CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A Passive Monitoring System Detecting Connected Devices

Bachelor's Thesis of Computer Science and Engineering

Filip Engbäck
Philip Lehmann Romøren
Filip Magnusson
Viktor Nambord
Simon Victor
Erik Åkerson

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Bachelor Thesis 2026

A Passive Monitoring System Detecting Connected Devices

Filip Engbäck
Philip Lehmann Romøren
Filip Magnusson
Viktor Nambord
Simon Victor
Erik Åkerson

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

A Passive Monitoring System Detecting Connected Devices
Filip Engbäck, Philip Lehmann Romøren, Filip Magnusson, Viktor Nambord, Simon Victor, Erik Åkerson

Supervisor: Atmane Ayoub Mansour Bahar, Department of Computer Science and Engineering
Examiner: Arne Linde, Department of Computer Science and Engineering
Graded by teacher: Magnus Almgren, Department of Computer Science and Engineering

Bachelor Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: AI-image showing a internet landscape. Image made with https://vdraw.ai/ai-image-editor

Typeset in LaTeX
Gothenburg, Sweden 2026

Abstract

With the increasing prevalence of daily internet usage, device identification and classification over wireless networks have become highly relevant topics. Most of the wireless network traffic is encrypted. Even though many users feel like that is enough to be secure, this project shows that there is still much information that can be extracted by a passive observer.

This project implemented a passive monitoring system capable of identifying devices and classifying their activity without being logged in on the network, only passive monitoring, which makes it almost impossible being detected.

The results showed that it was possible to infer certain device characteristics, including mobility patterns, vendor information, and user activity states such as streaming, browsing, or being idle.

These findings demonstrate that meaningful insights can be derived solely from metadata, despite the presence of encryption.

Acknowledgements

We would like to direct special thanks to our supervisor, Atmane Ayoub Mansour Bahar, who helped us along the way by guiding us and coming with excellent feedback. We would further like to thank Romaric Duvignau who provided this project and gave us the materials needed to conduct in controlled environments. Finally, we would like to thank Magnus Almgren, who took his time to read and mark all of our written assignments.

Filip Engbäck, Philip Lehmann Romøren, Filip Magnusson, Viktor Nambord, Simon Victor, Erik Åkerson, Gothenburg, May 2026

Contents

List of Figures

List of Tables

List of Acronyms

AP	Access Point
API	Application Programming Interface
BSSID	Basic Service Set Identifier
ESS	Extended Service Set
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
LA	Locally Administrated
MAC	Media Access Control
MAD	Median Absolute Deviation
NIC	Network Interface Card
OUI	Organizationally Unique Identifier
PCAP	Packet Capture
RSN	Robust Security Network
RSSI	Received Signal Strength Indicator
SSID	Service Set Identifier
STA	Station
TLS	Transport Layer Security
TSF	Timing Synchronization Function
TUI	Terminal User Interface
UUID	Universally Unique Identifier
VAP	Virtual Access Point
WLAN	Wireless Local Area Network
WPA2	Wi-Fi Protected Access 2
WPA3	Wi-Fi Protected Access 3

1

Introduction

This chapter introduces the background of *Wireless Local Area Networks* (WLANs) and the growing importance of security and privacy in increasingly connected environments, particularly with the expansion of *Internet of Things* (IoT) devices and public Wi-Fi usage. It explains how, despite modern encryption and privacy mechanisms, passive observation of wireless traffic can still reveal meaningful information through metadata and transmission patterns. Furthermore, it outlines the overall purpose and scope of this work, which focuses on designing a strictly passive monitoring system for analyzing Wi-Fi metadata and exploring what can be inferred without actively interacting with the network.

1.1 Background

WLANs are used all around the world in everyday lives to provide connectivity for a variety of devices such as phones, computers, TVs, and IoT devices. In recent years, the market for IoT devices has exploded, and in 2032 the expected number of IoT devices is projected to reach 37.48 billion worldwide [1]. The number of public Wi-Fi networks increases each year, and so does the number of intruders and information leaks [2]. As the number of connected devices and networks increases, several metrics can increase with it, such as electricity consumption, network latency, and number of users.

However, this increase of devices does also correlate to an increase of cyberattacks [3], as more connected devices mean more targets available, and a larger crowd for intruders to hide within. Some of the threats include Man-in-the-Middle attacks, data theft, malware injections, and malicious hotspots [2]. Furthermore, public Wi-Fi networks are sometimes non-encrypted, which makes it easier for attackers to spy and steal information. Yet, even when encryption protocols such as *Wi-Fi Protected Access 3* (WPA3) or *Transport Layer Security* (TLS) are used, a lot of information about a device can still be revealed [4]. As a result of these threats, it is increasingly important for users to understand the privacy and security risks associated with public Wi-Fi networks.

Information that can be observed includes metadata such as packet sizes, timing, protocol types and connection patterns which can reveal a lot about devices and their activities [4]. Devices connected to a network can also be identified by *Media Access Control* (MAC) addresses and *Internet Protocol* (IP) addresses since they are unique for a device on that network [5]. In addition, devices on a network can often be identified through a technique known as device fingerprinting. Device fingerprinting analyzes patterns in network behavior, such as protocol usage, packet timing, packet lengths, and traffic characteris-

tics, to infer the type of device generating the traffic [4]. This approach is particularly useful when traditional identifiers are unavailable or obscured, for example through MAC address randomization.

In order to observe this metadata, a method called packet sniffing is commonly used. Packet sniffing is a technique in which network packets are captured as they are transmitted across a network interface. Packet sniffing can often be captured using a *Network Interface Card* (NIC) in monitor mode, enabling passive observation of wireless traffic [6]. Wi-Fi networks can operate on several different frequencies, referred to as Wi-Fi channels, and devices may communicate on different channels at the same time. To achieve a more complete traffic capture, the monitoring interface periodically switches between channels, a process known as channel hopping, allowing traffic from across the network to be observed [7].

1.2 Purpose

The purpose of this project is to investigate how much information about connected devices and their activities can be inferred from strictly passively observed Wi-Fi metadata. By analyzing what characteristics can be derived from passive observation, the project aims to demonstrate that even encrypted wireless networks can reveal meaningful insight about users' devices and activity patterns.

1.3 Scope

This project focuses on the implementation and design of a passive wireless network monitoring system. The system operates by observing and analyzing *Institute of Electrical and Electronics Engineers* (IEEE) 802.11 protocol traffic using a device in monitor mode, together with network analysis tools. This allows it to capture frames and extract metadata such as protocol information, timing behavior, and other characteristics exposed by wireless communication.

The goal of the system is to be able to detect nearby access points and devices, keep track of their presence, analyze movement, and perform basic device fingerprinting to distinguish between devices based on observable device characteristics inferred from network traffic, including vendor identification. The system will also implement threshold based activity classification to estimate the type of activity a device is engaged in. The system will include an interactive visual interface to present the collected information in a user friendly form.

One of the most important rules for this project is that all monitoring is strictly passive. Passive monitoring only observes existing network traffic without transmitting additional packets [8]. In contrast, active monitoring generates packets of its own into the network, such as sending probe requests in order to discover connected devices [8]. This rule is crucial both to keep the technical scope feasible within academic level but also ethically responsible. This project is meant to be an exploratory study rather than a hacking or penetration testing tool.

All exploration and experiments part of this project will only be done on controlled networks and devices. The system only analyzes the metadata and protocol level information

that can be obtained without decrypting encrypted payload data.

Since the project is passively monitoring wireless traffic, privacy and ethical considerations are important. Although all experiments will only be conducted on controlled networks, similar techniques could be misused to monitor devices without user awareness. Therefore, the project emphasizes responsible handling of collected data and aims to increase awareness of privacy and security limitations in Wi-Fi networks. In addition, understanding connected devices and network activity may contribute to more efficient network and energy usage by identifying unnecessarily active devices.

1.4 Outline

To guide the reader through the remainder of this study, the thesis is structured as follows: Chapter 2 details the core theoretical foundations including MAC address randomization, management frames, and data frame traffic analysis. Chapter 3 outlines the technical method, documenting the work process, packet sniffing tools such as Scapy, and the specific evaluation criteria. Chapter 4 presents the detailed implementation of the pipeline, explaining how access points are grouped, hidden SSIDs associated, and how device fingerprinting, movement detection, and activity classification are executed. Chapter 5 compiles the experimental results of these distinct system components, followed by a critical analysis and evaluation of the system's performance in Chapter 6. Finally, Chapter 7 concludes the thesis by discussing system limitations, outlining paths for future work, and assessing ethical and privacy implications.

2

Theory

This chapter covers key theoretical aspects of Wi-Fi networks, including MAC address randomization and its limitations for device identification. It also explains how management frames and data traffic can be used for device fingerprinting and traffic analysis despite encryption and privacy mechanisms. Additionally, it introduces timing and synchronization concepts, such as clock skew and *Timing Synchronization Function* (TSF), as well as statistical filtering methods used for processing noisy wireless measurements.

2.1 MAC Randomization

Most modern devices use MAC randomization to hide the true identity of a device for privacy [9]. For each network a device is connected to, a new MAC address is generated and used for this specific network. The randomized MAC is then used for a certain amount of time until a new MAC is generated and used for the network. Different devices use different methods to randomize their MAC [9]. Furthermore, there are differences in how they calculate the randomized MAC address [9]. In order to determine if a MAC address is randomized, the *Locally Administered* (LA) bit can be observed, which is the second least significant bit in the first octet [10], [11]. If that bit is set, the MAC address has been modified from the device's factory-assigned address by the administrator. For example the following MAC address would likely be randomized:

$$A\mathbf{2} : B3 : C4 : D5 : E6 : F7$$

This is because in the first octet (A2) is the same as 1010 00**1**0 in binary. The LA bit is the highlighted, which is set to 1. Therefore, the MAC address is most likely randomized.

2.2 Management Frames

Management frames are used to give information about a device or an AP's capabilities [9]. The capabilities are stored as tags and give information such as which channels are supported, power capability and vendor specific information [9]. The management frames used for this project:

- **Beacon Frame**: On a network, APs show their presence on the network by periodically broadcasting beacons. A beacon's frame includes the *Service Set Identifier* (SSID) or network name, MAC address, channel, and other capabilities of the AP [12].

- **Probe Request**: Probe requests are sent from a device to detect nearby networks. Frames are sent in bursts over multiple channels to advertise the existence of the device to identify nearby networks [13], including its MAC address and sometimes also the SSID of a network. If an SSID is mentioned in the probe request, the mentioned AP responds to the request with a probe response.

  Modern devices use MAC randomization and each burst uses a different random MAC address, but in some cases it uses the same MAC address that was used to connect to the AP. The frequency and randomness of probe request, as well as the information extracted, differ for each device [14]. The bursts of probe requests are between 2-5 packets depending on the device, and last for around 20ms [14].

- **Association Request**: When a device has been authenticated on the network, the next step is for the client to associate with the AP [12]. An association request is sent from the device to the AP and then a response from the AP to the device, signaling either success or failure. The response from the AP is called a association response.

Association request and probe request frames expose a rich set of implementation specific features. Since these characteristics remain sufficiently stable across MAC address randomization events, they are well-suited for probabilistic device fingerprinting [15], [16]. The following features are extracted from association request and probe request frames as defined in the IEEE 802.11 standard and prior work on MAC layer fingerprinting [15], [16].

- **Information Element Ordering** The order in which information elements appear in the association request frame.
- **Supported Rates** The set of physical layer data rates supported by the device.
- **HT Capabilities** Describes the device's hardware capabilities.
- ***Robust Security Network* (RSN) Parameters** Specifies supported security mechanisms such as WPA2/WPA3.
- **Extended Capabilities** Indicates optional advanced protocol features.
- **Vendor-Specific Information Elements** Contain manufacturer-specific identifiers such as the *Organizationally Unique Identifier* (OUI).

These characteristics form the basis for device and access point fingerprinting. By analyzing relatively stable implementation-specific properties in management frames, it is possible to distinguish devices even when MAC addresses change or are randomized.

2.3 Traffic Analysis of Data Frames

While management frames can reveal information about a device's presence and capabilities, the device's actual network activity is carried through data frames [17]. These contain the traffic generated by applications, such as chat messages, web pages and streamed video. With modern encryption protocols, the contents of data frames are hidden, but metadata such as traffic rate, timing, direction and frame size remain visible to a passive observer [18], [19].

When the content of data frames is encrypted, information can still be inferred through traffic analysis, which studies observable metadata patterns rather than packet contents [18]. Since different activities often generate different traffic characteristics, metadata such as timing, direction and frame size can be used to infer what activity a device is engaged in [20], [21]. Wang et al. [22] compared traffic generated while using different applications on mobile devices and found that Chrome traffic contained moderately dense bursts of frames with varying sizes, while YouTube traffic contained denser bursts and a large share of similarly sized frames. By observing such metadata patterns over time, it is possible to estimate the underlying user activity [21].

2.4 Clock Skew and Timing Synchronization Function

Clock skew is used in this project as one of several fingerprinting indicators for determining whether two APs originate from the same hardware platform. In IEEE 802.11 networks, every AP broadcasts periodic timing information to synchronize connected clients. However, due to microscopic manufacturing imperfections in hardware components-specifically the quartz crystal oscillators that drive device clocks-different physical platforms tick at slightly different rates [23]. Because APs sharing the same physical hardware typically rely on the same internal oscillator, they exhibit nearly identical clock drift characteristics relative to a true time source. By measuring these minute deviations, we can infer physical co-location and hardware sharing, which serves as a critical feature for the hidden SSID pairing algorithm described later in Section 4.3.

Fundamentally, network synchronization aims to align local counting devices with a perfect time reference [23]. An ideal reference clock C_r progresses perfectly with true time t, denoted as $C_r(t) = t$, with a constant rate of change $dC_r(t)/dt = 1$. In practice, standard network nodes rely on economical quartz crystal oscillators, which inherently suffer from time varying inaccuracies [23]. Consequently, a local clock C_i operating with a relative frequency rate a_i (skew) and a time offset b_i is modelled mathematically as:

$$C_i(t) = a_i t + b_i \tag{2.1}$$

To achieve clock synchronization in a packet based network, timing information must be exchanged between two devices where timestamps, in form of C, from respective device are communicated between both devices. Clock synchronization is then performed by estimating and compensating for either the time offset, the frequency difference (clock skew), or both between devices [23].

In infrastructure based wireless networks, the AP serves as the timing reference for all associated *stations* (STAs) through a *Timing Synchronization Function* (TSF). The AP periodically transmits its current time within beacon or probe response frames, which include a timestamp representing the AP's clock value [23]. Upon receiving such a frame, a STA updates its local clock based on the received timestamp.

To estimate clock skew, two observations are taken from both the AP and the local STA clock using TSF timestamps. The system records two timestamp pairs: AP timestamps

$\text{TSF}_1, \text{TSF}_2$ and local device timestamps t_1, t_2. The corresponding time differences are defined as: $\Delta\text{TSF} = \text{TSF}_2 - \text{TSF}_1$ and $\Delta t = t_2 - t_1$.

The clock skew is then calculated as the relative progression error between the AP clock and the local clock. To express this skew in a standardized engineering metric, the deviation is converted into parts per million (ppm):

$$\text{skew}_{\text{ppm}} = \frac{\Delta\text{TSF} - \Delta t}{\Delta t} \times 10^6 \tag{2.2}$$

A skew_{ppm} value of 0 indicates perfect synchronization between the local clock and the reference AP clock. Deviations from zero represent clock drift, where positive or negative ppm values indicate that the AP's hardware oscillator is running faster or slower than the local reference, respectively. This distinct hardware signature is ultimately extracted and fed into the pairing framework detailed in Section 4.3.

2.5 Hidden SSIDs, Virtual Access Points, and BSSID Similarities

Hidden SSIDs cannot always be directly identified through passive observation. Therefore, the system relies on indirect indicators to infer whether a hidden AP belongs to a known *Extended Service Set* (ESS). Most APs include their SSID in beacon frames by default, broadcasting their identity to nearby devices. However, some APs omit this identifier, a mechanism known as a hidden SSID [24]. While this suppression prevents the SSID from appearing in passive scans, it can still be harvested through active scanning. When a client attempts to connect, it transmits a directed probe request; the AP then responds with a probe response frame revealing the SSID [17], [24]. Consequently, hidden SSIDs provide very limited security, a topic further explored in Section 4.3, and serve primarily to provide a layer of obscurity for casual users.

An AP can create virtual interfaces, allowing a single physical wireless interface to operate as multiple APs, and this concept is referred to as a *Virtual Access Point* (VAP). In other words, one physical wireless interface can host multiple APs, each with its own SSID, IP address, and MAC address [25]. Because SSID suppression is configured at the logical beacon level, a VAP can be independently set as a hidden SSID. While a hidden SSID is not inherently a VAP, the two are frequently linked in real-world deployments where a single device hosts both a public and a hidden administrative network [25], [24].

To manage these multiple logical interfaces, vendors often configure APs on the same device to have similar *Basic Service Set Identifiers* (BSSIDs). A common convention among manufacturers is to increment only the last octet of the BSSID, particularly when the Locally Administered (LA) bit is set [11]. Because this practice is based on industry convention rather than strict regulatory coding, its consistency depends entirely on the manufacturer's implementation [11].

These properties serve as features for the hidden SSID pairing algorithm. Because a hidden AP hides its identity in beacons, its ESS membership is initially unknown. However, by identifying structural similarities between the BSSID of a hidden AP and those of nearby broadcasting VAPs or APs, the system can probabilistically infer that they reside

on the same physical hardware. This relationship allows the algorithm to link a hidden network to a known ESS by proxy, providing a method for identification that does not rely solely on the arrival of an active probe response.

2.6 Hampel Filters

Hampel filter is a statistical tool to replace or remove isolated outliers in order to obtain a more accurate data window [26]. The filter depends on *Median Absolute Deviation* (MAD) and a chosen window length to identify outliers. To apply the filter to the chosen window several thresholds must first be defined.

The first threshold, the window size, refers to the number of elements in the moving window. Determining the suitable window size is mainly based on two factors; the amount of outliers and the sensitivity [26]. Larger windows are better to reduce the sensitivity and are more effective with fewer outliers, while smaller windows are better for frequent outliers but more sensitivity prone due to false positives [26].

Secondly, after the MAD is calculated by comparing every element by the median a second threshold (n_σ) is chosen, usually between 3 and 3.5 [26].

$$\text{MAD} = \text{median}|x_i - \text{median}| \tag{2.3}$$

$$\sigma = 1.4826 \cdot \text{MAD} \tag{2.4}$$

1.4826 is the scaling factor which comes from normal distribution, and is the relationship between the MAD and standard deviation [26], [27].

The last threshold is not directly chosen, but based on the previous two. The final threshold is calculated by multiplying the chosen n_σ of 3-3.5 and the calculated σ. If a sample from the window deviates (is larger than) from the threshold the Hampel filter regard x_i as an outlier and will replace it with the median [26].

$$|x_i - \text{median}| > n_\sigma \cdot \sigma \tag{2.5}$$

After this threshold, the window is filtered from deviating outliers and can be used further for movement detection.

3

Method

The overall work process and technical setup of the project are described, including how the system was designed, implemented, and divided into modular components for passive Wi-Fi traffic analysis. The method also introduces the tools used in the implementation and explains their roles in enabling packet capture, processing, and user interface development. The evaluation methods are outlined, detailing how each system component was tested in controlled environments to assess performance, accuracy, and limitations under realistic but controlled network conditions.

3.1 Work Process

The project began with a study of relevant theory and previous work to identify what information could be extracted or inferred by passively observing Wi-Fi traffic. The project was then divided into components to be developed in parallel before being merged into the final system. These components were AP mapping, movement detection, vendor classification, device re-identification and activity classification. The implementations of these components are described in Chapter 4.

A system was first developed allowing automatic switching to monitor mode, finding networks through channel hopping, filtering by a network, and capturing the traffic on the selected network. The development of each component branched off from this system. The components were then developed and tested individually to verify the behavior before they were integrated into the complete system.

3.2 Tools

This section presents the software tools used in the project to enable scanning, parsing, and visualization of wireless network data. It describes Python, Textual, Scapy and Airmon-ng, all of which together support the implementation of a passive Wi-Fi monitoring system.

3.2.1 Python

The programming language used for this project was Python[1]. The selection of Python is motivated by its extensive ecosystem of libraries. In particular, Python provides robust

[1] https://www.python.org

support for tasks such as data analysis and, as discussed in Section 3.2.3, packet manipulation [28]. The strong capabilities for scripting and integration of existing components, makes Python a suitable choice for developing applications aimed at demonstrating and testing conceptual theories.

3.2.2 Textual

Textual[2] is an open-source Python framework for building user interfaces that run in the terminal or a web browser. It enables developers to create sophisticated, interactive applications using a simple Python *Application Programming Interface* (API) [29], without needing traditional web technologies like HTML or JavaScript. The framework works by combining Python with concepts inspired by modern web development [30], such as component-based design, layouts, and styling systems similar to CSS. It provides prebuilt widgets (e.g., buttons, tables, and input fields) that can be arranged and updated dynamically to form a full user interface.

Internally, Textual is asynchronous [30], meaning it can efficiently handle events, user input, and background tasks without blocking the application. This event-driven architecture allows applications to remain responsive while performing multiple operations at the same time. Textual applications run primarily in the terminal but they can also be served as web applications without major code changes [30]. This cross-platform capability makes it suitable for tools that need to run locally, over SSH, or in a browser.

3.2.3 Scapy

Scapy[3] is a Python-based packet manipulation library used for network analysis, security testing, and protocol development [31]. It allows users to create, send, receive, and analyze network packets across many protocols [32], giving full control over packet structure and content. The tool works by interacting directly with network interfaces, often using raw sockets or low-level access to bypass parts of the operating system's networking stack [33]. This enables users to craft custom packets, sniff live network traffic, and automate tasks such as port scanning or traceroute.

Scapy follows an interactive and scriptable approach, where packets are defined in Python code [32], sent over the network, and matched with responses for analysis. It can decode captured packets and display useful information such as protocols, IP addresses, and payload data.

3.2.4 Airmon-ng

The tool used to enable monitor mode on a wireless interface and to disable the network manager is Airmon-ng[4]. This tool allows users to easily create and switch between monitor mode and managed mode wireless interfaces. It also provides functionality to identify and terminate processes that may interfere with the NIC when it is placed in monitor mode, such as the network manager [34]. Due to the convenience of switching between

[2]https://textual.textualize.io
[3]https://scapy.net
[4]https://www.aircrack-ng.org/doku.php?id=airmon-ng

different modes, this tool was selected instead of manually creating and configuring the wireless interfaces.

3.3 Evaluation Method

Evaluating the system was performed to ensure that the different components of the system functioned as intended. The NIC used across the evaluation methods was the ALFA Networks AWUS036AXML. The dwell time per channel was approximately 0.3 seconds. All testing was conducted in a controlled environment rather than in a real world deployment. When a test deviates from any of these specifications, it will be mentioned in the respective subsection. The methods used to evaluate each component are described below.

3.3.1 Controlled Environment Evaluation of AP Grouping and Pairing

The grouping and pairing components of the AP system were evaluated in a controlled environment, meaning that no real world deployment testing was conducted. The controlled environment consisted of two Wi-Fi routers with a total of four APs, where one of the APs was divided into two VAPs. The systems dwell time per channel was approximately one second and it had ten passes per channel during testing. The purpose of this setup was to identify edge cases that could occur in real world scenarios. The setup was designed to reproduce representative edge cases that are expected to occur in larger deployments, rather than to simulate the scale of a full production environment.

The evaluation method used to assess the system's ability to correctly group APs into their respective networks involved testing various SSID and security configurations across different APs and observing how the component grouped them. The evaluation of the hidden SSID pairing consisted of, one Wi-Fi router always had at least one AP broadcasting a visible SSID, while the second router had either all APs configured with hidden SSIDs or only a subset of them hidden. In all cases, both routers belonged to the same ESS.

The success of the hidden SSID pairing component was measured by the number of APs that were correctly paired with the appropriate ESS. The two Wi-Fi routers and their respective APs exhibited different characteristics, such as variations in clock skew and BSSID naming conventions. This allowed the evaluation to assess how effectively the system could pair APs when routers and their internal components and APs differed significantly from one another.

A limitation of this evaluation is that all testing was conducted in a controlled environment rather than in a real world deployment. As a result, environmental factors such as network congestion, interference from neighboring devices, and large scale AP deployments were not included in the evaluation and may affect system performance in practical applications.

3.3.2 Vendor Classification Evaluation

To evaluate the vendor classification component, testing was done with a device connected to an AP with 2 channels. The device was simulating real world activity to see if the system was able to identify the vendor of the devices. The component was also evaluated on its capabilities to extract the correct information and connect it to the correct device when MAC randomization was in use.

When simulating real world activity the device was switching between streaming, web browsing, looking at social media and being idle. Testing of identifying the vendor was done until a vendor was successfully identified, and an average of how long it took to give the result was recorded. Each test was done using a different combination of the activities mentioned above.

For testing the ability of the component to connect the correct information to the device, packets were captured for 150 seconds. The system was then evaluated with a percentage on the amount of information the system is able to connect to the device.

3.3.3 Movement Detection Evaluation

In order to evaluate the movement detection part of the system, testing in a controlled environment was done. The controlled environment included one AP on a single channel and three devices where two of them were control devices in stationary mode and one was actively moving.

The test was conducted over a 60 second window where the moving device was in constant motion while the two control devices was still the whole time. *Received Signal Strength Indicator* (RSSI) value, timestamp and MAC address were continuously recorded during the 60 second window. This setup was simulating real world scenario where computers often remain stationary while phones are more portable and in motion. The test was conducted to test the accuracy of the component and find the right window size and thresholds.

After the test, the component was evaluated for each window size from 10-130 and threshold from 0.5-4.0 to find the accuracy of each combination. Since smaller window sizes are faster to update the status in real time, a window size lower than 50 with a high accuracy should be prioritized.

3.3.4 Activity Classification Evaluation

Previous work has demonstrated that machine learning can be used to classify activities from traffic characteristics [19], [21]. In contrast, this project evaluates whether a simpler threshold-based approach can be sufficient for distinguishing between a limited set of activities in a passive monitoring system intended for real-time use.

The activity-classification component was evaluated by testing whether labeled traffic captures could be correctly classified as idle, browsing, or streaming. These three classes were chosen to represent different patterns of network traffic. The idle class represents a device that is connected to the network with the screen turned on, but where the user is not actively using any application. The browsing class represents interactive web use,

where the user navigates between pages and reads content, with autoplaying advertisements and videos paused when possible. The streaming class represents continuous video playback.

Each evaluation capture lasted five minutes. For the browsing and streaming classes, three different traffic sources were used to increase variation in the data. The evaluation dataset is summarized in Table 3.1.

Table 3.1: Number of evaluation captures collected for each activity class and traffic source, separated by device.

Activity class	**Traffic source**	**Android phone**	**Windows laptop**	**Total**
Idle	None	6	6	12
Browsing	Wikipedia	2	2	4
	Aftonbladet	2	2	4
	Amazon	2	2	4
Streaming	YouTube	2	2	4
	Twitch	2	2	4
	Disney+	2	2	4
Total		**18**	**18**	**36**

When this evaluation was performed, the ALFA NIC was not available. Instead, a Plexgear NIC, article number 61347, was used. Due to limitations of this NIC, the evaluation was performed on a 2.4 GHz network that did not use Wi-Fi 6, operating on a single channel. During the captures, both the monitoring device and the monitored device were positioned close to the router to keep the wireless signal conditions stable.

The classifier produces one activity prediction per 10-second traffic window based on the observed characteristics of data frames associated with the device. The implementation details of the classifier, including the selected thresholds and temporal smoothing rules, are described in Section 4.7. For the evaluation, the same decision logic as in the real-time implementation was applied to recorded traffic captures. The evaluation was therefore performed offline, but the classifier still processed the data sequentially, one time window at a time. It had no access to future time windows beyond the one currently being processed.

The classifier was evaluated at the window level by comparing the predicted label for each 10-second window with the known activity label of the corresponding capture. Since each capture lasted five minutes, each capture produced 30 traffic windows. The first window of each capture was excluded from the metric calculations, since the temporal smoothing logic cannot make a non-idle prediction without at least two windows. This resulted in 29 evaluated time windows per capture and 1044 evaluated time windows in total.

The evaluation was summarized using a confusion matrix and the metrics precision, recall, and F1-score. Precision measures how many windows classified as a given class actually belong to that class, while recall measures how many windows of a given class were

correctly identified. F1-score combines precision and recall into a single metric. For each metric, a macro score was calculated by averaging the score across all classes, indicating the classifier's overall performance across the three activity classes.

3.3.5 Device Re-Identification Evaluation

To evaluate the device re-identification component under MAC address randomization, multiple tests were conducted simulating device connection and reconnection scenarios in a controlled environment.

The first evaluation step focused on verifying that the system produces high similarity scores for MAC addresses originating from the same physical device. For each test, a device was connected and then reconnected to the same network, resulting in different randomized MAC addresses. The corresponding MAC pairs were identified and stored in *Packet Capture* (PCAP) files to establish ground truth for which MAC addresses belong to that device. Fingerprints were then extracted and compared to compute similarity scores.

The second evaluation step was whether the system can distinguish between different physical devices. This was done by comparing fingerprint similarities between MAC addresses belonging to different devices. The expectation was that similarity scores for different devices remain significantly lower than for same-device comparisons.

To further validate the robustness of the approach, an additional device was included in the evaluation. This enabled comparison across three independent devices, strengthening the reliability of the observed similarity patterns.

Based on the test results, a decision threshold for similarity scores was chosen. The threshold was chosen such that all device comparisons between the same devices resulted in similarity scores above the threshold, while all device comparisons between different devices remained below it. This threshold is later used to classify whether two MAC addresses comes from the same physical device.

4

Implementation

The system is implemented as a passive Wi-Fi monitoring pipeline based on IEEE 802.11 frame capture in monitor mode. It processes captured frames through sequential stages for parsing, aggregation, and real-time visualization of detected networks and devices. It also describes how access points are grouped, hidden SSIDs are associated, and how device fingerprinting, vendor classification, movement detection, and activity classification are performed.

4.1 System Flow

This project was implemented as a terminal-based Wi-Fi monitoring system using passive IEEE 802.11 frame capture in monitor mode. The overall methodology is structured as a packet processing pipeline consisting of four sequential stages; capture, parsing, aggregation and presentation, as illustrated in Figure 4.1.

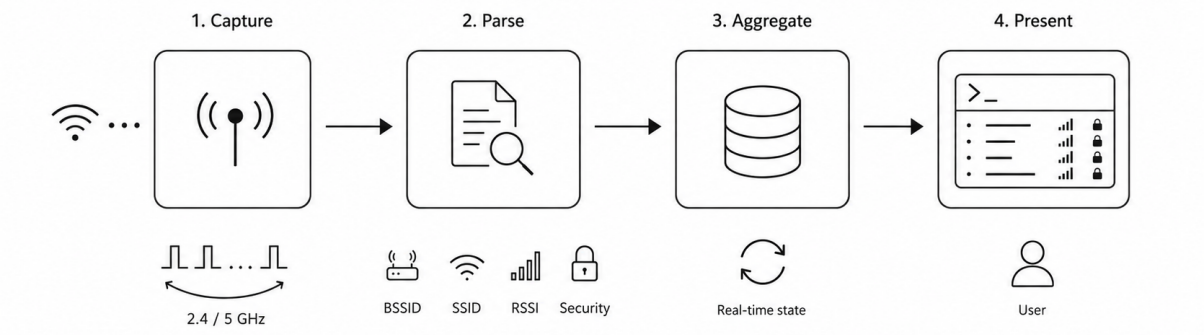


Figure 4.1: Overview of the packet processing pipeline, illustrating passive frame capture, packet parsing, real-time aggregation and user interface presentation.

Initially, the wireless network interface is set to monitor mode using Airmon-ng. When the user starts to scan for nearby networks a built in script is running to hop across a predefined set of channels within both the 2.4 GHz and 5 GHz frequency bands. During this channel hopping process, raw wireless frames are continuously captured.

Subsequently, each captured frame is parsed in order to extract the relevant attributes, including the BSSID, SSID, channel information, RSSI and security related parameters. The extracted data is transformed into structured observations, which are maintained within in memory state representations that are continuously updated in real-time. At

intervals, snapshots of this state are generated and transmitted to the user interface, where the current scanning status and detected network entities are visualized.

For the purpose of AP discovery, the system selectively processes beacon and probe response management frames, as these frame types contain essential information regarding AP identity and configuration. Detected APs are grouped according to their SSID and security characteristics, while individual BSSIDs are tracked separately with respect to their most recent channel, RSSI and vendor specific information. This design enables the representation of a single logical network composed of multiple physical AP radios and facilitates the analysis of channel distribution as well as the identification of the strongest observed signal within each group.

In the context of monitoring devices associated with a selected network, the user specifies a target network profile that includes known AP BSSIDs and their corresponding channels. The packet processing pipeline is then constrained to frames referencing these BSSIDs, thereby reducing the inclusion of unrelated traffic. From each relevant frame, device MAC addresses and associated metadata, such as RSSI, frame count and frame size are extracted and consolidated into per device records. These records are incrementally updated over time to support the estimation of activity patterns, movement tendencies inferred form RSSI variations and probable vendor identification. The monitoring interface is refreshed at fixed intervals, providing a continuous real-time overview of device associated with or operating in proximity to the selected network.

4.2 Grouping APs

Before monitoring devices within a wireless environment, the system must accurately determine which APs belong to the same ESS. Incorrect grouping leads to an incomplete or fragmented view of the network topology, causing errors in device tracking and roaming analysis. This section establishes the design rationale for a multi-parameter grouping heuristic and outlines the normalization process required to execute it.

4.2.1 Design Rationale and Alternative Approaches

The primary identifier advertised by an AP is the Service Set Identifier (SSID) contained in information element ID 0. However, relying solely on the SSID is insufficient for reliable network mapping. The SSID is a logical identifier rather than a globally unique one [17]. Different unrelated networks can use identical default SSIDs.

To overcome these limitations, a heuristic approach combining the SSID with security configuration metadata is selected. For a client to seamlessly roam and connect within an ESS, it requires a consistent security profile across all participating APs [35]. Inconsistencies in security parameters introduce severe operational and security risks, meaning an ESS must maintain uniformity in its deployment. Therefore, if two APs share an SSID but diverge in their security configurations, they almost certainly belong to different administrative domains. Combining these two independent layers creates a logical chain for ESS boundary discovery.

4.2.2 Extraction and Normalization of Network Parameters

To execute the grouping heuristic, the system must parse and compare security metadata across hardware. To prevent syntactic variations from breaking the logical grouping rules, raw data must be normalized into a standardized taxonomy before any grouping logic is applied.

The system first extracts the SSID from beacon frames, probe responses, and association responses. It then extracts security-related parameters primarily from information element ID 48 (the Robust Security Network (RSN) element). These values are then mapped to human readable labels using a predefined mapping.

After extraction, the raw security data is normalized to ensure consistency across different vendors and implementations. The resulting normalized security object contains: a security classification, a list of encryption algorithms, a list of authentication methods, and an operational mode. The normalization process begins by standardizing cipher and authentication fields by splitting composite values, converting all values to uppercase, and removing duplicates. Authentication values are further normalized to ensure that equivalent methods are treated consistently through semantic grouping. In particular, labels such as `WPA2-PSK` and `PSK` are mapped to `PSK`, `WPA3-SAE` and `SAE` are mapped to `SAE`, and `WPA2-Enterprise` and `EAP` are mapped to `EAP`.

4.2.3 Security Classification and Grouping Criteria

Following this normalization, a heuristic classification approach is applied, in which rule-based logic is used to determine the security classification. The security mode is derived from the normalized authentication field. Specifically, if the authentication field contains `EAP`, the mode is classified as `enterprise`, whereas if it contains `PSK` or `SAE`, the mode is classified as `personal`.

Thus, each access point is assigned a final security representation defined as a deterministic tuple:

$$\text{security} = (\text{type}, \text{encryption}, \text{auth}, \text{mode}) \tag{4.1}$$

Now let A and B be two access points. The condition for grouping access points into the same network is defined as follows:

$$\Big(\text{security}(A) = \text{security}(B)\Big) \wedge \Big(\text{ssid}(A) = \text{ssid}(B)\Big) \tag{4.2}$$

If this condition holds, then A and B are considered to belong to the same network; otherwise, they are not. In Figure 4.2 there is an example of how three APs are grouped by the algorithm to two different networks.

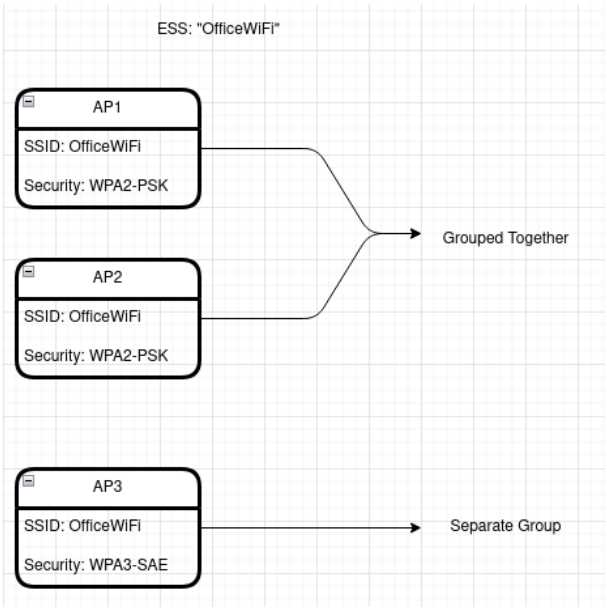


Figure 4.2: Example of AP grouping based on SSID and security configuration. APs with matching SSID and normalized security parameters are grouped into the same ESS.

4.3 Pairing Hidden SSID

After grouping the visible APs, the next step is the association of hidden SSIDs. The first step is to determine whether the system has already captured the hidden SSID from a probe response frame. Section 2.5 describes how this capture process works. If this is not the case, the pairing procedure is executed. This pairing process does not "crack" or directly reveal a hidden SSID. Instead, it infers which visible AP a hidden SSID belongs to by comparing hardware and signal characteristics. The features used for associating a hidden SSID with visible APs include RSSI, clock skew, BSSID similarities, OUI similarities, and *Universally Unique Identifier* (UUID).

The underlying concept is that a hidden SSID will exhibit relationships with all visible APs, and this relationship is assigned a similarity score between 0 and 1. Due to the chosen feature set, a hidden SSID may obtain high matching scores with multiple APs that belong to the same ESS. Because of this characteristic, a dynamic threshold is not used; instead, a fixed threshold is applied. The threshold is empirically chosen. This approach reduces the probability of false negatives, as it is possible that all candidate APs actually belong to the same network and therefore share very similar feature values. The final pairing is determined by selecting the highest-scoring visible AP above a fixed confidence threshold.

4.3.1 Hardware and Signal Based Similarity Metrics

A hidden SSID's BSSID is expected to exhibit a high degree of similarity with the other APs to which it belongs. In practice, APs often have very close or sequential MAC addresses. By employing the $S_{\text{similarity}}$ metric, it is possible to obtain a reliable indication of whether they are related.

A hidden SSID's MAC address ($b^{(\text{hidden})}$) is evaluated against a visible AP's MAC ad-

dress ($b^{(\mathrm{AP})}$) using a fine-grained, byte-wise comparison metric, $S_{\text{similarity}}$. This approach captures structural allocation patterns that standard Euclidean MAC distance metrics obscure. Each MAC address is represented as a 6-byte vector, MAC $= (b_1, b_2, b_3, b_4, b_5, b_6)$. The metric applies higher weights (w_i) to the final bytes, which dictate unique device identity, while lower weights are assigned to the vendor-specific prefixes. The differences are normalized linearly, with byte deltas greater than 32 being entirely disregarded to mitigate noise from completely unrelated devices:

$$S_{\text{similarity}} = \sum_{i=1}^{6} w_i \cdot \max\left(0,\ 1 - \frac{\left|b_i^{(\text{hidden})} - b_i^{(\text{AP})}\right|}{32}\right) \tag{4.3}$$

For computing the similarity scores for clock skew and signal strength, a similar approach is used, as the underlying reasoning is closely related. A hidden SSID and its corresponding VAP are expected to exhibit similar clock skew and signal strength values, since they originate from the same radio hardware. Consequently, the scoring for these two parameters is derived by computing the absolute difference between the values and applying an exponential decay function as normalization, where the parameter λ is determined empirically based on observed hardware variances:

$$S_{\text{skew}} = e^{-\lambda|\text{skew}_h - \text{skew}_{\text{ap}}|}, \qquad S_{\text{rssi}} = e^{-\lambda|\text{rssi}_h - \text{rssi}_{\text{ap}}|} \tag{4.4}$$

The final feature considered is the similarity of vendor specific information elements, in particular the OUI. Two access points are expected to share identical or at least highly similar vendor OUIs if they originate from the same device family or infrastructure. By comparing the sets of OUIs using the Jaccard index, a similarity score in the range $[0, 1]$ is obtained:

$$S_{\text{OUI}} = \frac{|O_h \cap O_v|}{|O_h \cup O_v|} \tag{4.5}$$

4.3.2 Score Fusion and Final Association Procedure

To unify these disparate metrics into a singular decision, a multiplicative fusion model is utilized. A multiplicative architecture was explicitly selected over an additive model to enforce strict feature consistency: if a candidate AP exhibits highly divergent clock skew or RSSI, the overall confidence must drop significantly, regardless of how closely their MAC addresses match.

However, to prevent missing features (e.g., an uncollected RSSI sample or an omitted OUI element) from completely destroying an otherwise strong match, each individual feature score S_i is mapped to a compressed range of $[0.5, 1.0]$. The baseline value of 0.5 acts as a neutral mathematical floor, ensuring that an unverified feature penalizes the product uniformly without zeroing out the entire calculation. The total fused score is defined as:

$$\text{Score} = \prod_i (0.5 + 0.5 \cdot S_i) \tag{4.6}$$

The UUID acts as an immediate shortcut; if a matching UUID is detected, the heuristic scoring engine is bypassed entirely, and the match is accepted. To ensure data integrity and remove transient noise, the system restricts comparisons to visible APs that have contributed a minimum of three captured management frames. Ultimately, the hidden SSID is grouped into the ESS of the visible AP that achieves the highest fused score above the fixed threshold.

4.4 Movement Detection

RSSI, as described earlier is the power level of a radio signal. Simply by measuring the radio strength, the system can detect if a device is in motion. However, RSSI strength is not only based on distance, but also disturbance. As a result, temporary outliers may occur which will affect the outcome. An outlier is a data point the deviates from the expected pattern. In order to prevent these deviations to ruin the result, Hampel filter is applied.

Hampel filter uses a window of RSSI values to calculate a median of the values in that window. After that a *Median Absolute Deviation* (MAD) is calculated where all values in the window is compared to the median. The formula looks like this:

$$\text{MAD} = \text{median}|x_i - \text{median}| \tag{4.7}$$

Furthermore, a new median is calculated based on the compared results, so a window with the values [-55, -60, -62, -66, -67] has the median of -62. The MAD will be the median of the values in the window compared to the median -62 which will give the new values [7, 2, 0, 4, 5]. The median of that new window is four. That new MAD is multiplied by 1.4826 which is the scaling factor from normal distribution. The formula looks like this:

$$\sigma = 1.4826 \cdot \text{MAD} \tag{4.8}$$

This result multiplied by three, which is the standard n_σ for "three sigma rule" becomes the threshold for detecting outliers [26]. If a value is greater than this threshold, it becomes replaced by the median value to ensure the device is only shown as moving when it is in motion and not misleading with single outliers [27].

In order to check if x_i is an outlier is if the following statement is true or not:

$$|x_i - \text{median}| > n_\sigma \cdot \sigma \tag{4.9}$$

Where n_σ is three as a standard. So when an outlier is greater than the threshold it is replaced by the median to create the filtered window. For the filtered window the MAD is once again calculated. It is multiplied by the scaling factor from normal distribution to create the standard deviation (σ) and if it is greater than the threshold of three it is marked as moving.

$$stdev > threshold \tag{4.10}$$

Stdev is the standard deviation (σ) calculated before as MAD $\cdot$ 1.4826, but this time from the filtered window. If this deviation is greater than the threshold it will be marked as moving.

The system requires at least ten RSSI values in order to create a filtered window and therefore be able to decide a device movement status. However, in order to create more accurate detection, a window of 40 samples will be used at a time. This is because more samples create a more stable median to compare the outliers against. The threshold chosen was three which usually is the standard for Gaussian-like noise [26]. However, it can range from two to five depending on how big the outliers are. Nevertheless, smaller n_σ may remove valid data (false positives) and higher n_σ may miss real outliers (false negatives) [26], [27].

4.5 Vendor Classification

Classifying the vendor of the MAC when MAC randomization is not in use, is done by checking the MAC in a lookup table, where each vendor has a specific MAC addresses they are allowed to use.

For devices where MAC randomization is used, more have to be done to connect the device to a vendor. Our system does this with the use of the management frames that a device sends. These frames are not encrypted and give information about the device. For our system, the management frames that are looked at are association requests and probe requests. In the management frames there is one particular interesting tag called the vendor specific tag. This tag shows the MAC of the network card's vendor, which is used to classify the vendor of the device.

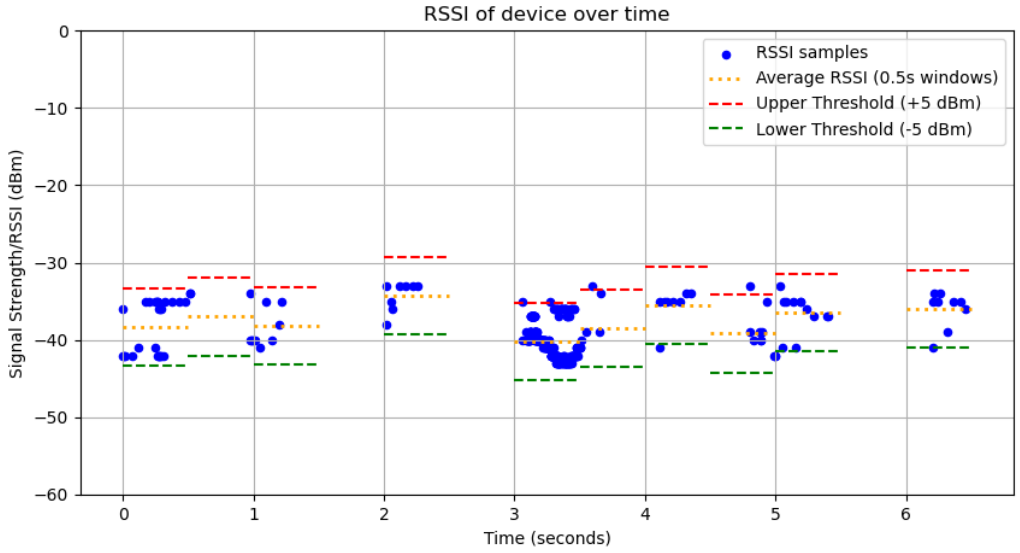
4.5.1 Association Request

Association requests are sent every time a device is connecting or trying to connect to an AP and use the same MAC address as the device uses for all other traffic on the AP. To classify the vendor, our system therefore captures the association requests and looks for the vendor specific tag in the management information to extract the vendor.

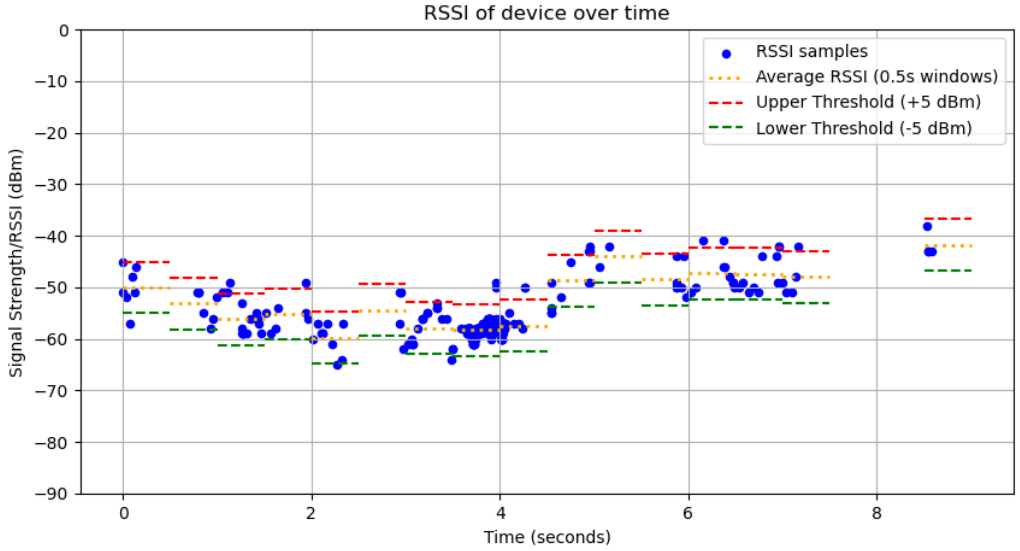
4.5.2 Probe Request

For devices already connected to the AP, probe requests are used to classify the vendor. Probe request for devices that use MAC randomization sometimes use different MAC addresses for probe requests, as for the MAC address it is connected to the AP with. Therefore, our system has to match the probe request with the device before classifying the vendor. To do this, our system looks through the last 10 seconds of the frames sent from the specific device and all probe requests sent the last 10 seconds. It then goes through all probe requests and first check if it uses the same MAC address as the device. If that is not a success, it compares the RSSI of the probe request to the RSSI of the packets sent at plus or minus 0.5 second. It does this by finding the average RSSI of the nearby packets and if the probe requests RSSI is the same with an error of 5 then the probe request is stored as a signal hit. The 0.5 seconds and RSSI error of 5 were chosen by testing a device's RSSI over 10 seconds. One test when the device was not moving shown in Figure 4.3a, and one when the device was moving shown in Figure 4.3b. In the

figure, the red and green dotted lines show the gaps for which the RSSI will be classified as a hit, which is calculated by making a range of plus and minus 5 dBm from the average RSSI in the 0.5 second time windows, which is shown as the yellow dotted line in the figures. The blue dots are the packets sent from the device. The figures show that the packets are mostly inside the RSSI range created by the upper and lower thresholds.



(a) Device not moving



(b) Device moving

Figure 4.3: RSSI over 10 seconds with calculated upper and lower thresholds

After the system has found all probe request that were a signal hit it loops through them and sees if the probe request were sent when the device had a gap in sending data to signalize that is sending a burst of probe requests. The time gap is set at 20 ms given the average burst time of probe requests explained in Section 2.2. All the probe request MAC addresses are then stored with a hit percentage based on how many time gaps the probe request hits. This is calculated using this formula:

$$hitPrecentage = hits/(hits + misses) \tag{4.11}$$

Then the system chooses the probe request MAC address with the best hit percentage and looks for the vendor specific tag in the management information to classify the devices vendor. For making the system more accurate, the hit percentage must be over 70% to be seen as probe request and device match.

The system also stores a value from 0 to 3 of how sure it is that it has found the devices vendor. If the sureness is 3 then the device does not use MAC randomization, and we are sure we have identified the vendor. If it's a 2 then association requests are used for vendor identification. We are then pretty sure, but since we only can extract the network cards vendor, its not always giving the vendor we are looking for. When the vendor sureness is 1 the system has used probe requests for identifying the vendor. Since a different MAC address is used for probe request and connection to the AP we are not very sure that the probe requests is correct and therefore that the vendor is correct. If the vendor sureness is 0 then the vendor is unknown. This value is used for the system to know if it needs to search for a vendor or not, the system only tries to classify a vendor using information from a sureness with a higher value than it is already assigned.

4.6 Device Re-Identification under MAC Randomization

This subtask addresses the problem of device re-identification under MAC randomization. When a wireless device disconnects from a network and later reconnects, it may use a newly randomized MAC address, causing the same physical device to appear as a new device.

To handle this problem device fingerprinting is applied. Instead of relying on MAC addresses, each device is represented by a fingerprint based on implementation specific characteristics observed during the connection process. When a device reconnects using a new randomized MAC address, its newly extracted fingerprint is compared against previously observed fingerprints.

The fingerprint is constructed from features extracted from IEEE 802.11 association request frames, including information element ordering, supported data rates, HT capabilities, RSN parameters, extended capabilities, and vendor specific information elements, as described in Section 2.2. The comparison produces a similarity score indicating how closely two fingerprints match. Since fingerprints are not perfectly identical across reconnections, due to variations in device state and network conditions, the method is probabilistic rather than deterministic.

Based on empirical evaluation (see Section 3.3.5) with multiple devices, a similarity threshold of 0.8 is selected from the values later shown in Table 5.4. If the similarity score between two fingerprints exceeds this threshold, the MAC addresses are classified as originating from the same physical device. Scores below the threshold indicate that the MAC addresses most likely belong to different devices.

4.7 Activity Classification

As described in Section 3.3.4, this project explores whether traffic analysis using simple threshold based rules can distinguish between user activities. The activity classification component was implemented to determine whether a device is idle, browsing, or streaming based on passively observed data frames.

4.7.1 Data Collection for Development

To develop the classifier, initial traffic captures were gathered for three activities: idle, browsing, and streaming. The idle class was captured with the device screen on, but with no applications open. Background processes such as automatic updates were not explicitly disabled in order to better represent realistic device behavior.

For the browsing class, the user visited Aftonbladet and interacted by reading articles and navigating between them. Autoplaying advertisements and videos were paused during these captures to avoid mixing browsing traffic with traffic resembling streaming. Since the classifier is based on observed traffic patterns rather than user intention, it cannot determine whether video traffic during a session comes from intentionally watching it or from autoplaying media. Therefore, the autoplaying videos were paused to keep a clear separation in patterns between the browsing class and streaming class. For the streaming class, YouTube was used to generate video-streaming traffic.

For each activity, the traffic was captured in 10-minute segments. Two segments were collected from an Android phone and two segments from a Windows laptop, resulting in 40 minutes of development data for each of the three activities. The traffic was captured using Scapy with additional filtering to allow monitoring of a specific network.

4.7.2 Threshold Selection

The development captures were divided into 10-second time windows. For each window, traffic statistics were calculated from the observed data frames. This allowed the traffic to be analyzed based on short-term traffic behavior, which is suitable for a real-time system.

Included in the calculated statistics were, number of packets, bytes per second, average frame size, and average interarrival times. These statistics were calculated separately for both uplink and downlink, as well as for the total traffic in each window.

To determine whether a time window represented an idle or active device, where active refers to either browsing or streaming, the clearest metric was the total number of bytes per second. As shown in Figure 4.4, the idle windows were concentrated at lower byte rates than the other classes. Based on this distribution, the 75th percentile of idle bytes per second was chosen as the threshold, in order to avoid the large spikes seen in some idle windows. This resulted in an idle threshold of 220 bytes per second.

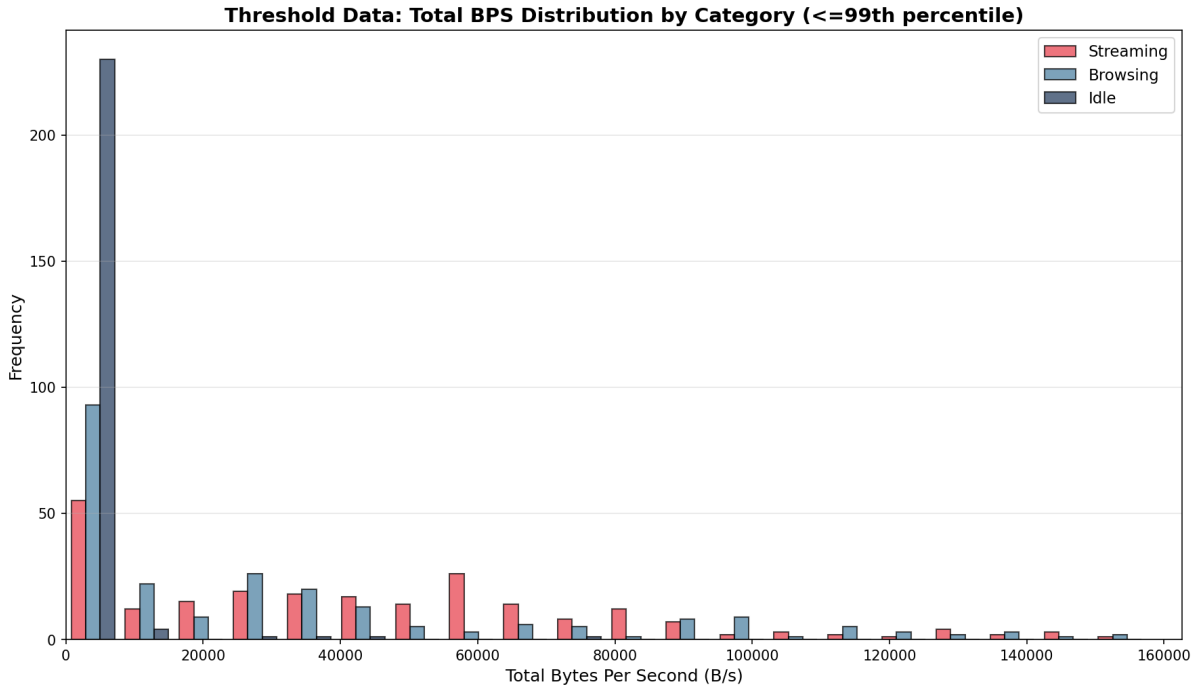


Figure 4.4: Histogram of bytes per second by category

To determine whether an active window represented browsing or streaming, the ratio between uplink and downlink measured in bytes was initially considered. Streaming traffic was consistently above a 90% downlink share in the phone captures. However, the laptop captures showed that browsing traffic could also exceed a 90% downlink share, since the laptop generated less uplink traffic during browsing. Therefore, this was not an effective metric for a system aiming to classify traffic from different kinds of devices.

A clearer distinction between browsing and streaming, which was more consistent across both the phone and laptop captures, was the average size of downlink data frames. As shown in Figure 4.5, most streaming windows had a larger average downlink frame size than browsing windows. An average downlink frame size threshold of 1200 bytes was therefore chosen to classify streaming. This threshold is above the 95th percentile of browsing windows, which was at 1116 bytes, while still placing 70% of streaming windows from the development data above the threshold.

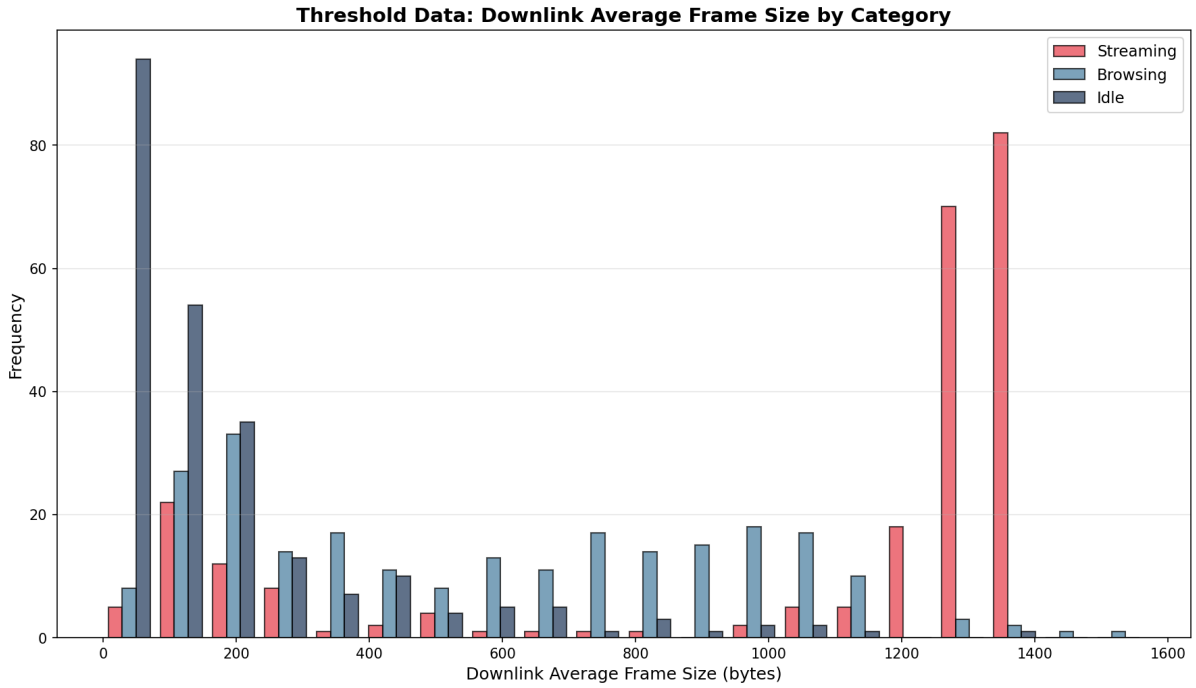


Figure 4.5: Histogram of average downlink frame size by category

The selected thresholds are used to mark whether a window shows signs of activity and whether it contains large downlink frames, as shown in Figure 4.6. Before a final prediction for a window is made, a temporal smoothing step, described in the next section, is applied.

```
active_window = bytes_per_second > 220
large_downlink_frame_window = downlink_avg_frame_size > 1200
```

Figure 4.6: Pseudocode showing the the threshold indicators for activity classification.

4.7.3 Temporal Smoothing

The threshold checks described in the previous subsection are applied to each individual 10-second window. To make the classifier more stable in the event of temporary traffic spikes or temporary drops in traffic, temporal smoothing is applied by considering the current window together with the two previous windows.

The first smoothing step determines whether the device should be considered active. As described in the previous subsection, a window is marked as active if its total traffic exceeds 220 bytes per second. The classifier considers a device as active if at least two of the last three windows, including the current one, exceed this threshold. This prevents a short spike in background traffic from changing the classified activity from idle to active, and also prevents a temporary drop in traffic from changing the activity from browsing or streaming to idle.

If the device is considered active, the classifier then determines whether the activity is browsing or streaming. A window is marked as streaming-like if its average downlink frame size exceeds 1200 bytes. The device is classified as streaming if at least two of

the last three windows exceed this threshold. Otherwise it is classified as browsing. Like the previous smoothing step, this prevents a browsing device with a temporary spike in downlink frame size from being classified as streaming, and prevents a streaming device with a temporary drop in frame size from being classified as browsing. The resulting decision process is summarized in Figure 4.7.

```
active_count = number of windows in the last 3
               where bytes_per_second > 220

if active_count < 2:
    label = idle
else:
    large_frame_count = number of windows in the last 3
                        where downlink_avg_frame_size > 1200

    if large_frame_count >= 2:
        label = streaming
    else:
        label = browsing
```

Figure 4.7: Pseudocode showing the temporal smoothing logic for activity classification.

This smoothing makes the classifier less sensitive to windows that differ from the surrounding traffic pattern. However, it also means that activity changes may require more than one window before they are reflected in the classification. Another thing to note is that when starting the classifier, at least two time windows are needed before a non-idle prediction can be made, and at least three time windows are needed before the temporal smoothing can be fully applied.

4.7.4 Real-Time Integration

After the thresholds and temporal smoothing rules had been defined, the activity classifier was integrated into the real-time monitoring system. The classifier processes incoming data frames as they are observed and maintains a separate activity state for each observed device on the monitored network.

When a data frame is observed, the system assigns it to the corresponding device based on the MAC address and stores the frame size, timestamp and traffic direction which is based on if the device's MAC address is in the source or destination field of the frame. When a data frame from a later 10 second time window is observed, the previous window is completed and its features calculated. Only the three most recent windows are stored for each device.

For each completed window, the system calculates the total bytes per second and average downlink frame size. These values are then compared against the selected thresholds and combined with the two previous windows using the temporal smoothing logic. The resulting label is stored as the latest activity classification for the device.

The classifier only emits a new result when a time window has been completed. Until then, the system keeps the latest completed classification to avoid updating the displayed activity based on an incomplete window.

5

Results

This chapter presents the results of the system's performance. A series of tests were conducted on the different components to see how well they performed and to assess how effectively the system fulfills its intended purpose. The results of these tests are shown in the sections below.

5.1 Results of AP Grouping and Hidden SSID Association

In this section, the grouping and pairing methods were evaluated to assess their effectiveness. The test setup consisted of two Wi-Fi routers: the RT-AX52 and the WiFi-Hub C2. The RT-AX52 provides two APs, one operating in the 2.4 GHz band and one in the 5 GHz band. The WiFi-Hub C2 provides three APs: one in the 2.4 GHz band and two VAPs in the 5 GHz band. Table 5.1 presents how the different routers were grouped or not grouped, where the Wi-Fi routers settings where applied to all its APs. Both Wi-Fi routers listed in Table 5.1 belong to the same ESS. In this context, "grouped" means that the algorithm identified the Wi-Fi routers and its associated APs as belonging to the same logical ESS, while "not grouped" means that no such relationship was detected.

Table 5.1: SSID and securityc configuration comparison between RT-AX52 and WiFi-Hub C2

RT-AX52 SSID	WiFi-Hub C2 SSID	RT-AX52 Security	WiFi-Hub C2 Security	Grouped
Tele1	Tele1	WPA2-personal	WPA2-personal	Grouped
Tele2	Tele1	WPA2-personal	WPA2-personal	No grouped
Tele1	Tele1	WPA2/WPA3-personal	WPA2-personal	No grouped

Table 5.2 and Table 5.3 present the performance of the pairing algorithm, which attempts to associate a hidden SSID AP with a visible AP. The algorithm operates within the same ESS and uses a similarity threshold of 0.7. The weights assigned to the individual bytes of the BSSID for similarity scoring are $[0.2, 0.2, 0.2, 0.1, 0.1, 0.2]$. Furthermore, the skew parameter is defined as $\lambda_{\text{skew}} = 1/5$, and the power parameter is $\lambda_{\text{power}} = 1/10$. These parameters were determined empirically by testing different configurations and selecting the values that produced the best overall performance. The higher weights assigned to specific BSSID bytes are motivated by the fact that these bytes typically exhibit greater variability and are therefore more informative for distinguishing between APs. Section 4.3 goes more in-depth about the context of these parameters.

In Table 5.2, where both APs of the RT-AX52 were hidden, no successful pairing was achieved for these APs, this resulted in an accuracy of 33%. In contrast, Table 5.3 shows that when one RT-AX52 AP was visible within the ESS, the system was able to correctly infer associations for all hidden APs in this configuration, this resulted in an accuracy of 100%.

Table 5.2: APs and hidden SSID pairing results where both RT-AX52 AP was a hidden SSID

Hidden	BSSID	Router	RSSI (dBm)	TSF Skew (ppm)	BSSID Similarity	Power Score	Skew Score	OUI Score	Confidence Score	Parent AP Match
No	34:53:d2:9c:f9:3c	WiFi-Hub C2	-37	-16.887	–	–	–	–	–	–
No	34:53:d2:9c:f9:3d	WiFi-Hub C2	-89	-18.353	–	–	–	–	–	–
Yes	36:53:d2:9c:fa:3e	WiFi-Hub C2	-89	-18.505	0.975	1.000	0.970	1.0	0.973	34:53:d2:9c:f9:3d
Yes	bc:fc:e7:ac:90:e8	RT-AX52	-44	-7.169	0.100	0.011	N/A	0.6	0.222	Rejected
Yes	be:fc:e7:ac:90:e8	RT-AX52	-36	-5.505	0.100	0.005	N/A	0.6	0.221	Rejected

Table 5.3: APs and hidden SSID pairing results where one RT-AX52 AP was a hidden SSID

Hidden	BSSID	Router	RSSI (dBm)	TSF Skew (ppm)	BSSID Similarity	Power Score	Skew Score	OUI Score	Confidence Score	Parent AP Match
No	34:53:d2:9c:f9:3c	WiFi-Hub C2	-35	-17.640	–	–	–	–	–	–
No	34:53:d2:9c:f9:3d	WiFi-Hub C2	-89	-18.678	–	–	–	–	–	–
No	bc:fc:e7:ac:90:e8	RT-AX52	-47	-6.757	–	–	–	–	–	–
Yes	36:53:d2:9c:fa:3e	WiFi-Hub C2	-90	-18.474	0.975	0.905	0.960	1.0	0.922	34:53:d2:9c:f9:3d
Yes	be:fc:e7:bc:90:e8	RT-AX52	-44	-7.346	0.887	0.741	0.889	1.0	0.776	bc:fc:e7:ac:90:e8

5.2 Movement Detection

When implementing the movement detection, testing had to be done simultaneously in order to find the suitable thresholds. As mentioned in Section 4.4 the chosen n_σ was three, which is the most commonly used and is called the "three sigma rule" due to its properties of removing the least amount of valid data and still cover the most outliers. To test the implementation, a PCAP from a controlled environment was used, where one device was known to be moving, and two was known to be stationary.

Other than the thresholds, the window size also had to be tested. In the beginning the size was set to be 60. This made the outcome somewhat stable. At least for the stationary devices due to few outliers having to be removed. However, for moving devices larger windows often remove valid data as outliers because of larger windows have time to get larger difference in RSSI. Additionally, in real time for moving devices it often took a very long time to get the first 60 samples in order to correctly show the movement status. Furthermore, the time to update a status shift took longer time than desired. Therefore, the window size was decreased to 40 in order to attain a high accuracy while still having good responsiveness.

Figure 5.1 shows the thresholds in the beginning with a window size of 60 and a threshold of 1.5. Since the orange area indicates a device in motion, it was very accurate for stationary devices. However, in real time for moving devices it often switched between moving and stationary, as can be seen in the alternating orange area on the top graph in Figure 5.1.

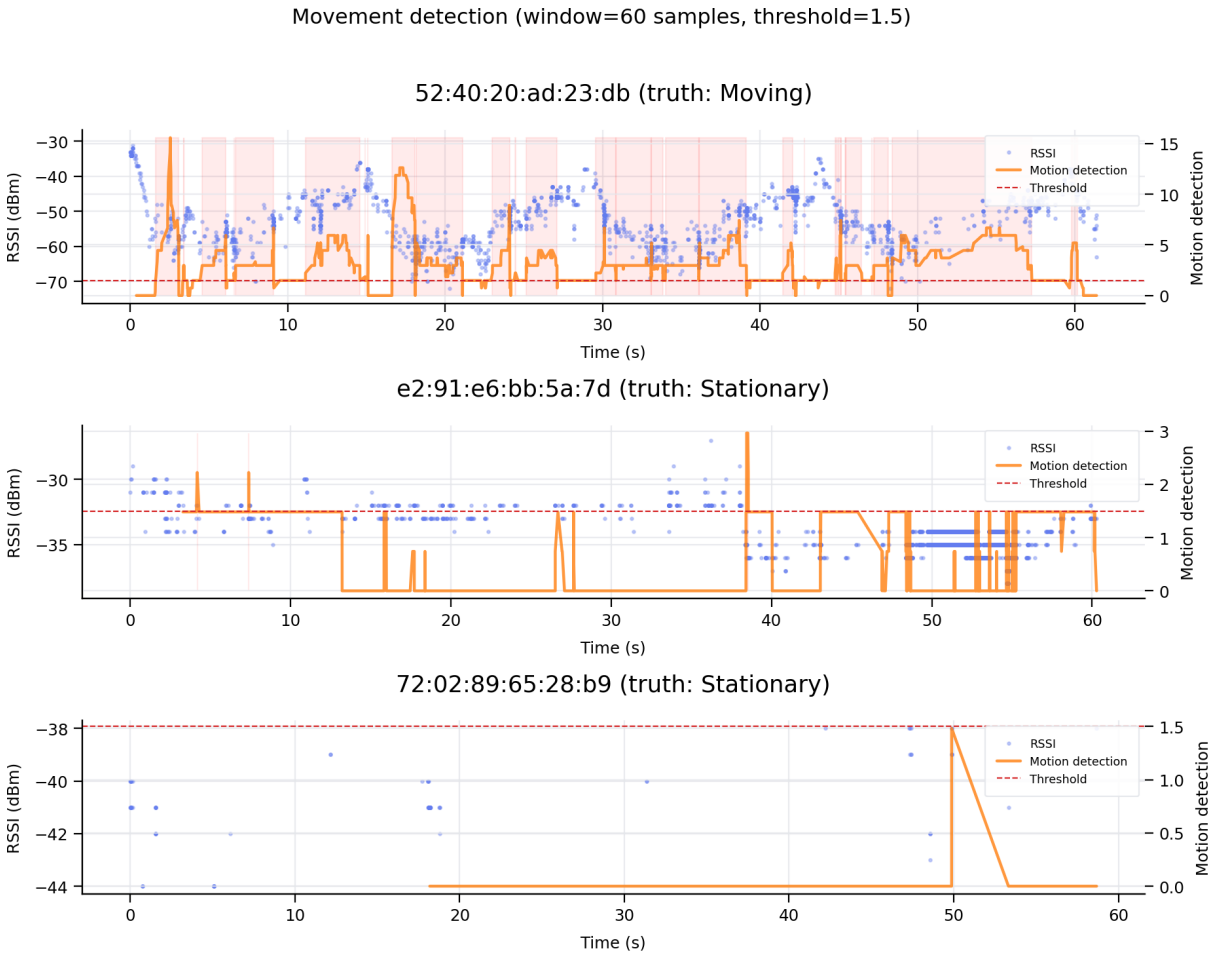


Figure 5.1: RSSI over 60 seconds with one moving device and two stationary. The window size is 60 samples and threshold to detect movement is 1.5. Every RSSI sample is represented as a blue dot. Dotted red line indicates threshold. Orange line represents motion detection. Orange area indicates estimated moving device.

In order to fix the moving accuracy there are two ways to solve the problem. One is to lower the threshold for when a device should be displayed as moving. This would result in higher accuracy for moving devices. However, the result would be the opposite for the stationary devices. The other way would be to decrease the window size to not smooth out real data as outliers.

Figure 5.2 shows the result after adjusting the thresholds. The result is very accurate for stationary devices. The only big exception is in the beginning where it needs more data to reach 40 samples in order to be calibrated and be more accurate. Furthermore, the moving accuracy is also very high, showing the correct outcome almost all of the time. In addition to being more accurate than the 60 window, it is also faster to update in real time due to the shorter sample window.

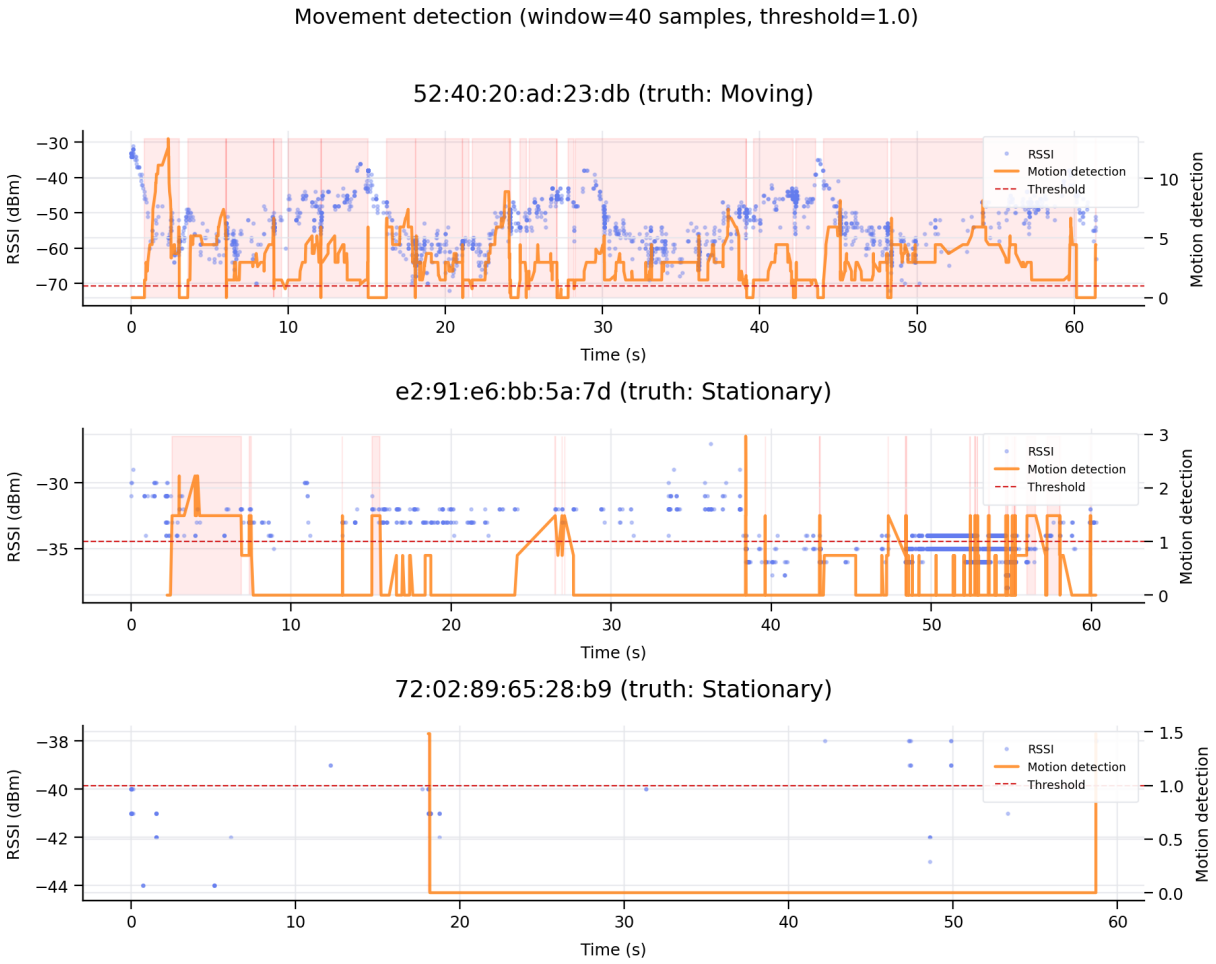


Figure 5.2: RSSI over 60 seconds with one moving device and two stationary. The window size is 40 samples and threshold to detect movement is 1.0. Every RSSI sample is represented as a blue dot. Dotted red line indicates threshold. Orange line represents motion detection. Orange area indicates estimated moving device.

When suitable thresholds were found, live tests were frequently made, where multiple devices were stationary, while phones were constantly moving. This was to test the responsiveness of how fast the system would recognize a difference in movement. In this test the shorter window size was the better option since shorter window size resulted in faster response. Even for the 40 window size it was unstable in the beginning before reaching a full window. This is because larger windows need more time to be filled while very short windows can be very inaccurate. As a result, the 40 sample window with 1.0 threshold was the suitable choice. Figure 5.3 shows the accuracy for different window sizes and thresholds that determines movement. For a faster response in real time, smaller window sizes are the better option. However, the result should still remain a high accuracy prediction. Therefore, window size 40 and threshold 1.0 is the best option, since it is faster and has an accuracy of 0.91. The accuracy was calculated by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Where TP = True Positive, TN = True Negative, FP = Fake Positive and FN = Fake negative.

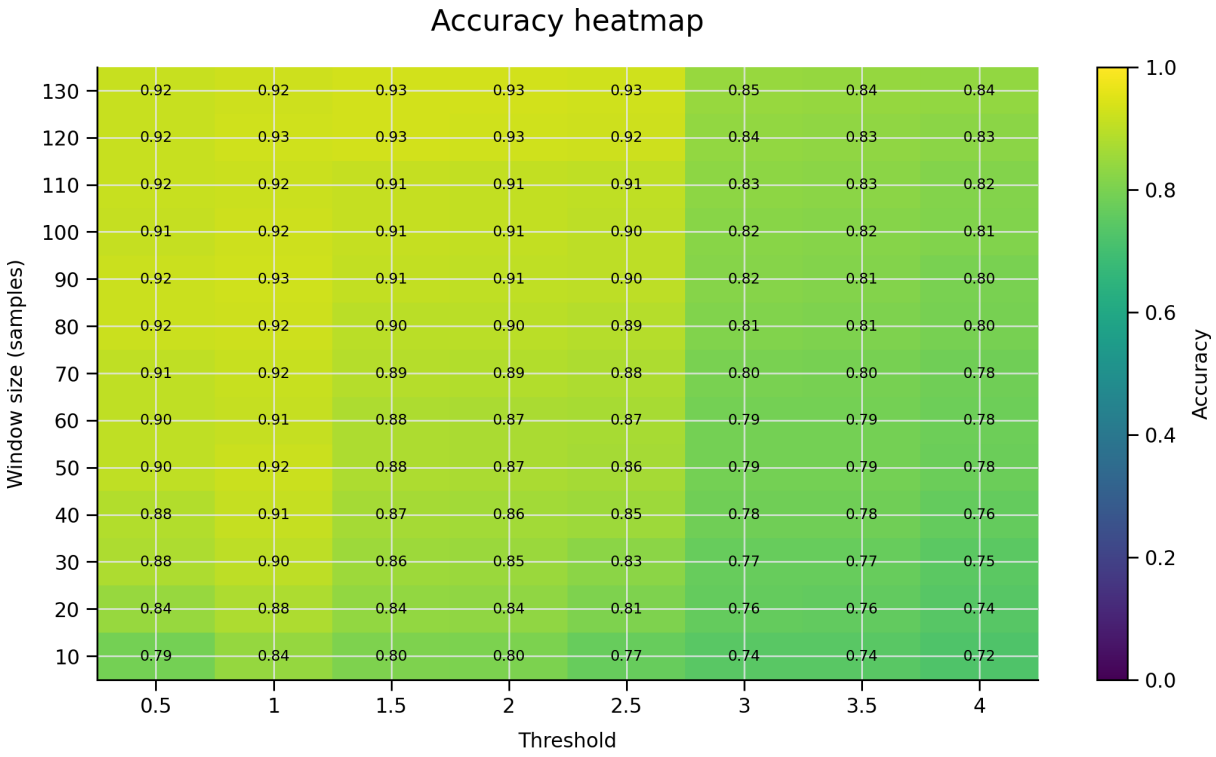


Figure 5.3: Heatmap of movement accuracy for window sizes between 10-130 and thresholds 0.5-4.0. Lighter color indicates higher accuracy

5.3 Vendor Classification

The vendor classification was tested in a controlled environment where only known devices were connected to our AP and only probe requests from known devices was analyzed. This was done by setting a maximum RSSI for probe request and making sure only our devices were inside of that area. The classification test was done on a iPhone 13 and a MacBook Air M5, with vendor Apple Inc.

Connection to the vendor were successful on average after 25 seconds, when the device was active, and always gave the correct vendor. All the times when a connection was successful, it was because the device sent a probe request with the same MAC address as it is connected to the AP with. Even though, as explained in Section 4.5, the system is able to link randomized MAC addresses to the same device, vendor information is not present in the management information of these packets. When a device connected to the AP, the system was always able to identify the vendor from the Association Request.

Figure 5.4 shows how the system is able to connect probe request with randomized MAC address with a device. The figure shows on the top, probe requests sent from the device we wanted to classify with randomized MAC addresses. To know that the probe request was sent from the correct device, no other devices were at a similar distance from the NIC doing the monitoring. The red dots on the bottom are probe request sent from other devices that where placed a bit further away. We can see from this test that no probe requests sent from another device is said to be coming from the targeted device, assuming there is no other device in close proximity. The test also shows the system is able to connect the correct probe request around 60% of the time.

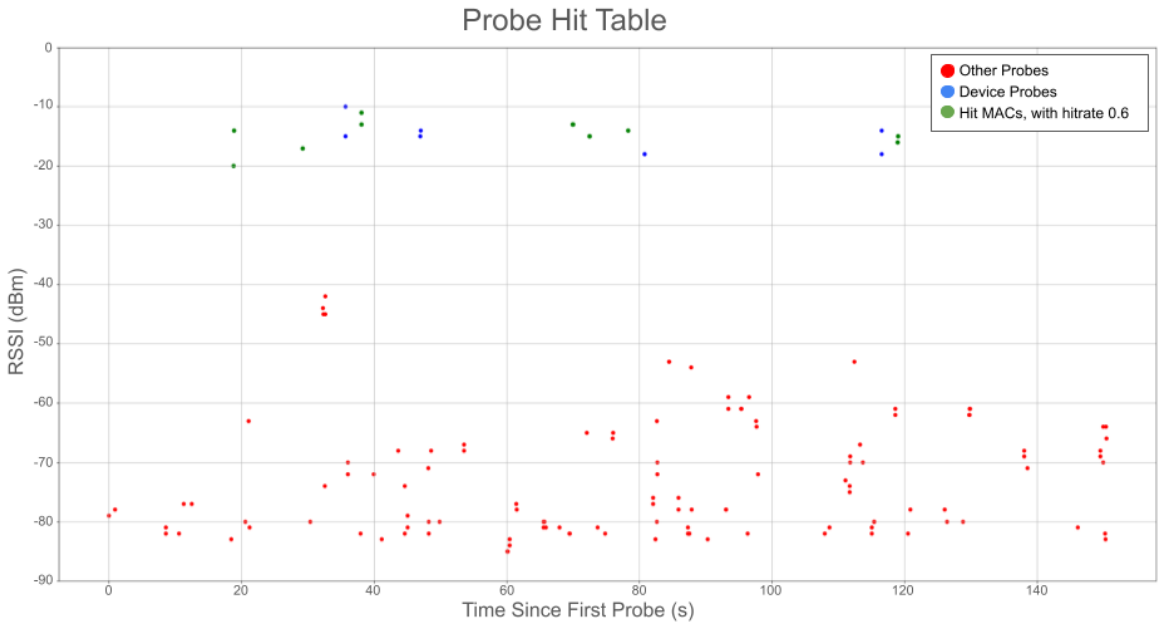


Figure 5.4: Probe Request captured over 150 seconds. The blue dots show the probe requests sent from the test device, the red dots show probe requests sent from other devices. The green dots show the probe request the system connects with the device.

5.4 Device Re-Identification Under MAC Address Randomization

The proposed fingerprinting approach was evaluated using three different physical devices: an iPhone 13, an iPhone 11, and an HP laptop. All devices employed MAC address randomization across reconnection events. For each device, association request frames were captured during an initial connection to the Wi-Fi network, after which the device disconnected and reconnected using a newly randomized MAC address. This process resulted in multiple distinct MAC addresses corresponding to the same physical device.

Figure 5.5 presents a fingerprint similarity heatmap comparing all observed MAC addresses. MAC addresses coming from the same physical device form distinct high similarity clusters along the diagonal, while comparisons between different devices consistently had lower similarity scores.

Based on the evaluation method explained in Section 3.3.5, MAC address pairs belonging to the same device achieved similarity scores in the range between 0.90 and 1.00, whereas comparisons between different devices resulted in scores between 0.52 and 0.68 (see Table 5.4). Based on this clear separation, a decision threshold of 0.8 reliably distinguishes between same devices and different devices.

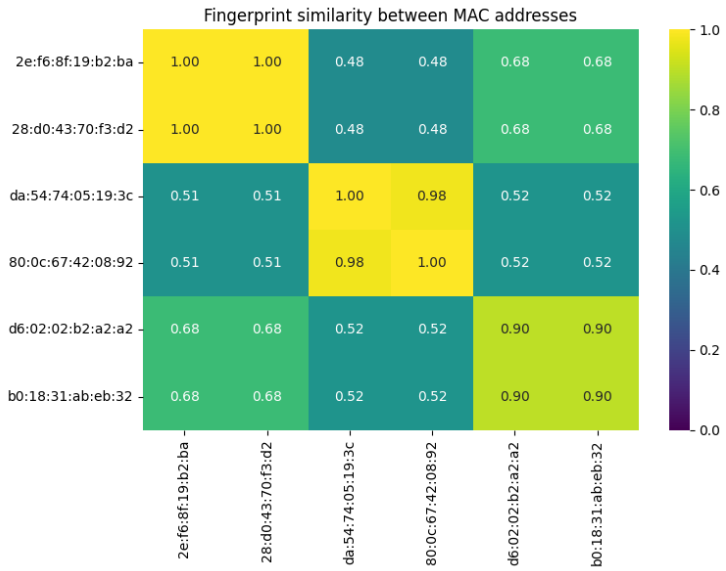


Figure 5.5: Heatmap of fingerprint similarity scores for all observed MAC addresses.

Table 5.4: Observed similarity score ranges for same devices and different devices.

Category	**Similarity Score Range**
Same device	0.90–1.00
Different devices	0.52–0.68

5.5 Activity Classification

This section presents the window-level results for the activity-classification evaluation, based on the evaluation method described in Section 3.3.4. Each 10-second window was compared with the known activity label of the corresponding capture. Table 5.5 shows the resulting confusion matrix, and Table 5.6 shows the precision, recall, and F1-score for each class, as well as the macro average.

Table 5.5: Confusion matrix for the activity classifier. Rows represent the true class and columns represent the predicted class.

	Predicted Idle	Predicted Browsing	Predicted Streaming
True Idle	283	61	4
True Browsing	11	322	15
True Streaming	3	23	322

Table 5.6: Classification metrics for the activity classifier. Precision, recall, and F1-score are reported separately for each class, with the macro average calculated across the three classes.

Class	Precision	Recall	F1-score
Idle	0.953	0.813	0.878
Browsing	0.793	0.925	0.854
Streaming	0.944	0.925	0.935
Macro average	0.897	0.888	0.889

The classifier achieved a macro F1-score of 0.889. Streaming had the highest F1-score at 0.935, while browsing had the lowest F1-score at 0.854.

6

Discussion

Based on the purpose of this project (see Section 1.2) we wanted to investigate how much information you can gather from devices with a passive monitoring system. The following sections discuss these findings in more detail, relating each subsystem to its strengths, limitations, and implications for passive wireless monitoring.

6.1 Discussion of AP Grouping and Hidden SSID Pairing Results

By identifying which APs belong to the same ESS, including both visible and hidden APs, the system becomes capable of continuously monitoring a device even when it transitions between APs within the network or tries to hide its connection to the network by connecting to a hidden SSID. These results demonstrate that even without prior knowledge of the network structure or AP relationships it is possible to get the ESS structure.

6.1.1 Impact of SSID and Security Configuration on ESS Grouping

From the result in Section 5.1, Table 5.1 demonstrates that when all APs shared the same SSID and identical security parameters, the system classified them as belonging to the same network. When the APs had different SSIDs but identical security parameters, they were classified as belonging to different networks. This behavior is consistent with the standard definition of an ESS, in which APs that belong to the same ESS are expected to share the same SSID. Therefore APs with differing SSIDs are, with high probability, associated with different ESSs [35].

A more notable case occurs when APs share the same SSID but differ in their security parameters. In such cases, the grouping method does not classify the APs as belonging to the same ESS. The reasoning for this is that, by standard, all APs in a network should have the same SSID and security configurations. But practical deployments do not always strictly follow this convention. Differences in security settings may arise due to configuration errors, transitional security implementations, or other deployment related factors [35]. Since no strict protocol dictates the exact configuration of APs within an ESS, absolute certainty cannot be achieved. But the standard is that APs with identical SSIDs but differing security configurations belong to separate ESSs. An additional possibility is that an AP with the same SSID but different security settings

could represent a rogue AP, but this lies outside the scope of this paper. Because of this reasoning, the system follows the standard assumption that network are configured according to established conventions.

6.1.2 Limitations and Behavior of Hidden SSID Pairing

The pairing results presented in Section 5.1 are in Table 5.2 and Table 5.3. Table 5.2 shows that, among the three APs with hidden SSIDs, only one was successfully paired. This observation can be further understood by examining Table 5.3, in which all APs with hidden SSIDs were successfully paired. In Table 5.2, the two APs with hidden SSIDs that failed to pair both belonged to the same Wi-Fi router, RT-AX53. In this configuration, none of the APs associated with the router had visible SSIDs. While Table 5.3 shows that each Wi-Fi router included at least one AP with a visible SSID. This outcome can be explained by the characteristics of the pairing method, which requires that an AP with a hidden SSID has at least one visible AP within the same ESS operating on similar hardware components. The features utilized by the pairing method are strongly dependent on hardware specific properties of the AP.

This dependency explains why, in both Table 5.2 and Table 5.3, the AP with MAC address 36:53:d2:9c:fa:3e is consistently paired, as it is associated with at least one corresponding visible AP sharing similar hardware characteristics. In Table 5.3, the hidden SSID associated with the RT-AX52 was successfully paired once a visible AP from the same Wi-Fi router became available, indicating that the presence of a visible AP with matching hardware characteristics enables the pairing process to succeed. This also highlights a limitation of the approach, namely that the system requires at least one visible AP sharing similar configurations with APs using hidden SSIDs. This limitation could be mitigated by allowing the system to wait until a client connects to the hidden SSID, since, as discussed in Section 2.5, the SSID can be inferred when a device associates with the AP.

The results in Table 5.3 can also be discussed in terms of the potential impact of the AP with MAC address be:fc:e7:bc:90:e8 belonging to a different ESS. In such a case the system incorrectly classify this as a false positive. Section 2.5 presents the theoretical background regarding the use of VAPs to create two distinct ESSs on the same physical hardware, which would explain this misclassification. Roth et al. [11] provides examples where different VAPs from different devices exhibit high similarity in their BSSIDs, which could affecting the pairing results in Section 5.1 negatively.

6.2 Movement Detection Based on RSSI Variations

As RSSI is based on signal strength and not distance, disturbances may cause problems. Anything that can absorb or reflect radio waves may interfere with the signal strength and therefore affect the outcome. Even though the distance is the most dominant factor for change in RSSI, other things such as walls, floors, ovens, humans or other thick objects may affect the values. This can lead to some inaccuracy due to stationary devices being affected by other environmental aspects. For example two humans carrying a large couch in between the device and the monitoring device, might temporarily negatively affect the signal strength and the system will detect this as a device moving further away from the monitoring device.

On the other hand, in real life environments these types of large objects are not very commonly interfering, especially not for longer periods of time. Since the system uses windows of 40 samples, smaller and shorter deviations are not affecting the outcome since they will be regarded as outliers.

One problem that the system does not take into account is the start up time before the device has the first 40 samples. It needs 10 samples to start showing a prediction. However, this prediction is very unstable since the window is very small. One way to prevent this is to not show any prediction at all until the full 40 samples have been filled. On the other hand, stationary idle device can take very long time to reach 40 samples since they send out less samples which would result in no displayed prediction at all.

One way to fix this problem would be to have time based windows instead of samples. However, this can also cause problems since it would be very inconsistent due to the window size having a large impact on the outcome. Devices with more activity would have larger windows than idle ones which would make the results vary and be more inaccurate. As a conclusion, we decided to keep the sample window over time windows.

6.3 Vendor Classification via Management Frames

For most devices given enough time, it was possible to classify the vendor. Even with devices already connected to the AP, probe requests with non randomized MAC address were sent out and made it possible to extract the vendor. But for devices that also used randomized MAC address for the probe requests, most probe request that were sent did not have the vendor specific tag, so identifying the vendor was often not possible. Even though we were able to connect the probe requests to the device, the information we could extract did not make it possible to classify the vendor. Furthermore, randomized probe requests were often only sent out when the device was looking for an AP, for example when going info the Wi-Fi settings on a phone. This also made identifying the vendor more difficult, because the device was not sending management frames.

For future development a database of fingerprints from the information presented in Section 2.2, can be used to identify the vendor and even device type. This can be done by comparing the fingerprint of the device to a database of fingerprints, similar to how devices for device re-identification in section Section 4.6 are compared.

6.4 Device Re-Identification Under MAC Address Randomization

The results demonstrate that MAC address randomization alone is not enough to prevent device re-identification in IEEE 802.11 networks, which is consistent with prior work on MAC layer fingerprinting and device tracking [15], [16]. Despite changes in MAC addresses across reconnection events, devices exposed highly stable fingerprints derived from association request frames, which are known to expose specific characteristics of the wireless hardware and software stack (see Section 2.2).

As shown in Figure 5.5, MAC addresses coming from the same physical device consistently resulted in high similarity scores between 0.90 and 1.00, while different devices remained

separated with lower scores. This separation suggests that the selected features effectively capture stable implementation characteristics rather than random network behavior.

The similarity threshold of 0.8 was selected based on experimental observations rather than assumptions. The heatmap visualization revealed a clear gap between similarity scores from the same devices and scores from different devices. Choosing a lower threshold could risk merging different devices as one, while a higher threshold could risk fingerprints belonging to the same device not being merged. The selected threshold therefore represents a balanced compromise between precision and recall.

6.5 Activity Classification

This section discusses the evaluation of the activity-classification component, the results of which are summarized in Section 5.5. As shown in Table 5.6, the threshold-based method was able to distinguish between idle, browsing, and streaming with relatively high performance. The classifier achieved a macro F1-score of 0.889, which indicates that the simple rule-based method performed reasonably well across all classes. This supports the idea that passively observable traffic characteristics can be used to infer user activity, and suggests that machine learning is not necessarily required to achieve useful classification performance for a small set of activities under the tested conditions.

Streaming achieved the highest F1-score. This is likely because streaming traffic often produces a more continuous and predictable downlink flow, despite occasional buffering. As shown in Table 5.5, streaming traffic was rarely classified as idle, and most errors were instead classified as browsing. This indicates that streaming windows more often failed to meet the average downlink frame size threshold than they fell below the bytes per second threshold for being an active window.

Browsing achieved the lowest F1-score. This is likely because browsing does not have one consistent traffic pattern, especially across different types of websites. When a page is loading, browsing can generate short bursts of high traffic, but when the user is reading an article, the traffic can instead become much lower. This explains why browsing can overlap with both idle and streaming, depending on whether the user is actively loading pages or passively reading.

The idle class had high precision but lower recall. This means that when the classifier predicted idle, it was usually correct, but a noticeable number of true idle windows were classified as browsing, as shown in Table 5.5. This suggests that the threshold used to determine whether a window was idle was too low to filter out some background traffic. Increasing this threshold would likely improve recall for the idle class, but could lead to more browsing windows on lightweight websites such as Wikipedia being incorrectly classified as idle.

Since the classifier uses temporal smoothing, there is a delay between a change in traffic patterns and the classifier changing its label. While this was intended to make the classifier more resistant to short temporary changes, it may also take longer to return to the correct classification if a drop or spike in traffic remains for multiple windows. Since no comparison was made against a version without temporal smoothing, the effect of this design choice cannot be measured separately. However, because the classifier uses 2-out-of-3 logic with 10-second time windows, a sustained change would be expected to affect

the final label after 20-30 seconds, depending on where in a time window that change occurs.

6.6 Terminal User Interfaces

The implemented system successfully performed both AP discovery and target network device monitoring in near real time. During discovery mode, the system continuously detected nearby APs across the configured channels and grouped them according to their SSID and security type, while preserving per BSSID attributes such as channel and RSSI, as shown in Figure 6.1. This enabled the identification of networks consisting of multiple physical APs and provided a coherent and stable representation of the surrounding wireless infrastructure.

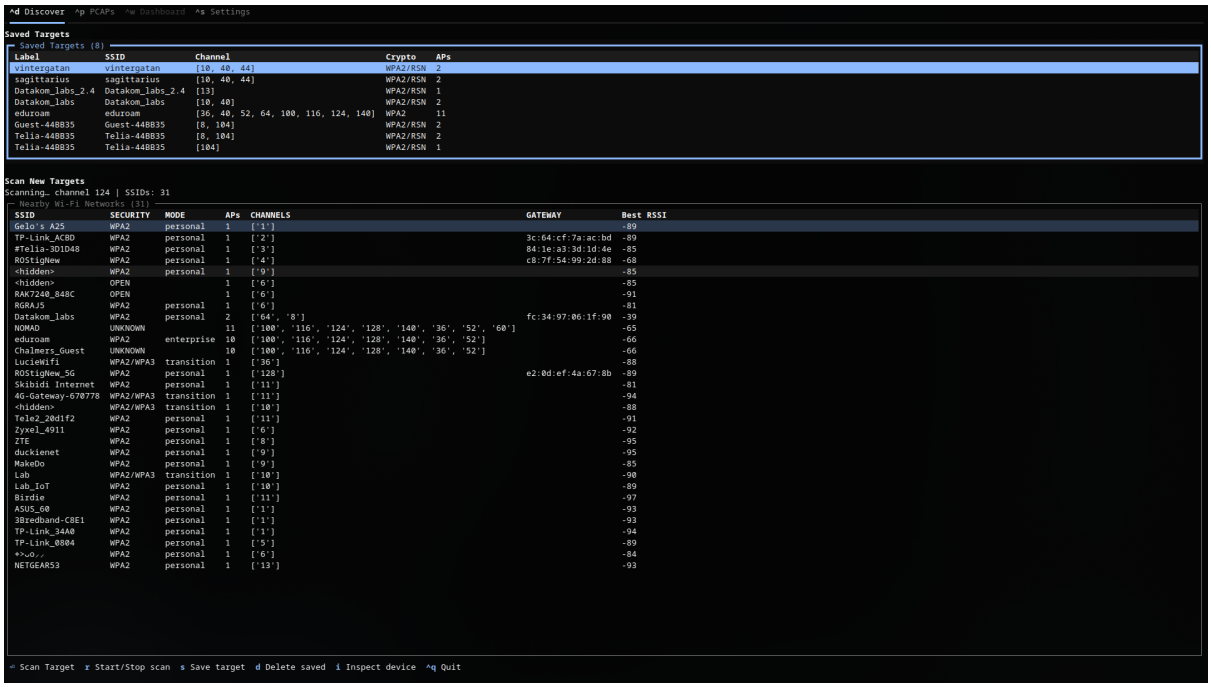


Figure 6.1: Discover page scans nearby Wi-Fi networks and groups APs by SSID and security type, providing a clear overview of the surrounding wireless infrastructure.

In monitoring mode, the system effectively tracked devices associated with a selected network by filtering captured traffic based on predefined target BSSIDs. Device records were continuously updated with relevant metrics, including frame count, RSSI history and inferred vendor information, thereby offering a dynamic view of network activity over time, as shown in Figure 6.2. The user interface remained responsive throughout continuous packet capture, and periodic state updates ensured that changes in observed devices were reflected in a clear and timely manner.

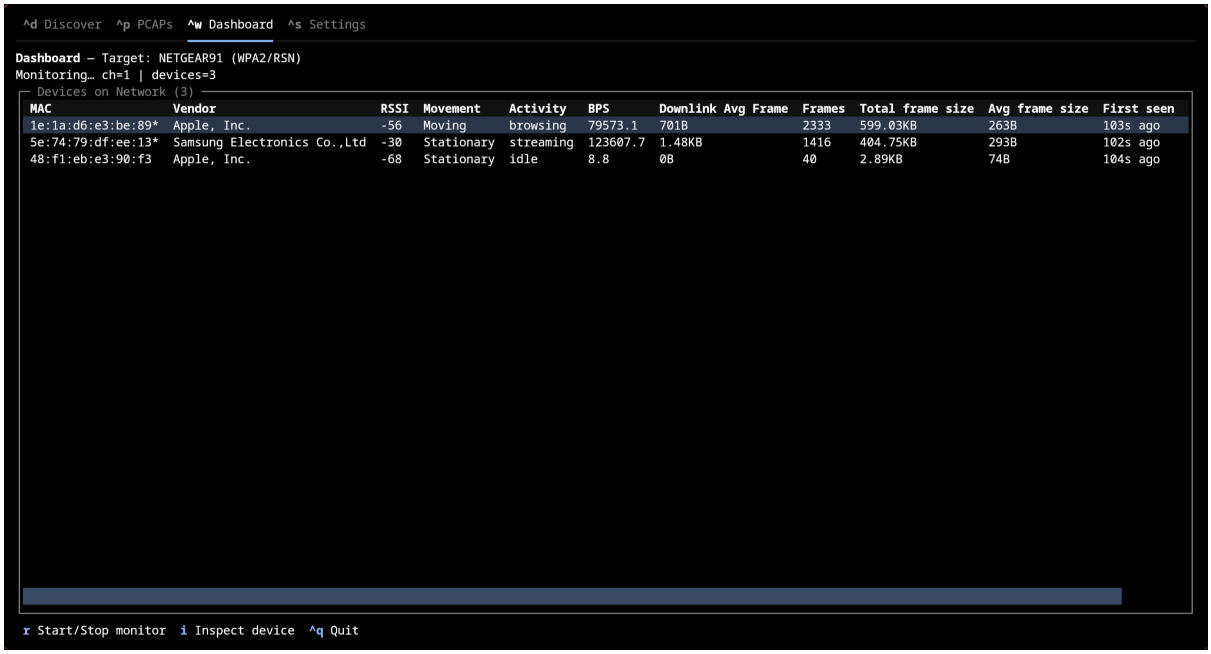


Figure 6.2: Dashboard page shows all detected devices within the selected network. The table provides an overview of each device, including MAC address, vendor, movement state, activity status, transmitted frames, average frame size and timestamps of first and last detection.

Furthermore, offline analysis using previously recorded PCAP files yielded results consistent with those obtained during live monitoring. The same parsing and aggregation pipeline was successfully applied to recorded data, demonstrating the reusability and consistency of the processing architecture across different data sources.

The results indicate that the proposed system is capable of reliably capturing, processing and presenting relevant Wi-Fi observations. The implementation provides a practical solution for terminal-based situational awareness, enabling users to monitor nearby wireless networks and associated devices in an efficient and structured manner.

7

Conclusion

This project investigated how much information can be extracted from devices present on a wireless network using a fully passive monitoring approach. The results demonstrate that, even when traffic content is encrypted and privacy mechanisms such as MAC address randomization are used, meaningful information about connected devices can still be inferred from observable IEEE 802.11 frame characteristics.

The implemented system successfully mapped local wireless environments by grouping APs into ESSs and pairing hidden SSIDs with visible hardware. Furthermore, the system was able to identify connected devices and their vendors, track movement via RSSI variations, and infer activity patterns based solely on passively captured traffic. Activity classification achieved an F1-score of approximately 89%, indicating that traffic metadata such as timing, rate, and packet size provide strong indicators of device behavior.

Altogether, the results confirm that passive wireless monitoring can offer significant insight into network usage and connected devices without requiring active interaction or device cooperation. These findings highlight both the capabilities and the privacy implications of passive Wi-Fi monitoring, emphasizing the need to carefully balance network analysis techniques with user privacy considerations.

7.1 Limitations of the System

This system is limited by the characteristics of the wireless environment, client behavior, and passive observation. First, the quality of detection is dependent on environmental factors such as signal attenuation caused by walls and floors, interference from neighboring networks, and channel congestion. These conditions may reduce the reliability of frame capture and lead to less stable measurements across different locations, thereby limiting the external validity of the results.

Second, system performance decreases as the number of nearby devices increases. A large number of active clients leads to higher levels of frame collisions and channel contention, while simultaneously increasing processing and storage demands. On hardware with limited resources, this may result in packet loss, delayed updates and reduced detection accuracy. Furthermore when monitoring is conducted on only a single channel at a time, or through channel hopping, a trade off arises between coverage and temporal resolution.

Third, fingerprinting and vendor classification relies heavily on association related management traffic, such as probe and association frames. Devices that are already connected and largely inactive may generate very few of such frames, while certain devices may intentionally reduce transmission due to power saving mechanisms. Additionally, modern

privacy preserving techniques, including MAC randomization, may obscure persistent device identities and contribute to either an overestimation or underestimation of device counts. Consequently, the absence of observed association requests should not be interpreted as conclusive evidence of the absence of a device.

Finally, several practical limitations must also be considered. Support for monitor mode varies depending on the network adapter, drivers and firmware specific behavior may influence the completeness of captured traffic.

Taken together, these limitations indicate that the system should be regarded as an approximate situational awareness tool rather than a definitive ground truth measurement system.

7.2 Future Work

Future work could focus on improving both the accuracy and the scope of the proposed monitoring system. In particular, more advanced machine learning models and larger datasets could be used to refine activity classification and distinguish between a wider range of user behaviors. Existing studies focusing exclusively on activity classification have shown that, with carefully selected machine learning models and well preprocessed data, classification accuracies of up to 97% can be achieved [36], as shown in Figure 7.1.

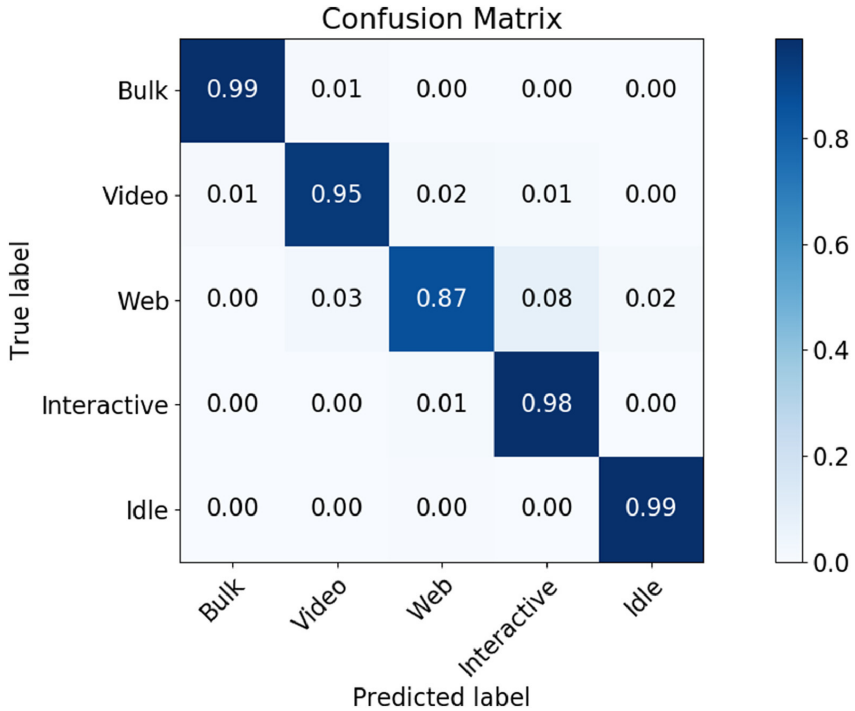


Figure 7.1: Example of online activity classification performance using machine learning. Reproduced from Labayen et al. [36] under the Creative Commons Attribution 4.0 International (CC BY 4.0) license.

Additional features derived from management and data frames, as well as longer observation periods, could further improve device fingerprint stability and robustness. and be used for better classification of devices vendor and type as mentioned in 6.3. Evaluating

the system in more diverse and crowded wireless environments, and in larger networks, where devices can roam between a large number of APs within an ESS, would provide additional insight into its scalability and real world applicability. Future studies should further investigate privacy preserving countermeasures and their effectiveness against passive inference techniques, in order to better balance network analysis capabilities with user privacy.

7.3 Ethical and Privacy Considerations

During the development of the project privacy and ethics had to be taken into account, as mentioned in Section 1.3 the developed tool could be used to breach the privacy of unaware bystanders. Trust in the digital infrastructure is therefore at risk, and the intended privacy is not being met. Similarly, freedom of speech can be affected by the weakening of privacy if, for example, a journalist's or activist's Wi-Fi activity can be gathered without their knowledge.

The ethical aspects that had to be taken into consideration during the project were to not infringe the autonomy and integrity of others. Testing had to be done in controlled environments where we made sure only our own devices were being monitored. To do this we used a designated AP that only we were connected to, furthermore in the saving of PCAP files only activity captured on the specified network is saved and not all data captured. Still, during tests we needed to monitor the results to be sure only our own packets and devices were being observed. The project should also be beneficial for the knowledge in security of encrypted Wi-Fi metadata fulfilling the purpose written about in Section 1.2.

The end product succeeds with this purpose, but the product has a risk to be used for harm as mentioned in Section 1.3. The product can be used with malicious intent, but due to the overall positive effects of spreading awareness of what is possible, and the positive use cases mentioned in Section 1.3, we conclude the project to be more beneficial than harmful.

7.4 Usage Of AI

In this project AI was used as a tool to help generate code, images and proofread text in the report. ChatGPT and Copilot were the main tools for code generation and the project text, VDRAW was used for AI image generation and teXgpt (Overleaf AI assistant) was used for more grammar checking and spelling. All AI generated content was validated to ensure no misleading information was generated.

Bibliography

[1] M. Dey and R. Jambhale. "IoT Statistics By Revenue, Market Size, Private Networks, Applications, Users and Facts". [Online]. Available: https://electroiq.com/stats/iot-statistics/.

[2] P. Bheevgade, C. Saha, R. Nath, S. Dabhade, H. Barot, and S. Junare, "The Rise of Public Wi-Fi and Threats", in Nov. 2023, ISBN: 978-981-99-5090-4. DOI: 10.1007/978-981-99-5091-1_13.

[3] S. Singh and S. Kumar, "The times of cyber attacks", *Acta Technica Corviniensis-Bulletin of Engineering*, vol. 13, no. 3, pp. 133–137, 2020.

[4] M. Shen, Y. Liu, S. Chen, L. Zhu, and Y. Zhang, "Webpage Fingerprinting using Only Packet Length Information", in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8761167.

[5] D. M. A. F. Al-Husainy, *MAC Address as a Key for Data Encryption*, 2013. arXiv: 1311.3821 [cs.CR]. [Online]. Available: https://arxiv.org/abs/1311.3821.

[6] M. L. Ali, S. Ismat, K. Thakur, A. Kamruzzaman, Z. Lue, and H. N. Thakur, "Network Packet Sniffing and Defense", in *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, 2023, pp. 0499–0503. DOI: 10.1109/CCWC57344.2023.10099148.

[7] T. Watteyne, A. Mehta, and K. Pister, "Reliability through frequency diversity: Why channel hopping makes sense", New York, NY, USA: Association for Computing Machinery, 2009, ISBN: 9781605586182. DOI: 10.1145/1641876.1641898. [Online]. Available: https://doi.org/10.1145/1641876.1641898.

[8] C. Chaudet, E. Fleury, I. G. Lassous, H. Rivano, and M.-E. Voge, "Optimal positioning of active and passive monitoring devices", New York, NY, USA: Association for Computing Machinery, 2005, ISBN: 159593197X. DOI: 10.1145/1095921.1095932. [Online]. Available: https://doi.org/10.1145/1095921.1095932.

[9] J. Martin et al., "A study of MAC address randomization in mobile devices and when it fails", *CoRR*, vol. abs/1703.02874, 2017. arXiv: 1703.02874. [Online]. Available: http://arxiv.org/abs/1703.02874.

[10] C. Matte and M. Cunche, "Spread of MAC address randomization studied using locally administered MAC addresses use historic", Inria Grenoble Rhône-Alpes, Research Report RR-9142, Jan. 2018. [Online]. Available: https://inria.hal.science/hal-01682363.

[11] J. D. Roth, J. Martin, and T. Mayberry, "A graph-theoretic approach to virtual access point correlation", in *2017 IEEE Conference on Communications and Network Security (CNS)*, 2017, pp. 1–9. DOI: 10.1109/CNS.2017.8228645.

[12] M. Thankappan, H. Rifà-Pous, and C. Garrigues, "Multi-Channel Man-in-the-Middle attacks against protected Wi-Fi networks: A state of the art review", *Expert Systems with Applications*, vol. 210, p. 118 401, 2022, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.118401. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422015093.

[13] n.a. "Monitoring Probe Request ssids". [Online]. Available: https://www.nzyme.org/knowledge/wifi/monitoring-probe-request-ssids.

[14] R. Rusca, F. Sansoldo, C. Casetti, and P. Giaccone, "What WiFi Probe Requests Can Tell You", in *Proc. IEEE 20th Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, USA, 2023, pp. 1–6. DOI: 10.1109/CCNC51644.2023.10060447.

[15] P. Robyns, B. Bonné, P. Quax, and W. Lamotte, "Noncooperative 802.11 MAC Layer Fingerprinting and Tracking of Mobile Devices", *Security and Communication Networks*, vol. 2017, no. 1, p. 6 235 484, 2017. DOI: https://doi.org/10.1155/2017/6235484. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2017/6235484. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2017/6235484.

[16] Wireless Broadband Alliance, "A Way Through MAC Randomization: Wi-Fi Devices Identification", Wireless Broadband Alliance, White Paper, Feb. 2022, Accessed: 2026-03-27. [Online]. Available: https://mentor.ieee.org/802.11/dcn/22/11-22-0653-00-0000-2022-march-wba-whitepaper-re-device-identification.pdf.

[17] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 8th. Harlow, UK: Pearson Education Limited, 2021, ISBN: 9781292405469.

[18] P. Dimou, J. Fajfer, N. Müller, E. Papadogiannaki, E. Rekleitis, and F. Střasák, "Encrypted Traffic Analysis: Use Cases & Security Challenges", European Union Agency for Cybersecurity (ENISA), Athens, Greece, Technical Report, Nov. 2019. [Online]. Available: https://www.enisa.europa.eu/publications/encrypted-traffic-analysis.

[19] M. Conti, Q. Li, A. Maragno, and R. Spolaor, "The Dark Side(-Channel) of Mobile Devices: A Survey on Network Traffic Analysis", *IEEE Communications Surveys & Tutorials*, vol. PP, Aug. 2017. DOI: 10.1109/COMST.2018.2843533.

[20] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing Android Encrypted Network Traffic to Identify User Actions", *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 162–175, 2016. DOI: 10.1109/TIFS.2015.2478741.

[21] F. Zhang, W. He, X. Liu, and P. G. Bridges, "Inferring users' online activities through traffic analysis", in *Proceedings of the Fourth ACM*, ser. WiSec '11, Hamburg, Germany: Association for Computing Machinery, 2011, ISBN: 9781450306928. DOI: 10.1145/1998412.1998425. [Online]. Available: https://doi.org/10.1145/1998412.1998425.

[22] Q. Wang, A. Yahyavi, B. Kemme, and W. He, "I know what you did on your smartphone: Inferring app usage over encrypted data traffic", in *2015 IEEE Conference on Communications and Network Security (CNS)*, 2015, pp. 433–441. DOI: 10.1109/CNS.2015.7346855.

[23] A. Mahmood, R. Exel, H. Trsek, and T. Sauter, "Clock Synchronization Over IEEE 802.11—A Survey of Methodologies and Protocols", *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 907–922, 2017. DOI: 10.1109/TII.2016.2629669.

[24] D. Ross, "Securing IEEE 802.11 Wireless LANs", Ph.D. dissertation, Queensland University of Technology, Brisbane, Australia, Jun. 2010.

[25] S. N. Sari, M. F. Edy Purnomo, and C. Miranda, "Performance Comparison Analysis of Wireless Access Point (WAP) and Virtual Access Point (VAP) Based on Mikrotik at SD Negeri 01 Jambearjo", in *2022 11th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, 2022, pp. 243–248. DOI: 10.1109/EECCIS54468.2022.9902900.

[26] A. W. Omer and T. H. Ali, "Dealing with the Outlier Problem in Multivariate Linear Regression Analysis Using the Hampel Filter", *KJAR*, vol. 10, pp. 1–17, Feb. 2025. DOI: https://doi.org/10.24017/science.2025.1.1.

[27] Pearson, R.K., Neuvo, Y., and J. Astola, "Generalized Hampel Filters", *EURASIP J. Adv. Signal Process*, p. 87, Aug. 2016. DOI: https://doi.org/10.1186/s13634-016-0383-6.

[28] Python Software Foundation. "What is Python? Executive Summary". Accessed: 2026-04-26. [Online]. Available: https://www.python.org/doc/essays/blurb/.

[29] Textualize. "What is Textual?" Accessed: 2026-05-17. [Online]. Available: https://textual.textualize.io/#what-is-textual.

[30] Textualize. "Textual". Accessed: 2026-05-17. [Online]. Available: https://github.com/Textualize/textual/blob/main/README.md#textual.

[31] Scapy Community. "What is Scapy?" Accessed: 2026-05-17. [Online]. Available: https://scapy.net/.

[32] Philippe Biondi. "Introduction". Accessed: 2026-05-17. [Online]. Available: https://scapy.readthedocs.io/en/latest/introduction.html.

[33] The Scapy community. "Scapy network stack". Accessed: 2026-05-18. [Online]. Available: https://scapy.readthedocs.io/en/latest/routing.html.

[34] Aircrack-ng Project. "Airmon-ng". Accessed: 2026-04-26. [Online]. Available: https://www.aircrack-ng.org/doku.php?id=airmon-ng.

[35] S. Frankel, B. Eydt, L. Owens, and K. Scarfone, "Establishing Wireless Robust Security Networks: A Guide to IEEE 802.11i", National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST Special Publication 800-97, Feb. 2007, 162 pages.

[36] V. Labayen, E. Magaña, D. Morató, and M. Izal, "Online classification of user activities using machine learning on network traffic", *Computer Networks*, vol. 181, p. 107 557, 2020, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2020.107557. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620312081.