University of Zurich[UZH]

MASTER THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

WiFi-based Crowd Safety System

Nicolas Spielmann
Zurich, Switzerland
Student ID: 15-704-505

Supervisor: Dr. Bruno Rodrigues, Eder Scheid, Prof. Dr. Burkhard Stiller, Simon Tuck
Date of Submission: January 27, 2022

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Master Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

The surveillance of the density of crowds has always been important for safety reasons. In recent years, this topic has evolved to provide business with important information about their potential customers. During the outbreak of the SARS-CoV-2 virus, there were regulations requiring persons to social distance. Such a crowd density monitoring system could be used to ensure social distancing.

This thesis proposes a person counting approach based on the Wi-Fi packets emitted by smartphones. It utilizes Wi-Fi sniffers, which monitor Wi-Fi traffic emitted by smartphone with WLAN enabled. The system is completely passive. The system uses RSSI values to estimate the distance between the Wi-Fi sniffers and the device, to then locate the device. During two experiments conducted indoor, it was evaluated, whether the capturing of packets works and whether the RSSI readings can be used to estimate the position of the smartphone. The proposed system was able to capture data emitted by smartphones. The RSSI values obtained indoor indicate, that the RSSI values are not precise enough for reliable trilateration.

Zusammenfassung

Die Überwachung von Menschenansammlungen war aus Sicherheitsgründen schon immer wichtig. In den letzten Jahren hat sich dieses Thema weiterentwickelt, um Unternehmen wichtige Informationen über ihre potenziellen Kunden zu liefern. Während des Ausbruchs des SARS-CoV-2-Virus gab es Vorschriften, die von Personen verlangten soziale Distanz zu bewahren. Ein solches System zur Überwachung der Personendichte könnte dazu verwendet werden, die soziale Distanz zu gewährleisten.

In dieser Arbeit wird ein Ansatz zur Personenzählung vorgeschlagen, der auf den von Smartphones gesendeten Wi-Fi-Paketen basiert. Dazu werden Wi-Fi-Sniffer eingesetzt, die den Wi-Fi-Verkehr überwachen, der von Smartphones mit aktiviertem WLAN gesendet wird. Das System ist komplett passiv. Das System verwendet RSSI-Werte(empfangene Signalstärke), um die Entfernung zwischen den Wi-Fi-Sniffern und dem Gerät zu schätzen und das Gerät dann zu lokalisieren. In zwei Experimenten, die in Innenräumen durchgeführt wurden, wurde evaluiert, ob das Aufnehmen von Paketen funktioniert und ob die RSSI-Messwerte zur Schätzung der Position des Smartphones verwendet werden können. Das vorgeschlagene System war in der Lage, die von Smartphones ausgestrahlten Daten zu erfassen. Die in Innenräumen ermittelten RSSI-Werte zeigen, dass die RSSI-Werte für eine zuverlässige Trilateration nicht genau genug sind.

Acknowledgments

I would like to express my sincerest gratitude to my supervisor Dr. Bruno Rodrigues for providing invaluable feedback and support in the process of my master thesis. I would also like to thank Simon Tuck for providing Wi-Fi sensing hardware. Last, I would like to thank Prof. Dr. Burkhard Stiller for the opportunity to realize my master thesis at the Communication Systems Group at the University of Zurich.

Contents

Chapter 1

Introduction

1.1 Motivation

Analysis of movement, capacity, and engagement is not only essential for the strategic planning of live marketing initiatives in retail spaces, trade shows, and promotional activities, but it is increasingly essential during critical health crises such as that of COVID-19, where social distancing and safe occupancy rates are a major concern. The current situation indicates that any step toward re-establishing society's regular economic activities shall be performed very carefully by health authorities and governments to prevent new infection waves. Thus, the use of existing and already rolled-out technology is essential and almost the only way (a) to crowd-source information concerning the health of individuals and (b) to ensure that social distancing rules are being respected. In this context, technical solutions that ensure compliance with regulatory requirements and provide valuable analytic and insights will become relevant for society and businesses to re-establish their normal flow. Operational benefits in digital tracking tools include the improved data quality and capacity to trace contacts of a more significant number of people/devices in a shorter period, including the ability to provide real-time situation awareness.

This thesis aims to provide an implementation of a real-time crowd monitoring system to ensure physical distancing in crowds using Wi-Fi sensors. The advantages of using Wi-Fi sensors are that they are significantly more cost-effective for comprehensive area coverage than comparable systems (for example, a camera-based crowd monitoring solution). Also, it is considerably easier to install and configure than similar systems (for example, a camera-based crowd monitoring solution). Furthermore, it is crucial to successfully map the mobile devices to the persons that should be tracked since one person might carry multiple devices or no device at all. It might also be possible to determine the position of a device more precisely whenever multiple sensors pick it up by using the signal strength to interpolate its location. This should then be visualized to give an overview of the current state of the crowd in the sensed area. Furthermore, whenever the number of persons in a given area is close to a certain threshold, there should be an automated alerting mechanism to prevent overcrowding.

1.2 Thesis Goals

The goals of this thesis are of different types of goals. The research goals of this thesis consist of following:

- Overview the state-of-the-art and define an algorithm for calculating the density thresholds in terms of signal strength and the number of signals captured. Involves expanding the scope of analysis of related work listed in this description, seeking to define a method and associated algorithms for crowd density assessment.
- Analyze and define a strategy for filtering Wi-Fi packets from previously captured sources. This involves analyzing the use of motion prediction algorithms (e.g., kalman filter) and defining heuristics that allow predicting with estimated error rate the population density of a space bounded by Wi-Fi coverage.
- Discover an optimal density of sensor coverage required for such a system and the hardware in use. In other words, define what is the optimal number of Wi-Fi sensors per 100 m2 further considering constraints of the hardware in use.

The engineering goals are:

- Implement a streaming data pipeline from the sensor fleet to the application, involving the filtering of Wi-Fi signals that are not relevant. For example, beacon frames, multicast MAC addresses, and OUIs that do not match smartphone manufacturers.
- Implement an alerting mechanism for sensors whose crowd density value exceeds a predefined safety threshold. The relevant application parameters should be configurable. E.g., the position of each sensor, crowd density thresholds, etc.

The developed prototype then needs to be evaluated concerning the analysis of crowd density and its error rate, and contrasted with results existing in the state-of-the-art. Further, the prototype should be contrasted with a ground truth verification of the crowd density (e.g., by using an alternative method to count the number of people in a given area - such as the restaurant in the university).

1.3 Methodology

The research goals of this thesis should be reached by doing research on the state of the art crowd monitor solutions. Furthermore, there needs to be done research on the fundamentals of wireless local area networks(WLAN), methods used to locate objects and other approaches to estimate crowd density.

To evaluate the crowd counting approach, multiple experiments should be conducted, where a ground truth is observed. This ground truth should the be used tweak the counting algorithm.

1.4 Thesis Outline

The thesis is structured as follows: Chapter 2 introduces the basics of WLAN, media access in Wi-Fi, Trilateration and discusses the current state of the art approaches in crowd monitoring systems. Chapter 3 describes the solution approach and how the system was designed. Chapter 4 describes how the designed system was implemented. Chapter 5 evaluates the implemented approach and decisions taken. Chapter 6 summarizes the contribution made by this thesis and outlines possible future research approaches and improvements.

Chapter 2

Fundamentals

2.1 Background

2.1.1 Wireless Communication

Wireless communication refers to the transfer of data or information between multiple location that does not use an optical or electrical conductor that functions as a medium to transfer the data. The most common technology used for wireless communication is the usage of electromagnetic radiation, so called radio waves.

These radio waves are divided into different frequency bands. The ultra high frequency (UHF) band, which ranges from 300 MHz to 3 GHz, and the super high frequency (SHF) band, which ranges from 3 to 30 GHz, are the frequency bands, which are used for Wi-Fi. The specific frequency bands used by Wi-Fi are the 2.4 GHz, 5 GHz, 6 GHz and 60 GHz bands.

These frequency bands are then divided into overlapping channels, to reduce interference. In the 2.4 GHz band, there are 14 channels with a spacing of 5 MHz except channel 14, which has a space of 12 MHz. Table 2.1 shows a list of channels used in the 2.4 GHz Wi-Fi. Channel 1 to 13 are commonly used in Europe, USA and Canada allow the usage of channel 1 to 11. Japan uses channel 1 to 14[1].

2.1.2 Wi-Fi

The Institute of Electrical and Electronics Engineers(IEEE) defines the 802 family of standards, which describe the standards and recommendations for networking. Such standards include Ethernet, LAN and WLAN. The family of 802.11 standards define Protocols on which WLAN is based. This standard is continuously evolving to increase performance and stability[2]. Wi-Fi is a certification done by the Wi-Fi Alliance based on the family of IEEE-802.11 standards. Therefore all Wi-Fi certified products are 802.11 compliant[3].

Channel	Frequency (GHz)
1	2.412
2	2.417
3	2.422
4	2.427
5	2.432
6	2.437
7	2.442
8	2.447
9	2.452
10	2.457
11	2.462
12	2.467
13	2.472
14	2.484

Table 2.1: 2.4 GHz channels

2.1.3 802.11 MAC

The 802.11 specification to establish network connections using wireless connections was invented by using the already defined standards for wired networking and evolve these to master the challenges introduced by wireless transmission of data. Therefore the media access control (MAC) in WLAN has been adapted to tackle the wireless nature of transmission. This included the adaption of the frame format. One of which was the decision to use four address fields. The 802.11 frame is defined as follows[1].

802.11 MAC Frame Format

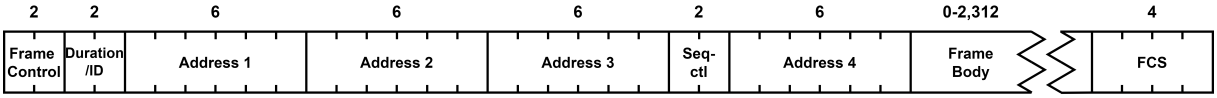

Figure 2.1: 802.11 frame as defined in [1]

Figure 2.1 shows the MAC Frame defined in 802.11. The fields of the frame are transmitted from left to right. The first two bytes of each frame are called the Frame Control subfield. These bytes are used to specify the crucial information of the data frame, such as type, subtype or power management as shown in Figure 2.2. There exist three major frame types in the 802.11 standard[1].

- ***Data Frames*** are used to transmit the actual data in the network. They carry the data used in the higher-level protocol. They can also perform some management functions such as acknowledging the reception of a packet.
- ***Control Frames*** are used to assist the delivery of data frames. For example a Request to Send (RTS) frame to gain control of the medium.

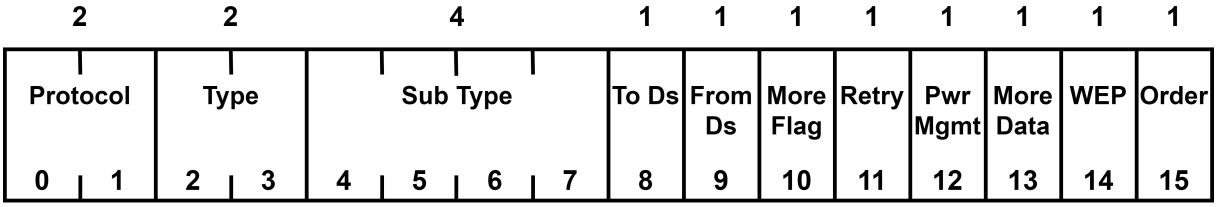

Figure 2.2: 802.11 frame control as defined in [1]

- ***Management Frames*** are mainly used to mange the network. They are used to authenticate a mobile station and join the network.

The Table C.1 gives a complete overview of all the different frame types and subtypes. Management and Control frames of a certain frame subtypes are only sent by either Access Point (AP) or mobile station. Data frames on the other hand are send by both AP and mobile station independent of the subtype of the frame. For Data frames, the **ToDS** and **FromDS** bits in the Frame Control subfields are relevant to identify, whether the frame was sent from an AP to a mobile Station or vice versa. Data frames with $ToDS = 1$ and $FromDS = 0$ identify a frame sent from a mobile station to an AP. Therefor, every frame can be identified as either a frame coming from an AP or from a mobile station. Table 2.2 list the frame subtypes sent by mobile stations. Data frames have different usage of the address fields depending on the combination of the **ToDS** and **FromDS** bit as described in the Table 2.3. Source Address(SA) is the MAC address of the station that generated the frame, Destination Address (DA) is the MAC address of the station that should process the frame and Basic Service Set ID(BSSID).

Subtype or Type	Name	Addition
0000	Association request	-
0010	Reassociation request	-
0100	Probe request	-
10	Data	ToDS = 1 and FromDS = 0

Table 2.2: Type and Subtype of Frames sent by mobile station

Function	ToDS	FromDS	Address 1	Address 2	Address 3	Address 4
To AP	1	0	BSSID	SA	DA	not used
From AP	0	1	DA	BSSID	SA	not used

Table 2.3: Use of address fields in data frame[1]

Wi-Fi-sniffing

Sniffing in general is the process of capturing packets sent over a network with the intent of analyzing the packages [4, p.647]. To capture packets sent in a wireless network, a special designed wireless sniffer can be used. This can be computer hardware specially designed to capture Wi-Fi packets [5, p.58]. However, capturing packets is not limited to specially

designed hardware, but can be done with any wireless device which supports monitor mode. Monitor mode is an operation mode for a wireless interface, in which the interface passively captures any data sent wirelessly. Many devices either support this mode out of the box or can be used with a modified firmware to enable monitor mode. Monitor mode does not only enable packet capturing, but also enables capturing of additional physical information, such as capturing the received signal strength (RSS) of the packet[6].

2.1.4 Number of mobile telephony customers

In 2007, it was reported, that 80% of Swiss people over the age of 15 years were in possession of a mobile telephone. This data was collected by a survey. A more recent estimation reports even higher number of market penetration, namely 127.1%. However, this market penetration percentage is calculated by an arithmetical method. The number of mobile phone subscriptions is divided by the population of Switzerland[7].

2.1.5 Trilateration and Multilateration

Trilateration is the process of determining the position of an object based on the distance measurement between the object and two known points. The distance between a know point $P_i(x_i, y_i)$ and $P(x, y)$ is defined as following in two dimensional space:

$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}, i = 1, 2 \tag{2.1}$$

Solving the resulting two-dimensional quadratic equations will result in the coordinates of the point of interest $P(x, y)$. The Figure 2.3shows an example of trilateration in two-dimensional space. As shown, the solution to this trilateration problem is ambiguous for all points, which are not co-linear with P1 and P2.

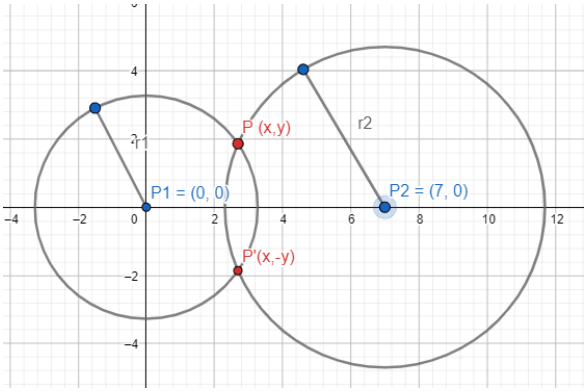

Figure 2.3: 2-D Trilateration Scenario.

A third non-co-linear measurement point is ideal for problems in two-dimensional space. For three-dimensional space, four points are needed to over-specify the system of equations and determine the exact position of $P(x, y, z)$ [8].

2.1.6 Signal Interference

The work of Woyach et al. [9] was the first instance in which the negative phenomena of multipath fading in wireless transmission was exploited to sense the movement of network nodes. Furthermore, they could also sense motions of bodies that were not part of the network but influenced the network by affecting signal propagation. They showed that Signal interference could be used to sense the presence of objects. Lee et al. [10] further extended this approach and were able to proof, that presence of humans increases the spread of Radio Signal Strength Indicator (RSSI) readings. This is due to the absorption of signals by the human body.

2.2 Related Work

2.2.1 Tracking Unmodified Smartphones Using Wi-Fi Monitors

Musa and Eriksson [11] have proposed a Wi-Fi-Based method to track cars and estimate their trajectory. They are utilizing the Wi-Fi detections of unmodified smartphones on-board the tracked vehicles. They are using Wi-Fi monitors sparsely placed along the paths which should be monitored. The monitors are spaced so that the distance between the monitors is approximately 600 meters since the Wi-Fi detection range tends to be between 250 and 300 meters. This reduces overlap and multiple detections of a single device. They then use a Markov model to estimate the car's trajectory since the vehicle is following a restricted path when driving on the road. The proposed solution only uses the Wi-Fi detections of the present smartphones, without requiring the phone to be modified or actively participate in this tracking. However, this approach uses a lot of constraints to track the path of vehicles accurately. There is no free movement possible in this approach, but rather the travel along a predefined underlying model.

2.2.2 Wi-Counter: Smartphone-Based People Counter Using Crowd-sourced Wi-Fi Signal Data

Li et al. [12] have proposed a crowdsourced approach to estimate the number of people present in a room. Their so-called Wi-Counter relies on the interference a human body has on the RSS. This was done by having the lecturer manually record the number of students present in the room. Simultaneously, the smartphone collected the RSS every 20 seconds. The crowd-sourced data was then cleaned to reduce noise and fed into a neural network trained to estimate the number of people present given the RSS. This resulted in an average person count error of less than 15% based on experiments with 50 people. Results presented the robustness of the approach independent of the movement of the people present. However, this method is limited to specific spaces similar to those used to crowdsource the data used for training the neural network. Furthermore, this method requires manual counting of people during the crowdsourcing phase.

2.2.3 Counting Public Transport Passenger Using WiFi Signatures of Mobile Devices

Myrvoll et al. [13] have implemented a system to track the number of passengers on board a public transport vehicle. They used the detected frames in the 2.4 GHz band to collect Wi-Fi data, which is then used to estimate the count of persons on board the vehicle. They are presenting promising results as the estimated count correspond the ground truth observed. The downside of the approach chosen is that the model relies on static MAC addresses. Since privacy is a growing issue, the primary mobile phone operating systems providers are developing mechanisms to randomize the MAC address of mobile phones. Therefore, mobile phones can not be uniquely identified anymore.

2.2.4 Electronic Frog Eye: Counting Crowd Using WiFi

Xi et al. [14] use an interference-based approach to estimate the number of people present. They use the Channel State Information from every OFDM channel instead of the RSS. They claim this information to be an acceptable grained value compared to RSS. However, their solution is only tested on a small number of persons.

2.2.5 FreeCount: Device-Free Crowd Counting with Commodity WiFi

Zou et al. [15] are also using an interference-based approach. With their information theory based approach for feature selection, they claim to be able accurately estimate the number of people with 96% of the time. This was achieved by using transfer kernel learning, which is able to transfer the knowledge from the source domain to be able to use this to solve the problem in the target domain.

2.2.6 ASIMOV: a Fully Passive WiFi Device Tracking

Ribeiro et al. [16]'s work is centered around eliminating MAC randomization used by most phones to anonymize the user. Ribeiro et al. [16] uses an RSSI based approach in combination with trilateration to locate smartphones. Based on the location estimate, he manages to determine, whether a new MAC address seen corresponds to a new device or is just a new MAC address of a device doing MAC randomization for privacy purpose. Ribeiro et al. [16] claims, that this RSSI based localization approach is precise enough to decide, whether a new encountered address in the same area corresponds to a new device or the already seen one.

2.2.7 BluePIL: a Bluetooth-based PassIve Localization Method

"Bluepil: a bluetooth-based passive localization method" [17] describes a similar approach to the one described in "ASIMOV: a Fully Passive WiFi Device Tracking"[16], with the

major difference, that the BluePIL project is not using WLAN packets and their RSSI, but relies on capturing Bluetooth signals and the RSSI of this signal. This is then used in the same way, to localize the device using RSSI and trilateration. The main disadvantage of the proposed approach is, that the usage of Bluetooth requires a constant Bluetooth connection, which might not be present all the time. However, the usage of Bluetooth might increase with smartphone manufactures deciding to build more smartphone without an audio jack, requiring a Bluetooth connection to connect peripherals.

2.2.8 CCount: Correlating RFID and Camera Data for High Precision Indoor Tracking

Rodrigues et al. [18] use another approach, where the data of captured Radio-frequency identification (RFID) tags were used, to estimate the amount of person present in a given area. In addition of capturing the RFID tags, a 3D camera was used to monitor the area. The goal of the work, was to correlate the data captured by the camera with the RFID tag sensed. They claim a correlation rate of 70-80% between the 3D cameras and the RFID tags. The RFID tags are used to have unique identifiers for the person counted and avoid duplicate counting. However, this approach requires the persons monitored to wear an RFID tag. This seems reasonable for events such as a congress, where every guest is handed a batch anyway. For other events, this approach might not be suitable.

Chapter 3

Design

3.1 Requirements

From the goals of this thesis is described in Section 1.2, there can be following requirements derived for the system:

- **R1** Require no active interaction of the counted persons.
- **R2** Be able to localize individuals based on their Wi-Fi signals.
- **R3** Give an approximate count based on localized individuals.
- **R4** Warn whenever an area is overcrowded, based on a pre-defined ocuppancy threshold.

3.2 Proposed Person Counting approach

As described in Section 2.1.4, almost every person in Switzerland has a mobile phone. Due to this fact, counting the amount of smartphones in a given area can be used to approximate the number of persons present. There are certain types of wireless frames, which are only sent by mobile stations as described in Section 2.1.3. Therefore, we can filter traffic of other APs.

Since a smartphone connected to an AP sends data constantly, these packets can be used for localization. Whenever a smartphone is not connected to an AP, it normally sends probe requests periodically to scan for available APs. Therefore, the idea is, to place Wi-Fi sniffers in an overlapping pattern to have at least four sensors covering the whole surveillance area. This is due to the requirements for an unambiguous location of the smartphone.

Every Wi-Fi probe packet, that is sent by a client, as described in Table 2.2, is logged with the corresponding sender address, its time, and RSS. The data collected by every

sensor would be saved on the sniffer itself. The data captured is saved in a Comma-Separated Values (CSV) file. The data is manually copied to a computer and fed to the positioning and counting algorithm for a more straightforward approach. The whole process is depicted in the Figure 3.1. The intended deployment scenario is illustrated in the Figure 3.2.

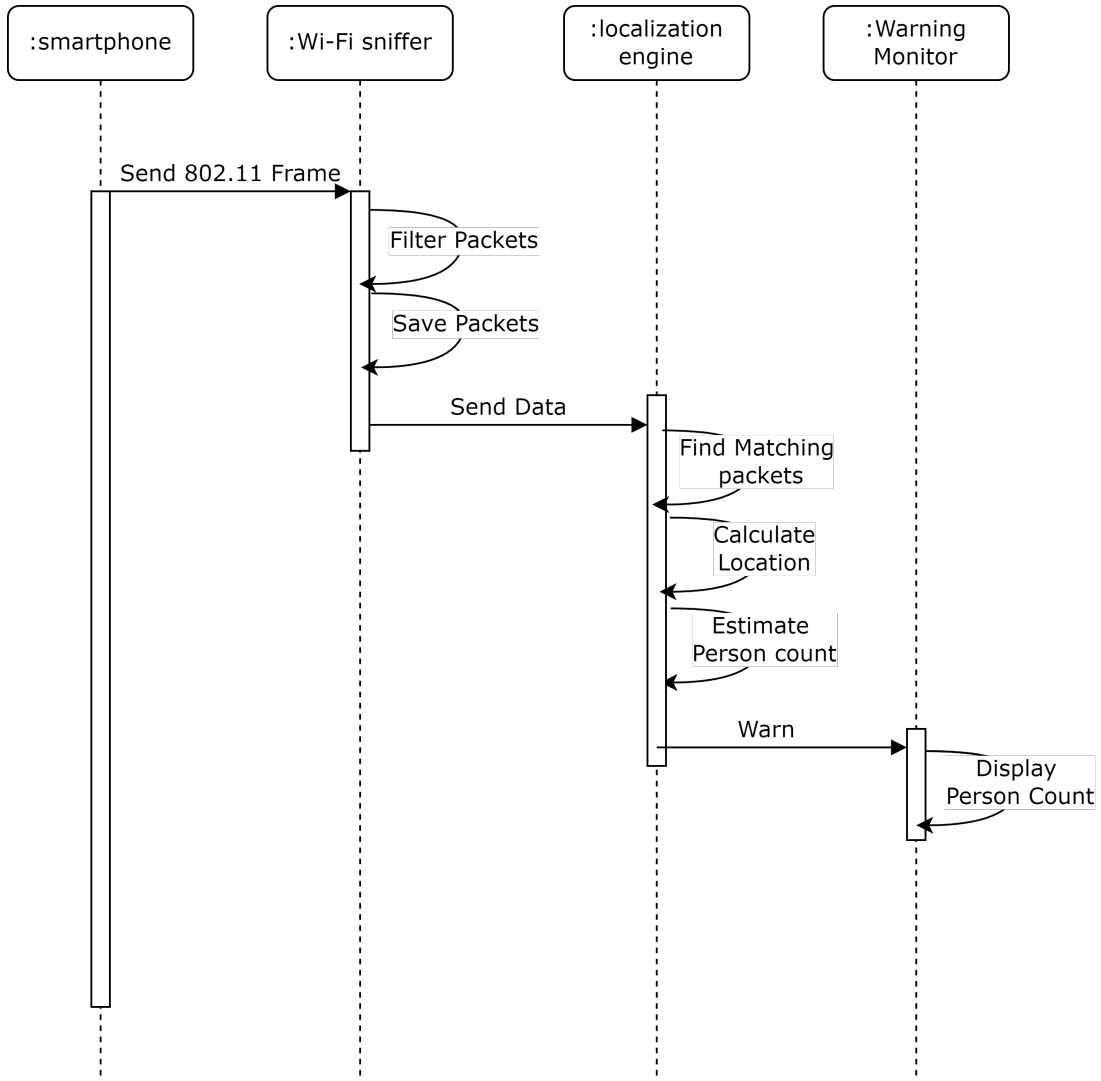

Figure 3.1: Sequence diagram

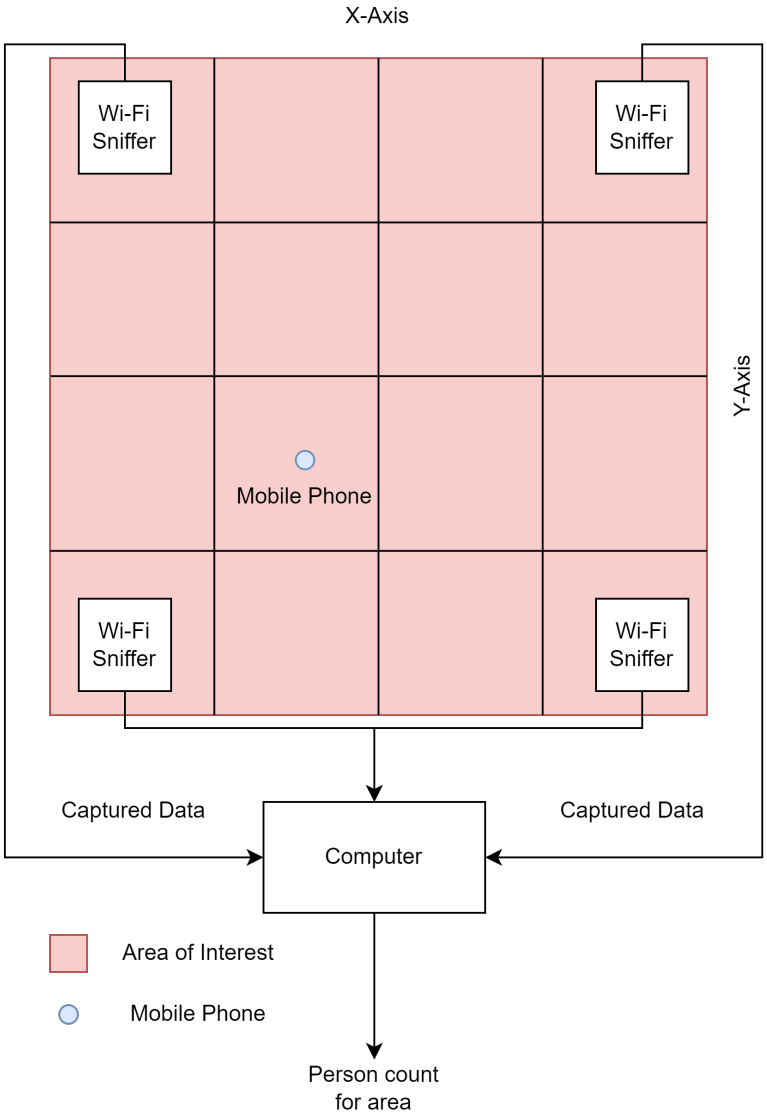

Figure 3.2: Sniffer deployment

3.3 Wi-Fi Sniffer

As described in Section 2.1.3, a Wi-Fi sniffer can be any device that supports monitor mode on the wireless interface. The core idea of the chosen person counting approach was to use trilateration to locate smartphones. Therefore, there was the need for multiple Wi-Fi sniffers to cover the area, which should be sensed. Therefore, a minimum of three sensors was needed for exact localization in two-dimensional space, as described in Section 2.1.5. It was decided to only consider two dimensional space, since the movement of the crowd observed tend to be two dimensional. Since the Raspberry Pi model Zero W is very cost-effective and also offers onboard Wi-Fi capability, this hardware was chosen to act as a Wi-Fi sniffer. The Pi Zero boards were fixed to a power bank responsible for power delivery to have a standalone Wi-Fi sniffer package. A picture of a sniffer package

is shown in the Figure 3.3.

Figure 3.3: Raspberry Pi Zero W and Powerbank used for sniffing

.

3.3.1 Software

Operating System

The first idea was to use a kali linux[19] image on the Pi and use the nexmon project to patch the firmware of the Pi[20]. The firmware patching needed to add support for a monitor mode on the wireless interface. Kali Linux was chosen since it is specially designed for penetration testing and other security research.

However, the Kali Linux operating system turned out to be very delicate when it comes to how the system is shut down. This issue is further described in Section 4.1.1. Therefore the dedicated Raspberry Pi operating system light [21](no graphical user interface) was used, since this operating system turned out to be more resistant to power outages. The nexmon patching did also not work as planned. Therefor a pre-compiled kernel which includes the nexmon patch was used[22].

3.4 Counting Algorithm

The idea was to count the person present in the given are for a point in time. It was decided to aggregate the data for short time intervals. This was done to have more robust counts to filter out the noise in the aggregated data. Therefore, the counting algorithm first creates time windows of the captured data. The data points would split up based on the MAC address in this time window. This subset of data will then be used to multilaterate the device. The multilateration function would calculate the relative position of the device. This position would need to be further processed by the localization engine to determine

how the device influences the person count in the given area. This process is depicted in the Figure 3.4.

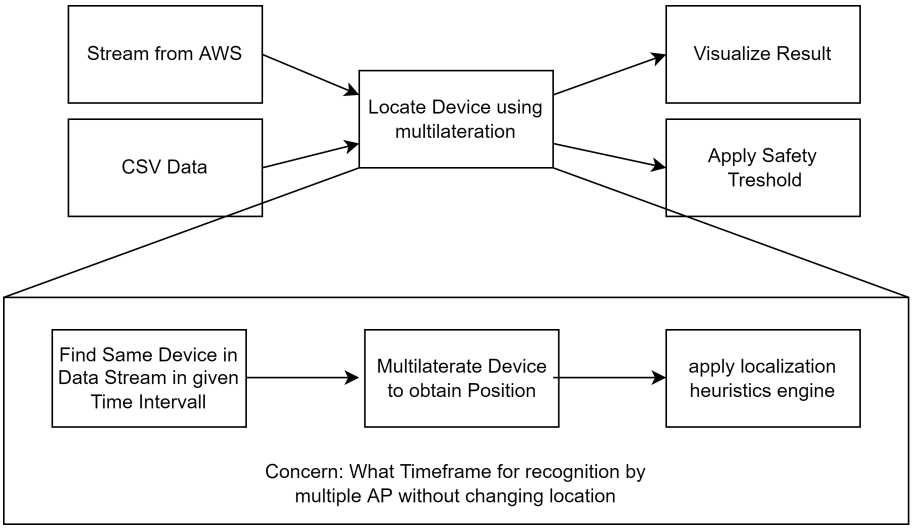

Figure 3.4: Counting Algorithm

3.4.1 Distance estimation

The Wi-Fi sniffer can sense the strength of the packets captured. This RSSI value can be used to estimate the distance between the device and the Wi-Fi sniffer. The distance is described by equation[23]:

$$P(d) = P(d_0) + 10\, n\, log_{10}(\frac{d}{d_0})$$

Where,

- $P(X)$ is the path loss at the Wi-Fi sniffer
- $P(X_i)$ is the path loss at reference distance d_0
- n is the path-loss exponent, which is used to incorporate different environments
- d is the estimated distance
- d_0 is the reference distance

Viswanathan [23] estimates n for inside a building with some obstructions between sniffer and smartphone to be between **4 and 6**. The value for reference path loss has to be experimentally determined.

3.4.2 Approach Discussion

- ***Usage of dedicated WI-Fi Sniffers*** This design approach uses dedicated Wi-Fi sniffers for packet capturing. This is contradicting the motivation of cost-effectiveness described in Section 1. However, this design approach was chosen to simplify the implementation and evaluation of the tracking and counting system. The proposed design and implementation can be adapted to use already deployed APs with the condition that these APs support monitoring and serve as AP. Once such a tracking system is established, the demand for APs capable of monitoring might increase.

- ***RSS based localization*** The proposed design uses the RSS as an indicator for the distance between the sniffer and the smartphone. This decision was made over using other metrics, such as Time of Arrival (TOA) based distance estimation, since it is a more straightforward approach. Using TOA-based distance estimation would require a very high precision clock synchronization of all the Wi-Fi sniffers. A deviation of a few nanoseconds would lead to increased errors in distance estimations.

Chapter 4

Implementation

4.1 Wi-Fi Sniffer

4.1.1 Operating System

As described in Section 3.3, Raspberry Pi Zero W's were used for Wi-Fi sniffing. As described in Section 3.3.1, the first idea was to use Kali linux. To simplify the setup of the Wi-Fi sniffer, there was the plan of creating a master image of the sniffer operating system with the required kernel patches already installed to allow the on board wireless chip to function as a monitoring interface. Therefore, the corresponding Kali Linux image was downloaded to create a boot medium for the Raspberry Pi. Then, the Raspberry Pi was initialized, and the steps described on the GitHub project of Schulz [20] were carried out to patch the kernel of the Raspberry Pi.

Unfortunately, during the execution of the download and installation commands of the Nexmon patching framework, there were many instances where the Raspberry Pi froze. Turning the Raspberry Pi of by disconnecting the power source led to bad sectors of the SD card. Kali Linux was very delicate to the power of. Therefore, the image was damaged beyond repair.

Therefore, the decision was made to switch to the dedicated Debian-based Raspberry Pi operating system with the hope of this operating system being more resistant. Patching the firmware using the approach described by Schulz [20] was not successful. Therefore, a kernel, which already included the firmware patch, was used. This resulted in an image that allows capturing on the newly added wireless interface ***mon0***.

4.1.2 Time Synchronization

The time of the Wi-Fi sniffer was intended to be used to timestamp the data. This timestamp is then used to identify package information captured by all Wi-Fi sniffers. Therefore, it is crucial to have a more or less valid time value on the sniffer. This time value

does not have to be exact to the nanoseconds since RSS is used for distance estimation instead of TOA but needs to be within an error range of less than one second. Therefore, clock synchronization over the internet is sufficient. As the Raspberry Pi computer does not include a real-time clock module[1], the time value of the Raspberry Pi is reset every time it is powered of. The Wi-Fi sniffers were configured to automatically connect to the WLAN network in the testing environment and synchronize the clock to fix this issue. The network router was configured so that the Wi-Fi sniffers would have a static IP address. Furthermore, SSH was enabled on the Wi-Fi sniffers. This configuration allows controlling the sniffers from a central computer for starting the capturing and coping of the data to the processing instance.

4.1.3 Package capturing

The first approach was to use airodump-ng to capture Wi-Fi packets[24]. Airodump-ng can capture raw 802.11 frames and write to a capture file. This file can then be analyzed using Wireshark[25]. The main problem was that airodump-ng does not include RSS measurements in the resulting capture file. Therefore, this data could not be used to estimate distances and is not suitable for the proposed approach.

A dedicated capturing library was used because airodump-ng did not satisfy the requirements. This library was Scapy. Scapy is a Python library, which can forge, dissect, emit or sniff network packages. Sniffed packets can not only be captured but also analyzed[26]. Therefore, filtering the packages based on the criteria defined in the Table 2.2 can directly be done on the sniffer. Thus, the data overhead is reduced, as only data relevant for the localization of the devices is stored.

```
1  from scapy.all import *
2  import time
3  import os
4
5  filename = "/home/pi/Documents/Capturing/Data/"
6  filename = filename + str(int(time.time())) + ".csv"
7  os.makedirs(os.path.dirname(filename), exist_ok=True)
8  f = open(filename, "w")
9
10 if len(sys.argv) >=  2:
11     iface = str(sys.argv[1])
12 else:
13     iface = "mon0"
14
15 if len(sys.argv) >=  3:
16     device = str(sys.argv[2])
17 else:
18     device = "99"
19
20 if len(sys.argv) >=  4:
21     duration = int(sys.argv[3])
22 else:
23     duration = 5
```

[1] An electronic device with a separate power supply used to measure the passing of time

```
24
25 until = time.time() + 60*duration
26
27 def callback(packet):
28     if packet.haslayer(Dot11):
29         if packet.type == 0 and packet.subtype == 0:
30             writePacket(packet,packet[Dot11].addr2)
31         if packet.type == 0 and packet.subtype == 2:
32             writePacket(packet,packet[Dot11].addr2)
33         if packet.type == 0 and packet.subtype == 4:
34             writePacket(packet,packet[Dot11].addr2)
35         if packet.type == 2 :
36             if packet[Dot11].FCfield.__getattr__("to-DS") == 1 and
                   packet[Dot11].FCfield.__getattr__("from-DS") == 0:
37                 writePacket(packet,packet[Dot11].addr2)
38
39
40 def writePacket(packet, senderAddress):
41     try:
42         dbm_signal = packet.dBm_AntSignal
43     except:
44         dbm_signal = "N/A"
45     f.write('{0};{1};{2};{3};\n'.format(senderAddress,dbm_signal,str(int
          (time.time())),device))
46     print(packet[Dot11].mysummary())
47
48 if __name__ == "__main__":
49     try:
50         while until > time.time():
51             sniff(iface=iface, prn=callback, timeout=60 )
52             for channel in range(1, 14):
53                 os.system("iwconfig " + iface + " channel " + str(
                       channel))
54     except KeyboardInterrupt:
55         print("Press Ctrl-C to terminate while statement")
56         pass
57     f.close()
```

Listing 4.1: Package Capturing

Listing 4.1shows the Python capturing script used on the Wi-Fi sniffer to capture, filter, and save the relevant 802.11 frames. The Python script first creates a file in a subfolder with the current timestamp as the filename. If no arguments are specified, the default capturing interface is ***mon0***. Scapy allows filtering the seized packages. Line 28 shows that only captured packets of the 802.11 type are considered. Line 29 to 37 are implementing the packet filter as described in Section 2.1.3. The *writePacket*() function calculates the RSS of the captured packet and writes the relevant data to the file.

Sniffing is done for five minutes, when no other duration is specified. To cover all the channels used by 2.4 GHz Wi-Fi, the sniffer sniffs only for a limited time on the track and then hops to the next channel using an operating system command to change the medium of the monitoring interface as implemented in line 51 to 53.

4.2 Localization engine

The localization engine is also implemented in Python. This decision was made, since Python allows easy, fast and efficient handling of large data sets by using the library pandas[27]. The state of the proposed system during development is not using a stream processing approach, but rather a data analysis approach, where the collected data is processed in retrospective.

Therefore, the captured data of the Wi-Fi sniffers is collected by using the Secure File Transfer Protocol (SFTP) and loading the created CSV file to the computer intended for evaluation. The data sets are then tagged with the sniffer id, which collected the data. Then, the data sets need to be combined in one big CSV file. The merging of the file can be achieved with the *mergeCSV.py* script. The resulting file is then processed by the localization engine.

As the Localization engine should be configurable based on the sensors and their placement, there is a configuration file used to define the sensors and their position. Furthermore, it is possible to define areas, for which a separate person count is done. The areas can be any rectangular shape. The area is also defined in the configuration file. This is done, by specifying the coordinates of the corners opposite of each other. For every area, a maximum count of persons can be defined. Furthermore, a fraction defined the threshold, whenever a warning should be printed to the console. Parts of a sample configuration file are depicted in Listing 4.2. All the other required configuration details are described in Section B.1.2.

```
1  Sensor:
2    0:
3      x: 425
4      y: 368
5    1:
6      x: 5
7      y: 421
8  Area:
9    0:
10     P0:
11       x: 0
12       y: 0
13     P1:
14       x: 511
15       y: 421
16     Max: 1
17     Threshold: 0.80
```

Listing 4.2: Part of an example Configuration

The localization engine code is depicted in Listing 4.3. The data is loaded by the program using pandas. The unix timestaps are converted to datetime, since this datatype simplifies time calculations. Furthermore, an output file is opened. This is done by the *loadData*() funtion Since the processing is done in retrospective, the minimum and maximum time and date of the captured packets is known. Pandas provides filtering functions. These are used to create subsets of the dataset, which contain all the packets captured in a 30 second slice. This dataframe is then passed to the trilaterate function.

```
1  import datetime
2  import yaml
3  from Service.trilaterateation import trilaterate
4  from Service.dataHandling import loadData
5  from Service.personCount import getPersonCount
6
7  def main():
8      df, minDate, maxDate, f = loadData(config)
9      print('Start Trilateration')
10     while minDate < maxDate:
11         endDate = minDate + datetime.timedelta(seconds=config['
               TimeWindow'])
12         relevant = df[(df['datetime'] > minDate) & (df['datetime'] <
               endDate)]
13         result = trilaterate(relevant,config)
14         if result != None:
15             personCount = getPersonCount(result,config)
16             f.write(minDate.strftime('%Y-%m-%d %H:%M:%S') + ',' +
                   endDate.strftime('%Y-%m-%d %H:%M:%S') + ',' + str(
                   personCount)+'\n')
17         minDate = endDate
18     f.close()
19     print('Calculations Done')
20
21
22 if __name__ == '__main__':
23     with open('Data/configuration.yaml') as f:
24         config = yaml.load(f, Loader=yaml.FullLoader)
25     print(config)
26     main()
```

Listing 4.3: Localization engine

The trilaterate function is depicted in Listing 4.4. The trilaterate function loops over all unique device addresses in the time window with the duration defined in the application configuration. For every second of the time window, it is checked, whether there exist three or more measurements. This is due to the prerequisites of trilateration as described in Section 2.1.5. The device id of the sniffer and the RSSI are then used to trilaterate the position of the smartphone as shown on line 18. For visualization purpose, every 30 second time window's trilateration results are saved to a scatter plot. For this, pyplot of the library matplotlib was used[28]. The marker type shows the different devices. An example of the resulting plot is shown in Figure 4.1.

The actual trilateration computation is done by the Python library easy-trilateration implemented by Alexander [29]. Easy-trilateration uses the non linear least square method of scipy to solve the system of equations to locate a point based on distance measurements. The nice thing about this easy-trilateration library is, that it can also trilaterate with an over specified input, for example four distance readings.

```
1  def trilaterate(df, config):
2      plt.style.use('seaborn-whitegrid')
3      devices = df['address'].unique()
4      times = df['datetime'].unique()
5      plotTime = df['datetime'].min()
```

```
6       result = {}
7       marker = 1
8       for device in devices:
9           devicedf = df[df['address'] == device]
10          x = []
11          y = []
12          for time in times:
13              currdf =devicedf[(devicedf['datetime'] == time)]
14              if len(currdf['sensorid'].unique()) > 2:
15                  readings = getReadingsForSensors(currdf)
16                  if device not in result:
17                      result[device] = {}
18                  xcord,ycord = trilaterateUsingRSSI(readings, config)
19                  result[device][time] = (xcord,ycord)
20                  x.append(xcord)
21                  y.append(ycord)
22          if len(x) > 0 and len(y) > 0:
23              plt.xlim(0, 540)
24              plt.ylim(0, 530)
25              scat = plt.scatter(x, y, marker=marker)
26              marker = marker +1
27              if marker == 12:
28                  marker = 1
29      if result == {}:
30          return
31      else:
32          filename = 'Data/Figures/'+re.sub(r'[^\w\-_\. ]', '_', str(
                plotTime))+'.png'
33          scat.figure.savefig(filename)
34          scat.figure.clf()
35          return  result
```

Listing 4.4: Trilateration

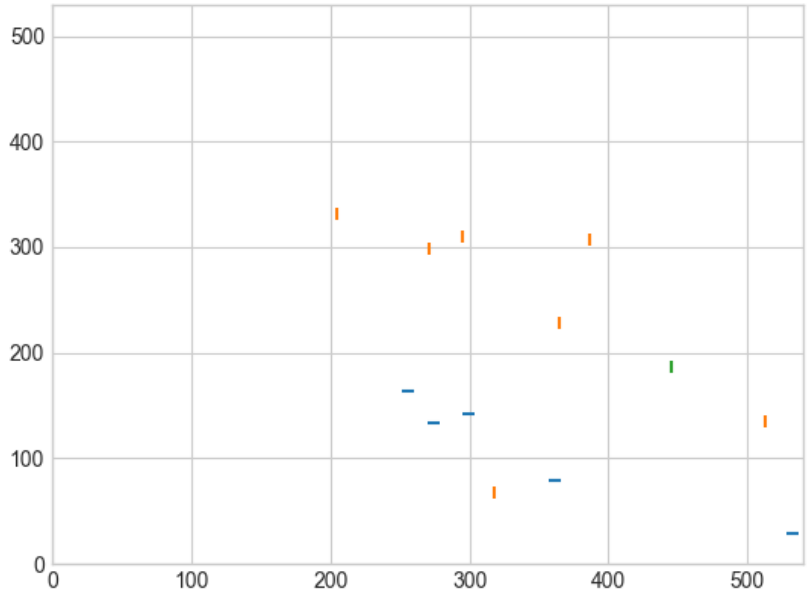

Figure 4.1: Scatter plot of trilateration result

.

After the devices have been localized using trilateration, their location is passed to the *getPersonCount*() function. This function does the actual person count. The basic implementation calculates the mean of the observed position values and utilizes this for person count. The current implementation counts persons multiple times, whenever defined areas are overlapping. The implementation requires the points defining the area to be ordered ascending in regards of their x and y values.

```
1  from collections import defaultdict
2  import numpy
3
4  def getPersonCount(result, config):
5      count = defaultdict(int)
6      for device in result.keys():
7          x = []
8          y = []
9          for time in result[device].keys():
10             xcord,ycord = result[device][time]
11             x.append(xcord)
12             y.append(ycord)
13         x_mean = numpy.mean(x)
14         y_mean = numpy.mean(y)
15         countDevice(x_mean,y_mean,count,config)
16     for area in count.keys():
17         if (config['Area'][area]['Max'] * config['Area'][area]['
               Threshold']) <= count[area]:
18             print("Area "+str(area)+" is overcrowded at "+str(count[area
                   ])+ " devices present")
```

```
19      return count
20
21
22  def countDevice(x,y,count,config):
23      for area in config['Area'].keys():
24          if (config['Area'][area]['P0']['x'] <= x and config['Area'][area
                ]['P1']['x'] >= x and config['Area'][area]['P0']['y'] <= y
                and config['Area'][area]['P1']['y'] >= y):
25              count[area] += 1
```

Listing 4.5: Person Count

Chapter 5

Evaluation

This chapter describes the experiments conducted to verify the approach taken. The first experiment conducted was to verify, that the Wi-Fi sniffers built were actually capable of capturing WLAN packets. It is was also used to determine the path loss needed for distance estimation.

The second experiment was conducted to validate the RSSI based localization approach. Furthermore, the idea was to evaluate, whether the movement of the devices can be accurately tracked. This tracking should then be used to identify devices entering or leaving the area of interest.

5.1 Experiment 1: Wi-Fi Capturing and range detection

5.1.1 Experiment Idea

The first experiment was conducted, to verify the capturing approach implemented on the raspberry pi. Its main goal was to verify, whether the packets sent by the smartphone are visible for the sniffer and can actually be captured. Furthermore, it was tested, whether the Wi-Fi sniffer was able to capture both, a non connected phone sending probe requests and a phone connected to WLAN sending data packets.

The second goal of the experiment was to determine the path loss, as this value is needed for distance estimation.

5.1.2 Methodology and Setup

Since there was no interest in trilateration and localisation of the smartphone, there was only one Wi-Fi sniffer used for this experiment.

The sniffer used is implemented as described in Section 4.1. The sniffer is built on a Raspberry pi zero W. Therefor the monitoring chip is a Broadcom BCM43438 radio chip, which supports bluetooth and 2.4 GHz WLAN.

The Wi-Fi sniffer was placed into the corner of and indoor room. The room has the size of 425 centimeters by 521 centimeters. The placement of the sensor was similar to the intended placement of the sensor when doing person counting. The sniffer was placed at sensor 2 position depicted in Figure 5.1.

The smartphones used for this experiment were an Oneplus 8 pro android phone[1], and a huawei mate 20 pro android phone [2]. The Oneplus phone is daily used and set up with multiple accounts and applications installed (e.g., instant messenger, mail, music streaming etc.). The phone is also equipped with a sim card. The second phone has no sim card inserted. There are also no accounts linked on the phone (no mail, no instant messaging). The phone has only YouTube installed.

The Wi-Fi sniffer was then started and put into sniffing mode. Following three scenarios where captured:

1. ***Scenario 1:*** The smartphone, which was connected to the WLAN, was placed at the distance of one meter from the sensor. The smartphone was left alone for the duration of five minutes, while the data was captured. The smartphone was configured, to not use mac randomization, to then be able to find the MAC address in the captured data.

2. ***Scenario 2:*** The smartphone, which was not connected to WLAN, but had WLAN enabled, was placed at the distance of one meter from the sensor. This smartphone was also left there for the duration of five minutes.

3. ***Scenario 3:*** The same smartphone with the same configuration as described in the first scenario was placed at the distance of 2 meters from the sensor and left untouched for five minutes.

4. ***Scenario 4:*** The same smartphone with the same configuration as described in the second scenario was placed at the distance of 2 meters from the sensor, and left untouched for five minutes.

The four scenarios were conducted with both phones to verify that both phones can be sensed. The *capturing.py* script as described in Listing 4.1 was modified to also include packet summary into the CSV file. This packet summary includes the packet type and subtype. This data is crucial to validate, that both, management and data packets, can be sensed.

The results of the five minutes of captured data should then be analyzed for the presence of the required packets for each scenario. The MAC address of the sensed smartphone should then be used to filter the data to only contain relevant data. The mean path loss at 100 centimeters should then be taken as $P(d_0)$ with a d_0 of 100.

[1] `https://www.oneplus.com/8-pro`

[2] `https://consumer.huawei.com/en/support/phones/mate20-pro/`

5.1.3 Results

Scenario 1

In the resulting data set of scenario 1 with the first phone, we could observe, that a majority of the captured frames are sent out by the sensed device (73% of 133 data points). When only looking at the packets sent by the phone, we could calculate a mean path loss of -45.9 dBm, with a standard deviation of 2.7.

For the second phone, we could observe, that the amount of packets sent by the phone is much smaller than with the first phone (19% of 143 data points). This observation might be due to the configuration of the second smartphone, with no accounts installed. When only considering the packets of the second phone, we calculated an average path loss of -38.5 dBm at 100 centimeters with a standard deviation of 1.2.

Scenario 2

In the resulting data set of scenario 2, recorded with the first phone, we could observe only a very small amount of management frames. All of the captured management packets were of the packet subtype of probe requests. There were no association or disassociation packets registered. 7 out of 132 packets captured were the probe requests send from the first phone. The mean path loss of the first phone was -50.4 dBm with a standard deviation of 1.7.

For the second phone, the number of relevant packets increased to 24 out of 88 captured frames. There were also no association or disassociation requests captured. The mean of the path loss for the second phone is -43.3 dBm, with a standard deviation of 6.6.

Scenario 3

In the data captured in scenario 3, we could see a decline in number of total packets captured. Also the number of packets we were looking for (data packets sent by the smartphone) was smaller . The frames from the first phone only made up 49 of the 83 frames. The path loss increased to a mean of -52.4 dBm with a standard deviation of 5.6.

For the second phone, the number of relevant frames increased to 32 of 58 captured in total. The mean path loss is -55.0 dBm with a standard deviation of 1.1.

Scenario 4

Similar to the second scenario, the number of management packets observed in scenario 4 is very small at only 6 of 168 packets relevant for the first phone. The mean path loss is measured to be at -59.3 dBm with a standard deviation of 3.3. There were only probe request reported, no association or disassociation frames.

The second phone showed up in the data in 16 of 103 packets. Similar as to the previous observations, there were no association or disassociation frames captured. The mean of the path loss is calculated to be -51.8 dBm at a standard deviation of 4.1.

Scenario	Phone	Path loss (dBm)	Standard deviation	Relevant packets	Total packets
1	1	-45.9	2.7	98	133
1	2	-38.5	1.2	28	143
2	1	-50.4	1.7	7	132
2	2	-43.3	6.6	24	88
3	1	-52.4	5.6	49	83
3	2	-55.0	1.1	32	58
4	1	-59.3	3.3	6	168
4	2	-51.8	4.1	16	103

Table 5.1: Summary of the data of experiment 1

5.1.4 Conclusion

From this first experiment, we can conclude, that the capturing of the packets works in general. However, when looking at the data in Table 5.1, there are some concerns raised regarding our capturing and localization approach as follows:

- When looking at the results of the second and fourth scenario, we can see, that the amount of data sent by the phone, is to small to have close to real time localization of smartphones using only probe requests. Most of the probe request captured by the sensors were even sent in burst. Therefore, the data is even more sparse. For the best case scenario using only probe request, we would only have five out of ten 30 second windows, which would include data generated by the probe requests.

- When looking at the number of data packets captured in scenario 1 and 3, we could see, that the phone with more accounts linked and more applications installed was sending up to 3.5 five times the amount of data packet compared to the second phone, which only had few applications installed. This issue might not be as crucial, as the devices carried by people tend to be configured in such a way that they utilize a lot of applications, which generate data traffic periodically.

- Regarding the path loss observed during the experiment, we can see, that the strength of the Wi-Fi signal varies depending on the hardware of the smartphone, as the first phone was weaker than the second phone, with the only exception being the third scenario. This creates some problem in distance estimation, as the calculated distance based on RSSI can not be accurate for all the devices encountered.

However, the impact of the distance estimation issue might have less impact as it suggests. Trilateration might still be done based on the ratio between the different distance estimations, rather than on the actual value.

5.2 Experiment 2 Wi-Fi based localization

5.2.1 Experiment Idea

The goal of the second experiment was to evaluate whether the smartphones, which should be used to count the persons present in a given are, can actually be locate using trilateration based on the RSSI measurements. The goal is to determine the accuracy of the calculated position based on a ground truth. Furthermore, the approach should be evaluated for the tracking of stationary phone and also for moving phones.

5.2.2 Methodology and Setup

Since the main goal of the experiment was to test trilateration, all of the four available sensors were used and placed in an almost square pattern in the same room, in which the first experiment was conducted. The placement of the sensors is depicted in Figure 5.1. The room is furnished with a table and chairs around the table. The sensors were all placed 90 centimeters above ground.

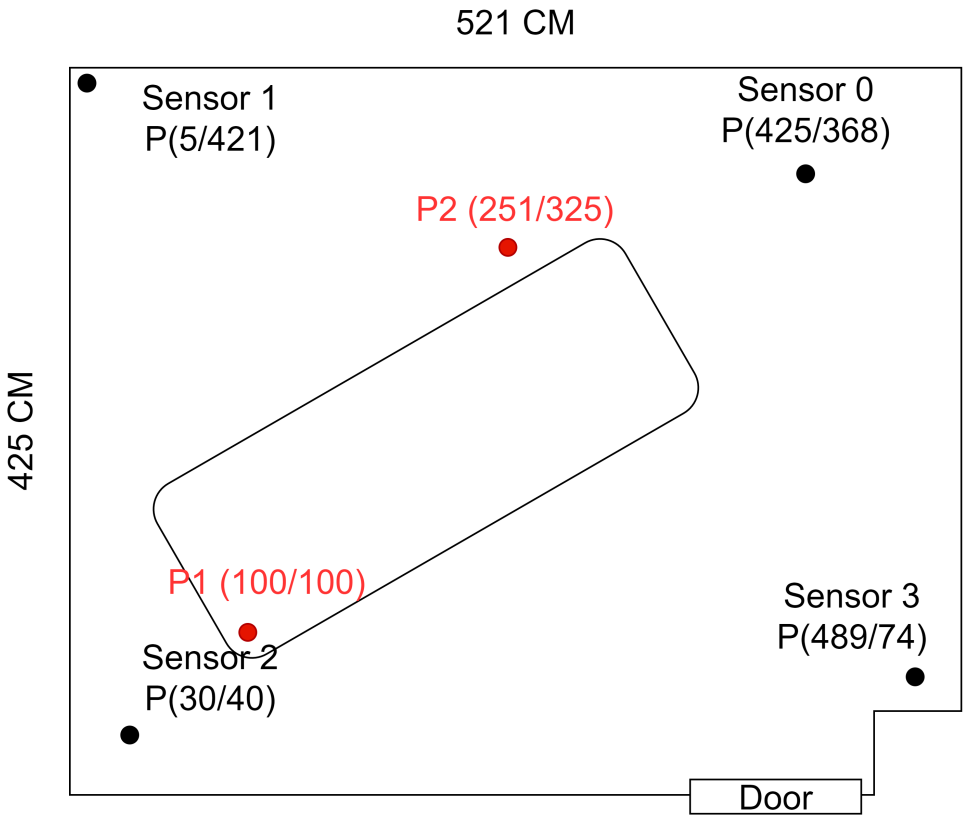

Figure 5.1: Sensor placement for experiment 2

.

In order to evaluate whether the approach taken can be used to locate stationary and moving phones, there were four scenarios played trough during the experiment. The scenarios were following:

1. ***Scenario 1:*** The smartphone is placed stationary at the point $P1(100/100)$ on the table and left there untouched for five minutes.

2. ***Scenario 2:*** The smartphone is placed in the front pocket of the pants of the test subject. The test subject enters the room through the door, circles around the table for one full circle and then sits down at the table at point $P2(251/325)$. The experiment duration is limited to five minutes.

3. ***Scenario 3:*** The smartphone is placed in the front pocket of the test subjects pants. The test subject, sitting at $P2(251/325)$, stays there for approximately one minute. The subject then gets up and leaves the room.

4. ***Scenario 4:*** The test subject is outside of the room with the smartphone still in the front pocket of the pants. The test subject then enters the room and circles the table for three minutes in comfortable walking speed, which is estimated to be about 139 centimeters per second [30]. The test subject then leaves the room. The whole duration of the experiment is five minutes.

Based on the results of the first experiment, the second experiment was only conducted with the first smartphone described in experiment one, since the scenario corresponds with the normal use case of a smartphone, where it is used with linked accounts and a sim card installed. The smartphone was also always connected to the WLAN, since the management frames would just have been to sparse to accurately track the device.

5.2.3 Results

Scenario 1

After the first scenario was conducted, we could observe, that the packets sent by the smartphone were not always captured by every sniffer. We filtered the capturing file by the sender address to only include packets sent by the smartphone. Sensor 0 captured 98 packets, sensor 1 captured 111 packets, sensor 2 captured 78 packets and sensor 3 captured 87 packets sent by the phone. When combining the data sets of all sensors into one data set, we get a total of 374 observations. When looking at the different timestamps of the observations, we could observe a total of 27 instances, where there were packets registered at three or more sniffers at the same time.

This data set was then processed by the localization engine implemented as described in Section 4.2. The localization engine calculated the position of the smartphone at the positions listed in Table C.2. Figure 5.2 shows a visualization of the resulting location estimation.

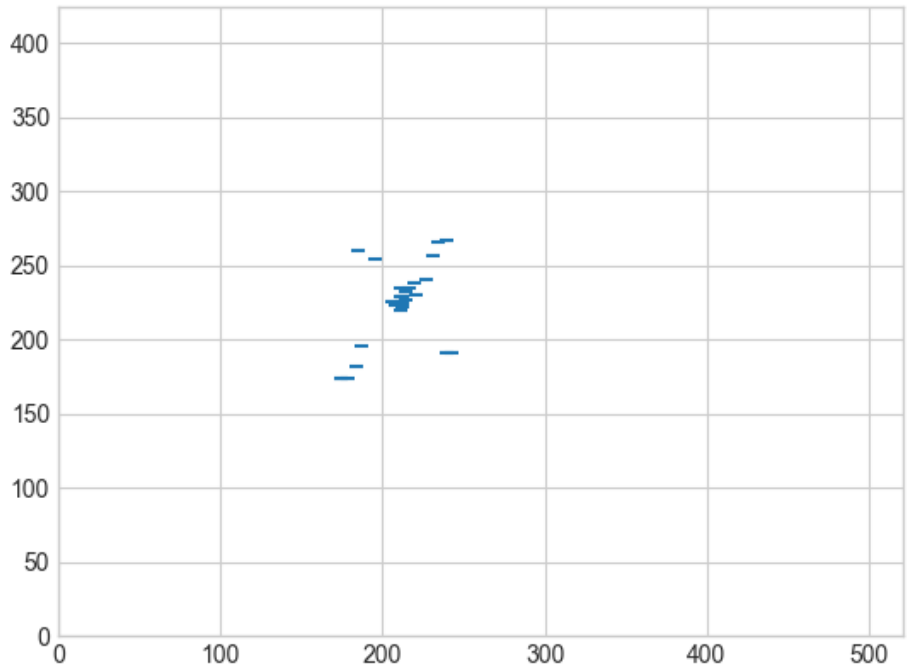

Figure 5.2: location estimation scenario 1

.

Figure 5.2 clearly shows, that the calculated location of the smartphone does not correspond to the actually location of the smartphone. The mean of all calculated positions is the point (205/222). Therefore, the calculated position is 160 centimeters of the actual position. The raw data shows, why the localization of the device is not working based on the trilateration algorithm. As the device is located at point $P1(100/100)$, the path loss to the nearest sensor, in this case sensor 2, should be the smallest. Sensor 0 should register the highest path loss, as it has the longest distance. During the first scenario, the mean RSS of the packets sent by the smartphone registered by sensor 2 is -48.0 dBm. The sensor with the smallest path loss, which should indicate the smallest distance, is sensor 1 with a mean path loss of -44.8 dBm. The highest mean path loss is registered at sensor 3, with a value of -51.1 dBm. It is clear, that this data can not be used to calculate position of devices accurately, as the path loss measurements are not accurate between the different sensors.

This issue regarding the path loss sensing might be due to different reasons. On one hand, the Wi-Fi sniffers might have sensors with different calibrations, resulting in different measurements of path loss. The other reasons for such inconsistent readings might be due to the influences on the radio waves, as the signal strength might heavily be influenced by multi path propagation[31].

Scenario 2

The data obtained in scenario 2 was very sparse. There were only a total of 19 packets sent by the smartphone registered. Out of these 19 data points, there was no case, where there were three or more matching timestamps. Therefore, the device could not be trilaterated. There were a lot of observations, where the signal strength of the captured packet could not be calculated by the Wi-Fi sniffer.

Scenario 3

During the third scenario, the number of obtained packets increased to 119. The experiment started at 10:38:28. The subject stood up at approximately 10:39:35. In the resulting data set, there were 8 instances in time, where at least three sniffers were capturing the packets of the smartphone. The smartphone could be located between 10:39:46 and 10:40:00. This time window corresponds to when the subject got up and left the room. This might indicate that the sensors might have been blocked from receiving the smartphone's signal in the stationary place. When processing the data set, the location of the smartphone can be visualized in Figure 5.3.

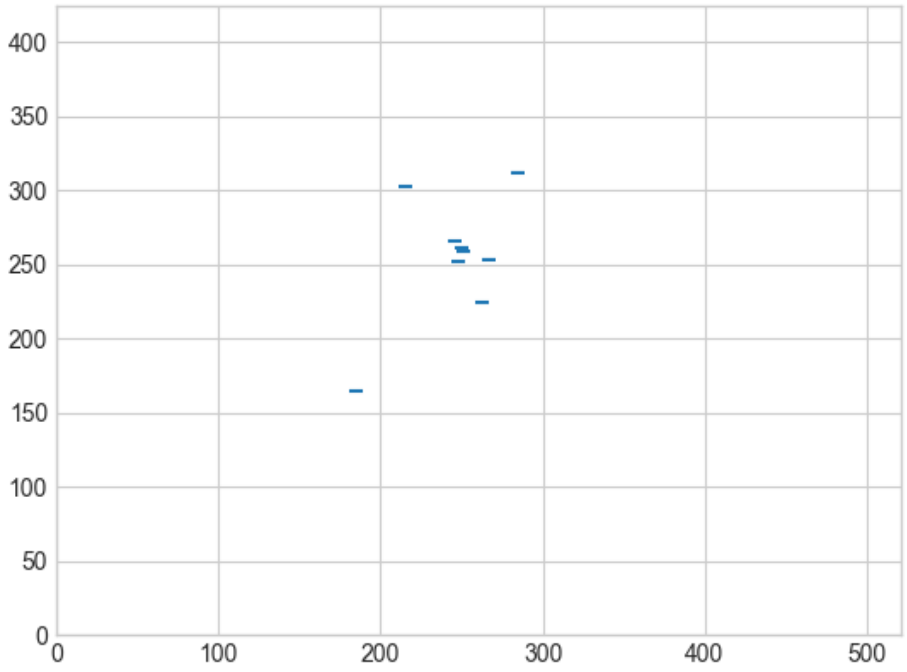

Figure 5.3: location estimation scenario 3

.

As already described in the results of the first scenario, the localization results obtained

from the data of the third scenario are more or less random and can not really be relied on for localization of the device.

Scenario 4

The data obtained in scenario 4 was sparse. There were a total of 45 packets sent by the smartphone registered. Out of these 45 data points, there was only one case, where there were three or more matching timestamps. Therefore, the device could only be trilaterated once. Capturing of Wi-Fi packets turned out to be very unreliable. A lot of packets are only captured by less than three Wi-Fi sniffers.

5.2.4 Discussion

Regarding the requirements described in Section 3.1, we can conclude following based on the experiment conducted:

- **R1:** In experiment 1, we could show that our approach is capable of registering smartphones without any interaction required, other than having WLAN enabled on the device. Therefore, detecting smartphones based on Wi-Fi signals is possible.
- **R2:** Based on the second experiment conducted, we could not reach the second requirement. This is mostly due to the fact, that the readings of the used sensors are not accurate enough.
- **R3:** The calculated positions of the smartphone during the first scenario of the second experiment were used to calculate the mean for every 30 second time window. The then obtained position of the smartphone was used in combination with the MAC address of the smartphone. For every distinct MAC address, the position is used to determine, where the device lies. Even though the positions of the devices were net accurate, the counting of the devices works as planned.
- **R4:** Based on the positions calculated in the first scenario in the second experiment, the warning function could be evaluated. For this, the maximum person count was set to 1. This led to the program printing out warning messages to the console. Therefore, the warning system would also work when the localization would be working.

During the time of the thesis, there were some unexpected events and complications, which influenced the development of this masters thesis. The main goal of this thesis was to build a counting algorithm based on already existing hardware and software. The Wi-Fi sniffer should be supplied. However, the supplied hardware and data access method did not satisfy the requirements to build a stream processing application doing person counting. Therefore, it was decided, that there was the need to implement our own Wi-Fi sniffers.

The implementation of the Wi-Fi sniffers was not planned as part of the initial goals thesis, but had to be done, to obtain data to evaluate a proposed counting approach. Due to the limited budget, a cost effective approach was chosen by using Raspberry Pi zero W boards, due to them not requiring any additional hardware to capture Wi-Fi.

However, since the default Kernel of the Raspberry Pi does not enable to use the Wi-Fi interface as a monitor. The process of patching the software of the Raspberry pi turned out to be very time consuming. Approximately six weeks were lost just to develop the Wi-Fi sniffers.

The covid situation during December 2021 and January 2022 worsened to a point, where meeting of bigger crowds were either forbidden or at least discouraged. Therefore, the evaluation of the implemented system in a real world scenario could not take place as it would have been planned.

The implemented Wi-Fi sniffers and the data captured by these sniffers could not be used to replicate the findings presented in [16] and [17]. This contradiction of findings might be due to the quality of the Wi-Fi sniffers used. Furthermore, the raspberry pi were also not able to reliably change the channel of the monitoring interface. This is due to the monitoring interface sharing the wireless chip with the normal WLAN interface used to connect to WLAN. Therefore, the sniffers were mostly only able to capture on the channel they connected to WLAN.

This indicates, that reusing already deployed WLAN hardware might not be as easy as expected. Most of the already used hardware might not even support monitoring of Wi-Fi packets. Furthermore, patching the AP to support monitor mode will most likely come with the cost of decreasing the performance of the AP, since it will use some of the hardware for monitoring.

Chapter 6

Final Considerations

6.1 Summary

During the period of this master thesis, the state of the art approaches for crowd density approximations have been studied. Based on the requirement, that the proposed system should work out of the box without any upfront sampling of the location, the decision was taken, to settle for an approach that uses Wi-Fi sniffers to capture devices Wi-Fi signature, instead of utilizing an interference based approach.

Based on this decision, a Wi-Fi sniffer was developed to capture packets that are only emitted by devices and not by APs. The implemented Sniffers use the on board Wi-Fi chip of the Raspberry Pi zero W. The RSSI of the captured packets were then used to trilaterate the smartphones. This was done after the data collection by the localization engine.

To evaluate, whether the capturing approach is suitable, there was one experiment conducted to verify, that devices could be sensed independent of their connection status, as long as WLAN was enabled. The second experiment conducted suggested, that the RSSI based localization in the chosen environment is not working as expected. The RSSI readings did not at all correspond to the distance to the sender. Therefore, the proposed counting approach relying on the position of the smartphone did not work as planned.

6.2 Considerations

Based on the results of the experiment conducted, we would suggest that an RSSI based localization of smartphones in an indoor environment is not accurate enough. This might be due to the Wi-Fi sniffers being inaccurate. Furthermore, signal propagation effects in indoor environment further complicate the distance estimation based on RSSI.

Another takeaway from this project is, that the utilized Raspberry Pi boards were sensitive to certain interactions. They did tend to freeze during the installation of packages.

Disconnecting the power source then lead to the file system being corrupted. This is crucial, as the Raspberry Pi were sometimes not powered of by a command, but by the powerbank running empty.

6.3 Future Work

The RSSI values obtained during this work turned out to be not accurate enough to localize a device in an indoor environment. Therefore, improving the system could be done in two different ways. On one hand, it could be evaluated, if other Wi-Fi sniffers produce better results and more accurate readings for the RSSI value. Another approach would include the utilization of a different metric to estimate the distance between the Wi-FI sniffer and the device.

On of such a metric might be the distance calculation based on TOA of the packets captured. This approach would then require a very precise time synchronization of the Wi-Fi sniffers, down to the milliseconds, since the passed time would be used to estimate distance. Since radio waves travel close to light speed through the air, a slight difference in time would lead to big errors in distance calculation.

The final implementation has a variety of points, where it could be improved. One of them would be to implement streaming on the sensors and adapt the localization engine to be capable to process data streams. Furthermore, the visualization of the data could be done in a close to real time way, where the sensed devices are visualized directly.

Bibliography

[1] M. Gast, *802.11 wireless networks: The definitive guide*. Southeast University Press Nanjing, JiangSu, China, 2006.

[2] "Ieee sa - ieee 802." [Online]. Available: https://standards.ieee.org/featured/802/index.html

[3] [Online]. Available: https://www.wi-fi.org/who-we-are

[4] M. Chapple, J. M. Stewart, and D. Gibson, *(ISC) 2 CISSP Certified Information Systems Security Professional Official Study Guide*. John Wiley & Sons, 2018.

[5] I. Management Association, *Cyber Law, Privacy, and Security: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2019.

[6] A. Blanco and M. Eissler, "One firmware to monitor'em all," in *Ekoparty Security Conference*, 2012.

[7] B. f. K. BAKOM, "Anzahl mobilfunkkundinnen und -kunden," https://www.bakom.admin.ch/bakom/de/home/telekommunikation/zahlen-und-fakten/sammlung-statisticher-daten/mobilfunk/anzahl-mobilfunkkundinnen-und-kunden.html.

[8] W. Murphy and W. Hereman, "Determination of a position in three dimensions using trilateration and approximate distances," *Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado, MCS-95*, vol. 7, p. 19, 1995.

[9] K. Woyach, D. Puccinelli, and M. Haenggi, "Sensorless sensing in wireless networks: Implementation and measurements," in *2006 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*. IEEE, 2006, pp. 1–8.

[10] P. W. Q. Lee, W. K. G. Seah, H.-P. Tan, and Z. Yao, "Wireless sensing without sensors - an experimental approach," in *2009 IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*, 2009, pp. 62–66.

[11] A. Musa and J. Eriksson, "Tracking unmodified smartphones using wi-fi monitors," in *Proceedings of the 10th ACM conference on embedded network sensor systems*, 2012, pp. 281–294.

[12] H. Li, E. C. Chan, X. Guo, J. Xiao, K. Wu, and L. M. Ni, "Wi-counter: smartphone-based people counter using crowdsourced wi-fi signal data," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 4, pp. 442–452, 2015.

[13] T. A. Myrvoll, J. E. Håkegård, T. Matsui, and F. Septier, "Counting public transport passenger using wifi signatures of mobile devices," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

[14] W. Xi, J. Zhao, X.-Y. Li, K. Zhao, S. Tang, X. Liu, and Z. Jiang, "Electronic frog eye: Counting crowd using wifi," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 361–369.

[15] H. Zou, Y. Zhou, J. Yang, W. Gu, L. Xie, and C. Spanos, "Freecount: Device-free crowd counting with commodity wifi," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[16] R. H. Ribeiro, B. B. Rodrigues, C. Killer, L. Baumann, M. F. Franco, E. J. Scheid, and B. Stiller, "Asimov: a fully passive wifi device tracking," in *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 2021, pp. 1–3.

[17] B. Rodrigues, C. Halter, M. Franco, E. J. Scheid, C. Killer, and B. Stiller, "Bluepil: a bluetooth-based passive localization method," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 28–36.

[18] B. Rodrigues, I. Cepilov, M. Franco, E. J. Scheid, C. Killer, J. von der Assen, and B. Stiller, "CCount: Correlating RFID and Camera Data for High Precision Indoor Tracking," University of Zurich, Department of Informatics, Tech. Rep., 01 2022.

[19] "Penetration testing and ethical hacking linux distribution," Jan 2022. [Online]. Available: https://www.kali.org/

[20] M. Schulz, "Seemoo-lab/nexmon: The c-based firmware patching framework for broadcom/cypress wifi chips that enables monitor mode, frame injection and much more," Nov 2021. [Online]. Available: https://github.com/seemoo-lab/nexmon

[21] R. P. Foundation, "Operating system images." [Online]. Available: https://www.raspberrypi.com/software/operating-systems/#raspberry-pi-os-32-bit

[22] "Re4son-pi-kernel," May 2019. [Online]. Available: https://re4son-kernel.com/re4son-pi-kernel/

[23] M. Viswanathan, "Log distance path loss or log normal shadowing model," *Accessed: Jul*, vol. 3, p. 2020, 2013.

[24] mister_x, "Aircrack-ng," Jan 2020. [Online]. Available: https://www.aircrack-ng.org/doku.php?id=airodump-ng

[25] U. Lamping and E. Warnicke, "Wireshark user's guide," *Interface*, vol. 4, no. 6, p. 1, 2004.

[26] P. Biondi, "Scapy documentation (!)," *vol*, vol. 469, pp. 155–203, 2010.

[27] W. McKinney *et al.*, "pandas: a foundational python library for data analysis and statistics," *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.

[28] J. Hunt, "Graphing with matplotlib pyplot," in *Advanced Guide to Python 3 Programming*. Springer, 2019, pp. 43–65.

[29] A. Alexander, "Easy-trilateration," Jul 2021. [Online]. Available: https://pypi.org/project/easy-trilateration/

[30] R. W. Bohannon, "Comfortable and maximum walking speed of adults aged 20-79 years: reference values and determinants," *Age and ageing*, vol. 26, no. 1, pp. 15–19, 1997.

[31] M. Hidayab, A. H. Ali, and K. B. A. Azmi, "Wifi signal propagation at 2.4 ghz," in *2009 Asia Pacific Microwave Conference*. IEEE, 2009, pp. 528–531.

Abbreviations

AP	Access Point
BSSID	Basic Service Set ID
CSV	Comma-separated values
DA	Destination Address
IEEE	Institute of Electrical and Electronics Engineers
LAN	Local Area network
MAC	Media Access Control
RFID	Radio-frequency identification
RSS	Received Signal Strength
RSSI	Radio Signal Strength Indicator
RTS	Request to Send
SA	Source Address
SFTP	Secure File Transfer Protocol
TOA	Time of Arrival
UHF	Ultra high frequency
WLAN	Wireless Local Area Network

List of Figures

List of Tables

Listings

Appendix A

Contents of the Submission

The submission is available under following **link**. The submission contains following files and folders:

1. Master Thesis as PDF (WiFi_based_Crowd_Safety_System.pdf)
2. Master Thesis as LaTeX-source code (WiFi_based_Crowd_Safety_System.zip)
3. Wi-Fi Counting source code as per 27.01.2022 (Wi-Fi-Counting.zip)
4. Datasets of all experiments (experiments.zip)
5. Raspberry Pi Image including capturing script as per 27.01.2022 (Capturing_Img.zip)

Appendix B

Repository description and running instructions

The submitted repository is structured as following:

- Data: contains input and output data
  - configuration.yaml: used for application configuration
- Pi_Script: contains the capturing script used on the Wi-Fi sniffer
  - capturing.py
- Service: contains the different service scripts
  - dataHandling.py
  - mergeCSV.py
  - personCount.py
  - trilateration.py
- venv
- main.py: main file

B.1 How to Run

:

B.1.1 Wi-Fi sniffer

To run the Wi-Fi sniffer, the user has to log in to the raspberry pi. The username is **pi** and the password is **raspberry**. For easier access, connect the pi to WLAN and then use SSH.

The capturing.py script can be run using following command from the home directory:

```
1 sudo python3 Documents/Capturing/capturing.py mon0 0 10
```

Where the first argument defines the script, the second the monitoring interface, the third specifies the sniffer id and the fourth argument is used to specify the capturing time in minutes.

B.1.2 Wi-Fi-Counting

To run the Wi-Fi-Counting program, the configuration file has to be modified to represent the current setting. Following attributes need to be specified:

- SensorType: either 'Pi' or 'Liveanalytics'
- File: relative path to the input data in format : 'address','rssi_mean','time', 'sensorid' for the 'Pi' configuration
- Output: relative path to the output file to store the counting results
- TimeWindow: size of timewindow in seconds
- n: n used for distance calculation
- P(d): path loss at distance d
- d: distance for P(d) measurement in centimeters
- Sensor: ID and position of the sensors see Listing 4.2
- Area: area configuration for person counting see Listing 4.2

Once the program is configured, the evaluation can be run by running the main method in the main.py file.

Appendix C

Tables

Type value	Subtype value	Name
00	0000	Association request
	0001	Association response
	0010	Reassociation request
	0011	Reassociation response
	0100	Probe request
	0101	Probe response
	1000	Beacon
	1001	Announcement traffic indication message (ATIM)
	1010	Disassociation
	1011	Authentication
	1100	Deauthentication
01	1010	Power Save (PS)-Poll
	1011	RTS
	1100	CTS
	1101	Acknowledgment (ACK)
	1110	Contention-Free (CF)-End
	1111	CF-End+CF-Ack
10	0000	Data
	0001	Data+CF-Ack
	0010	Data+CF-Poll
	0011	Data+CF-Ack+CF-Poll

Table C.1: Type and subtype identifiers in MAC 802.11[1]

Time	x	y
22:16:00	216	230
22:16:03	234	266
22:16:04	207	229
22:16:05	182	195
22:16:35	180	260
22:16:36	207	234
22:16:37	201	226
22:16:38	208	229
22:16:39	207	229
22:16:40	207	224
22:16:41	203	223
22:16:42	210	232
22:16:43	209	227
22:16:55	179	181
22:16:57	169	173
22:16:58	169	173
22:16:59	212	234
22:17:03	222	240
22:17:08	214	238
22:17:09	191	254
22:18:13	230	265
22:19:03	234	190
22:19:04	207	219
22:19:05	226	256
22:19:40	174	174
22:19:57	208	222
22:20:01	238	191

Table C.2: Trilateration result of scenario 1