Timesynchronization in NATO Narrowband Waveform (NBWF)

Pål Josten

Master of Telematics - Communication Networks and Networked Services (2

Submission date:	June 2013
Supervisor:	Øivind Kure, ITEM
Co-supervisor:	Bjørnar Libæk, Forsvarets Forskningsinstitutt

Norwegian University of Science and Technology
Department of Telematics

ABSTRACT

A common notion of time is essential to avoid collisions in distributed radio networks using TDMA. Local clocks with limited accuracy are normally used to control time in each individual node. These clocks require regular synchronization to prevent the local time of these clocks from diverging. Synchronization of these clocks is controlled by a synchronization algorithm that estimates the correct time based on the information exchanged between nodes.

This thesis present a survey of synchronization algorithms that can be used by NATO Narrowband Wave Form (NBWF). NBWF requires that synchronization must be performed without the use of dedicated synchronization messages. It is possible to achieve this by using the correct reception of messages inside certain TDMA slots. Reception of these messages enables the receiving node to estimate the local time of the sender without transmission of an actual timestamp. This information can be used by a synchronization algorithm in the receiving node to estimate the correct local time.

The survey of potential synchronization algorithms shows that there are several potential candidates for NBWF. A hybrid algorithm, divided into several layer, fulfills most of the NBWF requirements, and it is the most promising candidate. The actual synchronization of nodes is performed in the lowest layer by Discrete Network Synchronization (DNS) algorithm. Simulations with the DNS algorithm show that this algorithm might not be the optimal choice for NBWF networks. This thesis recommends that a modified version of the hybrid algorithm, utilizing the CS-MNS algorithm in the lowest layer, should be considered for further work with NBWF networks.

OPPSUMMERING

En felles oppfatning av tide er viktig for å unngå kollisjoner i distribuerte radio nettverk som benytter TDMA. Lokale klokker med begrenset nøyaktighet blir vanligvis benyttet for å styre lokal tid i hver enkelt node. Bruken av disse klokkene krever regelmessig synkronisering for å unngå at disse klokkene driver fra hverandre. Denne synkroniseringen styres av synkroniseringsalgoritmer som estimerer riktig tid basert på informasjon utvekslet mellom nodene.

Dette dokumentet presenterer en oversikt over potensielle synkroniseringsalgoritmer som kan bli benyttet i NATO Narrowband Wave Form (NBWF). NBWF krever at synkroniseringen skal kunne gjennomføres uten bruk av dedikerte synkroniseringsmeldinger. Dette er mulig å gjennomføre ved å benytte korrekt mottak av informasjon i bestemte TDMA luker. Mottak av en slik luke vil gjøre det mulig for mottakende node å bestemme avsendertidspunkt uten at det overføres et tidsstempel. Dette kan benyttes av en synkroniseringsalgoritme i mottakende node til å estimere korrekt tid.

Oversikten over potensielle synkroniseringsalgoritmer viser at det finnes flere potensielle kandidater for NBWF. Av disse peker en lagdelt hybrid algoritme seg ut som den kandidaten som oppfyller de fleste av NBWF sine krav. Selve synkroniseringen av enkeltnoder gjøres i det nederste laget hvor algoritmen Discrete Network Synchronization (DNS) benyttes. Simuleringer av DNS algoritmen viser at denne ikke er det optimale valget for NBWF. Det anbefales derfor at det jobbes videre med en modifisert utgave av hybrid algoritmen, som benytter CS-NMS algoritmen i det nederste laget.

PREFACE

The work described in this thesis was carried out at FFI (Forsvarets Forskningsinstitutt) in the period between January and June 2013. The opportunity to work with FFI was offered to me as a part of the international Master of Science program in Telematics, offered by the Department of Telematics at the Norwegian University of Science and Technology (NTNU).

I would like to express my gratitude to my supervisor at FFI *Bjørnar Libæk* for help with validating the simulator, comments, advice and guidance throughout the work with the thesis. I would also like to thank *Svein Haavik* at FFI and my main supervisor at NTNU *Prof. Øivind Kure* for sharing their knowledge with me in our meetings. Finally, I would like to thank FFI for giving me the opportunity to work on this challenging topic

Kjeller, 27 June, 2013.

Pål Josten

CONTENTS

Abstract i
Oppsummering ii
Preface iii
Contents iv
List of tables vi
List of figures vii
Acronyms ix
1. Introduction 1
    1.1. Scope and Objective 3
    1.2. Limitations and assumptions 4
    1.3. Organization of the thesis 5
2. Background 6
    2.1. Clocks 6
    2.2. Narrow Band Wave Form 10
        2.2.1. NBWF Link layer 10
        2.2.2. Prerequisites for Synchronization in NBWF 12
    2.3. Synchronization design issues and NBWF 20
3. Synchronization algorithms 26
    3.1. Network Time Protocol (NTP) 26
    3.2. Ad Hoc networks 27
        3.2.1. Solutions created for IEEE 802.11 ad hoc networks 27
        3.2.2. Hybrid solutions 30
    3.3. Sensor networks 31
        3.3.1. One way message exchange 31
        3.3.2. Receiver-to-receiver synchronization 33
        3.3.3. Two-way message exchange 34
        3.3.4. High propagation delay sensor networks 34
    3.4. Summary of algorithms suitable for NBWF 35
4. The hybrid algorithm proposed for NBWF 39
    4.1. Synchronization algorithm 39
    4.2. Hybrid network algorithm 40
    4.3. Distributed decision making 42

4.4. Implementation in the simulator 43
5. Simulations 46
5.1. Simulator overview 46
5.1.1. Validation of the simulator 48
5.2. Simulation outputs and statistics 50
5.3. Simulations 51
5.4. Simulation results 56
5.4.1. Initial simulations on the DNS algorithm 56
5.4.2. DNS algorithm in topology 1 64
5.4.3. DNS algorithm in topology 2 67
5.4.4. DNS algorithm in topology 3 70
5.4.5. DNS algorithm in a random topology 74
Summary of DNS simulation results 79
5.4.6. Simulation of the CS-MNS algorithm 82
6. Discussion 87
7. Conclusion 90
7.1. Future Work 91
References 92
Appendices 98
A. NBWF reference model 98
B. Result of simulator validation 99

LIST OF TABLES

Table 1: Acronyms x
Table 2: Important definitions and symbols 10
Table 3: NBWF prerequisites 20
Table 4: Important synchronization design issues 25
Table 5: Potential algorithms 35
Table 6: Important simulation parameters 52
Table 7: Performance of the DNS algorithm 79

LIST OF FIGURES

Figure 1: Relationship between local time and real time [7] 7
Figure 2: Time-critical path 9
Figure 3: TDMA frame and slot structure [2] 11
Figure 4: Type and use of slots [2] 11
Figure 5: NBWF synchronization points 14
Figure 6: NBWF time-critical path 15
Figure 7: NBWF synchronization 16
Figure 8: Two-way message exchange 21
Figure 9: Deterministic synchronization within a bound 22
Figure 10: State machine for the hybrid algorithm [32] 41
Figure 11: Pseudo code for the DNS algorithm 44
Figure 12: Model of simulator clocks 46
Figure 13: Synchronization Points used for validation 49
Figure 14: Topology 1 – Fully connected network 53
Figure 15: Topology 2 - A chain of nodes 54
Figure 16: Topology 3 – Two clusters of nodes 55
Figure 17: Effect of N_i for a static network of 10 nodes 56
Figure 18: 10 iterations of networks with 10 nodes and N_i=9 57
Figure 19: Topology 1 and topology 2 with N_i=Number of nodes 58
Figure 20: Simulation results for topology 2 networks 59
Figure 21: Topology 3 network with 20 nodes and different N_i values 60
Figure 22: Boxplot of successfully synchronized iterations of Topology 3 networks with 20 nodes and different N_i values 61
Figure 23: Topology 2 network with 5 nodes and different n_i values 62
Figure 24: Topology 1 network with 10 nodes and different n_i values 63
Figure 25: Rejected Iterations (topology 1) 64
Figure 26: Convergence for networks with 50 nodes (Topology 1) 65
Figure 27: Maximum relative offset with 99% confidence interval (topology 1) 66
Figure 28: Rejected iterations (topology 2) 67
Figure 29: Convergence for networks with 6 nodes (Topology 2) 68
Figure 30: Convergence for networks with 10 nodes (Topology 2) 68
Figure 31: Maximum relative offset with 99% confidence interval (topology 1) 69
Figure 32: Rejected iterations (topology 3) 70
Figure 33: Convergence for networks with 20 nodes (Topology 3) 71

Figure 34: Convergence for networks with 20 nodes and maximum distance of 15 Km (Topology 3) 72
Figure 35: Maximum relative offset with 99% confidence interval (topology 3) 73
Figure 36: Rejected iterations (random topologies high initial relative offset) . 74
Figure 37: Rejected iterations (random topologies low initial relative offset) .. 75
Figure 38: Convergence for networks with 50 nodes (random topology) 76
Figure 39: Maximum relative offset with 99% confidence interval (random topology) 77
Figure 40: Random change in topology for a network with 50 Nodes 78
Figure 41: Comparison between DNS and CS-MNS algorithm in topology 1 networks. 82
Figure 42: Comparison between DNS and CS-MNS algorithm in networks with random topologies 83
Figure 43: Convergence for networks with 50 nodes (CS-NMS topology 1).... 83
Figure 44: Convergence for networks with 50 nodes (CS-NMS random topology) 84
Figure 45: Comparison between DNS and CS-MNS algorithm in topology 2 networks 85
Figure 46: Comparison between DNS and CS-MNS algorithm in topology 3 networks. 85
Figure 47: NBWF reference model 98
Figure 48: Simulated and calculated data 101
Figure 49: The connection between synchronization points 102

ACRONYMS

Acronym	Description
AIE	Air Interface Encryption
ARQ	Automatic Repeat Request
ASP	Automatic Self-time-correcting Procedure (Algorithm)
ATSP	Adaptive Timing Synchronization Procedure (algorithm)
BBS	Black Burst Synchronization (Algorithm)
CDS	Connected Dominating Sets
CPM	Continuous Phase Modulation
CSMA	Carrier Sense Multiple Access
CS-MNS	Clock Sampling Mutual Network Synchronization (Algorithm)
DNS	Discrete Network Synchronization
DU	Dual Use slot
EW	Electronic Warfare
FFI	Forsvarets forskningsinstitutt
FTSP	Flooding Time Synchronization protocol
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
GU	General Use slot
GW	Gateway
HM	Heard Master
IBSS	Independent Basic Service Set
ID	Identity
IED	Improvised Explosive Device
IEEE	Institute of Electrical and Electronics Engineers
KFMP	Kalman Filter using Multiple Parents (Algorithm)
LLC	Logical Link Control
LNT	Local Network Time
LOS	Line of Sight
MAC	Medium Access Control
MANET	Mobile Ad hoc Networks
MASP	Modified Automatic Selftime-correcting Procedure (Algorithm)
MATSF	Multihop adaptive timing synchronization function
MELPe	Mixed-Excitation Linear Predictive enhanced

MV	Multicast Voice slot
NATO	North Atlantic Treaty Organization
NBWF	Narrow Band Wave Form
NTNU	Norwegian University of Science and Technology
NTP	Network Time Protocol
PTSF	Predictive Timer Synchronization Function (Algorithm)
RBCI	Radio Based Combat Identification
RBS	Reference Broadcast Synchronization (Algorithm)
SA	Situational Awareness
SDR	Software Defined Radio
SF	Superframe slot
TACSAT	Tactical Satellite
TATSP	Tired Adaptive Timing Synchronization Procedure (Algorithm)
TDMA	Time Division Multiple Access
TPSN	Timing-sync Protocol for Sensor Networks
TSF	Timing Synchronization Function
UHF	Ultra High Frequency
UTC	Universal Time
VHF	Very High Frequency

TABLE 1: ACRONYMS

1. Introduction

The introduction of Software Defined Radios (SDR) makes it possible to create waveforms that can be used on radios from different manufacturers and allow these radios to interact directly. Previously, this has been difficult because manufacturers created radios that only supported their own waveforms. Earlier attempts at standardization failed due to industrial protectionism [1], but use of multi-national task forces has increased the focus on this problem. Therefore, NATO has taken the opportunity created by the introduction of SDR and started work on new NATO owned waveforms for interoperable radios. The main objective is to create waveforms for VHF and UHF communication, resulting in the need for both a narrowband and wideband waveform.

NATO owned waveforms will enable direct interaction between military units, regardless of the tactical level, equipment providers and available infrastructure. Recently this has only been possible for voice traffic at low tactical levels. Full interaction, including data transfer, required complex gateways that had to be placed at higher tactical levels than the units using the gateways.

Work on the narrowband waveform (NBWF) is performed by a NATO workgroup[1] where Norway and Canada are the main contributors. NBWF must support the following requirements [2]:

- Simultaneous voice and data on channels with limited capacity.
- Support SDR.
- Provide information security and efficient signal coding.
- Push-to-talk voice capability with low delay and jitter.
- Friendly forces tracking and radio based combat identification (RBCI).
- Operation on a single 25 KHz fixed frequency radio channel.
- Relaying of voice calls over two relays.
- Prepared for frequency hopping as this functionality will be a future option.

Forsvarets Forskningsinstitutt (FFI) is leading the Norwegian effort. Currently FFI is focusing on the NBWF link layer. Much of the groundwork is completed, and Time Division Multiple Access (TDMA) is chosen as the medium access technology. The use of TDMA makes time synchronization an

[1] NATO C3B – CIS CaP-LOS Comms CaT

essential function for NBWF. All nodes must have a common notion of time to determine the correct start of TDMA slots and avoid collisions. A typical NBWF network will be a distributed system, where all nodes are connected via radio links. Maintaining a common notion of time that is fault tolerant is a well-known problem for distributed systems [3] [4]. There are several possible solutions that can achieve this. Three of them are described briefly here.

In theory, all nodes could be equipped with extremely precise physical clocks, synchronized at the time of production. This should enable sufficient accuracy for the lifespan of the system. Chip scale atomic oscillators such as SA.45s CSAC [5] are already commercially available. These devices promise to deliver a timing signal that is extremely accurate and stable. While the specifications are promising, there are a few considerations that make this approach unpractical for NBWF. First of all NBWF should work on a wide range of military radios and considerations such as chip size and price might make chip scale atomic clocks unsuited for most radios. Finally, the power consumption is still relatively high. NBWF need another solution, common to all radios.

The use of an external source broadcasting time signals, such as GNSS[2], could provide each node with highly accurate time. NBWF supports RBCI. As a result, a large part of the nodes should have a built in GNSS receiver or a GNSS receiver connected. These receivers are capable of delivering accurate timing signals that could be used to synchronize time. This solution is limited by GNSS coverage, and it is vulnerable to jamming of the GNSS signal. Nodes that loses GNSS signal will also lose the time reference provided by GNSS. NBWF cannot assume that all radios have access to GNSS or similar technologies at all times.

Quartz clocks are the most common solution in distributed systems. These clocks are small, inexpensive devices with limited precision. Nodes with quartz clocks need to be periodically synchronized in order to maintain time that is accurate enough to avoid TDMA slot collisions. Each node uses a synchronization algorithm to estimate the correct time and adjust the local clock. The synchronization algorithm bases the estimation on information that is exchanged between the nodes in the network. This is the intended approach for NBWF.

FFI wants to use the information provided by the TDMA structure in the link layer as a basis for time synchronization. The goal is to synchronize all nodes in the network without the use of dedicated synchronizations messages.

[2] GNSS: Global Navigation Satellite Systems such as GPS.

The main idea is to use correct reception of certain TDMA slots as synchronization points. The local time of the sender can be estimated by all receives as long as the information in these TDMA slots is received correctly. This approach should work as long as network affiliation and coarse synchronization are previously performed.

1.1. Scope and Objective

This thesis will try to find an algorithm that can be used to maintain synchronization after network affiliation and coarse synchronization. The algorithm must fulfill the following requirements:

- The maximum allowed time difference between any nodes in the network is 1 ms.
- The proposed solution must be able to maintain synchronization for the remaining network even if one or more nodes are lost.
- The proposed solution must limit the use of bandwidth and synchronization should be based on information piggybacked in existing traffic. Dedicated synchronization messages should be avoided.
    - A small amount of information, 1-2 bytes, can be piggybacked on existing traffic if necessary.

In addition to these mandatory requirements, the following requirements should be considered:

- The proposed solution should be able to utilize external synchronization, from a standard source of time such as Universal Time (UTC).
- The proposed solution should consider the impact of different demanding topologies, including the use of relaying.

The objective of this thesis will be to find an algorithm that can prevent collision of TDMA slots by maintaining precise local time in all nodes. This algorithm has to determine the time difference between nodes, modify the local clock and reduce the difference between the local clocks in all nodes. The most promising algorithm will be simulated, and the result of this simulation will indicate if this algorithm fulfills the NBWF requirements. The following research questions will be answered in order to find the best algorithm:

Question 1: *What does NBWF demand from a synchronization algorithm?*

Chapter 1 gives an overview of the requirements presented by FFI. NBWF and synchronization theory will be analyzed briefly to find additional requirements for the synchronization algorithm.

Question 2: *Which time synchronization algorithms are best suited for NBWF?*

A survey of work related to time synchronization in radio networks will provide a list of potential algorithms. Of these algorithms, one will be selected for further study and simulation.

Question 3: *Do the algorithm fulfill NBWF requirements for precision?*

Simulations on the selected algorithm must be performed to answer this question.

1.2. LIMITATIONS AND ASSUMPTIONS

The following assumptions have been made, based on input from FFI and the status of the NBWF link layer at startup of the thesis:

- The described algorithm is intended for previously established networks. It is assumed that all nodes have performed network affiliation and coarse synchronization. This have the following consequences:
  - All nodes will be able to decrypt the signal and detect receive TDMA slots.
  - The initial clock offset will be relatively small, because coarse synchronization is performed. Coarse synchronization is assumed to have a precision well below 1 ms.
- Additional network management mechanisms are needed in order to handle functions such as network affiliation, late entry, network merge and nodes leaving the network. These functions require dedicated synchronization messages and will not be a part of the algorithms described in this thesis. These mechanisms will not be discussed in detail, but they may affect the choice of algorithm.

The following limitations have been made, due to the complexity of the problem researched and the limited time available:

- The thesis will suggest an algorithm that can be used to maintain TDMA synchronization in the network. This algorithm depends on previous coarse synchronization and must be a part of a synchronization protocol providing additional management mechanisms. The implementation of the algorithm in this protocol is left for later work.
- Simulations will be performed in the Simula/DEMOS language because this programming language is already known by the author and supervisor.
- The thesis will evaluate existing algorithms and propose algorithms that fulfill the requirements of NBWF. The thesis will not propose new algorithms to meet these requirements.

1.3. ORGANIZATION OF THE THESIS

This thesis is organized in the following chapters:

- Chapter 2 presents an overview of time synchronization concepts and NBWF. The overview provides a foundation that will be used to compare different algorithms.
- Chapter 3 will give a brief overview of solutions and scientific works related to time synchronization. Potential algorithms for NBWF are compared and selected, based on this survey. The objective is to identify one algorithm that can be studied by simulation.
- Chapter 4 is a detailed description of the algorithm selected for simulation and how it can be implemented in NBWF.
- Chapter 5 describes the simulator, simulations and the results of the simulations
- In chapter 6, the obtained results are discussed.
- Chapter 7 answers the research questions posted in chapter 1. This chapter is also the conclusion of this thesis.

2. BACKGROUND

The progression of time has to be precisely maintained to ensure that different events happens at correct points in time. As an example, a NBWF node has to know the correct time to identify the correct TDMA slot and avoid collisions. It is useful to have a reference when comparing time in different clocks. This reference time is often labeled as *real time* or *reference time,* and it can be based on an imaginary perfect clock or be approximated by the use of extremely accurate clocks [3] [6].

The *global time* of a system is the common notion of time, shared by all the nodes in the system [3]. Inaccurate clocks make it impossible to have the same notion of time across all the nodes of a distributed system. As a result, nodes in distributed systems can only approximate global time and exact global time becomes an abstract notion.

2.1. CLOCKS

The physical clocks used in most distributed system are hardware devices based on an oscillator and a counter. Clocks are usually based on quartz-stabilized oscillator, but more accurate clocks use atomic oscillations [7]. This setup generates interrupts, or *ticks,* at a given rate. This marks the physical progress of time in the clocks. The term *software clock* or *logical clock* is used to describe the process, where ticks are counted, and time is increased after a given number of ticks [4]. It is the logical clock that is used for timekeeping and synchronization because it is impractical to change the oscillation frequency directly.

Each clock has its own *local time C(t),* where *C(t)* is the reading of the local time at real time *t*. A good clock [8] [9] [7] [6] is monotonic and chronoscopic, which ensures that a clock runs continuously and consistently. This enables the function utilizing the clock to observe an event at any point in time and observe all events in the correct order. Some algorithms allow the clocks to run backwards, which conflicts with the requirements for a good clock and can cause unpredicted effects.

Figure 1 shows an imaginary reference clock and two other clocks. The difference in time between a local clock and real time is known as *clock offset* [4]. The maximum difference between all local clocks and real time is known as the *accuracy*. Figure 1 also show that clocks progresses at different rates. This progress is ultimately governed by the frequency of the oscillator, but the *clock rate* of the logical clock can be adjusted as a result of synchronization. The reason is that several ticks are counted before time is increased. The mathematical expression for the rate is $\frac{dC}{dt}$ and the clock rate of an imaginary perfect clock is $\frac{dC}{dt}$=1 at all times [4] [10].

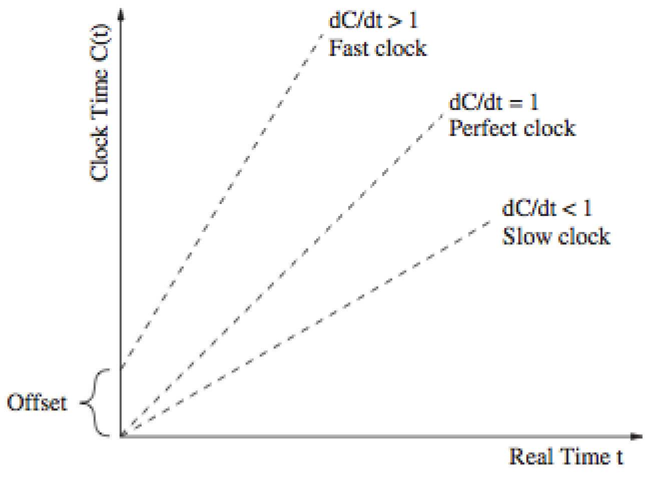


FIGURE 1: RELATIONSHIP BETWEEN LOCAL TIME AND REAL TIME [7]

It is not possible to create clocks were the clock rate is exactly the same as the reference clock because the quartz crystals will have small variations in frequency [7] [11] [6] [10]. This causes clocks to have a slightly different rate and they will gradually diverge from each other as time progresses. The difference in the rate between a clock and a perfect clock is known as *skew* (β) [4] [10] where $\beta = \frac{f_{reference} - f_{clock}}{f_{reference}}$ is 0 for a perfect clock. The difference in rate between two local clocks is known as relative rate. The skew of a clock should be bounded by a value ρ, specified by the manufacturer of the clock, such that the rate of the clock will be within this range when compared to the reference clock [4]:

$$1 - \rho \leq \frac{dC}{dt} \leq 1 + \rho \tag{1}$$

The value ρ is most often expressed as parts per million (ppm). A high precession quartz clock can have a skew where the value of ρ is in the range 1-10 ppm [7]. 1 ppm equals an error of 1µs per second or 60µs per minute. Two clocks, bounded by the same value ρ, can diverge from each other at a maximum rate of 2ρ. The maximum allowed relative time offset δ between any two clocks is known as the *precession.* The bound ρ and precision can be used to find how often the clocks need to be adjusted. This can be expressed as the maximum resynchronization interval, given by τ_{sync} [7]:

$$\tau_{sync} \leq \frac{\delta_{max}}{2\rho} \qquad (2)$$

In addition to the differences in skew, clock rates change because of factors such as aging, temperature and supply voltage [7] [11]. This change is known as *drift.* Drift is the difference of rate per unit of time for a fault free clock *C(t)* and real time. In other words, it is the second derivative of *C(t).*

Two nodes that need to synchronize their local clocks have to exchange information about their local time. Each node does this by generating a timestamp, which is the reading of the local time *C(t)* at real time *t.* This timestamp has to be sent to the other nodes and will experience various non-deterministic delays. These delays form the *time-critical path* of the message. The non-deterministic delays present significant sources of error in time synchronization because the delays make an exact comparison of timestamps difficult. This contributes negatively to the precision of the synchronization algorithm. As a consequence, reduction of the time critical path should be a fundamental aspect of all synchronization algorithms. These non-deterministic delays, also shown in Figure 2, are [10, 3] [12]:

- Creation delay: Time spent to construct the message at the sender and get it ready for the network interface. This is most often introduced due to the operating system, encoding and encryption.
- Access delay: Delay at the MAC layer before the medium can be accessed for actual transmission. This is due to medium contention.
- Send delay: This is the duration of the message transmission. The delay is dependent on message size and transmission rate.
- Propagation delay: This is the actual time the message uses on the transmission medium between the sender and receiver. This is related to the distance between the two nodes.
- Receive delay: This is the time needed for the network interface at the receiver to receive the message. This is equal to the send delay.
- Processing delay: This is the delay introduced by producing an interrupt that is read by the operating system, decoding and decryption.

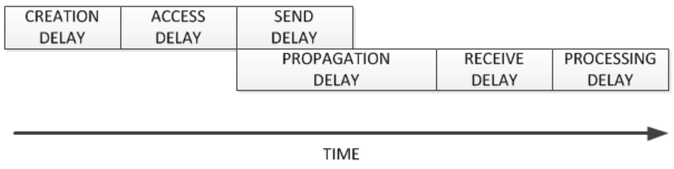


FIGURE 2: TIME-CRITICAL PATH

Table 2 is a short summary of the most relevant definitions and notations used in this thesis. The most common meaning of a definition is used in cases where the use of this definition varies in the available literature

Expression	**Notation**	**Definition**
Synchronization point		A point in time used for synchronization. This can be a timestamp or an event such as the correct reception of a message.
Timestamp		This is the value of the local time the sender transmits to the receivers as a part of a message.
Real time	t	Real time is used as a time reference, most often based on imaginary perfect clock.
(Approximate) global time		Global time is a common notion of time in distributed systems.
Local time	C(t)	This is the local estimate of the global time, maintained by a clock. It is this time that is used to record the occurrence of an event.
Offset	C(t)-t	The difference between the local time of a clock and the reference time t.
Accuracy		This is the maximum offset between any clock and real time.
Rate (clock rate)	C'(t)=dC/dT	This is the rate which the clock progresses at. A perfect real time clock has a rate equal to 1.
Skew	β=1- C'(t)	Differences in clock rate of a local clock and an imaginary perfect reference clock (real time). This causes the clocks to diverge from real time and clocks with other skew values.
Bound	ρ	A guaranteed bound on skew, provided by the manufacturer of the clock.

Relative offset	$\delta = C_A(t) - C_B(t)$	The time offset between two local clocks at a given point in time. This is referred to as drift in some publications.
Precision		This is the maximum relative offset between the local time of any two clocks in a distributed system.
Drift	$C''(t)$	Changes in the rate of a clock due to factors such as temperature and aging.
Time-critical path		The path of the message that contributes to non-deterministic errors.

TABLE 2: IMPORTANT DEFINITIONS AND SYMBOLS

2.2. NARROW BAND WAVE FORM

NBWF, as mentioned in chapter 1, is NATO's attempt at creating a narrow band waveform for the new SDRs. The waveform should work on channels with 25 KHz bandwidth, which NATO already uses for individual Line of sight (LOS) channel allocation. This has become a key design parameter, and it influences all levels of the protocol stack. As a result, the physical layer is capable of delivering raw data rates from 20 Kbps to 82 Kbps. The waveform must be able to operate at the lowest data rate [1] and this has influenced the design choices of the link layer.

2.2.1. NBWF LINK LAYER

FFI has done extensive work on the high level design of the NBWF link layer. The NBWF reference model in appendix A shows this. The requirement to deliver Push-to-talk half duplex voice, with low delay and jitter, has been a key design consideration. The result of this is that a TDMA based MAC protocol is used. TDMA is preferred over contention based protocol such as Carrier Sense Multiple Access (CSMA) protocol because these protocols are incapable of delivering sufficient voice quality at data rates as low as 20kbps [1].

The TDMA protocol used in NBWF is a form of dynamic TDMA protocol with a split channel scheme for voice and data. Voice capacity will be based on reservations, while data capacity can be delivered based on both reservations and contention. Contention will most likely be used for short data transmissions and reservation for long data transmissions.

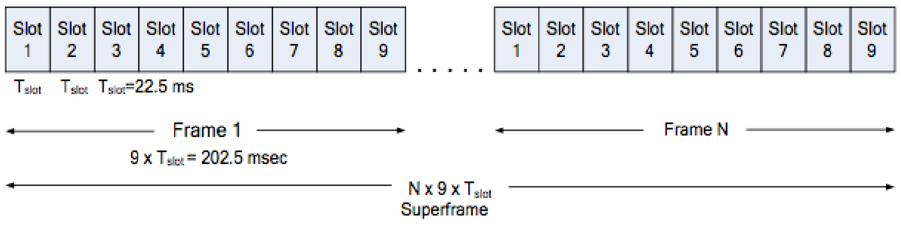


FIGURE 3: TDMA FRAME AND SLOT STRUCTURE [2]

The TDMA slot and frame sizes have been chosen as a compromise between data efficiency and voice delay. This has resulted in the TDMA structure shown in Figure 3. A slot length of 22.5 ms has been chosen because it corresponds with the MELPe[3] 2.4 Kbps frame length (22.5 ms). Each frame consists of 9 slots, and the total frame length is 202.5 ms. The frame length is as a compromise between voice delay and efficient data transfer. Frames will be organized within superframes that can vary in size. The number of frames in a superframe depends on the number of nodes in the network and the capacity they need.

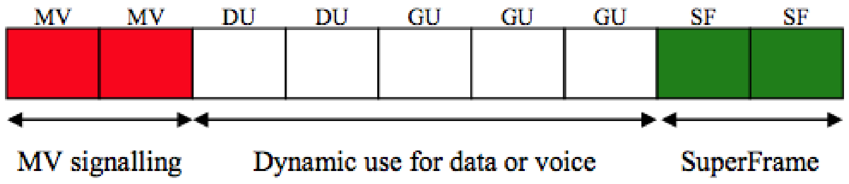


FIGURE 4: TYPE AND USE OF SLOTS [2]

As shown in Figure 4, there are 4 different time slots that can be used within a frame. Each Super Frame (SF) slot is permanently allocated to one node by network configuration. Multicast Voice (MV), Dual Use (DU) and General Use (GU) slots can be used by any nodes. Allocation of these slots is controlled through the use of reservation or contention. It is possible to use all SF slots and previously allocated MV, DU and GU slots for time synchronization. As it is shown in section 2.2.2, sufficient synchronization intervals can be achieved by using only SF slots. Synchronization becomes simpler to implement by using SF slots as all these slots are used in the same

[3] MELPe – Mixed-Excitation Linear Predictive enhanced voice codec. The prioritized voice coder in NBWF [1].

way. MV, DU and GU slots are not considered for synchronization as this would increase complexity, and this thesis only consider SF slots for synchronization.

It is not mandatory for a node to utilize all allocated SF slots. The node will only use a SF slot if there is information, such as SA messages, from higher layers to be transmitted. Other SF slots will be unused, and these cannot be used for synchronization. The minimum number of utilized SF slots will be governed by the number of SF beacons needed by other NBWF functions. Currently it is believed that such SF beacons will be transmitted inside a SF slot at least once every superframe. It is possible to increase the amount of beacons if it is needed by the synchronization algorithm. Estimations and simulations on the chosen algorithm will indicate if this is necessary.

The link layer also includes Air Interface Encryption (AIE). This is shown in the NBWF reference model, which can be found in appendix A. The main reason to include AIE is to encrypt the traffic in order to protect NBWF against traffic analysis and similar attacks. The decision to use AIE to secure restricted information is left to the individual radio design[4] [1]. AIE will encrypt all traffic over the radio interface. As NBWF uses a time based initialization vector, a node that performs late entry needs to know the global time of the network to compute the correct initialization vector. This will be simplified if the synchronization algorithm is capable of supporting external synchronization based on GNSS. If the global time of the NBWF network is accurate and based on GNSS it should be sufficient to set the local time of nodes doing late entry to GNSS time in order to compute the initialization vector.

2.2.2. Prerequisites for Synchronization in NBWF

The use of information in certain TDMA slots to share information about local clocks is essential to maintain synchronization without using dedicated synchronization messages. This information must be fed to a synchronization algorithm that can estimate the correct global time and adjust the local clock. As mentioned in section 1.2, network affiliation has to be performed before this is possible. It is assumed that this mechanism exist in the system and that it functions correctly. At the moment of writing, the NBWF project has no detailed description of the network affiliation mechanism but principal functionality is known.

[4] It is a trend in NATO to move the encryption of information closer and closer to the actual user. NBWF will support this, while still maintaining a minimum security through AIE.

Network affiliation can be performed passively or actively. Either way, the node requesting network entry, will receive coarse synchronization from one of the active network nodes. The effect of coarse synchronization is unknown at the moment of writing, but it is assumed that the relative offset will be well below 1 ms after coarse synchronization is completed. These two nodes will begin to diverge again because of differences in the clocks. The synchronization algorithm described in this thesis has to take over and maintain synchronization. This algorithm is used by all nodes in the network and it must maintain a network wide precision that is within the NBWF requirement.

Maintaining synchronization after network entry should be relatively straight forward to handle because the number of nodes and offset involved are small. However, the algorithm can potentially face more demanding circumstances with relative offset much higher than 1 ms. An example of this is when one or more nodes has lost communication for a period of time. The AIE encryption uses a time based initialization vector and will work as long as the offsets are <11.25 ms (1/2 a slot length). This will be the highest relative offset the algorithm has to handle. It is preferable that the algorithm is able to synchronize the network in these cases. The alternative is to use a network merge mechanism which requires dedicated synchronization messages. A relative offset between two nodes >11.25ms would make it impossible for the receiver to decrypt the signal and other mechanisms have to be used in order to synchronize these nodes. This ensure that information within a SF slot provides two points in time with a maximum offset of 11.5 ms.

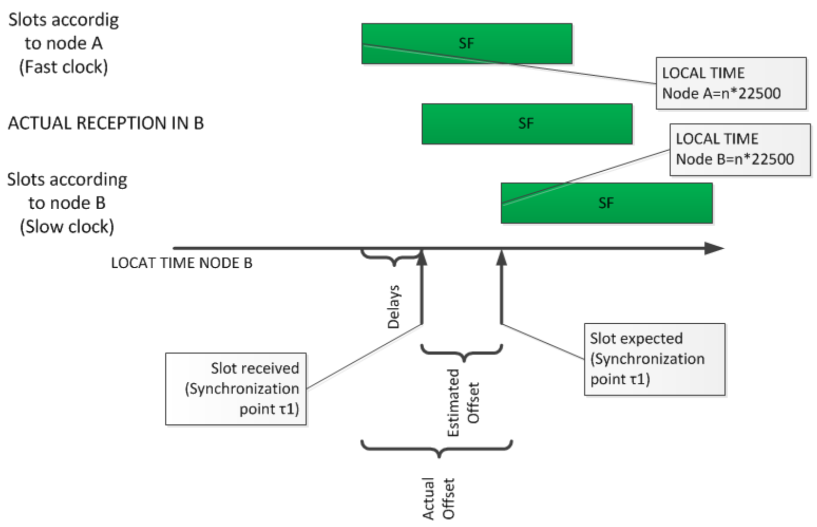

Slots accordig to node A (Fast clock)
SF
LOCAL TIME Node A=n*22500
ACTUAL RECEPTION IN B
SF
LOCAL TIME Node B=n*22500
Slots according to node B (Slow clock)
SF
LOCAT TIME NODE B
Delays
Slot received (Synchronization point τ1)
Estimated Offset
Slot expected (Synchronization point τ1)
Actual Offset

FIGURE 5: NBWF SYNCHRONIZATION POINTS

These synchronization points are shown in Figure 5. One synchronization point occurs when the message is received. A second synchronization point occurs when the sender start to transmit information in this slot. TDMA ensures that the receivers know the local time of the sender for this synchronization point because each slot will start at a known time. This procedure is comparable to one-way message exchange [7], without actually having to transmit any synchronization information between the nodes.

The different non-deterministic delays that effect the transmission of the message is described in section 2.1. The effect of the creation delay can be ignored because the sending node knows when the next SF slot occurs. This makes it possible to have the message ready before the slot starts. The use of TDMA also makes it possible to determine the access delay, because it is only dependent on the synchronization preamble and other known fields in the physical layer of NBWF [1]. This ensures that the access delay is a constant. The same is also true for send delay and receive delay because the length of a slot and the transmission rate are known. Both these delays become constants.

All three constants, shown in green in Figure 6, are known and can be compensated by implementation in algorithms and hardware. These constants can be ignored when the time-critical path of NBWF is evaluated. The processing delay can be ignored as the MAC layer will receive a signal directly from the physical layer when the last bit in the preamble is received, and the distance from the start of the slot to the end of the preamble is known. This leaves the propagation delay as the dominating non-deterministic delay in the time-critical path as shown in Figure 6.

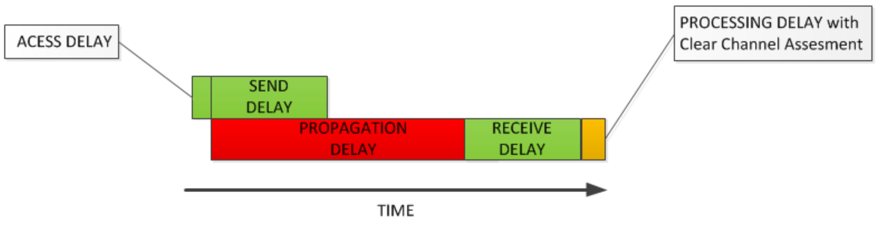


FIGURE 6: NBWF TIME-CRITICAL PATH

The maximum range in NBWF is estimated to be about 60 Km and the maximum non-deterministic error in NBWF will be <0.2 ms per hop. Figure 7 shows two nodes A and B where A begins to transmit the slot when the local time reaches $A\tau_1$. Node B expects to receive the TDMA slot when the local time in node B reaches $B\tau_1$ ($A\tau_1 = B\tau_1$). However, node B receives the slot at local time $B\tau_2$. Node B estimated the relative offset of node A to be the difference between these two synchronization points ($B\tau_2 - B\tau_1$). The actual relative offset consist of the estimated offset (δ) between the local clocks and delays (D) caused by the time critical-path between the two nodes. The delay is unknown to Node B, and in this case Node B estimates a relative offset that is too short. The error for this estimate is dependent on the distance between Node A and Node B. In other words, Node B is able to estimate the local time of Node A with a maximum error of 0.2 ms.

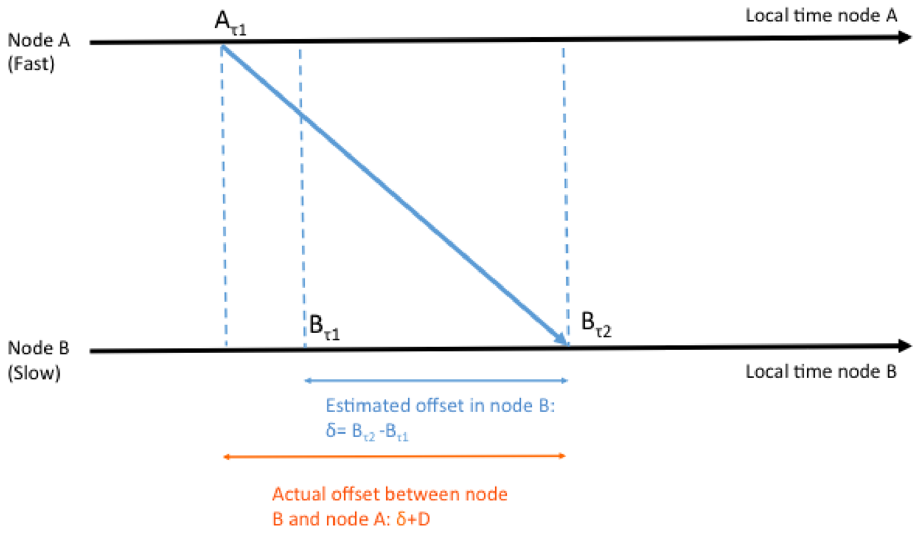


FIGURE 7: NBWF SYNCHRONIZATION

In NBWF, the relative offset must be so small that collisions of TDMA frames are avoided. The maximum allowed relative offset is governed by the guard time used in the physical layer. Guard time is a compromise between the ability to avoid collisions and the efficiency of NBWF. The guard time used in the physical layer must be large enough to avoid collisions while still being as small as possible to ensure an efficient data rate in NBWF. FFI has estimated that the critical value for relative offset needed to avoid collisions is between 1.5 ms and 2 ms [2] [1]. As mentioned in chapter 1, FFI has ensured a safety margin by requiring all nodes to be synchronized to a precision of 1 ms. As a result, collisions in NBWF are avoided as long as the algorithm maintains a maximum relative offset $\delta_{\max} = (C_A(t) - C_B(t)) \leq 1\ ms$, where $C_A(t)$ and $C_B(t)$ are the local time at any pair of nodes. The number of hops tolerated by NBWF will ultimately depend on the actual maximum relative offset of the chosen algorithm and propagation delays in the network.

The NBWF requirements state that the synchronization algorithm should operate without the use of dedicated synchronization messages. This will have an impact on the possibility to estimate the propagation delay of the signal. An unknown propagation delay will add uncertainty to the estimate of the local time. The safety margin introduced by FFI ensures that TDMA collision are avoided as long as the number of hops are low and distances are below the maximum 60 km expected for NBWF. This enables NBWF to use algorithms that do not take propagation delay into account.

All nodes in the network will have the opportunity to transmit in a dedicated SF slot at least once every superframe. Maximum time difference between two synchronization points from a single node is N*202.5 ms if no slots are lost. N is the number of frames (and most often number of nodes) in the network. A network of 50 nodes and N=50 will give maximum time between two SF slots of 10.125 s. The maximum allowed sync interval for NBWF if no rate adjustment is used, can be estimated by the use of equation (2). A maximum sync interval[5] of τ_{sync} =100 s is obtained by using ρ=5 ppm and a precision of 1 ms. As a result, several SF slots can be lost while still maintaining sufficient precision. This should also be true if only NBWF beacons as described in section 2.2.1, are transmitted in the SF slots.

Most recent works describing synchronization algorithms focuses most attention on how to transmit timestamps without collision. This is not a problem for NBWF as the use of TDMA and fixed SF makes it easy for all NBWF nodes to transmit information needed for synchronization. NBWF nodes should be relatively easy to synchronize for simple topologies. A very simple algorithm that updates the local time directly to the same value as synchronization point 1, which is an estimate of the local time at the sender should be enough for these topologies. This algorithm should be enough to achieve a precession where the local time of the two nodes only differs by the propagation delay between the two nodes. However, this approach will not be sufficient in a practical application of NBWF. A real life network would consist of up to 50 nodes in different demanding network topologies. As a consequence of this, NBWF need to utilize algorithms that can compensate for this.

Nodes in the network will be able to receive traffic from all nodes within range. This means that all non-contention based traffic in the MV, DU and GU slots can be used as input for synchronization, regardless of which node is the intended recipient for the data. The start of these slots can be used as synchronization points, and it will also make it possible to analyze which nodes transmits most data. A further study could reveal the effect of including these slots as a part of time synchronization. This could also be used in the estimation of the *quality* of the local time at each node. A node that transmits a high amount of data should have a higher quality than a node that only transmits data in SF slots.

The use of time quality can be extended further by using information from the link layer or by piggybacking 1-2 bytes of data in regular data transmissions. A node with a lot of one or two hop neighbors should have a

[5] When an algorithm that does not correct the clock rate is used.

higher time quality than a node with few neighbors. Information about the quality of the local hardware clock or the quality of an external reference can also be exchanged if piggybacking is used. This gives all the nodes a good overview of the network, without the use of time synchronization messages.

NBWF is intended to be used with LOS communication by military forces. This leads to several application related issues that need to be covered with NBWF synchronization. First of all it must support mobility because any number of the nodes can be on the move at a given point in time. In addition, it must support multiple hops of transmission. Voice traffic is limited to a maximum of two relays, but there is no limit to the number of hops for data traffic. Most networks will cover distances below 60 km, which is less than the theoretical range of the NBWF waveform. The result of this is that the need for relays is created by other properties of the network than distance between the nodes. These properties can be difficult terrain, electronic warfare (EW) by opposing forces or the utilization of IED jammers. The result of this is that NBWF networks might be separated into different clusters with a number of relays between them. However, the precision obtained by the algorithm does not have to support a large number of hops. It is possible to tolerate higher relative offset between nodes that are separated by several relays over longer distances because outside radio range of each other and little risk of collisions

NBWF is designed to be used in a wide range of new SDR together with several other waveforms. As a result, it is difficult to know the exact quality of these radios. However, it is reasonable to expect that military SDR will be high cost devices with strict quality requirements. This will also influence the hardware clocks used in these radios which will be high precession quartz clocks. AIE and other mechanism in NBWF will depend on the quality of the oscillators. This value has yet to be determined by the NBWF work group. Therefore, this thesis uses a conservative value of maximum bound on skew ρ=5ppm, based on the fact that NBWF are intended for high quality SDR. All SDR should have voltage and temperature compensating oscillators, making clock drift a relatively small issue. In addition, the synchronization intervals are relatively small and frequency change experienced by the hardware clock becomes negligible. As a result, the drift (changes in the rate of the clock) can be ignored for the synchronization intervals that NBWF will use. The local clock can be approximated by using a fixed clock rate, and C(t) can be simplified to this linear equation [13]:

$$C_i(t) = C_i(0) + \beta_i * t \tag{3}$$

Here, $C_i(0)$ is the initial offset at node *i* with respect to real time and β_i is a fixed individual skew for node *i*. This skew will have a fixed value in the interval from 1- ρ to 1+ ρ. A summary of these prerequisites is shown in Table 3.

Parameter	Value	Remark
$\boldsymbol{\delta}_{\mathbf{max}}$	$\leq 1\ ms$	Precision needed by NBWF.
Message exchange		The algorithm must not require dedicated synchronization messages and should use information in TDMA slots to generate synchronization points.
Non-deterministic error	<0.2 ms	Propagation delay at 60 Km.
Maximum time between to time samples from the same node	N*202,5 ms	This value might change if slots are lost due to different communication errors.
N	<50	This is the number of frames within a superframe.
$\boldsymbol{\tau}_{sync}$	100 s	The maximum allowed sync interval derived from δ_{max} and ρ.
Time quality		The quality of the local time in a node should be estimated and used as a part of the synchronization algorithm. This should include information such as: • Number of neighbors • Data traffic • Quality of external reference or physical clock
External reference		NBWF should support the use of external time references such as GNSS.
ρ	5 ppm	A conservative estimate for the bound on skew.
Clock rate	Constant: 1- ρ to 1+ ρ	A constant value is used because changes in clock rate due to drift is ignored.
Drift	0	Drift is ignored
NBWF networks		NBWF networks are flat without a hierarchy where all nodes can

		communicate with all other nodes within range. Typical size of the networks is expected to be below 60 km.
Mobility		The algorithm must support nodes that are moving.
Multiple hops	2 relays for voice	Must support up 2 relays for voice traffic and any number of hops for data traffic.
Maximum relative offset $\Delta\tau_{max}$	11.25 ms	This is the highest relative offset two or nodes can have before AIE decryption fails and mechanisms outside the scope of this thesis must be used.

TABLE 3: NBWF PREREQUISITES

2.3. Synchronization design issues and NBWF

Table 3 and the requirements listed in chapter 1 must be considered when algorithms are evaluated. In addition to these prerequisites, there are issues related to algorithm design that should be considered. Works such has the survey published by Sundararaman et al. [4] uses a set of synchronization issues to describe the different synchronization protocols. These issues are a result of different design choices, and they are useful when different algorithms are compared.

The exchange of timestamps is one of the most important design choices in an effective synchronization algorithm. It has a direct effect on the ability to compensate for non-deterministic delays [4]. Most solutions use a form of *sender-to-receiver* approach, where timestamps are transmitted from a sender to a receiver. This can be further divided into *one-way message exchange* and *two-way message exchange* [14]. In one-way message exchange, a message with a timestamp is exchanged between a sender and a receiver. This method can easily be adapted by NBWF as shown in Figure 7. Two-way message exchange provide more timestamps than one-way message exchange because several synchronization messages are exchanged. Figure 8 show a two-way message exchange between two nodes. In the case of the Network Time Protocol (NTP), Node A send the first message with its own timestamp included (τ_1). This is received by node B which creates a new message containing three timestamps (τ_1, τ_2, τ_3). Node A receives the message from node B and can use all 4 timestamps to estimate the propagation delays [4]. However, this approach is not compatible with NBWF as it requires some form of dedicated synchronization message exchange.

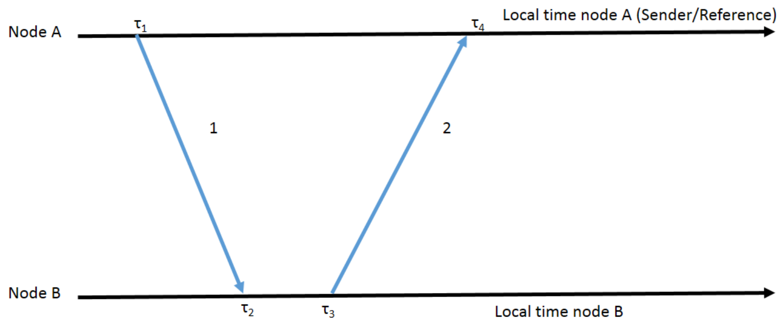


FIGURE 8: TWO-WAY MESSAGE EXCHANGE

An entirely different approach is to use *receiver-to-receiver* [4] synchronization. It is an attempt of using the properties of a broadcast medium to reduce the non-deterministic delays. The method uses the fact that all nodes in a small area network will receive a message at approximately the same time. All receiving nodes will exchange a timestamp based on the local time when they received the transmission. These timestamps are subsequently used to compute the offset. NBWF cannot use the receiver-to-receiver synchronization as originally described in Reference Broadcast Synchronization (RBS) [15], because this requires the use of dedicated synchronization messages. In addition, NBWF networks can cover large areas, and the received time stamp will be affected by the propagation delay. This will decrease the precision of receiver-to-receiver synchronization if it is used in NBWF. However, there are several recent improvements to the receiver-to-receiver approach that might be attractive to NBWF.

The ability to set bounds on the maximum allowed relative offset, due to skew, is another fundamental property of a clock synchronization algorithm. The majority of algorithms try to guarantee this bound with a certainty. These algorithms are known as *deterministic clock synchronization algorithms* [4] [9]. *Probabilistic algorithms* [4] [9] can only give a probabilistic guarantee for the offset, but they can achieve narrower bounds on relative offset. NBWF could work with both approaches, as long as they are able to bound the relative offset of the clocks to within 1 ms of each other. Deterministic algorithm are well known and should be preferred if they support all NBWF requirements.

Algorithms can use two different methods to adjust local clocks to run within the bound [4] [6]. Figure 9 shows the first method, were the local clock is changed instantaneously to the time estimated by the algorithm. This is known as *discrete or instantaneous time synchronization*. While this is straightforward to implement, it can lead to time discontinuities that conflicts with the monotonic and continuity requirements for a good clock. The other method is known as *continuous time synchronization*. This method is more complex than deterministic synchronization because the local clock is gradually adjusted in several steps towards the time value estimated by the algorithm. Both approaches can be supported by NBWF, but the use of discrete synchronization might influence the use of an initialization vector for AIE. This can be compensated by design, but continuous synchronization should be preferred.

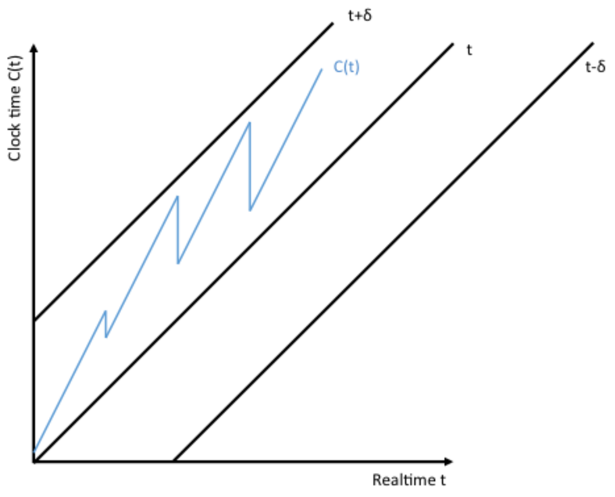


FIGURE 9: DETERMINISTIC SYNCHRONIZATION WITHIN A BOUND

Synchronization algorithms utilizing discrete or continuous time synchronization adjust the local clocks. The goal is to have all clocks running close to the approximate global time. This is known as the *clock correction* approach, and it is the most used method [4]. Another approach is to let the local clock run *untethered* while building a table of information about the clocks of the other nodes in the network. This table is used to transform the incoming timestamps to the local time of the current node. It is theoretically possible to use untethered algorithms for NBWF, but this will create a problem if the local time is going to be used for AIE initialization vector, frequency hopping and network affiliation. Clock correction algorithms should be used in NBWF.

Clock correction algorithms must at least correct the relative offset between two nodes. This ensures that the relative offset between these two nodes is as low as possible at the point of synchronization. After synchronization, the clocks will start to diverge from each other again because the rates are unequal. Sufficient precision and accuracy is ensured by selecting short synchronization intervals. Equation (2) indicates minimum time interval before asynchronization happens. The synchronization interval should be much shorter than this. Another approach is to correct both offset and rate. The goal is to have a relative rate that is as close to 1 as possible. Algorithms that synchronize both offset and rate are more complex, but they provide better precision and accuracy. In addition to this, algorithms that synchronize rate need fewer synchronization messages and can survive longer intervals without synchronization message. Algorithms that synchronize rate and offset should be preferred for NBWF, but it is possible to use both approaches.

Time synchronization will help the distributed system to reach a common notion of time that is an approximation of global time. This approximation is based on the local time in the individual nodes and the value will be computed by the synchronization algorithm. In a *master-slave* [4] synchronization structure, the approximation is based on the reading of the local clock of the master only. Master based algorithms are often simple and converge fast, but they leave the system vulnerable to a master node failure. This is a critical drawback for a military system such as NBWF and master based algorithms with a fixed structure should be avoided. Some works use master based synchronization with ad hoc structures. These structures are a result of master selection process that automatically selects a master from all available nodes. The node acts as a master for a given amount of time or until failure. A new master will automatically be selected after a failure or timeout. The network will continue to work with slight disturbances caused by change of a master. This disturbance should be small, and the algorithm must maintain precision well below the requirement of the system. The use of an ad hoc structure creates an algorithm that is robust to changes.

Other algorithms use a *mutual* (Also known as distributed or peer-to-peer) synchronization, where the local time of all nodes might be used to approximate the global time. Mutual synchronization ensures that the loss of a single node does not critically affect the rest of the system. Some algorithms use the result from a single node directly, but most often results from several nodes are gathered before each estimation of a new local time. An algorithm that supports mutual synchronization should be preferred but master-slave algorithms with an ad hoc based structure can also be used for NBWF.

A system that has no external references uses *internal synchronization* [3] [4]. This can be used in systems where the actual value of the approximate global time is of little interest, as long as it provides local clocks that can be used to maintain the order of events. Even if the system is synchronized with a reference time at one point, the approximation of global time will gradually diverge away from that reference. This may not be a problem as long as all the nodes diverge together.

In external synchronization [3] [4]one or more nodes are connected to a reference such as UTC. This ensures that the approximate global time of the distributed system is kept close to the reference. External synchronization is necessary for applications that depend on the real-time provided by references such as UTC. External synchronization is not strictly required for TDMA slot synchronization in NBWF. However, the use of external time will make network management, AIE encryption and frequency hopping easier. As a consequence, the algorithm chosen for NBWF should support the use of an external reference, but must also be capable of running autonomously. Table 4 is a short summary of these issues.

Issue	Applicable to NBWF	Remark
Receiver-to-receiver synchronization **Sender-to-receiver synchronization**	Both	The algorithm must use only two synchronization points. This is the only approach supported by NBWF.
Probabilistic **Deterministic**	Both	NBWF will probably work with both, but deterministic algorithms should be preferred.
Continuous **Discrete**	Both	NBWF will work with both approaches, but continuous synchronization should be preferred.
Clock correction **Untethered clocks**	Clock correction	Algorithm with clock correction should be used.
Offset **Rate and Offset**	Rate and offset	NBWF can support both, but algorithms that synchronize rate and offset should be preferred.

Mutual synchronization **Ad-hoc structure** **Master-slave synchronization**	Mutual synchronization or ad hoc based master synchronization.	Mutual synchronization should be preferred.
External **Internal**	Should support external	The algorithm should support the use of external synchronization.

TABLE 4: IMPORTANT SYNCHRONIZATION DESIGN ISSUES

3. SYNCHRONIZATION ALGORITHMS

Synchronization of clocks in a distributed system is a well-known problem with a lot of published scientific work. The first proposals were concerned with nodes in wired networks, such as the Internet. NTP is a well-known example of a protocol intended for Internet. An increased interest in Sensor networks and Mobile Ad hoc Networks (MANET) have shifted the focus to synchronization of nodes in wireless networks. This has resulted in a lot of new proposals for synchronization protocols and algorithms that are aimed at sensor networks or ad hoc networks.

Several survey articles have been written, providing an overview of protocols available at their time of writing. One of the most cited articles is the comprehensive work of Sundararaman et al. [4]. This survey gives a detailed overview of protocols available in 2005. Combining the main points in this survey, with the main points of section 2.2 and 2.3, makes it possible to form a quick overview of protocols that might be potential candidates for NBWF. This overview indicates that none of the protocols surveyed by Sundararaman et al. fits the NBWF requirements.

However, the surveys show that there is a continuous development of new algorithms and improvements on older algorithms. This makes it probable that many of the requirements might be solved by new and enhanced algorithms primarily made for Sensor networks or MANET. This thesis surveys recent scientific works related to sensor networks and ad hoc networks. Most of these works are based on or compared their results to one or more of the protocols surveyed by Sundararaman et al.

These newer works are categorized by the original protocols they relate to, and they are presented in the following sections. However, it is necessary to remember that many of these works are related to more than one original protocol. For simplicity, they are compared only with the works which they have closest relations. At the end of this chapter, the most promising solutions will be compared more closely in order to find potential algorithms for simulation.

3.1. NETWORK TIME PROTOCOL (NTP)

NTP is a well-known protocol for time synchronization that has been in use on the Internet since 1985 and has received several updates. The current version is NTPv4 [16] [17], which replaced NTPv3 [18] in 2010. NTP is a hierarchical protocol using stratum levels to indicate the distance from a reference clock. Reference clocks are traditionally based on extremely accurate atomic clocks. These clocks are at stratum level 0. Nodes directly connected to the reference clocks have stratum level 1. Nodes connected to stratum 1 nodes will have stratum 2 and so on. The stratum values are used to ensure that each node synchronizes with the node that is closest to the reference clock. The use of stratum enables NTP to support external synchronization (based on reference clocks) while still being robust. A stratum 1 clock would be the preferred source for synchronization as long as it is available, but it is possible to use sources with higher stratum if stratum 1 clocks become available. It is also possible to synchronize networks without any of the nodes connected to a reference clock by manually assigning a low stratum level to one or more nodes.

The synchronization algorithm uses estimation of round-trip delay and offset to synchronize the local clocks [8]. This algorithm requires four timestamps that are provided by first transmitting a request and then receiving a response with three timestamps. The need for four timestamps makes it impossible to employ the algorithm directly in NBWF without using dedicated synchronization messages. Based on this, no further research is performed, and NTP is not considered for NBWF.

3.2. AD HOC NETWORKS

Ad hoc networks come in a wide variety of distributed systems that require clock synchronization, but have no dedicated infrastructure. This ranges from IEEE 802.11 networks with relative short range to mobile ad hoc networks utilized by military forces that can cover large areas. All works described in section 3.2 use beacon mechanisms to transfer timestamps. This is useful to NBWF because it is a form of one-way message exchange. Most works intended for ad hoc networks offer improvements to the original IEEE 802.11 standard. However, there are also a few works that take an entirely new approach.

3.2.1. SOLUTIONS CREATED FOR IEEE 802.11 AD HOC NETWORKS

The IEEE 802.11 standard [19] uses Timing Synchronization Function (TSF) to synchronize the different nodes in the network. TSF also supports ad hoc networks, known as an Independent Basic Service Set (IBSS) in IEEE 802.11. The synchronization mechanism utilizes periodically transmitted beacons to exchange timing information. The responsibility of beacon transmission is shared between all nodes in the network. This is achieved by introducing a random delay, where the node with the shortest delay will send the actual beacon in the current beacon frame. All nodes in an IBSS are within radio range of each other and will normally receive the message. A node will synchronize its timer to the timestamp if the received time is later than the current local time, thus the global network time will be driven by the nodes with the fastest clocks.

The use of beacons in TSF makes it similar to the slot based approach proposed for NBWF. In NBWF, each received slot can be used as a beacon. However, there are several issues that make TFS unsuited for NBWF. TSF does not support multihop because it focuses on a single IBSS. The use of timestamps from clocks that only run faster will force the global time to follow these clocks. This might ensure slot synchronization, but it makes it difficult to support external synchronization. These issues were also identified by Sundararaman et al. [4] when they analyzed the protocol *Continuous clock synchronization in wireless real-time applications* suggested by Mock et al. [20] This protocol is an extension of IEEE 802.11 standard, which provides continuous clock synchronization for sensor networks.

TSF also suffers from problems with scalability due to the fact that the fastest clock will have a low probability of successfully sending a beacon [21] [22] [23]. This leads to asynchronism because the algorithm is dependent on the fastest node successfully transmitting a beacon. Adaptive Timing Synchronization Procedure (ATSP) [24] corrects this problem, by giving the fastest host a higher probability of transmitting a beacon. Tired Adaptive Timing Synchronization Procedure (TATSP) [25]improves this further, by eliminating problems that occurs when the node with the fastest clock leaves the network. NBWF does not suffer from a problem with beacon transmission collisions because TDMA is used. In addition, the two algorithms are not designed with multihop networks in mind. ATSP and TATSP offer little improvement over TSF when used in NBWF and are therefore not considered for NBWF.

Automatic Self-time-correcting Procedure (ASP) suggested by Sheu et al. [23] take the scalability problem into account as well as support for multihop wireless ad hoc networks. This algorithm relies on two main features. The first one is that the nodes with the fastest clocks will have a higher probability of transmitting a beacon. The second feature is that slower clocks can be rate adjusted to become faster host. This ensures that the synchronization signal of the faster clocks is spread in a multihop network [23].

Pande et al. suggest Modified Automatic Selftime-correcting Procedure (MASP) [22]. As the name suggest, it is a modification of ASP. MASP improves on ASP by modifying beacon contention and offset calculation.

Simulations of multihop networks show that both ASP and MASP have considerably lower average maximum relative offset than TSF. The same simulations also shows that MASP only performs slightly better than ASP [22]. In reality, these differences would make little difference in NBWF and ASP should be preferred. ASP is an algorithm that might solve the minimum requirements for NBWF, and it should be considered for NBWF. ASP requires one sequence number to be transmitted between the nodes, this could be piggybacked in a SF slot.

Wen et al. suggest a MAC level synchronization algorithm for IEEE 802.11 TDMA Ad Hoc networks [21]. According to the author this algorithm yields a better result than the Timing Synchronization Function (TSF) specified in the IEEE 802.11. The improvement has been achieved by changing selection of beacon nodes, based on the fact that typical propagation delay in IEEE 802.11 networks are low and can be neglected [21]. This change is inspired by the receiver-to-receiver approach used in RBS. In addition, the protocol support rate adjustment. While this algorithm prioritize fast clocks and adjust rates, the author does not mention mobility or multihop support. Therefore, ASP and MASP are preferred over this algorithm.

The ad hoc solutions described so far have only used information from faster clocks for synchronization. Clock Sampling Mutual Network Synchronization (CS-MNS) [26] tries to improve synchronization in ad hoc networks by utilizing information from all clocks. The main motivation for Rentel et al. was to create a mutual clock synchronization protocol that was simple and could work on different MAC layers. The designers sacrificed some precision in order to make this possible, but simulations indicate that the protocol performs better than ASP [26]. Clock offset, and rate adjustment is based on control theory, this makes it possible to adjust both offset and rate with a single timestamp. In addition, both slow and fast clocks can be used for synchronization. Rentel et al. does not discuss external synchronization, but except from that, the algorithm described is a potential candidate for NBWF.

A few later works suggest the use of connected dominating sets (CDS) in order to avoid asynchronization in large ad hoc networks. Asynchronization occurs when beacons from the fastest nodes are lost due to collisions. The use of dominating sets are originally a part of routing theory. These later works adapt this to ad hoc networks in order to reduce the chance of collisions. This is done by introducing a virtual infrastructure that segment the network into smaller parts, where only selected nodes are allowed to broadcast. Ad hoc networks that do not use CDS or similar techniques have a flat hierarchy where every node is equal.

Algorithms such as Multihop adaptive timing synchronization function (MATSF) [27] and Hierarchical PTSF [28] are two examples of algorithms utilizing CDS. As mentioned above, these algorithms are mainly created to avoid collision of beacons. This is not a problem in NBWF, where TDMA is used. The use of CDS in NBWF will add complexity without an improvement in relative offset or convergence time. As a result, CDS extensions to ad hoc algorithms will not be considered for NBWF.

There are other works related to IEEE 801.11 ad hoc networks that try to solve asynchronism or other problems related to IEEE 801.11. As with MATSF and Hierarchical PTSF, they offer little or no improvement to NBWF and often increase the complexity of the algorithm or protocol [29] [30].

3.2.2. HYBRID SOLUTIONS

Two recent algorithms use hybrid synchronization, which is a combination of master-slave and mutual synchronization. This approach combines the master-slave ability to use GNSS and the robustness of a mutual algorithm.

Black Burst Synchronization (BBS) [12] is a MAC level protocol intended for synchronization of ad hoc networks. The protocol uses tick synchronization and transmits synchronization information in tick frames. All nodes send their information within the same tick frame, and Black Burst encoding is used to avoid collisions. The encoding itself is of no interest to NBWF when TDMA is established because all nodes will have their predefined SF slots. However, BBS has several other promising features. First of all it supports both decentralized (mutual) and centralized (master-slave) synchronization. This should make it possible to use both internal and external synchronization. Secondly it can provide a deterministic bound for multihop precision, and it supports mobility. This makes it a possible candidate for NBWF.

Saarnisaari and Vanninen [31] [32] describes a hybrid algorithm for MANETs. They focused primarily on military networks when the algorithm was created, and it should fit well with NBWF. The algorithm uses a master-slave approach as long as GNSS is available in the network. A mutual synchronization algorithm named Discrete Network Synchronization (DNS) is automatically used when there are no GNSS capable nodes in the network. The hybrid algorithm enables the use of external references, without leaving the system vulnerable to master node failure. The algorithm only needs two flags and a timestamp to work and does not put strict demands on the underlying system. The algorithm is an attractive candidate for NBWF.

3.3. Sensor networks

Sensor networks are ad hoc networks that consist of specialized nodes. These nodes monitor a real world phenomena such as temperature and they are custom made for this task only [4]. However, sensor networks share a set of common features that makes it easier to compare them separately from more generic ad hoc networks. It is necessary to have these features in mind when sensor networks are compared [4]:

- Energy conservation is critical in most sensor networks. This can force the designers to make design choices that they otherwise would not make. Most synchronization algorithms will be a tradeoff between performance and energy efficiency.
- The nodes in sensor network are most often inexpensive devices. This will have an effect on the quality of the hardware components chosen which affects both the CPUs ability to do complex evaluations and the quality of the hardware clocks employed.
- The networks are often dense and cover small areas, and the resulting propagation delays are short. As a result, propagation delay is ignored by most protocols and algorithms designed for sensor networks. Some algorithms even use the ability to ignore propagation delay as an important part of algorithm design.
- A part of the solutions consider stationary nodes only.
- A part of the solutions are made for networks with much higher bandwidth than NBWF. This bandwidth enables more information to be exchanged between the nodes than what is possible in NBWF.

3.3.1. One way message exchange

The Flooding Time Synchronization protocol (FTSP) [33] is created with sensor networks in mind. The authors claim that the protocol performs better than other protocols because FTSP reduces the non-deterministic errors in the radio message delivery. The effect of reduced non-deterministic errors is that propagation delay remain the only significant part. This improves the precision of the FTSP protocol.

This effect is already achieved in NBWF by reducing the time critical path as described in section 2.2.2. Because of this, FTSP will have no specific advantages over other protocols without the ability to reduce the non-deterministic errors. The FTSP algorithm uses a single broadcasted radio message to synchronize the local clock. Differences in clock rates are compensated by using linear regression. The origin of the broadcasts are a single dynamically (re)elected node and the message is flooded through the network by other nodes. This creates an ad hoc structure that is more robust than a fixed spanning-tree [33]. According to the authors this makes the FTSP algorithm more robust against node mobility and topology changes. In addition to this, the authors claims that FTSP algorithm support multiple hops and a large number of nodes.

Kalman Filter using Multiple Parents (KFMP) is an algorithm for multihop TDMA sensor networks proposed by Zeng et al. [34] The algorithm uses observations from multiple parents to provide more precise results. A vector Kalman filter is used to combine these observations. This will, according to the authors, significantly reduce the clock error at the edge of the network. The algorithm is probabilistic and executed in several steps. A node has to use at least one superframe to find the number of one hop neighbors before the algorithm can be used to full efficiency. The algorithm is more complex than FTSP, but the authors claim that this leads to reduced relative offset. However, it is uncertain what the effect of these reductions will be if KFMP is employed in a NBWF network with high propagation delays.

Tjoa et al. describe a mutual synchronization algorithm that uses slot referencing to synchronize TDMA sensor networks [35]. The algorithm uses reception of a message in a TDMA slot to create a synchronization point. This is comparable to how NBWF will work, and no dedicated synchronization messages are needed. The algorithm gathers synchronization points from all nodes and stores the estimated relative offsets in a table. Synchronization is performed only after the node estimates that the largest relative offset exceeds a constant threshold. The new local time is based on the average of all offsets available in the table. The algorithm also use a timer to ensure that changes spread through the network.

A similar approach, created for TDMA ad hoc networks, is described in a more recent article written by Wang et al [36]. The authors describe the use of absolute slot numbering to create synchronization point. These synchronization points are used in combination with linear regression to synchronize the local clocks. The algorithm uses an ad hoc structure with an automatically selected root (master). Root selection seems to be very simple, the node with the lowest ID becomes the root. In addition to this, the authors assumes that all nodes know the route with shortest hops towards the root without describing how this is achieved. Both the article and the algorithm seem unfinished and the algorithm is not considered a suitable candidate for NBWF.

3.3.2. Receiver-to-receiver synchronization

Reference Broadcast Synchronization (RBS) [15] has received a lot of attention because the use of receiver-to-receiver based synchronization reduces the non-deterministic delays. This is achieved by broadcasting a beacon from a reference sender that is timestamped upon reception. These timestamps are exchanged between receivers of the original beacon. A new local time is time is estimated based on these timestamps. This leads to better precision, but requires additional message exchange between all nodes that receive a message from the reference sender.

The use of dedicated message exchange between the receivers and a reference broadcast is among the main reason that RBS cannot be used by NBWF. This synchronization mechanism needs dedicated synchronization messages, which is a conflict with the NBWF requirements. This is also confirmed by the evaluation of RBS by Sundararaman et al. However, RBS is used as a basis for a lot of new works that are not considered by Sundararaman et al. These works solve some of the original RBS problems and should be considered for NBWF.

In [37] Djenouri suggest a new protocol that distributes the reference function between all nodes in the network. This removes the single point of failure that the use of a reference sender in RBS introduces. In addition, the protocol allows information to be piggybacked, this eliminates the need for dedicated synchronization messages. The protocol is initially promising but have several shortcomings that make the proposed algorithm unsuited for NBWF. Firstly it does not consider mobility and assumes a static network. Secondly it assumes that all nodes are in close vicinity of each other. Thirdly it has limited support for multihop because two multihop nodes only synchronize at the moment when they need to exchange information. This is not a viable solution for TDMA networks. Finally, the protocol only supports a low number of nodes in the network because it requires a lot of piggybacked information. This makes it difficult to utilize the protocol in NBWF, where only 1-2 bytes of information can be expected to be piggybacked.

There are other algorithms [38] [39] [40] that suggest changes to RBS or are based on receiver-to-receiver synchronization. None of them does anything to change the extensive need of synchronization information to be exchanged. As a conclusion, none of the receiver-to-receiver based algorithms are suitable for NBWF.

3.3.3. TWO-WAY MESSAGE EXCHANGE

There are several algorithms [41] [42] that utilize two-way message exchange. A well-known example is Timing-sync Protocol for Sensor Networks (TPSN) [43], which supports multi hops and is one of the first algorithms to utilize MAC layer time stamping. While these algorithms might offer useful capabilities such as delay compensation, they require at least two-way message exchange. NBWF can only provide two separate synchronization points which makes it difficult to use algorithms based on two-way message exchange without adding dedicated synchronization messages. These algorithms are not considered for NBWF.

3.3.4. HIGH PROPAGATION DELAY SENSOR NETWORKS

There have been some focus on delay tolerant networks in some later works [14] [44] [45] [46]. These works describe extremely specialized sensor networks with high propagation delay. Initially these works might seem attractive to NBWF because they support networks with higher propagation delay than most other sensor networks. However, a closer inspection shows that none of these works fits NBWF. All these works require at least two-way message exchange. In addition, these sensor networks are specialized cases with exceptionally strict requirements on energy saving and long time periods between transmissions. The result is makes highly specialized algorithms, and they are unlikely to fit NBWF.

3.4. Summary of algorithms suitable for NBWF

	Node mobility	Rate sync	External/ Internal	Time quality	Mutual/ Ad hoc
ASP [23]	Described	YES	Internal	Fastest clock selected	Fastest node
CS-MNS [26]	Described	YES	Internal	Uses all nodes	Mutual
Hybrid(DNS) [31] [32]	Described	NO	Both	external mechanism	Hybrid
BBS [12]	Described	YES	Both	Uses all nodes	Hybrid
FTSP [33]	Described	YES	Internal	Node ID used for selection	Ad hoc tree
KFMP [34]	Not described	YES	Internal	Uses all nodes	Mutual
Tjoa et al. [35]	Not described	NO	Internal	Uses all nodes	Mutual

TABLE 5: POTENTIAL ALGORITHMS

Table 5 combines the findings in chapter 2 and 3 to present all the potential algorithms for NBWF. There are some synchronization design issues and NBWF prerequisites that are common to all algorithms. These are omitted in Table 5 but listed here:

- Deterministic: All algorithms are deterministic.
- Discrete: All algorithms use discrete adjustment of the local clock.
- Clock correction: All algorithms use clock correction.
- Multiple hops: The actual approach varies for each algorithm, but all algorithms are tested with multiple hops by the original authors. The effect on NBWF is unknown and must be tested by simulations to determine which algorithm is best suited.
- Dedicated synchronization messages: None of the algorithms need dedicated synchronization messages. Some of the algorithms need to exchange information, but this can be achieved by piggybacking 1-2 bytes of data to existing traffic in the SF slots.

All algorithms are based on mutual synchronization with the exception of FTSP and ASP. These algorithms use a master-slave approach with an ad hoc structure. Ad hoc structures might be effective in NBWF if they are supported by a reliable mechanism that select the master node based on information such as the number of neighbors and data traffic in the different nodes. Both algorithms lack this ability to use some sort of time quality to ensure that the best node is selected as master. The ASP algorithm is remarkably simple and synchronizes to information from faster nodes only. FTSP uses a root selection mechanism to select the master node. Both these approaches might cause problems if the master is placed at the edge of a network with several hops towards the main part of the network. This can cause unnecessary instability in the network if the master node is lost. Because of this, the ASP and FTSP are not as robust as the mutual algorithms.

The remaining algorithms uses mutual synchronization and can synchronizes the local clock with timestamps received from any node in the network. The ability to use timestamps from any node in the network make these algorithms less vulnerable to the loss of a single node. There is no need for root selection and the fact that the algorithms lack support for time quality is of less concern. These algorithms will only benefit from such functionality in network merge cases.

KFMP and the algorithm described by Tjoa et al. [35] are both mutual algorithms that estimates average values from several nodes before they adjust the local clock. KFMP can adjust both the offset and rate of the local clock. The ability to adjust rate could be utilized in NBWF to make the algorithm more robust towards communication problems. In addition to this KFMP uses averaging to reduce the maximum relative offset at the edge of networks with a lot of hops between nodes. This is a key feature in KFMP as it is created for sensor networks with short distances and a large number of hops. This positive effect for edge nodes is uncertain in NBWF as NBWF networks covers large distances but with relatively few hops (compared to sensor networks). As a consequence, the positive effect of utilizing the KFMP algorithm within NBWF might be small, when compared to the complexity needed to support the approach. In addition to this, the authors has not described how the algorithm will perform in networks with node mobility. The algorithm is complex to implement in the simulator, and the results are uncertain. Based on this, other algorithms will be preferred.

The algorithm described by Tjoa et al. is much simpler than the KFMP algorithm to implement. In addition to this, the algorithm is created with TDMA networks in mind. This should make it easy to implement in NBWF. Unfortunately, the algorithm cannot adjust the rate. In addition to this, the authors have only tested the algorithm on static networks. The effect of mobility is uncertain, and other algorithms should be preferred.

The CS-MNS algorithm is a mutual algorithm that differs from the two previous algorithm by the fact that it does not use averaging. The CS-MNS algorithm adjusts the local clock directly after reception of a timestamp from another node. This should help the CS-MNS algorithm to converge faster than the two previous algorithms that have to wait for timestamps from several nodes. The algorithm is simple to implement, and it is expected that it should fulfill the minimum requirements of NBWF. CS-MNS could be a candidate for a minimum solution for NBWF.

So far all the algorithms described only support internal synchronization. This approach ensure sufficient precision to keep all nodes synchronized, but the global time of the network will diverge from real time. The hybrid algorithm and the BBS algorithm use a hybrid approach that enables them to support external time sources while still being robust. The ability to use external sources when available can be used to ensure that the global time follows real time as closely as possible. The ability to follow real time can be utilized in NBWF to make implementation of network management, frequency hopping and AIE initialization vector much easier. These algorithms should be preferred over CS-MNS and the other algorithms, as long as they can deliver sufficient precision, accuracy and convergence.

BBS is a hybrid algorithm that can fulfill most of the NBWF requirements. The BBS algorithm primarily uses a master based synchronization algorithm but has the ability to fall back on mutual synchronization if the master becomes unavailable. External synchronization is available as long as master synchronization is used. The actual selection of a master and the use of external synchronization is not described, and the paper leaves the impression that only one master is considered [12]. The algorithms use of a single master might cause unnecessary instability when GNSS based synchronization is used. In networks with several GNSS capable nodes, BBS will have to fall back to mutual synchronization (or initiate master selection) if the master is lost. In addition to this, this the focus of the authors is mainly on the encoding of the synchronization messages and not on synchronization itself. The encoding of the signal is of no relevance to the algorithm described in this thesis as all nodes can use dedicated SF slots communication.

The hybrid algorithm designed by Saarnisaari et al. [31] [32], is made with military MANETs in mind and it is designed to work under a mechanism providing a distributed decision making capability [47]. This mechanism uses intelligent decisions, based on the number of neighbors and data traffic, to provide management capabilities such as coarse synchronization, late entry and network merge. Primarily the algorithm uses external synchronization, with the ability to fall back on mutual internal synchronization if all nodes connected to an external source fails. This is similar to the BBS algorithm with the exception that the hybrid algorithm designed by Saarnisaari et al. supports several masters without the need of a master selection process. The only real drawback is the lack of rate adjustment. On paper, the work described by Saarnisaari et al. promises support for all the requirements and prerequisites for NBWF. The hybrid algorithm will be studied in detail and simulated.

4. THE HYBRID ALGORITHM PROPOSED FOR NBWF

As described in section 0, a hybrid algorithm is suggested for time synchronization in NBWF. The hybrid algorithm is described in several articles where Saarnisaari and Vanninen are the main contributors. As mentioned, the algorithm is created for military MANETs and more specifically for networks with Frequency Hopping [47]. The algorithm is created to be simple to implement, without a specific link layer in mind. As a result, it should be easy to implement this algorithm in a TDMA network, and this has already been described in a master thesis by Toumivaara [48].

The complete work described by Saarnisaari and Vanninen is divided into 3 layers. The top layer handles network management functionality. The middle layer handles the hybrid functionality and selects master-slave synchronization if a GNSS signal is available otherwise it selects a mutual synchronization. The lowest layer is responsible for estimating offsets and keeping the local clocks within tolerable limits of each other. The different layers will be described below.

4.1. SYNCHRONIZATION ALGORITHM

The main objective of a synchronization algorithm is to minimize the relative offset between all the nodes in the network. Saarnisaari and Vanninen have suggested that the DNS algorithm is used for this purpose [31] [32]. The DNS algorithm supports multiple hops and can be used for distributed synchronization and centralized synchronization [31]. Saarnisaari based the DNS algorithm on the original work of Davies [49]. The main difference between the original algorithm and the algorithm proposed by Saarnisaari is the use of feedback.

Saarnisaari describes the local clock $T_i(k)$ in node *i* at synchronization instance *k* by the following equation [31]:

$$T_i(k) = T_0(k) + t_i(k) \tag{4}$$

$T_0(k)$ in equation (4) is the common reference time and $t_i(k)$ is the deviation from common time in node *i*. Actual adjustment of the local clock is done by adding a correction term $c_i(k)$ [31]:

$$T_i(k) = T_0(k) + t_i(k) + c_i(k) = T_0(k) + d_i(k) \tag{5}$$

The deviation from common time now becomes $d_i(k)=t_i(k)+c_i(k)$. Equation (6) shows that the correction term is recursive [31]:

$$c_i(k+1) = \alpha c_i(k) + h\epsilon_i(k+1) \tag{6}$$

The constant α is used to control the feedback, and the coefficient h is a used on the estimated adjustment value $\epsilon_i\ (k+1)$. Saarnisaari suggests the following values for the constants [31]: $\alpha=0.15$ and $h=0.75$. These values ensure both stability and sufficient convergence for the algorithm. The adjustment value ϵ_i is the average value of N_i computed deviation between node i and N_i other nodes. This can be expressed as [31]:

$$\epsilon_i(k+1) = \frac{1}{N_i}\sum_{j=1}^{Ni} \epsilon_{ij}(k) \qquad (7)$$

The DNS algorithm does not synchronize before the correct number of timestamps is received. The actual value of N_i has to vary with the number of nodes, system design and implementation. There are no mechanisms within the DNS algorithm that deals with missing timestamps or long periods without synchronization messages. DNS must wait indefinitely for arrival of missing timestamps instead of synchronizing with the information already available. This has to be handled by algorithms placed above DNS. Saarnisaari describes the following equation used to compute deviation between node i and another node j [31]:

$$\epsilon_{ij}(k) = d_j(k) - d_i(k) + \tau_{ij}(k) - \ddot{\tau}_{ij}(k) + n_{ij}(k) \qquad (8)$$

Here, $d_j(k)$ and $d_i(k)$ represent the deviation from the common reference time. The actual delay is represented by τ_{ij}. This is unknown, but the effect can be reduced by using delay compensation which is represented by $\ddot{\tau}_{ij}(k)$. The last part of equation (8) $n_{ij}(k)$ represent a delay measurement noise factor introduced by Saarnisaari [31].

4.2. HYBRID NETWORK ALGORITHM

The DNS algorithm can only be used to reduce and maintain low relative offset between nodes, and it depends on a timer provided by an additional algorithm to handle long intervals without synchronization messages. Saarnisaari and Vanninen have suggested a hybrid network algorithm that is capable of utilizing DNS [32]. The main objective of hybrid network synchronization is to enable the use of GNSS based synchronization as long as it is available and automatically switch to mutual clock synchronization if all nodes lose GNSS capability. In addition, the hybrid algorithm can be used with mutual synchronization only [32]. Two flags ensure that the correct choice is made when the algorithm is used in hybrid mode. The master flag (M) is set by nodes that have GNSS capabilities and the heard master (HM) flag is set by nodes that are direct neighbors of a master node or a node with the HM flag already set [32].

The hybrid algorithm synchronizes the local clock in discrete intervals. It will not synchronize the local clock after reception of a single timestamp but will wait for a given number of timestamps is received. This is done to make the algorithm more robust by ensuring that the correct decisions are made and that the best source of synchronization is used. It will also prevent nodes from unnecessary switches between mutual and GNSS based synchronization.

Saarnisaari and Vanninen created the algorithm without specific control channels in mind [32] but it includes mechanisms to control the transmission of synchronization messages. Figure 10 shows the state machine created by Saarnisaari and Vanninen.

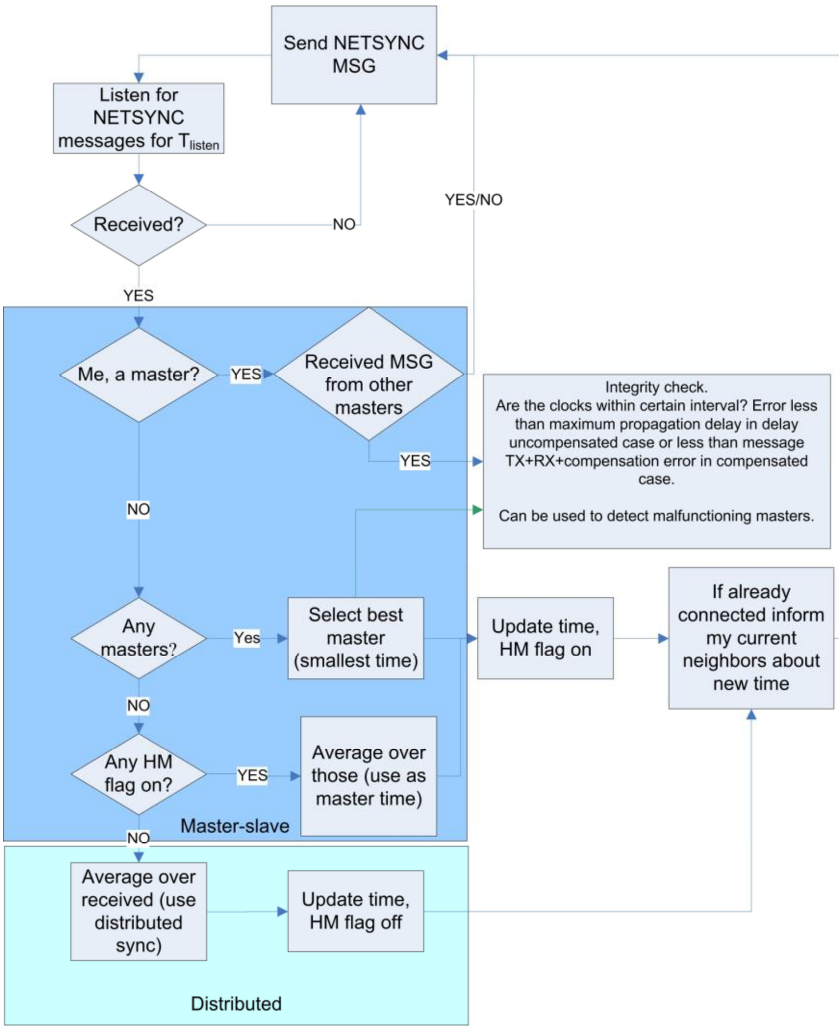

Send NETSYNC MSG
Listen for NETSYNC messages for T_{listen}
Received?
NO
YES/NO
YES
Me, a master?
YES
Received MSG from other masters
YES
Integrity check.
Are the clocks within certain interval? Error less than maximum propagation delay in delay uncompensated case or less than message TX+RX+compensation error in compensated case.
Can be used to detect malfunctioning masters.
NO
Any masters?
Yes
Select best master (smallest time)
Update time, HM flag on
If already connected inform my current neighbors about new time
NO
Any HM flag on?
YES
Average over those (use as master time)
Master-slave
NO
Average over received (use distributed sync)
Update time, HM flag off
Distributed

FIGURE 10: STATE MACHINE FOR THE HYBRID ALGORITHM [32]

The algorithm will wait for synchronization messages for a given amount of time T_{listen} and initiate transmission of a synchronization message if no messages are received during this period. T_{listen} can be varied based on the status of the nodes. This will ensure that GNSS capable nodes transmits more often.

Received synchronization messages will be handled based on the flags in the message and the status of the node receiving the messages. Nodes that have GNSS capability will not adjust itself based on synchronization messages. However, timestamps with the master flag set will be passed on to an integrity check mechanism. This check can be used to decide if the local or received GNSS clock have errors. However, Saarnisaari and Vanninen does not describe this integrity check any further [32] and it must be assumed that this will be a part of future work.

Nodes without GNSS capability, that receives one or more messages with the master flag set, will use one of these to estimate the offset directly. The timestamp with the lowest offset will be selected if several synchronization messages are available within the same period. The selected timestamp from a master will be used in a master-slave fashion and synchronization will be performed after an integrity check. These nodes will also set their HM flag and potentially initiate an extra synchronization message to inform all neighbors of the status change. This extra message is intended to help in network merge cases, but it is not implemented yet [32].

Nodes without GNSS capability that receive only one message with HM flag set will synchronize with this message in the same way as for a message with the master flag set. The average value of the relative offset is used if several timestamps with HM flag is received.

Finally, mutual clock synchronization will be used if a non GNSS capable node only receives messages with no flags set. The DNS algorithm is controlled by the hybrid algorithm, which feeds all timestamp to the DNS algorithm and ensures that the DNS parameter N_i will equal to the number of timestamps available at the moment. Saarnisaari and Vanninen suggested a slight alteration of the DNS coefficient h from 0.75 to 0.5 [32].

4.3. DISTRIBUTED DECISION MAKING

As mentioned, radio networks such as NBWF needs mechanisms to provide initial synchronization, late-entry and network merge. The hybrid algorithm described by Saarnisaari and Vanninen can only provide this functionality for GNSS capable networks [32]. Fortunately Saarnisaari and Vanninen, with the support of others, have created a distributed decision making method that can take care of this [47].

These management mechanisms are outside the scope of this thesis, but will be described briefly to give the complete overview of the approach suggested by Saarnisaari and Vanninen. This also shows that much of the intelligence is implemented in the decision making mechanisms. The actual synchronization performed by the combination of DNS and hybrid algorithm is similar to other approaches. These algorithms only gather information and synchronize in the same way as other mutual algorithms such as KFMP. The novelty is the ability to choose between mutual and GNSS based synchronization.

Node ID and network ID are common parameters found in many decision making systems [47] but the mechanism described by Saarnisaari et al. uses two additional parameters to ensure that decisions regarding network merge and late entries are performed as effective as possible. These parameters are information about the number of two-hop neighbors and data traffic [47]. Vanninen et al. suggested that each node piggybacked the information about their number of two hop neighbors and data activity in synchronization network. This information is used by the decision making system to make the best decisions in each case. Consistent decisions are ensured by letting the first node that receives a synchronization message from another network make the decision for the entire network it belongs to [47]. The distributed decision making mechanisms will not be simulated.

4.4. IMPLEMENTATION IN THE SIMULATOR

The DNS algorithm is adapted for the NBWF simulator by making the following changes. Both Equation (4) and equation (3) represent local time at a given moment. The simulator uses equation (3) to represent the local clocks of the nodes and equation (5) is changed based on this. An adjustment to local clocks $C_i(t)$ can be made by adding the correction term $c_i(k)$ to the synchronization point generated at the moment node *i* expected the TDMA slot. This synchronization point equals τ_1 in Figure 7. Therefore, equation (5) is replaced by equation (9) which provides the new local clock value $C_i(t)^+$ for node *i* after synchronization.

$$C_i(t)^+ = \tau_1 + c_i(k) \qquad (9)$$

Saarnisaari does not indicate a starting value for $c_i(k)$ and a starting value of $c_i(k)=0$ is used in the simulator.

Equations (6) and (7) are used unchanged. Equation (8) can be simplified as long as NBWF does not use delay compensation. As a result $\ddot{\tau}_{ij}(k) = 0$ and $n_{ij}(k)$ can be ignored. As shown in Figure 7 NBWF generates two synchronization point τ_1 and τ_2. The first synchronization point τ_1 is generated when node *i* expects the TDMA slot. The second synchronization point τ_2 is generated when node *i* receives the TDMA slot. Synchronization point τ_2 also includes the propagation delay and equation (8) is replaced by equation (10) which is a calculation of relative offset:

$$\epsilon_{ij}(k) = abs(\tau_2 - \tau_1) \qquad (10)$$

As previously mentioned, the number of synchronization points that are received before estimation is not specified by Saarnisaari. The maximum number of synchronization points that can be gathered will depend on the maximum allowed offset and the quality of the oscillators. As shown in section 2.2.2, NBWF can support a τ_{sync} value of 100s if the clocks have a maximum 5ppm skew. This is a high value, covering several superframes in a network with 50 nodes and should not be used as a basis for N_i. As a consequence, N_i should be decided based on simulations and the requirements of the hybrid algorithm. Pseudo code of the DNS algorithm is shown in Figure 11.

```
PROCEDURE DNS($\tau_1$, $\tau_2$, $N_i$)
    N := N+1;    //Count number of received timestamps from last sync
    array(N) := abs($\tau_2$- $\tau_1$);    //Equation 10 saved into an array.
    if N is equal to $N_i$ then begin    //Sync only after N sync points
        //Equation 7
        for w:=1 step 1 until N do begin;
            $\varepsilon_{ij}$ := $\varepsilon_{ij}$+array(w);
        end;
        $\varepsilon_i$ := $\varepsilon_{ij}$/N;
        $c_i$ := $\alpha$*$c_i$+h*$\varepsilon_i$;    //Equation 6
        estimatedTime := $\tau_1$+ $c_i$;  //Equation 9
        //Reset Counter, values and array after synchronization;
        N := 0
        for w:=1 step 1 until $N_i$ do begin
            eij_Array(w) := 0.0;
        end;
        $\varepsilon_{ij}$ := 0;
        $\varepsilon_i$ := 0;
        end
    else begin
        synched := false;
    end;
```

FIGURE 11: PSEUDO CODE FOR THE DNS ALGORITHM

The original plan was to simulate the DNS algorithm and the hybrid algorithm together. Functionality of the hybrid algorithm when GNSS is used is well described in the article by Saarnisaari and Vanninen [32] and it is assumed that most of their findings related to GNSS based synchronization applies to NBWF also. However, the functionality of the DNS algorithm is necessary to ensure the robustness of the hybrid algorithm. The motivation for the authors to select the DNS algorithm for mutual synchronization in the hybrid algorithm is unclear [32]. They have not described why they chose this algorithm over other potential mutual synchronization algorithms. The effect that parameters, such as the N_i value, have on precision and convergence is also unclear. Finally, Saarnisaari only simulated small networks in his original article about the DNS algorithm [31]. Because of this, simulations are performed on the DNS algorithm to verify if this is the best algorithm to use in combination with the hybrid algorithm.

5. SIMULATIONS

5.1. SIMULATOR OVERVIEW

The simulator is a DEMOS/Simula based program that is written with NBWF in mind. Several properties of NBWF have been used to simplify the simulator. The result is a simulator that can be used to simulate algorithms under the conditions given by NBWF. A model of the implementation of the local clocks is shown in Figure 12.

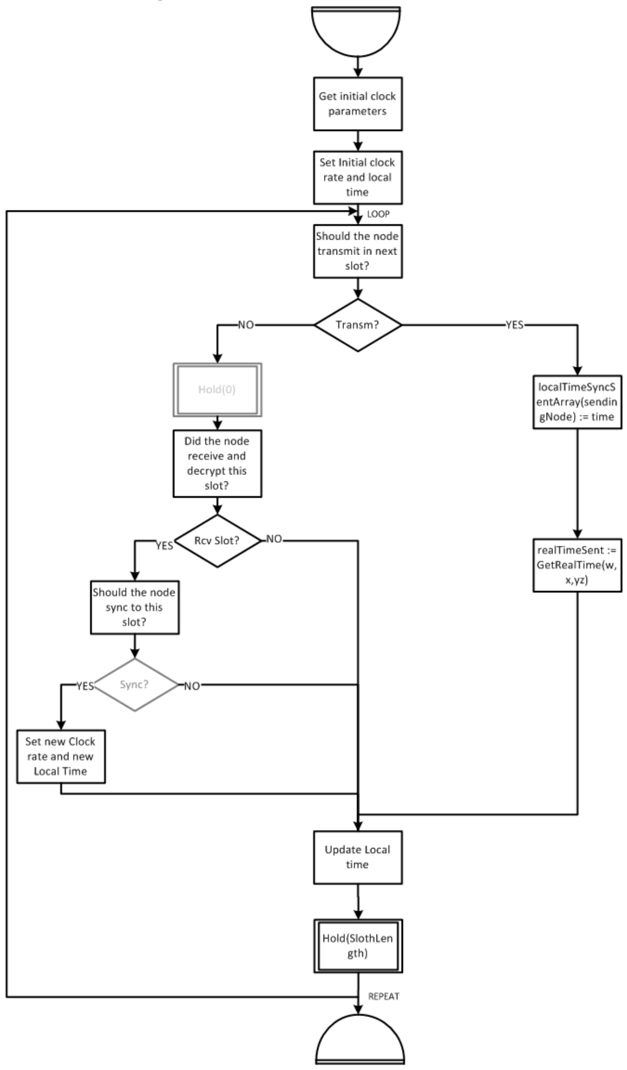

Get initial clock parameters
Set Initial clock rate and local time
LOOP
Should the node transmit in next slot?
Transm?
NO
YES
Hold(0)
localTimeSyncSentArray(sendingNode) := time
Did the node receive and decrypt this slot?
Rcv Slot?
YES
NO
realTimeSent := GetRealTime(w, x,yz)
Should the node sync to this slot?
Sync?
YES
NO
Set new Clock rate and new Local Time
Update Local time
Hold(SlothLength)
REPEAT

FIGURE 12: MODEL OF SIMULATOR CLOCKS

The following NBWF properties are crucial to the design of the local clocks and the simulator in general:

- AIE decryption becomes impossible for the receiver if the relative offset between two nodes is larger than ½ time slot (11.25ms). As a consequence, it can be assumed that the relative offset between two nodes that successfully communicate is lower than 11.25 ms. The algorithm simulated will only be in use after messages have been successfully decrypted. Therefore, reception in a SF slot always provides the necessary synchronization point.

- The short time critical path assumed in this thesis can be used to simplify the simulator. Propagation delay dominates the critical path, and it is the only random delay that is considered in this simulator.
- Drift can be ignored because the time interval between synchronization messages are short and the effect of drift will be small. Based on this, a linear equation (3) can be used for local time in the local clocks.

The simulator is controlled by a parameter file that controls the length of simulation, number of iterations and number of nodes. In addition, parameters such as offset, skew, propagation delay and connectivity matrix can be based on randomly generated values or values given by the user. The built in "replicate" function in Simula/DEMOS is used to ensure that each random iteration is started with a new seed. The simulator outputs trace files that are evaluated in Mathematica. Simplifications are made to enable node mobility, node positioning and topology changes in the simulator:

- Node mobility: Node mobility is performed by changing the propagation delay directly at a given point in time. The result is that mobility does not follow a realistic movement pattern, and this creates abrupt changes at deterministic intervals. This is not realistic when compared to a real life network. The simulation results can still be used to provide information about the lower limit of the performance as the algorithm is simulated under more demanding conditions than it would meet in real life.
- Node position: Each pair of nodes has a given propagation delay, but no absolute position. This might lead to some logical

inconsistencies in propagation delays when several nodes are involved. Such as: Node A has a propagation delay of 0.01 ms from both Node B and Node C while the propagation delay between Node B and Node C is 0.15 ms. As mentioned, this will create an illogical placement of nodes, but it should not impact the results of the simulations.

- Topology changes: Topology changes are only possible for simulations with random topologies. Simulations on topology 1-3 are performed with a static topology that is used throughout the entire simulation. Topologies are changed at deterministic intervals where the entire topology is changed by drawing a new random topology.

5.1.1. *VALIDATION OF THE SIMULATOR*

Initial validation of the simulator and the synchronization algorithms were performed by running basic deterministic parameterizations and check if these gave the expected results. The interpretation of the simulation results were based on trace files containing local time and offset at a given point in time. Errors were corrected directly, and validation was performed again to check the results.

In addition to the initial validations, a mathematical validation of the simulator was also performed. This was done by running the simulator with a basic algorithm created for this purpose. The algorithm uses the relative offset to adjust the local time of a node directly after reception of a message in a SF slot as shown:

$$\tau_2^+ = \tau_1 \tag{11}$$

Here, τ_1 and τ_2 are the synchronization points shown in Figure 7 and $\tau_2^+$is the new local time that the local clock is adjusted to. The purpose of this algorithm is to create synchronization results that are simple to validate. Figure 13 shows graphs of what the result of such an algorithm would look like for a NBWF network with only two nodes. The offset between node 1 (N1), node 2 (N2) and real time are zero at startup. The propagation delay between N1 and N2 is zero, and the clock skew of N1 is negative and the skew of N2 is positive.

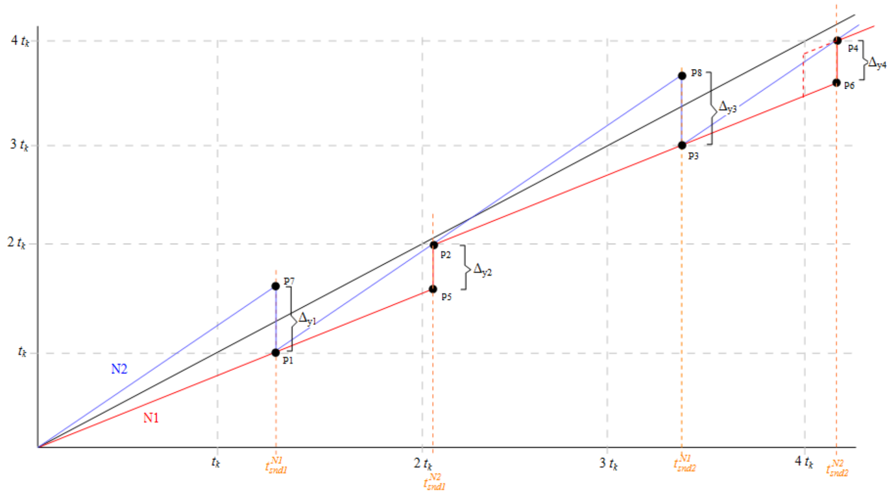


FIGURE 13: SYNCHRONIZATION POINTS USED FOR VALIDATION

The local time of the two nodes *N1* and *N2* are shown as blue and red graphs. These graphs have slopes controlled by the skew value, a_1 and a_2. The black line represents real time with zero skew and a rate equal to 1. Wake-up interval t_k represent the points in real time where the simulator wakes up. The messages used for synchronization are transmitted in SF slots at real time $t_{snd\#}^{N\#}$. The simulator wakes up for the first synchronization instance at real time t_k (i.e. at the start of the SF slot) and the simulator estimates that N1 transmit a synchronization beacon at real time t_{snd1}^{N1}. This beacon is immediately used by N2 which calculates a new local time based on equation (11). The result is that N2 can adjust the local clock to the correct value (due to zero propagation delay). N2 will transmit its synchronization message in the next SF slot, at real time t_{snd1}^{N2}. This allows N1 to synchronize.

A general mathematical expression for each synchronization point *P#* can easily be obtained based on a1, a2 and t_k. This expression is used show that the mathematical and simulated results are the same. In addition, two mathematical generalizations can be observed in order to verify the simulator. First of all it can be proven that a curve drawn through the points P0, P2 and P4 (or P0, P1 and P3) will be a straight line. Finally, it can be proven that the relative offset at each synchronization point for a node will be constant. A comparison between mathematically calculated values and a parameterization of the simulations yields identical result. Based on this, the underlying model of the simulator is validated. The mathematical expressions, calculations and simulation results can be found in appendix B.

5.2. SIMULATION OUTPUTS AND STATISTICS

The simulator creates one individual trace file for each iteration. A single iteration (and trace file) represent a unique network with unique random parameters. The trace files generated by the simulator contains the entire simulation including the transient period. The actual values saved for each synchronization point (SF slot) are: *simulator real time, average relative offset and maximum relative offset*.

The trace files are evaluated by Mathematica which reads all values from all iterations to tables before estimation begins. First, all average relative offsets are dropped as it was decided not to use these values. After this, the Mathematica tables contains *simulator real time* and *maximum relative offset.* Figures showing convergence and rejected iterations are based on data containing complete iterations that include transient periods. Figure 26 is an example of a figure showing the convergence and transient period of a network. The figure consist of two graphs. The first graph show the sample mean and confidence interval of all iterations for all synchronization points. The second graph show the percentage of networks where all nodes pair have a relative offset below 1 ms. After these figures are created, iterations are rejected based on the following rules:

- Iterations are rejected if the last maximum relative offset is above 11.5 ms. This represent a network where the synchronization algorithm has failed to synchronize all nodes. This network contains one or more nodes where the local clocks runs freely because they have lost the ability to decrypt AIE. The result is an extremely high maximum relative offset and a situation that must be solved with network merge or reentry. These iterations are marked “SYNC” in the figures showing rejected iterations.
- Iterations that have extremely long convergence time are also rejected from further estimation. These iterations are rejected based on the estimated mean of all maximum relative offset values of an entire iteration. A high mean value indicates that one or more nodes uses >1000 seconds before they converge towards values below 1 ms. This is clearly a failure of the algorithm that must be detected by other mechanisms in

NBWF. This should also lead to a network merge or a reentry. These iterations are marked “CNVRG” in the figures showing rejected iterations.

The rejected iterations are removed along with the transient period for the remaining iterations. These final results are used to estimate maximum relative offset for stationary networks. These estimates are performed in two steps. The first step estimates the mean value of all maximum relative offsets in an iteration. These values are used to create the box plots or used as a basis for step two. The second step uses the mean value from step 1 to estimates the final mean and 99% confidence interval based on all remaining iterations.

5.3. Simulations

Some NBWF network will be fully connected where all nodes can communicate with each other, but analysis of NBWF in section 2.2.2 showed that the synchronization algorithm also must support multiple hops. The introduction of multiple hops can create topologies that are more demanding to synchronize. Because of this, simulations will be performed with fully connected networks as well as topologies that includes multiple hops. In addition to this, simulations will be performed with random topologies. These random topologies will represent more typical NBWF networks which is similar to topology one, but not fully connected.

NBWF nodes can experience relative offsets as high as 11.5 ms before AIE decryption fails. Relative offsets as high as 11.5 ms are the result of short periods without communication. By using equation (2) and ρ=5 ppm, a maximum relative offset of 11.5 ms might be reached after about 20 minutes without communication. As long as the period without communication is short, AIE encryption will work, and the synchronization algorithm should receive the necessary messages to perform synchronization again. If this works, the use of additional network merge mechanisms can be avoided for short periods without communication. Initial simulations for all topologies will be performed with 11.5 ms maximum initial relative offset, which is the maximum relative offset tolerated by AIE. These simulations are performed to check if the algorithm can solve simple network merge case where the nodes still can perform AIE decryption of the signal.

It is expected that the mechanism providing network affiliation and coarse synchronization will have a precision better than 1 ms. However, 1 ms is used to add a safety margin and additional simulations are performed with 1 ms maximum initial relative offset. These simulations are performed to verify how the algorithm manages synchronization after a normal network affiliation. 1 ms is considered as the lowest maximum initial relative offset the algorithm must be able to synchronize to enable NBWF support.

All simulations are performed with all SF slots utilized and error free. This is done to ensure that it is easy to control the simulation results and have full control over how the algorithm effects network convergence. Table 6 show the most important simulation parameters used in the simulator. The different simulations will be described below.

Parameter	Value	Random/Deterministic	Distribution
Skew	-5 to 5 ppm	Random	Uniform
Maximum initial relative offset	0-11.5 ms or 0.1 ms	Random Random	Uniform Uniform
Propagation delay	0-0.2 ms	Random	Uniform
Movement (New propagation delay)	0-0.2 ms	Random	Uniform
Time between node movement	10.125 s	Deterministic	
DNS N_i	1-50	Deterministic	
Number of nodes	1-50	Deterministic	
Random topology	0 (no connection) or 1 (connection) for each node pair	Random	Bernoulli, p=0.95
Time between topology changes for random topologies	1012.5 s	Deterministic	

TABLE 6: IMPORTANT SIMULATION PARAMETERS

Initial simulations to determine the value of N_i

The article written by Saarnisaari [31] do not go into specifics about how the value of the N_i parameter affects the DNS algorithm. As a result, initial simulations are needed to find the effect of different values of N_i on NBWF networks using the DNS algorithm. In their work about the hybrid algorithm, Saarnisaari and Vanninen recommend that the N_i value is set high to avoid unnecessary switches between GNSS based synchronization and mutual synchronization. This might be useful in a network with a dedicated control channel that can transmit a number of messages relatively fast. NBWF lacks this dedicated control channel and uses SF slots instead. This makes it difficult to increase the number of transmitted messages to ensure convergence and precision. The first simulations will be performed with different values for N_i to find the optimal value to use in NBWF.

Topology 1

Represent a single broadcast domain where every node is within transmission range of all other nodes as shown in Figure 14. Simulations on topology one are performed to check if the algorithm can support NBWF under ideal conditions. Simulations will be performed for network sizes between 2 and 50 nodes. Maximum allowed distance is 60 km and nodes change position every 10 seconds while and the connection matrix is fixed.

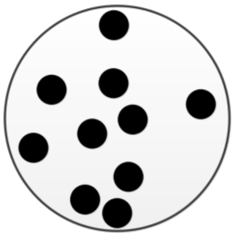


FIGURE 14: TOPOLOGY 1 – FULLY CONNECTED NETWORK

Topology 2

Topology 2 is created to simulate a chain of nodes as shown in Figure 15. In real life, this could represent a vehicle convoy utilizing IED jammers severely limiting the range of the radios or nodes separated by terrain features. The main objective of this topology is to provide insight in the effect of multiple hops on the algorithm. NBWF requires that voice is supported over 2 relays while data must work for any number of hops.

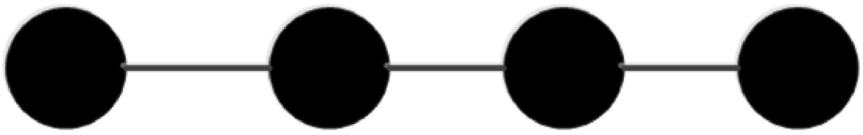


FIGURE 15: TOPOLOGY 2 - A CHAIN OF NODES

Simulations will be performed with up to 10 nodes to give better insight in the effect multiple hops has on synchronization. As for topology 1, the first simulations are performed with node movement every 10 seconds, maximum distances of 60 km and 11.5 ms maximum initial relative offsets. Each additional relay can lead to a 0.2 ms (equal to 60 km) increase in the relative offset. For 3 relays (5 nodes), the maximum error caused by propagation delay can be 0.8 ms. This leaves little room for the algorithm to achieve a precision better than 1 ms. It is expected that topology 2 networks will be difficult to synchronize with these extreme values, but the result of these simulations will show the algorithms ability to merge networks when initial relative offsets increases beyond 1 ms.

Simulations with more typical parameters must be performed for topology 2 to determine if the algorithm can support NBWF. The maximum initial relative offset is changed from 11.5 ms to 1ms. This represent the network after coarse synchronization. This represents a minimum initial relative offset the algorithm must be able to synchronize if it should be considered for NBWF. All topology 2 networks are performed with moving nodes that change position every 10 seconds and the connection matrix is fixed.

Topology 3

This topology represent a network divided into two clusters as shown in Figure 16. To make it more demanding, the two clusters are connected via a chain of 2 relays. This could represent two larger units that have lost direct communication due to terrain formation.

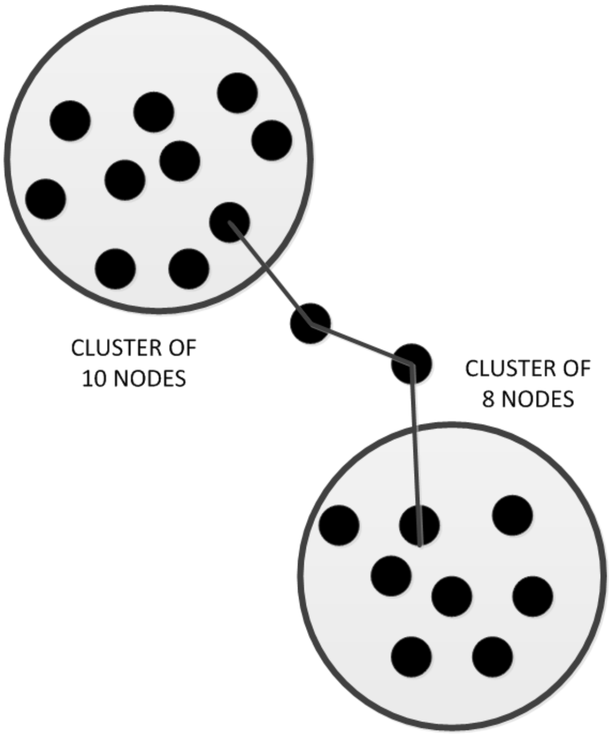


FIGURE 16: TOPOLOGY 3 – TWO CLUSTERS OF NODES

As for the two other topologies, a short period without communication is simulated by using high initial relative offsets at the start of the simulation. Simulations will be performed on networks with 20 nodes in a single fixed topology. The reason for this is that it is time-consuming and difficult to control such topologies in the simulator. The simulation results will be limited by the fact that only one of many possible combinations is simulated. However, it is expected that this topology will provide additional insight in how the algorithm performs

As for the two other topologies, additional simulations with maximum initial relative offset of 1 ms are performed. This is the minimum initial offset the algorithm must support. All simulations are performed with node movement and a fixed connection matrix.

5.4. Simulation results

5.4.1. *Initial simulations on the DNS algorithm*

It was assumed that N_i=*NumNodes-1* is a good value to use for the DNS algorithm. This is the lowest value that ensures that a node waits at least one superframe before it synchronizes. This should help prevent unnecessary switches between GNSS based and mutual synchronization in the hybrid algorithm. Simulations performed on a fully connected static (without node movement) network with 10 nodes uncovered a problem with this value. As Figure 17 shows, this value (N_i=*9*) causes the algorithm to converge towards a high and stable value. Figure 17 also show that this happens for multiples of N_i=*number of nodes-1* (N_i=*18*). Other N_i values do not show this tendency. Additional simulations were performed with different sized networks, and this confirms that this happens for networks of all sizes.

.

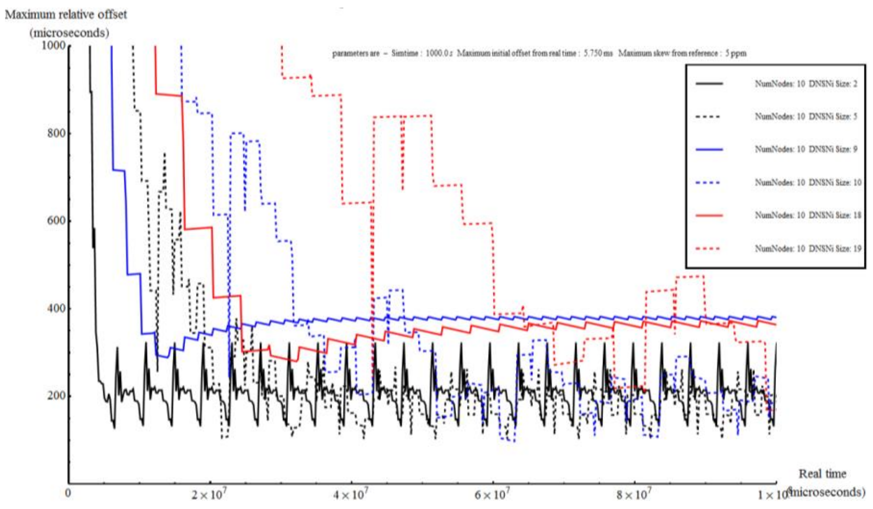


FIGURE 17: EFFECT OF N_I FOR A STATIC NETWORK OF 10 NODES

The tendency to converge towards a high and stable value when N_i=*number of nodes-1* was studied in more detail by doing a number of simulations on a fully connected network of 10 nodes. A closer inspection of the simulation results shows that only some of the iterations converge towards a high and stable value for networks with 10 nodes and *Ni=9*. This is shown in Figure 18.

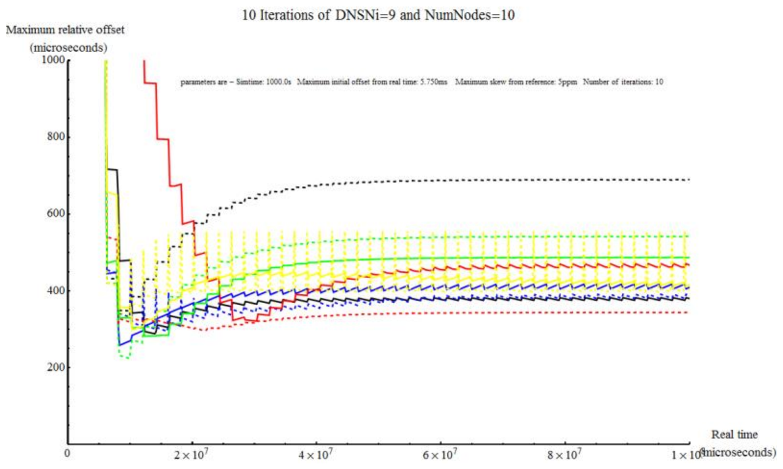


FIGURE 18: 10 ITERATIONS OF NETWORKS WITH 10 NODES AND N_I=9

Several additional simulations were performed to try to find an explanation for this behavior of *N_i=Number of nodes-1*. These simulations gave no conclusive answer, but the tendency disappears if node movement is enabled or for static networks with delay compensation. Further simulations might reveal the exact cause of this behavior. This is time consuming and not performed in this thesis. Based on this, *N_i=Number of nodes-1* should be avoided as much as possible. A close inspection of all 200 iterations of a simulation with 10 nodes and *N_i=10* shows stable results without a tendency to converge towards high value. Therefore, *N_i=number of nodes* is selected as a basis for further simulations, and considered the lowest value of N_i that can fulfill Saarnisaari and Vanninens intention to keep changes between GNSS and mutual synchronization at a minimum.

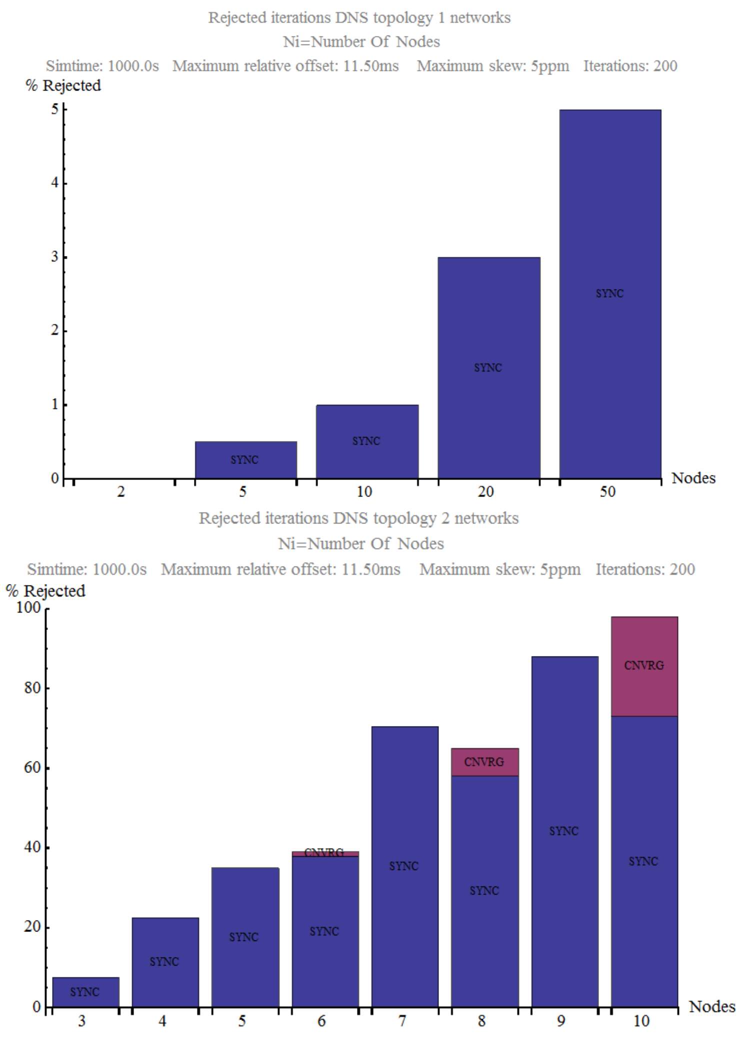


FIGURE 19: TOPOLOGY 1 AND TOPOLOGY 2 WITH N_I=NUMBER OF NODES

Figure 19 show the amount of rejected iterations for simulations with N_i=*number of nodes* for topology 1 (above) and topology 2 (below). A high number of rejected iterations indicates problems with synchronizing the network. The figure show that the DNS algorithm manages to synchronize topology 1 network with few problems. The number of rejected iterations are low (<5% for all network sizes) and is a result of high initial relative offset. For topology 2 networks, the DNS algorithm experiences problems with networks larger than 3 nodes (more than 1 relay) where >20% of the iterations are rejected. Most iterations are rejected because relative offsets are higher at the end of the simulation than the original initial offset. These represent networks that the algorithm were unable to synchronize and marked "SYNC" in the graphs. In networks with 6, 8 and 10 nodes, additional iterations were rejected because they had long convergence time. These are marked "CNVRG."

Simulations on topology 2 networks show another surprising property of N_i. The performance of the algorithm is inconsistent as the value of N_i varies with the number of nodes. Intuitively the maximum relative offset should increase as the network size increases (number of hops).

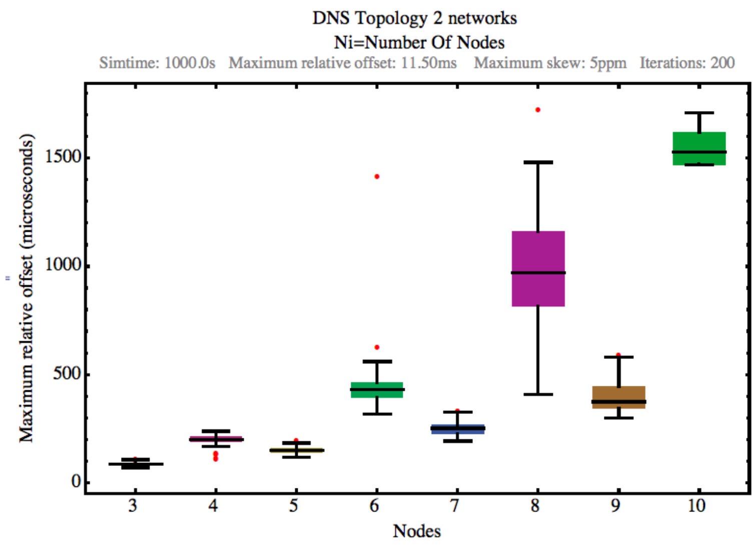


FIGURE 20: SIMULATION RESULTS FOR TOPOLOGY 2 NETWORKS

Figure 20, which show the result of all iterations that was not rejected, indicates that this is not the case. The figure shows that the algorithm performs best when N_i is an odd value. Simulations with longer simulation time (10000 seconds) and more iterations (500) confirms that this is not caused by few accepted iterations for the larger networks. The direct cause of this is not clear, but it is assumed that the algorithm performs better if N_i is an odd value.

Topology 3 is even more demanding to synchronize, and more than 90% of the iterations are rejected for a network of 20 nodes with $N_i=20$. For the remaining iterations, DNS was incapable of delivering sufficient precision within 1000 seconds (simulation time). Additional simulations on a topology 3 network with 20 nodes and different values for N_i, was performed to verify if this reduces the number of rejected iterations. Figure 21 show that the algorithm performs best with $N_i=3$ but with some improvement for $N_i=5$, $N_i=7$, $N_i=9$ and $N_i=10$.

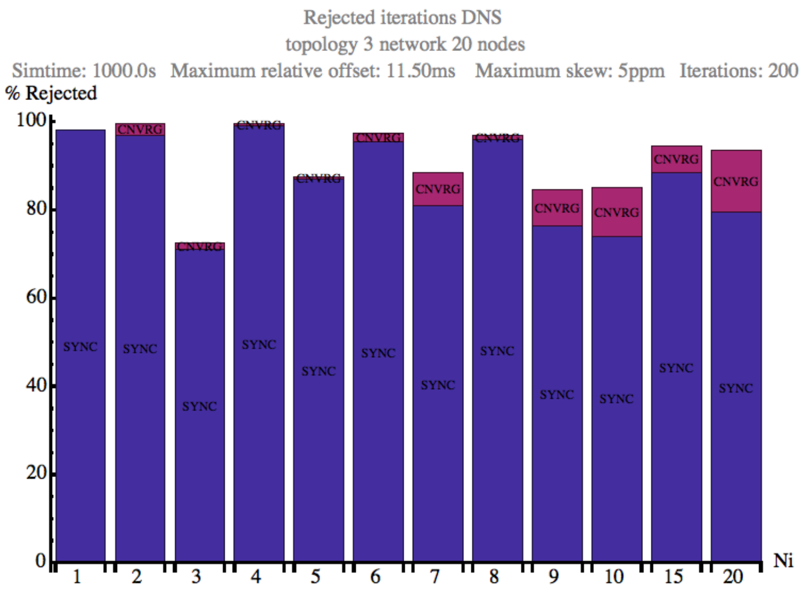


FIGURE 21: TOPOLOGY 3 NETWORK WITH 20 NODES AND DIFFERENT N_I VALUES

The result shown in Figure 21 and Figure 19 are not enough to reject the DNS algorithm outright because they are a caused by extreme initial relative offsets (as high as 11.5 ms) and high propagation delay (as high as 60 km). However, the results do indicate that the DNS algorithm is unable to do network merge for demanding topologies even with initial relative offset <11.5 ms.

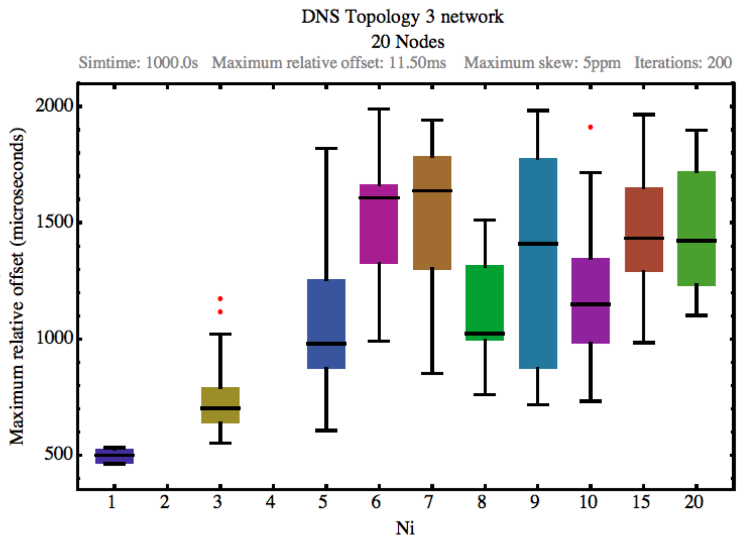
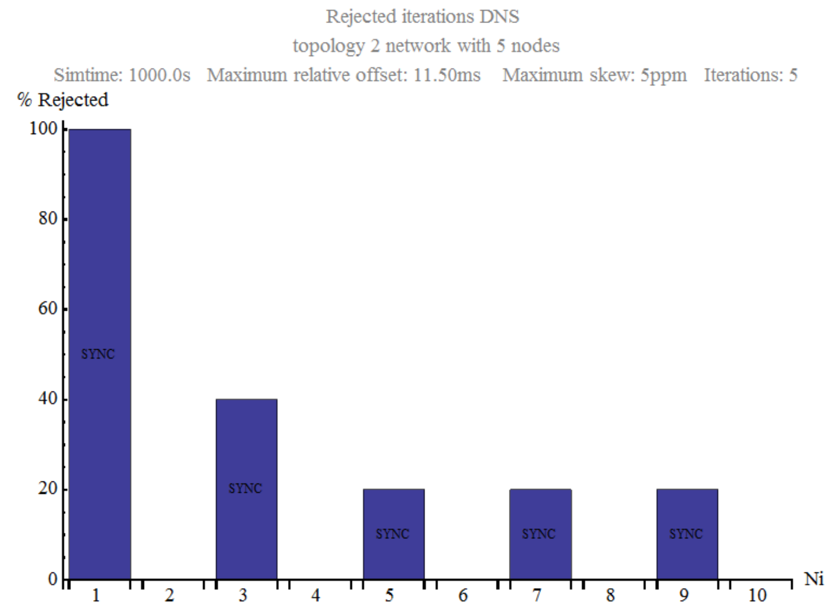


FIGURE 22: BOXPLOT OF SUCCESSFULLY SYNCHRONIZED ITERATIONS OF TOPOLOGY 3 NETWORKS WITH 20 NODES AND DIFFERENT N_I VALUES

Figure 22 is based on the same simulations as Figure 21 and shows the box plot of the successfully synchronized iterations. Figure 22 must be interpreted with the result of Figure 21 in mind because a lot of iterations are rejected. Figure 22 and Figure 21 show that $N_i=3$ or $N_i=5$ lead to best precision in NBWF. $N_i=1$ should not be used be due to extremely high numbers of rejected iterations.

The fact that DNS performs best with $N_i=3$ is supported by Figure 23 and Figure 24. Figure 23 show simulation results of a topology 2 network with 5 nodes and varying values of N_i. Figure 24 show a topology 1 network with 10 nodes and varying values for N_i. These results show that $N_i=3$ is the best value for the DNS because it provides better precision and shorter convergence time. The downside of using this value is that it will cause more switches between GNSS based and mutual synchronization when DNS is used by the hybrid algorithm. The switches between GNSS based and mutual synchronization will cause some fluctuations in time for the nodes involved. However, this should be preferred over the asynchronization caused by long convergence time introduced by high N_i values needed to avoid switches in the hybrid algorithm. $N_i=3$ seems to provide the best performance for complex networks (topology 3) and this value will be used for all subsequent simulations.



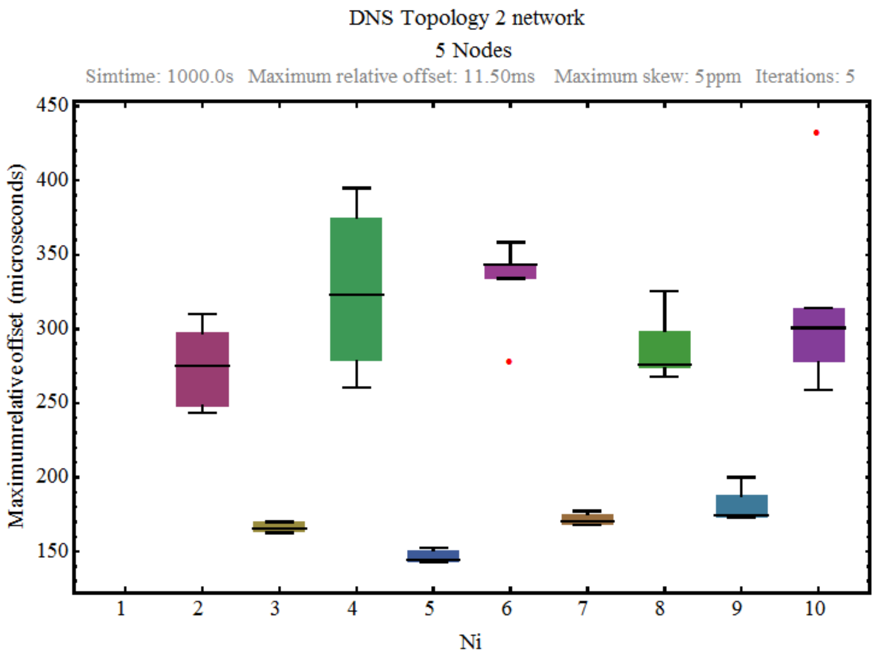
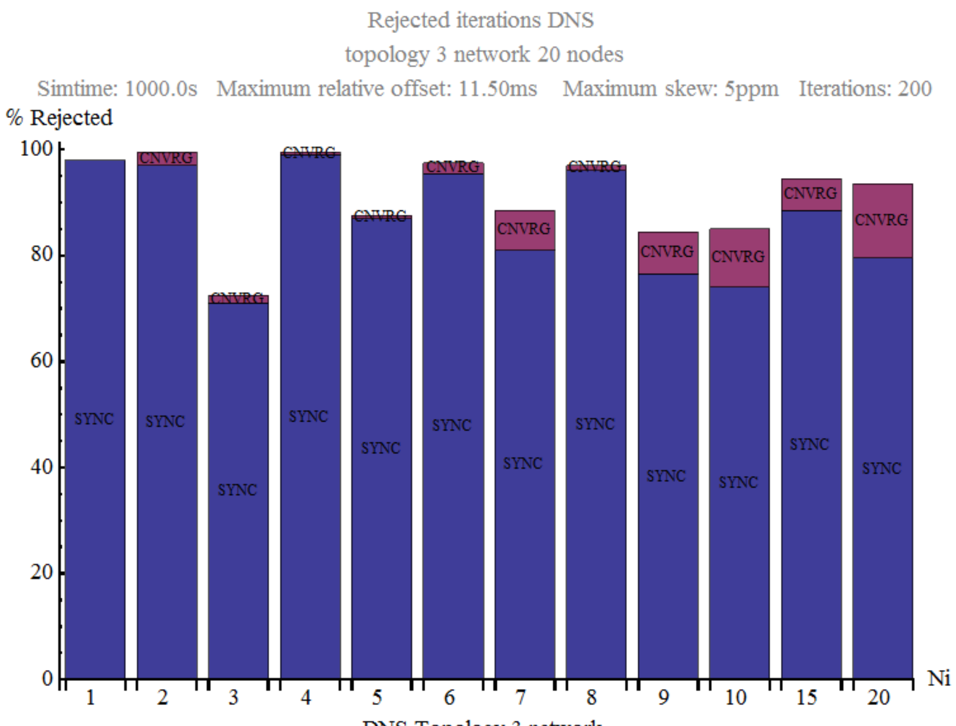


FIGURE 23: TOPOLOGY 2 NETWORK WITH 5 NODES AND DIFFERENT N_I VALUES



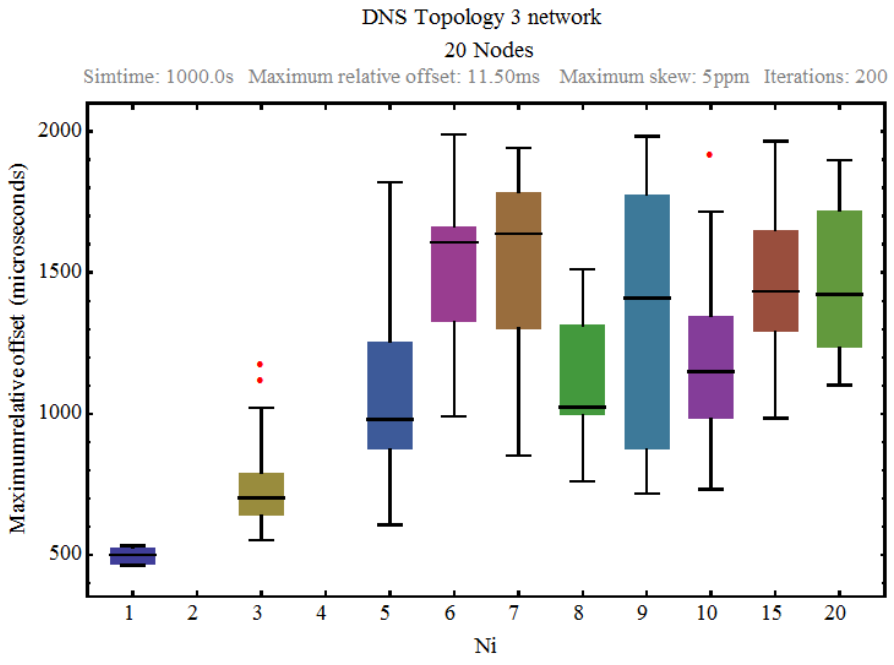


FIGURE 24: TOPOLOGY 1 NETWORK WITH 10 NODES AND DIFFERENT N_I VALUES

5.4.2. *DNS ALGORITHM IN TOPOLOGY 1*

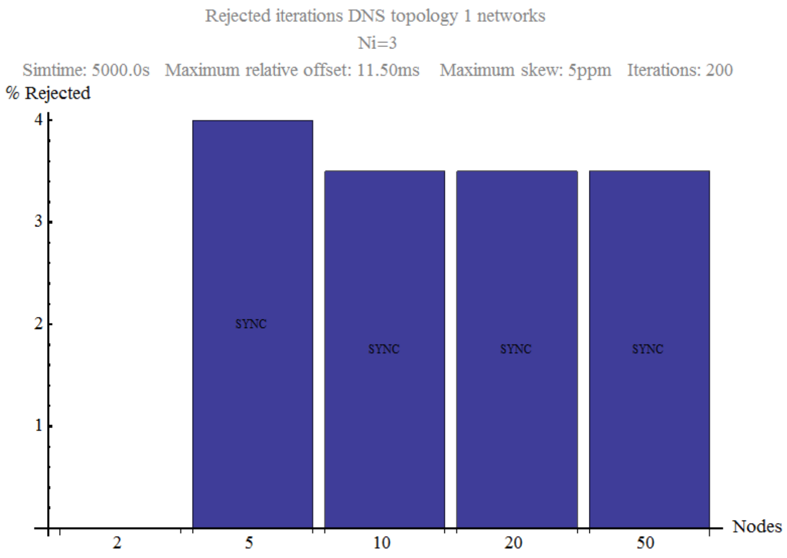


FIGURE 25: REJECTED ITERATIONS (TOPOLOGY 1)

Topology 1 is simulated for up to 50 nodes with a simulation time of 5000 seconds and 200 iterations. The number of simulations and simulation time provides enough to estimate confidence intervals with a confidence level of 99%. All simulations are performed with moving nodes and $N_i=3$. Figure 25 show the amount of rejected iterations for topology 1 networks with maximum initial relative offset <11.5 ms. About 4% of the iterations are rejected for networks of 5 nodes or more when the maximum initial relative offset is 11.5 ms. These iterations are rejected because they fail to synchronize the network due of high initial relative offset. This is supported by the fact that no iterations are rejected when the maximum initial relative offset is reduced to 1 ms.

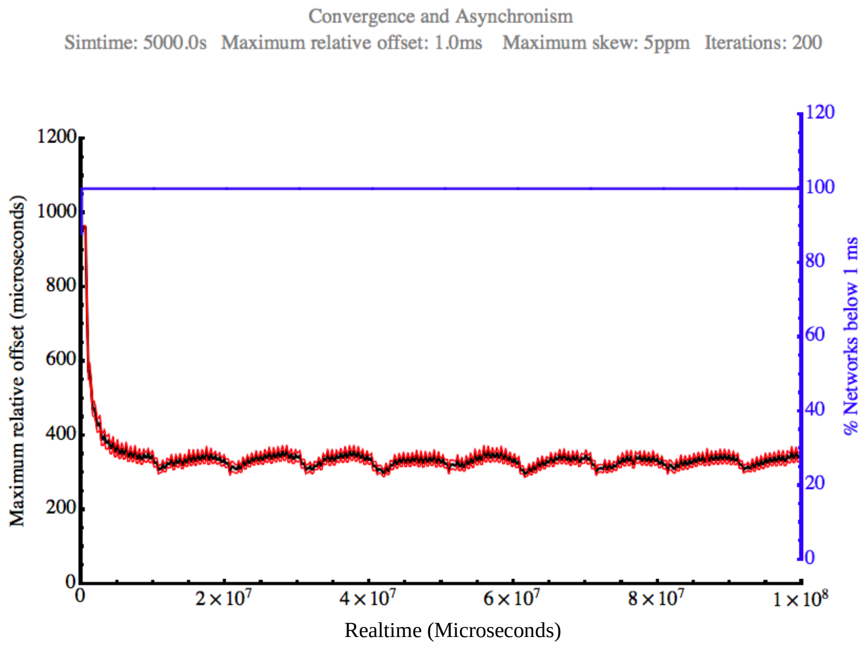


FIGURE 26: CONVERGENCE FOR NETWORKS WITH 50 NODES (TOPOLOGY 1)

Figure 26 show the mean value and 99% confidence interval for the maximum relative offset for all networks of 50 nodes. In addition to this, Figure 26 also shows the percentage of networks that have all node pairs within a maximum relative offsets of 1 ms. A network size of 50 nodes is chosen because this is the most demanding network simulated and smaller networks are expected to have even better performance. These two graphs give a clear view the transient period, convergence and performance of the DNS algorithm.

In Figure 26, the graphs show that the DNS algorithm is stable after convergence with no asynchronism because 100% of the networks have a maximum relative offset better than 1 ms. All networks converge to a value below 0.4 ms after about 10 seconds ($1*10^7$ µs). One more point of interest is that Figure 26 show the effect of node movement. The maximum relative offset reaches a minimum value before it starts to increase again. These minimum values are separated by 10 seconds ($1*10^7$ µs), which is the same value as the interval between node movements.

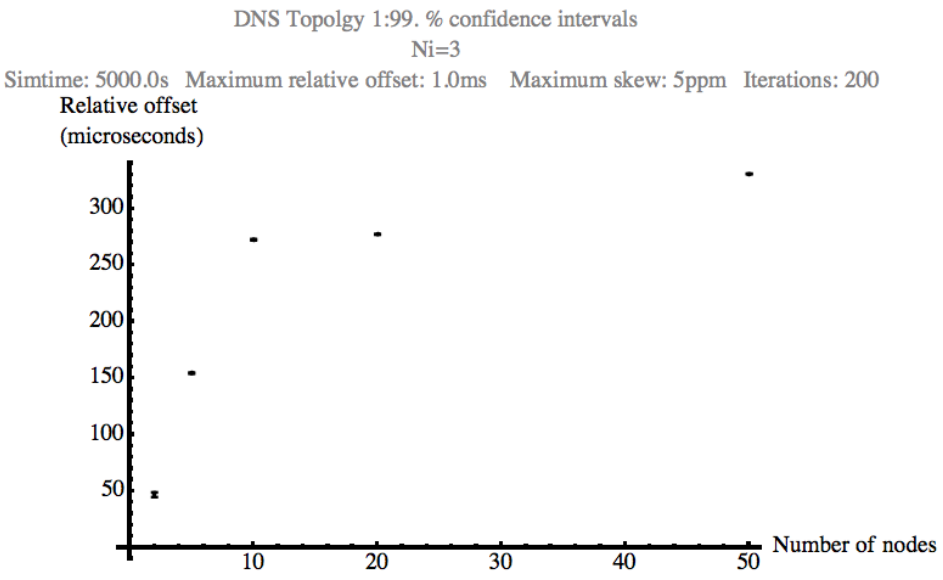


FIGURE 27: MAXIMUM RELATIVE OFFSET WITH 99% CONFIDENCE INTERVAL (TOPOLOGY 1)

Figure 27 show the estimated stationary maximum relative offset for topology 1 networks. The stationary value is found by removing the transient period. Based on the 10 s convergence time found in Figure 26, and with a safety margin, the transient period used by Mathematica is increased to 20s. The DNS algorithm is able to achieve a precision that is well within 1 ms for all simulated network sizes, with the exception of the rejected iterations shown in Figure 25. There is a small inconsistency in the simulation results for networks with 10 and 20 nodes. A larger difference between the estimated results for these two network sizes is expected. An increase in the number of iterations yields the same result, and the cause of this is unknown. Additional simulations show that this tendency is not as dominant for other values of $N_{i,}$ and it is concluded that the DNS algorithm supports topology 1. It is also concluded that the DNS algorithm is capable of performing network merge for most topology 1 networks as long as the initial relative offset is below 11.5 ms.

5.4.3. DNS ALGORITHM IN TOPOLOGY 2

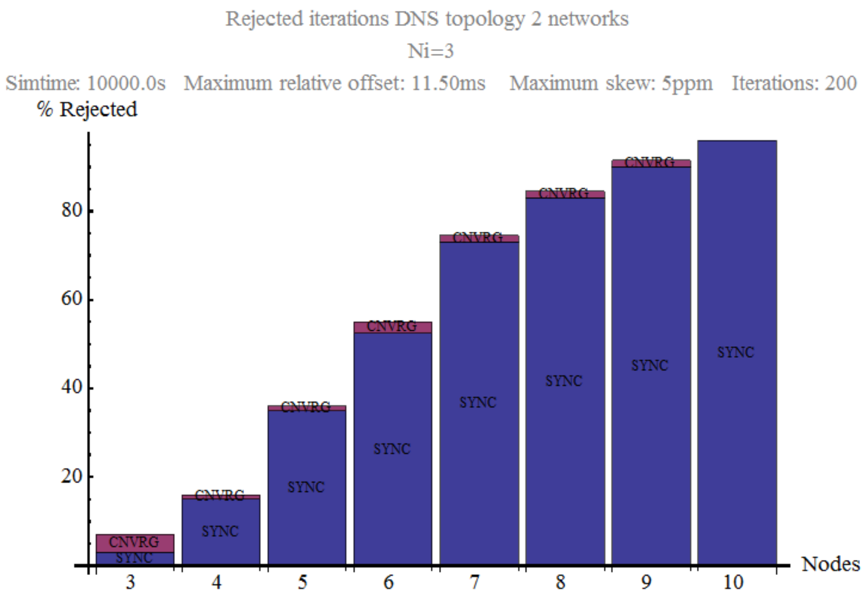


FIGURE 28: REJECTED ITERATIONS (TOPOLOGY 2)

Topology 2 is simulated with network sizes up to 10 nodes. A network of 10 nodes represents two nodes connected via 8 relays and a network size of 4 represents two nodes connected via 2 relays. The simulation time is increased to 10000 seconds to account for the possibility of longer transient periods. Figure 28 show the amount of rejected iterations for topology 2 networks when maximum initial relative offset is as high as 11.5 ms. About 15% of the iterations are rejected for a network with two relays (4 nodes). The amount of rejected iterations increases with the number of relays. This shows that the DNS algorithm has difficulty merging networks with 3 or more relays, even if the maximum initial relative offset is <11.5 ms. As for topology 1, no iterations are rejected if the maximum initial relative offset is reduced to 1 ms. The following results are estimates of simulations with a maximum initial relative offset of 1ms.

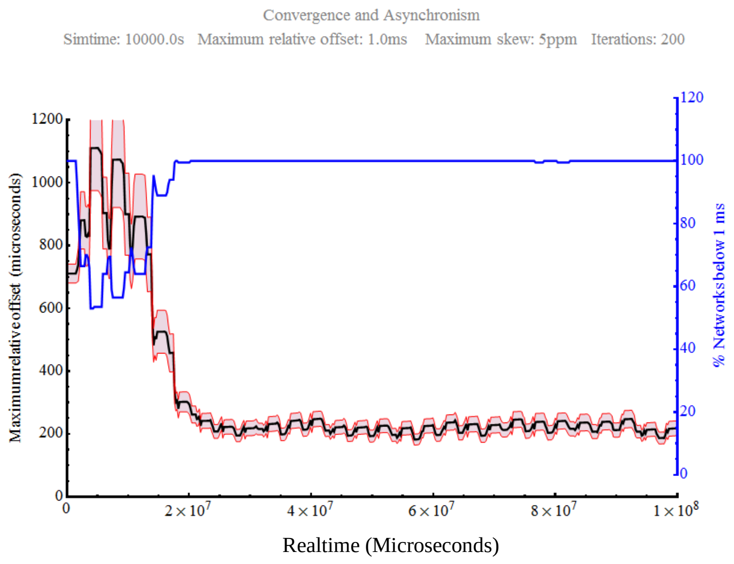


FIGURE 29: CONVERGENCE FOR NETWORKS WITH 6 NODES (TOPOLOGY 2)

Figure 29 show the mean value and 99% confidence interval together with the percentage of asynchronous nodes for networks with 6 nodes and maximum initial relative offset of 1 ms. This shows that all the networks converge to about 0.2 ms after about 20 seconds.

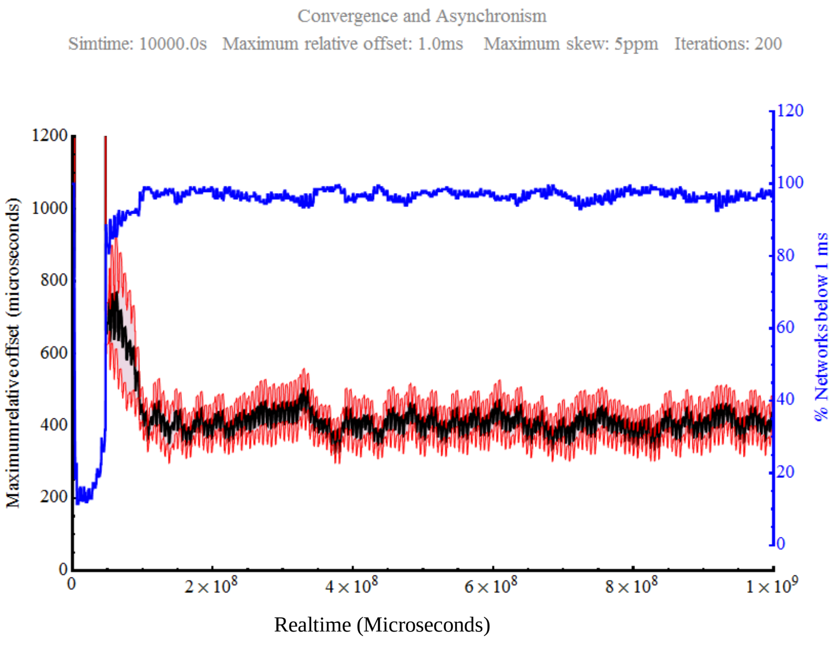


FIGURE 30: CONVERGENCE FOR NETWORKS WITH 10 NODES (TOPOLOGY 2)

Figure 30 show simulations of topology 2 networks with 10 nodes. In this figure, it becomes apparent that DNS struggles as it is unable to provide a precision better than 1 ms for some networks. Convergence time has also increased to 100 seconds. As a result, transient period is set to 200 seconds for the remaining estimations, where the stationary values are estimated.

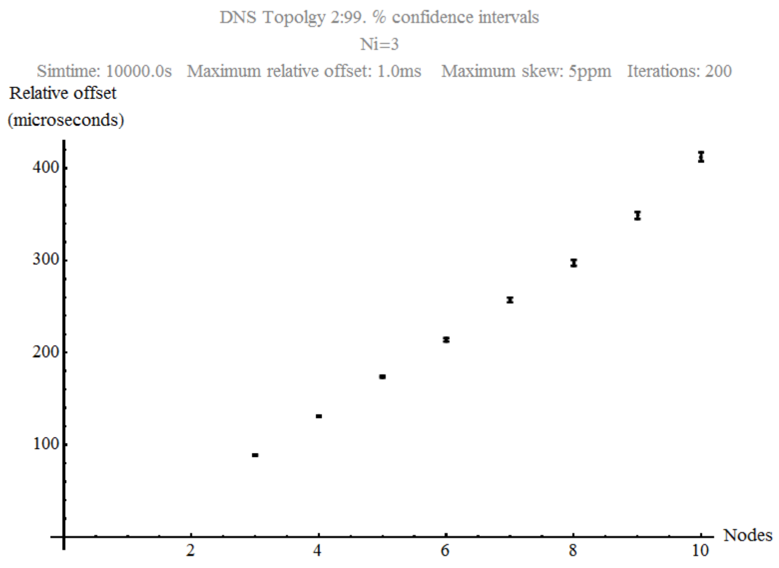


FIGURE 31: MAXIMUM RELATIVE OFFSET WITH 99% CONFIDENCE INTERVAL (TOPOLOGY 1)

Figure 31 shows the estimated relative offset for topology 2 networks when maximum initial relative offset is below 1 ms. All values are well within the 1 ms requirement of NBWF. The simulation results in Figure 29 and Figure 31 show that DNS can support at least 4 relays (6 nodes) and that it might be able to synchronize larger networks depending on initial parameters. This should be sufficient for most NBWF networks because TDMA collisions become more and more unlikely as the number of relays increases.

5.4.4. DNS ALGORITHM IN TOPOLOGY 3

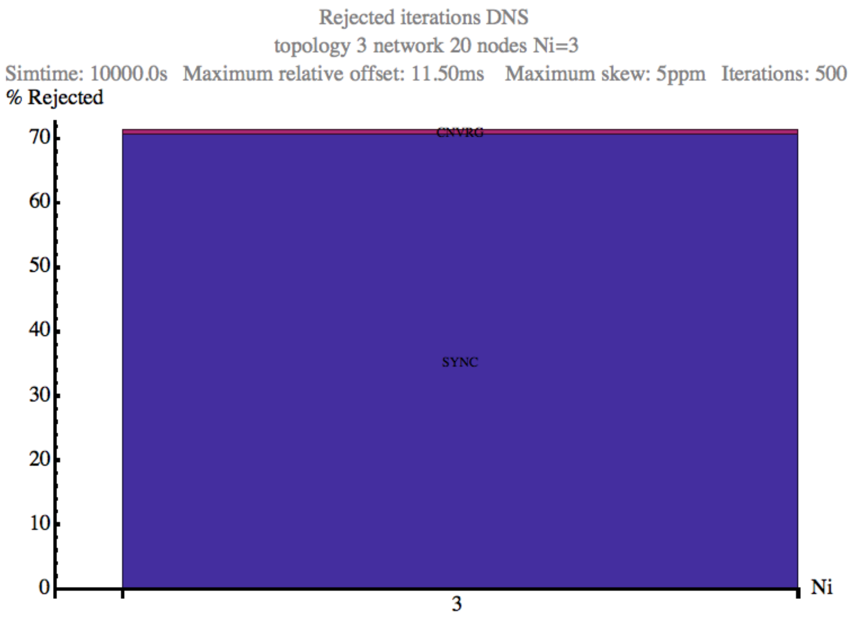


FIGURE 32: REJECTED ITERATIONS (TOPOLOGY 3)

As mentioned in section 5.3, topology 3 is only simulated for a network of 20 nodes. Simulation time is kept to 10000 seconds, and the number of iterations is increased to account for possible lost iterations. Simulation results, presented in Figure 32, show that over 70% of the iterations are rejected. As a result, it is concluded that the DNS algorithm is unable to merge topology 3 networks. An external network merge mechanism must be used, even if the maximum initial relative offset is below 11.5 ms. As for the two other topologies, no iterations are rejected when the maximum initial relative offset is reduced to 1 ms. This relative offset was used in the rest of the topology 3 simulations.

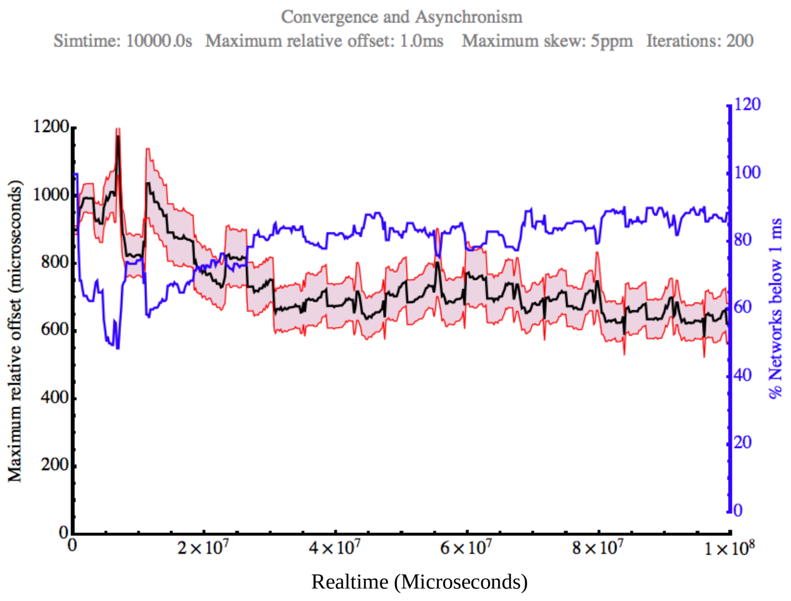


FIGURE 33: CONVERGENCE FOR NETWORKS WITH 20 NODES (TOPOLOGY 3)

Figure 33 shows mean value and 99% confidence interval for the maximum relative offset and the percentage of networks with maximum relative offset below 1 ms. The figure show that the networks reaches a mean value of about 0.7 ms with a confidence interval below 1 ms. However, the second graph show that the DNS algorithm does not manage to keep all the nodes in all networks within a maximum relative offset of 1 ms. About 80% of the networks have precision better than 1 ms for all nodes. In the remaining 20% of the network, one or more pair of nodes have relative offset above 1 ms, even if the confidence interval is below 1 ms. Figure 33 shows that the DNS algorithm manages to synchronize most, but not all, topology 3 networks when maximum distance as high as 60 km between each node is allowed.

The networks shown in Figure 33 can potentially cover distances over 200 km when 2 relays are used. From a military point of view, the probability that a topology 3 NBWF network cover over 60 km is low. In addition to this, the chances of TDMA collision are small when nodes are separated by such distances. Because of this, topology 3 is simulated with maximum distances of 15 km to check if the DNS algorithm manages to provide sufficient precision for all nodes.

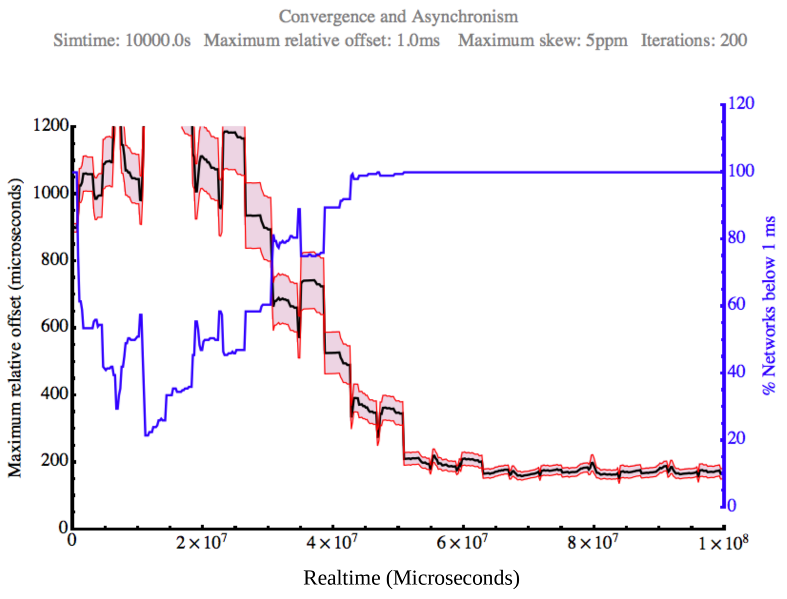


FIGURE 34: CONVERGENCE FOR NETWORKS WITH 20 NODES AND MAXIMUM DISTANCE OF 15 KM (TOPOLOGY 3)

Figure 34 show topology 3 networks with maximum distances of 15 km between each node pair. The result show that the network converges towards a value below 0.2 ms after about 65 seconds ($6.5*10^7$ µs). This shows that DNS can support the NBWF requirement of a precision of 1 ms in topology 3 as long as the total distance covered are lower than 60 km. This is also supported by Figure 35 which show that the estimated maximum relative offset is about 0.19 ms.

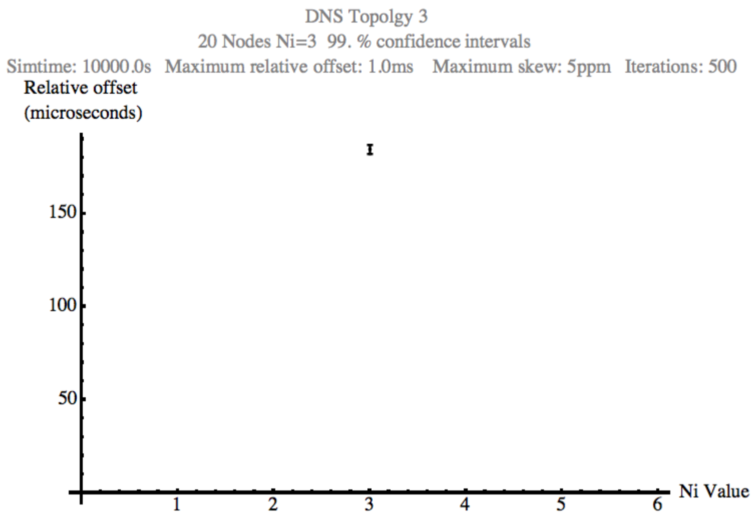


FIGURE 35: MAXIMUM RELATIVE OFFSET WITH 99% CONFIDENCE INTERVAL (TOPOLOGY 3)

5.4.5. DNS ALGORITHM IN A RANDOM TOPOLOGY

The specific conditions introduced by the different topologies affect the simulation results shown in sections 5.4.2 to 5.4.4. Random topologies (random connection matrices) are simulated to provide estimates of maximum relative offset for more normal NBWF network situations. These networks are similar to topology 1 networks, but they will not be fully connected. All simulations are performed with node movements every 10 seconds. Initial simulations are performed with a random topology that is created at startup and kept unchanged throughout the entire simulation.

Random topologies is created by drawing random values for the connection matrices. Each node pair will have a 95% change of connection. A consequence is split network topologies that it is impossible to synchronize. Iterations representing these network topologies will be treated by Mathematica in the same way as other iterations that fail to synchronize. These iterations will be rejected because some of the local clocks will diverge from each other instead of being synchronized.

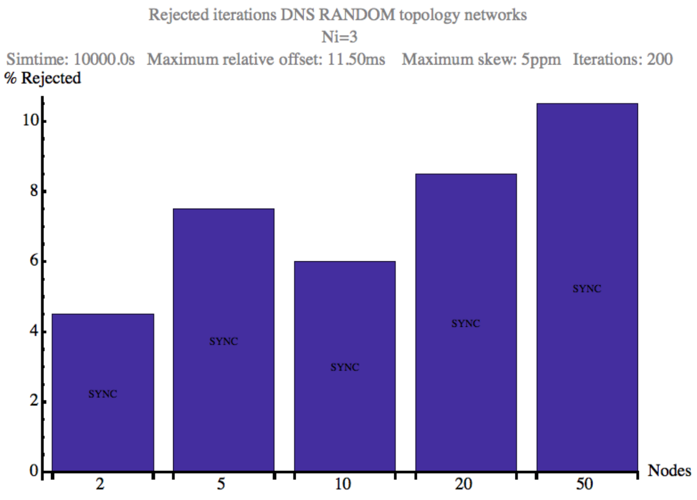


FIGURE 36: REJECTED ITERATIONS (RANDOM TOPOLOGIES HIGH INITIAL RELATIVE OFFSET)

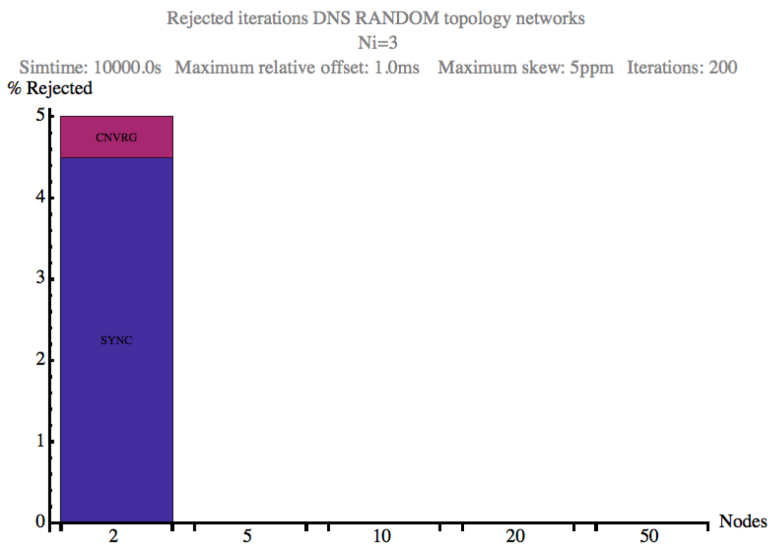


FIGURE 37: REJECTED ITERATIONS (RANDOM TOPOLOGIES LOW INITIAL RELATIVE OFFSET)

Figure 36 and Figure 37 show rejected iterations for a network with random initial topologies. For these simulations, the initial topology is kept constant during the entire simulation. This shows the effect of networks that are not fully connected. For network sizes from 5 to 50 nodes, the amount of rejected iterations is increased from around 4% to values from 6% to 10%. The increase in rejected iterations is a result of node pairs without connection creating networks that are more difficult to converge. For networks with 2 nodes, the amount of rejected iterations has increased from 0% to 5%. Figure 37 show that all networks with 5 to 50 nodes are synchronized when the initial relative offset is reduced to 1 ms. In addition to this, there are some changes in the number of rejected iterations for a network with 2 nodes that must be explained by a closer inspection of the results.

First of all, the small amount of networks labeled “CNVRG” in Figure 37 is a result of networks without connections and node pairs with relative rate close to 1. These nodes will slowly diverge from each other but has not reached a relative offset above 11.5 ms when the simulator ends. These two nodes do not communicate, but the low difference in the relative rate keep the local clocks close together. There are no networks labeled “CNVRG” in Figure 36 because the high initial relative offset causes a relative offset above 11.5 ms in all networks that fails to sync. The second important point to observe is that the total amount of rejected iterations are lower for networks with high initial offset

(4,5%) than for networks with low initial offset (5%). Intuitively, there should have been 5% split topologies and rejected iterations in both cases (Because there are 95% chance of connection). The difference is a result of the low number of iterations used. 200 iterations are not enough when a Bernoulli distribution with P=0.95 is used. Unfortunately, this was discovered too late to perform additional simulations.

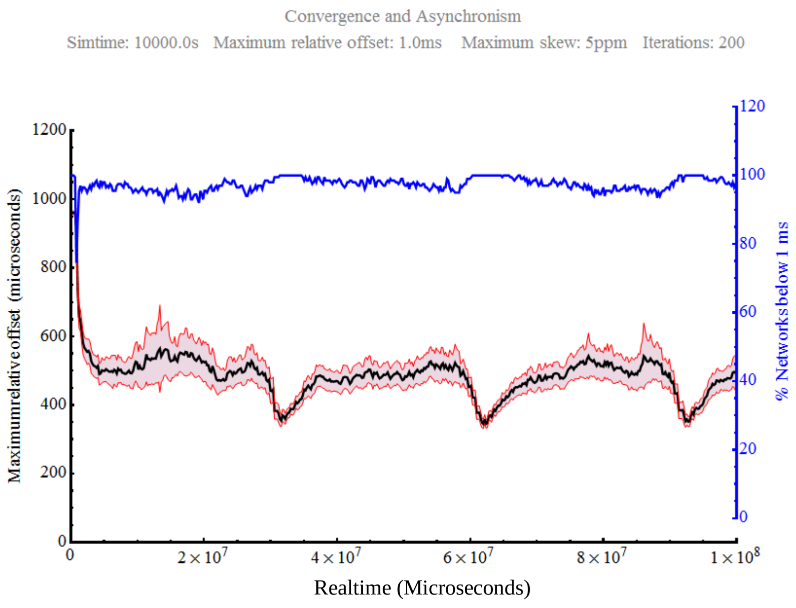


FIGURE 38: CONVERGENCE FOR NETWORKS WITH 50 NODES (RANDOM TOPOLOGY)

Figure 38 shows the mean relative offset and 99% confidence interval of a network with 50 nodes. The DNS algorithm converges towards a value from 0.4 to 0.6 ms, which is well within the NBWF requirement. The effect of node pairs without connection is clearly seen when compared to Figure 26 as the mean value is higher. In addition to this, the confidence intervals are larger. The result is that more variations in precision must be expected in networks with random topologies than in fully connected networks. Convergence time has increased from <10 seconds for fully connected networks to about 30 seconds for random topology networks.

Figure 38 also show the percentage of networks with all nodes within the NBWF requirement of 1 ms. This graph shows that the DNS algorithm struggles keeping relative offset of all nodes in all networks within 1 ms. The algorithm manages to achieve sufficient precision before the nodes starts to diverge from each other. This is probably caused by the combination of node movement and node pairs without connections.

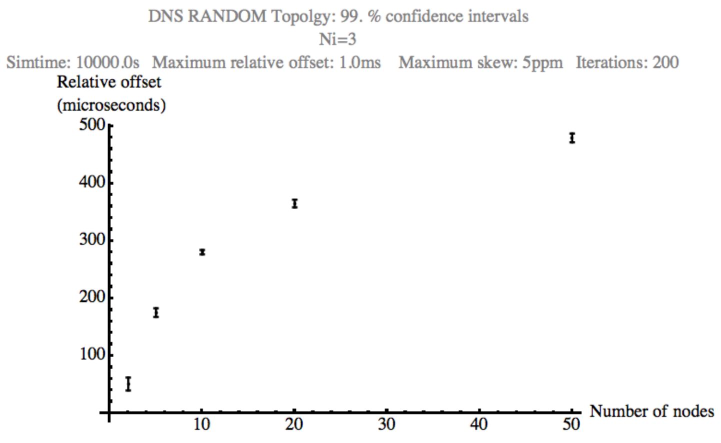


FIGURE 39: MAXIMUM RELATIVE OFFSET WITH 99% CONFIDENCE INTERVAL (RANDOM TOPOLOGY)

Figure 39 shows estimated maximum relative offset for networks with a random topology that is static throughout the simulations. This figure show that the effect of the random topology is higher mean offsets with wider confidence intervals when compared to similar topology 1 networks. This is the same effect as is shown in Figure 38.

The last simulation is performed with a changing topology where new topologies (connection matrices) are created about 1000 seconds apart to simulate the effect of changing terrain etc. Simulations was performed for 200 iterations and the result was evaluated by Mathematica. However, the output of the graph became unreadable when all 200 iterations was included because the sample mean of the maximum relative offset start to increase without a bound after a topology change. This was probably caused by maximum relative offsets that increases without bound when networks fail to synchronize, and these increasing relative offsets dominates the sample mean.

There was little time left to study these networks in detail and an output based on a selection of 25 iterations is included to show what changes in topologies might look like. Figure 40 shows the second topology change at about 2000 seconds ($2*10^9$ µs) for a network of 50 nodes. The change in topology results in a period of 30 seconds with higher maximum relative offset before the DNS algorithm manages to bring the maximum relative offset back to the same values as before.

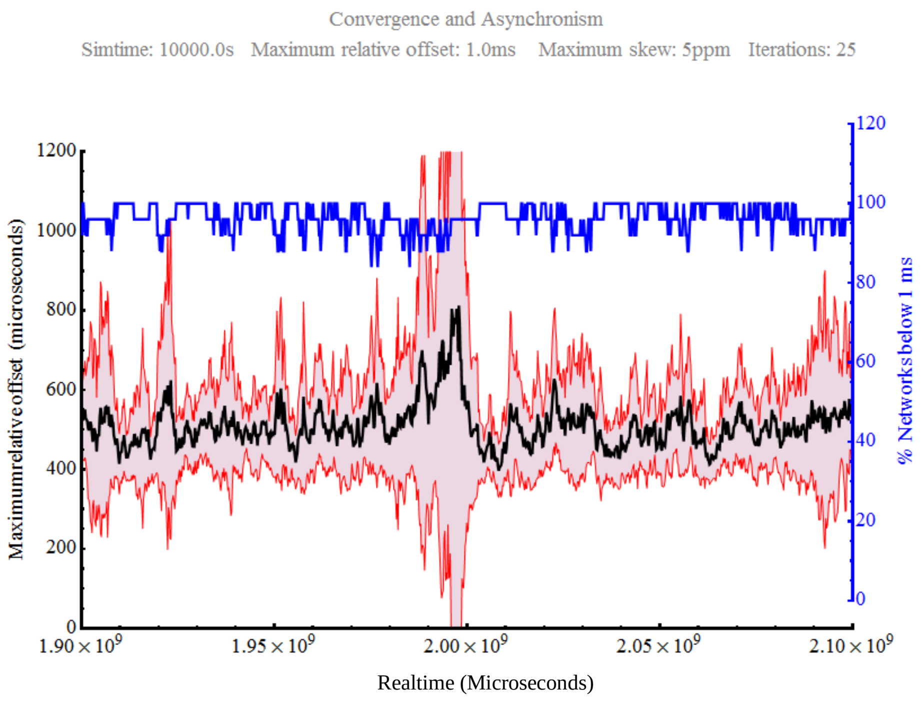


FIGURE 40: RANDOM CHANGE IN TOPOLOGY FOR A NETWORK WITH 50 NODES

The effect of topology changes must be studied further if the DNS algorithm is selected for NBWF. Except from that, the simulations described in section 5.4.5 show that the DNS algorithm supports random topologies, even if it struggles with a few percent of the networks with 50 nodes.

SUMMARY OF DNS SIMULATION RESULTS

Topology	Supports NBWF	Can merge networks with maximum offset <11.5ms	Convergence time	Maximum network size with maximum initial relative offset 1 ms. (Maximum tested)
1	Estimated relative offset < 0.4 ms	< 4% rejected iterations	< 10 seconds	50 (50) Nodes
2	Estimated relative offset <0.2ms (4 relays)	<3 relays	< 20 seconds (4 relays)	4 (8) Relays
3	Guaranteed for networks covering <60 km (including relays)	NO	< 65 seconds	20 nodes in two clusters separated by 2 relays.
Random	Slight problems with 50 nodes. Estimated relative offset <0.5 ms	< 10% rejected iterations	< 30 seconds	50 (50) Nodes

TABLE 7: PERFORMANCE OF THE DNS ALGORITHM

Table 7 show the simulated performance of the DNS algorithm in NBWF networks. The simulation results show that the algorithm manages to synchronize NBWF networks as long as $N_i=3$. The table also show that the DNS algorithm uses less than 65 seconds to converge any of these networks. Based on the simulation results, the following should be considered if the DNS algorithm is used in NBWF:

- The DNS algorithm must be used with low values for N_i to have sufficient performance for NBWF. Higher values of N_i lead to problems with both precision and convergence.
    - This influences the hybrid algorithm as a low value of N_i will cause more shifts between GNSS and mutual synchronization. While a high value cause severe problems with convergence and precision.
- All simulations are performed with all SF slots utilized and error free. This interval is short enough to ensure synchronization, but additional simulations are needed to find the maximum interval tolerated. This must also include the effect of errors in SF slots.

The simulation results also show that the DNS algorithm needs mechanisms for merge of complex networks even if the initial relative offsets are below 11.5 ms. The need for these additional network merge mechanisms is increased when compared to other algorithms because the DNS algorithm lacks the ability to synchronize the rates. This will cause the local clocks to diverge faster from each other when communication is lost. It is apparent that algorithms that synchronize rate will perform better and lead to fever network merge cases.

Originally the plan was to perform more simulations on the DNS algorithm in combination with the hybrid algorithm. Unfortunately, the implementation and simulation of the DNS algorithm uncovered several issues that raised questions about the use of the algorithm in NBWF. These issues do not fully disqualify the DNS algorithm as a candidate for NBWF, but they make it probable that there are better algorithms that could be used in combination with a hybrid algorithm. These issues are:

- Saarnisaari and Vanninen did not go into detail when they described the reason for selecting the DNS algorithm to be used in combination with the hybrid algorithm [32]. There is several other mutual synchronization algorithm that they could have used.
- The DNS algorithm can synchronize the offset only and leaves the clock rate unchanged. While a synchronization of the clock rate is not an absolute necessity, it would help maintain better accuracy and precision over periods without communication. This will lead to less use of network merge mechanisms. The ability to synchronize rate will

also lead to use of fewer beacons to maintain synchronization in an NBWF network.

- The effect of different sizes of the DNS parameter N_i is not documented in detail in the original work of Saarnisaari [31]. In addition, the findings in section 5.4.1 still leaves some uncertainties about the effect of this parameter.
- The original DNS algorithm is created without a timer that can be used to force synchronization in cases with long time difference between each synchronization point [31]. This can lead to long convergence time and in some cases unnecessary loss of synchronization. This is accounted for in some extent by the implementation of a timer in the hybrid algorithm, but it is still a weakness of the original DNS algorithm.
    - Other algorithms, such as KFMP [34] and the algorithm described by Tjoa et al. [35], use a timer or other mechanisms for this purpose.
- The simulations are performed with data in all SF slots. Networks with few utilized SF slots or heavy loss of slots make it even more difficult for the DNS algorithm to converge and reach a sufficient precision.

Some of these issues might be improved when the DNS algorithm is used in combination with the hybrid algorithm. Further simulations to find optimal values of the DNS parameters N_i, h and α might also improve precision and convergence. Even with these improvements it is believed that the DNS algorithm will need to utilize almost all SF slots to ensure that the algorithm manages to converge all NBWF networks. In addition to this, it is probable that the N_i value must be kept low even if the hybrid algorithm provides a timer. As a consequence of this, the next section investigates other algorithms that can replace the DNS algorithm.

5.4.6. *Simulation of the CS-MNS algorithm*

Three of the algorithms mentioned in section 3.4 are potential candidates to replace DNS in the lower layer of the hybrid algorithm. The algorithm described by Tjoa et al. and the KFMP algorithm are both mutual algorithms that could replace the DNS. In addition to this, the CS-MNS algorithm can be used if hybrid algorithm can tolerate periods of frequent shifts between GNSS based synchronization and mutual synchronization. Ability to synchronize rate is preferred because this can be used to increase the maximum allowed distance between active SF slots and help maintain synchronization for longer periods without communication. Both the KFMP algorithm and the CS-MNS algorithm have the ability to synchronize rate, but the CS-MNS algorithm is chosen for simulation because it is much simpler to implement. The CS-MNS algorithm is simulated for all topologies with a maximum initial relative offset of 1 ms, and the results are compared with similar networks utilizing the DNS algorithm.

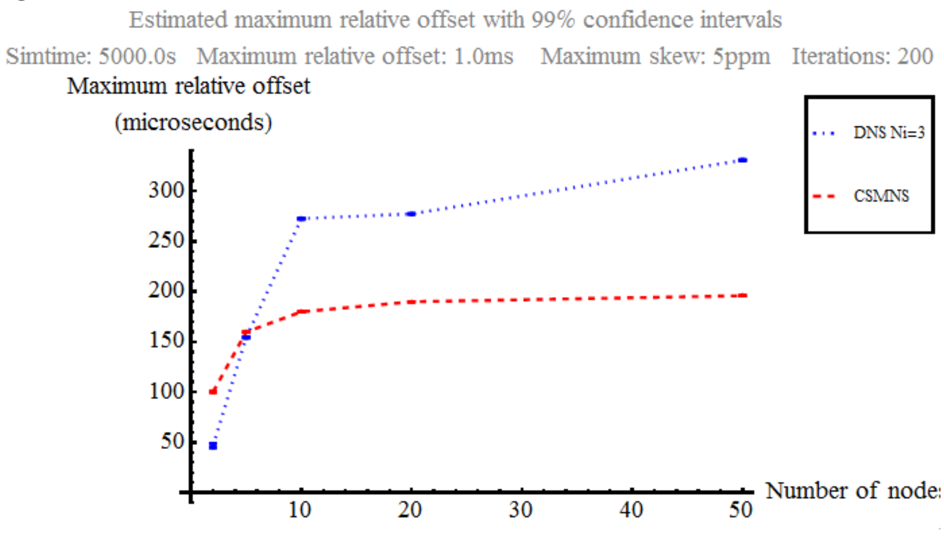


FIGURE 41: COMPARISON BETWEEN DNS AND CS-MNS ALGORITHM IN TOPOLOGY 1 NETWORKS.

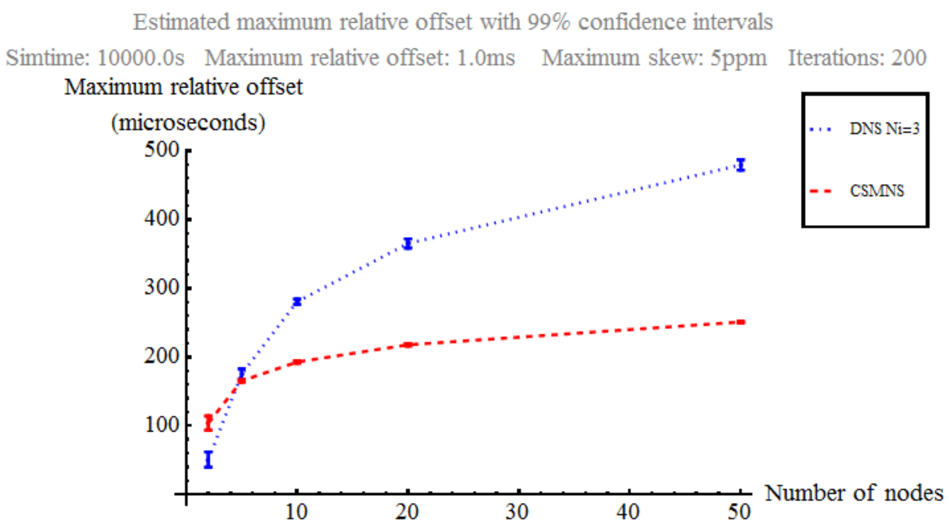


FIGURE 42: COMPARISON BETWEEN DNS AND CS-MNS ALGORITHM IN NETWORKS WITH RANDOM TOPOLOGIES

Figure 41 and Figure 42 show comparisons of the two algorithms for topology 1 networks and networks based on random topologies. Both figures show that the CS-MNS algorithm achieves better precision when the network size increases beyond 5 nodes.

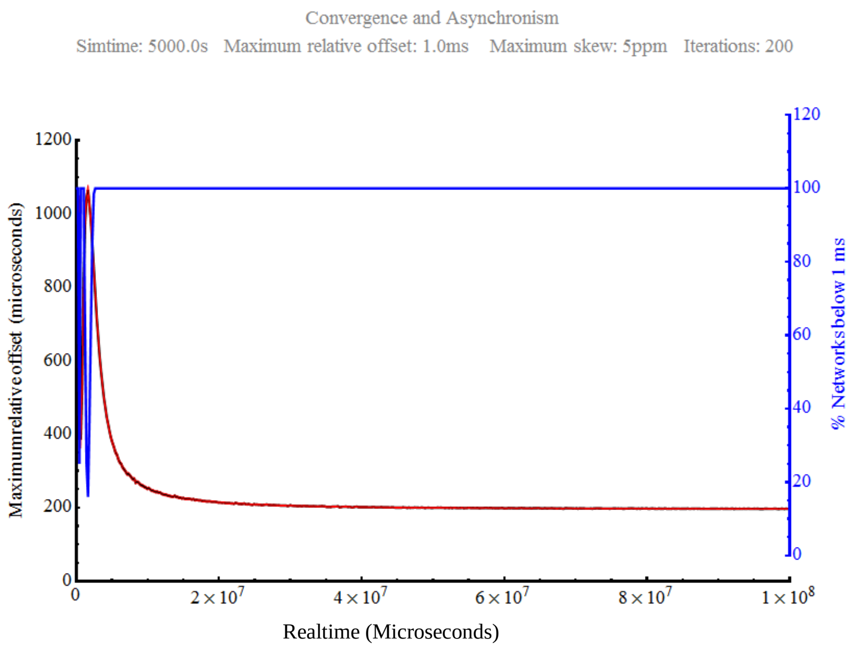


FIGURE 43: CONVERGENCE FOR NETWORKS WITH 50 NODES (CS-NMS TOPOLOGY 1)

Figure 43 shows that the CS-MNS algorithm manages to converge topology 1 network with 50 nodes to a stable value after about 30 seconds. For these networks the DNS algorithm (shown in Figure 26) manages faster convergence.

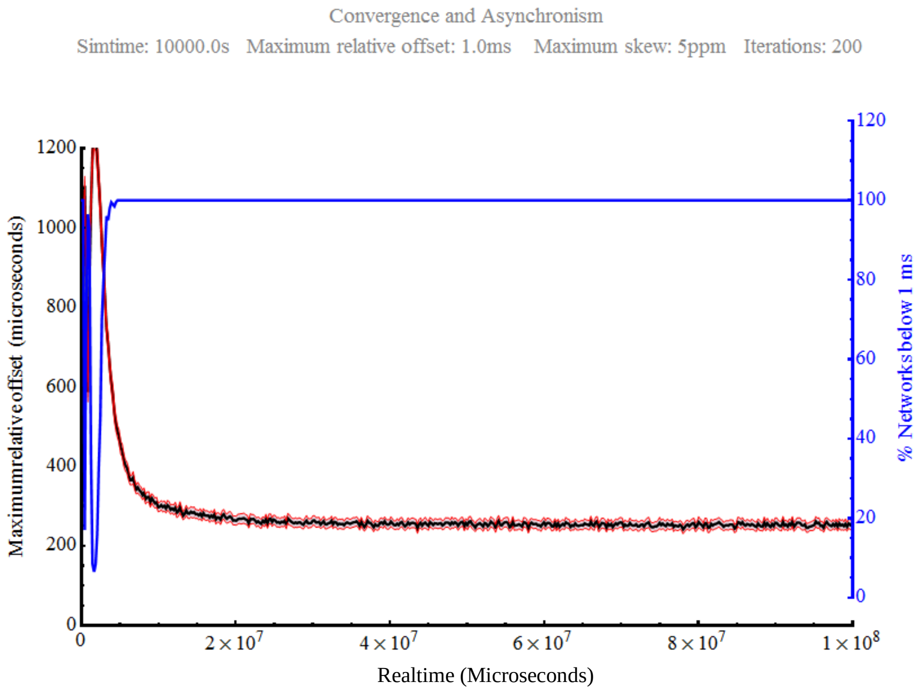


FIGURE 44: CONVERGENCE FOR NETWORKS WITH 50 NODES (CS-NMS RANDOM TOPOLOGY)

Figure 44 shows that CS-MNS manages to converge random topology networks with 50 nodes at about 30 seconds. The results in Figure 43 and Figure 44 show that CS-MNS the introduction of node pairs without connections only causes a slight increase in the convergence time. In addition to this, CS-MNS manages to maintain a precision better than 1 ms for all networks with random topology. The DNS algorithm has more problems with random topologies as it was unable to maintain a precision better than 1 ms for all random topology networks of 50 node (shown in Figure 38). This indicates that the CS-MNS algorithm are more robust to node pairs loosing connection.

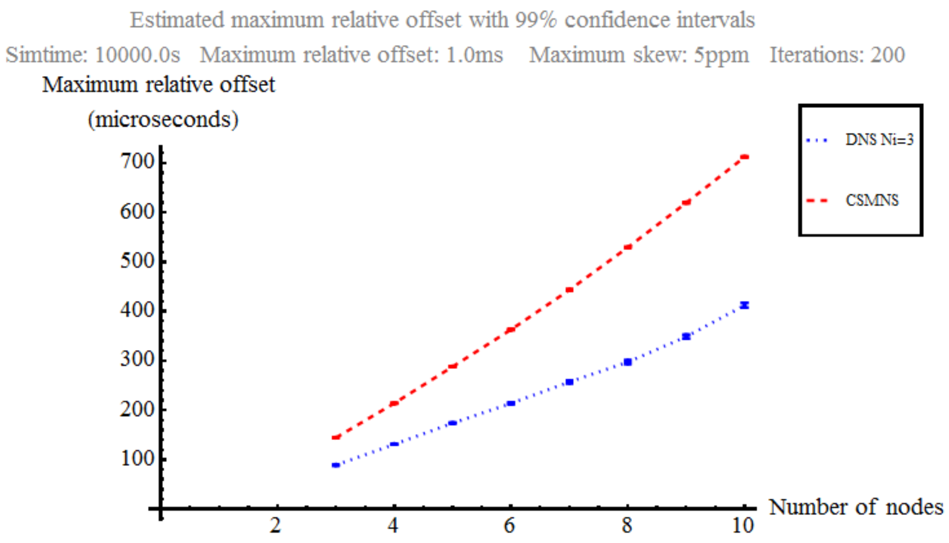


FIGURE 45: COMPARISON BETWEEN DNS AND CS-MNS ALGORITHM IN TOPOLOGY 2 NETWORKS

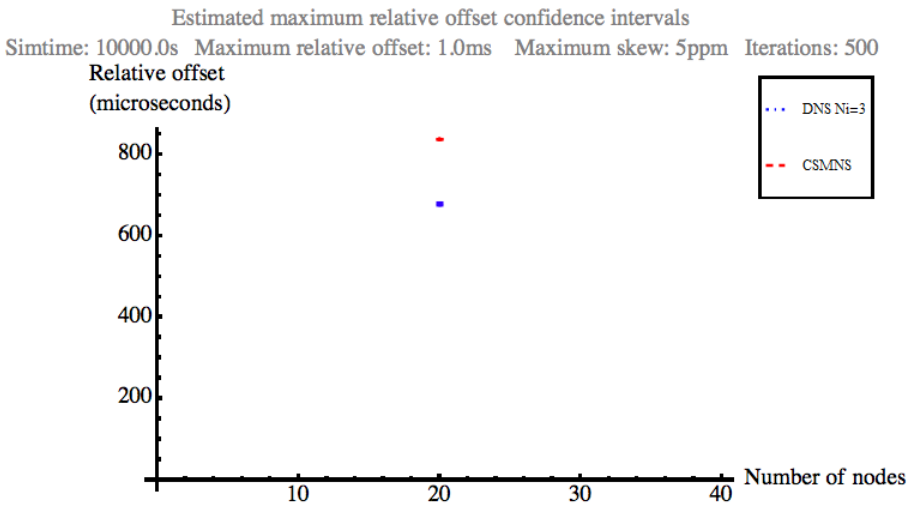


FIGURE 46: COMPARISON BETWEEN DNS AND CS-MNS ALGORITHM IN TOPOLOGY 3 NETWORKS.

Figure 45 and Figure 46 show comparisons of estimated maximum relative offset between DN and CS-MNS algorithm for topology 2 and topology 3 network. The results show that both algorithms achieves the required precision, but the DNS algorithm has a better precision in both these algorithms. The CS-MNS algorithm manages to converge topology 2 networks after about 30 seconds (not presented) this is slightly slower than the 20 seconds needed by the DNS algorithm. For topology 3(not presented), the CS-MNS algorithm and DNS algorithm have similar performance as both algorithms manages to converge the networks at about 60 seconds.

The simulation results are inconclusive However, there are additional factors that make it probable that CS-MNS is a better choice than DNS for NBWF networks:

- All simulations are performed with all SF slots utilized and error free. The result is short synchronization intervals, and the positive effect of rate synchronization becomes small. It is expected that the DNS algorithms ability to synchronize will be affected much more seriously than the CS-MNS algorithm when errors and long intervals between SF slots are introduced. This makes the CS-MNS algorithm more robust than the DNS algorithm.
- Additional simulations (not presented) with all SF slots enabled show that CS-MNS performs better than DNS, in all topologies, if a N_i value other than 3 is chosen.
- The CS-MNS is much simpler to implement, uses fewer parameters and is easier to control than the DNS algorithm. The CS-MNS algorithm does not show the same tendency to produce unexpected results as the DNS algorithm does with varying values of N_i.
- The DNS algorithm need only a single pair of synchronization point to synchronize. This should ensure faster convergence when errors are introduced to the network.

Based on this, CS-MNS should be preferred together with the hybrid algorithm for NBWF.

6. DISCUSSION

This thesis has surveyed a number of works related to sensor network and ad hoc networks to find potential synchronization algorithms for NBWF. The survey showed that most of the algorithms are unsuited for NBWF because there is no practical way to make them work without dedicated messaging. In addition to this, the requirement for a robust algorithm further limits the number of suitable algorithms.

All the potential algorithms found in the survey are based on one-way message exchange. These algorithms provides synchronization, but lacks the ability to compensate for propagation delays. The result of this is that the precision of the algorithms available for NBWF will be limited by the propagation delay. This is a result of choice to synchronize without dedicated synchronization messages and cannot be corrected by choosing another algorithm. Regardless which algorithm is chosen, NBWF will lack delay compensation.

Initially it was assumed that there existed a form of intelligent algorithms. The idea was that information such number of neighbors and data activity could be used to make an intelligent decision about nodes used for synchronization. The survey showed that mutual synchronization algorithms, such as CS-MNS, lacks this ability altogether. Other algorithms based on ad hoc structures, such as FTSP, used simple parameters such as node ID and network ID to select master nodes.

The mutual algorithm does not need this intelligence as they are designed to synchronize with the result from all nodes, but the lack of such ability is more of a surprise in algorithms using ad hoc structure. In networks with an ad hoc structure it would be preferable to use a master with a lot of neighbors instead of a master with few neighbors several hops away from the core of the network. One reason for the lack such algorithms might be that authors mainly have focused on new techniques to avoid collision between synchronization messages. Another reason might be that the ability to synchronize based on these choices add complexity to the algorithms. The work of Saarnisaari and Vanninen is the only suitable algorithm that uses this form of intelligent decision. However, the DNS algorithm used for synchronization is a simple mutual algorithm. Saarnisaari and Vanninen only use more complex decisions for functions such as network merge and late entry which is performed by a dedicated mechanism and not the DNS algorithm.

The survey shows that the hybrid algorithm suggested by Saarnisaari and Vanninen is the best choice for NBWF. This algorithm should be able to fulfill all NBWF requirements. In the hybrid algorithm, synchronization of networks without GNSS capable nodes are performed by the DNS algorithm. This algorithm is fundamental for NBWF because it ensures the robustness of the hybrid algorithm. Because of this, simulations were performed with the DNS algorithm to show how it performed in NBWF networks.

The simulation result showed that the DNS algorithm can be used to synchronize NBWF networks. The algorithm was capable of delivering sufficient precision and convergence as long as the DNS parameter N_i is set to a fixed low number. This parameter controls the number of messages received before the algorithm estimates a new global time and adjust the local clock. N_i is an important parameter when the DNS algorithm is used in the hybrid algorithm because the authors intended to use large N_i values to avoid unnecessary switches between GNSS based synchronization and mutual (DNS) synchronization.

Although Saarnisaari and Vanninen experienced long convergence time when they simulated the DNS algorithm together with the hybrid algorithm [32] they assumed that they could use a control channel to increase the number of synchronization messages. These messages could be used to help the DNS algorithm converge faster and enable larger values of N_i. In NBWF, it is not possible to increase the number of synchronization messages as the simulations were performed with all SF slots in use. The only way to ensure the DNS algorithms ability to synchronize is to allow frequent shifts between GNSS and mutual synchronization by allowing lower N_i values.

The fact that the DNS algorithm must use low N_i values together with other issues described in chapters 4 and 5 made it apparent that other algorithms might fit better with NBWF. While not absolutely critical, the ability to synchronize rate is beneficial to NBWF because this enables the network to survive longer periods without communication. Because of this, the algorithm replacing DNS should provide synchronization of both offset and rate and the CS-MNS algorithm was chosen.

The actual simulation results were inconclusive, but the CS-MNS algorithm is preferred over the DNS algorithm based on an overall assessment of the simulation results and the algorithms capabilities. For NBWF, it is recommended that the hybrid algorithms ability to switch automatically between GNSS based and mutual synchronization is combined with the CS-MNS algorithms ability to use the SF slots for effective synchronization. This should provide a foundation that could be used to create a robust and effective algorithm for NBWF. This combination is capable of delivering sufficient convergence time, precision and accuracy without using dedicated synchronization messages.

7. CONCLUSION

Synchronized local clocks are crucial to avoid collision of TDMA slots. Collision can be avoided by utilizing synchronization algorithm that can ensure precision better than 1 ms. In addition to this, NBWF requires that the synchronization algorithm must achieve this without using dedicated synchronization messages. A study of NBWF showed that correct reception of messages inside SF slots can be used to generate two synchronization points that can be used by the synchronization algorithm. The use of these messages makes it possible to utilize synchronization algorithms created for one-way message exchange to synchronize NBWF nodes without transmitting actual timestamps.

Potential algorithms were surveyed to find algorithms suited for NBWF. These algorithms were evaluated based on their ability to function in NBWF without the need for dedicated synchronization messages. In addition to this, abilities to support node mobility, multiple hops and use external reference time were key factors to find the best solution for NBWF. The hybrid algorithm described by Saarnisaari and Vanninen was selected for further study, based on the evaluation of potential algorithms.

The hybrid algorithm utilizes GNSS capable nodes for synchronization if available or mutual synchronization through the DNS algorithm if no GNSS capable nodes are available. Saarnisaari and Vanninens work indicated that GNSS based synchronization should work well with NBWF while the DNS algorithms ability to function in NBWF was uncertain. Simulations performed on the DNS algorithm show that it is possible to configure the DNS algorithm to deliver sufficient precision for NBWF networks. However, the simulation results together with an overall assessment of the DNS algorithm made it likely that other mutual algorithms are better suited for NBWF.

The CS-MNS algorithm was selected as a replacement for the DNS algorithm, and additional simulations were performed to check how the CS-MNS algorithm performed in NBWF networks. These final simulation results together with the overall assessment of the two algorithms made it clear that the combination of CS-MNS and the hybrid algorithm is the best basis for synchronization in NBWF networks. In addition to this, the simulations gave insight into the following NBWF issues:

- Maximum time between SF beacons: Synchronization is based on correct reception of messages in SF slots. The only guaranteed transmission inside

SF slots is a single beacon transmitted every superframe. This is enough to maintain synchronization.

- Accuracy of clocks: All simulations are performed with clocks that have a skew between -5 and 5 ppm. The simulations show that NBWF will work as long as the clocks have skews within this bound.
- Precision of network affiliation and coarse synchronization: Simulation results show that the algorithms manages to synchronize the networks as long as coarse synchronization has a precision of 1 ms or better.
- Convergence: The CS-NMS algorithm is capable of converging all simulated networks within 60 seconds.

7.1. Future Work

Because of the limited time available to this thesis, there are still several unfinished aspects around the combination of CS-MNS algorithm and the hybrid algorithm that should be studied further:

- The CS-MNS algorithm is only simulated separately without the hybrid algorithm. Further simulations should be performed where the combination of both algorithms are used. Important factor to consider for these simulations are:
    - Performance of GNSS based synchronization in NBWF.
    - The overall effect on precision caused by frequent changes between GNSS based synchronization and mutual synchronization.
    - The effect of errors and lost SF slots.
    - The amount of SF beacons (active SF slots) required to provide sufficient convergence.
    - The effect of topology changes.
- A study of the effect of including MV, DU and GU in the synchronization. Use of these slots could potentially speed up convergence.

REFERENCES

[1] S. Haavik, "Initial link layer protocol design for NBWF," Norwegian Defence Research Establishment (FFI), 2011.

[2] V. Jodalen, B. Solberg and S. Haavik, "NATO Narrowband Waveform (NBWF) - overview of link layer design," Norwegian Defence Research Establishment (FFI), 2011.

[3] H. Kopetz and W. Ochsenreiter, "Clock Synchronization in Distributed Real-Time Systems," *IEEE Transactions on Computers,* Vols. C-36, no. 8, pp. 933-940, 1987.

[4] B. Sundararaman, U. Buy and A. D. Kshemkalyani, "Clock Synchronization for Wireless Sensor Networks: A survey," *Adhoc Networks (Elsiver),* vol. 3, no. 3, pp. 281-383, 2005.

[5] Symmetricom, *Quantum SA.45s CSAC Datasheet*, Symmetricom, 2012.

[6] M. Ryu and S. Hong, "Revisiting clock synchronization problems: Static and dynamic constraint transformation for correct timing enforcement," School of Electrical Engineering, Seoul National University, Seoul, 1998.

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

[8] D. L. Mills, "Modelling and Analysis of Computer Network Clocks," Electrical Engineering Department, University of Delaware, 1992.

[9] K. Arvind, "Probabilistic Clock Synchronization in Distributed Systems," *IEEE Transactions on Parallel and Distributed Systems,* vol. 5, no. 5, pp. 474-487, 1994.

[10] F. Sivrikaya and B. Yener, "Time Synchronization in Sensor Networks: A survey," *IEEE Network*, vol. 18, no. 4, pp. 45-50, 2004.

[11] C. Lenzen, P. Sommer and R. Wattenhofer, “Optimal clock synchronization in networks,” in *Proceedings of the 7th ACM Conference on Embedded Network Sensors*, 2009.

[12] R. Gotzhein and T. Kuhn, "Decentralized Tick Synchronization for Multi-Hop Medium Slotting in Wireless Ad Hoc Networks Using Black Bursts," in *Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2008.

[13] M. L. Sichitiu and C. Veerarittiphan, "Simple, Accurate Time Synchronization for Wireless Sensor Networks," in *EEE Wireless Communications and Networking Conference (WCNC 2003)*, 2003.

[14] P.-H. Huang, M. Desai, X. Qiu and B. Krishnamachari, "On the Multihop Performance of Synchronization Mechanisms in High Propagation Delay Networks," *IEEE Transactions on computers*, vol. 58, no. 5, pp. 577-590, 2009.

[15] J. Elson, L. Girod and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proceedings of the 5th USENIX Symp. Operating System Design and Implementation (OSDI 02)*, 2002.

[16] D. Mills, J. Martin, J. Burbank and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," June 2010. [Online]. Available: http://tools.ietf.org/html/rfc5905. [Accessed 20 April 2013].

[17] Network Time Synchronization Research Project, "NTP version 4 release note," 3 October 2011. [Online]. Available: http://www.eecis.udel.edu/~mills/ntp/html/release.html. [Accessed 20 April 2013].

[18] D. L. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis," 1992. [Online]. Available: http://tools.ietf.org/html/rfc1305. [Accessed 20 April 2012].

[19] IEEE, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, IEEE, 1997.

[20] M. Mock, E. Nett, R. Frings and S. Trikaliotis, "Continuous clock synchronization in wireless real-time applications," in *Proceedings 19th IEEE Symposium on Reliable Distributed Systems (SRDS-00)*, 2000.

[21] Z. Wen, U. Heo and J. Choi, "A Novel Synchronization Algorithm for IEEE802.11 TDMA Ad Hoc Network," in *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*, 2007.

[22] H. K. Pande, S. Thapliyal and L. C. Mangal, "A new Clock Synchronization Algorithm for Multi-Hop Wireless Ad Hoc Networks," in *6th international conference on Wireless communication and sensor networks*, 2010.

[23] J.-P. Sheu, C.-M. Chao and C.-W. Sun, "A Clock Synchronization Algorithm for Multi-Hop Wireless Ad Hoc Networks," in *24th International Conference on Distributed Computing Systems, 2004. Proceedings.*, 2004.

[24] L. Huang and T.-H. Lai, "On the scalability of IEEE 802.11 Ad Hoc networks," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2002.

[25] T.-H. Lai and D. Zhou, "Efficient and scalable IEEE 802.11 Ad-Hoc-Mode Timing Synchronization Function," in *Proceedings of the 17th International conference on advanced information networking and applications*, 2003.

[26] C. H. Rentel and T. Kunz, "A Mutual Network Synchronization Method for Wireless Ad Hoc and Sensor Networks," *IEEE*

Transactions on mobile computing, vol. 7, no. 5, pp. 633-646, 2008.

[27] D. Zhou and T.-H. Lai, "An Accurate and Scalable Clock Synchronization Protocol for IEEE 802.11-Based Multihop Ad Hoc Networks," *IEEE transactions on parallel and distributed systems*, vol. 18, no. 12, pp. 1797-1808, 2007.

[28] P. Rauschert, A. Honarbacht and A. Kummert, "Synchronization of multihop ad hoc networks using connected dominating sets," in *IEEE International Symposium on Circuits and Systems*, 2006.

[29] J. Y. Lee, A. Verma, H. S. Lee and J. S. Ma, "A distributed time synchronization algorithm robust to traffic load for MANETs," in *VTC-2005-Fall. 2005 IEEE 62nd Vehicular Technology Conference*, 2005.

[30] S.-M. Chen, S.-P. Kuo and Y.-C. Tseng, "A Quorum-based Mechanism as an Enhancement to Clock Synchronization Protocols for IEEE 802.11 MANETs," *IEEE Communication Letters*, vol. 11, no. 4, pp. 313-315, 2007.

[31] H. Saarnisaari, "Analysis of a Discrete Network Synchronization Algorithm," in *MILCOM 2005 - 2005 IEEE Military Communications Conference*, 2005.

[32] H. Saarnisaari and T. Vanninen, "Hybrid Network Synchronization for MANETs," in *Communications and Information Systems Conference (MCC), 2012 Military*, Gdansk, 2012.

[33] M. Maróti, B. Kusy, G. Simon and Á. Lédeczi, "The Flooding Time Synchronization Protocol," in *Proc. of ACM SenSys 04*, 2004.

[34] Zeng, Yi, B. Hu and S. Liu, "Vector Kalman Filter Using Multiple Parents for Time Synchronization in Multihop Sensor Networks," in *Proceedings 5th Annual IEEE Communications Society*

Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '08.), 2008.

[35] R. Tjoa, K. Chee, P. Sivaprasad, S. Rao and J. Lim, "Clock drift reduction for relative time slot TDMA-based sensor networks," in *2004 IEEE 15th International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE Cat. No.04TH8754)*, 2004.

[36] F. Wang, P. Zeng and P. Zeng, "Slot Time Synchronization for TDMA-Based Ad Hoc Networks," in *2008 International Symposium on Computer Science and Computational Technology*, 2008.

[37] D. Djenouri, "R4Syn: Relative Referenceless Receiver / Receiver Time Synchronization in Wireless Sensor Networks," *IEEE Signal Processing Letters,* vol. 19, no. 4, p. 175.178, 2012.

[38] G. Cao and J. L. Welch, "Accurate multihop clock synchronization in mobile ad hoc networks," in *Proceedings of the 2004 International Conference on Parallel Processing Workshops (ICCPW04)*, 2004.

[39] B. Wehbi, A. Laouiti and A. Cavalli, "Efficient time synchronization mechanism for wireless multi hop networks," in *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*, 2008.

[40] D. Djenouri, "Estimators for RBS-Based Time Synchronization in Heterogeneous Wireless Networks," in *6Th IEEE International Workshop on Heterogeneous, Multi-Hop, Wireless and Mobile Networks*, 2011.

[41] R. Solis, V. S. Borkar and P. R. Kumar, "A New Distributed Time Synchronization Protocol for Multihop Wireless Networks," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, 2006.

[42] X. Chen and C. Li, "A Routing Based Time Synchronization Protocol for Multi-Hop Wireless Networks," in *2010 IEEE International Conference on Communications,* 2010.

[43] S. Ganeriwal, R. Kumar and M. B. Srivastava, "Timing-Sync protocol for sensor networks," in *Proceedings from First ACM conference Embedded Network Sensor System (SenSys)*, 2003.

[44] B. J. Choi and X. (. Shen, "Distributed Clock Synchronization in Delay Tolerant Networks," in *2010 IEEE International Conference on Communications (ICC)*, 2010.

[45] M. Sasabe and T. Takine, "A Simple Scheme for Relative Time Synchronization in Delay Tolerant MANETs," in *International Conference on Intelligent Networking and Collaborative Systems*, 2009.

[46] K.-L. Noh and E. Serpedin, "ADAPTIVE MULTI-HOP TIMING SYNCHRONIZATION FOR WIRELESS SENSOR NETWORKS (INVITED PAPER)," in *9th International symposium of signal processing and its application*, 2007.

[47] T. Vanninen, M. Raustia, H. Saarnisaari and J. Iinatti, "Frequency hopping mobile ad hoc and sensor network synchronization," in *MILCOM 2008 - 2008 IEEE Military Communications Conference*, 2008.

[48] H. Toumivaara, *REAL-TIME IMPLEMENTATION OF A DISTRIBUTED TDMA BASED MAC PROTOCOL,* Oulu: University of Oulu, Department of Electrical and Information Engineering. Master’s Thesis, 2009.

[49] A. C. Davies, "Discrete-time synchronization of digital data networks," *IEEE Transactions on Circuits and Systems,* vol. C, no. 7, pp. 610-618, 1975.

APPENDICES

A. NBWF REFERENCE MODEL

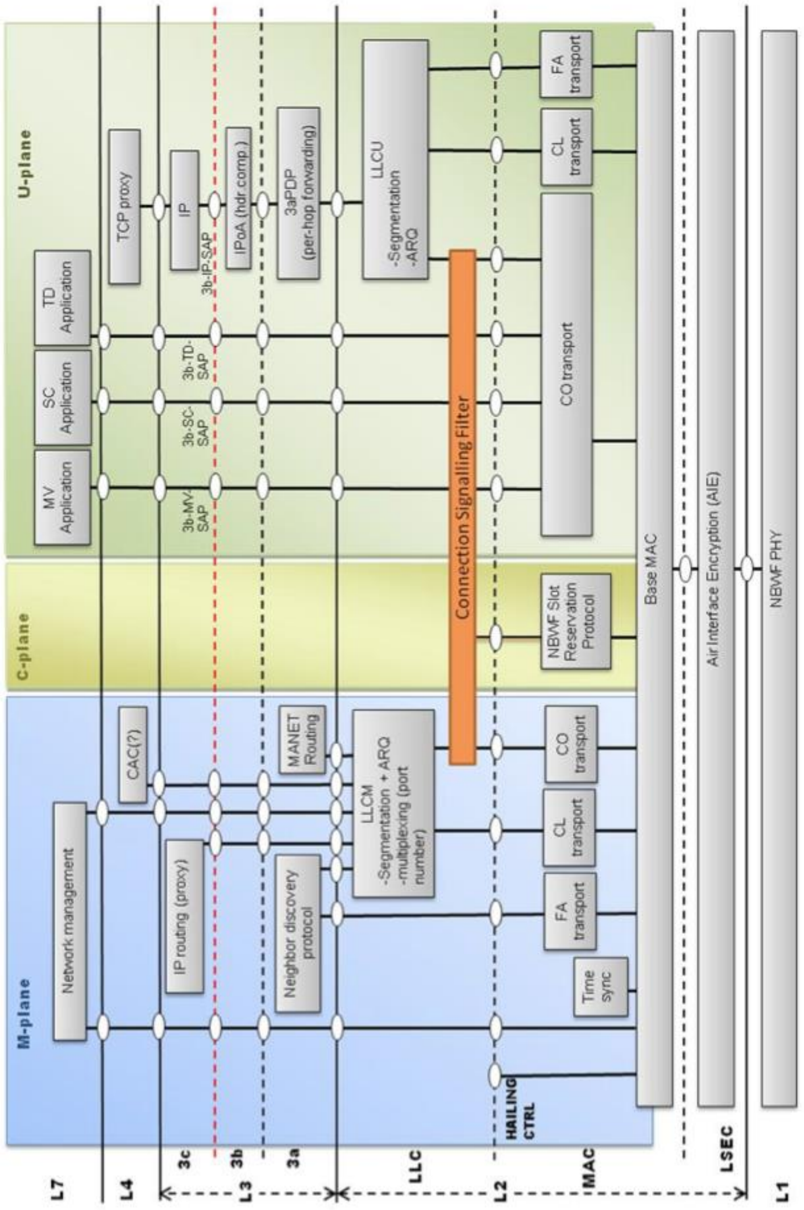


FIGURE 47: NBWF REFERENCE MODEL

B. Result of simulator validation

Mathematically it can be shown that the values of the different synchronization points P# can be expressed by a1, a2 and t_k, when t_k and β are known. This gives the following equations, solved in Mathematica:

$$Slope\ of\ N1:\ a_1 = 1 - \beta \tag{12}$$

$$Slope\ of\ n2:\ a_2 = 1 + \beta \tag{13}$$

$$Have\ x_1(\ t_{snd1}^{N1}):\ \boldsymbol{x_1} = \frac{\boldsymbol{t_k}}{\boldsymbol{a_1}} \tag{14}$$

$$Have\ a_2 = \frac{2t_k - t_k}{x_2 - x_1} \tag{15}$$

$$Solved\ with\ respect\ to\ x_2(t_{snd1}^{N2}):\ \boldsymbol{x_2} = \frac{\boldsymbol{(a_1 + a_2)t_k}}{\boldsymbol{a_2 * a_1}} \tag{16}$$

$$Have\ a_1 = \frac{3t_k - 2t_k}{x_3 - x_2} \tag{17}$$

$$Solved\ with\ respect\ to\ x_3(t_{snd2}^{N1}):\ \boldsymbol{x_3} = \frac{\boldsymbol{(a_1 + 2a_2)t_k}}{\boldsymbol{a_2 * a_1}} \tag{18}$$

$$Have\ a_2 = \frac{4t_k - 3t_k}{x_4 - x_3} \tag{19}$$

$$Solved\ with\ respect\ to\ x_4\ (t_{snd2}^{N2}):\ \boldsymbol{x_4} = \frac{\boldsymbol{2(a_1 + a_2)t_k}}{\boldsymbol{a_2 * a_1}} \tag{20}$$

$$Have\ a_1 = \frac{y_5 - t_k}{x_2 - x_1} \tag{21}$$

$$Solved\ with\ respect\ to\ y_5:\ \boldsymbol{y_5} = \frac{\boldsymbol{(a_1 + a_2)t_k}}{\boldsymbol{a_2}} \tag{22}$$

$$Have\ a_1 = \frac{y_6 - 3t_k}{x_4 - x_3} \tag{23}$$

$$Solved\ with\ respect\ to\ y_6:\ \boldsymbol{y_6} = \frac{\boldsymbol{(a_1 + 3a_2)t_k}}{\boldsymbol{a_2}} \tag{24}$$

$$Have\ y_7 = a_2 * x_1 \tag{25}$$

$$Have\ a_2 = \frac{y_8 - t_k}{x_3 - x_1} \tag{26}$$

$$Solved\ with\ respect\ to\ y_8\text{:}\ \boldsymbol{y_8} = \frac{\boldsymbol{(2a_1 + a_2)t_k}}{\boldsymbol{a_1}} \tag{27}$$

This gives the following points:

$$P1 = \{x_1, t_k\} \tag{28}$$

$$P2 = \{x_2, 2t_k\} \tag{29}$$

$$P3 = \{x_3, 3t_k\} \tag{30}$$

$$P4 = \{x_4, 4t_k\} \tag{31}$$

$$P5 = \{x_2, y_5\} \tag{32}$$

$$P6 = \{x_4, y_6\} \tag{33}$$

$$P7 = \{x_1, y_7\} \tag{34}$$

$$P8 = \{x_3, y_8\} \tag{35}$$

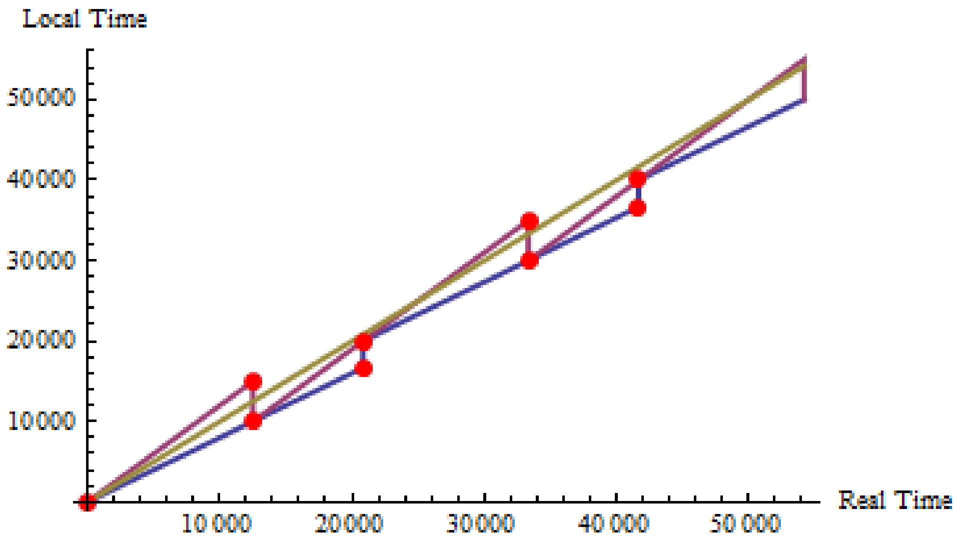


FIGURE 48: SIMULATED AND CALCULATED DATA

Figure 48 shows simulated data represented by the two lines and the points P0 to P8 represented by the red dots. The points were calculated in Mathematica with values β=0.2 and t_k =10000. The same values were also used in the simulations. It becomes apparent that the synchronization points are equal to the simulated data because the points and the graphs come together. This is expected and also shown in Figure 13.

The straight line through synchronization points, of a specific node, can be described by finding the slopes. The equation for the slopes are given below:

$$\textit{Slope for N1 synch points, P1 to P3: } a_3 = \frac{3t_k - t_k}{x_3 - x_1} \qquad (36)$$

$$\textit{Slope for N2 synch points, P2 to P4: } a_4 = \frac{4t_k - 2t_k}{x_4 - x_2} \qquad (37)$$

The equations for the straight lines through P0 and synchronization points are:

$$\textit{Line through sync points of N1 : } L_{N1} = a_3 * x_1 - t_k \qquad (38)$$

$$\textit{Line through sync points of N2: } L_{N2} = a_4 * x_2 - 2t_k \qquad (39)$$

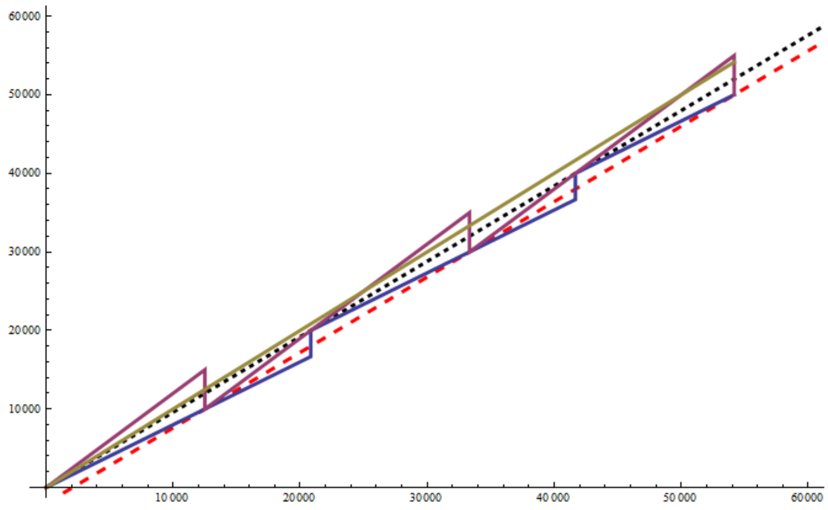


FIGURE 49: THE CONNECTION BETWEEN SYNCHRONIZATION POINTS

Figure 49 shows the same simulated data as in Figure 48 and the straight lines through the synchronization points. This confirms the linear connection between these points. These lines also shows that the clocks of both nodes will slowly drift away from real time.

It can be proven that the relative offsets at each synchronization point, for an individual node, are constant values. The equations for the relative offsets are:

$$\Delta_{y_1} = y_7 - t_k \tag{40}$$

$$\Delta_{y_2} = 2t_{k-} y_5 \tag{41}$$

$$\Delta_{y_1} = y_8 - 3t_k \tag{42}$$

$$\Delta_{y_1} = 4t_k - y_6 \tag{43}$$

The following equations, solved in Mathematica, shows that the relative offsets of the synchronization points for each node are equal:

$$\frac{\Delta_{y_3}}{\Delta_{y_1}} = \frac{\Delta_{y_4}}{\Delta_{y_2}} = 1 \tag{44}$$

Simulated data confirms this and shows that the relative offset at each synchronization point is constant for each node.