# Ultra-Wideband Communication and Relative Localisation for Swarming Robots

G.M. ter Horst

Master of Science Thesis

Delft University of Technology

Embedded and Networked Systems

# Ultra-Wideband Communication and Relative Localisation for Swarming Robots

Master of Science Thesis

For the degree of Master of Science in Embedded Systems
at Delft University of Technology

G.M. ter Horst

March 12, 2019

Faculty of Electrical Engineering, Mathematics and Computer Science
(EEMCS) · Delft University of Technology

### Author

G.M. ter Horst

### Title

Ultra-Wideband Communication and Relative Localisation for Swarming Robots

### MSc Presentation

26 March 2019

### Graduation Committee

Prof.dr.ir. K.G. Langendoen
Dr.ir. A. Noroozi
Ir.dr. C.J.M. Verhoeven

# Abstract

A robot swarm of zebros (Dutch: ZEs Benige RObots) is being developed at the TU Delft, with the purpose of forming a large self-deploying sensor network that survey remote locations without the need for any pre-existing infrastructure, or be used for evaluating swarming algorithms in the field. Towards this goal, zebros need to be able to communicate with and localise nearby neighbours. Existing localisation solutions showed to be inadequate for the task, because they generally do not allow for localisation without existing infrastructure [14] [39], do not allow scaling to a large swarm, either because they are very communication-intensive [36], because they make use of global knowledge [36] [56] or because they rely on robot behaviour, which introduces a dependency of localisation on a specific swarming algorithm [30] [50]. We present a fully-localised method of localisation called Tangolation, which estimates the location of a neighbour as a range and an angle, accompanied by a confidence value from 1 (low) to 5 (high). With Tangolation, two nodes estimate each other's next location from a range measurement using Two-Way Ranging (TWR), and the exchange of the nodes' displacement since the last range measurement. Therefore, the two nodes only need each other to determine each other's location as long as at least one of the two is moving, allowing Tangolation to scale well to larger swarms. No knowledge about the network topology or control over robot behaviour is needed, and communication between localising nodes only needs to happen sparsely, at a frequency of once every five seconds per neighbour.

A fully distributed Time Division Multiple Acces (TDMA) protocol named Anarchic TDMA (AN-TDMA) was devised to support communcation within the robot swarm, where nodes synchronise the slot start times without the need for a special coordinating node that indicates the start of slots or frames. AN-TDMA showed to increase the reachable channel utilisation ratios by 55% to 67% compared to the ALOHA method recommended by Decawave [21].

For TWR, the Decawave DW1000 transceiver was used, and the BNO055 IMU aided in displacement measuring. Tangolation was tested in a simulation with realistic conservative estimates of the noise over the range and displacement measurements. Even using conservative noise estimates, at least 95% of the estimated location angles are within the required 22° of the true value. Depending on the confidence level that Tangolation reports, 65% to 84% of the angle estimates are within 10° of the true value, which shows that Tangolation more than meets the set requirements.

# Acknowledgements

First of all I would like to thank prof. dr. ir. Koen Langendoen for his support and guidance during this thesis, and for taking the time to proofread even when the time to finish this thesis got rather tight.
Special thanks go to dr. Arash Noroozi, who stepped in as a second supervisor halfway down the project and immediately started asking challenging questions.
I would like to thank everybody in the Zebro project, for their knowledge they shared, for their help and support during this thesis project, and for providing me with free coffee.

Last but not least, I am very grateful for the mental support I received from my parents and Daniélle, who are always there for me.

Delft, University of Technology G.M. ter Horst
March 12, 2019

# Table of Contents

# Chapter 1

# Introduction

Swarm robotics have been subject to research for decades. The term "swarm robotics" was first introduced as a buzz word in 1988 by Beni, to replace the less catchy term "cellular automation" [6]. He used it to describe a group of robots that has some special characteristics that are found in swarms of insects or birds: decentralisation, asynchronous behaviour and simple homogeneous members.

The broad applicability of robot swarms and animal-like behaviour [4] [44] has resulted in a large number of existing swarms consisting of cheap and simple robots, such as the Kilobot swarm [49] and Colias [3] amongst others [27] [52] [54]. Not just robotics researches draw their inspiration from animals. Companies such as Boston Dynamics are inspired by animals in their robotics projects, as is clear from their Spot and SpotMini robots [9] [10]. However, the robots by Boston Dynamics are extremely expensive. Although they can get by on many different terrains, they are therefore not suitable for large homogeneous swarms. This in contrast with the existing swarms that consist of cheap and simple robots, but are made to be studied in a lab. They rely on line-of-sight communication [49] [54], existing routing and localisation infrastructure [3], and a managed environment. To the knowledge of the author, the only swarm that has been tested in an uncontrolled environment is a small swarm of at most ten Aquatic Surface Robots [27], with the use of GPS and centralised WiFi communication.

At the TU Delft, *zebros* are developed as a new type of swarming robot. These A4-sized, six-legged robots (Dutch: ZEs Benige RObots, hence the name) are designed as a simple, cheap and modular sensor platform, and can cope with uneven surfaces (see Figure 1-1). This swarm can serve as

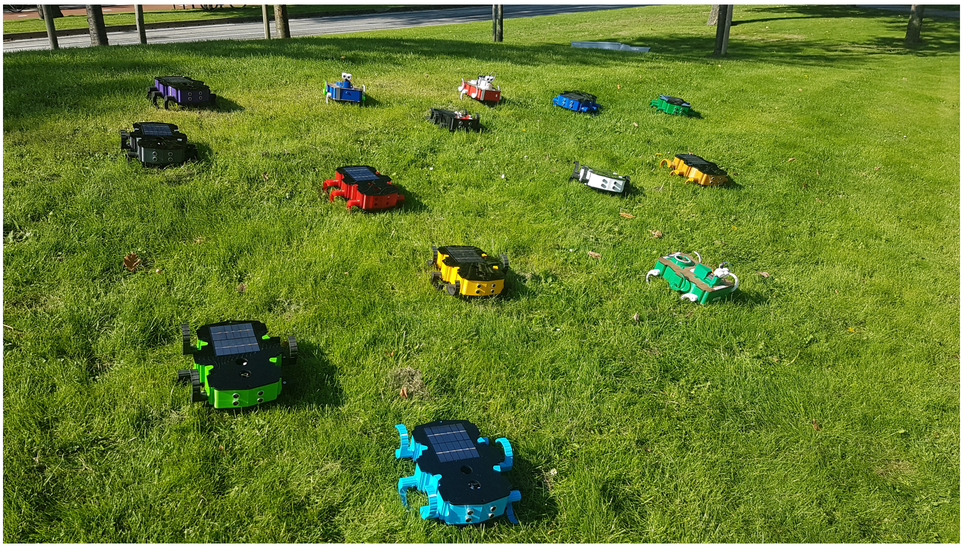


**Figure 1-1:** Zebros at the TU Delft campus.

a research platform for both indoor and outdoor sensing and monitoring, but also for animal swarm behaviour, or as an aid in search and rescue operations.

## 1-1 Problem Statement

Currently, a zebro can walk around autonomously. Swarming behaviour is the next step, and a number of swarming algorithms that allow group behaviour and collision avoidance have been designed for the zebros. However, these algorithms still only live in simulations. The missing link is a module that provides zebros with a location estimate of their nearby neighbours, in the form of a range and and angle, relative to where the robot is facing. With these relative locations, the swarming algorithms can be tested in the real world, in an uncontrolled environt.

Figure 1-2 shows the high-level architecture of a zebro. Swarming algorithms are part of *Toplevel*, and are the highest level in the hierarchy. *Toplevel* gathers information from *Observation* and *Localisation*, and tells *Locomotion* whether the robot should go forward, make a corner, or turn around. *Locomotion* then translates these commands to drive the six leg controllers of the zebro.

Previous work by J. Miog [40] has investigated the possibilities for wireless communication and localisation. He found that the ultra-wideband technology provided by Decawave seems promising due to its resilience against multipath fading, high datarate and low power consumption. Ultra-wideband

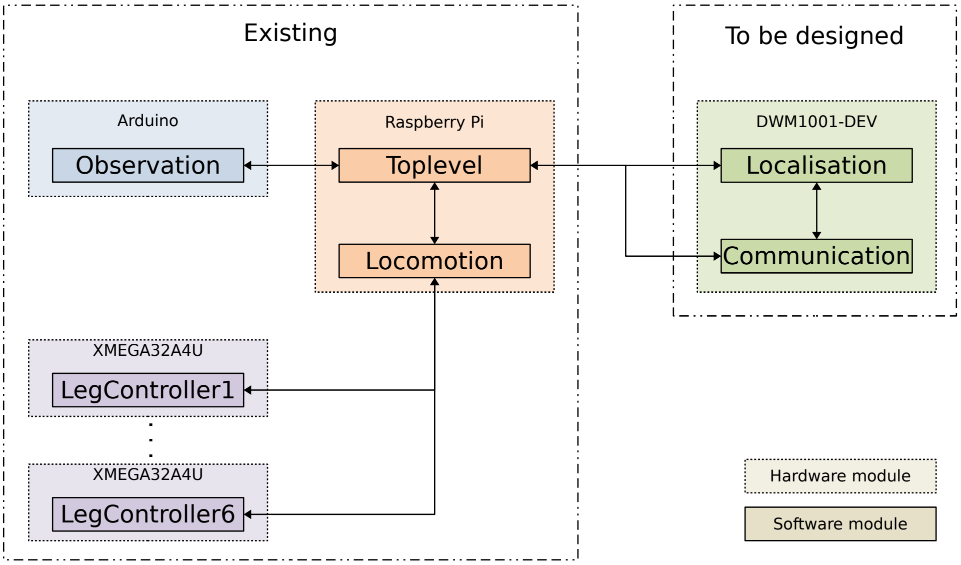


**Figure 1-2:** High-level functional architecture of a zebro.

also allows accurate message timestamping, making it an excellent candidate for Time-of-flight ranging [21] [40] with a claimed precision of $10cm$.

This thesis work will describe the design and implementation of the *Localisation* and supporting *Communication* modules as shown in Figure 1-2, to, as far as we know for the first time, allow a larger group of autonomous robots to swarm in an uncontrolled environment.

Decawave provides a small package in the form of the DWM1001-DEV development board, in which its DW1000 ultra-wideband transceiver is accompanied by a 64MHz nRF52832 ARM Cortex-M4 microprocessor with 64kB of RAM and 512kB flash [22]. If possible, this board is to be used, because it obviates the need for a custom designed PCB.

## 1-2 Requirements and constraints

The existing swarming algorithms give some sense of what is needed from a localisation point of view. Combined with the intended purpose of the zebro swarm, the following problem statement has been defined:

*Design and implement a pluggable module with ultra-wideband ranging that allows zebros to read out their neighbours' relative locations, such that*

R.1 The relative locations of the at least seven closest neighbours at a distance of $1 - 20m$ can be estimated [16].

R.2 The range accuracy shall be within 10% of the relative distance between neighbours [16].

R.3 The angular accuracy shall be at within 22°, and the average angular accuracy must be within 12°[16].

R.4 The module can be used effectively in both small and large swarms of $2 - 50$ nodes, preferably scalable to even larger swarms.

The requirements listed above should be met within the following constraints:

C.1 The localisation module does not need control over robot behaviour.

C.2 The localisation module does not need any external infrastructure to be in place.

## 1-3 Contributions

This work will consist of two major contributions. The main contribution is a localisation system for the zebros. The secondary contribution is a MAC layer protocol to support the localisation system, and possibly facilitate additional wireless communication. Both contributions will be introduced in more detail below.

### 1-3-1 Localisation

The first contribution will be the design and implementation of a pluggable hardware module that provides zebros with the relative locations their nearby neighbours, based on the DW1000 ultra-wideband transceiver. This localisation module is able to localise a neighbour with information just from that neighbour, so no information about or control of network topology is needed. The core idea is that two zebros determine *i)* their distance to each other every $\approx 5s$, and *ii)* exchange their displacement between the measurements. The combined info is sufficient to estimate a neighbour's relative location as a range and an angle. The solution is inherently scalable to large swarms because no information about network topology is needed, only local information is required, and communication only needs to happen infrequently. The reliance on at least one of two nodes moving is what gives this contribution its name: Tangolation.

### 1-3-2 Communication

The second contribution of this work will be the design and implementation of Anarchic TDMA (AN-TDMA), a TDMA-based MAC layer protocol that serves as support for the localisation layer. Nodes synchronise their belief about slot timings by listening to their neighbours, thereby obviating the need for a special coordinating node that indicates the start of slots and frames. Every node behaves identically, and can be added to or removed from the network at any time. The length of a slot has been designed such that it is long enough to support one Two-Way Ranging (TWR) message exchange.

## 1-4 Structure

This thesis report is structured as follows. An overview of the state-of-the-art will be given in Chapter 2. After that, Chapter 3 will describe the design of Tangolation, and the design of the supporting communication layer. Their implementations can be found in Chapter 4, followed by an evaluation in Chapter 5. A short conclusion is presented in Chapter 6, along with some discovered limitations and pointers for future work.

# Chapter 2

# Related Work

Both localisation and MAC protocols in the context of ad-hoc mesh networks have been discussed extensively in literature. To give the reader a context to place this work in, this chapter provides an overview of the state of the art. Localisation options are discussed in Section 2-1, and several decentralised MAC protocols are described in section 2-2.

## 2-1 Localisation in a mobile network

Localisation of nodes in a network is not a new problem, and has been covered extensively in literature. Most of this research covers localisation using some static infrastructure, such as anchors, or investigates node localisation of static nodes in a Wireless Sensor Network (WSN). Since these solutions rely on assumptions that are not valid for zebros, this literature will not be discussed. Instead, this section will purely focus on localisation in mobile networks without existing infrastructure.

### 2-1-1 Localisation using global state

When all distances between nodes in a network are measured, it becomes possible to estimate the topology of the network. The better connected a node is, the better the estimated location because of the higher number of measurements. It is shown in [28] that even when using Received Signal Strength (RSS) data, nodes in a static network can be localised fairly accurately, with better localisation where the network is denser. When nodes are moving, however, the topology of the network changes. Depending on the method of distance measurement, it may not be possible to determine the ranges between nodes at the same moment in time, resulting in a measured state that has never existed in reality. A solution to this problem is to include odometry data, and add timestamps to both range measurements and odometry [48].

Work by Liu [37] shows cooperative localisation for humans using distances and odometry data, where on a central computer range measurements and odometry measurements were merged with a particle filter to obtain location estimates. Range measurements were acquired using ranging nodes that four test subjects carried. Due to very noisy inertial measurements and quick degradation in range detection when the distance between nodes increases above $8m$, the accuracy was very low: generally around two meters, and at some point increasing to four to six meters. Assuming nodes $8m$ apart, the error in angle is thus between $atan(2/8) \approx 14°$ and $atan(6/8) \approx 37°$, which is more than the allowed error of at most $22°$ for the zebro localisation. For nodes closer than $8m$, the angular error will be even larger.

Roumeliotis and Bekey ([48]) showed that a Kalman filter can be used to track the global network topology in a partly decentralised way, by decomposing the propagation cycle into multiple smaller filters. Each of these filters runs locally on a robot, and information only needs to be exchanged during the update phase of the Kalman filter. The drawback of this method is that the number of nodes in the network needs to be known in advance. Additionally, during the update phase of the Kalman filter, everybody needs to communicate their new information to everybody.

In a group of more than two robots, the update phase can only take place when all information has propagated through the network. Therefore, it should be knowable when every node in the network is aware of some piece of information. Yang et al [56] solve this problem by the network topology and the communication scheme. A Kalman filter for relative localisation is used in a small group of Unmanned Aerial Vehicle (UAV)s and tested with different predefined communication schemes, which guarantee that information is propagated over the network in a certain number of steps. Depending on the communication scheme, information was guaranteed to be propagated through the network in one timestep (complete network connectivity) or three timesteps (pairwise communication between predefined pairs). Differences between the two methods were negligible with regard to localisation accuracy, showing that a completely connected network is not necessary as long as it is known in advance which nodes are connected to each other and thus the communication schemes can be known. For potentially large swarms such as the zebro swarm, it is not ensured that information is propageted in one timestep. Because the localisation module will have no control over the behaviour of zebros, it is also not known in advance who can communicate with who: as far as localisation is concerned, zebros are free to go wherever they want, thus changing the network topology continuously and inpredictably.

When it has to be guaranteed that information has been propagated through the network within a certain number of rounds, the information has to be remembered locally until and can only be discarded when all nodes know about it. Two bots that meet each other for the first time have to exchange all their past information to make sure every bot runs the same information through their Kalman filters. When there are no guarantees about information propagation, information has to be remembered indefinitely. This was solved by Leung et al.[36], who obviated the need for predefined communication schemes with an algorithm to determine when previous state information can be discarded, using the notion of checkpoints. When a checkpoint for timestep $t_i$ arises at timestep $t_n$ where $i < n$, it can be proven that at $t_n$ everyone in the network has knowledge of the state at $t_i$. Therefore, all information before $t_i$ does not need to be propagated to neighbours anymore, and can thus be discarded. This method has the added benefit that the number of nodes in the network does not need to be known initially, provided that the communication range is at least as long as the range measurement range. When using the intended ultra-wideband ranging method, ranging is achieved by communication. Therefore, the assumption that communication range is at least as long as the measurement range is a valid one for the zebro swarm. On the other hand, this method does introduce an increase in nominal memory usage, which increases with the size of the swarm. While the nominal memory usage is around $30kB$ for a network of seventeen nodes, peak memory usage far exceeds the $64kB$ available memory on the Cortex-M4 chip that we intend to use.

In summary, methods that infer a relative localisation estimate from a global state seem inapplicable to zebros, because all available information needs to be propagated over the swarm at all times. Methods that solve the problem of knowing when information has propagated to all nodes in the network either do this with the help of specific network topologies, or have high memory requirements.

### 2-1-2 Localisation using landmarks

Another option is to use known landmarks. Initially this has been researched in the nineties, mostly as a more flexible alternative to the basic line following robots that existed [42], [7], [5]. More recently, in 2016, the US airforce has investigated the possibility of localisation based on anomalies in the earths magnetic field [14]. However, these methods all assume the existence of known landmarks at predefined locations. This has the major drawback of using landmarks from the environment is that either it introduces a dependency on pre-existing infrastructure such as GPS or anchors, or it depends on the environment around the robot to be known.

Kurazume et al. note that robots within the swarm may serve as landmarks, by repeating what they call move-and-stop actions [34]. The robot swarm is divided into two groups, where each group alternately acts as stationary landmarks for the other moving group. Very recently, this approach has gained new interest. Work by Lockspeiser [38] investigates the accuracy of this method for several kinds of range measurement for robots on land. A similar routine is used by Matsuda [39] for autonomous underwater vehicles that alternately become static landmarks at the seafloor, as an alterative to localisation via vehicles at the surface with a GPS connection.

With these methods, localisation does not rely on knowledge and control over the environment. Instead, localisation has become an integral part of robot behaviour. Unfortunately, alternating landmarks thus directly violates the constraint that localisation may not have control over zebro behaviour.

### 2-1-3 Localisation using local knowledge

Recently, more research has emerged that focuses on localisation based purely on local knowledge, without the need for knowledge of a global state or common neighbours.

An example is the work of Guo et al [30], where a stationary UAV localises a moving neighbour based on distance measurements and its odometry data. While this method still relies on a common reference frame (e.g. magnetic North), Strader et al. [50] notice that when two robots move in a straight, intersecting path, no other data but relative distances are needed to determine that a neighbour must be in one of four locations given a strict set of assumptions. The cost is that very frequent measurements are needed, and data is only valid when both robots move in a straight line.

Allowing only movement in straight lines is a rather strict requirement, and testing with zebros showed that they generally do not walk in straight lines. Timing differences between when feet touch the ground introduce corners; in most cases deliberatly, but inherent small differences induced by small differences in motor speed or an uneven ground cause zebros to never walk completely straight. Trawny [53] removes the requirement of moving in straight lines by adding odometry data. He shows that even without a global reference plane, it is possible for two robots to localise each other when both move. The added complexity makes an unoptimized version of the algorithm take three to four seconds for a single robot-to-robot localisation. Optimization reduced this to $125ms$ on a desktop computer running matlab. This solution does direct neighbour-to-neighbour localisation, and thus does not need any information from robots other than the one it needs to localise.

It does not rely on any external factor, such as a global reference, and would therefore be perfect for both indoor and outdoor localisation, even in large swarms. The major drawback currently is the computational power that is needed even for the optimized algorithm.

Clearly, it is possible to estimate the location of a neighbour purely with information from that neighbour. However, existing methods have shown to be computationally expensive ([53]), require one stationary node (e.g. [30]) or are very communication intensive and rely on unrealistic assumptions ([50]).

## 2-2 MAC protocols for mobile nodes

This section discusses several types of MAC protocols, with a focus on moving ad-hoc mesh networks.

### 2-2-1 Carrier sense based medium access control

The standard MAC protocol for wireless sensor nodes is defined by IEEE 802.15.4 [33], supporting both a star and a peer-to-peer network architecture based on Channel Sensing Multiple Access with Collision Avoidance (CSMA-CA). Exactly one device can be elected as Personal Area Network (PAN) Coordinator, who will determine a unique PAN ID for the network that is not in use by any other network in communication range. An optional second responsibility is the transmission of beacon frames that signal the start of a new superframe, which consists of a slotted active period and an optional inactive period during which no communication happens.

In a Channel Sensing Multiple Access (CSMA) based wireless protocol, nodes listen if the channel is free, and send a Request To Send (RTS) message to claim the channel and request transmission to a certain neighbour. That neighbour answers with a Clear To Send (CTS) to make sure that its neighbours do not interfere with the upcoming transmission. When the inital transmitter has received the CTS, it is allowed to transmit its data.

In the early 2000s, Ding noticed that a protocol such as CSMA-CA, where the channel is claimed by a node using RTS and CTS messages, would have a dramatic impact on communication performance when the channel acquisition time is high [26]. It is shown that Time Division Multiple Acces (TDMA) performs at least as well as or better than CSMA-CA in terms of delay and throughput. Especially for ultra-wideband this is an important finding. Ultra-wideband preambles are generally long, ranging from 33% (best case, 64 symbol preamble and 127 byte data) to 99.9% (worst case, 4096 symbol preamble, 1 byte data). Because of the long preambles, the "short" RTS and CTS messages each take almost as much airtime as the actual message they claim channel acquisition for, thus defeating their purpose.

### 2-2-2 Time division based medium access control

More recently, it was shown that for ultra-wideband communications, a TDMA based protocol outperforms the more simple traditional ALOHA or CSMA-CA based protocols [47].

In the context of a constantly changing network, TDMA has the major drawback that it is generally scheduled by a central coordinator that marks the beginning of superframes and hands out timeslots to nodes, similar to the PAN coordinator from the IEEE802.15.4 standard [33]. Due to this centralised approach, traditional TDMA is inherently unscalable. Nodes can move outside the communication range of the PAN coordinator, and even with multiple PAN coordinators there is never the guarantee that every node is within communication range of one of them.

An alternative to centralised TDMA, is that of a distributed approach. A number of distributed slot selection algorithms exist [31] [43] [46] [57]. Some of these methods use TDMA combined with CSMA-CA [31] [43], but this would be desastrous for channel utilisation as explained in Section 2-2-1: the “short” messages to request access to the medium are only slightly smaller than the actual message that will be transmitted, thereby defeating their own purpose. Others need to transmit multiple messages before a slot can be determined, so when the network topology inevitably changes, a node needs to communicate with others to determine a new slot, thus likely introducing interference [46]. The method by Zhu requires the assumption that the network topology does not change during certain update phases [57]. For the intended zebro application, this assumption is not valid.

Degesys et al.[25] introduce the notion of *desynchronisation*, where nodes in a fully connected network do not try to perform their periodic operation at the same time, but rather at the opposite time. For a fixed superframe period, a fair schedule emerges when nodes try to schedule their transmission time exactly inbetween the nodes sending before and after.
Further research extending this notion to multi-hop networks resulted in a proposal for a two-hop version, where it was shown that the hidden node problem can by solved by looking at the constraint graph instead of the communication graph, and that that converges to a solution without collisions [24].

Muhlberger and Kolla [41] show that the constraint graph can be locally created when nodes communicate their neighbours’ phases. Their EXTENDED-DESYNC algorithm converges to a stable solution in a multi-hop network when communication links are symmetric. Real-world tests show that the actual performance is reasonably well predicted by its stochastic model [13] [12]. The approach of nodes listening to their environment to determine their schedule is a very appropriate method for a mobile robot swarm such as the zebro swarm. No central node is needed to manage scheduling, and timeslots can be reused in multi-hop networks, allowing the swarm to scale indefinitely in size as long as the density of the swarm is not too high. Possible difficulties are that nodes need to process each and every incoming message, even when it is not addressed to them, and the regular exchange of all neighbours' addresses and phases may be more data than fits in a message.

A similar synchronisation method was found by looking at fireflies [11], that were found to adapt their own flashing rate to that of their neighbours by adding a constant to their flashing phase until they are synchronized. This idea was adapted in the Reachback Firefly Algorithm for wireless sensor networks by adding a refractory period [35] and assuming MAC-level timestamping [55].

### 2-2-3 Frequency or code division based medium access control

While TDMA achieves resource sharing by synchronising nodes so they do not transmit at the same time, Frequency Division Multiple Acces (FDMA) and Code Division Multiple Acces (CDMA) allow simultaneous transmissions on separate channels.

Many FDMA or CDMA algorithms for ad-hoc networks rely on a static code assignment [32] [29], where a code or frequency is constant for a certain transmitter, receiver or for a specific pair of nodes. A drawback of this approach is that there should be as many channels as there are transmitters or receivers, respectively. In the case where each pair of $N$ nodes has a specific channel assigned, the number of channels even increases with $O(N^2)$.

To support more nodes than channels, Al-Meshhadany proposes an algorithm to use one subchannel for RTS and CTS messages to temporary claim one of the other subchannels, and use that claimed subchannel for the actual data transmission [2]. Assuming no interference and N subchannels, this method allows N-1 simultaneous connections.

Methods like these could also be applied on top of any TDMA scheme. Although the proposed ultra-wideband chip supports multiple frequencies, accurate ranging at a certain frequency requires an antenna designed for that frequency. Achieving orthogonality with the use of code division multiplexing is supported without having to change hardware, though tests by Decawave have shown that interference still exists even when using both different channels and different preamble codes [20].

# Chapter 3

# Design

This chapter covers the design of a solution to provide the zebros with decentralised localisation and communication. The implementation of the design will be covered in Chapter 4. Table 3-1 summarises the existing robot localisation methods from Chapter 2, and gives an overview of their strengths and weaknesses with regard to the context and requirements from Section 1-1.

Clearly, existing robot localisation methods do not tend to lend themselves well to large robot swarms in an uncontrolled, unknown environment. That motivates the design of a novel localisation method, of which this chapter explains the design choices and, to a lesser extent, some implementation details.

Section 3-1 covers the design of the localisation method, and the supporting MAC layer design is discussed in Section 3-2.

## 3-1 Localisation

From Table 3-1 it is easily seen that none of the existing localisation methods are both scalable to large swarms, and independent of any external infrastructure. Combining these two factors will drive the design of the localisation module, and the rationale given in Sections 3-1-1 and 3-1-2 will already provide some intuition about the final solution to the localisation problem, as described in Section 3-1-3.

**Table 3-1:** None of the investigated localisation methods meets all requirements within the context given in Chapter 1.

| | Nodes move uncontrolled | No external infrastructure | Scalable to large swarms | Links can change | Number of nodes can change |
|---|---|---|---|---|---|
| Distributed Multi-Robot Localisation [48] | Yes | Yes | No | Yes | No |
| Cooperative relative positioning of mobile users by Fusing IMU inertial and UWB ranging information [37] | Yes | Yes | No | Yes | No |
| Cooperative UAV Navigation using Inter-Vehicle Ranging and Magnetic Anomaly Measurements [56] | Yes | Yes | No | No | No |
| Decentralised Localisation of Sparsely-Communicating Robot Networks [36] | Yes | Yes | No | Yes | No |
| Absolute Postioning using the Earth's Magnetic Anomaly Field [14] | Yes | No | Yes | Yes | Yes |
| An Experimental Study of a Cooperative Positioning System [34] | No | Yes | Yes | Yes | Yes |
| Experimental Evaluation of Accuracy and Efficiency of Alternating Landmark Navigation by Multiple UAV's [39] | No | Yes | Yes | Yes | Yes |
| Relative Localisation for Quadcopters using Ultra-Wideband Sensors [30] | No | No | No | Yes | Yes |
| Cooperative Relative Localisation for Moving UAVs with Single Link Range Measurements [50] | No | Yes | No | Yes | Yes |
| On the global optimum of planar, range-based robot-to-robot relative pose estimation [53] | Yes | Yes | No | Yes | Yes |

### 3-1-1 Design for independency of external infrastructure

External infrastructure is a rather vague concept, and may need some further introduction to show in what direction it drives the design of our new localisation method.

The most obvious type of external infrastructure is probably the use of beacons or anchors. The way Decawave have setup their indoor localisation system is to place static nodes at known locations. Mobile nodes measure their distance to the static nodes using Two-Way Ranging (TWR) or Time Difference of Arrival (TDoA), and calculate their location. This method can serve an incredibly high number of mobile nodes, in particular when the static nodes are synchronized. In that case, a mobile node needs only to transmit a single message to let the static nodes calculate its location. However, having to place anchors for a robot swarm that is, given the requirements (Section 1-2) meant to cover possibly large, dangerous or hardly accessible areas is clearly not an option: robots need to localise the relative position of other robots without help of sensors that exist outside of the robot swarm. Thus, any sensor used in the localisation system must be placed on a moving robot.

The behaviour of zebros should be determined by the swarming algorithm, which is external to the localisation system in the case of the zebro swarm. That means that any localisation method can assume motion, but cannot steer zebros in a certain direction as is done for example by Kurazume [34].

Other methods that are considered depending on external infrastructure are those methods that require pre-existing maps of the environment in some way, such as landmark based methods, or the method from Lockspeiser who uses a map of the magnetic field of the area [38]. All these solutions require either some sensing of the environment beforehand, or control over the environment in some way.

### 3-1-2 Design for scalability

The main driving factor for the localisation system will be scalability. In his book *Scalable Internet Architectures*, Tom Hoff defines scalability as "How well a solution to some problem will work when the size of the problem increases".

Ignoring the localisation methods that use some form of external infrastructure, the solutions from Section 2-1 look at the problem in two ways.
The first way is to localise all robots in the swarm at once, for example by constantly estimating the current topology of the network. Examples of this method are described in Section 2-1-1. These methods make sure that every robot has the same knowledge about the state of the network. The

challenge is to do this efficiently, both in terms of communication and in terms of memory.
The other way is by locating each neighbour separately, following an approach such as that of Guo [30] or Trawny [53].

From the perpective of scalability, the second appoach clearly is superior. As given by the requirements, only the seven closest neighbours need to be located, and thus the complexity of this approach does not increase when the size of the swarm is increased. This is in contrast with the first approach, in which the update phase would require increasingly more memory and time.

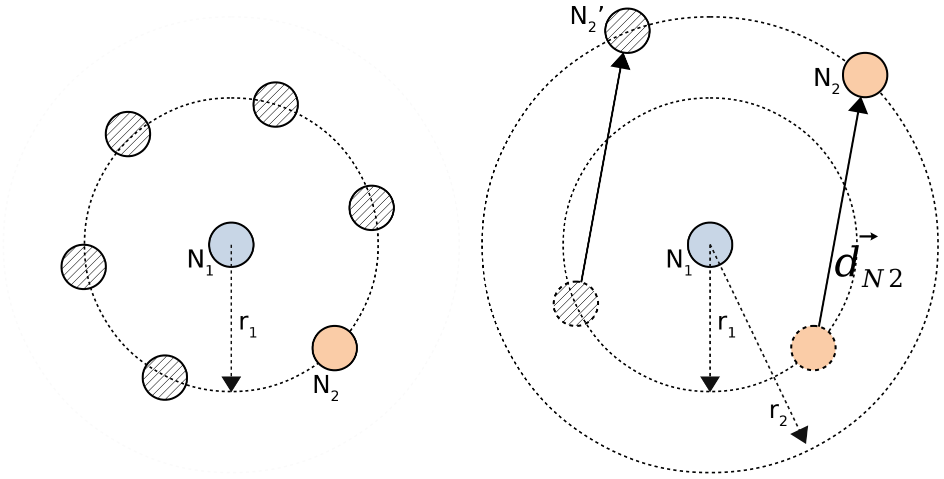
### 3-1-3 Tangolation

The previous sections have given a general outline of what an appropriate solution will look like: all sensors in the system are placed on robots, and two bots can localise each other without communication with any third robot.

The algorithms in Section 2-1 that locate a neighbour by interacting just with that neighbour, all base their localisation on range measurmeents. As noted in the preliminary research in [40], the ultra-wideband technology allows accurate ranging and provides additional communication possibilities.

With just range information, nothing is known yet about angles between nodes. Basic triangulation between nodes can give an insight in the local swarm topology. However, without a common reference the topology can be rotated in any way, still giving no indication about the relative location of neighbours. Additionally, any two neighbours need to share a common neighbour to form a triangle, and in a mobile network the sides would need to be measured at approximately the same point in time. The first problem can theoretically be solved. Algorithms to determine a common reference frame exist [53], but tend to be very computationally intensive. The second assumption, that any two neighbours are part of some triangle of which all sides are measured at the same time, directly contradicts with the restriction that localisation does not control swarming behaviour. There is no guarantee that two neighbours share a common neighbour at any time.

To get rid of the invalid assumption, and the problematic synchronous distance measurements, we propose a localisation method where one of the sides of the triangle is replaced by odometry data. This solution makes sure that one of the triangle sides is a vector instead of a scalar, leaving only two possible triangles in some reference frame. Figure 3-1 gives an impression of the simplified situation where node $N_1$ does not move. In the general solution, $N_1$ will move by $\vec{d}_{N1}$ and $N_2$ will move by $\vec{d}_{N2}$ between two rangings, so the movement of $N_2$ relative to $N_1$ can be written as $\vec{d}_{N2} - \vec{d}_{N1}$. A history of prior location estimates can be used to uniquely determine the



**(a)** After the first distance measurement, $N_2$ may be anywhere $r_1$ away (hatched).

**(b)** After the second ranging and exchanging displacements, $N_2$ can only be in two possible locations.

**Figure 3-1:** Tangolation

correct location, provided that the nodes do not consistently move exactly parallel with the exact same speed.

Secondly, instead of measuring the sides of the triangle at the same point in time, the non-odometry sides can be measured multiple seconds apart. Thirdly, this proposal removes the dependency on a common neighbour, allowing neighbour-to-neighbour relative localisation without any external help. It takes just two to tango, thus, *Tangolation.*

While this method removes the dependency on common neighbours and synchronisation, it replaces it by a dependency on movement. Since the sole purpose of a swarm is to move around, this assumption is deemed acceptable and valid.

The design suggestions from Miog [40] will be followed, and the localisation module will be equipped with an ultra-wideband transmitter that provides ranging and communication. Generating odometry data is regarded out-of-scope of this project, so a velocity estimate will need to be provided by the master device that is using the localisation module. The global reference frame can be provided by a magnetometer, or by algorithms such as the method by Trawny [53]. Keeping the method of generating a global reference frame internal to the localisation module makes sure that as far as the master device is concerned, no global reference plane is needed whatsoever.

## 3-2 Communication

As explained in Section 3-1-3, localisation is driven by wireless TWR. Interference in the wireless channel could cause a degradation of localisation performance, so it is important that there is adequate medium access control.

A naive ALOHA approach as recommended by Decawave [21] is expected to deliver more than 97% messages successfully, provided that the necessary channel utilisation is less than 0.186 [1]. For the TWR scheme with four messages that Decawave advises [21], that would mean that $0.97^4 = 88\%$ of the rangings are successful and 12% fail.

An estimation of the channel utilisation a single node requires is given by Equation (3-1).

$$\eta_{comm,robot} = f_{range} \cdot N_{nb\_range} \cdot t_{range} \tag{3-1}$$

In this equation, $f_{range}$ is the frequency with which a node initialises a TWR with each node, $t_{range}$ is the time it takes for a ranging to complete, and $nb_{range}$ is the number of neighbours a node wants to range with. Simulation tests with Tangolation (see Section 5-1-4) show that bots need to range approximately every five seconds with their neighbours, so $f_{range} = 0.2Hz$. The time that one TWR takes is measured to be $t_{range} = 3ms$ (see Section 5-2). The localisation module is required to locate seven neighbours, so $N_{nb\_range} \geq 7$. However, a node does not only need to range with neighbours it is locating. It should additionally check other neighbours to see if they are close enough to need localisation. As a conservative estimate, the number of neighbours a node needs to range with is therefore doubled to 14. Inserting these numbers into Equation (3-1) shows that one node will require approximately 0.84% of the channel. When the maximimum channel utilisation is 0.186 or 18.6%, there can only be 22 nodes in communication range and, as determined above, 12% of the rangings would be unsuccessful. This puts a strict constraint on the allowable swarm density, considering that the communication range may be much higher than the range in which the closest neighbours need to be located.

The area of the swarm can be much larger than the communication ranges. This might be regarded an advantage: nodes sufficiently far apart can transmit simultaneously without interference, which may allow for a larger number of nodes in the swarm than found from Equation (3-1). On the other hand, a node inbetween two simultaneously transmitting nodes may be within communication range of both, thereby introducing a *hidden terminal* problem (see Figure 3-2).

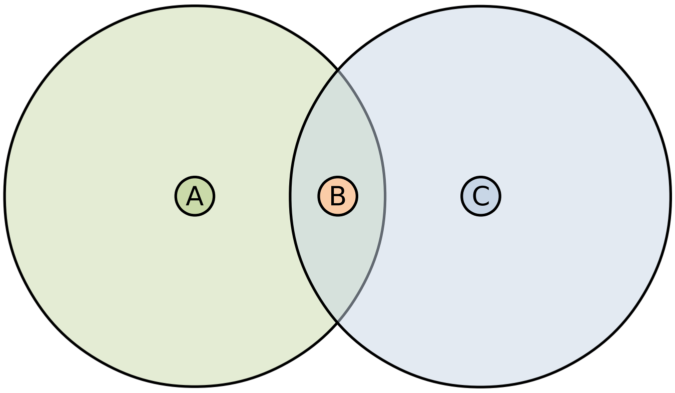


**Figure 3-2:** Hidden terminal problem: Nodes A and C are not within communication range, but simultaneous transmission will still cause interference at node B, which is in communication range of both.

Clearly, there are two main challenges: maximizing channel utilisation, and handling hidden nodes. Design decisions regarding each of these problems are covered in Sections 3-2-1 and 3-2-2, respectively. After that, the final MAC protocol is described in Section 3-2-3.

### 3-2-1 Design for maximum channel utilisation

It has been shown that a naive approach to medium access control will not provide a solution that is scalable to a larger swarm of robots. From Section 2-2 it is clear that some Time Division Multiple Acces (TDMA) scheme provides the best channel utilisation, and a distributed approach exists in the DESYNC method of Degesys [25]. Muhlberger shows a multi-hop extension to DESYNC in [41], where it is shown that a multi-hop network can synchronise itself without guidance of some time master.
Not only is TDMA regarded the best choice with respect to channel utilisation, the DW1000 transceiver will lend itself well for a TDMA based communication scheme. The transceiver has excellent timestamping capabilities that it requires for accurate range measurements [21], and it allows to control the transmission time of a message with a resolution of 8ns and an accuracy in the order of picoseconds [21].

In the DESYNC algorithm, every node transmits with the same period. Every node will always try to time their transmission exactly in the middle of the previous and successive transmitter, so after some periods every node has converged to its new phase.
The main drawback of this method is that the time between two successive nodes may become shorter than the size of a message. A node entering the network and timing its transmission between two nodes would then cause collisions. The period is known and fixed in the DESYNC algorithm, and thus

analogous to a frame in conventional TDMA. The maximum message size will be known in advance too, either because the application is known, or at least because most hardware does not support infinite messages. Therefore we propose to divide the period into slots just like in conventional TDMA schemes without loss of channel utilisation. The advantage is twofold: a new node entering the network has no influence on the phase of the other nodes, so they do not have to find a new phase to transmit at. Additionally, nodes entering the network can do so without causing collisions.

Synchronisation of time slots can be done similarly as in the DESYNC algorithm, simply by listening to neighbours, and adjusting to their tempo. As is seen in the original DESYNC paper [25], oscillations may arise when both nodes adjust their tempo. It should be investigated whether it makes sense to consistently only adjust to the slower or faster node, or to let both nodes adjust to each other.

### 3-2-2 Managing hidden nodes

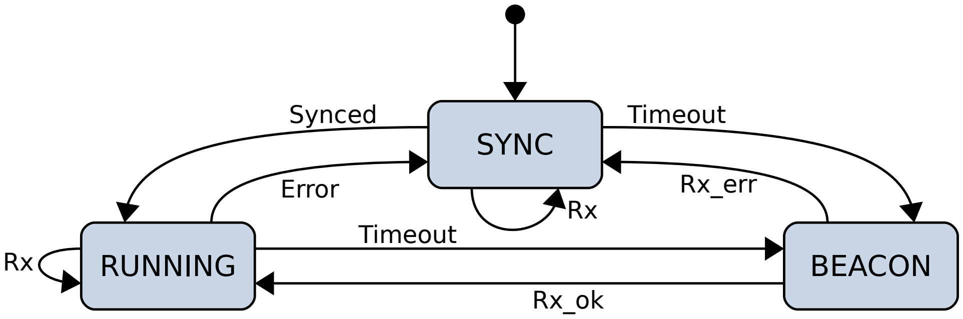
The previous section has proposed a method for distributed synchronisation. When the network is complete and there are fewer nodes than available slots, it is enough for a node to listen for occupied slots and choosing an unoccupied one. In a multi-hop network, this method of slot selection can introduce hidden terminals as in Figure 3-2, where node C may choose the same slot as node A when it senses that that slot is free.

It was proposed by Degesys [24] and shown by Muhlberger [41] that nodes can choose a correct slot when they are aware of the slot of both their neighbours and their neighbours' neighbours, which is every node in a two-hop vicinity. While this method of slot selection is devised for a network of static nodes, it should be researched whether it can be used in a dynamic mobile network.

### 3-2-3 Anarchic TDMA

The MAC layer will consist of two sublayers. The lower layer handles node synchronisation and determines when slots and frames start. The top layer determines which slot a node should take.

The most promising method of synchronisation for TDMA is to listen for when neighbours transmit, and adjust to their transmission times, either both ways, or just one-way. When each message $m$ carries its intended slot number $i$, every message $m_i$ can be used to synchronise to. We call this method Anarchic TDMA (AN-TDMA), referring to its unhierarchical and leaderless operation rather than implying chaos and a complete lack of rules.



| | |
|---|---|
| Rx: | Incoming message |
| Rx_ok: | Incoming message, correctly timed |
| Rx_err: | Incoming message, incorrectly timed |
| Synced: | Enough correctly timed incoming messages |
| Timeout: | No incoming messages for too long |
| Error: | Too many incorrectly timed messages |

**Figure 3-3:** AN-TDMA state machine.

Figure 3-3 shows the state machine each node follows to synchronise against its neighbours. Upon entering the network, a node starts in a *synchronisation* state, listening for neighbours to synchronise to for a random time. If no-one appears to be nearby, the listening timer times out and it moves to a *beaconing* state in which it will start broadcasting $m_0$ at the defined TDMA period. Other nodes entering the network at a later moment in time can then synchronise to these broadcasts. As soon as a message $m_i$ arrives at a *beaconing* node, it checks whether $m_i$ has actually been transmitted within slot $i$. If so (`Rx` in Figure 3-3), the node is part of a synchronised network, and goes into *running* mode. When $m_i$ does not appear to fall inside slot $i$ (`Error`), the transmitting node apparently is part of a differently synchronised network, and the *beaconing* node will switch to *synchronisation* mode to synchronise with its newly discovered neighbour.

The layer above the synchronisation layer will manage time slot selection, and is easily interchangeable once the framework is in place. Since the main topic of this thesis is localisation, and not communication management which only serves as a necessary byproduct, decentralised slot selection will not be covered in this thesis. It has shown to be a hard problem that deserves its own research, and although decentralised methods for stationary networks exist (e.g. [15], [46]), methods for mobile nodes seem more scarce. A properly optimized and tested slot selection algorithm for mobile nodes is therefore deemed out-of-scope for this thesis. In this work, slot selection will be performed naively by assuming a completely connected network.

However, Tangolation and slot synchronisation implementations shall not rely on this assumption. Implementations must be aware of the possibility that not every node may be in communication range, and that nodes can share a timeslot when the nodes are more than two hops apart.

# Chapter 4

# Implementation

For a more in depth understanding of how localisation and communication is implemented, this chapter will give an overview of the architecture and a number of implementation decisions as well as additional hardware choices. It will serve as documentation for those who intend to use, extend or adjust the communication and localisation systems.

The communication and localisation module that is developed consists of a DWM1001-DEV development board from Decawave, which combines a Nordic nRF52 Cortex-M4 microprocessor and a DW1000 ultra-wideband transceiver. The two are connected via SPI, the transceiver being the slave.

The implementation details and challenges regarding localisation are documented in Section 4-1. After that, Section 4-2 will cover the implementation of some features regarding communication.

A number of (apparent) bugs in the DWM1001 package have been found, and are briefly listed in Section 4-3.

## 4-1 Tangolation

Relative localisation of a node's neighbour depends on three inputs: ranging measurements with that neighbour, and both its neighbour's and its own movement since the last ranging with that neighbour. Measurements are noisy, and a naive implementation of Tangolation as described in Chapter 3 does not take that into account. To cope with noisy inputs, a particle filter is used to estimate a location after each ranging update.

The next sections elaborate further on the choice for and implementation of the particle filter, ranging and odometry measurements, respectively.

### 4-1-1 Particle filter

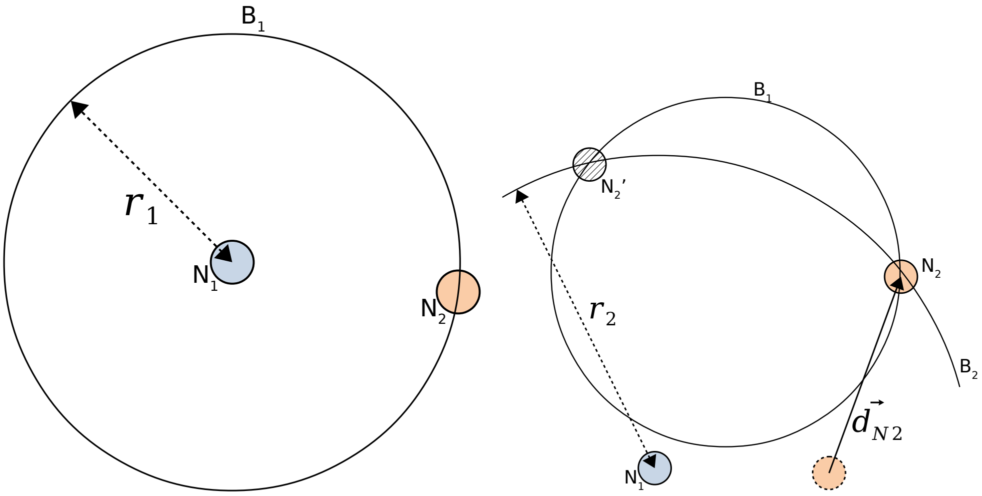
Two methods were found commonly in literature for data fusion in localisation: Kalman filters and particle filters.

Kalman filters are known to be an optimal estimator for linear problems. Real life problems tend to be non-linear, so the Extended Kalman Filter is used to linearise the problem around a current estimate, to still make the linear Kalman filter work for non-linear problems. The output of a Kalman filter is a mean state estimate, and an estimate of uncertainty around that mean.

As explained in Section 3-1-3, with only one triangle measurement (two rangings and the odometry vector) there are still two possible locations for the neighbour to be in, both with their own uncertainty. Kalman filters can only express uncertainty around a certain location, and not the uncertainty that a neighbour can be in one of two completely different locations. A particle filter, however, runs a number of predictions of the neighbours' track simultaneously. Its output consists of a number of possible locations of the neighbour, and an accompanying confidence measure. It is therefore able to express that a neighbour can be in one of multiple locations, and only converge to one of these when the other locations turn out very unlikely. For this reason, nodes will locate their closest neighbours with a separate particle filter for each neighbour.

Running a number of simulations of a neighbours' path tends to be expensive, both in terms of memory and in computation. To allow multiple particle filters (one for each neighbour that is tracked), a number of optimisations have been made in the form of a small number of particles, a precomputed probability density function that is implemented as a fixed-point lookup table. The particle filter algorithm consists solely of addition, subtraction, multiplication, bitshifts and lookups to run the algorithm fast on limited hardware.

Particle filters return a number of possible estimate solutions, but the localisation output should consist of a single location estimate. A possible method of drawing a single, definite conclusion is to take the average of all solutions. However, this will also take into account all outliers. Additionally, when there are two distinct problable locations, the average location will be somewhere where it is clear the neighbour is not located. A way around this could be to detect bounding boxes, and take the average of the box with the highest number of particles, but this gets complicated quickly. We have chosen to keep track of a particles' *age*, which starts at zero when a particle emerges for the first time, and is incremented when the particle survives the resampling phase. The particle with the highest age has been around for longest, and has thus proven to be a somewhat reliable estimation in the past. This way, a definite estimate can be given in $O(n)$, $n$ being



**(a)** Nodes $N_1$ and $N_2$ measures the distance between them.

**(b)** After $N_2$ moved by $\vec{d}_{N2}$ and $N_1$ is aware, it updates its belief.

**Figure 4-1:** Particle filter version of the design in Figure 3-1

the number of particles. A nice side-effect is that the particle age can serve as an estimation of confidence. In the context of the zebros, returning the particle age as an output provides *Toplevel* with a confidence value.

Figure 4-1 illustrates the particle filter implementation of the design of Figure 3-1. In Figure 4-1a, nodes $N_1$ and $N_2$ measure the distance between them. Then in Figure 4-1b, $N_2$ has moved by $\vec{d}_{N2}$. While measuring their range again, the nodes exchange their respective displacements. $N_1$ moves its initial belief by the displacement it receives from $N_2$, and compares that to the new ranging measurement.

### 4-1-2 Ranging

Decawave provides out-of-the-box ranging functionality between static anchors and moving tags with a proprietary and closed-source library for its DW1000 product. This library turned out to be highly tailored to their specific application, and only allows ranging between a device configured as anchor and another one as tag. No ranging was allowed between tags or between anchors themselves. This makes it difficult to use in a homogeneous swarm, as it would mean that bots from one half of the swarm (moving, but configured as anchors) could only range with bots from the other half of the swarm (moving, but configured as tags). Additionally, it allows ranging only and no other communication, and there is no control over the message scheduling. Unfortunately, efforts to tailor this library to our usecase turned out fruitless. Open-source drivers found on the internet only pro-

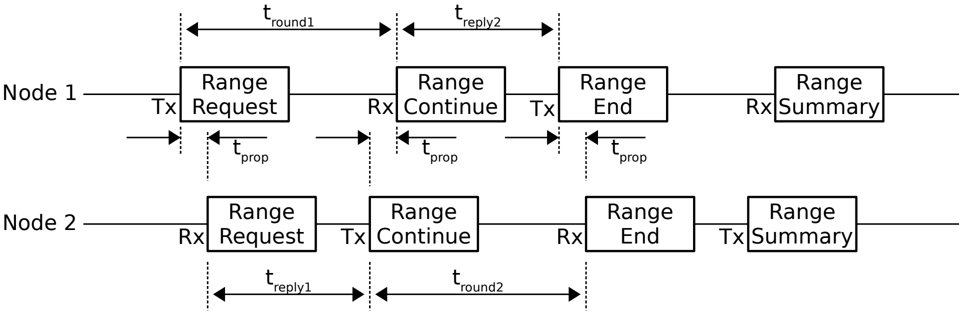


**Figure 4-2:** Two-way ranging scheme, after which both will calculate the same range.

vided blocking communication with the DW1000 transceiver, and did not support its builtin MAC capabilities. For these reasons, a custom driver has been developed. This driver uses the DMA to prevent the processor from blocking on communications, and makes use of the DW1000 MAC filtering to only receive specifically addressed messages.

Ranging is achieved with Two-Way Ranging (TWR). The ranging scheme that is used is shown in Figure 4-2, after which both nodes that participate in the ranging can calculate the distance between the two from the transmission and reception timestamps. See equation 4-1, of which Decawave shows that clock-induced errors are negligible and scaling linearly with the propagation time $\hat{t}_prop$ [21]. For a distance of $100m$, the clock-induced error (for the $20ppm$ clocks that the DW1000 uses) is in the order of millimeters.

$$\hat{t}_{prop} = \frac{t_{round1} \cdot t_{round2} - t_{reply1} \cdot t_{reply2}}{t_{round1} + t_{round2} + t_{reply1} + t_{reply2}} \tag{4-1}$$

All received messages are timestamped automatically with the DW1000's system timestamp, which is done in hardware by the DW1000 with a resolution of 15.65 picoseconds [21]. The *delayed transmission* capabilities of the chip allow the transmission time of a message to be pre-calculated and embedded in the message. The intended transmission timestamp has to be written by the driver to a special register in the DW1000 transceiver, which will then make sure the message is transmitted at the moment the DW1000 system timer matches the programmed timestamp.

### 4-1-3 Odometry

Nodes keep track of their estimated position relative to their own origin, which is to be the place where they first started walking, and is thus different for each bot. When two robots range, their current local coordinates are exchanged. The displacement vector since the previous ranging can then be calculated by simply subtracting the neighbours' coordinate at the previous ranging from the currently received coordinate. When odometry errors accumulate, the exchanged coordinates will have little to no resemblance to a robots actual position relative to its starting point. Since coordinates of a bot are only used to compare them to very recent coordinates of that same bot, this should not pose a problem.

Position tracking is done by integrating the estimated velocity, and a compass provides an approximated direction as well as a global reference frame. Samples of the velocity and compass are taken at $10Hz$. The scalar velocity estimate has to be provided by the master top-level controller as mentioned in Chapter 3. The current generation of zebros does not have the capabilities to measure their own velocity. Instead, the velocity that *Toplevel* (see Chapter 1) intends to go will be used as a crude estimate.

The Bosch BNO055 IMU [8] is responsible for providing a frame of reference that is approximately equal for all robots in the swarm. It combines an accelerometer, gyroscope and magnetometer, and performs automatic sensor fusion. Where most cheap magnetometers need manual calibration, this chip can do the calibration automatically, which is an incredible advantage because no human interaction is needed for magnetometer calibration. The status register provides an estimation for how well the device is calibrated. It can output the attitude in quaternions, so vector rotations can be calculated efficiently by the nRF52 microprocessor.

## 4-2 Communication

The implementation of the communication part of this thesis follows the design as specified in Section 3-2 closely, and thus needs just little further coverage. This section will discuss the intricacies regarding timed transmissions in 4-2-1, and the ranging implementation is covered in Section 4-2-2. Finally, the implementation of the Time Division Multiple Acces (TDMA) slots in 4-2-3 is described.

### 4-2-1 Timed transmissions

As mentioned in Section 4-1-2, the DW1000 can transmit a message at a predefined DW1000 system timer timestamp, which is not only crucial for ranging, but also very useful for TDMA. The DW1000 can be commanded to perform a *delayed transmission*, and transmit the data in its transmit buffer at the timestamp programmed in the `DXTIME` register. In an ideal world, this would lead to the situation marked "Ideal" in Figure 4-3: the microcontroller can issue the commands to fill the transmit buffer at any time

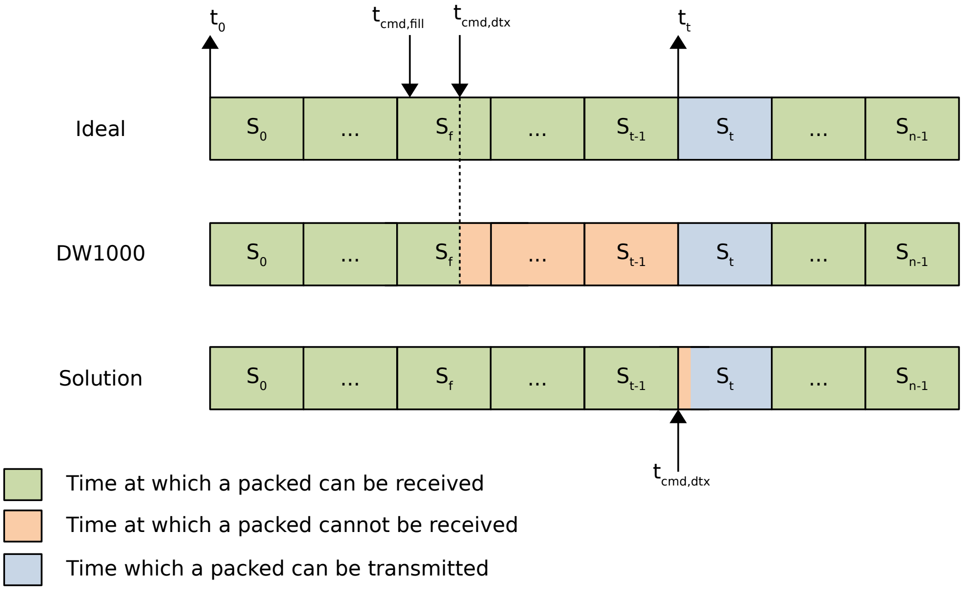

t0
tcmd,fill
tcmd,dtx
tt
Ideal
DW1000
Solution
S0
...
Sf
St-1
St
Sn-1
tcmd,dtx
Time at which a packed can be received
Time at which a packed cannot be received
Time which a packed can be transmitted

**Figure 4-3:** Node $N_t$ attempting to schedule a transmission for slot $S_t$.

$t_{cmd,fill}$ and then at $t_{cmd,dtx}$ tell the DW1000 that the transmit buffer can be transmitted at time $t_t$. The DW1000 would take care of all timings, and the microcontroller only needs to calculate when a message is allowed to be transmitted. The actual situation is indicated with the tag "DW1000" in the figure, and shows that as soon as the command is issued that starts delayed transmission, the DW1000 will turn off its receiver and cannot receive until the message has been transmitted.

The solution to this problem is to let the microcontroller time $t_{cmd,dtx}$ as close as possible to $t_t$, which implies that the microcontroller needs to know about the system time of the DW1000. It is possible to continuously read out the DW1000 system time, to the time until the message is to be sent. Some tests in reading out the DW1000 system time showed that this can take extremely long, sometimes almost half a millisecond. Reasons for why this can take so long have not been found, as we cannot look inside the transceiver. Therefore it was chosen to periodically read out the system time, and synchronize a local timer to the time that was read out. Instead of continuously checking the system time of the DW1000, the driver can now read from and set an interrupt on that local timer to get an estimate of the system time on the DW1000 with a simple linear extrapolation from the two latest DW1000 system time readings. The local time at the moment of starting the reading transfer is paired with the returned DW1000 system time. The advantage of taking the start of the transfer, is that that moment can be reliably determined whereas the end-of-transfer interrupt can be delayed by another interrupt.

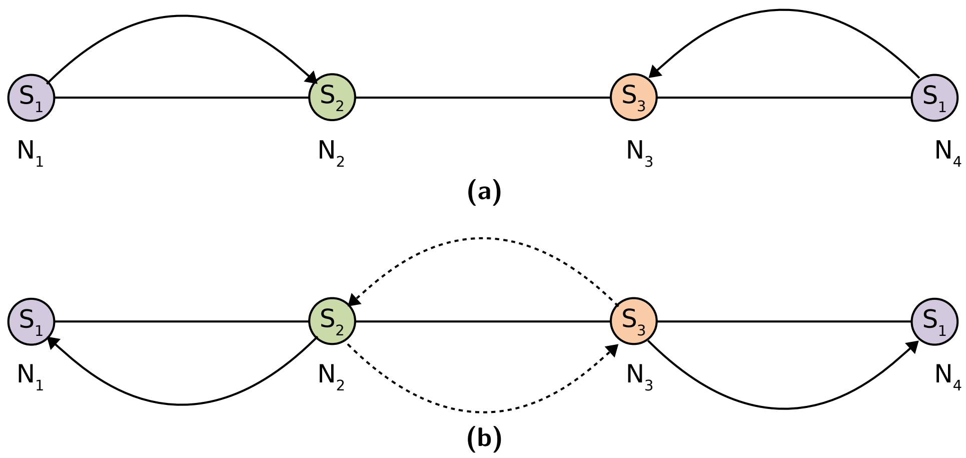


**Figure 4-4:** Neighbouring nodes can transmit in the same slot $S_i$.

The evaluation of this synchronisation is covered in Section 5-2-2.

### 4-2-2 Ranging

A complete TWR message exchange as sketched in Figure 4-2 will be performed in a single Anarchic TDMA (AN-TDMA) slot. Another possibility could have been to start a TWR with a single broadcast and let neighbours respond in their own time. This has the advantage that fewer messages are needed, and the low clock-induced error would make this possible. Additionally, it would automatically address all nodes in the neighbourhood, whereas addressed messages can only reach nodes the transmitter knows about. However, this method would also give rise to a situation where a zebro has a number of ranging requests pending it needs to answer, and it might have moved significantly by the time it has handled the last one. For this reason, TWR is performed in a single slot, to make sure a zebros' ranging comes as close to an atomic operation as practically possible.

The observant reader may notice that this method allows multiple neighbouring nodes to transmit in the same slot. After all, when a node is addressed with a Range Request, it is expected to answer. At first sight, this may render the reasoning about hidden nodes in Section 3-2-2 invalid.

As can be seen in Figure 4-4a, nodes $N_1$ and $N_4$ can both transmit in slot 1. By performing ranging in the same slot, their neighbours $N_2$ and $N_3$ now also both have to transmit in slot $S_1$ (Figure 4-4b). Their transmissions, however, only cause collisions at nodes that are in transmission range of both $N_2$ and $N_3$, and none of these nodes is an intended receiver. Therefore, the reasoning about hidden nodes in Section 3-2-2 still holds.

### 4-2-3 TDMA slot implementation

Generally, a TDMA slot contains a *guard time* in which it is guarenteed that no node transmits, and some time during which a specific node is allowed to transmit. The *guard* time is used to give nodes time to finish a previous message exchange and prepare for the next. Additionally, it accounts for clock differences between nodes and possibly some degree of non-determinism. In this particular case, the *guard* time should give a node sufficient time for

- handling the received message and re-enable the receiver,
- issuing the command to the DW1000 transceiver to start delayed transmission,
- some leeway, to account for the uncertainties in DW1000 system time approximation as introduced in Section 4-2-1.

Because AN-TDMA does not synchronise to a time master but to all neighbours equally, a node should time its transmissions at a known moment in the slot rather than anywhere within its allowed transmission time. For this reason, AN-TDMA slots contain a *valid* time segment in which a node is allowed to start its transmission, see Figure 4-5. The duration of the segment should be sufficiently long to account for clock drift and clock differences between nodes, but short enough to discover messages timed out-of-slot. In case of TWR, only Range Request messages (see Figure 4-2) need to start within the *valid* segment.

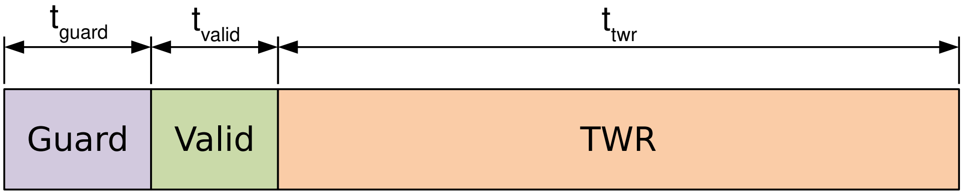


**Figure 4-5:** Slot segments in AN-TDMA.

The timestamp of a message that is transmitted or received by the DW1000 transceiver refers to the time of arrival of the `RMARKER`, which is located at the beginning of the PHY header (PHR) of the physical packet output by the transmitter. The layout of the physical packets that the DW1000 transmits are shown in Figure 4-6 [21].

The pre-amble and start-of-frame delimiter (SFD) are known and constant, so from the `RMARKER` timestamp, the beginning of the packet can be easily calculated. Transmitting nodes will time the start of their transmission in

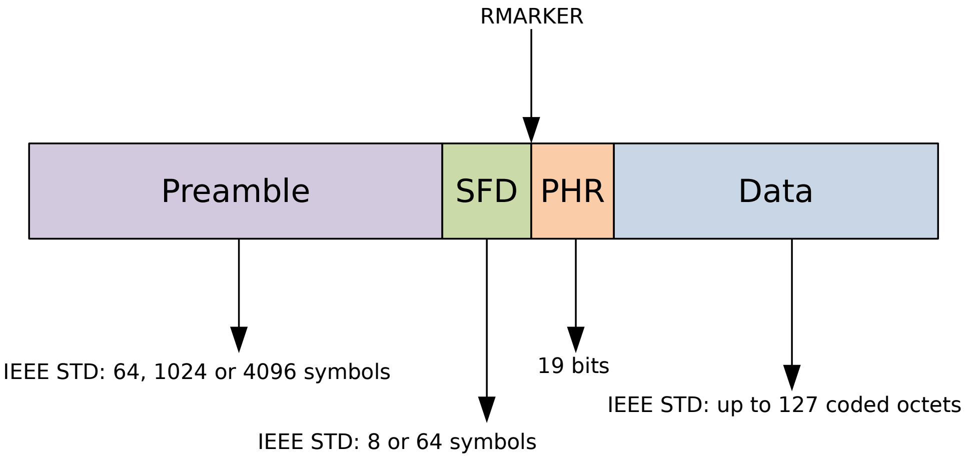

RMARKER
Preamble
SFD
PHR
Data
IEEE STD: 64, 1024 or 4096 symbols
19 bits
IEEE STD: up to 127 coded octets
IEEE STD: 8 or 64 symbols

**Figure 4-6:** DW1000 physical packet.

what they believe is the middle of the *valid* segment. Receiving nodes will adjust their own belief about the beginning of the slot whenever they receive a message in their *valid* segment. Messages timed outside a nodes' *valid* segment will trigger an error. When a node receives more erroneously timed messages than correctly timed messages, chances are high that its own belief about slot timings is wrong, and that its own transmissions are wrongly timed. As soon as a node notices this, it switches to its synchronisation mode, as depicted in the Design chapter in Figure 3-3.

## 4-3 Bugs

The DWM1001-DEV board with the DW1000 ultra-wideband tranceiver and the nRF52 microprocessor contains a number of bugs and annoyances, both in board design (marked `BOARD`) and in the transceiver (marked `DW1000`). To save a possible successor from having to debug the same errors again, the bugs are described concisely here. A workaround (`WORKAROUND`) was found for some of these, but not for all (`UNSOLVED`).

### DW1000 receive re-enable [`DW1000, UNSOLVED`]

Decawave has documented that an error in the receiver re-enabling can cause receive timestamps to be incorrect. Before re-enabling, certain error bits in the status register need to be cleared and the receiver needs to be reset to assure correct operation, according to the datasheet. However, the list of error bits that cause this transceiver bug ends in 'etc', so it is unknown if the current driver will let the DW1000 always behave correctly in all situations.

**DW1000 automatic receive re-enable** [DW1000, WORKAROUND]
The DW1000 chip has the ability to automatically re-enable its receiver after receive errors, or, when double receive buffering is enabled, after a successful reception. This setting has shown to generate regular spurious interrupt signals, that seem to be pulled low by the DW1000 immediately after firing. The pulse is still wide enough to be noticed by the microprocessor. In this project the setting is disabled, and the receiver is only re-enabled manually.

**DWM1001 SPI SlaveSelect leaks into VDD** [BOARD, WORKAROUND]
The communication and localisation module will be connected to a master via Zebrobus over SPI. The SlaveSelect of SPI is active-low, and thus generally high. It has been discovered that when the DWM1001-DEV board is unpowered but connected via SPI as a slave device, it will draw power from the SPI SlaveSelect pin, and pull VDD to 1.8 volts.

The BNO055 IMU is connected to the same VDD, and the $1.8V$ is enough for the IMU to stay powered, but not enough for the nRF52 microcontroller. When power is removed from the microcontroller, the continous communication with the BNO055 may be interrupted mid-communication, leaving the BNO055 in an invalid state in which no communication with the chip is possible. It cannot recover from this state, because it cannot be turned off due to the power leakage from the SPI SlaveSelect pin keeping it on.

As a workaround, the Zebrobus master can issue a command to start or stop communication between the microprocessor and the IMU, so the microprocessor knows to stop reading the IMU. After issuing this command, this way the microprocessor can be safely depowered while keeping the SPI connected.
The manual workaround at this moment is to always disconnect both the SPI and the power, but in the future a different PCB must be designed that shields the power circuit properly.

# Chapter 5

# Evaluation

The performance of both localisation and communication is evaluated in this chapter, in Sections 5-1 and 5-2, respectively.

## 5-1 Localisation

Real-world robot localisation tests with zebros require time constly and extensive human intervention, and are difficult to automate because of the absence of a ground truth to compare the localisation results with. Additionally, the current batch of zebros exhibit some bugs in their locomotion, causing them to stumble frequently. With the dependency of Tangolation on odometry estimates, real-world tests would not be representable. For this reason, Tangolation will be evaluated with results from a simulation which will be described in Section 5-1-4. However, first some concepts will be intruded in Section 5-1-1. Then the two inputs to Tangolation, ranges and a zebro's own displacement, are evaluated using real-world tests in Sections 5-1-2 and 5-1-3, respectively. Their error behaviours are analysed and fed into the Tangolation simulation, so the simulation can provide Tangolation with realistic noisy input. Finaly, section 5-1-4 then covers the results from the simulation.

### 5-1-1 Concepts and notations

Some shorthand notations and concepts need introducing for clarity.

**Range** $r$:
Subscript $r$ is used to refer to ranging measurements by the DW1000 transceiver.

For example, $\mu_r$ is the mean of a set of ranging measurements. Ranges are an input to the Tangolation algorithm.

**Displacement distance** $s$**:**
Subscript $s$ is used to refer to the straight line distance from some point A to some point B, both points on a same path a zebro walked. This straight line distance will from now on be called the *displacement distance* of a path. Displacement distances are an input to the Tangolation algorithm.

**Displacement angle** $\alpha$**:**
Subscript $\alpha$ is used to refer to the netto angle of a path walked by a zebro. We will mainly look at angle errors, so it does not matter whether the angle is with respect to a local or global reference frame. Displacement angles are an input to Tangolation.

**Localisation distance** $d$**:**
Subscript $d$ is used to refer to the scalar distance a zebro believes its neighbour is located at. This *localisation distance* is an output of the Tangolation algorithm, and is expected to be close the input $r$.

**Localisation angle** $\theta$**:**
Subscript $\theta$ is used to refer to the angle at which a zebro believes its neighbour is located at. In this thesis, zebros measure this *localisation angle* with respect to magnetic North.

**Error** $e$**:**
Error subscript $e$ is used to denote the difference between some set of measurements $x_{meas}$ and their actual values $x_{act}$. For example, $\mu_{r,e}$ is the mean of the set $r_e$.

**Swarming range** $r_s$**:**
Robots in the Tangolation simulation try to stay within some range from each other. For example $r_s = (r_{s,min} = 7, r_{s,max} = 15)$, a robot will move away from a neighbour when it notices it is closer than $r_{s,min} = 7m$, and move to a neighbour when it is further than $r_{s,max} = 15m$.

## 5-1-2 Ranging error modelling

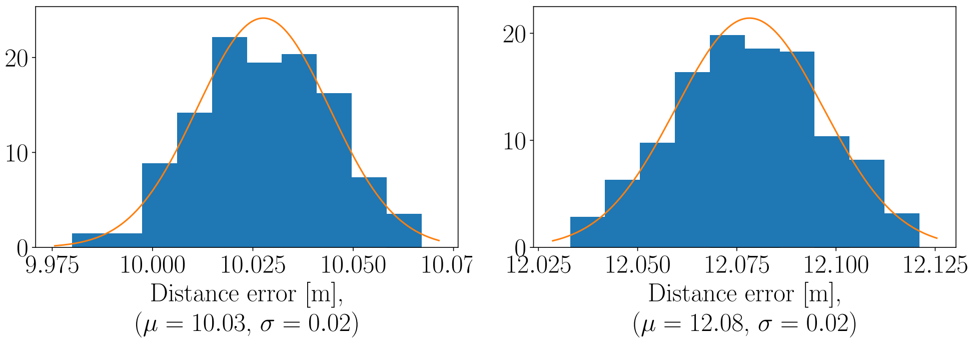
As a first test, ranging measurements are done to evaluate how a noisy ranging measurement can be modeled in the Tangolation simulation.

### Experimental setup

The DWM1001-DEV development board from Decawave is used, which combines a nRF52832 ARM Cortex-M processor, a DW1000 transceiver for ultra-wideband communication, and a 6.5GHz UWB channel 5 antenna.

**Table 5-1:** Transceiver settings for ranging measurements.

| | **Short range** | **Long range** |
|---|---|---|
| Bitrate [kbps] | 6800 | 128 |
| PRF [Hz] | 64 | 64 |
| Preamble length [bits] | 128 | 2048 |



**(a)** Test at 10.05m.

**(b)** Test at 12.06m.

**Figure 5-1:** Ranging measurement distribution.

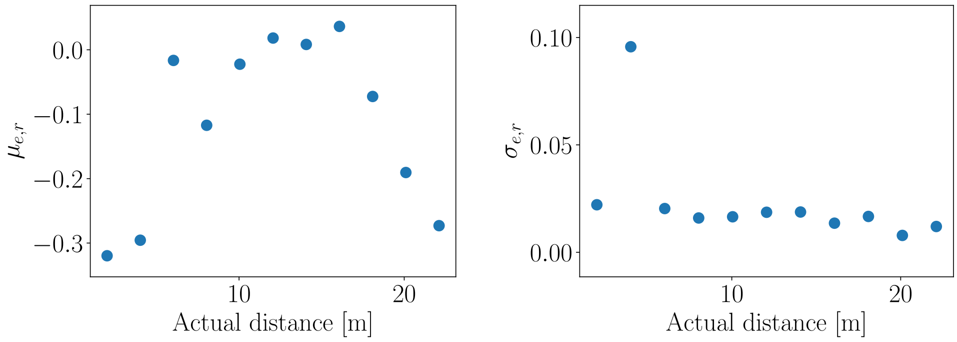
The software driver for the DW1000 transceiver that was implemented in C for this project manages the DW1000 and calculates the ranges.

Measurements are performed indoors at approximately $10cm$ from the floor, in the long hall at the 13th floor of the TU Delft EEMCS building. The floor consists of tiles of exactly $2.01m$, which provides a consistent, though slightly awkwardly spaced grid. All ranging tests are line-of-sight measurements, but in a noisy indoor environment.

Tests are done both with short-range and long-raneg transceiver settings. With the short-range settings, transmissions are kept short and the bitrate is high. For long range transmissions, a lower bitrate is used and the preamble is long to aid in packet discovery. The transceiver settings can be found in Table 5-1. To prevent redundancy, only short-range results will be shown in this section, as the long-range results show identical behaviour.

### Distance influence on ranging performance

Test results from some ranging measurements are shown in Figure 5-1. It can be seen that the error in the measurements in Figures 5-1a and 5-1b follows a Gaussian distribution, and thus can be modelled as such in the localisation simulation.



**(a)** Mean error vs. measurement distance.

**(b)** Standard deviation vs. measurement distance.

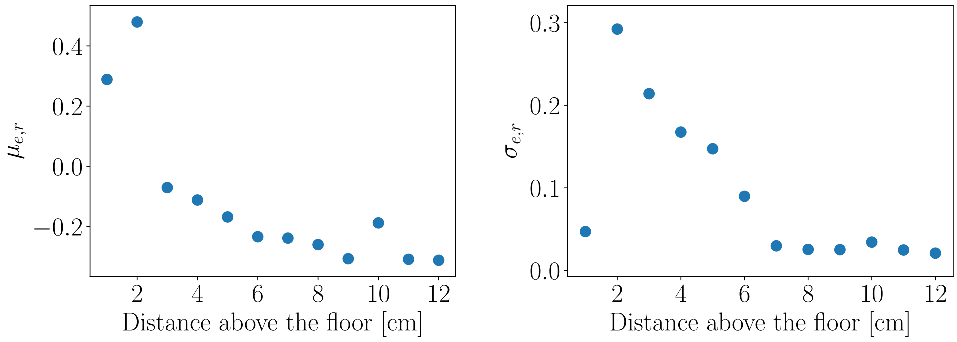
**Figure 5-2:** Ranging measurement distribution.

The means of the ranging errors $\mu_{r,e}$ and standard deviations $\sigma_{r,e}$ of all tests are graphed in Figure 5-2. It seems that the bias is dependent on the range, and the small standard deviations (Figure 5-2b) imply that measurements are very constant. The cause of the increasingly negative bias for distances $> 10m$ is documented by Decawave in one of their application notes [18], and is caused by the lower receive signal level. Unpublished work by Escudero [51] shows a simlarly increasingly negative bias when the ranging distance increases, of the same order of magnitude as in Figure 5-2a from $10m$ onwards.
However, neither of these sources explain a cause of the negative bias for distances $< 8m$. A possible cause could be errors in testing, causing the ground truth to be incorrect. However, tests were done in multiple runs, and it seems unlikely that the exact same measurement error was made multiple times.

The standard deviation of the ranging error (Figure 5-2b) at $\sigma_{r,e} \approx 2cm$ is shown to be small, constant, and independent of range. Additionally, it shows that the claim by the datasheet, which promises precision within $10cm$ [19], as approximately 99.7% of measurements will lie within $3\sigma_{r,e} \approx 6cm$ from the mean.

### Influence of proximity effects

The one outlier in terms of standard deviation is the measurement at $4.02m$. Reviewing the test showed that the antenna of one of the nodes pointed slightly downward and came close to the floor, thereby showing the necessity of some clearance around the antennas. Decawave makes a mention of proximity effects in its Design Decisions Application Note [17], and mentions



**(a)** Mean error vs. antenna distance from floor.

**(b)** Standard deviation vs. antenna distance from floor.

**Figure 5-3:** Rangings at $6m$ apart, increasing the distance from the floor.

a required clearance of $10mm$ in the DWM10001 board manual [23].

Figure 5-3 shows that when a receiver is placed close to some surface, ranging performance degrades quickly. Not only are measurements more noisy as indicated by the increase in $\sigma_{r,e}$, the mean measured range increases too. The leading-edge detection algorithm of the DW1000 transceiver seems to be unable to distinguish the first-path signal from its multipath reflections. The increase in $\mu_{r,e}$ of almost $0.6m$ implies that not one of the reflections of the close-by floor is taken as measurement, because then the difference in mean error would have been lower. Instead, it seems that so much noise is generated that the first distinguishable leading edge comes from a completely different reflection. A quick back-of-the-envelope calculation shows that, when assuming a symmetric reflection half-way along its path, the signal probably reflected from some object $1.37m$ from the actual first path, see the sketch in Figure 5-4. This is not at all surprising, given that the EWI halls are about $2.5m$ wide, and the experiment was performed roughly in the middle of the hall.

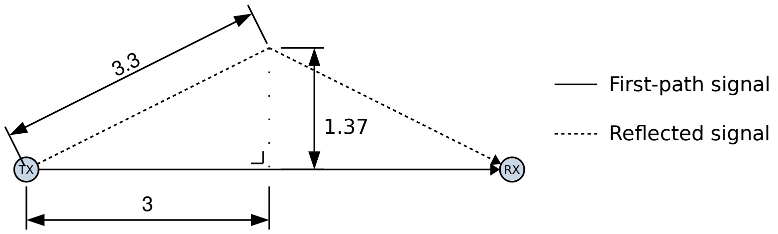


**Figure 5-4:** Ranging signal reflections.

Measurement results in Figure 5-3 show that the clearance needs to be significantly larger than the proposed $10mm$ to provide the ranging precision that the manufacturer claims. Unfortunately, that means that the localisation module cannot be placed inside the zebro body, but needs to be placed on the outside.

### 5-1-3 Displacement error modelling

Zebros measure their displacement by sampling and integrating over a velocity estimate and a direction. Although this is unusable for long-term self-localisation, it can be used to relate a zebro's current location to its location some seconds earlier which is what Tangolation will use the displacments for. Displacement angles and distances of a zebro are measured and compared to the zebro's estimate, to analyse the noise behaviour of the displacement errors.

#### Experimental setup

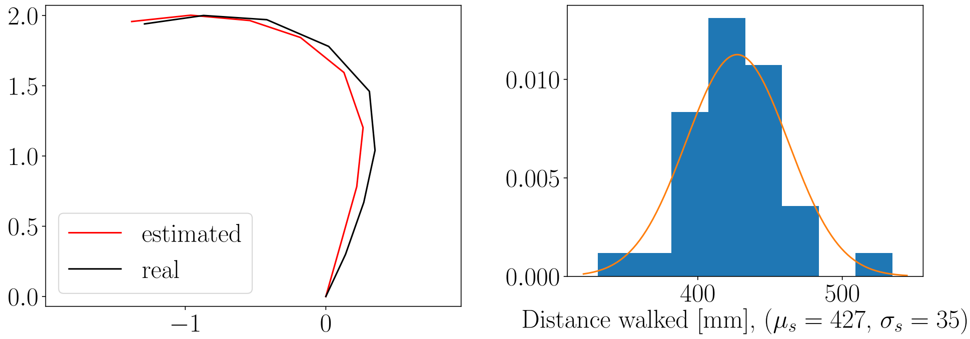
In this expirement, a zebro was released in a grassy field. As described in Section 4-1-3, the zebro estimates its location relative to its starting point by integrating over its velocity and direction, both of which are sampled at $10Hz$. For the experiment, the zebro's self-localisation estimate was sampled every five seconds ($0.2Hz$), and its location in the field at the moment of sampling was marked so the ground truth could be measured afterwards. With the estimated zebro velocity of $0.1m/s$, markers are thus expected to be placed approximately $0.1 \cdot 5 = 0.5m$ apart.

The displacement distance and angle between successive markers was measured, and compared to the estimates by the zebro.

#### Zebro displacement tracking accuracy

Figure 5-5 shows two examples of paths walked by a zebro, and its estimation of that path. The mean displacement distance from marker to marker is $427mm$, as plotted in Figure 5-5b.

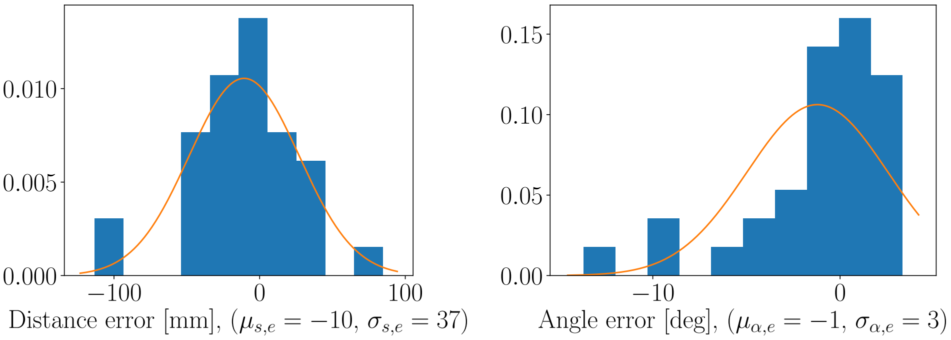
Figure 5-6a shows a histogram of differences between the actual displacement distances and estimated displacement distances between markers. The error $s_e$ behaves somewhat like a Gaussian. Fitting the displacement error with a normal distribution results in $X_{s,e} \sim \mathcal{N}(\mu_{s,e}, \sigma_{s,e}^2)$ , $\mu_{s,e} = 10mm$, $\sigma_{s,e} = 37mm$.
This Gaussian does not model the outliers at $-100mm$ and $75mm$ very well. These outliers are caused by a bug in the zebro's walking algorithm, that make it stumble from time to time.



**(a)** A path walked by the zebro.

**(b)** Measured distances between markers.

**Figure 5-5:** Zebro walking path and distance distribution, sampling at $0.2Hz$.



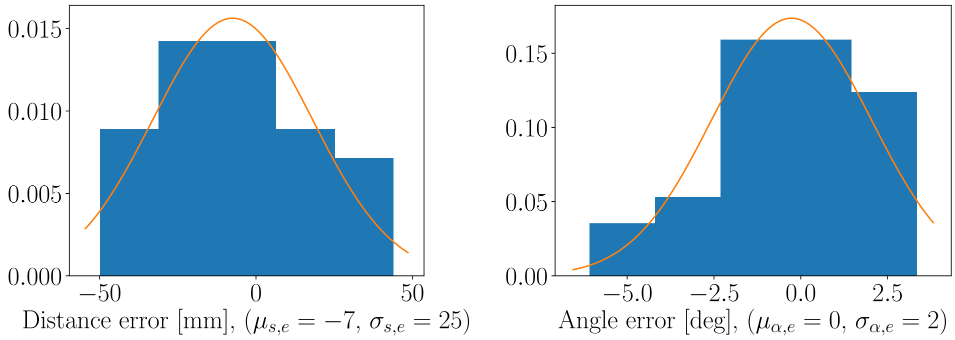
**(a)** Error in displacement distance tracking.

**(b)** Error in displacement angle tracking.

**Figure 5-6:** Errors in displacement tracking between markers, sampling at $0.2Hz$.

The current method of displacement tracking, which just uses the intended velocity due to the current absence of velocity tracking, does not take these stumbles into account as they are not being detected. When in the future bugs like these are fixed, and the velocity can be measured, displacement tracking will likely improve. For now, however, this is what we have to make do with.
The error in angle $\alpha_e$ as plotted in Figure 5-6b shows two similar outliers, for the same reasons. The fitting normal distribution gives $X_{\alpha,e} \sim \mathcal{N}(\mu_{\alpha,e}, \sigma^2_{\alpha,e})$ , $\mu_{\alpha,e} = -1°$, $\sigma_{\alpha,e} = 3°$.

Artifically removing the outliers caused by zebro stumbling results in a much smaller standard deviation, and thus a better prediction of the lenth of the path that was walked, as shown in the histograms in Figure 5-7. With a



**(a)** Error in displacement distance tracking, without zebro stumbles.

**(b)** Error in angle tracking, without zebro stumbles.

**Figure 5-7:** Errors in displacement tracking between markers, sampling at $0.2Hz$, outliers due to stumbling removed.

mean displacement $\mu_s = 427mm$ (see Figure 5-5b), the displacement error standard deviation goes down from $\sigma_{s,e} = 37mm$ (8.6% of the average walked displacement distance) to $\sigma_{s,e} = 25mm$, which is 5.8% of the average walked displacement distance. The mean of the errors $\mu_{s,e}$ is less than $10mm$, and is thus very close to zero: evidently, the velocity estimate provided by the *Toplevel* module is somewhat accurate in the long run.

The standard deviation of the angle error is very small at 3°, even before filtering out the outliers (Figure 5-6b). Over a displacement distance of $\mu_s = 427mm$, a 3° error comes down to approximately $15mm$, which seems well within the error of manual marker placement, and thus the actual error may be smaller.

The crude method of displacement estation and measuring, combined with the current bugs in zebro locomotion that causes stumbling makes for an error analysis of what may be regarded the absolute worst-case scenario.

### 5-1-4 Tangolation

The error behaviour of the inputs to the localisation algorithm Tangolation has been researched, so a simulation of Tangolation can provide realistic insight in the expected localisation accuracy and precision. The localisation performance will be split in *localisaton distance* performance, and *localisation angle* performance. The localisation distance refers to the output estimated distance to a neighbour, and localisation angle regards the estimated angle at which that neighbour is positioned. In this thesis, the angle is measured with respect to a global axis system, but nothing prevents the use of a local axis system.

### Experimental setup

Section 5-1-2 showed that ranging measurements have a consistent standard deviation of $\sigma_r = 0.02m$ independent of ranging distance. Displacement tests in Section 5-1-3 showed that distance tracking is not very precise, and the error has a standard deviation of 8.6% of the displacement distance in what may well be a worst-case scenario. Angular errors on the other hand are small, and show a standard deviation of $\sigma_{\alpha,e} = 3°$.

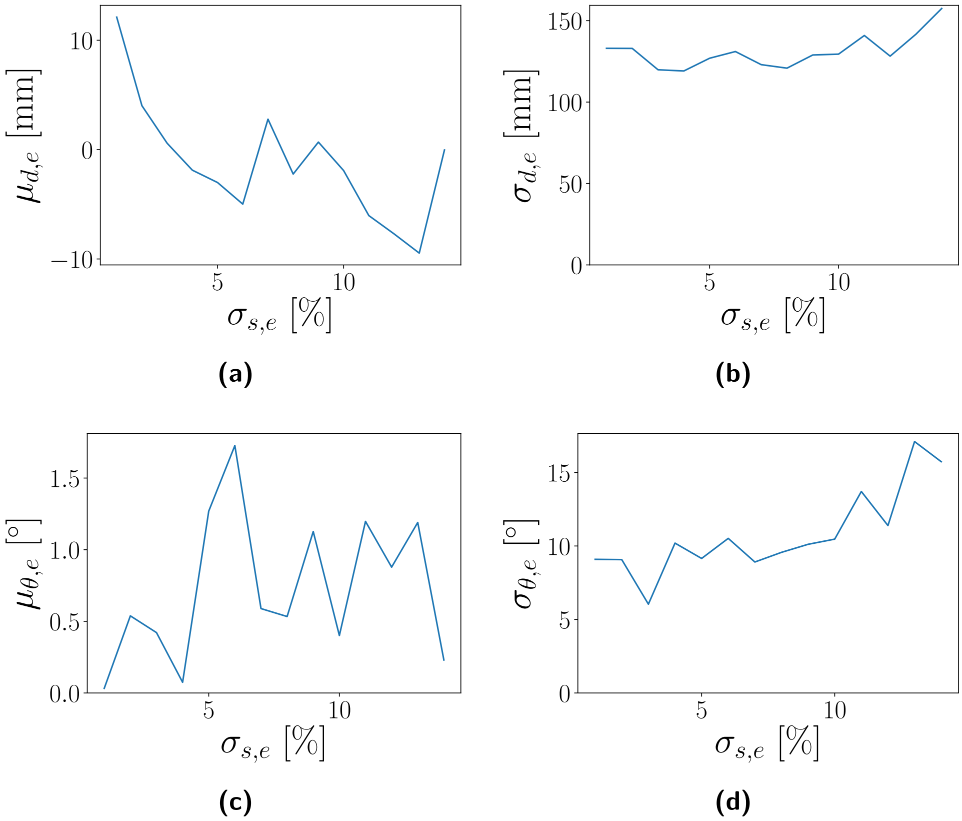
The error behaviours as described above were fed into a Python simulation, in which two robots move together through an infinite, empty world. The robots follow the flocking rules from Reynolds' computer program *boids* [45], where robots try to move in the direction they think their neighbours are going, try to remain close to them and at the same time try not to collide. Time is discretized in steps of $0.1s$, and in every step there is a chance of 0.5% for a robot to make a random corner of 45° to prevent it from going in the same direction eternally. Each robot is given the noisy distance to its neighbour and its neighbour's noisy displacement every $4 - 6s$. Combined with its own noisy displacement estimate, the robot runs the Tangolation algorithm to estimate the location of its neighbour. The displacement noise is implemented as a percentage of the actual displacement, because displacement is estimated by integration over a constant velocity estimate.

At any point in time, the neighbour's current location is extrapolated from its latest estimated location, its latest reported displacement vector, and the time since that last location estimate. This way, a robot in the simulation will always have some estimate of its neighbour's current location that it can use for swarming behaviour. Each simulation was run 50 times for 2000 timesteps with two robots.

### Varying the displacement distance error

As a first test, Tangolation robustness against noise in the displacement estimate is analysed, as displacment distance estimates are currently based on an intended velocity, and are not being measured. Insight in the behaviour of Tangolation with respect to noisy displacement estimates could for example be used as an argument for whether or not better displacement tracking in a next generation of zebros is required.

The main value of Tangolation lies in its ability to estimate the angle $\theta$ at which a neighbour can be found, after all the distance to the neighbour is easily measured and also already an input to the algorithm. Figure 5-8d shows that the uncertainty about the angle grows quickly when the displacement distance uncertainty increases. Considering that $\sigma_{s,e} = 8.6\%$ is a conservative estimate based on stumbling zebros, $\sigma_{\theta,e}$ can be expected to be around 10°, meaning that approximately 65% of the angle estimates are

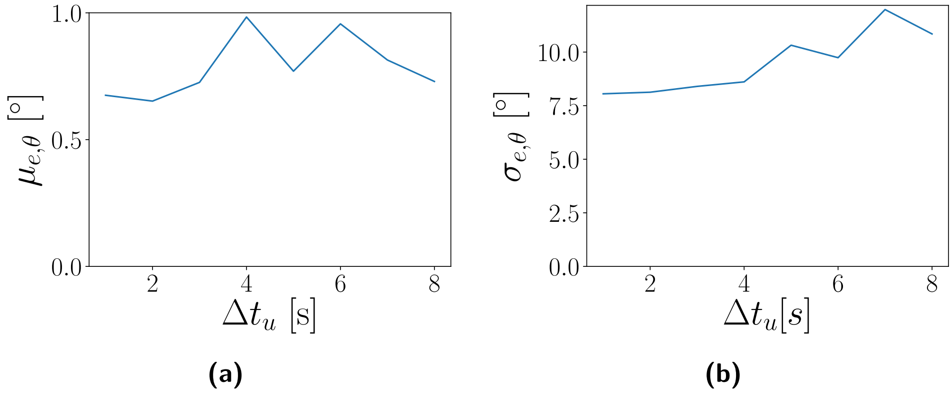


**(a)** **(b)** **(c)** **(d)**

**Figure 5-8:** Errors in $d$ and $\theta$ when changing $\sigma_{s,e}$ as $\%$ of displacement distance. $r_s = (7m, 15m)$, $\sigma_{r,e} = 0.02m$ and $\sigma_{\alpha,e} = 3°$.

within 10° of the true value, and more than 95% of the angle estimates are within the required maximum error of 22° with input errors that are modeled after the current suboptimal situation with stumbling zebros.

As expected, the mean errors $\mu_{d,e}$ and $\mu_{\theta,e}$ seem not so much influenced by the increase in displacement distance noise, because no bias has been introduced. The standard deviation of the localisation distance error hovers around $\sigma_{d,e} \approx 130mm$, so the output range estimate is considerably more noisy than the input range measurements. Evidently, fusing the input range data and the input displacement data does more bad than good for the distance estimation. This is not unexpected, given that the input distance estimates (the range data) were very accurate and precise to begin with, and the displacement data not so much. In this particular case, the *Toplevel* module might choose to only use Tangolation for an angular estimate, and combine that with the raw ranging measurements.



**(a)** **(b)**

**Figure 5-9:** Varying the update time $\Delta t_u$ with $r_s = (7, 15)$, $\sigma_{r,e} = 0.02m$, $\sigma_{s,e} = 8.6\%$, $\sigma_{\alpha,e} = 3°$.

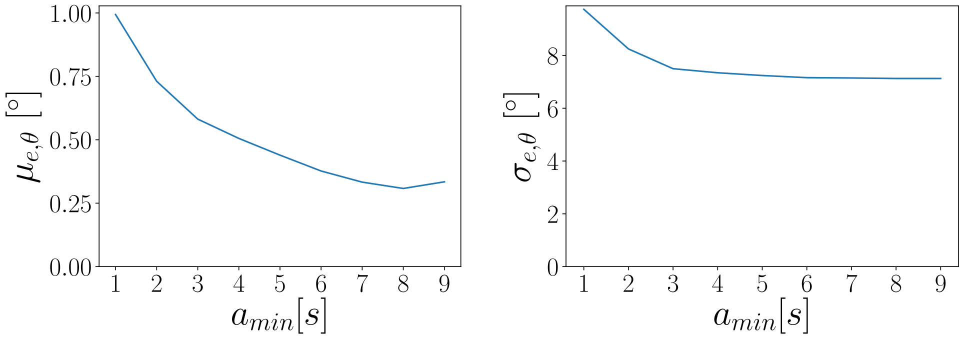
### Varying the time between updates

When the network becomes more densely populated, robots will have less possibilities to perform rangings with their neighbours. A larger time between updates $\Delta t_u$ will be the consequence, and this decrease in input data per lenght of time may affect the performance of localisation negatively. On the other hand, when $\Delta t_u$ is small, a zebro may not have had enough time to significantly move, and movement is crucial for Tangolation.
What we see in Figure 5-9 is that with update rates in the order of a second, Tangolation can provide fairly precise position estimates with a $\sigma_{\theta,e} \approx 8°$, keeping almost 99% of the angle estimates within 22°. Even when the update time is increased to $8s$, the standard deviation of the angle stays within 12°, and the averages of the angle error $\mu_{\theta,e}$ prove to be practically unaffected by an increased $\Delta t$. However, there is an argument to be made against long times between updates: a neighbour's location inbetween updates can only be estimated by extrapolation. Longer times between updates mean longer times a possible corner from a neighbour can go unnoticed.

### Varying the minimum particle filter decision age

As mentioned in Section 4-1-1, an *age* value has been assigned to each particle in the particle filter. For a user of Tangolation, such as *Toplevel* of Figure 1-2, this age can serve as a measure of confidence about the accuracy and precision of the estimated location. Normally, Tangolation uses the location estimate with the highest age as best estimate. In this test, a best estimate is only logged when its age $a$ is equal or highr than some $a_{min}$.

As is shown in Figure 5-10, the expected error in the localisation angle reduces significantly with increasing age. When the reported particle age is more than 5, $\sigma_{\theta,e}$ goes below 7°. That means that the percentage of angle



**(a)** **(b)**

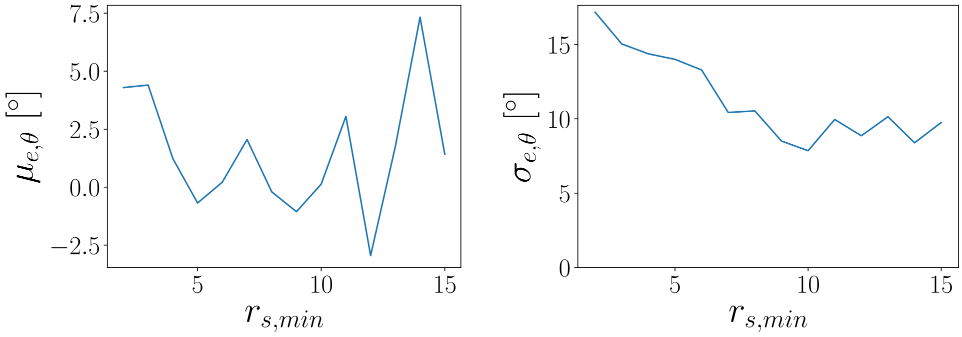
**Figure 5-10:** Changing the minimum particle age. $r_s = (7, 15)$, $\sigma_{r,e} = 0.02$, $\sigma_{\theta,e} = 3°$, and $\sigma_{s,e} = 8.6\%$.

estimations that are within 10° of the true value go up frmo 65% to more than 84%. Evidently, maximum age particle is a valid decisor, and its age can indeed serve as an indication of the quality of the localisation esimation. The figure shows no increase in quality of the estimation when the particle age exceeds 5, so the confidence value to return to *Toplevel* will lie in the range from 1 to 5.

### Varying the swarming range

To test how well Tangolation works when robots are programmed to stay close or far, the swarming range $r_s$ was varied, while keeping the swarming range width $r_{s,w} = r_{s,max} - r_{s,min}$ constant and equal to 4. This value was chosen large enough so robots do not constantly have to adjust to each other, as they would not have to do in a real swarm. On the other hand, the range is small enough to give a number of disjoint swarming ranges.

Figure 5-11 shows again that the mean of the localisation angle errors is unaffected by input changes, but $\sigma_{\theta,e}$ proves to be highly influenced by the distance to a neighbour. Intuitively, this could have been expected. As drawn in Figure 5-12, a localisation error $e$ close by results in a larger localisation angle error $\theta_e$ that the same error at a further distance. Thus, the current uncertainty about the exact movement of the zebros has a significantly higher influence on the localisation performance when zebros are closer to each other.



(a)

(b)

**Figure 5-11:** Changing the swarming range. $\sigma_{r,e} = 0.02$, $\sigma_{\theta,e} = 3°$, and $\sigma_{s,e} = 8.6\%$.

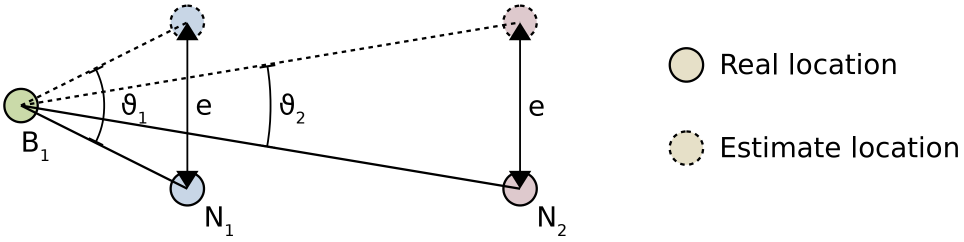


**Figure 5-12:** An absolute localisation error $e$ close-by results in a larger angular error than localistion error $e$ far away.

## 5-2 Communication

This section will test timings and types of synchronisation related to the Anarchic TDMA (AN-TDMA) implementation. Section 5-2-1 introduces some concepts and notations. Then the prediction of the DW1000 system timer with a timer local to the nRF52832 microcontroller is analysed in Section 5-2-2. The methods of node synchronisation as mentioned in Section 3-2-1 are compared in Section 5-2-3. Message timings are covered in Section 5-2-4, to verify whether AN-TDMA can provide a higher channel utilisation than ALOHA.

### 5-2-1 Concepts and notations

To keep explanations concise, some notations and concepts need to be introduced.

**Local timer** :
A 16MHz hardware timer local to the nRF52832 microprocessor.

**DW1000 system timer** :
The 125 MHz system timer of the DW1000 transceiver. The tranceiver can be programmed to transmit a message at a specific timestamp.

**Local timer synchronisation**:
The concept of using one of the hardware timers of the nRF52832 microprocessor to predict the system time of the DW1000, as previously explained in Section 4-2-1.

**Node**:
In this section, a node refers to a DWM1001-DEV board with an nRF52832 microprocessor and a DW1000 transceiver that is used for wireless communications.

**Node synchronisation**:
The notion of synchronising nodes so they have approximately the same belief about the start times of AN-TDMA slots.

**m**:
Subscript $m$ indicates a measured value.

**a**:
Subscript $a$ indicates an approximated value.

**e**:
Subscript $e$ indicates an error: $x_e = x_a - x_m$.

### 5-2-2 Local timer synchronisation

As already explained in Section 4-2-1, the microcontroller needs to know about the DW1000 system timer. Reading out this system time has shown to sometimes take very long.

#### Experimental setup

It was chosen to map periodic DW1000 system time readings to readings from a local hardware timer. At any point, the latest two measured datapoints $t_{m,i-1}$ and $t_{m,i}$, $\Delta t = t_{m,i} - t_{m,i-1}$ can be used to approximate some time $t_{a,i+1}$, where $t_{m,i} \leq t_{a,i+1} \leq t_{m,i} + \Delta t$. To test the local timer synchronisation, the DW1000 system time was read out every $10ms$.

#### Maximum errors in timer synchronisation

This dataset was used offline to determine the maximum errors between measured and approximated DW1000 system timestamps ($t_{e,max}$) when increasing $\Delta t$. The results are plotted in Figure 5-13.

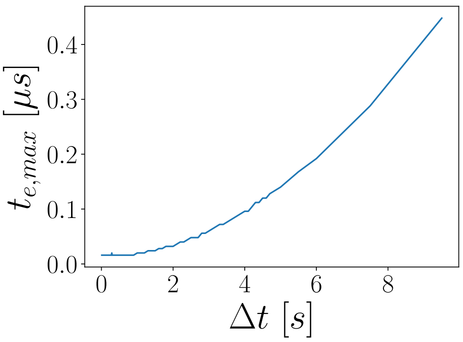


**Figure 5-13:** Maximum error in DW1000 system timer approximation for different sample times.

Evidently, even when sampling every second, the maximum approximation errors still stay within tens of nanoseconds which is practically negligible. The linear extrapolation method shows to be a good and very cheap estimator of the DW1000 system time, and will thus not significantly increase the needed guard time in the AN-TDMA slots.

### 5-2-3 Node synchronisation

A node in AN-TDMA synchronises its belief about the start time a slot to its neighbour's belief, every time it receives a message from its neighbour. Let the belief of node $N_i$ about the start of slot $k$ be ${}^{i}t_k$. $N_i$ becomes aware of an offset $t_{\mathit{offs}} = {}^{j}t_k - {}^{i}t_k$ between its belief ${}^{i}t_k$ and its neighbour's belief ${}^{j}t_k$, when $N_i$ receives a message that at ${}^{j}t_k$. To stay synchronised, at least one of the two nodes needs to adjust to the other. In the Design chapter, three methods of adapting to a neighbour's belief were proposed (Section 3-2-3):

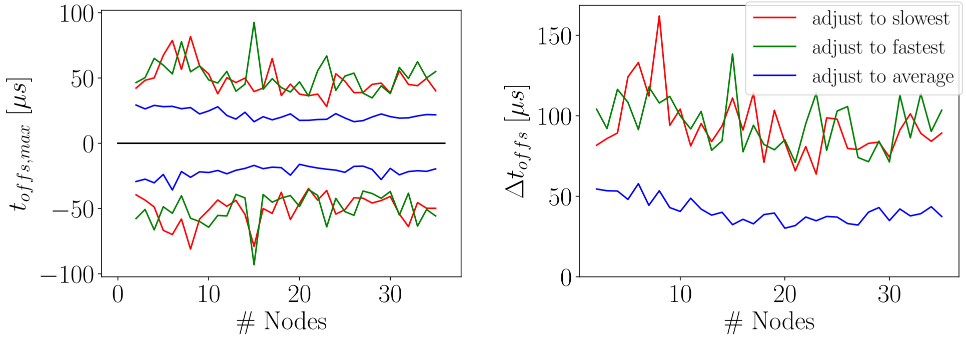
$$ {}^{i}t_{k,new} = \frac{{}^{i}t_k + {}^{j}t_k}{2} \tag{5-1}$$

$$ {}^{i}t_{k,new} = \max {}^{i}t_k, {}^{j}t_k \tag{5-2}$$

$$ {}^{i}t_{k,new} = \min {}^{i}t_k, {}^{j}t_k \tag{5-3}$$

Nodes may take the average of two beliefs (Equation (5-1)), always adjust to the slowest of the two (Equation 5-2) or always adjust to the fastest (Equation (5-3)).

Investigating which of these methods introduces the lowest maximum offset $t_{\mathit{offs},max}$ is interesting, because the expected maximum offset directly relates



**(a)** Maximum timing errors for increasing numbers of nodes.

**(b)** Required valid segment time for increasing numbers of nodes

**Figure 5-14:** Node synchronisation performance for increasing numbers of nodes

to the required duration of the *valid* segment in the AN-TDMA slot. When the *valid* segment duration can go down, the slot duration goes down and thus the maximum possible throughput increases.

### Experimental setup

The three methods were tested in a simulation for three different situations, because insufficient hardware was available to do any meaningful testing. The uncalibrated timer of the DW1000 transceiver is rated at $20ppm$ [21]. For a conservative estimate, clock offsets were chosen with a standard deviation of $20ppm$. Each test described below is performed 200 times with 36 nodes (unless stated otherwise), and 36 slots of $3ms$. Plotted errors are the extremes as found over all tests, and thus approximate worst-case behaviour.

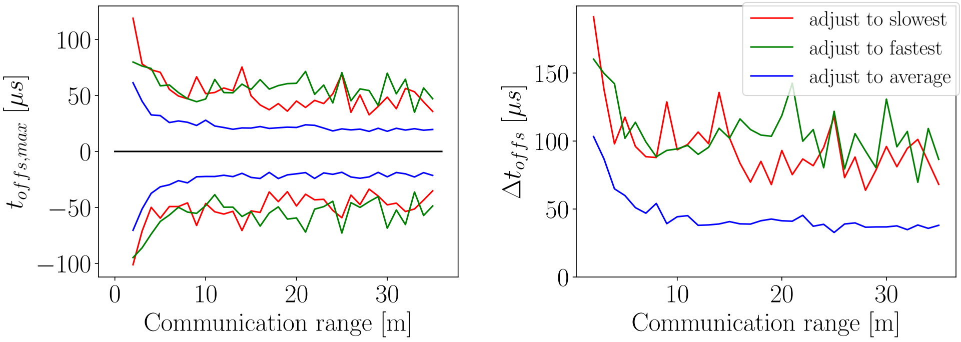
### Complete network, increasing number of nodes

First, we test the influence of adding nodes to a completely connected network. Figure 5-14 plots the maximum positive $t_{offs,max+}$ and negative offsets $t_{offs,max-}$. The difference $\Delta t_{offs}$ is the needed minimum *valid* segment time to account for all clock offsets.

The figure shows that, even though the method of Equation (5-1) might introduce oscillations (not visible in the plot), these possible oscillations are clearly preferable over the offsets introduced when all nodes have to follow along with the slowest or fastest node in the network. A likely reason is that when nodes adjust using Equation (5-2) and the slowest node in the network does not transmit for some time, the network adjusts consistently to the slowest *transmitting* node. However, the slowest node in the network will not adjust to the slowest transmitting node, because that node is considered the faster node. When the slowest node then starts transmitting in slot $k$, its belief about $t_k$ may have become vastly different from that of the rest of the network.

### Line network topology

The network may not be fully connected, so to test the influence of connectivity on the required *valid* segment time, the nodes are placed in a line one unit distance apart. Figure 5-15 shows that even when the network is only barely connected and nodes can only communicate with their two direct neighbours, adjusting the slot start time belief by averaging still only requires a *valid* window of $100\mu s$.



**(a)** Maximum timing errors for increasing connectivity

**(b)** Required valid segment time for increasing connectivity.

**Figure 5-15:** Node synchronisation performance for 36 nodes in a line topology.

When the length of the line is increased and the communication range is kept at one unit distance, connectivity is at its worst. Figure 5-16 illustrates that when nodes are added at the end of the line and the number of slots is set at the number of nodes, synchronisation performance decreases linearly with each added node, even when synchronisation is done with the averaging method. This should not come as a surprise. Each node can only hear from its two neighbours, and when the number of slots increases, the time between messages to synchronise to increases, and thus the clock-induced errors increase.

As argued in Section 3-2-2, timeslots can be shared by nodes at least two hops apart. This in the case of a line topology, three timeslots would be sufficient. The importance of slot reuse is depicted in Figure 5-17, which shows that adding more nodes does not decrease synchronisation performance very much, but using more slots does. As a rule of thumb, Figures 5-16 and 5-17

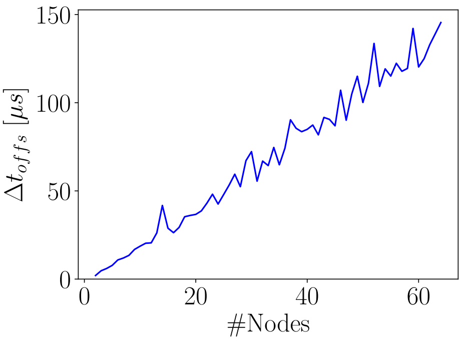


**Figure 5-16:** Line topology with synchronisation by averaging, adding nodes and slots.

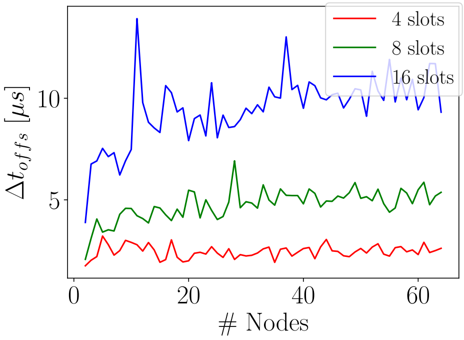


**Figure 5-17:** Line topology with synchronisation by averaging, adding nodes and reusing slots.

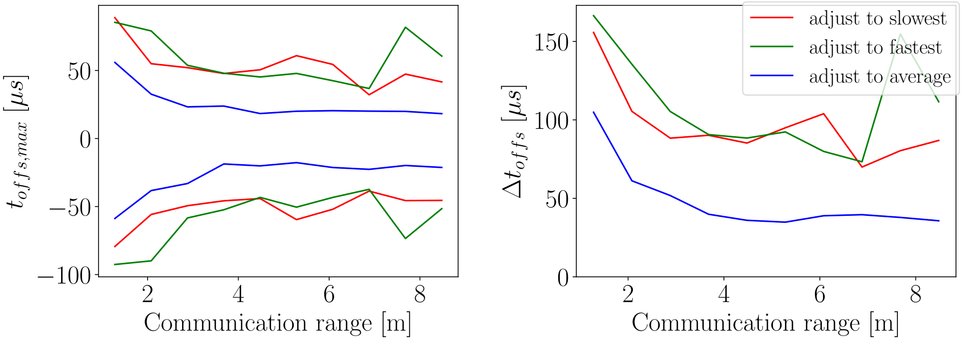
imply that for the used slot size of $3ms$ and clocks with a standard deviation of $20ppm$, the required *valid* range in the worst-case scenario of a line network is approximately twice the number of nodes in microseconds.

**Grid network, increasing communication range**
The more realistic approximation of a robot swarm is a grid network. As an approximation, the nodes are placed in a square and the communication range is gradually increased to simulate a sparsely or densely occupied swarm. See Figure 5-18. The results are in line with what was previously found, and an averaging update scheme is the clear winner. Nodes are shown to synchronise within a smaller range than with the line topology, so the metric of the *valid* range being twice the number of slots in microseconds holds.

### 5-2-4 Channel utilisation

The reason to implement AN-TDMA in the first place is to increase the channel utilisation $\eta$, as was discussed in Section 3-2-1. The range of the *valid* segment has been discussed in the previous section, which leaves the durations of the *guard* segment and the *Two-Way Ranging (TWR)* segments of the slot to be discussed.

For reference, the slot segments for TWR in AN-TDMA are shown again in Figure 5-19. This figure defines the time $t_{twr}$, which is the time it takes from the moment the command is issued to transmit the *req* message, to the time the interrupt arrives that signifies the arrival of the final *summ* message (see Figure 4-2 for reference). The *guard* time includes the time it takes to setup the first transmission and handle the last reception.



**(a)** Maximum timing errors for increasing connectivity.

**(b)** Required valid segment time for increasing connectivity.

**Figure 5-18:** Node synchronisation performance for 36 nodes in a square grid topology.

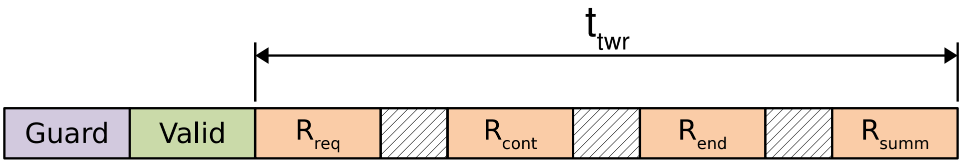


**Figure 5-19:** Illustration of $t_{twr}$.

### Experimental setup

Two nodes were set up to perform rangings with the short-range transceiver settings from Table 5-1, while adapting to each other's timing according to Equation (5-1). The TDMA frame consists of 32 slots, each with a length of 3ms. Slot selection was performed by taking the first free slot available, so the two nodes ended up in subsequent slots, and both nodes can initiate rangings to each other.

### Duration of TWR

Measurements of $t_{twr}$ are shown in Figure 5-20. Although the vast majority of rangings show to be fairly constant in duration at around $2,248\mu s$, a very small number takes at most $20\mu s$ more. The most likely cause is that one of the SPI commands was waiting in the transaction queue for a scheduled DW1000 system time reading to finish. If these irregularities pose a problem in the future, the system time readings can be scheduled more intelligently to prevent transactions during a TWR message exchange. One possibility is to read the system timer only after the *valid* time of a slot has passed and no message is being received yet. In the current version of AN-TDMA this

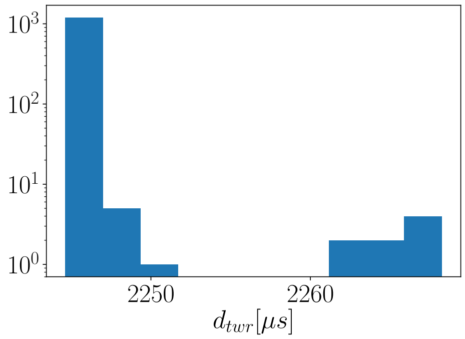


**Figure 5-20:** Measured times for a full TWR message exchange.

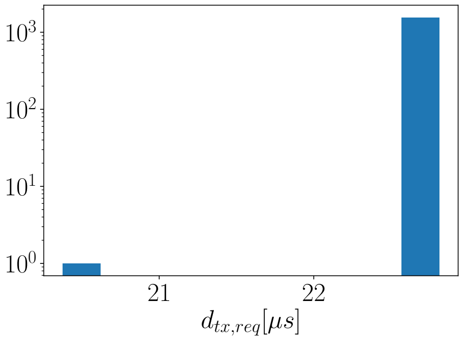


**Figure 5-21:** Measured time to prepare a *req* transmission.

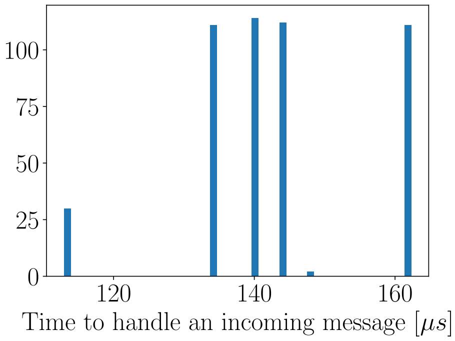


**Figure 5-22:** Measured time to process received messages.

is not implemented, so a conservative $t_{twr} = 2,500\mu s$ has been reserved for TWR.

### Duration of the *guard* time

The duration of the *guard* time is determined by the time it takes to handle the reception of a *summ* message and the time it takes to prepare the transmission of a *req* message. Data that is scheduled for transmission can be written to the DW1000 transmit buffer at any time in advance, so this does not influence the *guard* time. Therefore, transmission preparations are short and consistently take $22.7\mu s$, as plotted in Figure 5-21.

The time it takes to process a received message is taken from the moment the interrupt occurs that signals the availability of a new message, to the moment the receiver of the DW1000 has been re-enabled. When that command has finished, the DW1000 is ready to receive a new message and the next slot can start. The bars in Figure 5-22 nicely show the handling times of the five different message types that are being used: four for TWR, and

**Table 5-2:** Airtimes of the TWR messages.

| | **Message length** [bytes] | **Airtime** $t_{air}[\mu s]$ |
|---|---|---|
| Range request | 23 | 189 |
| Range continue | 35 | 201 |
| Range end | 51 | 219 |
| Range summary | 51 | 229 |
| Range total | | 838 |

one beaconing message to allow discovery by new neighbours. The *summ* message is the longest of the five, and takes $162\mu s$ to process. In total, the *guard* time should thus be at least $184.7\mu s$. Including some leeway for safety, the *guard* time is chosen to be $t_{guard} = 250\mu s$.

### Channel utilisation comparison

Table 5-2 shows the time the messages of TWR are actually in the air, using the short-range settings from Table 5-1.

The channel utilisation can then be determined by

$$\eta_{air} = \frac{t_{air}}{t_{slot}} = \frac{t_{air}}{t_{guard} + t_{valid} + t_{twr}} \tag{5-4}$$

Filling in the equation for 16, 64 and 128 slots results in channel utilisations of 30.1%, 29.1% and 27.8%, respectively. The differences are due to the increased required $t_{valid}$, as determined in Section 5-2-3. Compared to ALOHA, the increase in channel utilisation is 55% to 67%. Not only does AN-TDMA prevent the collisions that inevitably happen in ALOHA, it increases the throughput of the network as well.

# Chapter 6

# Conclusions

At the TU Delft, the six-legged zebros (Dutch: ZEs Benige RObots) are developed to form a large homogeneous robot swarm that can function as a self-deploying sensor network for monitoring remote locations without having to rely on existing supporting infrastructure. Multiple algorithms to let the zebros move and cooperate as a swarm have been developed. These algorithms require each zebro to communicate with its neighbours, and to have an estimate of where its neighbours are located. However, zebros do not have these capabilities yet. Existing methods found in literature either did not allow for localisation without existing infrastructure [14][39], or did not allow scaling to a large swarm, because they are very communication intensive [36] [50], or because they need global knowledge [36] [56].

To solve this problem, a framework for relative localisation named Tangolation has been designed and implemented, to estimate the location of a neighbour as a distance and an angle. The quality of the estimation is indicated by a confidence value from 1 (low) to 5 (high). In Tangolation, every five seconds two nodes simultaneously determine each other's location by measuring the distance to each other every five seconds, and exchanging the vector of their displacement between the measurements. By allowing localisation of a neighbouring zebro without requiring common neighbours or external infrastructure with only one message exchange per neighbour every five seconds, Tangolation does not put restrictions on the size of the swarm. The DW1000 ultra-wideband transceiver has been used for communcation and ranging measurements, and a communication layer called Anarchic TDMA (AN-TDMA) has been implemented to support Tangolation. Even though testing with real robots has not been possible due to timing constraints and locomotion bugs in the zebros, extensive testing has been done in a simulation with conservative estimates about noise behaviour in the real world.

The raw ranging measurements showed to have a standard deviation of $\sigma_r = 2cm$, and thus proved to be very precise already. The added value of Tangolation is its ability to estimate the angle a neighbour is located at. While displacement estimates are generally noisy when no external reference frame is available, Tangolation seems to be unimpressed due to the precision of the ranging input, and has shown to be fairly robust against uncertainties in displacement estimates. It has been shown that with the conservative determined noise estimates of $\sigma_r = 2cm$, $\sigma_{\theta,e} = 8.6\%$ and $\sigma_{d,e}$, 95% to 99% of the angle measurements are within the maximum error range of 22° from the true angle when robots are programmed to stay at a range of $6m$ or more from each other. Depending on the confidence value that Tangolation reports, 65% to 84% of the angle estimates is within 10° of the true value.

The AN-TDMA approach to communication has been shown effective, and increases reachable channel utilisation ratios by 55% to 67% compared to the ALOHA method recommended by Decawave. Additionally, it allowing collision-free transmissions without requiring any extra infrastructure, depending on the time slot selection method.

## 6-1 Limitations

Tangolation has shown degradation in performance in some situations, most notably when the swarm density increases. The uncertainty about the localisation error grows with a standard deviation of more than 15%, where only 85% of angle estimations have the required accuracy. It shows that Tangolation is not a be-all, end-all solution, and that it is not very well suited for short-range localisation. The main source of uncertainty at the moment, though, is its lack of testing time with actual zebros. Many hours of simulations with very conservative noise estimates have been run, but the real world has a tendency to be more chaotic and unpredictable than a computer simulation. Although the first results are very encouraging, Tangolation still needs to prove itself in the field.

Also the communication layer is limited at the moment, because of the current naive slot selection method. Distributed methods for slot selection in a near static network have been found abundantly, but no tried and tested method has been found for this specific usecase.

## 6-2 Future work

The currently open problems and some proposals for improval are given below.

### Research and implement smart slot selection

As mentioned before, the main open problem at the moment is to determine which slot is safe to select. A possible solution could be to piggy back data with neighbours' phases and corresponding addresses on the message that is periodically broadcast to announce presence. When a node discovers that its own slot is taken by a neighbours' neighbour, it would select a new slot, based on the two-hop neighbour information it collects from all its neighbours' broadcasts. Time slots are sufficiently long to allow for four Two-Way Ranging (TWR) messages, so when a node has too many neighbours, it might even split up the announcement message.

### Enhance ranging

Currently, nodes range one-to-one in a single slot. The number of messages can be almost cut in half when the slot duration is significantly increased, and allows for multiple simultaneous rangings. Clock-induced ranging errors are small [21], so nodes do not need to reply instantly. A node could broadcast its ranging request, and specify which neighbours it expects to respond at what point in the slot. When all addressed neighbours have responded, the initiating node can broadcast the third message containing all receive timestamps. At last, all addressed neighbours answer at their designated point in the slot. Using this method, the number of messages for ranging with 10 neighbours goes down from 40 to just 22. Note that the increased slot size requires a reeveluation of $t_{valid}$. Additionally, sending a single message in a slot that is designed for 22 messages would bring down channel utilisation dramatically, so one might want to distinguish between slots specific for ranging and slots for normal communication.

### Investigate replacing the particle filter with a Kalman filter

A discussion with dr. Rajan implied that a Kalman filter may be able to capture the situation of a neighbour being in one of multiple locations. A Kalman filter uses considerably less memory, and might be faster as it does not require multiple Monte-Carlo simulations to be run. Additionally, a Kalman filter might even produce better localisation estimates, although that should be researched first. Especially when the number of particles is low, as it is in our implementation of Tangolation, it becomes more likely that none of the Monte-Carlo simulations is close to the actual location, thus introducing an extra source of noise. A Kalman filter on the other hand has been proven to be an optimal linear filter when the model is correct.

**Remove the reliance on the magnetometer**

Cheap magnetometers are known for being noisy and easily influenced by the environment. Furthermore, they need to be manually calibrated before use. As mentioned in Chapter 2, methods have been found to accomplish localisation without a global reference frame, although these methods tend to be computationally intensive. Finding a way to replace or remove the magnetometer would make for a more resilient method of localisation in the real world.

# Bibliography

[1] Abramson, N. (1970). The Aloha System - Another Alternative For Computer Communications. In *American Federation of Information Processing Societies*, pages 1–17.

[2] Al-Meshhadany, T. and Ajib, W. (2007). New CDMA-based MAC protocol for Ad Hoc networks. In *IEEE Vehicular Technology Conference*, number November 2014, pages 91–95.

[3] Arvin, F., Murray, J. C., Shi, L., Zhang, C., and Yue, S. (2014). Development of an autonomous micro robot for swarm robotics. *IEEE International Conference on Mechatronics and Automation, IEEE ICMA 2*, pages 635–640.

[4] Bayindir, L. (2016). A review of swarm robotics tasks. *Neurocomputing*, 172:292–321.

[5] Becker, C., Salas, J., Tokusei, K., and Latornbe, J.-c. (1995). Reliable Navigation Using Landmarks. *IEEE International Conference on Robotics and Automation*, pages 401–406.

[6] Beni, G. (2004). From Swarm Intelligence to Swarm Robotics. *Lecture Notes in Computer Science3*, 37(2):305–312.

[7] Betke, M. (1995). Mobile Robot Localization using Landmark. *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, 13(2):251–263.

[8] Bosch (2014). BNO055 Intelligent 9-axis absolute orientation sensor.

[9] Boston Dynamics (2019a). Boston Dynamics Spot.

[10] Boston Dynamics (2019b). Boston Dynamics Spot Mini.

[11] Buck, J. (1988). Synchronous Rhythmic Flashing of Fireflies. *The Quarterly Review of Biology*, 63(3):265–289.

[12] Buranapanichkit, D. and Andreopoulos, Y. (2016). Distributed time division multiple access protocol for multi-hop wireless sensor networks. In *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, pages 1–4. IEEE.

[13] Buranapanichkit, D., Deligiannis, N., and Andreopoulos, Y. (2015). Convergence of desynchronization primitives in wireless sensor networks: A stochastic modeling approach. *IEEE Transactions on Signal Processing*, 63(1):221–233.

[14] Canciani, A. J. (2016). *Absolute Positioning Using the Earth's Magnetic Anomaly Field.* PhD thesis.

[15] Chaudhary, M. H. and Scheers, B. (2013). Progressive Decentralized TDMA based MAC : Joint Optimization of Slot Allocation and Frame Lengths. Number November.

[16] de Groot, J. (2017). *Swarm behaviour for the Zebro Robot (MSc Thesis).* Number November.

[17] Decawave Ltd (2014a). APH0007 Antenna Selection / Design. pages 1–8.

[18] Decawave Ltd (2014b). APS011: Sources of Error in DW1000 based TWR schemes. pages 1–21.

[19] Decawave Ltd (2015). ScenSor DWM1000 Module.

[20] Decawave Ltd (2017a). DW1000 Channel Interferece. How transmissions on one DW1000 channel can affect other channels and how to minimize that effect v1.1. Technical report.

[21] Decawave Ltd (2017b). DW1000 User Manual.

[22] Decawave Ltd (2017c). DWM1001 Datasheet.

[23] Decawave Ltd (2018). DWM1001 Datasheet. pages 1–23.

[24] Degesys, J. and Nagpal, R. (2008). Towards desynchronization of multihop topologies. *Proceedings - 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2008*, pages 129–138.

[25] Degesys, J., Rose, I., Patel, A., and Nagpal, R. (2007). DESYNC Self-Organizing Desynchronization and TDMA on Wireless Networks.pdf. *International Symposium on Information Processing in Sensor Networks*, pages 11–20.

[26] Ding, J., Zhao, L., Medidi, S. R., and Sivalingam, K. M. (2002). MAC protocols for ultra-wide-band (UWB) wireless networks: impact of channel acquisition time. *Proc. SPIE ITCom*, pages 1953–1954.

[27] Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S. M., and Christensen, A. L. (2016). Evolution of collective behaviors for a real swarm of aquatic surface robots. *PLoS ONE*, 11(3).

[28] Dunkels, A., Mottola, L., Tsiftes, N., Osterlind, F., Eriksson, J., and Finne, N. (2003). Relative Location Estimation in Wireless Sensor Networks. *Ieee Transactions on Signal Processing*, 6567(8):211–226.

[29] Fantacci, R., Ferri, A., and Tarchi, D. (2005). A MAC technique for CDMA based ad-hoc networks. *IEEE Wireless Communications and Networking Conference, WCNC*, 1(1):645–650.

[30] Guo, K., Qiu, Z., Meng, W., Nguyen, T. M., and Xie, L. (2016). Relative Localization for Quadcopters using Ultra-wideband Sensors. In *International Micro Air Vehicle Conference and Competition3*.

[31] Hsieh, T. H., Lin, K. Y., and Wang, P. C. (2015). A hybrid MAC protocol for wireless sensor networks. *ICNSC 2015 - 2015 IEEE 12th International Conference on Networking, Sensing and Control*, 16(3):93–98.

[32] Hung, W.-C., Law, K. L. E., and Leon-Garcia, A. (2002). A Dynamic Multi-Channel MAC for Ad Hoc LAN. *21st Biennial Symposium on Communications*, pages 31–35.

[33] IEEE Computer Society (2011). *IEEE Standard Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*. Number September. IEEE.

[34] Kurazume, R. and Hirose, S. (2000). An experimental study of a Cooperative Positioning System. *Autonomous Robots*, 8(1):43–52.

[35] Leidenfrost, R. and Elmenreich, W. (2009). Firefly clock synchronization in an 802.15.4 wireless network. *Eurasip Journal on Embedded Systems*, (1).

[36] Leung, K. Y. K., Barfoot, T. D., and Liu, H. (2010). Decentralized Localization of Sparsely-Communicating Robot Networks: A Centralized-Equivalent Approach. *IEEE Transactions on Robotics*, 26(1):62–77.

[37] Liu, R., Yuen, C., Do, T. N., Jiao, D., Liu, X., and Tan, U. X. (2017). Cooperative relative positioning of mobile users by fusing IMU inertial and UWB ranging information. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5623–5629.

[38] Lockspeiser, J. R., Don, M. L., and Hamaoui, M. (2017). Radio Frequency Ranging for Swarm Relative Localization. Technical report.

[39] Matsuda, T., Maki, T., Sato, Y., and Sakamaki, T. (2018). Experimental Evaluation of Accuracy and Efficiency of Alternating Landmark Navigation by Multiple AUVs. *IEEE Journal of Oceanic Engineering*, 43(2):288–310.

[40] Miog, J. (2018). Design recommendations for CARL, a Communication And Relative Localization system for swarming applications.

[41] Mühlberger, C. and Kolla, R. (2009). Extended desynchronization for multi-hop topologies. *9. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze*, pages 21–24.

[42] Nishizawa, T., Ohya, A., and Yuta, S. (1995). An Implementation of On-board Position Estimation for a Mobile Robot. *IEEE International Conference on Robotics and Automation*, pages 395–400.

[43] Patro, R. and Mohan, B. (2005). Mobile agent based TDMA slot assignment algorithm for wireless sensor networks. *International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II*, 2:1–5.

[44] Reinoso, G. and Guevara, G. (2005). Swarm Robotics: From Sources of Inspiration to Domains of Application. *Lecture Notes in Computer Science*, 37(2):305–312.

[45] Reynolds, C. W. (1987). Flocks , Herds , and Schools : A Distributed Behavioral Model 1. 21(July):25–34.

[46] Rhee, I., Warrier, A., Min, J., and Xu, L. (2009). DRAND: Distributed randomized TDMA scheduling for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 8(10):1384–1396.

[47] Ridolfi, M., van de Velde, S., Steendam, H., and De Poorter, E. (2018). Analysis of the scalability of UWB indoor localization solutions for high user densities. *Sensors (Switzerland)*, 18(6).

[48] Roumeliotis, S. I. and Bekey, G. A. (2002). Distributed Multi-Robot Localization. *IEEE Trans. on Robotics and Automation*, 18(5):781–795.

[49] Rubenstein, M., Ahler, C., and Nagpal, R. (2012). Kilobot: A low cost scalable robot system for collective behaviors. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3293–3298.

[50] Strader, J., Gu, Y., Gross, J. N., De Petrillo, M., and Hardy, J. (2016). Cooperative relative localization for moving UAVs with single link range measurements. *Proceedings of the IEEE/ION Position, Location and Navigation Symposium, PLANS 2016*, pages 336–343.

[51] Su, P., Escudero, C. J., and Garc, J. A. (2016). Assessment of UWB Ranging Bias in Multipath Environments. In *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 8–11.

[52] Tan, Y. and yang Zheng, Z. (2013). Research Advance in Swarm Robotics. *Defence Technology*, 9(1):18–39.

[53] Trawny, N. and Roumeliotis, S. I. (2010). On the global optimum of planar, range-based robot-to-robot relative pose estimation. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3200–3206.

[54] Turgut, A. E., Gökce, F., Celikkanat, H., Bayındır, L., and Sahin, E. (2007). Kobot: A mobile robot designed specifically for swarm robotics research. *ACM SIGACT News*, 38(1).

[55] Tyrrell, A. and Auer, G. (2008). Decentralized Slot Synchronization for Cellular Mobile Radio. *NTT DoCoMo Technical Journal*, 10(1):60–67.

[56] Yang, C., Strader, J., Gu, Y., Hypes, A., Canciani, A., and Brink, K. (2018). Cooperative UAV Navigation using Inter-Vehicle Ranging and Magnetic Anomaly Measurements. *2018 AIAA Guidance, Navigation, and Control Conference*, (January).

[57] Zhu, C. and Corson, M. S. (2000). An Evolutionary-TDMA Scheduling Protocol for Mobile Ad Hoc Networks. *Proc. Advanced Telecomm. and Information Distribution Research Program (ATIRP '00)*, 14.

# Glossary

## List of Acronyms

| | |
|---|---|
| **WSN** | Wireless Sensor Network |
| **RSS** | Received Signal Strength |
| **UAV** | Unmanned Aerial Vehicle |
| **PAN** | Personal Area Network |
| **TDMA** | Time Division Multiple Acces |
| **FDMA** | Frequency Division Multiple Acces |
| **CDMA** | Code Division Multiple Acces |
| **CSMA** | Channel Sensing Multiple Access |
| **CSMA-CA** | Channel Sensing Multiple Access with Collision Avoidance |
| **CTS** | Clear To Send |
| **RTS** | Request To Send |
| **TWR** | Two-Way Ranging |
| **TDoA** | Time Difference of Arrival |
| **AN-TDMA** | Anarchic TDMA |