Synchronized MAC layer for ultra-wideband wireless sensor network

Design, implementation, analysis, and evaluation

ALESSANDRO MONGE

KTH Information and Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

Synchronized MAC layer for ultra-wideband wireless sensor network

Design, implementation, analysis, and evaluation

Alessandro Monge
alessandromonge88@gmail.com

25 March 2013

MarnaTech AB
Falks väg 18
18254 Djursholm
Sweden

MarnaTech Industrial Supervisor: Peter Reigo
KTH Academic Supervisor: Gerald Q. Maguire Jr.
PoliTo Academic Supervisor: Guido Albertengo

School of Information and Communication Technology
KTH Royal Institute of technology

Stockholm, Sweden

Abstract

The necessity of interconnecting objects more and more, possibly with high mobility, has pushed the telecommunications industry to recently develop new wireless standards in order to guarantee tracking of devices and to provide integration with well-known worldwide networks such as the Internet. Within these standards the role played by power consumption is implicit, hence power consumption needs to be as low as possible in order to fulfill long-life requirements and to offer the opportunity of locating and moving smart objects in the coverage area while relying only on batteries as the device's power source. Considering the importance of identifying and tracking these smart objects with high accuracy and high precision, this document propose an implementation of an IEEE 802.15.4a MAC layer, exploiting ultra-wideband wireless technology, with a time synchronization algorithm included for precise Time Difference of Arrival indoor positioning.

Nevertheless, this thesis demonstrates the advantages of using UWB for indoor wireless communication, due to its accuracy in localization and its robustness against interference.

A demonstration network has been analyzed consisting of four main base stations optimistically displaced at the corners of a room gathering timestamps from a central tag moving within the space where the UWB signal is within range. These timestamps are collected in one of the base station which plays the role of the coordinator and sends this information to a server which computes the position of the tag using TDOA formulation.

The main focus of this work is the synchronization algorithms used to synchronize the four base stations and secondly to synchronize the coordinator with the tag. Particular interest is placed on the protocol, the kind of messages exchanged, and the procedure used to maintain a good level of synchronization and to avoid unwanted clock drifts.

Moreover the thesis gives some hints of potential future improvements and proposes a possible solution for large-scale scenarios involving the installation of additional base stations for higher coverage and integration of a larger number of tags, with a focus on synchronization, collision avoidance, and routing procedures to better fit the situation of a larger network and more tags.

As a result, all the assumptions and the methodologies applied give evidence of how difficult it is to meet contemporary requirements for position accuracy, low power consumption, limited memory, and small message exchange when utilizing low-power and lossy networks and to address problems which need to be further studied in the future. The results of this thesis project offer a good proof of the possibility to reach high accuracy in terms of localization when exploiting UWB radio technology and redundant time synchronization algorithms with the help of TDOA measurements.

Sammanfattning

Nödvändigheten av sammankopplade objekt mer och mer, eventuellt med hög rörlighet, har drivit telekombranschen till nyligen utveckla nya trådlösa standarder för att garantera spårning av enheter och att ge integration med välkända världsomspännande nätverk som Internet. Inom dessa standarder roll strömförbrukningen är implicit, måste därför strömförbrukningen ska vara så låg som möjligt för att uppfylla lång livslängd krav och erbjuda möjligheten att lokalisera och flytta smarta objekt i täckningsområdet medan enbart med hjälp av batterier som enhetens strömkälla. Med tanke på vikten av att identifiera och spåra dessa smarta objekt med hög noggrannhet och hög precision, detta dokument föreslå ett genomförande av en IEEE 802.15.4a MAC-lager, utnyttja ultrabredbandsteknik trådlös teknik, med en tidssynkronisering algoritm ingår för exakt tidsskillnaden för ankomst inomhus positionering.

Ändå visar denna avhandling fördelarna med att använda UWB för inomhus trådlös kommunikation, på grund av dess noggrannhet i lokalisering och robusthet mot störningar.

En demonstration nätverk har analyserats består av fyra huvudsakliga basstationer optimistiskt förskjutna i hörnen av ett rum samla tidsstämplar från en central tagg flyttar inom utrymme där UWB signalen är inom räckhåll. Dessa tidsangivelser samlas i en av basstationen som spelar rollen av samordnare och skickar denna information till en server som beräknar position taggen med TDOA formulering.

Tyngdpunkten i detta arbete är att synkronisering algoritmer som används för att synkronisera de fyra basstationer dels att synkronisera samordnaren med taggen. Särskilt intresse läggs vid protokollet, den typ av utbytta meddelanden och det förfarande som används för att upprätthålla en god nivå av synkronisering och för att undvika oönskade klocka drivor.

Dessutom avhandlingen ger några tips om potentiella framtida förbättringar och föreslår en möjlig lösning för storskaliga scenarier som innebär installation av ytterligare basstationer för högre täckning och integration av ett större antal taggar, med fokus på synkronisering, att undvika kollision och routing förfaranden för att bättre passa situationen i ett större nätverk och fler taggar.

Som ett resultat, som tillämpas alla antaganden och metoder vittnar om hur svårt det är att uppfylla dagens krav på positionsnoggrannhet, låg strömförbrukning, begränsat minne, och små utbyte av meddelanden vid användning med låg effekt och förstörande nätverk och att ta itu med problem som måste studeras vidare i framtiden. Resultaten från denna avhandling projekt erbjuder en bra bevis på möjligheten att nå hög noggrannhet vad gäller lokalisering vid utnyttjande UWB radioteknik och redundanta tid algoritmer synkronisering med hjälp av TDOA mätningar.

Acknowledgements

First and foremost I would like to thank my supervisor and examiner at KTH, professor Gerald Q. (Chip) Maguire Jr., not only for his invaluable guidance and constant availability to help but also for his high comprehension in understanding the difficulties I met while working in a small and recently-opened start-up.

I wish to thank also all the people I got to know during this last year and a half in Stockholm, who have shared with me unforgettable moments and have made my experience abroad more enthusiastic. Some of them have been important from the academic perspective, some other simply affectively and some other even both, but all of them have contributed in shaping the person I am now and for this reason will have always a unique place in my memory for the rest of my life.

Nevertheless, I want to thank my friends in Italy, who has never abandoned me and have contacted me constantly to remind I could count on them in case of necessity, spiritual or material.

Special thanks to my family, that despite of the distance and despite of the economic effort which was necessary to make me achieve this Double Degree in Sweden, has always stood by my side and has always encouraged me to keep going, advising me daily.

In addition I want to mention all the people who helped me through my education as an engineer in my home school, Politecnico di Torino, my supervisor there, whose dedication to the profession has made me grow my passion for technology and my willingness to learn every day something new.

Last, but not least, I would like to express my thankfulness to Peter Reigo and Marnatech in general, for their guidance and economic support, and for having broadened the perspective of this thesis from an academic project to a commercial product.

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ACK	Acknowledgement
AoA	Angle-of-Arrival
AODV	Ad-hoc On-demand Distance Vector
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSS	Chirp Spread Spectrum
DODAG	Destination Oriented Direct Acyclic Graph
DPS	Default Parameters Set
DYMO	Dynamic MANET On-demand Routing Protocol
EUI	Extended Unique Identifier
FCF	Frame Control Field
FCF	Frame Check Sequence
FTSP	Flooding Time Synchronization Protocol
HRTS	Hierarchy Referencing Time Synchronization protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IPHC	IP Header Compression
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ICMP	Internet Control Message Protocol
JTAG	Joint Test Action Group
LLN	Low power and Lossy Networks

LOS	Line of Sight
LTS	Lightweight Time Synchronization protocol
MAC	Medium Access Control
MANET	Mobile Ad-hoc Network group
MPDU	MAC Protocol Data Unit
MSDU	MAC Service Data Unit
MTSP	Multi-hop Time Synchronization Protocol
NEMO	NEtwork MObility
OLSR	Optimized Link State Routing Protocol
OOK	On-Off Keying
PAM	Pulse Amplitude Modulation
PHR	Physical layer Header
PPM	Pulse Position Modulation
PPDU	Physical Protocol Data Unit
PRF	Pulse Repetition Frequency
PSDU	Physical Service Data Unit
RBS	Reference Broadcast Synchronization
RFC	Request For Comments
RFID	Radio Frequency IDentifier
ROHC	RObust Header Compression
RPL	Routing Protocol for LLN
RSSI	Received Signal Strength Indication
RTLS	Real Time Localization Scheme
RTT	Round Trip Time
SDS-TWR	Symmetric Double Sided-Two Way Ranging
SHR	Synchronization header
SPI	Serial Peripheral Interface bus
TCP/IP	Transmission Control Protocol and Internet Protocol
TDMA	Time Division Multiple Access
TDoA	Time-Difference-of-Arrival

ToA	Time-of-Arrival
ToF	Time of Flight
TPSN	Timing-sync Protocol for Sensor Networks
UDP	User Datagram Protocol
UWB	Ultra-wideband
WLAN	Wireless Local Area Networks
μIP	Micro IP

Chapter 1

Introduction

This chapter gives a general introduction to the whole thesis, highlighting the area of expertise which is going to be treated, the problem which wants to be solved, the scope of the work as well as the necessary background and the structure of the paper.

1.1 General introduction to the area

The interest on wireless sensor networks has grown rapidly over the last few years following the shift from focusing on sensing to instead focusing on interconnecting small objects used in ones daily environment (such as home automation, industrial monitoring, sanitary management, and sometimes even outdoors). Moreover, there is an increasing desire to link these sensors with the Internet in order to guarantee interoperability and accessibility of the data by the relevant users.

Given these considerations, it seems reasonable to embed a TCP/IP stack in such sensing devices. It also makes sense to exploit IPv6's vast address space to enable each device to be globally addressable. Unfortunately, the TCP/IP protocol suite was not intended for this application due to the resource-constrains of the devices and the limitations of the underlying link layer. Additionally the network layers features consume too much energy.

For this reason new standards have been created, based on proprietary protocol stacks which define addressing and routing protocols. An example of such a protocol stack is the well-known Zigbee, whose specification for a suite of high level communication utilize the IEEE 802.15.4 standard MAC and physical layers. A brief introduction to ZigBee will be given in Chapter 2.

At the same time the IETF defined 6LoWPAN (IPv6 over Low power Wireless Personal Area Network): an adaptation layer, which provides all the services required by the network layer and those the link layer cannot offer. Due to its compatibility with the Internet world through IPv6 addressing, this standard seems to be the leading standard for the future for wireless sensor networks.

However, the choice of a routing protocol and network layer is not the only problem that must be addressed with designing wireless personal networks for the identification, localization, and actualization of objects; the main problem in localizing smart objects within a wireless network is the precision that wants

to reach, which very often depends upon on the accuracy of the synchronization model that is adopted and at which layer this model is defined. Considering the necessity of wasting as little energy as possible, it is not always feasible to find the right combination of wireless technology and higher layer protocols. There is a need for good clock synchronization while avoiding the expense of unnecessarily repeating a synchronization procedure as this would increase power consumption.

1.2 Problem statement

With the wide adoption and exploitation of the recent technologies (WLAN, Wi-Fi, Bluetooth) the desire for positioning objects and people is becoming more and more compelling. The required resolution to visualize the position of the target either on a personal computer or on a smartphone through downloadable applications seems to be constantly increasing.

Moreover, companies nowadays expect easy-to-build network infrastructures, with fast installation, reduced space, robustness to harsh environments, long-life batteries, and high coverage.

Among these wireless technologies, ultra-wideband (UWB) has been studied in the recent years, although it is still not largely applied. It offers numerous advantages with respect to other wireless technologies, but it needs to redefine network specifications in order to make it suitable for the common standards and protocols in use within the field of sensor networks and to make it compatible and easily connectable with other networks such as Internet, 3G, GSM, etc.

Specifically, MAC layer and network layer must be redesigned to reduce the number of computations performed by the devices and their complexity, to decrease the number of instructions and data that needs to be stored inside the small memories, and to understand the information that can be extracted from the UWB signal.

Additionally, the need for every more accurate calculation of the target's position requires the study of synchronization protocols that would be able to meet the requisites explained above and to guarantee exact results every time the position is to be updated. Very often the layer in which to realize the synchronization algorithm determines how tightly clocks can be synchronized, but it is not always clear what choice should be made.

1.3 Project's scope

The goal of this thesis project was to first design and implement a MAC layer for a simple network composed of four base stations (BSs), whose position is known, receiving time data from a personal tag, whose position is within range of all the four base stations but whose exact position is unknown. One base station, elected as coordinator, should then collect all the timestamp data from the other base stations and send this data to a cloud-based server via an Ethernet connection (and possibly one or more routers to reach the server). The server computes the location of the tag using this data.

Such a network should be able to perform high-precision synchronization over the BS-BS links, within the order of nanoseconds, and maintain this level

of precision over time in order to enable the tag to update its position once each second. For the initial setup the base stations are connected to mains power, hence they have no limits on their power consumption when implementing the synchronization algorithm. However, the deployment of such an algorithm should be done keeping in mind the possibility of using battery powered devices in the future.

On the tag side, a battery is employed to power the tag. There is no necessity for high-precision synchronization. It is preferable to keep the tag's power consumption as low as possible to lengthen the tag's battery life.

A side goal of the project is to investigate the best solution for synchronization and networking protocols if the system were to be enlarged to a situation scenario in which additional base stations are employed and the number of tags is dramatically increased (as their might be hundreds of devices embedded in objects or carried by people moving around in the region covered by the UWB network. In this case additional issues, such as packet collision, arise and a different approach should be taken to emulate the typical wireless technologies by adopting clustering to subdivide the network into different regions.

1.4 Required background knowledge

Throughout this document many communication protocols will be examined and a basic knowledge of both these protocols and computer networks in general will help the reader comprehend the contents of this thesis. Furthermore, references to basic signal processing via wireless links and outlines of C code related to the development of the necessary functions within the MAC and network layer will be included in the text.

For completeness, in Chapter 2 an exhaustive overview of all these background topics will be carried out together with brief explanations of the arguments closely related with the main subjects under discussion. In addition, the thesis has a list of acronyms and abbreviation (see page xi) for which fundamental computer networking knowledge will be assumed.

It is important to note that a reader with great expertise in this field could skip Chapter 2 and go directly to the method proposed in Chapter 3, but even such an advanced reader may find it useful to review the description of some of the network protocols which will be modified for the case under study (as a good understanding of these would give the reader a better understanding of the choices adopted and the motivations behind them).

1.5 Thesis outline

This thesis is divided into seven chapters, which follow a logically sequential order as well (as well as following the chronological order of the project). These chapters, until the sixth, are grouped in pairs leading to a structure in which the first two chapters represents an introduction and literature study, the second two chapters describe what has been done while the last two chapters are a collection of the data that was collected, results, and a complete evaluation of the goals achieved, together with comments and conclusions. In addition, a final chapter analyzes possible future work both with regard to research and

industrial development.

The first chapter gives a general idea of the technical area which the project is addresses, states the problem and the goals to be achieved. Chapter 2 gathers all the necessary knowledge a reader should be familiar with before reading the rest of the thesis. The third chapter gives a description of the hypothesis, the requirements, and the possible limitations of the design of the network to be implemented. Chapter 4 describes the procedure adopted to solve the problem together with the plans for all of the experiments and test that were conducted. Chapter 5 gives the details of the evaluation - including an estimate of the correctness, comparison, and performance. The sixth chapter gives a general overview of all the achieved goals and contrasts them with the initial project goals. Finally, the last chapter, is meant to be a springboard for possible future paths, which take into consideration both research to improve the proposed technique and to enlarge the industrial perspective of the work that has been done.

Chapter 2

Background

This chapter gives an overview of all the necessary knowledge to face the further reading of this thesis.

2.1 The Internet of Things

In 1999 Kevin Ashton first used the term Internet of Things (IoT) to refer to a set of uniquely identifiable objects that can be represented virtually in an Internet-like structure. One of the main prerequisites for the construction of a network of objects interacting between each other is to have radio-frequency identification (RFID) so that computers could inventory them; this can be achieved equipping each item of interest (even people) with a radio-tag.

The advantages of such an environment could transform our daily lives. Consider the cases of a factory, which would never run out of stock or generate waste; in farming tracking animals' health or location could improve the production of aliments; in offices location and working hours of people could increase management efficiency by knowing whether people are available or occupied in conferences or sales meetings and in hospitals, the safety of patients could be improved also reducing the amount of human time that must be spent per patient. These and many other fields are still objects of curiosity and study with regard to the words Internet of Things.

Nevertheless, the IoT is a difficult concept to define precisely and the best clarification is given by Ashton who stated: "Today computers - and, therefore, the Internet - are almost wholly dependent on human beings for information. Nearly all of the roughly 50 petabytes (a petabyte is 1,024 terabytes) of data available on the Internet were first captured and created by human beings - by typing, pressing a record button, taking a digital picture or scanning a bar code. Conventional diagrams of the Internet... leave out the most numerous and important routers of all - people. The problem is, people have limited time, attention and accuracy - all of which means they are not very good at capturing data about things in the real world. And that's a big deal. We're physical, and so is our environment... You can't eat bits, burn them to stay warm or put them in your gas tank. Ideas and information are important, but things matter much more. Yet today's information technology is so dependent on data originated by people that our computers know more about ideas than things. If

we had computers that knew everything there was to know about things - using data they gathered without any help from us - we would be able to track and count everything, and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best. The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so."[1]

Speaking about IoT it is possible to identify some key points, which allow the construction of an Internet-based network of objects:

- *Energy*: issues related with energy harvesting and low-power consumption are fundamental for the development of suitable chipsets for the IoT; these issues lead to research for new batteries and fuel cells.
- *Intelligence*: the concept of intelligence is related to context awareness that usually implies integration of memories and processing power, the ability to function in harsh environments, and affordable security.
- *Communication*: a multi-frequency band antenna should be integrated on-chip, optimized for size, cost, and efficiency; transmission speed and modulation schemes should be studied in order to achieve good transmission rates with low energy waste; communication protocols should be designed for Web oriented architectures where all the objects are combined to analyze events happening at various location throughout the network.
- *Integration*: devices should be integrated in every kind of package ranging from textiles and paper to human accessories and the device should be capable of operating in harsh environments while allowing cost-saving and eco-friendliness.
- *Interoperability*: probably the biggest challenge is to make the devices capable of understanding different standards and to enable communication between different kinds of networks.
- *Standards*: the necessity of a global standard directly follows from the necessity for interoperability.
- *Manufacturing*: obviously the cost of tags and antennas should be reduced as much as possible and production should occur on a large scale to minimize eco-impact and to decrease the dimensions and costs of the chips.

In contrast with these innovation trends there is a barrier imposed as privacy issues must be considered. It is still not clear which level of security should be guaranteed when object identification systems are involved.

2.2 Ultra-wideband

The first standard to be defined specifically for applications concerning IoT and wireless sensor networks was IEEE 802.15.4. This standard was first published in 2003 and revised in 2006. It specifies a Physical layer and the MAC layer for wireless networks with low power consumption requirements (usually referred to a for Low-Rate Wireless Personal Area Networks (LR-WPAN)). The systems

should be able to operate for multiple years on a battery and thus the communication protocols and the local computations should have very low complexity.

The communication via this standard was defined to work in an unlicensed, international frequency band at 2.4 GHz and to use a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) algorithm at the link level; ZigBee, WirelessHART and others have built upon this standardization. Recently ultra-wideband has been explored for the physical layer due to its advantages in positioning accuracy. A standard [7] was released some years later to properly treat communication using an ultra-wideband signal.

An ultra-wideband signal is defined by an absolute bandwidth of at least 500 MHz or otherwise by a fractional bandwidth of larger than 20% [2]. A fractional bandwidth is defined as the total bandwidth, divided by the centre frequency; the centre frequency is given by:

$$f_C = \frac{f_H + f_L}{2} \tag{2.1}$$

The two frequencies in the formula represent the upper frequency of the -10db emission point and the lower frequency of the -10db emission point.

The main characteristic of UWB systems, due to their large bandwidth, is the very short duration of waveforms, usually on the order of nanoseconds. Usually UWB systems transmit ultra-short pulses with a low duty cycle; this means that the ratio between the pulse transmission and the average time between two consecutive transmission is always very small; however, both low duty cycle schemes and continuous transmissions can be considered for UWB communications.

Having large bandwidth brings many advantages for positioning and communications such as penetration obstacles, high ranging (hence positioning) accuracy, and high-speed data communications, in addition to low power and low cost implementation. The good ranging accuracy leads to good positioning accuracy.

Spreading a signal over a large bandwidth also guarantees the possibility to build systems which interfere less with the existing systems as the spreading factor of low-rate data transmission can be very large (on the order of 1000). This allows not only transmissions over reasonable distances (10-50 m) but further improves the signal's robustness to narrowband interference originating from narrowband jammers or other UWB devices [3].

Thanks to its flexibility UWB can be used both in a *high data rate* scenario, where it provides wireless connectivity with a data rate ranging from 100 *Kbps* to 100 *Mbps*, between a host (typically a PC) and some other peripherals (mouse, keyboard, printer, etc.), (this is comparable with other short-range technologies such as Bluetooth), and *low data rate* scenario, making it suitable for wireless sensor networks. In the later scenario the data rate can range from a minimum of 50 *Kbps* up to 1 *Mbps* and the signal can reach distances of up to 100 m while offering position capabilities. UWB chips could cost on the order of US$ 1 for simple asset tracking and US$ 3-4 for communication and tracking nodes in an industrial environment [2].

2.2.1 Antennas

UWB antennas can be modeled as a front-end pulse shaping filter that affects baseband detection. A narrowband system does not have this problem due to the fact that front-end filters are typically designed as matched filters and because the fractional bandwidth is very small, hence the antenna frequency response can be designed to lie in the desired frequency domain. Unfortunately, for a UWB system pulse-waveform distortion is often present and the impulse response of the antenna can change with elevation and azimuth angle [2].

2.2.2 Propagation and channel model

Due to its really short pulses, it is highly inconvenient to treat an UWB signal directly within the frequency domain, hence it is necessary to envision it in the time domain and discard the common idea of a superposition of modulated sine waves when considering signal propagation via a medium.

Moreover, the huge transmission bandwidth enables super-resolution of the received signals at the receiver avoiding typical problems such as fading and enabling time-resolution of multiple paths. Ramirez-Mireles states in [4, 5] that compared to classical narrowband communications, in which the fading is of the order of 30-60 dB, the UWB level of nuisance is only 3-4 dB, thus fading is irrelevant, making time-resolvable multi-paths the dominant mechanism for energy transmission and capture. However, this leads to the necessity of treating carefully the pulse as carried by each of hundreds of individual paths, in contrast to several in a classic 5 MHz bandwidth system.

2.2.3 Modulations

In [6], Qiu, Liu, and Shen when choosing the modulation scheme take into consideration several aspects including data rate, transceiver complexity, spectral characteristics, robustness against narrowband interference, intersymbol interference, error performance, and so on. For pulsed UWB systems three main schemes were analyzed: pulse position modulation (PPM), pulse amplitude modulation, and on-off keying (OOK). These three modulation schemes are illustrated in Figure 2.1.

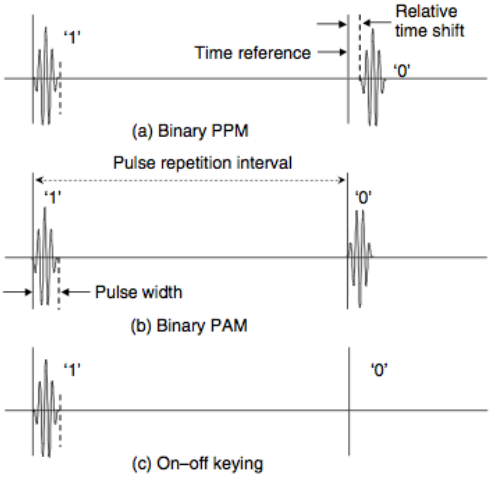


Figure 2.1: UWB modulation schemes

As can be seen from the picture, a '1' using PPM is represented by a pulse without any delay while a '0' is represented by a pulse shifted in time by a delay equal to τ, usually chosen to make the shifted version orthogonal to the original one. In contrast when using a PAM modulation the difference between a '1' and a '0' is perceived by studying the polarization of the pulse (positive for a '1' and negative for a '0'). Finally, when exploiting OOK, a '1' is represented by the presence of a pulse and a '0' by its absence.

2.3 IEEE 802.15.4a

This section presents a brief description of the definitions given by the IEEE 802.15.4a amendment.

2.3.1 PHY specification

In 2007 the Institute of Electrical and Electronics Engineers Standards Association (IEEE-SA) published an amendment to the original standard IEEE 802.15.4 specifying new physical layers (PHYs) to be added to the original standard.

This addition includes the standardization for two PHYs using UWB with designated frequencies in three different ranges: one below 1 GHz (from 250 to 750 MHz), one between 3 and 5 GHz (from 3244 to 4742 MHz), and the last one above 5 GHz (from 5944 to 10234 MHz) [7]. Moreover, three other PHYs were defined: one using Direct-Sequence Spread Spectrum and two using Chirp Spread Spectrum. The UWB PHY supports ranging capabilities, which are defined in the standard and are studied in the next chapter. These frequency bands and data rates of these alternative PHYs are summarized in Table 2.1.

Table 2.1: Frequency bands and data rates

PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbols rate (ksymbols/s)	Symbols
2450 DSSS	2400-2483.5	2000	O-QPSK	250	62.5	16-ary orthogonal
UWB sub-gigahertz	250-750	Variable	Variable	Variable	Variable	Variable
2450 CSS	2400-2483.5			250	166.667	
				1000	166.667	
UWB low band	3244-4742	Variable	Variable	Variable	Variable	Variable
UWB high band	5944-10234	Variable	Variable	Variable	Variable	Variable

Over these three main bands 16 different channels have been defined by the standard, as shown in Table 2.2. We assume that a UWB device should operate in one of the three frequency band and for each of these frequency bands at least one mandatory channel must be supported.

Table 2.2: UWB PHY channel frequencies

Channel number	**Center frequency (MHz)**	**UWB/band mandatory**
0	499.2	Sub-gigahertz
1	3494.4	Low band
2	3993.6	Low band
3	4492.8	Low band mandatory
4	3993.6	Low band
5	6489.6	High band
6	6988.8	High band
7	6489.6	High band
8	7488.0	High band
9	7987.2	High band mandatory
10	8486.4	High band
11	7987.2	High band
12	8985.6	High band
13	9484.8	High band
14	9984.0	High band
15	9484.8	High band

The IEEE 802.15.4a standard is supposed to furnish two main services: data service and management service. These two services are analyzed in the following sections, together with the Physical Protocol Data Unit (PPDU) definition and the symbol structure.

2.3.1.1 PHY data service

The data service takes care of the transmission of data at the physical layer and consists of two-way communication with ranging capabilities. The physical parameters to be used are first negotiated and at the end checked to achieve coherence.

A communication starts with a **PD-DATA.request** which asks for specific communication settings based upon four parameters:

- *UWBPRF* to choose the pulse repetition value of the transmitted PPDU among: NOMINAL_4_M, NOMINAL_16_M, and NOMINAL_64_M for UWB PHYs;
- *Ranging* to choose whether to use ranging exploiting both the ranging bit in the header and counter operation (ALL_RANGING) or only the ranging bit in the header (PHY_HEADER_ONLY);
- *UWBPreambleSymbolRepetitions* to set the preamble symbol repetitions to one of 16, 64, 1024 or 4096;
- *DataRate* to define the data rate to use for the communications based upon a numerical value ranging from 1 to 4 (DATA_RATE_1 corresponds to 250 *kb/s* and DATA_RATE_n corresponds to n*250 *kb/s*).

The communication is followed with a **PD-DATA.confirm** which allows the antenna to actually transmit and starts the ranging operations making use of seven parameters:

- *Status* to give feedback about the previous request;
- *RangingReceived* can be TRUE or FALSE and indicates if ranging is supported;
- *RangingCounterStart* is a 4-octet count to represent the time value at the beginning of a ranging exchange (set to all 0s if ranging is not supported);
- *RangingCounterStop* is a 4-octet count to represent the time value at the end of a ranging exchange (set to all 0s if ranging is not supported);
- *RangingTrackingInterval* is a 4-octet count of the time units in a message exchange over which the tracking offset was measured;
- *RangingOffset* is a 3-octet count of the time units slipped or advanced by the radio tracking system over the course of the entire tracking interval;
- *RangingFOM* 1-octet FoM to characterize the ranging measurements.

Following the confirmation a third packet called **PD-DATA.indication** is sent repeating some important parameters regarding the communication and the ranging operations, specifically: *UWBPRF*, *UWBPreambleSymbolRepetitions*, *DataRate*, *RangingReceived*, *RangingCounterStart*, *RangingCounterStop*, *RangingTrackingInterval*, *RangingOffset*, and*RangingFOM*.

2.3.1.2 PHY management service

This physical management service (PLME) defines two main primitives which are used to set the energy level (**PLME-ED**) and the state of the transceiver (**PLME-SET-TRX-STATE**). PLME-SET-TRX-STATE is one of RX_ON, TRX_OFF, FORCE_TRX_OFF, TX_ON, and RX_WITH_RANGING_ON. Additionally, for UWB systems another three primitives are defined:

- **PLME-DPS** attempts to set the Default Parameters Set (DPS) parameters;
- **PLME-SOUNDING** requests the PHY to respond with channel sounding information;
- **PLME-CALIBRATE** requests the PHY to respond with RMARKER offset information.

2.3.1.3 Frame format

The typical format of a UWB frame is shown in Figure 2.2. The Synchronization Header (SHR), which is actually the preamble of the packet, is added to aid the receiver algorithms related to AGC setting, antenna diversity selection, timing acquisition, coarse and fine frequency recovery, packet and frame synchronization, channel estimation, and leading edge signal tracking for ranging.

There could be four types of preambles (default, short, medium, and long) and which one must be used in the ongoing communication is specified in the *UWBPreambleSymbolsRepetitions* field in the PD-DATA.request primitive.

		Bits	**Octets**
		19	**Variable**
Preamble	**SFD**	**PHR**	**PSDU**
SHR		PHR	PHR payload

Figure 2.2: Format of the UWB PPDU

The preamble is subdivided in two different section: the SYNC section provides packet synchronization, channel estimation, and ranging sequence data and the SFD field (frame delimiter sequence). The length of these two sections is variable. The SYNC ranges from 16 to 4096 symbols, while the SFD can be between 8 and 64 symbols.

The two mandatory rates for the SHR are coming from the peak pulse repetition frequencies (PRFs) result for a preamble code length of 31 which always must be supported, hence the rates are therefore 1.01 *Msymbol/s* and 0.25 *Msymbol/s*. The Physical Layer Header (PHR) is sent at a nominal rate of 850 kb/s for all data rates above 850 kb/s and at a nominal of 110 kb/s for the nominal data rate of 110 kb/s. The Physical Service Data Unit (PSDU) is sent at the desired data rate according to the standard.

Once the PSDU is created (with a length ranging from 0 to 127 octets) the encoding process starts: first Reed-Solomon procedures are performed over the PSDU, then the 13 bits PHR is added, to which another 6 check bits are added; convolutional codes are then used over the PHY header and spreading is performed at one of the two possible nominal rates; finally the SHR is inserted.

2.3.1.4 Symbol structure

A UWB PHY symbol is able to carry two different types of information: the first one is used to locate the position of the burst of pulses while the second one is used to determine the polarity of the burst. The structure of this symbol is illustrated in figure 2.3.

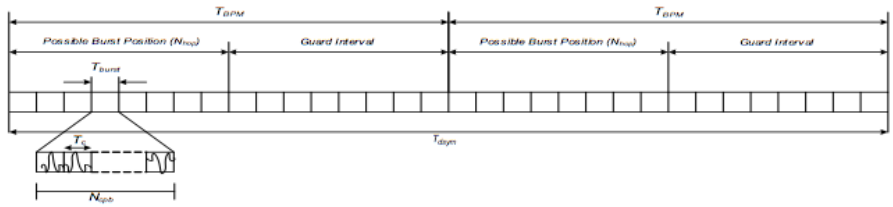


Figure 2.3: UWB PHY symbol structure [7]

The location of the burst in either the first half or second half of the symbol's slot indicates one bit of information. Additionally, the phase of the burst (either –1 or +1) is used to indicate a second bit of information.

2.3.2 MAC specification

In [7] the mechanism for channel access to be used is said to be ALOHA for UWB technology, but it is good to notice that this is not the only possibility. Ding et al. [8] compare different packet generation rate CSMA/CA and TDMA protocols' performances.

The standard describes two main services: the MAC data service and the MAC management service.

The data service follows the same process adopted for the physical service and again the parameters to be included in the different primitives are the same, although some more options are available which are dedicated to the status of the last MAC Service Data Unit (MSDU) transmitted or received.

The MAC management service supports the DPS, SOUNDING and CALIBRATE primitives and adds new features such as **MLME-BEACON-NOTIFY**, **MLME-RX-ENABLE**, and **MLME-SCAN**.

The frame description is not specifically treated in the IEEE 802.15.4a amendment but it is addressed in the original IEEE 802.15.4 standard [9], last revised in 2011, and the structure is shown in figure 2.4, along with the number of octets for each field.

2	1	4 to 20	n	2
FCF	Sequence number	Addressing fields	Data payload	FCS

Figure 2.4: Format of the UWB MPDU

The Frame Control Field (FCF) is used to distinguish the frame type, the addressing mode, and other control parameters. The sequence number is utilized

to match acknowledgement frames to data. The addressing field supports short addresses of 16 bits and long addresses of 64 bits and includes both source and destination addresses. The data payload is followed by the Frame Check Sequence (FCS) used for integrity verification.

The MAC layer and its packet structure is analyzed in detail in the rest of this thesis.

2.4 Time synchronization

In [10] Karl and Willig list all the possible applications of wireless sensor networks and observe that for measurement applications, event detection, and tracking one of the most important issues is the precision of localization of an asset or device.

The first consideration to take into account when speaking localization is timing and specifically about synchronization. Despite the necessity of knowing when an event occurs with high precision, time synchronization also determines the time resolution during the computation of the final position of a tag or a sensor moving in the network. The distance between sensors is a function of the time spent exchanging packets or readings between the devices involved and of course of the signal propagating over the network media. If the clocks of the receivers are not very accurate, then a seemingly small error can lead to significantly biased estimates of the location.

There are two principal ways of time synchronization: one is using dedicated time synchronization and the other is to combine the readings of multiple receivers in order to average the estimation error.

The clock structure is usually the same for all nodes [11] and it consists of an oscillator running at a specified frequency and a counter (i.e., a register) which is incremented in hardware after a certain number of oscillator pulses. The software implemented in the node only has access to this register, thus the time resolution is limited by the difference between two increments of this register. Therefore events happening between two increments can not be distinguished based upon the value of their timestamps.

As stated in [10], the software clock of node i at real time t can be expressed as function of the hardware clock by applying a simple transformation:

$$L_i(t) = \theta_i \cdot H_i(t) + \phi_i \tag{2.2}$$

where ϕ_i is called **phase shift** and θ_i is the **drift rate**, both involve the time resolution of the clock. From this definition some simple problems can be described:

- Devices are usually turned on at different times causing random differences in the their clocks' phase shift.
- Oscillators often have *a priori* a slight random deviation from their nominal frequency, called **clock drift** and sometimes **clock skew**. The clock drift can be due to impure crystals, but also due to several environmental conditions such as pressure, temperature, and so on. The clock drift is often expressed in parts per million (ppm) and gives the number of additional or missing oscillations a clock makes in the amount of time needed

for one million oscillations at the nominal rate. Clock skew describes the variation in the arrival time of the clock pulse through the circuitry and unfortunately can not be avoided, but it can be minimized.

- The oscillator frequency varies both on a short-term (due to temperature variations, air pressure, and so on) and a long-term (due to aging). This implies the necessity to repeat the synchronization algorithm at some minimum intervals in order to track these frequency changes.

When designing a synchronization algorithm it is really important to remember some performance metrics, such as energy costs and memory requirements.

Two main parameters are usually studied when describing time synchronization algorithms: **offset** between clocks and **drift** in the frequency of the clocks.

The literature regarding time synchronization algorithms for wireless sensor networks is extensive, however it is still not clear at which layer synchronization should be performed. Among all the protocols some of them are more widely used and will be introduced in this section.

2.4.1 RBS

In [12] Elson, Girod, and Estrin defined Reference-Broadcast Synchronization (RBS) as a receiver to receiver synchronization algorithm. A third party broadcasts a beacon without any timing information to all the receivers, and then the nodes exchange information regarding the time when they received the beacon in order to compare their different phase offsets.

The simplest version of RBS is composed of one broadcasted beacon and two receivers. When the algorithm is extended to a larger scale network with more receivers it is necessary to send more than one broadcast beacon. The receivers compare information with all their neighbors and average all the offset computations performed after each beacon; this increases the precision of the synchronization.

RBS differs from other algorithms because it removes the sender from the critical path. By removing the sender, the only uncertainty is the propagation and receive time. Figure 2.5 illustrates this concept.

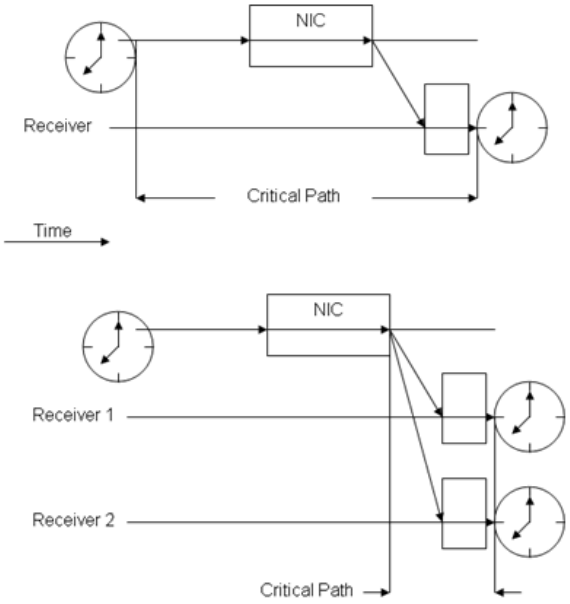


Figure 2.5: Comparison of a traditional synchronization system with RBS

Regarding the drift estimation Elson, Girod, and Estrin decided to exploit a simple least-square linear regression which offers a fast method for finding the best fit line through the phase error observations over time. The frequency and phase of the local nodes with respect to the remote node can then be recovered from the slope and the intercept of the line.

2.4.2 TPSN

Timing-sync Protocol for Sensor Networks (TPSN) [13] is a traditional sender to receiver based synchronization algorithm which builds a tree to organize synchronization between the nodes. The algorithm is divided into two main parts, the *level discovery phase* which creates the tree assigning a hierarchical level to each node in the tree and assigning level 0 to only one node in the topology (the root) and a *synchronization phase* when all the nodes start synchronizing with the nodes at level *i-1* until all the nodes are finally synchronized with the root.

2.4.2.1 Level Discovery Phase

During the level discovery phase the root node is assigned as level zero. Then this root node starts sending *level_discovery* packets to all its neighbors which assign themselves to level 1 and repeat the procedure by sending packets to all their neighbors; this process is repeated until all the nodes in the network are reached by the *level_discovery* packet.

2.4.2.2 Synchronization Phase

The synchronization concept underlying this algorithm is a two-way communication and as with the level discovery phase it begins at the root node and

propagates through all the network.

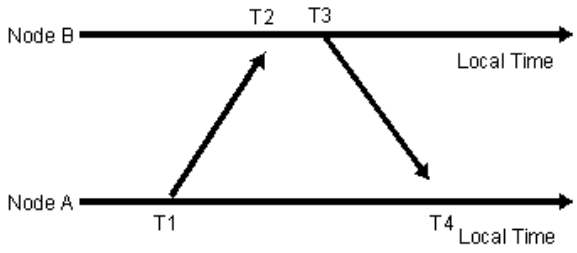


Figure 2.6: Two-way communication between nodes

Figure 2.6 shows the two-way messaging between a pair of nodes. Node A, which at the beginning is the root, sends a *synchronization_pulse* at time T1 to Node B; this packet also contains the level information about Node A. Node B receives the pulse at time T2 and sends back an *acknowledgement_packet* to Node A at time T3; this packet contains the level information about Node B and the three timestamps T1, T2, and T3. If Δ is the propagation delay and d the clock offset, then Node A is easily able to compute these two parameters and adjust its clock to match with the clock of node B through the formulas:

$$\Delta = \frac{(T2 - T1) - (T4 - T3)}{2} \tag{2.3}$$

$$d = \frac{(T2 - T1) + (T4 - T3)}{2} \tag{2.4}$$

The phase is initiated when the root sends a *time_sync* packet; the neighbors reached by this packet wait a while to start the two-way messaging and complete the adjustment of their clock with respect to the root node clock. At the same time, deeper level nodes in the network are able to hear at least the communication one level up, so they wait a random amount of time to start their two-way messaging with the upper level nodes. This process is performed until all nodes in the network are synchronized with the root node.

The main advantage, apart from being more precise due to the utilization of four different timestamps, is the possibility of working with multi-hop networks which eliminates the limit on ranging imposed by RBS.

2.4.3 FTSP

In [14] Maróti, Kusy, Simon, and Lédeczi propose another sender to receiver synchronization algorithm which claims to improve on the disadvantages of TPSN.

The structure of Flooding Time Synchronization Protocol (FTSP) is similar to TPSN in the way it synchronizes all the nodes with a root node. The root node transmits synchronization information within a single radio message to all the receivers. The receivers take note of the receive time of this packet. Having both the transmitting and receiving time, they are already able to compute the offset. In order to minimize high precision clock drift estimation is needed and FTSP utilizes linear regression.

The root is dynamically and periodically elected which allows topology changes and constitutes the main advantage over TPSN in terms of robustness and link failure.

As mentioned before, the literature regarding time synchronization protocols for wireless sensor network is quite extensive even though it is not always feasible to find appropriate algorithms for exploiting UWB technology.

In [15] a combination of receiver to receiver and sender to receiver algorithms is proposed as an implementation of two subprotocols, Hierarchy Referencing Time Synchronization (HRTS) and Individual-based Time Request (ITR).

Sichitiu and Veerarittiphan propose in [16] two sender to receiver protocols called TinySync and MiniSync.

The MultiHop Time Synchronization Protocol (MTSP) [17] uses a pair-wise synchronization protocol similar to Lightweight Time Synchronization (LTS) [18] and TPSN.

The synchronization problem will be reconsidered further in this thesis and an analysis of other possibilities is left to the reader as this general introduction to the issue is far from exhaustive for the case study under discussion. However, the examples of the protocols above should give the reader an idea of single-way or two-way message exchange and sender to receiver or receiver to receiver implementation, concepts which are going to be utilized again in the design of the network targeted in this paper.

2.5 Ranging, localization, and positioning

Once time synchronization is performed, communication between nodes can be used to extract information about their geometric relationship. Hence, the distance between two nodes or the angle between nodes can be estimated. This information can be used to compute the position of a device in the network. When distances are used the approach is called **lateration**, whilst when angle measurements are exploited it is called **angulation**. The simplest case of lateration gathers measurements from three different anchors (receivers at know positions), but sometimes in order to achieve higher precision, information from more than three anchors is collected; in this case we talk about **multilateration**.

The first step that must be taken to localize a device is obtaining distances measurements; this process is called ranging and exploits the possibility of measuring some radio characteristics at the receiver which can serve as an estimator.

The simplest characteristic that can be used is the Received Signal Strength Indicator (RSSI). The main disadvantage in measuring this characteristic is related with the fact that RSSI measurements are usually not constant but can oscillate quite much even when the sender and the receiver are not moving. For example, ranging errors of $\pm 50\%$ have been reported by Savarese [19]. Ramadurai and Sichitiu [20] have exploited the stochastic mapping RSSI values to distances.

The other most widely utilized approaches to achieving precise ranging are exploiting Time-of-Arrival (**ToA**), Time-Difference-of-Arrival (**TDoA**), and Angle-of-Arrival (**AoA**). These are each described in the following subsections.

2.5.1 ToA

Time-of-Arrival, sometimes also referred as time of flight, is the technique of distance measurement based upon knowing the time elapsing since transmission and reception of a signal and the propagation velocity of the signal used in the medium. If both the time of departure and the time of arrival can be timestamped, then the receiver is able to compute the distance between two anchors. We can relieve the receiver of this duty by sending an acknowledgement back and the transmitter can compute the round trip delay (assuming symmetric paths).

This technique is best in terms of position accuracy when measurements are gather from multiple anchors within the network. As shown in Figure 2.7, assuming it is desired to locate a device whose signal of three different antennas, then the distances collected by the three receivers allows to draw three different circles whose intersection determines the actual position of the transmitting device.

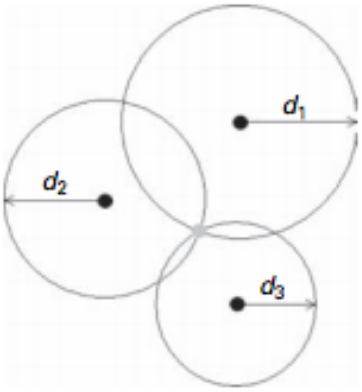


Figure 2.7: ToA trilateration technique [21]

Let d_1, d_2, and d_3, the three ranging measurements obtained from TOA as shown in the picture are the two coordinates x and y of the target node. These can be easily calculated by solving the three following equations:

$$d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}, i = 1, 2, 3 \tag{2.5}$$

However, depending on the medium in use, time of arrival requires really high resolution clocks in order to achieve high accuracy in positioning objects in the network. Some results are presented by Luo and Ge [71].

2.5.2 TDoA

With Time-Difference-of-Arrival, the difference between the arrival times of two signals traveling between the target node(s) and two different reference nodes is estimated. This locates the target node on an hyperbola, with foci at the two reference nodes. One simple way of obtaining TDOA estimation is to estimate the two different TOA at the two reference nodes and then compute the difference.

TDOA measurements can be achieved in absence of synchronization between the target node and the reference nodes guaranteeing that synchronization only needs to be performed among the different reference nodes. Another way is to perform cross-correlations of the received signals and to calculate the delay corresponding to the largest cross-correlation value.

In this case, when exploiting trilateration, as shown in Figure 2.8, each TDOA measurement determines an hyperbola. With three reference nodes two range differences can be obtained hence two different hyperbolas.

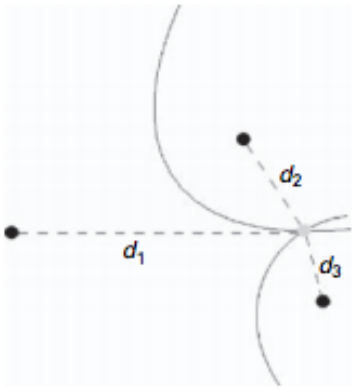


Figure 2.8: TDoA trilateration technique [21]

The two ranges differences can be expressed as follow:

$$d_i - d_1 = \sqrt{(x - x_i)^2 + (y - y_i)^2} - \sqrt{(x - x_1)^2 + (y - y_1)^2}, i = 2, 3 \qquad (2.6)$$

and the position can be computed from these two and the relation regarding the d_1 similar to the ones analyzed for ToA, but with parameter order inverted.

2.5.3 AoA

Unlike the other two previous models, AoA does not provide range information between two nodes, but instead exploits information about the direction of an incoming signal, thus, determining the angle between two nodes. This information can be obtained at an antenna array by measuring the differences in arrival times of the incoming signal at different antenna elements.

With these measurements two reference nodes are sufficient to determine the position of the target node by the intersection of two lines. The procedure is illustrated in Figure 2.9.

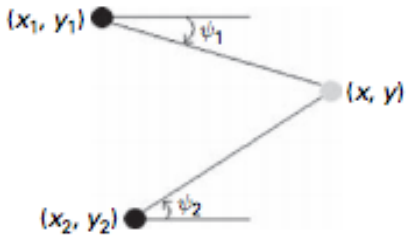


Figure 2.9: AoA triangulation technique [21]

Denoting with $\psi 1$ and $\psi 2$ the two angles as shown in the picture above it is easy to find the coordinates of the target node by simply solving the following two formulas:

$$\tan\psi_1 = \frac{y - y_1}{x - x_1} \qquad \tan\psi_2 = \frac{y - y_2}{x - x_2} \tag{2.7}$$

A comparison of these three techniques in terms of simulation for a UWB positioning system is given by Canovic in [73]. It is important to notice that these techniques could be exploited in a hybrid form using measurements from two different models (for example ToA/AoA or TDoA/AoA) with the goal of meeting particular accuracy constraints. An example of the first model for non-line-of-sight environments is given in [22], while a combination of TDoA and AoA has been studied for wideband CDMA cellular systems in [23], and in terms of performances in [24], as well as for indoor UWB Systems in [44].

2.6 6LoWPAN

IPv6 over Low-power Wireless Personal Area Network (6LoWPAN) is a network layer which allows the transport of IPv6 packets over IEEE 802.15.4 frames. An overview of the standard is given in RFC 4919 [25], while its definition is provided in RFC 4944 [26].

IPv6 defines certain requirements for the link-layer over which IPv6 packets are transported [27], however the IEEE 802.15.4 MAC layer does not fulfill these requirements in all the aspects. For this reason, 6LoWPAN defines not only a frame format, but also how to get a unique IPv6 address from either a 16-bit or 64-bit IEEE 802.15.4 MAC addresses and how to overcome the limitations of IEEE 802.15.4 frames [9].

As stated in RFC 2460 the minimum size for a MAC layer packet in order to transport IPv6 packets is 1280 octets, which is far away from the IEEE 802.15.4 maximum frame size limit of 127 octets. Moreover, of these 127 octets, a maximum of 25 are used for the header and if security is enabled another 21 additional octets are taken. This leaves only 81 octets free for the transportation of IPv6 packets. Since the IPv6 header is 40 bytes, only 41 bytes would be available for the transport layer. To meet these limitations, 6LoWPAN defines a fragmentation and reassembling mechanism that allows IPv6 packets to be split into smaller frames, making them suitable for transmission by the link-layer. This process is completely transparent to the Internet layer, but 6LoWPAN

applications should utilize small-sized packets in order to avoid as much fragmentation as possible, therefore 6LoWPAN defines a procedure to compress the IPv6 header.

2.6.1 6LoWPAN format

One problem due to the IEEE 802.15.4 format is that it does not include any identifier indicating what payload a data frame carries. There is no multiplexing information, hence it is impossible for the receiver to distinguish a 6LoWPAN packet from any other packet or even different kinds of 6LoWPAN packets. For this reason, 6LoWPAN defines a type identifier; 6LowPAN does not create a proprietary identifier but uses the first byte of the payload as a *dispatch* byte to give information about the type of payload and also further knowledge about the subtype. The two most significant bits of this octet are used to divide the types into 4 different categories, one of them (corresponding to 00) reserved for application using IEEE 802.15.4 other than 6LoWPAN. The four different categories are shown in Table 2.3.

Table 2.3: Two most significant bit meanings in the dispatch

00	Not a LoWPAN packet (NALP)
01	Normal dispatch
10	Mesh header
11	Fragmentation header

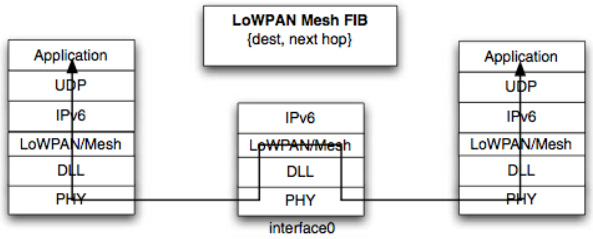
As 6LoWPAN is an adaptation layer working between IP and MAC, when talking about addressing, two kind of addresses are involved: link-layer and IP addresses. The link-layer on which it is supposed to work supports two kind of addresses (64-bit addresses and 16-bit short addresses), thus the complexity of 6LoWPAN is a little bit higher. In order to encourage direct compatibility with the Ethernet, the 64-bit MAC address is most used, thus the easiest way of constructing an IPv6 address is to combine a 64-bit prefix with a IEEE 802.15.4 address. If a short address must be used, a Personal Area Network (PAN) identifier is added to the field.

A direct consequence of the IEEE 802.15.4 frame addresses is the possibility for them to traverse multiple hops within a 6LoWPAN network, in which two main processes are involved: *forwarding* and *routing*.Frowarding can be performed at layer 2 or at layer 3.

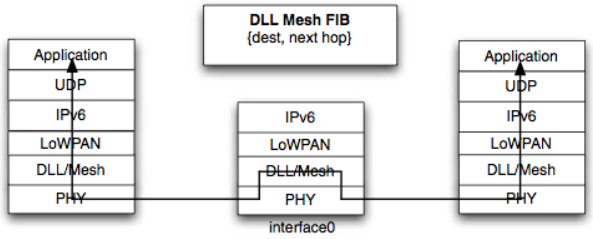
Usually in a 6LoWPAN network, forwarding is done by exploiting layer 2 and it is common to talk about a *Mesh-Under* model. In practice the model adds a mesh header to the 6LoWPAN format in which a source address, a destination address, and all the information about the middle hops are stored. In case routing is done at link layer, hence 6LoWPAN is completely transparent to this process and the mechanism is provided by another sublayer.

When forwarding is performed at layer 3 it is normal to talk about a *Route-Over* mechanism. In this case, no support from the adaptation layer is needed and fragmentation is performed at each hop along the path.

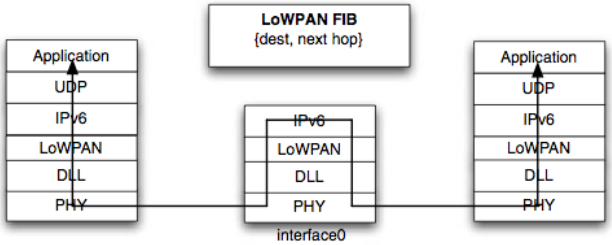
A visual overview of the three different mechanisms is illustrated in Figure 2.10.



(a) Mesh-Under adaptation layer



(b) Mesh-Under layer 2



(c) Route-Over layer 3

Figure 2.10: 6LoWPAN forwarding and routing [35]

2.6.1.1 Header compression

It has been stated previously that the maximum payload size provided by IEEE 802.15.4 is really small, half of which, in case of IPv6 usage, is already used by the IPv6 header. Even if fragmentation can be performed it is usually feasible and more convenient to fit all the IPv6 packet into an IEEE 802.15.4 packet due to the extras power consumption implied when implementing fragmentation.

Data compression could be applied but it is usually not helpful when forwarding small data items with the risk of losing important information necessary to decode subsequent packets.

For this reason, header compression is adopted. Again, this mechanism could be implemented in an end-to-end style, but is then limited to compressing the headers that are within the payload of the IP header, knowing that routers in the middle still need to see the entire IP address. Considering the biggest part of the header is occupied by the IP header, the best way is to utilize a hop-by-hop method allowing the packet to be forwarded on different links and to demand

agreement about compression directly neighbor to neighbor.

Since the 1990's header compression has been a focus of the IETF and many algorithms has been proposed ranging form the second generation *IP Header Compression* (IPHC) [28] and its extension to RTP [29, 30, 31] to the third generation *Robust Header Compression* (ROHC) [32].

ROHC employes a flow-based compression which is prone to loss of synchronization when packets of the same flow are lost leading to the necessity to add some acknowledgments techniques to the algorithm. Doing so, the protocol becomes really complex and not suitable for applications with constraints on the resources available.

6LoWPAN defines two models for header compression, one *stateless*, avoiding the synchronization problem and one *context-based*, which meets the requirement for globally routable IPv6 addresses.

2.6.1.2 Fragmentation

When an entire IPv6 datagram does not fit into an IEEE 802.15.4 MAC frame the packet is split into fragments each of which is incapsulated in a 6LoWPAN packet by adding a fragmentation header in order to recognize the IPv6 datagram they belong to. This header is composed of a packet identifier, an offset, and the total length information from the IPv6 original packet. The reason that fragmentation should be avoided is clear from the fact that this procedure adds 4 octet overhead to the first fragment and 5 octets to the second and all the subsequent fragments.

2.6.2 6LoWPAN Neighbor Discovery

The Neighbor Discovery protocol, first defined for IPv6 in RFC 4861 [33] and then optimized for low power, lossy networks [34] provides mechanisms to accomplish the following tasks: Router Discovery, Prefix Discovery, Parameter Discovery, Address Autoconfiguration, Address resolution, Next-hop determination, Neighbor Unreachability Detection, Duplicate Address Detection, and Redirect.

The original ICMPv6 messages defined within RFC 4861 are expanded for the lossy networks case with three new cases and two additional messages; and Redirect messages are not used with route-over topologies for this particular case. All of these messages are briefly described below:

- **RS** Router Solicitation prompts routers to generate RA;
- **RA** Router Advertisement disseminates prefix, context and link parameters;
- **NS** Neighbor Solicitation is in charge of address resolution and neighbor unreachability detection;
- **NA** Neighbor Advertisement is the response to NS;
- **ARO** Address Registration Option included in the NS to keep track of reachable neighbors;
- **6CO** 6LoWPAN Context Information Option is optional and similar to the PIO of RFC 4861 to gather prefix information;

- **ABRO** Authoritative Border Router Option is optional and necessary in case of route-over topology, to avoid the usage of most-recent prefix information if others are received from external routers;
- **DAR** Duplicate Address Request performs multihop duplicate address detection;
- **DAC** Duplicate Address Confirmation is the response to DAR.

When bootstrapping, hosts send a RS to discover routers; routers answer with an RA including a Prefix Information Option (PIO) and additionally 6CO and ABRO. If IEEE's 64-bit addresses are used, then no duplicate detection is needed otherwise the duplicate address detection procedure is performed exploiting DAR and DAC. The address registration feature is achieved by the NS and NA messages carrying the new option ARO (which contains the address of the sending interface and a lifetime value). A router that receives such a NS checks if the value is already in the cache and if not, it registers it in a new entry and keeps it until the lifetime value expires.

The optimizations obtained by this new version of the protocol are mainly related with the support of procedures like header compression and fragmentation even though the best achievement to be underlined is the host-initiated interaction which allows hosts to be sleeping, quite suitable capability for power-constrained networks.

2.6.3 6LoWPAN mobility and routing

This section is principally dedicated to explaining the routing algorithm for 6LoWPAN but also illustrates the possibility for devices to be mobile within large-scale networks. For this two concepts related with mobility are introduced:

- **Roaming** is the process in which a node moves from one network to another;
- **Handover** is the process in which traffic to and from a node is redirected so that packets can continue to be sent/received despite a change in link and possibly also a change in network layer address.

Reasons behind the necessity of performing roaming or handover procedures are in physical movement, radio channel impairment, network performance, sleep schedule, and node failure. The handover process requires a series of operations including establishing a new link, configuring a new IPv6 address, etc. IEEE 802.15.4 has a *node-controlled* approach for changing network (similar to Wi-Fi), rather than *network-controlled* (as in GSM systems). Two important algorithms are: Mobile IPv6 [36] and NEMO (NEtwork MObility) [37].

Routing and forwarding were introduced previously to give an idea of the possibilities; this section concentrates on routing performed at a network level using 6LoWPAN. For this reason it is important to keep in mind the main characteristics of 6LoWPAN networks: they forward packets on a single wireless interface, nodes have the same IPv6 address prefix, and packets are not meant to transit between two different subnets, thus the necessity of 6LoWPAN routers.

There are two main classes of routing algorithms used for 6LoWPAN networks.

- **Distance-vector**: in this approach each link is assigned a cost and when sending a packet from A to B the link with the lowest cost is chosen. A node keep soft-entries of the destinations needed together with their path cost. Routing information is updated either *proactively* (*a priori*) or *reactively* (on-demand). Due to their simplicity and their low signaling overhead, these algorithms are usually employed with 6LoWPAN.
- **Link-state**: in this model each node floods the network with its information about reachable destinations; after this flooding, each node creates a graph with information abut the structure of the entire network. Due to the high overhead and to large size of routing tables, link-state algorithms are not very suitable for 6LoWPAN networks and they are often applied offline at edge routers.

As mentioned in the previous paragraph, routing updates can be performed in two different ways within the distance-vector scenario.

- **Proactive routing**: nodes build up routing information before routes are needed, foreseeing possible traffic to the most likely destinations.
- **Reactive routing**: routes are discovered only when they are needed and there is little information stored during autoconfiguration phase.

Similar to the case of synchronization algorithms, the literature concerning routing protocols for WPANs is extensive but this thesis will focus on one particular group of the algorithms, specifically those designed by the IETF MANET group (formed in 1997). For reasons of completeness Routing Over Low power and Lossy networks (ROLL) is mentioned, even though its definition is still under development by the ROLL working group [40]. This algorithm will be considered when talking about Contiki, addressing it as IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), as stated by the working group.

2.6.3.1 AODV

The Ad-Hoc On-Demand Distance Vector (AODV) protocol [38] enables a node to establish and maintain routes in mobile multi-hop networks even with dynamic topologies. It creates routes only when needed and maintains entries for them only if there is active communication. AODV includes methods for local path repair, and includes a destination sequence number to ensure loop-free operation.

AODV is characterized by the use of three main messages: RREQ is broadcasted to the network in order to ask for a path to a particular destination and is responded to with a RREP. The RERR message is used to communicate that a link is broken.

AODV was the first reactive algorithm to be implemented and it has been used as a basis for many other routing protocols, such as the one implemented by ZigBee.

2.6.3.2 DYMO

Dynamic MANET On-Demand routing (DYMO) [39], also referred as AODVv2, is an improvement to AODV offering convergence in dynamic topologies, sup-

port for a wide range of traffic flows, and consideration for Internet interconnectivity.

Figure 2.11 shows a simple routing procedure for AODV algorithms valid for both AODVv1 and DYMO.

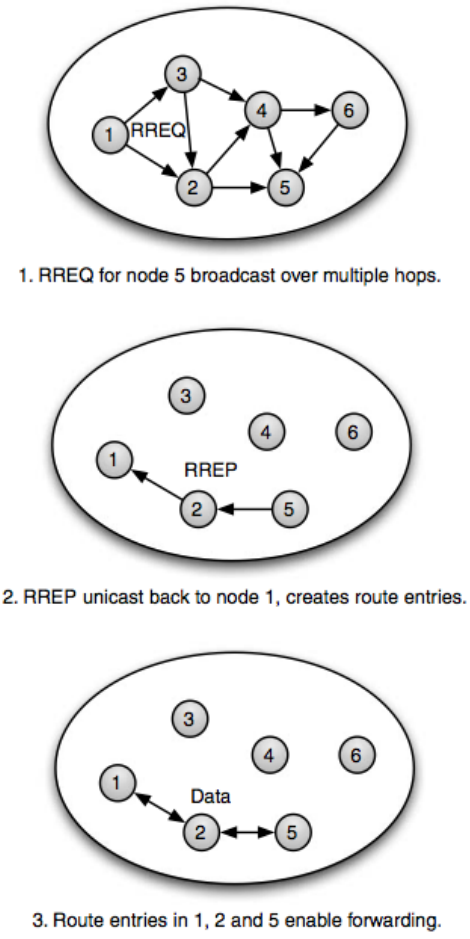


Figure 2.11: AODV example [35]

2.6.3.3 OLSR

The Optimized Link State Routing (OLSR) protocol [40] is a proactive link-state algorithm that builds routing tables by exchanging topology information among routers. In order to limit the flooding of traffic selected multipoint relay nodes are used as intermediate routers, thus enabling a cluster structure.

OLSR is most suitable for stable networks and therefore is not very widely exploited with 6LoWPAN, keeping in mind that WPANs usually employ a tree topology, which is not supported by a link-state routing model.

2.7 What have others already done?

This section describes two other popular systems which are implemented in wireless sensor networks and which are going to be referred to during the remainder of this thesis. Some background material regarding the MAC layer and network layer is presented as this material will be essential to understand the work that has been done.

2.7.1 ZigBee

ZigBee [65] is a well established set of specifications [41] for WPANs. It was designed specifically for low power consumption devices to guarantee long battery life.

2.7.1.1 ZigBee overview

ZigBee exploits the IEEE 802.15.4 standard for its lowest layers (PHY and MAC) and defines a protocol for the next upper layers of the Zigbee protocol stack, specifically network and application layers. The network layer permits the growth of networks to include a huge number of nodes and is assisted by the Application Support layer (APS). APS constitutes a connection between the network layer and the application layer. The application layer is subdivided into the ZigBee Device Object (ZDO) which assigns roles to devices and a user-defined application profile (plus security services). The overall Zigbee architecture is shown in Figure 2.12.

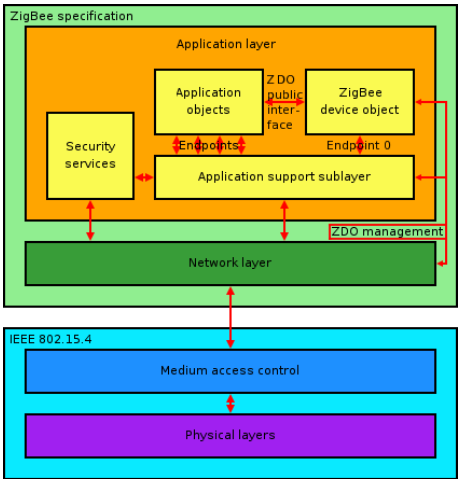


Figure 2.12: ZigBee protocol stack

The ZigBee specification defines three different kind of devices working within these layers in the network, working with 64-bit addresses with an option to use 16-bit short addresses to reduce packet size. These three different kinds of devices are:

- **ZigBee Coordinator Node** (ZCN) is unique in a network and it works

as a router to enable packets to reach other networks. It also stores information about the whole network.

- **Full Function Device** (FFD) is an intermediary router which work as a coordinator, but it stores less information, hence it has lower manufacturing costs.

- **Reduced Function Device** (RFD) can not relay data from other devices and is only able to communicate with a coordinator or FFD. A RFD needs the smallest amount of memory of memory, thus it is cheapest type of Zigbee node.

The APS is in charge of maintaining tables to enable matching between devices, discovery, and communication among devices. APS also determines the nature of a device (ZCN, FFD, or RFD) and commences and replies to requests to create a network.

Zigbee is defined to be able to work in three license-free frequency bands (however, a given device might only work in one of these bands). Zigbee was designed for low power consumption operations enabling a battery life lasting from months to years. ZigBee is characterized by high throughput and low latency for low duty-cycle applications. It uses CSMA/CA for channel access and devices have a typical range of 50 m.

This proprietary protocol is supposed to support three different kinds of traffic: *periodic* when the application dictates the rate and sensors wake up at this rate, *intermittent* when both the device and an application can request the network to wake up, and *repetitive* when the rate is fixed a priori with allocated time slots guaranteed to each device in order to prevent continuous communication.

To enable the transmission of data-traffic ZigBee can employ two different modes: beacon or non-beacon. The beacon mode is usually implemented when the coordinator is running on a battery while the non-beacon mode is applied when the coordinator is mains-powered. The first mode assumes that a device waits to wake up only when at periodic intervals the coordinator broadcasts its beacon; the second mode is designed for networks where the devices are sleeping most of the time and they wake up at random times to communicate changes in the sensor data they previously transmitted.

The standard is designed to enable three types of network topologies, *star*, *tree*, and *mesh* (see Figure 2.13). In a star topology the coordinator is the central node and all the others devices are usually RFDs (sometimes there could be also FFDs as leaf nodes). In a tree topology the coordinator is the node at level zero, the leaf nodes are usually RFDs and nodes at different levels of the network, communicating with other nodes at a deeper level, are FFDs. Finally, in a mesh topology, the choice of the kind of nodes is assigned consciously, but has no preferred pattern.

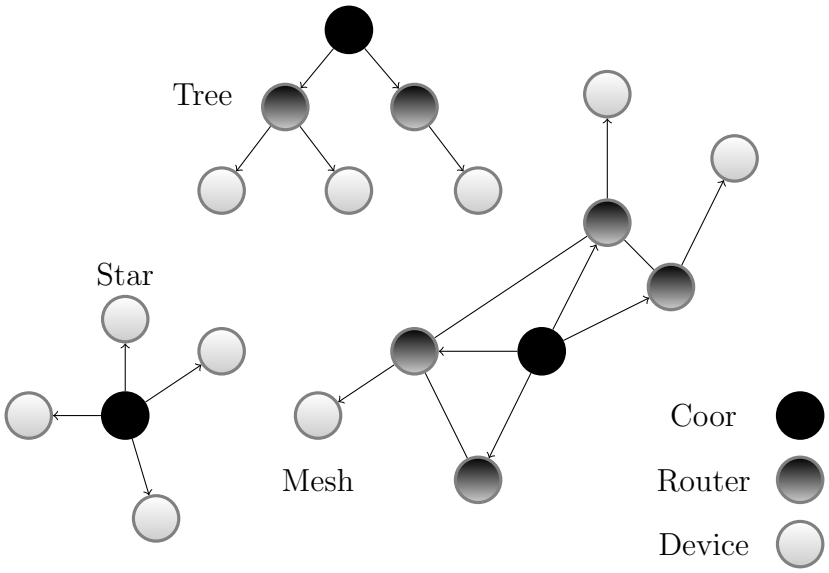


Figure 2.13: ZigBee topology models

ZigBee also defines a security mechanism to avoid intrusion of hostile parties. This security mechanism includes an Access Control List in order to accept messages only from known devices, 128-bit AES-based Encryption to prevent external devices from understanding ZigBee data, and Message Freshness Timers to reject timed-out messages.

2.7.1.2 ZigBee routing

The ZigBee routing algorithm is based on AODV in that each node keeps a record in its routing table for a particular route going from a source to a destination. Commonly the record is composed of at least a logical distance and the address of the next router on the path to the destination.

The routing process is initiated with route request being broadcast to all the neighbors in order to discover the best route to the destination. The process is repeated by all the neighbors until the destination is reached and answers with a route reply.

Thanks to the categorization of devices (explained in the previous section) the algorithm can be made more efficient in terms of minimizing the amount of RAM used by reducing the address space necessary for a routing entry (especially in devices such as RFD); in this case instead of recording the entire address of the node ZigBee allows the use of the address hierarchy to allow the use of a less efficient, but nonetheless usable, routing entry.

The coordinator of the network mostly works as an "aggregator", a device to which other nodes must send messages on a regular basis. In order to avoid the necessity for a node to discover every time the path to the aggregator, ZigBee offers a procedure to facilitate and optimize the discovery of the route to the aggregator by simply discovering all the paths to other nodes based upon a single route request broadcast.

Alternatively to distance vector routing, ZigBee also implements a second

algorithm, called "source routing". This can be used if the number of neighbors of the aggregator is so high that their routing tables could be overburdened. When applied, the routing information is stored in the routing table only at the aggregator, while other devices simply include all the path information in the frame itself. In order to start this procedure, the destination requesting to use source routing, sends a special message to the aggregator, called a Route Record command. Subsequently the aggregator will store the routing information in its routing table and from that point on all frame can be sent using source routing.

2.7.2 Contiki

Contiki [62] is an open source operating system designed for embedded systems in general, but for smart objects in particular. Contiki was written in C language due to its high portability. Contiki provides IP communication with the μIP TCP/IP stack. This was improved later on with the incorporation of the μIPv6 stack, a small IPv6 stack [42]. The footprints of the two stacks have values of less than 5 kB for the μIP and around 11 kB for μIPv6 stack [43].

For smart objects, IP was long believed to be too complex and heavyweight to be applicable. However, micro IP first within the memory provided by many micro-controllers and proved that an implementation of the IP stack was possible even in a constrained environment. While micro IP is a component of the Contiki operating system, it could be used as a stand-alone software package. This implementation includes the IP, ICMP, UDP, and TCP protocols and provides a routing protocol module to find out to where packets should be relayed.

2.7.2.1 RPL

RPL is a distance vector routing protocol that specifies how to build a Destination Oriented Directed Acyclic Graph (DODAG) using an objective function that, with a combination of metrics and constraints, is able to compute the best path from source to destination.

The graph building procedure starts at the root, usually called a LoWPAN Border Router (LBR). There can be multiple LBRs. Each LBR is configured by an administrator. RPL defines a set of ICMPv6 control messages to exchange graph-related information.

- **DIS** = DODAG Information Solicitation
- **DIO** = DODAG Information Object
- **DAO** = DODAG Destination Advertisement Object

The root starts by sending a DIO advertising to its neighbors the existence of a graph. These neighbors, according to the metrics and the constraints selected by the objective function, decide either to join the graph or not and compute their rank with respect to the root (which serves as the origin for coordinates in the graph hierarchy). If a neighbor is set to work as a router, it repeats the same procedure of the LBR to advertise this fact to all its neighbors. This process goes on until all "leaves" in the graph are reached and each node possesses a routing entry towards the root, building a so called *upward* routing graph. The DIS message is used to proactively solicit graph information via DIO.

Similar to the upward traffic, which flows from the leaf to the root, there could be download traffic, when the root wants to communicate with a leaf or when a communication request comes from outside the network. Routing tables need to be filled with information on how to reach individual nodes in the graph. This task is accomplished by the DAO message which is sent back by the neighbors to the node soliciting the existence of a graph with DIO. The message contains prefix information regarding the sub-network and is feasible for nodes at higher ranks to aggregate prefixes from different neighbors in order to create a shorter prefix which is again sent towards the root until all the nodes gather entries in their routing tables and a complete path to each prefix is built.

RPL provides techniques for loop avoidance and loop detection:

- Loop Avoidance is controlled by two mechanism, one that impedes a node from having a parent with a higher rank than itself by passing a variable within the DIO message called *max_depth* and another mechanism obliging nodes not to be greedy and try to go deeper in the graph in order to increment the number of parents.
- Loops in LLN are unavoidable, hence simple rules to detect them can be applied (for example, setting a direction bit in the RPL header). If a node set a bit 'down' in the header to assert that the packet needs to be forwarded towards its children, the next hop node checks its information in the routing lookup, and if finds out the message needs to be forwarded 'up', a loop is detected and the packet is discarded.

Finally, thanks to the possibility of building many graphs on the same physical topology and associating nodes with more than one parent (notice that to associate is a different concept from 'join'), multiple routes are guaranteed and local path repair is feasible in case of a node failure. In the unfortunate event that local path repair is not sufficient to determine a new path to the destination, RPL specifies the necessity of performing a global repair and redesigning the graph from scratch.

Chapter 3

Method

This chapter constitutes the main part of this thesis. It includes a description of the demonstration network which has been used, explicitly states the assumptions regarding the methods adopted, and gives a detailed summary of the system and device specifications.

Moreover, this chapter considers the methodologies chosen for the exchange of messages at the MAC layer and their format, the synchronization protocol used, and the algorithm introduced for clock drift compensation.

Finally the chapter gives a brief depiction of possible future scenarios, focusing mostly on the synchronization paradigm for a large-scale deployment, based on an optimized version of TPSN. This section is not considered further for the implementation due to the unavailability of physical devices to simulate the nodes behaviors, but the problem is treated from a theoretical point of view in order to address future developments.

3.1 Assumptions, network scenario and hardware specifications

The following two sections describe the considerations that must be taken into account regarding those aspects not directly connected with the thesis' focus or the specific network topology considered for the prototype's deployment. Moreover, this section describes basic functionality of the hardware which has been exploited for the different devices within the network.

3.1.1 Assumptions

First and foremost all the devices use a UWB signal for transmission purposes. The physical layer was implemented by a company providing chips and details of these chips are outside the scope of this thesis, we simply indicate that the chip that was used is based on the IEEE 802.15.4a amendment [7].

The mode of operation that was considered operates at 850 *kb/s* with a PRF equal to 64 *MHz* and total bandwidth of 500 *MHz*. This is not the only possible physical data rate, there are another two, one running at 110 *kb/s* and another running at 6.8 *Mb/s*, with same PRF and same bandwidth, which could be supported but were not analyzed in this thesis project. The channel utilized is

channel number 2 (see Table 2.2 on page 10) with a centre frequency equal to 3993.6 *MHz*. Other channels provided by the manufacturer are number 1, 3, and 4; however, not all the other are operational in the chips that we had access to.

At this data rate within this channel, operations are guaranteed over a ranging not higher than 43 *m*. The vendor expects to double this range for the next release of their chips.

The MAC addresses are expressed either following the EUI 64-bit standard or the 48-bit standard. The later is necessary due to the device processors working with 48-bit variables and the necessity of transmitting addresses either in 64-bit or 16-bit format. This issue is solved by a direct mapping from a 48-bit MAC address to a 64-bit MAC address as provided by the IEEE [45].

Consideration was given to battery powered operation, but for a first demonstration the devices are considered powered by a USB connection to laptops. This USB connection was also useful for debugging.

The synchronization algorithm applied is aims to get the best time accuracy and to work regardless of power consumption, as it was just developed for demonstration purposes; therefore it will need to be optimized to find the correct balance between synchronization accuracy and power consumed, especially in terms of the number of messages exchanged to get good time values to be processed by the Real Time Localization Scheme (RTLS) system and in terms of amount of time the devices spend sleeping.

The RTLS exploits TDOA as a localization scheme, but as this process simply gather timestamps and applies a formula (as already shown in the background chapter), the details of this computation are not addressed further in this thesis. All the TDOA computation are performed by an external device which is not considered further in this document. Instead we describe how these timestamps are collected and how they can guarantee a given position accuracy, rather than focusing on more how they are processed and used to compute the final position of devices moving within the radio coverture.

Localization must be performed at least each second in order to appreciate the continuous movement of the device and avoiding large jumps in position.

3.1.2 Network scenario

The network is composed of three main categories of devices:

- The **Coordinator** is the most important device in the network; it is in charge of collecting TDOA timestamp data from both the receiver in a direct communication with a device and from the other base stations that are within radio proximity. Localization is performed based upon multiple received from the different receiver in the network. The coordinator is attached to a server via USB or Ethernet link. The server receives the data for processing. The coordinator is in charge of discovering the other devices in the network, assigning them an address and communicating the parameters to be used for the communication. The coordinator is the reference for clock synchronization and is the most vulnerable part of the network.

- A **Base Station** (BS) is the device in charge of collecting both TDOA timestamps and communication with a node in the network. The BS sends

the timestamp data to the coordinator. This BS is sometimes referred as an anchor.

- The **Tag** is the device which we would like to localize in the network. It communicates via the BS to other network entities. This tag could be a personal tag, a normal wireless sensor network node or an asset tag However, for our testing purposes, the tag was considered plugged into a USB port of a personal computer.

The layout of the network which I have evaluated is shown in Figure 3.1. This is a simplest scenario. The squared shape space is most suitable in terms of the MAC layer topology; the room could have a different shape and base stations could be placed in different positions. It is up to the person who deploys the network to find the correct balance between radio coverage and base station placement.

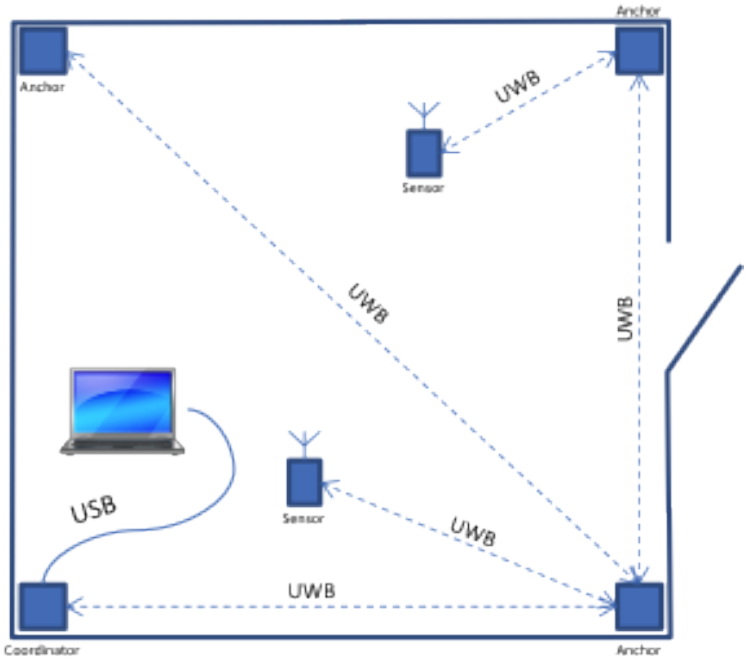


Figure 3.1: Schematic picture of demonstration system

As stated previously, all the devices are considered plugged into a mains power supply in order to test performances and make the network as simple as possible.

The position of the base stations is supposed to be well-known, as is the distance between them (even though we could computed the distance based upon Round Trip Time (RTT) measurements, but for sake of simplicity we assume that the deployer knows the physical position of each BS.

The coordinator should be turn on first, subsequently the base stations and eventually the tags (although only one tag was considered in the application scenario). However, in practice there is no required order in which the devices

should be turned on. The MAC layer implementation which will be described is supposed to understand any initialization scenario. Of course, initialization of the network does not actually starts until the coordinator is powered on, hence the other devices will simply wait for it, repeatedly sending a joining request.

The addresses are characterize by a PAN ID identifying the network and a subsequent MAC identifier. The roles of the devices were assigned statically but in a future scenario could be assigned dynamically, for example to support node failure.

3.1.3 Hardware specifications

The two components of the network (BS and tag) are characterized by two different kinds of chip.

The BS is an *STM32F107 ARM-Cortex M3* board core based on STM32F107VC micro-controller, working at up to 72MHz frequency, with 256KB Flash and 64KB RAM internal memory. The system frequency chosen for the case was 72 *MHz*, while the SPI frequency was 1 *MHz* [64].

The tag is based on a *TI MSP430F5438A* board working at up to 25MHz frequency, with 256KB Flash and 16KB SRAM. The chosen system's frequency for operations was 18 *MHz* while the SPI is working at the same frequency as for the base station [63].

The two components are modified for operations with UWB and they are completely compatible, even though further considerations will be analyzed when communication problems arise.

The UWB transceiver is using two different phase-locked-loop (PLL) for clock generation; the first one works on the PRF of 499.2 MHz while the second one multiplies this reference frequency for 128 (the number of pulses per symbol) to get a measurement of the the distance in units of difference in time between the pulses.

For debugging, a JTAG interface device is directly attached to the evaluation boards and exploited for printing values obtained on the laptop's screen. A tutorial concerning this testing device can be found at [46].

3.2 MAC layer operations

This section presents all the operations performed by the MAC layer starting with the initialization phase and going to the discovery and synchronization phase and continuing until stable communication is achieved. Also, explained in details the frame format follows the IEEE 802.15.4 standard.

3.2.1 Frame format

According to the IEEE 802.15.4 standard and considering the choices made for the operating mode the frame has a maximum payload of 127 bytes. This payload is divided into different fields.

- **FCF**: the Frame Control (FCF) field occupies the first two bytes and contains information regarding the frame type and other control flags

- **Sequence number**: occupies one byte and specifies the sequence number for the frame, if it is a response to a MAC command or to an acknowledgement
- **PAN ID**: the PAN identifier is a two byte identifier of the Personal Area Network; it should contain the field *0xffff* if the message is to be broadcasted (this is the case of messages sent from tags)
- **Destination address**: the destination address is the address to which a node wants to communicate and occupies eight bytes; again, the same assumptions as for the PAN ID are made in case of a broadcasted message
- **Source address**: is the address from which the communication wants to be generated; like the destination address, it occupies eight bytes
- **Data payload**: this is the user payload and occupies 104 bytes; in this thesis project contains particular information which will be explained lately in the thesis
- **FCS**: the Frame Check Sequence (FCS) is a security field composed of two bytes (we will ignore this value during this thesis project)

An overview of the final packet can be seen in Figure 3.2. It is important to notice that we assume in our tests that the network always has the same PAN ID, since in the standard, the distinction between destination and source PAN ID has been deleted as well as the two additional bytes occupied by the source PAN ID. Additionally, in the demonstration system no communication is possible between different networks, since we do not include an LBR.

2	1	2	8	8	104	2
FCF	n	panID	DestAddr	SourceAddr	Data Payload	FCS

Figure 3.2: MAC frame

As mentioned before the payload contains important information indicating from where the message was sent and the operation it wants to occur; these fields are analyzed case by case in the communication description. To give an idea of the kind of data that a packet could carry a list of possible values is given here for sake of clarity: code of the message (to identify the type of message received), timestamps (used to achieve synchronization), TDOA timestamps (to collect and send to the server), tagID (to inform the coordinator from which tag the TDOA timestamp is coming), clock offset information (to communicate once synchronization is computed), slot number (used to avoid collision once synchronization is achieved), and information regarding how to maintain synchronization (to be used by drift compensation algorithm).

3.2.2 State descriptions

Each device, according to the role it is playing in the network, during all the communication phases is working in a number of different states which are

presented in this section analyzing them case by case. Of course, as expected, some of the states will be clear from their names as they are common working modes in a wireless network interface (such as RX mode, TX mode, IDLE mode, SLEEP mode, etc.), but some other states are only partially understandable from their name, hence they are going to be described further in section 3.2.2.1. This subsection aims to give an idea, by means of diagrams, of the states that are relevant to the the implementation in terms of what needs to be performed in each state.

3.2.2.1 Possible states

First of all an introduction to all the possible states is given below.

- **INIT**: the first state (initial state) of all devices; in this state the role of the device is checked and depending on its role the operation of the device starts in a certain way.
- **SLEEP**: in this state the device is sleeping and is not consuming any power, just waiting for a defined time to wake up. The coordinator is assumed to never enter this state, as it should always be powered to guarantee communication with the server and to be always ready for receiving frames from other devices.
- **SLEEP_DONE**: when the clock reaches the correct value, it wakes up the device which passes through this state. The coordinator never enters this state as it never sleeps.
- **TX_WAIT**: the device enters this state when it sends a request to the physical layer in order to start the transceiver's transmission mode
- **TX_WAIT_ON**: if the transmission request is accepted by the physical layer, then the transceiver is turned on and the device is able to send data
- **TX_WAIT_CONF**: before sending data and upon receiving approval from the physical layer, the previous state of the communication is checked to determine which operation must be performed at the moment
- **RX_WAIT**: the behavior is similar to TX_WAIT, but for the receiving mode
- **RX_WAIT_ON**: the homonym of TX_WAIT_ON for the receiving mode
- **RX_WAIT_DATA**: in this state the device checks the packet that has been received and extracts the necessary information to be saved in its local memory
- **TXPOLL_WAIT_SEND**: this represents the state in which the device sends the first packet both to request the coordinator to join the network and to start synchronization procedures; only tags and base stations are able to send this packet. When the device is in this state the construction of the data packets is suspended.
- **TXRESPONSE_WAIT_SEND**: this state is entered only by the coordinator and is used to generate the response to POLL messages. The coordinator also keeps track of the synchronization procedure.

- **TXFINAL_WAIT_SEND**: after receiving a response from the coordinator, tags and base stations send an additional message to the coordinator in order to increase the synchronization accuracy by accumulating more timestamps.

- **TXREPORT_WAIT_SEND**: the coordinator acknowledges the device by sending a FINAL message with a REPORT containing all the necessary information to finish the synchronization procedure.

- **TXTDOA_WAIT_SEND**: finally, once synchronization is achieved, the tag is able to start sending TDOA timestamps by broadcasting a packet to all the base stations. The base stations timestamp upon arrival of the packet and resend this information to the coordinator which collect four TDOA packets and sends them to the server. Hence the coordinator is not subject to this particular state.

3.2.2.2 Coordinator state diagram

A visual explanation of the coordinator's state machine is shown in Figure 3.3.

As shown in Figure 3.3, in the initialization phase, after checking the actual device is the coordinator the node enters receiving mode and waits for the first poll from either a base station or a tag; upon the arrival of the first poll, unless packets are lost, the coordinator transitions into transmitting mode and sends a response to this message, then returns back to receiving mode and waits for a final message. When the final message is received the coordinator switches to transmitting mode and acknowledges the device with a report; then it goes back to receiving mode and waits for either the next poll from another device or the TDOA packet from any device. When the four-way message exchange procedure has terminated with all the devices in the network, then the MAC topology has been built.

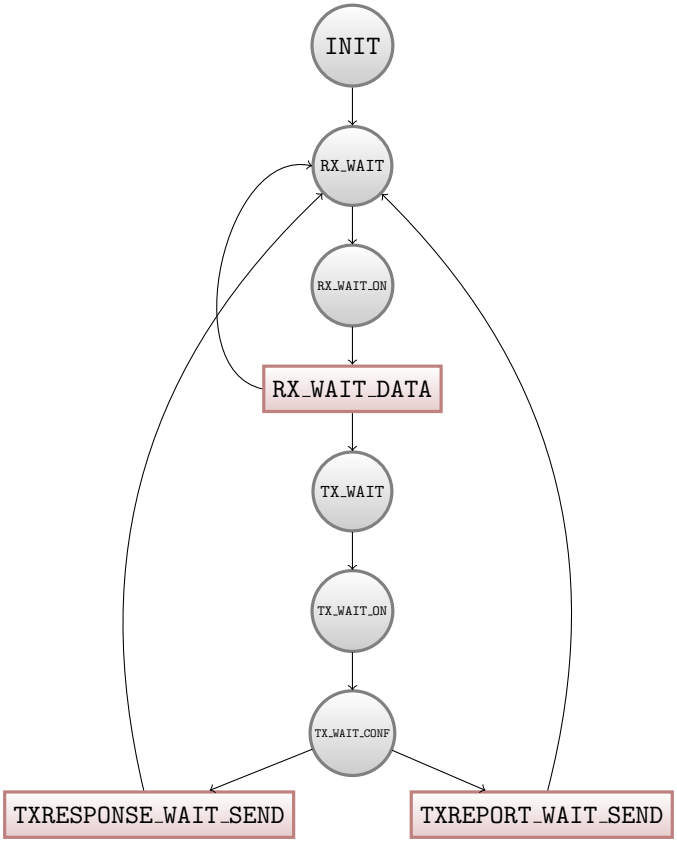


Figure 3.3: Coordinator states

It is important to notice that in general, when the coordinator is not performing any operation, it stays in receiving mode, in order to be always able to hear any kind of traffic being transmitted in the network and to decide what operations to perform.

3.2.2.3 Base station state diagram

On the base station side the states diagram is slightly different. This state machine is shown in Figure 3.4.

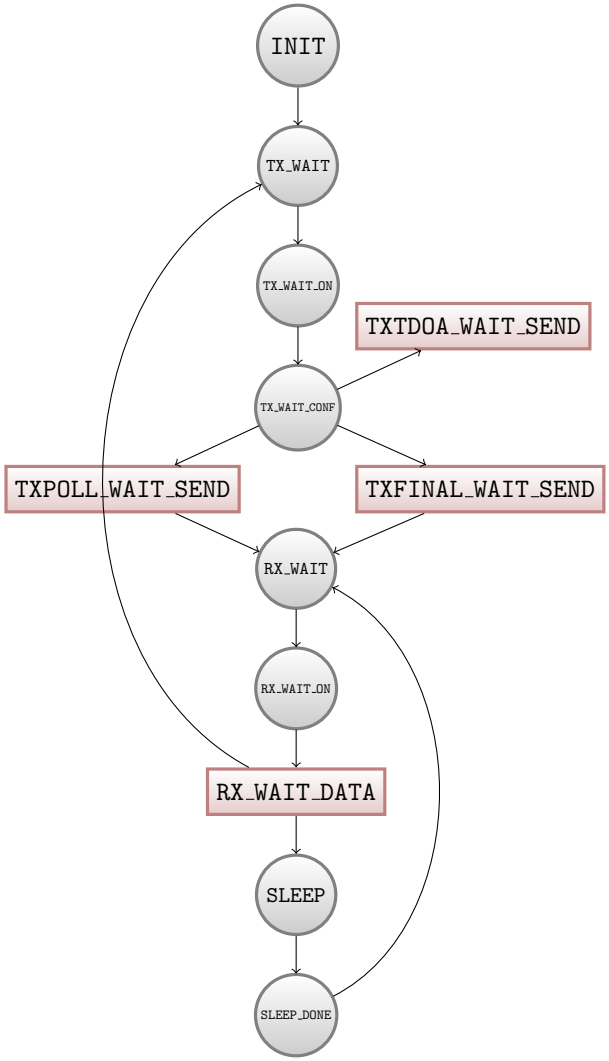


Figure 3.4: Base station states

After having determined during the initialization phase that the device is a a base station, it starts its operation in transmitting mode, sending a poll message to the coordinator (which has a known and static address); then the BS switches to receiving mode and waits for a response. Once the response is received the BS switches to transmitting mode again, and send its final acknowledgement to the coordinator, then waits for the report. After the report is received, the device goes to "sleep" for a certain amount of time as indicated by the coordinator in the report message. This gives time to the central device to discover all the nodes in the network and allows this device to reduce its power consumption.

Once the network is built, and after the number of ticks in sleep mode before waking up, the base station turns on in receiving mode and waits for a TDOA broadcast from the tag; upon arrival of a TDOA packet, the BS goes to transmitting mode and sends received time for this TDOA packet to the coordinator.

After this mechanism is completed, it is repeated every second or when the tag sends information. This periodic transmission of TDOA packets is used to keep track of the tag's movement, but between the transmission of two different TDOA from the tag, the base station can be put to sleep.

3.2.2.4 Tag state diagram

The tag state diagram can be derived from the base station one. Basically the operation flow follows the same pattern *except* for one small difference: when the sleep mode after receiving a report from the coordinator is over, the tag will enters the transmitting mode instead of receiving mode as the tag only needs to broadcast TDOA information to all the other devices, but it does not need to receive TDOA packets from anyone.

3.2.3 Communication installation

As mentioned in a section before, the communication is more fluent if the devices are turned on in order, starting with the coordinator, then the base stations, and finally the tag, but in practice there is no diversity if the order is changed. Base stations and the tag simply keep returning to transmitting mode and try repeatedly to send a poll message until a response is detected. The power consumption, in such an 'irregular' initialization phase is a little bit higher due to the transceiver turning on and off very fast, but since the transmitting mode is less expensive in terms of energy than the receiving mode, and keeping in mind that the demonstration system has power supplied to all node, hence there is no special problem from such an initialization configuration. However, consideration should be given in a future deployment with battery supplied devices, as there is likely to be a more dynamic environment for the initialization.

Once the devices are powered on, each base station and the tag sends their poll message addressed to the coordinator (thus also informing the coordinator of their own address). The first message arriving at the coordinator's receiver starts the pairing procedure only with the device whose message arrived first. The reason is that the coordinator would otherwise need to process multiple messages in its local memory and sends responses at different times. Additionally, this pairing messages are exploited for synchronization purposes and having the coordinator trying to communicate with multiple nodes could undermine the timestamping of transmissions leading to a miscalculation in the synchronization procedure.

While the coordinator and devices are pairing with each other and synchronization between them is achieved, the other devices keep trying to send poll messages until the coordinator responds. This procedure is repeated for all the devices in the network and all the nodes which are already paired with the coordinator will simply wait for a defined time in sleeping mode before turning on either in receiving or transmitting mode for the upcoming TDOA phase. The sleep is computed by the coordinator and the sleep time period for the node is communicated within the report message.

The TDOA phase starts with the tag sending an empty message broadcast, to all the network when the other devices are in listening mode. Upon arrival of the message at each device, the receiving time is timestamped and at the coordinator side, the received timestamp is simply saved in the local memory,

while at each base station the received timestamp is included in a new packet which is sent to the coordinator. How the messages being sent by the base stations avoid colliding is explained later in this thesis.

Once the coordinator collects exactly four TDOA timestamps, it is able to forward this information to the server which will apply the TDOA formula showed earlier in Chapter 2 to compute the position of the tag and show the location of the tag on a graphical interface with a map of the installed devices. It is important to be aware that, while the coordinator is sending this information to the server, it is also continuing network communication. This is possible since the the coordinator-server link utilizes a different interface (Ethernet in this case).

The TDOA phase is repeated each second, but it could be interrupted by a request by a node for resynchronization, due to the increasing clock drift of its micro-controller. In this case the TDOA information will be lost, as well as the movement of the tag (which will generate a jump on the map). The four-messages exchange procedure is repeated within the device that requested resynchronization. After completion of the synchronization process, the normal TDOA phase is reestablished.

3.3 Synchronization specifications

In this section, a theoretical analysis of the synchronization algorithm introduced roughly in the previous description and of the drift compensation algorithm is given, as well as a comparison with other standard synchronization mechanisms, also treated in Chapter 2.

3.3.1 Synchronization algorithm

The chosen synchronization is based on the Symmetric Double Sided-Two Way Ranging (SDS-TWR) protocol, described in the IEEE 802.15.4a amendment [47] and first introduced by Hach [48]. This algorithm has been used for ranging through Time-of-Flight (ToF) estimation in an asynchronous system, due to its better calculation based on two different samples of the propagation delay. Examples can be found in [49], where ranging errors are computed using a SDS-TWR with unequal reply time and in [50] applied to Chirp Spread Spectrum (CSS) technology. Even with the advantages mentioned, this algorithm itself does not take into account the error due to clock frequency offsets in coarse oscillators.

Sahinoglu and Gezici in [74] shows how this algorithm is implemented directly on the transceiver chip without using the processor's clock for timestamping of the arriving packets. Modules such as those from *Nanotron Technologies GmbH* implement this ranging in the chip.

As stated in the assumptions, the distance between the base stations is measured when installing the network. Thanks to this information and to the algorithm above cited it is possible to estimates the frequency offset of the two node clocks, as proposed in [51], which applies this technique to UWB systems. Once the frequency offsets are known synchronization can be performed with greater accuracy despite possible imperfections in the oscillators.

In figure 3.5 the algorithm is depicted using terms which will reappear later in the thesis.

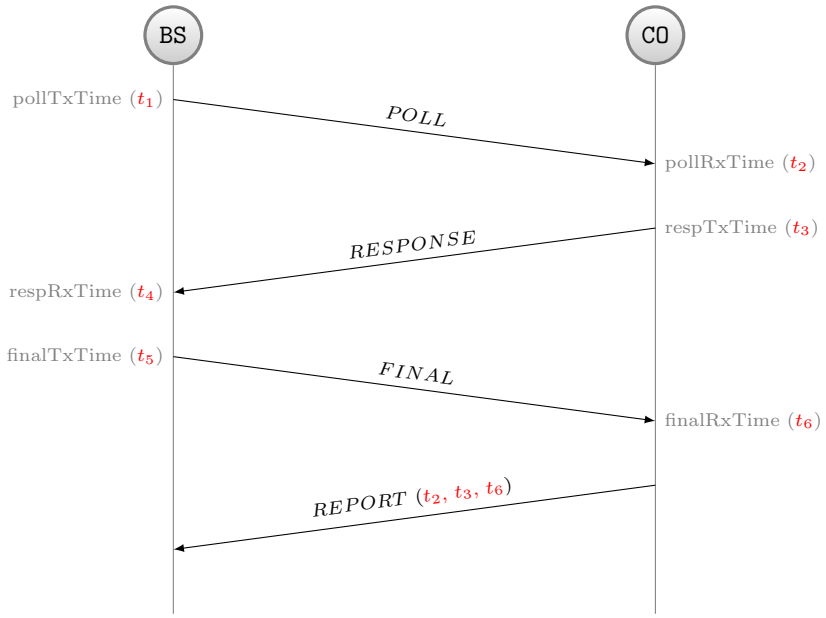


Figure 3.5: SDS-TWR

The base station can exploit six different timestamps (three saved in its local memory and three incoming from the coordinator in the report packet) and using the known distance between itself and the coordinator.

With the six timestamps a first estimate of the time of flight including the frequency offsets of the clocks can be done.

$$estToF = \frac{1}{4}(t_4 - t_1) - (t_5 - t_4) + (t_6 - t_3) - (t_3 - t_2) \tag{3.1}$$

Then if the counter ticks (n) and the frequency offsets (e) are introduced it yields:

$$estToF = \frac{1}{4}\frac{(n_4 - n_1) - (n_5 - n_4)}{f(1 + e_{BS})} + \frac{(n_6 - n_3) - (n_3 - n_2)}{f(1 + e_{CO})} \tag{3.2}$$

Knowing the speed of the signal's propagation in this media and indicating with d the distance between coordinator and base station, it is possible to compute another measured value for the ToF.

$$teoToF = \frac{d}{c} \tag{3.3}$$

The time interval between the two transmitting events and the two receiving events is the same (as both are the same ToF from the BS to the coordinator); writing this in terms of counter ticks and frequencies gives:

$$\frac{n_5 - n_1}{f(1 + e_{BS})} = \frac{n_6 - n_2}{f(1 + e_{CO})} \tag{3.4}$$

From equations 3.2, 3.3, and 3.4 the frequency offsets can be estimated and the final offset between the two clocks calculated as difference of these values, yielding equation 3.5.

$$offset = \frac{estToF - teoToF}{(n_5 - n_1) - (n_6 - n_2)} \tag{3.5}$$

This value can then be used to adjust the time at which communication must be performed between the devices.

It is important to notice that this algorithm is really expensive in terms of power consumption; it requires four transmissions and changing rapidly from transmitting mode to receiving mode. However, this method has been chosen because it guarantees the maximum time accuracy as it will be analyzed later in this thesis, hence it provides maximum localization resolution.

On the tag side the same procedure was adopted even though it is unrealistic for battery powered devices; moreover, the tag does not have the necessity of reaching such a level of synchronization, as it is not involved in TDOA operations (the TDOA timestamps are performed upon the packet's arrival at the anchors).

In the future this algorithm will change for the tag-coordinator link but for company issues. However, for the purposes of this thesis project we will assume that all the boards are powered for both simulation and demonstration purposes. The desire for rapid deployment led to this optimum, but energy expensive choice.

3.3.2 Drift compensation algorithm

Clock drift is the difference between two reading of the same clock at different times. Many issues are involved when talking about a clock drift: the power supplied to the clock is not constant, the temperature, and the environment can cause the oscillator to oscillate at different frequencies, the age of the oscillator can degrade the performances of the clock in terms of frequency and so on.

On of the most common algorithm used in wireless sensor network to compensate for clock drifts is linear regression to provide an estimation based on the measurements of the clock's frequency performed at different times. By simply fitting a linear equation to these values on a graph with actual time on the x axis and the clock's time on the y axis. Once this line is drawn, if the clock appears to be too slow or too fast to the 45 degree angle that should occur if the clock was perfectly accurate, we can correct the clock based upon the distance of a clock's time from the ideal line. Many examples can be found in the literature, utilizing for example Ordinary Least Squares [52, 54] or Recursive Least Squares [55] linear regression.

The problem with these methods is that they are very prone to rounding errors, especially when synchronization cannot be considered to be always perfect in terms of the offset computation.

Sommer and Wattenhofer propose a simple algorithm, based on the concept of moving average filter [53], to keep track of the drift by analyzing the offset computed during synchronization procedure. Every time synchronization is repeated, the new value of the offset is stored and is compared to the previous values using a weighted function as shown in Equation 3.6.

$$\Delta offset_{avg}(t) = \alpha \Delta offset(t) + (1 - \alpha)\Delta offset_{avg}(t - 1) \tag{3.6}$$

This formula smooths out the uncertainty coming from the synchronization process and avoids the synchronization process being repeated too often (an indispensable requirement for devices which needs to keep a high level of synchronization accuracy), as this would cause increased message overhead in the network.

This algorithm also helps from a power consumption perspective. Even though during the initialization phase the synchronization algorithm is too energy expensive, once the initialization phase is over this method avoids repeated synchronization, saving energy when the system is operated over long intervals.

If the offset is moving too far from the previous value, then a warning is raised to enter synchronization mode (by sending a poll), otherwise the operation proceeds as usual. Obviously, an initial static synchronization interval must be set based upon basic information about the oscillator's characteristics as provided by the oscillator's data-sheet.

3.4 Large-scale scenario

In this section a theoretical proposal for a large-scale scenario, with many more base stations and a huge number of tags, is given in terms of synchronization and the frame routing protocol.

3.4.1 Synchronization

We consider an indoor scenario in which the deployment of many base stations is necessary to cover the desired area and a very large number of objects and people are present - each carrying a tag. In this scenario is obvious that the algorithm above in the previous section is not suitable for many reasons. First of all, the packet exchange, if we consider only one coordinator, would be incontrollable. Secondly, the possibility of synchronizing each base station directly with the coordinator would be impossible due to lack of line of sight and the radio receiver's limited capabilities. Furthermore, when using batteries the algorithm unrealistic, at least for tag-anchor synchronization.

Kulakli [56], proposes some improvements to TPSN which would make it suitable for this scenario and making it applicable to the anchor-case synchronization.

In a wide environment TPSN builds a spanning tree in which through neighbor discovery, a level is assigned to each node. This procedure is repeated until all the nodes are discovered. If a node fails to receive a neighbor discovery message, it simply sends a level request packet to its neighbors. Each of these answers with a packet including its won level. Now the node assigns itself a level number that is the minimum level received from the neighbors plus one.

Synchronization is done for each node periodically. Each node sends a synchronization packet to its parent periodically in order to synchronize. These characteristics are further specified by Ganeriwal, Kumar, and Srivastava in [13].

The main issue with this method is that with the increasing number of anchors in the network, the node level will increase as exponentially, thus leading to a poorer synchronization accuracy that increases linearly with the depth in the spanning tree.

3.4.1.1 Clustered TPSN

In order to dodge the problem described above, a clustered solution is introduced, this enable the design of a mechanism to achieve synchronization directly between cluster roots and afterwards among nodes within a cluster.

By setting a maximum depth level for a cluster it is easy to build a tree in which, once the maximum level is reached, the counter starts again from zero and another cluster is created simply adding a new variable equal to the cluster level. This results in a tree such as that depicted in Figure 3.6.

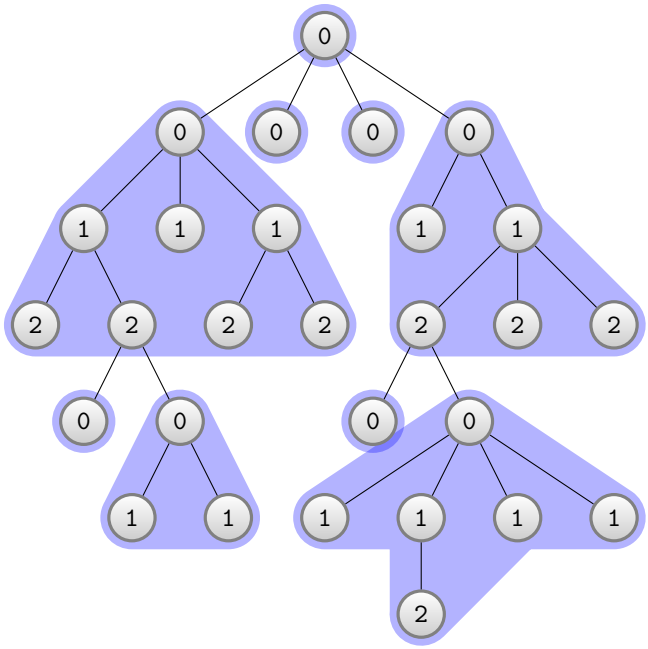


Figure 3.6: Clustered TPSN (Max Depth Level 2)

Obviously, in a very unfortunate scenario, the number of nodes at the same level could be huge leading to unbalanced clusters. To avoid this imbalance two enhancements can be exploited, one balancing the number of children a parent can have and another balancing the number of nodes within a cluster.

The first improvement, called **Balance Children Count**, uses the synchronization requests of the children during the synchronization phase. This is due to the fact that the old level discovery phase did not keep track at the parent of the number of children it may have. For this reason this new procedure is performed during the synchronization phase, so that as soon as a parent reaches the maximum number of children allowed, it replies to additional potential with a reject packet. The child node will then attempt to synchronize to another node until a parent who can host more children is encountered. Clearly this procedure needs to store the variable related with the number of children in the parent node. The procedure adds time complexity to the spanning tree construction, but considering that occurs during synchronization phase and that the actual tree is already built in advance during the discovery phase, thus this

balancing only improves the system *after* initialization.

The second enhancement, called **Balance Cluster Node Count**, builds directly upon the first optimization and again occurs after initialization. This optimization is introduced because not only may the number of children be unbalanced, but also the number of nodes belonging to a cluster could be imbalanced (leading to different sized clusters). To solve this, the information related to the number of children associated with a parent should be sent back to the root of the cluster which does a simple sum of all the values received minus one. If this number exceeds the maximum number of nodes for a cluster, again the node is rejected and the potential node asks for synchronization by another parent (possibly belonging to a different cluster), otherwise it will be rejected again until a free cluster is found.

Figure 3.7 shows the final result having a maximum number of children equal to 3 and a maximum number of nodes for a cluster equal to 6.

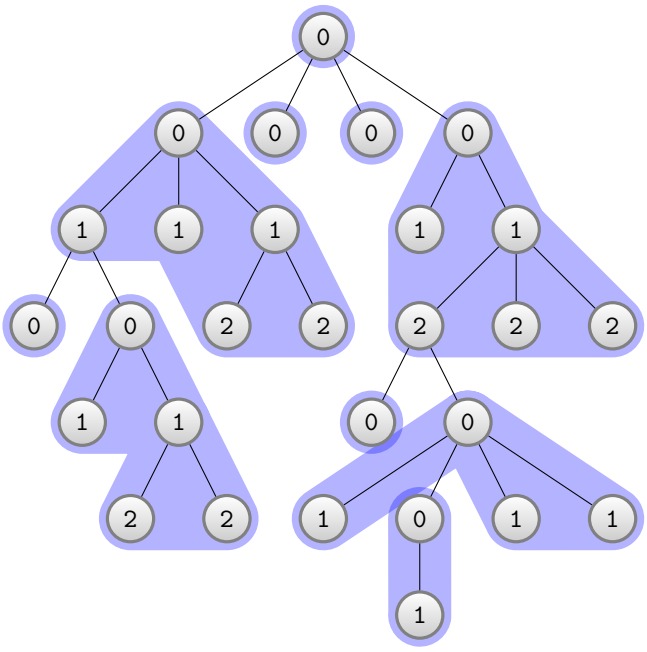


Figure 3.7: Final balance results

3.4.1.2 Inter cluster synchronization

Now that the spanning tree is built with a more optimal pattern, synchronization accuracy can be improved by means of an inter cluster procedure. Indeed, this method could be exploited when the spanning tree is constructed normally normally, i.e., following the TPSN standard, however, in this way, each node would be synchronized directly with the very first root causes the clustering algorithm just designed to be useless and leading to poor synchronization at the leaves.

The cluster division can be used to create a mechanism which synchronizes any node with any other node, whether to be a cluster root or another node in

the tree. In a normal TPSN environment, only the children know their parents when synchronization is requested by a child the information is passed directed to its parent. Normally the parent replies immediately but to achieve inter cluster synchronization another option arises.

If we imagine a scenario of three nodes in a row, which we call i, j, and k, where the first is the lowest level leaf and the last the highest level parent, the following operations occur:

1. Node i asks for synchronization with node j
2. Node j, instead of replying immediately, stores the requester's address and when (the time) the requester sent this request as this information is crucial for offset calculations
3. Node j prepares a new packet in which the requester's sent time is replaced with its own time and sends the resulting request to k
4. Node k acknowledges the request by sending a packet to node j
5. Node j adjusts its clock according to its own information and the new information which it got from k
6. Node k substitutes the original requester's sent time with the value stored when the message came from i and acknowledges the request with this own information and the information it got from k
7. Node i can now adjust its clock according to k's clock

This mechanism increase the number of packets sent in the network and the processing required by each of the nodes, but it also increases the level of accuracy by exploiting the anchor-anchor link.

3.4.1.3 Adaptive synchronization interval

As stated previously, in TPSN, synchronization is repeated periodically. By making use of the moving average filter presented in section 3.3.2 it is possible not only to smooth the uncertainties coming from the offset calculations and eliminate the drift contribution, but also to estimate how much the clocks differ from each other. This enables us to specify an adaptive synchronization interval.

If the freshly computed offset results are similar enough (i.e., within a guard interval) to the previous one (which requires the previous offset to be stored), then the synchronization interval could be increased by 1%; otherwise, if the clocks are drifting apart too much the synchronization interval could be reduced by the same percentage.

Again a first estimate of the interval must be made *a priori* when installing the network.

3.4.2 Routing

With all these premises, it is clear that RPL can help if synchronization is to be performed at network layer rather than at the link-layer. It is not completely defined at which layer synchronization should occur, the higher in the ISO-OSI stack we go, the more processing is required to access the clock information.

However, in an ideal scenario in which the most important parameter to determine the functionality of a WSN is power consumption, minimizing the number of messages exchanged could be an interesting solution.

With this in mind, it is clear that in a system designed to use RPL, building a DODAG could be used to implement TPSN within it. The main idea consists in merging the level discovery phase of TPSN with the graph solicitation provided by DIO messages. The graph is built according to metrics and constraints of RPL, but the same graph is valid for TPSN functions.

Afterwards, to provide a synchronization phase, which is composed of a pairwise message exchange, the DAO is used to request synchronization in addition to upward routing. Obviously, to complete this process, one more message must be designed so that the node not only routes the DAO information up to the next level but also answers the node requesting synchronization and communicating its position in the graph. This message is a simple ACK with the timestamps needed to calculate the offset.

Both the advantages of RPL and the enhancements analyzed for TPSN can be exploited by a single protocol if cluster configuration is provided for in RPL. This problem can easily be solved by adding information regarding the cluster ID into the RPL DIO and applying the maximum depth level used in the description of the TPSN optimization.

The Contiki operating system is based on the 6LoWPAN stack and it implements RPL. In [57], Xiaonan Wang and Huanyan Qian propose a 6LoWPAN wireless sensor network based on a cluster tree, which requires modifying the Contiki core and the RPL functions included in it, making it possible to meet all the requirements stated above and to build a system with clusters which supports both enhancements to TPSN (whose definition must be done within RPL) and implementing a clustered structure of the DODAG as created through neighbor discovery by RPL.

The resulting implementation guarantees a high level of synchronization accuracy and delivers all the features provided by RPL, such as loop avoidance, loop detection, and multiple graph topology (which sustains node failure also within a TPSN environment if nodes need to find a new source for synchronization).

The final protocol stack for such a system is depicted in Figure 3.8.

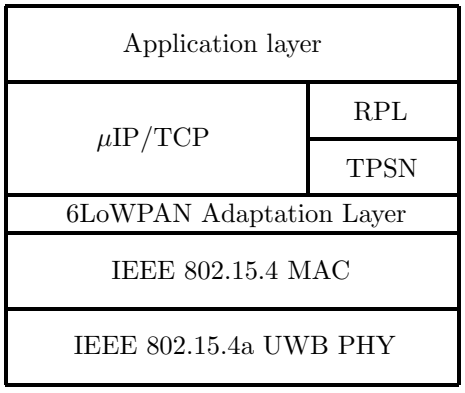


Figure 3.8: The resulting UWB WSN protocol stack

3.4.3 Handover

In the previous two sections the tag was never been mentioned (either in terms of synchronization or in terms of DODAG creation). As already stated, the tag which needs mobility and should have a very low power consumption in order to operate for a long time, even with small capacity batteries. The only functions related with the tag are positioning through TDOA and possibly delivering sensor data collected by the tag.

Therefore, synchronization at a deep level in the graph is not necessary at any point and it is actually futile to talk about a synchronization algorithm concerning such a low functionality device. Nor does routing represent a big obstacle, because the tag simply discovers in which cluster it is it broadcasts to all the members of a cluster. The four closest base stations will receive this information and will time stamp the packet's arrival and send this information to the administrator of the cluster. If the coordinator is directly connected to the server then the coordinator will send the information directly to the server, otherwise, it will act as a normal router to find the best path to the root by exploiting the algorithms above.

Two main issues come out of this analysis: the collision problem and the handover between clusters when a device is moving from one cluster to another.

The first problem is outside the scope of this thesis and we simply state that, in general, for the IEEE 802.15.4 standard the most common collision avoidance algorithms are ALOHA, CSMA, or CSMA/CA. We will simply assume that one of these algorithms is used.

The second problem, which can be seen as a normal handover, is addressed in this thesis. First it is worthy mentioning that 6LoWPAN provides a technique performing handover [35], another references to this subject concerning indoor clustered WSN can be found in [58].

The common background of these theories is that they are probably working with UWB radio technology, but they have not been tested.

In this thesis project we have chosen the 6LoWPAN handover mechanism because it facilitates operation with all the other systems employed *without* any further modification. Hence, we use the well-known NEMO algorithm, as described in Chapter 2.

For reason of completeness, a more profound understanding of a possible handover mechanism for UWB radio technology, even though not applicable to the presented scenario, can be found in [59].

Chapter 4

Implementation

In this chapter details of the implementation of the MAC layer and the synchronization algorithm for the network depicted in figure 3.1 are described. The choice of the software environments, listing of the parameters used, description of the messages exchanged, and a detailed analysis of the operations (as well as pseudo code for the algorithms exploited) are included in this chapter.

4.1 Software environments

The choice of the software environments to be used was determined by the hardware of the devices adopted for the development of the two main actors in the network.

For the tag Texas Instruments Inc. Code Composer Studio (CCS) [60] was used, whilst for the anchor side Keil's μVision [61] compiler was used.

CCS is a complete, Eclipse-based integrated development environment (IDE) that supports all MSP430 micro-controller devices with some limitations from a debugging perspective.

μVision is a complete open source IDE offering good support for debugging and compatibility with ARM-Cortex M3 micro-controllers.

All of the code was written in ANSI C for both sides taking into account the impossibility for the MSP430 working with 64-bit numbers, unlike the M3, and therefore applying the necessary adjustments.

The physical layer code was provided by the manufacturing company and was used only for calling functions to transmit and receive IEEE 802.15.4a frames and for configuration purposes. All the modification to this layer were made by another thesis work [72].

4.2 Packets description

In Chapter 3 different packets were mentioned for synchronized communication. These packets are actually the only packets that we have implemented as no application has been yet developed for the sensor tag.

Additionally there is a sixth packet, which carries the information sent to the server from the coordinator. This packet contains four TDOA timestamps,

but it's packet format has been defined to be compatible with the Ethernet interface, therefore it is out of the scope of this thesis.

The structure of each packet's payload is going to be described and usage will be analyzed while describing the network functions. However, first it is necessary to define some codes to indicate the type of each packet. These packet codes are included in the payload at the transmitter so that they can be recognized at the receiver. The packet code itself is simply a field in the user payload.

The packet code, expressed in hexadecimal, associated with each type of packet and the packet type names are shown in Table 4.1.

Table 4.1: Message Function Code list

Packet code	Packet name
0x21	POLL
0x10	RESPONSE
0x29	FINAL
0x2A	REPORT
0x30	TDOA

The MFC differ from the one implemented in [72] due to later improvements of the network.

4.2.1 Poll message

The poll message is the first message sent by anchors and tags. We assume that the coordinator is in receiving mode. The message is used by the anchor and the tags to inform the coordinator of their existence in the network and to request that the coordinator proceed with the synchronization procedure. The pool message also establishes a path for future communication.

The pool message only needs to carry the Message Function Code and therefore its payload has a length of 1 byte.

This polling message with its payload is depicted in Figure 4.1.

0	2	3	5	13	21	22
FCF	n	panID	Coordinator	Anchor/Tag n	MFC (0x21)	FCS

Figure 4.1: POLL frame

In the picture the starting ordinal byte offset of each field is shownThe complete polling MAC layer frame consists of 24 bytes.

The first byte of the FCF is set to *0x1* to indicate data type in the payload, while the second byte is set to *0xC* to indicate the use of EUI 64-bit addresses.

4.2.2 Response message

The response message is sent by the coordinator after receiving a poll message from any device. The response message is used to establish communication be-

tween the two devices and to continue the four-way message exchange necessary for synchronization.

Again this message does not carry any particular information except for the MFC and the length in bytes of the payload. Therefore the complete MAC frame, with same options for the fields as the poll message, is shown in Figure 4.2.

0	2	3	5	13	21	22
FCF	n	panID	Anchor/Tag n	Coordinator	MFC (0x10)	FCS

Figure 4.2: RESPONSE frame

4.2.3 Final message

The final message is the third message in the chain of synchronization and can be sent by the anchor or tag only upon receiving a response from the coordinator, hence only anchors and tags construct this frame.

As for the poll and the response, the message does not include additional information other than the MFC. Therefore it is the same length as the pool and response message and with same options in the FCF. The FINAL frame is shown in Figure 4.3.

0	2	3	5	13	21	22
FCF	n	panID	Coordinator	Anchor/Tag n	MFC (0x29)	FCS

Figure 4.3: FINAL frame

4.2.4 Report message

The report message is the last message necessary to permit adjustments of the clock based upon the coordinator's clock. Three timestamps are included in this packet to furnish the anchor or the tag with all the necessary data in order to perform offset calculations. Besides, considering that the initialization of the whole network could take a while depending on success of the pair-wise messaging and the number of devices involved, a field in the payload informs the device when to start the TDOA phase and the slot number assigned for this communication (a time slotted configuration is proposed to avoid subsequent collisions once synchronization is achieved and this method is explained in the following sections).

The FCF field is equal to the prior examples, the payload includes all the fields above described as shown in Figure 4.4. The abbreviations for the fields in the report are explained following the figure.

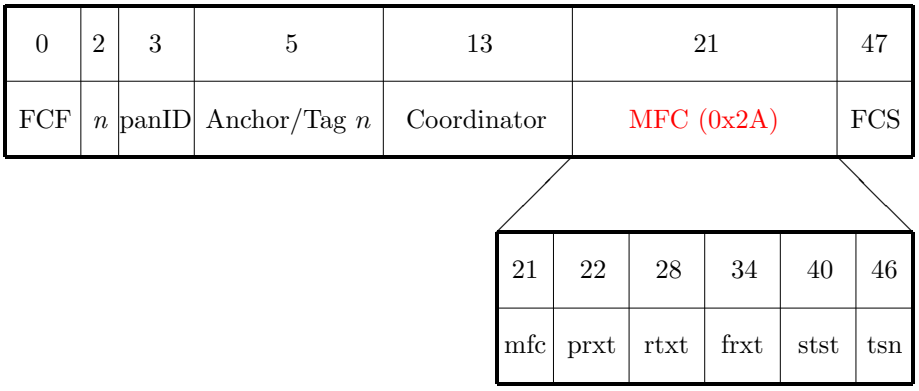


Figure 4.4: REPORT frame

As can be appreciated from this figure, the total size of the frame is 49 bytes and the user payload has a length of 26 bytes. The new fields introduced in this payload are:

- *prxt* (Poll Rx Time) is the timestamp for the poll receive time at the coordinator side
- *rtxt* (Response Tx Time) is the timestamp for the time the response is transmitted by the coordinator
- *frxt* (Final Rx Time) is the timestamp when the FINAL frame is received at the coordinator
- *stst* (Start TDOA Sensing Time) is the timestamp statically decided by the coordinator to inform the device when to start TDOA operations after the initialization phase is over
- *tsn* (Time-Slot Number) represents the time-slot number assigned to each specific device

The *stst* field is set to 0 when the pair-wise messaging is used only for synchronization and the network is not in the initialization phase.

It can be seen that each field carrying a timestamp is composed of six octets. This is due to the necessity of maintaining compatibility of the values used by the two different devices; On the tag side, in fact, CCS does not provide support for *unsigned long long* numbers for this special board.

4.2.5 TDOA message

The TDOA message comes in two different versions, depending whether the device constructing the message is a tag or an anchor; this is due to the fact that the tag is simply sending an empty packet (or it could contain sensor data), whose receiving time at the anchor side is used to perform the TDOA computations.

When the message arrives at the coordinator this message is simply timestamped and the timestamp value is stored in the local memory, while at the anchor side the message is timestamped and the value is then routed to the coordinator in another message, thus the necessity to define two different versions.

4.2.5.1 Tag version

The tag version of this packet is exactly the same as a normal poll, response, or final message and has the same FCF options - leading to the same dimension. Only the MFC is different.

0	2	3	5	13	21	22
FCF	n	panID	Coordinator	Anchor/Tag n	MFC (0x30)	FCS

Figure 4.5: TDOA frame (tag version)

4.2.5.2 Anchor version

The anchor version presents some differences due to the necessity of including the timestamp when a TDOA message is incoming from the tag. The message also includes the tag ID to tell the coordinator which device wants to update its position. This field is not currently necessary as only one tag is utilized for the demonstration system, but would be necessary if more tags are deployed in the same coverage area associated with the four anchors.

The structure of this packet and its payload are shown in Figure 4.6.

0	2	3	5	13	21	22	28	36
FCF	n	panID	Coordinator	Anchor/Tag n	mfc 0x30	tdoa	tag ID	FCS

Figure 4.6: TDOA frame (anchor version)

The field *tdoa* represents the timestamp of the arrival of the TDOA message from the tag and it is expressed as a 48-bit number. The field *tag ID* is the tag's address expressed as an EUI 64-bit format.

The total size of the frame is 38 bytes, with a payload of 15 bytes.

4.3 Network up and running

This section describes all the operations performed in the network until it is working in a stable mode and the tag's position update is occurring once every second (i.e., at 1 Hz).

4.3.1 Initialization phase

At the beginning of the initialization phase each device enters the INIT state. In this state a simple *if* based on the role the node plays in the network determines which device the code is executing on and this controls the following decisions regarding addresses and configuration (as shown in the code listing below).

```
if mode == tag then
    storeaddress
    initSyncInt = const
    appState = TX_WAIT
    nextState = TXPOLL_WAIT_SEND
else
    if mode == anchor then
        storeaddress
        initSyncInt = const
        appState = TX_WAIT
        nextState = TXPOLL_WAIT_SEND
    else
        if mode == coor then
            storeaddress
            appState = RX_WAIT
        end if
    end if
end if
```

If the device is a tag it copies its address from the local memory (using a vector of 8 bytes to store this 64-bit value) and then the transceiver is turned on and the transceiver is set to transmit mode. The next state is TX-POLL_WAIT_SEND. This state causes the first message in the synchronization process to be prepared. Also, the synchronization repetition interval is set to a constant value which will be changed after executing the drift compensation algorithm for the first time. The drift compensation increases this interval if possible to reduce the frequency of the four way message exchanges.

If the device is an anchor, same assumptions are made as in the tag case with the difference that the address is simply copied without caring about the 64-bit format.

If the device is the coordinator it simply tells the transceiver to enter receiving mode after loading the device's address from local memory (again in EUI 64-bit).

The addresses for anchors and tags are chosen on an incremental base starting from one and changing one number in the prefix to distinguish between tag list and anchor list. The prefix is furnished by the manufacturer of the boards.

4.3.2 Pair-wise exchange phase

As stated previously, now all the devices are in the correct mode of operation and the tag and anchors are ready to start their message exchange with the coordinator by sending the first poll.

In this phase no collision avoidance or collision detection techniques were considered; as the number of devices involved is really low a simple guard interval is specified by the coordinator in the last report message to each device telling it when to start entering in the TDOA phase. This guard interval (with a value of 10s) was chosen so that it is long enough to withstand some collisions and permit all the devices to conclude their synchronization phase.

The total time to exchange the four messages has been measured by printing on the screen the arrival time of the report for each device is obtained via the debugging serial port and has been always under 15 *ms*.

With this in mind it is possible to give a visual representation of the synchronization phase in Figure 4.7.

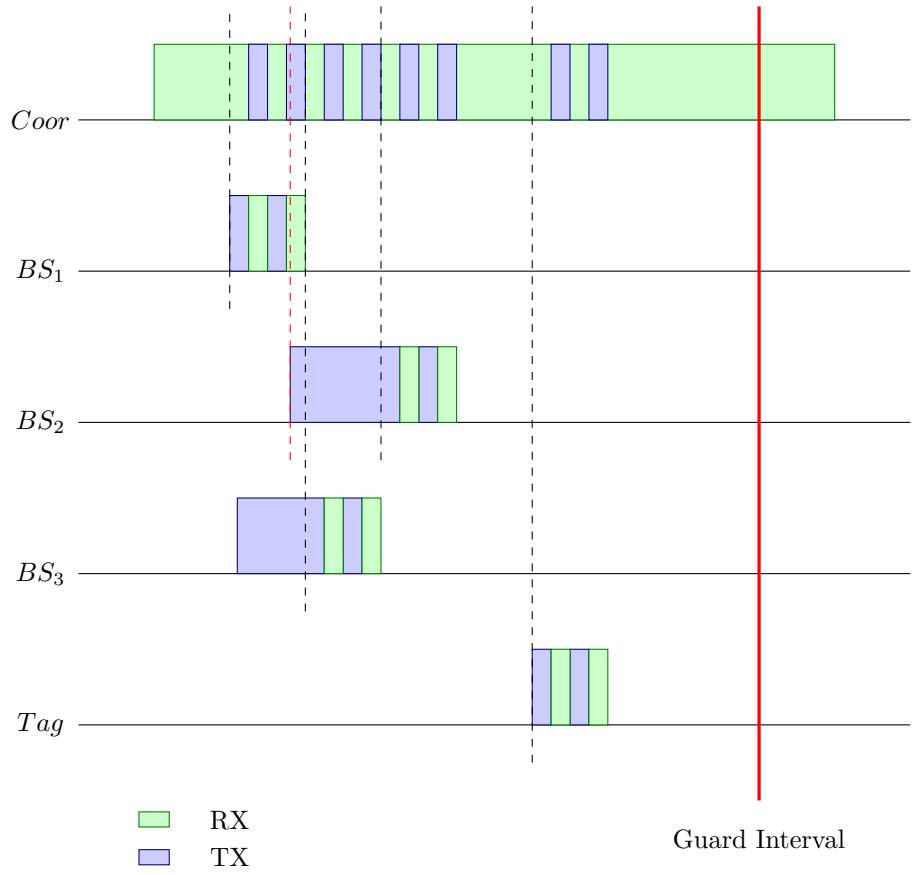


Figure 4.7: Pair-wise exchange phase

The first poll received by the coordinator determines the beginning of the procedure to guarantee synchronization with the device from whom this poll message was received.

4.3.2.1 Coordinator-anchor communication

The poll packet is prepared when the anchor is in TXPOLL_WAIT_SEND state. Once the frame is ready the anchor enter into a confirmation state (TX_WAIT_CONF) in which the previous state is used as a parameter to determine the operations to perform.

At the first step the previous case is TXPOLL_WAIT_SEND, that means, the anchor first timestamps the poll message transmitting time then the following application state is set equal to RX_WAIT in order to ask the transceiver to enter into receiving mode and waiting for the response from the coordinator.

On the coordinator side, upon the arrival of a packet, the frame is processed and the next operation to perform is determined by checking the MFC in the user payload. As it should be, in this case the MFC is 0x21, which means a

poll has been received, so a device is asking to join the network and at the same time to start synchronization procedure.

The coordinator is currently now in RX_WAIT_DATA mode and, considering that a poll has arrived, first the poll message received time is timestamped and stored in the local memory, secondly, the source address of the incoming packet is stored in order to use it as destination to which the response should be sent.

The application state is set to TX_WAIT, so that the device can ask the transceiver to switch to transmitting mode, whilst the next state is set to TXRESP_WAIT_SEND in which a new packet is created and sent back to the device which made the polling request.

Once the frame is ready, the coordinator, similar to the anchor enters the confirmation mode; checking the previous state (which is now TXRESP_WAIT_SEND) the coordinator timestamps the response transmitting time and save this timestamp in its local memory and sets the next state as RX_WAIT in order to reenter receiving mode and waits for the final message.

The communication occurs in exactly in the same way on the anchor side, the final transmitting message time is timestamped and as soon as the packet is sent, the anchor returns to receiving mode waiting for the report while the coordinator goes back to transmitting mode and afterwards to the TXREPORT_WAIT_SEND state.

At this step the frame is prepared according to the description in Section 4.2.4.

If we assume this anchor is the first one asking to join the network (despite the possibility that the tag got a reply sooner) the time slot assigned to it is 1. The time slots are created as a vector of numbers, each one representing a period 15 *ms* long (this choice has been made in order to guarantee future synchronization procedures can occur within a time slot if the sync interval is controlled so that synchronization is never lost but just needs to be adjusted; however a future TDOA phase does not need 15 *ms*, due to its one-message exchange). In ANSI C the first cell of a vector is at offset 0. In this configuration, the timeslot 0 is reserved to the tag to initially transmit TDOA information firstly and all the other timeslots are assigned incrementally to each anchor in order of appearance.

Once this procedure is completed the anchor receives this frame and has the possibility to compute its offset by exploiting the six timestamps it possesses, as well as to perform clock drift compensation, both operations are explained in the following section.

Thanks to the guard interval and to the offset adjustment, the anchor is able to enter SLEEP mode and save energy until it is waken up (at the end of the guard interval) in receiving mode to hear the the tag transmit a TDOA frame.

On the coordinator's side the state after TXREPORT_WAIT_SEND is set again to RX_WAIT by the confirmation mode checking, as mentioned earlier, the previous state. In this way the coordinator is able to hear other poll requests if construction of the network still going on, or simply listens to the channel for the TDOA frames coming from the tag or the other anchors, if the guard interval is over (note that with this configuration the coordinator never enters in SLEEP state).

In the particular case the final report is lost, as said the communication restarts with the anchor sending a poll to repeat the procedure, not having received the necessary information to compute the offset. In terms of vector

at the coordinator side, the values of the elements will be all set to "occupied" (putting 1 as a flag in the incremental element of the vector), thus, the new request from the same base station will be processed by checking the address associated with the occupied element in the vector. If a same address is detected, the coordinator will notify the report was lost previously and will associate the same element to the base station.

4.3.2.2 Coordinator-tag communication

The coordinator-tag communication flows in exactly in the same way with the only difference that the slot number communicated in the last report message is always 0 if the tag asks to join the network.

Note that this scenario foresees usage of only one tag, therefore only the 0 position is reserved; if more tags are deployed in the network then the algorithm must be improved to furnish each tag with a timeslot but still guaranteeing that all tag messages are sent before any anchor TDOA data.

If any packet is lost due to radio interference, errors in the frame, or because of a collision, in any state the devices find themselves, whether it is an anchor, a tag, or a coordinator, or if it is the final message which is lost or a poll which was never answered, then a timeout in each device tells the transceiver to enter the mode assigned to it during the initialization phase, which is RX_WAIT for the coordinator and TX_WAIT for anchors and tag. The next mode will be TXPOLL_WAIT_SEND. The procedure is started again and again until this initialization phase is completed and it is time for the synchronization phase.

4.3.3 Synchronization phase

The synchronization algorithm is written following the specifications described in Section 3.3.

All the operations are performed using 48-bit numbers. The appropriate functions were written to cope with this requirement. In particular 48-bit addition, subtraction, multiplication, and division routines were developed. In some operations numbers are converted back to 64-bit format due to the impossibility of performing long division in the other way, for example this happens when performing the last division showed in equation 3.5.

This division, even in 64-bit, returns un unsigned integer which leads to the inability to know if the offset is negative or positive. For this reason operations are done gradually, computing first the estimated ToF, then the theoretical ToF, and finally the two differences at denominator, everything in 48-bit.

Afterwards, the two values in the numerator and the two differences in the denominator are compared and two flags are set in order to keep track of the sign of the numerator and the sign of the denominator before the final division is performed.

Once the final operation is done, these flags are compared in a double conditional *if*, to determine the sign of the quotient. Depending on the sign the offset is used to apply the correct adjustment to the clock and the drift compensation function is called passing these values.

if $compare48bit(estTOF, teoTOF) == 1$ **then**
  $TOFflag = 0$

else
  if $compare48bit(estTOF, teoTOF) == 2$ **then**
    $TOFflag = 1$
  end if
end if
if $compare48bit(diff1, diff2) == 1$ **then**
  $DIFFflag = 0$
else
  if $compare48bit(diff1, diff2) == 2$ **then**
    $DIFFflag = 1$
  end if
end if
$offset = (estTOF - teoTOF)/(diff1 - diff2)$
if $(TOF == 1 \ \& \ DIFFflag == 0)||(TOF == 0 \ \& \ DIFFflag == 1)$
then
  $sign = neg$
else
  $sign = pos$
end if

The first time the drift compensation algorithm is called it simply stores the offset just computed in a variable called "average offset". It returns a static synchronization interval to the device, chosen based on reasonable criteria.

The second time synchronization is performed by a device (this is determined by means of a simple flag), when the drift compensation algorithm is called, it computes a new average using equation 3.6. Depending on how this value differs from the earlier before and from the recent value, the algorithm determines if it is necessary to decrease the synchronization interval or if it is possible to increase it.

The new interval is expressed in terms of how many rounds of time slots a device can wait before losing synchronization (keeping in mind its position in the vector). This is easily computed because the slots are defined to be 15 *ms* long and the number of devices is equal to 4 (excluding the coordinator which does not need a slot), so for this particular case a four-cell vector is used (which corresponds to 60 *ms*).

4.3.4 TDOA phase

The TDOA phase starts when the guard interval is over and all the devices are synchronized. The communication now takes place in the assigned slots where the tag is assigned slot 0 (hence, it is always the first) and the anchors are assigned incrementally to slot 1, 2, and 3.

At this time, the coordinator is in the RX_WAIT state, while the other devices, were in SLEEP mode until an instant before, thus the tag wakes up in TX_WAIT state and all of the anchors wake up on the RX_WAIT .

The first message transmitted from the tag via the network is an empty message sent with destination broadcast (0xffff). After this message is sent the tag goes to SLEEP mode until the next round starts (in 60 *ms*).

Next the first anchor, assigned time slot 1, enters transmitting mode, then the TDOA_WAIT_SEND state and prepares a packet including the address of

the tag and the receiving timestamp and sends it, until confirmed to the coordinator.

In the meanwhile, the coordinator remains in receiving mode and the other anchors are put into SLEEP mode for 15 ms, 30 ms, etc. depending on their slot number.

The second anchor wakes up and repeats the procedure and so on until all 4 timestamps are collected by the coordinator.

The above sequence is depicted in Figure 4.8.

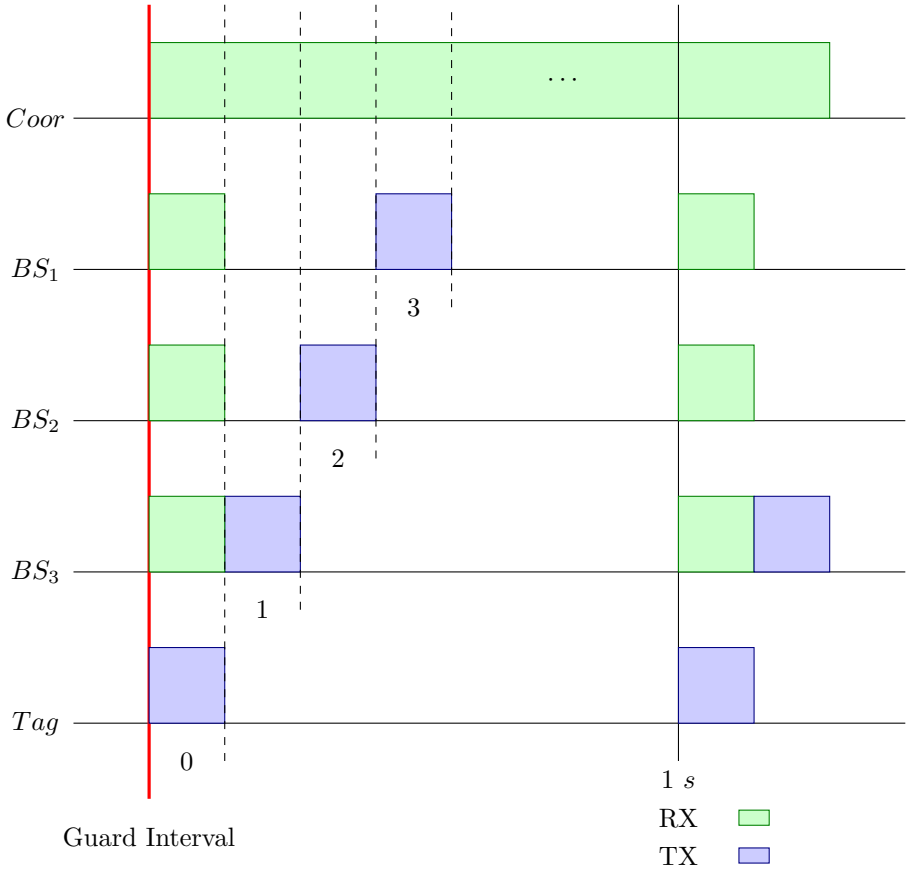


Figure 4.8: TDOA phase

The coordinator now sends the 4 timestamps to the server via an Ethernet connection. The server computes the position of the tag based on this data using the TDOA algorithm.

These procedure is repeated indefinitely in order to track with high accuracy the tag's movements. If a node is told to repeat synchronization, then the slot is used to do so and a jump in the tags position will be perceived at server side. The same happens if one packet or more of the four anchor's results is lost for other reasons.

Chapter 5

Analysis

This chapter presents an analysis of the proposed synchronization and ranging protocol with a focus on the debugging of the system and its performances, plus considerations concerning the synchronization accuracy and the power consumption due to the operations which were executed.

5.1 Debugging

As mentioned before, debugging was possible in two ways: one via software and another one via a JTAG interface. The later directly access the program stored in the chip. We can print values on the laptop screen using software such as Putty [59].

Printing on the screen is done by adding *printf* calls at critical points in the code.

In particular, printing was performed at the moment of transmission and reception and includes the addresses of the devices communicating and the MFC of the packet in order to see how many packets were lost lost and how the process described in Chapter 4 was working.

A set of experimental observations led to the discovery of two main issues: packet loss and the time required for the calls to *printf*.

5.1.1 Packet loss

First, the communication between anchors was really poor. Packet loss was really high and it was impossible to understand whether the problem was due to the radio technology or the software implementation.

It was observed that the entire four message exchange was completing successfully only once out of tens poll requests; sometimes even worse. The result was that the nodes were not able to finish the synchronization procedure within the guard interval. In Table 5.1, a summary of the experimental observations with different network configurations is shown.

Table 5.1: Experimental observations

Devices distance (m)	Number of restarts	Average time (s)	Average poll number	Poll pick number
1	20	6.52	34	62
1	50	5.77	28	79
1	100	8.93	51	64
2	20	7.03	37	65
2	50	4.59	31	56
2	100	8.25	44	81
5	20	5.49	29	69
5	50	6.28	38	72
5	100	7.21	42	53
10	20	9.23	47	67
10	50	11.71	51	62
10	100	7.45	32	54

Figure 5.1 shows a visual description of the data collected. The points have been fitted with a linear line a R^2 coefficient of 0.7255 has been found. This means that approximately, values for the time spent before concluding synchronization procedure, even in presence of high packet loss rate, can be predicted with a 73% of probability.

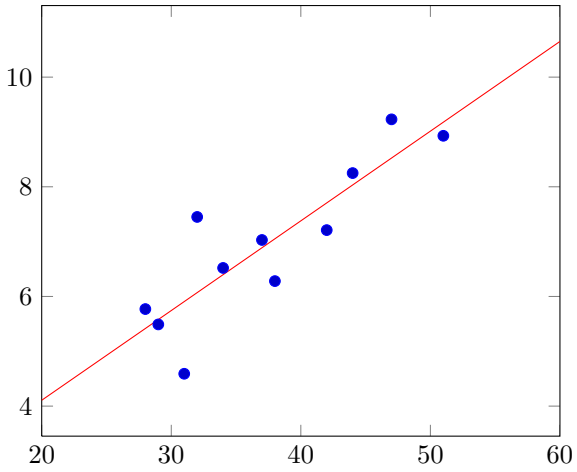


Figure 5.1: Average poll number versus average time

The *average time* was measured by averaging over all the network restarts the total time elapsing before completing the four-way exchange. To do this computation a simple chronometer was used. The number of polls was indicated by adding a *printf* line every time the anchor sent a packet with MFC code 0x21.

As can be appreciated from the table, these data are not very useful and do not seem to depend upon the distance between the devices.

On the other hand, communication between tag and anchor was essentially perfect and the four-way exchange was completed before the TDOA phase. The frame loss was measured to be under 0.8%. When repeating an experiment similar to the one described above. Usually when the poll frame from the tag was received by the coordinator, the processing go fine until the end, so the biggest contribution to the frame loss was due to the poll frame loss.

Collisions could be an explanation, but it is quite unlikely due to the small number of devices deployed in the network and the good performance of the coordinator-tag link. Another reason could have been related with the length of the timeout set at receiving mode used to wait not too long for a packet; this possibility was excluded because the measurements presented above were carried out with the maximum value possible for the timeout; therefore, unless still the packet cannot be received within this timeout, in which case there is nothing to do, the timeout is not a reason for having such a poor communication.

5.1.2 *printf*

The second issue is correlated directly with the *printf*. It was clear after performing some experiments (such as those above), that adding a *printf* to the code decreases the performance of the MAC communication. It was not possible to understand how and why this was happening but when more *printf*s were added to the code the packet loss increased, probably due to the additional amount of time spent by the chip processing the software; in the limit causing the expiration of the timeout in receiving mode (clearly this explanation is possible only if the poll was received at coordinator side, the timeout in fact was not an issue when the coordinator is constantly waiting in receiving mode). All the *printf* were called in a busy wait loop.

Table 5.2 shows a correlation between the number of *printf* calls in the code and the time spent by the devices before getting the report packet and completing the pair-wise communication. Each time is an average over 100 trials.

Table 5.2: *printf* behavior

***printf* number**	**average time (s)**
1	4.63
2	5.22
3	8.45
4	10.43
5	18.97

When one *printf* is used, it is placed in the RX_WAIT_DATA state portion of the code to check which packet has been received. When two are in-

cluded, one is place in the same position as before while the other before the first transmission (either TXPOLL_WAIT_SEND for the BSs and the tag or TXRESP_WAIT_SEND for the coordinator). If three *printf*s are to be exploited the third one is inserted before the second transmission (either TXFINAL_WAIT_SEND for the BSs and the tag or TXREPORT_WAIT_SEND for the coordinator). The forth *printf* is placed in the portion of code before the transmission of a TDOA message (i.e, in TDOA_WAIT_SEND). The fifth *printf* was added to obtain more precise results concerning this problem and was placed in the *main* function which calls the MAC layer implementation, printing the ID of the device in which the code was running.

Figure 5.2 shows a representation of the data above illustrated in a graph. The points have been fitted with a quadratic polynomial and a R^2 value of 0.9555 was found. It means with the increasing of number of *printf*s, the time spent before getting the report message can be predicted with 95% of certainty.

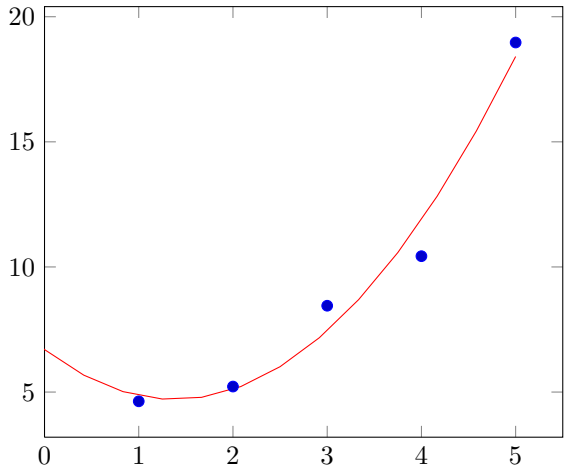


Figure 5.2: Average *printf* number versus average time

As can be seen the printing function negatively affects the performance quite a lot and especially when there are more than 2 *printf*s the synchronization procedure becomes really long. Moreover, it is not possible to see this from the table, but many times the procedure was not even performed due to the transceiver stopping.

Consequently the decision to stop the debugging with many printing functions and therefore it was impossible to investigate deeper in the problem presented in the previous section.

The serial link between Putty and the device was set two different speeds (first 9600 and then 11200) but results were similar in both measures. Optimization to print a string for Keil's compiler was applied and printing on the screen was appreciated to be fast upon packet reception.

5.2 Synchronization accuracy

The two problems presented above created a major obstacle for the evaluation of the synchronization accuracy and the testing of the proposed algorithm.

The algorithm was chosen due to the high redundancy of timestamps used to compute the offset which leads to greater precision than for all the other protocols presented in Chapter 2.

Many papers have been written to compare these synchronization protocols and find the best one (an example of such a comparison can be found in [66]) and the results are not always satisfactory depending on the application which needs to be supported and the size of the network.

The algorithm proposed here aims to get a synchronization error of the order of 0.1 *ns* and to use UWB as radio technology for getting position accuracy down to few *cm*. From the standard [**?**], with a chipping rate of 499.2 MHz it is possible to have maximum confidence interval of 100 *ps* which scaled by a factor of 1/2 give a maximum overall confidence interval of about 50 *ps* which represents also the smallest error in a timestamp value for a range.

Moreover, the vendor can exploit crystal characterization by choosing smaller units of measurements represented by the Least Significant Bit (LSB) of the timestamp (these value is set to a maximum of 2 *ns* in the standard but it is encouraged the choice of increasing it to reduce ranging error and have better ranging accuracy). This means that the LSB could reach a nominal value of:

$$LSB = \frac{1}{499.2MHz * 128} = 15.65ps \tag{5.1}$$

From a time of flight point of view this means the UWB signal, within this time, can travel:

$$d = 3 * 10^8 m/s * 15.65 * 10^{-12} s = 4.695mm \tag{5.2}$$

With this data in mind it is clear that applying TDOA, using 4 timestamps, it is possible to reach a position accuracy in the order of some *cm*.

Considering how SDS-TWR is designed, the algorithm implemented, takes advantage if knowing the distance between the devices, already computes an error that can occur between the frequency offset of the two clocks, therefore the precision depends only on the representation of the timestamps in a 48-bit scale and on the clock drift which is stated to be 2 *ppm* by the manufacturer data-sheet.

From [72] ranging measurements are performed to determine the distance between two devices in two different conditions: when the devices are in LoS and when they are not in LoS. The results show that in LoS environment it is possible to reach an accuracy of 25.6 *cm* when doing ranging between two devices at a distance of 38.96 *m*. Thus, it is reasonable to say that using the proposed algorithm, based on SDS-TWR, accuracy in the order of *cm* could be reach even when looking for localization of the tag in terms of position coordinates.

The only reason untrue values could arise is when exploiting drift compensation to increase the synchronization interval.This drift compensation algorithm averages the most recent offset and the previous value. Depending on how much it differs from the previous computation the synchronization interval is changed. This could lead to mistakes with the consequence of losing synchronization, thus necessitating restarting the network.

Unfortunately, further evidence of the behavior described above could not be achieved due to the fact that the real system was relying only on the usage of three base stations (the fourth was under repair), the connection with the server was not yet implemented and the devices SLEEP mode and interrupt handling were not yet supported in the current release (the manufacturer states in the data-sheet that a full operating device will be ready for the end of year 2013, beginning of year 2014).

5.3 Power consumption

Another problem related with the deployment of this algorithm regards its power consumption.

In the actual configuration all the devices are supposed to be powered by mains power hence no problem arises if the algorithm is working in ideal conditions (excluding the strange behavior described in Section 5.1).

Again, the aim of this accurate synchronization must in any case meet some power consumption requirements in order to be ready battery powered nodes, mainly in the tag, but likely also in the base stations. It is wanted for marketing purposes to build a system in which also the base stations are packaged devices that could be simply attached to walls or ceilings without the necessity of any additional cable.

As already mentioned, this algorithm is probably too expensive in terms of power consumption, but we can not be certain because the choice of the battery was still under discussion and the final configuration was unclear.

While it would be possible to eliminate running this algorithm on the tag side (this implies the modification of the time slot procedure for it, in fact this configuration was chosen due to the impossibility of waking up the base stations at random times when the tag wants to communicate, not having an interrupt handling mechanism available on the BS board), considering that the tag does not need high accuracy or even need much synchronization, we could implement a system in which synchronization is implemented for the coordinator-anchor links.

The final decision if to use battery powered base stations or mains power ones remains.

If we take the four-way exchange duration time as 15 ms and we divide it into two receiving periods and two transmitting periods, checking the data-sheet (provided by the manufacturer) for the transmitting and receiving current and considering a supply voltage of 3.3 V (again from the data-sheet), we get a final power consumption of:

$$E = 2V \cdot 7.5ms(I_t + I_r) = 2 \cdot 3.3V \cdot 7.5ms(17.8mA + 52.6mA) = 3.663mJ \quad (5.3)$$

No information about discharging time of the battery is given, but this value could be taken into account when making a decision about whether to consider battery powered BSs. When trying to implement this pair-wise exchange on some commercial batteries, it was possible to see that no battery analyzed and chosen to fit in the design package of the base station was able to withstand the procedure and have enough time to recharge for subsequent communications, thus impeding the life extension to some years.

Moreover the batteries were tried with the poor conditions described in Section 5.1, thus we assume that by jointly researching on more battery models and improving the communication performance a base station will be able to perform this algorithm for synchronization.

Chapter 6

Conclusions

This chapter presents some final considerations concerning the design, deployment, and evaluation of the proposed algorithm. It also illustrates the aspects that limited the results and how these problems could be solved in the future.

6.1 Goals

The MAC layer design is able to guarantee communication between all the devices in the network and supports UWB technology according to the IEEE 802.15.4a amendment.

The synchronization algorithm, based on the redundancy of timestamps given by the SDS-TWR, the ranging measurements done in [72] and the possibility of knowing the position of the anchors at deployment stage could achieve accuracy in the order of 0.1 *ns* and guarantee localization of the tags with high precision.

Moreover, with the higher level of accuracy in synchronization, from a theoretical point of view it could be possible to have larger synchronization intervals with the consequence of reducing the communication rate of the packets exchange and permitting long sleeping period if a tag requires updates less often than each second (in a normal scenario the object associated with the tag could be inactive for long periods of time), thus helping the possibility of exploiting battery powered base stations.

The time slots system avoids the possibility of collisions once the initialization process has terminated and therefore avoids this problem when there are few devices connected to the same coordinator. Additionally, the system of time slots is easily configurable by changing the length of the vector and computing the total time to configure the network so that the device can be told when they should wake up.

For a large-scale scenario, the method proposed seems to support a good level of synchronization and achieves it jointly with the construction of the DODAG, which reduces the number of messages exchanged during the initialization phase and supports techniques preventing malfunctions in case of a node failure.

The implementation is built on an open source operating system, which guarantees complete support in case of development difficulties and the possibility for a low cost, differently from a private stack implementation like ZigBee for

which it is necessary the purchase of a license.

Without considering the big advantage of having complete compatibility with the Internet protocols, which permits connection with other kinds of network and makes the system accessible from any laptop connected to the Internet (which suggest that there needs to be security considerations concerning signaling to and from the BSs and the server). This necessity of Internet integration was already suggested by the ZigBee Alliance itself [67].

6.2 Issues to be solved

In the analysis chapter some issues have already been raise.

Apart from the battery design, which was out of the scope of this thesis and was supposed to be done in advance in order to know the constraints and regardless of the testing of the localization accuracy, which would be easy once the Ethernet connection between the server and the coordinator is deployed, there remains some issues regarding the packet loss rate that need to be solved.

It is necessary to understand the core of the problem, why packet loss is so evident only for the communication between M3-based devices?

The debugging through *printf* suggest that it is impossible to proceed in this way in order to investigate more and better over this issue, especially now that the code is so long and problem could arise at any point.

Better idea would be to write a short simulation code, restarting from the bottom, at physical layer, and testing it step by step by building a small MAC layer with minimum functions to capture the problem font at a very low level and then address the right solution before running the all system with this characteristics. Or otherwise to use the local memory of each device to record data and dump this data after (without using *printf*s).

6.3 Required reflections

The MAC layer designed in this thesis project provides better understanding of how UWB technology can be used in a network environment, in which way a MAC layer with synchronization algorithm can be implemented and which is issues are related with its implementation. The proposed solutions address potential benefits to both operators and end users in terms of low manufacturing cost and small package devices as well as predicted high localization accuracy to offer the user a precise real time service for the tracking of assets.

Furthermore, the thesis project propose an implementation for a synchronization algorithm to exploit in a large-scale scenario which can be included on an open source operating system like Contiki and perform synchronization procedure together with routing routines through RPL, depicting the possibility of building an entire protocol stack for UWB using international standards and guaranteeing interoperability.

Considering the limitations in the work, the problems analyzed and the good foundation for the proposed future work orientate the reader towards a more trustworthy solution.

Chapter 7

Future work

This final chapter introduces possible areas of future work, either improvements to the decisions already made or aspects that were outside of the scope of this thesis project but that could be useful for the realization of a complete system that could be introduced to the market.

7.1 What has been left undone?

Throughout the report some aspects has been highlighted as critical and these need future study in order to overcome them.

7.1.1 Testing of the position accuracy by connectivity with the server

Once the packet loss issue is solved and synchronization proven to work according to the simulation's prediction, then localization accuracy should be tested to demonstrate position resolution at the *cm* level.

A server ready to receive four timestamps is already implemented thanks to the thesis project of a Jesper Köhl [68]; this server provides TDOA functions in order to calculate the actual position of an asset. Next step would be to implement a small function able to convert the MAC frame in IEEE 802.15.4 format into an Ethernet frame within the MAC layer designed, using the payload definitions expected by the server implemented as mentioned above.

7.1.2 Design of batteries able to support the system

The power consumption analysis gives an idea of the energy need for the system to run until synchronization is achieved.

Thus, it is important to find a proper battery able to withstand the transceiver operation for all the duration of the four-way exchange. It is a difficult task, especially on the tag side, where the space available for batteries is really small, therefore it is likely a future change of the algorithm for the tag is needed remembering that no high accuracy synchronization is needed for this element of the network.

7.1.3 Evaluation of clustered TPSN through simulation

The proposed clustered version of TPSN with all the suggested enhancements should be tested by means of simulation and it should provide results at least similar to the algorithm implemented here for a small scenario.

Moreover, the clustered TPSN power consumption performance must be proved to be better than the one proposed in this thesis in order to at least make it possible to also utilize batteries on the anchor side.

The aim of the future deployment is, in fact, to leave only the coordinator with a direct mains powered supply, due to its connection with the server, but exploiting battery powered base stations which could be easily deployed in any environment without the necessity of cables.

7.1.4 Development of the Network Layer using Contiki

Once TPSN is proven to have sufficient accuracy, it should be implemented in Contiki together with a modified version of RPL able to support both DODAG construction and the synchronization procedure.

This necessitates the modification of the DIO and DAO packets by including TPSN information and the design of a third packet sent as an acknowledgement to the DAO message with the scope of collecting the four timestamps required by TPSN to compute the offset.

Furthermore, handover in 6LoWPAN should be analyzed to see if it is ready to support the mobility of a large number of tags within the UWB radio coverture or if it needs further modifications.

7.2 Areas uncovered

This report did not focus on a complete simulation for the large-scale scenario and is limited to a proposal for a future design.

In this particular case, simulation by means of mathematical tools would be really useful, but it would be probably better to use a simulation tool designed for these purposes and with the description of hardware specifications for different chip models already included.

Cooja [69] is a network simulator embedded in the Contiki operating system. It offers the possibility of implementing different kind of WSN to analyze performances at the network level exploiting the Contiki system functions. It can utilize many commercial devices (such as the MSP430) and offers a wide choice regarding network topologies to be studied.

It would be really interesting to add a UWB physical layer definition within Cooja to permit the simulation of this kind of system whose characteristics are still not considered in the tool.

Finally, this document never mentioned security as an issue but, as in any kind of network, nodes could suffer an attack by external devices. This topic is matter of study in WSNs nowadays. As suggested by Victor Ariño Perez in his master's thesis [70] where an encryption model was studied to be embedded in Contiki.

References

[1] K. Ashton, "That 'Internet of Things' thing", *RFID Journal*, 2009-06-22

[2] X. Shen, M. Guizani, R. C. Qiu, T. Le-Ngoc, "Ultra-wideband wireless communications and networks", *Wiley*, 2006

[3] A. F. Molisch, P. Orlik, Z. Sahinoglu, J. Zhang, "UWB-based Sensor Networks and the IEEE 802.15.4a Standard - A Tutorial", *Mitsubishi Electric Research Laboratories*, 2006-10

[4] F. Ramirez-Mireles, "On the Performance of Ultra-Wideband Signals in Gaussian Noise and Dense Multipath" *IEEE Trans. Veh. Technol.*, vol. 50, no. 1, pp. 244–249, 2001-01

[5] F. Ramirez-Mireles, "Signal Design for Ultra-wideband Communications in Dense Multipath", *IEEE Trans. Veh. Technol.*, vol. 51, no. 6, pp. 1517–1521, 2002-11

[6] R. C. Qiu, H. P. Liu, and X. Shen, "Ultra-Wideband for Multiple Access", *IEEE Commun. Mag.*, vol. 43, no. 2, pp. 80–87, 2005-02

[7] IEEE Computer Society, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)", *IEEE Standard 802.15.4a^{TM}-2007*, 2007-08-31

[8] Jin Ding, Li Zhao, Sirisha R. Medidi, Krishna M. Sivalingam, "MAC Protocols for Ultra-Wide-Band (UWB) Wireless Networks: Impact of Channel Acquisition Time", *School of EECS, Washington State University, Pullman, WA 99164-2752*

[9] IEEE Computer Society, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)", *IEEE Standard 802.15.4^{TM}-2007*, 2011

[10] Holger Karl and Andreas Willig, "Protocols and architectures for wireless sensor networks", *Wiley*, 2005

[11] E. Anceaume and I. Puaut, "A Taxonomy of Clock Synchronization Algorithms", IRISA Research Report No. PI 1103, 1997

[12] Jeremy Elson, Lewis Girod, Deborah Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts", *Department of Computer Science, University of California, Los Angeles*

[13] S. Ganeriwal, Ram Kumar, M. Srivastava, "Timing Sync Protocol for Sensor Networks", in *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys, 2003, p. 138, available at http://portal.acm.org/citation.cfm?doid=958491.958508

[14] Miklós Maróti, Branislav Kusy, Gyula Simon, Ákos Lédeczi, "The Flooding Time Synchronization Protocol", Institute for Software Integrated Systems, Vanderbilt University

[15] H. Dai, R. Han, "TSync: A Lightweight Bidirectional Time Synchronization Service for WirelessSensor Networks", *ACM SIGMOBILE Mobile Computing and Communications Review*, pp. 125–139, 2004

[16] M. L. Sichitiu, C. Veerarittiphan, "Simple, Accurate time Synchronization for Wireless Sensor Networks", *Proceedings of Wireless Communications and Networking 2003 (WCNC)*, pp. 1266–1273, 2003-03

[17] J. So, N. H. Vaidya, "A Distributed Self-Stabilizing Time Synchronization Protocol for Multi-Hop Wireless Networks", *Technical report, Department of Electrical and Computer Engineering and Coordinated Science Laboratory, University of Illinois at Urbana-Champaign*, 2004-01

[18] J. v. Greunen, J. Rabaey, "Lightweight Time Synchronization for Sensor Networks", *Proceedings of the 2nd ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2003-09

[19] C. Savarese, J. Rabay, K. Langendoen, "Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks", *Proceedings of the Annual USENIX Technical Conference*, 2002

[20] V. Ramadurai, M. L. Sichitiu, "Localization in Wireless Sensor Networks: A Probabilistic Approach", *Proceedings of 2003 International Conference on Wireless Networks (ICWN 2003)*, pp. 300–305, 2003-06

[21] Zafer Sahinoglu, Shinan Gezici, Ismail Guvenc, "Ultra-wideband Positioning Systems: Theoretical Limits, Ranging Algorithms and Protocols", *Cambridge University Press*, 2008

[22] Genming Ding, Zhenhui, Lingwen Zhang, Ziqi Zhang, Jinbao Zhang, "Hybrid TOA/AOA Cooperative Localization in Non-line-of-sight Environments", *Beijing Jiaotong University*

[23] Li Cong, Weihua Zhuang, "Hybrid TDOA/AOA Mobile User Location for Wideband CDMA Cellular Systems", *IEEE Transactions On Wireless Communications*, vol. 1, no. 3, 2002-07

[24] N. J. Thomas, D. G. M. Cruickshank, D. I. Laurenson, "Performance of a TDOA/AOA Hybrid Mobile Location System", *University of Edinburgh*

[25] N. Kushalnagar, G. Montenegro, C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", *RFC 4919, Internet Engineering Task Force*, 2007-08

[26] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", *RFC 4944, Internet Engineering Task Force*, 2007-09

[27] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", *RFC 2460, Internet Engineering Task Force*, 1998-12

[28] M. Degermark, B. Nordgren, S. Pink, "IP Header Compression", *RFC 2507, Internet Engineering Task Force*, 1999-02

[29] S. Casner, V. Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", *RFC 2508, Internet Engineering Task Force*, 1999-02

[30] M. Engan, S. Casner, C. Bormann, "IP Header Compression over PPP", *RFC 3544, Internet Engineering Task Force*, 2003-07

[31] T. Koren, S. Casner, J. Geevarghese, B. Thompson, P. Ruddy, "Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering", *RFC 3545, Internet Engineering Task Force*, 2003-07

[32] R. Finking, G. Pelletier, "Formal Notation for RObust Header Compression (ROHC-FN)", *RFC 4997, Internet Engineering Task Force*, 2007-07

[33] T. Narten, E. Nordmark, W. Simpson, H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", *RFC 4861, Internet Engineering Task Force*, 2007-09

[34] Z. Shelby, S. Chakrabarti, E. Nordmark, "Neighbor Discovery Optimization for Low Power and Lossy Networks(6LoWPAN)", *RFC 6775, Internet Engineering Task Force*, 2012-11

[35] Z. Shelby, C. Bormann, "6LoWPAN: The Wireless Embedded Internet", *Wiley Series*, 2009

[36] D. Johnson, C. Perkins, J. Arkko, "Mobility Support in IPv6", *RFC 3775, Internet Engineering Task Force*, 2004-06

[37] V. Devarapalli, R. Wakikawa, A. Petrescu, P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", *RFC 3963, Internet Engineering Task Force*, 2005-01

[38] C. Perkins, E. Belding-Royer, S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", *RFC 3561, Internet Engineering Task Force*, 2003-07

[39] C. Perkins, I. Chakeres, "Dynamic MANET On-demand (AODVv2) Routing", *Internet-Draft, draft-ietf-manet-dymo-25, Internet Engineering Task Force*, 2013-01-04

[40] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, JP. Vasseur, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks", *Internet-Draft, draft-ietf-roll-rpl-19, Internet Engineering Task Force*, 2011-03-13

[41] ZigBee Alliance, "ZigBee Specification", *ZigBee Document 053474r17*, 2008-01-17

[42] M. Durvy, J. Abeillé, P Wetterwald, "Making Sensor Networks IPv6 Ready", *Proceedings of the Sixth ACM Conference on Networked Embedded Sensor Systems (ACM SenSys 2008)*, pp. 421-422, 2008-11

[43] Jean-Philippe Vasseur, Adam Dunkels, "Interconnecting Smart Objects with IP", *Morgan Kaufmann*, 2010

[44] Chin-Der Wann, Yi-Jing Yeh, "A Hybrid TDOA/AOA Positioning Technique for Indoor UWB Systems", *Department of Electrical Engineering National Sun Yat-Sen University*

[45] "Guidelines For 64-bit Global Identifier (EUI-64) Registration Authority", *IEEE Standards Assocation*, 2010-04

[46] "JTAG Tutorial" available at: *http* : *//www.corelis.com/education/JTAG_Tutorial.htm*

[47] IEEE Computer Society, "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)", *IEEE Standard 802.15.4a^{TM}-2007*, pp. 126-127, 2007-08-31

[48] R. Hach, "Symmetric Double Sided - Two Way Ranging", *IEEE 802.15 documents*, 2005-06

[49] Lee Jian Xing, Lin Zhiwei, Francois Chin Po Shin, "Symmetric Double Side Two Way Ranging with Unequal Reply Time", *Institute for Infocomm Research*

[50] Yan Zhang, "Precise Location Technology Based on Chirp Spread Spectrum", *Shandong Institute of Business and Technology*

[51] Yoonseok Nam, Hyungsoo Lee, Jaeyoung Kim, Kwangroh Park, "Two-Way Ranging Algorithms using Estimated Frequency Offsets in WPAN and WBAN", *Dongguk University, ETRI*

[52] Marcin Brzozowski and Hendrik Salomon, Peter Langendoerfer, "On Efficient Clock Drift Prediction Means and Their Applicability to IEEE 802.15.4", *IHP*

[53] Philipp Sommer and Roger Wattenhofer, "Symmetric Clock Synchronization in Sensor Networks", *ETH Zurich*, Computer Engineering and Networks Laboratory

[54] M. Maroti, B. Kusy, G. Simon, A. Ledeczi, "The flooding time synchronization protocol", *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004

[55] R. Solis, V. Borkar, P. Kumar, "A new distributed time synchronization protocol for multihop wireless networks", *45th IEEE Conference on Decision and Control*, 2006

[56] Ali Burak Kulakli, "Time synchronization in wireless sensor networks", *Master Thesis, Izmir Institute of Technology*, Computer Science, Izmir, Turkey, 2008

[57] Xiaonan Wang, Huanyan Qian, "Constructing a 6LoWPAN Wireless Sensor Network Based on a Cluster Tree", *IEEE Transactions on vehicular technology*, vol. 61, no. 3, 2012-03

[58] Chuan Chin Pu, "Sensing Task Handover for Indoor Clustered Wireless Sensor Network", *Symposium on Information & Computer Sciences*, 2011

[59] "PuTTY: A Free Telnet/SSH Client" available at: *http* : *//www.chiark.greenend.org.uk/ sgtatham/putty/*

[60] Texas Instruments Inc., "Code Composer Studio (CCStudio) Integrated Development Environment (IDE) v5" available at: *http* : *//www.ti.com/tool/ccstudio*

[61] ARM Holdings plc, "μVision IDE & Debugger" available at: *http* : *//www.keil.com/uvision/*

[62] "Contiki: The Open Source OS for the Internet of Things" available at: $http : //www.contiki - os.org/$

[63] Texas Instruments Inc., "MSP430F5438A", Data-sheet and User Guide available at: $http : //www.ti.com/product/msp430f5438A$

[64] ARM Holdings plc, "STM32F107", Datasheet available at: *http* : $//www.st.com/web/catalog/mmc/FM141/SC1169/SS1031/LN1564/PF221020$

[65] ZigBee Alliance, Products and Documentations available at: *http* : *//www.zigbee.org/*

[66] "A Comparative study of Time Synchronization Protocols in Wireless Sensor Network", *International Journal of Computer Applications (0975 – 8887)*, Vol. 36, n.11, 2011-12

[67] ZigBee Alliance, "ZigBee Alliance plans further integration of internet protocol standards", 2009-04-27

[68] Jesper Köhl, "Positioning algorithm and positioning server for a positioning system", *Examensarbete, KTH*

[69] "Get Started with Contiki", Cooja tutorial available at: *http* : $//www.contiki - os.org/start.html$

[70] Victor Arino Perez, "Efficient Key Generation and Distribution on Wireless Sensor Networks", *Master's Degree Project*, 2013-03

[71] Anqi Luo, Lei ge, "Indoor Location Detection using WLAN", *Master's Degree Project*, KTH, School of Information and Communication Technology (ICT), Communication Systems, CoS, 2010

[72] Byniam Shiferaw Heyi, "Implementation of Indoor Positioning using IEEE802.15.4a (UWB)", *Master's Degree Project*, KTH, School of Information and Communication Technology (ICT), Communication Systems, CoS, 2013

[73] Senad Canovic, "Application of UWB Technology for Positioning, a Feasibility Sudy", *Master's thesis*, Norwegian University of Science and Technology, Department of Electronics and Telecommunications, 2007

[74] Zafer Sahinoglu and Sinan Gezici, "Ranging in the IEEE 802.15.4a Standard", Mitsubishi Electric Research Laboratories, Cambridge, 2006-10

TRITA-ICT-EX-2013:59

www.kth.se