Michael Stocker, BSc

Design of a Decentralized and Synchronous UWB-based Localization System

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Telematics

submitted to

Graz University of Technology

Supervisor

Ass. Prof. Dr.techn. Boano Carlo Alberto
Dipl.-Ing. Großwindhager Bernhard

Institute of Technical Informatics

Head: Univ.-Prof. Dipl.-Inform. Dr.sc.ETH Kay Uwe Römer

Graz, May 2018

AFFIDAVIT

Kurzfassung

Die Möglichkeit die Position von Objekten innerhalb eines Gebäudes zu bestimmen, ist ein begehrtes Feature. Insbesondere im Kontext von Internet of Things (IoT) wird diese Funktion in naher Zukunft sehr gefragt sein.

Das Ziel dieser Arbeit ist es, ein skalierbares verteiltes und selbstorganisierendes Lokalisierungssystem basierend auf der Ultrabreitband (UWB) Funktechnologie zu entwickeln. Dieses System besteht aus statischen Knoten, die in einer beliebigen Umgebung verteilt sind. Jeder dieser Knoten besteht aus einem STM32 Nucleo Evaluierungsboard und dem ersten kommerziell verfügbaren IEEE 804.15.4 kompatiblen Ultrabreitband Transceiver DW1000. Nach einer Setupphase werden UWB Signale benutzt, um die Position mobiler Knoten zu bestimmen. Dabei unterstützt das System die passive Lokalisierung von mobilen Knoten, als auch Selbst-lokalisierung. Bei der passiven Lokalisierung ermittelt das Lokalisierungssystem die Position eines mobilen Knotens basierend auf einem vom letzteren ausgesendeten UWB Signal. Bei der Selbst-Lokalisierung nutzt ein mobiler Knoten die von dem Lokalisierungssystem gesendeten UWB Signale um seine eigene Position zu bestimmen.

Um dieses Ziel zu erreichen wurde das Problem in drei Teile aufgeteilt und untersucht: Lokalisierungsalgorithmen, Zeitsynchronisation und Clusterbildung. Um die Skalierbarkeit zu gewährleisten, ist es empfehlenswert die sogenannte time difference of arrival (TDOA) Methode zu benutzen. Basierend auf dieser Annahme wird in dieser Arbeit ein entsprechendes Lokalisierungs- und Kommunikationsschema vorgestellt.

Weiters wird die Lokalisierungsgenauigkeit und die Konvergenzrate verschiedener Algorithmen untersucht. Um die TDOA Methode nutzen zu können, müssen alle statischen Knoten zeitlich äußerst genau synchronisiert sein. Daher wird in dieser Arbeit das statistische Verhalten der Oszillatoren der Funkmodule analysiert und darauf basierend statistische Filter parametrisiert, um den Synchronisationsfehler zu minimieren.

Da es unmöglich ist, eine netzwerkweite Synchronisation im Nanosekundenbereich zu erreichen, ist es notwendig das Netzwerk der statischen Knoten in zeitsynchrone Cluster zu unterteilen. In dieser Arbeit wird ein deterministischer Clusteringalgorithmus vorgestellt und mit einem probabilistischen Clusteringalgorithmus verglichen. Für beide Algorithmen wird evaluiert, wie genau die Position mobiler Knoten mittels der gebildeten Cluster bestimmt werden kann.

Abstract

Determining the position of objects inside of a building is a highly desirable feature that is required by location-aware applications. Especially, in the context of Internet of Things (IoT) this feature will be in high demand in future.

This thesis aims to build a scalable, distributed, and self-organizing ultra-wideband (UWB) based localization system. This system is composed out of static UWB capable nodes placed in an arbitrary area. Each node is made of a STM32 Nucleo evaluation board and the first commercially available IEEE 802.15.4-compliant ultra-wideband (UWB) transceivers produced by DecaWave. As soon as the system started up, UWB signals are used to estimate the position of mobile nodes. The proposed system supports passive as well as self-localization. For the passive localization, the system estimates the position of a mobile node based on the reception of a single UWB signal at several anchor nodes. For self-localization, a mobile node can use the emitted UWB signals from the anchor nodes to estimate its position.

To reach this goal the problem is split into three separated parts, namely: localization techniques and algorithms, high-precise time synchronization, and clustering. To fulfill the requirement of a scalable solution the need for the time difference of arrival (TDOA) localization method is identified. Based on this, a scalable localization and communication scheme that supports passive as well as self-localization is proposed.

To find suitable algorithms to perform localization, the convergence rate and localization accuracy of different localization algorithms are analyzed.

The TDOA method requires that all nodes are time-synchronous in the sub-nanosecond range, thus, in this thesis, the statistical parameters of the clock uncertainties are determined and used to parametrize a statistical filter that minimizes the synchronization error between nodes.

As network-wide synchronization in the sub-nanosecond range is impossible to achieve, the need for time-synchronous clusters is identified. In this thesis, a deterministic clustering algorithm is proposed and compared to a probabilistic algorithm. For both algorithms, it is evaluated how well they can partition a network of many nodes into time-synchronous clusters that yield high localization accuracy.

Apart from the synchronization, all parts were evaluated theoretically offline. Thus, future work has to port the other parts to the Nucleo platform and evaluate the whole system online.

Danksagung

This master's thesis was conducted at the Institute for Technical Informatics at Graz University of Technology.

I would first like to thank my thesis supervisors Bernhard Großwindhager, Carlo Alberto Boano, and Kay Uwe Römer for their professional help and kind support.

I want to thank my family and friends for their support during my studies, without them, I would not have been able to finish my studies. I also want to thank my partner Teresa for her understanding and aid during the process of researching and writing this thesis.

Graz, April 2018 Stocker Michael

Contents

CHAPTER 1

Introduction

Positioning is the act of determining the position of any objects given some position-related information, e.g., distance to some anchor points. A coarse classification of positioning systems is done by separating them into indoor and outdoor positioning systems. While the outdoor positioning problem can be considered to be solved by the GPS system and its rivals Galileo, GLONASS and Beidou, indoor positioning is still an active field of research.

Being able to determine the position of objects inside a building precisely is a key enabler for many applications. Some examples are intelligent logistic centers, warehouses, hospitals or shopping malls. In logistic centers, an indoor localization system can be used to coordinate robots, people and to locate parcels. In hospitals, an indoor localization system is useful if expensive equipment has to be localized or to locate patients in case of an emergency. A system that provides such a service should ideally be insensitive to measurement uncertainties, resilient against failures, small and energy efficient, such that it remains almost unnoticed.

Several different techniques such as acoustic signals, radio waves or image processing can be employed to determine the position of objects inside buildings. The decision on which technique is used heavily depends on the use case and the resulting constraints. For example, image-based position estimation methods usually require expensive cameras as well as powerful processing units. If thousands of objects should be able to localize themselves or to be localized, such a system may be too expensive and energy consuming. Connecting thousands of objects is part of what is called the Internet of Things (IoT). In this domain, we are usually confronted with energy-constrained devices (i.e., battery-powered nodes). Hence, energy-demanding data and image processing may not be an option. Furthermore, a desired feature of IoT is that the presence of devices stays unnoticed, thus also, the size of a device plays an important role.

1.1 Motivation

This thesis aims to design a low-cost, precise, efficient, scalable and self-organizing indoor localization system. We use the first commercially available IEEE 802.15.4 compliant ultra-wideband (UWB) radio modules produced by DecaWave to achieve this goal. These modules allow to measure the duration of radio waves propagating through space, also called time-of-flight (TOF), with high precision. To be able to determine the TOF precisely, it is important to have synchronized clocks, as poorly synchronized clocks introduce errors into TOF and distance measurements. An indoor localization system based on UWB modules uses fixed anchor nodes with known position, and the position of mobile nodes is determined by measuring the distance to the anchor nodes by estimating the TOF. A localization system that directly uses the distances is based on the time-of-arrival (TOA).

To achieve high precision, scalability, and efficiency, we decided to use a localization method called time-difference of arrival (TDOA). This method essentially does not measure the distance to an anchor node, but rather the difference of the distances between the mobile node and two anchor nodes. This method has several benefits: first, synchronization is only required among the anchor nodes. Second, it allows determining the position of a mobile node with a single message. Third, it allows mobile nodes to localize themselves without any interaction with the anchor nodes, thus allowing for an arbitrary number of mobile nodes to participate. A TDOA system requires that all anchor nodes are synchronized with high precision, and in this thesis, we investigate how tight these nodes can be synchronized. As highly precise network-wide synchronization cannot possibly be achieved, clustering is used to form time-synchronized clusters. Within these synchronized clusters, mobile nodes can localize themselves or can be localized. As the localization precision depends on the distribution of anchor nodes within a localization cluster, we investigate how well simple clustering algorithms are suited to form localization clusters with high localization precision.

1.2 Contribution

The overall goal of this thesis is to design an efficient indoor localization system that is scalable and easy to deploy. Towards this goal, we split the problem into several smaller parts: for each part, we present a solution and all parts combined form the proposed indoor localization system. The first part is the localization part. The second building block is the synchronization part. The third block takes care of clustering. The fourth block is a proposed communication and localization scheme.

Regarding the *localization* part, the contribution of this thesis is to research and compare three localization algorithms and their convergence towards the final position under the presence of TOF measurement noise. We also evaluate if this localization algorithm requires a good initial guess of the final position, or if a weak guess (e.g., at the center of the cluster) is sufficient to ensure convergence.

The contribution of the *synchronization* part is fourfold. First, possible network-wide synchronization methods are analyzed and theoretically evaluated for our use case. Second, the statistical uncertainties of the crystal clocks of radio modules are determined. Third, the simple skew clock correction method and a statistical signal processing based clock correction method are compared regarding accuracy offline. Fourth, both methods were implemented on our test platform and are evaluated online.

In the *clustering* part we propose a deterministic clustering algorithm and compare it to a probabilistic algorithm. Probabilistic algorithms use random values to determine whether a node becomes a cluster-head or not. Deterministic algorithms use locally available information such as the number of unclustered neighbors or node position to decide on its role. We, therefore, apply both algorithms to a set of anchor nodes distributed on a map. The generated clustering is then used to calculate the theoretical localization accuracy for each point on the map and to compare the accuracy of the clustering generated by both algorithms.

We then propose a novel *communication and localization* scheme where passive, as well as self-localization of mobile nodes, is supported. Passive localization is performed by the sensor network by receiving a single message issued from the mobile node. Self-localization means that a mobile node localize itself by only receiving messages issued by the anchor nodes. In detail, the anchor nodes record the reception time-stamp of any messages and return the gathered data to the cluster-head. With this data, the cluster-head can determine the position of mobile nodes. Furthermore, the data messages can be used by a mobile node to perform self-localization to determine its location.

1.3 Thesis Outline

Chapter 2 gives an introduction to the topic of localization and synchronization in wireless sensor networks. To do so, we first describe topic-specific notations and explain different algorithm concepts that are used to achieve localization and synchronization in sensor networks. Afterward, we give an introduction to the three building blocks of the proposed localization system in this thesis, namely: localization, synchronization, and clustering. In the localization part of this chapter, we briefly discuss different radio-based localization techniques. In the synchronization part, we give an introduction to different time synchronization protocols and possible sources of synchronization errors. The clustering part gives an introduction to the topic of sensor network clustering.

In Chapter 3 we summarize available literature about TDOA systems built with the DecaWave chip, and discuss their weaknesses. Furthermore, the proposed indoor localization system and communication scheme are explained.

In Chapter 4 localization is discussed in more detail. Here we lay the mathematical foundation for position estimation given a set of distance (TOA) or difference of distance (TDOA) measurements. We first show closed-form solutions for TOA as well as TDOA systems and discuss why closed form solution might not find the optimal solution. We present the iterative Taylor-series expansion method to find the position given a set of measurements. In the evaluation section of this chapter, we compare the convergence performance of the Taylor-series expansion method with the Gauss-Newton and Maximum-Likelihood Gauss-Newton optimization method.

In Chapter 5 clock synchronization is discussed in more detail. First, an iterative model is derived from a discretized sinus oscillator. Then the same iterative state model is derived from stochastic differential equations (SDE), with this method, the relationship to the Allan variance can be shown. This relationship is then used to perform model identification, i.e., to identify the variances of the noise terms in the SDE equations. With the knowledge of the noise parameters, we create a Kalman filter and compare the performance with the simple skew correction model. In the evaluation section, we compare the Kalman filter with the simple skew model in an offline testing setup and online testing on the Nucleo board.

In Chapter 6 a probabilistic and deterministic iterative cluster algorithm is presented and evaluated regarding localization accuracy. To do so, we came up with three different realistic scenarios where anchors were placed in a room, in a hallway, or in an open area. We used those scenarios to simulate each clustering algorithm and calculate the localization accuracy via the Cramèr-Rao Lower Bound (CRLB).

In Chapter 7 the findings of this thesis are discussed, and further work is proposed.

CHAPTER 2

Localization and Time Synchronization in Wireless Sensor Networks

Advances in miniaturizing microelectronics over the last decade make it possible to develop small wireless, distributed sensors, and actuators. This new technology is now on edge to turn many people's vision into reality: a vision where hundreds or thousands of planned or ad-hoc deployed small devices can work together to capture current ambient conditions, identify presence or absence of objects and people, or determine the movement of these objects. Space and time have a close relationship and are playing an essential role in enabling these services, as both are used to distinguish between events and their effects that are captured by one or more nodes within networks of sensors. More precisely, time synchronization is important to be able to assign the change of sensory output of multiple devices to a particular event. Spatial localization is important to differentiate between events in a network that happened concurrently but at different locations. If a sensor network achieves high spatial resolution and precise time synchronization, it can be used to track objects along the deployed area. This is the use-case discussed in this thesis: thus, we focus on methods that can achieve high precision of space and time localization. Additionally, when it comes to reliable and efficient communication, precise time synchronization is essential, as it is required to coordinate individual nodes accessing a shared medium, using a method commonly known as time-division multiple access (TDMA). Being able to precisely determine when a packet is expected to arrive at a node is also advantageous if one wants to reduce the on-time of a radio receiver to minimize energy consumption.

2.1 Space-time

The concept of space-time in sensor networks was introduced in [1], where a unified view on the topics space and time in sensor networks was developed. In the world as we know it, a point in space is usually defined by a three-dimensional orthogonal space also known as the Euclidean or Cartesian coordinate system. Starting from a reference point O, each point in space can be reached by a linear combination of the 3-scaled basis vectors e_1, e_2, e_3. In space-time there is an additional fourth dimension, which accounts for the time of a point in space: hence, there is a fourth basis vector e_4. A consequence of the additional time dimension is that we do not speak of a point in space, but rather of an event that occurs in space-time.

A particular event in the space-time coordinate system is described by the reference point O and a scaled version of the four basis vectors (O, e_1, e_2, e_3, e_4). Given a point p with its coordinates (p_1, p_2, p_3, p_4), the event in space-time can now be expressed in terms of the previously defined coordinate system as follows: $p = O + p_1e_1 + p_2e_2 + p_3e_3 + p_4e_4$. With this in mind, we can define the spatial distance between two events as follows [1]:

$$d = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2} \tag{2.1}$$

The temporal distance is given by $|p_4 - q_4|$. With this notation, it becomes clear that localization and synchronization can be regarded as a process of determining the coordinates of an event p in space-time with respect to a coordinate system. Such an event can be the transmission or the reception of data packages, or, in other words, when their corresponding radio waves leave and arrive, at a sensor node at a particular position and time in space-time.

The previously mentioned separated view of distance metrics for spatial and temporal separation is, in fact, a simplification that is sufficient if one only wants to separate events above nano-second range. In physics and in particular in special relativity theory, the Minkowski geometry is used to describe space-time. The distance metric of this geometry is quite similar but differs a little bit as the additional time dimension is included in the metric (Equation 2.2). It is well known that Equation 2.2 yields zero if objects that connect two events are moving at speed of light, just like radio waves do. Hence, we can deduce from equation 2.2 that the time difference between two events (transmission and reception of a message), which are causally connected at speed of light, depends on their spatial distance.

In most wireless sensor network (WSN) applications this fact is neglected, and transmitted packets are regarded to arrive almost instantaneous at all receivers. However, if one wants to operate on timing information in sub-nanosecond range, this property has to be taken into account. On the one hand, this causes additional complexity for synchronization. On the other hand, it is this property of nature that allows us to perform spatial localization.

$$\begin{aligned}(\Delta s)^2 &= 0 = -(\Delta ct)^2 + (\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2 \\ (\Delta t)^2 &= \frac{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}{c^2}\end{aligned} \tag{2.2}$$

2.1.1 External and Internal Coordinate System

In [1], the authors differentiate between internal and external localization in space-time. External localization uses a given coordinate system to localize each event in space-time, while for internal localization no external reference coordinate system is given. In this case, the nodes have to agree on a single common coordinate system. An example of an external reference system would be the usage of GPS, which enables nodes to locate themselves spatially, as well as to synchronize their clocks to a global reference system.

2.1.2 Global and Local Coordinate System

Another attribute of localization in space-time is whether it happens in a global or local context. In a global context, the coordinates of two events in space-time must be known with respect to a single global reference coordinate system. If such a global context coordinate system can be established, it is easy to compare two events in space-time. In case of local context, groups of several nodes or even single nodes establish a local coordinate system, which hinders one to compare two events that are recorded by nodes in different local coordinate systems. The solution to this problem is to apply the coordinate transformation to bring recorded events into the same coordinate system. Of course, this can only be done if knowledge of the relationship between two coordinate systems is present at a certain stage in the transformation process. Border nodes connecting two local coordinate systems can be used for this purpose.

2.1.3 Space-time Location Algorithms

Over recent years, different centralized and distributed time and space localization algorithms have been proposed. The goal of all these algorithms is to minimize the error of the estimated and the true location of a node in space-time. Locating nodes in space-time with high precision allows establishing a local or global coordinate system among nodes. Only with the help of such a coordinate system, events that happen within the covered area of the sensor nodes can be precisely located in space-time. A concrete example for this is this thesis, where we try to establish a common space-time coordinate system among a group of anchor nodes. Only if this is successful, we can exploit Equation 2.2 and infer the position of an event (transmission of a packet from a mobile node) from the time difference of arrival events at the anchor nodes. Space-time location algorithms can be coarsely classified as distributed, distributed-centralized or centralized, as it was done in [2]. They can then be further classified into anchor-based or cooperative, single or multi-hop, relative or absolute location [2].

Centralized algorithms
Centralized algorithms such as presented in [3] use information of all nodes to calculate the position in space-time on a single computer. These algorithms usually produce superior performance regarding accuracy, as they can combine all available information of the network. The downside of this approach is that dealing with a considerable amount of data requires more processing power, which is not always desired in power-constrained sensor networks. Another problem is that all the information has to be gathered on a single point which requires additional communication effort resulting in higher energy usage, and poses the threat of a single-point of failure. Furthermore, these algorithms usually do not scale well with the size of a network: an increased number of participating nodes results in a higher computational effort and takes the algorithm longer to converge. In general, centralized algorithms are preferred if a network infrastructure exists for data aggregation, if the network is relatively small, and if accuracy is more important than energy consumption [2]. In [4] the authors propose a localization algorithm based on matrix completion with a modified Newton method, which seems to yield stable and good localization results while reducing the complexity dramatically.

Distributed algorithms
Distributed algorithms do not run on a centralized (sometimes dedicated) node, but rather on each participating node. These algorithms use only locally available information (e.g., distance and position of neighbors) to establish their position in space-time. Their performance is usually considered worse than those of centralized algorithms, though their energy consumption is lower [2]. During the execution of distributed algorithms, it is often required to establish a common value, such as a global time or a global space coordinate system among participating nodes. This kind of algorithm is called distributed consensus-based algorithms, where a parameter is iteratively estimated until all nodes agree on a common value. In [5] Calafiore et al. present a distributed Gauss-Newton algorithm that converges to the same solution as the centralized counterpart. Their solution is consensus-based, meaning that, for each update step, nodes have to exchange information with their neighbors, but no information has to be exchanged with a central station.

Distributed-centralized algorithms
Distributed-centralized algorithms are a combination of distributed and centralized algorithms. The idea is to apply centralized algorithms to overlapping clusters of the network. Border nodes which join two or more clusters function as mediators to stitch clusters together. With this method, it is possible to run centralized algorithms on a small subspace of nodes such that even low-power nodes can handle the amount of data. In [2] Stone et al. point out that this kind of algorithms yields higher accuracy than distributed algorithms while having a low communication overhead.

Cooperative
According to Stone et al. [2], localization algorithms can further be classified as anchor-based vs. cooperative-based. Anchor-based localization uses fixed anchor nodes to localize node in space-time, while cooperative-based algorithms use pairwise distances to establish their position. Anchor-based systems usually yield higher accuracy performance, as the positions of the anchors are known in advance or are determined with high precision during several iterations of a localization algorithm. Cooperative-based algorithms, instead, use the locally available information to localize nodes with respect to other nodes. Algorithms of any of the previously discussed classes (centralized, distributed, centralized-distributed) can be either cooperative or anchor-based [2]. Apart from this strict separation, combinations of both approaches are also possible. Nodes can for example use anchors to localize themselves with respect to the anchors; after that, they can use cooperative localization to refine their position estimation further. With this method, anchor-nodes can "inject" a global coordinate system into the cooperative localization process, which can only establish a local coordinate system.

2.2 Ultra-wideband and DecaWave DW1000 Radio

Ultra-wideband (UWB) communication systems refer to transceivers making use of high bandwidth for data transmission. There are several ways of emitting an UWB signal: the most common one is to emit very short pulses, which is also called impulse-radio UWB (IR-UWB). According to Federal Communications Commission (FCC) an UWB signal is a signal exceeding an absolute bandwidth $B >= 500MHz$ (resulting in a pulse width of $T_p \approx 1/B \leq 2ns$), or a signal exceeding a relative bandwidth $B_r > 0.2$ (or 20%).

Over the last few years, UWB got more attention for several reasons. First, it can achieve high throughput over short distances, which can meet the requirements of interconnected home entertainment systems. Second, it also allows a high density of nodes, which will be essential for the ever-increasing number of smart devices and sensor networks. Third, higher immunity to multi-path fading and a very good time-domain resolution allow for precise position estimation. The DecaWave DW1000 is the first low-cost IEEE 802.15.4-compliant UWB transceiver that can be used for both data transmission and highly precise time-stamping.

Another appealing feature of the DW1000 chip is the ability to output an estimation of the channel impulse response (CIR). The CIR determines how an input pulse is changed through the wireless channel. The CIR gives an idea of the surroundings, e.g., if there is line-of-sight (LOS) connection and information about reflections from walls or scattering from other objects.

2.3 Localization

The ability to localize sensor nodes in a network is a highly desirable capability. Only with this feature environmental monitoring can give sense to the collected data such as temperature, noise, movements, and air quality. Furthermore, the ability to precisely determine the position of nodes in sensor networks enables many new upcoming technologies such as robotics, automated driving, traffic monitoring, or inventory management. For all these applications, different requirements regarding accuracy, reliability, and speed of position estimation are posed. For example, in robotics, it is desired to have high precision position estimation with possibly many participants. Both self-localization and localization of a robot at a central computer might be required. For warehouse or inventory management this constraint might be relaxed, and localization of assets at a central entity can be sufficient.

Position estimation techniques can be coarsely classified into three categories: RSS profile-based, distance-based and angle of arrival (AOA)-based. In this section, we give an overview of radio-based localization techniques and compare their strengths and weaknesses.

2.3.1 Signal Strength Based Distance Measurement

Most radio devices allow determining the received signal strength (RSS) by reading a received signal strength indicator (RSSI). This value can be utilized to determine the relative distance between two nodes. As the RSS indicator is readily available in most devices, this technique is appealing, as coarse position estimation can be enabled without any additional hardware or communication effort.

The basis for this technique to work is the fact that the RSS value in free space decreases proportionally with the squared distance (d^2) between two nodes. The relationship between the transmitted power P_t and the received power P_r is given through Friis equation 2.3, where G_t and G_r are the transmitter and receiver antenna gain, and where λ is the radio wave length [6].

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2} \tag{2.3}$$

This free-space model is idealized and it is commonly known that P_r is influenced by reflection, diffraction and scattering. Furthermore, G_t and G_r as well as P_t can be device-dependent, causing additional localization error.

Based on empirical evidence, it is commonly accepted that the $P_r(d)$ can also be modelled as a log-normally random distributed variable with a distance-dependent mean, as given by Equation 2.4

$$P_r(d)[dBm] = P_0(d_0)[dBm] - 10 n_p log_{10}\left(\frac{d}{d_0}\right) + \chi_\sigma \tag{2.4}$$

where $P_r(d)$ is the distance-dependent received signal strength in *dBm*, $P_0(d_0)$ is the RSS value in *dBm* at a known distance d_0. χ_σ is a zero mean Gaussian-distributed random variable with a deviation of σ which represents the RSS fluctuation introduced by obstacles affecting the wave propagation. n_p is an empirically determined path loss exponent that is different for different environments. From this model, we can derive the estimated distance as shown in Equation 2.5. Contrary to Equation 2.4, P_r and P_0 are given in milliwatts instead of dB.

$$\hat{d} = d_0 \left(\frac{P_r}{P_0(d_0)} \right)^{-\frac{1}{n_p}} \tag{2.5}$$

The relationship between the true distance d and $\hat{d}$ is given by Equation 2.6.

$$\hat{d} = de^{-\frac{\chi_\sigma}{\eta n_p}} \tag{2.6}$$

where $\eta = \frac{10}{ln(10)}$. What we can observe from this equation is that the true distance highly depends on the environmental variable n_p and the random noise χ_σ. Especially in dynamic environments, this can lead to major distance measurements errors.

2.3.2 RSS Profiling

This technique also uses the signal strength value of the received signal and can thus be used with most of the existing radio devices. Contrary to the method discussed in Section 2.3.1, no relative distance to an anchor node is calculated based on the RSS value. The idea is to split a region into cells, where in each cell the signal strengths of all available anchor nodes are recorded. These measurements are then used to calculate an inverse radio map that contains the mean and standard deviation of the RSS value of each anchor node in each cell. To perform localization, this radio map can be used in conjunction with the Bayes theorem to calculate the most likely cell based on observed RSS values as follows [7]:

$$P(C_i|S) = \frac{P(S|C_i) \cdot P(C_i)}{P(S)} \tag{2.7}$$

where $P(C_i|S)$ is the probability of being in cell i given the observed signal strength vector $S = [s_j, ..., s_k]$ of anchors $j...k$. $P(C_i)$ is the prior probability of being in cell i and it is usually assumed to be uniformly distributed (e.g., $P(C_i) = \frac{1}{\text{Number of Cells}}$). The likelihood of observing the signal vector S in cell i is given by [7]:

$$P(S|C_i) = \prod_{j=1}^{N} P(s_j|C_i) = \prod_{j=1}^{N} \frac{1}{2\pi\sigma_{ji}} \cdot e^{\frac{s_j - \mu_{ji}}{\sigma_{ji}}} \tag{2.8}$$

where μ_{ji} is the mean value of the signal strength of anchor j in cell i and σ_{ji} is the standard deviation of the signal strength of j in cell i. To calculate the probability of being in a specific cell, Equation 2.7 is applied iteratively. After each iteration the posterior probability $P(C_i|S)$ is considered as the apriori probability $P(C_i)$. Already after a few (3-5) iterations, the probabilities converge to the estimated cell. While today this method is the most commonly used method for coarse localization, it has a big downside: the radio map has to be built and must be updated regularly as even small changes of objects in a room can influence the received signal strengths. Also, the localization accuracy is limited to a few meters and changes over time, depending on the presence or absence of people or objects.

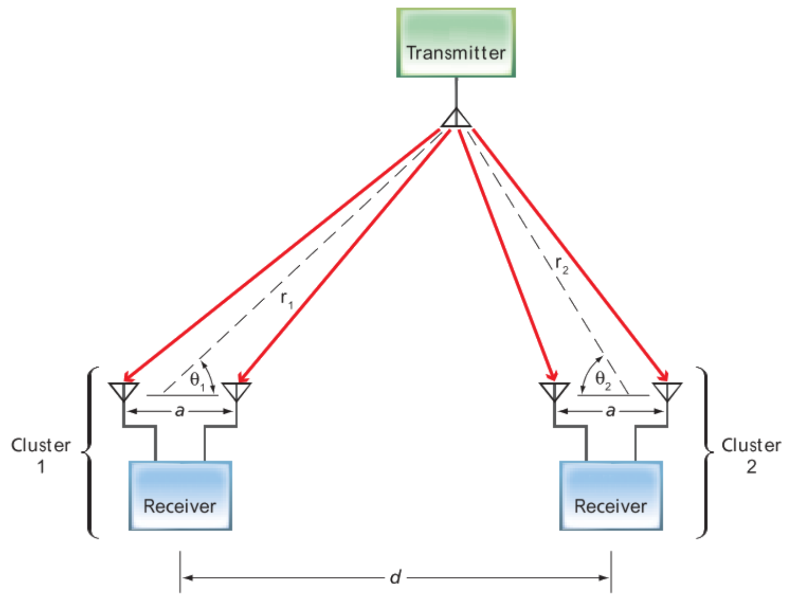


Figure 2.1: Illustration of the AoA method. Modified from [8]

2.3.3 Angle of Arrival

The angle of arrival (AoA) method uses the angle of incoming radio waves to infer the position of the transmitter. If several stations can measure the angle of a single incoming signal, these angles can be used to calculate the position by triangulation [2] [6]. Such a system is shown in Figure 2.1, where the angle of an incoming signal is evaluated at two antenna clusters, the position of the transmitter of the signal, is then found at the intersection of the two bearing lines. There exist different techniques to estimate the angle of an incoming signal. One method is to use an antenna array that allows changing the reception pattern by manipulating the phase offset of individual antennas. One can imagine that the beam of the antenna array is rotated electrically instead of rotating antennas mechanically. The angle of arrival is then found at the maximal reception strength [6]. Other techniques make usage of the time-difference-of-arrival (TDOA) or phase difference of arrival (PDOA) of the same radio wave at different antennas [9]. A major downside of this method is that the error increases with the distance, such that systems with low angular resolution are accurate in the near field, but not in the far field.

2.3.4 TOA

The Time-Of-Arrival (TOA) estimation method allows to measure distances: thus, it can be used in combination with trilateration to find the position. The simplest form of a TOA estimation system is when two nodes exchange a single message containing a global transmission time-stamp, where the receiving node must be able to determine the global reception time-stamp precisely. The distance can then be inferred from the time difference between the transmission and reception time. To be able to determine both time-stamps precisely, it is required that the clocks of both nodes are precisely synchronized and maintain global time, even small timing errors result in large distance errors. Once the distance to at least three anchor nodes is determined, the position of a mobile node can be found at the intersection of the individual circles as depicted in Figure 2.2a.

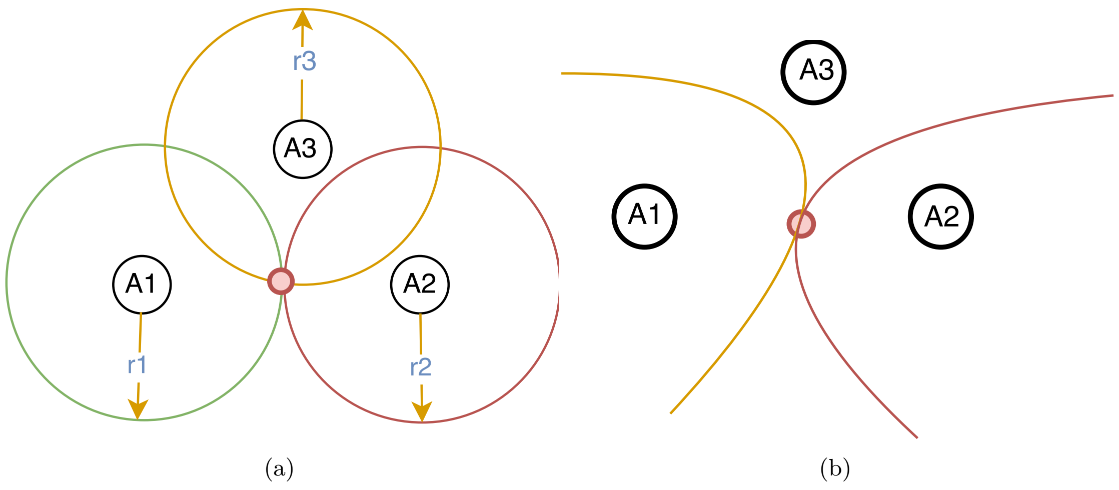


Figure 2.2: Figure 2.2a shows a visualization of the TOA localization process, the position is found at the intersection of the three circles. Figure 2.2b shows a visualization of the TDOA localization process, the position is found at the intersection of the hyperboloids. In case of only three anchor nodes, there may be two points where the hyperboloid intersects.

The simplest form of a TOA measurement is the one-way time-of-arrival method, where it is required that the clocks of both nodes are synchronized and maintain a global time. If this condition is met then the sending node (i) simply sends a packet containing the transmission time ($\hat{t}_{tx}$), the receiving node (j) records the reception time of the packet ($\hat{t}_{rx}$). The time of flight is then calculated by $\hat{t}_{ji} = \hat{t}_{rx,j} - \hat{t}_{tx,i}$ and the corresponding distance $\hat{d}_{ij}$ by multiplying $\hat{t}_{ji}$ with the speed of light c. For the rest of this thesis, we use a simplified notation and express the distance between a mobile node j to anchor node i as r_i. The accuracy of this method highly depends on the accuracy of the clock synchronization, as a difference in the current clock offset is falsely considered as a longer or shorter time-of-flight (TOF).

One way to compensate for different time offsets is by sending two messages: this method is called two way ranging (TWR). The node (i) which wants to determine its distance to another node j initiates the measurement by sending an initial message to node j. The transmission time of message 1 at Node i is denoted by $t^1_{tx,i}$ and the reception time of message 1 at node j by $t^1_{rx,j}$. For the response message, the transmission time at node j

is denoted by $t^2_{tx,j}$ and the reception time at node i by $t^2_{rx,i}$. The time-of-flight can then be calculated with Equation 2.9

$$\hat{t_{ij}} = \frac{(t^2_{rx,i} - t^1_{tx,i}) - (t^2_{tx,j} - t^2_{rx,j})}{2} \tag{2.9}$$

Although this method compensates for the clock offset errors, it cannot compensate for the clock skew error. As described in Section 2.4, quartz clocks can run at different speeds due to manufacturing imperfections as well as due to environmental influences such as temperature. The term $(t^2_{tx,j} - t^2_{rx,j})$ in Equation 2.9 is the so called turn-around time, i.e., the time that module j needs to send a response message. This value is subtracted from the time difference between the reception of the second message at node i and the transmission of the first message at node i $(t^2_{rx,i} - t^1_{tx,i})$. If the clocks of node j and i run at different speed the turn-around time recorded at node j does not correspond to the turn-around time from the point of view of node i. For example, a clock error of 1ppm (parts per million) with a turn-around time of 10ms would result in a turn-around error of 10ns, which is equal to an error of 3 meters.

The solution to this problem is the *symmetric double sided two-way ranging* (SDS-TWR) [10]. The SDS-TWR method uses an additional message to calculate the clock skew between two modules to compensate for it. Although this method yields high accuracy and works well with the DecaWave DW1000 modules, its downside is that three messages have to be exchanged with each anchor node.

2.3.5 TDOA

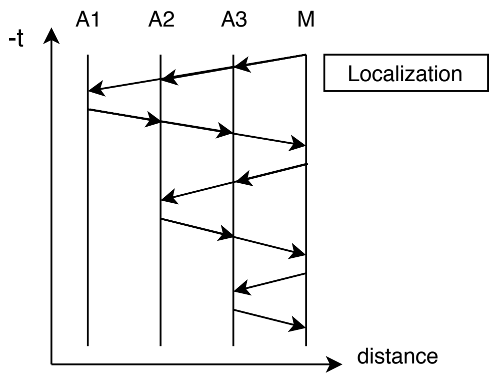
Time-Difference-of-Arrival (TDOA) uses the difference in time between messages arriving at two different base stations. The possible positions of a mobile node given a single TDOA measurement can be represented by a hyperbola (Figure 2.2b) where the base stations are in the foci of the hyperboloids. The position of a mobile node can be found at the intersection of at least 2 such hyperbolas. However, this can lead to ambiguity, as two hyperbolas can intersect at two points. To uniquely identify the position of mobile nodes, the TDOA method requires at least four anchor nodes with known position and synchronized clocks in 2D space [11]. The huge advantage of this method is that only one message has to be sent to the mobile node. Thus a node can be passively localized. This does also mean that the mobile node is not required to perform any clock synchronization or skew compensation.

2.3.6 TOA vs. TDOA

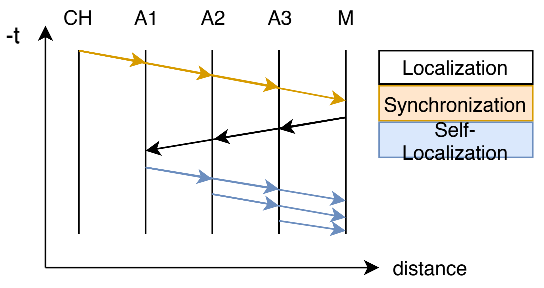
From the previous discussion, we can see that there are fundamental differences between the two localization methods. Both methods use time-stamps, but TOA can exploit additional information, i.e., the transmission time of a packet. As a simple example, we can compare both methods when using only one anchor node. In this case, the TOA method can determine that the mobile node's position is constrained to a circle around the anchor node. Contrary, the TDOA method is not able to constrain the mobile node's position. With two anchor nodes, the TOA method can constrain the position to either two or one places, whereas the TDOA method is only able to find an infinite number of possible positions lying on the hyperboloid. Only if there are four or more anchor nodes, the TDOA method can find a single position, while the TOA method requires only three anchor nodes [11].

Regarding localization accuracy, it is known that under the presence of time-stamping noise the TOA method performs better than TDOA. This is due to the geometry of the hyperboloids where small changes introduced by noise can have greater effects on the curve. However, in [12] the authors found through simulation that both methods perform almost equally well.

In Figures 2.3a and 2.3b we can see the messages exchanged for a TWR-TOA system and a TDOA system on a timeline. Something that can instantaneously be concluded from this image is that the TOA system requires several messages to be exchanged per anchor. In this illustration only two messages are exchanged, but, to compensate for the skew (SDS-TWR method), three or four messages have to be exchanged. Another advantage of the TDOA system that is illustrated in Figure 2.3b is the ability to support self-localization of mobile nodes. Once all anchor nodes are synchronized, they can send localization packets at pre-determined time-slots. A mobile node can receive the packages and compare the expected reception time with the true reception time. This information can be exploited in the same way as in the reversed system (passive localization) to perform localization. This localization can support an infinite number of self-localizing mobile nodes. Putting all together, we can conclude that the TDOA system can scale better at the cost of localization accuracy and additional complexity introduced by the synchronization system.



(a) TWR-TOA



(b) TDOA

Figure 2.3: Figure 2.3a shows an illustration of the message exchange of a TWR-TOA localization system. Figure 2.3b shows an illustration of the message exchange of a passive and self-localization TDOA system. *A1...A3* are anchor nodes. *M* is the mobile node and *CH* is a cluster-head.

2.3.7 Cramér-Rao lower bound

The Cramér–Rao lower bound (CRLB) is used in estimation theory to express a lower bound on the variance that can be achieved by an estimator. Given a set of anchor nodes and a variance of the measurement uncertainty the CRLB can be used to calculate how the measurement uncertainty influences the variance of the position estimation in each point. We follow the approach described in [13] and [12] to calculate the estimated position variance in each point given a set of anchor nodes with a measurement variance of 1. In Figure 2.4 we can see the CRLB calculated for a grid of 16x14 meters. What can be seen from this example is that the localization accuracy degenerates outside of the convex hull spanned by the anchor nodes. We use this method to calculate the localization accuracy in the clustering section.

In [12] Kaune et al. analyzed the theoretical performance of TOA and TDOA localization systems. In their work, they intuitively assumed that the performance of a TOA system is better because the angle of intersection of the circles generated by the TOA system is greater than the angle of intersection of the hyperboloids generated by a TDOA system. They further strengthened their assumption by calculating the angle of intersection for a given placement of anchors for each point on a map. Indeed, it was found that the angle of the intersection outside of the convex hull of the setup was lower for the TDOA system than for the TOA system. Kaune et al. then simulated localization for two scenarios where some stationary anchor nodes are placed on a grid. For each point of the grid, TOA measurements are generated, and white Gaussian noise is added. The TDOA measurements are obtained by calculating the difference between two TOA measurements. The simulation was run 100 times for each point of the grid, and an ML estimator was used to calculate the position given a set of TOA and TDOA measurements. The simulation showed surprising results:

" *The differences between TDOA and TOA localization are not significant. TDOA localization based on the full measurement set shows in the inner area between sensors better performance than the localization using a fixed reference sensor. Overall, TDOA and TOA localization yield roughly the same results. [...] These results do not support the expectations resulting from the angle study. One reason for the surprising performances is the larger dimensionality of the parameter vector for the TOA optimization process.* " [12]

Because of this results we decided to use their approach and calculate the CRLB for the TDOA system using the simpler formulas of the TOA case.

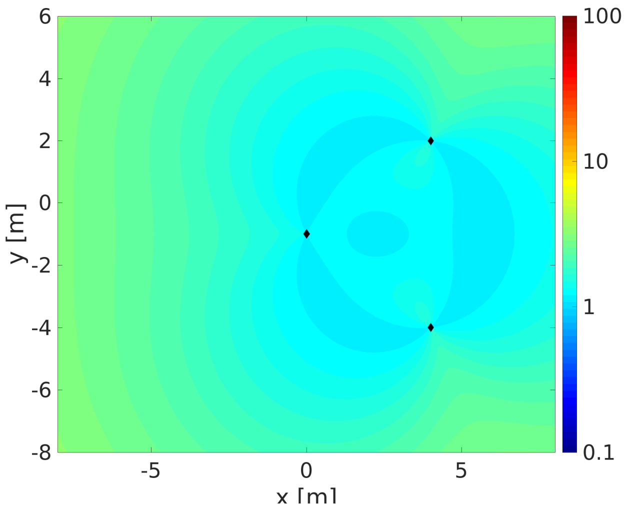


Figure 2.4: Estimated location uncertainty given three anchor nodes with a measurement uncertainty of $\sigma = 1$ meter. The colorbar on the right is logarithmic and shows the mapping between the localization variance and the corresponding color. According to [12] and [13] the TOA and TDOA are only differ marginally.

2.4 Time Synchronization

In this section, we give an introduction to computer clocks, how they work and what are the strengths of different types of clocks. Furthermore, we want to give an overview of existing synchronization protocols. These protocols can be classified into different synchronization schemes based on how many messages are exchanged and on whether they use a broadcast mechanism.

2.4.1 Computer Clocks

The clocks of all modern computers and embedded systems are made out of crystal oscillators. These are electronic oscillator circuits that use the mechanical resonance of piezoelectric materials to produce a periodically varying electric signal with a constant frequency. Because of their low cost and small size, quartz crystal oscillators are the most common type of electronic oscillators nowadays. A well-recognized problem is the stability of crystal oscillators, as their resonant frequency can highly depend on environmental factors such as temperature, shocks and power supply stability. Also, manufacturing imperfections influence the output frequency of the oscillator circuit, such that some modules run faster than others. In Figure 2.5 we can see the time reported by three different clocks A, B, and a reference clock running at frequency f. Clock A runs faster than the reference clock, thus its frequency is $f_A = f * (1 + e_A)$ where e_A

is the frequency deviation or skew of clock A. Clock B runs slower than the reference clock, thus its frequency is $f_B = f * (1 + e_B)$ where e_B is the frequency deviation or skew of clock B. The local time $C_A(t)$ reported by clock A is $C_A(t) = (1 + e_A) \cdot f \cdot t$, and $C_B(t) = (1 + e_B) \cdot f \cdot t$. The relative skew $\gamma = (f_A - f_B)/f_B$ [10] [14].

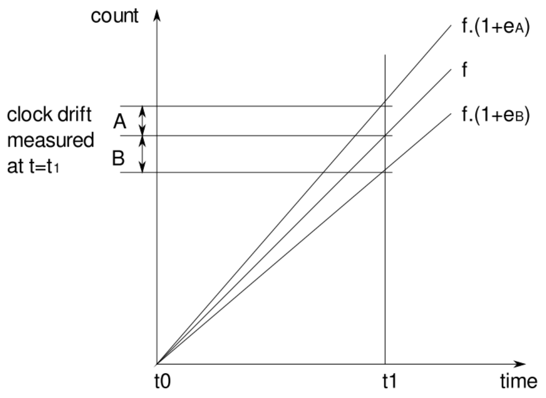


Figure 2.5: Shows the clock count of two clocks evolve over time. Taken from [10].

According to the different price, accuracy, and stability requirements, different types of crystal oscillators were developed [15]. Especially for applications such as our proposed TDOA system, high frequency stability is required where even small changes introduced by temperature change can have a huge impact on the accuracy.

Different types of temperature compensation techniques have emerged to solve this problem, each with their specific cons and pros [15].

Simple Packaged Crystal Oscillator (SPXO). These oscillators only include the main oscillator circuit and an output circuit without any temperature compensation. As they are cheap to produce, they are the most common crystal oscillators. However, they cannot be used if high-frequency stability is required. Typical frequency stability between 0°C and 70°C is ± 10ppm.

Temperature Compensation Crystal Oscillator (TCXO). This kind of oscillator uses analog or digital temperature compensation circuits. These circuits use the output signal of a temperature sensor to generate a correction voltage that is applied to the crystal oscillator circuit to compensate the temperature drift. Typical TCXO oscillators reach temperature stability of ± 0.5ppm over a temperature range of 0°C to 70°C. They are a good trade-off between SPXO oscillators and OCXO oscillators, as they are still affordable (10-100) and require less energy (0.04W) than OCXO oscillators [15].

Oven Controlled Crystal Oscillator (OCXO). These oscillators are embedded into a heating block. Via additional temperature sensors, a constant temperature is maintained within the block, such that the oscillator outputs a constant frequency. They reach a stability of ± 0.003ppm in a range of 0°C to 70°C, which is the highest accuracy of all three clock types. The downside of this type of oscillator is the relatively high power consumption of 0.6W and its high cost (more than 200$) [15].

2.4.2 Synchronization schemes

As indicated at the beginning of this section, time-synchronization protocols can be classified based on how many messages are exchanged as well as on whether broadcast messages are used or not.

One-Way
The simplest way to carry out pairwise synchronization are one-way message protocols, where one node (i) broadcasts a synchronization message containing the transmission time $t^n_{tx,i}$ and where the reception time at node j is denoted with $t^n_{rx,j}$. The time difference or offset θ^n at time instance n (or message n) n between the two nodes can be then calculated with $\theta^n_j + D = t^n_{rx,j} - t^n_{tx,i}$ at node j. D is the time a synchronization message needs to travel between the two nodes. Thus, D contributes to θ, leading to an imperfect offset synchronization [16]. If θ^n_j, θ^{n-1}_j and the interval τ (with respect to node **i**) between two messages are known then, the relative skew between two clocks can be computed as follows:

$$\gamma^n = \frac{\theta^n_j - \theta^{n-1}_j}{\tau} \tag{2.10}$$

For simplification, for the rest of the thesis it the reported time of the reference node is considered as the true time. This means that $e_{ref} = 0$ or $f_{ref} = f$, and $\tau = t_2 - t_1$ is always with respect to the reference clock.

Two-Way Synchronization
For the two-way message exchange synchronization protocols, node j replies to the first message of node i with a second message containing the reception time of the first message ($t^1_{rx,j}$) and the transmission time of the second message ($t^2_{tx,i}$). Node i then records the reception time $t^2_{rx,i}$ of the second message [16]. Assuming that the propagation delay and clock skew did not change over the time of the message exchange, node i can now determine the propagation delay D^n, and the offset θ^n by computing:

$$D^n = \frac{(t^{n-1}_{rx,j} - t^{n-1}_{tx,i}) + (t^n_{rx,i} - t^n_{tx,j})}{2} \tag{2.11}$$

$$\theta^n = \frac{(t^{n-1}_{rx,j} - t^{n-1}_{tx,i}) - (t^n_{rx,i} - t^n_{tx,j})}{2} \tag{2.12}$$

In comparison to the one-way message exchange, the two-way message exchange allows node i to check whether the calculated value D lies within certain bounds. For example, D should not be negative and should not exceed a certain threshold value D_{max}. This simple check allows dismissing synchronization messages that have unusually high propagation delays, which may be an indicator for erroneous measurements. The downside of this method is that the number of required message exchanges increases with the number of participating nodes [17]. Like for the one-way message exchange, we can use θ^n and θ^{n-1} to calculate the skew.

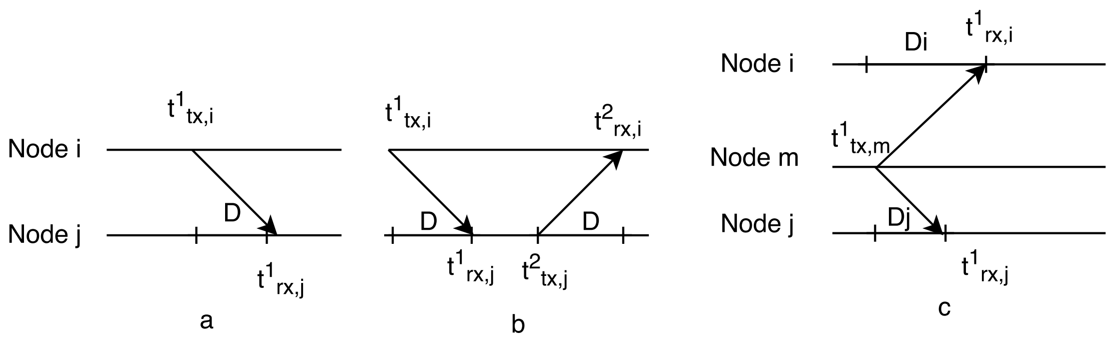


Figure 2.6: (a) One-Way synchronization, (b) Two-Way synchronization, (c) Receiver-receiver synchronization.

Receiver - Receiver Synchronization (Reference-Broadcast)
The receiver - receiver (or reference-broadcast) approach is different from the pairwise synchronization method, as it only considers the reception time of the same message at different receivers. No transmission time-stamp is needed thus it is not carried in the broadcast message [16]. This approach requires that the same message can be received by multiple receivers. Thus it only works in broadcast environments. The synchronization accuracy is mainly influenced by the propagation delay of the same message to the different receivers.

2.4.3 Uncertainties

Uncertainties in the reception time-stamp as well as in the transmission time-stamp can have a huge impact on the synchronization accuracy. As discussed in [18] the following different origins of uncertainties exist:

Send time
Depending on the used hardware, the medium access layer control (MAC) layer might not allow to set the transmission time and is influenced by the software stack. In this

case, the actual transmission time depends on various factors such as current system load and scheduling. Hence, the transmission time is non-deterministic.

Access time
Depending on the current network medium usage, it can take milliseconds or up to seconds until an error-free transmission of a packet can be guaranteed.

Transmission time
This is the time it takes a receiver to transmit a packet. While this is regarded as an uncertainty, it may not be relevant for the reception time-stamping.

Propagation time
This is the time a packet requires to travel from one node to an other. While this may be deterministic in a fixed setup with LOS condition between nodes, it is non-deterministic if the nodes move around or if the packet is forwarded via several nodes. The latter is mainly because of the random delays introduced by each hop.

Reception time
Likewise, for the send time, the MAC layer might not support time-stamping such that the recorded reception time is influenced by the software stack, i.e., by the system load, scheduling, etc. Also, if the hardware supports reception time-stamping, it cannot always determine the reception time with absolute certainty.

Uncertainties in the case of the DW1000 module
The DW1000 can precisely set the transmission time and measure the reception time. Though, the accuracy of the reception time depends on the strength of the received signal. Moreover, it does not perform any medium access checks. Thus, there is no access time delay. The uncertainty that is most relevant for this chip is the propagation delay, however, this delay is used to measure the TOF.

2.4.4 Multi-hop synchronization

One inherent property of sensor networks is that they can be distributed over large areas where not all sensors can communicate with each other, i.e., non-line of sight condition (NLOS). Thus, to establish a common time base in a large sensor network, multi-hop synchronization is required. As discussed previously, the synchronization accuracy vastly depends on access and measurement uncertainties, which are difficult to estimate on a logical link that is based on many physical hops. According to [17] four schemes can be identified and can be used to overcome this problem.

Out-of-band synchronization
This is the simplest scheme to achieve network-wide synchronization. It simply assumes that a large amount of master nodes is distributed in the network, such that every node

has a direct connection to at least one master node. In this case, out-of-band means that techniques other than the communication interface (wired or wireless) are used to synchronize the master nodes. An example is the GPS system. The downside of this scheme is that expensive GPS hardware and line-of-sight condition to the GPS satellites are required.

Clustering

This approach was proposed by the authors of the reference broadcast algorithm (RBS) [19], described in Section 2.4.5. The idea is to split a big sensor network up into small clusters, and all nodes within a cluster synchronize their clocks based on reference broadcast. As described in the previous section, the synchronization within a cluster can come in different variations: the simplest one is that all nodes synchronize to one reference node. To achieve network-wide synchronization, nodes can be part of several clusters. These nodes (gateway nodes) can transform the time of one cluster into the time of another. The size of the cluster influences the number of clusters in a sensor network. With increased size of clusters, the number of clusters is reduced, but more energy is required for transmitting packets. Moreover, a higher communication range also means that fewer nodes can communicate at the same time. A smaller cluster size means that more clusters are formed in the sensor network. Thus, if one sends a packet containing a time-stamp from one side of the cluster to another, the time-stamp has to be transformed several times, leading to an increased synchronization error.

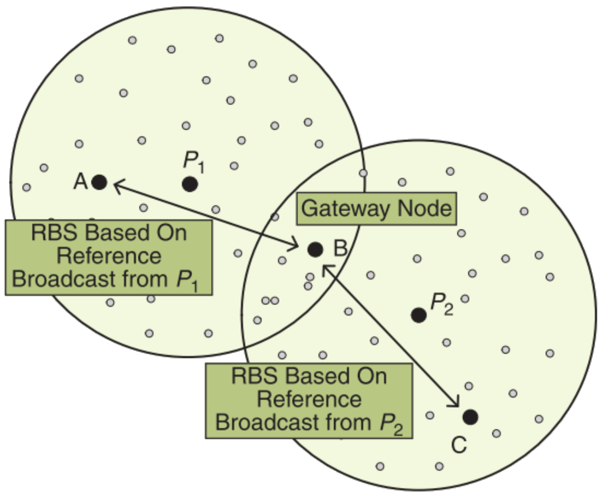

A
P_1
Gateway Node
B
RBS Based On Reference Broadcast from P_1
P_2
RBS Based On Reference Broadcast from P_2
C

Figure 2.7: How clustering is used to achieve network wide synchronization. All nodes of one cluster synchronize to their reference head (*P1* or *P2*). A gateway node (*B*) is used to translate time-stamp from one cluster to another. Taken from [20].

Tree Construction
This scheme is the most commonly used synchronization scheme in sensor networks. The idea is to construct a hierarchical topology in the form of a spanning tree. Starting from the root node, each child node is synchronized via pairwise synchronization. Three main problems are identified for this scheme: First, with increasing tree depth, synchronization errors accumulate such that nodes with a high level might be considerably worse synchronized than nodes that are directly connected to the root node. Then, again, if the depth is kept low (resulting in higher tree-width), many child nodes have to perform pairwise synchronization with their parent, resulting in an increased computational and energy demand for the parent nodes. Second, the tree construction algorithms must adopt to topology changes as quickly as possible. Furthermore, if the root node fails, a new root node must be elected as fast as possible. Third, if two physically nearby nodes have a large logical distance in the tree, their synchronization might be considerably worse compared to direct synchronization.

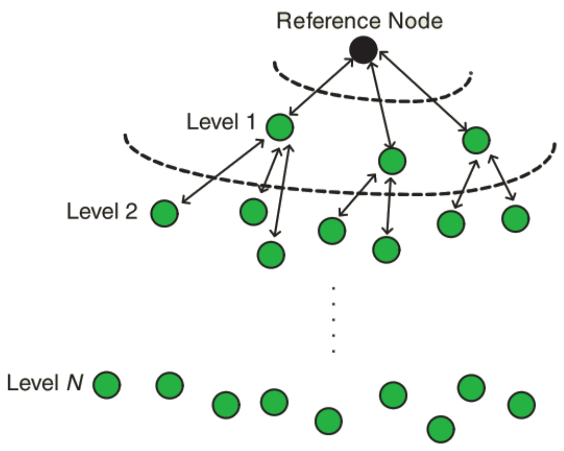


Figure 2.8: Illustration of a tree-based synchronization protocol. Nodes of level 1 use pairwise synchronization to synchronize their clocks to the reference node. As soon as they are synchronized, nodes at higher levels use them as reference node. Taken from [20].

2.4.5 Synchronization Protocols

The most prominent examples of each of the previously discussed synchronization schemes are presented next.

The Network Time Protocol (NTP)
NTP is the most widely used time synchronization protocol, as it is used to synchronize clocks of computers and other devices via the Internet. The NTP protocol assumes that

many time reference nodes are present in the network and each node synchronizes to one of the reference nodes.

Reference-Broadcast synchronization (RBS)
One of the most prominent receiver-receiver synchronization protocols is the reference-broadcast synchronization (RBS) protocol [19]. The simplest form of this protocol works in the following way:

- A transmitter broadcasts a synchronization packet.
- Two nodes record the reception time-stamp of this packet.
- The two nodes exchange their recorded reception time-stamps.

After they exchanged their recorded time-stamps, both nodes can deduce the current time of the other node by comparing their recorded time-stamps with the time-stamps of the other node. This very simple form of RBS does neglect the clock skew and assumes that the time-stamping is perfect. In practice, however, the reception time-stamp is Gaussian-distributed around the true value, and the clock skew is not zero. Hence, several reference broadcast messages have to be exchanged to estimate the clock skew and minimize the time-stamping error [19].

For bigger networks, the RBS protocol can be extended to work in multi-hop environments. Therefore, the network has to be clustered in such a way that each cluster is synchronized by one reference broadcast node. Gateway nodes that are part of two clusters can be used to transform the time coordinate system of one cluster into the coordinate system of another cluster.

Timing-sync Protocol for Sensor Networks (TPSN)
The TPSN synchronization protocol [21] uses the two-way message synchronization principle and works in two phases. During the first phase (level discovery phase), each node is assigned a level with only one node having level 0. This node is connected to an external time reference and starts to send level discovery packets. All nodes that receive this packet assign themselves to level 1. Afterward, all nodes of level 1 start to send level discovery packets. This process is repeated until all nodes are assigned to a level that is greater than the one of the parent node. During the second phase (*synchronization* phase) nodes of a lower level (level n-1) start to synchronize nodes of a higher level (level n) via two-way message exchange [22].

Tiny-Sync and Mini-Sync
Mini-Sync and Tiny-Sync [23] are using multiple round messages to obtain time-offset data-points, each data-point represents the clock offset between two nodes. As previously described in Section 2.4.2 (Equation 2.11), the round messages can be used to calculate the distance or propagation delay between two nodes such that this value is subtracted

from the time offset measurements. Both protocols keep multiple data-points to employ the line-fitting technique to estimate the offset and rate change between two modules.

Flooding Time Synchronization Protocol (FTSP)
The FTSP protocol [18] uses a flooding approach in which the time synchronization messages are flooded through the network. A master-node, usually the node with the lowest ID, starts to broadcast synchronization messages containing its local transmission time. Each nearby node receives several of these messages and computes the time-offset and skew (rate difference) by linear regression. Afterward, it starts to forward or re-broadcast the synchronization messages of the master node, but with updated time information, i.e., it compensates for the delay between the reception and transmission of the synchronization message. In contrast to the RBS protocol, FTSP contains a transmission time-stamp of the broadcasting node, such that a global time can be established among all nodes.

Glossy
Glossy [24] is another flooding approach to synchronize sensor networks. As described earlier, the FTSP flooding protocol simply starts to issue synchronization packets as soon the clock skew was determined. Some problems that can arise from such a broadcast protocol are discussed in [25]. Redundancy arises if a node starts to broadcast a synchronization message to nodes that have already received a synchronization packet in the last round. Congestion arises if many nodes try to re-broadcast a synchronization message at almost the same time, leading to a reduced performance of the network and synchronization accuracy. Collisions can happen if two nodes broadcast a synchronization packet at the same time. As noted in [24], finding a collision-free broadcast schedule is an NP-complete problem and is even more complicated if the topology changes suddenly. The Glossy protocol resolves this issues by considering concurrent transmission as an advantage rather than an issue. In their paper [24], Ferrari et al. describe how simultaneous transmission of the same packet can interfere constructively, allowing a node to decode a packet sent by two or more nodes with even higher probability. The huge advantage of Glossy compared to most other protocols is that it can run decoupled from other application tasks (as shown in Figure 2.9), such that no complicated synchronization message exchange has to be orchestrated.

2.5 Clustering

The act of finding a suitable partitioning of the network is called clustering, where each cluster is composed of a *cluster-head* (CH) or *reference-head*, and some *member* nodes. Since finding an optimal clustering is an NP-hard problem, all clustering algorithms are heuristic, meaning that no clustering method can guarantee to be optimal or perfect, but sufficient to meet immediate goals [26]. Clustering of nodes in sensor networks is performed with different objectives in mind. The main objective of clustering is to improve network lifetime by reducing energy consumption: this can be achieved by clustering in

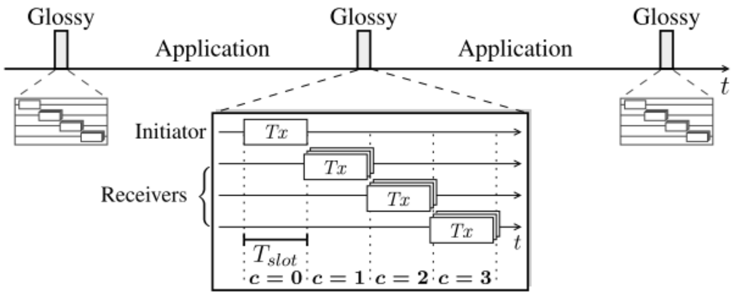


Figure 2.9: Glossy decoupling from application tasks. Taken from [24].

several ways. Many (WSN) applications require only aggregated values to be reported to a central entity: for example, it may be sufficient to report the mean, maximal, or minimum value of a sensed parameter within an area. In this scenario, a cluster-head is used to collect this values from its member nodes and only forwards a single value to the central entity. Another way to save energy using clusters is by reducing the transmission power: the member nodes can report sensed values to their central cluster-head with low transmission power (and range). The cluster-head then forwards gathered information to the next cluster-head with a higher transmission power. Moreover, lower transmission range can help to reduce collisions and congestions in sensor networks. An illustration of how data can flow through a clustered sensor network can be seen in Figure 2.10.

In the remainder of this section, we give an introduction and discuss different clustering algorithms. For this purpose, we follow the classification of cluster-head selection algorithms as it was done in [27]. Afsar et al. classify the clustering-algorithms based on their cluster-head selection mechanism as shown in Figure 2.11. As there are numerous algorithms we are only discussing the most prominent one of each class to get an idea of the underlying mechanism and its strengths and weaknesses.

2.5.1 Clustering characteristics

In [27] Afsar et al. define three main characteristics to classify different clustering algorithms: *cluster properties*, *cluster-head properties*, and *clustering process properties*.

Cluster properties cover properties such as the *cluster-size* (either equal or unequal), *cluster-count* (constant/preset or variable), *inter-cluster communication* (single-hop or multi-hop) and *intra-cluster communication* (single-hop or multi-hop).

Cluster-head properties define the properties of the cluster-head, such as whether they are mobile or stationary. Furthermore, properties are whether the node types are heterogeneous or homogeneous and the role of a cluster-head, i.e., if the CH simple relays data or if it performs data aggregation/fusion.

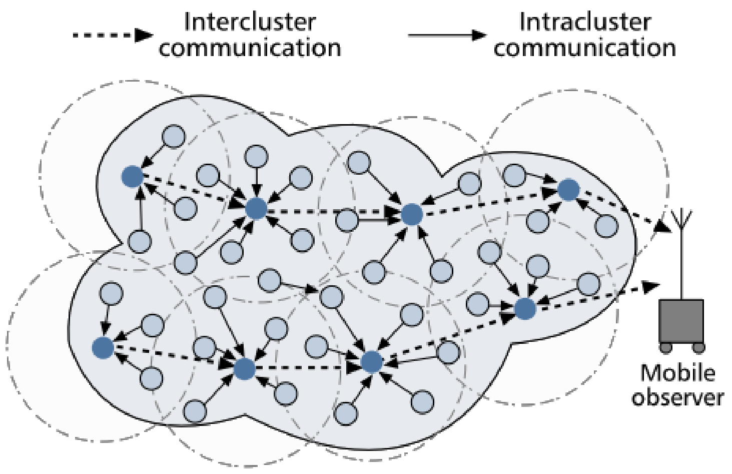


Figure 2.10: Illustration of data flowing through a clustered network. Member nodes report data to their cluster-heads via intracluster communication with low transmission power (and range). Cluster-heads forward aggregated information with higher transmission power (and range) to the next cluster-head. Taken from [26].

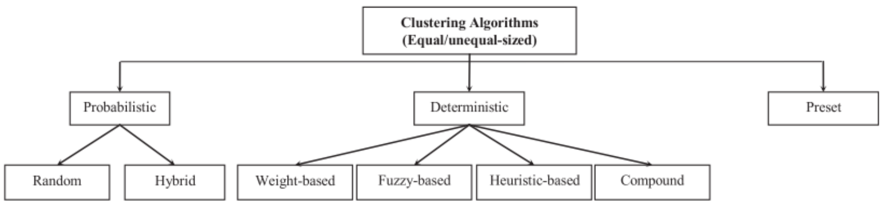


Figure 2.11: Classification of different clustering algorithms. Taken from [27].

Clustering process properties define properties of the clustering algorithms. They distinguish between the method, e.g., distributed or central: the objective function of the clustering process, and the CH election mechanism: *preset*, *probabilistic*, and *deterministic*. Other properties of the clustering-process are the algorithm complexity and whether the nodes are dynamic or static.

2.5.2 Probabilistic clustering

The objective of most probabilistic clustering algorithms is to maximize the network lifetime as much as possible. These algorithms usually have a low protocol overhead, as the decision of each node turning itself into a cluster-head is made randomly. The most prominent probabilistic algorithm is the LEACH protocol [28].

In LEACH, the time is divided into rounds, each round is further divided into a *setup* and *steady-state* phase. During the setup-phase the cluster-heads are elected randomly, i.e., each node decides to become a cluster head with a certain probability. During the *steady-state* phase, data is transmitted to the base-station. This approach ensures that each node is elected as cluster-head at least once, such that the load is distributed among all nodes in the network equally. In detail, each node is generating a random value between 0 and one during the setup phase. Each node that has not been previously elected as cluster-head computes a second value $T(n)$:

$$T = \frac{p}{1 - p\ (r\ mod \frac{1}{p})} \tag{2.13}$$

where p is the desirable percentage of cluster-heads in the network, and r is the current round. If a node has already been selected as cluster-head in the last $1/p$ rounds, T is simply set to 0. Each node then compares its randomly generated value with the calculated value T. If the randomly generated value is less than T a node elects itself as cluster-head and broadcasts a *cluster-head advertisement* packet. If the value is greater than T, a node simply sends a *join request* to its nearest cluster-head.

Hybrid probabilistic methods combine random methods with some kind of available parameters like residual energy or node degree. An extension of the LEACH protocol that uses residual energy in its election process was proposed in [29]. One approach they propose is to modify Equation 2.13 by adding the current residual energy, resulting in:

$$T = \frac{p}{1 - p\ (r\ mod \frac{1}{p})} \frac{E_{current}}{E_{max}} \tag{2.14}$$

where $E_{current}$ is the current energy and E_{max} is the maximal energy of a node. Simulations have shown that the network life-time was increased by up to 30%.

Another hybrid clustering algorithm is the *Clustering Algorithm via Waiting Timer* prosed in [30]. This algorithm randomly sets a waiting timer, and it declares itself as a cluster-head as soon as the timer expires. Additionally, a node decreases the waiting timer as soon as a neighbor is detected. Thus nodes compete with each other, as the nodes with higher degree tend to declare themselves as cluster-head earlier.

2.5.3 Deterministic clustering

Deterministic clustering algorithms do not use randomly selected values to decide their role in a sensor network. Instead, they use the locally available information to elect a suitable node as cluster-head, such as residual energy, node degree, or centrality. This kind of information can be exchanged via messaging, which can introduce an additional overhead depending on the diameter and number of nodes. As depicted in Figure 2.11, deterministic algorithms can be further classified into weight-based, fuzzy-based, heuristic-based, and compound.

Weight-based algorithms use some metrics to compute a weight that is then used for the selection of the CH. The idea of this kind of algorithms is that nodes compete with surrounding nodes where the node with the highest weight wins and is elected as a cluster-head. Usual metrics that are used in such algorithms are the residual energy, node degree, centrality, and distance to the base station. An example of such an algorithm is the Distributed Clustering Algorithm (DCA) proposed in [31]. This algorithm uses a generic weight value that is calculated locally based on some metrics such as residual energy or node degree. Each node decides on its role (cluster-head or member) solely based on the decision of the neighbor nodes with a higher weight, that is, if a node with higher weight decides to be a cluster head, it broadcasts a cluster-head advertisement message. Each nearby node with lower weight will then join this cluster-head. However, if no such advertisement message is received by a node, it declares itself a cluster-head and starts to send cluster-head advertisement messages.

Fuzzy-based clustering algorithms use fuzzy logic to select the best cluster-head. These algorithms consist of four main parts: a fuzzifier, a defuzzifier, a fuzzy inference engine, and fuzzy rules. The fuzzifier converts raw input data into a set of linguistic values. These linguistic values are then used by the fuzzy engine in combination with the fuzzy rules to produce an output. The output is then converted by the de-fuzzifier to the required data format [27].

Heuristic-based algorithms try to find the best set of cluster-heads and cluster-sizes by optimizing a fitness function. Different optimization algorithms such as particle swarm optimization (PSO), genetic algorithm (GA) or Ant and bee colony are used to find the solution given a fitness function with different parameters. As stated in [27], PSO has shown better performance than other algorithms. These algorithms often require a centralized approach thus do not scale well with bigger networks. An example of a heuristic based clustering algorithm is the Genetic Clustering Algorithm (GCA) [32]. It uses a genetic algorithm (GA) to prolong the network lifetime by optimizing the number of cluster-heads and the sum of all distances between the cluster-head and its members.

Compound-based algorithms use multiple phases to achieve a reasonable clustering. An example of such an algorithm is the ACE [33] algorithm, which minimizes the number of formed clusters across the network by minimizing the overlapping area. The clustering emerges from repeated local interaction and feedback between nodes. Thus it is also called an emergent clustering protocol. It is composed of two main phases: spawning and migration. During the spawning phase, a node decides to become a cluster-head if the number of neighbor nodes that would become a member of this node is bigger than a time-dependent value. The time-dependent value decreases exponentially over time: nodes with a high number of potential cluster members declare themselves a cluster-head earlier than nodes with a lower number of potential members. In the second (migration) phase the clusters are moved to decrease the overlapping area. To do so, a cluster-head evaluates nearby nodes that would be better suitable as cluster-head and hands over the cluster-head role if it can find one.

CHAPTER 3

Proposed Localization System

The previous chapter gave an introduction to sensor networks, time synchronization, localization and clustering. In this chapter, we introduce our proposed UWB-based TDOA indoor localization system. To do so, we first review existing UWB TDOA based indoor localization systems, and we then describe the localization and communication scheme of the proposed system.

3.1 Existing work

Although the principle of TDOA systems has been well known for decades, only with the availability of the DW1000 transceivers IEEE 802.15.4 compatible, high accuracy, and low-cost indoor localization systems can be built. Because of this, there is not much literature about TDOA systems using these new transceivers. Essentially, there are currently two groups with a couple of publications dealing with the topic of wireless synchronization and TDOA localization using the DW1000 transceivers.

In [13] Ledergerber et al. describe a self-localization TDOA system, meaning that the anchor nodes broadcast synchronization and localization messages such that a robot can localize itself. Ledergerber et al. first analyze the theoretical localization accuracy of TOA and TDOA systems and propose a lightweight synchronization and localization protocol. The lightweight one-way synchronization protocol uses a reference node that broadcasts two synchronization messages with a known delay. Afterward, all other anchor nodes calculate the skew and offsets from this synchronization messages and start broadcasting localization messages containing the corrected transmission time-stamp (with respect to the reference anchor). Ledergerber et al. also propose to use a Kalman filter for clock synchronization but it remains unclear how the filter parameters are derived, and no comparison with the simple extrapolation method is conducted.

In [34] Tiemann et al. propose an indoor localization system for an unmanned aerial vehicle (UAV) in GNSS denied environments. Their system uses TWR and they calculate the position via the Taylor-Series (TS) expansion method. The position is then converted into earth-centered earth-fixed (ECEF) coordinates, which can directly be used by existing navigation systems in UAVs. They analyze the influence of the antenna characteristics on the ranging performance and found that the error is not uniform and conclude that the orientation of the antenna has a significant impact on the rangings.

In [35] a multi-user TDOA based localization system is proposed and validated. Tiemann et al. use eight anchor nodes which are synchronized to a reference node via wireless clock synchronization. To ensure a high reception rate of the synchronization packets, they use different preamble codes for synchronization and localization messages, and this method is called SyncCDMA. Contrary to the system presented in [13] their localization system is inverted and receives localization messages rather than transmitting them. For localization, they used a closed form solution to estimate the position from the acquired TDOA measurements and analyzed the localization accuracy for different synchronization rates. They found that a synchronization rate of 1Hz is sufficient to have localization errors smaller than 20cm with a probability of 90%.

In [36] Tiemann et al. improve the system introduced in [35] and changed their localization algorithm to an extended Kalman filter (EKF). This system uses a wired backbone to transmit the recorded reception time of localization packets to a localization engine written in C++, the code of the localization back-end system is published online.

The biggest limitation of the localization systems proposed by the two groups is that they do not consider larger networks. We address this issue in Chapter 6 by investigating possible clustering algorithms to form time-synchronous localization clusters. The two groups also do not investigate the synchronization accuracy but rather evaluate the localization accuracy with several anchor nodes. Another weakness of the Kalman filter proposed by Ledergerber et al. is that they did not determine the clock uncertainty of the radio modules but rather used parameters that are typical for crystal quartz oscillators. We did this in Chapter 5. Moreover, Tiemann et al. and Ledergerber et al. do not investigate the convergence of their localization algorithms inside and outside of the convex hull of their localization algorithms. In this thesis, we investigate this in Chapter 4.

3.2 Requirements

In this section, we list the requirements for our proposed localization system.

- Self-organizing: The nodes of proposed localization system should be able to form time-synchronous clusters that allow localization within its convex hull and its surrounding with high accuracy. This process should be distributed and not governed by a central entity. Two simple algorithms are compared in Chapter 6.

- High localization accuracy: The accuracy of localization within a cluster mainly depends on how accurate the nodes within a cluster are synchronized, and how the nodes are distributed. The high synchronization accuracy is achieved via statistical signal processing, as described in Chapter 5. The distribution of nodes cannot be changed, but the clustering algorithm indirectly prefers clusters where the cluster-head is at its center, and the anchor nodes are distributed around the center. We evaluated the theoretical localization accuracy of clusters established by two simple clustering algorithms in Chapter 6.
- Scalable: The system should be scalable in two ways. First, there should be no constrains on the size of the localization network. Second, ideally, an infinite number of mobile nodes can participate.
- Self-localization: Self-localization allows mobile nodes to localize themselves solely by receiving localization messages issued by the anchor nodes. As this scheme does not require mobile nodes to transmit any messages an infinite number of mobile nodes can participate.
- Passive localization: In this scheme, a mobile node only has to transmit a single message. Only one message is enough for the proposed system to localize the position of the mobile node.
- Efficient: The number of messages used to localize a node should be minimized. The passive localization scheme only requires a single message, and for the self-localization scheme data, messages are used to further reduce the number of transmitted messages.

3.3 Overview

To allow localization over a large network, it is required to separate the network into smaller localization clusters, where each cluster has its own cluster-head that is used as a time reference and executes the localization algorithms. Hence, our synchronization protocol follows the receiver-receiver synchronization approach, and it is thus similar to the RBS protocol (introduced in Section 2.4.5), but requires line of sight (LOS) condition. Two different methods to correct clock errors are described in Section 5.

The proposed indoor localization system uses an external space coordinate system, as the position of the anchor nodes has to be known beforehand, and a local time coordinate system, as each cluster has its own time reference. The system is composed of several homogeneous nodes using the DW1000 UWB transceiver and Nucleo STM32 evaluation board. Homogeneous means, that each node can act as a cluster-member or cluster-head, and thus can perform data aggregation/localization or receiving/transmitting localization messages. The role of each node (e.g., cluster-member or cluster-head) is determined by the system itself during the start-up phase and is based on a simple deterministic clustering algorithm (see Chapter 6) that is executed on each node. Essentially, each node decides to become a cluster-head, if no other cluster-head is in its communication range and the number of unclustered neighbors exceeds a time-depending threshold value.

Once the start-up phase is completed, the cluster-head assigns data communication and localization time-slots to each of the cluster members. Afterward, the cluster-members start to receive localization messages during the *passive localization* phase, and return the recorded reception time-stamps to the cluster-head at their assigned timeslots during the *intra-cluster communication and self-localization* phase. As soon as the recorded reception time-stamps are received by the cluster-head, it runs the localization algorithm to localize a mobile node within the cluster. Another way to determine the position of a mobile node is to use the self-localization feature of the proposed system. As described in the next section, the data-messages of the cluster-members contain a transmission time-stamp that can be used in combination with the reception time-stamp by mobile nodes to localize themselves. A comparison and analysis of different localization algorithms is presented in Chapter 4.

The calculated position of mobile nodes are then forwarded to any destination within the network. How and when this inter-cluster communication happens has not been addressed yet. A simple solution may be to add a phase at the end of each round where inter-cluster communication between clusters happens. Another solution could be to use every x-th round for inter-cluster communication, where x has to be determined such that all packets receive their destination in reasonable time.

In Figure 3.1b one cluster of the proposed system is depicted where the orange arrows represent synchronization messages, the blue arrows TDOA data exchange, and the black arrows are localization messages. In Figure 3.1c the messages of the proposed systems can be seen on a timeline.

3.4 Protocol timing

In Figure 3.2 the timing of the protocol can be seen in more detail. The localization system executed periodically, each round is composed out of a *sync* phase, *intra-cluster communication and self-localization* phase as well as a *passive* localization phase. In the *sync* phase the cluster-heads broadcast synchronization messages that are used by the cluster members to synchronize their clocks and establish a common time-base. During the *intra-cluster communication and self-localization* phase all anchor nodes return the corrected reception time-stamps of all previously received localization packets to their cluster-head. As they send these messages at defined time-slots, a mobile node can use these messages to perform self-localization. During the *passive* localization phase all cluster members are listening to localization messages issued by mobile nodes, the recorded reception time-stamps are returned in the next round during the intra-cluster communication phase.

The reason for putting the intra-cluster communication before and not after the passive localization phase is that clock synchronization is higher at the beginning of each round

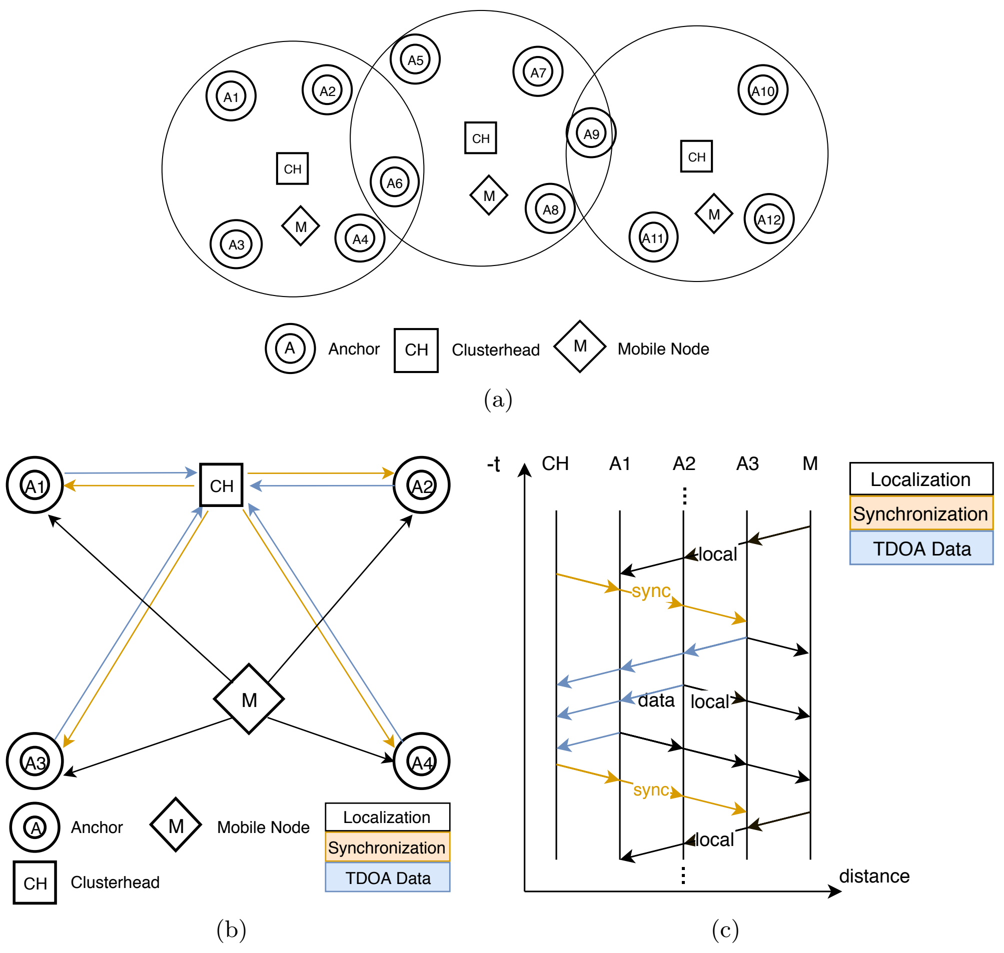


Figure 3.1: 3.1a shows a sensor network of 15 nodes, where 3 nodes are cluster-heads and the other nodes are cluster-members (anchor nodes). Two nodes (A9, and A6) are in the range of two cluster-heads, thus they can be used as gateway nodes. The mobile nodes are localized within each cluster. Additionally, there are three mobile nodes. 3.1b shows an illustration of one cluster of our proposed localization. In 3.1c the exchanged messages are depicted.

than at the end. As self-localization can support an arbitrary number of mobile nodes, all nodes that use self-localization benefit from a higher localization accuracy.

3.5 Conclusion

The proposed localization system can organize a network of many nodes into localization clusters via a deterministic clustering algorithm. Each cluster-head is used as a time-reference and performs localization within the range of the cluster. This can be seen as

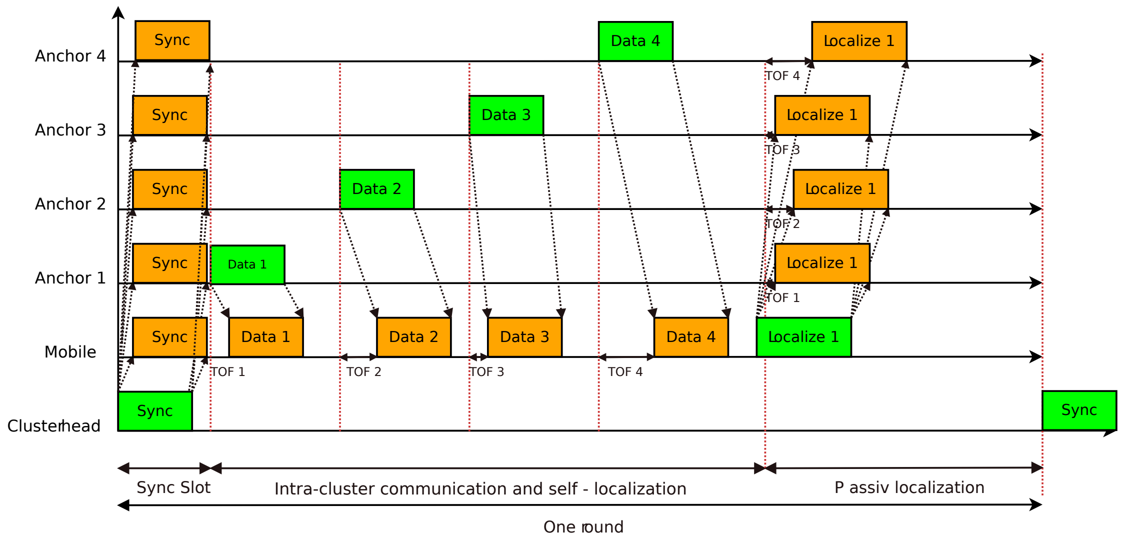


Figure 3.2: Illustration of the protocol timing, green means that a message is transmitted and orange that a message is received. The protocol works periodically. Each period is divided into the *Sync* phase, *intra-cluster communication and self-localization* phase, and *passive* localization phase. In the *Sync* phase all member nodes of a cluster listen for synchronization massages. During the *intra-cluster communication and self-localization* the recorded reception time-timestamps of the *passive* localization phase are returned to the cluster-head. During this phase mobile nodes can also localize themself by using the data messages as localization messages. During the *passive* localization phase mobile nodes can send localization packets, the reception times of this packets are recorded at each anchor and returned in the *intra-cluster communication and self-localization* phase to the cluster-head

data aggregation such that only the final position may be forwarded to a specific node in the network. The localization and communication scheme allows for self-localization as well as passive localization of mobile nodes. To the author's knowledge, this system is the first low-cost TDOA system that combines both schemes in the described way. As nodes are homogeneous, each node can substitute an erroneous cluster-head, which makes the system more resilient.

In the next chapters, the individual building blocks of the localization system are investigated in more depth. In detail, we describe possible localization algorithms in Chapter 4 and investigate their performance, two synchronization methods are compared in Chapter 5, and two clustering algorithms are simulated in Chapter 6 to investigate if they are able to establish a good clustering for localization.

CHAPTER 4

Localization algorithms

In this chapter localization via TOA and TDOA is investigated in more depth. The reason why both methods are described is that most literature and books are describing the TOA case and its localization methods. The intention was to show the similarities between TOA and TDOA, and that most TOA algorithms can solve the TDOA case with only slight adjustments. At the end of this section, three different localization algorithms for the TDOA case are compared, and their performance is evaluated.

4.1 Overview of TOA and TDOA

Trilateration is the process of calculating the position given a set of distances (TOA), or differences of distance measurements (TDOA). To uniquely determine the position of a mobile node with TOA distance measurements to at least three anchor nodes are required, in the three-dimensional space, at least four anchors are required [16]. In [37] several methods that can be employed to solve the trilateration problem are compared and analyzed. Some prominent ones are the analytical method (AM), the least squares method (LS), the Taylor series method (TS), and the Gauss-Newton (GN) method. While the analytical method is the simplest one it does not allow to use more measurements than undefined variables, hence, additional anchors cannot contribute to higher accuracy. The LS method allows to solve an overdetermined equation system; hence, additional measurements can improve the accuracy. However, as TOA, and TDOA yields a set of non-linear equations they first must be linearised, this results in a linearized least square (LLS) system. The TS estimation method is an iterative solution, and the equation system is first linearised around an initial estimated position and then used to update the estimated location iteratively.

Given the anchor nodes $i = 1...n$ with position $\mathbf{p}_i = (x_i, y_i)$, the location of the mobile node $\mathbf{x} = (x, y)$, the relationship to the distance r_i is expressed by equation 4.1 for the TOA case [16] [38].

$$\begin{aligned} &\mathbf{f}_{TOA}(x) = \mathbf{r}_{TOA} \\ &\mathbf{f}_{TOA}(x) = \begin{bmatrix} (x_1 - x)^2 + (y_1 - y)^2 \\ ... \\ (x_n - x)^2 + (y_n - y)^2 \end{bmatrix}, \mathbf{r}_{TOA} = \begin{bmatrix} r_1^2 \\ ... \\ r_n^2 \end{bmatrix} \end{aligned} \tag{4.1}$$

For the TDOA case the relationship is a little bit different. As can be seen in equation 4.2 the right hand side of the equation systems does not contain the distances r_i but rather the difference of distances m_i. In this example the difference is always with respect to anchor 1 but it can be any anchor node.

$$\begin{aligned} &\boldsymbol{f}_{TDOA}(x) = \boldsymbol{r}_{TDOA} \\ &\boldsymbol{f}_{TDOA}(x) = \begin{bmatrix} \sqrt{(x_2 - x)^2 + (y_2 - y)^2} - \sqrt{(x_1 - x)^2 + (y_1 - y)^2} \\ ... \\ \sqrt{(x_n - x)^2 + (y_n - y)^2} - \sqrt{(x_1 - x)^2 + (y_1 - y)^2} \end{bmatrix} \\ &\boldsymbol{r}_{TDOA} = \begin{bmatrix} m_{2,1} \\ ... \\ m_{n,1} \end{bmatrix}, m_{i,1} = r_i - r_1 \end{aligned} \tag{4.2}$$

4.2 Linearized Least squares (LLS)

Since there are two unknown parameters, namely x and y the system can be solved in closed form if there are only three independent equations (three distance measurements). In many situations, there are more anchor nodes available. Thus more than three measurements are available leading to an overdetermined system. In this case, we use the least squares (LS) technique to obtain and improve the location estimate [39]. In order to apply the LS technique the equation systems 4.1 and 4.2 have to be linearised (by arithmetic), thus the set of non-linear equations is transformed into a set of linear equations [16] [40].

In the case of TOA measurements, the estimated position is at the intersection of the circles of the TOA measurements, as can be seen in Figure 2.2a. To find a solution, i.e., the position of the mobile node, the relationship between measured distances and positions of the anchor nodes must be solved. The TOA system,in Equation 4.1, can be easily linearised by expanding the polynomials and subtracting equation one from all other equations, this removes the squared unknown (x^2 and y^2) from the equation system [39]. After some rearranging the following linear system is found:

$$2\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.3}$$

with $\mathbf{A}$ defined as

$$\mathbf{A} = \begin{bmatrix} (x_n - x_1) & (y_n - y_1) \\ ... & \\ (x_n - x_{n-1}) & (y_n - y_{n-1}) \end{bmatrix} \tag{4.4}$$

and $\mathbf{b}$

$$\mathbf{b} = \begin{bmatrix} r_1^2 - r_n^2 + k_n - k_1 \\ ... \\ r_{n-1}^2 - r_n^2 + k_n - k_{n-1} \end{bmatrix} \tag{4.5}$$

$$k_n = x_n^2 + y_n^2 \tag{4.6}$$

The estimated position of the mobile node $\hat{\mathbf{x}} = (\hat{x}, \hat{y})$ can then be calculated by using [16] [37] [38]:

$$\hat{\mathbf{x}} = \frac{1}{2}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{4.7}$$

Finding a closed form solution for the TDOA case is a little bit more complicated than for the TOA case. Specifically, we have to use the approach from [41], where one anchor (anchor 1) serves as a reference point and the whole system is transformed in such a way that the position of the reference anchor is mapped to the origin of the 3D Cartesian coordinate system [39] [37]. For this method, we have to start with an equation system that is similar to the one of the TOA system (4.1), but as can be seen in equation 4.8 the distance measurements of anchors $n > 1$ are transformed with respect to the position of anchor 1. Moreover, we will see later the solution $\mathbf{t}$ of the LS system is a solution with respect to the position of anchor 1.

$$\begin{bmatrix} (x_1 - x)^2 + (y_1 - y)^2 \\ (x_2 - x)^2 + (y_2 - y)^2 \\ ... \\ (x_n - x)^2 + (y_n - y)^2 \end{bmatrix} = \begin{bmatrix} r_1^2 \\ (r_1 + m_{2,1})^2 \\ ... \\ (r_1 + m_{n,1})^2 \end{bmatrix} \tag{4.8}$$

As in the TOA case the polynomials have to be expanded and the first equation has to be subtracted from equation $n > 1$. After some rearranging as described in [37],[39], the solution is:

$$\mathbf{t} = 1/2(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T(\boldsymbol{c} + 2r_1\boldsymbol{d}) \tag{4.9}$$

with

$$\boldsymbol{c} = \begin{bmatrix} k_2' - m_{2,1} \\ ... \\ k_n' - m_{n,1} \end{bmatrix} \mathbf{d} = \begin{bmatrix} -m_{2,1} \\ ... \\ -m_{n,1} \end{bmatrix} \boldsymbol{t} = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x - x_1 \\ y - y_1 \end{bmatrix} \tag{4.10}$$
$$x_i' = x_i - x_1, y_i' = y_i - y_1, k_i' = x_i'^2 + y_i'^2$$

As can be seen in equation 4.9 the solution depends on distance variable r_1. In order to arrive at a final solution this value has to be determined as described in [39]:

$$r_1 = \frac{\boldsymbol{d}^T \boldsymbol{B}\boldsymbol{B}\boldsymbol{c}}{2\boldsymbol{d}^T \boldsymbol{B}\boldsymbol{B}\boldsymbol{d}}, \boldsymbol{B} = \boldsymbol{I} - \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T \tag{4.11}$$

In reality, the position of the anchor nodes $\mathbf{x}_n = (x_n, y_n)$ cannot be determined with absolute accuracy, and also the measurements r_n are subject to noise. For this reason and because we loose information about the system during the linearization process, this method is not as accurate as iterative methods. However, the linearized equation systems cost function is unimodal meaning that the LS technique can find a globally optimal solution. This is important as iterative methods cannot guarantee convergence towards the optimal global solution if the initial guess is not set correctly [11].

4.3 Iterative solutions

There are several iterative methods such as the Taylor-Series-Least-Squares (TS-LS), Gauss-Newton (GN), or the Maximum-Likelihood Gauss-Newton (ML-GN) method. The idea of these algorithms is similar: first, an initial guess of the position is obtained by a non-iterative method. Second, this initial position is updated iteratively until some error threshold is reached. These algorithms can be applied either to static or moving objects [39] [11].

In this section we focused on the TS method, as it is easy to implement. To be able to have comparable results we also implement the Gauss-Newton, and Maximum-Likelihood Gauss-Newton (ML-GN) method, but do not explain in detail how they work.

4.3.1 Iterative optimization

For this method the non-linear equation 4.1 can be directly used to find a suitable solution. For this an error function has to be defined:

$$J_{NLS,TOA} = (\mathbf{r}_{TOA} - \mathbf{f}_{TOA}(\tilde{\mathbf{x}}))^T (\mathbf{r}_{TOA} - \mathbf{f}_{TOA}(\tilde{\mathbf{x}})) \tag{4.12}$$

$$\hat{\mathbf{x}} = \arg\min_{\tilde{\mathbf{x}}} J_{NLS,TOA}(\tilde{\mathbf{x}}) \tag{4.13}$$

Methods like Newton-Raphson, Gauss-Newton or steepest descent can be used to find a suitable value $\tilde{\mathbf{x}}$ such that the corresponding value of the error function $J_{NLS,TOA}(\tilde{\mathbf{x}})$ is minimized [11]. Given a known measurement error distribution, the Maximum-Likelihood approach maximizes the probability that the observed distance measurements match the estimated position. It can be shown that the ML estimator is equivalent to minimize the following error function:

$$J_{ML,TOA} = (\mathbf{r}_{TOA} - \mathbf{f}_{TOA}(\tilde{\mathbf{x}}))^T \mathbf{C}_{TOA}^{-1} (\mathbf{r}_{TOA} - \mathbf{f}_{TOA}(\tilde{\mathbf{x}})) \tag{4.14}$$

where $\mathbf{C}_{TOA}$ is the Gaussian-distributed error covariance matrix.

$$\hat{\mathbf{x}} = \arg\min_{\tilde{\mathbf{x}}} J_{ML,TOA}(\tilde{\mathbf{x}}) \tag{4.15}$$

When $\mathbf{C}_{TOA}^{-1}$ is proportional to the identity matrix, then the ML method reduces to the NLS method.

For the TDOA case, the cost function $J_{NLS,TDO}$ has to be replaced with:

$$J_{NLS,TDOA} = (\boldsymbol{r}_{TDOA} - \boldsymbol{f}_{TDOA}(\tilde{\boldsymbol{x}}))^T(\boldsymbol{r}_{TDOA} - \boldsymbol{f}_{TDOA}(\tilde{\boldsymbol{x}})) \tag{4.16}$$

and

$$J_{ML,TOA} = (\mathbf{r}_{TDOA} - \mathbf{f}_{TDOA}(\tilde{\mathbf{x}}))^T\mathbf{C}_{TDOA}^{-1}(\mathbf{r}_{TDOA} - \mathbf{f}_{TDOA}(\tilde{\mathbf{x}})) \tag{4.17}$$

where $\mathbf{C}_{TDOA}$ is the noise covariance matrix of the TDOA system.

4.3.2 Taylor-Series Expansion

In this method, a set of non-linear equations is linearised around a given initially estimated point by a Taylor series expansion. The new equation system is a LLS problem, and its solution can be easily found. Afterward, the new solution (estimated position) is used to update the initial guess. These steps are iteratively applied until the system converges to the estimated point.

First we have to define a function $f(x, y)$ that can describe range measurements.

$$f_i(x, y) = \sqrt{(x - x_i)^2 + (y - y_i)^2} = r_i \tag{4.18}$$

where r_i are the range measurements.

For the TDOA case f_i is adjusted and looks as follows:

$$f_i(x, y) = \sqrt{(x - x_{i+1})^2 + (y - y_{i+1})^2} - \sqrt{(x - x_1)^2 + (y - y_1)^2} = m_{i,1} \tag{4.19}$$

where $m_{i,1}$ are the range difference measurements with respect to anchor 1.

Given an initial position estimation

$$\mathbf{x}_v = [x_v, y_v]^T \tag{4.20}$$

the true position $\mathbf{x}$ is at position $x = x_v + \delta_x$ and $y = y_v + \delta_y$. The position estimation error is denoted by

$$\boldsymbol{\delta} = \mathbf{x}_v - \mathbf{x} = [\delta_x, \delta_y]^T \tag{4.21}$$

where $\mathbf{x}$ is the estimated position.

Expanding equation 4.18 by a Taylor series with only the first two terms leads to

$$f_{i,v} + a_{i,1}\delta_x + a_{i,2}\delta_y = r_i \tag{4.22}$$

where

$$\begin{aligned} f_{i,v} &= f_i(x_v, y_v), \\ a_{i,1} &= \frac{\partial f_i}{\partial x}\bigg|_{x_v,y_v} = \frac{x_v - x_i}{r_i}, \\ a_{i,2} &= \frac{\partial f_i}{\partial y}\bigg|_{x_v,y_v} = \frac{y_v - y_i}{r_i} \end{aligned} \tag{4.23}$$

For the TDOA system we have to use $f_{i,v}$ defined in Equation 4.19. $a_{i,1}$ and $a_{i,2}$ change accordingly:

$$\begin{aligned} &f_{i,v} = f_i(x_v, y_v), \\ &f_{i,v} + a_{i,1}\delta_x + a_{i,2}\delta_y = m_{i,1} \\ &a_{i,1} = \frac{\partial f_i}{\partial x}\bigg|_{x_v,y_v} = \frac{x_1 - x_v}{r_i} - \frac{x_{i+1} - x_v}{r_{i+1}}, \\ &a_{i,2} = \frac{\partial f_i}{\partial y}\bigg|_{x_v,y_v} = \frac{y_1 - y_v}{r_i} - \frac{x_{i+1} - x_v}{r_{i+1}} \end{aligned} \tag{4.24}$$

Equation 4.22 can be rewritten in Matrix form as

$$\mathbf{A}\boldsymbol{\delta} = \boldsymbol{D} \tag{4.25}$$

where

$$\begin{aligned} \mathbf{A}_{i,j} =& a_{i,j} \\ \mathbf{D} =& [r_1 - f_{1,v}, r_2 - f_{2,v}, ..., r_n - f_{n,v}]^T \end{aligned} \tag{4.26}$$

For the TDOA case with $f_{i,v}$ from Equation 4.19 is defined as:

$$\boldsymbol{D} = [m_{2,1} - f_{1,v}, m_{3,1} - f_{2,v}, ..., m_{n,1} - f_{n,v}]^T \tag{4.27}$$

The least square estimator of 4.25 is

$$\boldsymbol{\delta} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{D} \tag{4.28}$$

The initial position estimation $\mathbf{x}_v$ can be updated accordingly: $x_v = x_v + \delta_x$, $y_v = y_v + \delta_y$. These steps are performed iteratively until $\mathbf{x}_v$ converges to the true position. Similar to the ML-GN estimator also the TS-LS estimator can be extended to incorporate the noise covariance matrix to obtain a weighted TS-LS estimator. The covariance matrix $\mathbf{Q}$ is defined as

$$\mathbf{Q} = E[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T] = diag(\sigma^2, ...\sigma^2) \tag{4.29}$$

in the case of zero-mean Gaussian random variables this results in a diagonal matrix. Considering the noise covariance matrix equation 4.28 changes to:

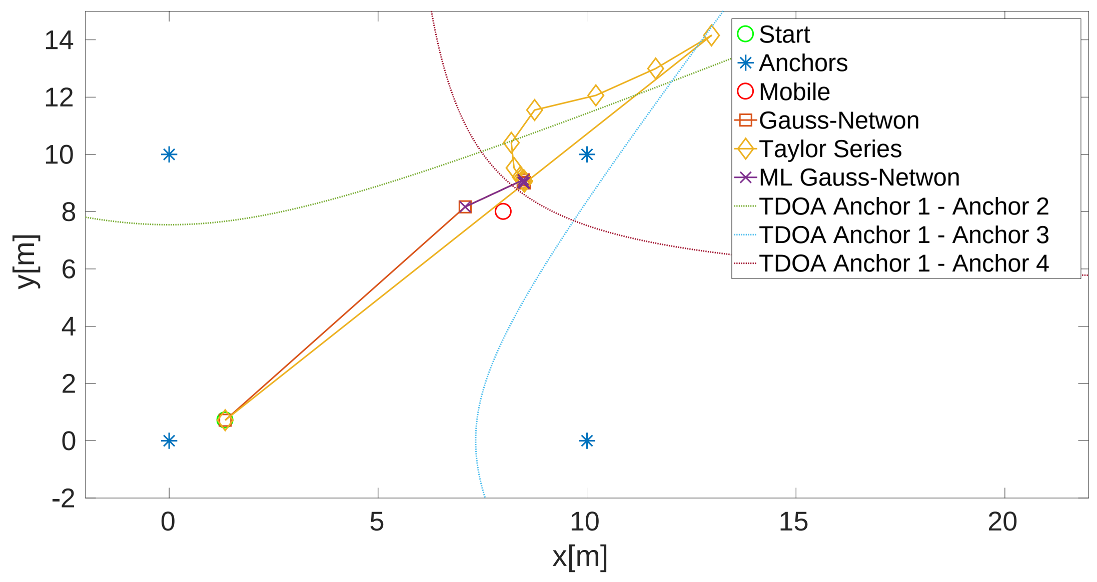
$$\boldsymbol{\delta} = (\boldsymbol{A}^T\boldsymbol{Q}^{-1}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{Q}^{-1}\boldsymbol{D} \tag{4.30}$$

4.4 Performance evaluation

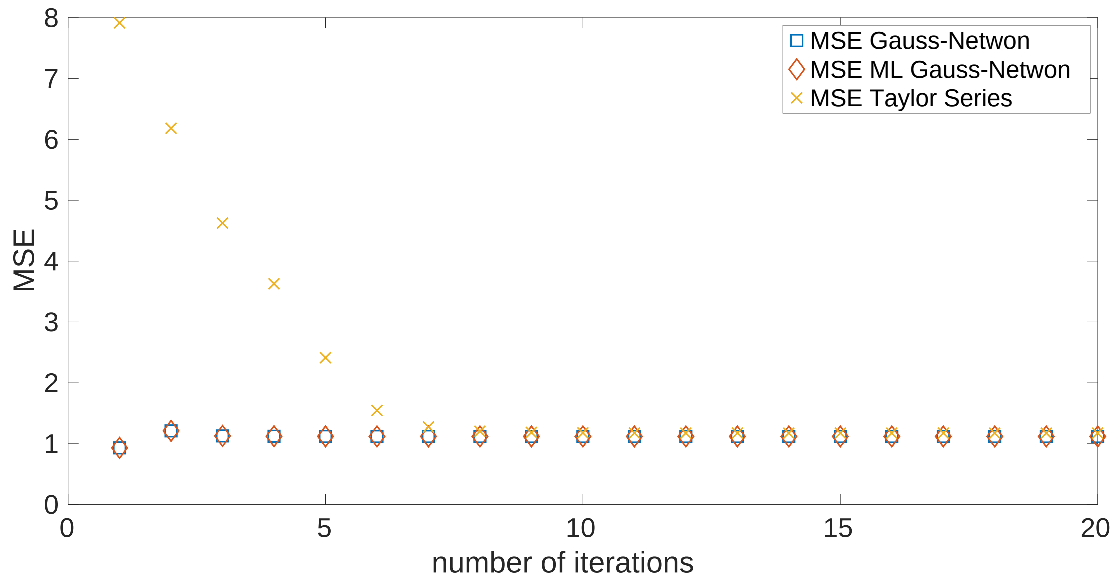
To be able to compare the performance of different position estimation methods we implemented the Taylor-Series expansion method, Gauss-Newton, ML Gauss-Newton, and the LLS method in Matlab for the TDOA case. As a starting point, we took the TOA example of the Gauss-Newton and ML Gauss-Newton from [11].

In Figure 4.1a we can see a simulated localization setup. The four blue asterisk symbols are the anchor nodes, the green circle is the initial position guess and the red circle is the position of the mobile node. The colored dotted lines are the TDOA hyperbolas of anchor two, three and four with respect to anchor 1 (left bottom anchor). We added normally distributed noise with standard deviation of 1 to each of the TDOA measurements; thus the TDOA lines do not intersect at a single point. The solid lines with the $\diamond$ symbol, the $\circ$ symbol, and $\square$ symbol represent one step of one of the localization algorithms, respectively. As it can be seen all algorithms end up at the same point but the Taylor-Series expansion takes considerably more steps. This issue can be further seen in Figure 4.1b, where the RMSE error of the estimated position is plotted for each algorithm after each iteration.

The bad performance of the TS method in Figure 4.1b and 4.1a seems to be a worst-case scenario that happens if the initial guess is bad. In Figure 4.2a, and 4.2b the initial guess was chosen close to the final position. In this case all algorithms converge at almost the same rate.

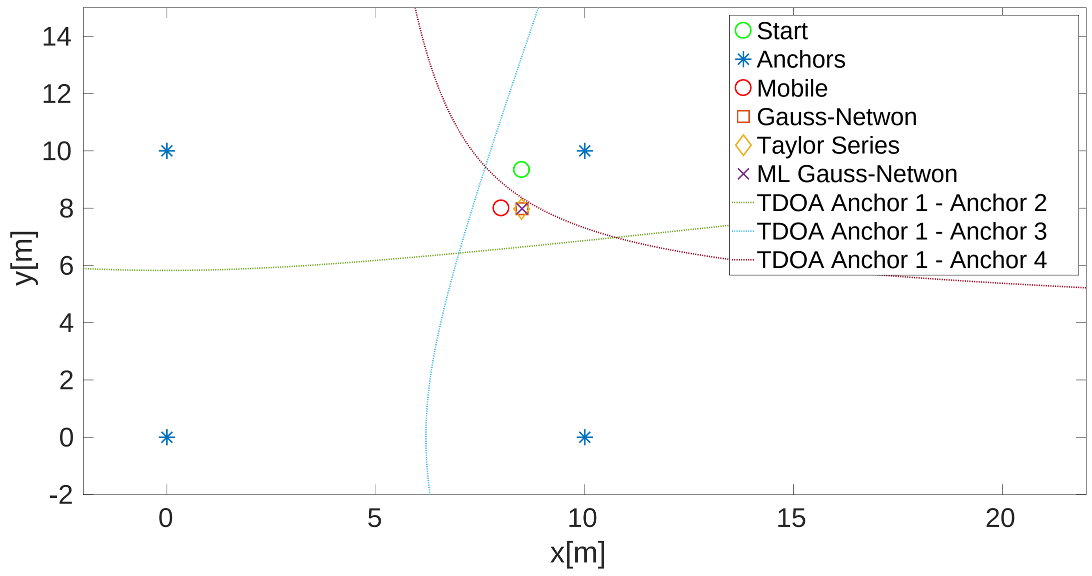


(a)

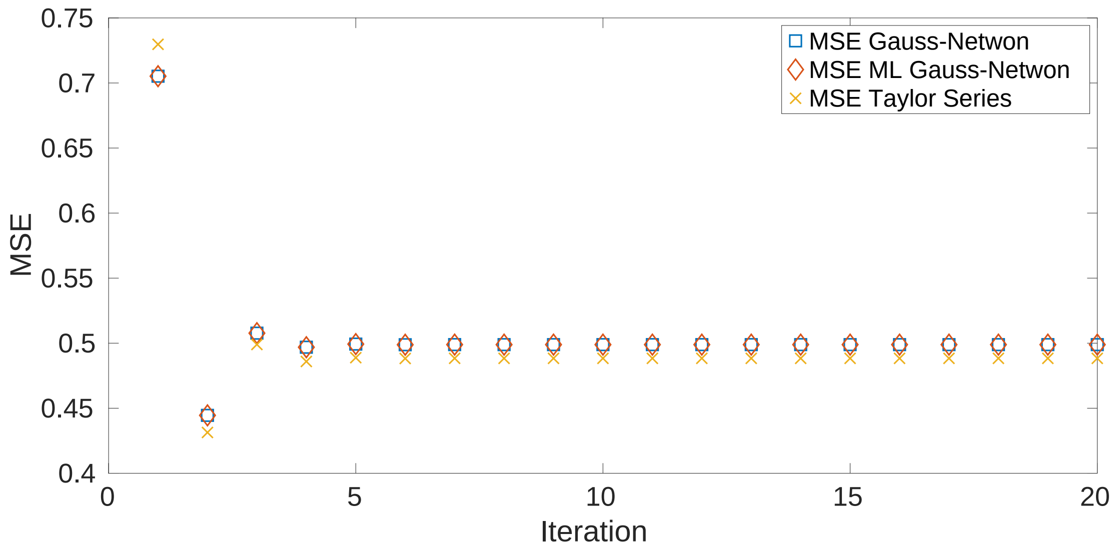


(b)

Figure 4.1: Simulation of different localization algorithms with a bad initial position guess. In this case the TS method takes considerably longer than GN and ML-GN to converge.



(a)



(b)

Figure 4.2: Simulation of different localization algorithms with an initial guess close to the final position. In this case the convergence performance and accuracy of the TS algorithm is almost equal to the other algorithms.

To further investigate the localization accuracy and the convergence of the iterative algorithms we simulated the localization process 2000 times, where the position of the mobile point is anywhere between x=[0, 10] and y=[0, 10]. The TDOA measurements were modified with Gaussian distributed noise with a standard deviation of 0.1m. We then compare the mean RMSE error of each algorithm in each iteration. As the ML Gauss-Newton estimator is the same as the Gauss-Newton estimator under equal measurement noise, we omit the simulation of this estimator and instead simulated the linearised Least Squares (LLS) estimator. As single simulations, where an algorithm does not converge influences the RMSE error greatly, therefore we decided to omit simulations where the RMSE error in the last iteration was greater than 10 meters. If that happens, we consider it not to converge.

The RMSE of each localization algorithm in each iteration was calculated with:

$$RMSE = \sqrt{\frac{1}{2L}\sum_{i=1}^{L}(\hat{x}_i - x)^{+}(\hat{y}_i - y)^2} \tag{4.31}$$

where L is the number of simulations. The mean CRLB value was calculated via

$$MeanCRLB = \sqrt{\frac{1}{2L}\sum_{i=1}^{L}(var(x_i) + var(y_i))} \tag{4.32}$$

where $var(x_i)$ and $var(y_i)$ are the variance of the x and y position of the mobile node in the i-th iteration. These values were calculated via the method described in [13].

We did this simulation with either a fixed starting point at the middle of the convex hull ($x_v = 5, y_v = 5$) (Figures 4.3a, 4.4a, 4.5a), and the estimated position of the LLS solution (Figures 4.3b, 4.4b, 4.5b). Moreover, we repeated the simulations with the position of the mobile node constrained to the inside of the convex hull and outside of the convex hull (Figures 4.4 and 4.5).

In Figure 4.3a we can see the mean error with the starting point at the center and in Figure 4.3b with the starting point at the estimated position of the LLS estimator, for both plots the positions of the mobile node were constrained to the inside of the convex hull. During the 2000 simulations the TS method did not converge once, the GN method two times and the LSS method delivered 38 results where the final position exceeded the 10 meter threshold.

We repeated the same simulations, but now the position of the mobile node was constrained the outside of the convex hull ($x = [10, 15], y = [10, 15]$). In Figure 4.4a we can see the RMSE with the initial point set to $x_v = 5, y_v = 5$ and in Figure 4.4b where the initial guess is taken from the LLS solution. For both simulations, the TS method did not converge two times, but GN and LLS deliver a reasonable estimation every time. This changed when we increased the possible location outside of the convex hull to $x = [10, 20]$, and $y = [10, 20]$. During these simulations, we could observe that the TS method did not converge 64 times, the GN 12 times and the LLS method did not deliver reasonable results five times. The results were even worse for the TS method if the initial point was set to the middle of the convex hull ($x_v = 5, y_v = 5$). In this case, the TS method did not converge 700 times, the GN method 13 times, and the LLS did deliver wrong results four times.

The plots can be interpreted as follows: The LLS estimator can deliver good initial estimates, but does not always find the right solution (especially outside of the convex hull) and is not as accurate as the TS or GN methods. The GN method seems to converge faster than the TS method (Figure 4.3a) , however, given a good initial estimation also the TS method converges fast, as shown in Figure 4.3b. If the mobile node is inside of the convex hull, the accuracy of the iterative methods is higher than the LLS method, but outside of the convex hull, the LLS method is even better. This seems counter-intuitive as the iterative methods are supposed to yield better results; future has to be done on this issue.

A detailed explanation why the LSS method may not always deliver good results under the presence of noise can be found in [11].

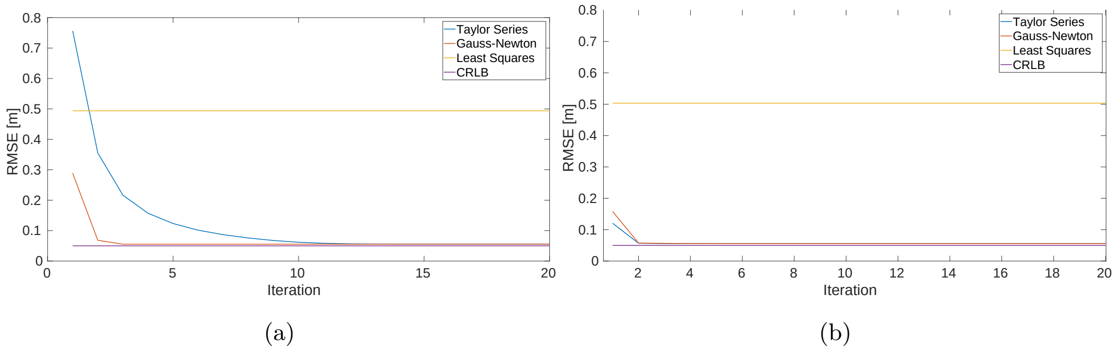


Figure 4.3: MSE error of the estimated position in each iteration with the position of the mobile node constrained to be inside of the convex hull. Figure 4.3a with initial guess at x=5,y=5. Figure 4.3b with the initial guess obtained by the LLS solution.

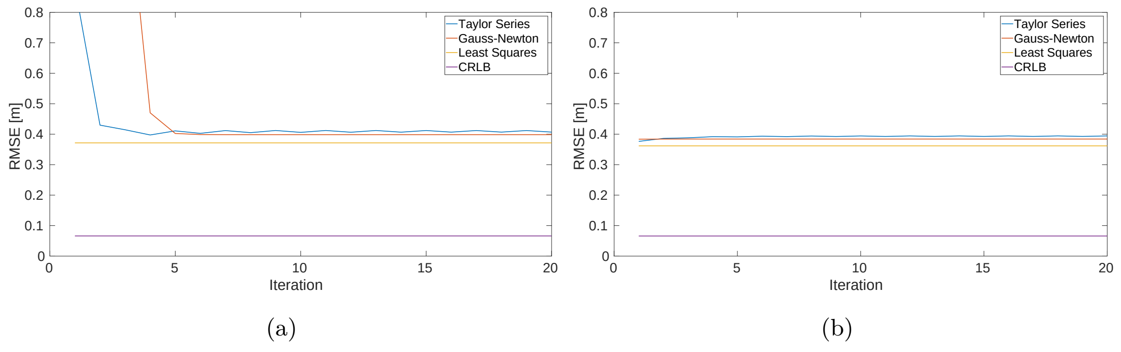


Figure 4.4: MSE error of the estimated position in each iteration with the position of the mobile node constrained to be outside of the convex hull. Figure 4.4a with initial guess at x=5,y=5. Figure 4.4b with the initial guess obtained by the LLS solution.

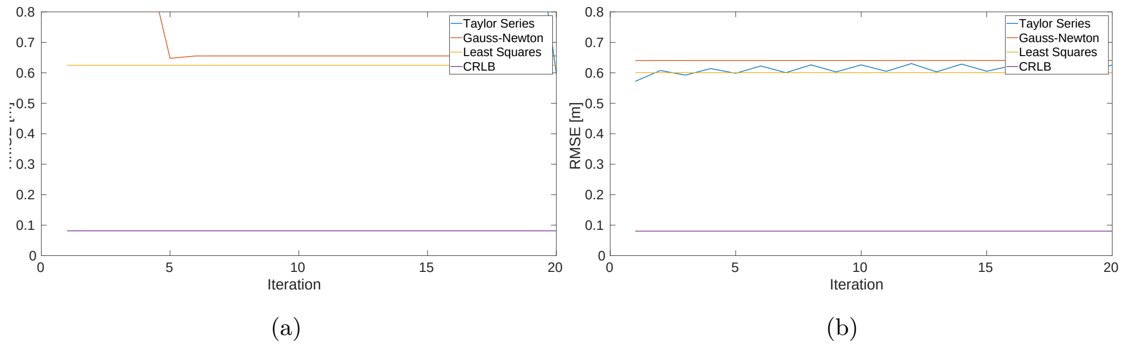


Figure 4.5: MSE error of the estimated position in each iteration with the position of the mobile node constrained to be further outside of the convex hull. Figure 4.5a with initial guess at x=5,y=5. Figure 4.5b with the initial guess obtained by the LLS solution.

4.5 Conclusion

In this chapter we evaluated three prominent TDOA localization methods, for this we analyzed the convergence rate and accuracy of one closed form and two iterative solutions and compared them with the theoretical lower bound. In general, the convergence of both iterative algorithms depends on a good initial estimate, but the GN method seems to be more robust if the initial estimate is wrong. The LSS method can be used to find a good initial guess, but as described in [11] also this method can fail under the presence of noise, thus, may produce bad initial estimates. If this happens both iterative methods may not converge. Adding a fifth anchor node seems to solve this problem as in this case all methods converge. To measure the performance of each algorithm we measured the execution time of one iteration of each algorithm on a notebook with an Intel Core i7 3517U processor with 1,9GHz. The TS method is the fastest and only took 0.06ms, followed by the GN method which took 0.15ms to complete, the lowest method is the LLS method which took 0.24ms to finish. However, as the iterative methods require several iterations, the LSS is the fastest. The biggest issue that we could identify is that both iterative methods may not converge if the initial guess is wrong, and also the LSS method does not always deliver a good initial guess. A solution might be to calculate the LLS solution with respect to different reference anchors and chose a plausible solution. Another issue that we could observe is that the iterative methods do not approach the CRLB if the mobile node is outside of the convex hull. Contrary to our implementation, in [12] Kaune et al. take the full measurement set to calculate the position. That is, each TDOA measurement is not only taken with respect to one anchor but with respect to all other anchor. This is one explanation why the results of our simulations are different to the results presented in [12].

CHAPTER 5

Synchronization

In this section, we present and evaluate the clock synchronization problem in more detail. To do so, we first present relevant literature which is used to derive an iterative and a stochastic differential equation (SDE) model of crystal clocks. After the model is derived, we show how to obtain the noise parameters from subsequent time offset measurements. These parameters are then used to parameterize an iterative filter, which is then evaluated offline as well as online one the STM32Nucleo platform.

5.0.1 Overview

In [42] the authors describe a distributed algorithm that can jointly estimate the clock offset as well as to perform localization. The proposed algorithm takes the noisy nature of TOA measurements into account and thus uses a probabilistic approach. The diffusion or averaging algorithm ensures that each clock converges to a global reference.

In [43] Zucca et al. show how to model atomic clocks with three-dimensional stochastic differential (SDE) equations, Zucca et al. further state that this model can also be applied to other types of clocks. Based on this model Zucca et al. derive a closed form and an iterative solution of the SDE equation system, the iterative solution can be seen as a state space representation of the clock and can be used to simulate clocks under the presence of noise. Based on the iterative solution Zucca et al. analyze how the noise propagates through the system and derive a link to the Allen variance, this link can be exploited to perform model parameter identification, i.e., to deduce the process noise parameter of a real system based on subsequent time-stamping measurements. This relationship is essential as optimal filtering is only possible if one knows how the noise influences the reported time.

The goal of [44] is to evaluate how clock accuracy, message exchange rate, and time-stamping accuracy influences the synchronization accuracy in networks employing the Precision Time Protocol (PTP). Giorg et al. first introduce an ideal sine wave oscillator.

Consequently, this oscillator is then used to derive an iterative representation of the clock. Giorg et al. then introduce Gaussian noise to the iterative clock model to model a realistic evolution of the clock over time, this ultimately leads to the idea to use a Kalman filter as state estimator. Afterward, the results from [44] are used to show how the required parameters, i.e., the process noise of the iterative model, can be determined. Giorg at a. then evaluate their method through simulation: The iterative model is used to simulate the evolution of the clock under the presence of process noise. The clock values are then corrected with the offset calculated by Kalman filter and compare the corrected values to the ones of the unregulated clocks. Giorg et al. also investigate how the time-stamping uncertainty influences the proposed synchronization method and conclude: If the time-stamping accuracy increases, also the clock offset error decreases to a certain limit that depends on the intrinsic stability of the unregulated clock, i.e., the process noise. In other words, synchronization of clocks below a certain threshold is not possible. Then again if the time-stamping uncertainty is increased the usage of accurate clocks cannot improve the synchronization accuracy. The conclusion is that both, the clocks process noise and the time-stamping accuracy must meet certain criteria to yield reasonable synchronization accuracy.

In [20] Wu et al. quickly analyze different synchronization protocols and their underlying nature, i.e., single message, multi-message as well as receiver - receiver synchronization. The authors further discuss the simple least square (LS) method for clock skew and offset estimation, and conclude that this method yields good results if the underlying distribution of message delay is Gaussian distributed. In the second part, the authors discuss more advanced statistical filtering methods such as the maximum likelihood (ML), linear programming (LP) and composite particle filters (CPF). The idea behind the CPF approach is to model the clock in the state space representation and to make usage of the Bayesian framework for state estimation. The authors state that if the problem is linear and with Gaussian noise, the Kalman filter provides the solution to the Bayesian framework. If the noise is not Gaussian, they take the approach from [45] to approach the non-Gaussian noise with a bank of Kalman filters.

5.1 Clock Model

Frequency or time deviation in clocks can be categorized into systematic deviations and random deviations. Systematic deviations originates from variations during the manufacturing process, temperature control, aging, or in the case of time deviation simply the delay when two clocks were switched on. Random deviations originate from random perturbation by thermo-mechanical noise in the electrical circuit or crystal structure, resulting in random fluctuation of phase and frequency. The random frequency fluctuation accumulates over time. Hence, it can be described as a random-walk model as the instantaneous time deviation depends upon the sum of all previous random steps.

In Figure 5.2 we see different types of errors that all accumulate to the instantaneous clock error between a reference and local time. The blue line denotes the true time.

The cyan line is the phase jitter or phase noise, which is responsible for random timing errors in the reported time, this error is categorized as random error. The red line is the skew and is caused by inaccuracy of the clock frequency, which is categorized as systematic error. The green line is the offset which is caused by different start-up time of two clocks and is also categorized as systematic error. Despite the fact that the skew is categorized as systematic it does also contain some randomness as it can vary over time due environmental influences (e.g., temperature change) and as previously mentioned random frequency changes. However, in most cases, the drift is assumed to be middle/long-term stable.

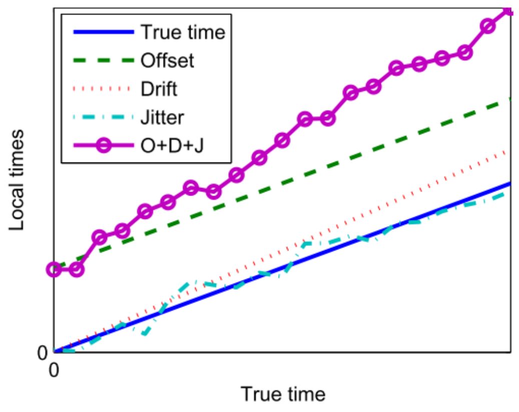


Figure 5.1: Clock errors. Taken from [47]

5.1.1 Sin Generator Model

The purpose of the following section is to derive an iterative state model of a clock from the *sin* model presented in equation 5.1. In Section 5.1.4 a more advanced model of the clock is presented.

The *sin* output of a frequency source can be modelled by equation 5.1 [46]. As can be seen, the correctness of the clock output depends upon the random process $\phi(t)$.

$$\begin{aligned} V(t) &= [V_0 + \epsilon(t)] sin(2\pi[f_0 t + \frac{\phi(t)}{2\pi}]) \\ \textbf{where} \\ f_0 &= \text{nominal frequency} \\ \phi(t) &= \text{phase deviation} \\ V_0 &= \text{nominal peak output} \\ \epsilon(t) &= \text{amplitude deviation} \end{aligned} \tag{5.1}$$

For the rest of this chapter we are ignoring the amplitude deviation $\epsilon(t)$.

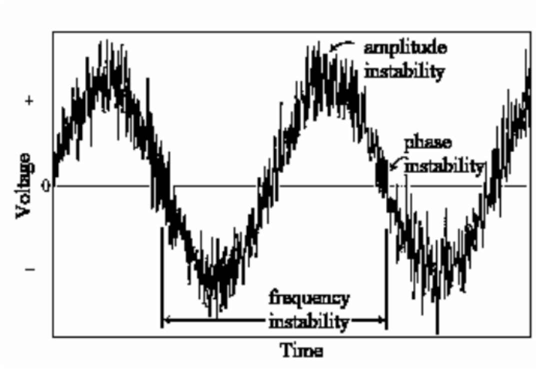


Figure 5.2: An illustration of the frequency, phase and amplitude instability of the sin clock model. The frequency instability is a result of the fluctuations of the period of an oscillator. The phase fluctuations is due to instability of the zero crossing. The fluctuations in the peak value results in amplitude instability. Taken from [46].

Let us consider the ideal time T_n and time $C(T_n)$ which is recorded via a non-ideal clock, both time stamps are recorded at event k_n. The difference of these two time-stamps is denoted by the instantaneous time offset $\theta(T_n)$:

$$\Theta(T_n) = C(T_n) - T_n \tag{5.2}$$

Next we will discover where this offset comes from. In the case of an ideal clock, equation 5.3 is satisfied, where k is an integer value. In real clocks, however, the occurrence of clock ticks is disturbed by a time depending $\phi(t)$ as shown in equation 5.4.

$$f_0 T_n + \frac{\phi_0}{2\pi} = k_n \tag{5.3}$$

$$f_0 T_n + \frac{\phi(t)}{2\pi} = k_n \tag{5.4}$$

Under the assumption that $\phi_0 = 0$ we can calculate the time T_k at step k of the ideal clock with $C(T_n) = k_n/f_0$ whereas the time-stamp of the non-ideal clock contains an error introduced by $\phi(t)$. In [44] Giorgi et al. state that $\phi(t)$ can be further separated into at least two components. First, the deviation of the oscillator's frequency f_i from its nominal frequency f_0, this deviation accumulates over time. Second, the instantaneous phase fluctuations expressed by $2\pi f_0 \psi(t)$, this corresponds to an instantaneous time offset fluctuation.

To get to an iterative representation of the clock model, T_n can be expressed as an accumulation of the differences of all previous time-stamps [44] :

$$T_n = \sum_{i=0}^{n-1} (T_{i+1} - T_i) \tag{5.5}$$

such that in the case of the optimal clock 5.3 with $\phi = 0$ the following equality holds.

$$k_n = f_0 \sum_{i=0}^{n-1} (T_{i+1} - T_i) \tag{5.6}$$

For the real clock this equation differs as we have to account for the frequency deviation and the phase fluctuation in every step. Here we can see f_i as the mean frequency deviation over an interval [44] .

$$\begin{aligned} k_n &= \sum_{i=0}^{n-1} f_i[T_{i+1} - T_i] + \frac{2\pi f_0 \psi(T_n)}{2\pi} \\ &= \sum_{i=0}^{n-1} [f_i[T_{i+1} - T_i] + f_0 \psi(T_n) \end{aligned} \tag{5.7}$$

In order to compare the optimal and the real clock we have to introduce the instantaneous time offset in 5.2 in equation 5.6 of the optimal clock [44].

$$k_n = f_0[\sum_{i=0}^{n-1} (T_{i+1} - T_i) + \theta(T_n)] \tag{5.8}$$

This allows us to equate equation 5.7 with 5.8, which can be then used to directly calculate the error term $\theta(T_n)$. The dimensionless value $(f_i - f_0)/f_0$ is the so called skew of the two clocks and is represented by $\gamma(T_n)$

$$\theta(T_n) = \sum_{i=0}^{n-1} [\frac{f_i - f_0}{f_0} (T_{i+1} - T_i)] + \psi(T_n) \tag{5.9}$$

As T_i is generated at a constant rate at the reference clock, the difference of subsequent T_i is always ΔT and thus we can replace T_n with n. With this we can express one time step of equation 5.8 via [44]:

$$\begin{aligned} \theta(n+1) - \theta(n) = \gamma(n)\Delta T + \psi(n+1) - \psi(n) = \\ \theta(n+1) = \theta(n) + \gamma(n)\Delta T + \psi(n+1) - \psi(n) \end{aligned} \tag{5.10}$$

where $\gamma(n) = \gamma$ is assumed to be constant which corresponds to a constant skew model.

To realistically model the clock behaviour we have to model how skew and offset fluctuation influence the iterative clock model. To do so both the skew fluctuation and offset fluctuation can be modelled as two uncorrelated Gaussian random processes with zero mean [44]:

$$\begin{aligned} \psi(n+1) &= \psi(n) + \omega_\theta(n) \\ \gamma(n+1) &= \gamma(n) + \omega_\gamma(n) \end{aligned} \tag{5.11}$$

The resulting iterative model is expressed by:

$$\begin{aligned} \theta(n+1) &= \theta(n) + \gamma(n)\Delta T + \omega_\theta(n) \\ \gamma(n+1) &= \gamma(n) + \omega_\gamma(n) \end{aligned} \tag{5.12}$$

5.1.2 Stochastic Differential Equation Model

Next the iterative model is derived from a stochastic differential equation (SDE) model. This is done to show the link to the Allan variance and to emphasis that the optimal solution of the offset and skew calculation can be found by a Kalman filter. The instantaneous clock offset is modelled by a three-state stochastic differential model as described in [43].

$$\begin{aligned} dX_1(t) &= (X_2(t) + \mu_1)dt + \sigma_1 dW_1(t) \\ dX_2(t) &= (X_3(t) + \mu_2)dt + \sigma_2 dW_2(t) \\ dX_3(t) &= \mu_3 dt + \sigma_3 dW_3(t) \end{aligned} \tag{5.13}$$

with initial condition

$$\begin{aligned} X_1(0) &= c_1 \\ X_2(0) &= c_2 \\ X_3(0) &= c_3 \end{aligned} \tag{5.14}$$

where the variable X_1 represents the phase deviation, the derivation $\dot{X_1}$ is the frequency deviation or skew and X_2 contributes to this frequency deviation. X_3 represents the change of the skew over time, it is called aging. $\sigma_{1,2,3}$ are the diffusion parameters of the noise components and represent the random nature of phase offset and frequency deviation. $W_{1,2,3}$ are Wiener processes with time independent (stationary) random increments, thus, $W(t_2 - t_1) = W(t_2) - W(t_1) = dW(t) \sim \mathcal{N}(0, t_2 - t_1)$. dW_1 is contributing to the phase deviation and is the white frequency noise (WFN). dW_2 is acting on the frequency deviation can be seen as the random walk frequency noise (RWFN). $\mu_{1,2,3}$ represent the deterministic part of the clock error, i.e., initial clock frequency offset (μ_1), constant frequency aging (μ_2) and time variable aging (μ_3).

Next, we are going to simplify the three-state model in 5.13. First, we can neglect X_3 as in the case of crystal oscillators the aging value is usually very small, in the order of $1 \cdot 10^{-7}s$ per year. Hence we can ignore this part of the differential equation system ($\mu_3 = \sigma_3 = 0$) [48]. Instead, X_3 could be used to model a frequency change introduced by environmental changes, e.g., temperature change. However, we assume that all radio modules have reached their working temperature and that the environmental temperature stays constant during the short synchronization intervals. Second, initial frequency offset μ_1 can also be expressed by the state variable X_2 such that the frequency deviation is now solely expressed by X_2. Accordingly we have to move the frequency offset μ_1 into the initial condition $X_2(0) = \mu_1 + c_2$. Third, we expect that there is no constant skew change thus μ_2 is set zero. The system 5.14 changes to:

$$\begin{aligned} dX_1(t) &= X_2(t)dt + \sigma_1 dW_1(t) \\ dX_2(t) &= \sigma_2 dW_2(t) \end{aligned} \tag{5.15}$$

This stochastic differential equation system (5.15) can be rewritten in matrix form

$$d\mathbf{X} = (\boldsymbol{F}\boldsymbol{X}(t) + M)dt + \boldsymbol{Q}d\boldsymbol{W}(t) \tag{5.16}$$

with

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \boldsymbol{Q} \quad = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}, \mathbf{M} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \boldsymbol{X} = \begin{bmatrix} X_1(t) \\ X_2(t) \end{bmatrix} d\boldsymbol{W} = \begin{bmatrix} dW_1(t) \\ dW_2(t) \end{bmatrix} \tag{5.17}$$

These differential equations are a strictly linear stochastic differential equation system, thus the solution can be obtain in closed form [49] [43] [50].

5.1.3 Solution of the SDE Model

To derive a discrete time solution of the SDE system (5.15) we first point out that the differential system 5.15 can be rewritten in its integral from [51]:

$$\begin{aligned} \boldsymbol{X}(t) &= \boldsymbol{\Phi}(t-t_0)\boldsymbol{X}(t_0) + \int_{t_0}^{t} \boldsymbol{\Phi}(t-s)d\boldsymbol{W}(s)ds \\ \boldsymbol{X}(t) &= \boldsymbol{\Phi}(t-t_0)\boldsymbol{X}(t_0) + \boldsymbol{G}_t \end{aligned} \tag{5.18}$$

The transition matrix $\boldsymbol{\Phi}$ is calculated by Taylor expansion:

$$\begin{aligned} \boldsymbol{\Phi}(t) &= e^{\boldsymbol{F}(t)} = \boldsymbol{I} + \boldsymbol{F}(t) \\ &= \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{5.19}$$

And the innovation of the stochastic process X(t) is described by the noise vector $\boldsymbol{G}_t$

$$\boldsymbol{G}_t = \begin{bmatrix} \sigma_1 \int_{t_0}^{t} dW_1(s)ds + \sigma_2 \int_{t_0}^{t} (t-s)dW_2(s)ds \\ \sigma_2 \int_{t_0}^{t} dW_2(s)ds \end{bmatrix} = \begin{bmatrix} \sigma_1 W_1(t) + \sigma_2 \int_{t_0}^{t} W_2(s)ds \\ \sigma_2 W_2(t) \end{bmatrix} \tag{5.20}$$

$$\begin{aligned} X_1(t) &= x_0 + y_0 t + \sigma_1 W_1(t) + \sigma_2 \int_{t_0}^{t} W_2(s)ds \\ X_2(t) &= y_0 + \sigma_2 W_2(t)) \end{aligned} \tag{5.21}$$

The closed form solution 5.21 contains the initial skew ($y_0 = \mu_1$) and initial phase offset (x_0). From this solution it can be seen that the instantaneous clock offset contains a deterministic part, which is expressed by $x_0 + y_0 t$, and the stochastic part, which is expressed by $\sigma_1 W_1(t) + \sigma_2 \int_{t_0}^{t} W_2(s)ds$. Where $W_1(t), W_2(t)$ are the Wiener processes obtained by integrating $dW_1(t)$ and $dW_2(t)$.

The covariance matrix Cov_t of the noise vector G_t is [43]:

$$\begin{aligned} \mathbf{Cov}_t &= E[G_t G_t^T] \\ \mathbf{Cov}_t &= \begin{bmatrix} \sigma_1^2 t + \sigma_2^2 \frac{t^3}{3} & \sigma_2^2 \frac{t^2}{2} \\ \sigma_2^2 \frac{t^2}{2} & \sigma_2^2 t \end{bmatrix} \end{aligned} \tag{5.22}$$

We now derive the iterative solution of the stochastic differential equation system. The motivation for this is twofold. First, we want to show the similarity between the model derived in section 5.1.1 and second the iterative solution is used in the next section to determine the noise parameter of the covariance matrix.

To move to the discrete iterative representation we can rewrite 5.16 by replacing t with discrete time intervals $t_0 < t_1 < t_N$ and $\tau = t_{k+1} - t_k$. Equation 5.21 is then rewritten as follows [49]:

$$\begin{aligned} X_1(t_{k+1}) &= X_1(t_k) + X_2(t_k)\tau + J_{k,1} \\ X_2(t_{k+1}) &= X_2(t_k) + J_{k,2} \end{aligned} \tag{5.23}$$

where J_k is

$$\begin{aligned} J_{k,1} &= \sigma_1(W_1(t_{k+1}) - W_1(t_k)) + \sigma_2 \int_{t_k}^{t_{k+1}} (W_2(s) - W_2(t_k))ds \\ J_{k,2} &= \sigma_2(W_2(t_{k+1}) - W_2(t_k)) \end{aligned} \tag{5.24}$$

and is normal distributed with zero mean covariance matrix $\mathbf{Cov}_\tau$: [49]

$$\begin{aligned} \mathbf{Cov}_\tau &= E[J_\tau J_\tau^T] \\ \mathbf{Cov}_\tau &= \begin{bmatrix} \sigma_1^2\tau + \sigma_2^2\frac{\tau^3}{3} & \sigma_2^2\frac{\tau^2}{2} \\ \sigma_2^2\frac{\tau^2}{2} & \sigma_2^2\tau \end{bmatrix} \end{aligned} \tag{5.25}$$

What is now left to do is to determine the noise parameters of the covariance matrix 5.25. As we will see in the next section, these values can be calculated by using the Allan variance.

5.1.4 Allan deviation

David Allan defined the Allan deviation in [52]. It is a two-sample variance and measures the variance between two subsequent measurements, rather than calculating the difference between the mean value and a sample as it is usually done for standard deviation calculation. It is defined as:

$$\sigma_y^2(\tau) = \frac{1}{2}\langle(\overline{y}_{n+1} - \overline{y}_n)\rangle \tag{5.26}$$

where τ is the time between sample $\overline{y}_{n+1}$ and $\overline{y}_n$. It can be used to identify different types of clock noise. In Figure 5.3 we can see the Allan variance for different values of τ. From this figure we can identify five different regions with different slopes; each of these regions corresponds to a different type of noise. In our case we are interested in the white frequency noise (WFN) and the random walk frequency noise (RWFN), we expect these two noise terms to be the major source of error in our clock model.

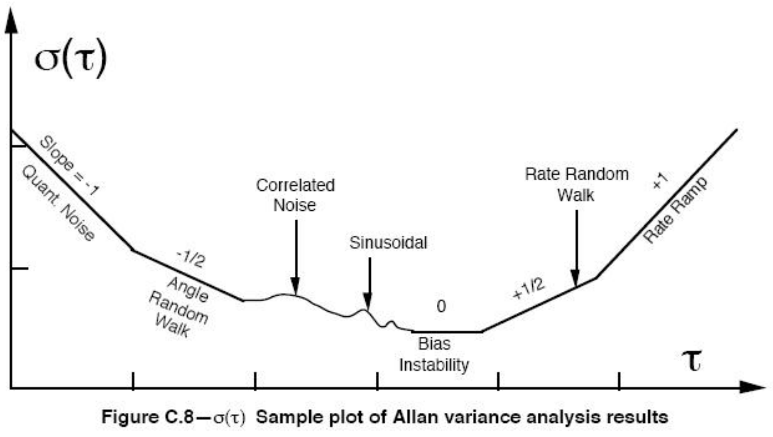


Figure 5.3: Sample plot of the Allan variance analysis results. WFN is the angle random walk with a slope of -1/2. RWFN is the rate random walk with a slope of 1/2. Taken from [53].

In [43] Zucca et al. show the relationship between the Allen variance and the parameters σ_1, σ_2, which allows one to identify the parameters of the covariance matrix ($\mathbf{Q}$) of the state space representation.

The frequency deviation is defined as

$$\begin{aligned} \overline{Y}_k &= \frac{1}{\tau}\int_{t_k}^{t_{k+1}} \dot{X}_1(t) * dt \\ &= \frac{1}{\tau}(X_1(t_{k+1}) - X_1(t_k)) \end{aligned} \tag{5.27}$$

and can be inserted into the previously defined Allen variance:

$$\begin{aligned} \sigma_y^2(\tau) &= \frac{1}{2\tau^2}E[(\overline{Y}_{k+1} - \overline{Y}_k)^2] \\ &= \frac{1}{2\tau^2}(E[(X_1(t_{k+2}) - 2X_1(t_{k+1}) + X_1(t_k))^2]) \\ &= \frac{1}{2\tau^2}(E[(\Delta)^2]) \end{aligned} \tag{5.28}$$

after inserting the iterative solution 5.23 iteratively into 5.28 $\mathbf{\Delta}$ is as follows:

$$\begin{aligned} \mathbf{\Delta} &= X_1(t_{k+2}) - 2X_1(t_{k+1}) + X_1(t_k) \\ &= J_{k+1,1} + [-J_{k,1} + \tau J_{k,2}] \end{aligned} \tag{5.29}$$

The expectation value and variance of the individual terms of equation 5.29 are given in [43] by

$$E[\Delta] = 0 \tag{5.30}$$

$$\begin{aligned}\sigma_y^2(\tau) &= E[\Delta^2] \\ &= \frac{1}{2\tau^2}(Var[\Delta] + E[\Delta]^2) \\ &= \frac{\sigma_1^2}{\tau} + \frac{\sigma_2^2\tau}{3}\end{aligned} \tag{5.31}$$

In this equation σ_1 refers to the white frequency noise (WFN) and σ_2 to the random walk white frequency noise (RWFN). The noise identification is then done by fitting equation 5.31 to values calculated by the Allan variance.

5.2 Clock Error Correction

As we saw in the last section, the time error between two modules contains an offset error, which originates from the different times when the modules are switched on, as well as a skew error which originates from imperfections of the clocks. When synchronizing clocks, we want to minimize the error between a reference and local clocks such that we can establish a common timebase among different nodes in a network. In the following section, we discuss the simple skew model (SKM) and statistical filtering that can be used to compensate for the errors.

5.2.1 Simple Skew Model

Lets assume that $t_{rx,n} = C(T_n)$ is the time of arrival of the n-th packet at the anchor node and $t_{tx,n} = T_n$ is the time of transmission of the n-th packet at the reference node and τ is the time between two subsequent time reference message broadcasts.

The time offset between two nodes is calculated as:

$$\theta(n) = t_{rx,n} - t_{tx,n} \tag{5.32}$$

What is left to do is to in cooperate the offset error that arises from the propagation delay. For this the distance r to the reference node has to be known. $\theta(n)$ then changes to

$$\theta(n) = t_{rx,n} - t_{tx,n} - \frac{r}{c} \tag{5.33}$$

where c is the value of the speed of light.

The skew is calculated by:

$$\gamma_n = \frac{\theta(n) - \theta(n-1)}{\tau} \tag{5.34}$$

Now lets assume we receive a packet at time T_i after two synchronization packet broadcasts have been received such that $T_{n-1} < T_n < T_i < T_{n+1}$. We can then use the calculated skew and offset of the last two preceding synchronization packets to transform the recorded local reception time into local time of the reference node. Recalling the notation of space-time we would call this a coordinate transformation, in particular the transformation of the locale time coordinate system into the coordinate system of a reference node. To calculate the corrected reception time-stamp $t_{rxc,i}$ of packet i we have to add the offset of the last reference packet and an extrapolated skew correction term to the reception time $t_{rx,i}$ [36] [14].

$$\begin{aligned} \theta(i) &= \theta(n) + \gamma(n)(t_{rx,i} - t_{rx,n}) \\ t_{rxc,i} &= t_{rx,i} + \theta(i) \end{aligned} \tag{5.35}$$

5.2.2 Kalman filter

Once the clock model (equation 5.12, and 5.23) has been identified it can be used to build a recursive estimator, the Kalman filter. The model can be rewritten in matrix form:

$$\begin{aligned} x(n) &= \mathbf{A}x(n-1) + \omega(n-1) \\ x(n) &= [\theta(n)\gamma(n)]^T, \omega(n) = [\omega_\theta, \omega_\gamma]^T \\ \mathbf{A} &= \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{5.36}$$

The Kalman filter holds information about the current state as well as the uncertainties of the states and works in two steps: prediction and update. During the prediction step, the Kalman filter estimates the next state based on the current state. In this step also the uncertainty of the states is increased. In the update step, new measurements are used to update the current state using a weighted average. The weights result from the uncertainties of the current states and the uncertainties of the measurements.

The state prediction step is defined as

$$\begin{aligned} \underline{\widehat{x}}(n) &= \mathbf{A}x(n-1) + \mathbf{B}u(n-1) \\ \mathbf{P}(n) &= \mathbf{A}\mathbf{P}(n-1)\mathbf{A}^T + \mathbf{Q} \end{aligned} \tag{5.37}$$

The state update equation of the recursive Kalman filter is defined as

$$\begin{aligned} \mathbf{K}(n) &= \mathbf{P}(n-1)\mathbf{H}^T[\mathbf{H}\mathbf{P}(n-1) + \mathbf{R}]^{-1} \\ \underline{\widehat{x}}(n) &= \underline{\widehat{x}}(n-1) + \mathbf{K}(n)(\underline{z}(n) - \underline{\widehat{x}}(n-1)) \\ \mathbf{P} &= [\mathbf{I} - \mathbf{K}(n)]\mathbf{P}(n-1) \end{aligned} \tag{5.38}$$

R is the measurement uncertainty and expresses the uncertainty of new measurements.

P(n-1) is the apriori error prediction covariance matrix while **P(n)** is the a posterior error covariance matrix. Essentially, these two matrices represent the uncertainty of the process at the current and next step and influence the Kalman-gain, thus determine how strong new measurement updates influence the state variables.

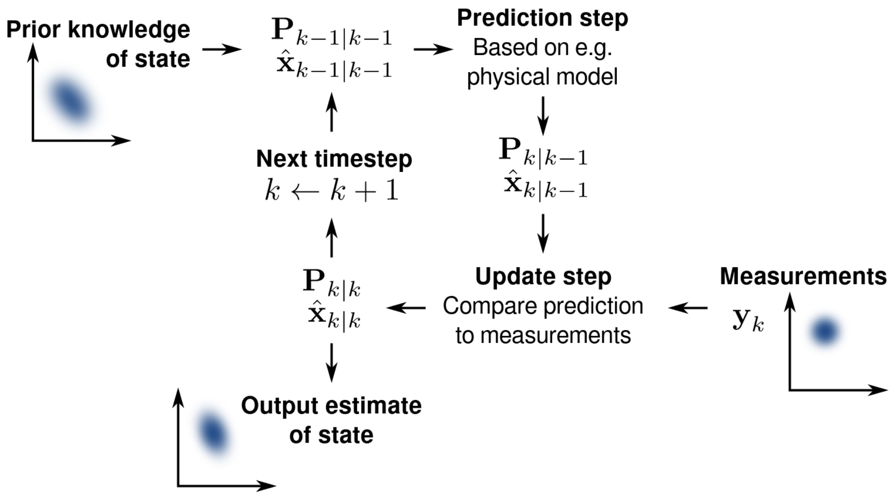

Prior knowledge of state
$\mathbf{P}_{k-1|k-1}$ $\hat{\mathbf{x}}_{k-1|k-1}$
Prediction step
Based on e.g. physical model
$\mathbf{P}_{k|k-1}$ $\hat{\mathbf{x}}_{k|k-1}$
Next timestep
$k \leftarrow k+1$
Update step
Compare prediction to measurements
Measurements
$\mathbf{y}_k$
$\mathbf{P}_{k|k}$ $\hat{\mathbf{x}}_{k|k}$
Output estimate of state

Figure 5.4: Concept of the Kalman filter. In the prediction step, the next state is predicted from the current state but also the uncertainty increases. In the update step, the current state is combined with the new measurements weighted by the state uncertainty and the measurement uncertainty. In this step, the uncertainty of the state is decreased. Taken from [54].

$\mathbf{K(n)}$ is the Kalman gain and is calculated from the apriori error covariance matrix and the measurement uncertainty matrix $\mathbf{R}$.

$\mathbf{Q(n)}$ is the process noise covariance matrix which determines how strong the uncertainty of a process evolves. As described in [44] it is a diagonal matrix with the time offset and skew uncertainty as its diagonal entries. The results from [43] suggest that it is not a diagonal matrix, but as the nondiagonal entries have little influence , thus, we follow the approach from [44].

$$\mathbf{Q} = \begin{bmatrix} \sigma_\theta^2 \Delta T & 0 \\ 0 & \sigma_\gamma^2 \Delta T \end{bmatrix} \tag{5.39}$$

$\mathbf{H}$ is the output matrix and is an identity matrix.

To use the Kalman filter to correct the reception time-stamp of an incoming packet i we use equation 5.35 but use $\theta(n), \gamma(n)$ from the Kalman filter.

5.3 Measurement Setup

For initial data generation to compare the simple skew model with the Kalman filter, we used a simple measurement setup where we use three modules. The reference module consists of a Nucleo board (Node A) and one DW1000 transceiver and is continuously transmitting synchronization packets containing the transmission time-stamp at a constant rate of $10Hz$, or every $10ms$, respectively. The receiving module (Node B) consists of a

Nucleo board and two DW1000 transceivers are connected to the same SPI bus but with different chip select lines. The receiver software works in polling mode and alternates the chip select pin to fetch new packets from either of the transceivers. For each message, the reception time-stamp is recorded and forwarded together with the synchronization message ID and the transmission time-stamp to MATLAB. In general, the evaluation works in the following way: First, a couple of synchronization messages are sent and the offset between the local time and the time of the reference node as well as the skew is calculated. Second, we use these values to correct the reception time-stamp of the next packet and compare the calculated value with the true transmission time-stamp. To compare two modules or localization packet time-stamps, we calculate the corrected time-stamp for two modules and compare these time values. Ideally, for both scenarios, the error should be zero.

Two datasets were recorded, each consists of 300000 data points, which correspond to roughly 50 minutes of recording time. In Figure 5.6a we can see the instantaneous time offset of the two modules of the first dataset. Additionally, we added two straight lines that connect the first and the last timestamps. The received time-stamps differ from those straight lines, hence, the time does not evolve in a strictly linear manner indicating a frequency change. This effect is even more visible if we plot the difference of subsequent timestamps as shown in 5.6b. We reckon that this is due to the temperature change after the modules are switched on as this behavior was not observed in our second dataset, which was recorded after the first dataset. Although our model does not account for this error, experimental results with an update rate of 10Hz suggest that the performance of the Kalman filter and the simple extrapolation method is not influenced by the constant temperature change. We reason that the rate of change is too slow compared to the update rate. During the online testing, however, we could observe that the temperature change decreases the synchronization accuracy.

5.4 Noise identification

We then used Matlab to calculate the Allan variance and used the curve fitting toolbox of Matlab (Listing 5.1) to fit equation 5.31 to the calculated Allan variance. In Figure 5.5 the Allan variance and the fitted curve is depicted. The calculated noise parameters can be seen in Table 5.1 and the measurement uncertainty was taken from [55] and adjusted experimentally.

Listing 5.1: Curve fitting in Matlab

```
f = fittype({'1/x','x/3'});
fitobject = fit(allan_timesteps,allan_variance, f)
sig1 = f.a;
sig2 = f.b;
```

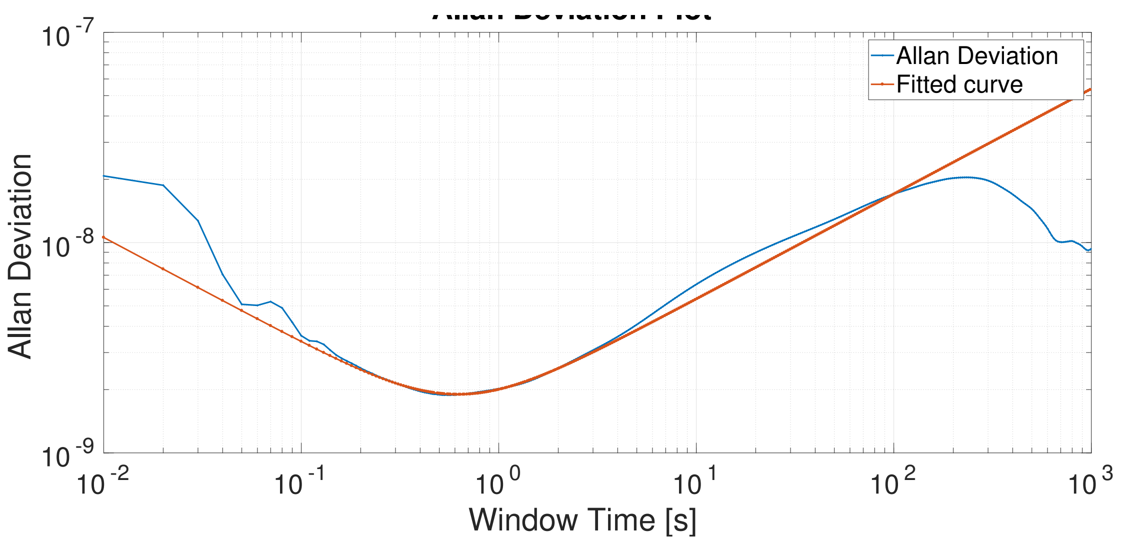


Figure 5.5: Allan variance and fitted curve.

Table 5.1: Kalman Parameter

Parameter	Value
σ_θ	2.05^{-9}.
σ_γ	4.93^{-9}

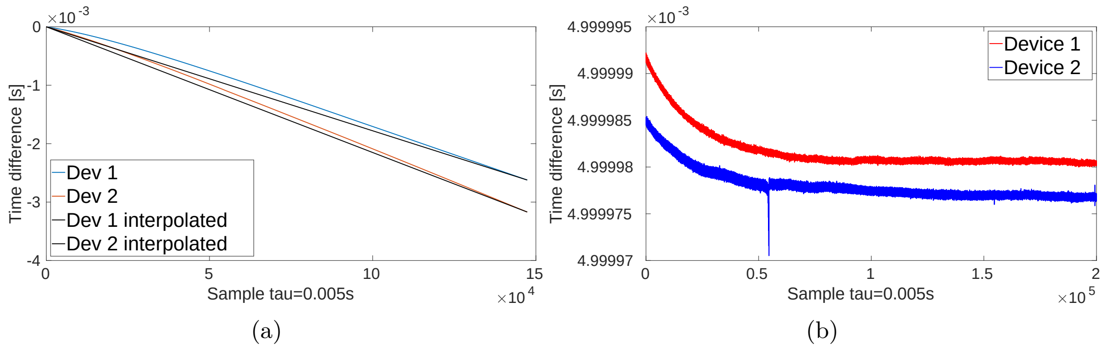
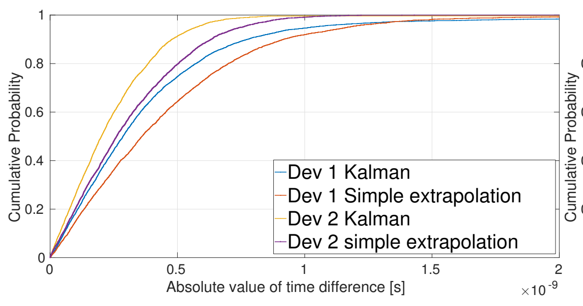


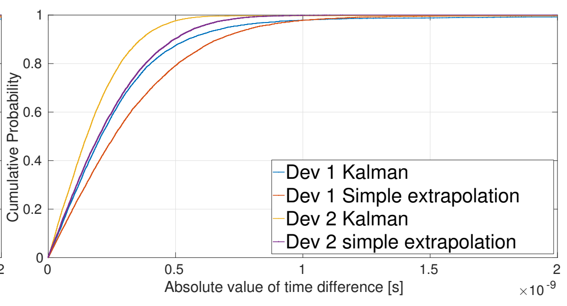
Figure 5.6: 5.6a shows the reception time of the same packets at two nodes. Figure 5.6b shows the difference of subsequent reception times, as the reference module broadcasts at 20Hz it should be ideally 5ms.

5.5 Offline evaluation

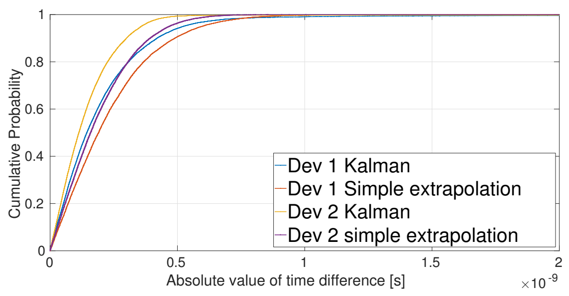
To compare the performance of the two synchronization methods we calculated the error between the corrected reception time-stamp and the transmission time-stamp embedded in the synchronization message, the errors are then visualized in CDF plots. We did this for different update rates 5Hz (5.7a), 10Hz (5.7b) and 20Hz (5.7c). As we can see the Kalman filter outperforms the SKM at high update rate. Another observation is that both synchronization methods performed by far worse for the second module, which we cannot explain at the moment and further tests with more modules have to be conducted.



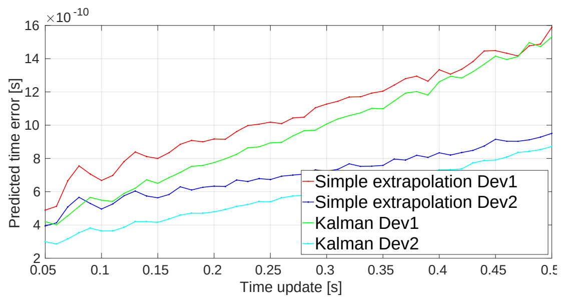
(a)



(b)



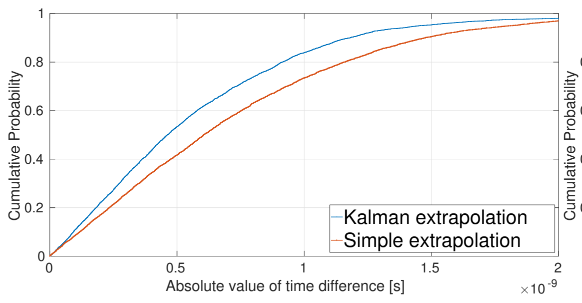
(c)



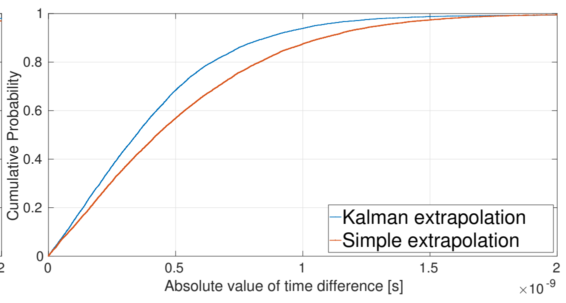
(d)

Figure 5.7: CDF of synchronization error between nodes and reference node depending on different update rates. Figure 5.7a depicts the CDF for an update rate of 5Hz, Figure 5.7b for 10Hz and Figure 5.7c 20Hz. Figure 5.7d shows the 90th percentile time offset error for different update rates.

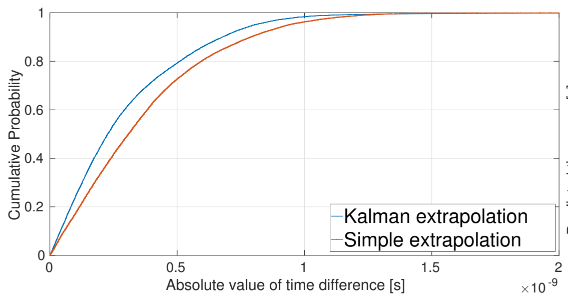
Like before we were interested in the synchronization error but this time in the synchronization error between the two nodes. The CDF of the error can be seen in Figure 5.8, again for update rates 5, 10, 20 Hz, respectively. We also calculated the 90th percentile of the error for different update rates which is shown in Figure 5.8d. What we can see in this figure is that the trend of the error follows the one of the first module in Figure 5.7d, hence, the overall performance is mostly influenced by the worse module.



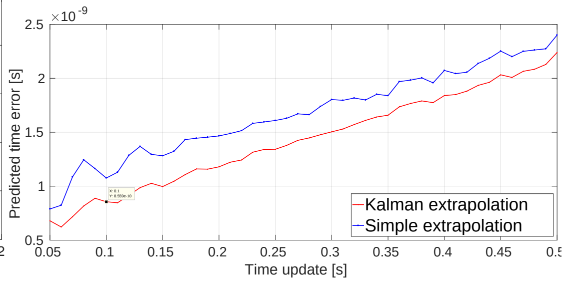
(a)



(b)



(c)

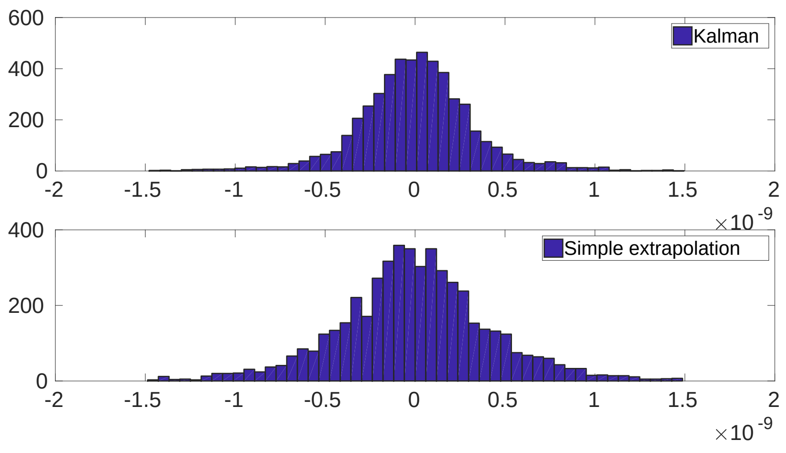


(d)

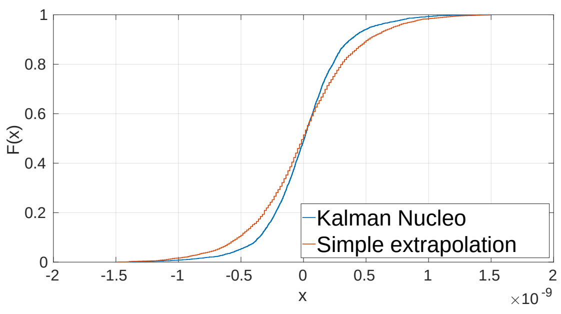
Figure 5.8: CDF of synchronization error between two nodes. Figure 5.8a depicts the CDF for an update rate of 5Hz, Figure 5.8b for 10Hz and Figure 5.8c 20Hz. Figure 5.8d shows the 90th percentile time offset error for different update rates.

5.6 Online evaluation

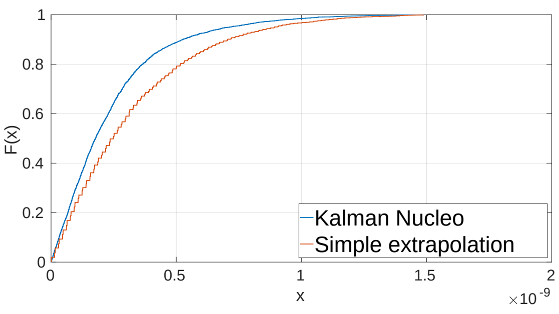
To evaluate the performance of both synchronization methods, we implemented them on the Nucleo board and used them to correct and compare the reception time-stamps of synchronization messages as well as the reception time-stamps of localization messages. Likewise, for the offline testing, we use node A to constantly broadcast synchronization messages which are received by node B where two DW1000 modules are connected, a third node (node C) is used to issue localization packets. For our first evaluation, we broadcast the synchronization packets with a rate of 10Hz and node C issues localization packets 80ms after every second synchronization message. In Figure 5.9 the error of the synchronization messages is shown. In Figure 5.10 the error of the localization messages is shown.



(a)

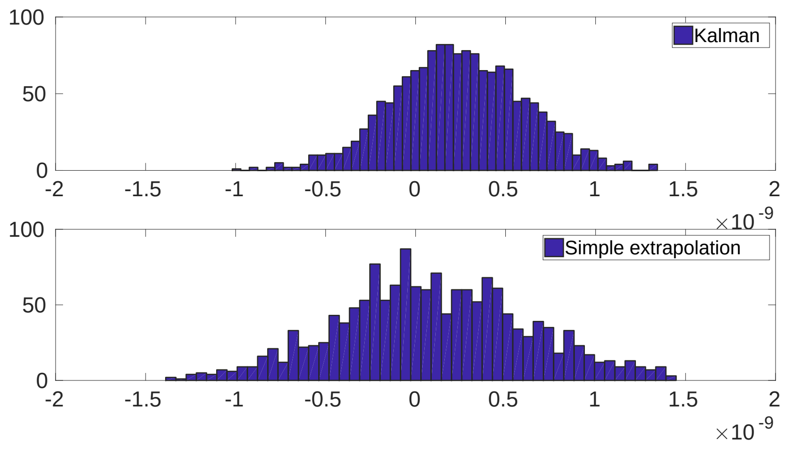


(b)

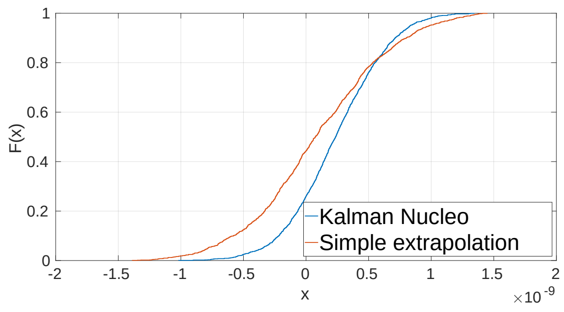


(c)

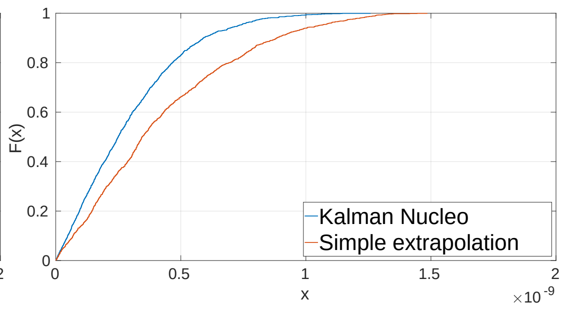
Figure 5.9: 5.9a shows the histogram of the error between the corrected timestamps (t_{rxc}) of synchronization messages between two modules. Ideally it should be zero. 5.9b shows the CDF, and Figure 5.9c shows the CDF of the absolute error with the mean removed.



(a)



(b)



(c)

Figure 5.10: 5.10a shows the histogram of the error between the corrected timestamps (t_{rxc}) of localization messages between two modules. Ideally it should be zero. 5.10b shows the CDF, and Figure 5.10c shows the CDF of the absolute error with the mean removed.

5.7 Conclusion

Precise time synchronization in the sub-nanosecond range is a challenging task as the accuracy depends on several factors. What we could observe during our online evaluation was that small temperature changes can influence the accuracy over short periods leading to an offset error of 1-2 nanoseconds. Initially, we assumed that once all radio modules have reached their working temperature, the clock skew stays the same over short synchronization intervals. What we could observe is that even blowing at one module changes the frequency and thus has an influence on the accuracy.

In general, the Kalman filter yields better results when high synchronization rates are used, but unfortunately, it did not improve the synchronization accuracy in the case of low update rates. What we could also observe is that the Kalman filter lags behind if sudden changes of the clock offset happen. Especially if the clock skew changes due to a temperature change the Kalman filter produces results where the mean value is not at zero but shifts by a constant value as can be seen in Figure 5.10b.

To compensate for this we temporarily introduced a third state in the Kalman filter to track the skew change introduced by the temperature change; the results were not distinct. In situations with a slow (almost constant) temperature change the third state helped to reduce the error. However, it also decreased the sensitivity of the Kalman filter to sudden changes, e.g., when we quickly changed the temperature. In this situations, the Kalman filter seemed to lag even more behind the simple skew model. It also made the Kalman filter more sensitive to wrong measurement updates as it required much longer to recover from single wrong measurement updates. Moreover, it was difficult to find a suitable noise parameter for the third state. One solution to this might be to reset and re-initialize the Kalman filter if such a case is detected. To make the two-state Kalman filter more sensitive one could increase the values of the process noise matrix, but this would also decrease the accuracy if the skew is stable.

CHAPTER 6

Clustering

While energy efficiency is the main motivation for most clustering algorithms, our requirements are different. The main objective is to maximize the localization accuracy within the sensor network. To achieve reasonable localization accuracy, it is required that all anchor node's clocks are synchronized in sub-nanosecond range. As we discussed earlier, multi-hop synchronization may not be suitable for large sensor networks with tight synchronization bounds. To overcome this problem, we have to partition the network into smaller clusters. In each cluster, all members (anchor nodes) synchronize their clocks to the cluster-head such that each cluster forms a small TDOA system on its own. For our system, the cluster-head performs several tasks such as coordinating the communication of the anchor nodes, the localization of mobile nodes within its cluster, or simply data aggregation by removing redundant TDOA measurements.

One challenge along the way of building a distributed and self-organizing localization system is to find a suitable clustering such that each cluster allows for high localization accuracy. On top of that, it is also required that such a system yields high overall localization accuracy, meaning that there should be no areas between clusters where localization accuracy is reduced.

So far no clustering algorithms with the objective to maximize localization accuracy within the convex hull of a cluster but also optimizing the overall localization accuracy exist. As a starting point, a simple probabilistic and deterministic clustering algorithm shall be evaluated through simulation.

For the evaluation, several realistic scenarios where anchor nodes are placed with some randomness on a map are created. The two clustering approaches are then used to compute a possible clustering for the given maps. Afterward, we used the approach from [13] to calculate the Cramér-Rao lower bound (CRLB) to calculate the localization uncertainty in each point of the map.

6.1 Requirements

The requirements for a suitable clustering algorithm depend on the exact use case, however, some requirements are universal for a localization system:

- High local localization accuracy: Within each cluster, the localization performance should be high. To achieve this, the anchor nodes should be arranged circularly around the center of the cluster.
- High global localization accuracy: The overall localization performance should be high. Thus, there should be no areas between clusters where the localization accuracy is low.
- Little overlapping areas: The clusters should overlap as little as possible.
- Scalable: The algorithm should be able to partition large sensor networks.

Some additional requirements that are important for our system are:

- Self-organizing: The algorithm should be self-organizing, meaning that only locally available information is used and processed to find a reasonable partition of the sensor network.
- Distributed: No central unit should be responsible for the clustering.

For this thesis it is assumed that nodes are not running on battery and the anchor nodes are fixed. Hence the requirement mentioned next are relaxed.

- Fast convergence: A suitable clustering should be found as fast a possible. Although this requirement is fundamental in many situations, it is relaxed for our system as we assume to have static anchor nodes. Thus, it is of no harm if the clustering takes longer during the setup phase.
- Little overhead: As few messages as possible should be exchanged. Also, this requirement is relaxed in our system as clustering is only active during the setup phase.
- Network lifetime: The clusters should be arranged in such a way that the network lifetime is increased. Again this is relaxed for our system as the static nodes do not run on battery.

From the previously discussed requirements, and the structure of the proposed system described in Chapter 3, the following cluster-characteristics are identified:

- **Cluster properties**: The cluster-size should be as big as possible but can be unequal and the cluster number as small as possible, thus can be variable. Inter-cluster communication is required but might be realized via gateway nodes (gateway nodes are nodes that are part of two clusters and can be used to exchange data between clusters). Intra-cluster communication is single-hop, i.e., each member of a cluster can directly communicate with its cluster-head.

- **Cluster-head properties**: The cluster-head is expected to be homogeneous, thus the cluster-head is of the same type as all other nodes. The cluster-head is stationary, and it does perform data aggregation and fusion.

- **Clustering process properties**: The clustering process has to be deterministic, i.e., it has to be weight-based such that nodes that can create large clusters with high localization accuracy are preferred over nodes that can only form small clusters with low localization accuracy. Moreover, it has to be dynamic and must be able to react if nodes stop working.

To the author's knowledge there exists no clustering algorithm that was intentionally designed to solve our objective: finding a clustering of our sensor network that allows localization with high accuracy within the whole covered area.

6.2 Clustering algorithms

In this section, a probabilistic and a deterministic iterative algorithm are presented. Both algorithms are quite similar but the deterministic uses a time-depending function to decide on each nodes role, thus, it is more predictable. Both algorithms are later analyzed.

6.2.1 Probabilistic

The probabilistic algorithm works iteratively. In each iteration, all nodes broadcast their current state, i.e., clustered, unclustered or cluster-head if its state has changed since the last iteration. This allows all nodes to maintain their *Neighbours* list which is important for the next step.

To become a cluster-head three conditions must be met. First, there is no other cluster-head in communication range. Second, at least two other unclustered nodes must be in range, which is required to localize a node in the 2D plane. Third, each node generates a random value between 0 and 1, only if the value is greater than 0.8 it decides to become a cluster-head, thus only with 20% probability a node becomes a cluster-head.

Algorithm 6.1: Probabilistic algorithm

```
Result: Decide on nodes role
role := unclustered;
while $time < maxtime$ do
    $random$ ← generate random value between 0 and 1;
    CHs ← get CH in range;
    UCs ← get unclustered nodes in range;
    if $number(CHs) > 1$ then
        join clusterhead;
        role := clustered;
    end
    if $number(CHs) == 0$ and $random > 0.8$ and $number(UCs) > 3$ then
        role := clusterhead;
        return
    end
end
```

6.2.2 Deterministic

The deterministic algorithm works similar to the previously described probabilistic algorithm. The main difference is that no random value is used to decide on the role of a node, it rather uses an exponentially decaying time depending function (equation 6.1) as proposed in [33]. In detail, as soon as the decaying time depending function is smaller than the number of unclustered neighbors the node becomes a cluster-head.

$$f = \exp(-k_1 \frac{t}{t_{max}} - k_2)d \tag{6.1}$$

Where $k_1 = 2.3$ and $k_2 = 0.1$ were empirically found by [33]. t is the local time of a node and d is the desired cluster size. t_{max} is the maximal duration time of the protocol. The idea behind this method is that equally sized clusters are preferred over clusters with different size and that the cluster-size can be controlled via the parameter d. Furthermore, it leads to a repulsion effect between clusters as nodes that are close to the border of a cluster have less unclustered neighbors, thus they become late a cluster-head.

Algorithm 6.2: Deterministic algorithm

```
Result: Decide on nodes role
role := unclustered;
while $time < maxtime$ do
    CHs ← get CH in range;
    UCs ← get unclustered nodes in range;
    if $number(CHs) > 1$ then
        join clusterhead;
        role := clustered;
    end
    f ← $\exp(-k_1 \frac{time}{timemax} - k_2)d$ ;
    if number(CHs)== 0 and f < number(UCs) and number(UCs)> 3 then
        role := clusterhead;
        return
    end
end
```

6.3 Simulation

Our simulation framework is built in Matlab, and it can simulate the communication between nodes, i.e., send and receive messages within the communication range of the nodes. It also allows to simulate and report collisions when two nodes send messages at the same time, this feature, however, was not used for simplicity.

Each node in our simulation framework holds the following properties:

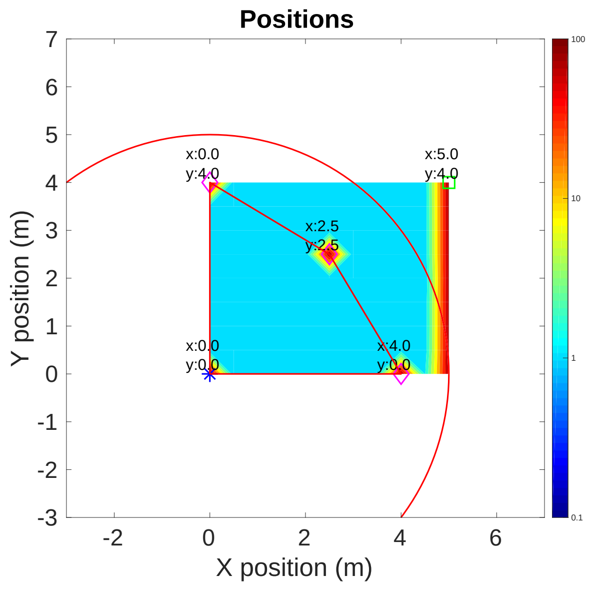
- Position
- *cstatus.* This field represents the cluster role. It can be either *unclustered*, *clustered* or *cluster-head.*
- *Neighbours.* This field contains all IDs of nodes in range and their *cstatus.*
- *MyFollowers.* This field contains all nodes that are following a node. A node A considers a node B following itself if the cluster ID of node B corresponds to the cluster ID of node A, or if the cluster ID of node B is 0.

- *Time.* Each node holds its local time, however, for simplification it is assumed that all nodes are switched on at the same time.

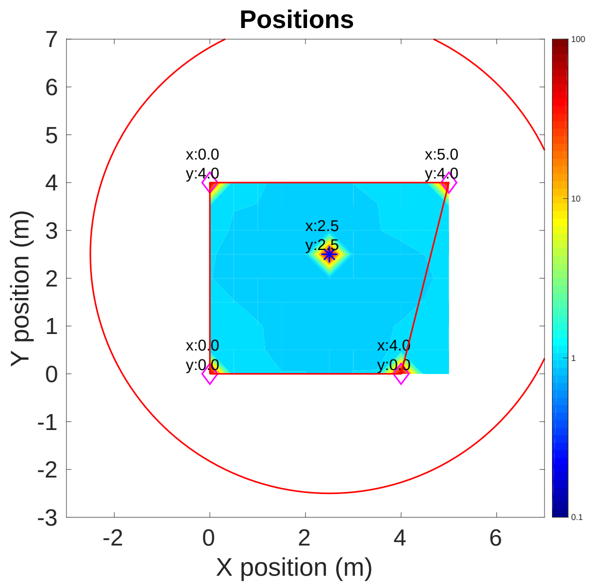
In this section, three scenarios are presented where it is shown that the deterministic algorithm is advantageous over the probabilistic. This is done by calculating the CRLB of any point in the area where the anchor nodes are placed after the clustering algorithms were used to calculate a clustering. The background color indicates the calculated localization uncertainty in each point. For this simulation, we assume that there is a standard deviation of 1 meter between all anchor nodes which does not correspond to the previously empirically determined variance in the synchronization section. This can be justified by the fact that we are only interested in the comparison of the performance of the individual clustering algorithms and not in the actual localization accuracy. For all simulation, we assume that the range of each sensor is 5 meters, the iteration time is $t_{max} = nr_{nodes} \cdot 10$, where nr_{nodes} is the number of nodes, the desired cluster size d was set to 10 in all simulations.

6.3.1 Room

In this example, we assume that one node is placed in each corner of a room with a dimension of 5x4 meters. A fifth node is placed in the middle of the room. The communication range of each node is 5 meters such that the only node that can synchronize all nodes at once is the node at the center of the room. As can be seen in Figure 6.1a, the probabilistic algorithm might select one of the nodes in the corner as the cluster-head. It can be seen that not all nodes are part of the cluster and the localization accuracy is low in the upper right part of the room. In contrast, the deterministic algorithm (Figure 6.1b) is able to successfully select the node in the middle of the room as the cluster-head.



(a)



(b)

Figure 6.1: Simulated clustering of the room example. Blue nodes are cluster-heads, green nodes are unclustered and pink nodes are clustered. The background color is the calculated CRLB at the position given a set of anchor nodes with a fixed variance of 1 between all anchor nodes. Figure 6.1a is the clustering generated by probabilistic algorithm. Figure 6.1b is the clustering generated by deterministic algorithm.

6.3.2 Hallway

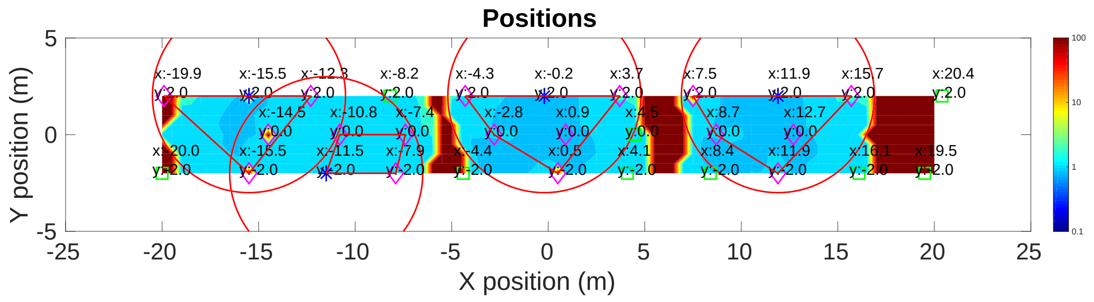
In this scenario a hallway with a size of 40x5 meters is simulated, where anchor nodes are placed in a regular pattern on each side of the hallway. Additionally, nodes are placed in the center of the hallway. In order to add some randomness in the anchor placement each node position is randomly moved by maximal ±0.5 meters. As we can see in Figure 6.2a the probabilistic approach leads to some areas where localization accuracy is very low (red areas), and eight anchor nodes are neither a cluster-head nor clustered. For the same map, the deterministic algorithm yields better results as can be seen in Figure 6.2b. For this algorithm, only two nodes are not part of any cluster, and there is no area in between two clusters where the localization accuracy is reduced. To further confirm the intuition we simulated this scenario 200 times and stored the localization uncertainty of each point of each simulation. These values were then used to calculate the overall probability that an area is exceeding a certain localization uncertainty. The results can be seen in Figure 6.4a where the cumulative distribution function of the localization accuracy is depicted. From this figure, we can deduce that the deterministic clustering algorithm leads to clustering where the localization variance does not exceed 1.6m in 90% of the area. Contrary, the probabilistic algorithm performs worse and does not even reach the 90% mark.

6.3.3 Open area

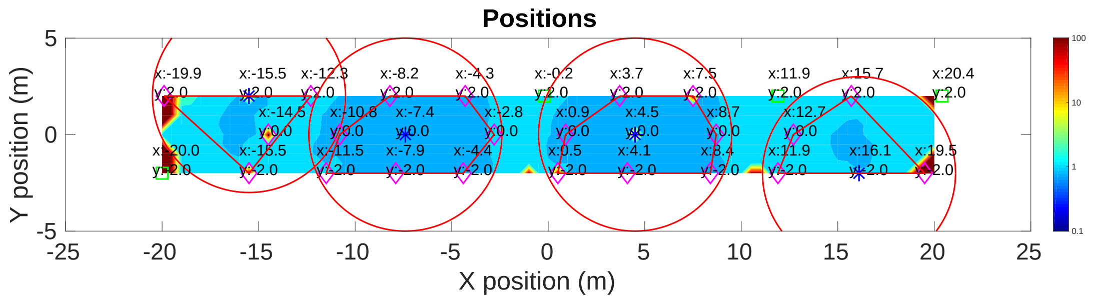
In this scenario a large area of 20x20 meters was simulated 200 times. Again the anchor nodes were placed in a regular pattern and randomized by uniformly distributed value between ±0.5 meters. As can be seen in Figure 6.3a the probabilistic algorithm produces clusters of equal size but large areas of the clusters are overlapping. Additionally, there are large areas where localization performance is low and several anchor nodes are unclustered.

In contrast, the deterministic algorithm leads to a clustering with larger clusters and the repulsive effect leads to less overlapping areas as can be seen in Figure 6.3b. Furthermore, there are fewer areas with poor localization accuracy, and only one anchor node is not part of any cluster.

As in the hallway example we calculated the cumulative probability of each point in the map exceeding a certain localization uncertainty (Figure 6.4b). As can be seen in this figure the overall localization accuracy is worse than in the hallway example, but the deterministic algorithm does still perform better than the probabilistic algorithm. The localization accuracy is high at the center of the area, but the localization performance is low at the border of the grid.

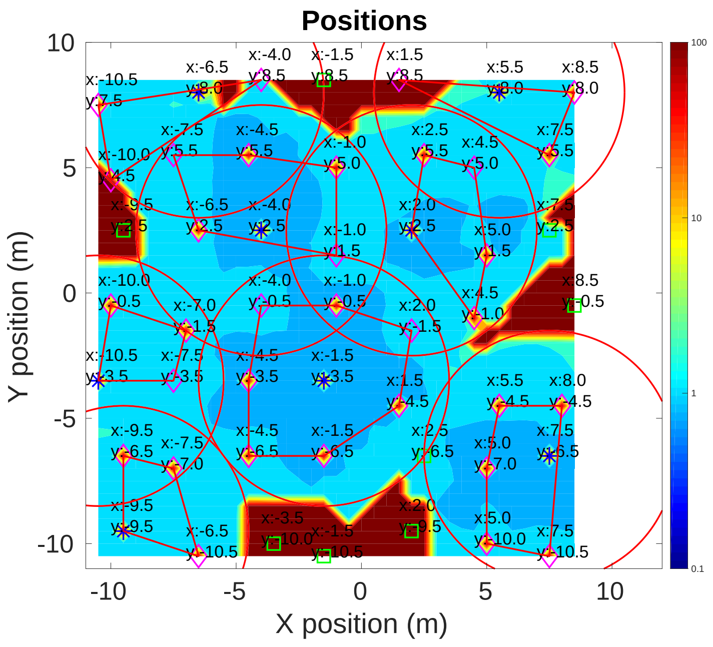


(a)

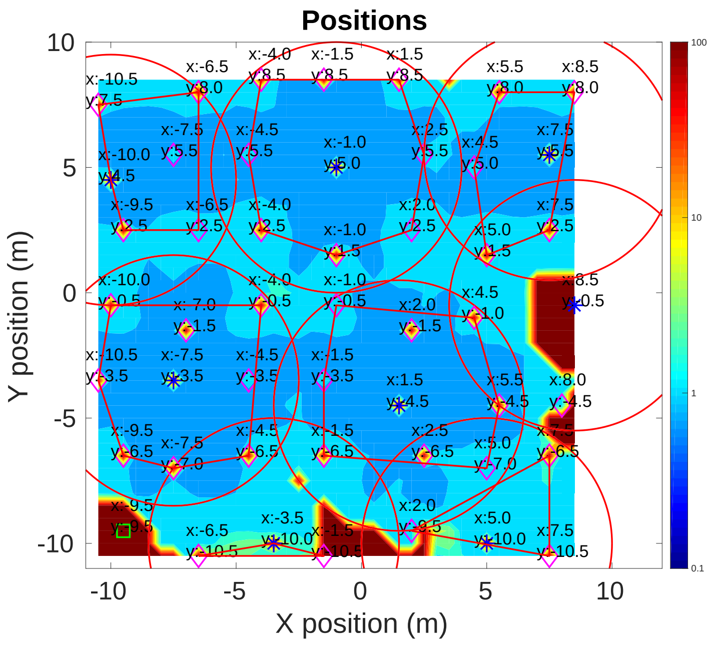


(b)

Figure 6.2: Simulated clustering of the hallway example. Blue nodes are cluster-heads, green nodes are unclustered and pink nodes are clustered. The background color is the calculated CRLB at the position given a set of anchor nodes with a fixed variance of 1 between all anchor nodes. Figure 6.2a is the clustering generated by probabilistic algorithm. Figure 6.2b is the clustering generated by deterministic algorithm.

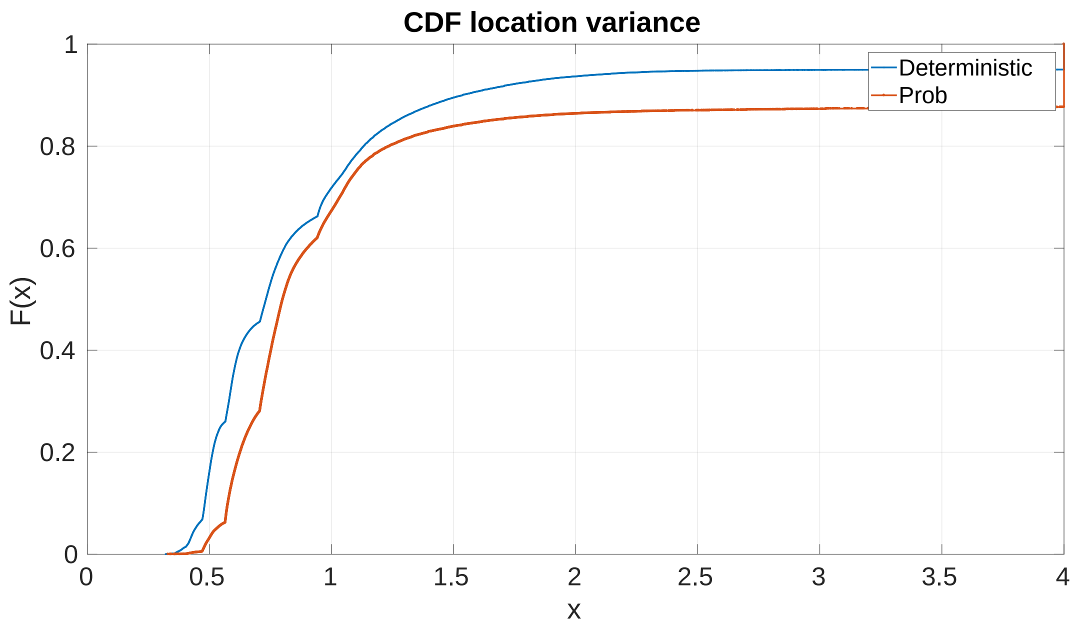


(a)

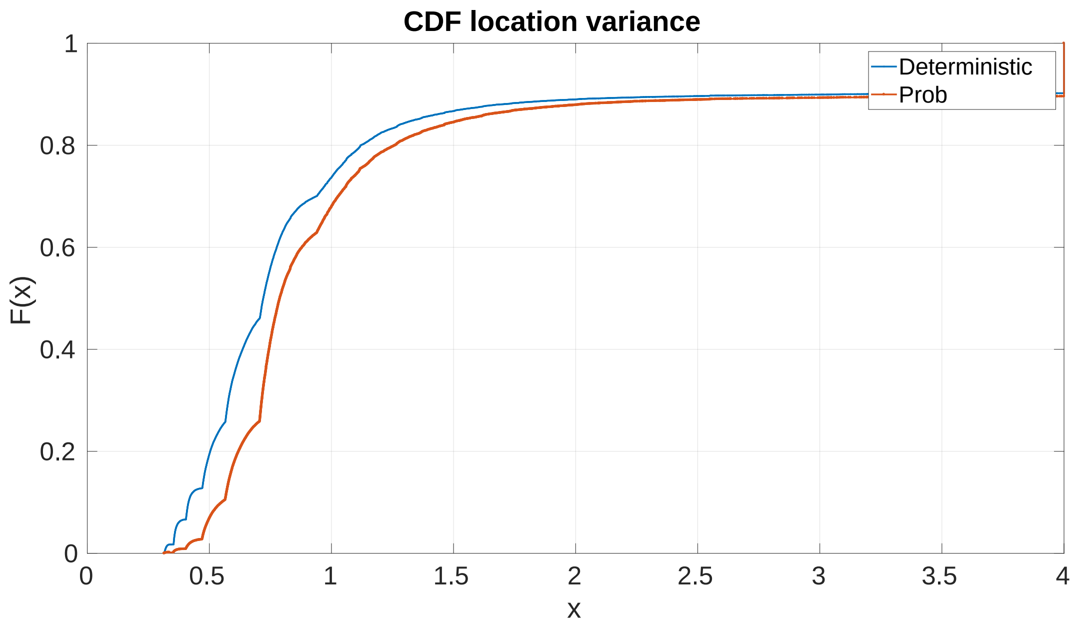


(b)

Figure 6.3: Simulated clustering of the open area. Blue nodes are cluster-heads, green nodes are unclustered and pink nodes are clustered. The background color is the calculated CRLB at the position given a set of anchor nodes with a fixed variance of 1 between all anchor nodes. Figure 6.3a is the clustering generated by probabilistic algorithm. Figure 6.3b is the clustering generated by deterministic algorithm.



(a)



(b)

Figure 6.4: CDF of the CRLB for the hallway example (Figure 6.4a) and open area (Figure 6.4b) after 200 simulations.

6.4 Conclusion

In this section, the performance of two simple iterative clustering algorithms was evaluated. Our simulation showed that already a very simple probabilistic iterative clustering algorithm could yield reasonable localization accuracy for different scenarios. The improved deterministic version of the simple iterative algorithm seems to produce higher overall localization accuracy as well as produce less overlapping areas and leaves fewer nodes unclustered. Further improvements may take the spatial information into account such that clusters, where the cluster-head is located at the center of the clusters, are preferred over clusters where the cluster-head resides at the border of a cluster.

CHAPTER 7

Conclusions and Future Work

This thesis proposes an indoor localization system that is self-organizing, scalable, and efficient. The system follows the "throw and go" principle, that is, homogeneous nodes can be randomly placed within a building, and the system can organize itself such that efficient localization with high accuracy is possible. Along the way to a final system design, several challenges must be solved, namely: localization, synchronization, and clustering. To allow for an efficient localization system the need for a TDOA system was identified in Chapter 2. Suitable algorithms for positioning, their performance and their sensitivity to noise are investigated in Chapter 4. The findings are that the linearised closed form solution can give reasonable results inside of the convex hull of a cluster if the measurement uncertainties are small. However,it fails if the measurement uncertainties are increased or if the final position is far outside of the convex hull. In general, the iterative algorithms perform better regarding localization accuracy and both methods converge to the CRLB inside of the convex hull of the cluster. The Gauss-Newton algorithm is more robust to wrong initial guesses than the TS method. Outside of the convex hull both methods require good initial guesses to converge. Future work may investigate how to verify the estimated position of the closed form solution, and how to find alternative solutions. A non-line-of-sight detection can be employed to discard measurements from anchor nodes that do not have line-of-sight condition. Moreover, at the moment it is assumed that the position of the anchor nodes are known beforehand, thus, suitable algorithm to identify the position of the anchor nodes via pairwise distance measurements has to be found.

In TDOA systems the measurement uncertainties heavily depend on the clock synchronization accuracy. In this thesis, two clock synchronization methods are compared in Chapter 5. The statistical parameters of the crystal oscillators of the DW1000 modules were identified via the Allan variance, and used in the Kalman filter for optimal offset and skew estimation. Compared to the simple skew estimation method the Kalman filter yields superior results with high synchronization rates, however, the simple skew estimation method is almost equally good at synchronization rates below 5Hz. During the evaluation, a high-temperature dependency was observed resulting in a change of the skew value. As the Kalman filter combines previous states with new measurement updates, it can lag behind the true skew value if only two states are used. To compensate for this, a third state was temporarily introduced in the Kalman filter to track the change of the skew value. On the one hand, this yields better results when the change of skew was constant. On the other hand, it took the Kalman filter much longer to converge to the true values if sudden changes or synchronization messages with large reception time uncertainties occurred. Future work might investigate this issue to find methods to reinitialize the Kalman filter if this happens such that a third state can be used.

In Chapter 6 two simple clustering algorithms were compared, and the theoretical localization accuracy was computed for clustered networks. The results suggest that the deterministic algorithm produces slightly higher localization accuracy and less overlapping clusters. Furthermore, the simulation for a single room in Section 6.3.1 shows the superior of the deterministic algorithm if only one cluster is required and the ideal cluster-head is the node which can connect the most nodes to its cluster. However, the results from Chapter 4 suggest that the localization algorithms do not always converge outside of the convex hull. Hence, the convergence of these algorithms should be investigated in more detail for larger areas with a clustered network. Further work may investigate how the spatial distribution can be taken into account for the cluster-head selection algorithm. Another issue that was not addressed in this thesis is a coarse time synchronization that is required by the time-depending deterministic clustering algorithm presented in Section 6.2.2. A solution might be the use of the Glossy synchronization method, but it has to be investigated if this method works reliably with many nodes using the DW1000 module.

The proposed communication and localization protocol in Chapter 3 allows for efficient and accurate passive localization as well as self-localization. The messages required for self-localization are placed at the beginning of each round. Thus, the accuracy for self-localization is increased. With this design choice, an arbitrary number of nodes can localize themselves with high accuracy. Something that has not been taken into account is the inter-cluster communication and how cluster interference can be mitigated. As proposed in Chapter 3 different preamble codes can be used for interference mitigation, and inter-cluster communication can take place at defined time intervals.

At the current stage, only the synchronization and message exchange is implemented on the Nucleo platform. To make the system usable, also the localization and clustering algorithms have to be ported to the Nucleo platform. Afterward, the localization accuracy must be evaluated for different real-world use cases.

CHAPTER 8

Appendix

This chapter describes the message format used to establish synchronization and to transmit TDOA data. As the clustering is not implemented yet we do not describe these messages. However, as the clustering requires only to transmit/receive the role of neighbor nodes these messages are expected to be rather simple.

uint8 packet_type: Each message contains an 8-bit packet-type identifier to distinguish between *synchronization*, *localization*, and *data* messages.

8.1 Synchronization Messages

enum_sync_t: Especially during the development phase, it was handy to let the reference node specify when data should be returned to the reference node and when mobile nodes are allowed to transmit data. With this value, we assign a whole round to a specific purpose. Possible values are:

- Only mobile nodes are allowed to send localization messages.
- Only anchor nodes are allowed to return data to the cluster-head
- None of the above, only synchronization is allowed.

Synchronization message:
Each synchronization message contains the following four fields:

- *uint8 packet_type*: This value identifies the type of the packet, it is used to call the appropriate packet handler.

- *uint32 pkt_id*: The packet ID or sequence number is an increasing number that is incremented by the reference node. It is used to identify the packet synchronization message uniquely. The purpose of this value is twofold: first, it is used to identify synchronization packet loss to adjust the skew estimation mechanism accordingly. Second, it is used for debugging purposes of relating any kind of events between two separate nodes.

- *uint8 txt[5]*: This is the 5 byte transmission time.

- *enum_sync_t interval_type*: This is the type of the current synchronization interval. As described previously, we assign each synchronization interval a special purpose such as allowing mobile nodes to broadcast *localization* packets or to allow anchor nodes to transmit data to the cluster-head.

- *uint8 crc[2]*: This array contains the CRC value of the received packet. Note that the DW1000 module already checks for the correctness of the CRC value. In case it is not correct the *RX ready* flag is not set.

8.2 Localization

Localization message format
The localization packet format is used by mobile nodes when they want to be localized. It contains all the information required by the TDOA system to relate the reception time-stamps of a localization packet at different anchor nodes.

- *uint8 packet_type*: This value identifies the type of the packet, and it is used to call the appropriate packet handler.

- *uint8 node_id*: This id is used to identify the node that issued a localization packet uniquely. This value can be assigned dynamically when a node registers at the TDOA system or at a cluster reference node. For this thesis, the node id is assigned statically.

- *uint32 local_packet_id*: This value is used to uniquely identify a localization packet.

TDOA measurement message
The packet type *packet_tdoa_meas_t* is used by the anchor nodes to return the measured reception time of a previous localization packet to the cluster head. At the current state, only one measurement is returned, but in a later version, the length (and therefore the number of returned TDOA measurements) can be extended.

- *uint8 packet_type*: This value identifies the type of the packet, and it is used to call the appropriate packet handler.

- *uint8 anchor_id*: This id is used to identify the anchor node that sent a data packet uniquely. This value can be assigned dynamically when clusters are spawned. For this thesis, the anchor id is assigned statically.
- *uint32 local_packet_id*: This is the packet id of the localization message that is reported to the cluster-head.
- *uint8 node_id*: This is the mobile node's id, which sent the localization message.
- *uint64 rxt*: This is the reception time of the localization message with id *local_packet_id* issued by mobile node with id *node_id* and received at anchor with id *anchor_id*.
- *uint8 crc[2]*: This array contains the CRC value of the received packet. Note that the DW1000 module already checks for the correctness of the CRC value. In case it is not correct the *RX ready* flag is not set.

For a final version, it would not be required to include the *local_packet_id* to match the reception time of the same localization packet at different anchor nodes. Instead, we propose to use the global time and to define a margin value, i.e., if a packet is received at anchor 1 at t_1 it is matched to a packet received at anchor 2 if its reception time t_2 lies within the margin value ma such that $t_1 - ma < t_2 < t_2 + ma$. Currently, we are using a 6-byte long value for the global time. Hence, we require 7 bytes (6 bytes + 1 byte for the *node_id* for a single TDOA measurement set to be sent to the cluster head).

The DW1000 module allows to send 127 byte in standard mode and 1024 byte in extended mode. The IEEE 802.15.4 header takes 7 bytes: *packet_type*, *anchor_id* and *crc* requires 4 bytes, all together 11 bytes. To gain the maximal amount of TDOA measurement data that can be transmitted in a single packet we have to subtract 11 from the maximal allowed transmitted bytes resulting in 116 bytes for the standard mode and 1013 for the extended mode, respectively. Dividing this values by 7 (6-byte time-stamp + 1-byte node_id) gives us the final amount of TDOA measurements that can be transmitted to the cluster head in a single packet. In the case of the standard mode, this would allow us to return 16 measurements and for the extended mode 144 measurements. For further optimization, we can reduce the maximal value of *rxt*. This can be done if we assume that all nodes receive the same message within the same synchronization interval. Thus the most significant bits do not differ for all measurements. For example, during a synchronization interval of 100ms, 34 bits are enough to express each time-stamp within the 100ms range such that the required size for a single TDOA measurement reduces to 42 bits. This results in up to 22 measurements for the standard mode and 192 measurements for the extended mode.

List of Figures

List of Tables

Bibliography

[1] K. Römer and F. Mattern, “Towards a Unified View on Space and Time in Sensor Networks,” *Computer Communications*, vol. 28, pp. 1484–1497, 2005.

[2] K. Stone and T. Camp, “A survey of distance-based wireless sensor network localization techniques,” *International Journal of Pervasive Computing and Communications*, vol. 8, pp. 158–183, 2012.

[3] R. T. Rajan and A.-j. V. D. Veen, “Joint Ranging and Synchronization for an Anchorless Network of Mobile Nodes,” *IEEE Transactions on Signal Processing*, vol. 63, no. 8, pp. 1925–1940, 2015.

[4] T. L. N. Nguyen and Y. Shin, “Matrix completion optimization for localization in wireless sensor networks for intelligent IoT,” *Sensors (Switzerland)*, vol. 16, no. 5, pp. 1–11, 2016.

[5] G. C. Calafiore, L. Carlone, and M. Wei, “A distributed Gauss-Newton approach for range-based localization of multi agent formations,” in *Proceedings of the IEEE International Symposium on Computer-Aided Control System Design*, pp. 1152–1157, 2010.

[6] G. Mao and B. D. O. Anderson, “Wireless sensor network localization techniques,” *Computer Networks*, vol. 51, no. 10, pp. 2529–2553, 2007.

[7] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, “Practical robust localization over large-scale 802.11 wireless networks,” in *Proceedings of the 10th annual international conference on Mobile computing and networking - MobiCom ’04*, p. 70, 2004.

[8] L. B, “Ultra-Wideband Angle-of-Arrival Tracking Systems,” Tech. Rep. December 2010, 2010.

[9] I. Dotlic, A. Connell, H. Ma, J. Clancy, M. Mclaughlin, A. Chambers, and P. Street, “Angle of Arrival Estimation Using Decawave DW1000 Integrated Circuits,” *Positioning, Navigation and Communications (WPNC), Workshop*, 2017.

[10] Decawave Ltd, “Application note: Sources of error in DW1000 based two-way ranging (TWR) schemes,” pp. 1–21, 2014.

[11] R. M. Buehrer and S. Venkatesh, “Fundamentals of Time-of-Arrival-Based Position Locations,” *Handbook of Position Location: Theory, Practice, and Advances*, pp. 175–212, 2011.

[12] R. Kaune, “Accuracy studies for TDOA and TOA localization,” *2012 15th International Conference on Information Fusion (FUSION)*, pp. 408 –415, 2012.

[13] A. Ledergerber, M. Hamer, and R. D’Andrea, “A robot self-localization system using one-way ultra-wideband communication,” *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 3131–3137, 2015.

[14] A. Engel and A. Koch, “Accelerated Clock Drift Estimation for High-Precision Wireless Time-Synchronization,” in *2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)*, pp. 627–635, 2015.

[15] H. Schwartz, T. Kunz, N. Charles, and Z. Hui, “Frequency Accuracy & Stability Dependencies of Crystal Oscillators,” Tech. Rep. September, 2014.

[16] W. Dargie and C. Poellabauer, *Fundamentals of Wireless Sensor Networks: Theory and Practice.* Wiley Publishing, 2010.

[17] K. Römer, P. Blum, and L. Meier, “Time Synchronization and Calibration in Wireless Sensor Networks,” in *Handbook of Sensor Networks: Algorithms and Architectures*, pp. 1 – 39, 2005.

[18] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, “The flooding time synchronization protocol,” in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys ’04, (New York, NY, USA), pp. 39–49, ACM, 2004.

[19] J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 147–163, Dec. 2002.

[20] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, “Clock Synchronization of Wireless Sensor Networks,” *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 124–138, 2011.

[21] S. Ganeriwal, R. Kumar, and M. B. Srivastava, “Timing-sync Protocol for Sensor Networks,” in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pp. 138 –149, 2003.

[22] S. M. Lasassmeh and J. M. Conrad, “Time Synchronization in Wireless Sensor Networks: A Survey,” in *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, pp. 242–245, 2010.

[23] S. Yoon, C. Veerarittiphan, and M. Sichitiu, “Tiny-sync: Tight time synchronization for wireless sensor networks,” *Transactions on Sensor Networks*, vol. V, pp. 1–33, 2007.

[24] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 73–84, 2011.

[25] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '99, (New York, NY, USA), pp. 151–162, ACM, 1999.

[26] O. Younis, M. Krunz, and S. Ramasubramanian, "Node Clustering in Wireless Sensor Networks: Recent Developments and Deployment Challenges," *IEEE Network*, vol. 20, no. 3, pp. 20–25, 2006.

[27] M. M. Afsar and M.-H. Tayarani-N, "Clustering in sensor networks: A literature survey," *Journal of Network and Computer Applications*, vol. 46, pp. 198–226, 2014.

[28] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, vol. 2, pp. 1–10, 2000.

[29] M. J. Handy, M. Haase, and D. Timmermann, "Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection," in *4th International Workshop on Mobile and Wireless Communications Network*, 2002.

[30] C.-y. Wen and W. A. Sethares, "Automatic Decentralized Clustering for Wireless Sensor Networks," *EURASIP J. Wirel. Commun. Netw.*, vol. 2005, no. 5, pp. 686–697, 2005.

[31] S. Basagni, "Distributed clustering for ad hoc networks," *Proceedings Fourth International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*, pp. 310–315.

[32] S. Mudundi and H. Ali, "A new robust genetic algorithm for dynamic cluster formation in wireless sensor networks," in *Proceedings of the 7th IASTED International Conferences on Wireless and Optical Communications, WOC 2007*, pp. 360–367, 2007.

[33] H. Chan and A. Perrig, "ACE: An Emergent Algorithm for Highly Uniform Cluster Formation," *Wireless Sensor Networks*, vol. 2920, pp. 154–171, 2004.

[34] J. Tiemann, F. Schweikowski, and C. Wietfeld, "Design of an UWB Indoor-Positioning System for UAV Navigation in GNSS-Denied Environments," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–7, 2015.

[35] J. Tiemann, F. Eckermann, and C. Wietfeld, "Multi-User Interference and Wireless Clock Synchronization in TDOA-based UWB Localization," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–6, 2016.

[36] J. Tiemann, F. Eckermann, and C. Wietfeld, “ATLAS - An Open-Source TDOA-based Ultra-Wideband Localization System,” in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–6, 2016.

[37] G. Shen and R. Zetik, “Performance Comparison of TOA and TDOA Based Location Estimation Algorithms in LOS Environment,” in *5th Workshop on Positioning, Navigation and Communication*, pp. 71–78, 2008.

[38] K. W. Cheung, H. C. So, and Y. T. Chan, “Least Squares Algorithms for Time-of-Arrival-Based Mobile Location,” vol. 52, no. 4, pp. 1121–1128, 2004.

[39] D. Kegen, I. Sharp, and P. Y. J. Guo, *Ground-Based Wireless Positioning*. 2009.

[40] M. Pelka, “Position Calculation with Least Squares based on Distance Measurements,” tech. rep., Lübeck University of Applied Sciences, 2015.

[41] J. Smith and J. Abel, “The spherical interpolation method of source localization,” *IEEE Journal of Oceanic Engineering*, vol. 12, no. 1, pp. 246–252, 1987.

[42] C. Abou-rjeily, “Joint Distributed Synchronization and Positioning in UWB,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, no. 4, pp. 1896–1911, 2006.

[43] C. Zucca and P. Tavella, “The Clock Model and Its Relationship with the Allan and Related Variances,” *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 52, no. 2, pp. 289–296, 2005.

[44] G. Giorgi and C. Narduzzi, “Performance Analysis of Kalman-Filter-Based Clock Synchronization in IEEE 1588 Networks,” *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 8, pp. 2902–2909, 2011.

[45] J. H. Kotecha and P. M. Djuric, “Gaussian Sum Particle Filtering,” *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2602–2612, 2003.

[46] I. Standards and C. Committee, “IEEE Std 1139-2008 (Revision of IEEE Std 1139-1999) IEEE Standard Definitions of Physical Quantities for Fundamental Frequency and Time Metrology–Random Instabilities,” vol. 2008, no. February, 2009.

[47] A. Mallat and L. Vandendorpe, “Joint estimation of the time delay and the clock drift and offset using uwb signals,” in *2014 IEEE International Conference on Communications (ICC)*, pp. 5474–5480, June 2014.

[48] M. A. Lombardi, “Fundamentals of Time and Frequency,” in *The Mechatronics Handbook*, pp. 17.0 – 17.18, 2002.

[49] G. Panfilo and P. Tavella, “Atomic clock prediction based on stochastic differential equations,” *Metrologia*, vol. 45, no. 6, p. S108, 2008.

[50] B. Øksenda, *Stochastic Differential Equations, Linear Filtering, and Chaos Expansion.* 1993.

[51] L. Galleani, "A tutorial on the two-state model of the atomic clock noise," *Metrologia*, vol. 45, no. 6, p. 175, 2008.

[52] D. W. Allan, "Statistics of Atomic Frequency Standards," *Proceedings of the IEEE*, vol. 54, no. 2, pp. 221–230, 1966.

[53] X. Niu, Q. Wang, Y. Li, Q. Li, and J. Liu, "Using inertial sensors in smartphones for curriculum experiments of inertial navigation technology," *Education Sciences*, vol. 5, pp. 26–46, 03 2015.

[54] P. Aimonen, "Basic concept of kalman filtering," 2007.

[55] D. Neirynck and M. Mclaughlin, "Comparison of wireless clock synchronization algorithms for indoor location systems," in *2014 IEEE International Conference on Communications Workshops (ICC)*, pp. 157–162, Decawave Limited, 2014.