Synchronous sleep in a wireless mesh network

- using synchronous sleep schedule, in a wireless sensor network for indoor climate control, to save power enabling battery powered network

FREDRIK ISAKSSON

KTH VETENSKAP OCH KONST

KTH Industrial Engineering and Management

Master of Science Thesis
Stockholm, Sweden 2014

Synchronous sleep in a wireless mesh network

- using synchronous sleep schedule, in a wireless sensor network for indoor climate control, to save power enabling battery powered network

Fredrik Isaksson

Master of Science Thesis MMK 2014:81 MDA 494

KTH Industrial Engineering and Management

Machine Design

SE-100 44 STOCKHOLM

KTH Industriell teknik och management

Examensarbete MMK 2014:81 MDA 494

Synchronous sleep in a wireless mesh network

- using synchronous sleep schedule, in a wireless sensor network for indoor climate control, to save power enabling battery powered network

Fredrik Isaksson

Godkänt 2014-09-29	Examinator Martin Edin Grimheden	Handledare Lei Feng
	Uppdragsgivare Tritech Technology AB	Kontaktperson Henrik Jonhed

SAMMANFATTNING

För att styra inomhusklimat behövs det ett styrsystem med återkoppling. Det finns flera olika sätt att återkoppla, men en av möjligheterna är att använda ett trådlöst nätverk med sensorer. I trådlösa sensornätverk vill man alltid spara energi för att få längre batteritid. Det finns flera olika tekniker för att uppnå det. Den vanligaste är att man begränsar tiden radio-delen är aktiv på olika sätt.

Detta examensarbete går igenom vad som är intressant för att styra klimatet inomhus, trådlösa sensornätverk och en strategi för att spara energi genom att synkronisera tiden mellan alla noder, beräkna tidsdriften, och låta dem sova mellan rapporteringen av sensordata. Detta gäller även meshnätverk, där vissa noder vidarebefordrar meddelanden mellan andra noder. Föreslagen som nämns veriferas med hjälp av simulationer i Simulink och TrueTime.

Resultatet visar på att det mycket väl är möjligt att låta noderna sova, även i meshnätverk, för att spara energi. Detta gäller både vid användning av tidsynkronisering som tar hänsyn till tidsdrift och endast tidsskillnaden mellan noderna. En diskussion om nödvändigheten av att kompensera för driften tas också upp; är det nödvändigt att lägga till den komplexiteten jämfört med vad man vinner i energibesparing?

KTH Industrial Engineering and Management

Master of Science Thesis MMK 2014:81 MDA 494

Synchronous sleep in a wireless mesh network

- using synchronous sleep schedule, in a wireless sensor network for indoor climate control, to save power enabling battery powered network

Fredrik Isaksson

Approved 2014-09-29	Examiner Martin Edin Grimheden	Supervisor Lei Feng
	Commissioner Tritech Technology AB	Contact person Henrik Jonhed

ABSTRACT

To control the indoor climate a control system with feedback is needed. There are several different ways to provide feedback; one possibility is to use a wireless network of sensors. In wireless sensor networks, it is always desirable to save energy for longer battery life. There are several different techniques to achieve it: among them the most common one is to limit the time the radio is active in various ways.

This thesis goes through what is relevant to control the indoor climate, wireless sensor networks and a strategy to save energy; by synchronizing the time, and calculate the time drift between all nodes, and allow them to sleep between the reporting of sensor data. This also applies to mesh networks, where some nodes forward messages between other nodes. The suggestions in the thesis is verified with simulations in Simulink and TrueTime.

The result shows that it is very well possible to allow nodes to sleep, even in mesh networks, in order to save energy. This applies for time synchronization that only compensates for time offset, and for synchronization that calculates the time drift as well. A question regarding compensation of the time drift is also addressed; is it necessary to add to the complexity, compared to what you gain in energy conservation?

FOREWORD

Till minne av Stefan Isaksson

1970-05-07 till 2014-03-30

Hoppas du spelar fotboll i himlen.

I’d like to thank Henrik Jonhed, my supervisor at Tritech, for coming with advice along the road. I also want to thank Lei Feng, my supervisor at KTH who has helped me with new ideas and the simulations. Without that help, this thesis wouldn’t have become what it is. In addition, a special thanks to Boe Sjöberg at Tritech who helped with arrangements regarding the thesis.

It has been a long spring, and summer for that matter, but I have gained new knowledge in areas regarding the thesis but also outside when discussing technology with employees at Tritech.

Fredrik Isaksson

Stockholm, 2014

ABBREVIATIONS

CAV	*Constant Air Volume*
HVAC	*Heating, mechanical Ventilation and Air-Conditioning*
MAC	*Medium Access Control*
MPC	*Model Predictive Control*
OCXO	*Oven Controlled Crystal Oscillator*
P2P	*Peer-To-Peer*
PLM	*Product Lifecycle Management*
PMV	*Predicted Mean Vote*
PPD	*Predicted Percentage Dissatisfied*
RDC	*Radio Duty Cycling*
TCXO	*Temperature Compensated Crystal Oscillator*
TT	*TrueTime*
VAV	*Variable Air Volume*
WSN	*Wireless Sensor Network*

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

This master thesis examines the possibility of synchronous sleep in a wireless sensor mesh network by using time synchronization to save power. This is to enable the whole network to be battery powered. The use of this feature can be seen in indoor climate control, where continuous measurement is not always needed but long battery life of the nodes is desired.

1.1 BACKGROUND

A modern indoor climate control system does not always need measurements continuously, but rather has a periodic sample of for example temperature and humidity. This period can be up to one hour, during that time there is no need for the sensors to be awake, especially if they are battery powered. To enable this in a wireless mesh network where nodes are relaying messages, the nodes have to be awake at the same time. That means that the time has to be synchronized amongst the nodes.

By doing so, it is possible to enable a fully battery powered mesh network for indoor climate control. Of course it is possible to implement time synchronisation and synchronous sleep in other systems, if it’s suitable. One example is to monitor (instead of controlling) the indoor environment or buildings. Or in an even wider perspective it can be useful when new technologies, such as internet of things, is being more available and used for applications when periodic data over a long time is desired, not limiting to indoor climate measurements.

1.2 PURPOSE

The purpose of enabling synchronous sleep is to make it possible to let the nodes be battery driven. By doing so one can install sensors in new places and without restrictions due to cables. This might be interesting when retrofitting an older building with a modern indoor climate control system; instead of having to install cables one can place sensors wherever needed.

1.3 DELIMITATIONS

Wireless Sensor Network is a wide field with extensive research in many areas. Therefore it would be difficult to do research in every aspect of WSN, leading to some delimitation in this thesis. Below the delimitations are listed in bullet points and further on described in detail.

THE THESIS WILL NOT INCLUDE

- Sensor evaluation
- Developing time synchronization protocol over wireless network
- Developing network construction or reconstruction
- Evaluating different hardware
- Minimizing time awake

The final hardware would use some sensors (ex. temperature, humidity) but the performance of the sensors is not crucial for this thesis, thus is not within the scope of the thesis to evaluate sensors.

Even though time synchronization was crucial for this thesis it is a field of research on its own. Due to complexity it is not included in the thesis to examine the performance of time synchronization used compared to other protocols. Network constructing and reconstruction is excluded to keep the complexity at a reasonable level.

Power management on hardware level is a topic on its own and to keep the scope within a reasonable limit, different hardware (ex. transceivers, micro controllers) will not be evaluated even though it is an important part of the whole system.

The thesis will concentrate on maximizing the time asleep rather than trying to minimize the time awake. However further development should consider both aspects to maximize the performance regarding power usage.

1.4 METHOD

To answer the question if it possible to enable a battery powered wireless sensor network with mesh capability for feedback in indoor climate systems; a study of existing research within relevant areas has been conducted. This is to create an understanding of what is crucial for the success of the thesis. In addition it creates a base of what technologies can be used and an understanding of indoor climate control and what requirements there are on such systems. This is accounted in chapter 2.

Simulations was done in Simulink (with TrueTime) to identify critical points in the system, especially in comparison with the requirements. This is to gather knowledge of how an eventual implementation should be done and what is required from the hardware.

2 FRAME OF REFERENCE

This section is divided into three parts. First part will go through generally about indoor climate, and how to control it. The second part will go through wireless sensor networks (WSN) and last synchronous sleep is explained and what is needed to achieve that.

2.1 INDOOR CLIMATE

Keeping temperature and humidity within a certain range is essential for comfort. There are standards that regulate what climate that is suitable depending on several factors. ASHRAE 55 and ISO 7730 are two example, they were developed in parallel [1] [2]. The factors are for instance the obvious one; temperature. In addition relative humidity and air velocity are important aspects. However it is not only these three elements that need to be considered. ISO 7730 mentions factors such as thermal balance of human body as a whole. This is influenced by aspects such as clothing and physical activity.

Clothing and physical activity can vary from person to person. Therefore ISO 7730 have something called predicted mean vote (PMV); that is what temperature is most suitable depending on factors as metabolism, mechanical power, clothing, air temperature and relative airspeed amongst other factors [2]. Since this can vary between person to person it is also possible to calculate predicted percentage dissatisfied (PPD), predicting how many will feel it is too hot or too cold. ASHRAE 55 uses these terms in the same way.

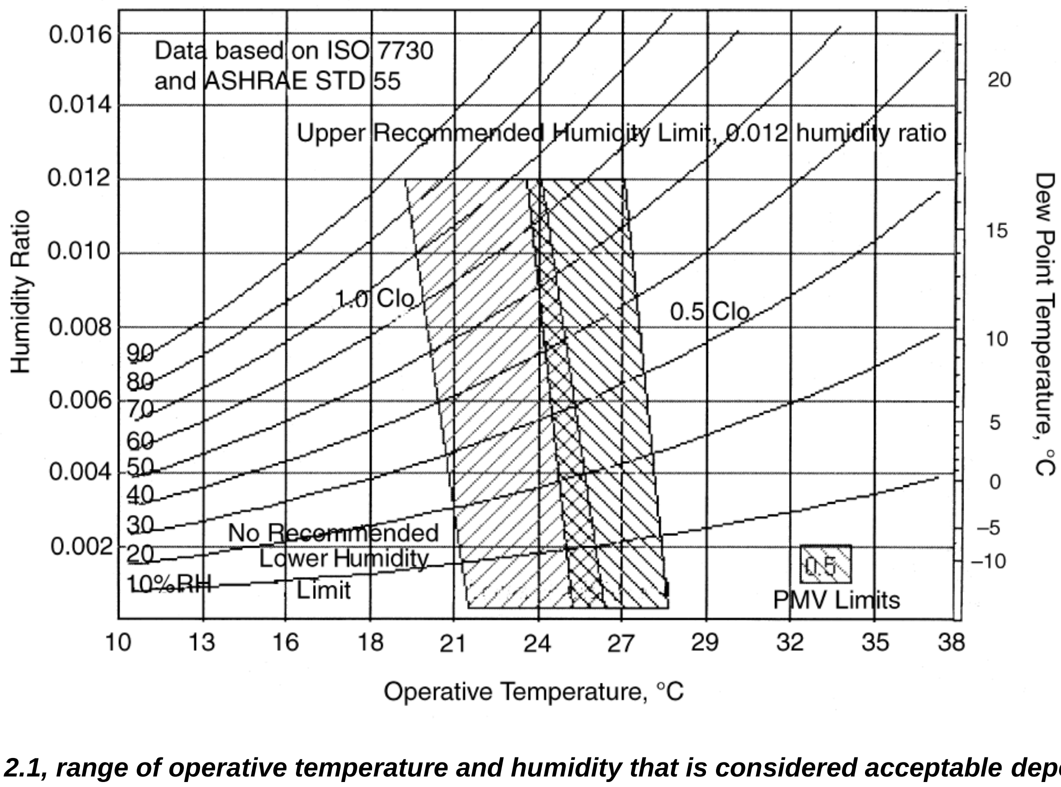

Figure 2.1, range of operative temperature and humidity that is considered acceptable depending on clothing. [1] (dew point temperature sv. Daggpunktstemperatur, RH is abbreviation for relative humidity)

ASHRAE 55 has an example with a certain limit on air speed and clothing. Figure 2.1 shows the recommended temperature interval depending on humidity and clothing. There are two types of intervals depending on the clothing insulation (1.0 Clo and 0.5 Clo); these are typical factors of clothing when the outdoor environment is cool respectively warm [1]. To be able to achieve this some control of the environment indoors is needed. This is explained in section 2.1.1.

2.1.1 CONTROL

To keep a comfortable climate indoors, it is needed to be able to affect temperature and humidity [1]. In addition CO2-control can be desirable [3]. There are different ways of controlling the climate indoors, especially temperature. By using a thermostat the heating can be on or off, depending on the temperature, and has a deadband that is for example 1˚C [4]. This might not be suitable for bigger areas since the heating will not be activated if the thermostat does not indicate a heat demand. Therefore the heating might not be activated even though it is cold in a room nearby [5].

Another way of controlling the temperature is using a sensor placed in the return air path. It is mentioned that this method is not taking the radiant heat felt by the occupants into regards of the control and they may or may not feel comfortable [6]. It can be a thermostat placed in the return air path or a temperature sensor connected to a controller using PI/PID. It is mentioned that on/off based and PI/PID control is mostly used due to simpler implementation and tuning [7]. Note however that it is not that common to use PID controllers in heating, mechanical ventilation and air-conditioning (HVAC) since they (PID controllers) cannot reflect the effects of the outside temperature. In addition there are difficulties with implementing a PID controller when the system has multiple-input and multiple-output [8].

Buildings account for around 40% of the total energy use in Europe it is therefore a necessity to use intelligent control of the energy for the future energy system [9]. There is the possibility of using model predictive control (MPC); however that concept has some drawbacks mentioned, such as computational power required and the need of a mathematical model of the building itself. The control could use inputs such as temperature outside, weather forecast data, and room temperature error. By using MPC it was possible to achieve savings of 17-24% [8]. Using MPC with an economic factor as well saves money by start heating in advance or postpone the heating depending on the energy price. By doing so it is possible to do economic savings around 35% [9].

A common system for controlling the indoor environment is HVAC. It uses the temperature of the air to heat or cool the environment. At first it was mostly constant air volume (CAV) that provided cool, dry, still indoor air that was considered as the thermal comfort at the time. However, during the oil shock in the seventies a more energy efficient system was needed and variable air volume (VAV) was introduced [10]. Later on it has been identified that heat pumps combined with water based floor heating systems will be one of the main sources for heating buildings [9].

Depending on the control and system used, different sample time is needed. There is a vast range of suggested sample times, one suggests a sample time between 1-15 minutes and uses 5 minutes in the end [7] and in addition one uses 15 minutes [11]. Furthermore [9] uses 30 minutes and [12] uses an hour. Much depends on how the system works. As [13] mentions, some systems use the thermal inertia of the building itself. Other systems may control the heat of the air, and [7] mentions the time constant of the indoor air as 27 minutes and [9] mentions it as 1 hour. As comparison [9] states the thermal time constant of the floor as 186 hours and [13] mentions numbers between 24 and 300 hours depending on the type of the building and how old it is. So the sample time is heavily depended on the type of system.

To gather this data some sort of system (network) is needed. This thesis will look into how the use of a wireless network can be used in this context. One of the reasons is the cost and complexity of installing a wired network when retrofitting a building. Another cause is the possibility to monitor the environment for a while even though it is not part of the feedback of the control loop.

2.2 WIRELESS SENSOR NETWORK

Wireless sensor networks (WSN) are sensors that cooperatively relay information between the nodes of the network to a receiving node or gateway. These networks use a mesh network which is a point-to-point network. These networks have the advantage of being fault-tolerant and reconfigurable. However, a significant amount of overhead is attached with setting up the network and relaying information long distances.

For WSNs to be practical, they would ideally be self-sufficient in power. While many sensor-devices rely on batteries, these devices eventually have to be recharged, changed, or even wasted when the battery runs out [14]. Some authors [15] propose using innovative techniques to prolong battery-lifetime by using for example solar power to recharge the battery. There is several radio duty cycling protocols for lowering relaying nodes power consumption [16] [17] [18]. More of this is mentioned in section 2.2.2.

In particular, one improvement to the current state-of-the-art in mesh net-protocols is command the sensor to sleep at a given time for a given duration. This could potentially decrease static-power consumption dramatically. Recently, synchronous sleep protocols have had success in decreasing static power consumption on WSNs [19].

There are two types of (physical) structures when it comes to WSN [14], structured and unstructured. A structured WSN is when the nodes are deployed in a planned manner. By doing so one can gain advantages such as better network coverage and lower network maintenance and management costs. As opposite, an unstructured WSN is when the nodes are spread within an area and create an ad-hoc network. This is often left unattended and used for a large numbers of nodes. Therefore it is difficult to perform network maintenance and management.

Depending on where the network is deployed the numbers and type of nodes that is needed can vary. An open indoor environment may require fewer nodes than an indoor setting with plenty of walls and other obstacles. An outdoor environment may require more nodes due to the size of the area that the nodes are deployed in. Regarding of the type of the nodes; depending on how harsh environment it is (especially indoors versus outdoors) different types of casings is needed to protect the circuits [14].

2.2.1 MESH

Mesh network is a network topology allowing nodes to send messages via other nodes. By routing through several nodes the message propagates to the addressed node. This is an advantage if the power of the radio transmissions are limited or it is within an area with many obstacles like walls. A typical mesh network consists of two types of devices: mesh nodes and router nodes where the routers have capability of relaying messages and routing functions for the mesh network. One could say that routers form the backbone of the mesh network [20].

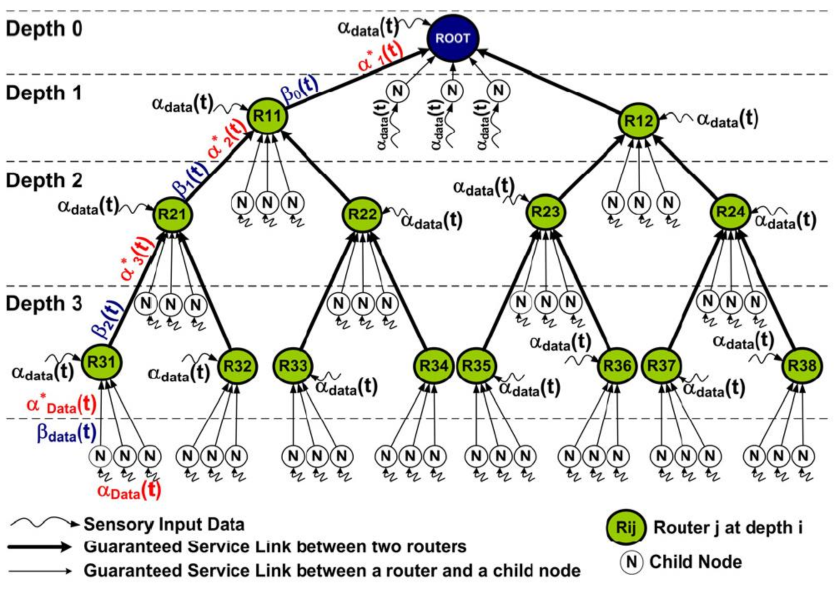

Figure 2.2, cluster tree mesh topology [21]

An example is ZigBee. It can form mesh networks connecting numerous (hundreds to thousands) nodes. ZigBee uses three kinds of devices; coordinator, router and end device. The coordinator initiates the network formation and can bridge the network. ZigBee routers relay messages from end devices further, thus making the mesh networking possible. The ZigBee end devices contain sensor(s) and actuator(s), and can only communicate with either a ZigBee router or coordinator [14]. This topology is called cluster tree and is illustrated in Figure 2.2.

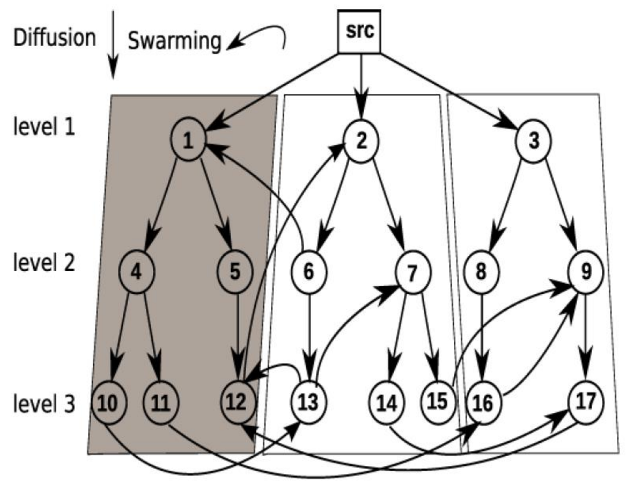

Figure 2.3, peer-to-peer mesh network, not all connections are shown, such as all cross communication between the nodes [22]

Another topology for mesh networks is peer-to-peer (P2P). Instead of having end devices with reduced functionality as in ZigBee the end devices have router functionality as well. By doing so, it is possible for all nodes to communicate with each other, as shown in Figure 2.3. This reduces the network to two types of nodes, coordinator/gateway and routers/nodes. Looking at the definition of what P2P network is:

> *"A distributed network architecture may be called a Peer-to-Peer (P-to-P, P2P, ...) network, if the participants share a part of their own hardware resources (processing power, storage capacity, network link capacity, printers, ...). These shared resources are necessary to provide the Service and content offered by the network (e.g. file sharing or shared workspaces for collaboration): They are accessible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource (Service and content) providers as well as resource (Service and content) requestors (Servent-concept). " [23]*

However, the same author mentions both pure- and hybrid P2P networks. For a network to be pure P2P one must be able to remove a terminal (node) and the network shall not suffer any loss of network service. A hybrid network is if the network has a central terminal that is crucial for network service [23]. Due to the fact that (1) a WSN might have a coordinator and/or gateway (as ZigBee) and (2) it is wireless and might suffer from loss of network service if any terminal is removed, it is possible to say that a WSN with a P2P topology is a hybrid P2P network.

2.2.2 POWER MANAGEMENT

It is well stated that transmitting, listening or receiving messages is the biggest power consumer in wireless nodes [16] [18] [24] [25]. Therefore it is desirable to reduce the time listening, receiving or transmitting messages. This can be achieved by software. Using algorithms reducing the time the transceiver is active one could save a large amount of power, thus prolonging the battery life significantly. Using a technology called Radio Duty Cycling (RDC) one can reduce the active time of the transceiver by turning it on and off continuously like a PWM signal. This can be done in both an asynchronous and synchronous way.

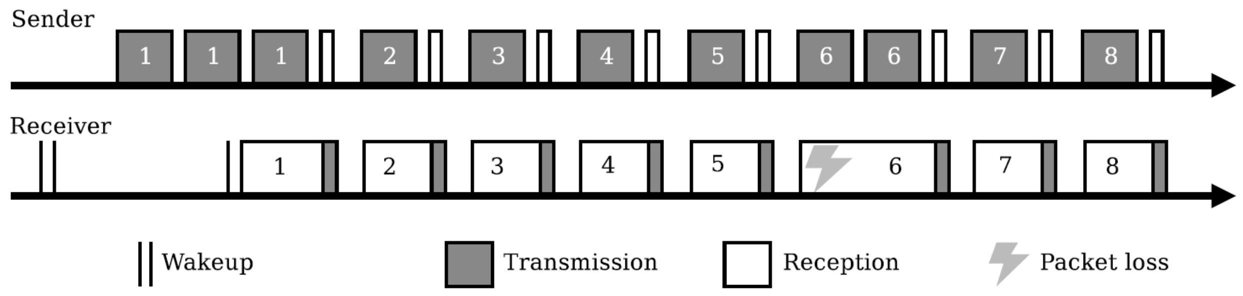

Figure 2.4, an example of asynchronous radio duty cycling, the sender sends multiple packages, with one package lost [26]

An asynchronous way is to have a node trying to send the message multiple times until another node matches it and receives the message. The sending node does this with a certain frequency and the non-sending nodes listen if anything being sent in a certain frequency. Since the sender is sending for an amount of time, in a certain frequency, these two phases will eventually match. When doing so, the listening node will acknowledge the sending node and will thereafter listen to the message from the sender; all this is shown in Figure 2.4. The main issue here is to have a combination of a frequency and duty cycle minimizing the active time, but still allowing the listeners to notice if there is a message being sent within a reasonable timeframe.

As opposite of the asynchronous way a synchronous method is to have specified timeslots to send and listen. This requires some sort of synchronisation and robustness from time drift, often as an offset from wakeup of the transceiver until the rest of the nodes are active [27].

It is noted [17] that using a fixed period of listening is not adaptive to changes in the amount of traffic and is therefore not suitable for an environment with a dynamic flow of information. Using a fixed period for a specified amount of periodic data is not suffering from this. Therefore it exists several radio duty cycling protocols, both for networks with a dynamic amount of data traffic as well as static amount of periodic data. This can be seen comparing S-MAC [27] with PMAC [17]; it shows that with a high load of traffic an adaptive protocol has better throughput which can be crucial in some applications

As mentioned, transceivers use most of the power in a typical sensor node, and it is especially idle listening that is considered as the biggest waste of energy [17]. However there are other events that are considered as waste as well. Collision of packages, that is, multiple nodes are trying to send at the same time, is an obvious waste because there was no information passing through and the nodes have to send again.

Another waste identified by [27] is a node receiving messages not meant for it, this is called overhearing. It is identified that control packages overhead is to be seen as waste since it does not provide useful information and hinders the more useful messages being sent.

These protocols aim to lower the power consumption in WSNs, however it is stated by [14] that depending on what kind of node (ex. ZigBee router/end device, see section 2.2.1) and the kind of power supply, a protocol optimized for sensors (ZigBee end device) may not be suitable for mesh routers. This is due to the fact that usually routers have an external power supply but where end devices are battery powered [18] [20].

2.2.3 CULTUREBEE

An example of WSN in use is CultureBee, developed by Linköping University [28]. The purpose of CultureBee is preserving cultural heritage buildings. It utilizes ZigBee standard for local WSN and is remotely monitored and is measuring temperature and humidity. It is also used to control radiators and dehumidifiers.

The system is composed by three parts. First there is the WSN, consisting of a ZigBee network. Then there is the local server that acts like a gateway between the WSN and web service. And last, the main server, which provides web interface for the user, data synchronisation with local servers. The web interface provides the user with remote monitoring and control. The WSN is based on a ZigBee network, as discussed in 2.2.1, it consists of three types of components; end device, router and coordinator. The routers and coordinator is mains powered. There is also type of node they call control node. This is a router that also can provide ON/OFF switch and 0-10V analogue control signal.

Testing of the power consumption of the end devices showed that it is possible to achieve 3382 days of battery lifetime depending on setup (such as frequency of sensing and reporting data). It is not discussed if it is possible to let the routers be battery powered and what battery time they would have [29].

2.3 SYNCHRONOUS SLEEP SHEDULE

If the WSN does not need to send data continuously but rather have a fixed sample time one could save energy by letting the nodes sleep when no data is sent. This is especially suitable for longer sample time periods. If a node can be asleep for most of the time, one could gain a vast amount of battery time.

This is particularly interesting for the routing nodes in the network. Since a routing node always has to be on (or else it will miss relaying the other nodes messages) but the end nodes, that does not need to relay messages, can sleep whenever they desire. It is mentioned by [18] [20] that in many situations infrastructural nodes (ex. ZigBee routers) have constant power supply and have therefore the possibility to be constant awake. But as stated in section 2.2.2, idle listening is a waste of energy, and if the end nodes are sleeping, the routers should not waste energy, thus sleep as well.

Previous research [19] [30] has implemented a synchronous sleep schedule. This protocol relies on a coordinator broadcasting a sleep command that propagates through the network. However, it is noted that it is for sensor networks in a small scale. Worth mentioning as well is that the sleep command propagates through the network and therefore the nodes will wake up one by one. This might not be the most robust solution if one or more nodes miss the sleep command. The result of missing the sleep command would be that one or more nodes are awake, thus wasting power. If one use time synchronisation within the network and a periodic sleep schedule, a node could miss a time synchronisation event, but still be able to sleep and wake up for the next period if enough overhead and good enough time synchronisation.

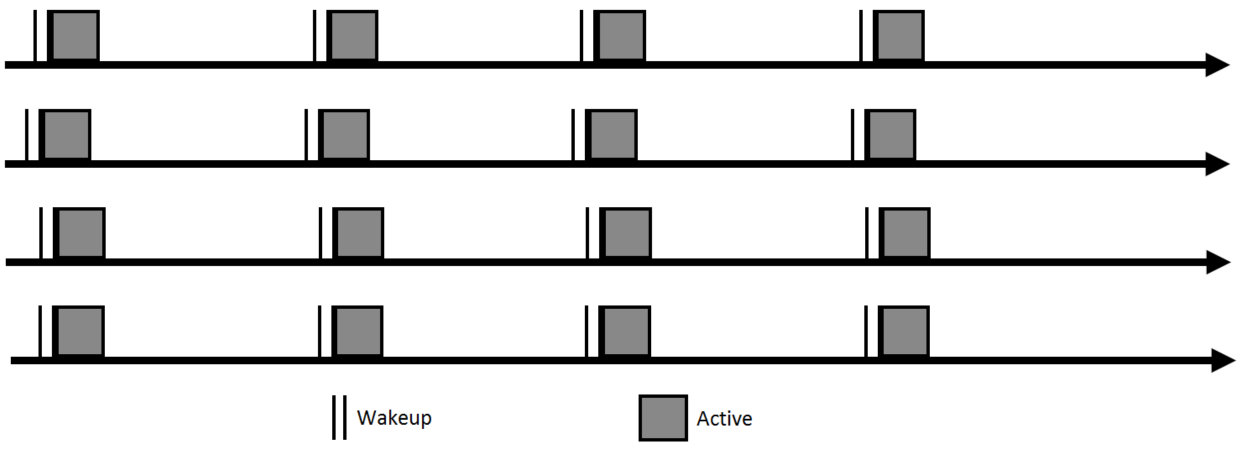

Figure 2.5, synchronous sleep were the nodes wake up at the same time. The scale is not linear, the active time can be much smaller than the time sleeping, note the offset in the wakeup but overlapping active time.

For example; if an indoor climate system only needs updates of the temperature and humidity every hour, it is possible that the nodes could sleep, depending on the throughput in the network, for 59.5 minutes. Thus only being active for half a minute, giving it a duty cycle of less than 1% comparing to RDC that have a duty cycle that varies between 1% up to 66% [16] [18] [31] . In battery life time that turns out to be 77 and 128 days depending on how many packages one are sending per minute according to [32] (each package is 87 bytes). With 1 package per minute the battery life was 128 days, and with 60 packages per minute the corresponding battery life was 77 days. This was with a battery with 880mAh of initial charge.

Another example would be if the nodes messages and synchronisation would take less than 30 seconds. If it would take less than 15 seconds the duty cycle would be less than 0.5%. In addition, it can be of interest if other protocols (such as RDC) can be used in combination of synchronous (long time) sleep lowering the overall duty cycle even more. These examples provide some numbers to get an understanding of the relationship between different technologies. In Figure 2.5 that would be inactive time of 59.75 minutes, and the active part being 15 seconds (note that the scale in the figure is not fitting this example).

To implement synchronous sleep there are two key components identified. One is the capability of synchronizing time between the nodes. The other is the time keeping of each node.

2.3.1 TIME SYNCH

Time synchronization in WSN is an area of research on its own. There is plenty of research within the area due to the importance of having correct timestamps on the messages. This is as a result to the fact that some WSN for example track objects or events and to migrate the data collected the timestamp have to be correct [32] [33] [34] [35].

There are several challenges. Especially when it comes to WSN, mesh networks, a node doesn’t communicate directly to other nodes (except when it is its neighbour), the messages have to jump through several nodes. This leads to that a single node cannot be chosen as a reference that other nodes can synchronize to with direct communication. Another challenge is the risks of a wireless network, e.g. packages can be lost, nodes can run out of battery or other errors that lead to reconstruction of the network [36].

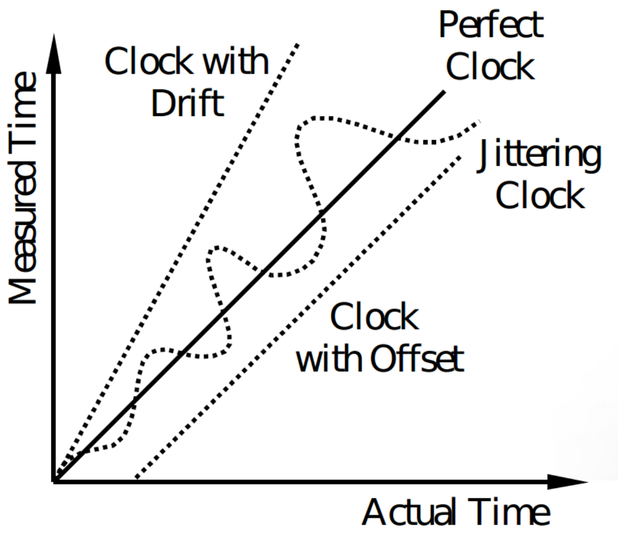

Figure 2.6, graph showing different errors in clock [37]

The time synchronisation issue can be separated into two key components. One is the offset between the clocks and the other is drift. The offset is a static error due to imperfect time synchronization. One can minimize that by using effective time synchronization protocols; however it is not possible to avoid offset in the whole network. Clock drift is a dynamical error that can increase (or decrease) the time difference between two nodes. The difference is shown in Figure 2.6. A time synchronisation protocol minimises the offset but does not always compensate the drift [36].

Network time synchronisation methods rely on message exchange between nodes. By including a timestamp in a synchronisation message the receiving node is able to compare the timestamp with the time when it received the message. Between the sender and receiver there are several sources of error that makes it impossible to exactly compare the two local clocks. There are four identified sources of error in network time synchronisation and are identified as; send time, access time, propagation time, and receive time [35].

- Send time is the time spent constructing the message, switching context in the operating system and sending it to the network interface for transmission.
- Access time is the delay in the network interface, this is dependent on the Medium Access Control (MAC) protocol used and waiting for a possibility to send
- Propagation time is how long it takes between the senders network interface to the receivers
- Receive time is the time it takes for the receiving network interface to receive the message and transfer it to the host

There are several strategies to overcome or bypass these errors. One technique bypasses the send time by broadcasting its synch message and let the receivers compare the time when they received the message with each other. This is called Reference Broadcast Synchronisation [35]. There are more techniques; however it is not plausible to include an acceptable level on the description of these due to the complexity, the mentioned protocol is an example, not an acceptable description.

2.3.2 TIME KEEPING

The clock on each node can't be in total synch due to drifting in the clocks, even if the clocks are perfectly synchronized at the beginning. This is due to imperfections in the clock [32]. However for the synchronous sleep to work in longer sleeps, the drift must be kept to a minimum, thus allowing the node to minimize the overhead during wakeup and thereby increasing the efficiency. Depending on the drift, the nodes might need to wake up earlier (more often than the sample time) to synchronize again.

The accuracy of the clock is described in parts per million (ppm). Common numbers for a Real Time Clock (RTC) crystal oscillator is somewhere between 20 to 30 ppm (i.e. they have an inaccuracy between -30ppm and 30ppm) [38] [39]. In some cases the accuracy needs to be higher. To achieve higher accuracy there are different strategies. Since the crystal oscillating frequency is depended on the temperature some techniques compensate for it.

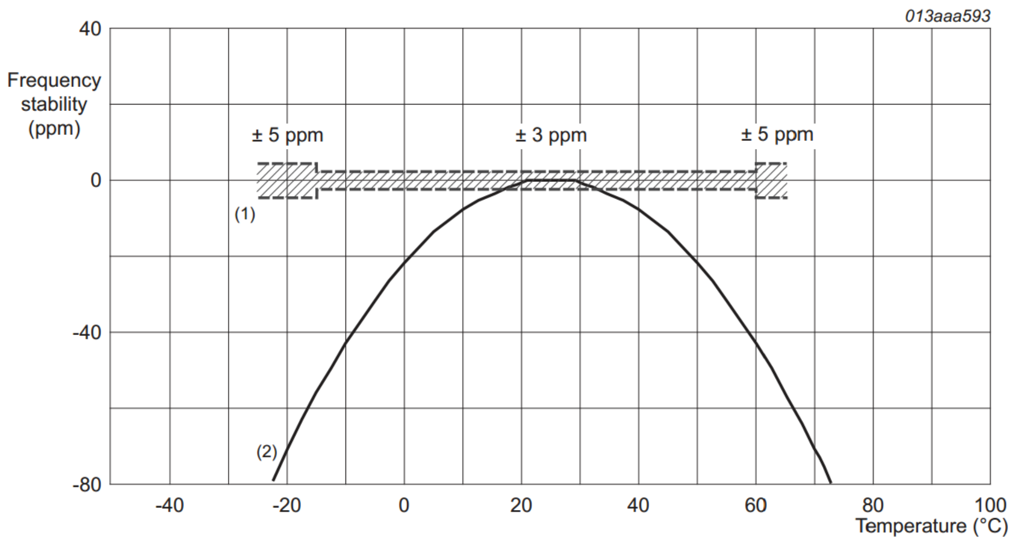

Figure 2.7, (1) shows typical compensated frequency and (2) typical uncompensated tuning-fork crystal frequency [40]

One of them is called Temperature Compensated Crystal Oscillator (TCXO) and there are both digital and analogue compensated. The temperature coefficient is between -0.034 to 0.035 ppm/$^{\circ}C^2$ [41] [42] [43] and by measuring the ambient temperature it is possible to calculate and thus reduce the drift as shown in Figure 2.7 to 2-5 ppm [40] [44] [45] depending on the temperature span. Instead of compensating for the temperature, one strategy is to keep the crystal at a constant temperature, above ambient temperature. This is called Oven Controlled Crystal Oscillator (OCXO). However, this requires power and is generating waste heat. Therefore it might not be suitable for power critical applications.

As a comparison, looking at datasheets for an OCXO gives the numbers 50-170mA depending on the supply voltage and ambient temperature [46]. While a TCXO RTC from the same company have a power consumption of 800nA. Compare this again with products from the same manufactory, an uncompensated RTC draws 150-350nA depending on model [39].

It is possible to compensate the drift with the microcontroller itself if it is possible to measure the temperature. How decent the compensation is depends on how accurate the crystal itself is and how precise the temperature sensor is. Atmel claims it is possible to achieve 5ppm accuracy if the temperature range is within 15°C - 35°C and it is calibrated with an external oscillator with high accuracy, such as an OCXO [47]. Compare it with Figure 2.7 and one realises that the temperature span is rather limited compared to a TCXO. However, Atmel mentions if the temperature sensor is calibrated a higher accuracy is achievable but does not give any number.

3 THE PROCESS

It can be discussed if a thesis such as this should be implemented on real hardware or simulated. The decision of simulating was due to several aspects. Firstly, this thesis is done by one person and the possibility to divide work for preparing implementation and simulation is therefore limited. Another aspect is time, because the goal of the thesis is to look into long sleep times in WSN; it would take far too long time to conduct experiments. The decision was not only based on how time consuming it would be, when doing model based development one does simulation first to investigate if it is possible to implement and optimise before doing it on real hardware.

3.1 EVALUATING SIMULATION ENVIRONMENTS

Before simulating an evaluation of different simulators were done. Three different simulators were evaluated; these were Cooja, TrueTime (TT) and OMNeT++. Each of those has pros and cons and those are presented below.

3.1.1 COOJA

Cooja is a simulator written in Java, there are readymade images of Linux machines that one can install as virtual computer, minimizing the time installing and configuring the environment. Cooja emulates Contiki motes (Contiki is an open source OS), meaning that the C-code written for the motes (nodes) can be simulated before implemented on hardware. Contiki supports mesh network [48]. However, the test simulations that were made with Cooja for evaluation ran at 5-10% speed (the time in the simulation were 10-20 times slower than real life).

3.1.2 TRUETIME

TT is a simulator based on Matlab/Simulink real-time systems. TT supports individual time drift for each of the node, but it is not possible to change it during simulation, as well as offset. It comes with an Ad hoc On-Demand Distance Vector routing example, providing mesh capability for TT. The wireless network part lacks the functionality of turning the receiver off and the fact that listening also drains power, as discussed in 2.2.2 [49].

3.1.3 OMNET++

OMNeT++ is a component-based C++ simulation library and framework for building networks. It is not a network simulator itself but by programming components in C++ and assembling those into larger models using a high-level language one can get advanced network models. The advantage is that one gain reusability. It comes with models of different protocols, wired as well as wireless [50].

3.2 SIMULATION REQUIREMENTS

3.2.1 REQUIREMENTS OF THE SYSTEM

Concluded from the theory, some requirements can be decided. Especially those relevant for the sleep schedule implemented. The most important requirement is how long the nodes should be able to sleep. One hour is the longest time mentioned (see section 2.1.1); therefore the main requirement is for the nodes to be able to sleep for one hour. However, the nodes may miss a time synchronisation event and therefore must be able to handle that as well. To summarize:

- Nodes must be able to sleep for at least one hour
- Nodes must be able to handle missed time synchronisation
- The protocol must not add additional errors

3.2.2 ASSUMPTIONS

Some assumptions and simplification of the simulation are made both due to complexity and time limit. One of those is the influence of the temperature on the drift. As discussed in section 2.3.2, the temperature has influence on the drift; this is not included in the simulations, even though it is possible. To sum up the assumptions:

- The temperature is stable indoors and does not affect the drift
- Nodes does not get faulty, but may miss radio traffic (loss probability)

3.2.3 SIMULATION SCENARIO

A simulation scenario where the gateway is placed in one room with a number of nodes, and another group of nodes are in a neighbouring room. The connection is bridged between the rooms with two nodes. This includes that some node’s messages needs to be relayed by several other nodes to arrive at the gateway and in the same way with the time synchronisation. It also includes the critical connection between two rooms.

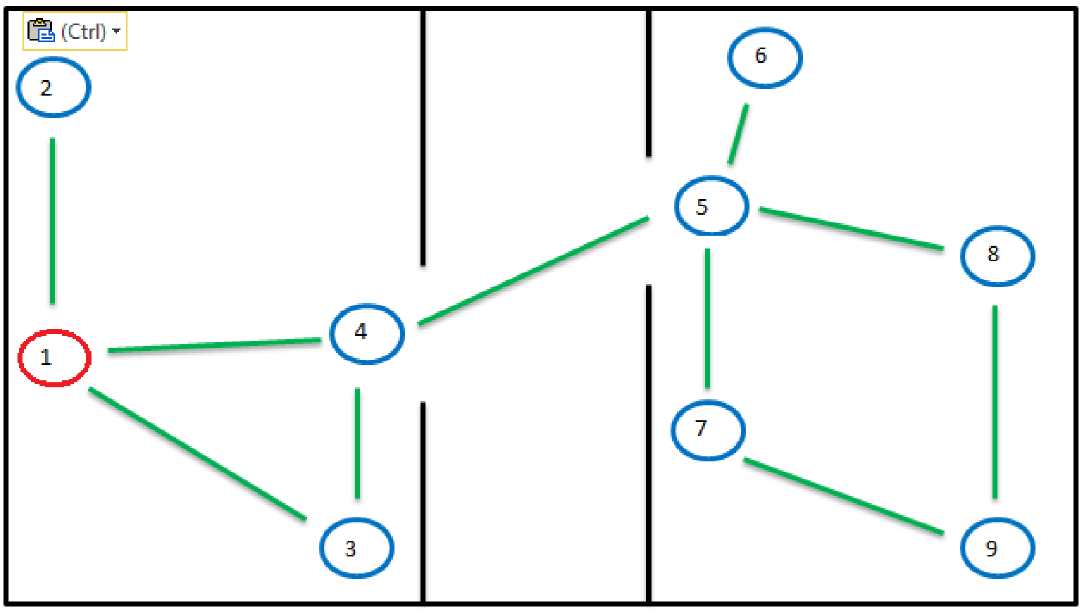

Figure 3.3.1 Network layout where the red ring is the gateway, the blue are nodes and the green lines represent communication ways

In Figure 3.3.1 the connections between the nodes are showed as well as the placement. This is only an imaginary layout; in TT the nodes have been placed so they only are in range with those they should have connection with. This is since TT doesn't have the functionality of including walls and other obstacles in the radio path. Another important factor is the loss probability, in the simulations it is set to 0.05. The sleep time is set to be 4096 seconds after some time (it has shorter sleep time in the beginning to let all nodes get synched properly). 4096 is the exponent of 2 that is closest (but above) one hour. To summarize this:

- Loss probability is 0.05 (5%)
- The sleep time will eventually be 4096 seconds
- The nodes are fixed in position

Message delay is measured to be 2e-4 seconds

3.3 SETTING UP SIMULATION ENVIRONMENT

One of the most important aspects of the thesis is the time drift. Therefore it was decided to use TT as simulation environment. To get a reasonable simulation environment some issues had to be solved. Below these issues and solutions are described. During the time when each solution was developed each was tested to verify that it worked separately, before implementing it into the whole simulation.

3.3.1 DRIFT

An extra layer was added on top of TT, giving the possibility to change the time drift over time. It also includes an offset. Even though not implemented, this layer enables to let the drift be influenced by the temperature. Each node has been set with an offset, to simulate that the initial time is not set correctly, between 0 and 5 second by using the built in random function in Matlab (giving a random number between 0 and 1, and then multiply it with 5). In the same way each of the nodes has individual drift of somewhere between -30 and 30 ppm, to simulate the drift in RTC as mentioned in section 2.3.2. This is also done with the random function so the user does not alternate the factors. The gateway has been set to 0 offset and 0 drift to make it easier to compare, and since it is the node initiating the time synch.

3.3.2 TIMESYNCH

Since TT does not have a function that synchronizes the time between different nodes, this had to be done. Therefore a time synchronization protocol was implemented. The time synchronization implemented takes both the drift and offset into account. This is a one-direction-synchronization, in other words; one node (in this case the one acting as gateway) broadcasts a time synchronization message with a timestamp, this is done every 2 seconds in the simulations. In a real implementation one may want to implement so that the gateway only broadcasts when the nodes are supposed to be awake. This is possible, however to save time, this was not implemented in the simulations.

The receiving node compares the timestamp of the received message with the time when the message was received and adjusts the offset accordingly (with the network delay in mind) and calculates a drift factor. When a node have received and processed a time synch, it will broadcast a time synch message for nodes further away. This leads to that all the nodes will receive a time synchronization. Below is a proof that shows how and why it works. The notation used in the proof is the following

Tx - Time when timesynch message is sent (according to sending node)

Rx - Time when timesynch message is received (according to receiving node)

df - Node's drift factor compared to the gateways time

dfc - Node's calculated drift factor

υ - Static network delay

γ - Dynamic network delay

The static network delay is from building the message, sending it on the physical layer, receiving it and time stamp the message. This takes a certain amount of time. However, there can be a dynamic delay as well, such as the network medium is busy with another node sending data and thus delay when the message is being sent.

If one does not consider the offset, but only the drift, as in a drift factor of 1 would be in perfect synchronisation (considering the time rate). A node with a drift factor higher than 1 would be a node with too high tick rate (the time is going too fast), and vice versa, a drift factor below 1 would be a node’s time going too slow.

Considering this, it is possible to compare the node that is sending the time synch message sending time, and the time (according to the receiving node) when the message is received. This leads to the possibility to calculate a factor, describing the difference in time rate, in other words; the drift factor. Note that the initiating node is the gateway, whose time is considered as perfect (drift factor of 1 from the start). The drift factor can be calculated as shown in (1).

$$dfc_n = \frac{Rx_n - Rx_1}{Tx_n - Tx_1} \tag{1}$$

Where n is number of time synchronisations received. Rx_n can be expressed with, however since there can be disturbances Rx_n is expressed as following

$$Rx_n = (Tx_n + \upsilon_n)df + \gamma \tag{2}$$

Where υ_n is a random delay caused by unpredictable behavior in the network (such as the radio channel is already busy), and γ is a network delay caused by the time it takes from the timestamp is made until the receiving node gets the message and timestamp it. Combining (2) in(1), leads to

$$dfc_n = \frac{((Tx_n + \upsilon_n)df + \gamma) - ((Tx_1 + \upsilon_1)df + \gamma)}{Tx_n - Tx_1} \tag{3}$$

(3) can be simplified to

$$dfc_n = df + \frac{(\upsilon_n - \upsilon_1)df}{Tx_n - Tx_1} \tag{4}$$

If there is no message lost, meaning there is no permanent block of the message, the random variable υ_n is bounded. In addition the message transmission is periodic or sporadic; then the sequence $\{dfc_n \mid n = 2,3,4...\}$ converges to df i.e.

$$\lim_{n\to\infty} dfc_n = df$$

The assumption of no message lost ensures the correctness from Eq.(1) to Eq. (3). If the message is sporadic, then $Tx_n - Tx_{n-1} \geq \delta$, where δ is the smallest gap between two consecutive occurrences. If the message is periodic, then $Tx_n - Tx_{n-1} = T$, where T is the period. In both cases, $Tx_n - Tx_1 \geq (n-1)\cdot\delta$ or $Tx_n - Tx_1 \geq (n-1)\cdot T$.

Therefore $Tx_n - Tx_1$ approaches to infinity when n goes up. The condition of υ_n is bounded then guarantees that the second term in Eq. (4) approaches to 0, i.e. after a long time, the disturbances influence less, and a correct drift factor can be calculated.

3.3.3 MESH

TT does not support mesh network from the start; however there is an example with Ad-hoc On-Demand Distance Vector (AODV) routing protocol. It contains bugs and was only able to let certain nodes send. This bug was traced and solved and the AODV implemented allows cross talking nodes, and multiple node sending messages.

3.3.4 SLEEP/DEACTIVATE NETWORK

TT wireless network lacks some functionality that is important for the simulations in this thesis. It is important that a node cannot deactivate its receiver, limiting when messages can be received. This is explained in the evaluation of TT, see section 3.1. This was solved by adding a flag that is set high when the node is sleeping, and vice versa when it is awake. It does not include acknowledge (ACK) of received messages, in other words; a node still believes it successfully sent a message to a sleeping node. Despite this, the ability to not receive messages (such as time synchronization broadcasts) when asleep is weighted higher.

To disable sending and receiving while sleeping, a global variable was added (unique to every node), and if that was set, the node was sleeping. In the interrupt handler that is triggered by a received message, an if-statement was added. If the node is sleeping, the message was disregarded and nothing else done, if the node is awake it calls the function that handles incoming messages.

When the node awakes it is idle for a while, ready to relay messages, time synch and more. After a random time the node sends its data. The reason for the idle time is to make sure all the nodes are awake before sending data. In numbers this would be; the nodes are awake for example a total time of 4 seconds, using the random function each node will send its data after somewhere between 2 and 3 seconds.

3.3.5 INITIAL SEQUENCE

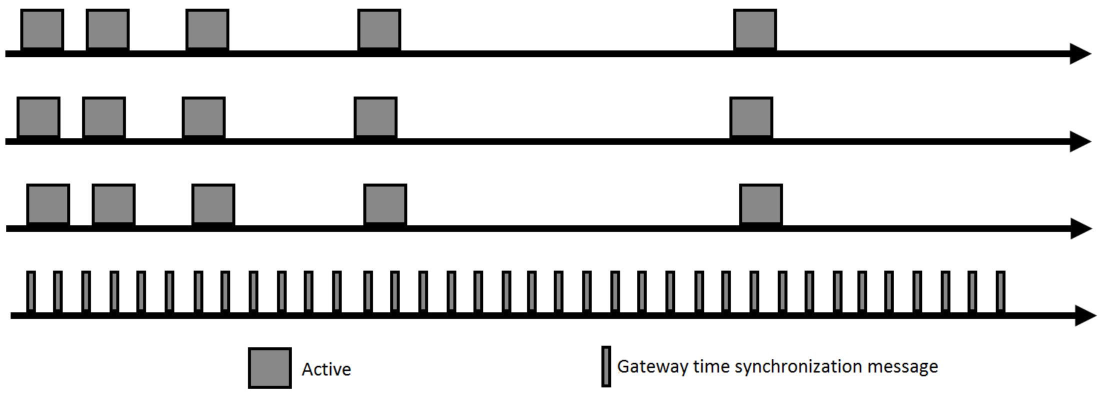

Figure 3.2, initial sequence, principle picture

In the start of the simulation the node does not sleep for one hour. Instead they are awake and waiting for a time synchronization message as shown in Figure 3.2. This is done several times to ensure that all nodes get in synch. Gradually the sleep time is increased by the gateway until it reaches 4096s. By having this initiating sequence the nodes does not have to be in synch from the start.

4 RESULTS

In this chapter the result of the simulations are shown. Three things are investigated; is it possible to let the nodes sleep for one hour and does the drift compensation (not offset adjustment) affects this. How does the drift factor error evolve after a long time? And last, how does the time synch perform in regards of the offset. To show that the result is valid, three runs with different settings are shown, even though more simulations have been done the behavior did not change between the runs. The tables are showing the initial values generated by Matlab using its random function.

4.1 DRIFT FACTOR ERROR

It is shown in 3.3.2 that the error of the drift factor calculations should be minimized after a long time. Simulations were made for a time of five days to see if the error actually converged towards zero.

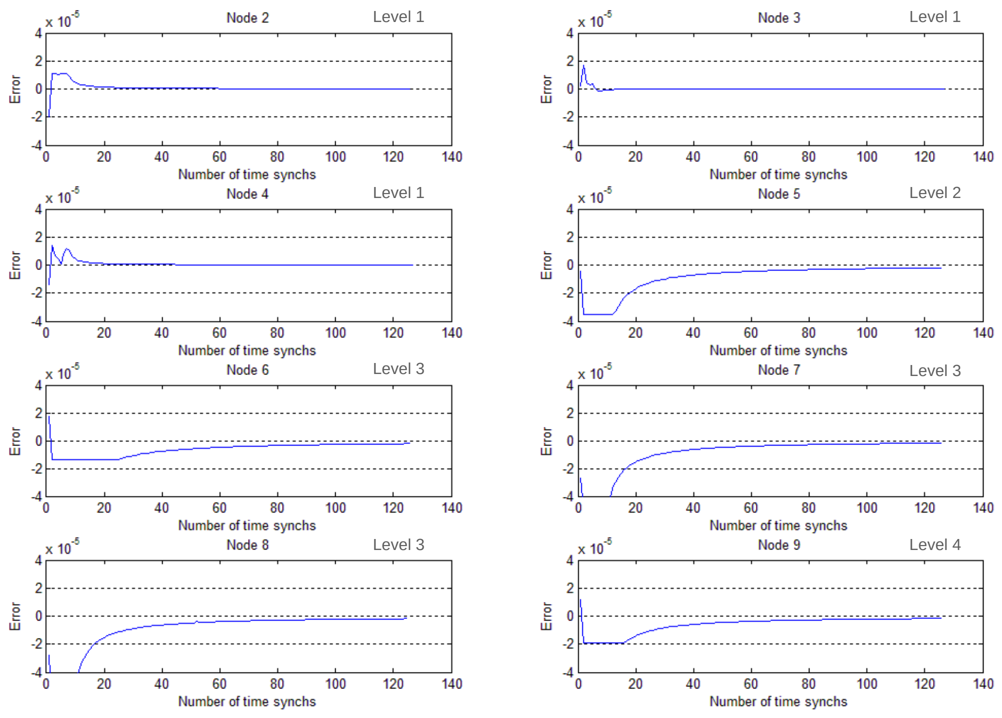

Figure 4.1, first run, showing the drift factor error over number of time synchs

In Figure 4.1 we can see how the drift factor error changes after a number of time synchs. The error in the end (and the initial values) was as following:

Table 4.1, first run, result and initial values

Node	**Last drift factor error [ppm]**	**Time drift factor [ppm]**	**Start offset [s]**
2	0.1653	-19.7835	3.3201
3	-0.0172	2.1511	4.1455
4	0.11604	-13.9583	0.88078
5	-2.0056	-4.1288	2.3783
6	-2.1822	17.1133	0.65326
7	-1.8161	-26.9187	3.1375
8	-1.8050	-28.2547	0.68097
9	1.6752	11.6733	2.5784

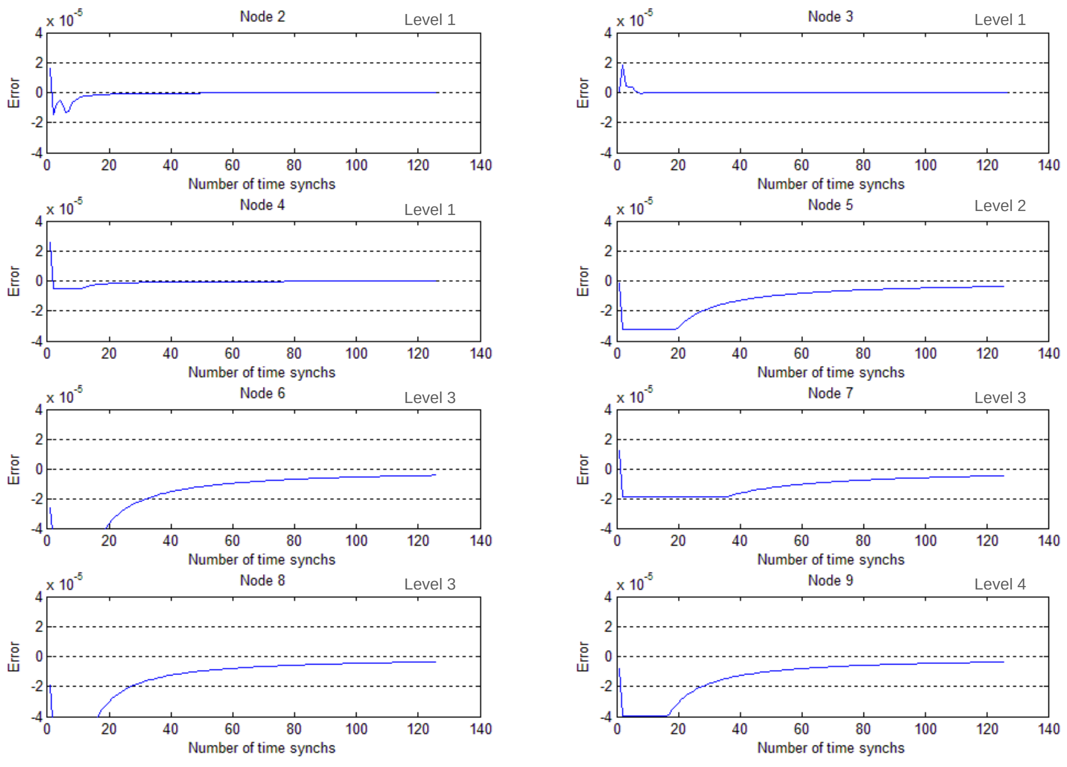

Figure 4.2, second run, showing the drift factor error over number of time synchs

In Figure 4.2, we can see a bit of different behaviour, note how big the error is for some nodes in the beginning. The end result of the second run as well as the initial values was as following:

Table 4.2, second run, result and initial values

Node	Last drift factor error [ppm]	Time drift factor [ppm]	Start offset [s]
2	-0.1331	16.0094	1.5132
3	-0.0052	0.70937	4.4178
4	-0.2160	25.9763	0.43336
5	-3.6908	-1.5599	1.9852
6	-4.3844	-26.2825	3.9466
7	-4.7022	11.9293	3.9572
8	-3.5480	-18.7227	1.599
9	-3.6336	-8.4368	0.16265

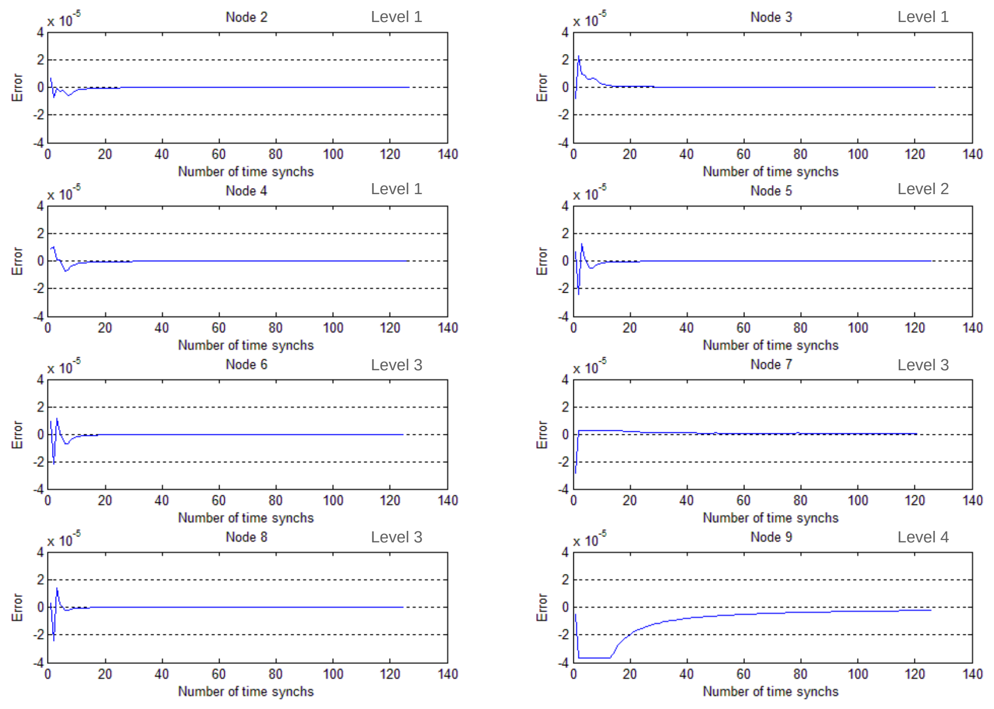

Figure 4.3, third run, showing the drift factor error over number of time synchs

In Figure 4.3 we can see the same overall behaviour as before. The end result and initial values this time was:

Table 4.3, third run, result and initial values

Node	Last drift factor error [ppm]	Time drift factor [ppm]	Start offset [s]
2	-0.0578	6.9577	0.79423
3	0.0669	-7.9582	3.5442
4	-0.0741	8.9985	4.4909
5	-0.0534	6.5078	2.0542
6	-0.0749	9.1023	3.2088
7	0.2370	-28.4307	1.0191
8	-0.0255	3.1483	0.81937
9	-2.3160	-5.4452	4.7437

4.2 TIME SYNCH ACCURACY

Another aspect of the time synch is how accurate it is regarding the offset just when the time synch was performed. In addition the disturbances effect compared to the depth of the network.

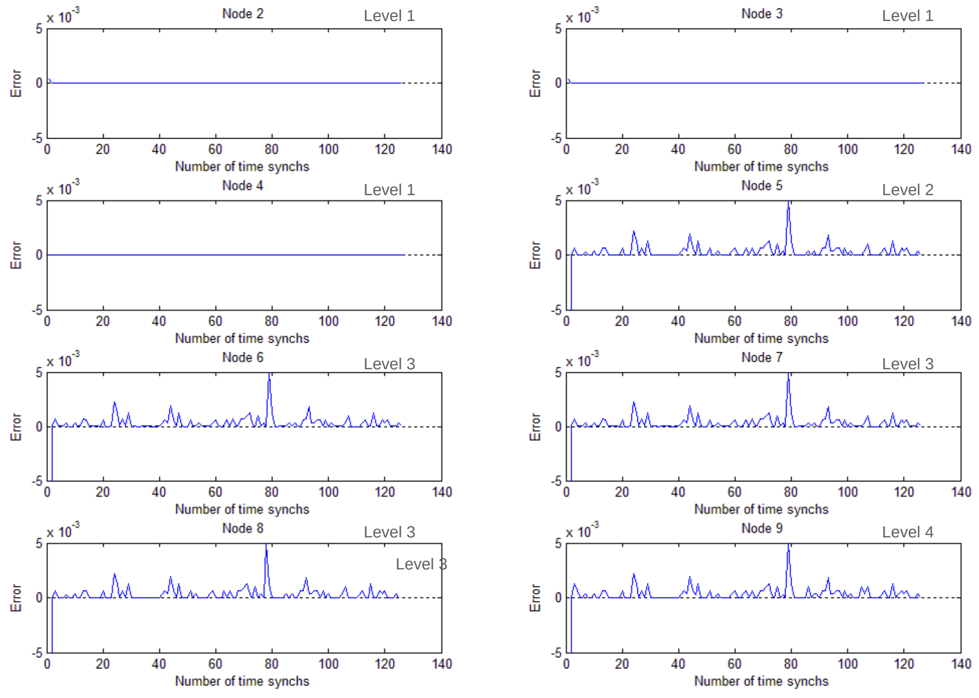

Figure 4.4, first run, the time synch error

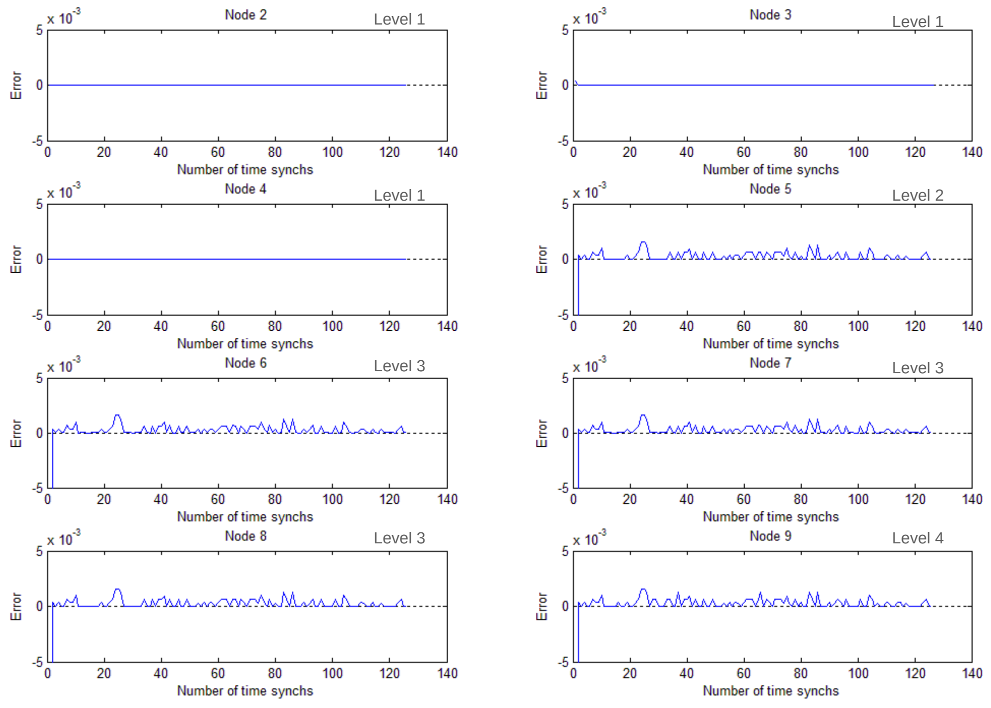

Figure 4.5, second run, the time synch error

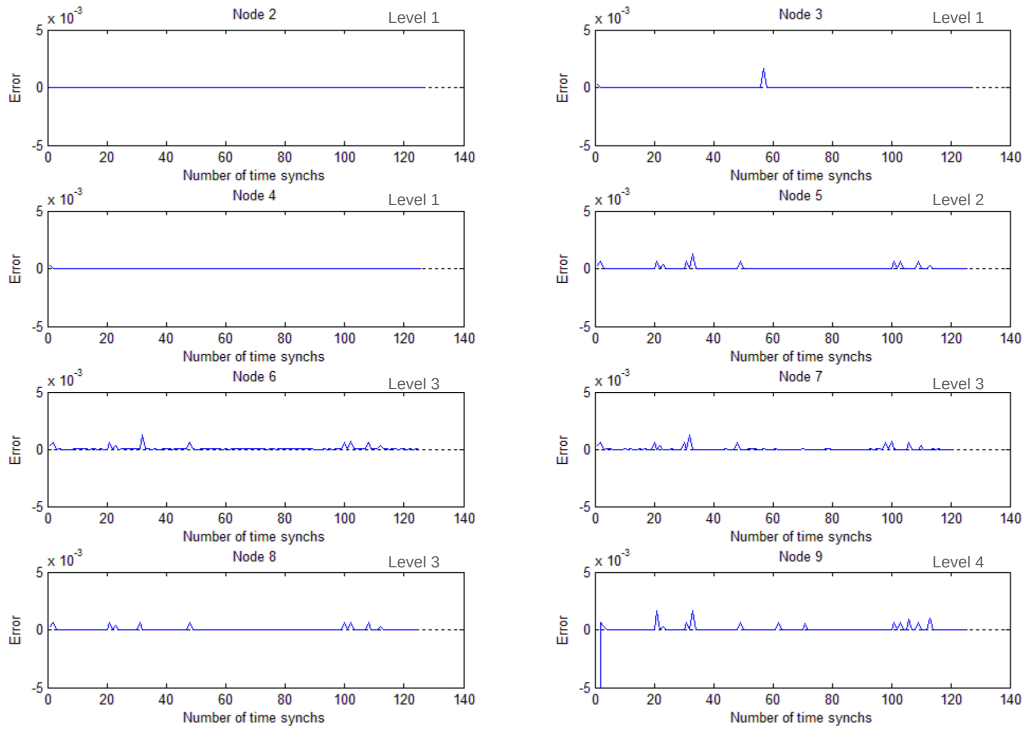

Figure 4.6, third run, the time synch error

4.3 SLEEP

Simulations were made to answer the crucial question if it is possible to let the nodes sleep a longer while and then send data in a mesh network without additional fault. That is fault caused by the sleep schedule and not faults or losses due to the radio medium or other disturbances outside the sleep schedule.

During the first simulations the nodes were awake for 8 seconds. It showed that there was no problem, even when the drift wasn't considered but only the offset. Therefore simulations letting the nodes being awake for a shorter time (4s) were conducted.

When going towards shorter time awake, a phenomenon occurred when starting the simulation. Since the nodes are awake a shorter time, one or more nodes could miss the starting time synch, thus becoming out of phase. Due to this, some nodes missed several sleep periods and did not get time synched for a long time.

An important aspect is the combined effect of the time synch error and drift factor error. The accuracy of when the nodes wake up is dependent of this.

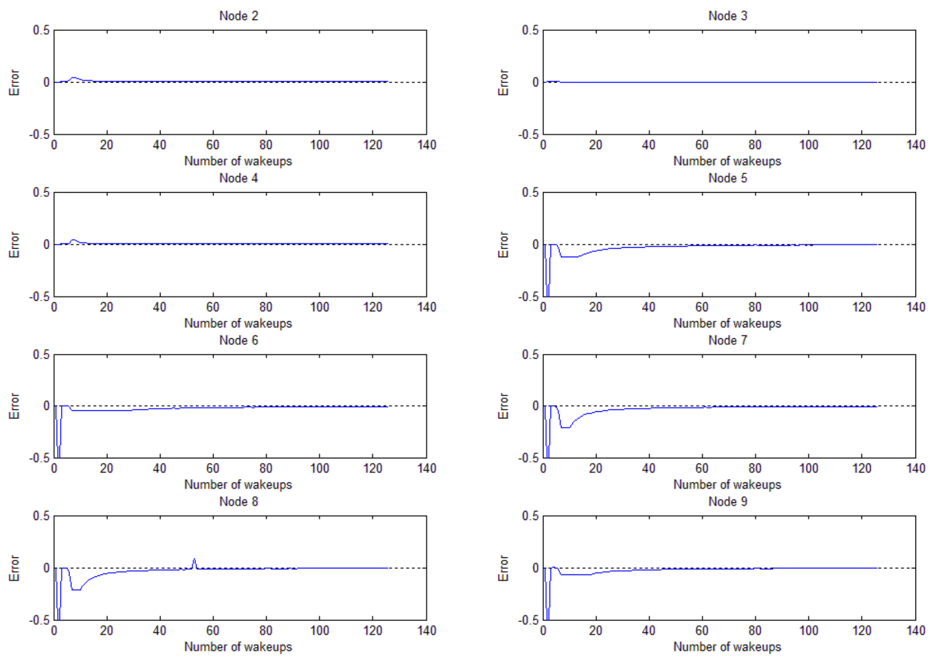

Figure 4.7, first run, wakeup error

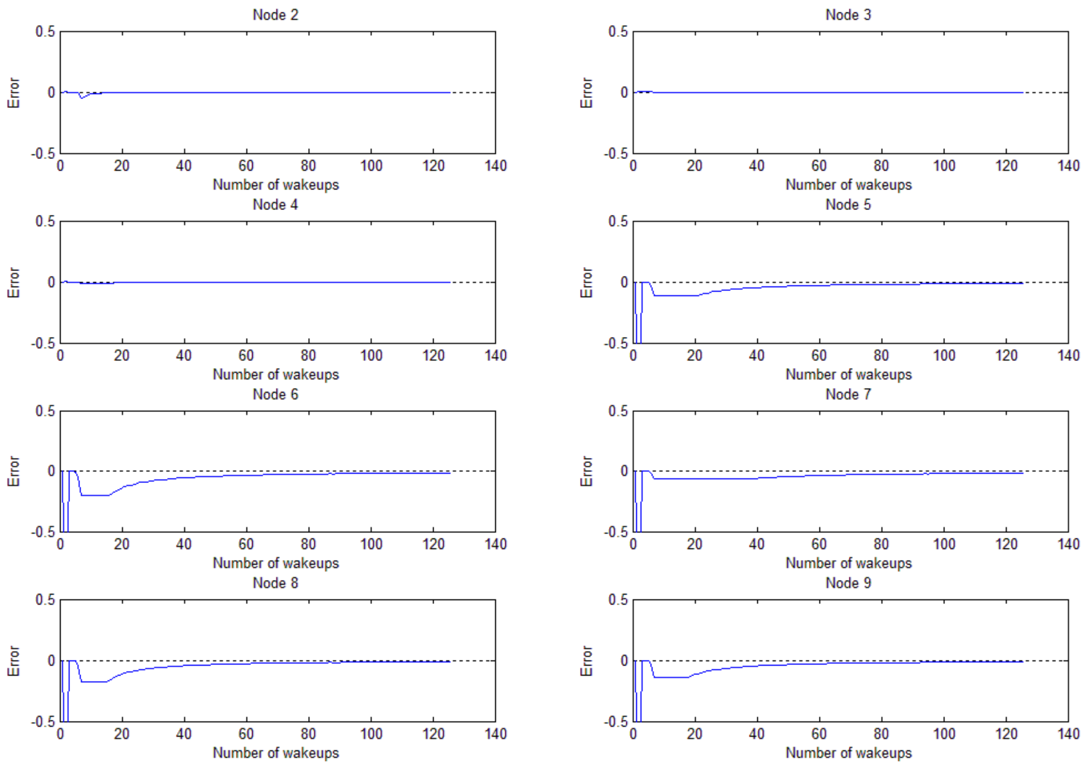

Figure 4.8, second run, wakeup error

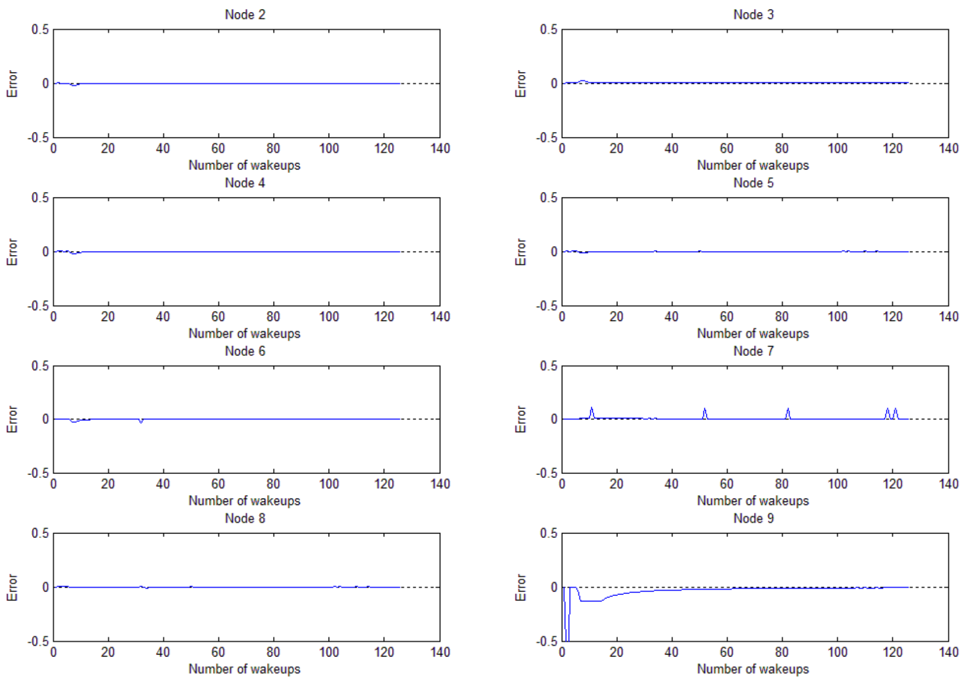

Figure 4.9, third run, wakeup error

4.4 RESULTS WITHOUT DRIFT COMPENSATION

Below are the results from simulating the same scenario but without drift compensation. The drift factor error will not be presented since it's not calculated.

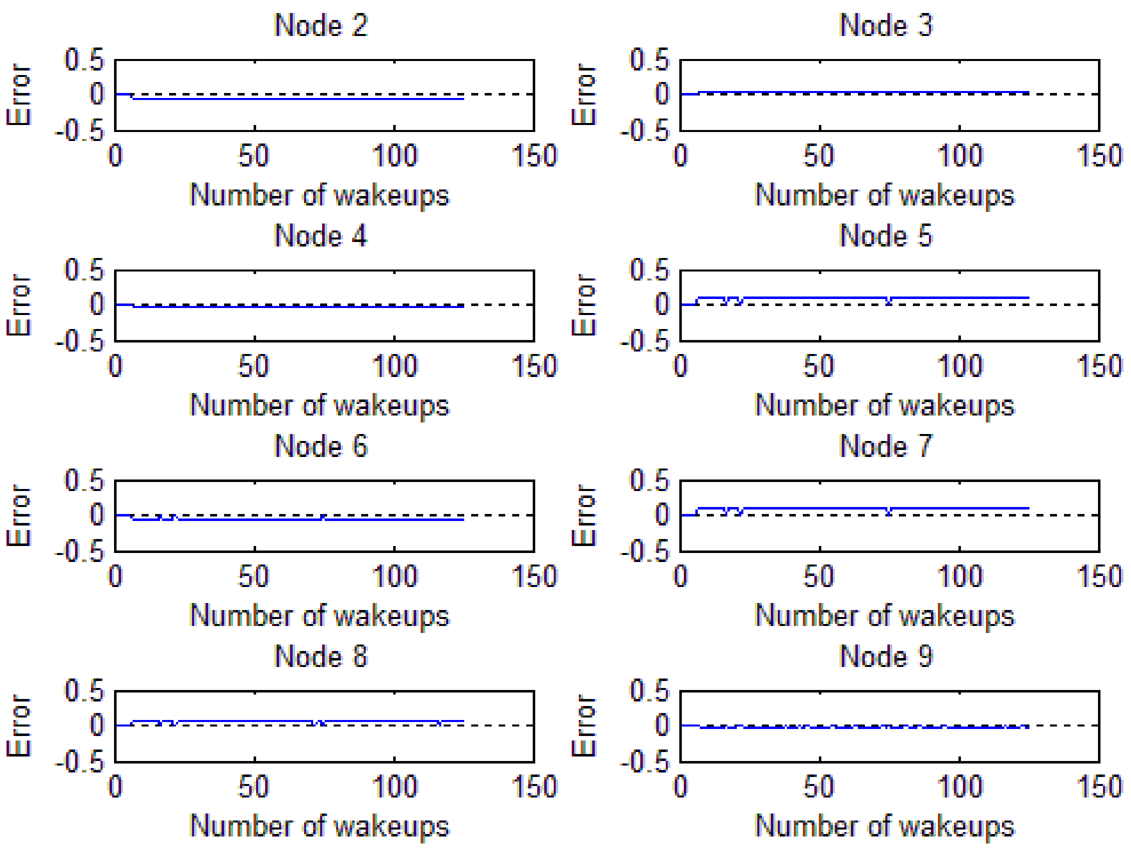

Figure 4.10, first run without drift compensation, wakeup error

Table 4.4, first run without drift compensation, initial values

Node	Time drift factor [ppm]	Start offset [s]
2	-22.3808	4.5669
3	7.9416	0.4877
4	-13.2901	2.7344
5	27.4504	4.8244
6	-20.5432	4.8530
7	27.4300	2.4269
8	18.0168	0.7094
9	-4.6943	4.5787

As one can see the wakeup error becomes constant. This is quite intuitive since the drift factor is calculated more accurate after some time; thus making it possible for the nodes to wake up at a more precise time.

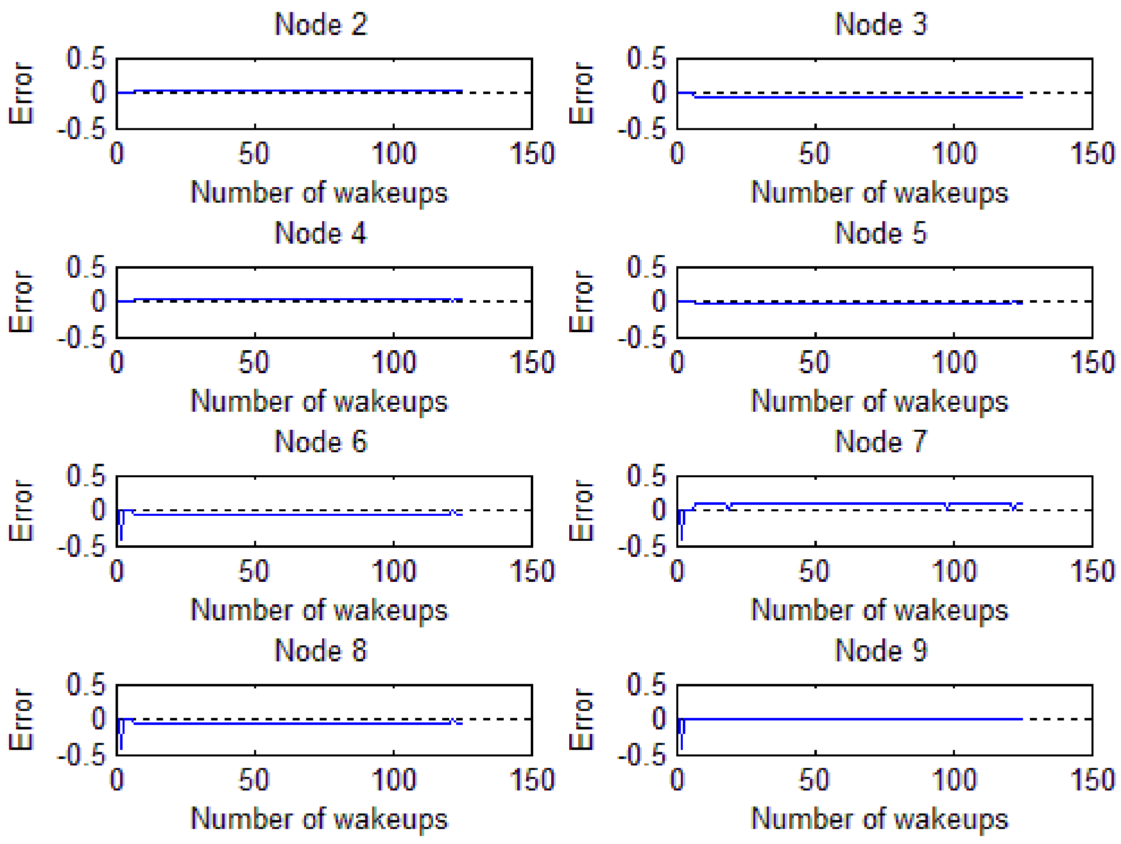

Figure 4.11, second run without drift compensation, wakeup error

Table 4.5, second run without drift compensation, initial values

Node	**Time drift factor [ppm]**	**Start offset [s]**
2	6.9147	0.6603
3	-18.4448	2.2348
4	12.3906	2.0049
5	-11.8549	0.4209
6	-20.2797	3.4554
7	26.0378	1.7956
8	-20.0415	4.2845
9	0.80421	4.5878

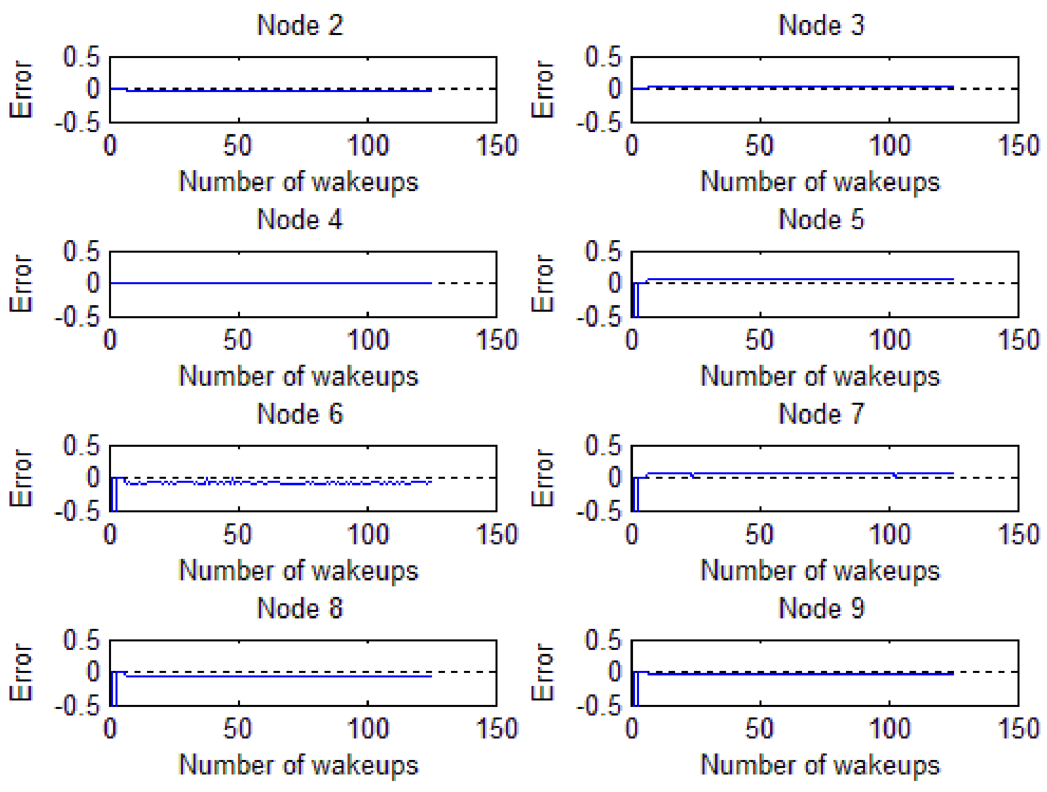

Figure 4.12, third run without drift compensation, wakeup error

Table 4.6, third run without drift compensation, initial values

Node	Time drift factor [ppm]	Start offset [s]
2	-10.8748	3.7233
3	6.9408	3.5947
4	-1.7686	1.7208
5	19.836	2.6021
6	-22.5876	2.5967
7	18.6665	3.1784
8	-16.5659	3.5678
9	-10.903	2.3706

As well in Figure 4.10, Figure 4.11, and Figure 4.12 we can see the same behaviour of the error getting constant, it is especially visible when the sleep time gets longer after a while. Theoretically the maximum error can be 0.1221s if having a 30ppm drift and a sleep time of 4096s. Comparing it with the maximum error when the drift is compensated (roughly 5ppm accuracy) one get an error of 0.0203s. These statements do not consider the time synch error.

5 DISCUSSION AND CONCLUSIONS

Here some discussion about the result is presented. Why did the result look as they did, can something be told out of that? Afterwards the conclusion; is it possible to use time synchronisation to enable synchronous sleep in an indoor climate control system?

5.1 DISCUSSION

5.1.1 USE OF DRIFT FACTOR CALCULATION

As one can see in section 4.3 the wakeup error is becoming smaller over time. In other words; the nodes wakes up more precise (in time) after some time. Therefore it is possible to minimize the overhead and by doing so reducing the active time.

How much could the active time be reduced then? The maximum wakeup error was identified to be between -0.1221s and 0.1221s when the drift was not compensated. And when the drift was adjusted for, the error was below 5ppm after a while (giving an error less than 0.0203s). The exception is when a node has missed a time synchronisation and has to calculate the next wakeup time itself. As seen in Figure 4.9, the error grew to approximately 0.125s, thus leading us to the error can grow with at least 0.1s when a node misses a time synchronisation. This gives that the wakeup error could be 0.2221 s for the system with uncompensated drift (even though it wasn't seen in those simulations conducted). The difference is about 0.1s between a system that consider the drift and compensate for it and a system that does not do that.

Considering the overhead needed and that one may want to add a safety factor to it, to be sure that a node does not miss the time slot when they are supposed to be awake. An example would be if one uses the safety factor 4 we get 0.5s overhead for the system with drift factor compensation and 0.9s for the system without. The nodes where awake for 4s during the simulations, giving the possibility to reduce the time awake with 10%. Of course, this changes with how long time the nodes are awake and what safety factor is used for the overhead.

But if that is not considered worth the effort, the drift does not need to be calculated, if the nodes are awake long enough. It might be less effort to just compensate the offset, not just in one way; code, time and complexity. One may be able to use cheaper hardware and spend less time on developing and testing thus saving money.

Another aspect is robustness. If a node misses a time synch, one want to be sure it does not miss it again during the next wakeup. By checking that a node have been time synched before going into sleep mode one can shorten the sleep time (not by much, but maybe in the size of 0.5-1 second) and let it be awake for a longer time. In addition, before going to sleep the same period as the node missed the time synch, one can let the node be awake for a bit more as well. By doing so the system gets more robust against missed time synchronisation events and thus disturbances.

5.1.2 DISTURBANCE IN TIME SYNCH

Analysing the error in the time synchronisation one can see that an error that occurs in one node, propagates deeper in the network. A node deeper in the network cannot get better time synchronisation than the node it synchronises with. It can be seen in Figure 4.4, node 5 has some error, and the same error can be seen in the nodes after node 5 (see Figure 3.3.1). Since the time synchronisation used is a one way protocol, it is hard to prevent this; the error will propagate throughout the network.

Something that can be discussed is the use of a two way time synchronisation that is less prone to disturbances or making a statistical model of the error depending on how deep in the network it is. Due to the fact that disturbances are hard to predict (that's why they are disturbances) a statistical model can be hard to create; especially for different environments. Therefore it might be a better idea to use a two way time synchronisation to make it more robust to disturbances.

But as mentioned in section 5.1.1, it is always a balance between effort and result. It takes more effort to create a two way synchronisation, but it becomes more robust. So the question is, how robust does the system need to be? Depending on the context one may want to distribute sensors over a wide area but does not care if a few nodes are lost so to say. However in the context of indoor climate control one may not want to lose nodes since it can affect the comfort of the inhabitants.

Regarding robustness, in this context the nodes are in known places and if a node is lost there is a possibility to replace the whole node, the battery or let it synchronise directly with the gateway. Even though it might be possible, it can affect the perceived comfort of the inhabitants.

5.2 CONCLUSIONS

Can the use of time synchronization in a mesh network enable synchronous sleep to save power? Based on the result that of letting the nodes sleep over one hour and wake up synchronous thus lowering the overall active time; it is possible. Another important conclusion; for this context with a limited sleep time of 1 hour, it is possible to ignore the time drift and just compensate for the offset each time synch. This can lead to less complex development. To compare it to the requirements:

- The nodes can sleep for over one hour
- The nodes can handle missed time synchronizations
- The protocol does not add additional errors in the communication

Therefore it should be possible to use it for an indoor climate control system.

How does it compare to 1% radio duty cycle as mentioned in 2.3? With an active time of 4 seconds and a sleep time of 4096 seconds gives a duty cycle of 0.1%. That is a factor 10 lower, with the limitation of sending in specific time slots which is not suitable for all applications. In other words, the battery life could be up to 1280 days (when using a 88mAh battery), but mind that the comparison is between the synchronous sleep schedule and a protocol that can send data all the time.

6 RECOMMENDATIONS AND FUTURE WORK

Depending on what the goal is one can either implement this on real hardware to evaluate and/or use it in a real system. Another possibility is to investigate different aspects with the help of simulations. Some aspects that might be especially interesting to study are mentioned below.

6.1 OPTIMIZE OVERHEAD/TIME AWAKE

As of now, the time awake (including overhead) is set to a fixed value. As mentioned in 5.1.1 one could reduce the active time by minimizing the overhead when the drift factor has been calculated thoroughly.

6.2 EVALUATING PERFORMANCE WHEN RANDOM DISTRIBUTED

The simulations have not been conducted with a random placement of the nodes; instead they were placed in a planned manner to fit the context of indoor climate control. Therefore it is hard to say how the network would act depending on network layout. A simulation (or implementation on real hardware for that matter) with random placement could show if there are any issues with it, or if it is fully possible.

6.3 INVESTIGATE RESOLUTION OF TIME

On a real microcontroller the resolution of time is limited. It might be a 16-bit timer register used, but in the simulations the resolution was not limited. Since this could have a direct impact on the performance when implementing it on hardware one may want to investigate it. Even though it may affect the drift calculations the biggest impact would be on offset and wake up time.

6.4 INVESTIGATE TEMPERATURE CHANGES

Even though the context is to keep a comfortable (and a rather stable) temperature the effect of letting the temperature fluctuate, thus affecting the time drift, have not been investigated. This could be especially interesting for other contexts were the nodes sleep for longer time and/or in an environment where the temperature fluctuates over time.

6.5 IMPROVE TIME SYNCHRONIZATION

As discussed in section 5.1.2 there are disturbances in the time synchronization. Therefore a part of future work can be improving the performance and robustness of the time synchronization or implement an existing one with proved performance.

6.6 PERFORMANCE IMPACT DUE TO NETWORK SIZE

In the thesis only a limited network was simulated. How does the sleep time and time synchronization get affected by the size of the network? Is there a correlation between the size on the network and how short the nodes can be awake? This might be especially interesting for projects that are interested in implementing the sleep schedule on large network.

REFERENCES

[1] American Society of Heating, Refrigerating and Air-Conditioning Engineers, Thermal environmental conditions for human occupancy, Ashrae, 2004.

[2] International Organization for Standardization, ISO 7730, ISO, 2005.

[3] S. K. Brown, “High quality indoor environments for sustainable office buildings,” in *CRC for Construction Innovation*, 2008.

[4] L. Peeters, J. Van der Veken, H. Hens, L. Helsen and W. D’haeseleer, “Control of heating systems in residential buildings: Current practice,” *Energy and Buildings,* vol. 40, no. 8, pp. 1446-1455, 2008.

[5] L. Peeters, J. Van der Veken, H. Hens, L. Helsen and W. D’haeseleer, “Control of heating systems in residential buildings: Current practice,” *Energy and Buildings,* vol. 40, no. 8, pp. 1146-1455, 2008.

[6] V. Jain, V. Garg, J. Mathur and S. Dhaka, “Effect of operative temperature based thermostat control as compared to air temperature based control on energy consumption in highly glazed buildings,” in *12th Conference of International Building Performance Simulation Association*, Sydney, 2011.

[7] K. Vinther, V. Chandan and A. G. Alleyne, “Learning/repetitive control for building systems with nearly periodic disturbances,” *Control Conference (ECC) IEEE,* pp. 1198-1203, 2013.

[8] S. Prívara, J. Široký, L. Ferkl and J. Cigler, “Model predictive control of a building heating system: The first experience,” *Energy and Buildings,* vol. 43, no. 2, pp. 564-572, 2011.

[9] R. Halvgaard, N. K. Poulsen, H. Madsen and J. B. Jorgensen, “Economic model predictive control for building climate control in a smart grid,” *Innovative Smart Grid Technologies,* pp. 1-6, 2012.

[10] R. de Dear, “Revisiting an old hypothesis of human thermal perception: alliesthesia,” *Building Research & Information,* vol. 39, no. 2, pp. 108-117, 2011.

[11] H.-Y. H. Xing, “Building load control and optimization,” Massachusetts Institute of Technology, 2004.

[12] J. van Schijndel and P. Steskens, “System Identification for Indoor Climate Control,” in *7th International Conference on System Simulation in Buildings*, Liege, 2006.

[13] J. Persson and D. Vogel, “Utnyttjande av byggnaders värmetröghet,” Lunds Tekniska Högskola/Lunds Universitet, Lund, 2011.

[14] J. Yick, B. Mukherjee and D. Ghosal, “Wireless sensor network survey,” *Computer Netowrks,* no. 52, pp. 2292-2330, 2008.

[15] T. R. H. S. J. Voigt, “Utilizing solar power in wireless sensor networks,” *Local Computer Networks IEEE,* pp. 416-422, 2003.

[16] A. Dunkels, The contikimac radio duty cycling protocol, Swedish Institute of Computer Science, 2011.

[17] T. Zheng, S. Radhakrishanan and V. Sarangan, “PMAC: an adaptive energy-efficient MAC protocol for wireless sensor networks,” in *Parallel and Distributed Processing Symposium*, Phoenix, 2005.

[18] P. Suarez, C.-G. Renmarker, A. Dunkels and T. Voigt, “Increasing ZigBee network lifetime with X-MAC,” in *Proceedings of the workshop on Real-world wireless sensor networks*, Glasgow, 2008.

[19] D. Gao, Y. Niu and O. W. Yang, “Synchronous sleep andwake in IP-enabled wireless sensor networks,” *Electrical and Computer Engineering,* pp. 161-164, 2009.

[20] I. F. kyildiz, X. Wang and W. Wang, “Wireless mesh networks: a survey,” *Computer networks,* vol. 47, no. 4, pp. 445-487, 2005.

[21] A. Koubaa, M. Alves and E. Tovar, “Modeling and worst-case dimensioning of cluster-tree wireless sensor networks,” in *Real-Time Systems Symposium, 27th IEEE International*, 2006.

[22] N. Magharei and R. Rejaie, “Prime: Peer-to-peer receiver-driven mesh-based streaming,” *IEEE/ACM Transactions on Networking,* vol. 17, no. 4, pp. 1052-1065, 2009.

[23] R. Schollmeier, “A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications,” in *Proceedings First International Conference on Peer-to-Peer Computing*, Linköping, 2001.

[24] Q. Ren and Q. Liang, “An Energy-Efficient MAC Protocol for Wireless Sensor Networks,” *IEEE Globecom,* pp. 157-161, 2005.

[25] R. Subramanian and F. Fekri, “Sleep scheduling and lifetime maximization in sensor networks: fundamental limits and optimal solutions,” in *Proceedings of the 5th international conference on Information processing in sensor networks*, Nashville, 2006.

[26] S. Duquennoy, F. Österlind and A. Dunkels, “Lossy links, low power, high throughput,” in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, Seattle, 2011.

[27] W. Ye, J. Heidemann and D. Estrin, “An energy-efficient MAC protocol for wireless sensor networks,” in *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, New York, 2002.

[28] “CultureBee - wireless monitoring of culture heritage,” [Online]. Available: http://www.culturebee.se/faq. [Accessed 11 August 2014].

[29] J. Zhang, A. Hyunh, Q. Ye and S. Gong, “Culturebee-A Fully Wireless Monitoring and Control System for Protecting Cultural Heritage,” *Enabling Technologies: Infrastructure for Collaborative Enterprises,* pp. 250-255, 2011.

[30] D.-Y. Gao, L.-J. Zhang and H.-C. Wang, “Energy saving with node sleep and power control mechanisms for wireless sensor networks,” *The Journal of China Universities of Posts and Telecommunications,* vol. 18, no. 1, pp. 49-59, 2011.

[31] L. Tang, Y. Sun, O. Gurewitz and D. B. Johnson, “PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks,” in *INFOCOM, 2011 Proceedings IEEE*, IEEE, 2011, pp. 1305-1313.

[32] E. Nataf and O. Festor, "Online Estimation of Battery Lifetime for Wireless Sensor Network," Inria, Nancy, 2012.

[33] Y.-C. Wu, Q. Chaudhari and E. Serpedin, “Clock synchronization of wireless sensor networks,” *Signal Processing Magazine, IEEE,* vol. 28, no. 1, pp. 124-138, 2011.

[34] B. Jiang, K. Han, B. Ravindran and H. Cho, “Energy efficient sleep scheduling based on moving directions in target tracking sensor network,” *Parallel and Distributed Processing,* 2008.

[35] “Simple, accurate time synchronization for wireless sensor networks,” *Wireless Communications and Networking, IEEE,* vol. 2, pp. 1266-1273, 2003.

[36] F. Sivrikaya and B. Yener, “Time synchronization in sensor networks: a survey,” *IEEE Network,* vol. 18, no. 4, pp. 45-50, 2004.

[37] L. Schenato and F. Fiorentin, “Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks,” *Automatica,* vol. 47, no. 9, pp. 1878-1886, 2011.

[38] P. Sommer and R. Wattenhofer, “Symmetric Clock Synchronization in Sensor Networks,” ETH Zurich, 2008. [Online]. Available: http://www.tik.ee.ethz.ch/file/d9e498034f81e2f523963208db107be7/realwsn08_slides.pdf. [Accessed 16 April 2014].

[39] “Crystal | Crystal and Osciallators | DigiKey,” DigiKey, 14 April 2014. [Online]. Available: http://www.digikey.com/product-search/en/crystals-and-oscillators/crystals/852333?k=32kHz. [Accessed 14 April 2014].

[40] “Micro Crystal - Realtime Clock Modules,” Micro Crystal Switzerland, 2013. [Online]. Available: http://www.microcrystal.com/index.php/products/real-time-clocks. [Accessed 14 April 2014].

[41] “PCF2127AT Product data sheet,” NXP Semiconductors, 11 July 2013. [Online]. Available: http://www.nxp.com/documents/data_sheet/PCF2127AT.pdf. [Accessed 16 April 2014].

[42] “Quartz crystal list | Quartz Crystal / Oscillator,” Citizen Finetech Miyota, 2014. [Online]. Available: http://cfm.citizen.co.jp/english/product/cvo_list1.html. [Accessed 14 April 2014].

[43] “ECS, Inc. International, THRU-hole and Surface Mount Crystals,” ECS International, [Online]. Available: http://www.ecsxtal.com/store/pdf/ECS-3x8X%202x6X%201X5X.pdf. [Accessed 14 April 2014].

[44] “Introduction to Quartz Frequency Standards - Static Frequency versus Temperature Stability,” IEEE Ultrasonics, Ferroelectronics, and Frequency Control Soceity, 2014. [Online]. Available: http://www.ieee-uffc.org/frequency-control/learning-vig.asp?chapter=vigstatc. [Accessed 15 April 2014].

[45] “DS3231 Extremely Accurate I2C-Integrated RTC/TCXO/Crystal,” Maxim intergrated, January 2013. [Online]. Available: http://datasheets.maximintegrated.com/en/ds/DS3231.pdf. [Accessed 16 April 2014].

[46] “AB-RTCMK-32.768kHz REAL TIME CLOCK MODULE WITH 32.768kHz TCXO,” Abracon, 30 September 2012. [Online]. Available: http://www.abracon.com/Precisiontiming/AB-RTCMK-32768kHz.pdf. [Accessed 16 April 2014].

[47] “Micro Crystal - Oven Controlled XTAL Oscillators,” Micro Crystal Switzerland, [Online]. Available: http://www.microcrystal.com/index.php/products/oscillators/ocxo. [Accessed 22 April 2014].

[48] “Atmel | APPLICATION NOTE | AT03155: Real-Time-Clock Calibration and Compensation | SAM3 / SAM4 Series,” Atmel, 2014. [Online]. Available: http://www.atmel.com/Images/Atmel-42251-RTC-Calibration-and-Compensation_AP-Note_AT03155.pdf. [Accessed 22 April 2014].

[49] Thingsquare, "Get started with Contiki, Instant Contiki and Cooja," [Online]. Available: http://www.contiki-os.org/start.html. [Accessed 3 September 2014].

[50] Lunds Universitet, "Automatic Control - TrueTime," [Online]. Available: http://www.control.lth.se/truetime/. [Accessed 3 September 2014].

[51] OMNeT++ Community, "OMNeT++ Network Simulation Framework," [Online]. Available: http://www.omnetpp.org/. [Accessed 3 September 2014].