Computer Networks 2

Project 2 Report

Team 15

Katsarikas Georgios AM: 2019030139

Taouxis Georgios Apollon AM: 2019030011

Tzedakis Emmanouil AM: 2019030076

1. Introduction

The 2nd project's goal was the implementation of a Speedtest application using TCP sockets as a tool to measure the TCP Downlink, which refers to the data flow from the client side to the server side. The application follows a simple client-server architecture with another device used for packet sniffing and analysis by utilizing the Wi-Fi Doctor application of the previous project. The wireless network's performance was validated across different scenarios in terms of frequency bands, distance between the server and the access point as well as the server's mobility.

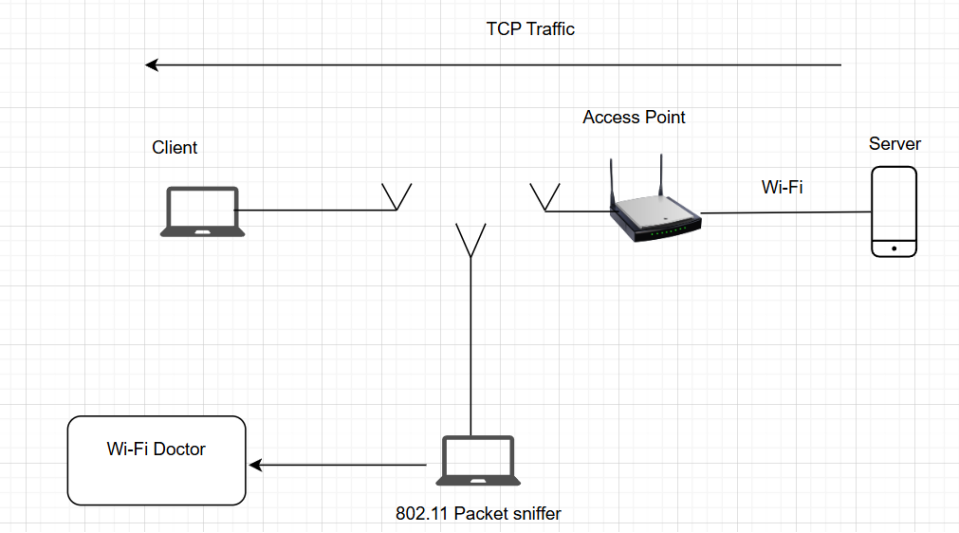
Each speedtest experiment lasts 30 seconds. The system's server prints the data sent rate in 2-second windows as well as the aggregated throughput after the end of the experiment. Additionally, packet sniffing is performed during this 30 second time-window and stores the captured traffic into a .pcap file which is then used by the Wi-Fi doctor for the acquisition of important metrics such as the RSSI,Throughput(estimated), Data rate, Frame loss, Rate gap . After running the same scenarios with **iperf,** we plotted the results for each metric and compared the findings with the ones produced by iperf to validate the application's reliability.

2. Design

The topology of our implementation consists of the following devices:

- A Windows 11 Laptop using WSL 2 as the client
- An Android mobile phone using Termux, an Android terminal application and Linux environment, acting as the server
- An Ubuntu 24.04 Laptop acting as the device for packet sniffing and analysis.

An overview of the system used for the speedtest application is shown below:



Implementation details

server.c: Implements a TCP server that receives a stream of data from the client. Initially, it creates a TCP socket with **socket(AF_INET,SOCK_STREAM,0),** defines socket options to allow address reuse and then binds to a specified port, listening for incoming connections.

Upon successfully running **accept(),** a new socket is created for for the communication with the client and the **handle_client()** proceeds to process the connection.

For the throughput measurement, it receives data in a loop with **recv()** into a byte-buffer of a fixed size. It tracks the cumulative bytes received, keeps track of time since start and reports every 2 seconds until the 30 second mark is reached.

Finally, it periodically prints the throughput for each time-window as well as the final average throughput and closes the client socket upon completion/disconnection.

client.c: Connects to the server and continuously sends data.
Initially, it attempts to connect to an IP address and port that are specified using the preprocessor directive **#define** and uses retry logic for connection establishment with a maximum of 3 attempts if the initial attempt fails.

Then, it fills a fixed-size buffer with the character ‘A’ and keeps on sending it using **send()** in a loop. It keeps track of the total bytes sent to compute the bit rate in Mbps and the Data sent in MB. It reports performance every 2 seconds and stops the data transmission after 30 seconds. It is also displaying connection status, the throughput at each interval and a message indicating the connection is over.

3. Evaluation

In order to evaluate our results, we initially used the following formula for the throughput estimation (Wi-Fi Doctor):

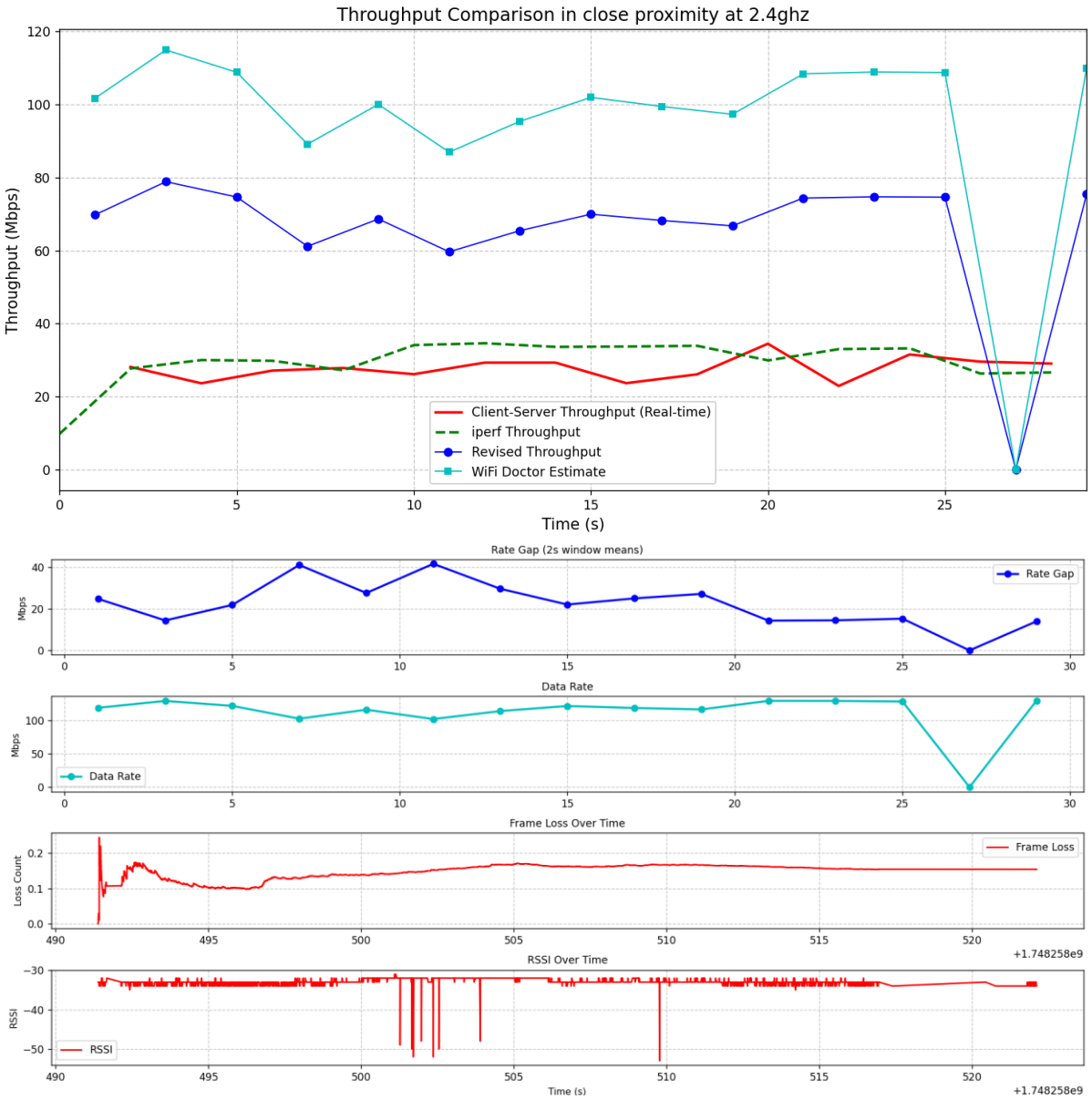
$$Throughput = DataRate \cdot (1 - FrameLossRate)$$

The revised throughput is calculated based on the **channel utilization** parameter and the theoretical throughput as shown below:

$$RevisedThroughput = TheoreticalThroughput \cdot ChannelUtil$$

Since the channel utilization is at most equal to 1, the revised throughput is almost always less than its theoretical counterpart.

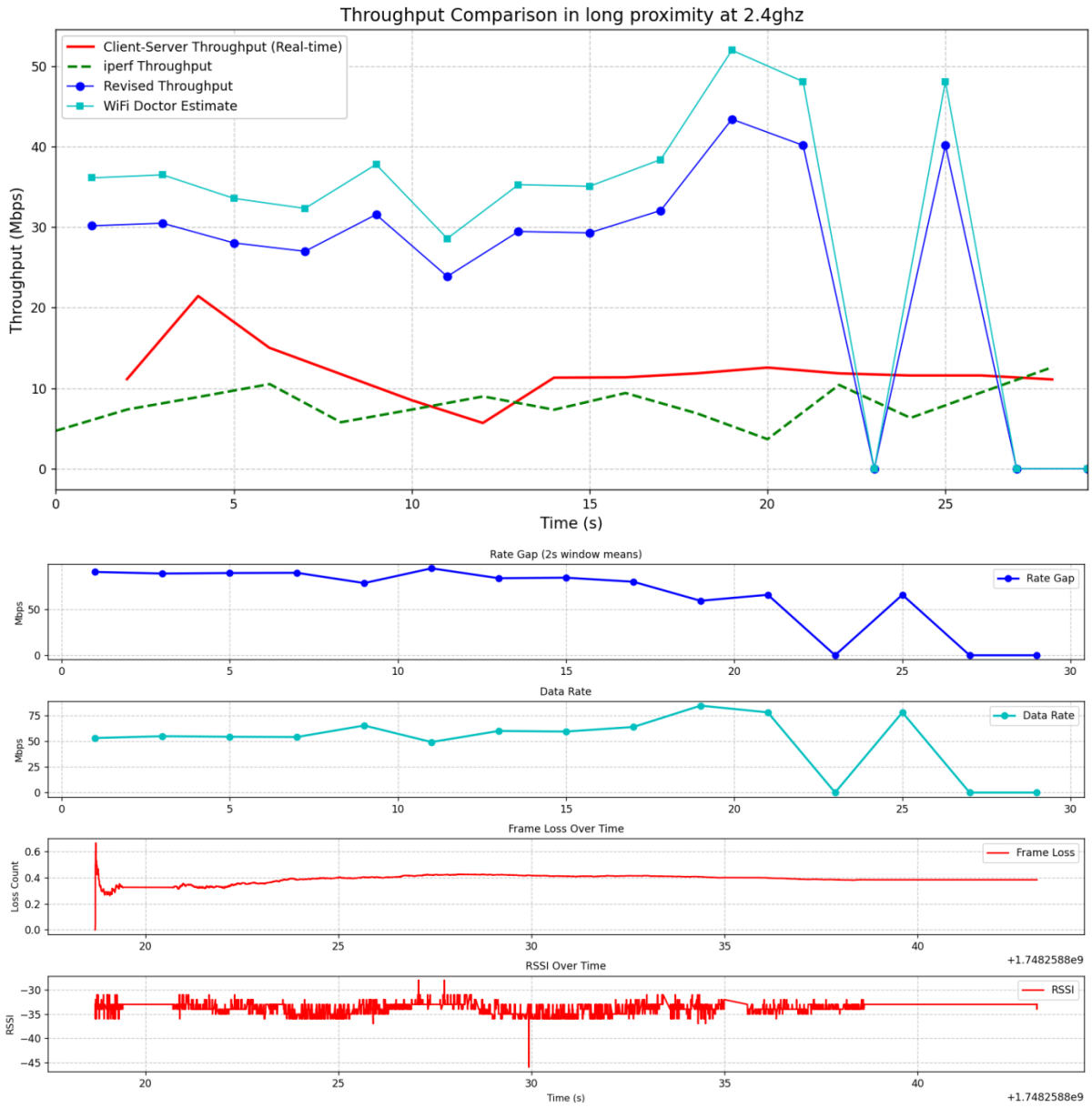
- **Scenario 1: 2.4GHz, server close to the access point**



In the first scenario, the speed test application showed the strongest and most stable performance with the highest overall throughput across all scenarios at 2.4GHz. The client-server throughput remains stable between 25-30 Mbps during the 30 second time-window apart from some small interruptions around the 25th second.

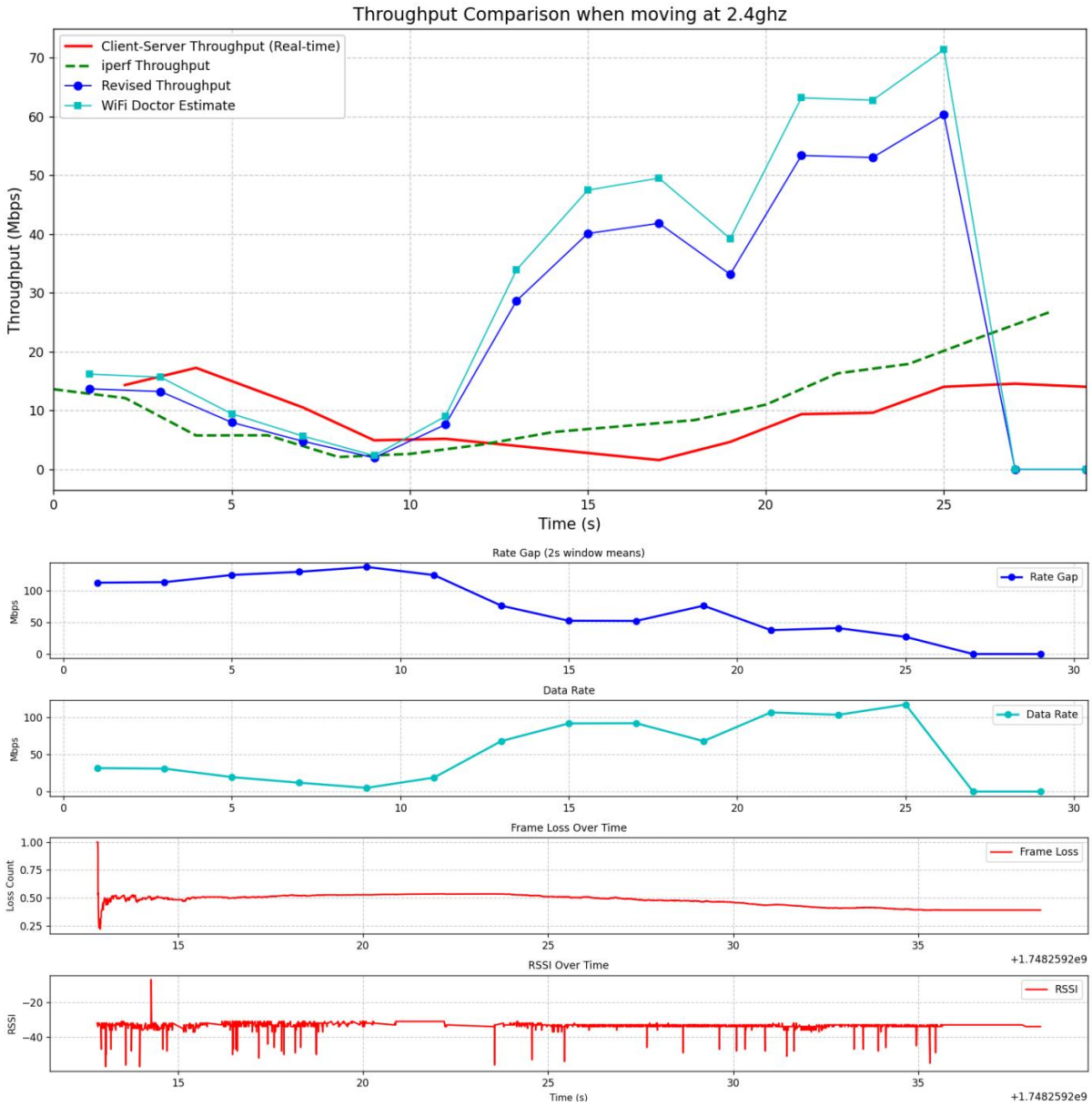
The revised throughput formula tracks the throughput in a more accurate way compared to the Wi-Fi doctor. Additionally, the sudden throughput drop during the final 2 seconds of the experiment, indicates a brief channel interference. However, the frame loss over time remained stable around 0.1 which confirms the good channel conditions.

- **Scenario 2: 2.4GHz, server far from the access point**



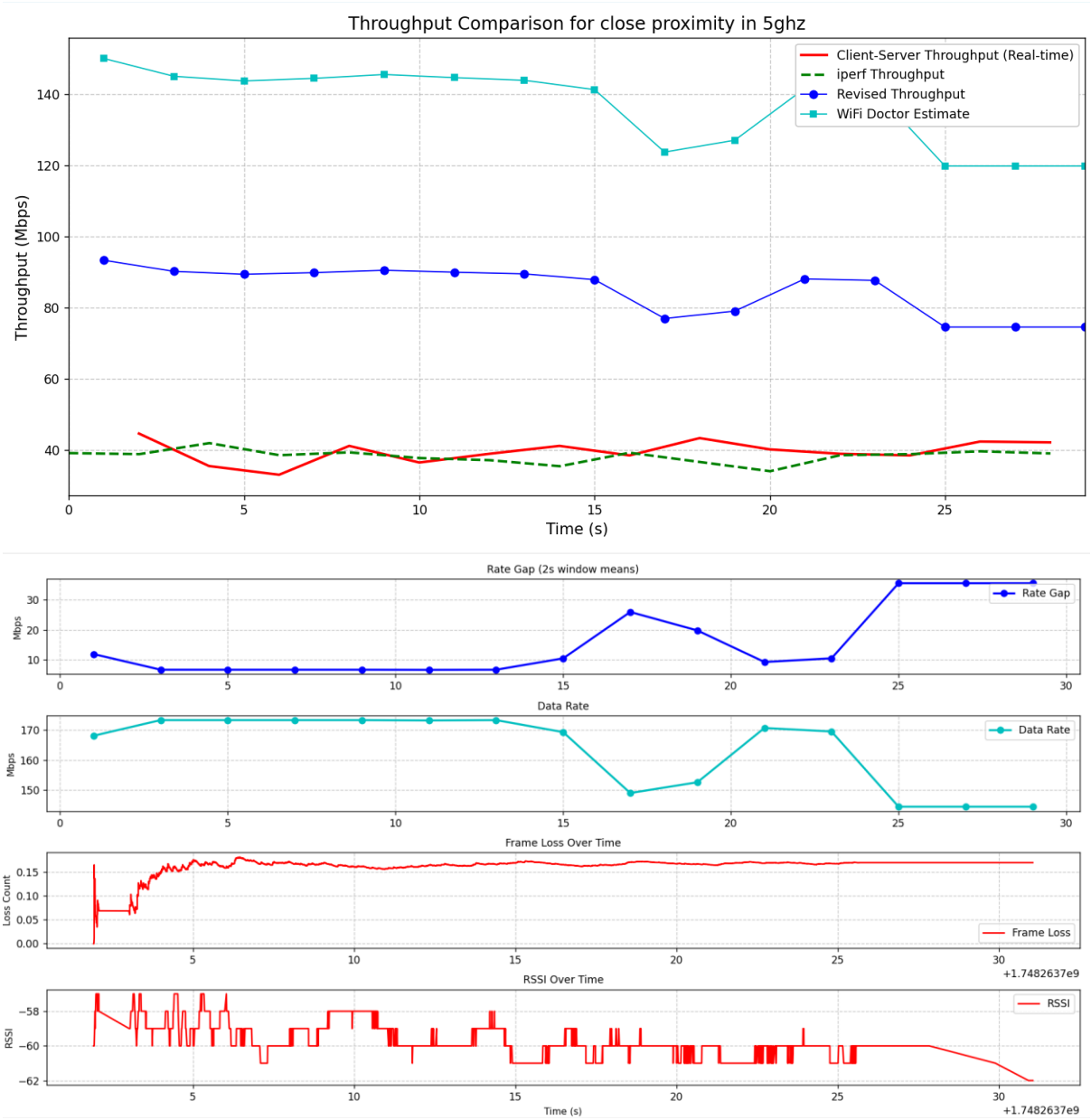
The second scenario shows significant performance degradation with a drop of 60% of the measured throughput compared to the close-proximity scenario. The throughput shows high variability across the first half of the experiment, but it remains stable around a value of 10 Mbps for the rest of it. This pattern is also confirmed by the increased frame loss rate (~40%) indicating low SNR. Compared to the previous experiment, the revised throughput and the Wi-Fi doctor estimation show an improvement in throughput calculation but both fail to predict the throughput variations since the server is now much further from the access point.

- **Scenario 3: 2.4GHz, server moving away from access point**



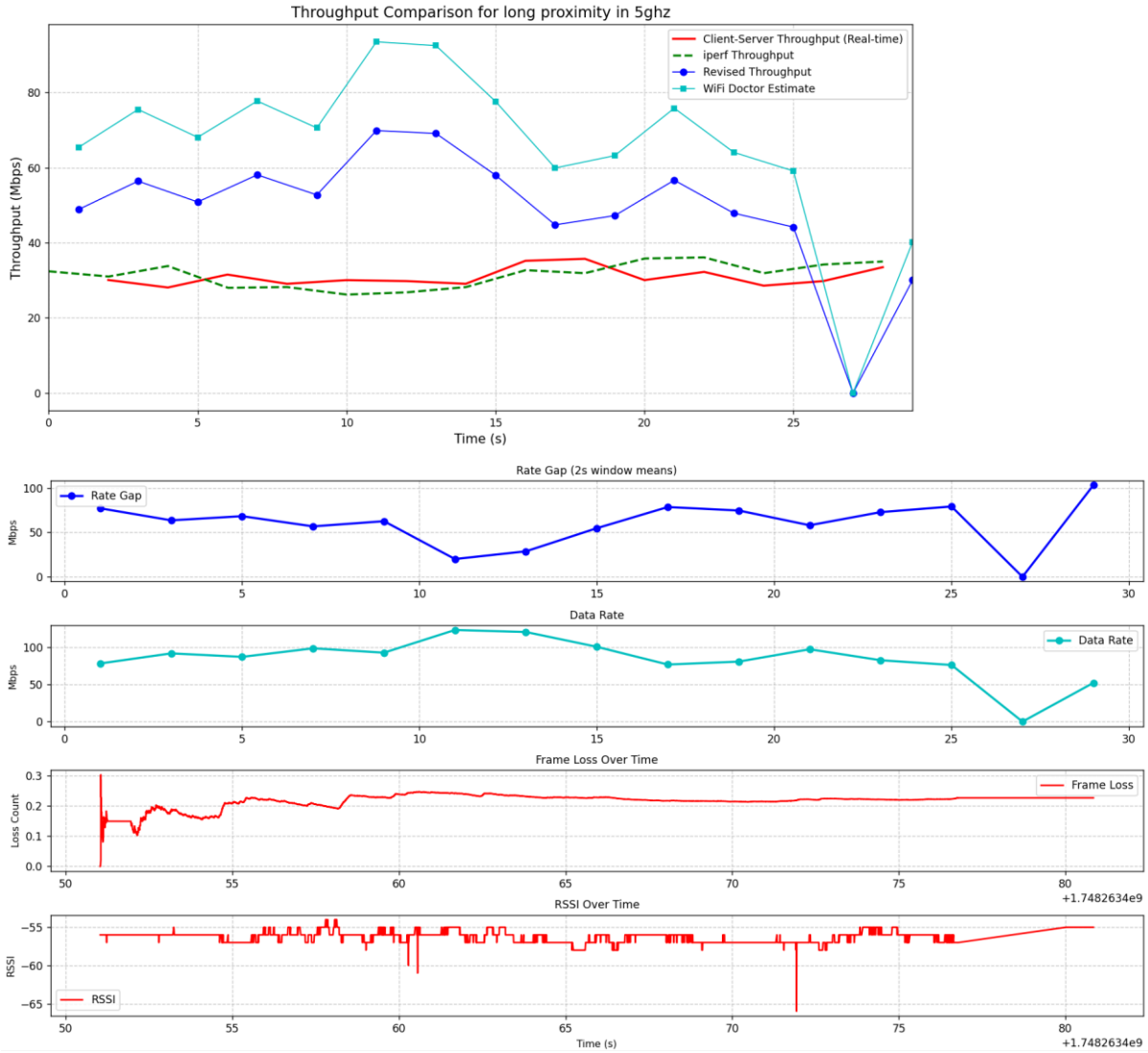
The third scenario aims to showcase the effect of varying distance between the server and the access point. In this experiment, the server is initially close to the access point, then moves away for the first 12 seconds and then returns close to the access point in a stable pace. The revised throughput and the Wi-Fi doctor estimations almost align with the results from the iperf tool and are close to the real-time client-server throughput.

- **Scenario 4: 5GHz, server close to the access point**



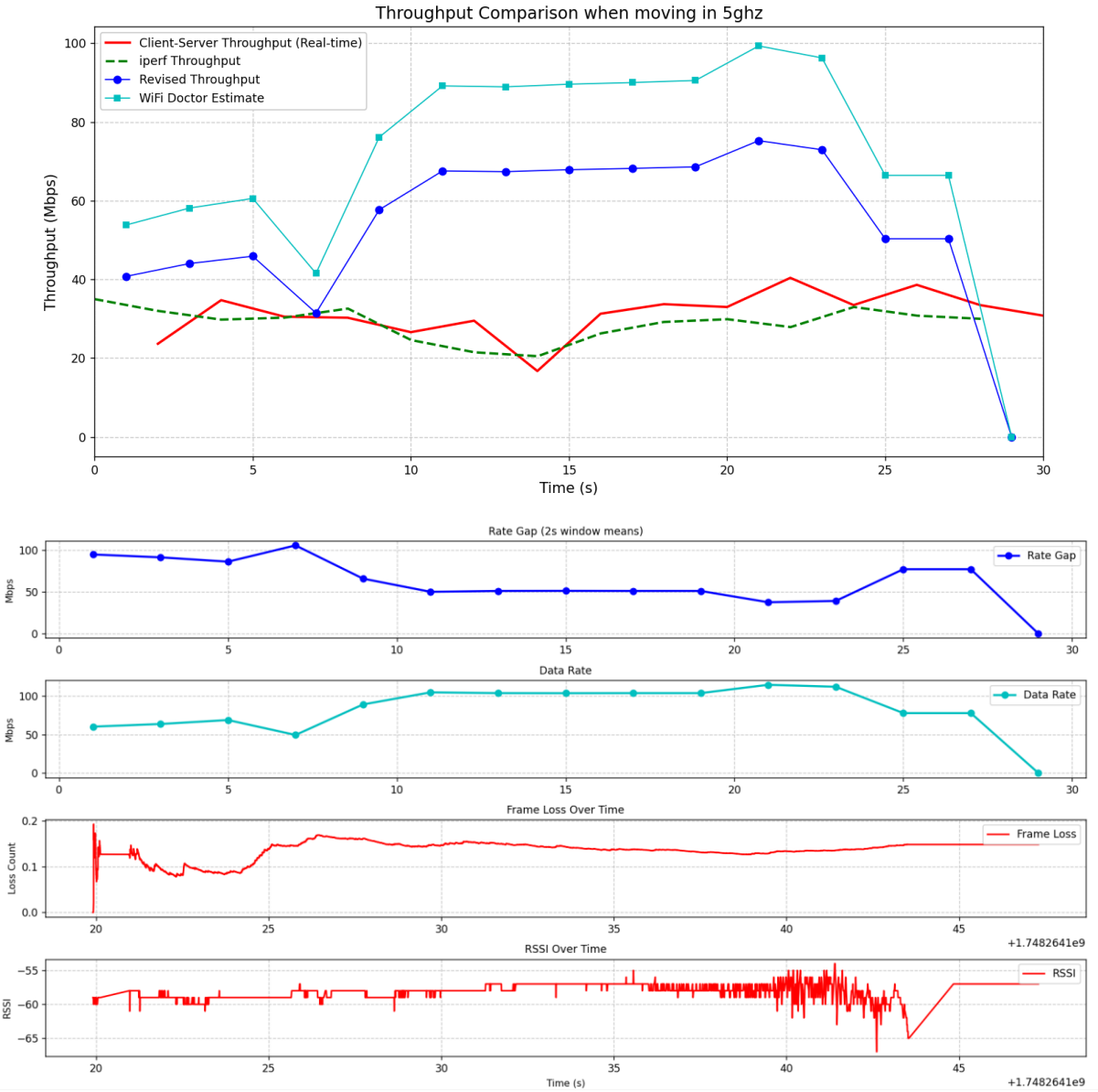
In a similar way to the 1st scenario, the throughput is consistently high compared to the other cases. The real-time throughput is close to the benchmark of iperf tool which proves the speed test application is accurate under ideal conditions. Even though the revised throughput estimation is not that accurate compared to the real-time and iperf results, it shows an improvement of almost 2x the estimation provided by the Wi-Fi doctor, suggesting that the real throughput does not depend solely on frame loss rate and data rate.

- **Scenario 5: 5GHz, server far from the access point**



The far-proximity scenario reveals a significant performance drop and high variability on the througput, both common characteristics of a low SNR connection. Despite its fluctuations between 28-32 Mbps, the real-time throughput mirrors the throughput calculated by the iperf tool. The estimations cannot provide fully accurate results for the throughput since they are not taking into consideration distance-related factors such as interferences.

- **Scenario 6: 5GHz, server moving away from the access point**



The final scenario proves how challenging the 5GHz networks are in non-static environments. The real-time throughput as well as the one calculated by iperf show sudden changes between 20 and 35 Mbps, thus making it difficult for the estimations to predict a correct value for the throughput. The Wi-Fi doctor estimation provides values with high variability, whereas the revised throughput is more conservatively calculated. Another important observation is the drop of the RSSI metric, indicating how difficult it is to maintain a strong connection on 5GHz networks as the distance between the server and the access point increases.

Key takeaways

- The values at the edges of the throughput estimation cannot be estimated in an accurate way due to the discrepancy between the device where the packet sniffer is running on and the client-server devices. Given the fact that it is impossible to start the packet sniffing at the exact moment the client-server connection is established, we expect to see some inaccuracies during the end of each experiment
- The channel utilization is used as factor for the revised throughput estimation due to the percentage of the channel utilization being empty, leading to the drop in the throughput value. This means that there was space for the transmission of more data, but it turned out to be unused.

4. References