Computer Networks 2

Project 1 Report

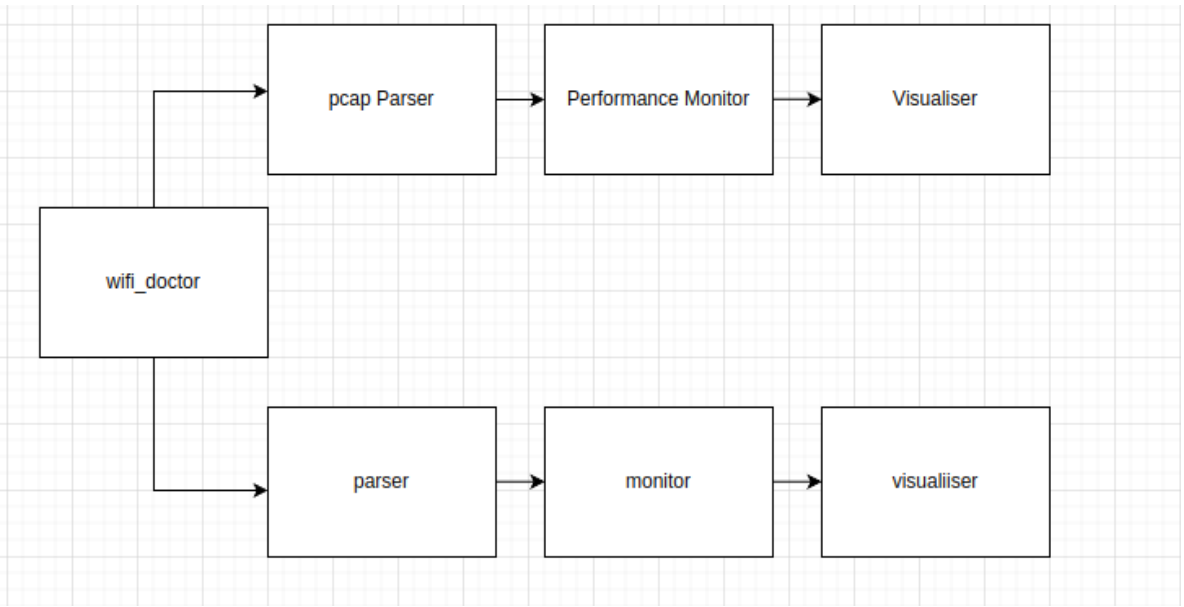
Katsarikas Georgios	**AM: 2019030139**
Taouxis Georgios Apollon	**AM: 2019030011**
Tzedakis Emmanouil	**AM: 2019030076**

1. Introduction

The first project's goal was the implementation of a 'Wi-Fi doctor', a program that offers insight information about a network. It offers the opportunity for data extraction, monitoring, analysis and visualization of metrics which can determine whether the network works optimally or not. The language for the tool's implementation is Python due to the plethora of available libraries for packet capturing, plotting and command line arguments integration.

2. Design

The project has the following structure:



i. Wifi_sniffer

Initially, the tool captures packets using the Wi-Fi sniffer module. Based on the user's choice of interface and channel it executes the command **sudo iwconfig <interface> channel <channel number>** and prints a message if it succeeds in doing so. Then, it runs **wifi_sniffer** using the

function **LiveCapture [3][4]** from the pyshark library and returns a packet capture (pcap) file and prints a message upon success. This function is executed as many times as the amount of channels specified by the user and returns a list of pcap files.

ii. Parser

Receives the list of pre-existing pcap files plus the ones captured by the sniffer and extracts the BSSID,Transmitter MAC, PHY, Channel, Frequency, Signal strength, SNR, Timestamp from the beacon frames [2]. It returns a list of text files with each containing all those parameters separated by '::'.

iii. Monitor

Receives the density files produced by the parser and proceeds to analyze the data to procure metrics in order to visualise network density as well as throughput and rate gap

iv. Visualizer

Proceeds to visualise the metrics produced by the monitor using pandas and matplotlib

v. Wi-Fi doctor

The file which contains the main function. If the user does not directly specify an interface to capture packets from, then the program proceeds to read from 2 pre-existing files: one from a home network and one from a university network, both dual-band. Then, it calls the parser function to extract data, monitor function to analyse them and the visualizer function to plot them. An example of a correct execution is **python3 wifi_doctor.py --interface wlan0mon --channels 1,6,11 –duration 10**

However, before running the tool, the user needs to make sure the interface is set on monitor mode. This is done by running the following commands:

```
sudo airmon-ng check kill
sudo airmon-ng start wlan0
```

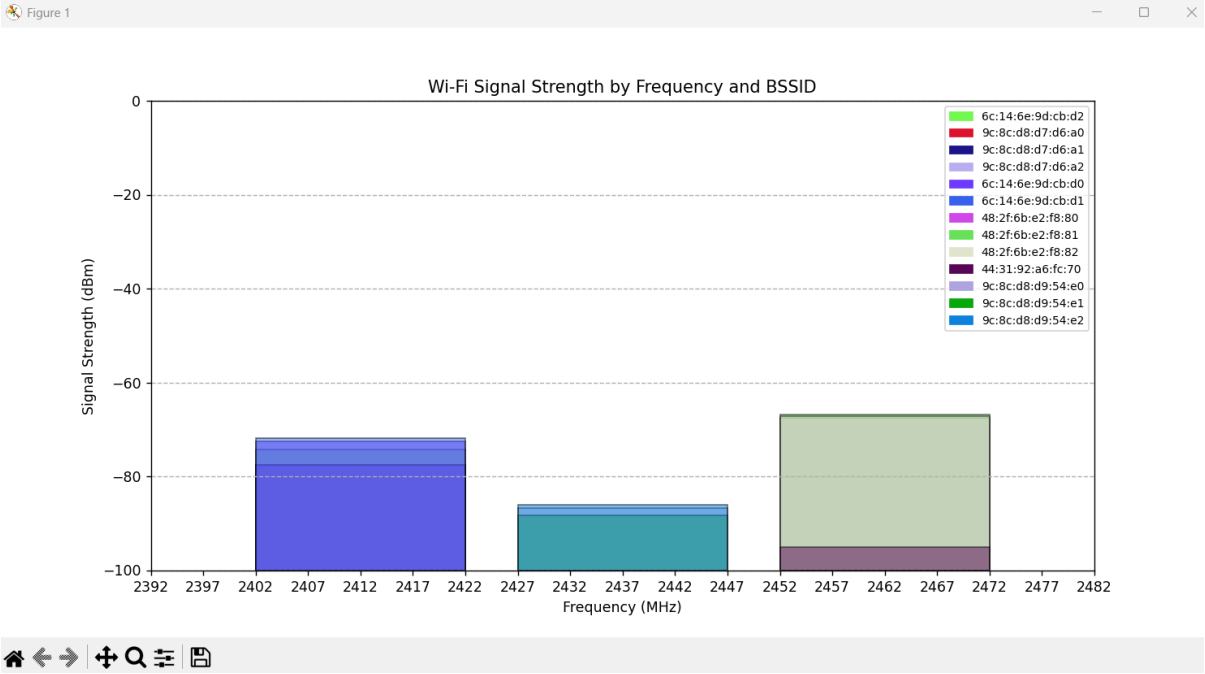
Furthermore, you can run **./setup.sh** to activate monitor mode, run the program with the parameters you want and then deactivate monitor mode. On windows, you can only run the offline mode by simply typing **python wifi_doctor.py** in the command line.

3. Evaluation

- **Density metrics evaluation**

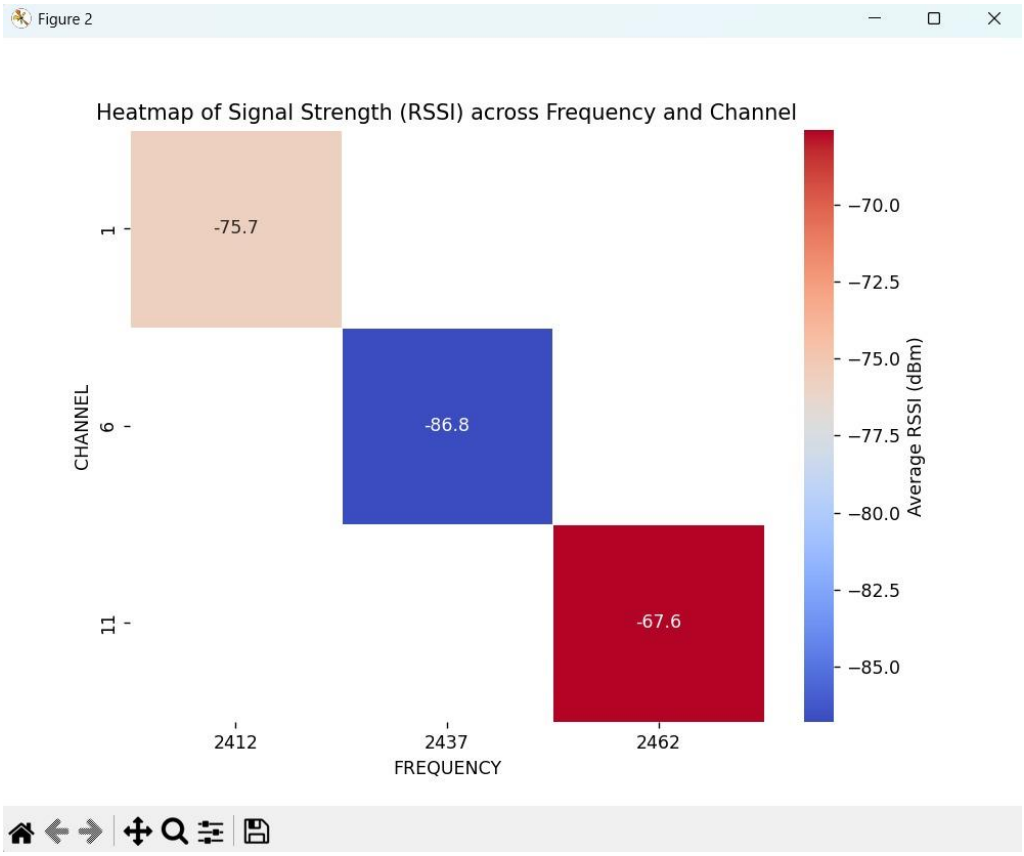
  For the first part of the project, our task was to implement density metrics to demonstrate the traffic throughout the network for multiple channels. The following metrics are provided:

i. Wi-Fi Signal strength by Frequency and BSSID



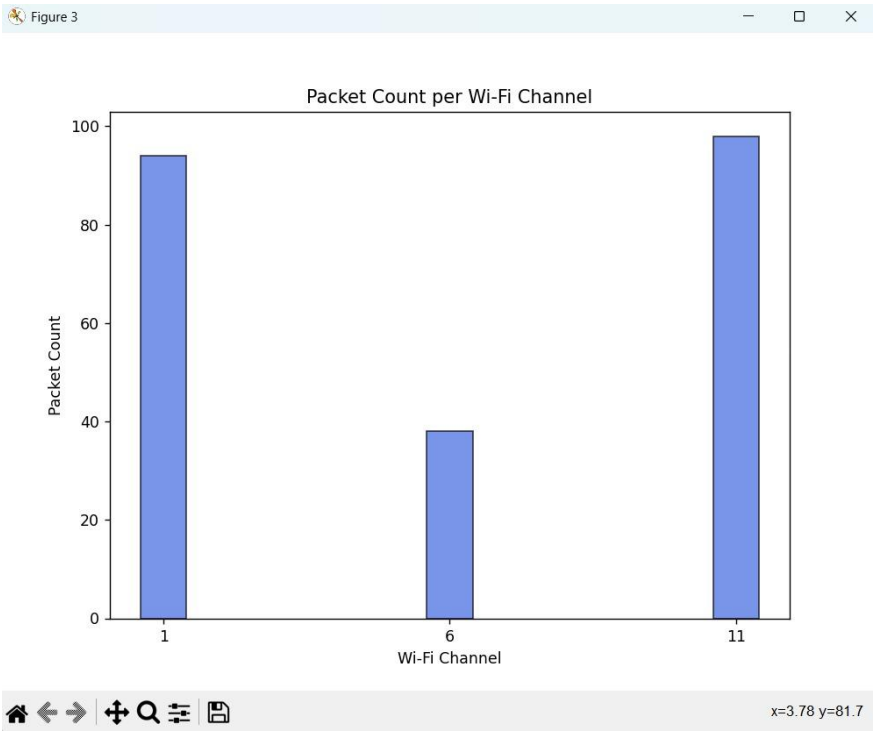
This metric measures how strong the signal is for different BSSIDs (Basic Service Set Identifiers) across various frequencies at the 2.4 ghz band. It is useful for the identification of access points (APs) that provide stronger or weaker signals at specific frequencies, which is crucial for troubleshooting coverage issues. Moreover the graph shows explicitly the overlap and the congestion of the network. Although it is not shown here a graph in the 5 ghz band would prove once again that the 2.4 ghz band is indeed more congested and has bigger overlap between channel than the 5 ghz band.

ii. Signal strength across different frequencies and channels



This metric maps signal strength variations across channels and frequencies using a heatmap. It is an important metric because it is able to identify congestion and interference on specific channels and optimize channel selection to improve performance

iii. Packet count per Wi-Fi channel



Packet count per Wi-Fi channel tracks the number of packets transmitted over each channel. This metric is important because higher packet counts indicate congestion, which can slow down network performance. By monitoring packet density, overused channels can be identified and the network's traffic can be distributed more efficiently between channels.

iv. RSSI over time

RSSI over time visualizes how the signal strength changes over a period. RSSI fluctuations are indicators of channel interference and high presence of noise

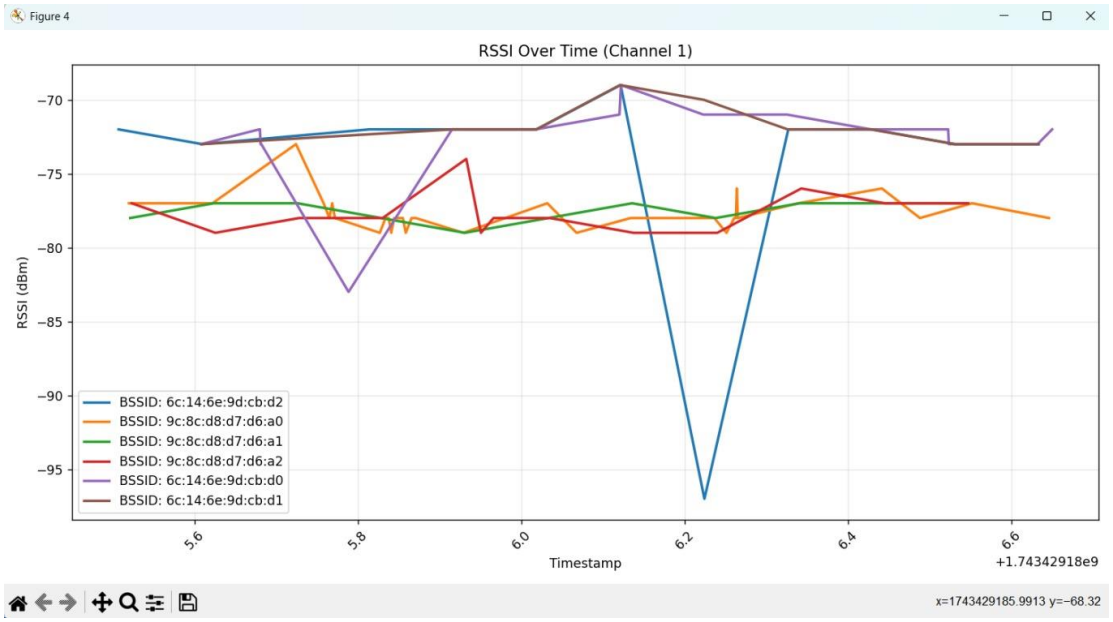


Figure 4
RSSI Over Time (Channel 1)
RSSI (dBm)
−70
−75
−80
−85
−90
−95
BSSID: 6c:14:6e:9d:cb:d2
BSSID: 9c:8c:d8:d7:d6:a0
BSSID: 9c:8c:d8:d7:d6:a1
BSSID: 9c:8c:d8:d7:d6:a2
BSSID: 6c:14:6e:9d:cb:d0
BSSID: 6c:14:6e:9d:cb:d1
5.6
5.8
6.0
6.2
6.4
6.6
Timestamp
+1.74342918e9
x=1743429185.9913 y=−68.32

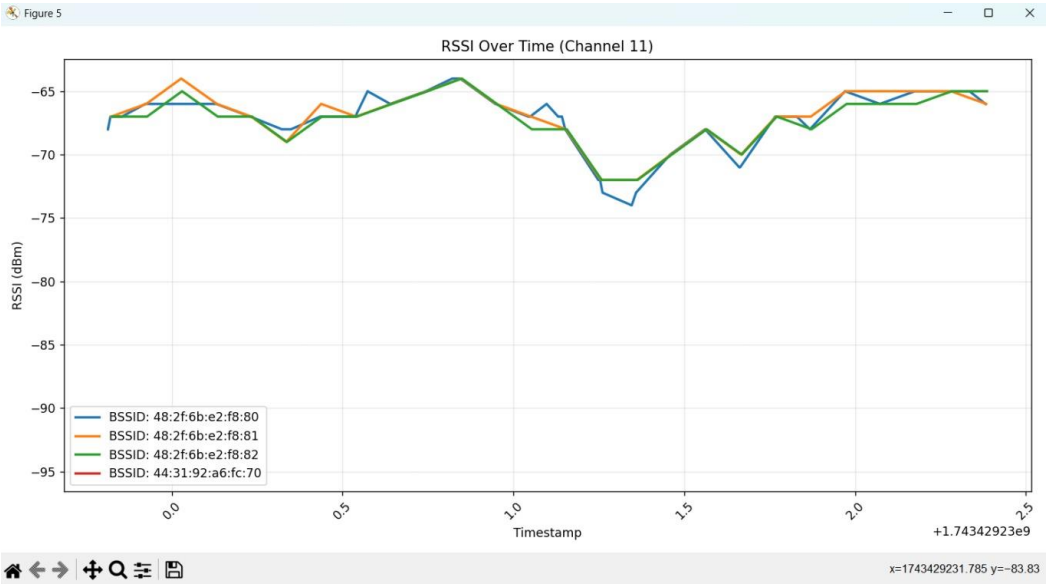


Figure 5
RSSI Over Time (Channel 11)
RSSI (dBm)
−65
−70
−75
−80
−85
−90
−95
BSSID: 48:2f:6b:e2:f8:80
BSSID: 48:2f:6b:e2:f8:81
BSSID: 48:2f:6b:e2:f8:82
BSSID: 44:31:92:a6:fc:70
0.0
0.5
1.0
1.5
2.0
2.5
Timestamp
+1.74342923e9
x=1743429231.785 y=−83.83

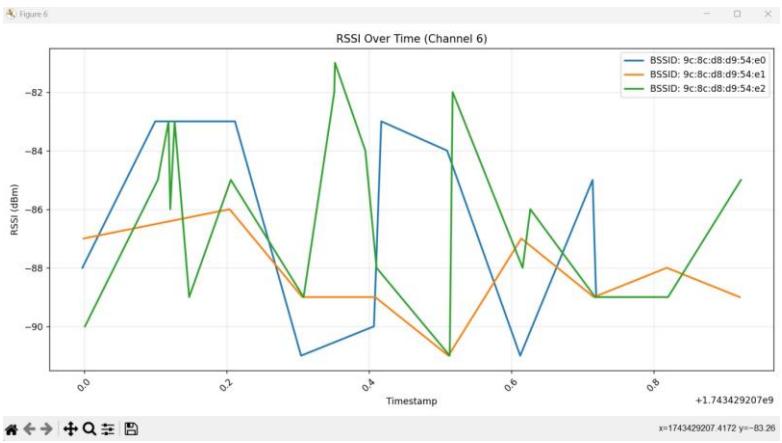
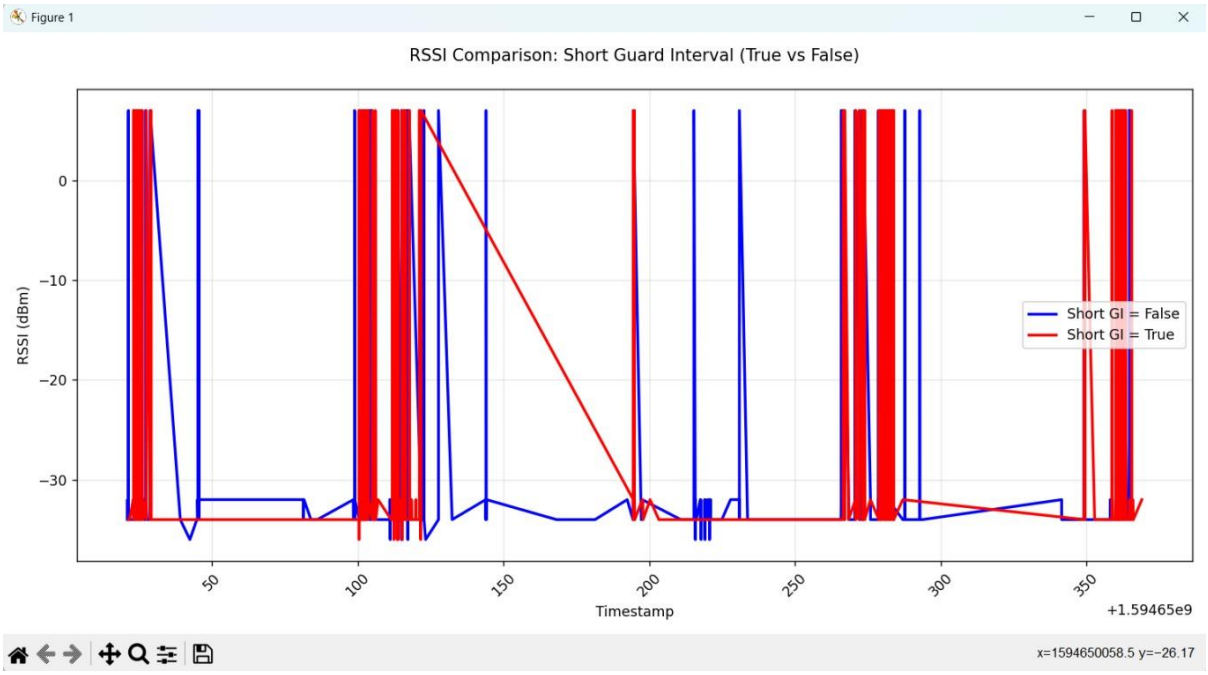


Figure 6
RSSI Over Time (Channel 6)
RSSI (dBm)
−82
−84
−86
−88
−90
BSSID: 9c:8c:d8:d9:54:e0
BSSID: 9c:8c:d8:d9:54:e1
BSSID: 9c:8c:d8:d9:54:e2
0.0
0.2
0.4
0.6
0.8
Timestamp
+1.743429207e9
x=1743429207.4172 y=−83.26

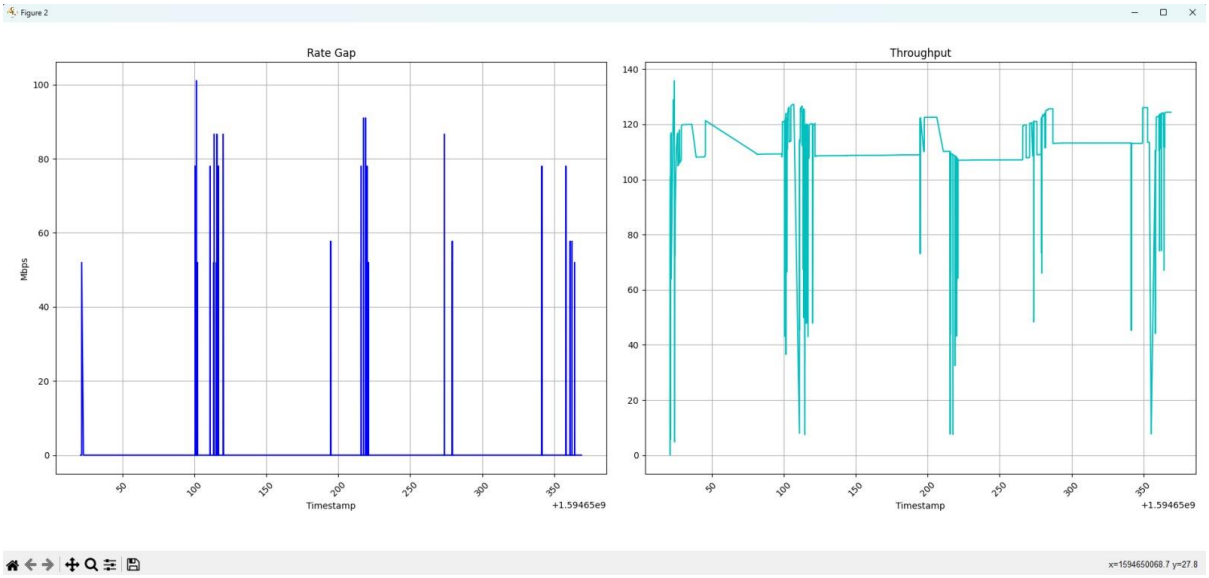
For the second part, we implemented the following metrics for the network's throughput performance:

i. RSSI comparison: short guard interval



RSSI comparison with short guard interval (SGI) shows the signal strength differences when SGI is set to True (enabled) or False (disabled). SGI is a setting that reduces the time between data transmissions, allowing for faster speeds but requiring a stable connection. If SGI results in lower RSSI values or fluctuating performance, it may mean that enabling it leads to more packet loss, reducing overall efficiency.

ii. Rate gap and Throughput



Rate gap and throughput are two critical indicators of Wi-Fi performance. The rate gap metric showcases inconsistencies in data transmission speeds, which may be caused by packet loss, interference or network congestion. Large gaps mean that the connectivity is relatively poor, leading to lower efficiency.

Throughput is used for the measurement of the actual data transfer rate. It is calculated by using the following formula:

$$T = DR \cdot (1 - FLR),$$

where DR is the data rate in Mbps and FLR is the frame loss rate. In order to produce the above throughput diagram over time we calculated the throughput for every timestamp between the specified transmitter and destination address given in the instructions.

If throughput has some spikes over time, it can indicate network congestion, retransmissions, or weak signal conditions, all of which impact the network's performance. By analysing both metrics, it is possible to identify and address performance bottlenecks in the network.

Another important note that can be extracted from both plots is that the rate gap plot explains in great length the throughput bottlenecks as they do appear at the same time intervals where rate gap spikes.

as the rate gap increases, the throughput decreases. This happens because of the large difference between the highest and lowest data rates being achieved.

4. Related work

[1] Pefkianakis et al. “Characterizing Home Wireless Performance: The Gateway View”, IEEE INFOCOM 2015

5. References

[2] https://www.engeniustech.com/wi-fi-beacon-frames-simplified/

[3] https://kiminewt.github.io/pyshark/

[4] https://thepacketgeek.com/pyshark/intro-to-pyshark/