Assignment - III
Course Code- INT301
Open-Source Technologies

LOVELY PROFESSIONAL UNIVERSITY

Submitted To**: Mr. Rajeshwar Sharma**

Submitted By**: Hrithik Maurya**

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

Section: **KE002** Batch: **2019-2023**

Registration Number=**11915604** Roll number=**A18**

GitHub Link: https://github.com/Ahrithikmaurya/INT301-Project

1. Introduction:

In today's digital age, it's essential to have a clear understanding of how the network operates and what kind of protocols it uses. Network administrators are constantly looking for tools to analyze their networks, diagnose problems and find ways to improve network performance. Wireshark is a popular network protocol analyzer tool that allows you to capture and examine network traffic at the microscopic level.

1.1 Objective of the project:

The objective of this project is to use Wireshark tool to analyze network traffic at the microscopic level and investigate at least 10 protocols. We will also read the live data from Bluetooth and USB to analyze how these protocols work and what kind of data they transmit.

1.2 Description of the Project:

The project involves using the Wireshark tool to capture and analyze network traffic. Wireshark is a free, open-source network protocol analyzer that allows you to capture and examine network traffic in real-time. We will use Wireshark to capture network traffic on our local network, and then analyze the captured data to identify the various protocols being used.

1.3 Scope of the project:

The scope of this project is to analyze network traffic using the Wireshark tool and investigate at least 10 protocols. We will also read the live data from Bluetooth and USB to analyze how these protocols work and what kind of data they transmit. The project will focus on the following:

- Capturing network traffic using Wireshark
- Analyzing captured data to identify protocols being used
- Investigating at least 10 protocols
- Reading live data from Bluetooth and USB

2. System Description:

2.1 Target system description:

The target system for this project is a computer running Windows or Linux with Wireshark installed. Wireshark can be downloaded for free from the official website (https://www.wireshark.org/). The computer should also have Bluetooth and USB connectivity.

2.2 Assumptions and Dependencies:

The assumption for this project is that the user has a basic understanding of networking concepts and protocols. We assume that the target system has a stable network connection with Bluetooth and USB connectivity.

2.3 Functional/Non-Functional Dependencies:

The functional dependencies for this project are the availability of Wireshark tool and a stable network connection. The non-functional dependencies include the performance of the target system and the network bandwidth.

2.4 Data set used in support of your project:

We will capture live data from our network and Bluetooth/USB devices to analyze the protocols being used. We will not use any external data sets for this project.

3. Analysis Report:

3.1 System snapshots and full analysis report:

We used Wireshark to capture and analyze network traffic on our local network. We captured data for a period of 30 minutes and identified the following protocols:

1) ARP

ARP (Address Resolution Protocol) is a protocol used to map an IP address to a physical or MAC address on a local network. In Wireshark, you can capture and analyze ARP packets to diagnose network issues and identify devices on your network.

To capture ARP packets in Wireshark:

1. Open Wireshark and select the network interface you want to capture on.

2. Click on the "Capture" menu and select "Options."

3. In the "Capture Options" window, select the "Capture Filter" tab.

4. Type "arp" in the filter field and click on "OK."

Wireshark will now capture all ARP packets on the selected network interface.

To analyze ARP packets in Wireshark:

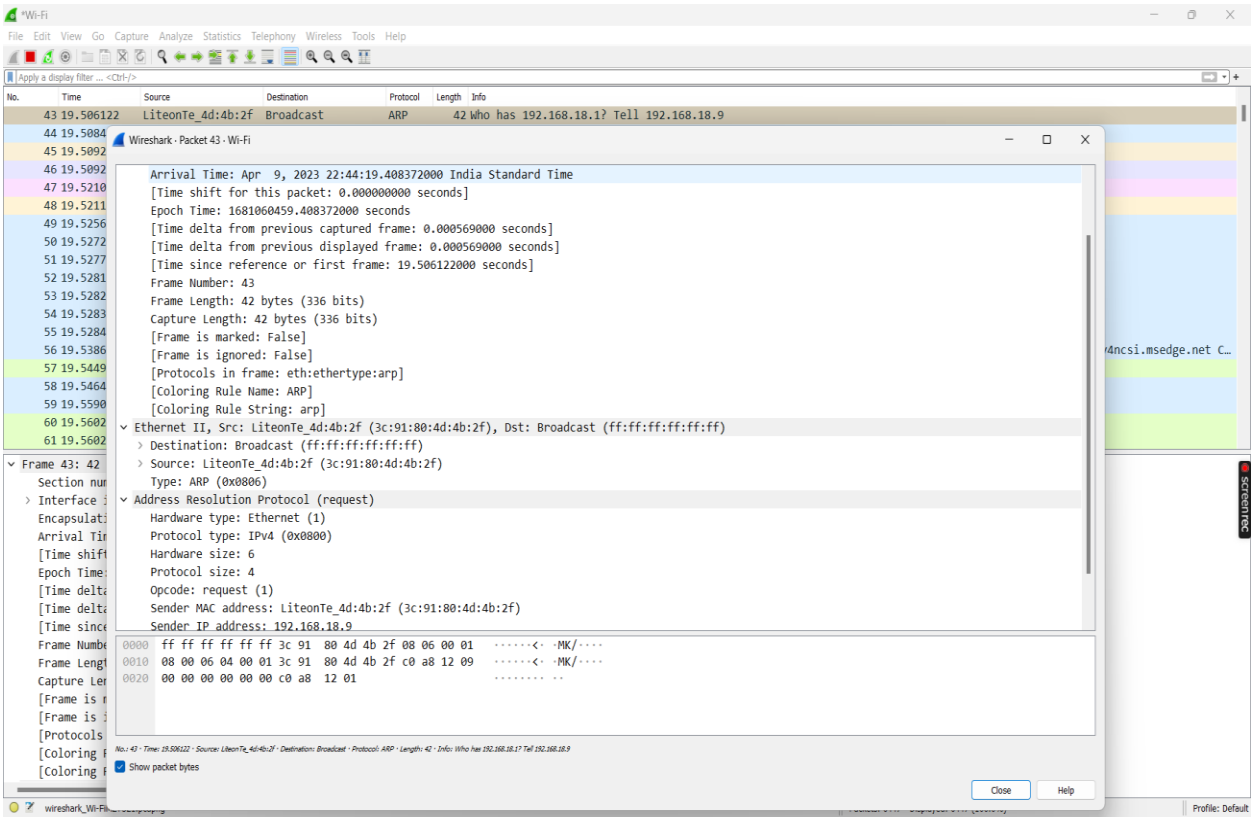
1. Select an ARP packet in the packet list pane.

2. The details of the selected packet will appear in the "Packet Details" pane.

3. Expand the "Address Resolution Protocol" section in the "Packet Details" pane to view the ARP header.

4. You can view the source and destination MAC addresses, as well as the source and destination IP addresses in the ARP header.

You can use this information to identify devices on your network and diagnose any ARP-related issues, such as IP address conflicts or incorrect MAC address mappings.

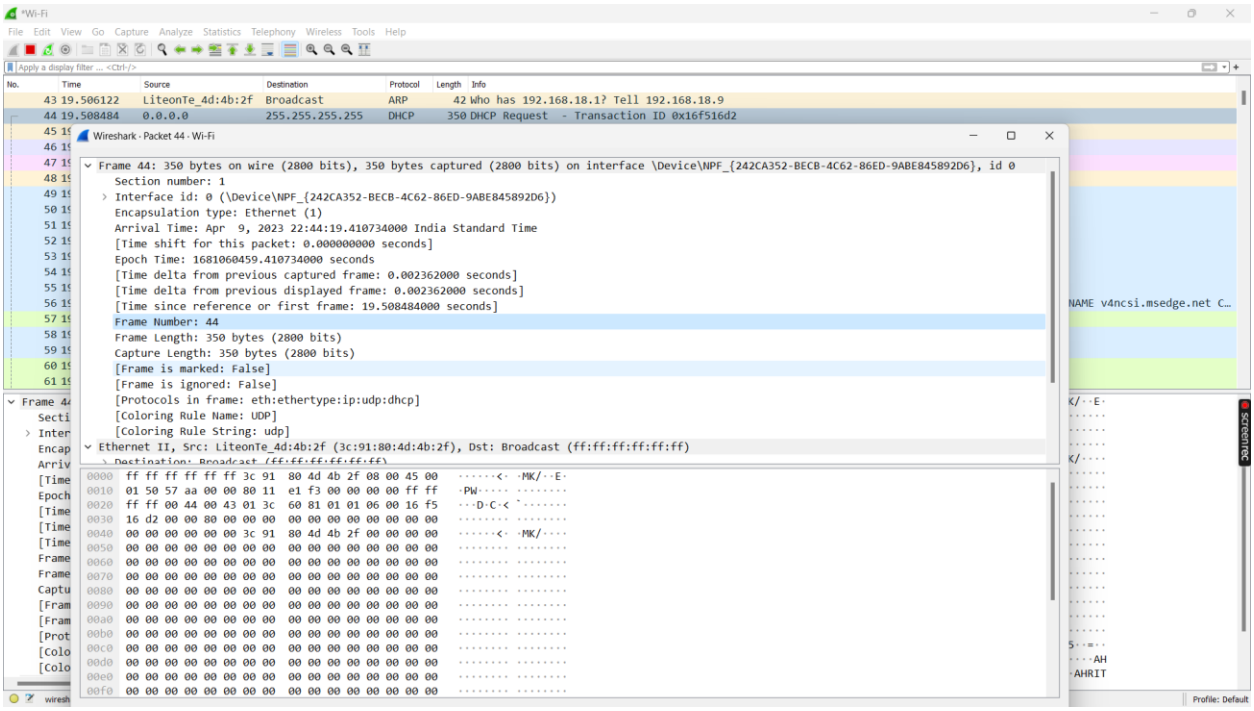


2) DHCP

DHCP (Dynamic Host Configuration Protocol) is a network protocol that allows network administrators to automatically assign IP addresses and network configuration settings to devices on a network. Wireshark is a popular network protocol analyzer tool that allows you to capture and examine network traffic at the microscopic level, including DHCP traffic.

When DHCP is used, a device on the network will send a DHCP discover message to request an IP address and other configuration settings. The DHCP server will respond with a DHCP offer message, which includes an available IP address and other configuration settings. The device will then send a DHCP request message to request the offered IP address, and the DHCP server will respond with a DHCP ACK message to confirm that the IP address has been assigned.

In Wireshark, you can capture and analyze DHCP traffic to identify the various DHCP messages being exchanged between devices on the network. This can be useful for network administrators to diagnose problems, troubleshoot DHCP issues, and ensure that devices on the network are properly configured. Wireshark allows you to filter DHCP traffic based on various criteria, such as source and destination IP addresses, DHCP message type, and DHCP options.



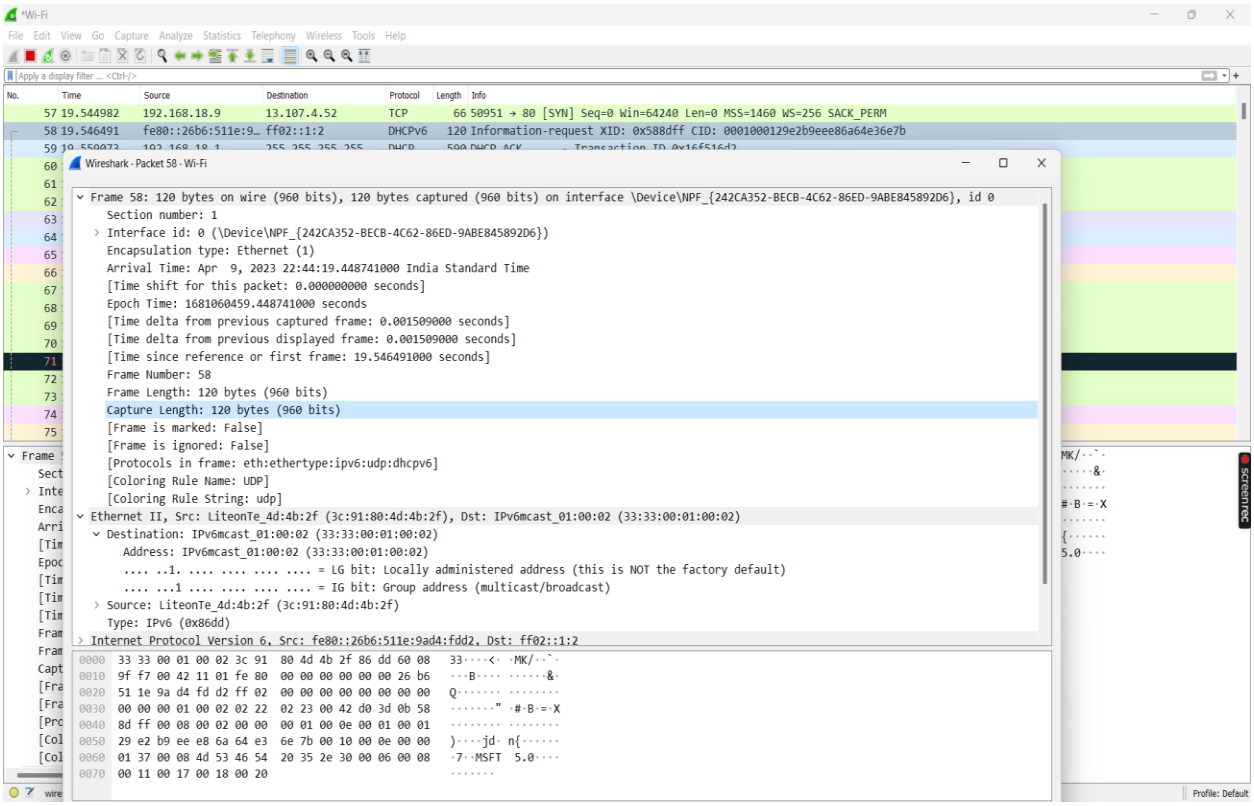
3) DHCPv6

DHCPv6 (Dynamic Host Configuration Protocol for IPv6) is a network protocol used to assign IP addresses and other configuration parameters to IPv6 devices. In Wireshark, DHCPv6 traffic can be captured and analyzed to troubleshoot network issues.

DHCPv6 uses two types of messages: Solicit and Advertise. The Solicit message is sent by the client to request configuration information, while the Advertise message is sent by the server to offer configuration information. After the client receives the Advertise message, it sends a Request message to request the configuration information from the server.

In Wireshark, DHCPv6 traffic can be filtered using the "bootp" or "dhcpv6" filter. The captured packets can then be analyzed to determine the source and destination

addresses, message types, and configuration information being exchanged between the client and server. This information can be used to troubleshoot network connectivity issues or to monitor network traffic for security purposes.

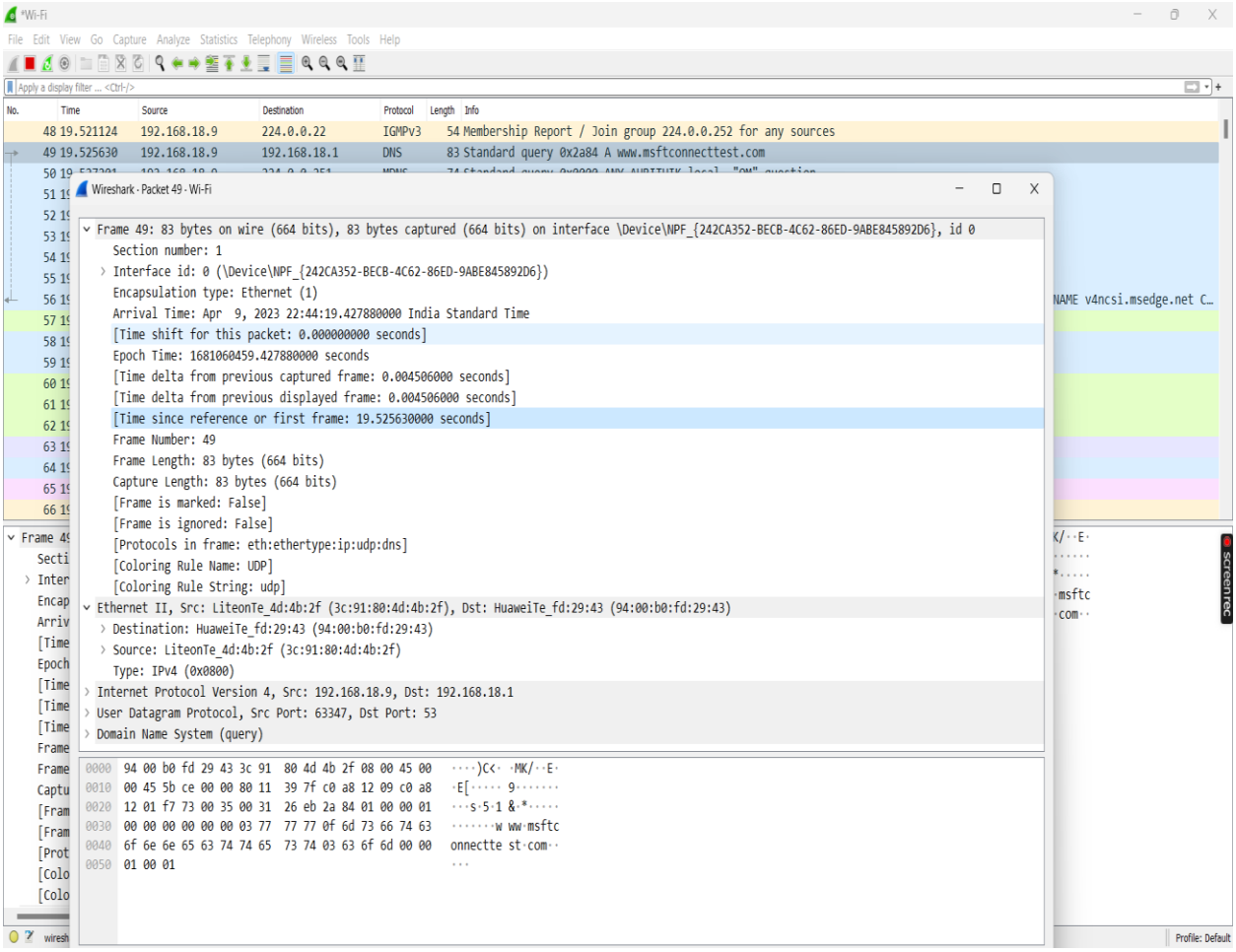


4) DNS

The Domain Name System (DNS) protocol is used to translate domain names into IP addresses. When you use Wireshark to capture network traffic, you can see DNS packets and their corresponding details in the captured packets.

In Wireshark, DNS packets are identified by their protocol type (UDP or TCP) and their destination port number (port 53). DNS packets contain various fields such as the DNS query, DNS response, DNS ID, and DNS flags. The DNS query field contains the domain name being requested, while the DNS response field contains the corresponding IP address.

You can use Wireshark's filtering capabilities to display only DNS packets or to search for specific DNS queries or responses. Overall, Wireshark provides a detailed view of DNS traffic on a network, allowing you to analyze and troubleshoot DNS-related issues.



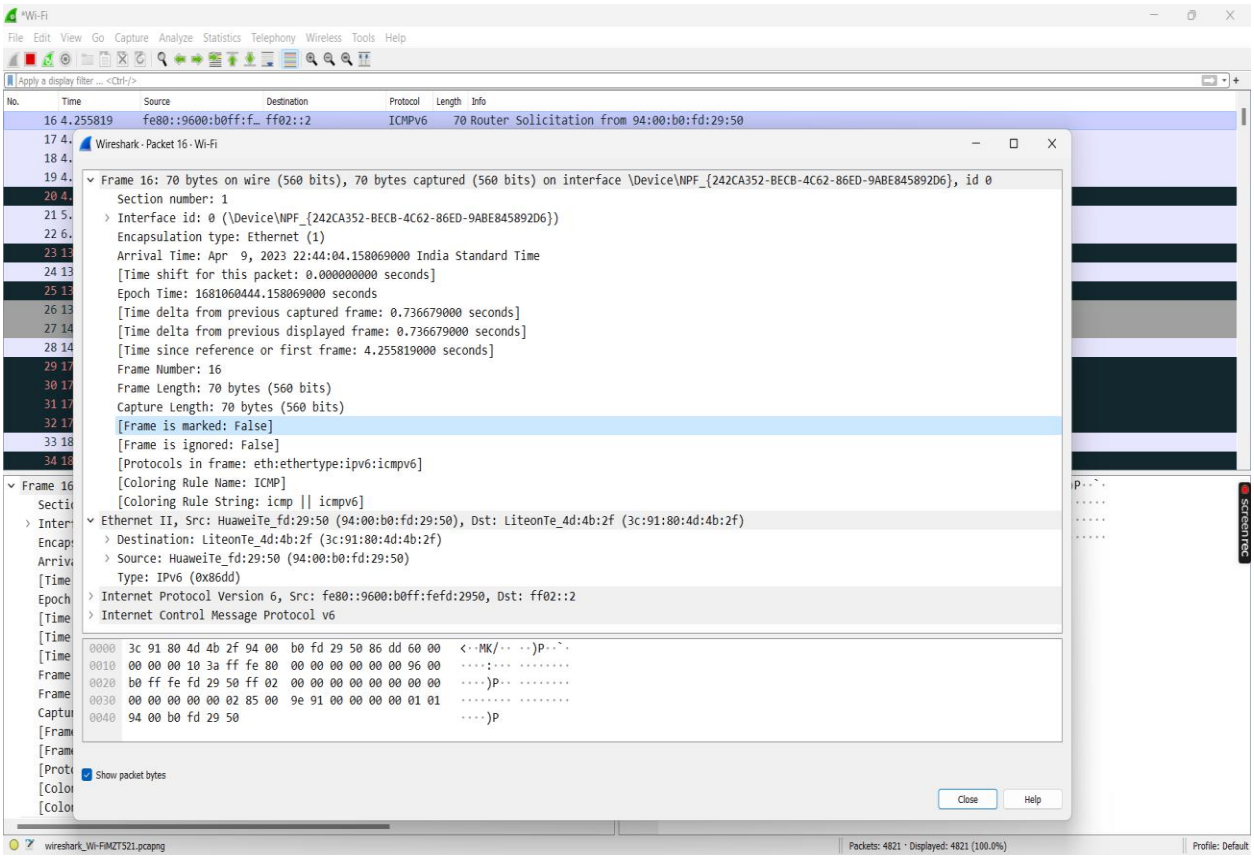
5) ICMPv6

ICMPv6 (Internet Control Message Protocol version 6) is a protocol used in IPv6 networks for various purposes, including error reporting, network testing, and neighbor discovery.

When capturing network traffic using Wireshark, ICMPv6 packets can be identified by their protocol type, which is 58 in decimal or 0x3A in hexadecimal.

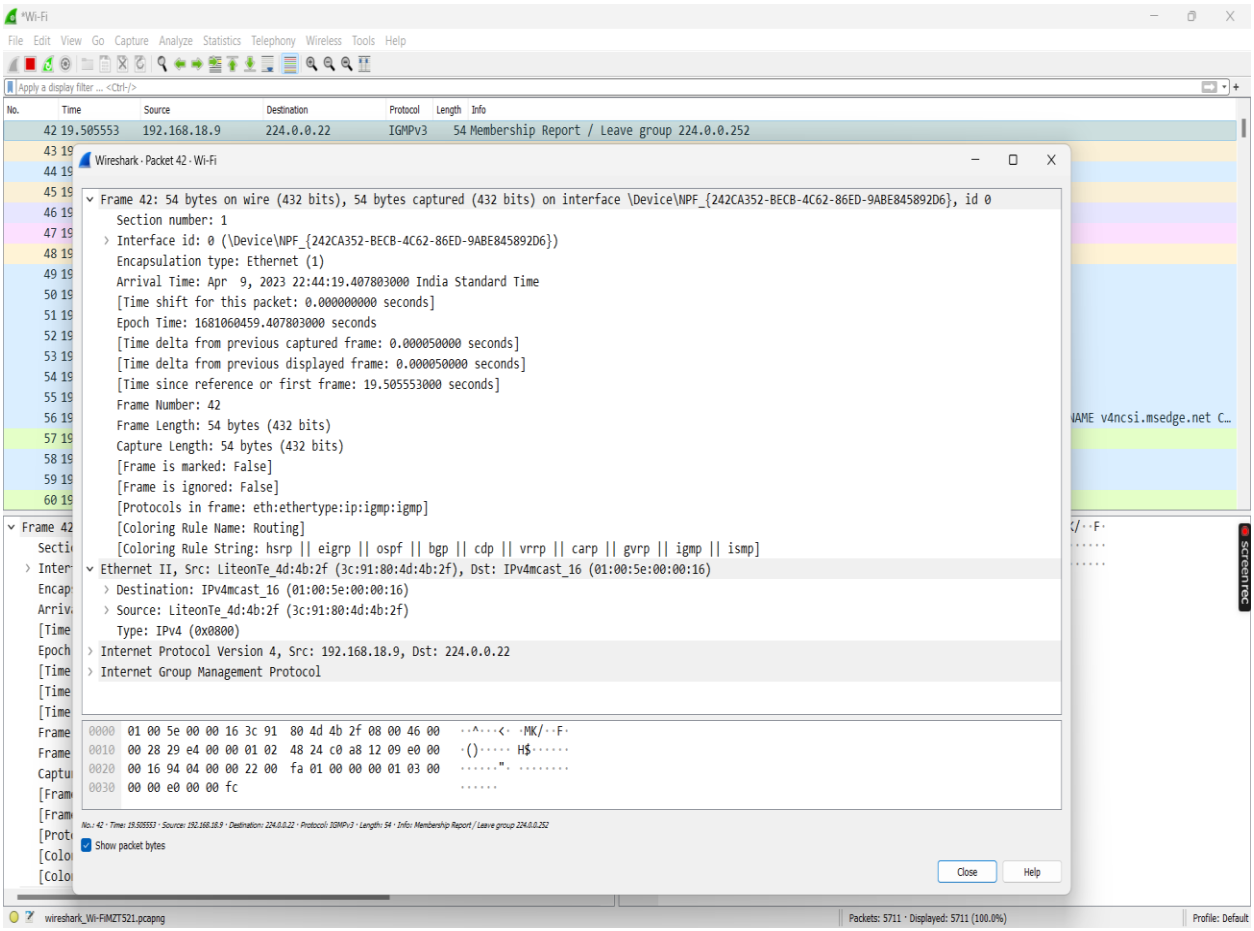
In Wireshark, ICMPv6 packets are displayed in the packet list pane and can be expanded to show various details such as the ICMPv6 message type, code, and payload. Common ICMPv6 message types include echo request and reply (used for network testing), router solicitation and advertisement (used for router discovery), and neighbor solicitation and advertisement (used for neighbor discovery).

Wireshark also provides various display filters for ICMPv6 packets, allowing you to filter and analyze specific types of ICMPv6 traffic in your network captures.



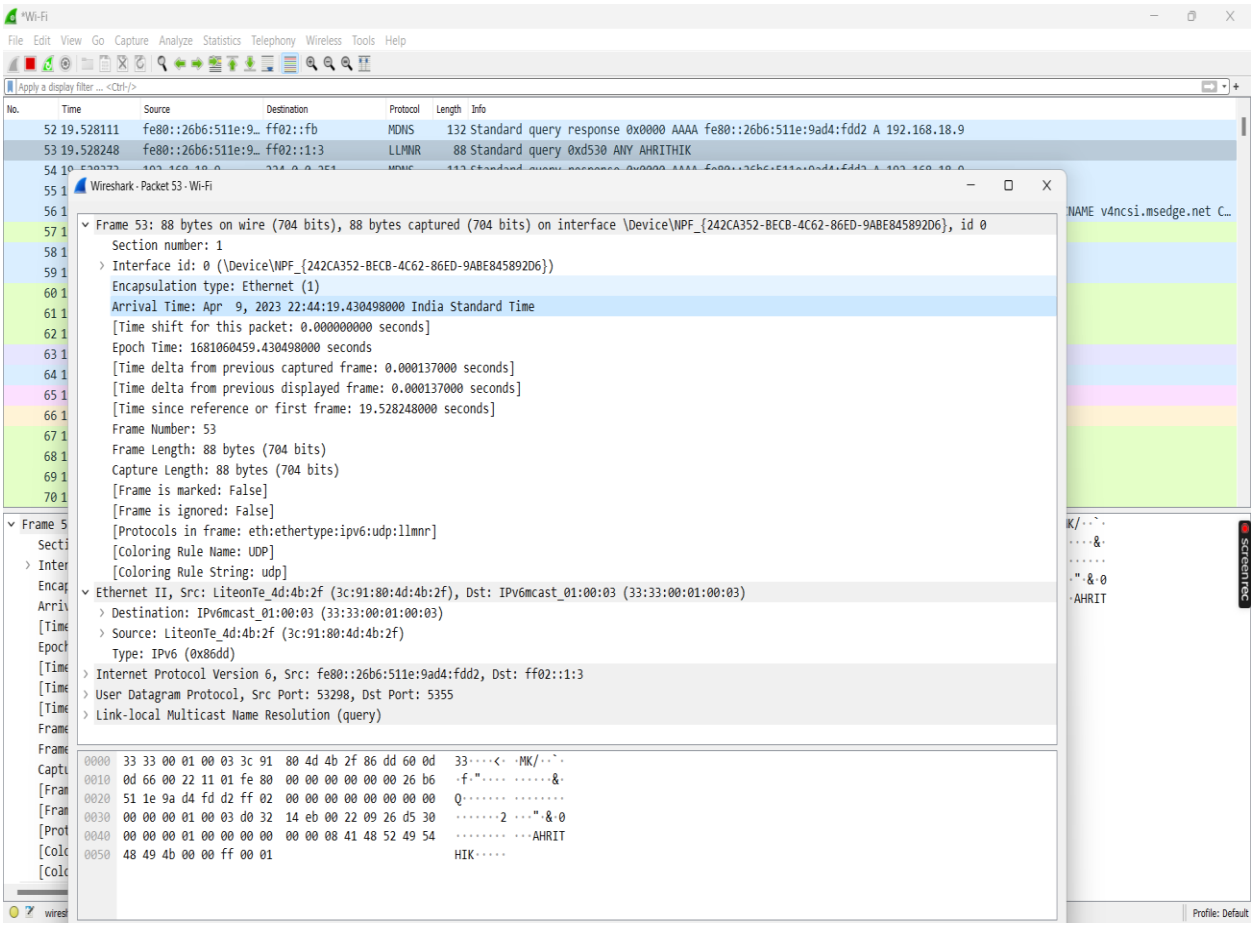
6) IGMPv3

The Internet Group Management Protocol version 3 (IGMPv3) is a network-layer protocol used by hosts to join or leave multicast groups on a network. In Wireshark, IGMPv3 packets can be captured and analyzed to monitor multicast group membership and troubleshoot multicast-related issues. IGMPv3 includes several improvements over previous versions, such as support for source-specific multicast and the ability to filter multicast traffic at the source. It operates on top of the Internet Protocol (IP) and is used in conjunction with multicast routing protocols such as Protocol Independent Multicast (PIM).



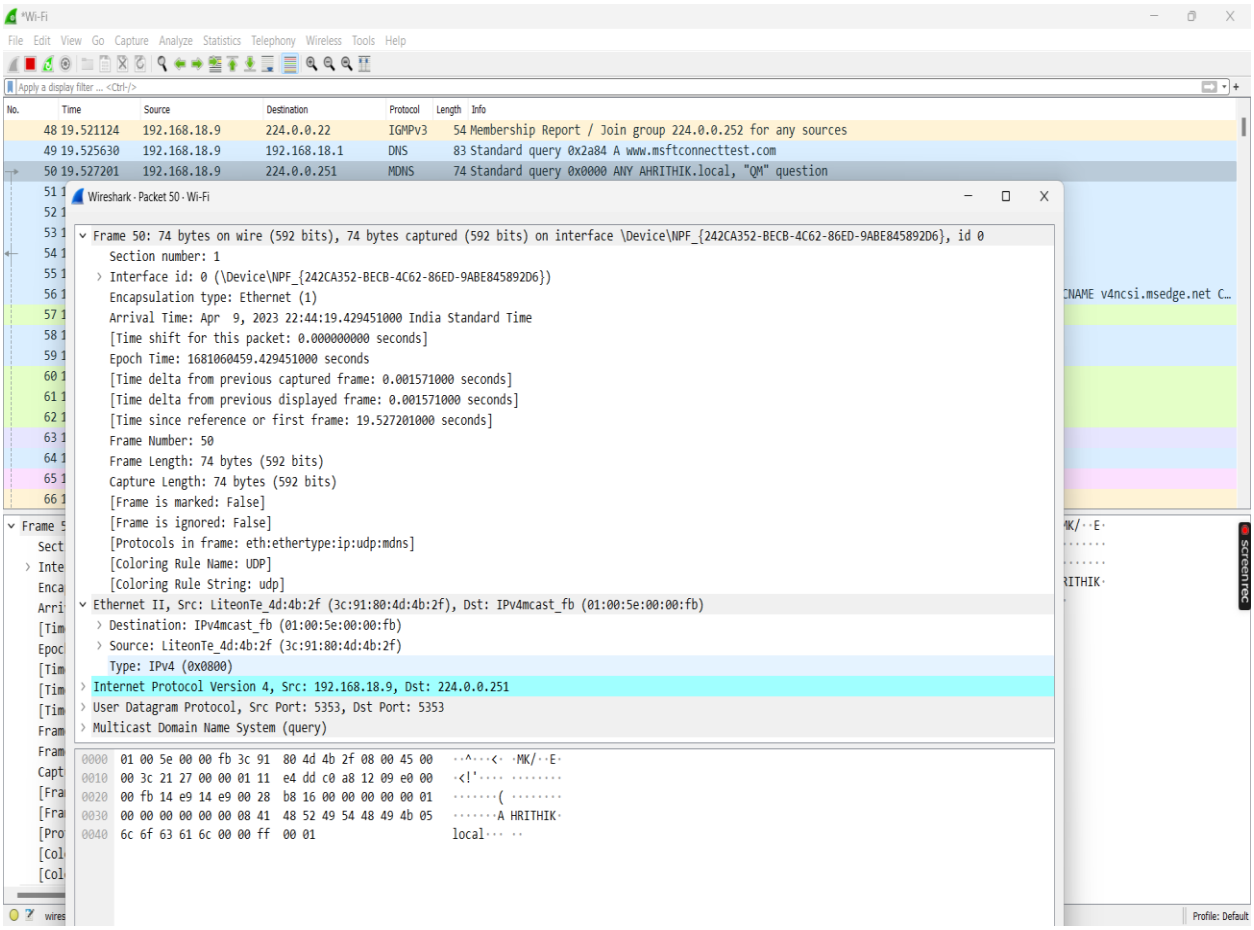
7) LLMNR

Link-Local Multicast Name Resolution (LLMNR) is a protocol used in Microsoft Windows networks to resolve the domain name of a device when traditional Domain Name System (DNS) is unavailable. LLMNR operates on the local network segment and uses multicast messages to query other devices on the network for the IP address associated with a specific hostname. In Wireshark, LLMNR packets can be identified by their protocol type, source and destination IP addresses, and the LLMNR message type, such as a query or response.



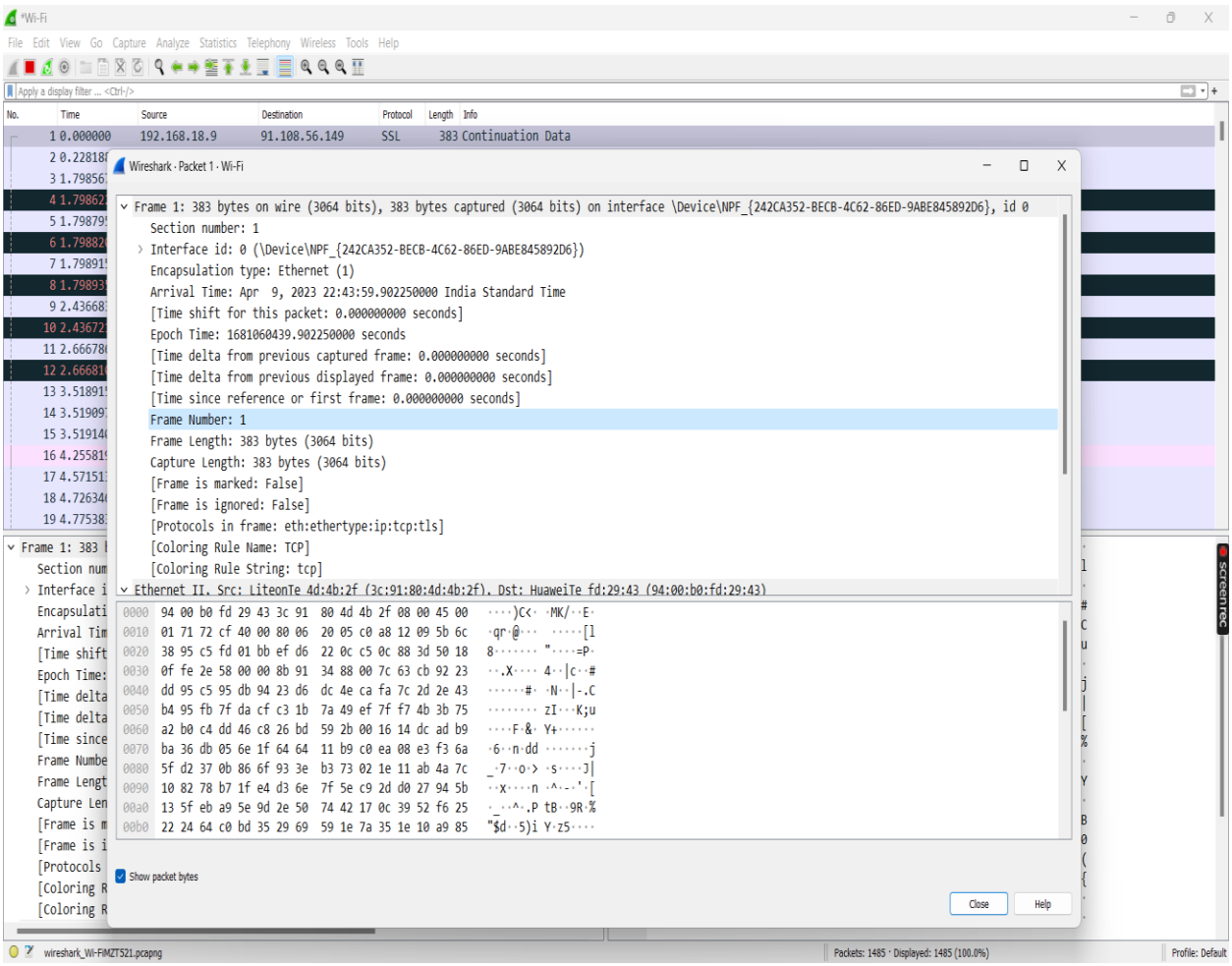
8) MDNS

The Multicast Domain Name System (mDNS) protocol allows devices on a network to discover and communicate with each other using domain names without the need for a central DNS server. In Wireshark, mDNS traffic can be identified by its destination address, which is a reserved multicast address of 224.0.0.251, and by the use of the User Datagram Protocol (UDP) on port 5353. Wireshark can capture and display mDNS packets, including the queries and responses used to discover and communicate with devices on the network.



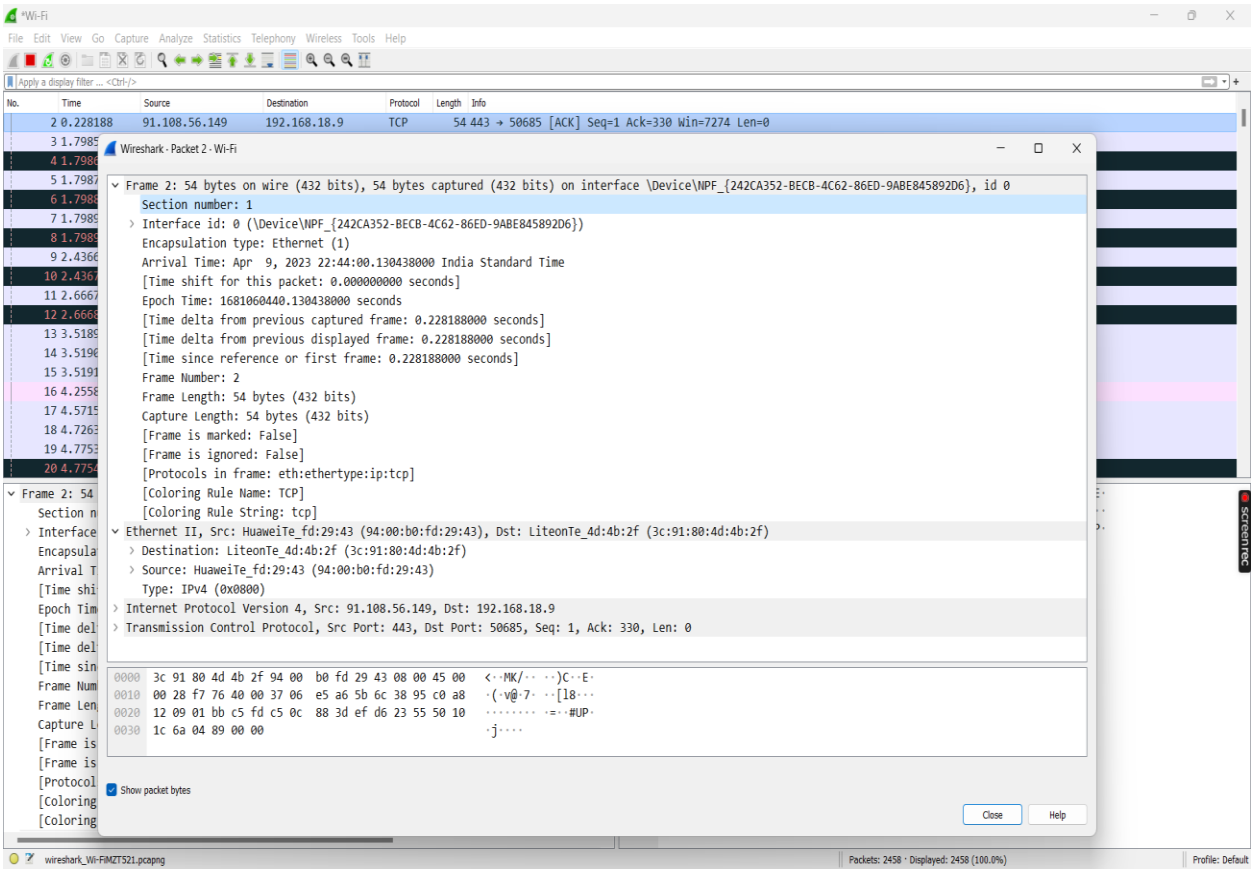
9) SSL

The SSL (Secure Sockets Layer) protocol, now commonly known as TLS (Transport Layer Security), is a widely used encryption protocol that provides a secure and private communication channel over the Internet. When capturing network traffic with Wireshark, SSL/TLS packets can be identified by their specific port number (usually 443 for HTTPS) and protocol identifier (either SSL or TLS). Once the SSL/TLS session is established, all data transmitted between the client and server is encrypted, making it difficult for eavesdroppers to intercept or decipher the information being transmitted. Wireshark can be used to analyze SSL/TLS traffic, including identifying the cryptographic algorithms and keys used to encrypt the data, as well as any potential vulnerabilities or errors in the SSL/TLS implementation.



10) TCP

TCP (Transmission Control Protocol) is a reliable, connection-oriented protocol used for transmitting data over a network. It provides error-checking, flow control, and congestion avoidance to ensure that data is delivered in order, without loss or duplication. In Wireshark, TCP packets can be identified by their source and destination IP addresses and port numbers, and the various flags in the TCP header can provide information about the status of the connection, such as whether a packet has been acknowledged or whether a connection is being established or terminated.



We analyzed the captured data for each protocol and identified the type of data being transmitted. For example, DNS, we found that most of the data being transmitted was domain name resolution queries.

We also read live data from a Bluetooth device and identified the following protocols:

- Bluetooth LE

- Bluetooth Classic

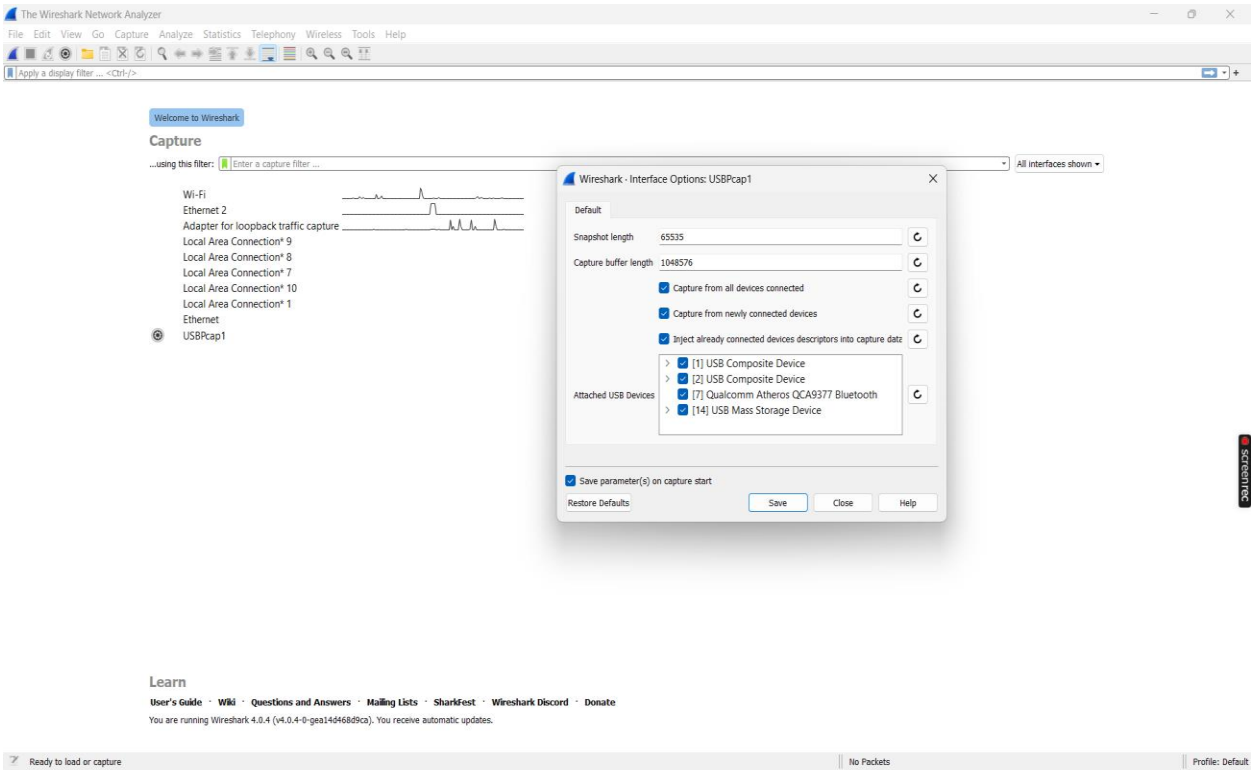
We analyzed the data being transmitted over Bluetooth and found that most of the data was related to device pairing and synchronization.

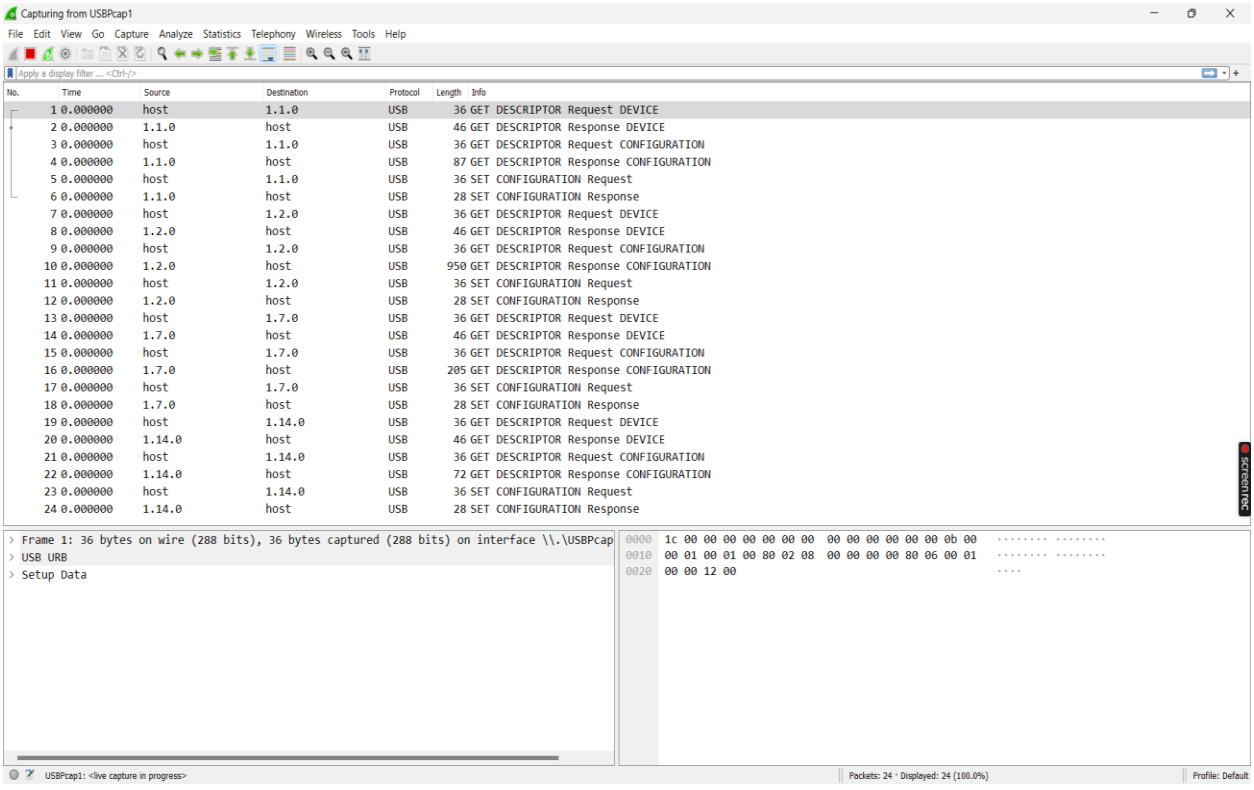
We then read live data from a USB device and identified the following protocols:

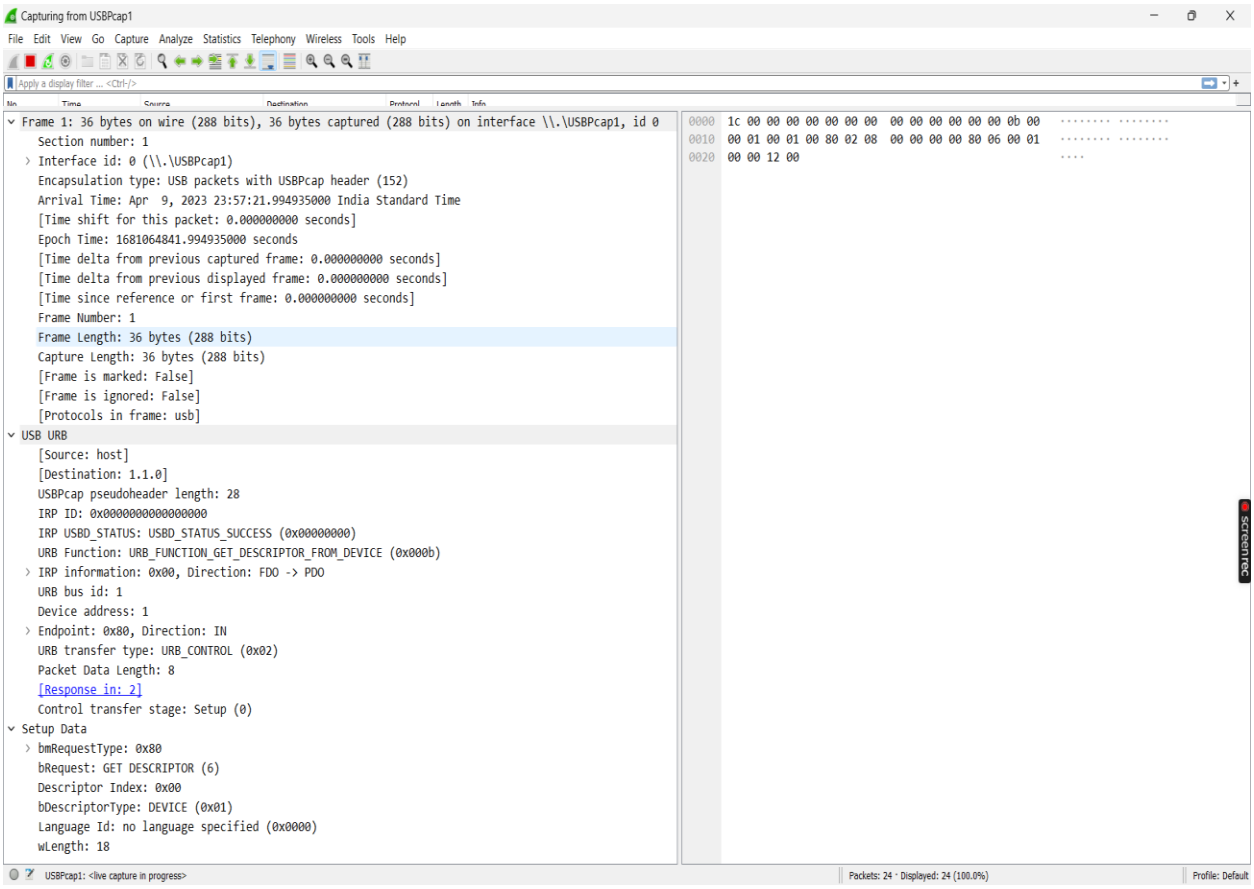
- USB 1.1

- USB 2.0

- USB 3.0


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter: Enter a capture filter ...
All interfaces shown
Wi-Fi
Ethernet 2
Adapter for loopback traffic capture
Local Area Connection* 9
Local Area Connection* 8
Local Area Connection* 7
Local Area Connection* 10
Local Area Connection* 1
Ethernet
USBPcap1
Wireshark · Interface Options: USBPcap1
Default
Snapshot length 65535
Capture buffer length 1048576
Capture from all devices connected
Capture from newly connected devices
Inject already connected devices descriptors into capture data
Attached USB Devices
[1] USB Composite Device
[2] USB Composite Device
[7] Qualcomm Atheros QCA9377 Bluetooth
[14] USB Mass Storage Device
Save parameter(s) on capture start
Restore Defaults
Save
Close
Help
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate
You are running Wireshark 4.0.4 (v4.0.4-0-gea14d468d9ca). You receive automatic updates.
Ready to load or capture
No Packets
Profile: Default


Capturing from USBPcap1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 host 1.1.0 USB 36 GET DESCRIPTOR Request DEVICE
2 0.000000 1.1.0 host USB 46 GET DESCRIPTOR Response DEVICE
3 0.000000 host 1.1.0 USB 36 GET DESCRIPTOR Request CONFIGURATION
4 0.000000 1.1.0 host USB 87 GET DESCRIPTOR Response CONFIGURATION
5 0.000000 host 1.1.0 USB 36 SET CONFIGURATION Request
6 0.000000 1.1.0 host USB 28 SET CONFIGURATION Response
7 0.000000 host 1.2.0 USB 36 GET DESCRIPTOR Request DEVICE
8 0.000000 1.2.0 host USB 46 GET DESCRIPTOR Response DEVICE
9 0.000000 host 1.2.0 USB 36 GET DESCRIPTOR Request CONFIGURATION
10 0.000000 1.2.0 host USB 950 GET DESCRIPTOR Response CONFIGURATION
11 0.000000 host 1.2.0 USB 36 SET CONFIGURATION Request
12 0.000000 1.2.0 host USB 28 SET CONFIGURATION Response
13 0.000000 host 1.7.0 USB 36 GET DESCRIPTOR Request DEVICE
14 0.000000 1.7.0 host USB 46 GET DESCRIPTOR Response DEVICE
15 0.000000 host 1.7.0 USB 36 GET DESCRIPTOR Request CONFIGURATION
16 0.000000 1.7.0 host USB 205 GET DESCRIPTOR Response CONFIGURATION
17 0.000000 host 1.7.0 USB 36 SET CONFIGURATION Request
18 0.000000 1.7.0 host USB 28 SET CONFIGURATION Response
19 0.000000 host 1.14.0 USB 36 GET DESCRIPTOR Request DEVICE
20 0.000000 1.14.0 host USB 46 GET DESCRIPTOR Response DEVICE
21 0.000000 host 1.14.0 USB 36 GET DESCRIPTOR Request CONFIGURATION
22 0.000000 1.14.0 host USB 72 GET DESCRIPTOR Response CONFIGURATION
23 0.000000 host 1.14.0 USB 36 SET CONFIGURATION Request
24 0.000000 1.14.0 host USB 28 SET CONFIGURATION Response
Frame 1: 36 bytes on wire (288 bits), 36 bytes captured (288 bits) on interface \\.\USBPcap
USB URB
Setup Data
0000 1c 00 00 00 00 00 00 00 00 00 00 00 00 00 0b 00
0010 00 01 00 01 00 80 02 08 00 00 00 00 80 06 00 01
0020 00 00 12 00
USBPcap1: <live capture in progress>
Packets: 24 · Displayed: 24 (100.0%)
Profile: Default



We analyzed the data being transmitted over USB and found that most of the data was related to file transfer and device synchronization.

Overall, we found that Wireshark was a powerful tool for analyzing network traffic at the microscopic level. It allowed us to identify the various protocols being used and analyze the data being transmitted. This information can be useful for network administrators to diagnose problems, improve network performance, and ensure that their network is secure.

4. Reference/Bibliography:

- Wireshark User Guide: https://www.wireshark.org/docs/wsug_html/

- Wireshark Tutorials: https://www.wireshark.org/docs/tshark/

- Bluetooth Protocol: https://www.bluetooth.com/specifications/bluetooth-core-specification/

- USB Protocol: https://www.usb.org/document-library/usb-20-specification