ANALYSIS OF ADAPTIVE MULTI-HOP TIME SYNCHRONIZATION IN LARGE WIRELESS SENSOR NETWORKS

BY

NATTAPHOL SANGJUMPA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING IN INFORMATION AND COMMUNICATION
TECHNOLOGY FOR EMBEDDED SYSTEMS
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2016

ANALYSIS OF ADAPTIVE MULTI-HOP TIME SYNCHRONIZATION IN LARGE WIRELESS SENSOR NETWORKS

A Thesis Presented

By

NATTAPHOL SANGJUMPA

Submitted to

Sirindhorn International Institute of Technology

Thammasat University

In partial fulfillment of the requirement for the degree of

MASTER OF ENGINEERING IN INFORMATION AND COMMUNICATION TECHNOLOGY FOR EMBEDDED SYSTEMS

Approved as to style and content by

Advisor and Chairperson of Thesis Committee	Asst. Prof. Somsak Kittipiyakul, PhD
Co-Advisor	Assoc. Prof. Steven Gordon, PhD
Committee Member and Chairperson of Examination Committee	Kamol Kaemarungsi, PhD
Committee Member	Prof. Akinori Nishihara, PhD
Committee Member	Assoc. Prof. Komwut Wipusitwarakun, PhD
Committee Member	Asst. Prof. Prapun Suksompong, PhD

JULY 2017

Acknowledgments

I would like to thank my co-advisor, Prof. Steven Gordon, who was my previous advisor for the first and major part of my Master study. He gave me continuous support. I am very grateful for his patience, motivation, passion, and endless knowledge. His guidance helped me throughout my research and writing of this thesis. Without his persistent help, this thesis would not have been completed.

This work would have not been completed either without the insightful comments and supervision of Dr. Kamol Kaemarungsi, from National Electronics and Computer Technology Center.

My gratitude also goes to my current advisor, Prof. Somsak Kittipiyakul. He has helped me revising the thesis significantly.

I am very grateful for comments and suggestions from the rest of the thesis committee: Prof. Akinori Nishihara, from Tokyo Institute of Technology, Prof. Komwut Wipusitwarakun, and Prof. Prapun Suksompong. They have tried to understand my work and my English.

Without my seniors and classmates in Information and Communication Technology for Embedded Systems program, my life at SIIT would not be so fun. Thank you for all fun we had while studying and for helping me conducting simulations. I am also grateful to my friends in the SIIT Network Lab for offering me advice and support.

Finally, I would like to thank my family members who have supported me spiritually throughout my life, especially my mother who has constantly reminded me to maintain a good health while studying. She has always been understanding and supportive along all the paths I have taken. All of the words are not enough to describe my feelings toward my family.

The research presented in this thesis was supported via scholarships, travel funding, and research assistantship from the following organizations: Thailand Advanced Institute of Science and Technology, National Science and Technology Development Agency, Tokyo Institute of Technology, Sirindhorn International Institute of Technology, Thammasat University, and the National Research University Project of Thailand, Office of Higher Education Commission.

Abstract

In this thesis, we study time synchronization problem in multi-hop wireless sensor networks. Specifically, we perform a simulation-based study of the Adaptive Multi-Hop Time Synchronization (AMTS) protocol proposed by Noh et al [1] for different network topologies. AMTS is an extension of the Timing-sync Protocol for Sensor Networks (TPSN) protocol [2] and aims to increase the energy efficiency of time synchronization in large-scale and long-lived multi-hop wireless sensor networks.

Given that AMTS has been analyzed for some simple topologies like chain networks, here we analyze AMTS for more general two-dimensional topology such as grids. We also look at the effect of key parameters of AMTS such as re-sync period, the number of beacons per pairwise synchronization, and the synchronization mode. The performance metrics we look at are synchronization error, overhead, and energy consumption.

We perform our simulation using the OMNeT++ simulator. Our simulation results show similar results as what have been found earlier with the chain topology. Specifically, we found that: i) For all topologies, as the network size increases, all three performance metrics get worse; ii) When the re-sync period increases, the timing error becomes worse while the overhead and energy consumption are better; and iii) For the amount of sensing activity assumed in the simulation, all three performance metrics are better for the session-initiated (SI) mode, compared to the always-on (AO) mode.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Distributed systems such as wireless sensor networks (WSNs) maintain the clocks of their nodes in such a way that the clocks are in synchronization. This synchronization is one of the most complex problems in WSNs [3].

Time synchronization is required in sensing time-sensitive physical quantities (e.g., pressure, humidity, temperature), in controling time-sensitive actuators (e.g., solenoid, electric motor), or in data gathering and transmission. The need for unobstrusive and remove monitoring is the main motivation for deploying a WSN [2].

Time synchronization algorithm allows different nodes to automatically time synchronize with other nodes. In these algorithms, nodes do not need to pre-configure to access particular root or coordinator node: they can self-configure to find the best facility (e.g. node routing, time synchronization, mode selections) on-demand. This is valuable in WSNs as such network can be highly dynamic (mobility and failure of nodes). Ordinarily, time synchronization can be performed in a topological structure, with a coordinator node broadcasting sync packeta (beacon) to end device nodes in a network. Many protocols and algorithm have been developed for time synchronization in WSN (for example, [2, 1, 4, 5, 6]).

In this thesis, we study an adaptive multi-hop time synchronization [1] in wireless sensor networks which typically have resource-constrained nodes. These nodes collectively sense the environment and deliver the sensed data to some collector nodes. The nodes have limited hardware capabilities, therefore extra communication packet inside the protocols are designed to give optimization for performance given the pressure. One limitation is on-board clocks: nodes may not have a separate hardware clock, with the clock only implemented as a counter in software. The result is that clocks on sensor nodes may be highly inaccurate; manufacturing tolerance, age, temperature, pressure and other factors may effect to different clock rates on different component of sensors [4].

1.2 Problem Statement

A key component in WSNs is time synchronization protocol. A popular method for time synchronization is the pairwise synchronization [7], where a node (node A) is synchronized to another node (node B) by exchanging timing messages node. Node A calculates the clock offset of it own clock to the clock of node B and adjusts the clock offset accordingly, so that both nodes are in sync. Several protocols (such as [8]) have been designed for WSNs and support pairwise synchronization so that the whole network is in synchronization with a reference node.

A popular protocol for WSNs is Timing-Sync Protocol for Sensor Networks (TPSN) [9]. The protocol starts with a level discovery phase to assign each node with a level which turns the network into a hierarchical structure for performing time synchronization in the second phase. In the second phase, pairwise synchronization is applied from a node to a node in a lower level, such that eventually all nodes are synchronized to the reference node.

There are many time synchronization protocols that extends TPSN, such as [10, 11]. Some authors proposed adaptive-clock synchronization protocols, such as [8, 12, 1], that optimize the network synchronization protocol with the aim of achieving a specific synchronization accuracy with minimal energy consumption.

In this thesis, we are particularly interested in the Adaptive Multi-hops Time Synchronization (AMTS) protocol [1]. AMTS significantly extends TPSN with the goal of minimizing the overall energy consumption in large-scale and long-lived network. For long-lived network, the synchronization must be done regularly and due to the drift (called skew) of the local clocks, it is wise to estimate and adjust for the clock drift as well. The period for re-synchronization can be dynamically adjusted. In addition, for large-scale networks that have many nodes, usually only a small subset of nodes are active in transmission of sensed data. Hence, only these nodes are required to be kept in sync with the reference node, while the other nodes can be left out of sync. That means for large-scale networks, we may not need to perform network-wide synchronization for every nodes.

An important design trade off for time synchronization protocols is that of clock accuracy versus network traffic overhead and energy consumption. Keeping the clocks of all nodes synchronized with a reference node can require significant traffic in the network, which in turn requires energy consumption for packet transmission and reception. Although for small, simple network topologies (e.g., a chain) the overhead can be tolerable, as networks grow, the overhead may be too much for most applications.

1.3 Aims of the Thesis

In the original work proposing the protocol [1], AMTS has been analysed for a chain topology. Similarly, TPSN, which AMTS is based on, has been analysed for a chain topology as well but with larger number of nodes (50 nodes) [4]. However, the network topologies found in WSNs are more general than the chain topology.

Hence. in this thesis we are interested in studying performance of AMTS in a more realistic network topology such as grid. Compared to the chain topology which is considered as one-dimensional form, the grid topology is two-dimensional form.

The reason we want to study a two-dimensional node distribution such as the grid topology is as following. There are multiple nodes with the same synchronization level in the hierarchical structure after the Level Discovery Phase. This means during the Synchronization Phase there might be collision when multiple nodes perform the pairwise synchronization at the same time. This may result in more overhead and more energy consumption. In addition, there is more flooding and collision during the Level Discovery Phase.

Although there are many performance metrics to measure performance of a synchronization protocol, in this thesis we look at three performance metrics: synchronization error, overhead, and energy consumption, for the different network topologies. We also consider the effect of the location of the reference node and the key parameters of AMTS.

1.4 Scope of the Thesis

The focus of this thesis is in multi-hop time synchronization in WSNs, specifically evaluating the performance of AMTS. Here is the main scope of the study of our thesis:

1. We perform simulation-based study of AMTS in two different topologies: chain and grid.
2. The location of the time reference node is at the boundary in the chain topology and at the boundary and center for the grid topology.
3. The network sizes vary from 2 to 26 nodes.
4. We look at three performance metrics: synchronization error, overhead, and energy consumption.
5. Heterogeneous devices with a mix of CPU and clock capabilities are not considered. Our focus is on a simple, typical WSN with all devices having the same capability.

1.5 Contributions of the Thesis

We hope that our thesis provides the following contribution:

1. A better understanding of the time synchronization protocols in multi-hop wireless sensor networks, specifically the TPSN and AMTS protocols. The reader might get a better understanding how the protocols work and the effects of some key parameters such as re-sync period and synchronization modes.
2. Simulation of AMTS protocol using OMNeT++ simulator. OMNet++ is publicly available and a useful tool to model many wireless network systems and protocols.
3. Comparison of some important performance metrics, which are synchronization error, overhead, and energy consumption.

1.6 List of Publication

The research leading to this thesis has been published in an international conference:

- N. Sangjumpa, S. Gordon, and K. Kaemarungsi, "Analysis of adaptive multi-hop time synchronization in large wireless sensor networks," in 2016 7th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES), March 2016, pp. 79-84.

1.7 Thesis Organization

The rest of thesis is organized as follows:

In Chapter 2, we provide background and details on time synchronization in WSNs. We discuss the pairwise synchronization protocol which is important for time synchronization in wired and wireless networks. Then we move to time synchronization over multi-hop wireless networks and specifically discuss the TPSN protocol and the AMTS protocol which is based on TPSN and the specific time synchronization protocol we study in our thesis.

Next in Chapter 3, we discuss in details the different network topologies and performance metrics we use for studying of the performance of AMTS. We also gives the parameters of AMTS we use for simulation on the OMNet++ simulation tool.

The simulation results are given in Chapter 4. There we discuss what we have found from our simulation.

Finally, Chapter 5 provides a summary of the thesis and discusses some possible future work.

Chapter 2

Time Synchronization in Wireless Sensor Networks

2.1 Wireless Sensor Networks

Advances in micro-electro-mechanical systems (MEMS) technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate untethered in short distances. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks based on collaborative effort of a large number of nodes [4].

A wireless sensor network consists of spatially distributed autonomous devices using sensors to monitor physical or environmental conditions. A WSN system incorporates a gateway that provides wireless connectivity back to the wired world and distributed nodes.

Some important characteristics of WSNs are as following:

1. Tolerance to node failures: Some sensors are limited in mobility and may fail or be blocked or loss battery energy by physical or data attacks.

2. Scalability: Some networks may be scaled as large as thousands nodes.

3. Dynamic network topology: Nodes may be highly mobile. Some nodes can become malfunction due to hardware failure or battery depletion.

4. Hardware constraints: Most sensors are small in size and are battery operated. As a result, CPU processor, memory and storage are typically very limited. Also, wireless communication data rates and duty cycles are limited.

5. Production cost: Sensor nodes are cheap and built with decent hardwares. The concept is that a mass of cheap sensors is better than a few of expensive sensors.

6. Energy consumption: Divided into two parts, one is that related with a adjustable power of transmission, and another is relative power consumption of the remaining circuit for communication system in WSNs [8].

7. Data transmission: Operation in networks for processing to send, store, and receove data in a essential way [13].

2.2 Time Synchronization in WSNs

2.2.1 Motivation for Time Synchronization

Wireless sensor networks (WSNs) consist of resource- constrained nodes that collectively sense the environment and report back to central nodes or servers. The nodes have limited hardware capabilities, therefore specialize communication protocols are designed to give optimal performance given the constraints. One limitation is on-board clocks: nodes may not have a separate hardware clock, with the clock only implemented as a counter in software. The result is that clocks on sensor nodes may be inaccurate; manufacturing tolerance, aging, temperature, pressure and other factors may lead to different clock rates on different sensors [1]. Hence researchers have developed time synchronization protocols for WSNs.

Time synchronization is an important issue in multi-hop ad hoc wireless networks such as sensor networks. Many applications of sensor networks need local clocks of sensor nodes to be synchronized, requiring various degrees of precision. Some intrinsic properties of sensor networks, such as limited resources of energy, storage, computation, and bandwidth, combined with potentially high density of nodes make traditional synchronization methods unsuitable for these networks. Hence, there has been an increasing research focus on designing synchronization algorithms specifically for sensor networks. This article reviews the time synchronization problem and the need for synchronization in sensor networks, then presents in detail the basic synchronization methods explicitly designed and proposed for sensor networks [14].

Synchronization of Timing is computer science and engineering majors, that is focus in coordinate otherwise independent clocks. Alike when collection set accurately, real clocks will diverge after some amount of time as clock drift, caused by clocks counting time at slightly different rates[15] .

However, we focus the part of significant profit and more traditional distributed systems, that we can easily and efficiently deploy time servers. Moreover, most machines can contact each other, following for an almost simple diffusion of information. These assumptions are no longer valid in many wireless networks, evident sensor networks. Nodes are resource constrained, and multi-hop routing is lavish. In addition, it is often important to optimize logic for energy consumption. These and other observations have led to the design of very different clock synchronization algorithms for wireless networks. In the following, we consider one specific solution in the AMTS protocol.

Time synchronization is a procedure for providing a common notion of time across a distributed system. It is crucial for WSNs when they perform a number of fundamental operations, such as:

1. Data fusion
   Data merging is a major operation in all distributed networks for processing and integrating the collected data in a meaningful way, and it requires some or all nodes in the network to share a common time scale.

2. Power management
   Energy efficiency is a key factor when designing WSNs since sensors are usually left unattended without maintenance and battery replacement for their lifetimes after deployment. Most energy-saving operations strongly depend on time synchronization. For instance, duty cycling (sleep and wake-up modes control) helps the nodes to save huge energy resources by spending minimal power during the sleep mode. Thus, net-

work wide synchronization is essential for efficient duty cycling and its performance is proportional to the synchronization accuracy.

3. Transmission scheduling
   Many scheduling protocols require time synchronization. For example, the time division multiple access (TDMA) scheme, one of the most popular communications schemes for distributed networks, is only applicable to a synchronized network.

4. Miscellaneous
   Many localization, security, and tracking protocols also demand the nodes to time stamp their messages and sensing events. Therefore, time synchronization is one of the most important research challenges in the design of energy-efficient WSNs.

2.2.2 Pairwise Synchronization

A key component of WSN time synchronization protocols is pairwise synchronization (see [7, 2]), where a sender node and receiver node exchange synchronization messages such that one can calculate the clock offset between the two nodes and adjust its clock accordingly so that it is synchronized with the other node.

Specifically, as shown in Fig. 2.1, suppose node A wants to synchronize with node B. A and B follow the following steps:

1. Node A sends a *synchronization pulse packet* (sometimes referred to as "**beacon**") containing the time stamp $T1$, which is the time this packet is sent from A and measured by the local clock at A.

2. Node B receives this packet at time $T2$ which is measured by the local clock at B. Hence,
   $$T2 = T1 + \Delta + d \tag{2.1}$$
   where Δ is the clock offset at B relative to A (i.e., the time at B is $T1 + \Delta$ while it is $T1$ at A) and d is the propagation delay between the two nodes.

3. Node B sends back an *acknowledgement packet* at time $T3$, measured by the local clock at B. The packet contains the times $T1, T2$ and $T3$.

4. Node A receives the packet at time $T4$, measured at his local clock. That is,
   $$T4 = T3 - \Delta + d \tag{2.2}$$
   since the time, measured by the local clock at A, that the ack packet is sent is $T3 - \Delta$. Here we assume that the clock offsets and the propagation delays from A to B and B to A at the two different times $T1$ and $T3$ stay unchanged.

5. Given the knowledge of $T1, T2, T3$, and $T4$, node A can calculate the clock offset, from (2.1) and (2.2), as
   $$\Delta = \frac{(T2 - T1) - (T4 - T3)}{2} \tag{2.3}$$
   and the propagation delay as
   $$d = \frac{(T2 - T1) + (T4 - T3)}{2} \tag{2.4}$$

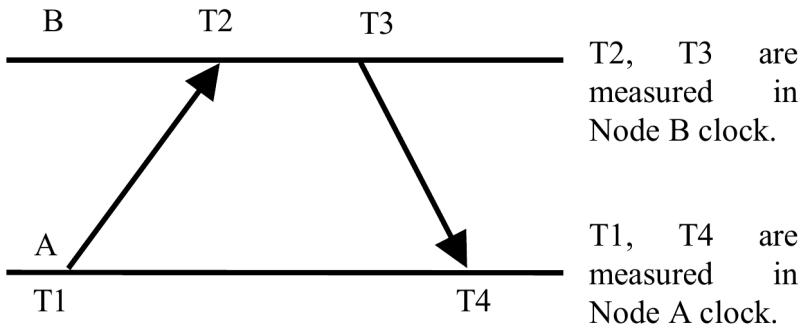

Figure 2.1: Pairwise synchronization: Two-way message exchange between nodes A and B [2]

With the clock offset Δ, node A can adjust its local clock forward by Δ and now it is in sync with node B.

Note that since the node A who will adjust the clock offset is the one who sends the synchronization pulse packet, this approach of synchronization is called the *sender-receiver synchronization* approach.

An alternative approach for synchronization in wireless environment where multiple receivers can overhear the same transmission from a node is the *receiver-receiver synchronization*. An example of protocols using this receiver-receiver synchronization approach is Reference Broadcast Synchronization (RBS) [16].

2.2.3 Time Synchronization over Multi-hop Wireless Network

In small wireless networks, all nodes can directly talk and perform pairwise synchronization to the reference (root) node. Each node takes turn (to avoid collision) sending beacon to the root node, receives the acknowledgement packet from the root node, and determines and adjusts its clock offset to be sync with the root node.

However, in large networks where some nodes cannot talk directly to the root, pairwise synchronization can be done in multiple stages. Conceptually, in the first stage the nodes that can directly talk to the root synchronize their clocks first. In the next stage, other nodes than can talk to this first level of nodes can perform their own pairwise synchronization to some of the first-level nodes and become in sync with the first-level nodes and the root. This is sequentially done until all nodes are in sync.

This is the concept of Timing-sync Protocol for Sensor Networks (TPSN) [2]. TPSN protocol is divided into two phases: Level Discovery Phase and Synchronization Phase.

Level Discovery Phase

The level discovery phase is performed at the start of the network operation to assign levels to each node. The idea is that the root/reference node is level 0, its 1-hop neighbors are level 1, its 2-hop neighbors are level 2, and so on. The steps whose flowchart is shown in Fig. 2.2 are as following:

1. If the root node is not selected, select a root node using an appropriate leader election algorithm. Assign a zero level to the root node.

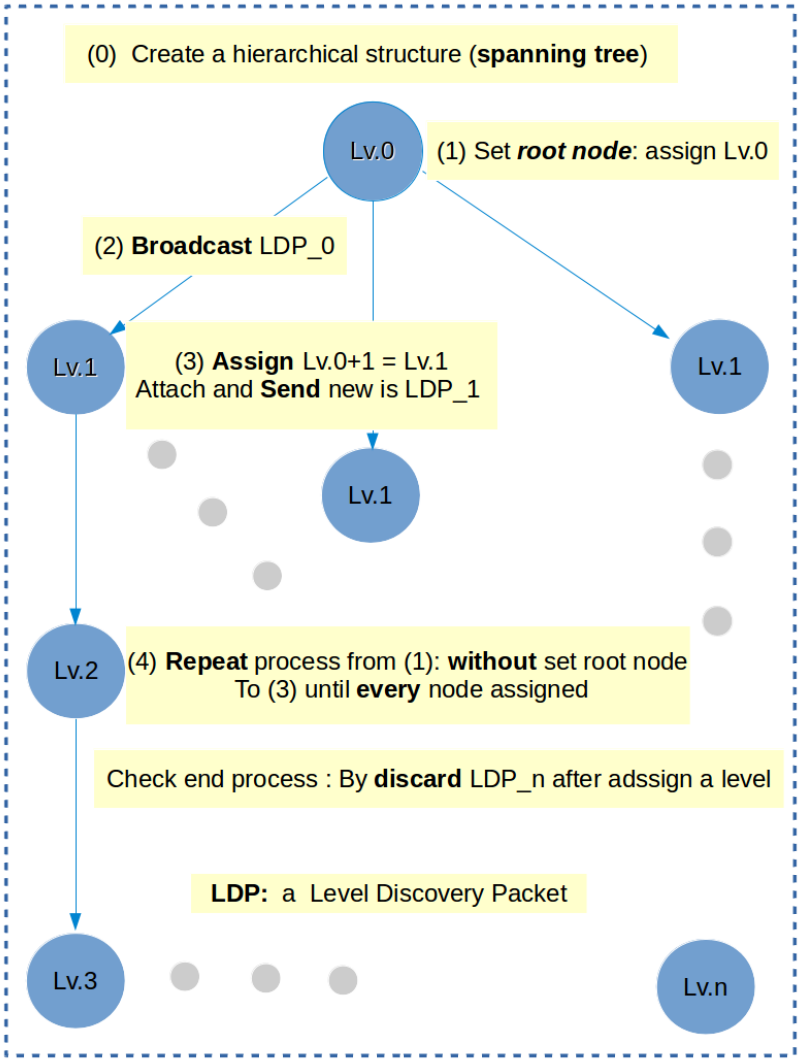

Figure 2.2: Flowchart of Level Discovery Phase

2. The root node initiates this phase by broadcasting a *level discovery packet* (LDP) containing the identity and the level of the sender (i.e., level 0).

3. The immediate neighbors of the root node receive this packet and assign themselves a level that is one greater than the level they have received, i.e., level 1.

4. After establishing its own level, each node broadcasts a new LDP packet containing its own level.

5. This process is continued until every node in the network is assigned a level.

A simple flooding algorithm or more sophisticated but efficient minimum spanning-tree algorithms can be used to broadcast LDP packets. To avoid flooding congestion, after a node has been assigned a level, it ignores any future LDP packets.

An example of how the level assignment works out is shown in Fig. 2.3 for a 5-by-5 grid network and a root node on top of the grid. In this example, for convenience we assume the transmission range is just enough to cover only the adjacent neighbors of the sender. The

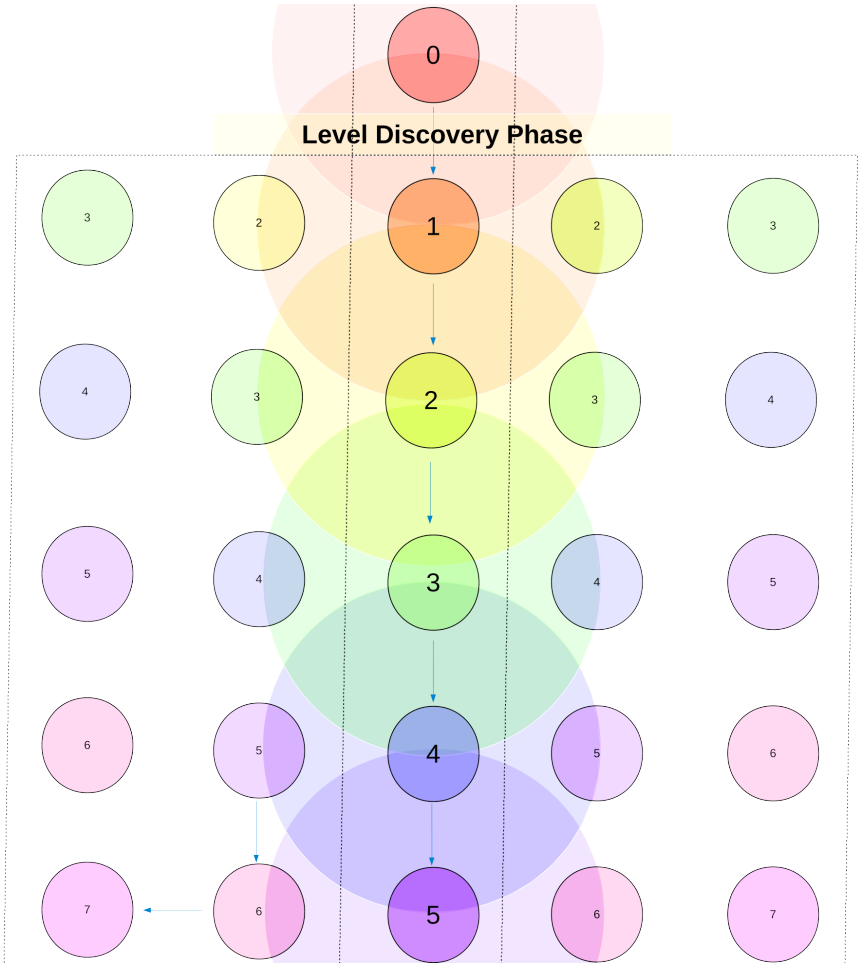

Figure 2.3: Example of Level Discovery Phase for a 5-by-5 grid network and a root node.

number in each node shows its assigned level. The levels are 0 to 7 in this example. The arrows illustrate the flow of the LDP packets. We show the arrows only for the center column just for illustration. In reality, there are three arrows out from the level-1 node, for example.

Synchronization Phase

In this phase, the sender-receiver pairwise synchronization as discussed in Section 2.2.2 is performed for every edge (or branch) of the hierarchical structure established in the earlier phase. A node, say node A, in level i sends out a beacon (a sync pulse packet) containing the identity, level, and timestamp $T1$. Due to established hierarchical structure, it is guaranteed that a node in level $i-1$, say node B, is a neighbor of node A and will receive the packet and send back an acknowledge packet with the times $T1, T2$, and $T3$. The originator node A in level i can thus calculate the clock offset Δ and adjusts its own clock to that of node B.

The steps in this phase are as following:

1. The message exchanges in this phase starts with the root node broadcasting a *time-sync* packet.

2. On receiving this packet, nodes belonging to level 1 wait for some random time before they initiates the pairwise synchronization message exchange with the root node. This randomization is to avoid the contention in medium access.

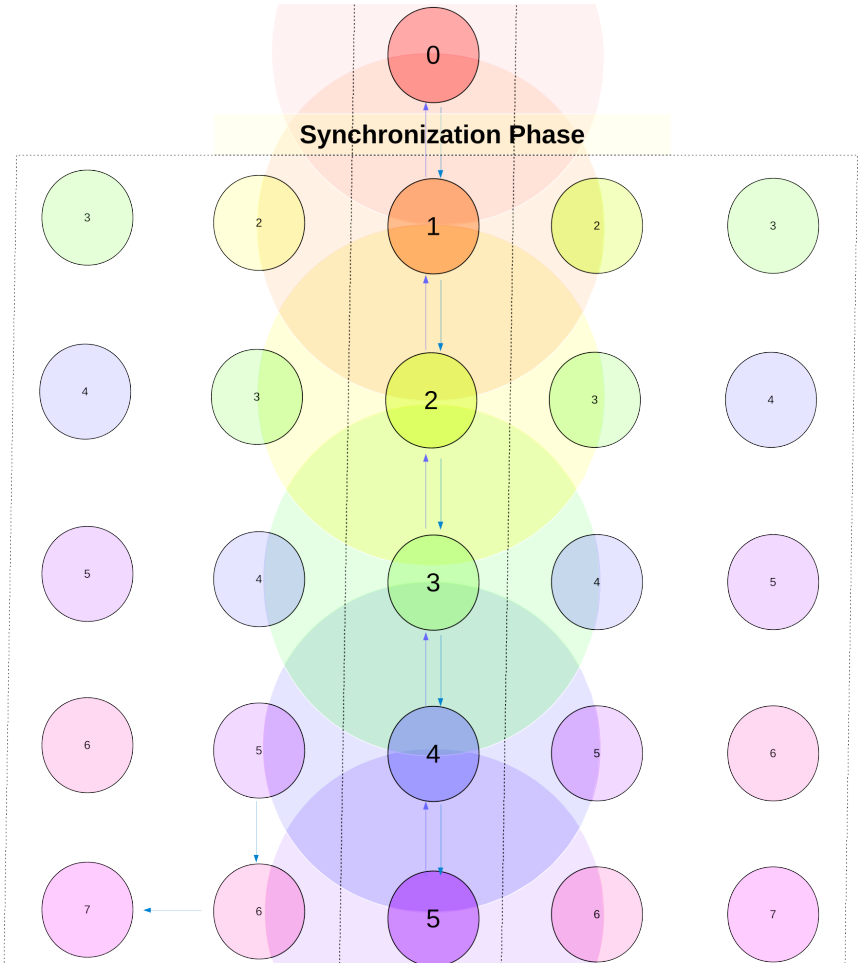

Figure 2.4: Example of Synchronization Phase for a 5-by-5 grid network and a root node.

3. On receiving back an acknowledgement, these nodes adjust their clock to the root node.

4. The nodes in level 2 will overhear this message exchange and back off for some random time before they start their own pairwise synchronization message exchange. This randomization is to make sure that the nodes in level 2 start the synchronization after the nodes in level 1 have already become synchronized with the root node.

5. This process is carried out until every node in the network is synchronized to the root node.

Note that in real sensor networks, packet collisions can happen often and retransmission of the beacon is necessary.

Fig. 2.4 illustrates the pairwise synchronization for the center column of the same 5-by-5 grid network in Fig. 2.3. In this example, there is only one node in level 1. When this node receives the time-sync packet of the root node (node 0), it sends out an LDP packet. The root node and all three level-2 nodes hear the packet but only the root node responds with an acknowledgment packet, while the level-2 nodes wait random time before starting their own pairwise synchronization with the level-1 node.

2.3 Adaptive Multi-Hop Time Synchronization

2.3.1 Motivation

In real hardwares and system, the clock of a sensor node typically drifts over time which makes the clock offset (the Δ in Section 2.2.2) varies over time and need to be estimated from time to time. Hence, synchronization must be done periodically to keep the synchronization error within a limit specified by the application.

To lengthen the *re-sync period*, which is the period where a synchronization happens, it would be better to estimate the rate that the clock drift. This rate is called *clock skew rate*. While the offset of two clocks is the actual time difference between them, the skew is the frequency difference between them [8].

For example, suppose the clock skew rate of node A relative to node B is kept fixed at 5 ms/s, then if two clocks are in sync initially at time 0, then at time 2 seconds later, the clock at A shows 2.010 s while that at B is 2.000 s. If we know this skew rate, we can simply adjust the clock at A automatically to compensate for such change. Thus, correct estimation of the skew rate makes the two clocks in sync for a longer duration and hence the re-sync period can be significantly extended. The estimation of the skew must be done regularly since in reality the skew is also not constant over time.

To be able to estimate the skew rate, we need to do the pairwise synchronization multiple times so that we know how the clock offset changes over time. However, this requires more number of messages exchanged and hence more energy consumption from the already-constrained battery of the sensor node.

There have been many proposed protocols to estimate the skew rate and enhancing the energy efficiency of the synchronization process, such as [8, 10, 12, 1].

2.3.2 AMTS

In this thesis we are interested in studying the Adaptive Multihop Timing Synchronization (AMTS) protocol [1] in different network topologies. AMTS is a time synchronization protocol that enhances TPSN to be more energy efficient for large-scale and long-lived sensor networks.

AMTS has several adaptive features such as adjusting the synchronization mode, the period of network-wide timing synchronization (called the *re-sync period*), and the joint clock offset and skew rate estimation in order to increase the re-sync period and achieve long-term reliability of synchronization. AMTS is also robust to high latencies and network delays due to the clock estimators.

AMTS consists of three phases:

1. Level Discovery Phase
2. Synchronization Phase
3. Network Evaluation Phase

The level discovery phase is exactly the same as that in TPSN. The synchronization phase is similar to that in TPSN but there are multiple pairwise synchronization message exchanges in order to estimate the clock skew. What makes AMTS much different from TPSN is the network evaluation phase, where the optimization of the network-wide synchronization parameters and synchronization modes happens.

Next we discuss the three phases in details.

Level Discovery Phase

The level discovery phase is exactly the same as that in TPSN, presented in Section 2.2.3. The result of this phase is a hierarchical structure or a spanning tree of the network, where the root or reference note is assigned level 0 and other nodes are assigned levels $1, 2, \ldots$ accordingly in the hierarchical structure.

Synchronization Phase

This phase performs pairwise synchronization between pairs of nodes by exchanging timing messages. The set of nodes involving in the pairwise synchronization depends on the synchronization mode of the network. There are two modes:

1. *always on* (**AO**): always maintain network-wide synchronization. In this mode, the pairwise synchronization is done in every edges of the spanning tree.

2. *sensor initiated* (**SI**): synchronize only when it needs to. In this mode, only the nodes participating in the particular multi-hop data transmission synchronize with each other. The nodes that are not required to transfer multi-hop data do not need to be re-synchronized.

AMTS performs joint clock offset and skew rate estimation in order to increase the resync period and achieve long-term reliability of synchronization.

To account for the clock skew, AMTS uses a more general clock model than the one in Section 2.2.3 which assumes a constant clock offset and propagation delay. As shown in Fig. 2.5, to estimate the skew a number N of pairwise synchronization must be performed in a synchronization period. Let the superscript (R) denotes the real time.

Let $T_{1,1} = 0$ be the reference time at the real *local* time $T_{1,1}^{(R)}$ of Node A and for the i-th ($i = 1, 2, \ldots, N$) pairwise message exchange,

$$T_{1,i} = T_{1,i}^{(R)} - T_{1,1}^{(R)} \tag{2.5}$$

and

$$T_{4,i} = T_{4,i}^{(R)} - T_{1,1}^{(R)} \tag{2.6}$$

denote the *relative* time stamps based on the local time observations $T_{1,i}^{(R)}$ and $T_{4,i}^{(R)}$ at Node A. The timestamps $T_{1,i}$ and $T_{4,i}$ in the ith message exchange are measured by the local clock of A, while the timestamps $T_{2,i}$ and $T_{3,i}$ are measured by the local clock of B.

The *reference* clock offset (time difference) at time $T_{1,1}^{(R)}$ is denoted as

$$\theta_A = \theta_A^{(R)} + T_{1,1}^{(R)} \tag{2.7}$$

where $\theta_A^{(R)}$ represents the *real* clock offset between the two nodes.

For the ith message exchange, the delay of the sync-pulse packet from A to B is modeled as $d + X_i$ and the delay of the ack packet from B to A is $d + Y_i$ where d represents the fixed portion of the delay and X_i, Y_i the random portions. The analysis of AMTS in [1] assumes that X_i, Y_i are i.i.d. Gaussian random variables with mean 0 and variance σ^2.

Suppose θ_B is the clock skew (frequency difference) of B's clock relative to A's clock and stays constant during the N message exchanges.

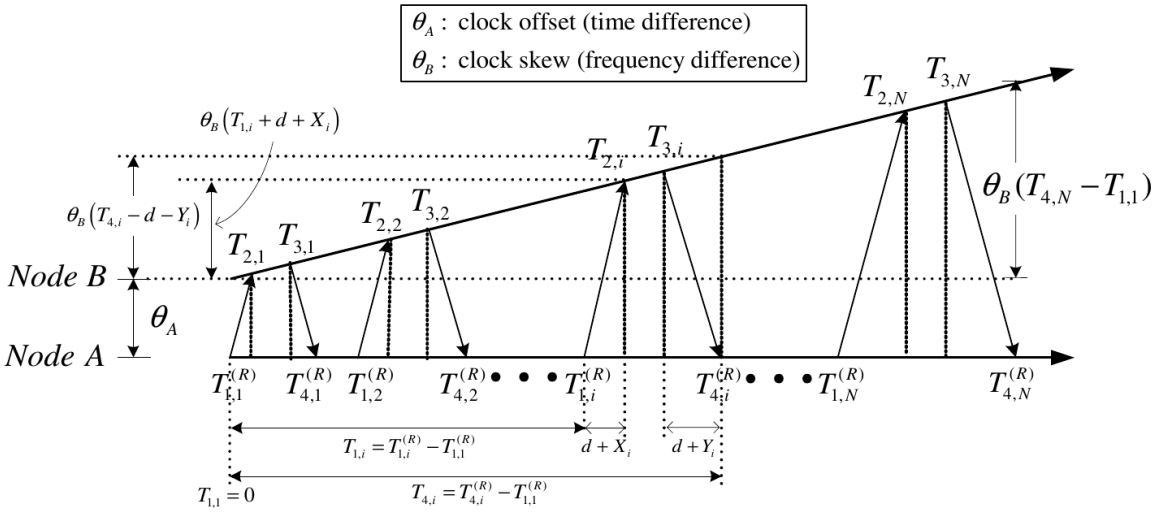

Figure 2.5: Message exchanges between master-slave nodes having clock offset and skew [1].

The timestamp at B in the ith uplink message is

$$T_{2,i} = (T_{1,i}^{(R)} + d + X_i) + \theta_A^{(R)} + (T_{1,i}^{(R)} - T_{1,1}^{(R)} + d + X_i)\theta_B \tag{2.8}$$

where the term $T_{1,i}^{(R)} + d + X_i$ is the real time the packet arrives at B, $\theta_A^{(R)}$ is the real clock offset between the two nodes, and $(T_{1,i}^{(R)} - T_{1,1}^{(R)} + d + X_i)\theta_B$ is the additional clock offset due to the skew θ_B during the time duration $T_{1,i}^{(R)} - T_{1,1}^{(R)} + d + X_i$.

Using (2.5) and (2.7), we can rewrite $T_{2,i}$ as

$$T_{2,i} = (1+\theta_B)(T_{1,i} + d + X_i) + \theta_A \tag{2.9}$$

which might be easier to understand as the time (at B) when the packet is received is $T_{1,i} + d + X_i$ scaled up by the factor $1+\theta_B$ and added by the offset θ_A between the two clocks.

Similarly, the timestamp at B in the ith downlink message $T_{3,i}$ is represented by

$$T_{3,i} = (1+\theta_B)(T_{4,i} - d - Y_i) + \theta_A \tag{2.10}$$

From (2.9), subtracting $T_{2,1}$ from $T_{2,N}$ removes d and θ_A, giving

$$T_{2,N} - T_{2,1} = (T_{1,N} - T_{1,1} + X_N - X_1) + \theta_B(T_{1,N} - T_{1,1} + X_N - X_1) \tag{2.11}$$

Similarly, from (2.10) subtracting $T_{4,1}$ from $T_{4,N}$ gives

$$T_{4,N} - T_{4,1} = (T_{3,N} - T_{3,1} + Y_N - Y_1) - \theta_B(T_{4,N} - T_{4,1} - (Y_N - Y_1)) \tag{2.12}$$

For short notations, let denote the difference between the first and the last timestamps of the $j = 1,2,3,4$ times as

$$D_{(j)} = T_{j,N} - T_{j,1} \tag{2.13}$$

Also let denote new random variables $P = X_N - X_1$ and $R = Y_N - Y_1$.

We can write (2.11) and (2.12) as

$$\begin{aligned} D_{(2)} &= D_{(1)} + P + \theta_B(D_{(1)} + P) \\ D_{(4)} &= D_{(3)} + R + \theta_B(D_{(4)} - R) \end{aligned}$$

Since P and R are i.i.d. Guassian random variables with mean 0 and variance $2\sigma^2$, we can determine the joint pdf and the likelihood function (see details in [1, 10]) and arrive at the maximum likelihood estimator of the clock skew θ_B as

$$\hat{\theta}_B = \frac{D_{(2)}^2 + D_{(3)}^2}{D_{(1)}D_{(2)} + D_{(3)}D_{(4)}} - 1 \tag{2.14}$$

Now, with the clock skew estimator, the clock offset estimator can also be estimated using the maximum likelihood principle as (see details in [1] and [10])

$$\hat{\theta}_A = \frac{\bar{U}' - \bar{V}'}{2} \tag{2.15}$$

where $\bar{U}' = \sum_{i=1}^{N} U_i'$, the mean of the sequence $\{U_i'\}_{i=1}^{N}$, and $\bar{V}' = \sum_{i=1}^{N} V_i'$, the mean of the sequence $\{V_i'\}_{i=1}^{N}$ and for $i = 1, \ldots, N$,

$$U_i' = T_{2,i} - T_{1,i} - \hat{\theta}_B T_{1,i} \tag{2.16}$$

$$V_i' = T_{4,i} - T_{3,i} + \hat{\theta}_B T_{4,i} \tag{2.17}$$

Hence, the given joint clock offset and skew estimators require N message exchanges in a sync period.

Network Evaluation Phase

In this phase, AMTS examines the total amount of message exchanges for synchronization during the last sync period and adjusts the duration (re-sync period as well as the number of timing message exchanges (beacons) per each pairwise synchronization) of the next sync period to minimize the average number of message exchanges. it selects the synchronization mode to be either always on (AO) (always maintain network-wide synchronization) or sensor initiated (SI) (synchronize only when it needs to) based on the network status and parameters. When the network traffic occurs rarely and synchronization delay is not a critical problem, selecting the SI mode is a better choice to save network resources [1].

In addition to the re-sync period and the sync mode, another important parameter that must be optimized is the number of timing message exchanges (beacons) per each pairwise synchronization, N. Increasing N increases the synchronization accuracy but in the meantime increases the number of message exchanges and hence decreases the energy efficiency. That is, there is a trade-off between accuracy and energy consumption.

1. Adaptive Clock Sync Algorithm (ACA)

AMTS uses ACA to select sync mode (AO or SI) to minimize the number of required timing messages.

To determine the number of required timing messages in each mode, let us define some network parameters as following:

- B: number of branches (or edges) in a spanning tree of the network.
- τ: re-synchronization period.
- $\bar{h}$: average number of hops per unit time.
- δ: latency factor reflecting the amount of allowed delay in data transmission [17].
- N: number of beacons per pairwise synchronization.

In the always on (AO) mode, the number of required timing messages per a pairwise synchronization is $2N$. Since there are total of B edges in the spanning tree ($B = n-1$ where n is the number of nodes in the network), the total number of timing messages per a sync period (which lasts for τ time units) is $2NB$, giving the average number of timing messages per unit time in AO mode as

$$\overline{M}_{AO} = \frac{2NB}{\tau} \tag{2.18}$$

On the other hand, in the sensor initiated (SI) mode, the sensor which performs synchronization requires on average $\bar{h}$ hops per unit time. Each hop requires one pairwise synchronization which in turns requires $2N$ messages exchanged per pairwise synchronization . Hence, the average number of timing messages per unit time in SI mode is

$$\overline{M}_{SI} = 2N\bar{h} \tag{2.19}$$

Note that during the synchronization phase, the root node collects the information of the total number of hops occurred in the last sync period and determines the average number of hops per unit time $\bar{h}$ in the network.

Hence, the decision to select which mode giving the minimum average number of timing messages per unit time is to select AO if the scaled average number of messages per unit time in AO mode is smaller than the average number of messages per unit time in SI mode, and vice versa. That is,

$$\delta\overline{M}_{AO} \lessgtr_{SI}^{AO} \overline{M}_{SI} \tag{2.20}$$

resulting in the rule

$$\tau \gtrless_{SI}^{AO} \frac{B\delta}{\bar{h}} \tag{2.21}$$

AMTS uses the scaled factor (latency factor) δ to represent the magnitude of the delay in the network and to allow the AMTS protocol to set preference for the AO mode. More delay-dependent networks assume a larger value of δ where $0 \leq \delta \leq 1$. For sensor networks always requiring network-wide synchronization (only AO mode is desired), δ should be set to 0. On the other extreme, for delay-dependent networks which we would prefer SI mode to save time used for synchronization, δ should be set close to 1.

During sync period τ, a portion of the time is used for synchronization while the leftover portion is for transmission of sensing data. Let τ_{max} denotes the maximum duration of the portion of sensing data transmission (we call it the *non-sync* phase). This value is specified from the hardware specification of the clock and the accuration of the estimators. Also let τ_{sync} denote the duration of the synchronization phase. Hence, the sync period τ is

$$\tau = \tau_{max} + \tau_{sync} \tag{2.22}$$

2. Optimum Number of Beacons (N) and Re-sync Period (τ)

When the network operates in the always-on (AO) mode, we need to determine the maximum non-sync phase τ_{max} and hence the re-sync period τ in (2.22) which depends on N, the number of beacons per pairwise synchronization and the accuracy of the clock offset and skew estimators.

To find the maximum non-sync phase τ_{max} which is the maximum duration of the sensor data transmission in a sync period, we need to first model how the clock timing mismatch (ε) develops with time. Right after synchronization is done for a node (say at time 0), the timing error is given as $\varepsilon = \varepsilon_0$, which is the clock offset error (the real clock offset minus the estimated clock offset from (2.15)). As time passes by by time t, the timing error is assumed to grow as

$$\varepsilon = \varepsilon_0 + \varepsilon_s \, t \tag{2.23}$$

where ε_s is the clock skew error (the real clock skew minus the estimated value from (2.14)). In [1], the clock offset and skew errors are modeled as normal distribution.

Hence, the maximum timing mismatch ε in (2.23) happens when $t = \tau_{max}$ and is a normal distribution, $\varepsilon \sim N(0, \sigma_\varepsilon^2)$ where

$$\sigma_\varepsilon^2 = \sigma_{\varepsilon_{0,N}}^2 + \sigma_{\varepsilon_{s,N}}^2 \tau_{max}^2 \tag{2.24}$$

In practice, we are given some specification on the maximum timing mismatch ε_{max} and the sync error probability P_s that the error (absolute term) stays below the maximum value ε_{max}. That is,

$$P_s = Pr(|\varepsilon| \leq \varepsilon_{max}) \tag{2.25}$$

where the probability can be looked up from the Q-table of normal distribution. For example, when P_s is set to be 0.1% and $\varepsilon_{max} = 10$ ms, then the standard deviation of the maximum timing mismatch is $\sigma_\varepsilon = 3.04$ ms.

As shown in [1], for $N \geq 2$ the clock skew can be estimated and the maximum non-sync phase τ_{max} is given as

$$\tau_{max}^{(N)} = (N-1)\sqrt{\frac{\sigma_\varepsilon^2 - (\frac{1}{N})\sigma_{\varepsilon_{0,1}}^2}{\sigma_{\varepsilon_{s,2}}^2}} \tag{2.26}$$

where we have used the superscript (N) to denotes the dependency on N. For $N = 1$,

$$\tau_{max}^{(1)} = \sqrt{\frac{\sigma_\varepsilon^2 - \sigma_{\varepsilon_{0,1}}^2}{\sigma_{\varepsilon_{s,1}}^2}} \tag{2.27}$$

where

- $\sigma_{\varepsilon_{0,1}}^2$ is the variance of the clock offset error using only one message exchange in the pairwise synchronization
- $\sigma_{\varepsilon_{s,2}}^2$ is the variance of the clock skew error using only two message exchanges, and
- $\sigma_{\varepsilon_{s,1}}^2$ is given by the specification of the crystal oscillator.

Hence, the re-sync period as a function of N is

$$\tau^{(N)} = \tau_{max}^{(N)} + \tau_{sync}^{(N)} \tag{2.28}$$

where $\tau_{sync}^{(N)}$ is the synchronization time with N beacons and is estimated at the root node for different values of N.

From (2.18), the average number of the message exchanges in the AO mode is

$$\overline{M}_{AO}^{(N)} = \frac{2NB}{\tau^{(N)}} \tag{2.29}$$

We can choose the optimal value of N that minimizes $\overline{M}_{AO}^{(N)}$.

Fig. 2.6 shows the flowchart of the AMTS. The sync period (SP) is the number of the sync period. So the protocol starts with the first period, then second and so on.

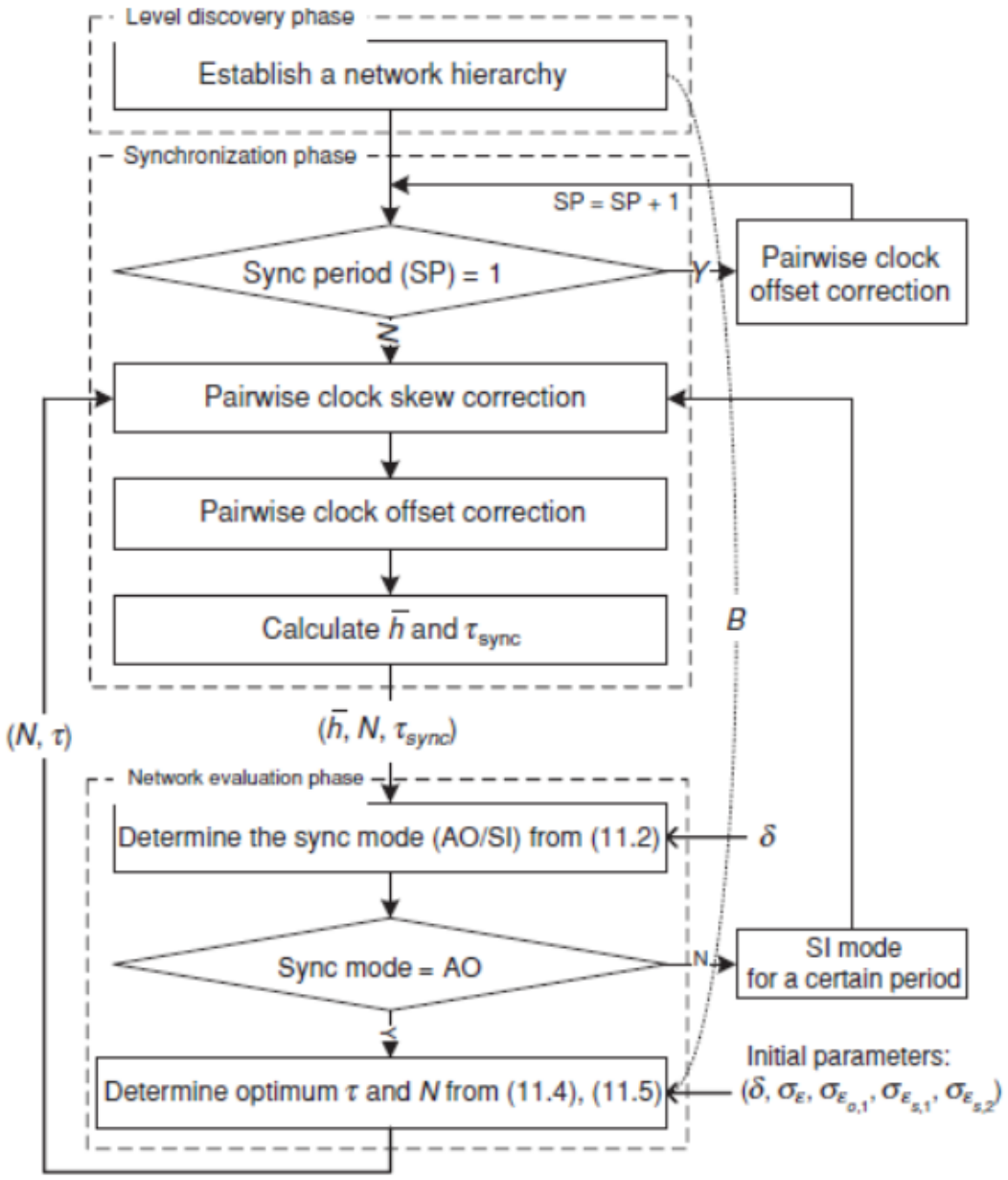

Figure 2.6: Flow Chart of AMTS protocl [1].

Chapter 3

Analysis Methodology

3.1 Purpose of Analysis

In this thesis we are interested in studying the Adaptive Multihop Timing Synchronization (AMTS) protocol [1] in different network topologies. In this chapter we discuss the topologies and the different performance metrics we use to study the performance of AMTS under these different topologies.

In general, a key design trade off for time synchronization protocols is that of clock accuracy versus network traffic and energy consumption. Keeping the clocks of all nodes synchronized with a reference node can require significant additional network traffic (overhead), which in turn requires energy consumption especially for packet transmission and reception. Although for small, simple network topologies (e.g. a chain) the overhead can be tolerable, as networks grow, the overhead may be too much for most applications.

This is due to two main reasons:

1. With large networks and different topologies (compared to a chain), the number of neighbors may vary significantly. This impacts on the number of transmissions and receptions, an in turn energy consumption.

2. With large networks significant radio interference between nodes can arise (and interference from other transmitters not in the WSN is more likely). This can lead to varying transmission delays (due to packet collisions and retransmissions) decreasing the accuracy of the pairwise synchronization.

In this thesis, we are interested in finding the performance of AMTS in the following three scenarios at different re-sync periods:

1. Chain

2. Grid with the center root node

3. Grid with the root node outside the grid

We hope that our thesis will enhance the understanding of both AMTS and TPSN protocols in general network topologies.

3.2 Network Topologies

The network topology or the spatial distribution of nodes in a WSN may impact the performance of synchronization protocols such as AMTS. To consider the impact of topology on AMTS, we analyze the following topologies:

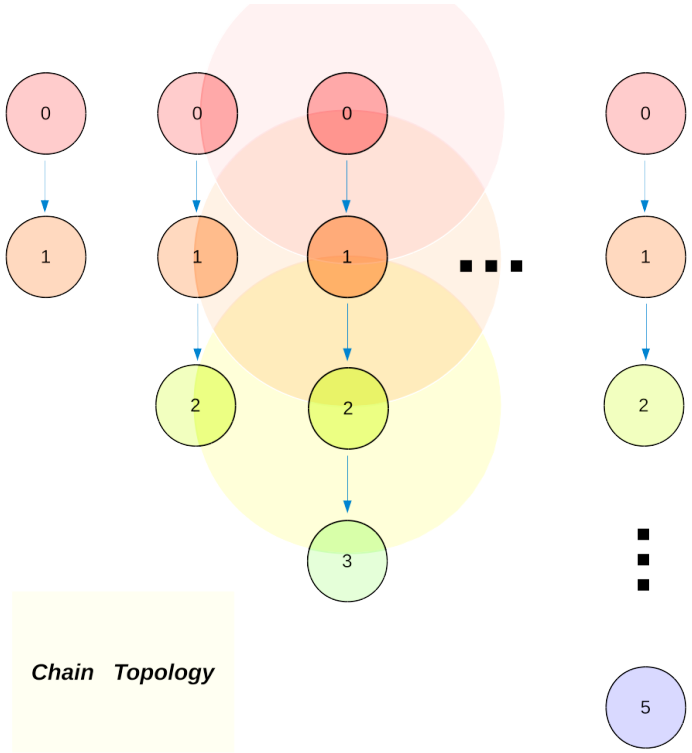

Figure 3.1: Illustration of the chain topology with $2,3,\ldots,6$ nodes ($n = 1,\ldots,5$)

1. **Chain (linear) topology**, consisting of n nodes and a root node. We let the first node of the chain be the root node. Fig. 3.1 illustrates the chain topology with $2,3,\ldots,6$ nodes ($n = 1,\ldots,5$). Each node is labeled with its level which is assigned in the level discovery phase of AMTS. This illustrations for this topology and others (below) have arrows showing the flow of the level discovery packet and also the sequential order of the pairwise synchronization. For convenience, the wireless transmission range is assumed to be just enough to cover the adjacent neighbors. For example, node 1 (level 1) can only communicate with nodes 0 and 2.

2. **Grid (square) topology with center root node** or **"Grid" topology**, consisting of $n \times n$ nodes with equal horizontal and vertical separation. The center root node is one of the $n \times n$ nodes and is located closest to the center of the grid. Fig. 3.2 illustrates the grid topology with $n = 2,3,4,5$, respectively. Notice that the root node (level 0) is nearest to the center of the grid. The arrows show a flow of the LDP packets and order of the pairwise synchronization. There are many possible ways for the flow of LDP packets and order of pairwise synchronization but they all give the same number of messages exchanged (excluding possible collision and retransmission) and the same energy consumption due to the equal distance between neighbors.

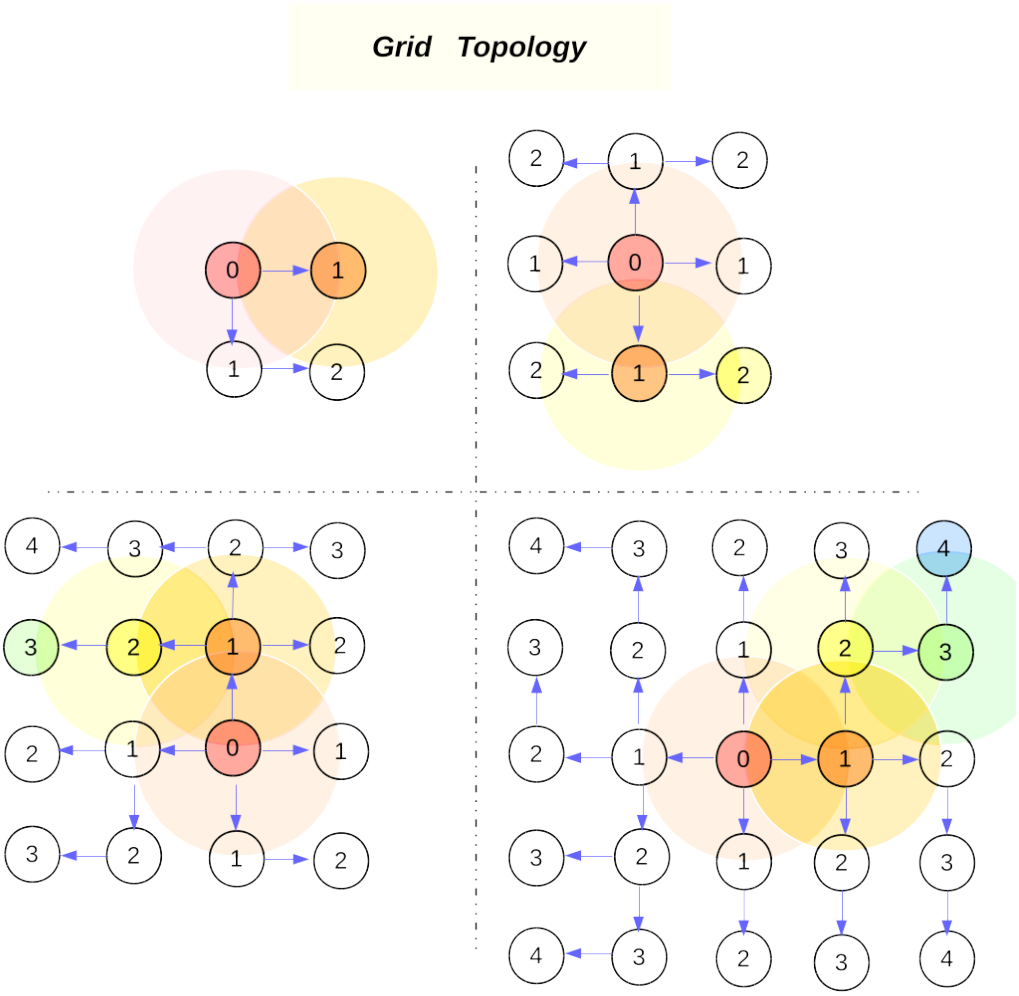

Figure 3.2: Illustration of the grid topology with $n = 2, 3, 4, 5$, respectively

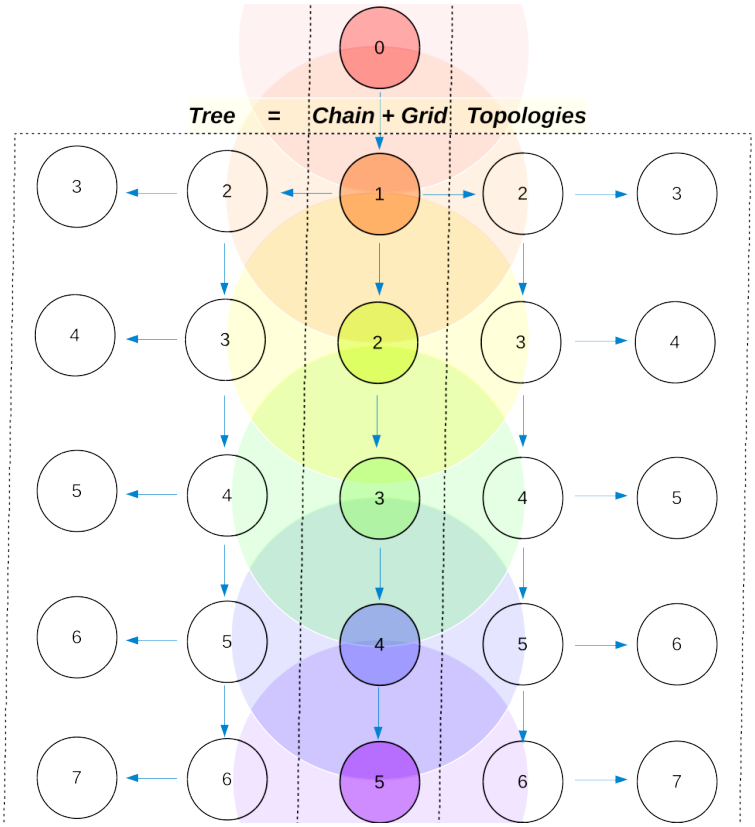

Figure 3.3: Illustration of the tree topology for 5-by-5 topology with the root node on the top of the grid.

3. **Grid with the root node outside the grid** or **"Tree" topology**[1], consisting of $n \times n$ nodes as in the grid topology but here the root node is put as an extra node and is located at the center on top of the grid. Fig. 3.3 illustrates the grid topology with $n = 5$ and the root node 0.

Note that while there are only 5 levels (levels 0 to 4) in the 5-by-5 "grid" topology (Fig. 3.2), there are 8 levels (levels 0 to 7) in the 5-by-5 "tree" topology (Fig. 3.3). This difference is due to the location of the root node. If the root node at the center, there are less number of levels in the spanning tree of the network. However, there is more chance for collision if there are more number of nodes in the same level.

[1]Due to our misunderstanding of the topology, we have been calling this topology a "tree", although every topology (including the chain and grid topologies), after the level discovery phase, ends up with a spanning tree. For convenience and short reference, we will keep using this name "tree" to refer to this specific topology. We will refer to the grid topology with the center root node as "grid". Please apologize for such misleading terminology.

3.3 Performance Metrics

To analyze the performance of AMTS we consider the following three metrics:

1. **Timing error** [seconds per day]: The timing error is defined as the average per-node timing error of all nodes in the network. The per-node timing error of a given node is defined as the maximum clock timing mismatch between the local clock at the node and the clock at the root node. Lower timing error means higher clock accuracy.

2. **Overhead** [bits per second per node]: There is communication overhead in synchronizing clocks. Specifically with AMTS, nodes need to discover levels and then periodically perform pairwise synchronization. The overhead is reported as the average number of bits per second sent per node.

3. **Energy consumption** [Watts]: While the processing required for AMTS is quite low, significant energy may be consumed by both transmitting packets and receiving packets. The energy is measured as the network-wide energy per unit time (which is actually a power quantity) average over time.

We use the following Mica2 motes model to determine the energy consumption [18]:

$$E = t_{idle} \times e_{idle} + t_{sleep} \times e_{sleep} + t_{Tx} \times e_{Tx} + t_{Rx} \times e_{Rx} \tag{3.1}$$

where for state s in one of the four states: idle, sleep, Tx, Rx, e_s is the energy consumed when the node is in state s and t_s is the time the node spends in that state s.

3.4 Simulation Tools

We use OMNet++ 4.6.0 network simulator [19] to simulate the different topologies with configurable parameters. MiXim-2.3 and iNet frameworks allow us to analyze AMTS protocol. We implemented the three phases of AMTS in ZigBee nodes within OmNet++, i.e. network level discovery, synchronization and network evaluation phases.

OMNET++ is a discrete event simulation environment publicly available since 1997 [20]. OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators [19].

MiXiM is an OMNeT++ modeling framework created for mobile and fixed wireless networks (wireless sensor networks, body area networks, ad-hoc networks, vehicular networks, etc.). It offers detailed models of radio wave propagation, interference estimation, radio transceiver power consumption and wireless MAC protocols (e.g. Zigbee). [21]

Fig. 3.4 shows an example screen of OMNet++ while Fig. 3.5 shows an example code.

3.5 Clock Model

At any point of time, if the time at UTC is t, the time in the clock of machine p is $C_p(t)$. In a perfect world, $C_p(t) = t$ for all p and t. This means $dC/dt = 1$. However, due to the clock inaccuracy discussed above, a timer (clock) is said to be working within its specification if

$$1 - p <= \frac{dC(t)}{dt} <= 1 + p \tag{3.2}$$

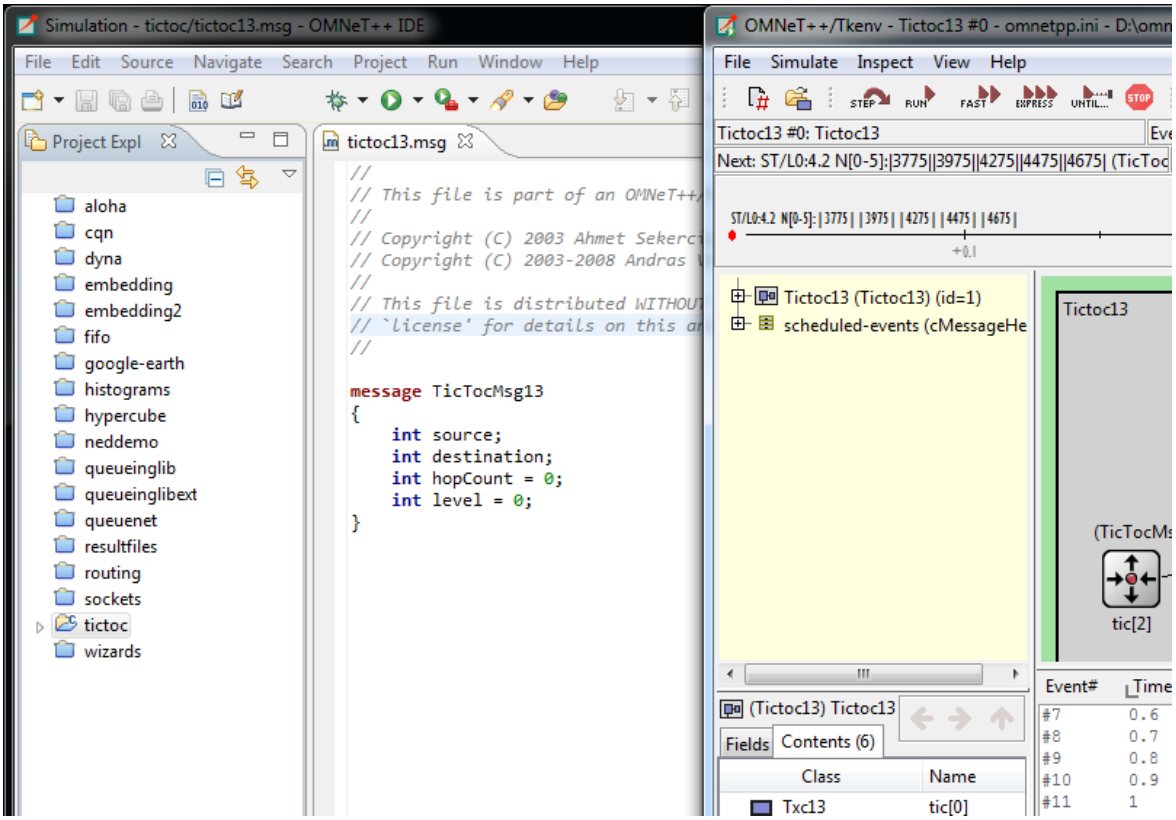

Figure 3.4: Example Screen of OMNeT++ Simulator

where constant p denotes the maximum skew rate. A typical value of the maximum skew specified by the manufacturer hardware. We note that the clocks of two nodes are synchronized to one common time at some point of in time, but they do not stay synchronized in the future due to clock skew. Even if there is no skew, the clocks of different nodes may not be the same. Time differences caused by the lack of a common time origin are called clock phase offsets.

3.6 System Parameters

- Topology: chain, grid, and tree topologies with n between 1 and 5.
- Synchronization protocol: We consider only two protocol: no synchronization (as a reference) and AMTS.
- Synchronization period: The time between performing synchronization with AMTS. Typically in the order of 10s of minutes.
- AMTS mode: AMTS supports two modes, AO and SI, and we collect results for each mode.

The parameters in our simulation-based analysis are in Tables 3.1, 3.2, and 3.3. We use the values the same as those in the AMTS paper [1].

```
class TicTocMsg13 : public ::cMessage
{
  protected:
    int source_var;
    int destination_var;
    int hopCount_var;
    int level_var;

  private:
    void copy(const TicTocMsg13& other);

  protected:
    // protected and unimplemented operator==(), to prevent accidental usage
    bool operator==(const TicTocMsg13&);

  public:
    TicTocMsg13(const char *name=NULL, int kind=0);
    TicTocMsg13(const TicTocMsg13& other);
    virtual ~TicTocMsg13();
    TicTocMsg13& operator=(const TicTocMsg13& other);
    virtual TicTocMsg13 *dup() const {return new TicTocMsg13(*this);}
    virtual void parsimPack(cCommBuffer *b);
    virtual void parsimUnpack(cCommBuffer *b);

    // field getter/setter methods
    virtual int getSource() const;
    virtual void setSource(int source);
    virtual int getDestination() const;
    virtual void setDestination(int destination);
    virtual int getHopCount() const;
    virtual void setHopCount(int hopCount);
    virtual int getLevel() const;
    virtual void setLevel(int level);
};
```

Figure 3.5: Example codes for OMNet++

Table 3.1: Energy Consumption Parameters

Parameter	Value	Units (SI)
e_{sleep}	0.06	mW
e_{idle}	0.138	mW
e_{Rx}	9.6	mW
e_{Tx}	1.1	mW
$e_{CPU(Active)}$	7.6	mW
$e_{CPU(Inactive)}$	0.237	mW

Table 3.2: Network Parameters

Parameter	Value	Units (SI)
Area size (X,Y)	(1000, 1200)	Gab 200m
Location (X, Y, Z)	(500, 100, 0), (500, 300, 0), ...	-
Simulation duration	1440	minutes
Max. Tx power	1.1	mW
Min. Radio Power (Sensitivity)	-100	dBm
Carrier Frequency	2.4	GHz
Network Type	Flood	-
Packet sizes (NW.and APP.Layer)	88	bytes

Table 3.3: AMTS Parameters

Parameter	Value	Units
B	depends on topology	
N	125	
SP	1, 2, 3, ...	periods
$\hbar$	1 to 5	hops
Synchronization Mode	AO and SI	modes
τ_{max}	2.094	msec
τ_{sync}	0.729094	msec
τ	10, 20, 30, 50, 60	minutes
Latency Factor (δ)	0 to 1	-
ε_{max}	10	ms
ε_o	50	μs
ε_s	4.75	$\mu s/s$
P_s	0.01	
σ_ε	3.33	ms
$\sigma_{\varepsilon_{o,1}}$	16.67	μs
$\sigma_{\varepsilon_{s,1}}$	1.58	$\mu s/s$
$\sigma_{\varepsilon_{s,2}}$	1.72	$\mu s/s$

Chapter 4

Simulation Results and Discussion

4.1 Organization of Results

We perform simulation of the AMTS protocol using OMNet++ for the three topologies (in Section 3.2): chain, grid, and tree for the parameters given in the previous chapter. We collect the three performance metrics: timing error, overhead, and energy consumption as described in Section 3.3. We also vary the re-sync period (τ) to study the effect of the network size. As a reference to the performance of AMTS, we also perform no synchronization (No-AMTS) for comparison.

For each set of parameters we run 15 collections and take the average for the performance metrics. The period of simulation is one day.

4.2 Explanation of Results

Here we present and discuss the simulation results for different topologies and different key parameters such as network sizes, re-sync periods, and operation modes.

4.2.1 Result: Chain Topology

For the chain topology (see illustration in Fig. 3.1) with different number of nodes, n, the three performance metrics are shown in Fig. 4.1 to 4.3. For chain topology, the chain length is equivalent to the number of nodes (excluding the root node).

1. Timing error

Fig. 4.1 shows that, for the same re-sync period (e.g., 20 minutes), the timing error slightly increases with the chain length n. This is because the clock offset error is accumulated from nodes in previous levels. However, the clock offset error is small.

When comparing AMTS with different re-sync period but for the same chain length, we see that having longer re-sync periods result in higher timing error, as we would expect from the model of the timing mismatch in (2.23), which says that the error grows with time.

Note that the timing error for no synchronization (No-AMTS) is worse than those with AMTS. Hence, the clock offset and skew estimator in AMTS seem to work properly.

2. Overhead

Fig. 4.2 shows the overhead at different chain length and re-sync period. The major part of the overhead is the number of the timing messages exchanged for pairwise synchronization. For chain length n, the number of branches (or edges) is also $B = n$. In the AO mode,

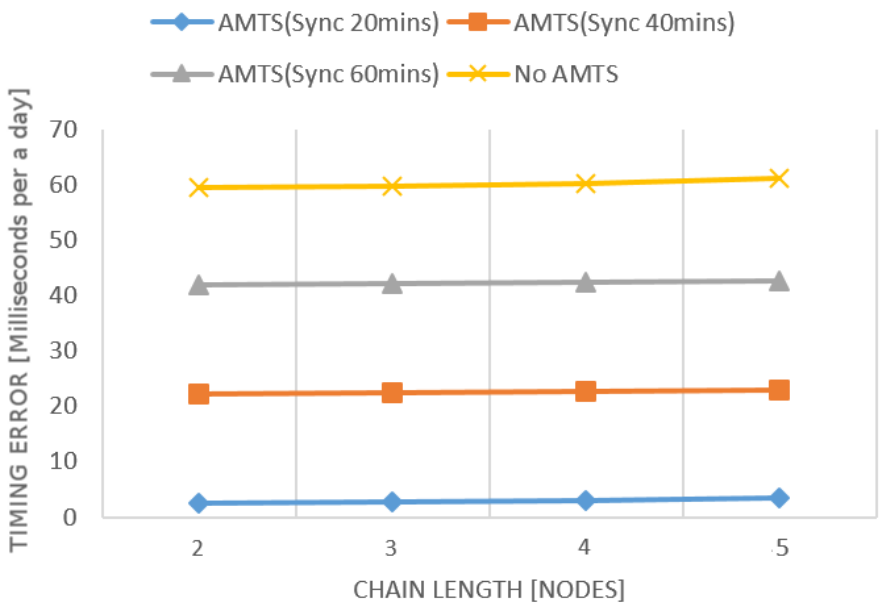

Figure 4.1: Chain topology: timing error performance

the average number of timing messages per unit time is expressed in (2.18) as

$$\overline{M}_{AO} = \frac{2NB}{\tau} = \frac{2Nn}{\tau} \tag{4.1}$$

where τ is the re-sync period and N the number of beacons per sync period.

The result in Fig. 4.2 shows this trend that the overhead depends linearly with the chain length n and inversely with the re-sync period τ. The timing error and overhead results in Fig. 4.1 and Fig. 4.2 confirm the

Note that the overhead for No-AMTS is zero since no synchronization is performed; hence, it is not shown.

3. Energy Consumption

Following the overhead result, the energy consumption result in Fig. 4.3 shows that more energy is required for more chain length and/or shorter re-sync period as expected. However, it is interesting to see that the energy consumption for AMTS at different re-sync periods is much larger than that without synchronization. This may mean that synchronization under AMTS requires high energy. This has to be investigated in future work.

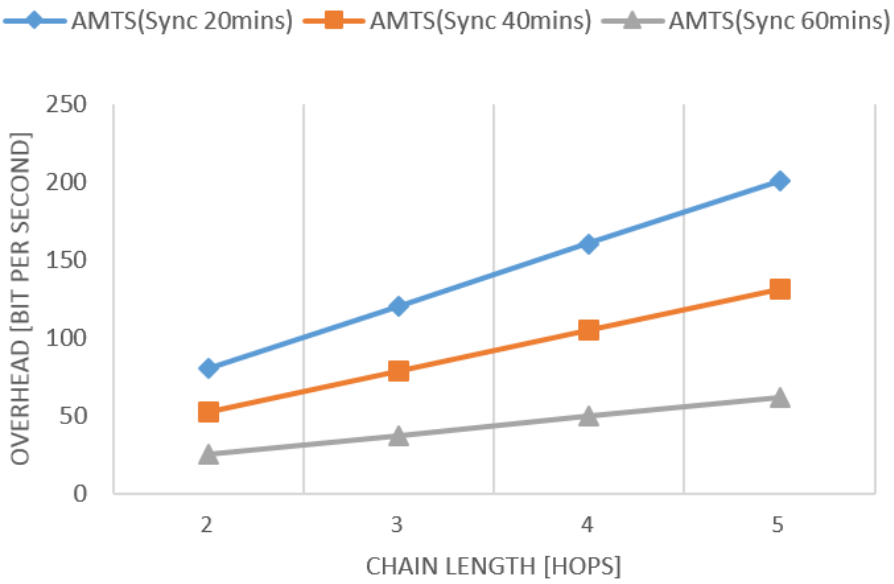

Figure 4.2: Chain topology: overhead performance

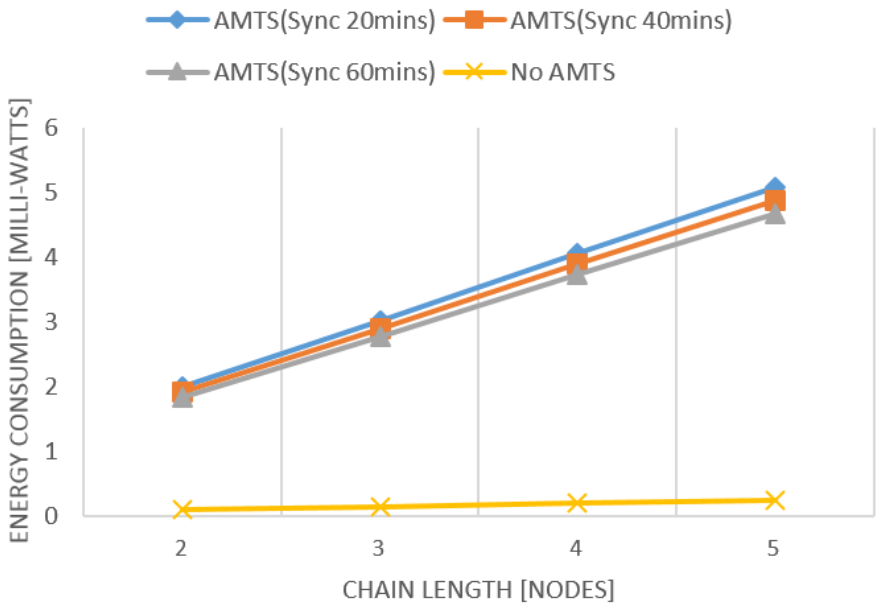

Figure 4.3: Chain topology: energy consumption performance

4.2.2 Result: Grid Topology

Fig. 4.4 to 4.6 show the results for the grid topology (see illustration in Fig. 3.2) with the network size is $n \times n$ where $n = 1, \ldots, 5$. The root node is assigned to be one of the nodes closest to the middle of the grid.

Unlike the chain topology where the network size and the chain length are the same, the grid topology has network size of n^2 while the grid width is n.

Also here the root node is put at the middle of the network, unlike in the chain topology where the root node is at the edge of the network.

1. Timing error

Fig. 4.4 shows that the timing error under AMTS decreases with the re-sync period but increases with n. However, we could not understand why the timing error seems to grow exponentially or quadratically with the value of n, although it is tempted to relate the timing error to be proportional to the network size which grows as n^2. But we think that the timing error should depend linearly with the number of levels in the network. As shown in Fig. 3.2, the number of levels (not counting level 0) for 2-by-2, 3-by-3, 4-by-4, and 5-by-5 network is 2, 2, 4, and 4, respectively. The number of levels seem to grow linearly with n. This is another reason we think the timing error should grow linearly with n. Again, this can be investigated in the future.

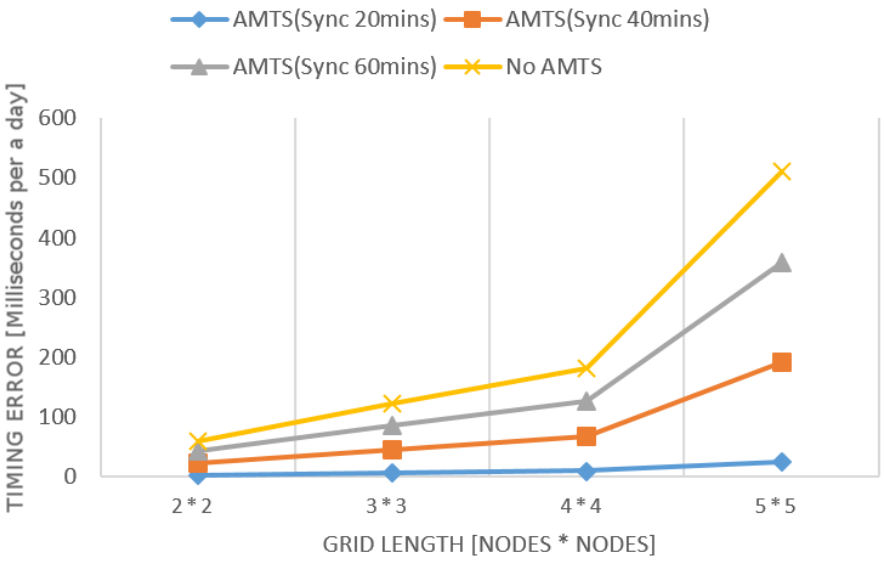

Figure 4.4: Grid topology: timing error performance

2. Overhead

Fig. 4.5 shows the overhead performance. Unlike the chain topology where the average number of timing messages per unit time is $\overline{M}_{AO} = \frac{2Nn}{\tau}$, here it is $\overline{M}_{AO} = \frac{2N(n^2-1)}{\tau}$ for the grid topology. Hence, we would expect that the overhead should grow with n^2. However, Fig. 4.5 does not show this and even worse, it shows that the overhead does not increase when $n = 4$ changes to $n = 5$. Again, this investigation is another future work. There may be other overheads that we need to take into account. However, at least the dependence of

the overhead on the re-sync period τ seems to be correct, with a larger re-sync period gives a smaller overhead.

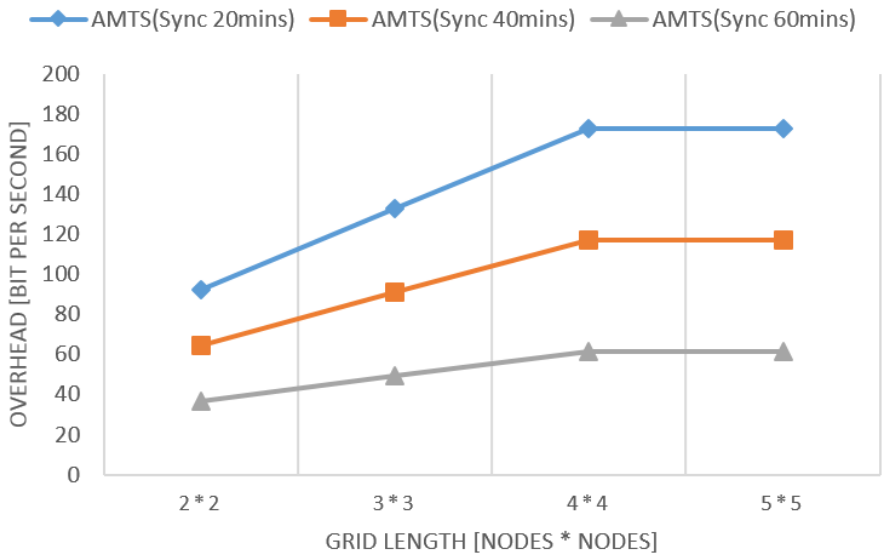

Figure 4.5: Grid topology: overhead performance

3. Energy Consumption

The energy consumption in Fig. 4.6 shows similar trend as that for the chain topology. However, now the difference between AMTS and No-AMTS seem small at $n = 2$.

4.2.3 Result: Tree Topology

Fig. 4.7 to 4.9 show the results for the tree topology (see illustration in Fig. 3.3) with the network size is $n \times n$ where $n = 1, \ldots, 5$. The root node is assigned to be an extra node (not counted as the n^2 nodes in the network) outside the grid. Now the network size is $n^2 + 1$ nodes and the root node is at the boundary of the network.

Since the results for the three performance metrics are very similar to those for the grid topology shown earlier, we do not go into detailed discussion here.

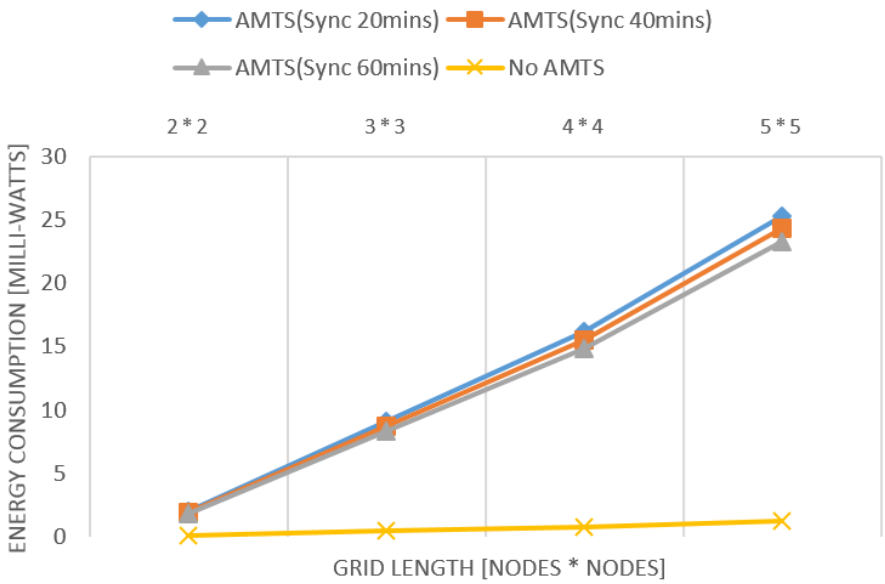

Figure 4.6: Grid topology: energy consumption performance

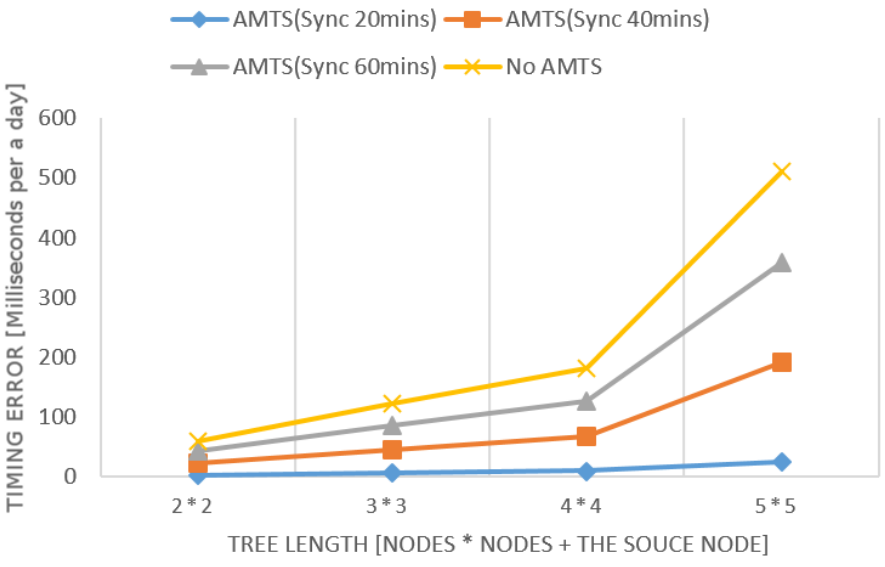

Figure 4.7: Tree topology: timing error performance

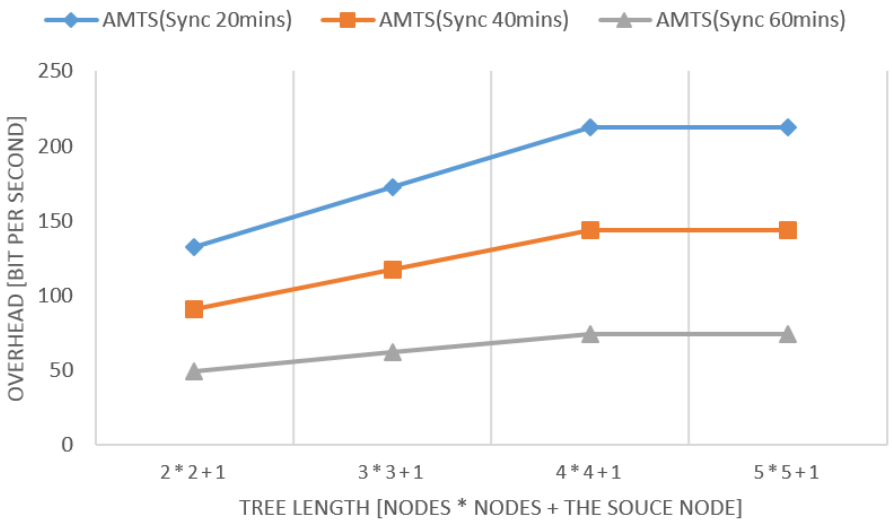

Figure 4.8: Tree topology: overhead performance

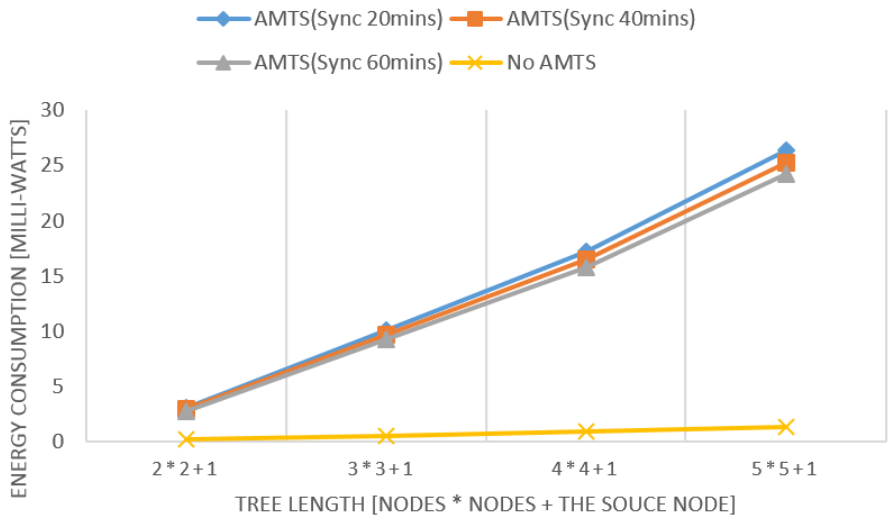

Figure 4.9: Tree topology: energy consumption performance

4.2.4 Result: Re-Synchronization Period

In this part, we consider specific network for each of the three topologies and vary the re-sync period τ. We set $n = 5$ for all topologies. That is, we consider the chain topology with chain length 5, the 5-by-5 grid topology, and the 5-by-5 tree topology. However, please be reminded that for the 5-length chain topology and the 5-by-5 tree topology, the root node is at the boundary of the network, while for the 5-by-5 grid topology, the root node is right at the middle of the network.

In addition, the main difference in the 5-by-5 grid topology, and the 5-by-5 tree topology is in the location of the root node and hence different number of levels. From Fig. 3.1 to 3.3, the number of levels (not counting the zero level) for these chain, grid, and tree topologies are 5, 4, and 7, respectively. The 5-by-5 grid topology has less number of levels than the 5-by-5 tree topology but each level has more number of nodes.

1. Timing error

As shown in Fig. 4.10, the timing errors for the three topologies become more different for larger values of the re-sync period. For the same value of re-sync period, the timing error for the tree topology is the worst, while that of the chain topology is the best.

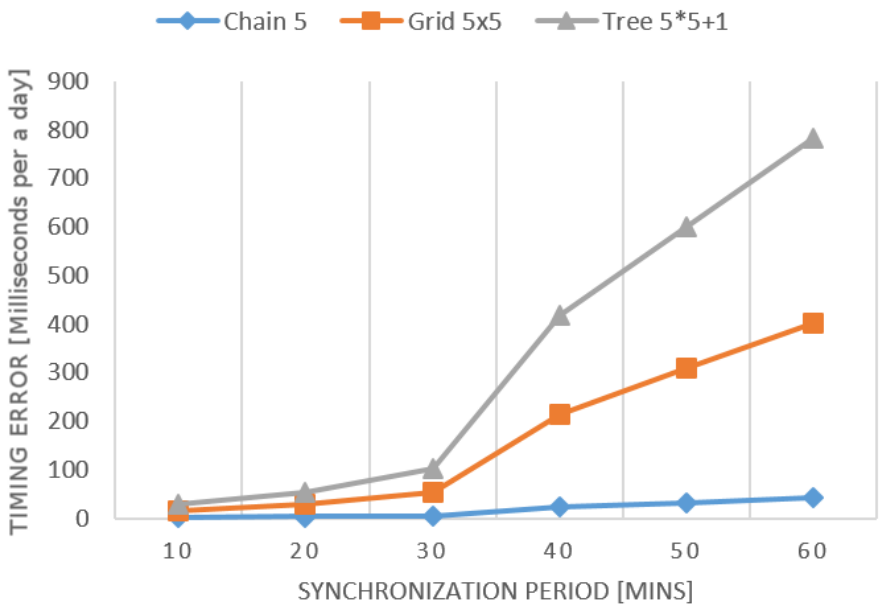

Figure 4.10: Different synchronize periods: Accuracy Performance

2. Overhead

As shown in Fig. 4.11, for any topologies the overhead decreases with the re-sync period as expected. Moreover, the overhead for the tree topology is the worst while that for the chain topology is the best. We do not expect such large different overhead for the grid and tree topology since the average number of timing messages per unit time is $\overline{M}_{AO} = \frac{2NB}{\tau}$ where the number of branches in the grid topology is $n^2 - 1$ while that of the tree topology is n^2 (no minus one since there are $n^2 + 1$ nodes in the tree but n^2 in the grid). For $n = 5$ here, $n^2 - 1$ and n^2 is almost the same and hence both grid and tree have almost the same

average number of timing messages per unit time. This large difference is left for future investigation.

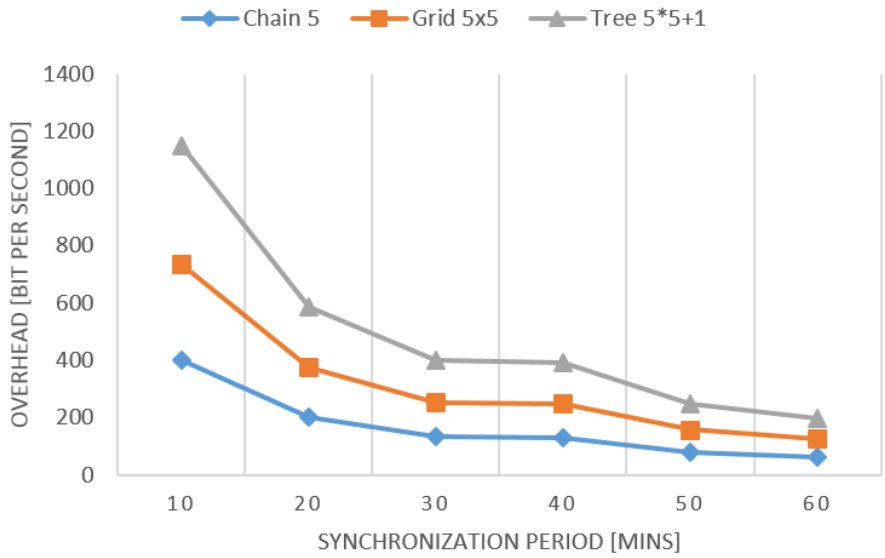

Figure 4.11: Different synchronize periods: Overhead Performance

3. Energy Consumption

As shown in Fig. 4.12, the energy consumption decreases with the re-sync period and it is the highest for the tree and lowest for the chain. Since energy consumption is related to the overhead, again we do not expect to see such large difference between the tree and the grid topology.

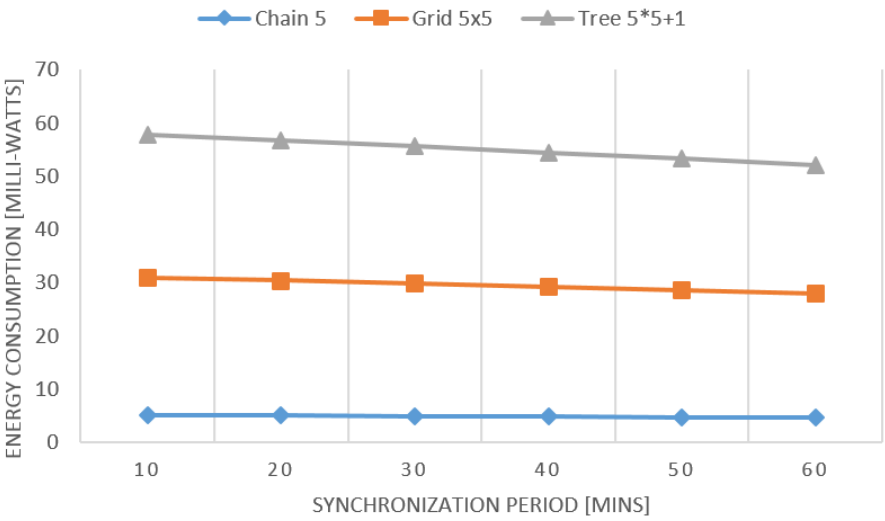

Figure 4.12: Different synchronize periods: Energy Consumption Performance

4.2.5 Result: Synchronization Mode

Here we compare the performance for the two synchronization modes of AMTS: always-on (AO) and session-initiated (SI) modes and the performance of No AMTS (no synchronization) in Fig. 4.13 to 4.15. The result depends on the number of nodes active in the SI mode. For the particular parameters we use, the SI mode has better performance than the AO mode in all of the three metrics: timing error, overhead, and energy consumption.

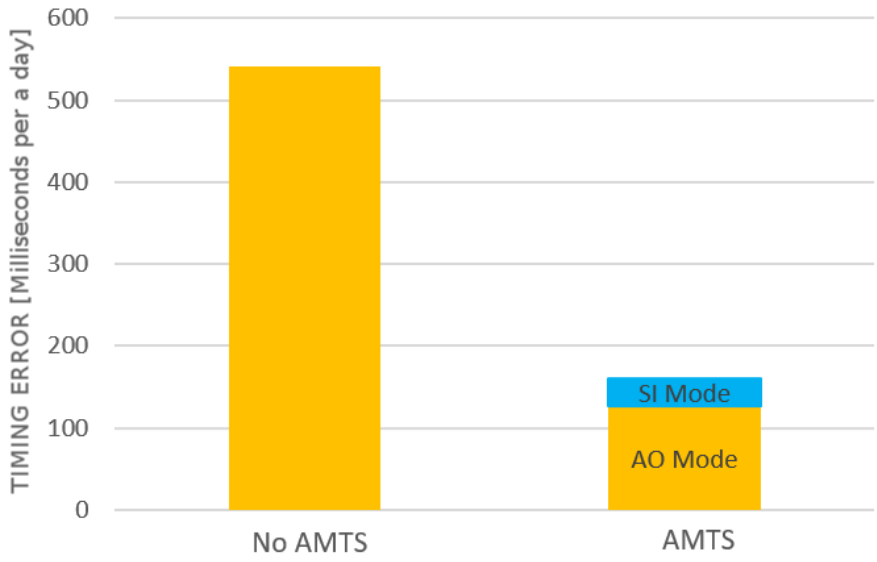

Figure 4.13: AMTS modes: Accuracy Performance

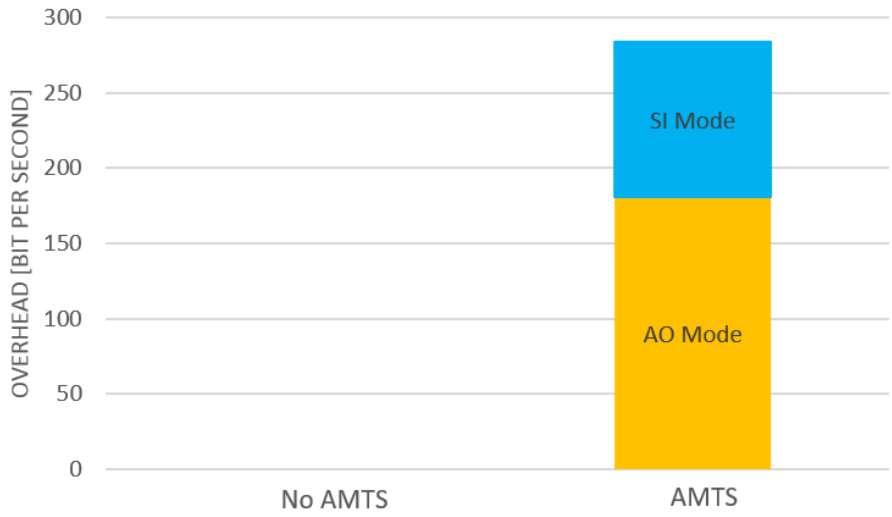

Figure 4.14: AMTS modes: Overhead Performance

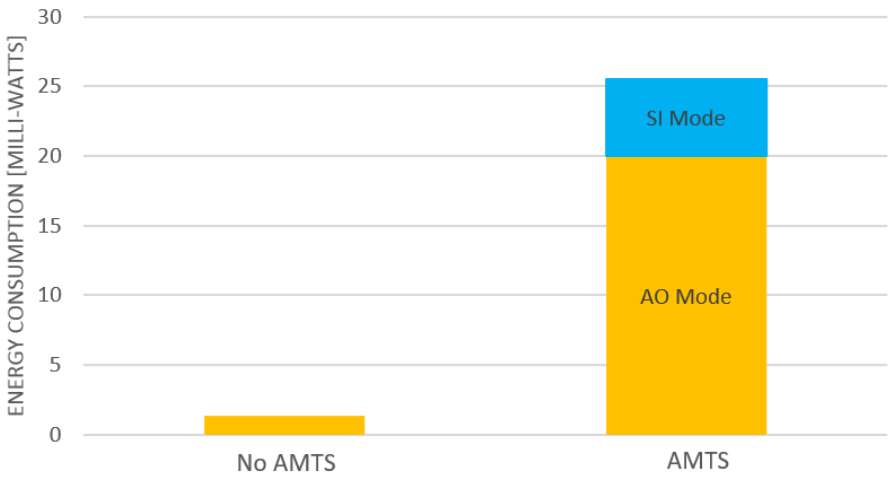

Figure 4.15: AMTS modes: Energy Consumption Performance

Chapter 5

Conclusions

5.1 Thesis Summary

In this thesis, we have performed a simulation-based study for the AMTS synchronization protocol which is an extension of the TPSN protocol and aims to increase the energy efficiency of time synchronization in large-scale and long-lived multi-hop wireless sensor networks.

We perform our simulation using the OMNeT++ simulator which is a discrete event simulation environment publicly available. OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators.

We focus our thesis to understand the performance of AMTS in three different network topologies: chain, grid, and tree topologies. The performance metrics we look at are synchronization error, overhead, and energy consumption.

The simulation results show that

1. For all topologies, as the network size increases, all three performance metrics get worse.

2. When the re-sync period increases, the timing error becomes worse while the overhead and energy consumption are better.

3. For the amount of sensing activity assumed in the simulation, all three performance metrics are better for SI mode, compared to the AO mode.

5.2 Future Work

There are some possible work that could be done in the future:

1. As we have noted in the discussion of the simulation results in Ch. 4, there are several behaviors and values of the results that we cannot quite explain. These could be investigated further for future work.

2. It is also interesting to see the performance of AMTS, when compared with TPSN, to see the improvement due to the Network Evaluation Phase, which is the third phase in AMTS. This phase is added by AMTS, on top of the first two phases which are proposed in TPSN. In the Network Evaluation Phase, the re-sync period, the number of beacons per pairwise synchronization, and the synchronization mode are optimized. In this thesis, we have only compared AMTS with no AMTS (no synchronization).

3. To see that AMTS can really work for large-scale networks, simulation should be done for large networks. In this thesis, we have only considered network with up to 26 nodes.

References

[1] K. L. Noh and E. Serpedin, "Adaptive multi-hop timing synchronization for wireless sensor networks," in *9th International Symposium on Signal Processing and Its Applications*, Feb 2007, pp. 1–6.

[2] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*. New York, NY, USA: ACM, 2003, pp. 138–149.

[3] E. Serpedin and Q. M. Chaudhari, *Synchronization in Wireless Sensor Networks: Parameter Estimation, Performance Benchmarks, and Protocols*. Cambridge University Press, 2009.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[5] W. Su and I. F. Akyildiz, "Time-diffusion synchronization protocol for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 384–397, 2005.

[6] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6–28, Dec 2004.

[7] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, Oct 1991.

[8] S. PalChaudhuri, A. K. Saha, and D. B. Johnson, "Adaptive clock synchronization in sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*. Berkeley, California, USA: ACM, 2004, pp. 340–348.

[9] K. L. Noh and E. Serpedin, "Pairwise broadcast clock synchronization for wireless sensor networks," in *2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2007, pp. 1–6.

[10] K. L. Noh, Q. Chaudhari, E. Serpedin, and B. Suter, "Analysis of clock offset and skew estimation in Timing-sync Protocol for Sensor Networks," in *IEEE Globecom 2006*, Nov 2006, pp. 1–5.

[11] W. Su and I. F. Akyildiz, "Time-diffusion synchronization protocol for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 384–397, April 2005.

[12] S. Ganeriwal, I. Tsigkogiannis, H. Shim, V. Tsiatsis, M. B. Srivastava, and D. Ganesan, "Estimating clock uncertainty for efficient duty-cycling in sensor networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 3, pp. 843–856, June 2009.

[13] M. Ye, C. Li, G. Chen, and J. Wu, "EECS: an energy efficient clustering scheme in wireless sensor networks," in *PCCC 2005. 24th IEEE International Performance, Computing, and Communications Conference, 2005.*, April 2005, pp. 535–540.

[14] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Network*, vol. 18, no. 4, pp. 45–50, July 2004.

[15] A. S. Tanenbaum and M. Van Steen, *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[16] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, Dec. 2002.

[17] A. A. Syed and J. Heidemann, "Time synchronization for high latency acoustic networks," in *25TH IEEE International Conference on Computer Communications (IEEE INFOCOM 2006)*, April 2006, pp. 1–12.

[18] K. Mikhaylov and J. Tervonen, "Novel energy consumption model for simulating wireless sensor networks," in *2012 IV International Congress on Ultra Modern Telecommunications and Control Systems*, Oct 2012, pp. 15–21.

[19] "OMNet++." [Online]. Available: https://omnetpp.org/

[20] A. Varga and R. Hornig, "An overview of the OMNeT++ Simulation Environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 60:1–60:10.

[21] "MiXim Project." [Online]. Available: http://mixim.sourceforge.net/