

Mariem CHaabene

Embeded Systems and IoT
2024–2025

Wi-Fi Optimization via Machine Learning

Wi-Fi Dual-AP Data Collection for Machine Learning-Based Optimization

from 09/06/2025 to 29/08/2025

Universitat Pompeu Fabra
Carrer de Roc Boronat, 138, Sant Martí, 08018 Barcelona, Espagne

Under the supervision of:

- **Company supervisor:** Boris Bellalta, boris.bellalta@upf.edu
- **Phelma Tutor:** Nicolas Castagne, nicolas.castagne@grenoble-inp.fr

Confidentiality: No

Phelma

Bât. Grenoble INP – Minatec
3, Parvis Louis Néel – CS 50257
38016 Grenoble Cedex 01
Tél +33 (0)4 56 52 90 00 Fax +33 (0)4 56 52 91 03
<http://phelma.grenoble-inp.fr>

Abstract

This work addresses the design and implementation of an experimental testbed for Wi-Fi performance evaluation and dataset collection, with the objective of enabling Machine Learning-based optimization of the wifi network in order to improve throughput, latency, and reliability.

The proposed testbed integrates two OpenWrt-based Access Points controlled by an orchestration machine and two client devices executing automated traffic generation and metric collection scripts. A fully automated data collection pipeline was developed, producing a dataset of more than 1,000 labeled experiments under diverse network configurations.

The report presents the system architecture, the implementation methodology, and the obtained experimental results.

Ce travail présente la conception et la mise en place d'une plateforme expérimentale destinée à la constitution d'un jeu de données dont le but de l'utiliser dans des algorithmes d'apprentissage automatique. L'objectif est de permettre l'optimisation de réseau wifi afin d'améliorer le débit, la latence et la fiabilité des communications.

La plateforme proposée repose sur deux points d'accès OpenWrt pilotés par une machine d'orchestration, ainsi que sur deux terminaux clients exécutant des scripts automatisés de génération de trafic iperf et de collecte de métriques. Une chaîne de collecte de données entièrement automatisée a été développée, produisant plus de 1 000 expériences étiquetées dans des configurations variées.

Le rapport décrit l'architecture du système, la méthodologie adoptée et les principaux résultats expérimentaux obtenus.

Contents

Glossary	3
List of Figures	5
1 Introduction	6
1.1 Context	6
1.2 Problem Statement	6
1.3 Objectives	6
1.4 Plan of the Report	6
2 Background and State of the Art	7
2.1 Wi-Fi Performance Challenges	7
2.2 Machine Learning for Wi-Fi Optimization	7
3 Exploration of Changeable OpenWrt Wi-Fi Parameters	8
3.1 Overview of the Experimental Open Wrt Environment	8
3.2 Changeable OpenWrt Wi-Fi Parameters	8
3.2.1 Transmission Power	8
3.2.2 Channel (Frequency)	8
3.2.3 Channel Bandwidth	9
3.2.4 Contention Window (WMM)	9
3.2.5 MCS and NSS Control	9
4 Single-AP Experiment Pipeline	10
4.1 System Architecture	10
4.2 Pipeline Overview	10
4.3 Software Components and Roles	11
4.3.1 iperf_single.sh	11
4.3.2 run_iperf_scenarios.sh	11
4.3.3 collect_wifi_metrics_json3.sh	11
4.3.4 collect_loop3.sh	11
4.3.5 wifi_test_orchestrator.sh	12
4.4 Workflow Diagram	12
4.5 Server-Side Metric Synchronization	12
4.6 Dataset Organization in JupyterLab	13
4.7 Operational Notes	13
4.8 Results and Discussion	13
4.8.1 Impact of RSSI and Bandwidth on TCP Throughput	13
4.8.2 Transmit-Power Influence on Received Signal Strength	14
4.8.3 Latency as a Function of RSSI for TCP vs UDP	15
4.8.4 Protocol-Level Throughput Comparison	16
4.8.5 Interpretation of TCP Throughput vs MCS Index	17
5 Dual-AP Experiments: Direct Duplicatin of Exp1	18
5.1 Objectives	18

5.2	System Architecture	19
5.3	Adaptations comparing to Exp1	19
5.4	Execution Flow (per AP)	19
5.5	Output files	20
5.6	Observed Issue: Inter-AP Desynchronization	20
5.7	Conclusion	20
6	Dual-AP Experiments: Controller-Based Dual-AP Solution	21
6.1	Problem Statement	21
6.2	Objectives	21
6.3	Design Principles	21
6.4	System Architecture	22
6.5	Measurement Pipeline	22
6.6	Comparison with Naïve Duplication	23
6.7	Software Components and Roles	23
6.8	Workflow Diagram	24
6.9	Results and Discussion	24
6.9.1	iPerf throughput & TX_DELTA	24
6.9.2	Results and Discussion on the Final Dataset	25
6.10	Reproducibility (Procedure)	26
7	Conclusion and Perspectives	26
	Annexes	28

Glossary

AP	Access Point. Device that provides Wi-Fi connectivity to clients.
OpenWrt	Linux-based open-source operating system for embedded routers, enabling fine-grained configuration of wireless parameters.
MCS	Modulation and Coding Scheme. Defines the modulation type and error correction rate in IEEE 802.11, impacting data rate and robustness.
NSS	Number of Spatial Streams. Determines the degree of MIMO (Multiple-Input Multiple-Output) parallelism in Wi-Fi transmissions.
CW_{min}/CW_{max}	Contention Window bounds used in the CSMA/CA medium access protocol. Define the random backoff range before channel access.
iPerf3	Widely used traffic generator and performance measurement tool supporting TCP and UDP flows, with JSON output format.
JSONL	JSON Lines format (one JSON object per line), convenient for large log files and streaming datasets.
tmux	Terminal multiplexer for running multiple shell sessions persistently on the same terminal.
UCI	Unified Configuration Interface. The command-line configuration system used in OpenWrt.
iw	Command-line tool to configure and query Linux wireless interfaces.
hostapd	Daemon for access point and authentication server management in Linux systems.
TCP	Transmission Control Protocol. A reliable, connection-oriented transport protocol ensuring in-order delivery.
UDP	User Datagram Protocol. A lightweight, connectionless transport protocol with no retransmissions.
RSSI	Received Signal Strength Indicator. Measurement of the signal power observed at the receiver.
SNR	Signal-to-Noise Ratio. Ratio between signal power and noise floor, determining link quality.
RTS/CTS	Request to Send / Clear to Send. Mechanism to reduce collisions in Wi-Fi by reserving the channel before transmission.
PHY	Physical layer of the OSI model, dealing with modulation, coding, and RF transmission.

MAC Medium Access Control layer, responsible for channel access, retransmissions, and frame addressing.

TX_DELTA

AP-side throughput metric, computed from differences in transmitted byte counters over time.

List of Figures

1	Single-AP experiment pipeline.	10
2	Single AP workflow	12
3	TCP throughput vs RSSI, colour-coded by bandwidth. Error bands = 95 % CI (LOESS).	13
4	Measured RSSI (dBm) as a function of transmission power (TX).	15
5	Average latency (ms) as a function of RSSI (dBm), split by transport protocol.	15
6	Average latency (ms) as a function of RSSI (dBm), split by transport protocol.	16
7	TCP throughput vs MCS index (80 MHz, NSS=2, short GI)	17
8	Impact of TCP Window Size on Throughput	17
9	Dual-AP experiment pipeline (direct duplication).	19
10	Controller based dual AP pipeline	22
11	Controller-Based Dual-AP	24
12	Comparison of iPerf throughput and TX_DELTA estimated throughput.	25
13	C1: Travail et communiquer en environnement international et interculturel	39
14	C2: Concevoir ou réaliser des solutions d'ingénierie, permettant de répondre à un cahier des charges	40
15	C3: Mettre en œuvre une démarche d'innovation et de recherche, Concevoir des solution d'ingenierie	41
16	C4: Travail et coopérer dans une équipe	42
17	C5: Tenir compte des transitions technologiques, environnementales et sociétales (RSE)	44

1 Introduction

1.1 Context

The increasing demand for high-throughput, low-latency, and reliable connectivity has made Wi-Fi optimization a critical challenge, particularly in multi-Access Point (AP) environments. With the proliferation of connected devices and dense deployments, ensuring stable and efficient wireless communication is more essential than ever.

1.2 Problem Statement

Traditional static configurations (e.g., channel assignment, bandwidth, transmit power, MCS/NSS selection, contention parameters) often fail to deliver optimal performance in dynamic and dense Wi-Fi environments. This issue is amplified in multi-AP scenarios, where interference, channel overlap, and unstable client associations can severely degrade network quality [1].

To address these challenges, adaptive network management approaches have been introduced, with Machine Learning (ML) playing an increasingly important role. By learning from real measurements, ML algorithms can predict and apply optimal configurations in real time, adapting to changing network conditions [2]. Recent studies confirm this trend, showing that combining Wi-Fi monitoring with ML enables accurate performance prediction and proactive optimization [3]. However, the effectiveness of these approaches depends on the availability of accurate, large-scale, and representative datasets that capture the relationship between Wi-Fi parameters and network performance metrics [4].

1.3 Objectives

The objective of this internship was to design and implement a robust experimental framework capable of automatically generating large-scale datasets for Wi-Fi performance. More specifically, the work aimed to:

- Build a reproducible single- and dual-AP testbed allowing remote configuration of key Wi-Fi parameters (channel, bandwidth, transmit power, MCS/NSS, contention window) on OpenWrt APs via automated SSH scripts.
- Implement an automated pipeline for traffic generation and synchronized metric collection, including iPerf3 flows (TCP/UDP) and Wi-Fi statistics (RSSI, retries, failures, noise floor, latency).
- Produce clean, analysis-ready datasets in structured JSONL format, capturing both input parameters (AP configurations, traffic settings) and output metrics (throughput, latency, packet loss, jitter) suitable for training ML models.

1.4 Plan of the Report

The remainder of this report is organized as follows:

Section 2 reviews the relevant background and state of the art, covering Wi-Fi performance challenges and recent advances in ML-based wireless optimization.

Sections 3–4 detail the single-AP methodology and results, including the exploration of configurable OpenWrt parameters (Section 3) and the orchestration pipeline, metric collection, and synchronization (Section 4).

Sections 5–6 extend the study to dual-AP scenarios: Section 5 highlights the limitations of naïve duplication and Section 6 introduces the proposed controller-based orchestration framework, its reliability mechanisms, and the final dataset.

Section 7 concludes the work and outlines future perspectives.

Annexes provide the scripts and procedures referenced throughout.

2 Background and State of the Art

2.1 Wi-Fi Performance Challenges

Wi-Fi networks operate in shared, interference-prone environments. Factors such as channel congestion, adjacent-channel interference, signal attenuation, and multipath fading can significantly affect throughput and latency. In dense deployments, overlapping APs can cause severe interference and unstable associations, leading to degraded performance [5, 6]. Moreover, performance is influenced by a variety of configurable parameters, including:

- **Channel number/frequency** (interference exposure and propagation);
- **Channel width** (e.g., 20/40/80 MHz; spectrum efficiency vs. robustness);
- **Transmit power** (coverage vs. interference footprint);
- **MCS index** (data-rate vs. error resilience);
- **NSS** (MIMO spatial streams);
- **RTS/CTS** (collision control in congested environments).

Static configurations of these parameters often fail to adapt to real-time variability of wireless channels and client distributions, especially in dynamic environments such as offices or public spaces.

2.2 Machine Learning for Wi-Fi Optimization

Traditional approaches to Wi-Fi optimization rely on heuristics or rule-based systems, which lack the adaptability to react optimally under diverse and changing conditions. ML offers a promising alternative by enabling data-driven decisions; such approaches have been increasingly applied to Wi-Fi optimization [7]. Recent work has demonstrated real-time Wi-Fi monitoring and throughput prediction using ML-based models [3].

Supervised learning can model the mapping between network configurations and resulting performance metrics, enabling prediction of optimal settings. Reinforcement learning can learn policies for dynamic reconfiguration based on feedback from the environment. To apply these methods effectively, high-quality datasets are essential and must provide (i) **input features** (Wi-Fi configuration parameters, link-layer statistics, and channel conditions),
(ii) **output labels** (throughput, latency, jitter, packet loss),
(iii) **variety** (diverse network states)
and (iv) **sufficient volume** to train robust models.

In this context, the present work focuses on building a synchronized, dual-AP dataset with rigorous acceptance criteria, designed to support ML-based Wi-Fi performance prediction and optimization.

3 Exploration of Changeable OpenWrt Wi-Fi Parameters

3.1 Overview of the Experimental Open Wrt Environment

Experiments were conducted on Access Points (APs) running OpenWrt, a Linux-based firmware that enables both web-UI and command-line configuration (via SSH). This environment provides low-level access to MAC/PHY settings and allows *fine-grained* control of the wireless interface[8, 9, 10, 11].

Each AP was accessed over SSH; configurations were applied using `uci`[9] and `iw`[11](and persisted through `wifi reload`). This enabled shell scripts to automate parameter changes and to run controlled experiments systematically.

3.2 Changeable OpenWrt Wi-Fi Parameters

In this subsection, we detail the OpenWrt knobs that can be reconfigured (channel/frequency, channel width, transmit power, EDCA/WMM contention window, and MCS/NSS) and quantify how each affects key performance indicators (throughput, latency, and stability).

3.2.1 Transmission Power

Principle. Transmission power controls coverage and interference footprint; higher power improves SNR at distance but increases co-channel interference.

Exploration. Two PCs were used: PC1 to change AP settings; PC2 to generate continuous ICMP pings while varying transmission power between 23 and 15 dBm.

Observations.

- Near the AP, reducing power from 23 to 15 dBm had little effect on latency (typically 13–15 ms).
- At longer distance (outside the room), latency spikes up to 124 ms were observed.

Conclusion. Transmit power materially affects latency as path loss increases. Lower power can limit interference for nearby clients but must be raised for distant/obstructed clients. (see Annex)

3.2.2 Channel (Frequency)

Principle. The operating channel determines interference exposure and collision probability.

Exploration. The AP initially operated on channel 40 (5200 MHz). We tested:

- *iw with interface down/up*: applied the change but caused client disconnections and up to 41% packet loss during the switch.
- *uci+wifi reload*: applied the change persistently, with a short outage and about 15% packet loss.

Conclusion. Channel switching on an active AP introduces a brief service interruption. The `uci+wifi reload` method is more reliable and less disruptive. (see Annex)

3.2.3 Channel Bandwidth

Principle. Channel width (20/40/80 MHz) trades robustness for peak data rate: larger widths increase capacity but are more sensitive to interference.

Exploration. Automated switching between HE20 (20 MHz) and HE80 (80 MHz) was implemented; each change was verified from the active config while continuous pings monitored transient latency.

Conclusion. HE20 provides more stable communication in noisy environments; HE80 increases peak throughput but raises susceptibility to interference. (see Annex)

3.2.4 Contention Window (WMM)

Principle. The contention window (CWmin/CWmax) governs backoff aggressiveness and thus latency/fairness. In our tests we modified the *Best Effort (BE)* access category.

Exploration. CWmin/CWmax were changed via `uci` WMM parameters (propagated to `hostapd`); settings were verified, and ping/throughput tests were run.

Conclusion. Smaller CW reduces average latency but increases collision risk/unfairness; larger CW increases delay but mitigates contention. (see Annex)

3.2.5 MCS and NSS Control

Principle. PHY data rate depends on NSS, modulation order, coding rate, symbol rate, and protocol efficiency. Higher MCS/NSS increases throughput but requires higher SNR.

Exploration. Specific MCS/NSS values were forced using `iw` (e.g., `vht-mcs-5 <NSS>:0-9`). Bitrate reports confirmed the settings, and measured throughput matched expectations. Automatic rate adaptation was restored by clearing the forced configuration.

Conclusion. Forcing MCS/NSS enables precise control of the throughput/robustness trade-off, confirming that OpenWrt can override automatic rate selection when supported by the driver. (see Annex)

4 Single-AP Experiment Pipeline

Objective: Deploy an automated pipeline on a *single* Access Point (AP) that (i) applies Wi-Fi configuration changes,
(ii) runs randomized `iperf3` traffic tests,
(iii) collects Wi-Fi and traffic metrics with consistent timing,
and (iv) stores results as structured JSON Lines (`.jsonl`) for analysis.

4.1 System Architecture

- **Access Point (AP):** Provides Wi-Fi service; configurable through OpenWrt (`uci/iw`). Configuration commands are issued remotely from the client laptop (SSH session).
- **Client (control & collection):** Triggers `iperf3` traffic, collects Wi-Fi metrics on the AP via SSH/OpenWrt, and merges them into JSONL snapshots.
- **Server:** Receives `iperf3` traffic and exports server-side logs for cross-validation.

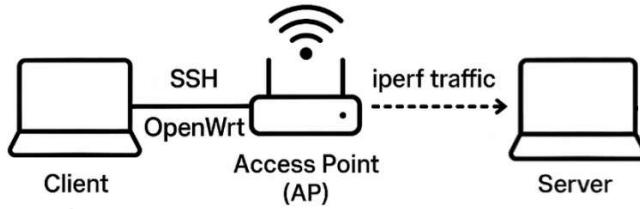


Figure 1: Single-AP experiment pipeline.

4.2 Pipeline Overview

1. **Orchestration.** `wifi_test_orchestrator.sh` iterates predefined configurations (channel, Tx power, bandwidth), applies changes via `uci/iw`, reloads Wi-Fi, waits for stabilization, then triggers traffic tests.
2. **Traffic generation.** `run_iperf_scenarios.sh` samples randomized scenarios (TCP/UDP, streams, window/target rate) and calls `iperf_single.sh`.
3. **Single test execution.** `iperf_single.sh` runs one `iperf3` test (`-J`), extracts protocol/throughput/loss/jitter and writes one JSON line; raw JSON is kept for debugging.
4. **Wi-Fi snapshot.** `collect_wifi_metrics_json3.sh` queries the AP (`iw dev ... info, station dump, survey dump, uci`), measures latency (ICMP), and attaches the current traffic label.
5. **Continuous collection.** `collect_loop3.sh` runs as a daemonized loop with period $\Delta t = 30\text{ s}$; it calls the snapshot script, validates JSON, merges with the most recent traffic metrics, and appends to `wifi_log2.jsonl`.
6. **Final dataset.** The file `wifi_log2.jsonl` consolidates Wi-Fi parameters, per-client stats, traffic metrics and timestamps; it is later imported into Python for analysis.

4.3 Software Components and Roles

4.3.1 iperf_single.sh

Function. Execute a single `iperf3` test with a unique label and export a compact record.
Steps.

- Write the label to `/tmp/current_traffic_label`.
- Run `iperf3 -J` for a fixed duration (e.g., 60 s).
- Extract: protocol, throughput, loss, jitter, requested bandwidth/window.
- Append one JSON line to `traffic_metrics.jsonl`;

4.3.2 run_iperf_scenarios.sh

Function. Build and execute randomized test scenarios. Each test:

- Generates a unique traffic label.
- Calls `iperf_single.sh` with selected options (TCP/UDP, streams, window/target rate).
- Waits sufficiently (e.g., 65 s) to ensure the Wi-Fi collector captures the test interval.

4.3.3 collect_wifi_metrics_json3.sh

Function. Produce a complete Wi-Fi snapshot of:

- *Global configuration:* channel, frequency, bandwidth, Tx power, contention window (CWmin/CWmax), MCS/NSS.
- *Environment:* noise floor (dBm), channel busy time (%).
- *Per-client:* MAC, RSSI (dBm), retries/failures, TX/RX bitrate (Mb/s), TX/RX bytes, MCS.
- *Latency:* ICMP RTT (min/avg/max).

Extraction uses `iw/uci`, outputs parsed to JSON. The current traffic label is read from `/tmp/current_traffic_label`. Output: `wifi_metrics2.json`.

4.3.4 collect_loop3.sh

Function. Run a continuous loop (period $\Delta t = 30$ s), invoke the snapshot script, validate JSON, merge with the latest traffic metrics, and append to `wifi_log2.jsonl`. Runs directly on the AP; designed to be launched in a `tmux` session.

4.3.5 wifi_test_orchestrator.sh

Function. End-to-end automation:

- Stop any running `collect_loop3.sh`; start one collector for the whole campaign.
- Iterate over parameter sets (e.g., channels {36, 44, 48}, Tx power {10, 17, 23} dBm, bandwidth {20, 80} MHz).
- For each config: apply `uci/iw`, reload Wi-Fi, wait for stabilization, run all traffic scenarios.
- Stop collection cleanly at the end and persist logs.

4.4 Workflow Diagram

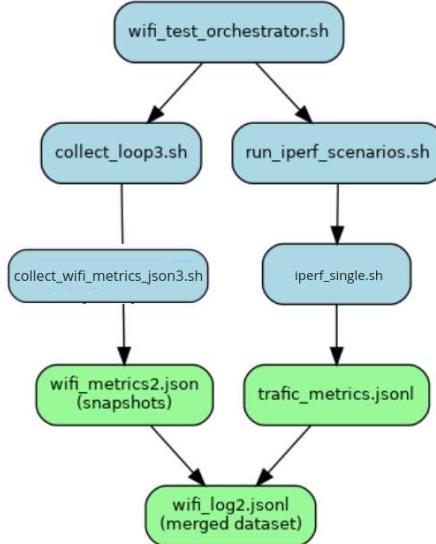


Figure 2: Single AP workflow

4.5 Server-Side Metric Synchronization

In addition to collecting Wi-Fi and client-side `iperf3` metrics on the access point, we also collected `iperf3` server-side metrics (received throughput, packet loss, jitter) directly from the laptop acting as the server. These metrics provide the server's perspective on traffic quality and allow cross-validation of client-side results.

To synchronize both sides, we used the `-extra-data` flag of `iperf3`, which embeds a custom label in the JSON output. After fine-tuning both client and server scripts, we were able to collect synchronized data from both ends. The client produced `wifi_log2.jsonl` (Wi-Fi and client-side iPerf data), and the server produced `trafic_metrics_server.jsonl` (server-side results).

To transfer the server-side file to the client device, the following command was used:

```
scp -O chaabenm@192.168.1.182:/tmp/trafic_metrics_server.jsonl /root/trafic
```

4.6 Dataset Organization in JupyterLab

At the end of the collection process, we loaded both datasets into Python and merged them using `pandas`. The common field used for merging was `traffic_label`, which is shared by both files (thanks to `-extra-data`).

The final dataset (**3607 rows**) includes: (i) Wi-Fi configuration (channel, NSS, bandwidth, Tx power, etc.), (ii) client-side metrics (latency, estimated throughput, RSSI, retries, bitrates), and (iii) server-side metrics (received throughput, packet loss, jitter). All notebooks were executed in `tmux` sessions to avoid data loss.

4.7 Operational Notes

- Long-running scripts (`collect_loop3.sh`, server listener) are launched in `tmux` and host sleep is disabled to prevent interruptions.
- Labels propagated via `-extra-data` ensure unambiguous alignment of Wi-Fi snapshots and traffic tests.

4.8 Results and Discussion

4.8.1 Impact of RSSI and Bandwidth on TCP Throughput

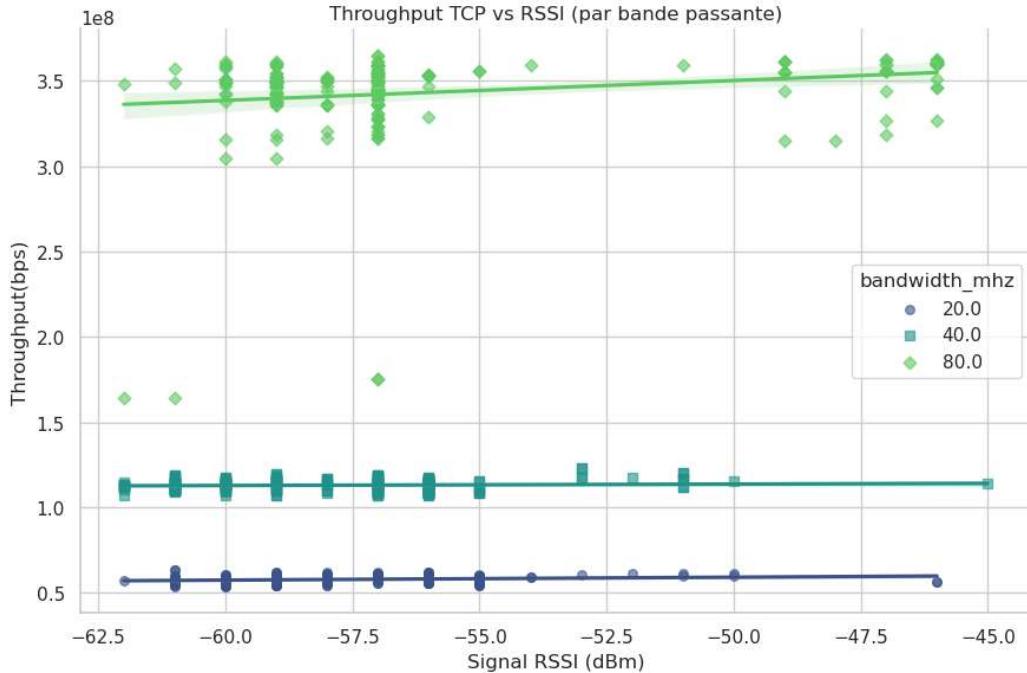


Figure 3: TCP throughput vs RSSI, colour-coded by bandwidth. Error bands = 95 % CI (LOESS).

Figure 3 explores the influence of signal strength (RSSI) and channel bandwidth on TCP throughput. Three well-defined performance “plateaus” emerge from the data:

- **20 MHz** → approximately **60 Mbps**
- **40 MHz** → approximately **110 Mbps**

- **80 MHz** → approximately **340 Mbps**

These values are consistent with theoretical expectations from Wi-Fi physical-layer limits. Within each bandwidth group, the regression line (LOESS-smoothed) appears nearly flat. This suggests that, once a minimum signal threshold is met (around -62 dBm), increases in RSSI do not significantly affect throughput.

Theoretical context. According to IEEE 802.11 standards, the maximum physical-layer data rate is determined by:

$$\text{Data Rate (Mbps)} = \text{NSS} \times \text{BW}_{\text{MHz}} \times \text{Efficiency(MCS)} \times \text{Symbol Rate}$$

In this expression:

- **NSS** is the number of spatial streams.
- **BW** is the channel bandwidth (e.g., 20/40/80 MHz).
- **Efficiency(MCS)** is the number of bits per symbol (depends on MCS index).
- **Symbol Rate** is typically fixed for a given standard (e.g., 802.11ac).

Once RSSI is strong enough it sustains a stable MCS and once that MCS is stable, higher RSSI values no longer translate into higher throughput. And **bandwidth becomes the dominant factor** and throughput becomes plateaued and bounded by bandwidth.

Thus, increasing **BW** has a linear and immediate effect on achievable throughput, which explains the three performance tiers observed in the data. In contrast, **RSSI only determines the maximum usable MCS**. Once that MCS is stable, throughput becomes plateaued and bounded by bandwidth.

4.8.2 Transmit-Power Influence on Received Signal Strength

Figure 4 shows the distribution of received signal strength indicator (RSSI) values across three transmit power settings: 10, 17, and 23dBm. Each boxplot summarizes the variability and central tendency of RSSI measurements recorded at the receiver.

Observations.

- The median RSSI increases slightly with TX power, from approximately -58 dBm (at 10dBm) to -56 dBm (at 23dBm), indicating improved signal strength.
- The interquartile range (IQR) narrows with higher TX power, suggesting more consistent and stable RSSI values.

Interpretation. These results confirm that increasing the transmit power improves the received signal strength. Additionally, the reduced IQR at higher TX power indicates that the received signal becomes more stable, potentially improving modulation robustness.

Conclusion. Transmit power has a measurable but limited effect on RSSI in this setup. While stronger TX power improves signal strength and reduces variability, the magnitude of change remains small. This reinforces the idea that RSSI is already sufficiently high across all tested TX levels to sustain stable links.

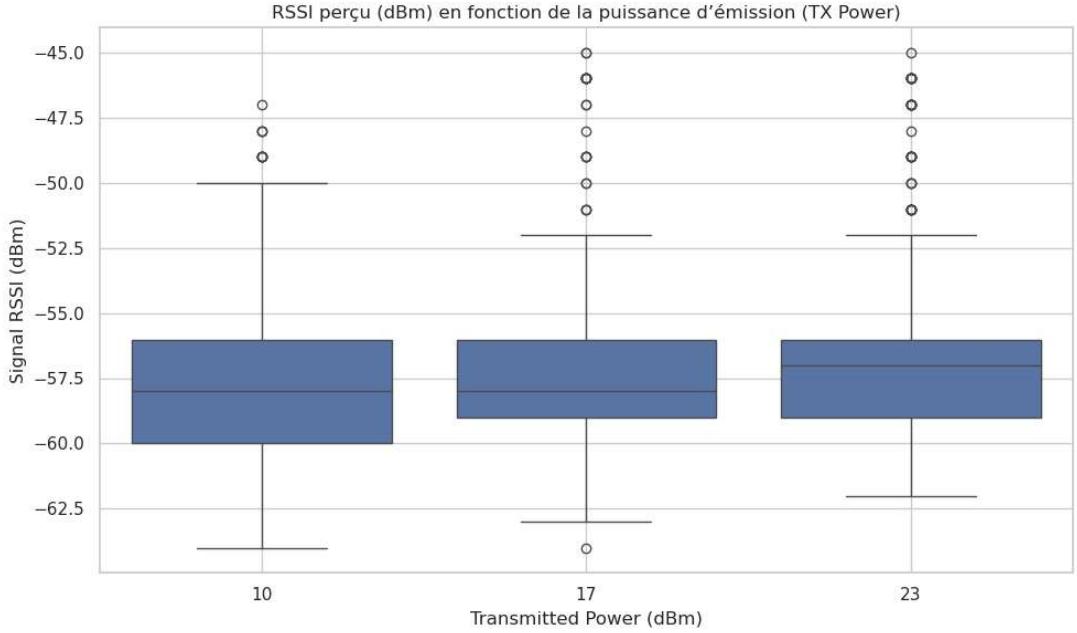


Figure 4: Measured RSSI (dBm) as a function of transmission power (TX).

4.8.3 Latency as a Function of RSSI for TCP vs UDP

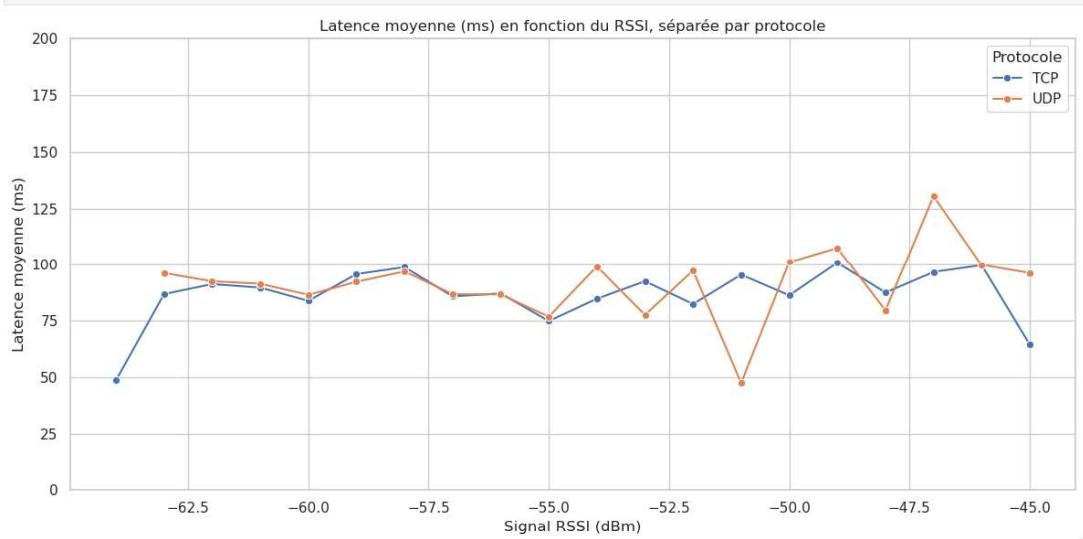


Figure 5: Average latency (ms) as a function of RSSI (dBm), split by transport protocol.

RSSI impact (Figure 5). Latency exhibits a modest decreasing trend as RSSI improves from very low levels (e.g., -63dBm) up to about -55dBm . This aligns with theoretical expectations: better signal strength reduces packet loss and retransmissions, leading to faster delivery.

Beyond -55dBm , however, this improvement plateaus. The latency stabilizes and no longer decreases with further RSSI gains. This suggests that once a minimum signal quality is achieved to support reliable MCS.,. Additionally, the plot reveals differences between protocols:

- **TCP** shows a more stable latency curve across RSSI levels, likely due to its congestion control, flow regulation, and ACK mechanisms.
- **UDP** demonstrates higher variability, especially at higher RSSI levels. However, in some instances, it achieves slightly lower minima, which is advantageous for real-time traffic.

4.8.4 Protocol-Level Throughput Comparison

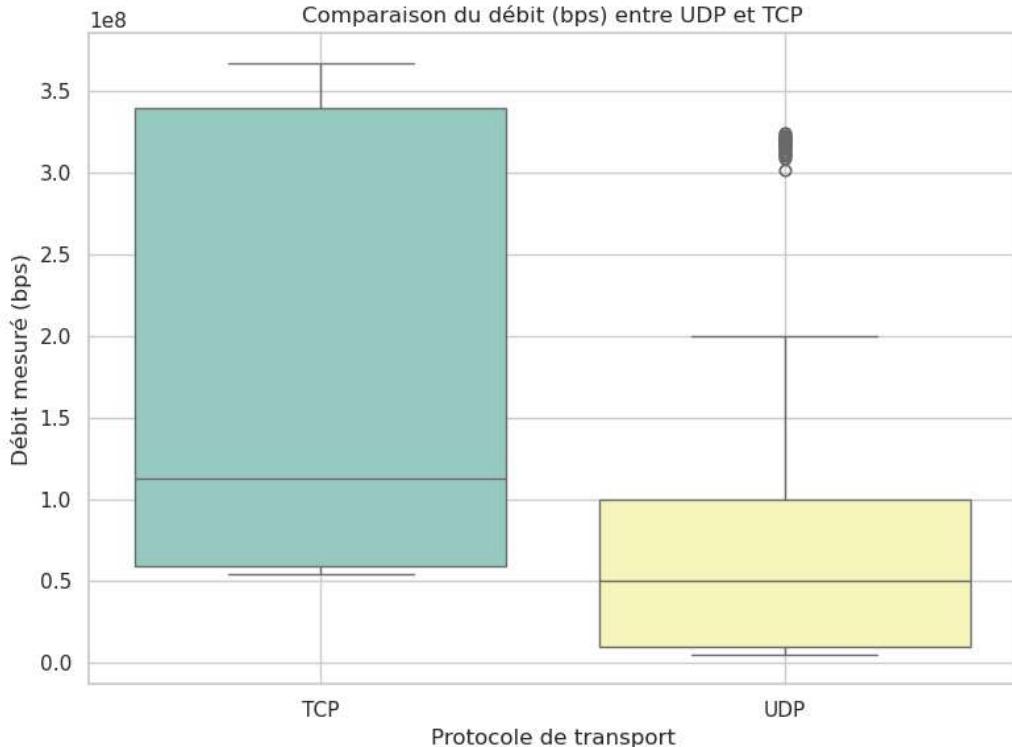


Figure 6: Average latency (ms) as a function of RSSI (dBm), split by transport protocol.

Protocol-Level Throughput Comparison (Figure 6). This boxplot illustrates the distribution of measured throughput (in bps) for both TCP and UDP traffic over the test.

Findings. The TCP protocol exhibits a significantly wider throughput range, with a median around 110 Mbps and upper values reaching nearly 360 Mbps. UDP, on the other hand, shows a narrower distribution centered around 50 Mbps, with most values below 200 Mbps.

Interpretation. The fundamental difference lies in how each protocol manages transmission. TCP is connection-oriented and includes built-in mechanisms such as congestion control, flow control, and acknowledgments. These allow it to dynamically probe and saturate the available bandwidth based on network conditions. Therefore, TCP adapts to utilize the maximum capacity of the physical layer (PHY), particularly in clean wireless environments.

UDP, by contrast, is a connectionless protocol that lacks such adaptive mechanisms. It sends packets at a user-defined constant rate without feedback from the receiver. In our experiments, most UDP tests were configured with a fixed transmission rate (e.g.,

100 Mbps), which artificially capped the maximum achievable throughput regardless of the actual link capacity. This explains why UDP appears less performant in the plot, despite its lower protocol overhead in theory.

Conclusion. TCP’s intelligent rate adaptation enables it to outperform UDP in raw throughput under typical conditions. However, this does not imply that TCP is always preferable: UDP remains valuable for latency-sensitive applications where throughput consistency is less critical than timely delivery.

4.8.5 Interpretation of TCP Throughput vs MCS Index

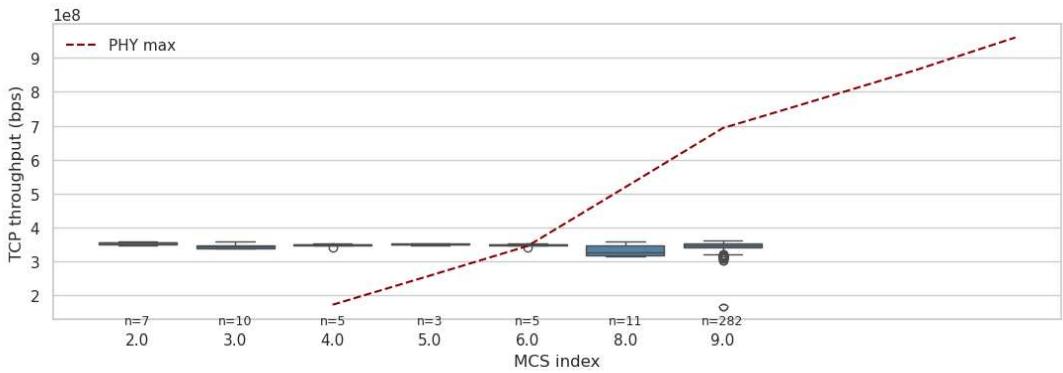


Figure 7: TCP throughput vs MCS index (80 MHz, NSS=2, short GI)

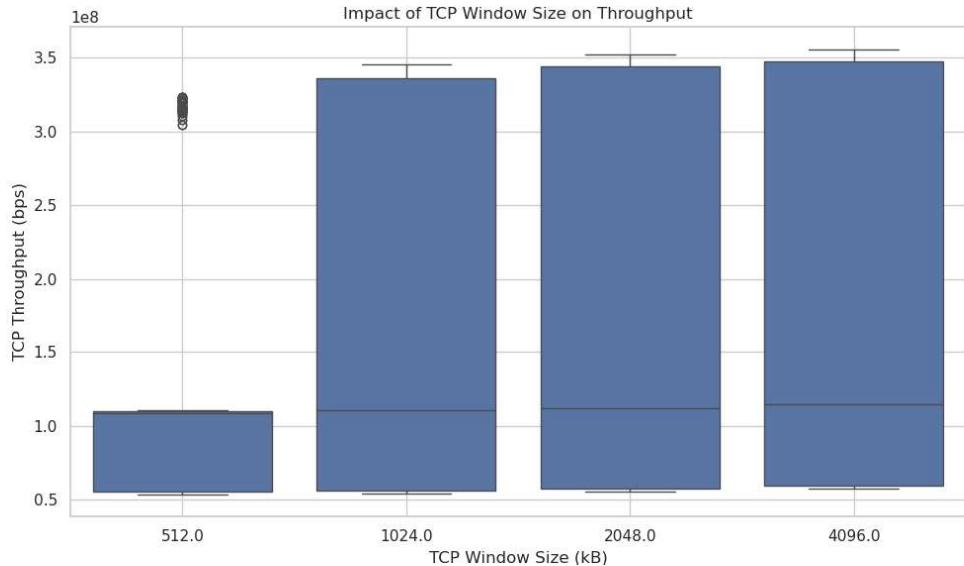


Figure 8: Impact of TCP Window Size on Throughput

Figures 7 and 8 highlight a key performance limitation in Wi-Fi TCP transmissions. Although the physical (PHY) data rate increases significantly with the MCS index—reaching up to 960 Mbps at MCS 9 with short guard interval (400 ns)—the actual measured TCP throughput remains flat around 330–350 Mbps. This plateau suggests that the bottleneck lies not in the wireless channel, but at the transport layer, and more specifically in TCP’s congestion and flow control behavior.

TCP throughput is constrained by bandwidth-delay product:

$$\text{Throughput} = \frac{\text{TCP window size}}{\text{RTT}}.$$

Assuming a round-trip time (RTT) of 6 ms and a TCP window of 512 kB, we can estimate the theoretical throughput:

$$\frac{512 \times 1024 \times 8 \text{ bits}}{0.006 \text{ s}} \approx 682 \text{ Mbps.}$$

Yet, in practice, the throughput rarely reaches this value. Figure 8 shows that TCP throughput increases with the window size but quickly saturates around 340 Mbps once the window exceeds 1 MB. This threshold corresponds to the flat region in Figure 7, where higher MCS indexes no longer yield higher throughput.

This observation confirms that **the TCP window size is necessary, but not sufficient** to reach PHY-layer capacity. In our dataset, even with large windows (up to 4 MB), throughput remains capped. This suggests that **other bottlenecks are present**, such as:

- MAC-layer contention and backoff delays (CSMA/CA): Wi-Fi stations must wait for a clear channel before transmitting. In dense environments, this leads to random backoff delays, reducing efficiency and increasing latency.
- Protocol overhead (e.g., headers, retransmissions): Each packet includes headers from multiple layers (MAC/IP/TCP), and retransmissions add extra traffic. These reduce the proportion of useful data on the channel.
- Conservative TCP congestion control (e.g., CUBIC): TCP algorithms cautiously increase the send rate and react sharply to perceived congestion, often underutilizing available bandwidth in high-capacity links.
- Packet loss or jitter: Even low levels of loss or delay variation cause TCP to scale back its transmission rate, leading to lower sustained throughput.

Additionally, the use of a short guard interval in 802.11ax does increase the nominal PHY rate, but this benefit is not visible at the transport layer due to the cumulative overhead across the stack.

The boxplot in Figure 8 confirms that **TCP throughput only reaches its ceiling once the window is sufficiently large**, and then remains relatively unchanged. Achieving high throughput over Wi-Fi requires not only a high PHY rate, but also optimal tuning of protocol parameters, especially the TCP window size, and more generally, a deep optimization of the full protocol stack.

5 Dual-AP Experiments: Direct Duplicatin of Exp1

5.1 Objectives

The dual-AP experiment (Exp2) aimed to extend the single-AP pipeline (Exp1) to a multi-AP setup. The objective was:

- to capture the wifi and iperf metrics for different configurable wifi parameters in a multi APs operating simultaneously.
- Build a joint dataset combining synchronized measurements from AP1, Server1 and AP2, Server2 to see wifi comportment in a multi AP environment in order to provide richer features for ML-based Wi-Fi optimization.

5.2 System Architecture

The setup reproduced the single-AP pipeline on two APs, running in parallel and independently:

- **AP1 and AP2:** Two OpenWrt access points, configurable via SSH (`uci/iw`), each running the same data collection scripts as in Exp1.
- **Clients:** Two laptops acting as iPerf senders, each associated to a distinct AP.
- **Servers:** Two receivers collecting iPerf server-side metrics for each client stream.

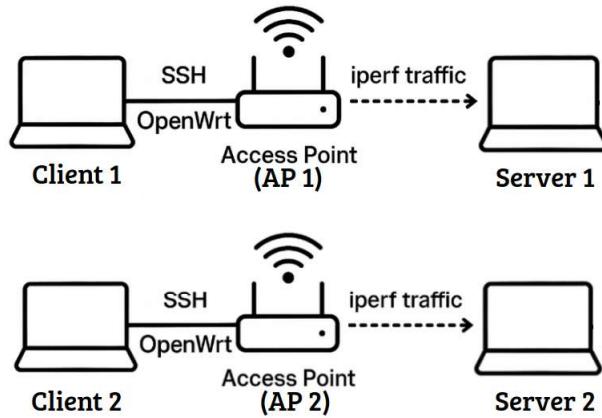


Figure 9: Dual-AP experiment pipeline (direct duplication).

5.3 Adaptations comparing to Exp1

Two main adaptations were introduced comparing to Exp1 the first was to shorten campaign duration because this time we have many more combinations and the second one was to randomize AP configurations for each AP in order to avoid generating identical sequences for both APs. So each AP generated the full configuration space (~576 combinations), shuffled it, and selected one configuration per run. This avoided systematic twin pairs between AP1 and AP2.

5.4 Execution Flow (per AP)

For each run, an AP:

1. Selected a random configuration, applied it, reloaded Wi-Fi, and waited for stabilization.

2. Launched 1 randomized iPerf tests, each with a unique `traffic_label`.
3. Logged client-side metrics locally (`/tmp/last_iperf_metric.jsonl`).
4. Triggered `collect_loop3.sh` to capture Wi-Fi snapshots, merge with the latest iPerf data, and append to `wifi_log2_apX.jsonl`.
5. Meanwhile, server listeners recorded throughput, loss, and jitter.

5.5 Output files

At the end of a campaign, the following files were produced:

- `wifi_log2_ap1.jsonl` and `wifi_log2_ap2.jsonl`: AP-side datasets (Wi-Fi + client metrics).
- Two `traffic_metrics_server.jsonl` files: server-side datasets with received throughput, packet loss, and jitter.

The intended approach was to merge AP1 and AP2 datasets horizontally (by timestamp or traffic label) to build paired AP1–AP2 states with an acceptance of 5s marge between both APs.

5.6 Observed Issue: Inter-AP Desynchronization

Although both loops were launched simultaneously, a time drift up to ~ 40 s appeared between AP1 and AP2. This dysynchronization between data capturing for AP1 and AP2 will be harmful later for ML training. In fact those two captures will be treated as if they were ran simultaneously which is not the case. The delay is caused by variable execution times (reloads, parsing delays..) and environmental variability (radio conditions, CPU load, interference)).

So this delay even a small drift (5 s) would be tolerable if both APs stayed under the same configuration window. However, asynchronous reconfigurations produced inconsistencies. For instance:

- t : AP1 = Config-A, AP2 = Config-B (AP1 snapshot).
- $t + 5$: AP1 switches to Config-C (AP2 snapshot).
- $t + 10$: AP2 switches to Config-D .

Merging by closest timestamp could merge the AP1 snapshot at t and AP2 snapshot at $t+5$. So we will have the config (C,B) that was just an intermediate state so we will have the config of the (C,B) pair but the metrics of (A,B) , introducing a wrong line in our dataset which will be harmful to the ML training.

5.7 Conclusion

The direct duplication of Exp1 to a dual-AP setup confirmed feasibility and produced richer datasets. However, it also revealed a structural limitation: the absence of synchronization between AP1 and AP2. The observed drift (up to ~ 40 s) led to asynchronous configurations and inconsistent merged pairs, introducing noise into the dataset.

But each merged line must reflect a truly simultaneous state ($\text{AP1} + \text{AP2} + \text{iPerf}$). Achieving this requires central orchestration which lead us to a controller-based architecture.

6 Dual-AP Experiments: Controller-Based Dual-AP Solution

6.1 Problem Statement

Naïve duplication of Exp1 to two APs produces desynchronisation: (i) configuration applied at different instants, (ii) iPerf starts misaligned, (iii) inconsistent AP1/AP2 merges. Result: non-auditable pairs, unusable for ML.

6.2 Objectives

The controller-based design eliminates the drift observed with the direct duplication approach. Specifically, we aim to:

- **Enforce synchronisation** across configuration, traffic generation, and metric capture on both AP1 and AP2;
- **Guarantee traceability**: each dataset line represents a truly *simultaneous* state ($\text{AP1} + \text{AP2} + \text{iPerf}$) under a unique label;
- **Apply rigorous quality control** (accept/reject per run) based on timing constraints and JSON validity, producing data that is fit for ML.

6.3 Design Principles

- **Decoupling of phases.** Configuration (**Phase A**) is strictly separated from measurement (**Phase B**): the controller applies exactly one shuffled Wi-Fi configuration per AP (no traffic), wait until it is done for both of them and then triggers one iPerf scenario per run.
- **Label as join key (and gating).** A controller-issued, unique `traffic_label` binds AP1, AP2 and iPerf artefacts; AP collectors(The loops that at the beginning works continuously) are now *label-gated* and do not capture until they received the label from the controller and then emit *exactly one* merged line per label (Wi-Fi snapshot + TX_DELTA + iPerf summary).
- **Atomic per-label export.** Each AP writes its per-label line in wifi log2.jsonl and in a temporary file containing only this line then `mv` (atomic rename) to `/tmp/by_label/<LABEL>.jsonl` ensuring the controller never reads partial JSON.
- **Time gating (quality control).** The controller accepts a pair only if both start and end timestamps align within ≤ 6 s; otherwise the run is rejected and logged, yielding a synchronised, auditable, ML-ready dataset.

6.4 System Architecture

- **Controller:** SSH orchestration; Phase A applies config; Phase B generates label and launches `iperf_with_txdelta.sh <LABEL>` on both APs; waits artefacts (temporary file containing the capture for this label; merges or rejects).
- **AP1/AP2 (OpenWrt):** Apply config; run one iPerf scenario; collector `collect_loop_exp2.sh` merges {Wi-Fi snapshot + TX_DELTA + iPerf} into one JSON; append to `wifi_log2.jsonl`; export per-label file atomically.
- **Clients:** Two laptops acting as iPerf senders, each associated to a distinct AP.
- **Servers:** Two receivers collecting iPerf server-side metrics for each client stream.
- **Switch:** Connect both APs and both Clients in the same LAN.

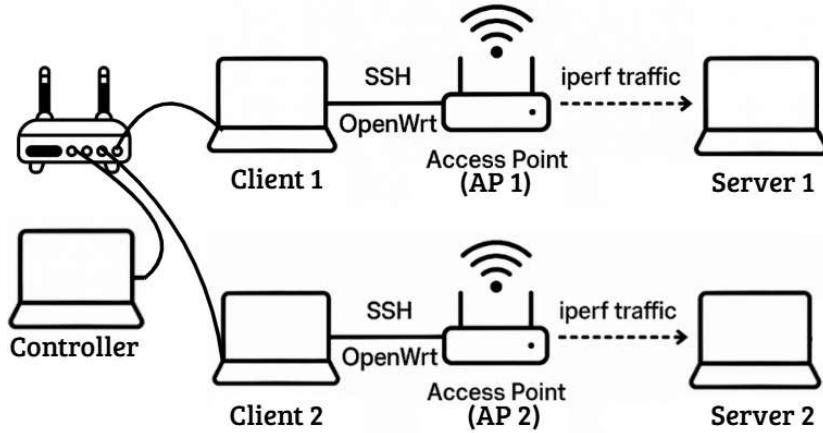


Figure 10: Controller based dual AP pipeline

6.5 Measurement Pipeline

Phase A (Configuration, no traffic).

1. The Controller sets up each AP WiFi configuration via this script: `wifi_test_orch_oneshot.sh`.
2. AP pops one config from shuffled queue (`/tmp/wifi_conf_queue.txt`): {channel, BW, TX power, NSS, CWmin/CWmax, RTS/CTS}.
3. Apply via UCI batch + `wifi reload`.

Phase B (Measurement, synchronised).

1. Controller creates label `RUN_YYYYMMDD_HHMMSS_i`; launches `iperf_with_txdelta.sh <LABEL>` on AP1/AP2.
2. On AP: `iperf_with_txdelta.sh` selects one scenario (TCP/UDP), calls `iperf_single_exp2.sh` (JSON + `iperf_start/end_ts`); computes per-client $\text{TX_DELTA} = \Delta \text{TX bytes}$ (before/after), writes to `tx_delta_log.jsonl`.

3. The loop: run `collect_wifi_metrics_json3.sh` (channel, BW, noise, busy%, per-client RSSI/bitrates/bytes, ICMP RTT); read label; merge into one JSON; append to `wifi_log2.jsonl` and export atomically to `/tmp/by_label/<LABEL>.json`.
4. Controller: poll both APs for per-label files (`texttt/tmp/by_label/<LABEL>.json`); accept if time-gated; merge into `dataset_dual_ap.jsonl` else log `dataset_dual_ap_rejected.jsonl`.

6.6 Comparison with Naïve Duplication

- Multi-scenario loop → **one scenario per call** (`iperf_with_txdelta.sh`).
- Local labels → **controller-issued label** (shared AP1/AP2/iPerf).
- Continuous capturing → **label-gated single capture** (exactly one JSON per label).
- No AP traffic view → **TX_DELTA** (per-station Δ TX bytes).
- Ad-hoc merge → **time-gated QC** (start/end \leq 6 s) + accept/reject logs.

6.7 Software Components and Roles

(a) Controller — dual_ap_controller.sh. Key functions: Orchestrates Phase \tilde{A} /Phase \tilde{B} , collects L1/L2, and merges or rejects. Outputs: `dataset_dual_ap.jsonl` (accepted) and `dataset_dual_ap_rejected.jsonl` (rejected).

(b) AP collector loop — collect_loop_exp2.sh. Period 30,s; **one line per label**. Steps: label read → Wi-Fi snapshot → TX_DELTA attach → iPerf gating → JSON validation via the controller.

(c) Wi-Fi snapshot — collect_wifi_metrics_json3.sh. : configuration (channel, frequency, BW, TX power, WMM), survey (noise, busy), ICMP RTT, and station stats (RSSI, retries, bitrates, bytes, MCS). Produces `wifi_metrics2.json` (label and timestamp included).

(d) iPerf + TX_DELTA — iperf_with_txdelta.sh. Receives the label; selects a *single* iPerf scenario (TCP -P1,2,4,8,16, UDP -b5M...400M, TCP -w, default -t 10). Measures per-client TX bytes before/after via iw, computes tx_delta_bytes and estimated_throughput_bps, appends to `tx_delta_log.jsonl`.

(e) Single iPerf run — iperf_single_exp2.sh. Run the iperf traffic with the given parameters, extracts (TCP) `end.sum_sent.bits_per_second` or (UDP) `end.sum.bits_per_second`, `lost_percent`, `jitter_ms`; produces one JSON line (with `iperf_start_ts`/`iperf_end_ts`) to `traffic_metrics.jsonl` and `/tmp/last_iperf_metric.jsonl`.

6.8 Workflow Diagram

Produced files on the controller:

- `dataset_dual_ap.jsonl`: accepted, synchronised AP1+AP2 pairs (timestamps and labels);
- `dataset_dual_ap_rejected.jsonl`: rejected pairs (missing line, desync \$>\$ 6,s, invalid JSON).

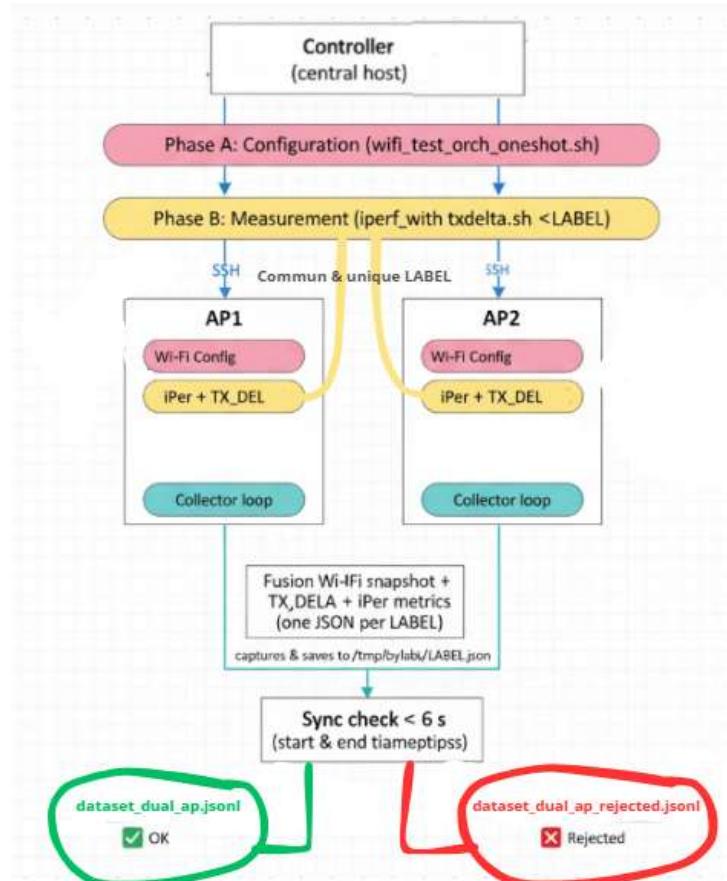


Figure 11: Controller-Based Dual-AP

6.9 Results and Discussion

At the end of the collection process, we loaded the `dataset_dual_ap.jsonl` file in `jupyterlab` and turn it using `pandas` into a 1000 line dataset.

6.9.1 iPerf throughput & TX_DELTA

In our dataset we rely on two complementary throughput measures:

iPerf throughput corresponds to the application-level goodput reported by iPerf3 at the client side. It is expressed directly in **bits per second** over the test interval (`iperf_start_ts` to `iperf_end_ts`), it excludes MAC-layer retries and any frames not received by the client; it's the net payload seen by the receiver

By contrast, **TX_DELTA estimated throughput** is computed at the AP side in `iperf_with_txdelta.sh`: the script samples per-station transmit byte counters (`iw station get ... tx bytes`) immediately before and after the labeled run, then takes the difference and divides by the run duration. Formally, for a client station:

$$\text{TX_DELTA throughput (bps)} = \frac{8 \times (\text{TX_bytes}_{\text{after}} - \text{TX_bytes}_{\text{before}})}{\Delta t}$$

where `TX_bytes` are counters expressed in `bytes`, and $\Delta t = \text{iperf_end_ts} - \text{iperf_start_ts}$. This AP-side metric thus represents the volume of bytes attempted for transmission by the AP, including retransmissions and MAC-layer overhead.

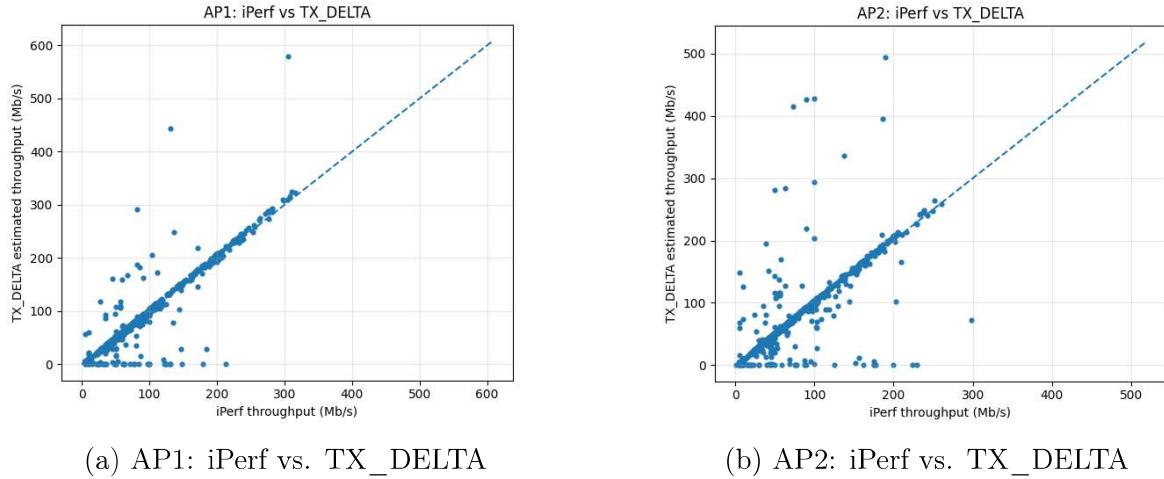


Figure 12: Comparison of iPerf throughput and TX_DELTA estimated throughput.

Figure 11 compares the two measures for AP1 and AP2. Most points align closely with the diagonal $y = x$, showing that TX_DELTA is consistent with iPerf in the majority of runs. Deviations *above* the diagonal reflect situations where the AP transmitted more bytes than the client actually received as useful payload (e.g., link-layer retries or losses), while the few points *below* the diagonal correspond to counter or timing mismatches. The correlation remains high (≈ 0.92 for AP1, ≈ 0.80 for AP2), confirming TX_DELTA as a reliable cross-check and an indicator of efficiency beyond what iPerf alone can reveal.

6.9.2 Results and Discussion on the Final Dataset

The final dataset comprises over 1,000 synchronized runs collected under the controller-based dual-AP architecture. Each entry aggregates a complete experiment cycle, including Wi-Fi configuration parameters (channel, bandwidth, TX power, NSS, contention window, RTS/CTS), traffic metrics from iPerf (protocol, throughput, jitter, loss), AP-side TX_DELTA counters, and per-station Wi-Fi statistics (RSSI, retries, bitrates, bytes). Importantly, every run is tagged with a unique `traffic_label`, allowing the controller to merge artefacts from AP1 and AP2 only if temporal alignment (start/end timestamps) was within the 6-second tolerance. Runs failing this criterion were systematically excluded and logged, ensuring that the dataset is auditable and machine-learning ready.

This structure yields several benefits: (i) a balanced coverage of protocol scenarios (TCP/UDP with varying rates and parallel streams), (ii) a wide distribution of link conditions due to randomized radio configurations, and (iii) dual-AP synchronization, enabling

comparative analysis of interference and shared-medium effects. The resulting dataset thus provides both reliability (through strict acceptance rules) and diversity (through randomized scenarios), positioning it as a solid foundation for training and evaluating ML models in Wi-Fi optimization tasks.

6.10 Reproducibility (Procedure)

1. Deploy scripts under `/root` on both APs and the controller; ensure dependencies (`iperf3`, `jq`, `iw`, `uci`).er scenario to capture as well the server metrics
2. Let the servers in listning mode via `iperf3 -s` or `iperf lsn`
3. Start the collect loop in the background of both APs (After cleaning the old ones
4. Start the controller: `"./dual_ap_controller.sh"`after cleaning it
5. Monitor:`dataset_dual_ap_rejected.jsonl` (controller) and `wifi_log2.jsonl` (APs).

Conclusion. A central controller synchronises AP configurations and iPerf sessions; paired runs are accepted only when both start and end timestamps align within ± 6 s. This yields an ML-ready, auditable dataset while remaining compatible with OpenWrt tooling (Bash, `iPerf3`, `jq`, `iw`, `uci`).

7 Conclusion and Perspectives

This work presented the design and implementation of a controller-based dual-AP measurement framework for Wi-Fi performance optimization. Starting from a single-AP setup, we identified the limitations of naive duplication—namely, temporal desynchronisation and inconsistent pairing of metrics—and addressed them by introducing a central controller. The controller enforces a strict separation between configuration and measurement phases, issues unique traffic labels, and validates synchronisation within a six-second tolerance before merging results.

The final outcome is a dataset of over 1,000 synchronized runs, each combining iPerf traffic metrics, AP-side TX_DELTA counters, and detailed Wi-Fi snapshots. The dataset is both diverse, thanks to randomized scenarios and radio configurations, and reliable, due to rigorous acceptance and rejection policies. This makes it suitable for downstream machine learning tasks such as channel allocation, or throughput prediction.

Beyond the dataset itself, the proposed pipeline demonstrates good practices for experimental reproducibility: atomic per-label exports, robust error handling, and full traceability of rejections. The approach can be extended to more than two APs, to finer-grained synchronisation mechanisms (e.g., NTP or hardware timestamps), and to richer metrics (e.g., spectral scans or airtime usage).

In summary, this work delivers both a methodological contribution—an orchestrated and auditable measurement pipeline—and a practical asset: a large-scale, high-quality dataset ready for machine-learning-based Wi-Fi optimization.

Bibliography

- [1] B. Bellalta, “Ieee 802.11ax: High-efficiency wlans,” *IEEE Wireless Communications*, vol. 23, no. 1, pp. 38–46, 2016.
- [2] C. Jiang, H. Zhang, Y. Ren, and Z. Han, “Deep learning for wireless traffic prediction and classification,” in *IEEE ICC 2019*, pp. 1–6, IEEE, 2019.
- [3] F. Chen *et al.*, “Real-time wifi performance monitoring and prediction with machine learning,” *Future Research in Sustainable Information Processing*, vol. 2, no. 1, pp. 1–10, 2020.
- [4] X. Chen, Y. Fang, and P. Yang, “Wi-fi data-driven network optimization: Challenges and opportunities,” in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1241–1246, IEEE, 2020.
- [5] A. Mishra, M. Shin, and W. A. Arbaugh, “An empirical analysis of the ieee 802.11 mac layer handoff process,” in *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement*, pp. 93–102, ACM, 2003.
- [6] G. Fang, L. Qiu, Z. Zhong, P. Zhang, and S. Banerjee, “The impact of interference on wireless network performance,” in *Proceedings IEEE INFOCOM 2010*, pp. 1–9, IEEE, 2010.
- [7] H. Mao, R. Netravali, and M. Alizadeh, “A survey on machine learning for networking,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.
- [8] “Wi-fi configuration on openwrt.” <https://openwrt.org/docs/guide-user/network/wifi/basic>, 2025. Accessed Sep. 21, 2025.
- [9] “Uci (unified configuration interface) guide.” <https://openwrt.org/docs/guide-user/base-system/uci>, 2025. Accessed Sep. 21, 2025.
- [10] J. Malinen and contributors, “hostapd: Ieee 802.11 access point and authentication server.” <https://w1.fi/hostapd/>, 2025. Accessed Sep. 21, 2025.
- [11] “iw — nl80211-based cli for wireless devices.” <https://wireless.docs.kernel.org/en/latest/en/users/documentation/iw.html>, 2025. Accessed Sep. 21, 2025.

Annexes

This section gathers supplementary material referenced in the report.

List of Annexes

- A.1** Transmission Power & Latency (script and observations)
- A.2** Channel Change Procedures
- A.3** Bandwidth Switching (HE20/HE80)
- A.4** WMM / Contention Window Configuration
- A.5** Forcing MCS/NSS

- A.6** Prise en compte des impacts écologiques et sociétaux (DDRSE)
- A.7** Preuve de la compétence C1: Travailler et communiquer en environnement international et interculturel
- A.8** Preuve de la compétence C2: Concevoir ou réaliser des solutions d'ingénierie, permettant de répondre à un cahier des charges
- A.9** Preuve de compétence C3: Mettre en œuvre une démarche d'innovation et de recherche
- A.10** Preuve de compétence C4: Travailler et coopérer dans une équipe
- A.11** Preuve de compétence C5: Tenir compte des transitions technologiques, environnementales et sociétales (RSE)

Annex A.1 — Transmission Power & Latency

To begin experimenting with Wi-Fi parameter optimization, I focused on transmission power. The goal was to observe how adjusting the AP's transmit power affects the latency experienced by a wireless client.

I used two machines:

- **PC1:** connected via Ethernet to the OpenWrt router (used to control the AP).
- **PC2:** connected via Wi-Fi to the AP (used to send ICMP ping requests).

On PC1, I created and executed the following script to vary the transmission power of the AP:

```
#!/bin/sh

echo "Before changing power..."
iw dev phy1-ap0 info | grep txpower

echo "Start ping on Wi-Fi client (PC2)..."
sleep 10

echo "Set txpower to 15 dBm..."
iw dev phy1-ap0 set txpower fixed 1500
sleep 2
iw dev phy1-ap0 info | grep txpower

echo "Observe ping behavior..."
sleep 15

echo "Restore txpower to 23 dBm..."
iw dev phy1-ap0 set txpower fixed 2300
sleep 2
iw dev phy1-ap0 info | grep txpower

echo "Test completed."
```

In parallel, on PC2, I launched:

```
ping 8.8.8.8
```

The IP address 8.8.8.8 corresponds to Google's public DNS server, commonly used for testing connectivity due to its high availability and low response time.

Observations:

- When close to the AP, reducing the transmit power from 23 dBm to 15 dBm had little effect on latency (remaining around 13–15 ms).
- When moving further away (outside the room), significant variations appeared, with latencies spiking up to 124 ms:

```

icmp_seq=7 time=124 ms
icmp_seq=8 time=96.0 ms
icmp_seq=9 time=79.5 ms
icmp_seq=10 time=105 ms
...
rtt min/avg/max/mdev = 13.849/33.065/124.436/29.265 ms

```

Key observations unchanged; referenced in Sec. 3.1.

Annex A.2 — Channel Change Procedures

1. Initial Attempt: Direct Channel Change

We first tried changing the frequency directly using:

```
iw dev phy1-ap0 set freq 5200 80 5210
```

However, this returned the error:

```
command failed: Resource busy (-16)
```

This suggests that the Wi-Fi driver does not allow channel switching while the interface is active, especially in AP mode.

2. Temporary Interface Shutdown

Next, we disabled the interface before attempting the channel switch:

```
ip link set phy1-ap0 down
iw dev phy1-ap0 set freq ...
ip link set phy1-ap0 up
```

This approach worked and applied the channel change successfully. However, the test revealed that changing the AP’s channel while clients are connected:

- causes temporary disconnections,
- increases latency,
- and leads to significant packet loss (up to 41%).

3. Configuration Update Using UCI

We also tested channel switching by modifying the configuration with UCI:

```
uci set wireless.radio1.channel='36'
uci commit wireless
wifi reload
```

This method did not successfully apply the change. The system appeared to ignore the new setting — likely because the interface remained active during the reload.

4. Full Wireless System Restart (Most Reliable Method)

The most effective method involved a complete shutdown and restart of the wireless system:

```
wifi down  
uci set wireless.radio1.channel='36'  
uci commit wireless  
wifi up
```

This approach ensures:

- the change is properly applied,
- it persists after reboot,
- and it causes only a brief network outage (15%).

The final script using this method is shown below:

```
#!/bin/sh  
echo "Initial_channel:"  
iw dev phy1-ap0 info | grep channel  
echo "Disabling Wi-Fi..."  
wifi down  
sleep 2  
echo "Switching to channel 36..."  
uci set wireless.radio1.channel='36'  
uci commit wireless  
wifi up  
sleep 10  
echo "Current_channel:"  
iw dev phy1-ap0 info | grep channel
```

Annex A.3 — Bandwidth Switching (HE20/HE80)

```
#!/bin/sh  
# Affiche la bande initiale  
echo "Largeur initiale :"  
iw dev phy1-ap0 info | grep width  
# Passage HE80 HE20  
echo "Passage HE80 (20 MHz)..."  
wifi down  
uci set wireless.radio1.htmode=HE20  
uci commit wireless  
wifi up  
sleep 15  
iw dev phy1-ap0 info | grep width  
# Retour HE20 HE80  
echo "Retour HE80 (80 MHz)..."  
wifi down
```

```

uci set wireless.radio1.htmode= HE80
uci commit wireless
wifi up
sleep 15
iw dev phy1-ap0 info | grep width
echo "Test termin ."

```

Annex A.4 — WMM / Contention Window

```

uci set wireless.default_radio1.wmm_ac_be='cwmin=31,cwmax=1023,aifs=3'
uci set wireless.default_radio1.wmm_ac_vi='cwmin=7,cwmax=15,aifs=2,txop=300'
uci commit wireless
wifi reload

```

The changes were applied successfully. To verify this command can be used:

```

uci get wireless.default_radio1.wmm_ac_be
which returned:

```

```

cwmin=31 cwmax=1023 aifs=3
cwmin=7 cwmax=15 aifs=2 txop=3008

```

indicating the new WMM settings were correctly stored and active on the 5 GHz access point (`default_radio1`).

Annex A.5 — Forcing MCS/NSS

Forcing specific Modulation and Coding Scheme (MCS) values allows us to evaluate how a fixed data rate performs under different radio conditions. This is particularly useful for testing the relevance and performance of adaptive rate algorithms in real scenarios. The general syntax `iw dev <iface> set bitrates ht-mcs-5 0-7` is used to constrain the allowed MCS values, but its effectiveness depends on the PHY family (HT, VHT, or HE) supported by the connected devices on the 5 GHz band.

For example, in HT mode (802.11n), the following command executed without error:

```

iw dev phy1-ap0 set bitrates ht-mcs-5 7          # force HT-MCS 7

```

However, since the connected stations were already negotiating higher-order VHT or HE rates, this restriction had no visible effect. This was confirmed by inspecting the current transmission rates using `iw dev phy1-ap0 station dump | grep 'tx bitrate'`, which revealed one station using VHT and another using HE.

Next, I attempted to set a fixed VHT rate:

```

iw dev phy1-ap0 set bitrates vht-mcs-5 9:9          # returns "Invalid argument"

```

This failed because the correct VHT (or HE) syntax requires the spatial stream count, in the form `NSS:MCS`. Once corrected, the commands were accepted. What means 1 stream? Is that we are using a single spatial stream? Can we set it two 2? Can we allow for all MCSs, but change from 1 to 2 Spatial streams?: (By “1 stream”, I meant setting the Number of Spatial Streams (NSS) to 1, which forces the use of a single spatial stream. It is possible to set it to 2 streams with a command like:

```
iw dev phy1-ap0 set bitrates vht-mcs-5 2:9
```

provided both the AP and the client station support multiple spatial streams (the maximum supported NSS for each device can be checked using iw list).

However, it is not possible to vary only the NSS (Number of Spatial Streams) without specifying an exact MCS or an MCS range: the iw command requires you to indicate the NSS together with either a specific MCS or a range of MCS values.

That said, it is possible to set a range of MCSs for a given NSS, which effectively sweeps across all MCS values for that chosen NSS. For example, to allow all MCS 0–9 on NSS=1, we can use:

```
iw dev phy1-ap0 set bitrates vht-mcs-5 1:0-9 )
```

```
iw dev phy1-ap0 set bitrates vht-mcs-5 1:9      # VHT-MCS 9 on 1 stream
iw dev phy1-ap0 set bitrates he-mcs-5 1:11      # HE-MCS 11 on 1 stream
```

The changes were reflected in the bitrate report:

```
tx bitrate: 390.0 MBit/s 80MHz VHT-MCS 9 VHT-NSS 1
# or
```

```
tx bitrate: 1200.9 MBit/s 80MHz HE-MCS 11 HE-NSS 2 \h1{Here , we have 2 spa
```

Finally, I restored automatic rate adaptation by clearing the forced configuration:

```
iw dev phy1-ap0 set bitrates
```

These experiments show that OpenWrt can indeed enforce fixed MCS values, provided the proper PHY family and syntax are used.

Annex A.6 — Prise en compte des impacts écologiques et sociétaux (DD&RSE)

Mon empreinte GES pendant le stage. Durant mon stage, mon empreinte carbone provient principalement de :

- **Déplacements** : trajets domicile–UPF en métro (environ 2×45 minutes par jour). L'impact carbone du métro est faible : en ordre de grandeur, $\approx 4 \text{ gCO}_2/\text{km}$. Pour un trajet journalier d'environ 20 km aller-retour, cela représente $\approx 80 \text{ gCO}_2/\text{jour}$.
- **Électricité au poste** : utilisation de 5 PC portables, chacun $\approx 50 \text{ W}$. Soit $5 \times 50 = 250 \text{ W} \times 6 \text{ h/jour} \rightarrow 1.5 \text{ kWh/jour}$. En prenant un facteur d'émission de $0.25 \text{ kgCO}_2/\text{kWh}$ (mix européen), cela équivaut à $1.5 \times 0.25 = 0.375 \text{ kgCO}_2/\text{jour}$.
- **Banc d'essai réseau** : deux points d'accès, chacun entre 8 et 12 W. En prenant une moyenne de $10 \text{ W} \times 2 = 20 \text{ W} \times 3 \text{ h/jour} \rightarrow 0.06 \text{ kWh/jour}$ ($\approx 0.015 \text{ kgCO}_2/\text{jour}$).
- **Autres équipements** : switch $< 10 \text{ W}$ cumulés. Sur 3 h/jour, cela représente 0.03 kWh/jour ($\approx 0.0075 \text{ kgCO}_2/\text{jour}$).
- **Ordre de grandeur global** :
 - Déplacements : $\approx 0.08 \text{ kgCO}_2/\text{jour}$
 - Poste de travail : $\approx 0.375 \text{ kgCO}_2/\text{jour}$
 - Réseau (APs + switch) : $\approx 0.02 \text{ kgCO}_2/\text{jour}$

Soit un total d'environ $0.475 \text{ kgCO}_2/\text{jour}$ en période de tests intensifs.

Actions de réduction et bonnes pratiques.

- Campagnes de tests plus courtes et ciblées. les scripts de collectes de donnés/reception d'iperf en fin de test.
- Transports : marche/métro uniquement.
- Transports : Prendre le train/Bus au lieu de l'avion pour retourner en France.

Aspects éthiques, diversité et déchets. Le projet de stage a soulevé plusieurs enjeux éthiques et sociétaux. Tout d'abord, la protection de la vie privée a été une priorité : les adresses MAC et identifiants collectés ont été systématiquement anonymisés et les fichiers de données ont été stockés de manière sécurisée.

La diversité et l'inclusion constituaient également une dimension importante, notamment au sein d'une équipe internationale : l'anglais a été choisi comme langue de travail afin de permettre à chacun de s'exprimer et de contribuer sans barrière linguistique. Par ailleurs, un effort conscient a été fait pour respecter toutes les personnes indépendamment de leur origine, de leur religion ou de leurs idées, afin de préserver un climat de collaboration sain et respectueux. Sur le plan du bien-être au travail, l'organisation des journées a pris en compte la nécessité de préserver la santé : des pauses régulières ont été observées pour éviter la fatigue visuelle, et une alternance entre position assise et debout a été mise en place afin de limiter les tensions posturales.

Enfin, la gestion des déchets a également été intégrée à la démarche. Le matériel informatique utilisé était majoritairement issu du réemploi, évitant ainsi des achats superflus et prolongeant la durée de vie des équipements. Lorsqu'il s'agissait de consommables ou d'emballages, ceux-ci étaient systématiquement triés afin de respecter les filières de recyclage. À terme, l'ensemble de ces pratiques contribue à donner au projet une dimension responsable, qui ne se limite pas à la technique mais prend en compte ses impacts sociaux et environnementaux. Cette approche éthique et inclusive renforce la valeur du stage, en démontrant que l'ingénierie peut et doit intégrer des préoccupations humaines et durables dans sa pratique quotidienne.

Utilité du projet. L'utilité de ce projet peut être analysée à différents niveaux, allant de la structure d'accueil jusqu'à mon propre parcours d'apprentissage.

Pour la structure, il s'agit avant tout d'un travail qui s'inscrit dans un champ de recherche en plein essor : l'optimisation des réseaux sans fil par des approches d'intelligence artificielle et de machine learning. Les réseaux Wi-Fi constituent aujourd'hui une ressource critique surtout dans des environnements critiques comme les hôpitaux, les écoles.. Pouvoir disposer d'un contrôleur intelligent capable d'adapter dynamiquement des paramètres tels que le canal, la largeur de bande, le schéma de modulation ou encore la puissance d'émission représente un enjeu stratégique, car cela permet non seulement d'améliorer la qualité de service, mais aussi de tendre vers une utilisation plus sobre et efficiente des ressources radio. À terme, ce type d'outil peut renforcer l'image d'innovation de la structure et ouvrir la voie à de futures collaborations de recherche dans le domaine des smart networks.

Du point de vue **des utilisateurs finaux**, l'apport du projet est également significatif. Un réseau plus stable et plus rapide, avec une latence réduite et un meilleur débit, constitue un confort quotidien non négligeable pour tout usager connecté. Dans des environnements bruités ou saturés, où plusieurs réseaux Wi-Fi coexistent et interfèrent, les améliorations apportées par le projet peuvent se traduire concrètement par une diminution des micro-coupures, une continuité de service renforcée et une meilleure fluidité lors d'activités sensibles comme les visioconférences ou les transferts de données lourdes.

Enfin, **sur le plan personnel**, ce projet a représenté une étape d'apprentissage particulièrement riche. Il m'a permis d'acquérir une méthodologie complète de collecte et de traitement de données, en mettant en place des scripts automatisés capables d'enregistrer des métriques réseau en temps réel. J'ai ainsi développé une compréhension approfondie du fonctionnement interne des points d'accès Wi-Fi, de leurs configurations avancées via OpenWrt, et des mécanismes de communication qui régissent la qualité d'un lien sans fil. Au-delà de la technique, ce stage m'a appris à gérer un projet de bout en bout : définir des objectifs clairs, mettre en place une organisation reproductible, analyser des résultats en gardant une vision critique et ajuster les expériences en fonction des contraintes réelles. Cette expérience m'a également sensibilisé à l'importance de documenter les processus et de toujours relier la technique à son utilité concrète pour la société. En résumé, l'utilité de ce projet réside autant dans les bénéfices apportés à la structure et aux utilisateurs que dans la croissance de mes compétences d'ingénierie en devenir.

Évaluation des impacts positifs et négatifs du projet. L'évaluation des impacts du projet ne peut se limiter à la seule phase d'usage ; elle doit intégrer une réflexion plus large inspirée de l'analyse du cycle de vie. Au stade de la fabrication, l'impact environnemental est relativement faible, puisque l'ensemble des équipements utilisés (points d'accès, switchs, contrôleur, PC portables) provenait déjà de ceux existant dans le lab et il n'y a pas la nécessité d'achats neufs. Cela illustre une démarche de réutilisation et de prolongation de la durée de vie du matériel. La distribution est également inexisteante dans ce projet, le matériel étant disponible localement, ce qui réduit encore les impacts logistiques sauf mon transport pour arriver au labo ce qui est négligeable. La phase d'usage constitue le cœur de l'analyse : la consommation électrique des APs et des ordinateurs a été le principal poste d'impact. En ce qui concerne la maintenance et la fin de vie, le projet encourage des pratiques de mutualisation et de durabilité : les scripts développés sont documentés et réutilisables par d'autres étudiants ou chercheurs, ce qui augmente leur valeur au-delà du stage. Les équipements peuvent être réparés et reconfigurés facilement, grâce à l'usage d'OpenWrt qui prolonge l'autonomie et la flexibilité des points d'accès. En cas de fin de vie matérielle, les dispositifs peuvent intégrer les filières de recyclage électronique (DEEE), mais la stratégie privilégiée reste le réemploi, qui maximise la valeur environnementale.

En revanche, il convient de rester attentif aux effets rebond : une meilleure qualité de connexion pourrait inciter à une consommation accrue de données, annulant une partie des gains énergétiques. Ainsi, l'impact global du projet peut être jugé majoritairement positif, à condition que les pratiques de sobriété, de mutualisation et de réemploi soient systématiquement intégrées dans la démarche.

Propositions de solutions d'ingénierie durable. L'un des principaux défis écologiques et sociétaux liés à mon projet est la sobriété numérique, c'est-à-dire la capacité à fournir une qualité de service optimale tout en minimisant la consommation énergétique par gigaoctet transféré. Ce problème est d'autant plus important que les usages numériques explosent et que la consommation énergétique du secteur des télécommunications croît rapidement. Pour y répondre, une première piste d'ingénierie consiste à concevoir un contrôleur multi-objectif qui n'optimise pas uniquement les performances (débit, latence), mais qui intègre aussi un indicateur énergétique. Cette approche suppose de développer des méthodes de caractérisation précises : par exemple, instrumenter les points d'accès avec des wattmètres connectés afin de mesurer la consommation réelle en fonction des paramètres choisis (canal, largeur de bande, MCS, puissance). Ces données pourraient ensuite être intégrées dans les modèles d'apprentissage afin d'entraîner un système capable de prendre des décisions équilibrées entre qualité de service et sobriété énergétique.

Il serait également pertinent d'introduire un mécanisme de détection d'effets indésirables, comme l'effet rebond : si une amélioration de la qualité de connexion conduit à une explosion de l'usage (streaming haute définition, téléchargements massifs), l'impact environnemental global peut devenir négatif. Dans ce cas, le contrôleur pourrait proposer des modes de fonctionnement « sobriété » ou « éco » à certains moments de la journée, par exemple en limitant automatiquement la largeur de bande aux heures creuses.

Enfin, une dimension importante est la réutilisation. Les scripts développés doivent être documentés et mis à disposition de futurs projets, ce qui évite de repartir de zéro et favorise une amélioration incrémentale collective. De même, l'usage d'OpenWrt permet une grande autonomie et prolonge la durée de vie des équipements en leur offrant des

fonctionnalités avancées au-delà de leur usage commercial initial. Ainsi, les solutions proposées ne visent pas seulement à réduire l'empreinte carbone immédiate, mais s'inscrivent dans une démarche plus large de durabilité, d'adaptabilité et de résilience des systèmes complexes. Elles démontrent que l'ingénierie peut apporter des réponses innovantes à des problèmes techniques mais aussi environnementaux.

Auto-évaluation (0–4).

- Empreinte GES : 3
- Réduction/bonnes pratiques : 3
- Éthique/diversité/déchets : 4
- Utilité (organisation/utilisateur/perso) : 4
- ACV et mitigations : 3
- Ingénierie durable : 2
- Gestion des incertitudes : 2

Annex A.7 — Preuve de la compétence C1: Travail et communiquer en environnement international et interculturel

Au cours de mon parcours académique et professionnel, j'ai évolué dans des environnements où la diversité culturelle et internationale était centrale. J'ai commencé ce parcours en venant en France pour poursuivre mes études d'ingénieur, après deux années de classes préparatoires en Tunisie. Pendant ces années, j'ai côtoyé des étudiants de multiples nationalités. Cette immersion m'a appris à travailler et à communiquer avec des personnes ayant des idées, des traditions et des cultures différentes. J'ai compris que ce que je pensais évident dans mon pays ne l'était pas ailleurs, et que chacun a ses propres valeurs et perspectives. J'ai ainsi appris à respecter ces différences et à les considérer comme une richesse. Au Graduate School, la diversité était encore plus marquée : mes camarades venaient d'Algérie, du Brésil, d'Argentine, d'Espagne, de France et d'autres pays. Le partage de nos expériences m'a permis de découvrir des points de vue nouveaux, d'élargir ma vision et de renforcer mon ouverture d'esprit. Mon stage en Espagne, à l'Universitat Pompeu Fabra de Barcelone, a constitué une étape plus exigeante. Partie seule, j'ai dû m'adapter rapidement, m'intégrer dans un nouvel environnement et créer des liens. J'ai appris à travailler dans un contexte où les méthodes de communication et les façons de collaborer variaient selon les nationalités. Cette expérience a développé mon adaptabilité, aussi bien dans la gestion de projets que dans la vie quotidienne. La maîtrise des langues étrangères a été un atout essentiel. L'anglais, langue principale de travail, m'a permis de progresser dans l'expression technique et scientifique. L'espagnol, utilisé dans les échanges quotidiens, m'a aidée à m'intégrer et à saisir les implicites culturels, au-delà des mots. Ces expériences m'ont également confrontée à des situations d'ambiguïté interculturelle : une remarque ou un geste ne sont pas toujours interprétés de la même façon selon la culture. J'ai appris à accueillir ces différences avec tolérance et à ajuster ma communication, afin de rester efficace tout en respectant la diversité. En résumé, mon parcours entre la Tunisie, la France et l'Espagne illustre ma capacité à travailler et communiquer dans des environnements internationaux. J'ai su m'adapter, collaborer avec des profils variés, développer une réelle ouverture d'esprit et utiliser mes compétences linguistiques pour dépasser les barrières culturelles.

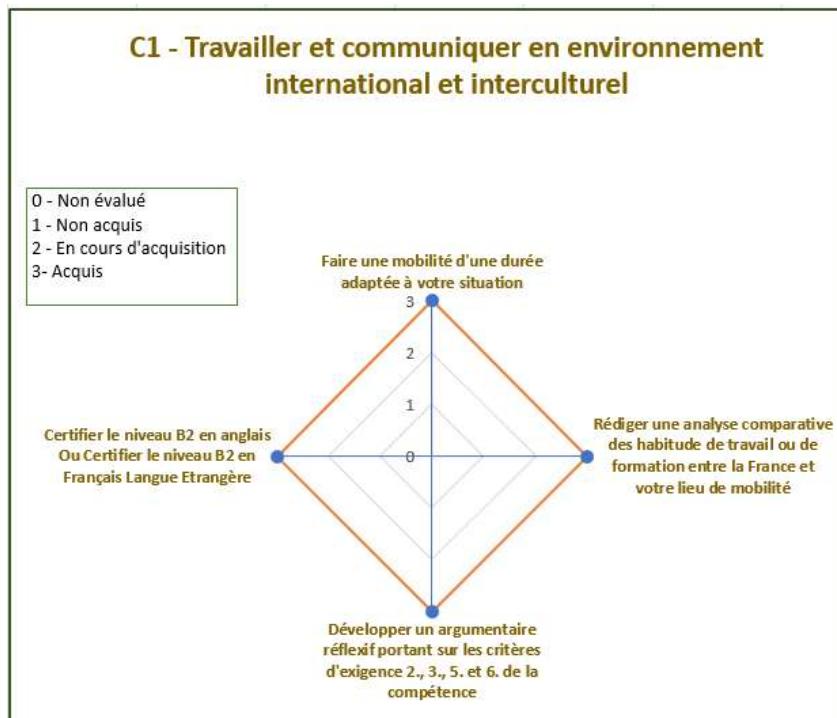


Figure 13: C1: Travailler et communiquer en environnement international et interculturel

Annex A.8 — Preuve de la compétence C2: Concevoir ou réaliser des solutions d'ingénierie, permettant de répondre à un cahier des charges

Tout au long de mon parcours académique et extrascolaire, j'ai pu développer la compétence de trouver des solutions d'ingénierie pour répondre aux cahiers de charges proposés. Pour commencer, dans les projets de bureaux d'études en électronique, chaque projet était accompagné d'un cahier des charges que nous devions étudier, comprendre et respecter dans un temps donné. Par exemple, lors de la conception de l'amplificateur audio sur circuit imprimé (PCB), on nous a donné des spécifications précises. J'ai donc appris à lire et analyser ces exigences, à concevoir une architecture adaptée, à choisir les composants, à utiliser KiCad pour la conception, et à faire des recherches supplémentaires pour proposer une solution fonctionnelle. En d'autres termes, j'ai appris à transformer un cahier des charges en une solution d'ingénierie testée et validée.

Par la suite, j'ai renforcé cette compétence en tant que chef de projet à la Junior Étude Phelma (JEP). Dans ce cadre, je ne me suis pas limitée à suivre un protocole établi : j'ai aussi appris à proposer une démarche de travail adaptée aux besoins du client. Avec l'étudiant chargé de la réalisation, nous avons défini une méthodologie claire, en incluant les ressources et les délais nécessaires. J'ai aussi mobilisé mes connaissances et celles de mes ressources externes (autres membres de la JEP) pour tester et analyser de manière critique la solution développée. Et à la fin, on a documenté le travail (méthodes, résultats, organisation) pour en permettre la réutilisation ultérieure par l'entreprise. Cette expérience m'a montré l'importance de conjuguer rigueur technique et gestion de projet afin de livrer des solutions d'ingénierie robustes, efficaces répondant au cahier des charges proposé.

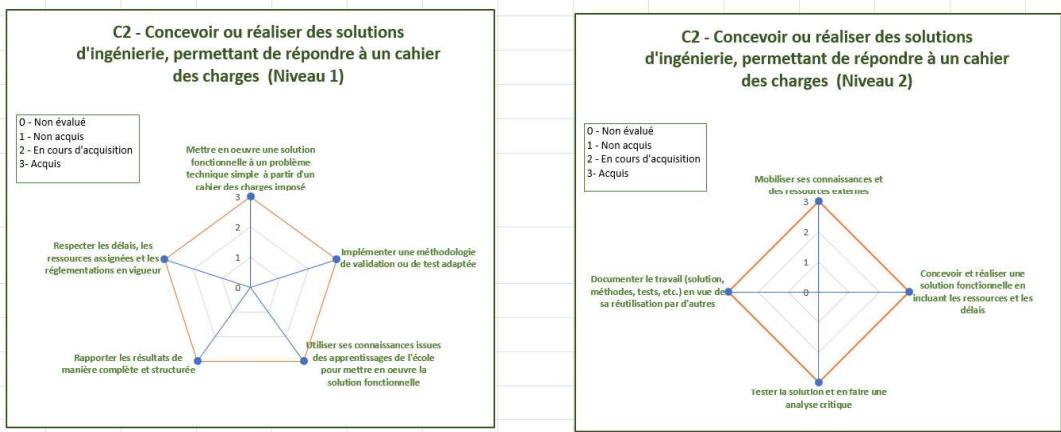


Figure 14: C2: Concevoir ou réaliser des solutions d'ingénierie, permettant de répondre à un cahier des charges

Annex A.9 — Preuve de compétence C3: Mettre en œuvre une démarche d'innovation et de recherche, Concevoir des solution d'ingenierie

Dans le cadre de mon stage de recherche à Barcelone, j'ai eu l'opportunité de conduire un projet novateur visant l'optimisation des réseaux Wi-Fi par l'intégration de techniques de machine learning. Ce projet constitue une illustration complète de la mise en œuvre d'une démarche d'innovation appliquée.

Au début, l'idée était assez simple : nous voulions explorer comment les paramètres d'un réseau Wi-Fi pouvaient être optimisés dynamiquement. Nous avons commencé par une phase de recherche sur l'état de l'art et l'identification des paramètres Wi-Fi les plus influents (comme le canal, la largeur de bande, la puissance d'émission). Une première série d'expériences a été menée sur un point d'accès unique, afin de constituer un dataset de référence et de valider nos hypothèses initiales.

Rapidement, nous avons rencontré des défis : les résultats obtenus avec un seul point d'accès ne suffisaient pas à capturer la complexité des environnements réels. Nous avons donc étendu l'expérimentation à une configuration avec deux points d'accès, doublant ainsi la complexité des scénarios. Cette démarche progressive nous a permis d'ajuster notre approche : face aux premiers résultats peu concluants, nous avons affiné nos méthodes de collecte et de modélisation.

Au fil du projet, j'ai appliqué une démarche d'ingénierie où chaque difficulté était l'occasion de repenser nos choix et de proposer des ajustements. Nous avons ainsi mis en place une stratégie d'échantillonnage aléatoire (type Monte Carlo) pour éviter une exploration exhaustive trop lourde. J'ai également dû faire preuve d'esprit critique en évaluant la faisabilité de chaque nouvelle approche et en proposant des améliorations continues.

En fin de compte, le projet a permis de poser les bases d'un contrôleur Wi-Fi intelligent capable d'optimiser la configuration réseau en temps réel. Ce travail démontre non seulement ma capacité à mettre en œuvre une méthodologie de recherche et d'innovation (en passant par l'état de l'art, la modélisation, les tests et les ajustements), mais aussi à adapter cette démarche aux contraintes réelles et aux imprévus. J'ai su valoriser les résultats obtenus en les inscrivant dans une perspective de sobriété numérique, répondant

ainsi à des enjeux écologiques et sociétaux actuels.

En conclusion, cette preuve de compétence met en lumière ma capacité à analyser et formaliser une démarche d'innovation (N1) ainsi qu'à la réaliser concrètement (N2), en intégrant les incertitudes, en résolvant des problèmes complexes et en valorisant les résultats de façon adaptée aux enjeux de notre époque.

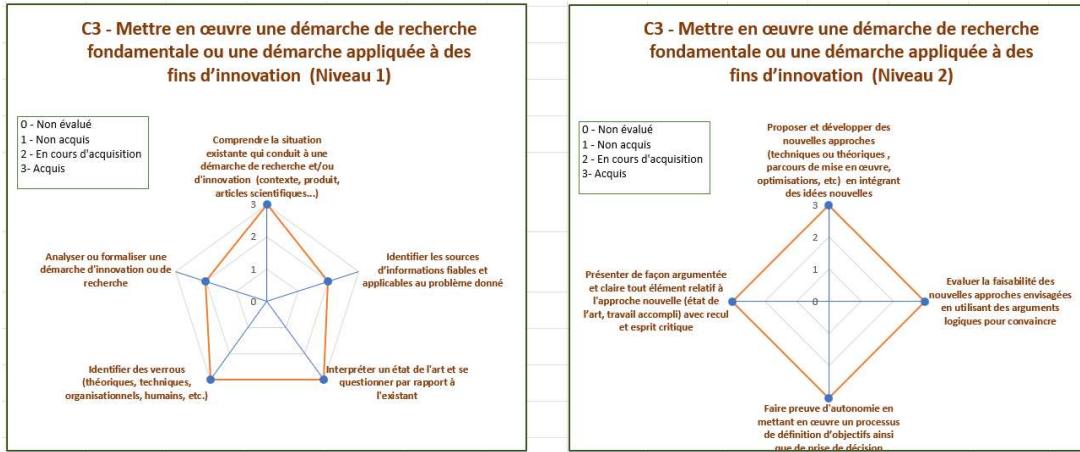


Figure 15: C3: Mettre en œuvre une démarche d'innovation et de recherche, Concevoir des solution d'ingenierie

Annex A.10 - Preuve de compétence C4: Travailler et coopérer dans une équipe

Dans le cadre du projet de Génie Logiciel (GL), j'ai eu l'opportunité de travailler pendant un mois et demi au sein d'une équipe pluri-écoles composée d'étudiants de Phelma et de l'Ensimag. L'objectif était ambitieux : développer un compilateur pour le langage Deca.

Dès le début, nous avons structuré le travail en plusieurs étapes : découpage du projet en sous-tâches, planification rigoureuse afin de respecter les délais imposés, puis répartition claire des rôles et des responsabilités. Chaque membre de l'équipe connaissait ainsi ses missions et les objectifs à atteindre.

Nous avons adopté une méthodologie agile, avec des sprints : chaque sous-partie terminée devait être validée collectivement. Nous produisions également des livrables réguliers pour suivre l'avancement. En parallèle de mes contributions au code, j'ai assumé le rôle de scrum master. Ce rôle impliquait de coordonner l'équipe, de suivre l'avancement, et de m'assurer du respect des délais fixés.

Au-delà des aspects techniques, j'ai également pris en charge la gestion humaine du projet. J'ai veillé à écouter les opinions de chacun, même si toutes les suggestions n'étaient pas nécessairement retenues, afin de maintenir un climat de travail respectueux et inclusif. J'ai aussi mis en place les outils de gestion de projet, réparti les rôles (coordination, tests, développement), et assuré le reporting régulier aux encadrants extérieurs de l'Ensimag, qu'ils soient spécialistes en informatique ou en gestion de projet.

Enfin, j'ai dû gérer plusieurs situations sensibles : la démotivation ponctuelle de certains membres, la nécessité de redistribuer des tâches pour avancer efficacement, ou encore la résolution de conflits internes afin de préserver une dynamique de groupe positive. J'ai également assuré la communication des résultats globaux auprès des décideurs, en présentant à la fois les aspects techniques et organisationnels.

Cette expérience m'a permis de démontrer ma capacité à structurer un projet complexe, à coordonner une équipe, et à le mener à bien dans un cadre collaboratif et professionnel.

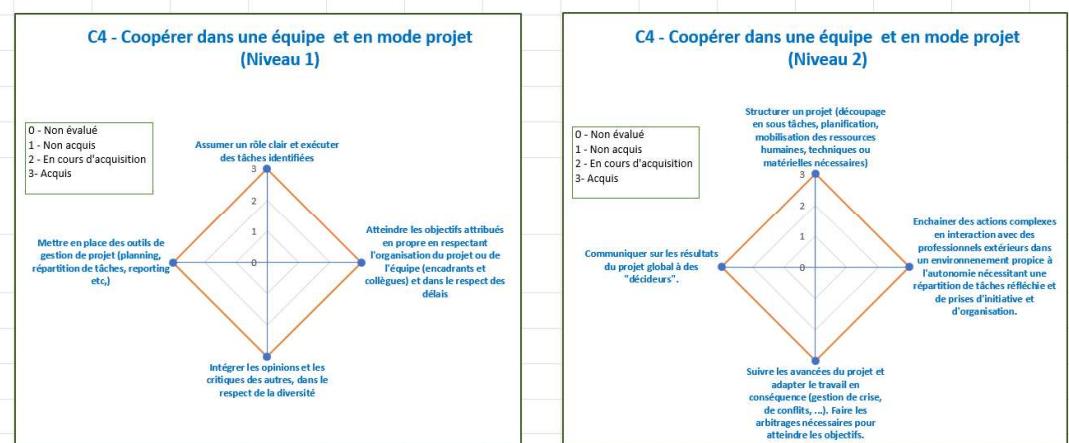


Figure 16: C4: Travailler et coopérer dans une équipe

Annex A.11 - Preuve de compétence C5: Tenir compte des transitions technologiques, environnementales et sociétales (RSE)

Au sein de mon expérience dans le Graduate School, j'ai développé une compétence essentielle : celle de prendre en compte l'ensemble du cycle de vie d'un produit technologique, en intégrant les dimensions environnementales, sociales et sociétales.

Dans ce cadre, nous avons acquis une vision très claire de tout le cycle de vie des produits. Cela commence par l'extraction des métaux, en considérant non seulement leur impact environnemental, mais aussi les conditions de travail des personnes qui y sont exposées. Nous avons également réfléchi aux impacts du transport des matériaux, qui contribue à l'empreinte carbone et à la pollution. Puis nous avons étudié la phase d'utilisation du produit, en évaluant son efficacité énergétique et ses impacts sur les utilisateurs, avant de réfléchir à la fin de vie du produit : recyclage, réduction des déchets, et comment minimiser les impacts sur la biodiversité et la santé.

Un exemple concret a été notre participation à un hackathon chez STMicroelectronics, où nous avons dû proposer des solutions concrètes pour rendre la microélectronique plus durable. Cela nous a permis de relier la théorie à la pratique, en tenant compte de toutes les parties prenantes et en adoptant une approche systémique.

En résumé, cette expérience m'a appris à analyser et à évaluer les transitions technologiques, environnementales et sociétales de façon holistique, et à proposer des solutions qui minimisent les impacts tout en étant socialement responsables.

Aussi, lors de mon stage ouvrier chez SICOAC, une entreprise tunisienne spécialisée dans la fabrication de tuyaux en polyéthylène (PE) et en PVC, j'ai pu observer et participer à une démarche intégrant à la fois les transitions environnementales, technologiques et sociétales. L'entreprise, certifiée ISO 9001 et ISO 14001, place la qualité et la durabilité au cœur de ses activités.

Ma mission a consisté à découvrir les pratiques mises en place pour réduire l'impact écologique tout en respectant les normes de qualité. J'ai notamment participé aux tests de conformité en laboratoire (traction, allongement, retrait à chaud, essais de pression), mais aussi au contrôle du taux de carbone noir dans les tuyaux en PE, essentiel pour garantir leur robustesse tout en limitant la consommation de ressources. Ces tests s'inscrivent dans une logique de conformité environnementale et de fiabilité produit.

J'ai également observé les actions concrètes de l'entreprise pour réduire son empreinte environnementale, telles que le recyclage des tuyaux défectueux, réintégrés dans la chaîne de production afin de limiter le gaspillage. En parallèle, la gestion de la sécurité et des conditions de travail des employés a été mise en avant : port des équipements, respect des consignes et suivi régulier des conditions de l'usine.

Cette expérience m'a permis de comprendre comment une entreprise industrielle intègre les principes de responsabilité sociétale (RSE) dans ses pratiques quotidiennes. Elle m'a aussi sensibilisée à l'importance de la coordination interdépartementale, puisque les décisions commerciales avaient un impact direct sur la production, les achats et la logistique. Le résultat principal est que j'ai pu analyser et participer à la mise en œuvre concrète de la RSE, en reliant la qualité des produits, la durabilité des processus et le bien-être des employés.

Ces deux expériences complémentaires – le Graduate School et mon stage ouvrier chez SICOAC – m'ont permis de développer une approche globale et intégrée des transitions

technologiques, environnementales et sociétales. J'ai appris à relier la réflexion théorique sur le cycle de vie et la durabilité des produits à des pratiques industrielles concrètes, tout en prenant en compte les enjeux humains, organisationnels et écologiques.

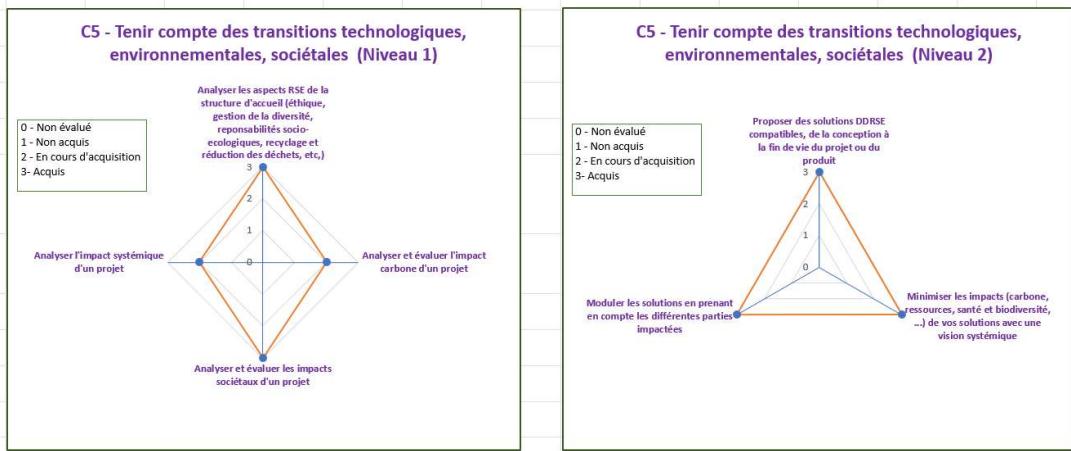


Figure 17: C5: Tenir compte des transitions technologiques, environnementales et sociétales (RSE)