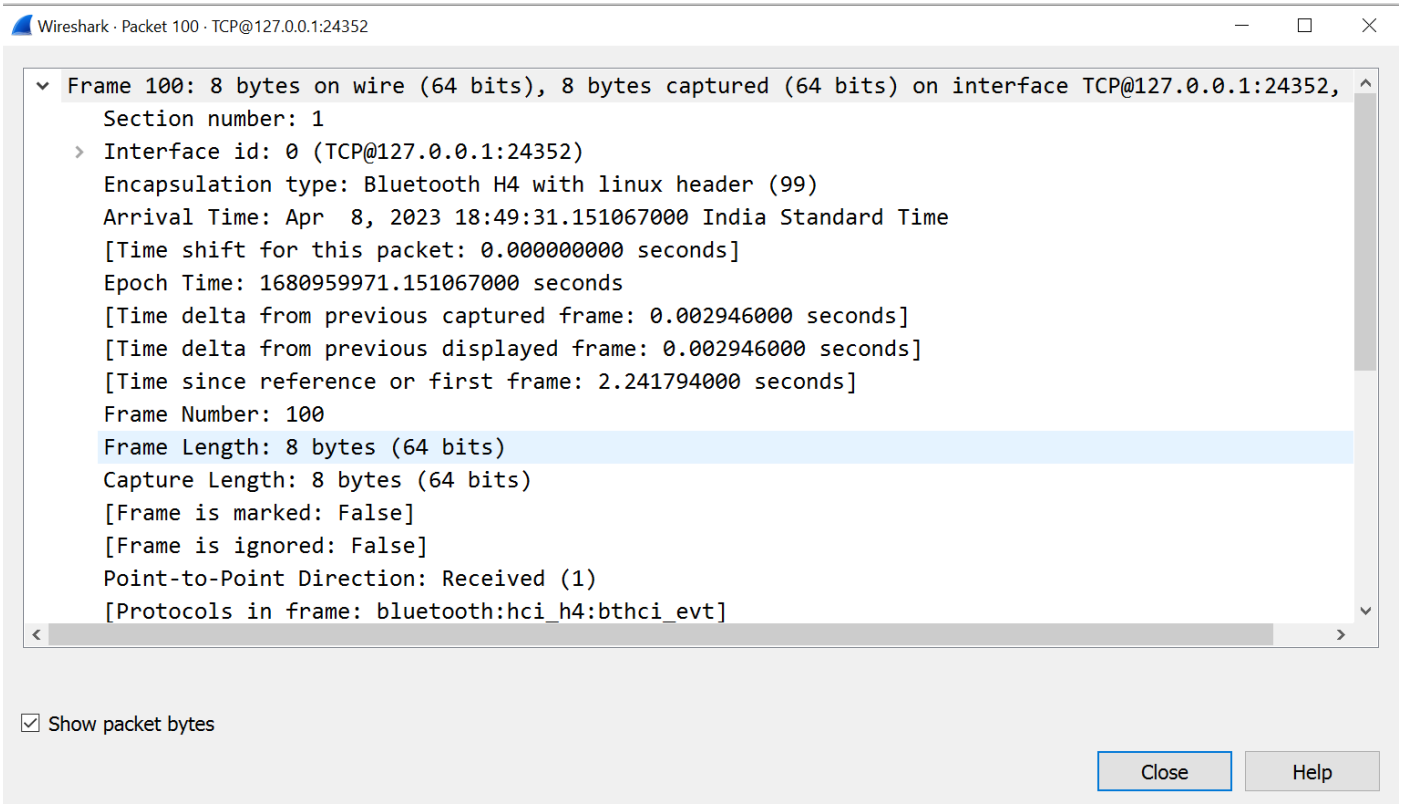


Fig. 5.9.1. Detailed analysis of HCL_EVT protocol

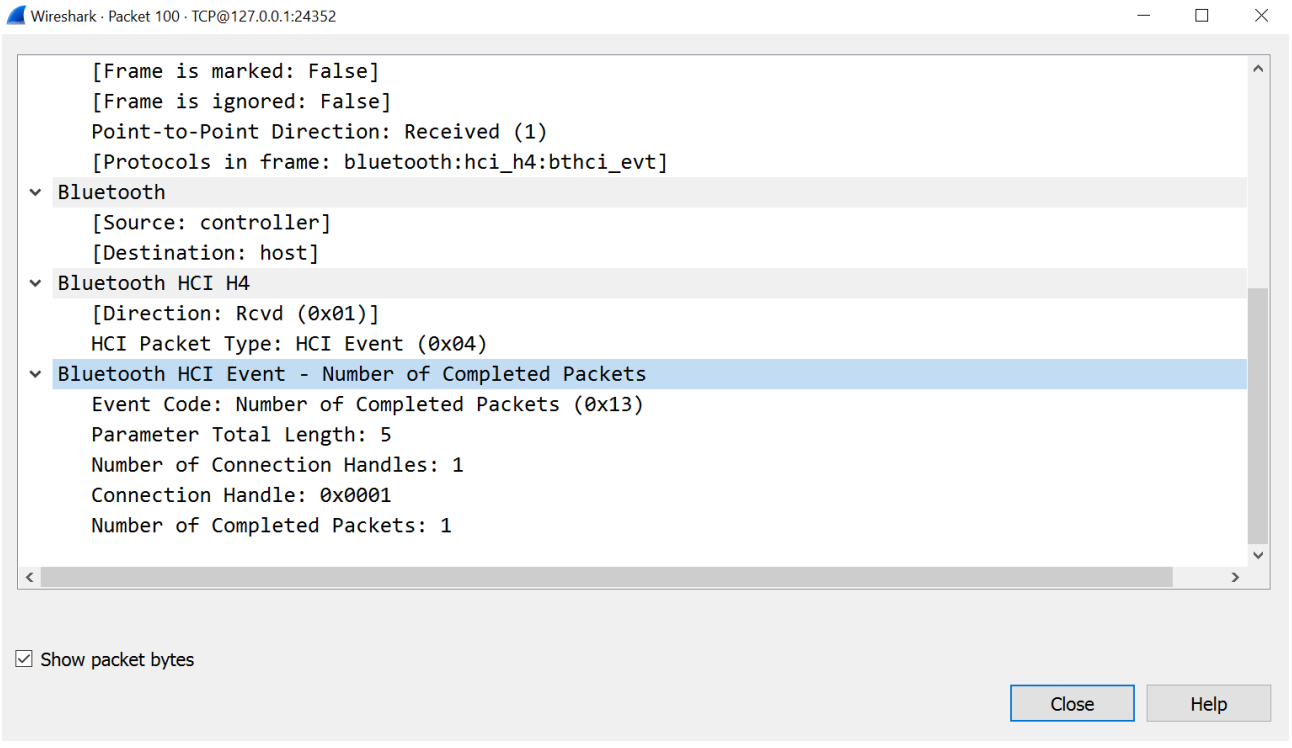


Fig. 5.9.2. Detailed analysis of HCL_EVT protocol

# CHAPTER- 4

## 4.1. CONCLUSION:

In conclusion, this project aimed to explore network protocols by using Wireshark to analyze network traffic at a microscopic level. The project investigated ten different protocols including TCP, UDP, HTTP, DNS, NBNS, LLMNR, TLS, QUIC, IGMP, and SSDP, as well as reading live data from Bluetooth and USB. Through the use of Wireshark, we were able to gain a better understanding of how each protocol works and their specific network behaviors. The I/O graphs generated in Wireshark helped to visualize the data being transmitted and allowed for further analysis. This project provides valuable insights into network protocol analysis and demonstrates the importance of using network monitoring tools like Wireshark in troubleshooting and optimizing network performance.

## 4.2. ACKNOWLEDGEMENT:

I would like to express my sincere gratitude to all those who have supported and contributed to the successful completion of this project. Firstly, I would like to thank my teacher, Dr. Manjot Kaur for providing me with guidance and valuable feedback throughout the project. I would also like to extend my appreciation to my friends for their support and encouragement. Lastly, I would like to express my gratitude to the developers of Wireshark for providing such a powerful tool for network analysis.

## 4.3. REFERENCES:

1) Wireshark. (2021). About Wireshark. Retrieved from https://www.wireshark.org/about.html (accessed March 20,2023)
2) StackOverflow.com. Retrieved from https://stackoverflow.com/questions/22085989/how-to-capture-and-analyze-bluetooth-packets-using-wireshark (accessed March 22,2023)
3) Microsoft. (2021). Microsoft Windows Browser Protocol. Retrieved from https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-wbt-server/4f3a4d50-1ca2-4682-bd3a-b3c6b0e6e45f (accessed March 23,2023)
4) IETF. (2021). User Datagram Protocol. Retrieved from https://tools.ietf.org/html/rfc768 (accessed March 23,2023)
5) IETF. (2021). Transmission Control Protocol. Retrieved from https://tools.ietf.org/html/rfc793 (accessed March 23,2023)
6) IETF. (2021). Hypertext Transfer Protocol. Retrieved from https://tools.ietf.org/html/rfc2616 (accessed March 24,2023)
7) IETF. (2021). Domain Name System Protocol. Retrieved from https://tools.ietf.org/html/rfc1035 (accessed March 24,2023)
8) IETF. (2021). Link-Local Multicast Name Resolution (LLMNR) Protocol. Retrieved from https://tools.ietf.org/html/rfc4795 (accessed March 25,2023)
9) IETF. (2021). Simple Service Discovery Protocol (SSDP). Retrieved from https://tools.ietf.org/html/rfc6763 (accessed March 26,2023)
10) IETF. (2021). Internet Group Management Protocol (IGMP). Retrieved from https://tools.ietf.org/html/rfc3376 (accessed March 26,2023)
11) IETF. (2021). QUIC: A UDP-Based Multiplexed and Secure Transport. Retrieved from https://tools.ietf.org/html/rfc9000 (accessed March 30,2023)

12) IETF. (2021). Internet Control Message Protocol (ICMP). Retrieved from https://tools.ietf.org/html/rfc792 (accessed March 30,2023)

## 4.4. GITHUB LINK:

This report can also be found at : https://github.com/tanmoykrbera/INT301CA3