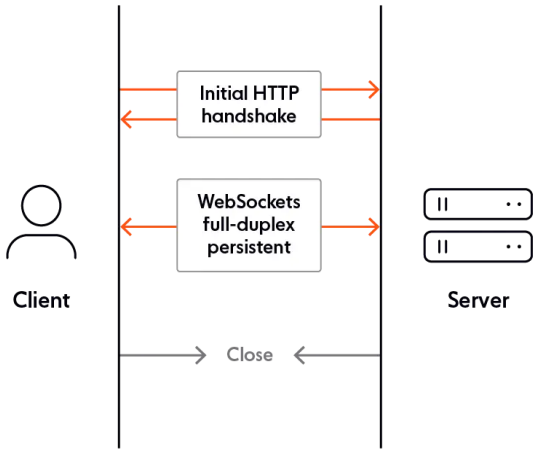
WebSocket Protocol

Abhishek Raj, En-21114004

Overview:

HTTP and WebSocket are both communication protocols used in client-server communication. HTTP is unidirectional, where the client sends the request, and the server sends the response. At the same time, WebSocket is bidirectional, a full-duplex protocol that is used in the same scenario of client-server communication. WebSocket communication involves a handshake, messaging (sending and receiving messages), and closing the connection.



Why WebSocket are used?

The advantage of WebSocket is that it enables real-time communication between the client and server without frequent HTTP requests/responses. This brings benefits such as reduced latency and improved performance and responsiveness of web apps.

Due to its persistent and bidirectional nature, the WebSocket protocol is more flexible than HTTP when building real-time apps that require frequent data

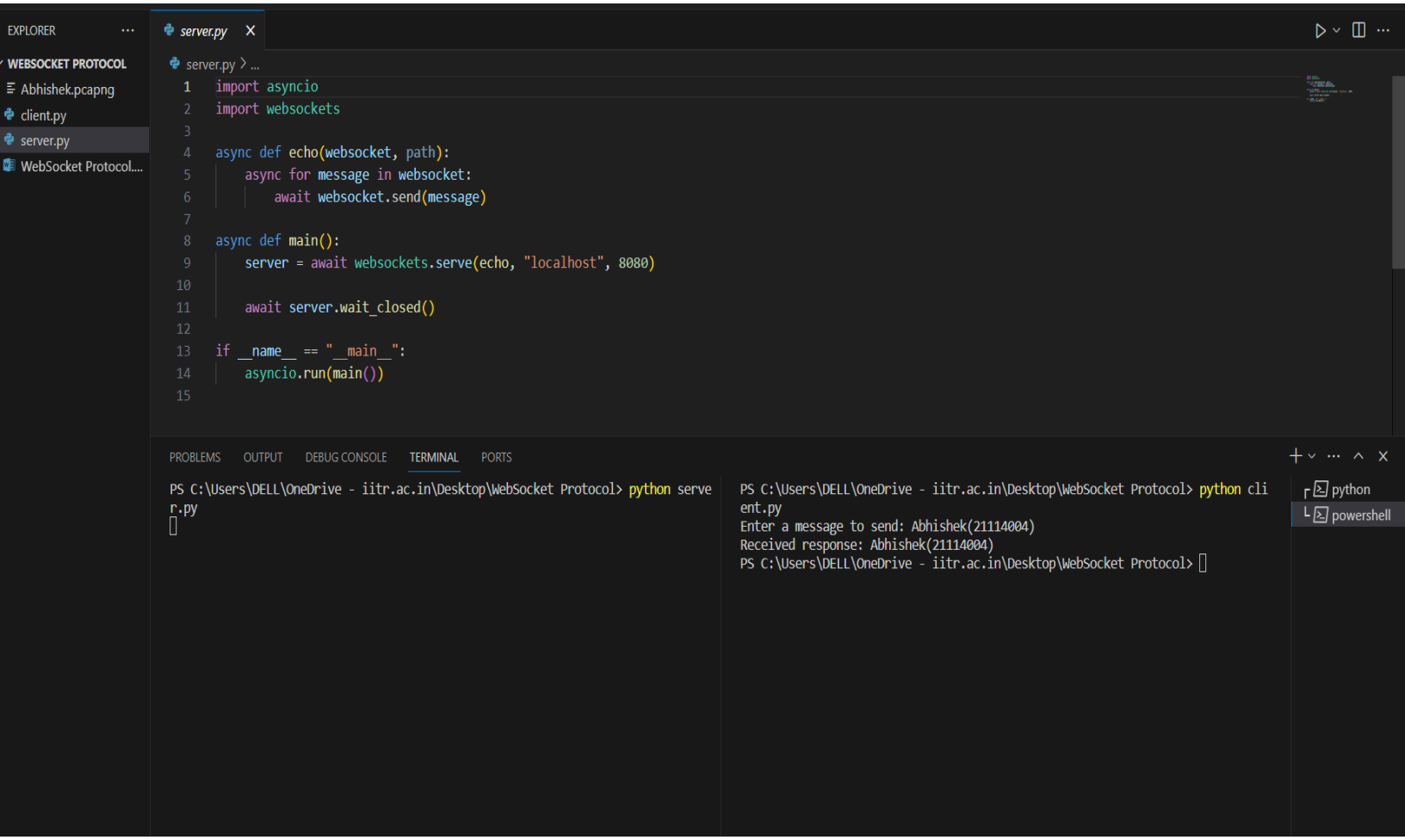
exchanges. WebSocket is also more efficient, transmitting data without repetitive HTTP headers and handshakes. This can reduce bandwidth usage and server load.

Use cases of WebSocket:

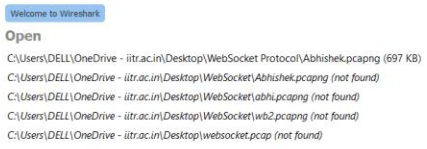
- Chatting
- Live feeds
- Multiplayer gaming
- Real-time Location Tracking
- Data synchronization (in Bitcoin)

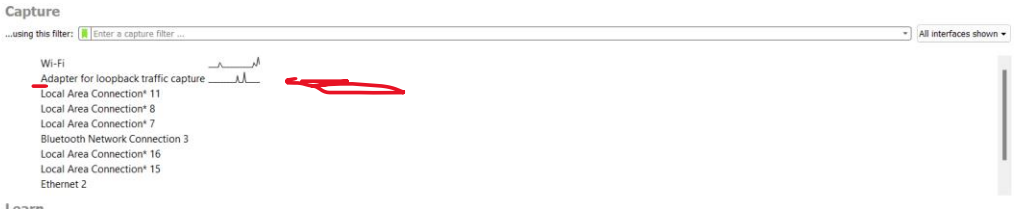
Analyze the WebSocket traffic using Wireshark.

We make WebSocket server and Client file, then pass the message I.e., Abhishek (21114004).

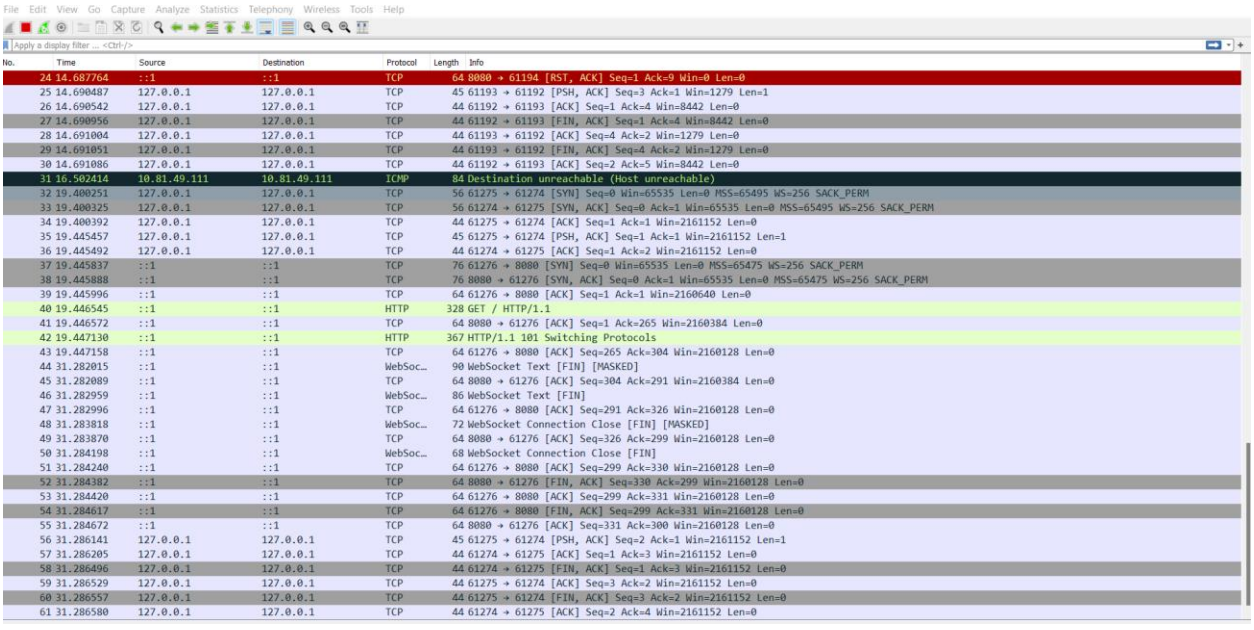


To capture the traffic for Wireshark and then analyze it. So, let's begin with capturing the traffic. When we open Wireshark, it displays all the available network interfaces with live network traffic.

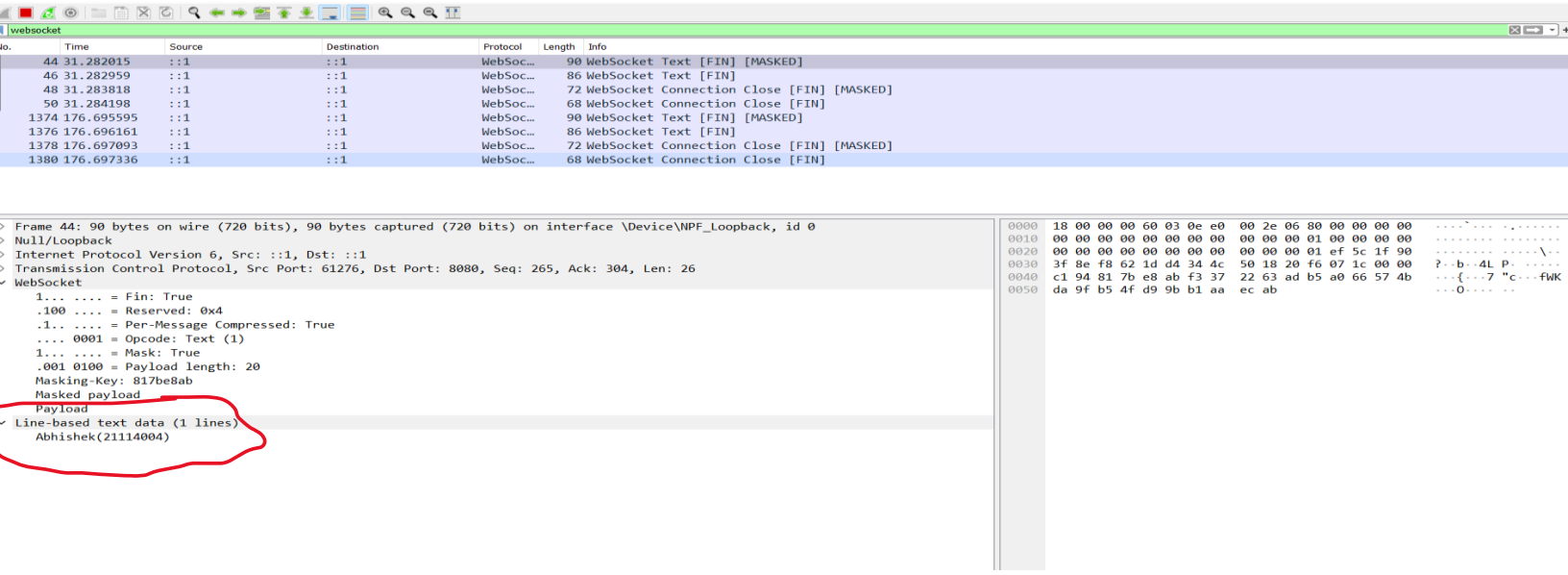




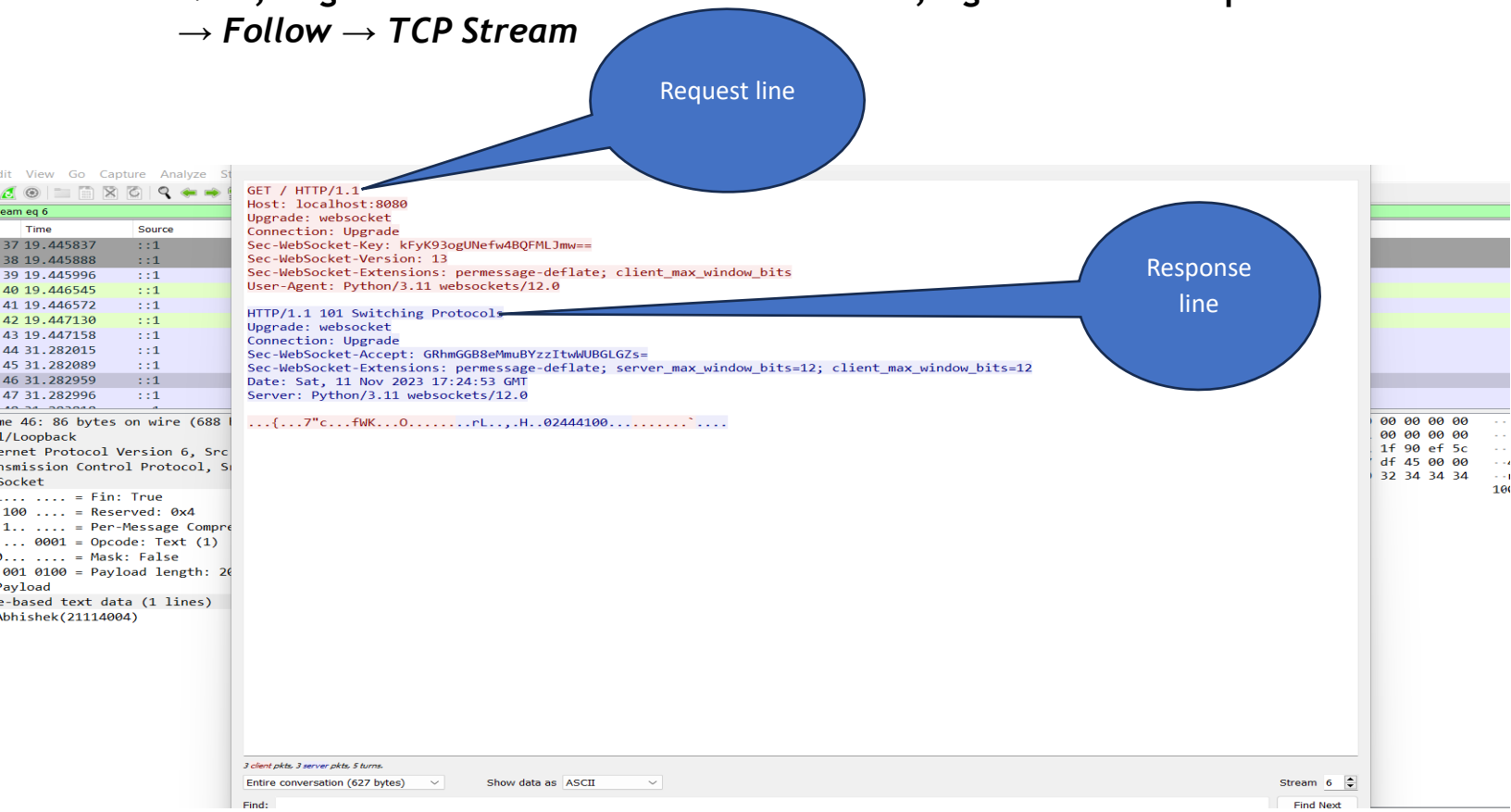
Here we choose Adapter for loopback traffic capture and then capture the packets.



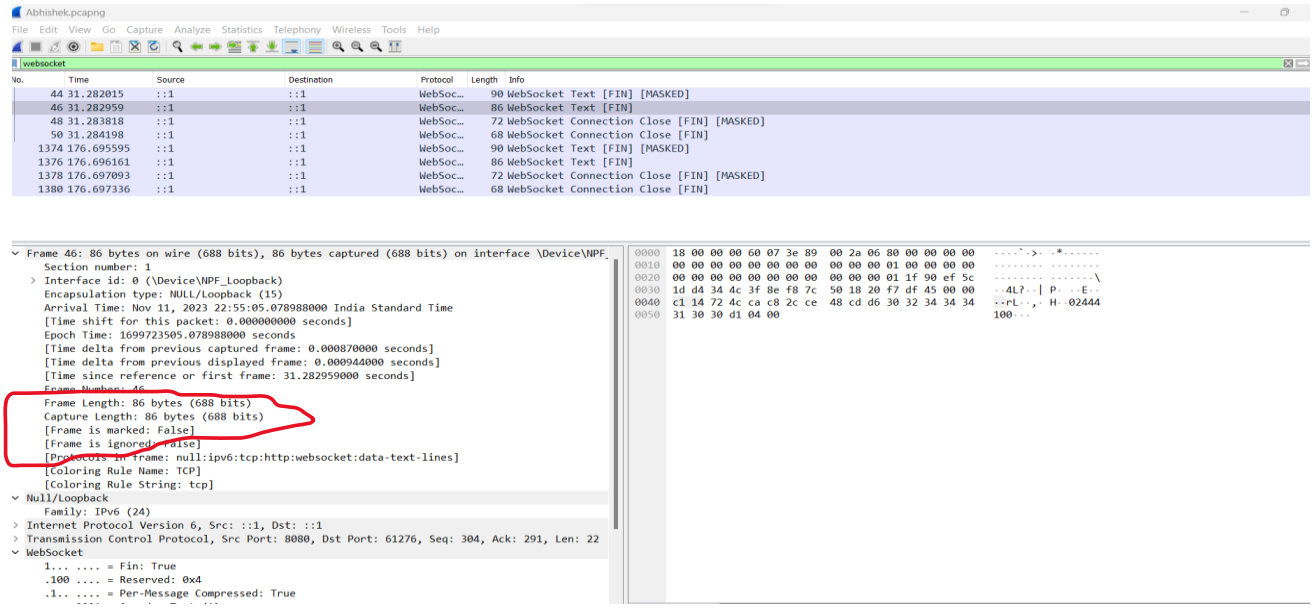
Let's try to analyze the network packets captured so far. First, since the initial handshake is on HTTP protocol, let's filter the packets for ***WebSocket*** protocol.



Next, to get a detailed view of the handshake, right-click on the packet → *Follow* → *TCP Stream*



Analyzing the performance of WebSocket involves monitoring key metrics to understand how well your WebSocket-based application is performing. Here are some steps and considerations for analyzing WebSocket performance:



1. Packet Capture and Analysis:

- Use tools like Wireshark to capture WebSocket traffic.
- Analyze the captured packets to ensure that the WebSocket connections are being established correctly and that messages are being exchanged efficiently.

2. Latency Measurement:

- Analyze the packet timestamps to calculate the latency.

3. Throughput Analysis:

- Measure the throughput or data transfer rate of your WebSocket connections.
- Evaluate how efficiently your application can send and receive data over WebSocket.
- Calculate throughput manually by dividing the amount of data transferred by the time taken.