Network-based Localization of Wi-Fi Devices

Bachelor Thesis

David Schunke

RWTH Aachen University, Germany
Chair of Communication and Distributed Systems

Advisors:

Dipl.-Inform.	Nicolai Viol
Dipl.-Inform.	Martin Henze
Prof. Dr.-Ing.	Klaus Wehrle
Prof. Dr.	Bernhard Rumpe

Registration date: 2012-12-22
Submission date: 2013-05-20

I hereby affirm that I composed this work independently and used no other than the specified sources and tools and that I marked all quotes as such.

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, den 17.05.2013

Abstract

This thesis focuses on the design and implementation of a network-based localization system of Wi-Fi devices. For this, our system uses access points to collect measurements of the received signal strengths from the wireless traffic and imposes no active participation of the target device in the localization process. The actual estimation of the localization is delegated to an existing localization framework that provides different deterministic and probabilistic methods comparing the RSS measurments to a prior-build radio propagation map. Based on this implementation, the localization error is evaluated in various scenarios of different complexity. The localization accuracy of different devices are compared to investigate the device adaptation of our system. Additionally, the results are compared to a client-based implementation. The evaluation showed that our system is capable to determine a devices location with an error of few meters. In our test environment, an error of up to $1.75m$ could be achieved and degrades to more than $18m$ in difficult scenes such as stairwells. Overall, the network-based implementation reached in our test environment a higher accuracy than the client-based solution.

Acknowledgments

I want to thank my supervisors Nicolai Viol and Martin Henze for their support and effort on this work. Further, I want to thank Prof. Dr.-Ing. Klaus Wehrle for the opportunity to work on this topic and making this thesis possible. I also want to thank Prof. Dr. Bernhard Rumpe for agreeing to be the second assessor.

Contents

1 Introduction

The approach of determining an objects location is an ancient purpose. In evolution, several animals such as bats and dolphins have adapted methods using echo systems to orientate in environments where vision is limited, so they can move and hunt in totally darkness. Humans use localization techniques in navigation in a long time. In former times, celestial alignments where used to navigate ships to their intended destination. Today, more sophisticated techniques are applied to handle maritime and air traffic safe and efficient, due to the drastically increased amount of traffic. In personal usage, as more and more people carry mobile devices such as smartphones and tablets, the number of location-based applications that use the physical position increases as well. Ubiquitous examples are map and route traveling services and even advertisments use location information to provide individually fitted offers. Hence, the importance of accurate localization systems keeps increasing.

The most often applied system is the Global Positioning System (GPS) which uses a network of satellites and can determine a devices position with an error around $1m$ [22]. The usage of GPS requires additional hardware which increases costs and energy consumption. Furthermore, GPS is in general not feasible in indoor environments because of interferences on the signals. Other systems use, e.g., GSM[1] networks and the positions of radio towers, but its accuracy ranges from several meters up to kilometers.

For indoor environments, there exists no easy to apply solution yet that could be called state-of-the-art. Research on this topic varies in the source of information as well as the localization process itself and in general, the required accuracy in indoor localization is higher than in outdoors. Today, implemented systems for indoor localization are extremely adapted to their use case and environment to achieve an accuracy of few meters.

[1]Global System for Mobile Communications

1.1 Motivation and Challenges

Due to the desire to build an easy to adapt localization system, many approaches tend to use already existing infrastructure such as Wi-Fi networks since Wi-Fi is a common technique in wireless communication. Other common technologies that could serve as information sources are, e.g., ZigBee that is often used in Wireless Sensor Networks (WSN), Bluetooth, infrared or ultrasound. Even measurements from specialized sensors such as an accelerometer or a compass could be used. In this work, we will focus on the usage of Wi-Fi radio signals.
There are many approaches on indoor localization that are based on Wi-Fi, but most approaches use a client-based implementation where the target object for which the location has to be determined serves as measuring device. This requires an active participation of the client in the localization process and mostly needs specific software installed on the target device.
Another approach is to use Wi-Fi access points (APs) as measuring points to collect information about the target device. In contrary to the mentioned client-based implementation, this is called a network-based implementation. This eleminates the requirement of an active participation and thus introduces further possible use cases in which knowledge of the localization process at the target object is not intended. Nevertheless, this makes the localization approach dependend on the wireless network traffic emitted by the target device. Furthermore, no additional software is required on the target device.

Our goal on this topic is to build a simple to adapt network-based indoor localization system and evaluate its accuracy. Due to this, in this thesis we will focus on the use of Wi-Fi radio signals because of the widely spread usage of Wi-Fi and the easy deployment of Wi-Fi APs. We will build a network-based architecture using a priorly implemented localization framework, evaluate this system in various scenarios and compare the results to a client-based solution. Furthermore, we will evaluate the amount of wireless traffic a standard Wi-Fi device emits and investigate approaches on increasing the amount. Finally, we will go into the topic of privacy and security, researching client-sided possible counter measures against our system.

1.2 Outline

This thesis is structured into seven chapters. This introduction is followed by the background chapter stating the fundamentals of Wi-Fi, the process of frame capturing and how the location of an object can be determined. In Chapter 3, we present the localization framework that we use in this work, other approaches on network-based indoor localization of Wi-Fi devices, and discuss the differences to our implementation. Chapter 4 illustrates the design of our system and explains the architecture, each component and discusses possible counter measures against our system. Chapter 5 then explains the technical implementation and is followed by the evaluation chapter. The last chapter concludes the results of our research and points out possible future tasks.

2

Background

In this chapter we describe the fundamentals of the main topics that are required to apprehend the process of localization and our solution. For this, we give an overview of the functional principle of Wi-Fi and explain the necassary headers and protocols. We take a brief look at packet capturing and how to obtain information from network traffic and finally at the localization process itself.

2.1 Wi-Fi

Wi-Fi is a specification based on the IEEE 802.11 standard [14]. Due to simplification reasons, in this work we can reduce the term of a Wi-Fi device to every device that can participate in an IEEE 802.11 network. This may cover notebooks or smartphones as well as wireless sensor nodes, network routers and APs.
Regarding to the OSI model, IEEE 802.11 defines the medium access and addressing on the data link layer as well as the physical transmission using radio signals [18]. As all kinds of waves, radio waves are affected by different physical phenomena such as reflection, absorption, refraction, interference and more. There exist standards for communicatating in the 2.4, 3.6, 5, and $60GHz$ frequency bands which refer to the radio wave length, but the typically used frequencies are 2.4 and $5GHz$. For this work, the mainly important difference between these frequencies is that the $2.4GHz$ band is also used by other technologies such as Bluetooth or microwaves. Hence, interferences are more probable in the 2.4 than in the $5GHz$ frequency band.
Additionally to reduce interferences between nearby but independent Wi-Fi networks, the frequency band is divided into several channels. The channel distribution of the $2.4GHz$ frequency band is shown in Figure 2.1. Due to a bandwith of $22MHz$, adjacent channels overlap which can lead to channel overhearing. In that case a device can receive a packet on a different channel than it was sent on which leads to a corrupt received signal strength (RSS), which we describe in the following.

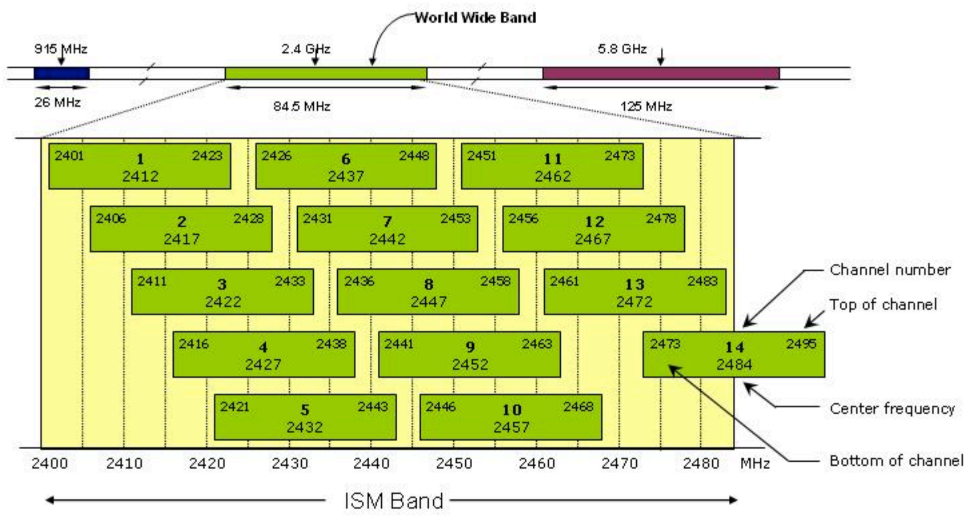


Figure 2.1 The figure illustrates the channel distribution in the $2.4GHz$ frequency band [4].

2.1.1 Signal Strength

The signal strength is a value describing the power level a packet has been received with. It is measured in dBm which can be expressed by the equation

$$dBm = 10 * log_{10}(1000 * mW)$$

where $0dBm$ equals $1mW$.
After transmission, the value decays exponential by the distance according to the inverse-square law, so that a receiving device gets a signal strength proportional to its distance to the transmitter. The typical received signal strength in IEEE 802.11 networks range from $-10dBm$ to $-90dBm$. According to the formula above, the higher or less negative the signal strength is, the better is the link quality.

RSSI vs. Signal Strength

The Received Signal Strength Indicator (RSSI) is a metric indicating the quality of a link by using the received radio signal strength. The IEEE 802.11 standards do not define how this value has to be calculated, so it depends on the vendor of the wireless network interface controller (WNIC).
The standard only defines the RSSI value as an optional 1 byte integer parameter, which may be added to the received packet. Typically, the value ranges from 0 to a RSSI-max value or is returned in percentage and used internally by the WNIC to determine the link quality. More precise, the WNIC checks whether the value is beneath a defined threshold to determine, e.g., when to check for a better channel or another AP with a better link to. The RSSI value can be obtained using monitor mode, on which we will go into in Section 2.2.1.

2.1.2 Modes and Addressing

The usage of Wi-Fi can be divided into two basic modes: ad-hoc and infrastructure mode. In ad-hoc mode, the devices communicate directly with each other, called

peer-to-peer. In infrastructure mode, the communication is handled via a central unit. Clients connect to an AP which then redirects the message to the actual recipient.
In both modes, the addressing is realized by an unique identifier: the MAC-address. Every Wi-Fi device has a MAC-address that is bounded to the network interface controller (NIC). A NIC checks all messages it receives if its own MAC-address is stated as recipient. For this and to correctly interpret the message, a standard format for transmission is required, which we describe in the following.

2.1.3 IEEE 802.11 frames

The IEEE 802.11 standard defines a general frame format as shown in Figure 2.3.

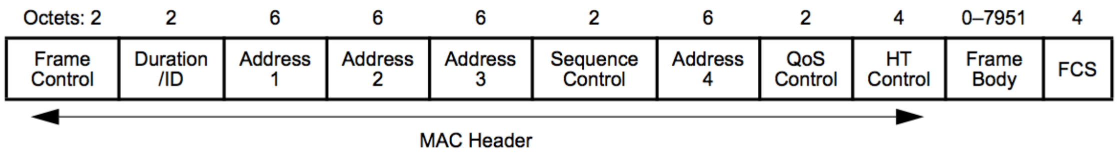


Figure 2.2 General frame format of IEEE 802.11 frames with defined length in bytes [18].

The frame control field consists of two bytes defining the protocol version, type and subtype of the frame plus further management bits. There are three different frame types defined: (1) *management* for communication maintenance, (2) *control* to facilitate the delivery of data frames, and (3) *data* frames for the actual transmission of content.
Duration holds of 2 bytes and is only used for control frames indicating the remaining time needed to receive the next frame.
Frames may contain up to four address fields. The first address is always the recipient and the second one the transmitting address. The third address field is only present in management and data frames and contains the missing original source or destination field, depending on values of the frame control field. The fourth is only used for Wireless Distributed Systems (WDS).
The remaining fields are not required in this work. Their meaning and usage can be looked up in [18].

The first three fields *frame control*, *duration* and *address 1* have to be present in every possible frame, forming the minimal possible *MAC header*. The header is followed by a *frame body* of variable length containing the content specific to the type of the frame. The last part of an IEEE 802.11 frame is the *Frame Check Sequence* (FCS) holding 4 bytes and using a Cyclic Redundancy Check (CRC) for error detection.

2.2 Packet Capturing

Packet capturing describes the practice to obtain messages from the network traffic even if they are not addressed to the capturing device. For this, the actual addressing

process as described in Section 2.1.2 is circumvented. Packet capturing is mostly done for debugging but can also be used to illicitly obtain information about the participating devices or even in network attacks.
The most common toolchain for this approach is packet capture (pcap) [12]. UNIX systems implement the libpcap and Windows uses an adaptation of libpcap called WinPcap. Libpcap is a low-level library written in C and implemented as fat binary, so it supports multiple instruction sets and can be used on 32 and 64-bit systems. It gives functionallity to monitor network traffic as a so called sniffer, to obtain low-level information from the NIC and even to inject network traffic. For this purpose, it needs advanced permission called monitor mode, which is described in the following.

2.2.1 Monitor Mode

NICs support different modes they can be set to, one is the monitor mode (sometimes also called RFMON for radio frequency monitor). Unlike the promiscuous mode, in which the NIC only receives packets within its associated broadcast domain, the monitor mode allows to obtain all packets a NIC can physically receive. It then adds an additional header to the packet with physical and data link information, such as, e.g., the RSS value or the channel it has been captured on. The type of this link-layer header depends on the chipset and a NIC can support multiple different types. In this work, we use the so called Radiotap header, which we describe in the following.

2.2.2 Radiotap Header

The Radiotap header format provides additional information either from the driver to a userspace application, or vice versa [10]. It is flexible in its size and structure, giving the developer flexebility to define an arbitrary number of fields depending on a bitmask.

```c
struct ieee80211_radiotap_header {
    u_int8_t      it_version; /* set to 0 */
    u_int8_t      it_pad;
    u_int16_t     it_len;     /* entire radiotap data */
    u_int32_t     it_present; /* bitmask for present fields */
}
```

Listing 2.1. This listing shows the radiotap header format as a C struct.

The format starts with the main radiotap header, defining the header version (always set to 0), the entire radiotap header length including the data fields and a bitmask of the present radiotap data fields. The *it_pad* field is only for padding the size to 8 instead of 7 bytes. It is directly followed by the data fields in strict order in which the present bitmask is defined. The data fields differ in length and type, but are fixed to their definition. This means that fields may not be variable in their length. The field for, e.g., the RSS holds the bit number 5 and contains an 8-bit signed integer with a value in dBm. A complete list of defined fields can be obtained from [10].
The last data field is immediately followed by the IEEE 802.11 frame.

2.3 Localization

The process of localization is defined as determining either the physical or a logical position of an object. In general, this task covers two main topics: *what type of information is used to obtain information about an object* and *how to process this information to an actual location.* For both parts exist various approaches and can be replaced by a different one without harming the other part. For calculating the distance there exist several interesting approaches [20]:

Received Signal Strength (RSS)
: As explained in Section 2.1.1, the RSS of a radio signal is proportional to the distance between transmitter and receiver following the inverse-square law [16][24]:

$$RSS \sim \frac{1}{Distance^2}$$

This states that the RSS value is proportional to the inverse of the squared distance. Hence, the distance can be calculated from the RSS value.

Time Of Arrival (ToA)
: In ToA - also called Time of Flight (ToF) - a transmitter sends its system timestamp to the receiver that will calculate the distance. The receiver simply subtracts this timestamp from its own system timestamp and thus obtains a time difference according to the distance between the two devices. The distance can be calculated by combining this time difference with the priorly known propagation speed of the signal, which is the velocity of light for radio signals. This method is highly error-prone to time shifts, hence the two systems need to be highly time synchronized.

Time Difference of Arrival (TDoA)
: TDoA uses two ToA measurements. Instead of one signal, a transmitter emits signals of different types, e.g., ultrasound and radio signals. As mentioned before, radio signals travel with the velocity of light while ultrasound propagates with sonic speed, thus significantly slower than radio signals. Therefore, the receiver can compute a difference between the times both signals arrive. From this, the process continues analog to ToA. This method is less error-prone to time shifts since it only relies on one single system clock and, thus does not require time synchronization.

Angle of Arrival (AoA)
: AoA - also called Direction of Arrival (DoA) - determines the angle a signal arrives at a receiver. This can either be done by directional hardware such as antennas or microphones. Another way is to use two receiving devices with well known distance between them. Using again the TDoA at these two receivers and the velocity of the signal, these information can be converted to the directional angle of receiver and transmitter. This method is very error-prone to reflection and multipaths of signals.

After having relative distance or direction information about the target, the location can be computed using several procedures [20]:

Trilateration

This method uses the estimated distance and the geometry of circles or spheres to determine the location of the target object. Assuming a two-dimensional space, at least three anchor points with a measured distance to the target object are required to calculate the location of the target. At each anchor point, a circle with the radius of the distance is formed and the intersection of all circles represents the targets location.

Multilateration

Contrary to trilateration, multilateration uses TDoAs instead of distances. Considering two receivers and a measured TDoA, the possible locations of the transmitter form one half of a two-sheeted hyperboloid. Adding a third receiver creates a second TDoA and thus a second hyperboloid. The two hyperboloids then intersect in a curve of possible locations. With a fourth receiver, a third TDoA and hence a third hyperboloid can be formed intersecting with the two already found hyperboloids in a unique point. The searched location of the target is then the intersecting point.

Triangulation

Triangulation uses angles to estimate the location. In a two-dimensional space, two measured angles are required. From these angles and one known distance, a triangle can be formed with the target object fixed at one point using the law of sines.

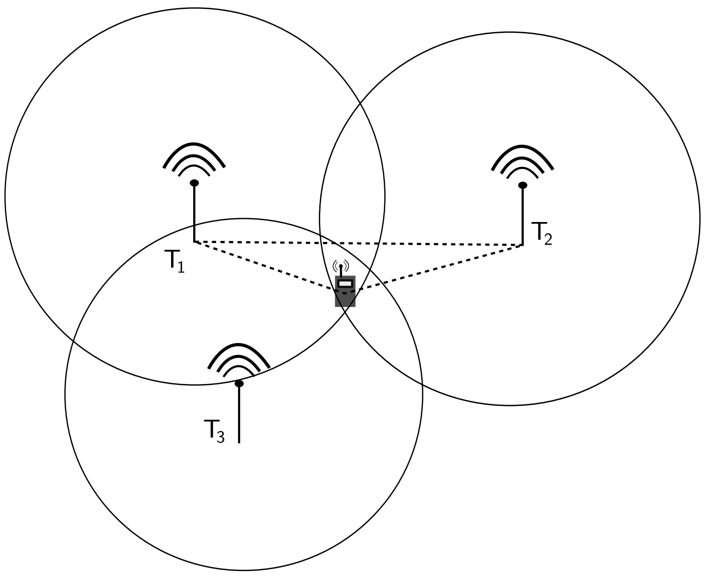


Figure 2.3 The figures illustrates the triangulation method in a two-dimensional space. [25].

Fingerprinting

Fingerprinting uses the raw RSS measurements to determine the targets location. The RSS data is compared to a priorly build RSS database or map. The best fitting stored location according to the RSS values is then chosen as the determined location.

Probabilistic Methods

The previous methods can be called deterministic as they compute a single resulting determined location. Contrary to that, probabilistic methods use likelihood ratio theory to assign a probability to each possible location. Statistical models are used for this and the probabilities are related to the measurements and the specific timestamp.

2.3.1 Accuracy and Error Containment

As mentioned before, radio signals are affected by various physical phenomena. Absorption and interferences change the signal strength so that a wrong distance will be calculated from the RSS value. Reflection multiplicates the original single to numerous additional signals, which is called multipath. When corrupt distance values are then used in, e.g., trilateration, the interesction does not form a single point, but an area.
When there is a direct line of sight between transmitter and receiver available, signals are still slightly influenced by, e.g., temperature, light or air pressure. Additionally, in indoor environments, there usually is no direct line of sight because of walls, furniture and moving people. Hence, it is in general not possible to determine a targets location absolutely accurate. There exist several different approaches in the field of optimization. Covering this topic closer goes beyond the scope of this paper.

3

Related Work

This chapter discusses related approaches on indoor localization. In Section 3.1, we present a raytracing-based localization framework that we use in this work for the calculation and evaluation process. In Section 3.2, we discuss other network-based approaches on localizing Wi-Fi devices and their differences to our work.

3.1 Localization Framework

The localization framework has been developed and implemented at RWTH Aachen University by Dirk Rothe et al. [25] [28]. It covers functionality for graphical visiualization, evaluation and the actual localization process based on bayesian inference using RSS readings. First, it generates a signal strength database using recorded RSS reference measurements at well known locations and a radio propagation model. The RSS measurements for the localization process are then compared to the signal strength database and the location is determined by estimating the best fitting result regarding the used algorithm.

Architecture

The framework is build as a single server instance using a webserver to receive the measurement data from a single or multiple sources via HTTP. It consists of five major components:
The Optimizer trains the radio propagation model optimizing the parameters regarding the different materials in the environment using initially supplied reference measurements at well known locations.
The Simulator is the organization unit responsible for distributing the calculation jobs to nodes of a GPU cluster and recovers the results which are then send back to the origin.

The Localizer implements the localization algorithms and uses the optimized propagation model for estimating the location relating to the RSS measurements.
The Evaluator aggregates the error over the estimated locations and provides information about the performance and debugging information. It also gives a graphical output of the results and the error that can be accessed via a webpage.
The fifth component is stated as "General Infrastructure" [25] and contains the HTTP-Server, database and further applications.

In the presented test scenario, a client-based implementation for gathering the RSS measurements has been realized on an Android device. The client observes the traffic from the deployed APs, collects the RSS measurements and sends them to the framework. Furthermore, it implements additional functions for debugging and visualization.

Radio Propagation Model

The major error source for localization approaches in indoor environments can be reduced to the different objects and materials between transmitter and receiver of the Wi-Fi radio signal as they induce the phenomena such as reflection and absorption. To improve the validity of the used signals and therefore also the accuracy of the finally estimated location, the different materials can be considered in the calculation process. For this the propagation of radio signals in the given environment is modelled to create a signal strength map. The radio propagation is related to the position of one AP, so it is not possible to move the AP afterwards without corrupting the signal strength map. Due to this and the purpose to finally visualize the estimated locations on a map, the positions of the used APs have to be preconfigured with well known coordinates relative to the environmental scene. The scene geometry is given as a 3D-model in form of an .obj-file created with the software Blender [1]. This model consists of multiple objects representing the different materials such as walls and doors. Each material has a parameter for reflection and transparency which then have to be optimized.
In this approach, a so called PHOTON raytracer [27] is used for parameter optimization, modelling a radio wave as a single particle and then simulating the paths for all waves [25]. Afterwards, different smoothing methods are applied to the simulated paths and finally a signal strength map is created with a denoted average error within 6-8 dBm.

Localization

In the localization process, the best fitting estimated location regarding the propagation models from all AP and the measured RSS readings is taken as the determined location. For this task, three algorithms can be used in this framework which are all based on bayesian decision theory [25]:

Hidden Markov Model (HMM)
HMM is a statistical model taking physical constraints into account, i.e., consecutive locations are dependend in the physical world [29] [28] [23]. In general,

a Markov Model says that for temporal discrete processes, the future is independet of the past given the present state. In case of localization this means that the temporal adjacent location of an object depends on the present one, not on past locations. Referring to localization using RSS measurements, there are two types of parameters: the location z_t at time t and the RSS values of the APs as vector x_t. The location, i.e., state, is the searched parameter and therefore called "hidden". Instead, the RSS measurements are observed which depend on the location and are called "emissions". This is illustrated in Figure 3.1.

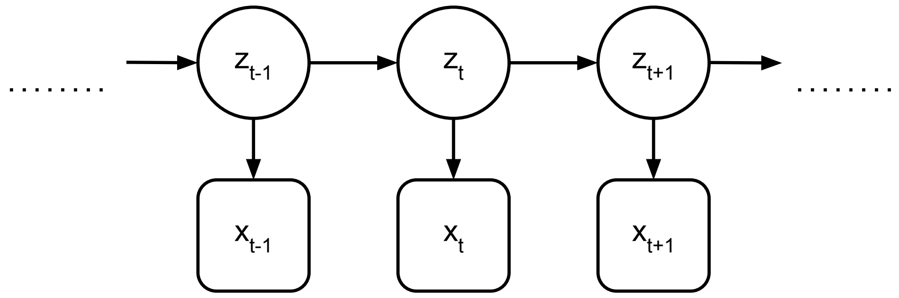


Figure 3.1 The figure illustrates the dependencies in a Markov chain. The z_t represent the hidden states and the x_t the emissions.

Be t the current time, the state z_t only depends on the previous state z_{t-1} and the emission x_t only on the state z_t. Neither z_t nor x_t depend on the previously observed emission x_{t-1}. Therefore, these dependencies form a chain, called Markov Chain.

Considering these dependencies, each future state has a certain probability. In HMM are the probabilities for all possible matching locations calculated and the final resulting one as the determined location is the state with the maximum probability. More advanced information on this model can be obtained from [25].

Particle Filter (PF)

Similar to HMM, PF considers the dependency of adjacent states and locations are connected in a Markov chain. The difference to HMM is, that PF uses a heuristic strategy to determine the location. PF estimates the distribution model for only one location, hence in general, it finds only a local optimal result which may deviate from the global optimum. In HMM all possible distributions are estimated and therefore the global optimum is found. This leads to less computing effort for PF compared to HMM, so it needs less hardware resources respectively less computation time and can therefore be efficiently used on less powerfull devices.

Least Mean Square Error (LMSE)

LMSE is a simplistic model and a basic method in linear optimization. In contrast to HMM and PF, the dependencies of adjacent locations or measurements are neglected and only the present state is taken into account. Be x_t

the vector of measurements from the APs at time t, it is simply compared to the values of the signal strength map by calculating the Euclidean distance. From all results, the minimum is taken as the best fitting location.

Offline vs. Online

HMM and PF can be used either a posteriori on an entire set of measurements (offline) or live estimating the location regarding the available set at that time (online). In offline estimation, the end location is used to detect larger deviations from the estimated path by considering the dependency of adjacent states. Unlikely locations regarding the entire estimated path are then discarded and locations with a less probability is used that fits better to the path. LMSE only estimates the optimal location for one measurement set and does not consider a dependency of adjacent states. Therefore, this approach is not feasible for LMSE.

Results

The test scenario of the presented work on this localization framework contained of 22 deployed APs on an 8000 m^3 indoor environment. In this environment, the progation model achieved an accuracy of $4dBm$ and the used PHOTON raytracer could be used with two different device classes, although the RSS behaviour was different.
The evaluation of the localization algorithms showed different accuracy results [25]. Using HMM and PF, an average error of less than $3m$ could be achieved with a deviation of up to $1m$ between offline and online estimation. With LMSE the average error was about $3.5m$. There were difficult parts of the environment taken into account of the measurements such as stairways were the radio propagation were significantly worse. Therefore, removing such parts from the mean error, the accuracy could be improved by about $0.5m$.

3.2 Other Network-Based Approaches

In the following, we discuss other approaches on network-based indoor localization of Wi-Fi devices. First, we present a work that uses APs to collect RSS measurements from the wireless traffic. The second approach uses APs to measure the Time-of-flight of Wi-Fi frames between the target device and the transmitter.

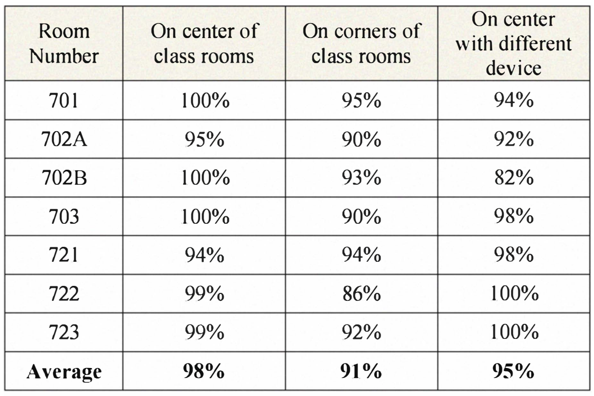
3.2.1 Access Points as Signal Strength Data Collectors

Kao et al. presented a network-based application for indoor localization using RSS values from Wi-Fi devices [21]. It uses APs to gather the RSS measurements from the target devices and sends the readings to a localization server that performs the localization process.

The architecture of this approach contains of three major components: (1) numerous APs serving as RSS data collectors, (2) a localization server processing on the RSS data of all APs, and (3) the target Wi-Fi client.

The APs collect the RSS data from the wireless network traffic and simultaneously handle the network communication. In the presented test environment, ASUS WL500 APs were used running a version of OpenWRT [7]. To collect the RSS data, a slightly modified version of the open-source monitoring tool Kismet is used which records the RSS data and client information to a database [5].
The localization server runs a MySQL database server to which the RSS data is stored. The pre-collected RSS data ranged from $-60dBm$ to $-88dBm$. RSS values at locations where no measurements had been received were stored with a value of $-90dBm$.
The localization server performs the actual localization process using fingerprinting. This comprises two phases: (1) a so called offline phase to collect RSS data of the pre-defined environmnet and create a RSS map, and (2) an online phase in which RSS data is gathered in real time and then compared to the pre-collected values. The location is finally determined as the minimum result of the LMSE method, i.e., the minimum result of the Euclidean distance function regarding the mean of RSS values.
Additionally in the presented test environment, the localization server runs a webserver providing the location-based content and a webpage which serves as subscription unit for the clients. Only clients that access this webpage will be tracked by the system.
The third part is the target client that will be located. There is no additional soft- or hardware necessary so that a client can be located by the system. As already mentioned, in the presented test environment a client is only been tracked after accessing a provided webpage. This is no technical restriction to the presented system as a client could also been tracked all the time while it is in range. In the example application, content such as files and information according to designated locations and times are then provided to the user via the webpage.

In the experiments, the accuracy has been evaluated at different locations in the test environment. In this approach, locations were based on rooms and the localization error was calculated as the percentage of correct determined rooms. Three APs were deployed on the same floor covering the entire area. The pre-collected RSS data was measured each time at the center of each room. The accuracy then was evaluated taking measurements at the center of each room, at the corners, at one floor above and again at the center of each room using a different mobile device than the pre-collected RSS data was recorded with. The achieved average accuracy ranged between 91% at the corners and 98% at the centers. Even at one floor above, the average accuracy was about 93%. Full results are shown in Figure 3.2.
The evaluation showed, that already with conventional tools that were slightly customized, a simple created RSS fingerprinting map and the minimum necessary amount of three deployed APs, an average error of less than 10% could be achieved.



Room Number	On center of class rooms	On corners of class rooms	On center with different device
701	100%	95%	94%
702A	95%	90%	92%
702B	100%	93%	82%
703	100%	90%	98%
721	94%	94%	98%
722	99%	86%	100%
723	99%	92%	100%
Average	**98%**	**91%**	**95%**

Figure 3.2 The average accuracy evaluated in the experimental test environment [21].

3.2.2 Positioning using the Time-of-flight

In this approach, Ramirez et al. presented a network-based solution using the ToF of Wi-Fi signals [17]. This work targets a simple deployment by a software-based solution which can be installed on off-the-shelf APs through a firmware upgrade. The presented work uses APs to send Wi-Fi frames to the target device and receive the corresponding acknowledgement (ACK) frames. From this, the round-trip-time (RTT) is measured which is simplified to twice of the ToF. Three different methods were implemented for obtaining the required timestamps.

The first method uses the timing synchronization function (TSF) specified in the IEEE 802.11 standard. TSF is used to synchronize the timers of all participants in an IEEE 802.11 network and is based on a $1MHz$ clock, so a resolution of 1μs is offered. The main limitation to this method is, that only received frames, i.e., the ACK frames, contain timestamp information from the TSF. To obtain a timestamp from transmitted frames, an additional Wi-Fi device is used. This device receives the transmitted frames as well as the ACK frames and is set close to the transmitting AP to minimize the difference in the ToF according to the distance between these devices. The additional device then calculates the distance by using the time when the transmitted frame was received subtracted from the time the ACK frame was received, resulting in the total time (T_{total}). Further the processing time ($T_{process}$) which is the delay the target device needs between receiving the transmitted frame and sending the ACK. The formula

$$d = \frac{c \cdot (T_{total} - T_{process})}{2}$$

where d holds the distance is used to calculate the distance. Details on the contained parameter c were not described. Furthermore, a gauss filter is used to increase the accuracy of the distance estimation [17].

The second method is based on the CPU clock and thus can offer a higher resolution than the TSF. In this implementation, the necessary timestamps are obtained by the hardware driver of the WNIC. Each time when a frame is sent as well as received by the WNIC, a hardware interrupt is raised which are processed by the corresponding functions of the driver. The driver software was modified so that in these functions all necessary information such as the timestamp are collected. To minimze delays according to the processing time of sending respectively receiving the frames, the timestamps are taken by the driver as soon as possible. The distance is then estimated with the same formula as in the first method. This solution requires access to the driver software but needs no time synchronization as both timestamp are obtained from the same clock.

The third method uses again the TSF but eleminates the need of an additional device to obtain the timestamp from transmitted frames. As in the second method, the driver software is modified to call a function which obtains the card clock time for a transmitted frame. When the ACK is received, the timestamp can be obtained from the frame. This method also requires no time synchronization as both timestamps rely on the same clock. For the distance estimation again the formula from the first method is used.

The presented work takes no further look at the actual calculation process of the location from the estimated distances and only mentions that algorithms such as mutlilateration can be used. Further research were done on the type and amount of send frames as the computation load of WNIC and CPU can affect process of obtaining the timestamp. The results to these tasks can be looked up in [17]. Additionally, a filter is used to prevent from outliers in implausible distance values.

The evaluation of the three methods showed all a mean error within few meters. All three methods were implemented and simultaneously evaluated. The second method showed the best performance with an average error of $1.5m$. The second and third method showed some important outliers and the average error was 2.8 respectively $4.4m$. According to the fact that all three methods were run simultaneously, those outliers were lead back to deviations in the clock.

For the approach of building an easy to deploy system that can be integrated in an existing Wi-Fi infrastructure, the first method requiring additional devices is not feasible. The second and third method require access to the driver software which depends on the device whether this is feasible in an already present infrastructure.

3.2.3 Similarities and Differences to our Work

Kao et al. defined the localization process as determining the room in which the client is located in. In our work, we determine the clients location regarding a coordinate system. This leads to a more subtle localization and offers also the possibility to determine the corresponding room by mapping the coordinates to rooms.
Furthermore, with LMSE only a single deterministic method for the location estimation had been used. The localization framework that we use in our approach offers with HMM and PF two probabilistic methods to estimate the position and additionally the LMSE method. Moreover, the HMM and PF methods are available as offline and online versions, thus in our work, we compare the results of five localization methods.

In the work of Ramirez et al., the shown accuracies of all three methods are comparable to the results shown by the localization framework in Section 3.1 that we use in this work. The main advantage of this approach using the ToF is the robustness against environmental effects in contrary to the effects when using the RSS.

Instead it is influenced by deviations in the used clocks and the load of the CPU what restricts the scalability of such a solution. Furthermore, this requires the client and the entity that collects the measurements to be in the same network. In our approach, we do not require the client to be associated at all.

4 Design

This chapter describes the architecture of our network-based localization system and gives an overview of defined requirements of each component. Further, we describe an approach on increasing the available information about a target device. Finally, we discuss different approaches on counter measures against our system.

4.1 Scenario

There exists no approach on indoor localization yet that provides a simple to deploy and cost efficient solution with high accuracy. Therefore, in this work, we challenge the task to build an easy to adapt and efficient localization system of Wi-Fi devices. Nowadays, Wi-Fi is a widely used technology in wireless local area networks (WLAN) and ad-hoc networks. Standard devices such as notebooks or smartphones support Wi-Fi communication and according to this, Wi-Fi infrastructure is also commonly available in indoor environments. Due to the goal to bulid a simple to deploy system, we decided to design a pure software-based system that only relies on a Wi-Fi infrastructure and thus can be installed through a software upgrade on standard hardware such as APs. In order to only use a software solution, this also reduces the costs as no additional hardware is required. Further advantage of using Wi-Fi is, that all necessary information about the client device that will be located is already available through the Wi-Fi respectively IEEE 802.11 protocol, e.g., WNICs already measure the RSS values.
As the entire localization process, including data collection and the actual location estimation, is done in a Wi-Fi infrastructure, this leaves no tasks at the target device. This also reduces the implementation effort as no changes have to be made on that device. Furthermore, no active participation of the target device in the localization process is required, thus the location of a Wi-Fi device can be determined without recognition. This introduces new possible applications, e.g., surveillance or inventory monitoring, but also leads to privacy aspects as the location of private devices that support Wi-Fi such as smartphones directly relate to the location of

people. Therefore, we further discuss possible counter measures against our system. Moreover, our system relies on the emitted wireless network traffic of the target device to collect RSS measurements. Hence, if the target device emits no traffic, we are not able to estimate its location. Due to this, we also investigate a method to induce a Wi-Fi device to emit wireless traffic.

4.2 Architecture

The system architecture consists of three major components: (1) several sniffer which serve as data collectors, (2) one server to collect the measurements from the different sniffer, and (3) the localization framework described in Section 3.1 that performs the actual localization process. Figure 4.1 shows an illustration of the architecture.

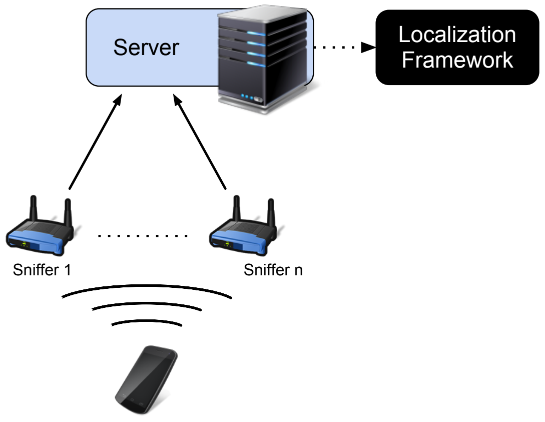


Figure 4.1 The figure shows an overview of the system architecture containing the sniffer, server and the localization framework.

At least three sniffer placed at different locations are necessary to enable the localization process. The server can be implemented on any device, but has to be a central entity due to the design of the communication process with the localization framework. Additionally, a central entity can simply be implemented and suffices the tasks of the server.
All sniffer have to be connected to the server and the server to the localization framework to be able to deliver the data to the successive entity. These connections can be realized in any form, e.g., using a wired local area network, an ad-hoc network or even Wi-Fi. The only restriction is not to use the same or an overlapping Wi-Fi channel as the target device uses for communication. We define this constraint to prevent from side effects on the RSS measurements through channel overhearing. In addition, the connection between the sniffer and the server have to enable time synchronization.

Time Synchronization

The sniffer and server have to be time synchronized so that time related measurements from different sniffers can be merged. Furthermore, some statistical models rely on a chronological order such as the hidden markov model (HMM) or particle filter (PF) as described in Section 3.1. The localization framework is not required to be time synchronized.

4.3 Sniffer

The tasks of the sniffer are to extract RSS measurements from the network traffic emitted by the target Wi-Fi device and forward these measurements to the server. This defines several requirements.
First of all, the sniffer has to be able to capture arbitrary wireless network traffic and receive Wi-Fi frames from the WNIC. For this, it is necessary that a wireless network interface can be set to monitor mode as this is required to capture frames. From the received frames, the sniffer has to be able to extract the RSS value and assign this measurement unambiguously to a Wi-Fi device.
Furthermore, there exist different Wi-Fi channels which a target device can use for communication. As described in Section 2.1, adjacent channels overlap which leads to channel overhearing and results in corrupt RSS measurements when a frame is captured through channel overhearing. Therefore, the sniffer first has to determine the right channel and set the used network interface to use this channel, so that the possible optimum in the RSS measurement is reached.

As there are possibly multiple Wi-Fi devices present in the environment, the sniffer has to be able to filter out the frames corresponding to the target device that has to be located. In general, this is no restriction to our system and location estimation of multiple devices simultaneously is possible as well. In order to be able to focus on a specific target device, we define this requirement.

4.4 Server

The server functions as sink for the collected data from all sniffers. Therefore, the server has to be able to receive the data from the sniffers. Moreover, it is responsible for preprocessing the collected data to the format the localization framework requires. Hence, the localization framework that we use in this work defines how the server processes on the collected data.
The localization framework provides two different modes to estimate the location from the measurements: (1) an online mode in which measurements are passed directly to the framework enabling live-tracking of a device, and (2) an offline mode in which a sequence of measurements are processed a posteriori allowing a higher accuracy in the estimated locations. In order to enable live-tracking as well as an a posteriori estimation for higher accuracy, we impose the server to provide both ways.
For (1), this requires the server to directly merge the collected data grouping by the target device and the sniffer which received the data. This imposes the server to be able to process on the received data and simultaneously gather further measurements from the sniffers. Additionally, it has to pass the merged data to the localization framework in the required format as well without influencing the other processes.
For method (2), it is necessary to store the data to some kind of data storage and

provide functionality to merge the data to the required format. Furthermore, this is also necessary to be able to repeat the localization process on the identical measurement data.

4.5 Frame Injection

Our system relies only on the emitted wireless network traffic of the target device to obtain the necessary information and be able to estimate its location. Thus, when the target device does not emit any network traffic, our system is not able to determine its location.
Different tests have shown, that the amount of emitted network traffic drastically varies between devices. Some devices periodically transmit Wi-Fi frames due to background processes or network scans. Others only emit traffic when a user performs tasks that use Wi-Fi. This also depends on the mode a client WNIC is set to. Due to this, we will also evaluate further tests on the available amount of traffic.
In order to induce the target device to emit Wi-Fi traffic, we manually send different Wi-Fi frames addressed to the target device and investigate the resulting behavior. Furthermore, we will test the transmission of ICMP echo request packets [15] to force the target device to send ICMP echo reply packets. For this, the target device has to implement the Internet Control Message Protocol (ICMP) which is a transport protocol based on IP. This is usually the case on common Wi-Fi devices. Additionally, this requires the target device to be associated to a Wi-Fi network as the ICMP echo request packets are transmitted via IP.
If the target device reacts to a manually transmitted frame, this enables the location estimation even when no traffic is available.

4.6 Counter Measures

A network-based localization system offers the possibility to locate and track a Wi-Fi device without any recognition at the target. In the case of private devices such as smartphones or tablets where the location of the device relates to a persons location, our system invades the persons privacy. To give a person the possibility to prevent from being located, we investigate different possible counter measures against our system. This covers (1) reducing the signal strength of transmission, (2) vary the used signal strength in a designated range, (3) switching the used Wi-Fi channel, and (4) switching the MAC-address of the used WNIC.
Due to the limited scope of this work, we only discuss these methods and will not implement them.

Reducing the Signal Strength

In a Wi-Fi network, a device only needs one AP in range for communication. As described in Section 2.3, the process of localization requires at least three anchor points with information about the target to be able to determine the location. Therefore, the signal strength the target device uses for transmission could be dynamically reduced to the level that is required so that only the nearest AP receives the emitted traffic. This still enables communicating in the Wi-Fi network but eliminates the possibility of localization as only one AP can receive the transmitted frames. This method assumes, that the sniffer are implemented on the APs of the Wi-Fi network. Otherwise, it cannot be ensured that only one sniffer receives the emitted traffic.

Varying the Signal Strength

Another approach on modifying the signal strength is to vary the used signal strength level for transmission randomly in a designated range. The minimum level still has to enable communicating in the Wi-Fi network. This would consequently lead to randomly varying RSS measurements at the sniffer and hence, to varying results in the estimated locations. This method does not prevent from localization but could decrease the accuracy of the estimated locations and thus, disguises the real position.

Wi-Fi Channel Switching

This method takes advantage of the different available Wi-Fi channels. When communicating in a Wi-Fi network, the used channel for transmission is stated by the AP, thus a device can not freely switch between all existing channels. But due to the fact of overlapping channels, a client could randomly switch between the stated channel and the overlapping ones. This leads to channel overhearing as described in Section 2.1, but still enables communication. The channel overhearing then leads to varying RSS measurements and again to varying results in the estimated locations. This method also only decreases the localization accuracy and does not prevent from localization in general.

MAC-Address Switching

MAC-addresses are used to uniquely identify the target device. If the MAC-address changes, our system considers the new address to belong to a different device, hence all previously collected data and estimated locations can not be linked to this new address. The information our system can link to a device then depends on the frequency of the MAC-address switching. The shorter the interval between two switches is, the less information can be linked to that device. This method could lead to temporal failures in the Wi-Fi communication, because the Wi-Fi protocol relies on the MAC-address as well to assign received frames. This could be solved at a client by storing old MAC-addresses so that frames addressed to those MAC-addresses can still be received, but would lead to problems at the AP.

5 Implementation

The following chapter describes the details of our implementaion. We give general information of the development and used third-party software. Then we take a look at the implementation of each component. Furthermore, we discuss solutions for additional tasks, that came up during the design phase.

5.1 Development

We decided to use the high-level language Python to implement our system. Python is in general an interpreted scripting language with available interpreters on all common operating systems, but can also be compiled into standalone executable programs using third-party tools. It offers the use of different paradigms such as object-oriented, procedural, imperative or functional programming. It uses duck typing thus it is dynamically typed, but also uses strong typing in several places. Python has a module architecture and offers a large amount of libraries. Also interaction with C libraries can be realized and it has several build in data structures such as dictionaries and lists. Hence, Python offers a faster development and deployment to the different used hardware than other languages such as C. A Python script is in general slower than a compiled C program, but this was no limitation to our system.

5.1.1 OpenWRT

OpenWRT is a project providing a linux based distribution for embedded devices such as APs [7]. It is an open-source project and licensed under the GPL. OpenWRT has its origin in the Linksys GPL sources for the WRT54G wireless router and is meant to provide an easy to modify, fully accessible and featured operating system for routers. Software packages can be installed via the opkg package management system or added before compilation. There exist several forks or similar projects such as FreeWRT [3] or DD-WRT [2].

For our work, OpenWRT offered a fully configurable system which was required to access monitor mode and use a Python interpreter to develop on an AP. We installed the version Backfire 10.03.1 on Linksys WRT160NL wireless routers with additional packages such as a Python interpreter and several libraries, which are described in the following.

5.1.2 Third-Party Software

We made recourse to several open-source libraries. The most important were the C library Libpcap, which we described in Section 2.2, and the Python module ctypes that we used to build a Python wrapper of Libpcap, on which we go into in Section 5.1.4. Twisted offered simple methods to make the module asynchronous. Other used libraries such as NumPy and SimpleJson are used for convenience, but the used functions could also be reimplemented if necessary.

Ctypes

Ctypes is a functional library that provides C compatible data types and allows calling C libraries [9]. We used this library to build a pure Python wrapper for the Libpcap, which we describe in Section 5.1.3.

Twisted

Twisted is a powerful module for event-driven network programming published under the open-source MIT license [13]. It provides functionalities for asynchronous programming, supports several network protocols such as HTTP or UDP, and also offers off-the-shelf classes for networking, e.g., a simple webserver that were used for debugging.

NumPy / SimpleJson

NumPy is a scientific Python module that provides support for more sophisticated mathematical functions such as multi-dimensional matrices [6]. We used the function `median()` to calculate the median from a list of RSS measurements.
SimpleJson is a simple and fast JSON encoding and decoding module [19].

5.2 Sniffer

The sniffer extracts the RSS values from the Wi-Fi traffic of the target device and forwards the data to the server. Due to this, it has three responsibilities: (1) determining the Wi-Fi channel to listen on, (2) capturing the Wi-Fi frames and extract the data, and (3) forwarding the data to the server.
First of all, a wireless network interface in monitor mode is required to enable frame

capturing, as we described in Section 2.2.1. If no interface in monitor mode is available, the sniffer prevents from starting. Then, a filter is set to only receive frames with the MAC-address of the target device as its source. After that, the capturing process is started using our implemented wrapper of the Libpcap, called PyCap.

PyCap

To only use pure Python code in the implementation of our system, we have build a wrapper for the C library Libpcap using ctypes. It is devided into two major parts: first the `functions.py` that loads the Libpcap and creates Python functions which have to be defined in their argument and return data types. The second part is the `types.py` which defines several Python classes inheriting the `ctypes.Structure` according to their implementation as a C struct in the Libpcap.
PyCap is implemented as a package and can easily be accessed with the standard `import` key word.

Frame Capturing

There are two different modes available in Libpcap to capture frames: the *loop* and the *dispatch* mode. Both modes execute a callback function for each frame that is captured to process on it. The important difference is, that the loop mode blocks when in an iteration no frame is available. It then waits until a frame is captured. Thus, during this waiting phase, the sniffer blocks and no other statement can be executed.
The dispatch mode does not wait but processes the frames that are available. Hence, when there is no frame available, the dispatch mode terminates and other statements can be executed. We decided to use the dispatch mode as the non-blocking becomes important for the channel determination, which we explain later in this section.

Libpcap passes the frame raw and unformatted to the callback function with an additional pcap header, see Listing 5.1.

```c
struct pcap_pkthdr {
    struct timeval    ts;     /* time stamp */
    bpf_u_int32       caplen; /* captured length */
    bpf_u_int32       len;    /* actual length */
};
```

Listing 5.1. The listing shows the pcap header format as C struct.

The pcap header holds information about the time the frame has been received, the amount of bytes that have been captured and the actual length of the frame. To refer the data later to a timestamp, we use the time from this header.
The raw frame contains of two major parts: (1) the radiotap header as explained in Section 2.2.2, and (2) the IEEE 802.11 frame. We only use the data field for the radio frequency signal power at the antenna, but due to the variable length of the radiotap structure, we have to check in the bitmask which data fields are present. Attention should be paid that the bitmask is stored in network-byte-order which is defined as big-endian. Table 5.1 shows a section of the data fields that have to be

checked for their presence, including their assigned bit number and the size in byte. A complete list can be looked up from [10].

Bit Number	Field	Size in Bytes
0	TSFT	8
1	Flags	1
2	Rate	1
3	Channel	4
4	FHSS	2
5	Antenna Signal	1

Table 5.1 Overview of the data fields that have to be checked for their presence.

To calculate the offset of the antenna signal data field, the sizes of the present data fields before the antenna signal field have to be summed up, plus 8 byte for the radiotap header. The resulting offset can then be used to extract the RSS value.
If the traffic of more than one device is captured, the source MAC-address has to be extracted as well. The field *address 2* of the IEEE 802.11 frame contains the source address, see Section 2.1.3 for further description. To access the IEEE 802.11 frame, the *it_len* field of the radiotap header can be used which holds the entire length of the radiotap data. The IEEE 802.11 frame directly follows the radiotap data, thus the offset equals the length of the radiotap data.
Other information such as the radiotap data field for the channel have been used for debugging, but are not relevant in this work.

After the necessary information have been collected, the sniffer sends the timestamp, measured RSS value and the corresponding MAC-address to the server using UDP for transmission. The data is encoded as a JSON string due to simplification. We decided to use UDP for communication due to its usage of the internet protocol (IP), thus the sniffer and server are not required to be located in the same network and can communicate via a local area network (LAN) as well as over the internet. Furthermore, UDP defines a minimum of protocol mechanism and is natively supported in various programming languages, hence can easily be implemented.

Channel Determination

The channel of the target device has to be determined to prevent from channel overhearing which would lead to corrupt RSS values as described in Section 2.1. For this, the sniffer switches once to each available channel after a certain timeout. In between, it stores the measured RSS values according to the used channel. The channel is then determined by comparing the collected RSS values. Due to attenuation, tests have shown that the RSS values of frames that have been received through channel overhearing are between 5 and $15dBm$ lower. Thus, the channel with the highest RSS value is determined as the channel the target device is transmitting on. This process is only done once the first time the sniffer is started and does not consider channel switching at the target device. To take this into account, a more sophisticated method has to be applied.

5.3 Server

The server functions as sink for the measurements from the different sniffer entities. It has three major tasks: (1) merging the different measurements according to their timestamp and relating MAC-address, (2) logging the data to a data storage, and (3) forwarding the merged data to the localization framework.
The server is implemented as a single script based on the event-based networking framework Twisted. To be able to simultaneously receive data from the sniffer and process on them, all tasks are implemented asynchronously using the reactor pattern [1][26] implemented as the class `reactor` in Twisted.
First, the reactor maintains a UDP socket for receiving the data from the sniffer. The reactor then listens to a UDP socket on a designated port and delegates incoming packets to our class `UDPLineReceived()` which inherits the `DatagramProtocol` of Twisted. This takes care of the IP header and only passes the data of the packet to a method in our `UDPLineReceived()` class. The data - formatted as a JSON string - is decoded and stored to a temporal buffer.
Furthermore, the reactor runs a `LoopingCall` task which executes the method `processBuffer()` periodically after a designated timeout. This method processes the buffer and merges the different measurements, grouping the data (1) by the corresponding target MAC-address and (2) by the sniffer which has recorded that measurement. If more than one measurement from one sniffer is available, the median is taken from the RSS values to reduce the impact of outliers. Each execution of this method represents a localization step. We decided to use a timeout of $200ms$, thus 5 localization steps per second, which seemed to suffice for accurate live localization considering the movement speed of a human. After each execution of the `processBuffer()` function, the buffer is cleared and the merged data is passed to the localization framework.

Integration of the Localization Framework

The localization framework is an independent and separate entity. It implements a webserver and provides an interface for receiving HTTP post packets. At the end of the `processBuffer()` function, another function is called to send the merged data to the server. This is also implemented asynchronous to prevent from blocking of the LoopingCall due to delays in the HTTP respectively TCP transmission.
The data of the packet has to be formatted as JSON string, illustrated as an example string in Listing 5.2.

```
[
  {'rssi': -52, 'ts': 1368028727.763011, 'ssid': 'platoon',
    'bssid': '00:22:15:21:b0:c2'},
  {'rssi': -81, 'ts': 1368028727.763011, 'ssid': 'platoon',
    'bssid': '00:22:15:24:25:97'}
]
```

Listing 5.2. This is an example data JSON string containing measurements from two different sniffer corresponding to the device called *platoon*.

[1]A design pattern for event-based handling of service requests.

The JSON string contains an array with an object for each sniffer that received RSS measurements. Each object has to comprise four key-value pairs:

rssi: the RSS measurement value as a number

ts: the corresponding timestamp, also as a number

bssid: a string of the MAC-address of the AP the RSS has been measured with

ssid: a string holding the name which the localization framework uses to generate the output

The localization framework then generates a webpage with graphical output of the localization results and further debugging values using the committed ssid value.

Logging

In order to be able to use the identical measurement values to repeat the localization process, the server stores all incoming data from the sniffer as well as all outgoing data to the localization framework in a database. We decided to use *sqlite3* due to its simple structure [11]. It is available on all common operating systems and requires no further administration software. A database contains of one single formatted file including all data and structures. Especially, the required storage capacity is lower compared to other database management systems, thus it can be implemented on devices with low available storage capacity, e.g., on an AP.

5.4 Frame Injection

To induce a device to emit Wi-Fi traffic, we send different artificially created IEEE 802.11 frames addressed to the target and analyse if the device responds to them. The Libpcap provides the function `pcap_inject()` to manually emit network traffic. Similar to the frame capturing, this function requires a network interface in monitor mode. A byte buffer containing the designated raw frame is passed to the `pcap_inject()` function. Furthermore, it is necessary to pass a valid radiotap header in addition to the actual frame to the inject function. Otherwise, the NIC rejects the transmission of the frame. Listing 5.3 shows an example.

```
static const unsigned char buffer[] = {
  /* start of radiotap header */
  0x00                                        // version
  0x00,                                       // padding
  0x19,0x00,                                  // header length
  0x6f,0x08,0x00,0x00,                        // present flags
  0x04,0xbb,0xc1,0x00,0x00,0x00,0x00,0x00,    // timestamp
  0x12,                                       // flags
  0x18,                                       // data rate: 12.0 Mb/s
  0x6c,0x09,                                  // frequency: 2412
  0x80,0x04,                                  // type: 802.11g
  0xc5,                                       // ssi signal: -59 dBm
```

```
    0xa5,                                       // ssi noise: -91 dBm
    0x00,                                       // antenna: 0
    /* start of RTS frame */
    0xb4,0x00,                                  // fc
    0xd9,0xd9,                                  // duration
    0x55,0x44,0x33,0x22,0x11,0x00,              // ra
    0x00,0x11,0x22,0x33,0x44,0x55,              // ta
    0x0e,0xa5,0x4e,0xef                         // fcs
};
```

Listing 5.3. An example of a radiotap header followed by an IEEE 802.11 RTS control frame as a C byte buffer.

The first 25 byte define the radiotap header. The last 24 byte represent the RTS frame. We implemented the transmission of IEEE 802.11 RTS and data frames with empty payload using the inject function.
For the transmission of ICMP echo request packets, we used the tool *ping* [8] which is a common tool on UNIX operating systems.

6

Evaluation

This chapter presents the evaluation of our work. First, we will show measurements of the available wireless traffic amount of standard Wi-Fi devices in different operating modes. We further will investigate approaches on increasing the wireless traffic amount. Finally, the evaluation of our implemented network-based localization system will be presented.

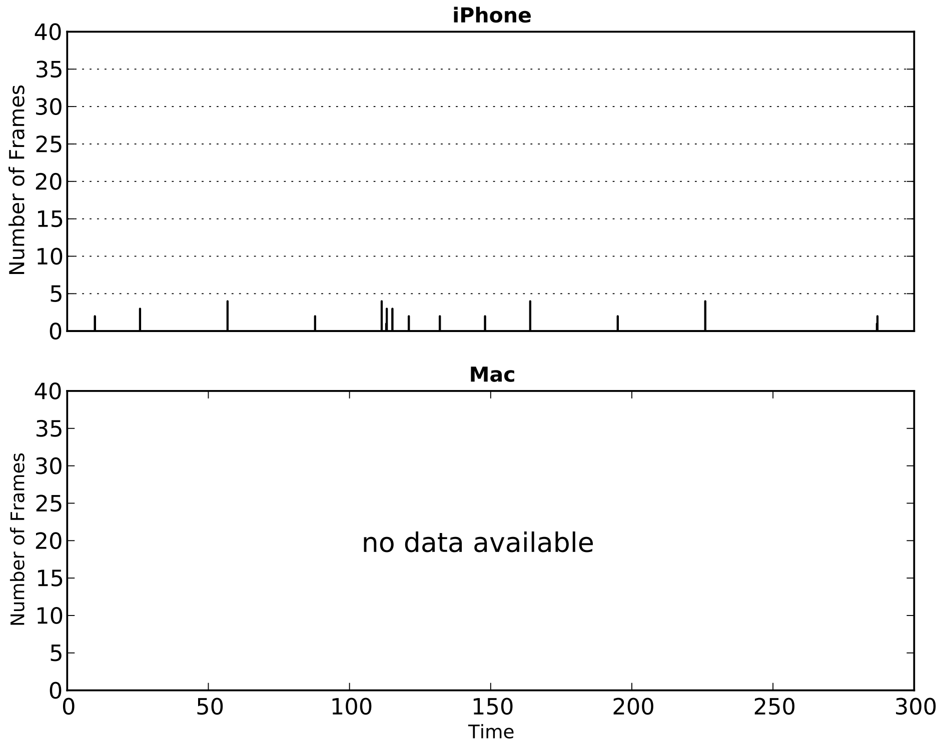
6.1 Traffic Amount

Our implemented system only relies on the emitted wireless traffic of the target Wi-Fi device. Thus, we are not able to determine the location of a device if no wireless traffic is available. Due to this, we evaluated the amount of traffic a standard device emits.
There are in general three different modes a Wi-Fi device can operate in: (1) the unassociated mode where a device is not connected to any network, (2) the associated mode where a device is connected to an AP providing a Wi-Fi network, and (3) the ad-hoc mode in which all devices participating in the network directly communicate with each other. We evaluated the amount of traffic in all three modes. For this, we compared the results of two standard Wi-Fi devices with each other: an Apple iPhone 4s running iOS in version 6.1.3, and an Apple MacBook Air running OS X in version 10.8.3. Both devices set to the appropriate mode running no additional process except standard background tasks of the operating system, due to the fact that our system imposes no tasks or active participation on the target device. The traffic amount of both devices has been measured simultaneously over a period of 5 minutes in each mode. For this, we used a single sniffer running on a Linksys WRT160NL wireless router placed next to both devices. The server that serves as sink for the measurements of the sniffers has been running with an iteration timeout of 200 ms, thus it processed 1500 iterations during the entire measurement period. Each iteration represents one location estimation, hence this configuration allowed

up to 5 localizations per second and a maximum of 1500 localizations during the 5 minute measurement period.

Unassociated

In the unassociated mode, a device has no connection to a network or device, thus no data transmission is expected. Only IEEE 802.11 management probe request frames are transmitted to actively seek for available Wi-Fi networks [18]. Figure 6.1 shows the traffic amount of both devices.



iPhone	**Max. time interval:**	60.8s	**Avg. time interval:**	18.5s
	Min. time interval:	0.2s	**Frames / Iterations:**	41 / 16
MacBook	**Max. time interval:**	n/a	**Avg. time interval:**	n/a
	Min. time interval:	n/a	**Frames / Iterations:**	n/a

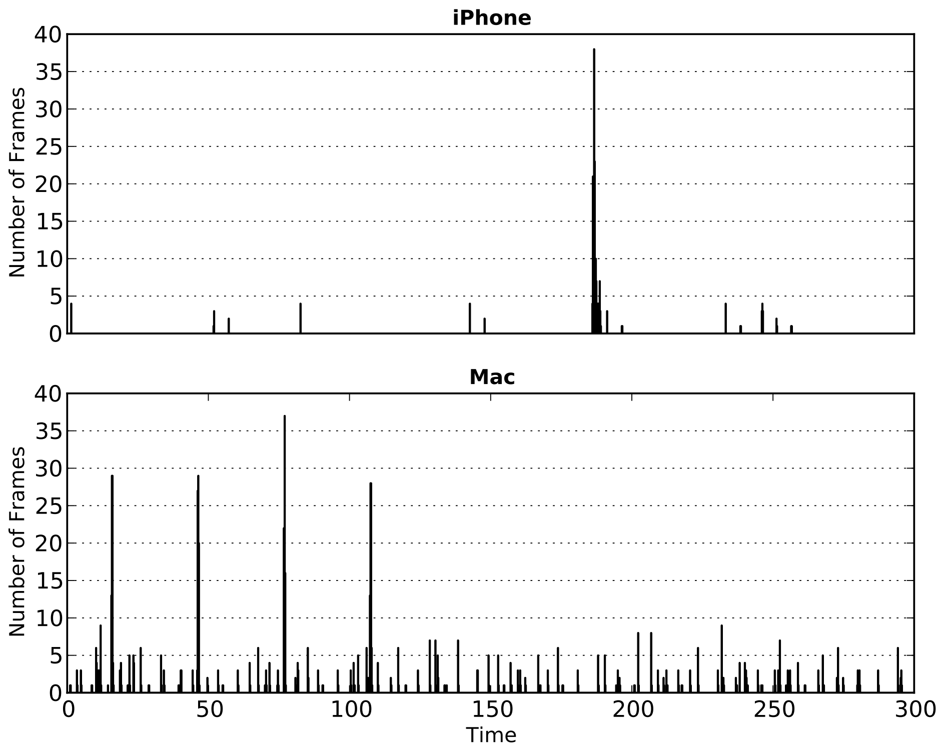
Figure 6.1 Graphical and numerical demonstration of the traffic amount while both devices were in the unassociated mode. The MacBook did not transmit any frames at all.

The iPhone emitted in the mean around every 18.5 seconds Wi-Fi traffic. The maximum time interval between two transmissions was 60.8 seconds and the minimum was 0.2 seconds, thus there were frames available in two consecutive iterations of the server. Overall, the iPhone transmitted 41 Wi-Fi frames that matched in 16 iterations of the server during the measurement period of 5 minutes. These transmissions arise from the background process in iOS that automatically seeks for available Wi-Fi networks. Refering to these results, our system would be able to estimate 16 locations of the iPhone with a mean time interval of 60.8 seconds in between.

The MacBook did not emit any traffic at all, thus there were no measurements available. This is because the operating system OS X does not automatically seek for available Wi-Fi networks. Hence, as this is the only traffic that is emitted in the unassociated mode, the MacBook transmits no Wi-Fi frames without being instructed and our system is not able to estimate the location of the MacBook at any time.

Associated

In the associated mode, a Wi-Fi device is connected to a wireless network through an AP that maintains the network. In contrary to the unassociated mode, the devices not only transmit probe request frames when associated to a network but use that connection for data communication. To evaluate the traffic amount in this mode, we established a Wi-Fi network using an additional Linksys WRT160NL router and both devices had been connected to this network. Figure 6.2 shows the captured traffic amount.



iPhone	**Max. time interval:**	60.0s	**Avg. time interval:**	7.3s
	Min. time interval:	0.2s	**Frames / Iterations:**	199 / 36
MacBook	**Max. time interval:**	7.0s	**Avg. time interval:**	1.3s
	Min. time interval:	0.2s	**Frames / Iterations:**	812 / 237

Figure 6.2 Captured traffic amount of the iPhone and MacBook while both were associated to a network. The results are presented graphically as well as numerically.

Associated to a network, the iPhone transmitted averagely every 7.3 seconds Wi-Fi frames with a maximum interval of 60.0 seconds in between. Overall, it transmitted

199 frames during the capturing period divided into 36 iterations of the server, which allows us to estimate 36 locations every 7.3 seconds in the mean. Regarding the minimum interval of 0.2 seconds, some estimated locations are directly adjacent. The higher traffic amount in contrary to the unassociated mode comes from different background processes running in the background of the operating system that use the connection to communicate data information. The peak at second 187 of 38 frames came from a higher communication effort in those tasks.

The MacBook also uses its connection to the associated network for data communication in running background tasks. This happened averagely every 1.3 seconds using overall 812 frames that matched 237 different iterations, thus our system was able to determine 237 locations with a maximum time interval of 7.0 seconds. There are several peaks in the recorded measurements in which the MacBook transmitted up to 37 frames in a single server iteration, due to again temporal high communication effort at the corresponding processes.

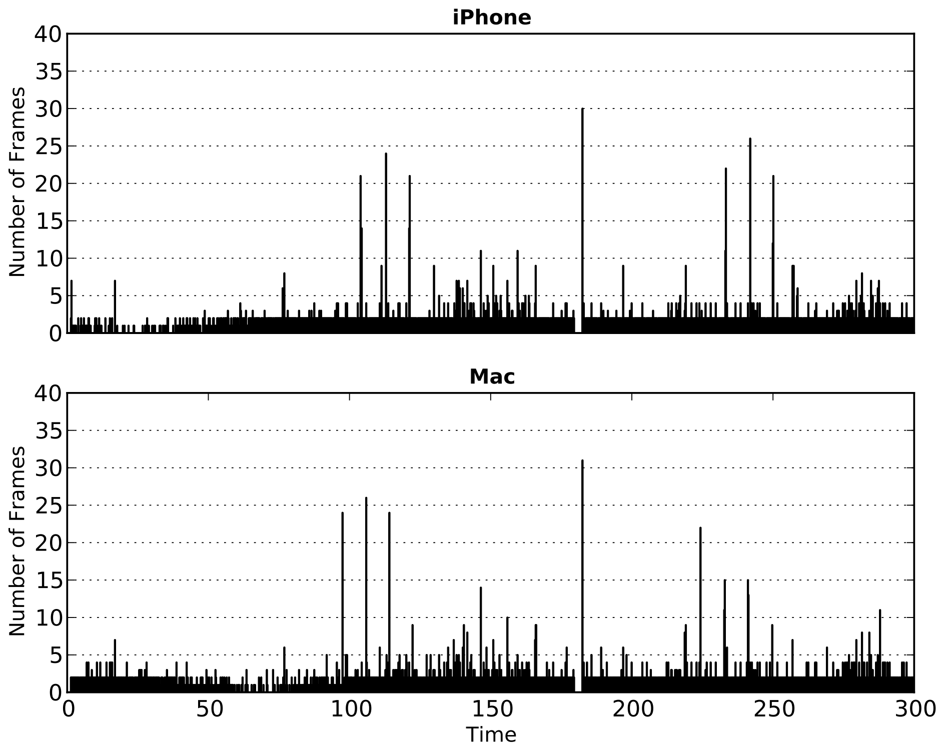
Ad-hoc

In contrary to the associated mode, in an ad-hoc network all participating devices are directly connected to each other, called peer-to-peer. Hence, there is no central entity such as an AP that maintains the network, but all devices maintain their connections to the other devices. This constantly causes wireless traffic as a device has to send Wi-Fi frames to manage its connections. Figure 6.3 shows the measurement results of the traffic amount while the iPhone and the MacBook were connected to the same ad-hoc network.

In the ad-hoc mode, both devices constantly emitted network traffic with a mean time interval of 0.3 seconds. Overall, the iPhone send 2908 frames in 1238 server iterations overall and the MacBook 2974 matching 1262 iterations. Hence, regarding the total number of 1500 server iterations during the measurement period, there were frames available in more than 80 percent of all iterations. For both devices, the maximum time period in which no frames were available was 3.0 seconds beginning at second 180. Since this period occured for both devices, this leads us to the assumption that this is dued to a dropout at the sniffer. Nevertheless, this has no impact on the measurement quality.

6.1.1 Summary

The measurement results show, that the traffic amount highly depends on the used Wi-Fi mode and varies between different devices.
While the iPhone emitted few traffic when unassociated due to automatic background scans for available wireless networks, the MacBook showed no activity at all. Hence, when the target device is unassociated to any Wi-Fi network, the ability of our system to estimate its location relies on the performed network scans of the device.
When a device is associated to a Wi-Fi network, both devices communicate data information via the network due to background tasks of the operating system. Therefore, in this mode our ability to locate a target device depends on running applications on that device. In contrary to the iPhone, the MacBook showed a much higher



iPhone	**Max. time interval:**	3.0s	**Avg. time interval:**	0.3s
	Min. time interval:	0.2s	**Frames / Iterations:**	2908 / 1238
MacBook	**Max. time interval:**	3.0s	**Avg. time interval:**	0.3s
	Min. time interval:	0.2s	**Frames / Iterations:**	2974 / 1262

Figure 6.3 A graphical and numerical representation of the traffic amount while both devices were connected to the same ad-hoc network.

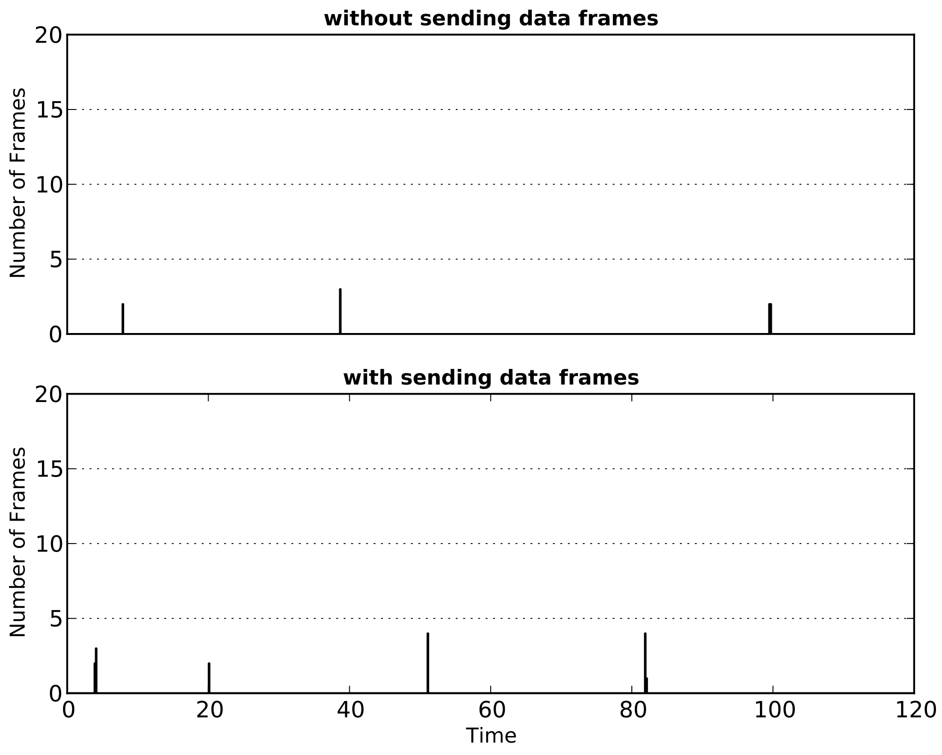
amount of traffic which enables a higher resolution of the estimated locations.
In ad-hoc mode, devices have to constantly maintain their connections to other devices. For this, devices permanently emit wireless network traffic, thus we are able to locate a device in ad-hoc mode independently of running applications or background tasks.

6.2 Frame Injection

The evaluation results in Section 6.1 showed, that the wireless traffic amount emitted by a Wi-Fi device highly depends on running applications and background tasks on that device. Furthermore, there were large differences between the modes. In the unassociated mode, the only traffic came from the scans seeking for available Wi-Fi networks. When associated to a network, the devices also used the connection to communicate data dued to background applications of the operating system. In order to reduce the dependency of running background tasks of our system, we evaluate the behaviour of a Wi-Fi device regarding artificially transmitted Wi-Fi frames addressed to the target device. As the devices showed constant high traffic in ad-hoc mode, we only test the behaviour in unassociated and associated modes. In all tests, we used the same Apple iPhone 4s as in the traffic amount evaluation, running iOS in version 6.1.3. We placed the measuring sniffer running on a Linksys WRT160NL router next to the iPhone. Further, we set the iPhone online and idle, meaning no additional tasks were performed except the processes corresponding to the operating system. All measurements were performed over a period of 2 minutes.

Unassociated

While the iPhone was unassociated, it occasionally send IEEE 802.11 probe request frames to seek for available Wi-Fi networks. We tested the devices behaviour on two different artificially transmitted frame types, addressed to the iPhone.
First, we send data frames with an empty payload. The frames were send directly successive with no additional timeouts between two transmissions. The source address in the data frame header was set to `00:11:22:33:44:55`. Figure 6.4 shows the recorded traffic amount during the test period.

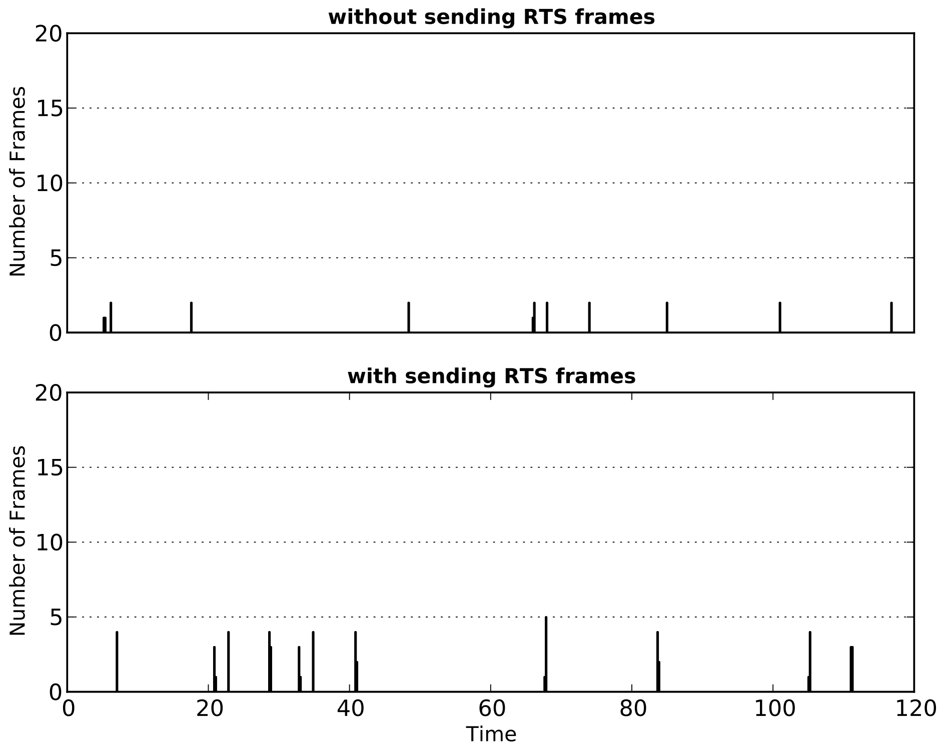


without data frames	**Max. time interval:**	60.8s	**Avg. time interval:**	30.6s
	Min. time interval:	0.2s	**Frames / Iterations:**	9 / 4
with data frames	**Max. time interval:**	31.0s	**Avg. time interval:**	15.6s
	Min. time interval:	0.2s	**Frames / Iterations:**	16 / 6

Figure 6.4 The results of the traffic amount measurements with and without artificially transmitting IEEE 802.11 data frames to the target device while it was unassociated to any network. The values are presented graphically and numerically.

Without the data frames, the iPhone again occasionally transmitted probe request frames. In average every 30.6 seconds, the iPhone emitted wireless traffic with the maximum interval of 60.8 seconds in between two transmissions. Overall, 9 frames matching 4 iterations of the server were captured.
While sending the data frames to the iPhone, the maximum interval of 31.0 seconds and average interval of 15.6 seconds were markedly lower comparing to the measurement without sending data frames. But overall, the iPhone transmitted 16 frames in 6 iterations of the server. This amount is only slightly higher and contained exclusively probe request frames, thus we assume that this is dued to fluctuations in the period of the probe request sendings and not because of the data frames. Therefore, the iPhone did not react on the data frames by transmitting any Wi-Fi frames.

In the second test, we used IEEE 802.11 RTS frames to induce the iPhone to transmit Wi-Fi frames. The frames were again transmitted directly successive with no additional timeout between the transmissions and the source address was set to `00:11:22:33:44:55` as well. The measurement results are illustrated in Figure 6.5. Without the transmission of RTS frames, the iPhone emitted in the mean every 10.1



without RTS frames	**Max. time interval:**	30.8s	**Avg. time interval:**	10.1s
	Min. time interval:	0.2s	**Frames / Iterations:**	21 / 12
with RTS frames	**Max. time interval:**	26.6s	**Avg. time interval:**	5.8s
	Min. time interval:	0.2s	**Frames / Iterations:**	56 / 19

Figure 6.5 The figure shows the captured amount of traffic without and with artificially sending IEEE 802.11 RTS frames while the device was unassociated to any network. The results are presented graphically and numerically.

seconds wireless traffic with a maximum interval of 30.8 seconds. 21 frames have

been captured in total, matching 12 server iterations.
While we were transmitting the RTS frames, the iPhone transmitted averagely every 5.8 seconds Wi-Fi frames. The maximum interval was 26.6 seconds. The total amount of 56 frames in 19 server iterations is slightly higher compared to the amount without the transmission of RTS frames. We again reason this to fluctuations in the background scans of the iPhone for seeking for Wi-Fi networks, due to the fact that we were flooding the iPhone with RTS frame transmissions during the entire test period. Hence, the total frame amount of 56 frames is far too low to be reasoned to the RTS frame transmission. Therefore, the RTS frames did not influence the traffic amount emitted by the iPhone.

Associated

When the iPhone is associated to a network, different tasks of the operating system use the connection for data communication. We tested the impact of ICMP echo requests on which the iPhone had to react with an ICMP echo response message. Those transmissions are realized using data frames. In this test, we send echo request messages to the iPhone with no further timeout in between. Figure 6.6 shows the results of the traffic amount measurement.

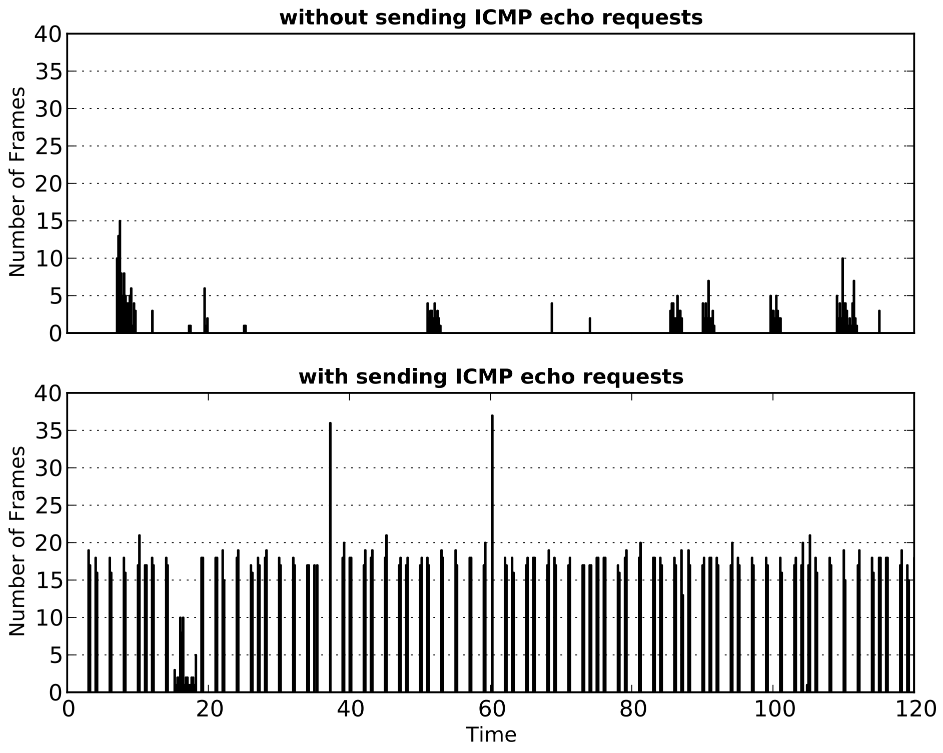
Without the echo requests, the iPhone emitted in the mean every 1.4 seconds Wi-Fi frames with a maximum interval of 25.8 seconds. Overall, the iPhone transmitted 274 frames in 76 server iterations during the test period of 2 minutes.
While we were sending the echo requests, the traffic amount was nearly 10 times higher. In total, the iPhone send 2656 Wi-Fi frames in 162 server iterations. The average interval between two transmissions was 0.7 seconds with a maximum of 1.8 seconds. These results show, that the iPhone reacted on the ICMP echo requests and send echo response messages back. Hence, when the iPhone is associated to a network, we are able to increase the traffic emitted by the device, so that our system is independent from data communications of running tasks on the device.

6.2.1 Summary

The measurements have shown, that the test device does not react on IEEE 802.11 RTS or data frames when it is unassociated to any network. When it is associated, we could increase the traffic emitted by the test device by sending ICMP echo request messages. We assume both methods to be dependend on the test device and for the usage of echo request messages, the device has to implement the ICMP. Therefore, the results might vary between different devices.

6.3 Localization

This section presents the evaluation of the implemented network-based system. We will describe the test environment, used hardware and general setups. Further, we will present and analyse the results of the localization estimation in three different setups.



without echo requests	**Max. time interval:**	25.8s	**Avg. time interval:**	1.4s
	Min. time interval:	0.2s	**Frames / Iterations:**	274 / 76
with echo requests	**Max. time interval:**	1.8s	**Avg. time interval:**	0.7s
	Min. time interval:	0.2	**Frames / Iterations:**	2656 / 162

Figure 6.6 A graphical and numerical representation of the traffic amount while the device was associated to a Wi-Fi network. The upper chart shows the amount without sending ICMP echo requests, the lower while sending those requests.

6.3.1 Scene

The evaluation was conducted on the second floor of the E1 computer science building of the RWTH Aachen University. We deployed 5 sniffer each on a Linksys WRT160NL wireless router distributed to cover the entire environment. A graphic showing the test environment including the five deployed APs is given in A.1. Each AP was connected via ethernet to the present local area network. Our server was implemented on a standard desktop computer connected via ethernet to the same network. In this evaluation, we concentrated on the location estimation regarding a two-dimensional map, but this is neither a limitation to our implemented system nor to the used localization framework.
The localization framework was configured with well known positions of the 5 sniffer in the environment. Further, it required reference RSS measurements in the test environment also at well known locations to build the radio propagation map for each sniffer as described in Section 3.1. These reference measurements were performed using a Lancom AirLancer USB-300agn Wi-Fi stick as target device plugged into an Apple MacBook Air running a version of Ubuntu 11.10 as virtual machine. Overall, the measurements were taken at 11 distributed locations recording 100 RSS values

at each. Furthermore, the localization framework provides three different resolutions for rastering the radio propagation map: $20cm$, $40cm$ and $60cm$. To evaluate our system, we used a resolution of $40cm$.

In the test environment, we defined 5 paths of different complexity and took measurements of three different test devices while traveling the paths. Each path contained a number of well known locations at which we took timestamps during the measurements. All paths have been traveled forwards and backwards to double the timestamps at well known locations. Table 6.1 shows the properties of all paths.
The column *timestamps* denotes the number of timestamps that have been taken at

Path	Timestamps	Avg. time	Figure
room-change	6	14.36	A.2
double-room-change	6	18.22	A.3
floor-straight	10	41.86	A.4
stairwell	6	22.67	A.5
stationary	5	11.48	A.6

Table 6.1 Properties of the defined paths in the test environment denoting the number of timestamps and average travel time for each path. The corresponding figures are denoted in column *Figure* and can be found in the appendix.

well known positions. The *avg. time* value shows the mean time we spend time to travel the paths, thus describes the mean measurement period for each path. The column *figure* gives reference to the corresponding plot given in the appendix.
The path *room-change* was defined to investigate the results when changing from one room to another. The *double-room-change* describes the same for changing the room twice in quick succession. In *floor-straight* we travelled a narrow floor with rooms to both sides and *stationary* was used to investigate the results when the test device stays at one point.

6.3.2 Accuracy Test

We have travelled each path ten times recording the measurements for all three test device setups simultaneously, thus they are all bound to the same environmental influences. In the following, we describe the setup of the three different target devices and discuss the evaluation results.

Setup 1 - Network-based with a Lancom AirLancer USB-300agn Wi-Fi stick

The first setup was the same that we used to record the reference measurements, thus a Lancom AirLancer USB-300agn Wi-Fi stick running under Ubuntu 11.10. As we have seen in Section 6.1, the ad-hoc mode offers the highest amount of wireless traffic emission. Due to this, we set the device to ad-hoc mode so that we received as much RSS measurements as possible without any required active participation at the device.

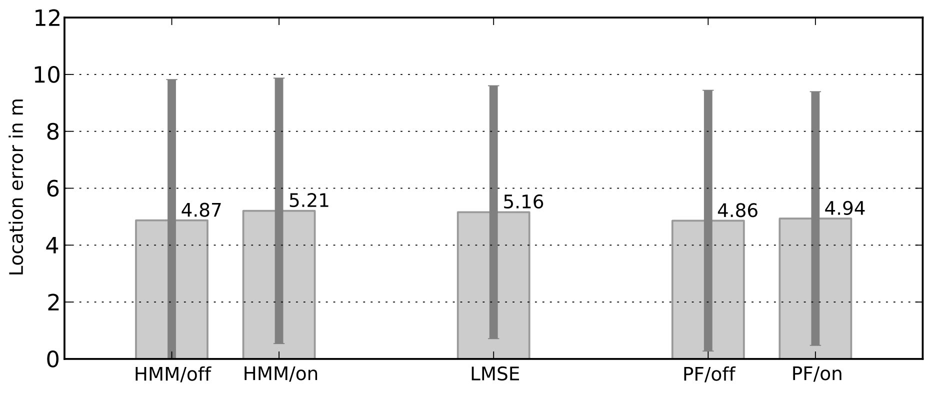


Figure 6.7 Localization errors of the AirLancer Wi-Fi USB stick for the different algorithms averaged over all paths. The standard deviation is also delineated.

As can be seen in Figure 6.7, the localization accuracies for the different algorithms are all similar to each other around $5m$ deviation. The values are given as the mean over all paths. For both HMM and PF as exptected, the offline location estimation showed a better accuracy than the online method, due to the a priori path optimization. For HMM, the difference between offline and online method amounts to $34cm$, for PF only $8cm$. Overall, the best accuracy had been achieved offline with PF, the worst online with HMM.
Further, Figure 6.7 shows the standard deviation for each algorithm which is quite large. This is dued to significantly higher values for the path *stairwell* and can be seen in Table 6.2. The table shows detailed localization errors for each path over

Setup 1 - localization error in m					
Path	HMM/off	HMM/on	LMSE	PF/off	PF/on
room-change	2.64	3.08	3.13	2.57	2.92
double-room-change	3.17	3.18	3.38	3.45	2.98
floor-straight	1.99	3.46	3.13	2.51	3.13
stairwell	14.72	14.47	14.01	13.97	13.81
stationary	1.84	1.84	2.14	1.81	1.84
Mean	4.87	5.21	5.16	4.86	4.94
Std. deviation	4.95	4.67	4.44	4.58	4.46
Median	2.64	3.18	3.13	2.57	2.98

Table 6.2 Detailed accuracy errors for each evaluated path over all algorithms regarding setup 1. For LMSE, there is no difference between online and offline version as described in Section 3.1. A graphical aggregation is given in Figure 6.7.

all algorithms. As can be seen, the *stairwell* path shows for all algorithms an error about $14m$ while the error at the other paths vary between $1.81m$ for *stationary* and $3.46m$ for *floor-straight*. Therefore, the values for the *stairwell* path worsen the mean and standard deviation values compared to the median. The significantly higher error is based on the difficult environment at this path. An elevator and much thicker walls at the stairwell massively influence the radio waves and thus the RSS value at the sniffer. Some sniffer partly even did not receive signals from the test

device at all while at the stairwell.
A notable error difference within a single path can be seen for the *floor-straight* path. The offline version of HMM could achieve an error of $1.99m$ while the error for the online version is about $3.46m$ and for the offline PF version $2.51m$. Further, the error from the offline PF version for *double-room-change* is unexpectedly nearly about $0.5m$ higher compared to its online version.
In total, the offline PF version showed the best performance over all estimations. The LMSE could achieve results with only a slightly higher error compared to HMM and PF.

Setup 2 - Network-based with an Apple MacBook Air

To evaluate the device adaptation of our system, we used a different device than when recording the reference measurements for building the radio propagation map. Therefore, in this setup we used an Apple MacBook Air running OS X 10.8.3 as target Wi-Fi device. Just like in the first setup, we set the device to ad-hoc mode to provide as much wireless traffic as possible without required active participation.

As shown in Figure 6.8, the different localization errors over the five algorithms lay within up to $68cm$. The best results could be achieved with the offline PF

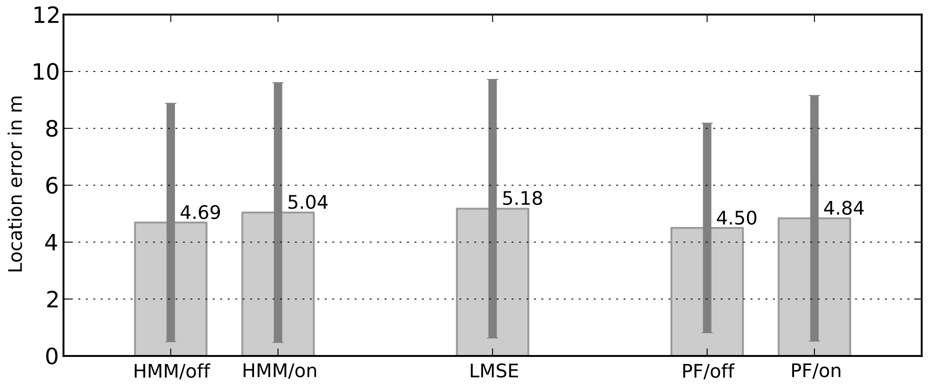


Figure 6.8 Localization errors and standard deviation for the MacBook Air regarding the different localization algorithms.

method with $4.50m$ and the worst with the LMSE method with $5.18m$. The standard deviation is again quite large for all algorithms. Detailed values over all paths are shown in Table 6.3.

As in setup 1, the *stairwell* path showed significantly higher errors for all algorithms due to the impact of the difficult environment on the radio signals. Also the error for the *double-room-change* path is for all algorithms much higher compared to the other paths, except the *stairwell*.
The least error of $1.75m$ could be achieved for the *room-change* path with the offline HMM. Except for the *stationary* path were for HMM and PF the offline error equals the online error, the differences between offline and online are quite large. For the most paths, the offline method expectedly achieved a better accuracy. Only for the path *double-room-change*, the error for the online version is significantly lower compared to the offline version with differences of $1.02m$ for the HMM and $1.24m$ for

Setup 2 - localization error in m					
Path	HMM/off	HMM/on	LMSE	PF/off	PF/on
room-change	1.75	2.46	2.67	1.99	2.58
double-room-change	4.86	3.84	3.73	4.78	3.54
floor-straight	2.24	2.99	3.19	2.43	2.84
stairwell	12.75	14.08	14.20	11.54	13.40
stationary	1.84	1.84	2.10	1.76	1.76
Mean	4.69	5.04	5.18	4.50	4.84
Std. deviation	4.19	4.56	4.54	3.68	4.32
Median	2.24	2.99	3.19	2.43	2.84

Table 6.3 Detailed listing of the resulting localization errors for each path over all algorithms regarding setup 2. The corresponding graphical aggregation is shown in Figure 6.8.

PF. For this path, the LMSE method could achieve the second least error after the online PF method.
Overall paths, the offline PF method reached the least error in the mean and standard deviation, but regarding the median, the offline HMM achieved best results. The least accuracy was reached with the LMSE method.

Setup 3 - Client-based with a Lancom AirLancer USB-300agn Wi-Fi stick

As our system implemented network-based, we want to compare the localization results to a client-based implementation. For this, we set up an additional wireless network interface on each AP in access point mode, so that each AP generated IEEE 802.11 beacon frames in an interval of $15ms$. To collect RSS measurements, we implemented a sniffer on an Apple MacBook Air running Ubuntu 11.10. As WNIC, we again used the Lancom AirLancer USB-300agn Wi-Fi stick and set the interface to monitor mode.

Figure 6.9 shows, that in the mean the offline HMM reached the best accuracy with $6.19m$ and the LMSE the least accuracy with $6.47m$. The difference is with $28cm$ quite small. The standard deviation is again quite large due to very high errors for the *stairwell* path, as shown in Table 6.4.

The error of more than $18m$ for the *stairwell* was between 4 and 7 times higher than for the other paths. The best accuracy was reached with the offline PF method for the *stationary* path. In this setup, the differences between the algorithms are quite small with some cm. A large difference can be seen for the path *floor-straight* where the offline HMM method reached an error of $2.89m$ and the second best accuracy was $3.56m$ with the offline version of PF. As expected, the offline methods achieved a higher accuracy than the online versions, except for the outlier path *stairwell*. Comparing the median values, the offline HMM reached the best result with $3.56m$ and the online HMM the least accuracy with $3.99m$. The difference of $43cm$ is quite low.

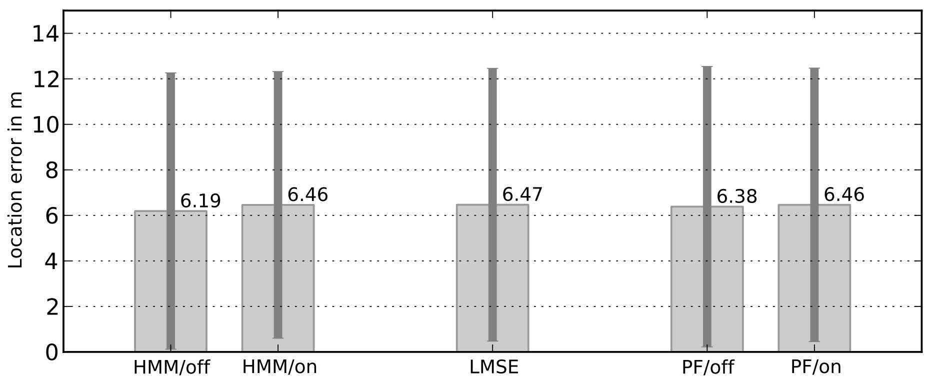


Figure 6.9 Localization errors and standard deviation of the client-based implementation regarding all algorithms.

Setup 3 - localization error in m					
Path	HMM/off	HMM/on	LMSE	PF/off	PF/on
room-change	3.56	3.99	3.63	3.77	3.87
double-room-change	4.10	4.10	4.28	4.00	4.07
floor-straight	2.89	3.99	3.92	3.56	3.86
stairwell	18.26	18.08	18.36	18.62	18.39
stationary	2.14	2.14	2.15	1.97	2.12
Mean	6.19	6.46	6.47	6.38	6.46
Std. deviation	6.07	5.86	5.99	6.16	6.01
Median	3.56	3.99	3.92	3.77	3.87

Table 6.4 Detailed localization errors for setup 3 according to the graphical aggregation in Figure 6.9. The results are listed for all paths over all algorithms.

6.3.3 Summary

Comparing the results of the three different setups, we can see that the differences between the five algorithms are quite small, whereas the difference between the paths are quite large. Over all setups, the *stairwell* path achieved accuracies that generally do not suffice in indoor scenarios and can be reached with already available solutions such as GPS. All other paths could be estimated with errors within few meters up to an overall best accuracy of $1.75m$ in setup 2. Except the *stairwell*, the *double-room-change* path where we changed the room twice within a short time reached the least accuracy. Further, the offline versions of the HMM and PF algorithms mostly reached better results than the online methods, with few exceptions. In total, the best results were reached for the *stationary* path where we remained at a certain location during the entire measurement period.

Against our expectation that the setup we used to collect the reference measurements would achieve the highest accuracy, the best results were reached with setup 2. This shows, that our system can reach comparable results for different devices without making any adjustments in the localization estimation. Furthermore, both network-based setups showed better performances than the client-based implementation. This indicates that a network-based system is not restricted to a lower accuracy than a client-based implementation where the target devices actively participates in the localization process.

7

Conclusion

In this work, we challenged the task to design and implement an easy to adapt and efficient indoor localization system of Wi-Fi devices. As there exists no state-of-the-art solution yet and the amount of location aware applications increases, the demand for an efficient solution increases as well. This has lead to an utterly software-based implementation that can easily be installed on Wi-Fi standard hardware. This solution relies on the information that the wireless network traffic emitted by the client device provides. For the actual localization process, we used an existing localization framework that estimates the targets location based on RSS measurements compared to a priorly build radio propagation map.

The evaluation has shown, that our system is able to determine a clients location with an error of up to less than $2m$. We have investigated diverse test paths in our environment resulting in significant differences of the localization accuracy. This has shown, that the major challenge of RSS based localization systems are differences in the environmental effects on radio signals. Overall, we could build with few effort a network-based indoor localization system that is capable to determine a clients location within few meters.

We further investigated the generally available amount of wireless traffic emitted by standard Wi-Fi devices. As we could see, the traffic amount highly depends on the operating mode of the WNIC and differs between devices as well. Therefore, we examined possibilities to artificially increase the emitted traffic amount and could partly reach a higher amount.

As our system introduces the possibility of localizing Wi-Fi devices with no awareness on the client side, it also induces the demand for counter measures due to privacy protection. For this, we have discussed different approaches to limit the localization accuracy or even prevent from being located at all. Due to the limited scope of this thesis, we could not further investigate the results of those approaches.

7.1 Future Work

Our work and implementation has brought up different future tasks and research topics on network-based localization of Wi-Fi devices.

First of all, there are parts of our implementation that can be improved. We implemented our entire system in Python using a Python interpreter. To eliminate the necessity of an interpreter and simplify the installation procedure, it could be implemented and provided as a software package via package management tools. Furthermore, the architecture could be extended so that the entire system such as filter settings or the processing interval at the server can be controlled from a single point. Moreover, the channel the target device is transmitting on is only determined once in the beginning of the capturing process. To take channel switching at the target device into account, a more sophisticated method could be implemented.
Also the effect of the amount and distribution of the sniffer on the localization error would be interesting to analyse. Additionally, methods of pre-processing the RSS measurements or involving further information sources to improve the localization accuaracy could be investigated.

Our research on methods to induce a higher emitted traffic amount could only achieve partly positive results. Therefore, the impact of other IEEE 802.11 frame types or more sophisticated methods offer further research opportunities.

As we could only discuss ideas of counter measures against our system, it would be interesting to implement them and analyse their impact on the localization capability. It also offers research on further approaches of counter measures as well as privacy and security in localization systems in general.

Bibliography

[1] Blender foundation. `http://www.blender.org/`.

[2] dd-wrt.com. `http://www.dd-wrt.com`.

[3] Freewrt.org. `https://openwrt.org`.

[4] geekzone.co.nz. `http://www.geekzone.co.nz/inquisitor/2996`.

[5] Kismetwireless.net. `http://www.kismetwireless.net`.

[6] Numpy.org. `http://www.numpy.org/`.

[7] Openwrt.org. `https://freewrt.org`.

[8] Ping linux manpage. `http://linux.die.net/man/8/ping`.

[9] Python.org. `http://docs.python.org/2/library/ctypes.html`.

[10] Radiotap.org. `http://www.radiotap.org`.

[11] Sqlite.org. `http://www.sqlite.org/`.

[12] Tcpdump/libpcap. `http://www.tcpdump.org`.

[13] Twistedmatrix.com. `http://twistedmatrix.com/`.

[14] Wi-fi alliance. `http://www.wi-fi.org`.

[15] Internet Control Message Protocol. RFC 792, RFC Editor, 1981. `http://tools.ietf.org/html/rfc792`.

[16] Boithais, L. *Radio Wave Propagation.* McGraw-Hill Book Company, 1987.

[17] Casacuberta, I., and Ramirez, A. Time-of-flight positioning using the existing wireless local area network infrastructure. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on* (2012), pp. 1–8.

[18] IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (ANSI/IEEE Std 802.11, 1999 Edition (R2003)).* Institute of Electrical and Electronics Engineers, Inc., March 2012.

[19] Ippolito, B. `http://simplejson.readthedocs.org/`.

[20] J. Figueiras, S. F. *Mobile Positioning and Tracking.* John Wiley and Sons, 2010.

[21] Kao, K.-F., Liao, I.-E., and Lyu, J.-S. An indoor location-based service using access points as signal strength data collectors. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on* (2010), pp. 1–6.

[22] NAVSTAR GPS - United States Department of Defense. *GPS Standard Positioning Service (SPS) Performance Standard*, September 2008.

[23] Nicoli, M. B. Hmm-based tracking of moving terminals in dense multipath indoorenvironments. *Eusipco 2005* (2010).

[24] Paris, D. T., and Hurd, F. K. *Basic Electromagnetic Theory*. McGraw-Hill Book Company, 1969.

[25] Rothe, D., et al. Indoor localization of mobile devices based on wi-fi signals using raytracing supported algorithms. Master's thesis, RWTH Aachen University, Germany, Ferbruary 2012.

[26] Schmidt, D. C., et al. *Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects*. John Wiley and Sons, 2000.

[27] Schmitz, A., and Kobbelt, L. Efficient and accurate urban outdoor radio wave propagation. *Electromagnetics in Advanced Applications (ICEAA)* (Sept. 2011), 323–326.

[28] Viol, N., Link, J., Wirtz, H., Rothe, D., and Wehrle, K. Hidden markov model-based 3d path-matching using raytracing-generated wi-fi models. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on* (2012), pp. 1–10.

[29] Wallbaum, M., and Wasch, T. Markov localization of wireless local area network clients. In *WONS* (2004), vol. 2928 of *Lecture Notes in Computer Science*, Springer, pp. 1–15.

A

Appendix

A.1 List of Abbreviations

AoA	Angle of Arrival
AP	Access Point
CRC	Cyclic Redundancy Check
DoA	Direction of Arrival
FCS	Frame Check Sequence
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HMM	Hidden Markov Model
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LAN	Local Area Network
LMSE	Least Mean Square Error
NIC	Network Interface Controller
PF	Particle Filter
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
RTT	Round-Trip-Time
TDoA	Time Difference of Arrival
ToA	Time of Arrival
ToF	Time of Flight
TSF	Timing Synchronization Function
WDS	Wireless Distributed System
WLAN	Wireless Local Area Network
WNIC	Wireless Network Interface Controller
WSN	Wireless Sensor Network

A.2 Localization Paths

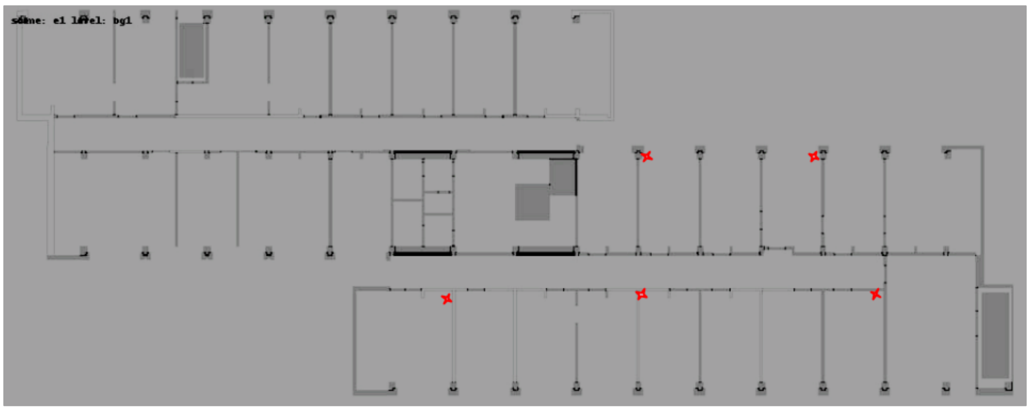


Figure A.1 Our test environment at the second floor of the E1 computer science building of the RWTH Aachen University. The five deployed APs are denoted with red crosses.

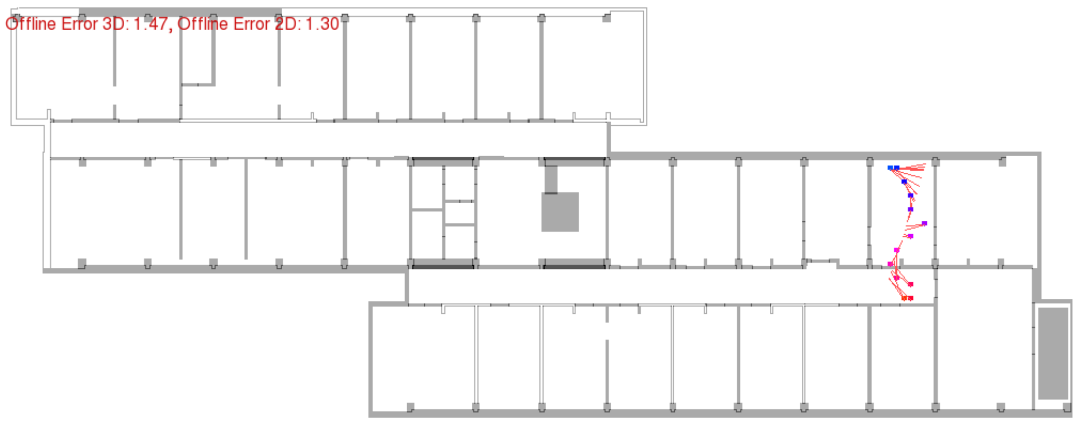


Figure A.2 The path *room-change* starts the floor and visits another room directly after the start. The path is travelled forwards and backwards, ending again in the floor. We defined three different locations for both directions, were we took the timestamp. The points show the estimated, the red lines the error to the real location.

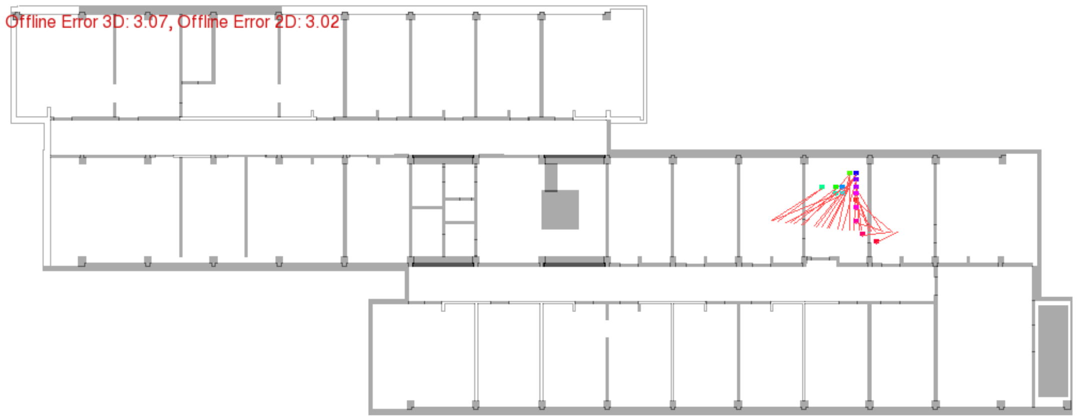


Figure A.3 The *double-room-change* path starts in the middle of a room in the upper right and visits two other rooms in a short interval. The path is travelled forwards and backwards and contained three different locations in each direction, were we took the timestamp. The points show the estimated, the red lines the error to the real location.

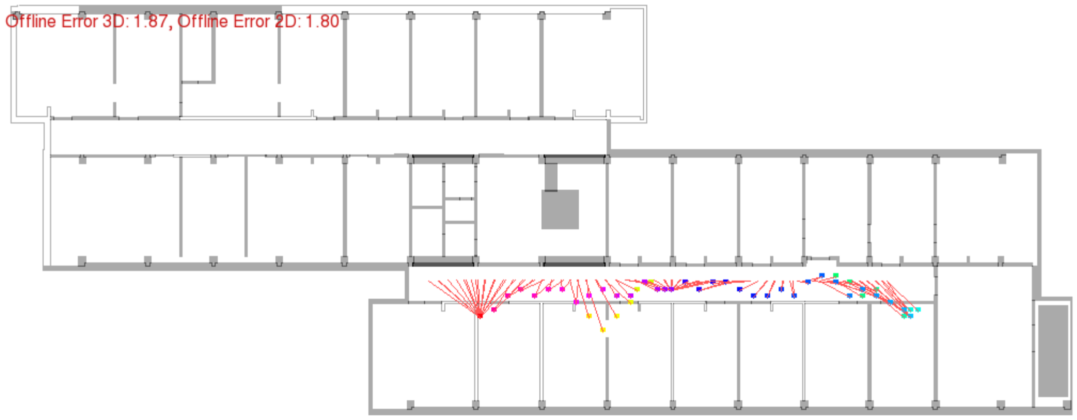


Figure A.4 In the *floor-straight* path, we travelled along the entire floor forwards and backwards. For each direction, there were five different locations at which we took timestamps. The points show the estimated, the red lines the error to the real location.

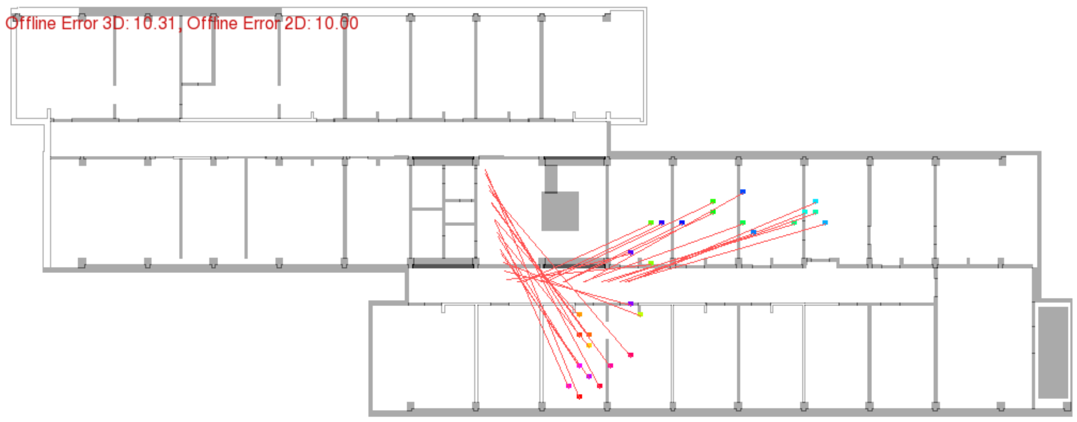


Figure A.5 The *stairwell* path starts in the corner of the stairwell and leads into the floor. We travelled the path forwards and backwards with three different timestamp points for each direction. The points show the estimated, the red lines the error to the real location.

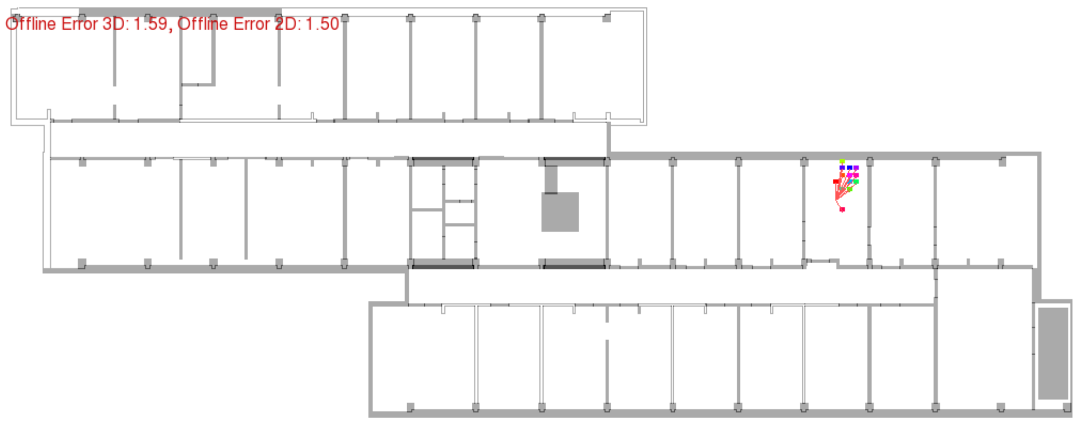


Figure A.6 For the *stationary* path, we remained at one location during the entire measurement period. For each period, we took five timestamps in arbitrary intervals. The points show the estimated, the red lines the error to the real location.