



University of
Zurich^{UZH}

Optimizing MAC Scavenger's Data Streaming Pipeline

*Künzler Fabian
Zurich, Switzerland
Student ID: 15-941-842*

Supervisor: Bruno Rodrigues, Eder Scheid
Prof. Dr. Burkhard Stiller
Date of Submission: March 21, 2021

Abstract

Passive wireless tracking is used to track an individual with a wireless capable device over an area covered by a network. This technology could help to monitor occupancy levels in public transport systems and buildings. To avoid being tracked, countermeasures, such as MAC-Randomization, have been implemented on WiFi capable devices. A thesis at the University of Zurich analyzed these countermeasures and presented a system called MAC-Scavenger which circumvents them.

This thesis analyses the basic background concepts of wireless networks and passive wireless tracking. It investigates the MAC-Scavenger and its data gathering process in detail and derives requirements for an improved system. The main requirement is to simplify the existing MAC-Scavenger and therefore to make the passive wireless tracking technology available for non-computer experts. Another requirement, inferred from studies in related research, includes the secure handling of the gathered data. Based on these requirements the Web-Scavenger is implemented and documented in the scope of this thesis. The documentation shows the high level architecture as well as key implementation parts. In the end the requirements are evaluated against the implemented system.

Zusammenfassung

Das passive drahtlose "tracking" wird verwendet, um eine Person mit einem wireless-fähigen Gerät über einen von einem Netzwerk abgedeckten Bereich zu verfolgen. Dies könnte helfen, die Belegung in öffentlichen Verkehrsmitteln und Gebäuden zu überwachen. Um zu vermeiden, verfolgt zu werden, wurden Gegenmassnahmen wie MAC-Randomisierung, auf WiFi-fähigen Geräten eingeführt. Eine Masterarbeit an der Universität Zürich analysierte diese Gegenmassnahmen und stellte ein System namens MAC-Scavenger vor, das diese umgeht.

Diese Bachelorarbeit analysiert die grundlegenden Konzepte von drahtlosen Netzwerken und passivem drahtlosem Tracking. Sie untersucht weiter den MAC-Scavenger und dessen Datenerfassungsprozess im Detail und leitet daraus Anforderungen an ein verbessertes System ab. Die Hauptanforderung besteht darin, das bestehende MAC-Scavenger System zu vereinfachen und damit die passive drahtlose Ortungstechnologie auch für Nicht-Computer-Experten zugänglich zu machen. Weitere Anforderungen, die aus Studien in verwandten Forschungsgebieten abgeleitet werden, sind der sichere Umgang mit den gesammelten Daten. Basierend auf diesen Anforderungen wird der Web-Scavenger im Rahmen dieser Arbeit implementiert und dokumentiert. Die Dokumentation zeigt sowohl die High-Level-Architektur als auch die wichtigsten Teile der Implementierung. Abschliessend werden die Anforderungen anhand des implementierten Systems evaluiert.

Acknowledgments

I would like to thank my supervisors Dr. Bruno Rodrigues, Eder Scheid and Prof. Dr. Burkhard Stiller at the Communication Systems Group of the University of Zurich. Especially, I want to thank Dr. Bruno Rodrigues for his constant support and his feedback during our weekly meetings.

My thanks go also to my family for their patience and the final proofreading at the end of my thesis.

Contents

Abstract	i
Acknowledgments	v
1 Introduction	1
1.1 Motivation	1
1.2 Thesis Goals	2
1.3 Methodology	2
1.4 Thesis Outline	3
2 Background	5
2.1 WLAN	5
2.1.1 IEEE 802.11 standard	6
2.1.2 Dataframes	13
2.2 Passive and active Scanning	18
2.3 MAC-Scavenger	19
2.3.1 Goal of MAC-Scavenger	19
2.3.2 Implementation and Deployment	20
2.4 Related Work	21
2.4.1 Passive WiFi Monitoring of the Rhythm of the Campus	21
2.4.2 Tracking Unmodified Smartphones Using Wi-Fi Monitors	22
2.4.3 Correlating WiFi Activity to Human Occupancy	23
2.4.4 Signals from the Crowd	23

2.4.5	Bluetooth	24
2.4.6	Discussion	25
3	Prototype Design	27
3.1	Requirements	27
3.1.1	Deployment	28
3.1.2	Operation	28
3.1.3	Security	29
3.2	Web-Scavenger Approach	30
3.3	Architecture	30
3.3.1	Database	33
3.3.2	Deployment and Operation	35
3.4	Features	38
3.5	Implementation	39
3.5.1	Technologies	39
3.5.2	Website	40
3.5.3	Desktop Application	50
4	Evaluation	53
4.1	Requirement based evaluation	53
4.1.1	Deployment requirements	54
4.1.2	Operation requirements	56
4.1.3	Security requirements	58
5	Summary and Conclusions	61
5.1	Future Work	62
	Bibliography	62
	Abbreviations	67

<i>CONTENTS</i>	ix
List of Figures	68
List of Tables	70
List of Listings	71
A Web-Scavenger Githlab Repository	75
B Documentation	77

Chapter 1

Introduction

1.1 Motivation

Tracking the behavior of individuals becomes increasingly important within a marketing context to improve the offer of products and services. Not only are government agencies with three letters interested in the whereabouts of persons, so are companies. Tracking a person over a defined area such as a public transport system [34] or a university campus [21, 45, 47] could give the tracker essential information about the individuals. This information can be used for crowd control, for city planners [21], for emergency services, for lowering energy consumption of buildings [45] and also to tailor advertisement to future customers [7].

An easy and cost effective way to track individuals in our digital society is by sensing the packets emitted by our personal, WiFi capable devices. The gathered information often tells much more about the users than just their location [5].

A study from the University of Illinois focused on estimating the trajectory of vehicles along a highly trafficked road. In the end they compared the estimated trajectory against GPS ground truth. They discovered that an estimation with Probe Request Frames is feasible, but they did not provide a way to deploy the Monitors and the system for average users [33].

A study from the MIT investigated if the human occupancy of campus buildings can be estimated through WiFi traffic. They discovered that WiFi signals reveal human occupancy patterns. However, the researchers did not offer a straightforward program which made the technology available to the wider public [45].

To make the technology available for a wider public, the usage must become easier. A simple deployment and operation allows users to use the system independently, which ultimately reduces cost and allows everyone to profit.

To the best of my knowledge none of the existing systems provide a "plug-and-play" deployment, meaning that a huge effort needs to be put in to set up and configure the

tracking system in the desired location. This means that average users can not profit from the advantages that passive wireless tracking provides.

This thesis tries to change that. The goal is to improve the existing MAC-Scavenger system [6] by automating the deployment in a way that the WiFi gathering devices only need to be physically put in the respective place with a minimum of manual configuration steps. This would make the technology broadly available.

1.2 Thesis Goals

The first objective of this thesis is to give an introduction into passive WiFi tracking and discuss related work. However, the key objective of this thesis is to improve, respectively to automate, the deployment of the existing MAC-Scavenger system [6]. The list below summarizes the goals.

- Introduce theoretical foundations of passive WiFi tracking.
- Discuss related work with focus on deployment.
- Automate the deployment of the MAC-Scavenger such that fewer manual steps are needed.
- Simplify deployment so that an average user can deploy the MAC-Scavenger.
- Simplify operation such that non computer experts can operate the MAC-Scavenger.

In summary, the aim of this thesis is to improve the existing MAC-Scavenger system and not to add new features to the MAC-Scavenger. Therefore the code of the MAC-Scavenger will be altered as little as possible.

1.3 Methodology

This thesis follows an exploratory approach and provides an investigation into how the current MAC-Scavenger system can be improved. Through qualitative data and bibliographic research the existing system is analysed. This knowledge is then used to derive requirements for the practical implementation of an improved system. The following steps describe the process of this bachelor's thesis to achieve its goals:

- Analysis of passive wireless tracking: In the first step, comprehensive knowledge of wireless network communication is obtained. Furthermore, passive wireless tracking, MAC randomization and the MAC-Scavenger system were analyzed.

- Definition of requirements: In a second step, the MAC-Scavenger was configured to further understand the deployment and operation process. From these experimental trials requirements for an improved system were derived. These requirements aimed to improve the MAC-Scavenger such that the goals set in section 1.2 can be achieved
- Implementation: The implementation consists of the high level architecture design and the actual programming. In this step the prototype was created and the website and the Desktop Application were implemented. Additionally, the prototype was deployed to a server for the evaluation phase.
- Evaluation: In the last step the previously defined requirements were qualitatively evaluated against the developed system.

1.4 Thesis Outline

Chapter 2 explores the core concepts of WiFi communication and networks with a strong focus on active scanning. It further discusses related work in the area of passive wireless tracking. **Chapter 3** presents the Web-Scavenger prototype. While it starts with the derived requirements, it explains the high level architecture later. The chapter closes by showing implementation details and presenting some major features. In **Chapter 4** the developed system is assessed by comparing the state of the system to the desired state, namely the requirements. **Chapter 5** summarizes the thesis and shows areas where future work can be conducted.

Chapter 2

Background

2.1 WLAN

The first known deployed Wireless LAN, known as AlohaNET, was deployed from 1970 to 1976 in Hawaii to connect the islands of the Hawaiian chain [23]. It took another 20 years until the first commercial WLAN product called WaveLAN was brought to market in the early nineties [22].

Wireless communication provides its users with many advantages such as mobility and the simplicity of extending an existing network [6].

On the other hand, wireless communication's most significant disadvantage is data transmission over the shared medium air. This could lead to interference from electromagnetic waves emitted, for example, by electrical locomotives or other transmitting wireless stations [40].

To combat the multiple access problem, respectively, the interference from other stations, procedures like Space Division Multiple Access (SDMA), Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA), and Code Division Multiple Access (CDMA) were developed. SDMA states that when two transmitters are far away from each other, their signals cannot interfere. In TDMA, each sender has a specific time slot on a frequency, whereas in FDMA, each sender has its frequency during the whole transmission. CDMA, in the end, is a protocol to encrypt data such that the data packages can be stacked on top of each other [40]. Another disadvantage related to the used, shared medium is that anyone within reach of the signal can manipulate or tap into it. Finally, the last drawback have been the relatively slow data rates compared to wired connections [40].

With technologies and protocols advancing, the advantages of wireless communication over time outweighed the disadvantages, which led to WLAN becoming an elementary part of today's society.

The three leading wireless solutions for networks are the IEEE 802.11, HiperLAN, and HomeRF. The 802.11 standard is the most popular one [40].

2.1.1 IEEE 802.11 standard

The IEEE 802.11 standard is a wireless standard defined by the IEEE and was developed 1997 [40]. The Institute for Electrical and Electronical Engineers, abbreviated with IEEE, is a board of experts from the fields of electrical engineering, computer science, medicine and many more which roots go back to 1884 [1].

The problem which arose with the release of the first wireless products was that there existed many different standards. If a customer had once committed to a standard he could not interchange a device with the counterpart of a different company because they used another standard. Therefore the main goal of the Institute is to develop and publish standards for network technologies [40].

The protocols used in the 802.11 standard are all standardized by the IEEE committee. Furthermore, the committee assigns the unique MAC address with which WiFi capable devices can be identified, to companies [6, 40]. The importance and the exact functionality of the MAC address will be explained in the OSI model section. As already discussed, the main problem of wireless communication is that the air is a shared medium that every wireless capable station can access. The 802.11 standard uses a decentralized media access protocol called CSMA/CA to solve this problem. CSMA/CA is the acronym for Carrier Sense Multiple Access Collision Avoidance [40]. In CSMA/CA, every station, like a mobile phone or laptop, listens to the carrier medium, at a predefined frequency range. Once they sense an empty medium for a certain period (DIFS), the stations start a randomly generated back-off time. The first station which back-off time reached zero will start sending while the other stations sense a busy medium and stop their back-off time such that a collision of data can be avoided [8]. This is different from the CSMA/CD, which is used in the wired Ethernet solution where collision is allowed [40].

Generally, three different types of media can be used to send data. These media types are infrared, radio frequencies, and microwaves, of which radio frequencies are the most common ones for wireless solutions today [36]. These electromagnetic waves are divided into different bands, which have a predefined frequency range [6]. The 802.11 standard uses the band with a frequency range of 2.4 GHz [40]. This band is called the ISM (Industrial, Scientific, Medical) and is free to use [17, 40].

Over the years, many improvements have been made in the technological field of wireless communication and computer science, hence the original standard needed to be extended over time. A summary of the essential improvements can be found in Table 2.1.

Protocol	Improvements	Launch Year
802.11a	Extends the existing 802.11 standard for the 5 GHz frequency band by using the OFDM procedure [40, 12].	1999 [12]
802.11b	Defines a Bit transmission layer also called physical layer which works with the DSSS procedure to achieve higher data rates [40, 12].	1999 [12]
802.11c	”Provides required information to ensure proper bridge operations, which is required when developing access points” [12].	2001 [12]
802.11d	Enables international roaming so that WiFi cards adapt to the local regulations [40, 12].	2001 [12, 40]
802.11e	Extends the MAC layer to enable performance improvements and the implementation of Quality of Service [40].	2005 [40]
802.11f	Defines an Inter Access Point Protocol (IAPP) to standardize communication between Access points [40, 12].	2003 [40, 12]
802.11g	Introduces a wireless high speed physical layer which works on the ISM band and without TPC and DFS [40].	2003 [12, 40]
802.11h	Extends the 802.11a standard by introducing TPC and DFS. These extensions are necessary to fully exploit the 802.11a potential [40, 12].	2003 [12, 40]
802.11i	”Enhances WLAN security to replace the previous security specification Wired Equivalent Privacy (WEP)” [12].	2004 [40, 12]
802.11j	Enables the wireless LAN to run on higher frequency bands, which was introduced for the Japanese market [12].	2004 [12]
802.11n	Improves the MAC and PHY Layer to enable a high throughput [40]. It also introduced security improvements [12].	2009 [12, 40]
802.11p	Improvements on the Physical Layer to optimize communication between devices in vehicular environments [40, 12].	2010 [12]
802.11r	Improves the roaming, respectively the hand-over between access points [40, 12].	2008 [12]
802.11s	Introduces mesh networks which enable a wireless connection between access points [40, 12].	2011 [12]

Table 2.1: 802.11 Protocol Family

Network architecture

This section focuses on the two network architecture solutions that are feasible with the previously described IEEE 802.11 standard. There are two types of network architecture within a wireless LAN. The first solution is the **Ad-hoc** network architecture. This architectural form is often used by the wireless LAN technology named Bluetooth, which is described in section 2.4.5 [44]. The second form is the **infrastructure architecture** which is most common in 802.11 and HiperLAN2 standards [44].

Ad-hoc Network: This architectural form describes a simple structure on how to connect multiple wireless capable devices. In its simplest form, only two stations with WLAN cards are necessary. These two stations will then directly communicate with each other

by exchanging data packages over specific frequency channels [40, 44]. To communicate, the stations need to be in reach of each other's radio range or a station must be able to forward traffic to the final station [44]. Since there is no central Access Point (AP), that coordinates the medium access, the stations themselves manage it. This means a higher complexity at the station's ends [44].

The Ad-hoc network depicted in Figure 2.1 shows three stations that form an IBSS. A Basic Service Set or BSS defines a set of stations within the same radio coverage [44]. In an infrastructure network, such a BSS can be seen as one cell of the entire cellular network [40]. An IBSS or Independent BSS is not dependent on connections outside its cell and therefore works independently. The most significant advantages of an Ad-hoc network is its flexibility, since adding or removing a station is simple [44, 40].

Infrastructure Network: Stations in an infrastructure network do not communicate directly with each other but indirectly over Access Points (AP) [44]. Access Points are unique stations that, most of the time, are at the center of a BSS. This leads to an increased reach of a BSS of up to 100 meters [40]. Furthermore, the Access Point controls the medium access and connects a BSS via the distribution system (DS) with other wired or wireless networks. Therefore multiple BSS can create one network called Extended Service Set, without all stations being within each other's radio range [44, 40]. A distribution system connects the different BSS by connecting the respective APs. It can either be a wired or wireless distribution system [40]. APs might also serve as wireless Gateways, which enable wireless internet access. If an AP has a routing function and a DHC protocol (DHCP) and a NAT, it can be connected to a DSL modem, allowing all stations within the ESS to access the internet by being routed over their respective AP [40].

In Figure 2.2 an infrastructure network is depicted. The BSS overlap each other. This way, the ESS provides coverage for a larger area than a single BSS could do. The overlapping BSS have another advantage since they enable roaming. Roaming describes the ability of a mobile station to move within the network and seamlessly connect to a new AP [40]. An AP is responsible for sending beacon frames, which are described in more detail in section 2.2. The main goal of a beacon frame is to announce the presence of an AP. This way, the client station can switch when the signal is too weak or the data rate is too low, which allows a seamless change inside the network [40]. The scanning of the station or, more simply put, the search for an AP can be initiated by a poor signal or a low data rate. Depending on the brand of the device, some stations are always searching for new APs [40].

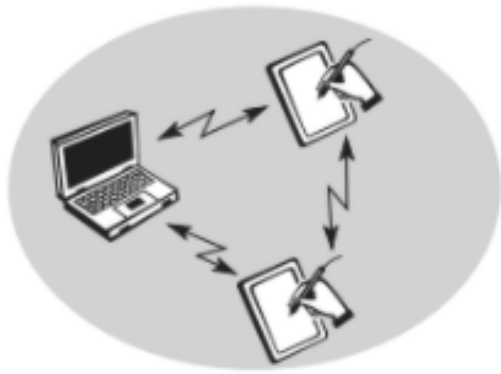


Figure 2.1: Ad-hoc Network. Figure by [44].

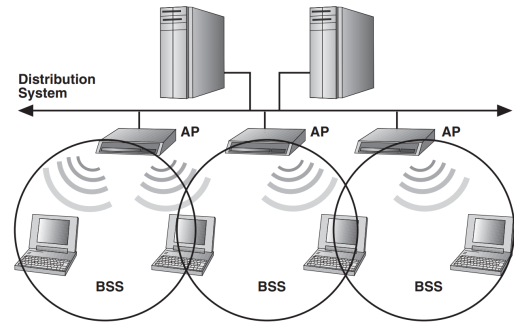


Figure 2.2: Infrastructure Network. Figure by [40].

OSI model

As already mentioned, in the early days of wireless communication there existed many different standards which made communication between stations from different producers impossible [40]. Hence the Open Systems Interconnection Reference Model (OSI model) has been established. The OSI model is divided into 7 layers which all provide services to the upper layer and use the services from the lower layer [40]. Each of these layers follows a set of rules named protocols [40]. Only the MAC and the PHY Layer protocols are defined by the 802.11 standard [44].

The OSI model is depicted in Figure 2.3. Two stations communicate by passing data down in their station to the physical layer which then modulates the data onto the carrier medium. In the receiving station the physical layer demodulates the signal and pushes it up to the respective Peer Layer [40]. Only layers on the same level can communicate with each other. Those layer pairs are called Peers. Therefore all Peers need to have the same protocols in place for the communication to work [40].

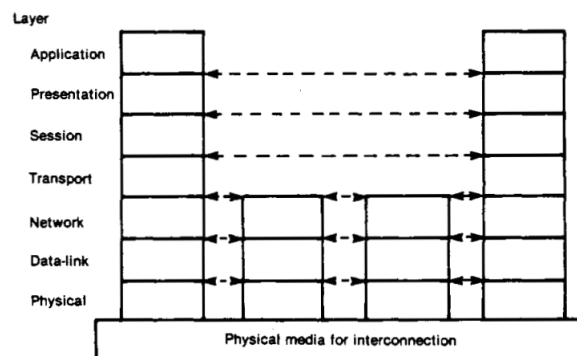


Figure 2.3: OSI model. Figure by [11]

The MAC Layer, which is a sub-layer of the Data Link Layer [41], and the PHY Layer are the most important layers for this thesis but for completeness a short overview over

the other layers is given in the Table 2.2 below.

Layer	Functionality
Application Layer	"[The Application Layer] as the highest layer of OSI does not provide services to any other layers" [11]. It includes information about processes used by the network users [40].
Presentation Layer	It includes functions to connect network resources such as printers. Additionally, it decides how the data is presented to Application Layer, which includes the length of a data frame [40].
Session Layer	This layer establishes and ends a connection to another device and structures the data independently from the Presentation Layer [40].
Transport Layer	It transfers data from the network layer to end systems [11]. Moreover, it splits data into smaller fragments when sending and puts them together when receiving [40].
Network Layer	The Network Layer provides routing of the data through networks and sub-networks by assigning source and final address [40, 11]
Data Link Layer	This layer has the task to ensure a smooth data transfer by detecting errors through a check sum [40, 11]. It consists of the MAC sub-layer, defined by the 802.11 standard, and the Logical Link Control sub-layer [41].
Physical Layer	"[It] provides the mechanical, electrical, functional, and procedural standards to access the physical medium" [11].

Table 2.2: OSI model layers

Physical Layer: This layer is divided into two sub-layers. The first sub-layer is the Physical Layer Convergence Protocol (PLCP). It provides a Service Access Point (SAP) to the MAC Layer, which is independent from the transmission technology [44, 40]. The PLCP is responsible to transform the data into a format which can be sent over the shared medium [40]. The PLCP additionally provides the Clear Channel Assessment signal (CCA), which the MAC Layer can use to check, if the shared medium is idle [44]. Moreover, it enables the exchange of management information between the MAC and the PMD Layer [40].

PMD stands for Physical Medium Dependent and is the other sub-layer of the PHY Layer. The PMD sub-layer provides the hardware to send and receive data over the shared medium. To send packets, the PMD brings the data into the format compatible with the respective transmission technology and then modulates it onto the carrier signal [40]. The modulation can be split into digital modulation, which is the transformation of digital data into an analog signal, and analog modulation, which shifts the frequency of the analog signal up to the carrier frequency [44]. The frequency shift, respectively the analog modulation, is necessary for the signal to travel a longer distance [40]. Demodulation is the opposite way at the receiver's end to transform the analog signal into digital data.

The IEEE 802.11 standard defines three different standards for the physical layer [44].

Each of these standards uses its own PMD to modulate the information onto the carrier signal. The three standards are DSSS, FHSS and infra-red [44]. DSSS and FHSS are spread spectrum solutions, which gives the signal the advantage of a certain degree of protection against eavesdropping and interference [40].

MAC Layer: As discussed above, signals are transmitted over a shared medium in wireless communication which can cause several problems. A signal could suffer interference from other signals emitted by other wireless stations. The consequence of conflict heavy communication is a loss of performance [40]. Therefore, the main goal of the MAC Layer is to control the medium access and the frame creation [40, 44]. This includes functionality to sense the carrier medium, to confirm the reception of a message and to split long data frames, which are likely to suffer from interference, into smaller frames [40]. Additional tasks include support for roaming, authentication and power conservation [44].

The 802.11 standard defines two main access mechanisms: The Distributed Coordination Function (DCF) and the Point Coordination Function (PCF). DCF is a decentralized approach to achieve the main goal of the MAC Layer and can be used both in infrastructure and ad-hoc networks. PCF on the other hand is a centralized approach which means it is only available in infrastructure networks [40, 44]. Both DCF and PCF are also referred to as Distributed Foundation Wireless Medium Access Control (DFWMAC) plus the respective coordination function at the end [44].

There exist two versions of the DFWMAC-DCF mechanism: An original version which uses a simple CSMA/CA medium access procedure and an extension of the original one [44]. The original DFWMAC-DCF using CSMA/CA is depicted in Figure 2.4. Once the PHY Layer senses an idle medium all stations wait for a certain Interframe Space (IFS). There exist multiple IFS with different priorities. The higher the priority the shorter the time period defined by an IFS. DIFS stands for Distributed Interframe Space. During this time, stations with higher priority frames, such as management frames, can access the medium [40]. Throughout the DIFS the PHY Layer checks if the medium is busy or not, by using the CCA signal. If the PHY has not registered any frames during the DIFS, the station starts counting down a randomly generated time, called Backoff-time [40, 44]. This is necessary to avoid multiple stations accessing the medium after a DIFS, which could cause collisions. When the first station reaches nil it starts sending its data [40].

The other stations in the system receive the data frame as well. All stations except the sending and the receiver can update their Network Allocation Vector (NAV) with the information extracted from the data frame header [40]. The NAV is an essential part of the carrier sense functionality of the MAC Layer, called virtual carrier sense function [40]. The vector stores the time the station is prohibited to access the medium. Each station has its own NAV, which is updated every time a frame is received by the station [40]. The virtual carrier sense function ensures that ongoing operations, where the medium is idle but management frames still need to be exchanged, are not interrupted [40].

Once the data is transmitted the receiver waits for a Short Interframe Space (SIFS), which is the shortest IFS. If the check sum is correct it answers to the sender with a 14 Byte long Acknowledgment frame (ACK) to confirm the reception of the data [40]. If the sender does not receive an ACK, the data is automatically sent again [44]. ACK frames are only sent when an unicast transmission took place [44, 40].

Thereafter, all stations sense with their PHY Layers an idle medium. Furthermore, each station's NAV signals the end of a transmission. Consequently, all stations observe the medium for a DIFS and then start their random Backoff-time. Stations which did not have the opportunity to send in the first cycle will just continue to count down the old Backoff-time [44].

The time between the end of the DIFS and the start of the data transmission is called contention period. During this time collision is possible due to the random Back-off time [40].

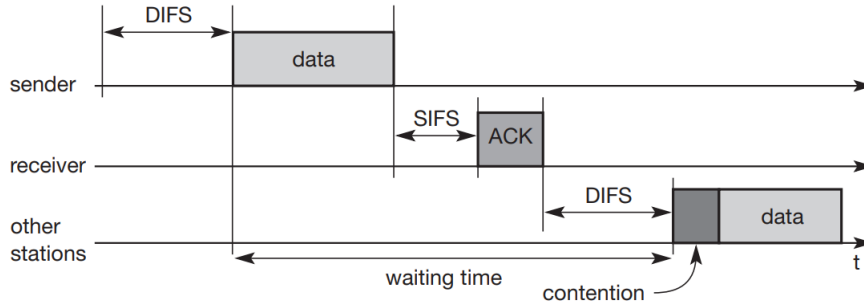


Figure 2.4: DFWMAC-DCF CSMA/CA. Figure by [44]

If two stations cannot reach each other, hence cannot update their NAV. Both stations sense the medium as free. If these two decide to send at the same time to a station they both can reach, a collision follows at the receiving station. This problem is called the Hidden-Station-Problem, depicted in Figure 2.5 [40, 44].

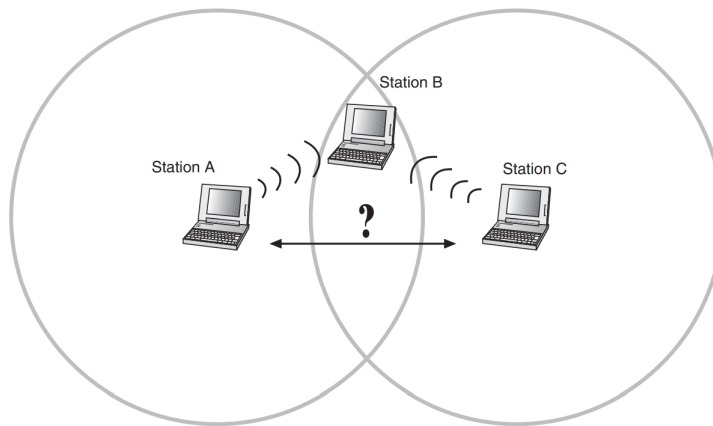


Figure 2.5: Hidden-Station-Problem. Figure by [40]

To combat this, the IEEE introduced the RTS/CTS extension to the DFWMAC-DCF mechanism. If a station senses an idle medium during the DIFS and the Backoff-time, it sends a control frame with the name Request-To-Send (RTS) instead of directly sending a data frame. After the SIFS, the receiver answers with a Clear-To-Send (CTS) frame [44]. The station which is hidden from the sender, can update its NAV according to

the information presented in the CTS. This can prevent a collision at the receivers end [44, 40]. This extension of the DFWMACDCF mechanism is presented in Figure 2.6

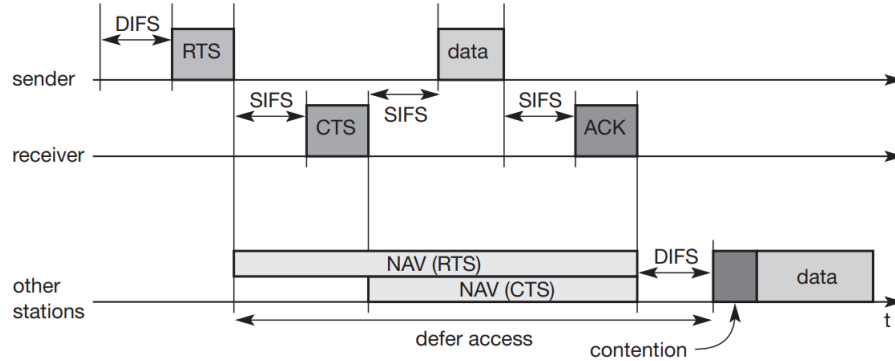


Figure 2.6: DFWMAC-DCF RTS/CTS extension. Figure by [44]

The last mechanism is the point coordination function (PCF), which provides a time-bounded service [44]. In this centralized function a point coordinator regulates the shared medium access [40]. The Access Point, which represents the point coordinator, polls all stations in the contention free period and allows the DCF to coordinate. Once the Access Point wants to send data, it waits until it senses an idle medium and waits for a PIFS [44]. PIFS is a Point Interframe Space, which has a higher priority than a DIFS [40]. After the waiting period it starts sending data downstream to the first station which can respond a SIFS later. Subsequently, all stations were polled and the AP ends the contention free period with the CF_{end} marker. This starts the contention period which uses the DCF [44].

2.1.2 Dataframes

As seen above, data frames are an essential part of managing the medium access and transferring data in a wireless network. In comparison to a wired Ethernet solution, wireless networks need more control and management information to function properly [40]. Three different main frame types are defined in the 802.11 standard [40, 6].

- **Data frames** are mainly responsible for transmitting user data but can additionally be used to communicate information to control the medium access. Data frames are specified with 01 in the type field [40].
- **Control frames** are primarily used to manage the medium access of stations in a network [40]. Furthermore, they provide reliability functions to the MAC Layer and assist in the delivery of data frames [6, 17]. RTS, CTS, ACK, Power Save Polls (PS) and CF_{end} are examples [40, 6, 17].
- **Management frames** enable a station to find an access point and associate or disassociate with a network. Furthermore, they allow synchronization and data management in the power saving mode [40, 44].

Frame Control structure

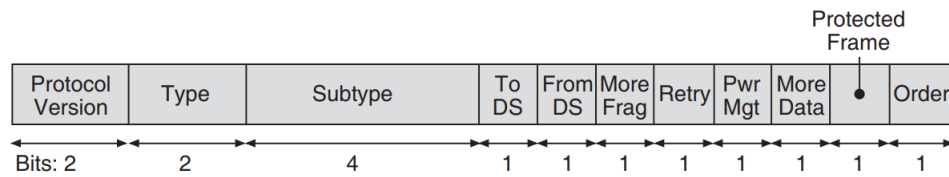


Figure 2.8: 802.11 Frame Control structure. Figure by [40]

- **Protocol version** is a 2 Bit long field which indicates the protocol version. If extensions of the current standard would cause incompatibilities, the extension would be flagged with a version number [40, 44].
- **Type** represents the frame type and is either 00 for management frames, 01 for control frames or 10 for data frames [40, 44].
- **Subtype** indicates the exact frame type. For example, if the type defines the frame as a control frame, the Subtype field defines whether it is a CTS or RTS or another Subtype [40, 44].
- **To DS** and **From DS** define the transmission path. "To DS" specifies if a frame should be sent over the DS to its destination and "From DS" indicates if the frame leaves a DS [40]. Therefore, these fields influence how many addresses are specified in the MAC frame.
- **More Fragments** indicates whether more fragments will follow after the current one [44, 40].
- **Retry** specifies if the current frame has already been sent before. This information helps to eliminate duplicates at the receiver's end [44, 40].
- **Power Management** shows whether the station goes into power-save mode after a successful transmission [44, 40].
- **More Data** informs the receiver of the frame that the sender has more data ready to be sent [44, 40].
- **Protected Frame**, which was formerly called WEP, shows whether the high-level data is transmitted in cipher or not. Nevertheless, the MAC Header will be in clear text even if the protected frame is set to 1, meaning the high-level data is encrypted [40].
- **Order** advises the receiver to forward the fragmented data in order of reception [40, 44].

Probe Request Frame

Probe Request Frames are management frames in the 802.11 standard [40]. They are the most important frame in the context of this thesis, because they allow the MAC-Scavenger system to fingerprint devices [6]. Their main goal is to discover nearby APs, respectively, wireless networks [40, 6]. They achieve this by being broadcasted in bursts on a high-frequency [6]. The amount of probe requests in each burst and the time between bursts depends on various factors such as screen state, battery state, proximity to the network, airplane mode, WiFi settings and Bluetooth activation [6].

The Probe Request Frame, depicted in 2.9, consists of a fixed part called the MAC Header and a variable part called Frame Body. All fields depicted in 2.9 are mandatory [17]. The Frame Body consists of the SSID, the station's supported data rates, and optional Information Elements (IE) [46, 40]. The SSID is the Service Set Identifier and is simply the name of a wireless network. If a station is searching for a specific previously connected wireless network, the SSID is specified. Otherwise, a broadcast-SSID is set, which indicates that the device wants to connect with any available AP [30, 17, 40]. If the station supports the network's required data rates, a Probe Response Frame is sent back to the station [40].

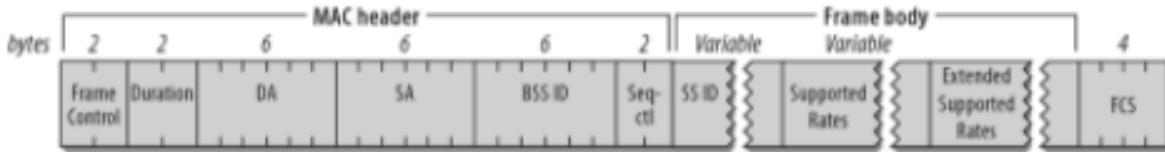


Figure 2.9: 802.11 Probe Request Frame. Figure by [17]

IEs, or tagged parameters, on the other hand, are used for advertising the support of some functionalities [46]. A list of some IEs can be found in 2.10.

- HT capabilities info
- Ordered list of tags numbers
- Extended capabilities
- HT A-MPDU parameters
- HT MCS set bitmask
- Supported rates
- Interworking - access net. type
- Extended supported rates
- WPS UUID
- HT extended capabilities
- HT TxBeam Forming Cap.
- HT Antenna Selection Cap.

Figure 2.10: Information Elements (IE). Figure by [46]

The information, such as the SSID, the source address or the IEs, constantly leaked during the search for APs by the Probe Request Frames can also be used to draw conclusions about the user (*c.f.* Section: 2.4.4) [30, 5].

MAC Identifier

MAC addresses, or Extended Unique Identifiers (EUI), identify network-capable hardware. The EUI 48 Bit long (EUI-48) and the EUI 64 Bit long (EUI-64) identifier are the most commonly used for various standards, with the EUI-48 identifier being the most used in the wireless IEEE 802.11 standards [2]. Therefore the EUI-48 is the MAC address format this thesis focuses on. Figure 2.11 depicts different compositions of the EUI-48, including an example of the ACDE48234567 MAC address in binary and hexadecimal form in the last two rows [3]. Three MAC address compositions are offered by the IEEE: The MAC address block large, medium and small of which the MA-L is considered the most widely used to create MAC addresses of the IEEE 802.11 standard [2]. Thus this thesis and the MAC-Scavenger focus on the MA-L product [6].

octet identifier	0	1	2	3	4	5
MA-L	24-bit OUI			24-bit extension		
MA-M	28-bit MA-M base				20-bit extension	
MA-S	36-bit OUI-36					12-bit extension
example value (hex)	AC	DE	48	23	45	67
example value (binary)	1010 1100	1101 1110	0100 1000	0010 0011	0100 0101	0110 0111

Figure 2.11: EUI-48 structure. Figure by [3]

The MAC address contains the Organizationally Unique Identifier (OUI) and the Company ID (CID) [3, 13, 40]. The IEEE Registration Authority (IEEE RA) assigns the globally unique OUI to the respective company [3]. Both the information which OUI is taken and the information which vendor owns which OUI is publicly available [30, 40]. The CID, on the other hand, is assigned by the company, which holds the OUI [6].

Organizationally Unique Identifier (*c.f.* Figure 2.12) contains 3 octets, each 6 Bit long. The OUI's initial octet is the most significant one, with the last two Bits of the initial octet being the least significant. These two are called M-bit (least significant Bit) and X-bit (second least significant Bit). The M-Bit value is normally 0, and so is the X-Bit, which indicates that it is an OUI [3].

When the EUI is used as a MAC address, the two least significant octets of the initial octet of the MAC address have a special significance [6, 3, 13].

When the M-Bit, also called I/G-Bit, is set to 0, the address is a unicast identifier, whereas a 1 specifies multicast identifiers [13, 40, 3]. The least significant Bit (X-Bit), also called U/L-Bit, shows if the identifier is a Globally Administered Address (GAA) or a Locally Administered Address (LAA). If the U/L-Bit is set to 0, the identifier is a GAA [13, 3, 40].

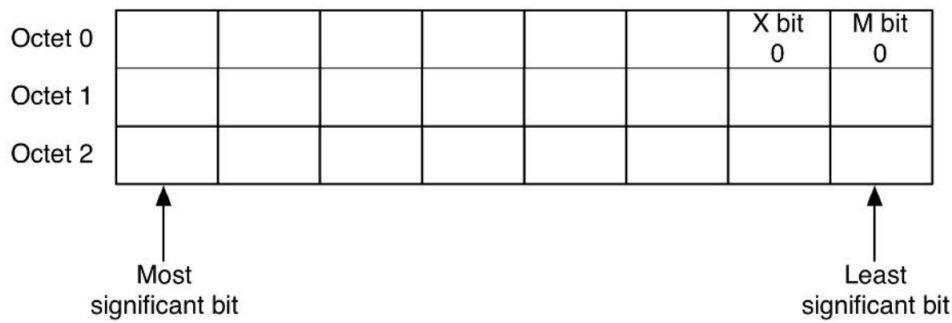


Figure 2.12: OUI-24 structure. Figure by [3]

Company ID (*c.f.* Figure 2.13) also contains 3 octets in the MA-L version. Each octet is 6 Bit long. Like the OUI, the first octet is the most significant one. The M-Bit, X-Bit, Y-Bit, and Z-Bit are the four least significant Bits [6, 3]. "In the CID, these have the values 0,1,0, and 1 [...]" [6]. Both the OUI and the CID can be represented as a base-16 number of six hexadecimal digits or as octets separated by hyphens [1, 6].

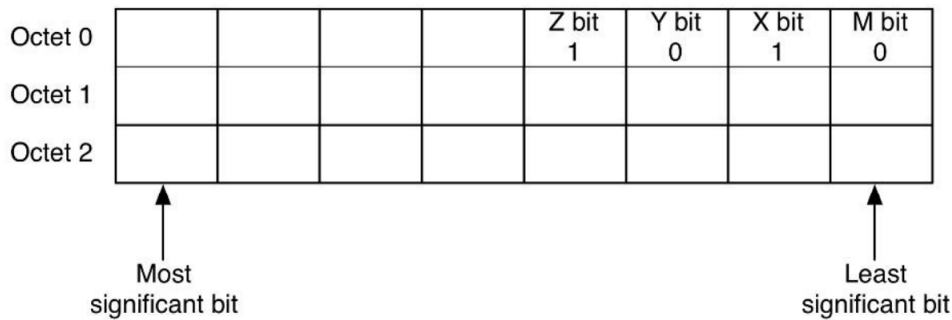


Figure 2.13: CID structure. Figure by [3]

Local and Global identifiers are MAC-addresses, which can be distinguished by the value in the U/L-Bit. The LAA is either assigned by the network administrator or it is generated from a component in a wireless LAN. The LAA is globally not unique in comparison to the EUI-48 Globally Administrated Address. The Global Administered Address is assigned by the holder of the OUI [13, 40, 6].

2.2 Passive and active Scanning

APs broadcast beacon frames over the shared medium to invite stations to join their network. From the station's point of view, this is called passive scanning, since it just listens on all channels, if not specified otherwise, for such a beacon frame. A beacon frame includes a timestamp, beacon interval, capability information, SSID and supported rates. An extensive list can be found in [40]. The AP roughly sends ten beacon frames per second. Over the beacon frame, stations receive all necessary parameters which are required to join the network. Furthermore, the stations are able to synchronize their time

with the AP's time [40].

Active scanning requires the phone's WiFi option to be enabled [43]. During active scanning, the station broadcasts Probe Request Frames on potentially multiple channels to discover a nearby network. The station can either specify a preferred AP (SSID) to connect to or specify a broadcast SSID [40, 43].

This method is often referred to as passive monitoring since no active search from the AP itself is required.

2.3 MAC-Scavenger

2.3.1 Goal of MAC-Scavenger

To avoid collision on the MAC Layer protocol, the IEEE made it mandatory for companies which sell devices operating on the MAC Layer, to assign a globally unique address to their products [6]. Moreover, as seen in section 2.1.2, the MAC address of the emitting device is part of every frame [27]. It was also shown that the address is displayed in clear text even if the Protected Frame Field is set to 1.

Capturing those frames is relatively easy and is associated with low costs since only a commercial WiFi network card that supports monitor-mode is necessary [18]. Another alternative to identify WiFi capable devices is to analyze the radio waves emitted by the PHY Layer. However, contrary to the MAC frame identification, the PHY Layer identification depends on costly equipment [18].

The information on the MAC frames, especially the MAC address, can be used to identify a device and possibly track it over a public transport system [34], a college campus [47, 21] or on high streets [7]. The revealed SSID of previously connected APs can, furthermore, in combination with geographical locations of WiFi networks, reveal the socioeconomic status and give indications on the relationships between individuals [5, 30].

To combat these potential privacy violations, the MAC address randomization was introduced [28, 6]. MAC randomization is a technique implemented by various vendors, which replaces the real MAC address in Probe Request Frames with a randomly generated one. In what time interval and on what circumstances a random MAC address is generated, is defined by the operating system of the device [46].

However, many studies show that device fingerprinting is possible even when MAC randomization is used [46, 18, 6]. The MAC-Scavenger system follows two main approaches. The first one is the Received Signal Strength Indicator (RSSI), which measures the emitted radio signal's strength. The second approach for fingerprinting devices uses the Information Elements included in the Probe Request Frame, which are still enough to identify a device [18, 46, 6]. Through these approaches, the MAC-Scavenger system can tell if specific radio traffic is originated from the same device or not. Furthermore, it can count the number of devices that are present and circumvent the countermeasures, allowing the MAC-Scavenger to localize devices in the monitored area [6].

2.3.2 Implementation and Deployment

To achieve these goals the MAC-Scavenger uses Monitors to collect data and a Sync to coordinate the analyzing and data gathering process [6]. The high-level architecture can be seen in Figure 2.14. The Monitor is responsible for the gathering of Probe Request Frames. The Monitors are devices equipped with WiFi cards, which are reconfigured in a way that they listen to all traffic and not only to the data stream addressed to them [6]. The Second part is the Sync, which communicates with the Monitors and coordinates the data gathering process [6]. The whole MAC-Scavenger is controlled by the user who can enter commands via the Shell on the Sync and the Monitor device.

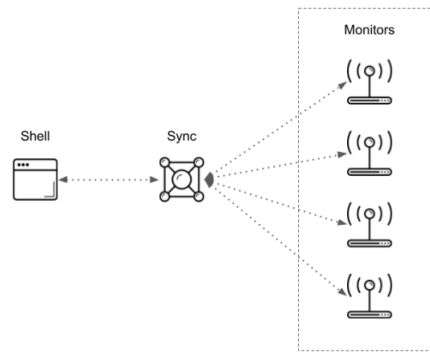


Figure 2.14: MAC-Scavenger Implementation. Figure by [6]

To have the MAC-Scavenger fully operational a certain sequence of steps need to be executed. This sequence of steps is called deployment. The list below gives a high-level overview of what needs to be done to deploy and operate the MAC-Scavenger:

- Configure every single Monitor device, which includes the following:
 - (Change keyboard layout if not familiar with the American layout).
 - Connect device to the internet.
 - Install Github.
 - Download MAC-Scavenger.
 - Install all dependencies for the MAC-Scavenger manually via the Python package manager (pip), which also needs to be installed.
- Configure the Sync device with the same steps as described above.
- Configure a MongoDB on the Sync device.
- Start Monitor devices via the Monitor Shell [6].
- Add the Monitor to the Sync by entering the port, name and IP-address of the Monitor in the Sync Shell. This needs to be done for every Monitor device [6].
- Use the *Save* command to store the information of the Monitor [6].

- Eventually load the saved Monitors into the Sync by using the *Load* command [6].
- Use the *Start* and *Stop* command to control the data gathering process [6].
- Use the *Analyze* command to get a result displayed in the Sync Shell [6].

2.4 Related Work

This section presents five different studies in the area of wireless tracking. Each study was examined from three different perspectives. The main focus was on what application field and opportunities were researched. The two sub-foci were concentrated on how the studies managed the deployment and what problems in terms of performance they encountered. The section closes with a short discussion about the mentioned related work.

2.4.1 Passive WiFi Monitoring of the Rhythm of the Campus

In 2015 a study was carried out at the Delft University of Technology in the Netherlands. The goal was to gain an insight into the occupation, duration of stay, and moving patterns between different faculties, the library, and the auditorium [21]. The main area of interest was the interiors of the central library and the auditorium.

To analyze the occupation, 20 WiFi Monitors were physically deployed in different spots. During one week, data from Probe Request Frames had been collected in the library, auditorium, and five faculties [21]. The placement of small devices needed to be approved due to potential privacy issues [21].

The data gathered from these devices was statistically filtered and divided into several different user groups due to their duration in the monitored area. As a result of the number of Probe Requests, the data was simplified to detections per 15 and 60 minutes [21]. Simultaneously, ground truth data was collected with the help of an online questionnaire and by physically counting people.

By combining the ground truth data and the data from the monitoring process, a function was derived which helps calculating the correct amount of people based on the WiFi activity. The estimation was then displayed in an online dashboard.

The study concludes that it is possible to see how many people are at a location at a particular time. Furthermore, it is possible to see how different user groups use certain buildings more often than others. Overall it is possible to identify moving patterns between buildings [21].

The study also uncovered several weaknesses in the deployment. The system, for example, had problems to differentiate between devices inside and outside of the building. To tackle this problem, the study proposes the use of directional WiFi antennas [21]. Also, the placement of more Monitors inside the building instead of the entrance will increase

coverage and better results. Additionally, the study states that the coverage should be tested extensively during device distribution due to obstacles such as walls [21]. Another problem was that some devices were detected by multiple sensors, which led to duplicates. This was solved by data analysis in retrospect. Furthermore, static wireless devices such as routers, printers, and Monitors are not automatically excluded during performance [21]. Also, the study observed that some devices were not detected. In case of failure of one of the Monitors, the researchers had a warning system in place, which helped them react to such an event. This measure helped to minimize data loss [21].

2.4.2 Tracking Unmodified Smartphones Using Wi-Fi Monitors

The University of Illinois conducted this study in Chicago. They developed a method based on Viterbi's algorithm, which estimates the trajectory of a device by using intercepted frames as input [33]. Additionally, they developed methods that prompt devices to send additional messages than just Probe Request Frames. They studied the feasibility of WiFi tracking through many real-world deployments. One of these deployments was at a 2.8 kilometer long, highly trafficked road [33].

During the 12-hour trial, seven Monitors were spaced 400 meters apart on the 2.8 kilometer long road. Additionally, the system had a server to which the Monitors were connected. The signals are then reported on a second-to-second basis to the central server, which was responsible for trajectory estimation and analysis [33].

At the end of the experiment, the estimations of selected devices were compared to GPS ground truth. They show that the mean error achieved by the system of relatively cheap firmware is about 67 meters out of the 23'000 observed devices [33]. During the deployment on the highly frequent street, they used several self-developed methods to prompt the devices to send more frames.

Those methods either emulate popular SSID or extracted SSID from the Probe Request. By broadcasting beacon frames and pretending to be an AP with such an SSID, devices are prompted to send Association Frames. As long as the station is associated with the fake AP, the devices send null frames due to the "hostapd" energy-saving function of the 802.11 standard [33]. This leads to more packets transmitted and therefore more data for trajectory estimation.

The study showed that WiFi tracking and trajectory estimation could be done quickly with commercially available, cheap hardware. Furthermore, they evaluated their methods to prompt a device to send more packets and proofed their functionality. They conclude that the estimation becomes better with denser Monitor deployment [33]. They also identified possible problems for the outdoor deployment of such a system. Although the Monitors consume little power, they need a source of power. During shorter deployments, batteries can serve as a power source. For more extended operations, they propose to work together with the local municipality to position Monitors in windows of shops on the street. The second problem is the Uplink connection to the server since, in remote areas, no internet connectivity might be available or, if available, rather costly [33].

As the study presented before, they faced duplicate detections and detections of stationary devices. These problems were solved through applications on the server. In contrary to the study from Delft University, problems regarding privacy were considered harmless since the tracking was restricted to a limited area [33].

2.4.3 Correlating WiFi Activity to Human Occupancy

Emissions from commercial buildings contribute nearly 12% to the total of the US-American annual greenhouse gas emissions [45]. A study at MIT developed a system called MIT Ethernet to continuously measure heating, ventilation, and energy consumption of buildings at the MIT Campus. The system's goal is to provide energy efficiency information, which then helps develop strategies to lower the environmental cost of buildings. One part of this system used WiFi activity to estimate human occupancy [45].

The system uses the existing infrastructure of nearly 5000 APs to estimate human occupancy at the scale of the entire campus and the fair resolution of a single room [45]. The system was able to reveal the daily profile of human activity patterns at the campus. Nevertheless, the study found some biases in measuring the WiFi activity. The researchers mention areas and lectures, such as exams, where students are required to switch off their phones [45]. Furthermore, students could leave their laptops connected at their work station and go for lunch, what counteracts efficient recognition [45, 43]. Additionally, the unequal distribution has an impact on the inaccuracy of human occupancy estimation [45].

The study concludes that the most accurate measuring system consists of a network of dedicated occupancy sensors. However, since such infrastructure is costly, the best compromise is offered by a system that monitors WiFi activity. The data provided by such a system is a good approximation while costs are kept low since it is based on current infrastructure [45].

2.4.4 Signals from the Crowd

An Italian study conducted in 2013 investigated if information on social structure and socioeconomic status can be inferred by the information revealed by the Probe Request Frames. In a 3-month long campaign, they collected frames from several national and international events, a university campus, a train station and a mall [5].

To collect data, the researchers joined the event with their laptops, which used commercially available hardware. After data acquisition, the researchers removed all frames with the wrong FCS [5]. They were able to collect 11M probes from over 160K different devices [5].

In the next step, the researchers managed to extract the OUI and assigned the vendors to the detected devices based on the public IEEE OUI listings. With the help of the detected SSID of the devices, they could design a system that automatically extracts the social connections among the smartphone owners [5]. The idea behind it is that

people with relationships or the same background often use the same vendor and have similar SSIDs. For example, when two friends, each equipped with a device, go to a local restaurant both devices will automatically connect to the restaurant's WiFi. Both devices will search again for that specific WiFi in the future. Hence it is possible to link them. The study also states that the SSID might reveal sensitive information such as the workplace or residential area which are privacy sensitive informations [5]. The researchers also plan to include information on popular public WiFi networks, such as Guglielmo, which has over 9K hotspots across Europe, in future work [5].

Furthermore, it was possible to develop a system that automatically associated an SSID with a specific language. Through this system, it was possible to differentiate the crowds of national and international events [5].

Additionally, the researchers were able to recognize the socioeconomic differences between different crowds. This conclusion was mainly based on the distribution of vendors. For example, an overrepresentation of more expensive devices indicates a more wealthy crowd [5].

The researchers finish by saying that they "believe [that] this [study] is just a first step towards a new, non-invasive methodology for uncovering non-online social networks" [5].

2.4.5 Bluetooth

Bluetooth is beside the 802.11 standard, another standard that specifies wireless communication. Bluetooth was developed by five companies (Ericsson, Nokia, IBM, Intel, and Toshiba), which are called the Bluetooth Special Interest Group (SIG) [40]. There are Bluetooth standards from the Bluetooth SIG and the IEEE, which defines the IEEE 802.15.1-Standard [40]. Bluetooth enables point-to-point or point-to-multi-point communication between the devices, which form a Piconet (Personal Area Network) with one master and several slaves in its simplest form [40].

A study conducted at the University of Zurich tried to use the Bluetooth technology to develop a system with the name BluePIL, which identifies and localizes Bluetooth capable devices [19]. The identification is more straightforward than with the 802.11 standard since no randomization strategy is used for the Bluetooth address. BluePIL uses a combination of information captured through BTBR/EDR packets, signals, and localization algorithms to achieve its goals [19].

Three experiments were conducted with the BluePIL system. The first experiment achieved an accuracy in localization of 1 m in a 12m² area, whereas the second experiment had an accuracy of about 1.4 m in a 25m² area. The third experiment was unsuccessful since it was not able to determine the number of persons in a room due to the lack of Bluetooth enabled devices [19].

2.4.6 Discussion

The studies, such as [45], [33], [21] and [6], show that identifying moving patterns and occupation is possible through WiFi signals. Although the localization via WiFi signals imposes some problems, WiFi tracking is a good compromise between monetary costs and accuracy. Problems of usage revealed by the researched studies are listed below:

- **Too few Monitors** led to a less accurate result in multiple studies [21, 33].
- **Duplicate device detection** skewed the estimated amount of detected devices [21, 33]
- **Detection of stationary devices** also skewed the estimated amount of detected devices [33, 21].
- **Not everyone is carrying an active WiFi capable device**, which skews the overall result even more. This can be due to usage restrictions, socio-economic status of the individual or because the person decided to switch the device off or not carrying one at all [45, 33, 21, 19].
- **Noisy surroundings** can further lead to heavy interference.

Every deployment of a monitoring system was deployed by computer experts. No study provides an easy "plug-and-play" deployment. The study at the Delft university [21] used an online dashboard, which helped to show the analyzed result. A similar approach was observed in the study from the University of Illinois [33], where they used a centralized server which was connected to the Monitors. The Monitors then sent data to the centralized server where it was analyzed and presented to the user.

The related work also unveiled problems with the deployment of a passive wireless tracking system:

- **Legal procedures** might be extensive and impose a huge obstacle for WiFi tracking in the future. None of the studies above focused on the privacy aspect but most of them had to make legal considerations [21, 33]. This problem cannot be overcome by automation.
- **Physical placement and connection** is another problem which could lead to the failure of the whole system. Battery consumption and Uplink connectivity might still be problematic in some areas. Moreover, broken devices must be repaired or physically replaced as quickly as possible, which cannot be automated either [21].
- **Extensive testing** of the system before usage can solve or minimize problems such as duplicate device detection, stationary device detection or noisy surroundings. This can be automated.

None of the studies examined has developed or presented a way for an easy and simple handling of their system, which would average users allow to use the passive wireless tracking technology. All systems were deployed by computer experts, which monitored the system closely.

Chapter 3

Prototype Design

This section presents the prototype of the improved MAC-Scavenger system, called Web-Scavenger. The naming and architecture is further explained in chapters 3.2 and 3.3.

3.1 Requirements

This chapter defines the requirements, which are derived from the goals of this thesis and the findings in related research. Since the main goal is to simplify and automate the usage of the MAC-Scavenger, this thesis orientates itself on the guidelines from TechRepublic and TechTerms for user friendly programs. TechRepublic is a company which specialises in best-practice standards for IT-technologies [50]. TechTerms, on the other hand, is a free online dictionary for IT-terminology. The most important characteristics for a simple and user friendly program described by TechRepublic and TechTerms are listed below.

- **Simple to install** [48].
An average user without specific knowledge in computer science can install the program and in case of obstacles, there is a documentation which assists the user to overcome them.
- **Pleasant, easy-to-navigate GUI** [48].
An average user without specific knowledge in computer science can navigate the program to retrieve the information he wants to see.
- **Intuitive**
"In order to be user-friendly, an interface must [...] make sense to the average user and should require minimal explanation for how to use it" [10].
- **Easy to troubleshoot** [48].
An average user without specific knowledge in computer science can detect, understand and resolve errors.
- **Does not need third-party software** [48].
An average user without specific knowledge in computer science is not required to read into documentation and instructions of additional third party software.

3.1.1 Deployment

The list below describes the requirements for the deployment. The deployment is defined as the sequence of steps that need to be taken until the MAC-Scavenger *Add* command, which adds Monitor devices to the Sync, can be executed.

1) GUI instead of MAC-Scavenger Shell and terminal

The Web-Scavenger provides a GUI, which is intuitively to understand and operate. It also displays error messages for the most common errors.

2) No Git-technology necessary

The improved system allows to download and deploy the MAC-Scavenger without knowledge or usage of the Git-technology. The removal of third party-libraries such as Git helps to reduce complexity and create a more user friendly, hence simpler, program [48].

3) Easy to understand and accessible documentation

The improved system provides a simple, more extensive documentation about the MAC-Scavenger, which is also easy to access. The documentation consists of instructions how to do the initial setup of the Tinkerboard and how to deploy and operate the new improved system. The documentation improves the program since it makes it easier to navigate the GUI and to solve errors. [48].

4) No database setup required

The Web-Scavenger prevents the user from configuring a MongoDB, which reduces the third-party software percentage the user has to deal with. This makes the overall application simpler to use [48].

5) Automate the Installation of Python libraries

In a simple program the user does not need to deal with third libraries and their complicated installation [48]. The improved system handles the installation and possible errors automatically.

6) Deployment in less than 10 clicks

The Web-Scavenger requires only ten simple mouse clicks to complete the deployment of the MAC-Scavenger. This means that after ten clicks the user is ready to add new Monitor devices to the Sync. Such a short deployment together with the simple, uncomplicated user interface simplifies the program according to [48, 10].

3.1.2 Operation

This section describes the requirements for the improved system during operation. Operation includes adding and removing new Monitor devices, starting and stopping the data gathering process, analyzing the data and dealing with error messages.

7) GUI instead of MAC-Scavenger Shell and terminal

Similar to the deployment requirements, the Web-Scavenger does not require the user to know how to navigate or use the MAC-Scavenger Shell. Furthermore, the user does not need to know the exact *Start*, *Stop* and *Analyze* commands because they can be executed by simple self-explanatory buttons. All these commands are presented to the user on one compact graphical interface.

8) No knowledge of network architecture required

The Web-Scavenger does not require the user to understand the Monitor-Sync-Architecture described in section 2.3.2. Furthermore, no knowledge about IP-addresses or ports is required.

9) Easy to understand and accessible documentation

Similar to the deployment, the operation of the new system and its functionality is well documented which allows easy troubleshooting.

10) *Save* and *Load* are automated

The original *Save* and *Load* commands are executed automatically by the Sync when they are necessary.

11) Easy troubleshooting

The Web-Scavenger provides a graphical error message for the most common errors that might occur. Additionally, the user is able to see his past actions and errors in a simple history.

12) From finished deployment to analyzed results in less than 5 clicks

The process from a newly configured system to analyzed data results is possible in less than five mouse clicks.

13) Remote control of the MAC-Scavenger

The improved system can be controlled remotely, meaning one does not have to be in the same physical place as the Sync.

3.1.3 Security

This Chapter presents the requirements regarding cyber-security aspects. The related work shows that legal procedures needed to be taken to handle privacy relevant information. Often these legal procedures impose restrictions or guidelines on how to protect sensible data. So far the MAC-Scavenger does not have functionality to protect its data. Therefore, the new improved system implements a few simple measures for privacy protection. These measures specifically aim to secure the Sync node which is where most data is stored.

14) Token based authentication

A user receives a token upon login with which he can authenticate himself for 30 minutes. The user can login with his individual username and password. Upon each service call from the user the token is validated. If the token is valid the requested

service and data is provided. If not, the user is automatically logged out. Moreover, the user has the ability to manually logout.

15) Encrypted data

To improve security of the data in the database, the improved system implements functionality to hash passwords and to encrypt and decrypt sensible user information.

16) Administrator view

An administrator can restrict access of users and needs to explicitly accept a user upon his first login. To perform this, the administrator has its own graphical user interface which is different from a user without administrator rights.

17) SSH-key instead of default root user login

The Sync device can no longer be accessed remotely with the public default root username and password. Instead, only authorized personal can access the Sync via a SSH-key.

3.2 Web-Scavenger Approach

The proposed system to improve the MAC-Scavenger is called Web-Scavenger. The MAC-Scavenger is a combination of the words MAC and *scavenger*. The word MAC indicates that the MAC-Scavenger solves the challenge of fingerprinting devices imposed by the MAC-randomization. *Scavenger* on the other hand means *hunter*, which references the process of rummaging through information to extract important information pieces [6]. The Web-Scavenger follows the same portmanteau word approach, by combining the word *web* and *scavenger*. In this context, the word *web* refers to the world wide web, since the improved system's graphical user interface is implemented as a website. The word *scavenger* is used to link the Web-Scavenger to the original MAC-Scavenger system. The Web-Scavenger refers to both the website component as well as the Desktop Application component of the newly developed system.

A web based approach was chosen because it allows the user to access the system remotely from anywhere in the world as long as he or she has internet access. Moreover, a website can be accessed from different devices such as laptops, phones and tablets. It is also independent from the operating system on these devices. Related studies [21, 33] followed a similar approach where a centralized component, which analyzed and displayed the results, was implemented as a "server". This centralized part represents the Sync in the MAC-Scavenger, therefore the Sync is part of the website component of the Web-Scavenger.

3.3 Architecture

This chapter presents the high-level architecture and the database model of the Web-Scavenger. The interaction between the different components of the Web-Scavenger de-

scribed below is graphically portrayed in Figure 3.3.

The Web-Scavenger consists of two different main programs. The first program is a Desktop Application which runs on the Monitor device. The second application is the website consisting of front-end, back-end and a restful API.

To start the Web-Scavenger a user has to log into his account on the website. Upon login the user is assigned a token with which he can authenticate himself over the restful API. Access to the different services is also gained through the token. Once the user has logged in, he is able to download the Desktop Application from the website to the individual Monitor devices. Through the Desktop Application the user is able to do some configuration of the Monitor device. The configuration for example includes naming the Monitor device.

Once this has been done, the Desktop Application will start communicating with the web server. To allow the web server such a communication, it needs to start looking for incoming traffic. This is done by the user who tells the server to listen to incoming traffic via the Web-Client. After a connection between both the Desktop Application and the web server has been established the Desktop Application shares IP-address, port and name with the web server which in return confirms the reception of the data. Finally the Desktop Application starts the Monitor program.

The Monitor program is part of the MAC-Scavenger program and has not been altered. The communication of the Desktop Application to the Monitor is a one way communication. This approach was chosen to keep the MAC-Scavenger in its most original form. The Monitor receives all parameters such as port, name and IP-address from the Desktop-Client and tries to establish a connection to the Sync node [6]. The Sync is also part of the original MAC-Scavenger system.

On the web server side, after the Desktop Application has connected to the server, the user can start and stop the data gathering process over the Web-Client. The communication of the restful API to the Sync is also a one directional communication. After the Sync program has been started, it loads the data of all necessary devices into the Sync, before starting the data gathering pipeline (*c.f.* Figure 3.1). This pipeline includes starting and setting up the Monitor and then receiving the data from several Monitors. After the data stream has been received, the Sync reassembles the data and removes errors. Finally, the data is stored in the MAC-Scavenger database [6].

Upon starting the process on the Sync, the Monitor device (Monitor program) receives the instruction to dump packets with the help of tshark [6]. Tshark is a part of the hacking tool Wireshark and allows to capture packets in a network [49]. The Monitor node pipeline (*c.f.* Figure 3.2) furthermore filters out the Probe Request Frames and extracts important data fields such as the MAC address and IEs. In the end of the Monitor node pipeline the data is sent to the Sync program which then stores it into a database [6].

After the data gathering process the user can request the analysis of the gathered data via the website. After checking the validity of the token, the web server starts the Analyzer program from the original MAC-Scavenger. The Analyzer then responds with the analysis which then gets forwarded to the Web-Client where the user can review it.

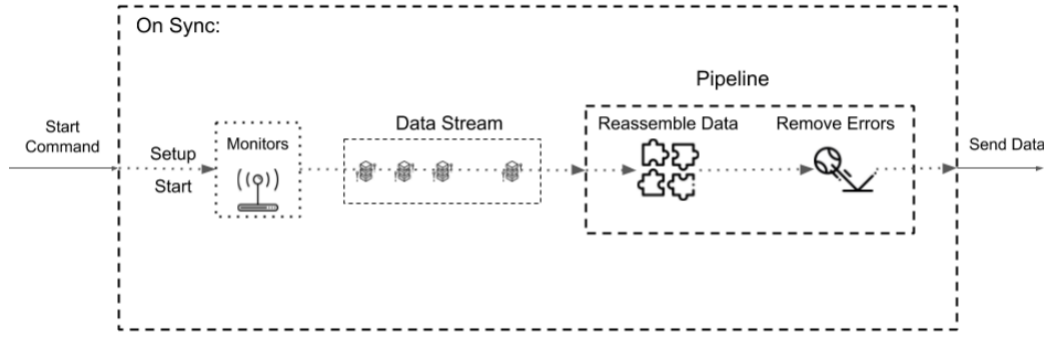


Figure 3.1: Data Gathering Pipeline on the Sync nodes. Figure by [6]

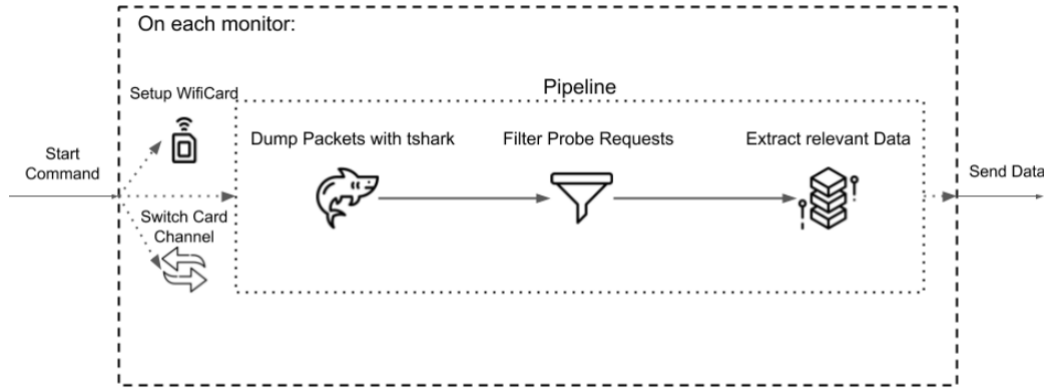


Figure 3.2: Data Gathering Pipeline on the Monitor nodes. Figure by [6]

The Figure 3.4 shows the architectural design of the web server including the front-end in more detail. The Web-Scavenger has two different databases. The Web-Scavenger database (left database in Figure 3.4) is responsible for storing all user data. Therefore, the Login Controller only accesses the Web-Scavenger database. Additionally to the user data, device related data is also stored in the Web-Scavenger database. Once the user starts the MAC-Scavenger Sync, the Service Controller retrieves all device related data for a specific user from the Web-Scavenger database and transforms it into a format which is readable by the Sync. Now, the Sync can access the device information and starts the data gathering process together with the Monitor nodes (*c.f.* Figure 3.3).

During the data gathering pipeline described above, the data extracted from the Probe Request Frames is stored in the MAC-Scavenger database (right database in Figure 3.4). The front-end and ultimately the user is able to steer the MAC-Scavenger data gathering pipeline through the Service Controller which is able to start, stop and add new devices. When the user requests the analysis, the request is forwarded from the Service Controller to the Analyzer which then responds with the analysis of the data gathered so far.

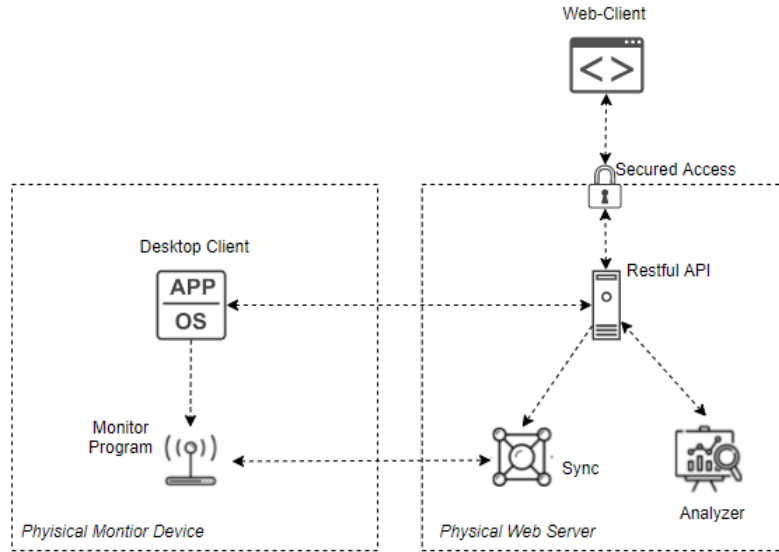


Figure 3.3: Interaction of App- and Web-components of the Web-Scavenger

3.3.1 Database

This section describes the relational database of the Web-Scavenger. Therefore, the relational model presented here represents the one implemented in the Web-Scavenger database in Figure 3.4.

There are three tables in the Web-Scavenger database. The main table is the "users" table where all users and their attributes are stored. The *user_id* is the primary key of a user and identifies him uniquely. The *role* represents the user type.

Currently, there are only two user types implemented: *Admin* and *Guest*. The *Admin* user type has additional features available such as monitoring the other users and allowing or denying access. *Guest* users on the other hand are normal users which can use the Web-Scavenger but cannot control access for other users. If a *Guest* user is permitted to use the Web-Scavenger or not can be checked with the *accepted* attribute.

Username and *password* represent the credentials with which a user can log into his account. Both strings cannot be longer than 50 characters. Additional restrictions, such as minimum length, apply to the *password* (*c.f.* Subsection 3.5.2). Since no feedback is given from the Sync program to the Service Controller (*c.f.* Figure 3.4), the Web-Scavenger must store the status of the Sync separately. Therefore, the running *attribute* shows whether the Sync is currently running or sleeping.

The table with the name "devices" contains all devices which a user has. The relation between the user and the device table is a zero to many relationship. Even though the user can have zero devices, at least one is necessary to actually start the MAC-Scavenger. The devices are identified by a generated ID called *device_id*. Additionally, each device has a *name*, *port* and an *IP-address*. All these attributes are provided by the Desktop Application to the web server. The *user_id* in the "devices" table represents a foreign key, with which a device can be assigned to a user and vice versa. When the user starts the

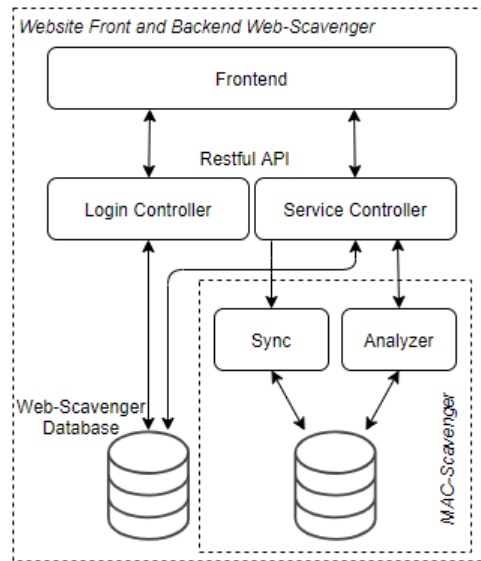


Figure 3.4: High-Level Architecture of the Web-Scavenger Website Component

data gathering pipeline, he is required to select the devices which should gather data from the pool of added devices. The *added* attribute shows the Service Controller if it needs to reformat the data of the device, so that the Sync can use it in the data gathering process. If the *added* boolean is set to false for example, the device will be ignored by the Service Controller and hence the Sync will not use it for data gathering purposes.

The "events" table stores all events for the history feature (*c.f.* Subsection 3.5.2). An event could be the execution of a command, an error or a warning. An event is characterised by an *event_id* which identifies an event uniquely. The *title* attribute is a short string message which describes the event. The *device* attribute includes the IP-address of a device which was connected to an event. If no device was part of the event, this attribute remains empty. The *device* attribute is not a foreign key like the *user_id* is. Last but not least, the *time* attribute shows when an event occurred.

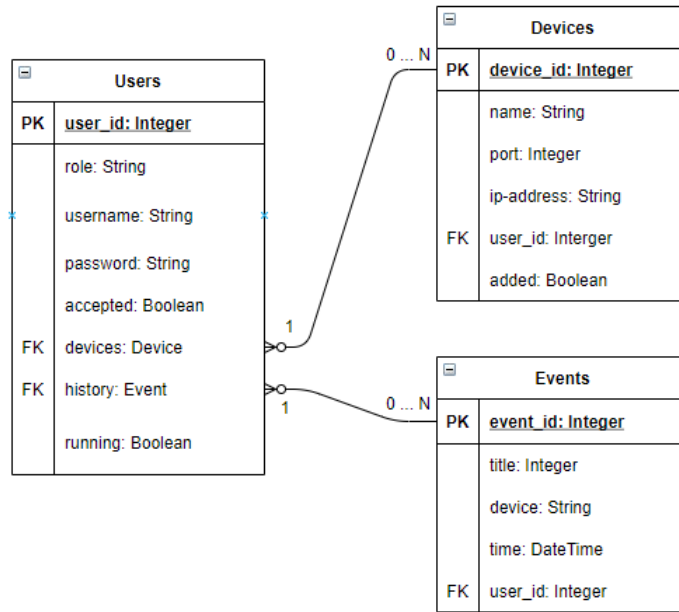


Figure 3.5: Database structure of Web-Scavenger-database

3.3.2 Deployment and Operation

This chapter focuses on the user perspective during the deployment and operation. As described in subsection 3.1.1, the deployment is defined as the sequence of steps that need to be taken until the MAC-Scavenger *Add* command, which adds Monitor devices to the Sync, can be executed. Therefore the deployment goes until the Monitor devices are found by the web server (*c.f.* Figure 3.6). Hence the operation consists of all actions after the *Add* command, including the *Add* command.

The process starts when the user wants to use passive wireless monitoring due to reasons mentioned in section 2.3.1 and 2.4. First of all, the user needs to register himself on the Web-Scavenger site. After entering his credentials and confirming the password, the user is directed to the Login-Page. In the background all requests from registering to login are handled by the Login Controller. Following the registration, a new entry in the “users” table of the Web-Scavenger database is created. When a user wants to login over the website, his credentials are checked against the username and password stored in the database. Furthermore, the admin is prompted to review the newly registered user to allow him access. If the admin has not granted access to the new user, he or she will not be able to login.

After the login, the user can either continue by downloading the Desktop Application, if he or she is familiar with the functionality. Alternatively, the user can read the documentation first. The documentation not only contains information on how to deploy and operate the MAC-Scavenger, but also on how to setup a Tinkerboard as a Monitor node. A Tinkerboard is a single board computer, which was mainly used as Monitor device in the scope of this thesis [20]. After downloading the Desktop Application the user needs to configure the Tinkerboard and start the Desktop Application. Subsequently the Desktop Application window asks the user to name that specific Monitor device. After a name is

inserted, the user has the option to start or stop the Monitor program over the Desktop Application (*c.f.* Figure 3.17). Errors during the operation of the Monitor program are displayed in the Desktop Application window. In the next step, the user must place the Monitor in the area he or she wants to monitor.

Back on the website, the user, who is still logged in, needs to search for the Monitor device he has just configured. If the search is successful the new device will appear on the screen including the previously configured data for that specific device. If no device is found an appropriate message is displayed. This functionality is handled by the Service Controller (*c.f.* Figure 3.4).

After the devices are found by the Service Controller, the user can select the devices which should be used for the next data gathering session. This section is similar to the *Add* command in the original MAC-Scavenger process. Therefore, this action belongs to the operation. Subsequently, the user needs to start the Sync through the Service Controller, which then initializes the pipeline described in Figure 3.1 and 3.2. If no devices are selected a message is displayed informing the user that he needs to select a device (*c.f.* Figure 3.12).

In the end the user wants to review and analyse the data gathered by the Web-Scavenger system. Consequently, he or she stops the data gathering pipeline and instead uses the analyse functionality through the Service Controller. The Service Controller then forwards the results from the MAC-Scavenger Analyzer to the Web-Client where the user can review it. In the end of the process, the user logs out from the website.

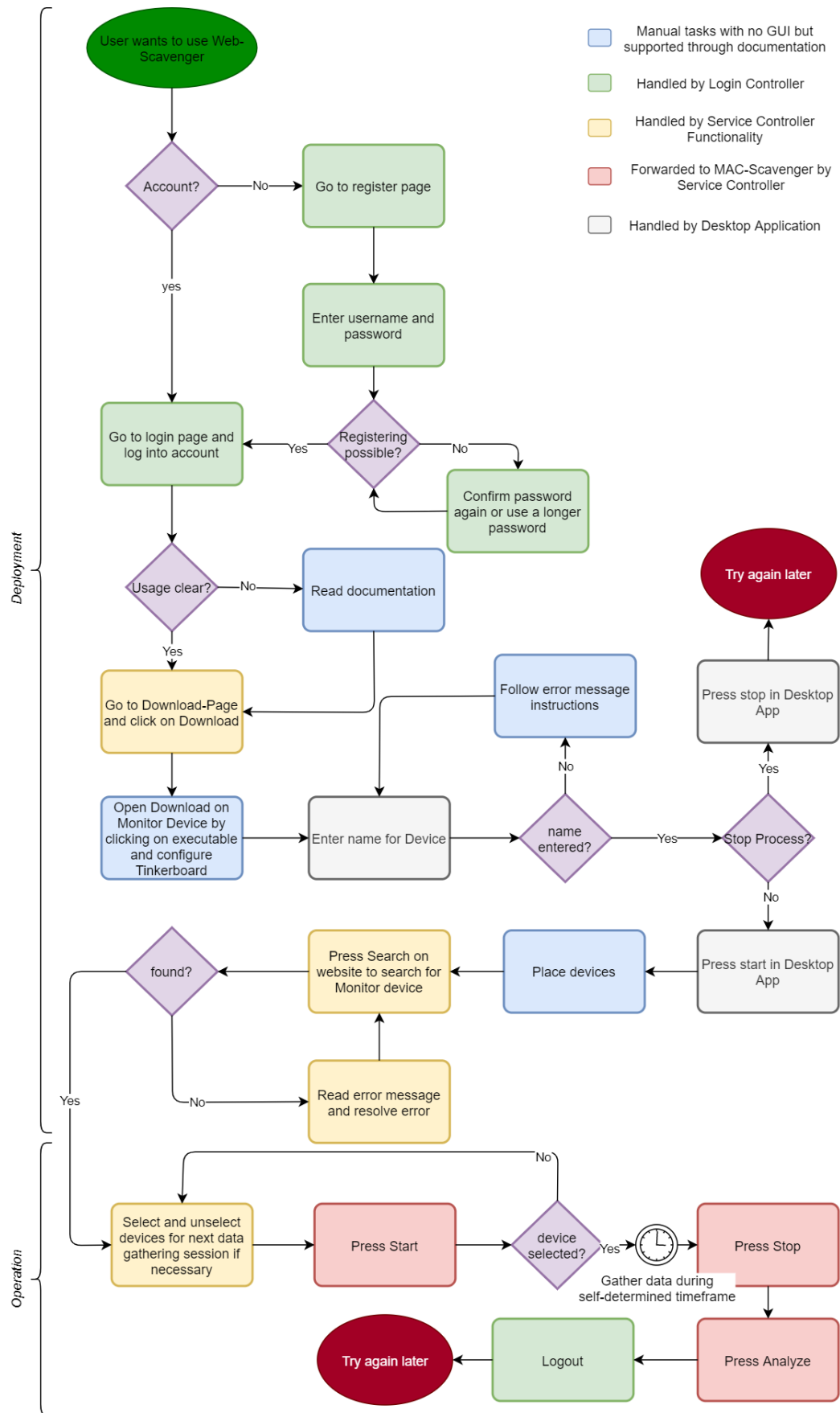


Figure 3.6: Process to deploy and operate the MAC-Scavenger

3.4 Features

The list below shows all the features of the new Web-Scavenger. The second column also reveals the component in which the feature is implemented.

Feature title	Component	Feature Description
Authentication	Login Controller	The login and token based Authentication are both part of this feature. The login allows the program to assign a token to the user, which is required to access data and services of the Web-Scavenger.
Admin view	Login Controller	The administrator can allow and restrict access for several users over a admin-specific user interface.
Register	Login Controller	The registration process allows users to apply for the usage of the Web-Scavenger. Once the data has been checked by the administrator they are allowed to login.
Desktop Application	Desktop Application	The Desktop Application provides a graphical user interface for the Monitor node. It starts the Monitor program, the Installation Script (<i>c.f.</i> Installation Script) and helps the web server to find the Monitor node (<i>c.f.</i> Search feature). Additionally, computer specific parameters such as IP-address and port number are retrieved and handled automatically.
Installation Script	Desktop Application	The Installation Script automatically downloads all necessary libraries required for the Monitor program to run.
Search	Service Controller and Desktop Application	The Search functionality in the Service Controller of the web server and the Desktop Application allows the exchange of computer specific data such as IP-address and port and therefore allows both programs to start a communication channel.
History	Service Controller	The history gives the user a graphical interface to review the past actions and errors.

Device Reformatter	Service Controller	The reformatting feature allows the Service Controller to write a YAML-file with device specific information, which is required by the MAC-Scavenger Sync.
Device Selection	Service Controller	Through the selection feature the user can choose devices for the data gathering pipeline without having to delete them from the database.

Table 3.1: Features of the Web-Scavenger

3.5 Implementation

This section is structured into two parts. The first part shows implementation details of the website component while the second part focuses on the Desktop Application.

3.5.1 Technologies

Figure 3.7 shows which technologies are used in which component. The front-end of the Web-Scavenger is written in the programming language Typescript together with the React framework. It is operated by Nginx as a reverse proxy. The back-end is written in Python with help of the Flask web-framework. More details can be found below.

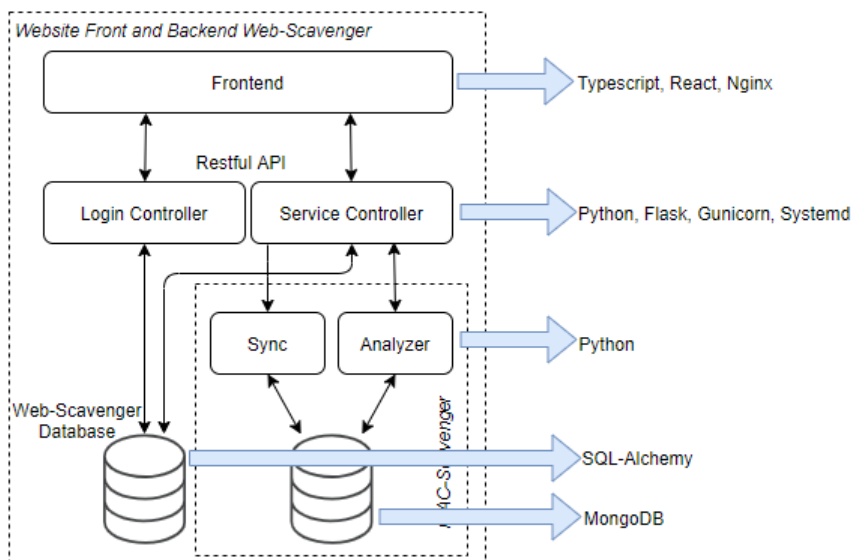


Figure 3.7: Technologies used in Web-Scavenger components

Typescript and React

Typescript is "an open-source language, which builds on JavaScript" [29]. It introduces types to the code, which helps to document and structure the application [29]. React is the framework which allows to build user interfaces with JSX. JSX is a combination of JavaScript and XML and helps to structure the code in components [16].

Python and Flask

Python is a object-oriented, functional-oriented and aspect-oriented programming language [6]. A big advantage of Python is the modularity due to its libraries and packages which can be added freely to the program [39]. The MAC-Scavenger is written in Python which makes the interaction with the Web-Scavenger, which is a Python program, straight forward [6]. Flask, on the other hand, is a web application framework, which provides libraries, tools and technologies to build a web application [25].

Deployment Website

Technologies for deployment and operation of the website are Nginx, Gunicorn and Systemd. "Nginx is a open source software for web serving, reverse proxying [...] and more" [15]. In the context of this work, the Nginx is used as a reverse proxy, which passes requests from the client on to the Flask back-end and the response back to the client [42]. Gunicorn is a Web Server Gateway Interface to run web applications. It calls the program written in the Flask framework [24]. Systemd is an initialization system and service manager for Linux [26]. In the context of the deployment of the Flask back-end it is used to run the Python program in the background.

Databases The Web-Scavenger uses the SQL-Alchemy database, whereas the MAC-Scavenger relies on a MongoDB. The MongoDB stores objects in JSON-format, which makes the MongoDB highly flexible [31]. The MAC-Scavenger stores IEs, MAC addresses and the location estimation of these devices in the MongoDB [6]. The SQL-Alchemy is a relational database, which comes as Python library. Its ORM (Object-Relational-Mapper) makes it easy to map Python objects into the database [9, 38].

3.5.2 Website

Dashboard

In Figure 3.8 the main Dashboard of the website of the Web-Scavenger is shown. The Dashboard of the Desktop Application is discussed in section 3.5.3. After the successful login the user with a valid token is directed to the Dashboard-Screen.

The screen depicted in Figure 3.8 shows the Dashboard-Screen of an *Admin* user. Therefore, on the left side of the screen all registered *Guest* users are listed. The *Admin* can easily restrict access for a user by clicking on the box next to the user's name. If the box is checked, it means that full access to the Web-Scavenger functionality is granted. Upon registration of a new user the access is forbidden until an *Admin* user allows it. The Dashboard of the *Guest* user looks alike, however there is no list of other users available. And the option to restrict the access is also not given.

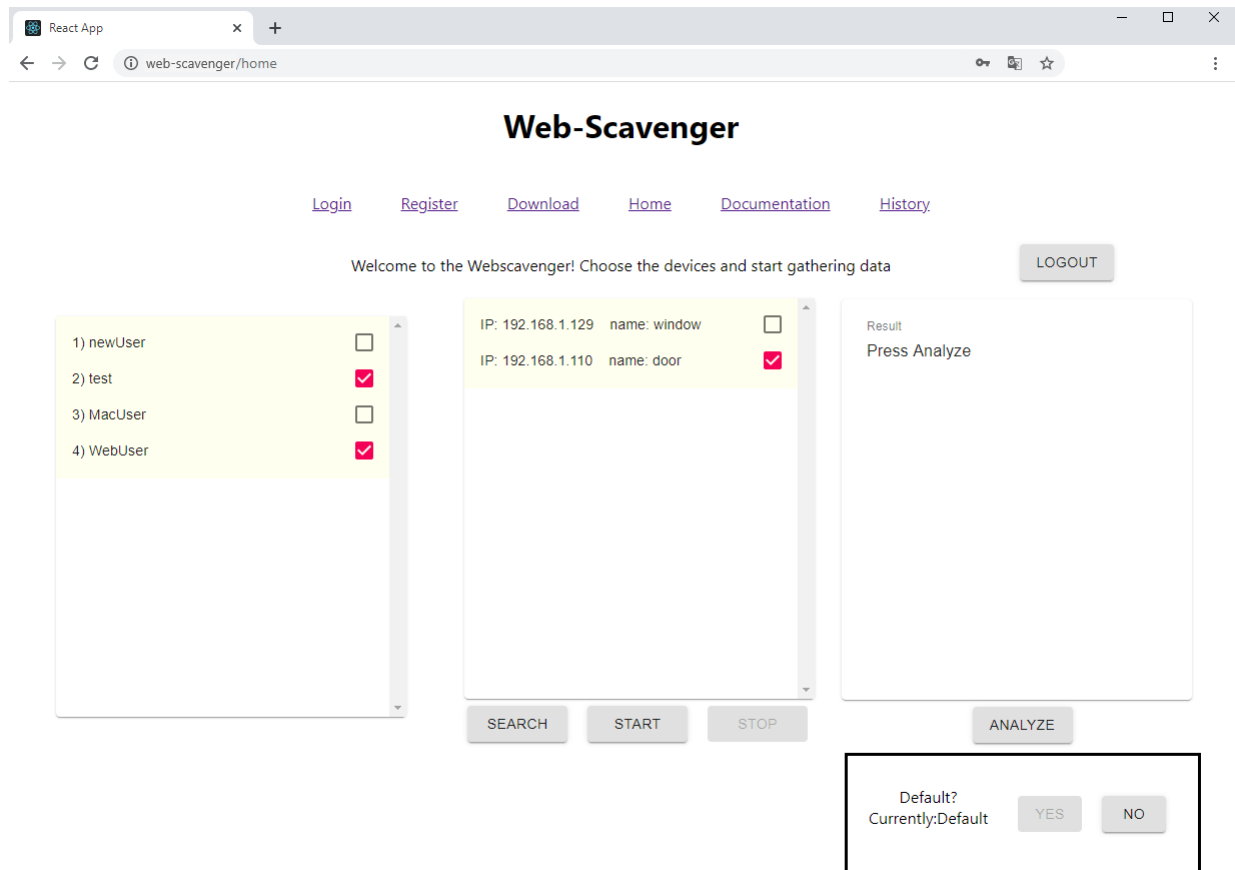


Figure 3.8: Dashboard of the Web-Scavenger

The card in the middle of the screen in Figure 3.8 shows all Monitors, which were added to the Web-Scavenger. The IP-address of the device and the name given by the user over the GUI in the Desktop Application are displayed. When the box next to the device's information is ticked, it means that the device will be used for data gathering purposes when the Start-Button is pressed. This allows the user to dynamically choose which Monitors should be used during the data gathering process without having to engage with the physical device.

The right side of the screen is dedicated to the analyze functionality of the Web-Scavenger. When pressing on the Analyze-Button the Web-Scavenger will analyze the gathered data with the default configuration. If instead of the default configuration a custom configuration should be used to analyze the data, then this needs to be arranged in the box below the Analyze-Button. Through pressing on the No-Button the custom mode is set. Located above the three Cards is the navigation bar. Every user with a valid token can navigate to the Download-Page, Documentation-Page or the History-Page. An alternative to the navigation bar is to enter the new URL in the browser window. For example to go to the Documentation-Page, `/docs` needs to be added to the URL. Additionally to the navigation bar, a short instruction is provided to the user together with a Logout-Button, in case the user wants to log out.

The Download-Screen is shown in Figure 3.9. The navigation bar is on top, together with

a short instruction on what to do. The logo of the Web-Scavenger can be seen in the middle of the screen. The user will start the download of the Web-Scavenger Desktop Application (*c.f.* Section 3.5.3) through clicking on the picture. The Desktop Application is inside a zip-folder, which also contains the whole MAC-Scavenger. So, in case a person wants to use the original MAC-Scavenger system he or she can download the whole source code from the website without using Git.

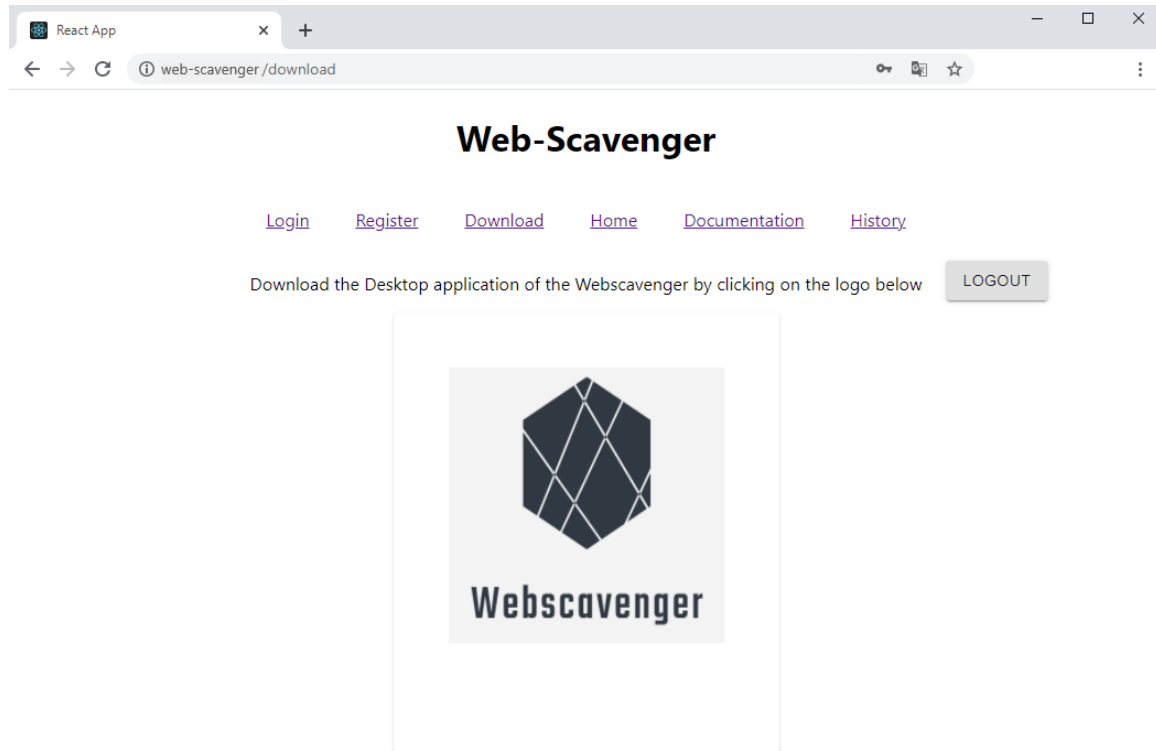


Figure 3.9: Download-Page Web-Scavenger

When clicking on the Documentation-Link in the navigation bar, the user is directed to the Documentation-Screen, which can be seen in Figure 3.10. Additionally to the navigation bar, there is the navigation menu for the PDF-file. With the Prev- and Next-Button the user can switch between pages. With the two Icon-Buttons next to them, the user can zoom-in and -out of the pdf. The whole documentation is a pdf saved in the front-end code. This allows for quick updates in case of additional functionality, since no changes to the code are necessary when updating the documentation. The current pdf documentation can also be found in the appendix of this thesis.

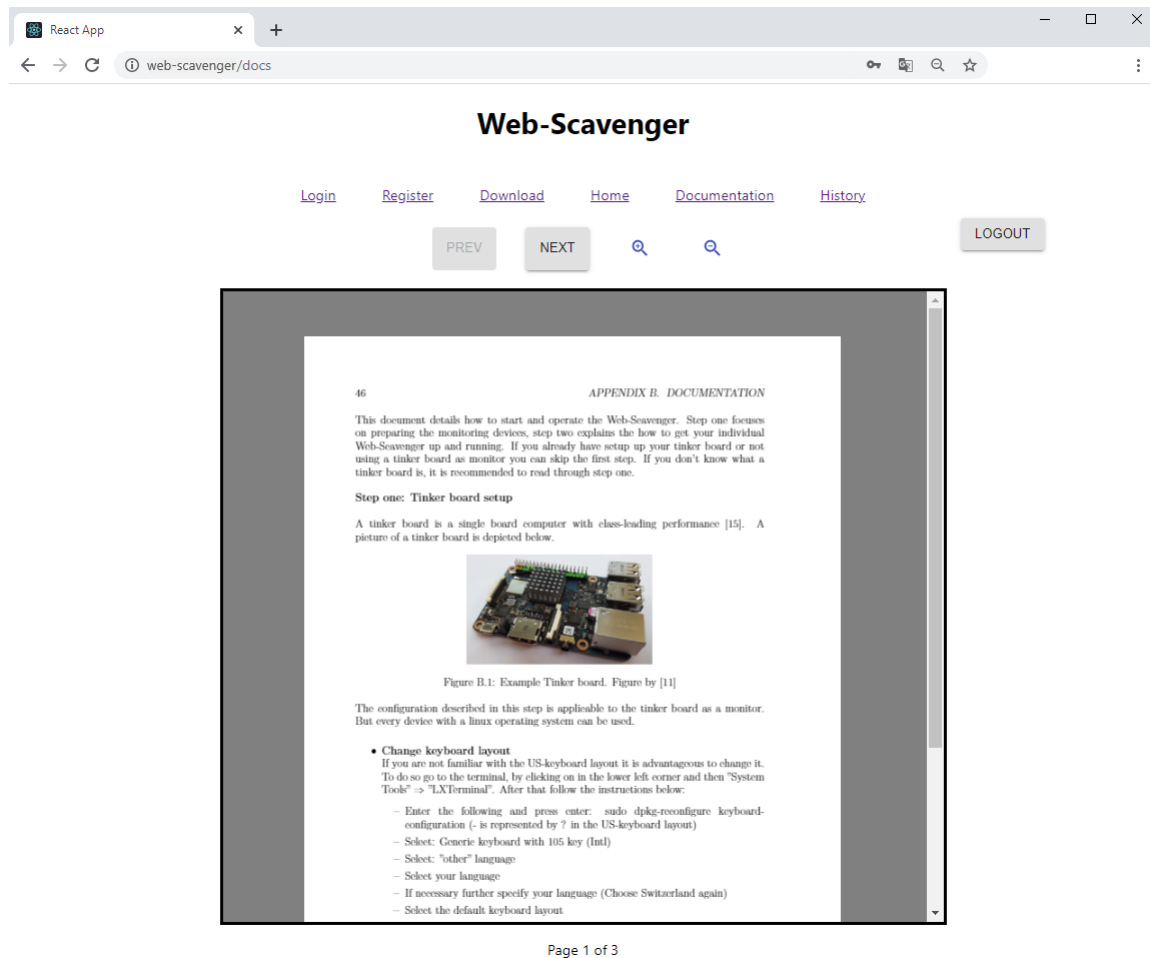


Figure 3.10: Download-Page Web-Scavenger

Start, Stop and Analyze

After the Start-Button in the Dashboard, depicted in Figure 3.8, is pressed, the Web-Scavenger executes the code shown in Listing 3.1. The code in the function *start_scavenger* is simplified for the sake of clarity. The error handling for example, including updating the history in case of an error with an appropriate message, is omitted.

The function *start_scavenger* retrieves the username from the token and also retrieves all devices associated with that user. Afterwards, a loop iterates through all devices and selects those which were selected for the data gathering process by the user by checking the *added* attribute. The loop also filters out the attributes which are unnecessary for the MAC-Scavenger execution.

The *write_device_file* function then writes all selected devices into a YAML-File, which can be read by the Sync. This is the Device Reformatter discussed in the Feature section (*c.f.* Section 3.4). Therefore, the *write_device_file* automates the *Load* command of the MAC-Scavenger.

After formatting the devices a subprocess is started. The *subprocess.Popen()* operation

starts the MAC-ScavengerShell.py script located in the MAC-Scavenger. It further stores the ID of that subprocess in the web application configuration, such that it can be accessed later to stop it. In the end, if no error occurred, the history is updated and an appropriate message is sent back to the client.

```

1      def start_scavenger:
2          #error handling omitted
3          username = get_jwt_identity()
4          devices = User.find_by_username(username).devices
5
6          list_of_devices = []
7          for device in devices:
8              if device.added:
9                  list_of_devices.append(device.map_to_config())
10         write_device_file(list_of_devices, file_path_mac)
11
12         check_if_more_than_one_device(list_of_devices)
13
14         commands = "python MacScavengerShell.py"
15         process = subprocess.Popen(commands, universal_newlines=True
16                                     , cwd=file_path_mac, shell=True)
17
18         app.config['MAC_PID'] = process.pid
19
20         history_update(user, EventTypes.started.value[0])
21
22         return {"message": "MacScavenger was started"}, 200
23
24     def write_device_file(list_of_devices, file_path_mac):
25         file_path = os.path.join(file_path_mac, 'config', 'config.
26                                 yaml')
27
28         yaml.emitter.Emitter.process_tag = noop
29
30         with open(file_path, 'w') as file:
31             yaml.dump(list_of_devices, file)
32
33     class DeviceConfig:
34         def __init__(self, host, name, port):
35             self.host = host
36             self.name = name
37             self.port = port

```

Listing 3.1: Start Data Gathering Process

The Stop-Button, when pressed, executes code that stops the subprocess and all child processes recursively. This has to be done because the MAC-Scavenger as subprocess has its own subprocesses which need to be stopped as well.

The Figures 3.11 to 3.13 show different results for the starting mechanism and the *analyze* function. Figure 3.11 depicts a running Web-Scavenger with a device named *door*. This can be inferred from the green message at the bottom or the disabled Start-Button. Figure 3.12 illustrates one of many possible error messages. In case of Figure 3.12, it is a warning that at least one device needs to be selected to start the data gathering pipeline. In the

Figure to the right, the output of the analyze functionality can be seen. The output was generated in Default-Mode.

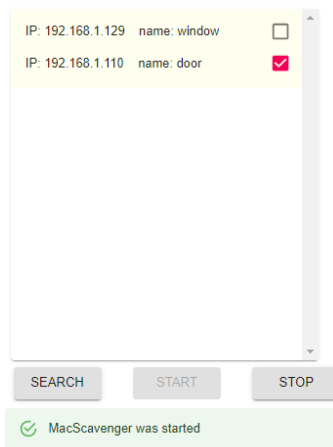


Figure 3.11: Start data gathering process



Figure 3.12: No Device selected for Start

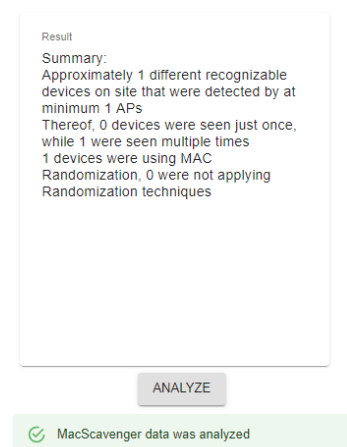


Figure 3.13: Analyze data

History

The history feature allows the user to review his past actions to identify errors. Figure 3.14 shows the History-Screen of the Web-Scavenger. On top, there is the navigation bar together with the Logout-Button. Below the navigation bar are the events of the history. An event in the history displays the time of the event together with a short description. If the event is related to a device, the IP-address of that particular device is also shown in the event description. This can be observed in the event on top. Additional information is provided through the coloring of the event. The color depends on whether it is a warning, an error or a successful event. The history, furthermore, never consists of ten or more events because it is implemented as a FIFO (First in, First out) Queue. A FIFO Queue removes the oldest event, which is the first event inserted in the Queue, first [35].

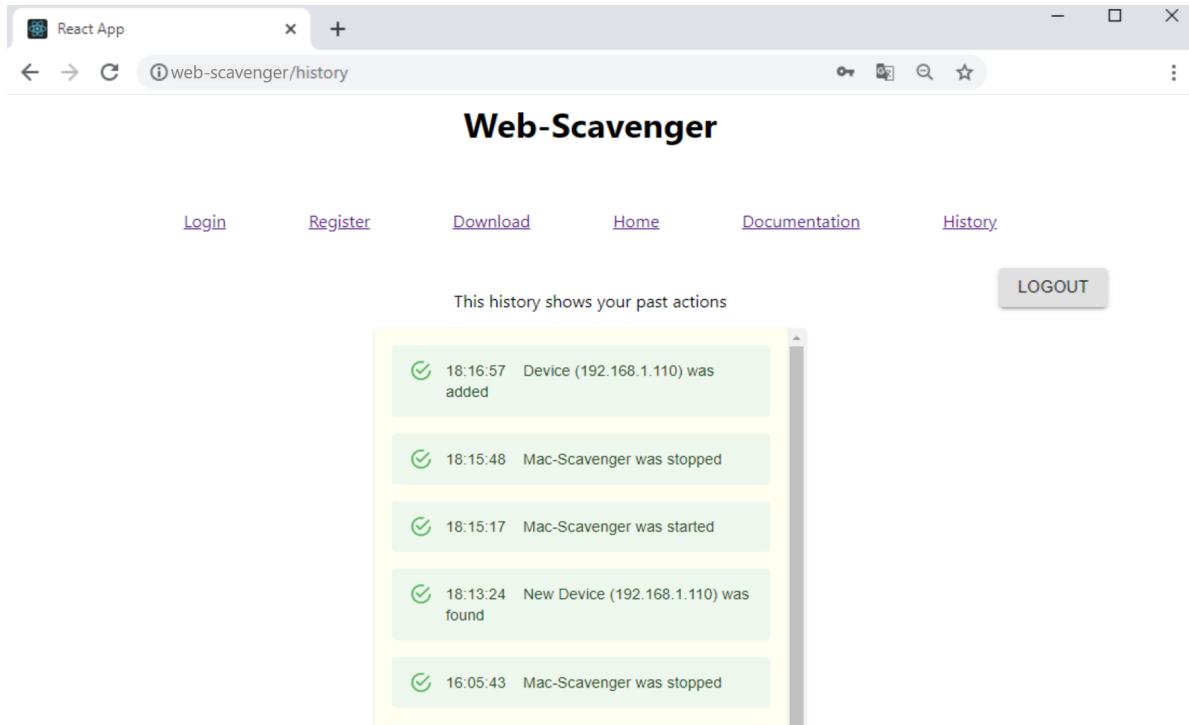


Figure 3.14: History Web-Scavenger

Search and Device Selection

The search feature is one of the most important features of the Web-Scavenger, since it allows the user to search and connect to Monitor devices without having to know the port or the IP-address. After the Desktop Application is configured, (*c.f.* Section 3.5.3) the user clicks on the Search-Button on the Web-Scavenger Dashboard (*c.f.* Figure 3.8) to initiate the search for new Monitor devices. The code displayed in Listing 3.2 shows the creation of the connection and the exchange of data from the web server's perspective.

```

1  def get_device_info():
2      server_socket = socket.socket(socket.AF_INET, socket.
          SOCK_STREAM)
3      working = True
4      #IP-address of the webserver
5      host = '#####'
6      port = 2000
7
8      server_socket.settimeout(60)
9      server_socket.bind((host, port))
10     server_socket.listen(5)
11     print("Server is listening")
12
13     timeout = time.time() + 60    # 1 minute from now
14     data=None
15
16     while True:
17         try:

```

```

18         client, address = server_socket.accept()
19         print("Connection with Client"+ str(address) +"was
           established")
20     except socket.timeout:
21         print("timeout")
22         working = False
23         break
24
25     msg = client.recv(1024)
26
27     data = pickle.loads(msg)
28     print('received data')
29
30     if address[0]:
31         client.send(bytes("OK", "utf-8"))
32         client.close()
33         data['ip_address'] = address[0]
34         break
35     if time.time() > timeout:
36         print("timeout")
37         working=False
38         break
39
40     return working, data

```

Listing 3.2: Python Socket

The *get_device_info* function is called in the beginning of the search feature. The function establishes a server socket and starts to listen for incoming traffic on a specific port. The Desktop Application knows the IP-address of the web server, since it is the predefined place where the website is hosted. Once the connection with the client has been accepted, in this scenario the Desktop Application, the server receives a message. This message contains all information necessary for the connection of the Sync program later in the data gathering pipeline. The data includes IP-address, port and name. In the end the function returns a boolean, which indicates a successful data exchange, and the data itself.

Both the boolean and the data are returned to the Service Controller, which called the *get_device_info* function. The Service Controller validates the received data to check if the device is not already in the database. To identify the user the JW-Token (JWT) is used. After the validation process, the Service Controller creates a new device object and stores it in the Web-Scavenger database. Additionally, the Service Controller updates the history accordingly.

Login

Before a user can use the Web-Scavenger, he or she needs to register on the Registration-Screen shown in Figure 3.16. A username and a password need to be entered to register. The user is required to confirm the password before pressing on register. Furthermore, the password needs to be longer than 6 characters. The username also has some restrictions. For example the username should be unique and it should not include blank spaces or be

empty. If such a restriction is violated, an appropriate error message is displayed on the screen.

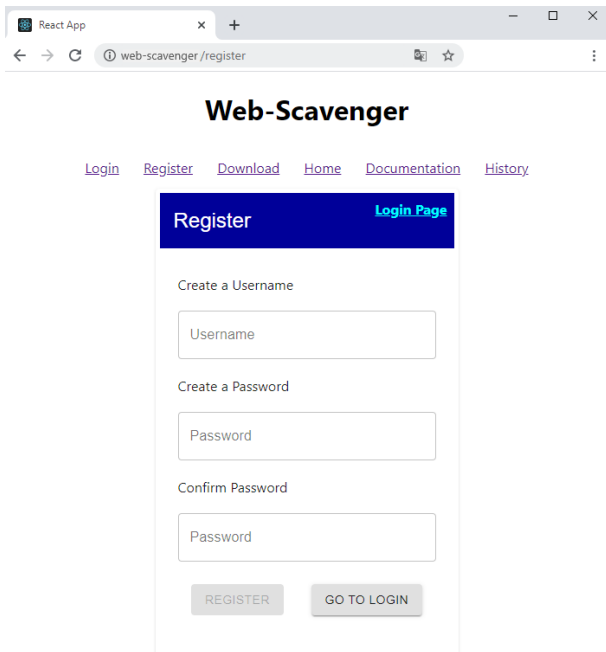
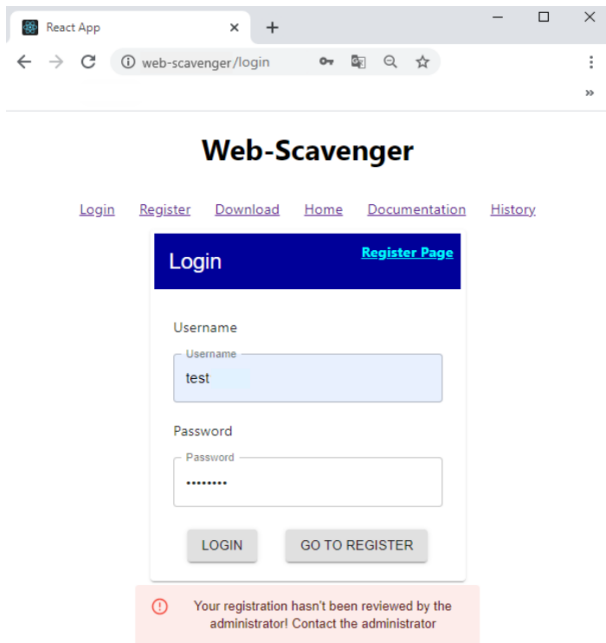


Figure 3.15: Start data gathering process Figure 3.16: No Device selected for Start

The user is directed to the Login-Screen (*c.f.* Figure 3.15) after a successful registration. The error message that is shown on the Login-Screen in Figure 3.15 stems from the fact that the administrator has not reviewed the account yet. Therefore, the administrator has to tick the box next to the user in the Dashboard depicted in Figure 3.8. Subsequently, the user with the name *test* is allowed to access the Dashboard, the Documentation-Page and the History-Page.

The access is controlled through a JWT access_token. JSON Web Tokens (JWT) signed with a secret key are the industry standard for Authorization [4]. The JWT used in the Web-Scavenger includes among other things the username and an expiration time. Each token is only valid for a specific period of time, namely 30 minutes in case of the Web-Scavenger. The `jwt_flask_extended` library, which can be installed using `pip`, uses the secret key in the website's configuration to validate the token and to make the username readable. An additional decorator, shown in Listing 3.3, was implemented to check whether the user is registered in the database. This serves as a supplementary security check.

For each request, for which Authorization is necessary, the API executes the `jwt_flask_extended` library's `jwt_required` decorator and the `restricted` decorator. Additionally, functionality for a refresh token is partly implemented.

```

1  def restricted():
2      def decorator(func):
3          @wraps(func)
4          def wrapping_func(*args, **kwargs):
5              token = request.headers
6              bear_token = token['Authorization'][7:]
7              decoded_token = jwt.decode(bear_token, app.config['
              SECRET_KEY'])
8              username = decoded_token['identity']
9
10             user = User.find_by_username(username)
11
12             if not user or not user.accepted:
13                 abort(409)
14             return func(*args, **kwargs)
15         return wrapping_func
16     return decorator

```

Listing 3.3: Custom Security Decorator

Considering that privacy violations are a potential issue according to the related work, the stored data must be protected. Functions to hash the password and to encrypt and decrypt were implemented, but are not used in the deployed version for the sake of simplicity.

```

1  @staticmethod
2  def encrypt(message: bytes, key: bytes) -> bytes:
3      return Fernet(key).encrypt(message)
4
5  @staticmethod
6  def decrypt(token: bytes, key: bytes) -> bytes:
7      return Fernet(key).decrypt(token)
8
9  @staticmethod
10 def generate_hash(password):
11     return sha256.hash(password)
12
13 @staticmethod
14 def verify_hash(password, hash):
15     return sha256.verify(password, hash)

```

Listing 3.4: Encryption and Decryption Functions

Another security issue that was resolved is the problem of the root-user-access of the physical device on which the database is located. For example, an adversary could gain access to the database, if he knew the publicly available default password of the root user. A prerequisite is that the person who uses that physical device did not disable the root user. In the Web-Scavenger the database is located on the web server. This web server is configured in a way that it can only be accessed via SSH-key and the login via credentials is disabled for all users. This provides an additional security layer to the sensitive data.

3.5.3 Desktop Application

The Desktop Application is executed on the Monitor device and has three essential purposes. The first goal of the application is to download all necessary Python libraries for the Monitor program of the MAC-Scavenger. For this process the application uses a Shell Script, which relies on the linux command *sudo* and the Python package manager *pip* to install these dependencies automatically. Therefore, the Desktop Application can only run on computers, which have installed the *sudo* command. The script also returns a custom error message in case an error occurs during the automatic download.

The second purpose of the Desktop Application is the creation of a connection to the web server where the website is hosted. This allows the automatic exchange of the device name, port and IP-address of the Monitor device. Therefore the user does not need to know the concept of the IP-address or the address itself. The connection is established through a socket similar to the Listing 3.2, but from the client side's perspective.

Finally, the last objective of the Desktop Application is the start of the Monitor program of the original MAC-Scavenger. To start this program the application has to forward all information such as name, port and IP-address automatically to the Monitor program.

The GUI of the Desktop Application is depicted in Figure 3.17. In the middle is a simple text field, where the user has to enter the name of the device. A preferred name is the location where the device will be located to gather data. In the example shown in Figure 3.17 the device will be placed at the door. If no name is entered before starting the Monitor, the application will prompt an appropriate error message to the user. After the user has entered a name he can start or stop the Monitor program. After the start button is pressed the following steps are executed automatically: Download of all libraries, establishing a connection to the web server and starting the Monitor program.



Figure 3.17: History Web-Scavenger

The Desktop Application can be downloaded from the website's Download-Screen seen in Figure 3.9 by clicking on the Web-Scavenger logo.

Chapter 4

Evaluation

This section evaluates the implementation of the Web-Scavenger against the requirements which are derived from the related work and the goal of this thesis. For each requirement, the current status in the MAC-Scavenger and the improvements made, which resulted in the Web-Scavenger, are discussed. A completely filled circle (●) next to the requirement name indicates, that the requirement is completely satisfied. If the requirement is only partially satisfied, the circle is only filled half (◐). An empty circle (○) shows that the requirement is not met by the Web-Scavenger.

Due to time limitations of this thesis, a system usability scale evaluation could not be carried out.

4.1 Requirement based evaluation

The table summarizes the evaluation in the following chapters. It can be seen that all requirements related to either Operation or Deployment are completely met by the Web-Scavenger system. Only requirement 15 is partially satisfied.

Requirement number	Category	Fulfilled
RQ 1	Deployment Requirement	●
RQ 2	Deployment Requirement	●
RQ 3	Deployment Requirement	●
RQ 4	Deployment Requirement	●
RQ 5	Deployment Requirement	●
RQ 6	Deployment Requirement	●
RQ 7	Operation Requirement	●
RQ 8	Operation Requirement	●
RQ 9	Operation Requirement	●
RQ 10	Operation Requirement	●

RQ 11	Operation Requirement	●
RQ 12	Operation Requirement	●
RQ 13	Operation Requirement	●
RQ 14	Security Requirement	●
RQ 15	Security Requirement	◐
RQ 16	Security Requirement	●
RQ 17	Security Requirement	●

Table 4.1: Requirement Evaluation Summary

4.1.1 Deployment requirements

1) GUI instead of MAC-Scavenger Shell and terminal ●

The original MAC-Scavenger relied on the terminal and the MAC-Scavenger Shell for it to be deployed. All the dependencies had to be installed manually via the terminal. It was furthermore necessary to navigate to the program in the terminal and start both the MAC-Scavenger Shell Script and the MAC-Scavenger Monitor Script through Python commands. The inputs such as the IP-address, port and name of the Monitor device needed to be entered in the terminal in the right syntax, without any instructions of the MAC-Scavenger.

In the Web-Scavenger a GUI is used instead of a terminal. The user does not need to navigate to the folder in the terminal and then start scripts. Instead, the Monitor can be started via a simple double click. The inputs are given via a GUI which provides instructions and enforces the right syntax. Additionally, the Web-Scavenger automatically inputs information, such as IP-address and port, to the MAC-Scavenger program whenever possible. The terminal is not necessary in the Web-Scavenger implementation. Consequently, this requirement is fulfilled.

2) No Git-technology necessary ●

The MAC-Scavenger project needed to be searched on Github. After the project was found on the website and Git was installed, the user had to clone into the repository via the terminal. Alternatively, the user could also use a Git-Graphical-User-Interface such as Git-Desktop. But such a GUI would have to be downloaded additionally to Git.

The improved MAC-Scavenger, alias Web-Scavenger, allows the user to click on a simple link to download the whole MAC-Scavenger and the Desktop Application of the Web-Scavenger. Git does not need to be installed or configured so that this requirement is met.

3) Easy to understand and accessible documentation ●

The main documentation in the MAC-Scavenger is a README-file in the source code. Thus, it is not accessible for an average user who wants to use the program instead of changing the source code. Additional documentation is provided in the

Master Thesis "MAC Scavenger" [6]. But the documentation in the thesis mostly discusses the mathematical concepts of the MAC-Scavenger code and provides very little practical instructions on how to use it.

Under the Documentation-Link on the website of the Web-Scavenger an extensive documentation can be found. It does not only explain how to operate the Web-Scavenger but it also provides instructions on how to configure the physical Monitor device and how to use the original MAC-Scavenger. Through the website the documentation is also accessible to every user without having to search the web and read a scientific paper. This accessible, non-scientific documentation allows for easy troubleshooting and, therefore, makes the overall deployment simpler.

4) No database setup required

The MAC-Scavenger system requires a database (*c.f.* Figure 3.4). In the original version a MongoDB was used. Since the Sync needed to be downloaded on a physical device, the MongoDB needed to be installed and configured accordingly.

The Web-Scavenger implements the Sync part of the MAC-Scavenger in its website component. The Sync program runs on the web server where the website of the Web-Scavenger is hosted. Therefore, the administrator of the web server is responsible for configuring the environment, including the MongoDB, on the web server. Hence, the average user of the Web-Scavenger does not need to setup or install the Sync program or a database.

5) Automate the Installation of Python libraries

The deployment process of the MAC-Scavenger includes the download of all Python libraries which are necessary for the Monitor and the Sync program to work. These libraries need to be identified first and then downloaded manually one by one via terminal.

In the Web-Scavenger the Sync program does not need to be downloaded at all, since it is running on the web server upon starting it through the website. The Python libraries of the Monitor program on the other hand are installed automatically through a Shell Script. The Shell Script is started once the user has defined a name for the Monitor device (*c.f.* Section 3.5.3). The user of the Web-Scavenger does not need to be concerned with the download, since it is handled automatically by the Desktop Application.

6) Deployment in less than 10 clicks

The deployment in the new Web-Scavenger is defined as the process until the *Add* command from the original MAC-Scavenger can be executed. Therefore, the *Search* command is included in the Web-Scavenger deployment (*c.f.* Figure 3.6). This calculation assumes that a user is already registered on the website and accepted by the administrator, but not logged in yet. Clicks are further defined as mouse clicks. The exact sequence of clicks is shown in the list below. The number on the left shows how many clicks are necessary up to and including the action on the right side.

- 1 Click on Login-Button.

- 2 Click on Download-Link in the navigation bar to go to the Download-Page.

- 3 Click on the Web-Scavenger Logo to download the Desktop Application.
- 5 Double click on the executable Desktop Application on the Monitor device.
- 6 Click into the input field to name the device.
- 7 Press start in the Desktop Application.
- 8 Press search on the website.

After that sequence the deployment process is finished. Subsequently, the new device appears on the screen and is added automatically per default. This can be seen by the tick next to the device on the Dashboard (*c.f.* Figure 3.8). The whole deployment of the Web-Scavenger takes only 8 clicks which satisfies this performance requirement.

4.1.2 Operation requirements

7) GUI instead of MAC-Scavenger Shell and terminal ●

In the MAC-Scavenger, a user had to know the name and the syntax of the commands to operate the program. For example he needed to know what the *Load* and *Save* commands do. He also needed to know in which sequence the different commands had to be used. All these commands were documented briefly in the README-file in the source code. Additionally, all commands needed to be used in the MAC-Scavenger Shell, which required additional knowledge of terminal commands.

The Web-Scavenger, similar to the deployment requirement, does not need to be used with the terminal. Instead, all operations can be executed over a graphical user interface. The Dashboard presented in the Implementation Section of this thesis shows the compact interface, where all commands are represented by self-explanatory buttons. If the user wants to start the data gathering process, he only needs to click on start instead of loading or adding devices first through predefined syntax. Additionally, a documentation is provided in the application itself, which helps the user to understand the sequence in which the commands need to be executed. Since the Web-Scavenger can be used entirely through the GUI this requirement is considered to be met.

8) No knowledge of network architecture required ●

In addition to the commands, a user had to have an understanding of IP-addresses and ports if he wanted to use the MAC-Scavenger. Furthermore, a user had to know how to retrieve this information from the different devices. Moreover, primitive knowledge of the Sync-Monitor-Architecture was required. For example when a user wanted to analyze or start the data gathering process, he needed to know the difference between the Sync and the Monitor program, since the functionality of these programs differs. And the user needed to know that there is only one Sync in the network and that he can't run Sync commands on the Monitor and vice versa.

Due to the fact that only the Monitor node needs to be configured in the Web-Scavenger, the user does not need to understand the architecture behind it. The

commands that can be executed are presented in the GUI, which helps to mitigate confusion. Furthermore, IP-addresses and ports are retrieved automatically by the Desktop Application of the Web-Scavenger. Through a connection with the web server this data is also automatically exchanged with the Sync. This allows a device to appear in the device section of the website without the user having to manually setup a connection between devices. Through automation, the Web-Scavenger achieved that a user does not need to deal with neither the IP-addresses nor with the network architecture.

9) Easy to understand and accessible documentation ●

The MAC-Scavenger is documented in the README-file in the source code and partially in the Master thesis "MAC Scavenger" [6]. Therefore the documentation was not really accessible for the average user, who does not want to interact with the code or a scientific paper.

The Web-Scavenger provides an extensive documentation on the website under the Documentation-Link which does not only cover the Web-Scavenger but also how to use the original MAC-Scavenger system.

10) Save and Load are automated ●

The *Save* command saves the configuration and data of a Monitor device on the Sync device in the original MAC-Scavenger [6]. But it only saves devices which are currently added to the MAC-Scavenger system. Once a Monitor device is removed, it needs to be added again, meaning the whole deployment process has to start again from the beginning. The *Load* command makes sure to load the previously saved configurations into the MAC-Scavenger. It may require a path if the devices are saved to a custom location [6].

The Web-Scavenger automates these two commands and, therefore, satisfies the 10th requirement defined in section 3.1.2. All devices are saved in the Web-Scavenger automatically after they were found by the Sync program running on the web server (*c.f.* Section 3.5.2). Even if they are not currently used in the MAC-Scavenger data gathering pipeline, they remain in the database. This means the deployment pipeline does not have to be started again from the beginning when a previously not used device wants to be loaded/added again. Once the user clicks on the Start-Button on the Web-Scavenger Dashboard all devices which are checked (have a tick next to them) are loaded into the Sync program (*c.f.* Section 3.5.2).

11) Easy troubleshooting ●

Error messages in the MAC-Scavenger are displayed in the terminal. There is no documentation provided which describes how to resolve an occurring error. Furthermore, it is not always clear what the error is referring to or if the error is just a warning message. In the future work section of the master thesis "MAC Scavenger" it is also mentioned, that error handling should be improved [6].

The Web-Scavenger solves this problem in the GUI by giving an extensive feedback through colored error messages. Additionally, warning messages and messages for successful execution of commands are displayed. The Web-Scavenger not only displays its own error messages but also forwards the messages generated from the MAC-Scavenger to the user where necessary. Together with the documentation,

which is displayed on the website, the user is enabled to find the error by himself. More and more detailed error messages, together with the documentation allow the user to identify and fix errors easily.

12) From finished deployment to analyzed results in less than 5 clicks ●

The deployment in the new Web-Scavenger is defined as the process until the *Add* command from the original MAC-Scavenger can be executed. Therefore, the *Search* command is included in the Web-Scavenger deployment (*c.f.* Figure 3.6) and everything after is considered as part of the operation. This calculation starts after all necessary devices are found by the search functionality. The number on the left shows the number of clicks that are necessary up to and including the action on the right side.

- 1 Press on Start.
- 2 Click on Stop-Button.
- 3 Press the Analyze-Button.

The Web-Scavenger only needs three simple mouse clicks from starting the data gathering process to showing the analyzed results. All these buttons are located on the same interface which allows the user to operate the Web-Scavenger efficiently.

13) Remote control of the MAC-Scavenger ●

To control the original MAC-Scavenger, the user had to be in the same location as the Sync node. In addition, the user also needed to connect a keyboard and a display to the Sync device to give commands.

The Web-Scavenger on the other hand can be controlled remotely due to the website component. The user can give commands such as start, stop and analyze through simple mouse clicks on the Web-Client. Moreover, he or she can use any device to access the website. This means that the user is able to steer the data gathering process from a laptop, cellphone or tablet. The user is not restricted to a Operating System. The remote access capability of the Web-Scavenger allows more flexibility on the user side and, thus, simplifies the usage of the original MAC-Scavenger system.

4.1.3 Security requirements

14) Token based Authentication ●

The privacy aspect was identified as a possible future challenge of passive WiFi monitoring in the related work section. The sensitive information that is gathered through the MAC-Scavenger must be protected. One part of the protection is, that only authorized people can access functionality to start, stop and analyze data. Therefore, the Web-Scavenger implements a token based Authentication.

The user can log into his account with his credentials. Upon login the user receives a token which is valid for 30 minutes. Each screen besides the Login- and Register-Screen can only be accessed with a valid token. Moreover, for each protected request to the back-end a valid token is required. The token holds information such as the

expiration time, the username and a secret key. If the token has not expired, the username is present in the database and the secret key is correct, the token is considered valid and access is granted. The Web-Scavenger introduced Authentication, which was in-existent in the original MAC-Scavenger.

15) Encrypted data

In the original MAC-Scavenger the data was stored in clear text in the database. This is a potential security risk, because a potential security breach gives the adversary easy access to the stored data.

To mitigate the risk of a data leak, the Web-Scavenger implements functions to encrypt and decrypt data. The key, with which the data is encrypted, is randomly generated by the cryptography library in Python. Both the *encrypt* and *decrypt* function are implemented (*c.f.* Listing 3.4) but are not used in the deployed Web-Scavenger for the sake of simplicity. In addition to encrypting the data, passwords are hashed through a hash function. This makes it more difficult to retrieve the actual password from the database in case the database is hacked. The function is implemented, but not used for the same reason as the *encrypt* function.

16) Administrator view

The administrator can restrict access for users of type *Guest* through his own administrator view (*c.f.* Figure 3.8). This procedure secures the data even more, since it imposes another barrier to access the data. The administrator view in the Web-Scavenger only differs slightly from the normal view. The difference is that the Card on the left on Figure 3.8 is only visible for users of type *Admin*. That is why this requirement is satisfied.

17) SSH-key instead of default root user login

In the original MAC-Scavenger, the the gathered data was located on the device where the Sync was running. This device was considered to be a Tinkerboard. The Tinkerboard comes with a publicly known default password for the root user. Changing the password or creating a new user is a difficult process for an average user. And even when the password was changed, the device could still be vulnerable to brute force attacks, since there is no limit for password trials.

The Web-Scavenger has all its data stored on the web server, which is also a Tinkerboard. To secure the Tinkerboard not only was a new user created, but also the access via password is disabled. The Tinkerboard can only be accessed through a SSH-key, which makes the Web-Scavenger safer than its predecessor.

Chapter 5

Summary and Conclusions

The MAC-Scavenger is responsible for device fingerprinting. This means the MAC-Scavenger system can circumvent the MAC randomization countermeasure and uniquely identify devices and then track them over the area covered by the network [6].

The goal of this thesis was to simplify the operation and deployment of the MAC-Scavenger in a way that only a small number of configuration steps have to be executed manually. To achieve that, requirements for an improved program were derived from the analysis of related work and the goal of this thesis. The main requirement was to develop a graphical user interface with which the user can interact while the hidden code handles most operations automatically in the background. This thesis presents the architecture and implementation of the developed system. In the end, it compares the new system against the requirements and shows that 94% of the requirements are fulfilled completely.

Due to the Sync-Monitor network architecture and the requirements a web based approach was chosen. Therefore, the new system was named Web-Scavenger. The Web-Scavenger consists of two main components: A Web component and a Desktop Application. The Web component, which is the website, interacts with the Sync program of the original MAC-Scavenger and automates commands such as *Save* and *Load*. It also simplifies commands by automatically inserting the correct input parameters such as the IP-address and port for the original *Add* command.

Another focus of the website lies on securely handling sensitive data, which was identified as a possible challenge of passive wireless tracking in the related work section. The Web component uses hash-functions, token based Authorization and encryption to securely handle gathered data.

The Desktop Application helps the Web-Scavenger to exchange input data for the Sync automatically with the Web component. Additionally, it provides a graphical user interface for the remaining manual configuration on the Monitor device and allows to download dependencies automatically.

The evaluation of the Web-Scavenger shows that the majority of requirements were achieved by the new system. It further shows that the Web-Scavenger provides a non-complex graphical user interface for the MAC-Scavenger and automates many tasks which

had to be executed manually in the MAC-Scavenger. To summarize, the Web-Scavenger simplifies the MAC-Scavenger system and makes the wireless tracking technology understandable and accessible for a wider audience.

5.1 Future Work

The legal aspect of wireless passive tracking might become a big challenge in the future, when technologies continue to evolve. This concern is also expressed in multiple related theses. But currently there exists only very few research on legal aspects and privacy rights regarding the passive wireless monitoring technology.

In recent years, some substitutes came to market which might rival the monitoring technology described in this work in the future. Eden Senses, an ETH spin-off, received a contract from the Swiss Public Transport Authority (SBB), to install their air Monitor devices in trains in the year 2020 [32]. Those Monitors measure humidity, concentration of CO₂ and more key indicators in the air. Through computer algorithms, these input factors result in an accurate estimate of the number of people in a train wagon [32]. In comparison to passive wireless monitoring, there applies no data protection regulation to the input factor air. Future research can bring value by comparing both data measuring methods.

Further research might also focus on the Web-Scavenger itself. The Web-Scavenger strongly relies on the MAC-Scavenger which was designed as a proof of concept. Therefore, improvements in accuracy or error handling in the MAC-Scavenger can be advantageous to the Web-Scavenger.

Bibliography

- [1] IEEE Standards Association. *History of IEEE*. [Online]. Available: <https://www.ieee.org> (Visited on Oct. 13, 2020).
- [2] IEEE Standards Association. *MAC Address Block Large (MA-L)*. [Online]. Available: <https://standards.ieee.org/MA-L> (Visited on Oct. 26, 2020).
- [3] IEEE Standards Association. *Company ID (CID)*. [Online]. Available: <https://standards.ieee.org/CID> (Visited on Oct. 26, 2020).
- [4] Auth0. *Introduction to JSON Web Tokens*. [Online]. Available: <https://jwt.io/introduction> (Visited on Feb. 2, 2021).
- [5] Barbera, Marco V and Epasto, Alessandro and Mei, Alessandro and Pertta, Vasile C and Stefa, Julinda. *Signals from the crowd: uncovering social relationships through smartphone probes in Proceedings of the 2013 conference on Internet measurement conference*. 2013. p. 265-276. [Online]. Available: <https://dl.acm.org/doi>.
- [6] Baumann, Lenz. *MAC Scavenger A Passive Method Handling MAC Randomization on Mobile Devices*. Master thesis, University of Zurich, 2020. [Online]. Available: <https://files.ifi.uzh.ch>.
- [7] Betzing, Jan Hendrik. *Beacon-based customer tracking across the high street: perspectives for location-based smart services in retail*. Completed Research, University of Munster, 2018. [Online]. Available: <https://www.smartmarketsquare.de>.
- [8] Bianchi, Giuseppe and Fratta, Luigi and Oliveri, Matteo. *Performance evaluation and enhancement of the CSMA/CA MAC protocol for 802.11 wireless LANs in Proceedings of PIMRC'96-7th International Symposium on Personal, Indoor, and Mobile Communications*. 1996. p. 392-396. [Online]. Available: <https://ieeexplore.ieee.org>.
- [9] SQLAlchemy authors and contributors. *SQLAlchemy*. [Online]. Available: <https://www.sqlalchemy.org/> (Visited on Jan. 23, 2021).
- [10] Per Christensson. *User-Friendly*. [Online]. Available: <https://techterms.com> (Visited on Jan. 29, 2021).
- [11] Day, John D and Zimmermann, Hubert. *The OSI reference model in Proceedings of the IEEE*. 1983. p. 1334-1340. [Online]. Available: <https://ieeexplore.ieee.org>.

- [12] Deng, Der-Jiunn and Chen, Kwang-Cheng and Cheng, Rung-Shiang. *IEEE 802.11 ax: Next generation wireless local area networks in 10Th international conference on heterogeneous networking for quality, reliability, security and robustness*. 2014. p. 77-82. [Online]. Available: <https://ieeexplore.ieee.org>.
- [13] Eastlake, Donald and Abley, J. *IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters in Febrero*. 2013. p. 0-27. [Online]. Available: <https://www.hjp.at>.
- [14] Steve Emms. *Review: Asus Tinker Board S Single-Board Computer*. [Online]. Available: <https://www.linuxlinks.com> (Visited on Jan. 11, 2021).
- [15] F5, Inc. *What is NGINX?*. [Online]. Available: <https://www.nginx.com> (Visited on Jan. 11, 2021).
- [16] Facebook Inc.. *React*. [Online]. Available: <https://reactjs.org> (Visited on Jan. 11, 2021).
- [17] Gast, Matthew. *802.11 wireless networks: the definitive guide*. O'Reilly Media, Inc. 2005.
- [18] Gu, Xiaolin and Wu, Wenjia and Gu, Xiaodan and Ling, Zhen and Yang, Ming and Song, Aibo. *Probe Request Based Device Identification Attack and Defense in Sensors*. 2020. p. 1-5. [Online]. Available: <https://www.mdpi.com>.
- [19] Halter, Cyrill. *BluePIL: Fully Passive Identification and Localization of Bluetooth Devices in Near-Real-Time*. Ph.D. dissertation, University of Zurich, 2020. [Online]. Available: <https://files.ifi.uzh.ch>.
- [20] ASUSTeK Computer Inc.. *tinker board, tinker your way to the future*. [Online]. Available: <https://www.asus.com> (Visited on Jan. 11, 2021).
- [21] Kalogianni, Eftychia and Sileryte, Rusne and Lam, Marco and Zhou, Kaixuan and Van der Ham, Martijn and Van der Spek, S and Verbree. *Passive wifi monitoring of the rhythm of the campus in Proceedings of The 18th AGILE International Conference on Geographic Information Science*. 2015. p. 9-14. [Online]. Available: <https://d1wqtxts1xzle7.cloudfront.net>.
- [22] Kamerman, Ad and Monteban, Leo. *WaveLAN®-II: a high-performance wireless LAN for the unlicensed band in Bell Labs technical journal*. 1997. p. 118-133. [Online]. Available: <https://onlinelibrary.wiley.com>.
- [23] Kuo, Franklin F. *Computer Networks—The ALOHA System*. Hawaii University Tech. Rep. 1981. [Online]. Available: <https://apps.dtic.mil>.
- [24] Unknown. *Gunicorn*. [Online]. Available: <https://gunicorn.org/> (Visited on Jan. 11, 2021).
- [25] Kushal Das. *Introduction to Flask*. [Online]. Available: <https://pymbook.readthedocs.io> (Visited on Jan. 11, 2021).

- [26] Linode. *What is systemd?*. [Online]. Available: <https://www.linode.com> (Visited on Jan. 11, 2021).
- [27] Martin, Jeremy and Mayberry, Travis and Donahue, Collin and Foppe, Lucas and Brown, Lamont and Riggins, Chadwick and Rye, Erik C and Brown, Dane. *A study of MAC address randomization in mobile devices and when it fails* in *Proceedings on Privacy Enhancing Technologies*. 2017. p. 365-383. [Online]. Available: <https://content.sciendo.com>.
- [28] Matte, Célestin. *Wi-Fi tracking: Fingerprinting attacks and counter-measures*. Ph.D. dissertation, Université de Lyon, 2017. [Online]. Available: <https://tel.archives-ouvertes.fr>.
- [29] Microsoft. *What is TypeScript?*. [Online]. Available: <https://www.typescriptlang.org> (Visited on Jan. 18, 2021).
- [30] Minnaard, Wicher. *Out of sight, but not out of mind: Traces of nearby devices' wireless transmissions in volatile memory* in *Digital Investigation*. 2014. p. 104-111. [Online]. Available: <https://www.sciencedirect.com>.
- [31] MongoDB, Inc.. *What Is MongoDB?*. [Online]. Available: <https://www.mongodb.com> (Visited on Jan. 18, 2021).
- [32] Moritz Kaufmann. *Mit Daten dieses Startups wollen die SBB ihre Züge diskret überwachen*. [Online]. Available: <https://nzzas.nzz.ch> (Visited on Feb. 15, 2021).
- [33] Musa, ABM and Eriksson, Jakob. *Tracking unmodified smartphones using wi-fi monitors* in *Proceedings of the 10th ACM conference on embedded network sensor systems*. 2012. p. 281-294. [Online]. Available: <https://dl.acm.org/doi>.
- [34] Myrvoll, Tor A and Håkegård, Jan E and Matsui, Tomoko and Septier, François. *Counting public transport passenger using WiFi signatures of mobile devices* in *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017. p. 1-6. [Online]. Available: <https://ieeexplore.ieee.org>.
- [35] Per Christensson. *FIFO*. [Online]. Available: <https://techterms.com> (Visited on Feb. 10, 2021).
- [36] Edward C. Prem. *Wireless Local Area Networks* in *Recent Advances in Networking Paper, Ohio State University*. 1997. p. 1-12. [Online]. Available: <http://www.netlab.boun.edu.tr>.
- [37] PyPA. *pip - The Python Package Installer*. [Online]. Available: <https://pip.pypa.io> (Visited on Jan. 11, 2021).
- [38] Python Software Foundation. *SQLAlchemy 1.3.23*. [Online]. Available: <https://pypi.org> (Visited on Jan. 11, 2021).
- [39] Python Software Foundation. *What is Python? Executive Summary*. [Online]. Available: <https://www.python.org> (Visited on Jan. 11, 2021).

- [40] Rech, Jörg. *Wireless LANs: 802.11-WLAN-Technologie und praktische Umsetzung im Detail*. Heise Verlag. 2012.
- [41] Rountree, Derrick. *Security for Microsoft Windows system administrators: introduction to key information security concepts*. Elsevier. 2011.
- [42] Scaleway SAS. *How to Configure a Nginx HTTPs Reverse Proxy on Ubuntu Bionic*. [Online]. Available: <https://www.scaleway.com> (Visited on Jan. 11, 2021).
- [43] Scheuner, Joel and Mazlami, Genc and Schöni, Dominik and Stephan, Sebastian and De Carli, Alessandro and Bocek, Thomas and Stiller, Burkhard. *Probr-a generic and passive WiFi tracking system in 2016 IEEE 41st Conference on Local Computer Networks (LCN)*. 2016. p. 495-502. [Online]. Available: <https://ieeexplore.ieee.org>.
- [44] Schiller, Jochen H.. *Mobile communications*. Pearson education. 2003.
- [45] Vaccari, Andrea and Samouhos, Stephen and Glicksman, Leon and Ratti, Carlo. *MIT enernet: correlating WiFi activity to human occupancy in Proceedings of Healthy Buildings*. 2019. p. 1-5. [Online]. Available: <https://senseable.mit.edu>.
- [46] Vanhoef, Mathy and Matte, Célestin and Cunche, Mathieu and Cardoso, Leonardo S and Piessens, Frank. *Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms in Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. 2016. p. 413-424. [Online]. Available: <https://dl.acm.org/doi>.
- [47] Vu, Long and Nahrstedt, Klara and Retika, Samuel and Gupta, Indranil. *Joint bluetooth/wifi scanning framework for characterizing and leveraging people movement in university campus in Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*. 2010. p. 257-265. [Online]. Available: <https://dl.acm.org/doi>.
- [48] Wallen, Jack. *10 things that make software user-friendly*. [Online]. Available: <https://www.techrepublic.com> (Visited on Jan. 29, 2021).
- [49] Wireshark. *tshark*. [Online]. Available: <https://www.wireshark.org/docs> (Visited on Jan. 29, 2021).
- [50] Zdnet, A Red Ventures Company. *About Us*. [Online]. Available: <https://www.techrepublic.com> (Visited on Jan. 29, 2021).

Abbreviations

ACK	Acknowledgment (Frame)
AP	Access Point
API	Application Programming Interface
BSS	Basic Service Set
BSSID	Basic Service Set Identification
BTBR/EBR	Bluetooth Basic Rate / Enhanced Data Rate
CCA	Clear Channel Assessment
CDMA	Code Division Multiple Access
CF _{end}	Contention Free End
CID	Company Identifier
CRC	Cyclic Redundancy Code
CSMA/CA	Carrier Sense Multiple Access Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access Collision Detection
CTS	Clear-To-Send (Frame)
DCF	Distributed Coordination Function
DFS	Dynamic Frequency Selection
DFWMAC-DCF	Distributed Foundation Wireless Medium Access Control
DHCP	Dynamic Host Configuration Protocol
DIFS	Distributed (Coordination Function) Interframe Space
DSL	Digital Subscriber Line
DS	Distribution System
DSSS	Direct Sequence Spread Spectrum
ESS	Extended Service Set
EUI	Extended Unique Identifiers
FCS	Frame Check Sequence
FDMA	Frequency Division Multiple Access
FHSS	Frequency Hopping Spread Spectrum
FIFO	First In First Out
GAA	Globally Administered Address
GHz	Gigahertz
IAPP	Inter Access Point Protocol
IBM	International Business Machine
IBSS	Independent Basic Service Set
IEEE	Institute of Electrical and Electronics Engineers
IFS	Interframe Space
I/G (Bit)	Individual/Group (Bit)

IP-address	Internet Protocol Address
ISM-Band	Industrial, Scientific, Medical -Band
JWT	Json Web Token
LAA	Locally Administered Address
LAN	Local Area Network
MAC	Media Access Control
NAT	Network Address Translation
NAV	Network Allocation Vector
OFDM	Orthogonal Frequency Division Multiplexing
ORM	Object Relational Mapper
OSI model	Open Systems Interconnection model
OUI	Organizationally Unique Identifier
PCF	Point Coordination Function
PIFS	Point Interframe Space
PIP	Preferred Installer Program
PLCP	Physical Layer Convergence Protocol
PMD	Physical Medium Dependent
PS	Power Save (Mode)
RA	(IEEE) Registration Authority
RSSI	Received Signal Strength Indicator
RTS	Request-To-Send (Frame)
SAP	Service Access Point
SBB	Schweizer Bundesbahnen
SDMA	Space Division Multiple Access
SIFS	Short Interframe Space
SIG	(Bluetooth) Special Interest Group
SSID	Service Set Identifier
TDMA	Time Division Multiple Access
TPC	Transmission Control Protocol
U/L (Bit)	Universal/Local (Bit)
URL	Uniform Resource Locator
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network

List of Figures

2.1	Ad-hoc Network. Figure by [44].	9
2.2	Infrastructure Network. Figure by [40].	9
2.3	OSI model. Figure by [11]	9
2.4	DFWMAC-DCF CSMA/CA. Figure by [44]	12
2.5	Hidden-Station-Problem. Figure by [40]	12
2.6	DFWMAC-DCF RTS/CTS extension. Figure by [44]	13
2.7	802.11 MAC frame structure. Figure by [40]	14
2.8	802.11 Frame Control structure. Figure by [40]	15
2.9	802.11 Probe Request Frame. Figure by [17]	16
2.10	Information Elements (IE). Figure by [46]	16
2.11	EUI-48 structure. Figure by [3]	17
2.12	OUI-24 structure. Figure by [3]	18
2.13	CID structure. Figure by [3]	18
2.14	MAC-Scavenger Implementation. Figure by [6]	20
3.1	Data Gathering Pipeline on the Sync nodes. Figure by [6]	32
3.2	Data Gathering Pipeline on the Monitor nodes. Figure by [6]	32
3.3	Interaction of App- and Web-components of the Web-Scavenger	33
3.4	High-Level Architecture of the Web-Scavenger Website Component	34
3.5	Database structure of Web-Scavenger-database	35
3.6	Process to deploy and operate the MAC-Scavenger	37
3.7	Technologies used in Web-Scavenger components	39

3.8	Dashboard of the Web-Scavenger	41
3.9	Download-Page Web-Scavenger	42
3.10	Download-Page Web-Scavenger	43
3.11	Start data gathering process	45
3.12	No Device selected for Start	45
3.13	Analyze data	45
3.14	History Web-Scavenger	46
3.15	Start data gathering process	48
3.16	No Device selected for Start	48
3.17	History Web-Scavenger	50
B.1	Example Tinkerboard. Figure by [14]	78

List of Tables

2.1	802.11 Protocol Family	7
2.2	OSI model layers	10
3.1	Features of the Web-Scavenger	39
4.1	Requirement Evaluation Summary	54

Listings

3.1	Start Data Gathering Process	44
3.2	Python Socket	46
3.3	Custom Security Decorator	49
3.4	Encryption and Decryption Functions	49

Appendix A

Web-Scavenger Githlab Repository

The source code of the Web-Scavenger can be found on Gitlab under the following link:

`https://gitlab.com/search?utf8=E29C93&search=web-scavenger&group\_id=&project\_id=&snippets=false&repository\_ref=&nav\_source=navbar`

Appendix B

Documentation

The following pages show the description of the Web-Scavenger which allows the user to safely operate it. This description is an essential part of the deployment because it answers common questions in an easy way. These explanations can also be found on the Web-Scavenger Website under the title Documentation.

This document details how to start and operate the Web-Scavenger. Step one focuses on preparing the monitoring devices, step two explains how to get your individual Web-Scavenger up and running. If you already have setup your Tinkerboard or not using a Tinkerboard as Monitor you can skip the first step. If you don't know what a Tinkerboard is, it is recommended to read through step one.

Step one: Tinkerboard setup

A Tinkerboard is a single board computer with class-leading performance [20]. A picture of a Tinkerboard is depicted below.

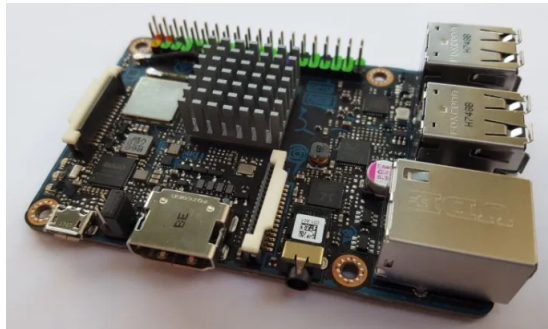


Figure B.1: Example Tinkerboard. Figure by [14]

The configuration described in this step is applicable to the Tinkerboard as a monitor. But every device with a linux operating system can be used.

- **Change keyboard layout**

If you are not familiar with the US-keyboard layout it is advantageous to change it. To do so go to the terminal, by clicking on in the lower left corner and then "System Tools" $\Rightarrow$ "LXTerminal". After that follow the instructions below:

- Enter the following and press enter: `sudo dpkg-reconfigure keyboard-configuration` (- is represented by ? in the US-keyboard layout)
- Select: Generic keyboard with 105 key (Intl)
- Select:"Other" language
- Select your language
- If necessary further specify your language (Choose Switzerland again)
- Select the default keyboard layout
- Select: "No compose key"
- Select: "No" for a combination of "control+Alt+Backspace"
- Reboot the system by clicking in the lower left corner and then "logout" $\Rightarrow$ "reboot"

- **Connect to the Internet**

Alternatively to a connection through Wifi an ethernet connection can be established by plugging in the ethernet cable.

- Click in the right corner on the connect symbol
- Select your Wifi
- Enter your password (Attention: Make sure to have the right keyboard layout first)

- **Download pip3**

Pip3 might already be installed depending on your operating system version. Pip3 is the package manager for Python which allows to install Python packages [37] which are needed for the Web- and the Mac-Scavenger. Depending on what Python versions you have installed you can omit the 3 in the following commands. If you haven't updated the Python version the commands below are the ones which should be used in the LXTerminal.

- Check if pip3 is installed: `pip3 - -version`
- Install pip3 with: `sudo apt-get install python3-pip`
- Restart terminal
- Check if pip3 is installed: `pip3 - -version`

After these steps your Tinkerboard is ready to use the Web-Scavenger. But if you want to use the original Mac-Scavenger System and control it over the terminal you can do that as well. A short description on how to do that will be given in the end of step 2.

Step two: Prepare Web-Scavenger and operate it

Before starting with step two, make sure to finish step one. In the beginning of this section the usage of the Web-Scavenger is explained. In the end this documentation gives a short explanation on how to use the Mac-Scavenger without the Web-Scavenger.

- **Register and login**

Go to the website of the Web-Scavenger and register yourself with a username and a password. Once the Admin has approved you, you will be able to log into your account and use all functionalities.

- **Download Desktop App**

With your monitoring device log into your account and navigate to the Download section by clicking on Download. Once you are on the page click on the Web-Scavenger picture to initiate the download. After the download has finished unzip the content and go to the dist folder and start the application by double clicking on the executable. If your Anti-Virus-Software does not allow the execution just start the App.py file in the root folder via the terminal.

- **Use the Desktop App**

Once the App opens give the monitor a name. Make sure to enter a name otherwise a warning will be shown. Also, the name should not consist of blanks and not be longer than 50 characters. An appropriate name would be the place where you want to place the monitor. For example "door" or "window". You can also terminate the app by clicking in the upper right corner or by clicking on "stop"

- **Start Desktop App**

After you entered a name and pressed start, the app will initiate the download of all necessary packages for the monitoring device. Do not interrupt it.

- **Control Monitors**

While the monitor is downloading the dependencies which can take between 1-3 hours, you can setup the other monitors in the mean time. Once the download of the monitor has finished. Log into the Web-Scavenger Website and press "Search". Your monitors should appear in the window above with the correct name and the "IP-address".

- **Position Monitor**

Once the monitor-devices (Tinkerboards) are registered on the website, you can disconnect them from a monitor and put them into the location where you want them to gather data.

- **Start Data gathering**

By pressing on the Start-Button the monitors start gathering data. A successful start should be prompted to you by a green message. Furthermore should the Stop-Button be enabled after the start.

- **Stop and analyzing**

The advantage of the Web-Scavenger is that you can analyze the data while the gathering process takes place. For this just press on the Analyze-Button. The whole data gathering process can be stopped by clicking on the Stop-Button.

- **Check History**

The history shows you your previous actions and also if any errors occurred and gives you tips how to avoid or solve them.

- **Logout**

After you starting, analyzing or searching for devices, it is always good to logout to protect sensible information beeing leaked to others. To do this, use the Logout-Button.

The above section covered the use of the Web-Scavenger in detail. The part below shortly describes how to operate the Mac-Scavenger in its original form. Keep in mind that additional configurations and downloads need to be made to get the Mac-Scavenger working. To use the Mac-Scavenger download the App from the website to your Sync and to your Monitor devices by clicking on the Web-Scavenger logo or clone it (Install git beforehand). After that install all dependencies for the Sync and the monitor. Start the sync by executing the MacScavengerShell.py in the command line. To use the Monitor execute the following command: `python3 MacScavengerMonitor name port`. To add the device and save it for later use the following command in the Shell in the Sync: `add name IP-address port` or load previously saved devices by using the `load` command. After that you can use "start" to start the gathering process.