# Εργαστήριο Δικτύων ΙΙ – Άνοιξη 2025

## 1ο Project – Wi-Fi Doctor

### 1. Project Overview

In this project you will have to design and implement a Wi-Fi network performance monitoring / troubleshooting system, aka "Wi-Fi Doctor". A Doctor can provide both the "diagnosis" of a health problem and the "treatment". The Wi-Fi Doctor project will only focus on the "diagnosis" (performance analysis) aspects of Wi-Fi issues (performance bottlenecks), at the *device side*. The "treatment" of Wi-Fi bottlenecks is out of the scope of this project.

An overview of a Wi-Fi Doctor system is shown in the figure below.

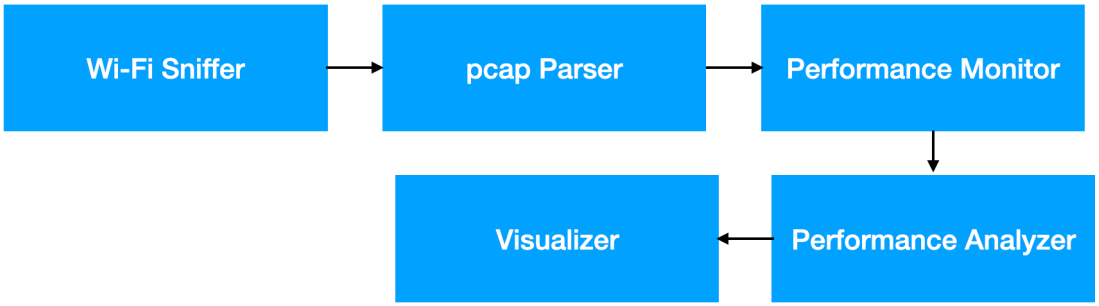


Figure 1: Wi-Fi Doctor

**Wi-Fi Sniffer:** It will monitor the packets transmitted over Wi-Fi. You will use *Wireshark* to monitor the Wi-Fi network as you learned in lab 1. The output of the Wi-Fi Sniffer will be a .pcap file.

**pcap Parser**: It will take as input a .pcap file and it will extract all the related information required for the performance monitor. There is no restriction on the programming language you can use to implement the pcap Parser or the other modules of the system. One option is to use python pyshark [1].

**Performance Monitor**: It will leverage the Wi-Fi parsed parameters to monitor the performance of the Wi-Fi Network. These parameters can include:

- BSSID: e.g., TUC
- Transmitter MAC address: e.g. 00:14:a5:cb:6e:1a
- Receiver MAC address: e.g. 00:14:a5:cd:74:7b
- Type/subtype: e.g., QoS Data
- PHY Type: e.g. 802.11n
- MCS Index: e.g., 15 (if available)
- Bandwidth: e.g., 40 MHz
- Spatial Streams: e.g. 2
- Short GI: True/False
- Data rate: e.g., 300 Mbps
- Channel: e.g., 3
- Frequency: e.g., 2422MHz
- Signal strength: e.g., -57 dBm
- Signal/noise ratio: e.g., 39 dB
- TSF Timestamp: 3745174720

The Performance Monitor will analyze the above parameters and will calculate additional performance indicators such as throughput. The Performance Monitor needs to identify if the performance is good or bad to accommodate certain applications. We will elaborate on its function in the following sections.

**Performance Analyzer**: The Performance Monitor seeks to answer the question; *how good is my Wi-Fi network performance*? The Performance Analyzer seeks to answer the question; *why is my performance good or bad*? The Performance Analyzer needs to analyze the performance in 3 dimensions:

(a) Device configuration: e.g., PHY Type, Bandwidth, Spatial Streams
(b) Link performance: e.g., Signal Strength, Signal/Noise Ratio
(c) Interference / Network Load

**Visualizer**: Presents the Performance Monitor's and Analyzer's outputs in a comprehensive way.

The project can be split in 2 parts, as we discuss in the following sections.

### 1.1 Wi-Fi Network Density

Wi-Fi network density is a "proxy" of network load and interference. Leverage your Wi-Fi Doctor to measure the network density in:

- (a) Enterprise/campus network (TUC)
- (b) Home network

Analyze the performance for at least 2 wireless channels (ideally one channel at 2.4 GHz and another at 5GHz). So, in total you can have at least 4 pcaps (2 networks times 2 channels per network).

For Wi-Fi network density analysis you need to collect .pcap using Wireshark and parse them, through your pcap Parser. The Performance Monitor will process only the following fields from the *802.11 beacon frames*:

- BSSID: e.g., TUC
- Transmitter MAC address: e.g. 00:14:a5:cb:6e:1a
- PHY Type: e.g. 802.11n
- Channel: e.g., 3
- Frequency: e.g., 2422MHz
- Signal strength: e.g., -57 dBm
- Signal/noise ratio: e.g., 39 dB

Then it should provide density metrics which include ALL the above parameters. For example, a Wi-Fi environment can be considered denser when it has more SSIDs with higher signal strength.

**Note**: The Performance Analyzer is not relevant for this part of the project.

The visualizer you should provide the output of your analysis. This could be plots, tables etc. Be creative!

After designing the density metrics you need to compare the density of the Wi-Fi networks and channels you monitored.

### 1.2 Wi-Fi Network Performance

In the 2nd part of the project you will estimate a theoretical downlink (from the Wi-Fi Access Point (AP) to the device) throughput. You will also understand, how to root cause throughput bottlenecks. Different from Part 1.1 where the device is operating in "Monitored" mode, sniffing the neighboring networks, here the device is operating in "Managed" mode. In "Managed" mode the device monitors the *802.11 data frames* between the AP and the device [2] [6].

Since setting your Wi-Fi adapter on "Managed" mode can be challenging, for this part of the project we will provide you the .pcap. The pcap Parser will fetch the metrics described in section 1 and feed them to the Performance Monitor.

Performance Monitor: It will calculate a theoretical downlink throughput as follows:

- Throughput = Data_Rate * (1-Frame_Loss_Rate)

The Data Rate will be in Mbps and it is provided in the 802.11 radio's information in Wireshark. The Frame Loss Rate can be calculated, by monitoring the "retry" flag, provided by the Wireshark. Follow the instructions in [5] to calculate the Frame Loss Rate.

Performance Analyzer: It will provide insights about the throughput performance. Why is throughput good or bad? It will leverage the following metrics to identify performance bottlenecks:

- PHY Type
- Bandwidth
- Short GI
- Data Rate
- MCS Index
- Signal strength (dBm)
- Rate Gap as defined in Equation (1) of [3]. Use the tables in [4] to calculate to the expected PHY Rate $R_E$. Note PHY Rate is the same as Data Rate as reported in [4]

The Performance Analyzer should be able to analyze why Data Rate is low or high by analyzing:

- Wi-Fi configuration (PHY Type, Bandwidth, Short GI) based on [4]
- Wireless channel (MCS Index, Signal Strength) [4] [6]
- Wireless channel / interference [3]

Traces: For Part1.2 you will be provided *HowIWiFi_PCAP.pcap* for your analysis. You will analyze the downlink data traffic from the AP (2C:F8:9B:DD:06:A0) to the device (00:20:A6:FC:B0:36).

Alternatively, you can collect your own traces. For example, connect your Wi-Fi device A to an AP and create downlink traffic. Use another device B in Monitor mode, operating on the same channel with device A, to sniff the 802.11 packets received by device A. Try a scenario where A is close to the AP and far from the AP. Try congested and non-congested settings.

Visualizer:

It first provides a throughput estimation across the trace. It provides statistics (min/mean/median/75P/95P/max) and timeseries analysis of the metrics discussed in the bullets of 1.2. Correlate the metrics (e.g., throughput vs Signal Strength).

## 2. Deliverables

For this project you will need to deliver in a zipped folder with:

- Your code along with a README file to explain what each file of the code is doing
- A report describing your Wi-Fi Doctor system

The report will be a .doc or .pdf, in single column format, up to 10 pages. It can be either in Greek or in English. Its structure will be similar to an academic research paper, with the following sections.

- **Title**

- **Introduction**: Includes the goal of the project, a short description of your system and a short description of your findings and results.

- **Design**: It should be a more elaborate version of Figure 1, with the necessary description of the algorithms used to implement your Wi-Fi Doctor.

- **Evaluation**: Presents the results from parts 1.1 and 1.2. Use the plots and tables, produced by the Visualizer, to illustrate your findings. You need to explain your findings (elaborate not only on the ***what*** you observe but also ***why*** you observe it as well). You can use additional references for your analysis.

- **Related work**: Cite and briefly describe the research papers use used to design your system.

- **References**: List of references (similar format to section 3).

Longer reports are not necessarily better. Be precise!

## 3. Grading

Part 1.1: 50%

Part 1.2: 50%

## 4. References

**[1]** Python pshark: https://pypi.org/project/pyshark/

**[2]** https://travelingpacket.com/2019/12/19/802-11-finding-the-mcs-rate-in-wireshark/

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

**[4]** https://www.watchguard.com/help/docs/help-center/en-US/Content/en-US/WG-Cloud/Devices/access_point/deployment_guide/Checklist_Charts/mcs_80211n_ac.pdf

**[5]** https://semfionetworks.com/blog/calculate-802-11-retry-rate-with-wireshark/

**[6]** https://documentation.meraki.com/MR/Wireless_Troubleshooting/Analyzing_Wireless_Packet_Captures