Wi-Fi Watch: A Tool to Analyze WLAN Services

Abdeldjallil Benziane
UCB Lyon1, ENS Lyon, CNRS
LIP UMR 5668, Lyon, France

Anthony Busson
UCB Lyon1, ENS Lyon, CNRS
LIP UMR 5668, Lyon, France

Thomas Begin
UCB Lyon1, ENS Lyon, CNRS
LIP UMR 5668, Lyon, France

***Abstract*—Despite the widespread deployment of WLANs, there is a relative lack of open-source tools to continuously verify or conduct post-mortem analysis of whether the WLAN and its services remain available. We present Wi-Fi Watch, an open-source tool that enables WLAN administrators to easily deploy monitoring nodes. These nodes automatically collect and analyze performance metrics from the standpoint of a station. Wi-Fi Watch produces a user-friendly dashboard interface and logs outcomes.**

***Index Terms*—WLAN, performance, monitoring, troubleshooting, user experience.**

I. Introduction

Wireless Local Area Networks (WLANs) have become ubiquitous in our everyday lives. Based on the IEEE 802.11 standard, modern WLANs provide fast and seamless network connectivity to WLAN users, referred to as stations (STAs). WLANs' success is notably due to their relatively simple and cost-effective deployment. Access to WLANs is often commercialized (e.g., in hotels) while other WLANs support critical applications (e.g., connected tablets for university exams). In such settings, it is important for the WLAN administrator to be able to continuously verify, or conduct a post-mortem analysis, ensuring that the WLAN and its services (e.g., DNS, DHCP) remain available. As far as we know, only a few tools provide such services. Several software-based solutions can be installed on end-user devices such as smartphones or computers (e.g., Wi-Fi Analyzer [1]). However, these solutions are generally not open source and do not provide centralized data collection capabilities, limiting logging and post-processing analysis. Commercial proprietary solutions based on dedicated hardware probes and closer to our approach are also available (e.g., Aruba UXI [2]). In this demo, we present an open-source tool, called Wi-Fi Watch, that enables WLAN administrators (or a network controller) to easily deploy monitoring nodes to automatically collect and analyze performance metrics from these nodes. Wi-Fi Watch allows WLAN administrators to access and to verify the WLAN behavior from the standpoint of a STA.

II. Overview of Wi-Fi Watch

Network infrastructures in companies, train stations or university campuses typically include a wired network and a WLAN. The WLAN can be composed of one or multiple access points (APs) while the infrastructure generally provides multiple services such as DHCP, DNS, and Syslog.

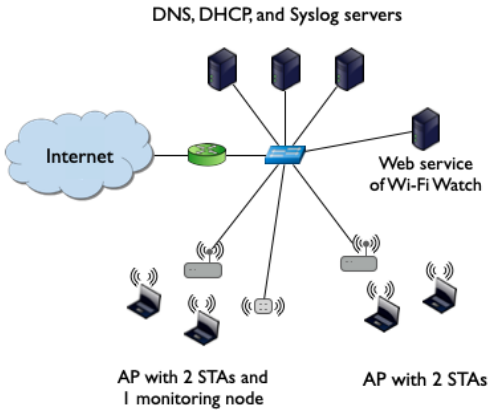


Fig. 1. Example of a network infrastructure supporting Wi-Fi Watch with one monitoring node and a server hosting the associated web service.

In this paper, we consider the WLAN of such a network infrastructure. We propose the Wi-Fi Watch tool, which uses a set of monitoring nodes – such as simple Raspberry Pis – and is designed to be deployed where the WLAN services are to be tested. Additionally, Wi-Fi Watch includes a web service to monitor the collected data, which can be deployed anywhere on the wired network of the infrastructure. Fig. 1 illustrates an example of a network infrastructure that supports the Wi-Fi Watch solution, with one monitoring node.

a) Performance metrics collection: The monitoring nodes run Linux, specifically a Debian-based distribution in our implementation. They must be equipped with two network interface cards: a Wi-Fi interface for measurements and testing, and an Ethernet interface for communicating with the web service. Monitoring nodes maintain a list of available WLANs for which the received signal strength is above a given threshold defined by the Wi-Fi Watch user. Based on this list and using the web service, the Wi-Fi Watch user then selects the SSID and (one of) its associated AP(s) using the AP's MAC address. The monitoring node can now start to passively collect measurements on the physical network layer such as the current radio channel busy time, the modulation and coding scheme (MCS) in use, and the Received Signal Strength Indicator (RSSI). Note that the busy time corresponds to the fraction of time during which the radio channel is sensed as occupied by the Wi-Fi interface. Wi-Fi Watch also derives an estimated residual capacity metric

This work was funded by the French National Research Agency (ANR) under the project ANR NEMIoT (ANR-23-CE25-0006).

TABLE I
COLLECTED METRICS WITH THEIR ASSOCIATED LINUX COMMANDS.

Metric	Linux tools
Busy time	`iw`
MCS	`iwconfig`
RSSI	`iwconfig`
Association time	`wpa_supplicant`
DHCP time	`dhclient`
Application time	`ping`, `wget`

calculated as $(1 - B) \times R_{\text{MCS}}$, where B denotes the busy time, ranging from 0 to 1, and R_{MCS} represents the link capacity determined by the selected MCS. In parallel to these passive measurements, the monitoring node actively tests the different steps involved in the WLAN connection process. This requires that credentials of a valid WLAN user account have been previously configured through the web service (using a web form or a plain text file with the passwords safely stored, e.g., encrypted). These steps include the AP-STA association delay as well as the time needed to obtain a response from the DHCP server (which provides the STA's IP address, its default gateway, and its DNS server). Lastly, monitoring nodes also assess Internet connectivity by sending ICMP echo requests to a given address specified through the web service or by measuring the download time of a given web page referenced by its URL. Note that all these tests are performed periodically at a configurable frequency. Fig. 3 illustrates the graphical interface of the web service used to configure the different parameters of the monitoring nodes. Fig. 4 shows a snapshot of the real-time visualization that the Wi-Fi Watch user can obtain from the tool. Performance metrics include channel utilization, link capacity in Mbps (derived from the MCS), latency, and residual capacity metric, displayed through several graphs. We report in Table I the list of performance metrics collected on the monitoring nodes along with their corresponding Linux tools used to obtain them.

b) Logging: All the tests and measurements described above (e.g., AP association/authentication, DHCP, DNS, ICMP echo) generate syslog messages that are sent to the Syslog server of the network infrastructure, the IP address of which must be configured beforehand. The logs include the timestamp, the AP being tested, and a severity level that depends on the test outcome, which can take up to 8 different values. In our case, we use three levels: (i) The informational level (level 6) to indicate that the operation completed successfully in less than 4 s; (ii) The warning level (level 4) to inform that the operation completed in between 4 s and 15 s; (iii) The error level (level 3) to notify that the operation has failed.

This collection of logs is crucial for our tool, as it enables WLAN administrators to retrospectively analyze performance over specific periods if connectivity outages or network issues have been reported. Note that logs can be processed by third-party tools to automatically generate alarms and notifications (e.g., Grafana, Elasticsearch). We show an excerpt of the logs generated by Wi-Fi Watch during one of our experiments in Fig. 5. We leave for a separate work the issue of dealing with the potentially large amount of logs.

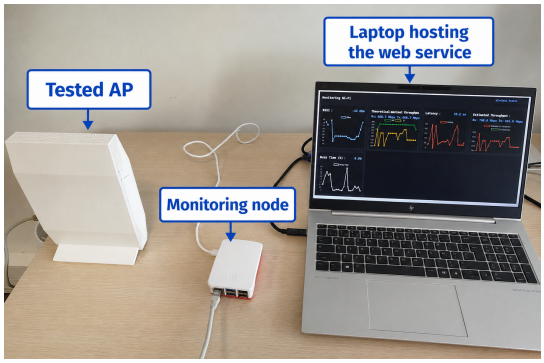


Fig. 2. Our testbed for Wi-Fi Watch.

III. IMPLEMENTATION DETAILS

a) Installation: Our solution can be downloaded from a Git repository [3]. The web service is containerized and deployed using Docker Compose and a Dockerfile. No software installation is required on the monitoring nodes. They only need to run a Debian-based operating system with an active SSH server and a user account with administrative privileges.

The user of Wi-Fi Watch is required to provide an Ansible inventory file containing the IP addresses and credentials of the monitoring nodes to be used. The deployment of the required tools and scripts on the monitoring nodes is then performed automatically through an Ansible playbook.

b) Web service: The web service is implemented in Python and relies on a Flask server. The system administrator interacts with the platform through a web interface and its associated functionalities. User actions are translated into requests to the backend API of the web service. For example, HTTP POST requests are used to submit the actions or measurements to be performed on the monitoring nodes, such as measuring the association time between the monitoring node and the tested AP. The web service subsequently executes an Ansible playbook that orchestrates the requested measurements and tests on the monitoring nodes. Fig. 2 depicts our experimental testbed with the main Wi-Fi Watch components: AP, monitoring node, and web service.

IV. PROPOSED DEMO AT MSWIM 2026

During the conference, we will present a live demonstration of our tool using one laptop to host the web service and several Raspberry Pi devices acting as monitoring nodes deployed within the conference venue. Participants will be able to access the web service in real time and monitor the performance of the conference WLAN. We will also provide attendees with direct access to the web interface from their own laptops and assist volunteers in deploying the web service on their laptop for hands-on experimentation.

REFERENCES

[1] "Wi-Fi Analyzer," https://www.wifianalyzer.info/.
[2] "HPE Aruba Networking UXI," https://www.hpe.com/us/en/networking/user-experience-insight.html.
[3] "Wi-Fi Watch," https://github.com/anthonybusson/Wi-Fi-Watch.

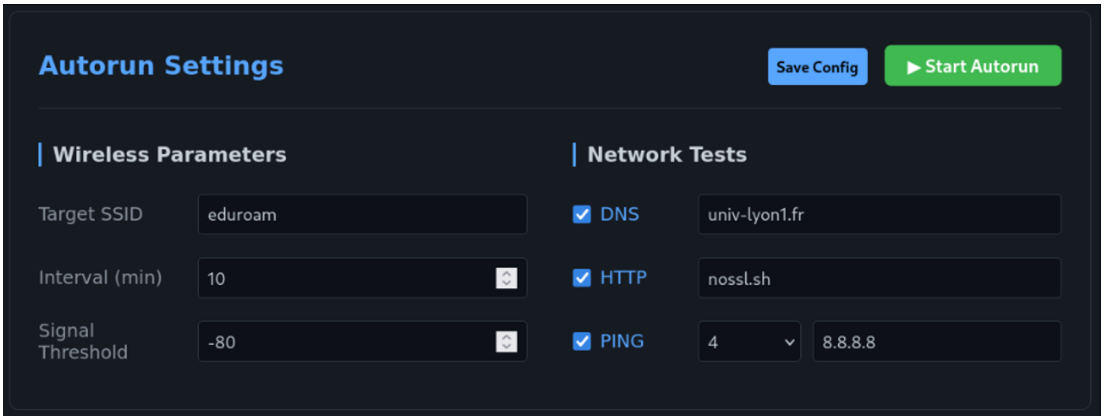


Fig. 3. Web page used to configure the Wi-Fi Watch parameters.



Fig. 4. Snapshot of the performance measurements displayed in real time on the web service.

```
LAPTOP-37PRQERO WifiMonitoring[12182]: [INFO] [ASAU] BSSID: 26:fd:a3:8b:d8:8b | Duration: 2.87 s – SUCCESS
LAPTOP-37PRQERO WifiMonitoring[12182]: [INFO] [DHCP] BSSID: 26:fd:a3:8b:d8:8b | Duration: 0.188 s – SUCCESS
LAPTOP-37PRQERO WifiMonitoring[12182]: [INFO] [WirelessInfo] BSSID: xxxx | Rx MCS: VHT-MCS 5 Mbps | Tx MCS: VHT-MCS 5 Mbps | Busy Time: 0.39 % | RSSI: -37 dBm
LAPTOP-37PRQERO WifiMonitoring[12182]: [INFO] [WirelessInfo] BSSID: xxxx | Rx MCS: VHT-MCS 6 Mbps | Tx MCS: VHT-MCS 6 Mbps | Busy Time: 0.00 % | RSSI: -37 dBm
....
LAPTOP-37PRQERO WifiMonitoring[12182]: [INFO] [DNS] BSSID: xxxx | Duration: 0.437 s – SUCCESS
LAPTOP-37PRQERO WifiMonitoring[12182]: [INFO] [ICMP] BSSID: xxxx | Aver. Latency: 104.024 ms | Min Latency: 26.666 ms | Max Latency: 140.172 ms | Pkt Received: 4/4 – SUCCESS
LAPTOP-37PRQERO WifiMonitoring[12182]: [INFO] [HTTP] BSSID: 26:fd:a3:8b:d8:8b | Duration: 0.188 s - SUCCESS
```

Fig. 5. Example of logs generated by a monitoring node.