Clock Synchronization: Combining IEEE 1588 and Adaptive Oscillator Correction Method

by

Waheed Ahmed, B. Eng.

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs

in partial fulfillment of the requirements for the degree of

Masters of Applied Science in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering (OCIECE)

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada, K1S 5B6

January 2012

The undersigned recommend to
the Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Clock Synchronization: Combining IEEE 1588 and Adaptive Oscillator Correction Method

submitted by

Waheed Ahmed, B. Eng.

in partial fulfillment of the requirements for

the degree of Master of Applied Science in Electrical and Computer Engineering

_____
Chair, Dr. Howard Schwartz, Department of Systems and Computer Engineering

_____
Thesis Supervisor, Dr. Thomas Kunz

_____
Thesis Supervisor, Dr. Howard Schwartz

Carleton University
January 2012

Abstract

The thesis proposes to combine the IEEE 1588 protocol and the adaptive oscillator correction method (AOCM) to improve master-slave synchronization. This is done by extending the AOCM to a slave clock to train a model using the IEEE 1588 synchronization updates from a master clock. The model corrections are applied to the slave clock during the waiting period in between the synchronization updates and when the slave-master network experiences an outage. The NS-2 results indicate that the slave accuracy improves up to 10 times for no traffic networks depending on the slave clock training period.

In traffic scenarios, the slave accuracy is affected by the asymmetric delays such that the degree of accuracy is one half of the asymmetric latencies. During network congestion, the slave accuracy improves because the delays are less asymmetric. During an outage in a traffic scenario, the solution only improves the slave clock stability.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Thomas Kunz for guiding me throughout my masters program and supporting me in every step of the thesis work. His wide knowledge and constructive feedback helped me a great deal.

I would also like to express my deep gratitude to my co-supervisor, Prof. Howard Schwartz for his support and invaluable feedback during the thesis research period.

Finally I would like to thank my family and friends for their great support and encouragement throughout this period.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AOCM	Adaptive Oscillator Correction Method
CS-MNS	Clock Sampling Mutual Network Synchronization
CTE	Cumulative Time Error
DAC	Digital to Analog Converter
DOCXO	Double Oven Controlled Crystal Oscillator
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineers
IRIG	Inter-Range Instrumentation Group
ITU	International Telecommunication Union
NS	Network Simulator
NTP	Network Time Protocol
OCXO	Oven Controlled Crystal Oscillator
PPB	Parts Per Billion
PPM	Parts Per Million
PPS	Pulse Per Second
PTP	Precision Time Protocol
RLS	Recursive Least Squares
RMS	Root Mean Square
RPEM	Recursive Prediction Error Method
RTT	Round Trip Time

1 Chapter: Introduction

1.1 Overview

Almost every organization heavily relies on computers these days, all of which rely on clocks. Clocks are well-known to drift as they are usually made from inexpensive oscillator circuits or battery backed quartz crystals, resulting in drifts of seconds per day. The important question is what happens to these computers and the processes running in these networks if their clocks do not agree with each other or with the correct time? The short answer is that bad things start to happen. If the network clocks are not synchronized, processes could fail, data could be lost, security could be compromised, legal implications could be faced and most importantly the organizations lose credibility with customers and their business partners [2]. It is also important to realize that the solutions to network synchronization issues are usually easy to implement, inexpensive and highly effective, for example a time server costing $5000 can support a network of thousands of computers. [1] [2]

The objective of the thesis is to study different methods that can be used to synchronize clocks in a network, to understand the problems causing bad synchronization in a network and propose a solution to improve the synchronization accuracy. This also involves understanding why the clocks drift, what are the factors affecting the accuracy and the stability of the clock oscillators and what can be done to counter those factors.

1.2 Outline

The chapters in this thesis are organized in the following manner.

Chapter 2 provides background information about the elements affecting the clock accuracy and stability. It also talks about the network factors affecting the clock synchronization and how different protocols are commonly used to improve clock accuracy.

Chapter 3 provides a description of the IEEE 1588 synchronization protocol and reviews the state-of-the-art literature related to the performance of IEEE 1588 clock synchronization and the adaptive oscillator correction method to counter the temperature and ageing effects of the clock oscillator.

Chapter 4 presents the proposed solution to the clock synchronization problem by combining the IEEE 1588 protocol and the adaptive oscillator correction method (AOCM). It also provides details of the clock agent and algorithm implemented in NS-2 simulator, and presents the NS-2 simulation results of temperature and ageing effects on clock accuracy.

Chapter 5 presents the simulation results using the proposed solution from Chapter 4. The test cases are derived from the ITU standard document [20].

Chapter 6 concludes the thesis and provides directions for future work.

1.3 Contributions

The contributions of this thesis include the following:

- A solution to the clock synchronization problem is proposed. The idea is to combine the IEEE 1588 protocol and the adaptive oscillator correction method. The purpose of the adaptive oscillator correction method [19] is to train a clock locked to a GPS signal and apply that training model to correct the clock in holdover mode (i.e. when

the GPS signal is lost), hence improving the oscillator accuracy and stability in holdover mode. The first step of the thesis is to apply the corrected oscillator method to a simulated master clock which has a GPS signal reference. The next step extends the adaptive oscillator correction method to the slave clock, using the IEEE 1588 synchronization updates it receives from its master clock.

- A clock agent is implemented in the NS-2 simulator to simulate a real clock. Environmental changes are also implemented such as changes in temperature and ageing affects of the clock. These changes cause the clocks to drift.
- The proposed solution to combine the 1588 protocol and the adaptive oscillator correction method is implemented in NS-2.
- NS-2 test cases are implemented based on the ITU document [20] covering various network conditions and network loads. The test cases are run to simulate the proposed solution and the results are analyzed.
- The NS-2 results indicate that the proposed solution improves the slave accuracy up to 10 times for no traffic networks depending on the length of the slave clock training period. The solution results in traffic networks indicate that the slave accuracy is affected by the asymmetric delays such that the degree of accuracy is one half of the asymmetric latencies. In case of network congestion, the slave accuracy improves because the delays are less asymmetric. In a traffic scenario, when there is an outage, the solution only improves the slave clock stability.
- The proposed solution can be enhanced by accounting for the asymmetric latencies by using IEEE 1588 transparent clocks and by employing the hardware time stamping for the synchronization updates.

2 Chapter: Background Information

In this chapter, first an introduction to crystal oscillators is provided. In the second section, the factors affecting the stability and accuracy of a clock oscillator are presented; most importantly the effect of temperature and ageing factors is reviewed. The third section of this chapter reviews the elements affecting the clock synchronization in a network and different protocols that can be used to improve clock accuracy, signifying the importance of the IEEE 1588 synchronization protocol in terms of its accuracy.

2.1 Crystal Oscillators

The time-keeping element in the most consumer devices such as wristwatches, clocks, radios, computers, cell phones etc. is the underlying crystal oscillator. A crystal oscillator is an electronic oscillator circuit which creates an electrical signal with a very precise frequency by using the mechanical resonance of a vibrating crystal of piezoelectric material. The frequency is used to keep track of time. Commonly used piezoelectric resonators are the quartz crystal and hence the oscillators using them are termed as crystal oscillators. [18]

2.2 Factors affecting oscillator stability and accuracy

A clock accuracy and stability is directly related to the frequency accuracy and stability of the underlying crystal oscillator. The frequency accuracy of an oscillator is a measure of offset from a specified target. The factors establishing the frequency accuracy

of an oscillator include the temperature, ageing and retrace[1]. The frequency stability of a crystal oscillator is determined over a period of time by measuring the oscillator frequency variations from its operational frequency. The factors affecting the frequency stability of an oscillator include the reference signal noise (if the oscillator is locked to a reference signal such as GPS), oscillator tuning port noise, supply rail noise and the crystal vibration [18]. Figure 1 presents accuracy and the stability examples for a frequency source. [3]

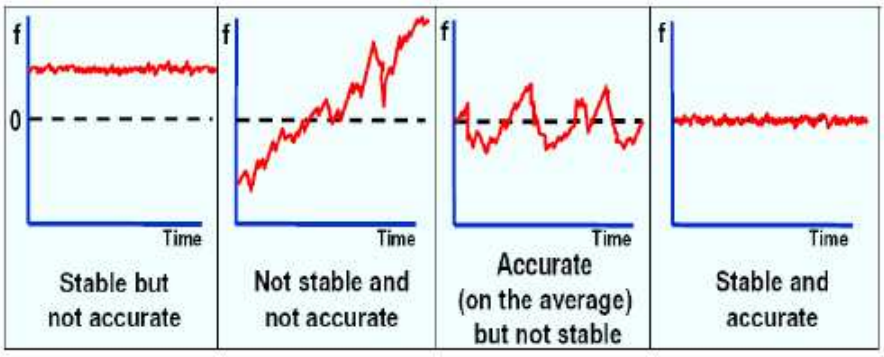

Figure 1: Accuracy and Stability of a Frequency Source. [3]

The most important factors affecting the accuracy of oscillators are temperature and ageing and these factors are reviewed next.

2.2.1 Temperature

Temperature plays a major role in the frequency accuracy of a clock. The frequency temperature dependency is directly related to the crystal cut of the oscillator.

[1] "When power is removed from an oscillator for several hours, then re-applied on it again, the frequency of this oscillator will stabilize at a slightly different value. This frequency variation error is called retrace error." [18]

The frequency-temperature characteristic using the AT-cut crystal cut method is shown in Figure 2. Here φ is the cut angle and with varying cut angles, the frequency-temperature dependency varies. It is observed that in general the crystal cuts show a cubic dependency on temperature. [4]

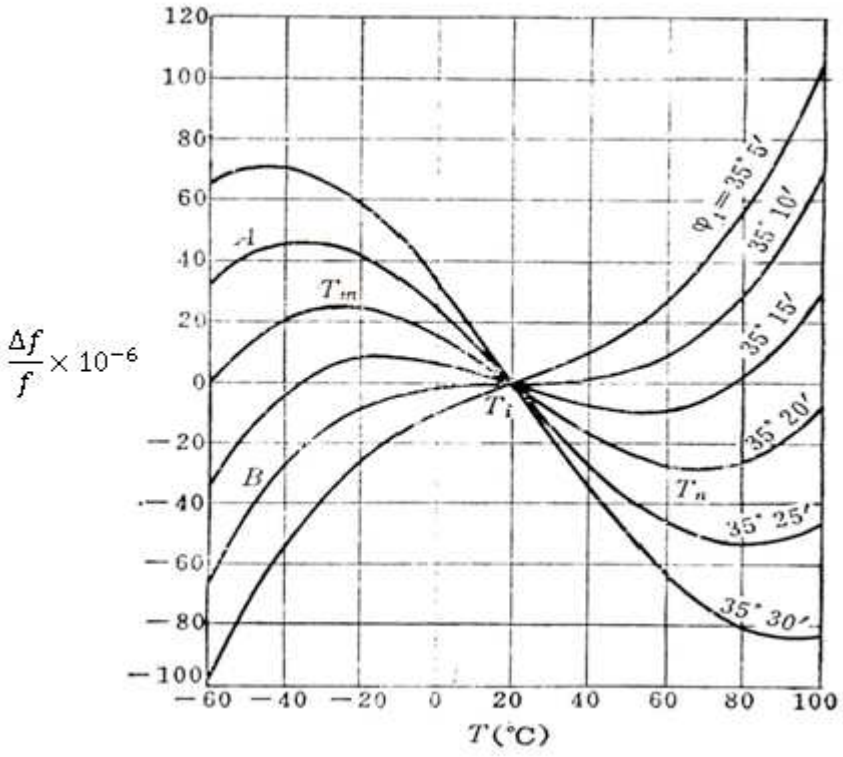

Figure 2: Frequency-Temperature Dependency using AT-cut. [4]

2.2.2 Ageing

The ageing of a crystal oscillator is the process where the crystal frequency changes over time. Generally the ageing effect over time is not linear, however when the ageing effect is observed over a short period of time, it can be considered as linear as shown in Figure 3.

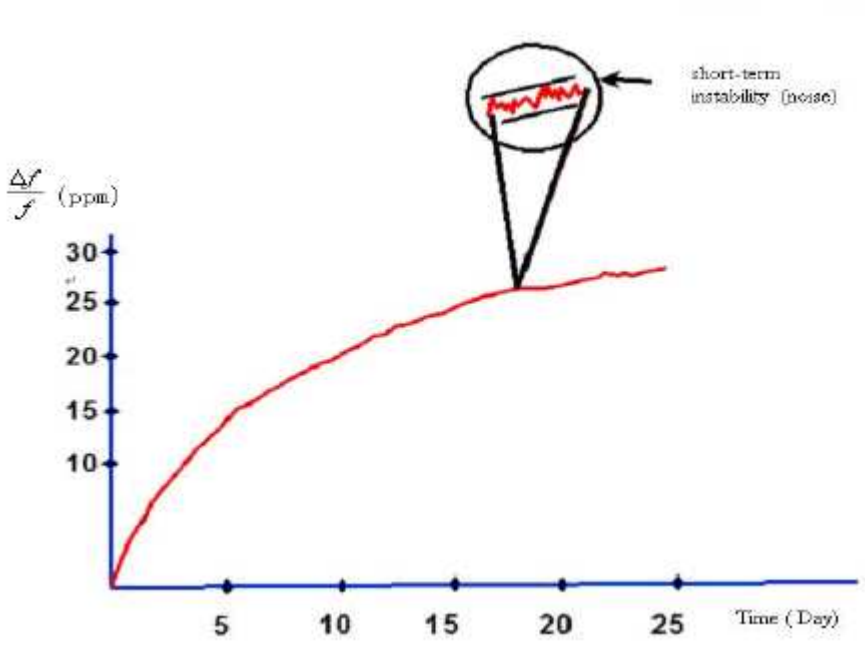

Figure 3: Ageing Effect of an Oscillator. [3]

2.2.3 Other Factors

Some other factors affecting the frequency accuracy of a clock are Retrace, Thermal Hysteresis, Frequency Pushing and Pulling etc. Similarly some other elements affecting the frequency stability are Tuning port noise, Reference signal noise, Power supply noise etc. [5] [6]

It should be noted that among all the factors, the temperature and ageing are the most important and dominant factors which affect the accuracy of oscillators, e.g. from Figure 2 for a temperature range of -60ºC to 100°C, the frequency drifts are -100 ppm to 100 ppm, whereas from Figure 3, for a 25-day period, the frequency drifts are up to 30 ppm for the given oscillator. On the other hand, the Retrace factor can for example cause frequency drifts of only 10 ppb in a 2-week period. [18]

Also the temperature or the ageing effect may induce more frequency errors than the other depending on the thermal environment. If the thermal environment is stable, the aging will dominate the frequency stability of the oscillator. On the other hand, if the thermal environment is undergoing variations in a time period which is shorter than the time required for the oscillator to drift significantly with respect to the aging rate then the temperature effect on the frequency stability of the oscillator will dominate. [18]

2.3 Network Synchronization

Network synchronization means that one tries to set the time on two or more clocks in a network to be the same. The hidden problem lies in the term "setting" the time, for example the length of time it takes to "set" the time. There is latency associated in processing of time packets by the operating system. There is also network latency created by the hubs, routers, switches, cables and other hardware existing between clocks.

The most commonly used network protocols for the clock synchronization are NTP (Network Time Protocol) and IRIG (Inter-Range Instrumentation Group) time codes. In an NTP-based LAN network, the network devices add latency and jitter that reduce the clock accuracy to 1 to 2 milliseconds [7]. For an NTP-based WAN network, the accuracy drops to 1 to 20 milliseconds [7]. However, these are far from guaranteed results when taking into consideration the fact that the switches and routers used on LANs and WANs are adding further latency and the fact that many NTP clients run on non-real-time Operating Systems which further decreases the synchronization accuracy. For example, on Windows, if the system is busy performing high priority tasks, the clock accuracy could be reduced to 10 to 50 milliseconds. On the other hand, the IRIG B

protocol can improve accuracy using dedicated coaxial cables that are used to carry time stamp information between IRIG B clocks. The use of dedicated cables and hardware clocks in the IRIG time code provides increased accuracy of 1 to 10 milliseconds compared to NTP's accuracy of 10 to 50 milliseconds for a typical NTP client running on Windows OS [7]. The more recently developed protocol is the IEEE 1588 Precision Time Protocol (PTP). The IEEE 1588 PTP overcomes the latency and jitter issues through hardware time stamping at the physical layer of the network resulting in 20 to 100 nanoseconds clock accuracy. The IEEE 1588 PTP is cost effective as it uses existing Ethernet LANs. The key factor in achieving the higher accuracy in IEEE 1588 is the use of boundary or transparent clock and hardware assisted time stamping. The hardware time stamping reduces the operating system latency while the deployment of boundary or transparent clocks reduces the network latency. Figure 4 shows a comparison of NTP, IRIG time code and IEE 1588. [7] [8]

Protocol	Sync Accuracy	Interconnect	Required Clock H/W & S/W
NTP	1-10 milliseconds	Ethernet LAN or WAN	H/W or S/W server, S/W clients
IRIG	1-10 microseconds	Coaxial Cable	H/W master and slaves
IEEE 1588	20-100 nanoseconds	Ethernet LAN	H/W master and slaves

Figure 4: Comparison of the Synchronization Requirements of NTP, IRIG time code and IEEE 1588. [7]

3 Chapter: Related Work

In this chapter, the first section presents the details of the IEEE 1588 synchronization protocol. The second section of the chapter reviews the state-of-the-art related to the performance of IEEE 1588 clock synchronization. The third section reviews the Adaptive Oscillator Correction Method used to counter the temperature and ageing effects of the clock oscillator.

3.1 IEEE 1588 Precision Time Protocol

In this section, the IEEE 1588 Precision Time Protocol (PTP) is briefly discussed. The basic message exchange mechanism between a master clock and a slave clock is presented that is used to do the time stamps and the offset calculations. The concept of boundary and transparent clocks is also mentioned.

3.1.1 Introduction

The IEEE 1588 PTP standard [9] specifies a clock synchronization protocol to synchronize real-time clocks in the nodes of a distributed system. A node within a distributed system can be modeled as containing a real-time clock that may be used by applications within the node for various purposes such as generating timestamps for data or ordering events managed by the node. The protocol provides a method for synchronizing the clocks of participating nodes to a high degree of accuracy and precision. The PTP standard [9] discusses in detail the Precision Time Protocol and the node, system, and communication properties necessary to support PTP.

3.1.2 PTP System

A PTP system is a distributed, networked system which consists of a combination of PTP and non-PTP devices. PTP devices include ordinary clocks, boundary clocks, end-to-end transparent clocks, peer-to-peer transparent clocks, and management nodes, some of which are briefly discussed later in this section. Non-PTP devices include bridges, routers, and other infrastructure devices including application devices such as computers and printers. Figure 5 shows a packet switched network supporting the IEEE 1588 protocol, showing a master clock (locked to a GPS signal), slave clocks and transparent clocks.

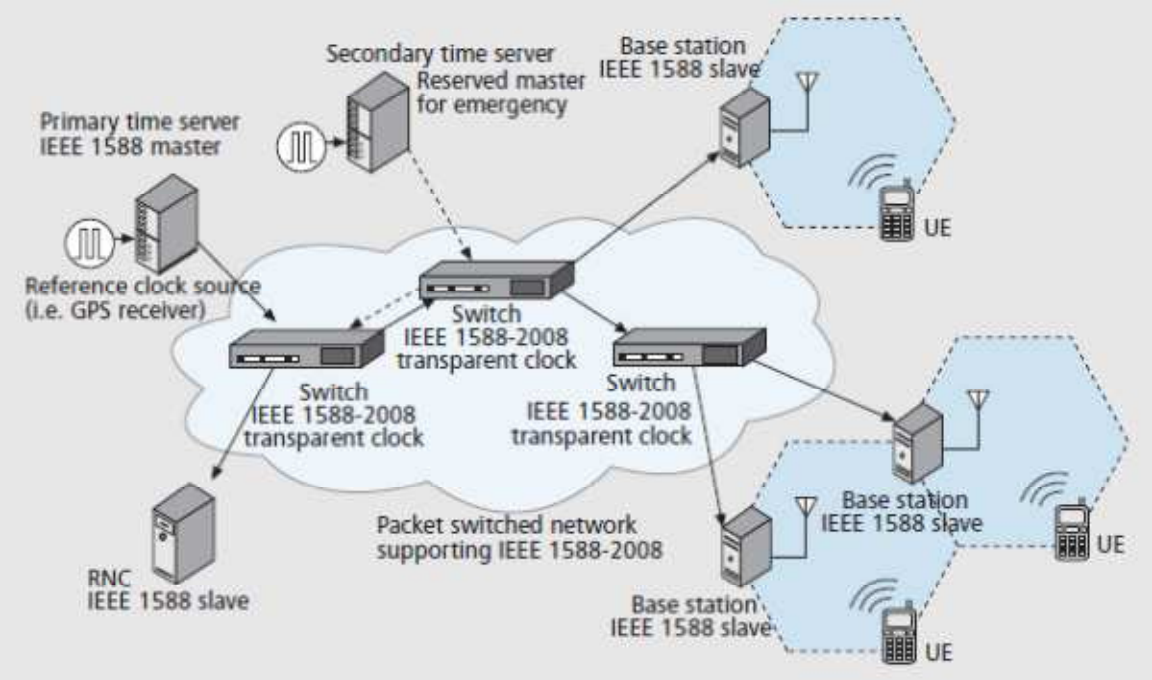

Figure 5: Clock synchronization of wireless base stations. [15]

The clocks in a distributed system are structured into a master−slave synchronization hierarchy where the top clock in the hierarchy determines the reference

time for the entire system. This clock is referred to as the grand master clock. PTP timing messages are exchanged between the clocks to achieve synchronization. Here the slaves use the timing information to adjust their clocks to the time of their master in the hierarchy. [9]

The standard [9] defines event and general PTP messages. Event messages are timed messages as an accurate timestamp is generated at both transmission and receipt. General messages on the other hand, do not require accurate timestamps. The Sync, Delay_Req, Follow_Up, and Delay_Resp messages are used to generate and communicate the timing information needed to synchronize ordinary and boundary clocks using the delay request-response mechanism. These message types and various PTP devices are described in the next sub-sections.

3.1.2.1 PTP Message Types

Sync - A Sync message is an event message that is transmitted by a master to its slaves. It either contains the time of its transmission or is followed by a Follow_Up message containing this time. It may be used by a receiving node to measure the packet transmission delay from the master to the slave.

Follow_Up - The Follow_up is a general message. In a two-step ordinary or boundary clock, the Follow_Up message communicates the time of transmission of a particular Sync message

Delay_Req - A Delay_Req message is an event message. It is a request for the receiving node to return the time at which the Delay_Req message was received. This request is responded to by using a Delay_Resp message.

Delay_Resp - The Delay_Resp message is a general message that communicates the time of receipt of a Delay_Req message.

3.1.2.2 PTP device types

The following types of PTP devices are discussed in the standard [9].

3.1.2.2.1 Ordinary clock

An ordinary clock is a clock that has a single Precision Time Protocol (PTP) port[2] in a domain and maintains the timescale used in the domain. It may act as a source of time (a master clock), or may synchronize to another clock (a slave clock).

3.1.2.2.2 Boundary clock

A boundary clock has multiple Precision Time Protocol (PTP) ports in a domain and maintains the timescale used in the domain. It may serve as the source of time (a master clock), and may synchronize to another clock (a slave clock).

3.1.2.2.3 Transparent clock

A transparent clock is a device that measures the time taken for a Precision Time Protocol (PTP) event message to transit the device and provides this information to clocks receiving this PTP event message.

End-to-end transparent clock

An end-to-end transparent clock is a transparent clock that supports the use of the end-to-end delay measurement mechanism between slave clocks and the master clock.

---

[2] Precision Time Protocol (PTP) port is a logical access point of a clock for PTP communications to the communications network. [9]

The end-to-end transparent clock forwards all messages just as a normal bridge, router, or repeater. The time the message takes to traverse the transparent clock, called resident time, is measured by the residence time bridge. These residence times are accumulated in a special field, the correctionField, of the PTP event message or the associated Follow-Up message. This correction is based on the difference in the timestamp generated when the event message enters and leaves the transparent clock.

Peer-to-peer transparent clock

A peer-to-peer transparent clock is a transparent clock that provides corrections for the propagation delay of the link connected to the port receiving the PTP event message.

3.1.3 Delay Request-Response Mechanism

Delay Request-Response Mechanism is the basic synchronization mechanism in the IEEE 1588 PTP standard to synchronize clocks in a master-slave hierarchy. It uses the PTP timing messages (Sync, Delay_Req, Delay_Resp, and if required, Follow_Up) on the communication path linking the two clocks. Figure 6 shows the basic pattern of synchronization message exchange to synchronize a slave clock to a master clock.

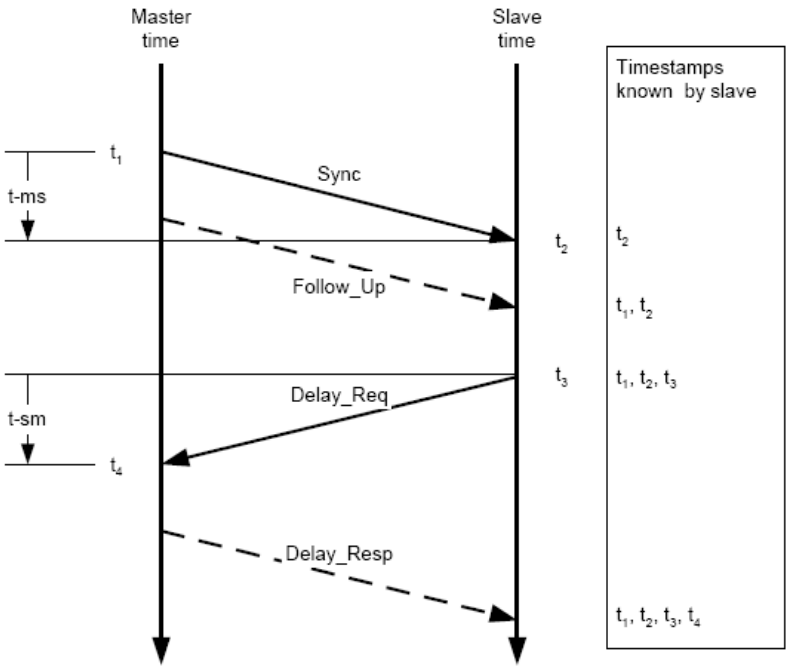

Figure 6: Basic synchronization message exchange in Delay Request-Response Mechanism. [9]

The message exchange pattern between master and slave clocks is as follows:

- The master sends a Sync message to the slave and notes the time of its transmission t_1.
- The slave receives the Sync message and notes the time of reception t_2.
- The master conveys to the slave the timestamp t_1 by:
    - Embedding the timestamp t_1 in the Sync message. This requires some sort of hardware processing if highest accuracy and precision are desired.
    - Embedding the timestamp t_1 in a Follow_Up message.
- The slave sends a Delay_Req message to the master and notes the time of its transmission t_3.

- The master receives the Delay_Req message and notes the time of reception t_4.
- The master conveys to the slave the timestamp t_4 by embedding it in a Delay_Resp message.

After the exchange of these messages, the slave possesses all four timestamps t_1, t_2, t_3 and t_4. These timestamps can be used to calculate the offset of the slave's clock with respect to the master and the mean propagation time of messages between the two clocks, which is the mean of t-ms and t-sm in Figure 6. The mean propagation time and offset computations shown below are given in the IEEE 1588 standard [9] Section 11.2 and 11.3.

$$Mean\ Propagation\ Time = \frac{[(t_2 - t_1) + (t_4 - t_3)]}{2} = \frac{[(t_2 - t_3) + (t_4 - t_1)]}{2}$$

$$Clock\ Offset =$$

$$t_2 - t_1 - Mean\ Propagation\ Time - correctionField\ of\ Sync\ message$$

(no Follow_Up message)

$$Clock\ Offset = \frac{[(t_2 - t_1) - (t_4 - t_3)]}{2}$$

The standard [9] does not specify how to apply the clock offset correction to the slave clock. However in this thesis, the clock offset correction is directly applied to the time stamp of the slave clock i.e.

$$Slave's\ New\ Time\ Stamp = Slave's\ Old\ Time\ Stamp - Clock\ Offset$$

The computation of offset and propagation time assumes that the master-to-slave and slave-to-master propagation times are equal. Any asymmetry in propagation time

introduces an error in the calculated value of the clock offset. The computed mean propagation time differs from the actual propagation times due to the asymmetry. [9]

The above calculations also assume that the slave and master frequency are exactly the same, or nearly exactly the same. If this assumption is not made then there will be a set of simultaneous equations solving for the slave frequency and offset using the two "exact" timestamps from the master, and two "erroneous" timestamps from the slave. The simultaneous equations can typically only be solved by assuming delay symmetry, but other methods are also possible. [11]

3.2 Performance of IEEE 1588 Protocol

This section provides a survey of research literature related to the performance of IEEE 1588 clock synchronization.

3.2.1 FPGA based Ethernet Switch

The author of [12] evaluates an FPGA-based Ethernet switch implementation of IEEE 1588 clock synchronization and presents the performance results. An Ethernet switch bridges multiple networking segments and allows precision time synchronization between all nodes required to support IEEE 1588. A switch hardware shown in Figure 7 implements a complete 8-port 10/100 Mbps Ethernet PHY device and consists of four dual-port 10/100 Mbps Ethernet PHY devices and a single FPGA.

Figure 7: Switch Hardware. [12]

A simple master device which can produce proper sync and follow-up messages and can respond to delay-request messages to a slave has been implemented using standard FPGA prototyping hardware. Test signals like a 1pps (1 pulse per second) signal and 1 KHz clock waveform are implemented using a physical port. An oscilloscope is fed with master and slave signals to visualize verification of proper synchronization and jitter performance. In the experiment, the frequency of 125 MHz for both slave and master devices is used. Hence both master and slave devices introduce a jitter of 8ns. Overall a minimum of 16ns jitter and offset deviation can be expected from the experiment setup. The experiment is performed using different sync frame transmit intervals. The values are recorded when they become stable which is approximately after 15 sync messages. Table 1 presents the measured performances in terms of offset and jitter.

Table 1: Performance Results using FPGA based Ethernet Switch. [12]

Sync Interval (s)	Offset Min (ns)	Offset Max (ns)	Offset Average (ns)	Jitter (ns)
3	-408	+504	0.9	912
2	-192	+324	-0.2	516
1	-92	+88	-0.9	180

From Table 1, it is obvious that the slave synchronizes with the master clock within 1ns on average. Also the sync interval clearly has an effect on the results. The smaller the sync interval, the smaller is the jitter. [12]

3.2.2 IEEE 1588-2008 Adapters

The authors of [13] present a system with IEEE 1588-2008 adapters that allow clock synchronization on the order of sub-microsecond using existing Gigabit Ethernet equipment. The IEEE 1588-2008 adapter measures the residence times, and runs the peer delay mechanism of the IEEE 1588 standard [9] to compensate for the time error caused by the queuing delays in the Gigabit Ethernet equipment. The authors use ordinary Gigabit Ethernet equipment to operate as a Peer-to-Peer Transparent Clock (P2P TC). A P2P TC is shown in Figure 8.

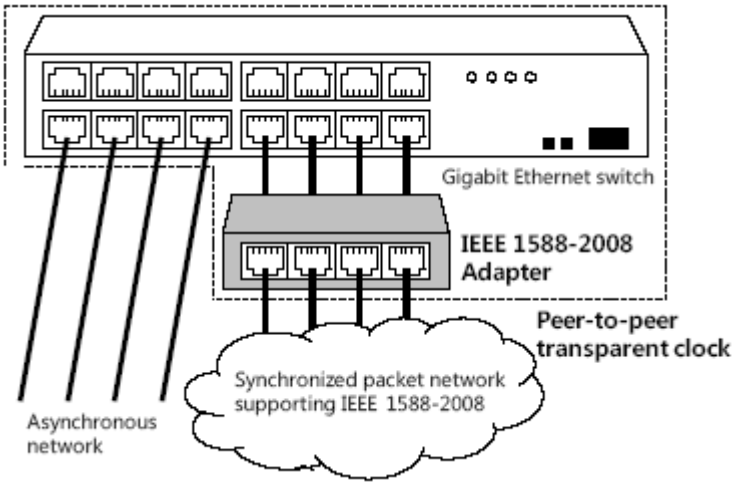

Figure 8: A peer-to-peer transparent clock using an IEEE 1588-2008 adapter over an ordinary Gb Ethernet Switch. [13]

The authors present two schemes of clock synchronization, one using ordinary switches and the delay request-response mechanism and the other using peer-to-peer transparent clocks (P2P TCs) and the peer delay mechanism. To evaluate their schemes, the authors create a test setup in which four computers are connected to a simple network using two ordinary gigabit Ethernet switches as shown in Figure 9.

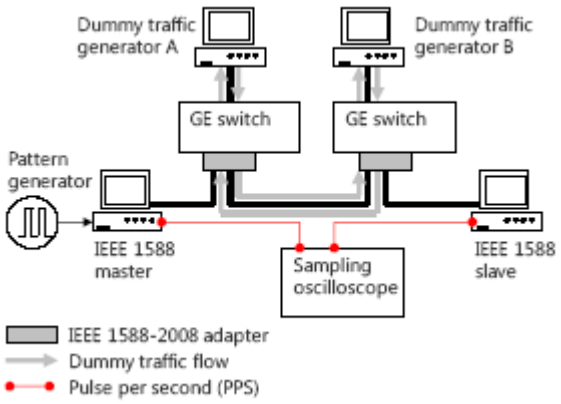

Figure 9: Test setup to measure synchronization accuracy. [13]

The leftmost computer generates the SYNC messages as an IEEE 1588 master device. The rightmost computer, which acts as an IEEE 1588 slave device receives the SYNC messages and estimates time and frequency of the master. The middle two computers generate dummy traffic to interrupt message exchanges between master and slave devices. Both master and slave devices are connected to high-speed digital sampling oscilloscope to measure the time error between master and slave devices. The results obtained using the proposed IEEE 1588-2008 adapters and using ordinary clocks are shown in Figure 10 and Figure 11 respectively.

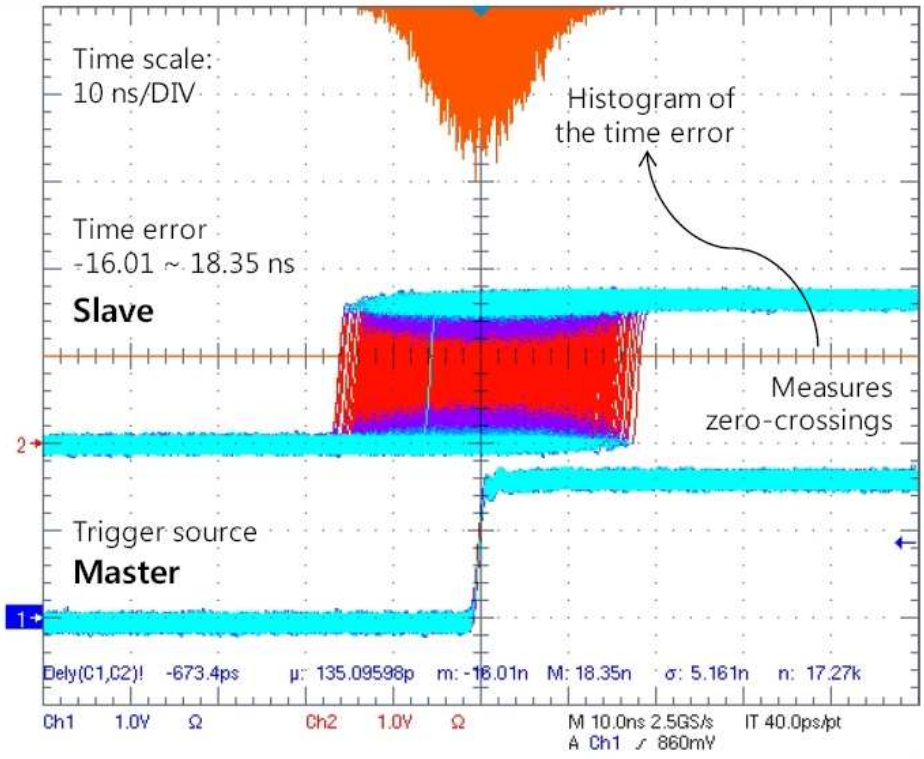

Figure 10: Oscilloscope screenshot measuring the time error of the slave using the IEEE 1588-2008 adapters. [13]

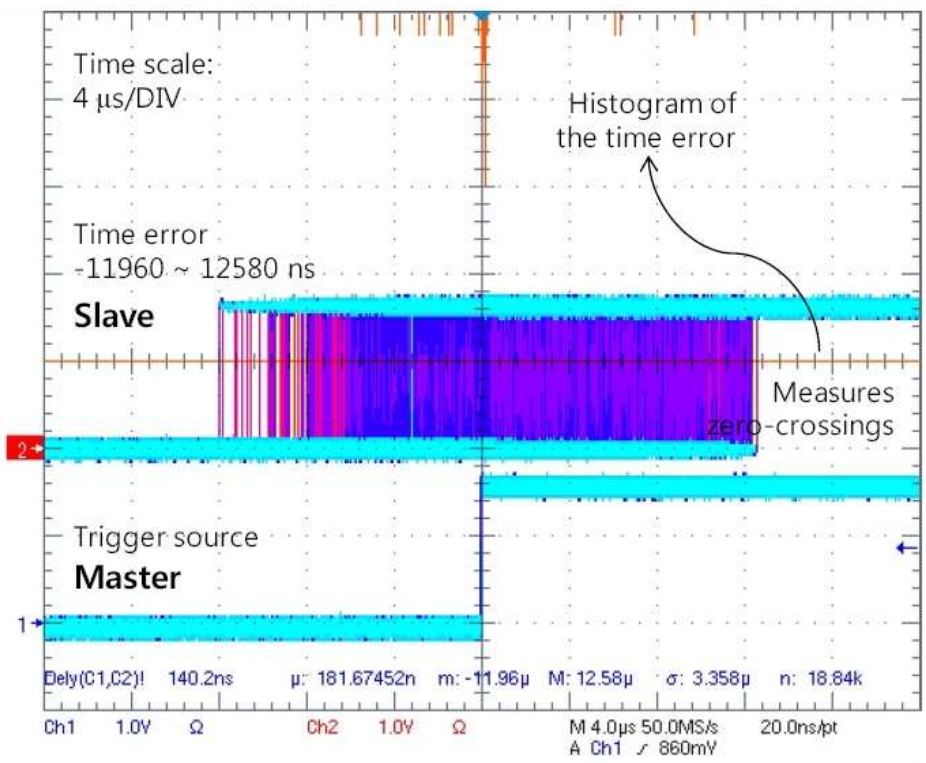

Figure 11: Oscilloscope screenshot measuring the time error of the slave using only the ordinary clocks. [13]

The results are collected using the master PPS signal as a trigger source and capturing the slave PPS signal for 10 minutes. Figure 10 indicates that the rising edges of the slave signal are distributed over the interval from -16.01 ns to 18.35 ns, whereas in Figure 11, when ordinary clocks are used, the slave signal is distributed over the interval from -11960 ns to 12580 ns. The large time error in Figure 11 is caused by queuing delays in the switches. With IEEE 1588-2008 adapters these errors are corrected and hence we have a synchronized network. Hence an accuracy of ±20ns overall can be obtained using the proposed IEEE 1588 adapters as P2P transparent clocks. [13]

3.2.3 Hardware assisted Time Stamping

The authors [14] from Zurich University of Applied Sciences (ZHW) have used hardware assisted time stamping to evaluate IEEE 1588 clock synchronization. Their test setup consists of ZHW's IEEE 1588 evaluation kit which is capable of delivering two time stamps per transmission/reception of Precision Time Protocol (PTP) event messages at the same time. The first time stamp is recorded at the Medium Independent Interface (MII) between MAC and PHY chips. The second time stamp is taken at the entry point of the network interface driver's interrupt service routine. The ZHW's IEEE 1588 evaluation kit, as shown in Figure 12, consists of an embedded PC with a 300 MHz Geode CPU and the network interface on the main board. A time analyzer board consisting of a TSU (time stamping unit) and an adjustable clock is connected to the PC/104 bus. The clock can be monitored and compared with other clocks by the one pulse per second (1PPS) output.

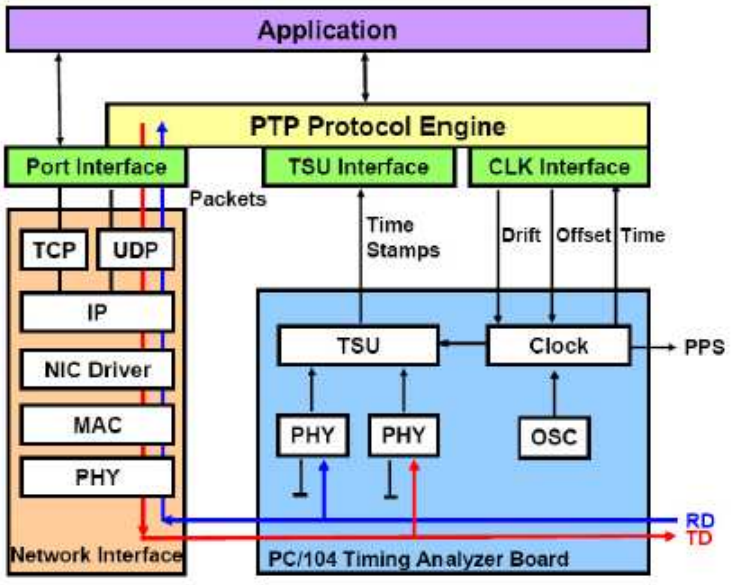

Figure 12: The IEEE 1588 evaluation kit. [14]

The authors employ the hardware assisted time stamping technique using two setups. The experiment conditions are listed below:

- Sync interval = 2s
- Clocks used are equipped with cheap quartz crystal oscillators, Jauch VX3MH-5000, 50 MHz, ±50 ppm and they drift from each other approximately by 9μ/s when running free.
- Temperature is constant at room temperature.
- Hub used for first setup: Addtron (100Base-Tx Class2)
- Switch used for second setup: Allied Telesyn FS708
- High Resolution Timing Analyzer (HRTA) is used to compare the 1PPS outputs of master and slave devices. Transitions of input lines are time stamped with a resolution of 20 ns.

The first setup consists of a master and a slave clock connected via hub. The histogram in Figure 13 shows the offset of the slave with respect to the master clock determined by comparing the PPS signals of the two nodes under stable conditions. The variation lies within ±80ns. The observed variation is the result of hub jitter, the transceiver's jitter and the time stamp's resolution of HRTA.

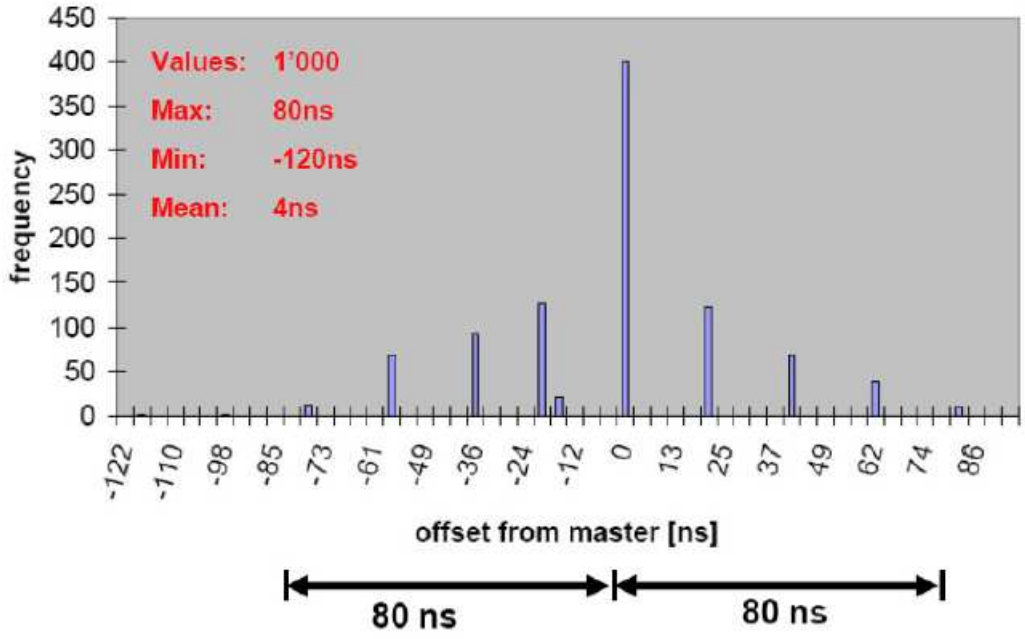

Figure 13: Synchronization accuracy using hub. [14]

In the second setup, the hub is replaced by a switch without any load. Figure 14 shows the synchronization accuracy which varies beyond ±500ns.

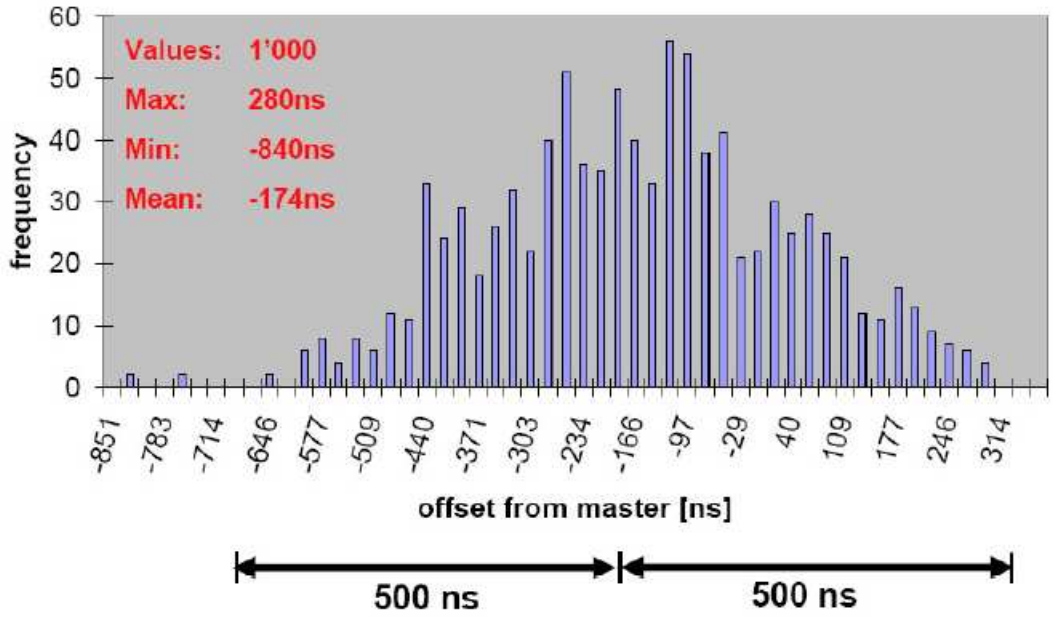

Figure 14: Synchronization accuracy using switch without any load. [14]

Next TCP data stream is added at about 20% of the available bandwidth in the path of SYNC messages while in the path of Delay_Req messages, the TCP stream consumes 2% of the available bandwidth. This results in a small queuing delay and in low path asymmetry. The synchronization accuracy is in the range of ±400µs due to delayed SYNC messages as shown in Figure 15

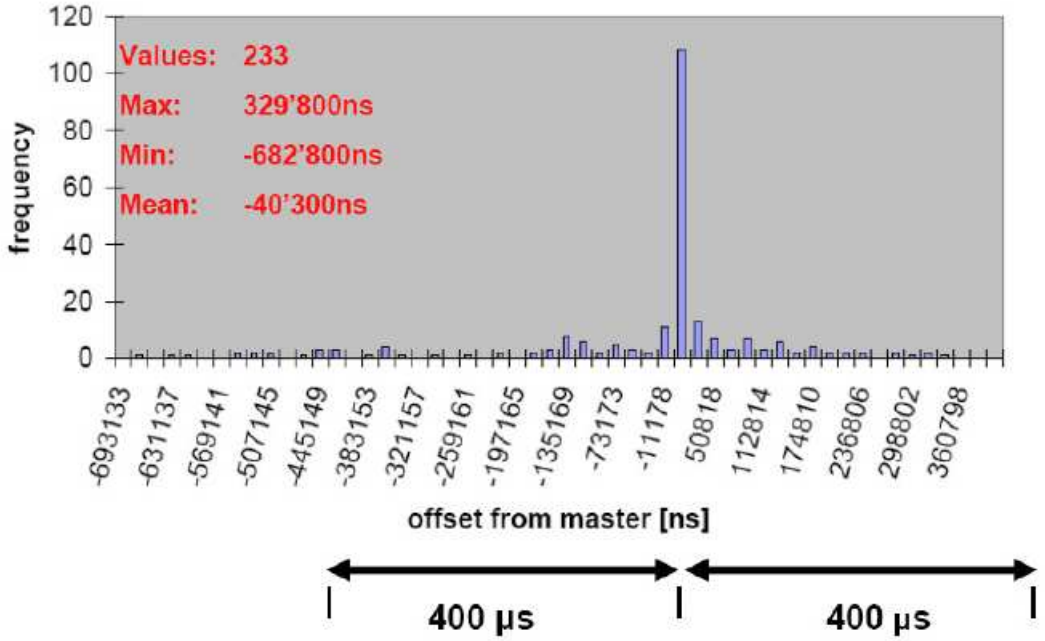

Figure 15: Synchronization accuracy using switch with asymmetric load. [14]

Filter algorithms could be used to sort out extremely delay messages under stable conditions. [14]

3.2.4 Using Gigabit Ethernet Switch and Combined Ordinary Peer-to-peer Transparent Clock to support IEEE 1588

The authors of [15] discuss practical considerations to be taken in the design and implementation of time synchronization systems using IEEE 1588. The authors use a

model based on a Gigabit Ethernet switch and a combined ordinary clock (OC) and a peer-to-peer transparent clock (P2P TC). One of the important issues undermining the synchronization performance of a system is the asymmetry of message propagation delay. IEEE 1588 PTP devices assume that the propagation delay is symmetric for both directions, which is generally not true. The Gigabit Ethernet (e.g. 100BASE-TX) switch uses all four pairs for both directions so that the asymmetry almost disappears or is a small constant that can be eliminated using a correctional constant. Other time errors that may arise could be the result of quantization errors, oscillator instability and phase-locked loop (PLL) internal jitter. Quantization errors arising from rate computations can be reduced to a few nanoseconds by using a relatively long dividend (e.g. an 80 bits dividend) in division calculations. Also if the wander and drift of the oscillators are not faster than the correction mechanism interval then the synchronization accuracy is not significantly affected. The authors perform an experimental evaluation of their combined OC and P2P TC model. The network setup uses the implemented switches supporting IEEE 1588 and connects five devices to do time synchronization. One device generates dummy traffic to introduce random queuing delays in the switches. An oscilloscope screenshot measuring the time error between the outputs of two devices is shown in Figure 16. It can be seen from the figure that the timing output of the slave device is within the range of -19.8ns to 29.0ns of the master device.

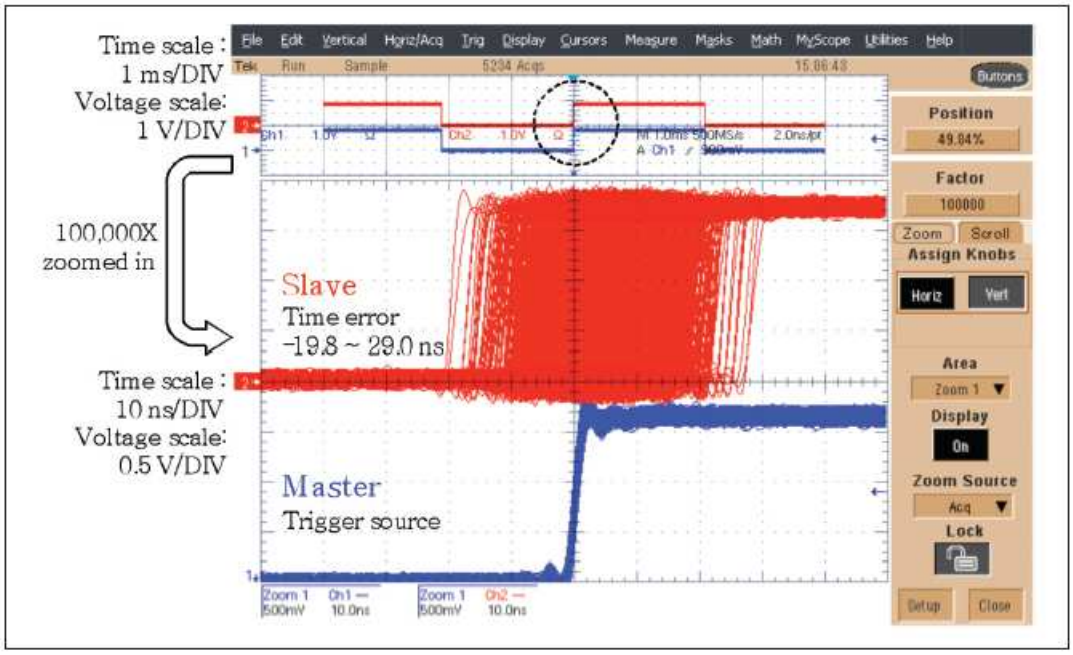

Figure 16: Oscilloscope screenshot measuring time synchronization performance. [15]

3.2.5 OPNET Simulation Results over Ethernet

The synchronization accuracy of IEEE 1588 PTP standard is dependent on the configuration of the network, i.e. the type of network devices, the number of devices in the network that contribute to the traffic, the processing delays and the point of time-stamping of the frames at the server and client. The authors of [16] conducted simulation studies on the IEEE 1588 PTP clock accuracy using OPNET to investigate the above mentioned factors. The simulations results for a hub based Ethernet network and a switch based Ethernet network are presented here. The simulations were done under varying load conditions. Also the authors compared the difference in clock accuracy achieved using time-stamping at the PTP layer and at the physical layer. The various time stamping techniques are described below.

- **Time stamping outside MAC layer** – PTP packets are time stamped at the PTP layer and therefore include protocol and processing delays at the PTP and other intermediate layers. A single M/G/1 queue process model is used to estimate all protocol processing delays.
- **Time stamping inside MAC layer** – PTP packets are time stamped just before they leave the physical layer and hence protocol and layer processing delays are not incurred. In case of a retransmission, the time stamp when the packet was last sent is recorded.

The results of different simulation scenarios are presented in the following subsections.

3.2.5.1 Hub Based IEEE 802.3 Ethernet

The basic simulation setup consists of 8 workstations (a master, a slave and other nodes) connected using a hub. In an unloaded hub network, with time stamping inside the MAC layer, the delay computed at the client was 5.9366 μs and the offset was 0 while with time stamping outside the MAC layer, the delays were highly variable as they also include protocol and layer processing delays. The average offset computed was 79.1258 μs. When the hub based network is loaded, with time stamping inside the MAC layer, the results obtained are the same as in the no load hub situation. With time stamping outside the MAC layer, and the load varied, the calculated offsets were highly variable due to processing delays. The results observed are shown in Table 2.

Table 2: Data obtained in a loaded hub network with time stamping outside the MAC layer. [16]

Load	4.5%	9%	20%	45%	70% (seed 99)	70% (seed 177)
Offset in µs	41.208	17.95	364	397	-100	-650

3.2.5.2 Switched IEEE 802.3 Ethernet

The basic simulation setup consists of 8 workstations (a master, a slave and other nodes) connected using a switch. In loaded switched network, if time stamping is done inside the MAC layer, then the delays due to packet buffering in the switch affects the synchronization accuracy and increases with the increasing traffic load. If time stamping is done outside the MAC layer in a loaded network, then the offsets are highly variable due to variable protocol processing delays. The results are shown in Table 3.

Table 3: Data obtained in a loaded switched network. [16]

Load	3.5%		17.5%		35%		60%	
Seed	99	1177	99	1177	99	1177	99	1177
Offset in µs (time stamping inside the MAC layer)	0.015	0.379	1.02	1.3	2.04	2.42	4.26	5.67
Offset in µs (time stamping outside the MAC layer)	-110	-16.7	4.9	79.7	17	-36.5	216	-7.8

The authors suggested that it is not feasible to use IEEE 1588 in switched Ethernet especially when the network is loaded more than 35%. They suggested to use the transparent clocks at the switch to improve the synchronization accuracy.

3.3 Adaptive Oscillator Correction Method

The authors of [19] present an adaptive correction algorithm to enhance the oscillator performance in holdover mode. The oscillator used in the timing module on a base station is usually phase locked by a one pulse per second (1-pps) GPS signal. When the GPS signal is lost, the timing module enters holdover mode and the timing module accuracy becomes dependent on the local oscillator. The proposed adaptive correction model can improve the oscillator performance in this situation. According to 3GPP2 recommendations, the CDMA systems must not exceed 10 μs cumulative time error (CTE) during an 8 hour holdover period. [19]

The 10 μs CTE amounts to 0.35 ppb of oscillator drift and therefore requires an expensive oscillator such as a double oven controlled crystal oscillator (DOCXO) to fulfill the stability requirement over a 75°C operational temperature range of the oscillator. The authors suggested employing an adaptive model of the timing module during the locked period and then using the resulting model to correct the oscillator drifts in holdover mode. They demonstrated their algorithm on a relatively cheaper oscillator i.e. a single oven controlled crystal oscillator (OCXO). The timing module on which their model is based is shown in Figure 17. The timing module mainly includes the following

- The GPS receiver module which receives a precise 1-pps GPS signal
- A 10 MHz OCXO

- The frequency multiplier to generate a frequency source
- The digital phase detector to count the numbers of period of the frequency source
- The correction signal calculator to compute the correction signal based on the count value
- The adaptive oscillator model.
- A digital-to-analog converter (DAC) to control the 10 MHz OCXO, and also to feed the adaptive oscillator model in holdover mode.
- A temperature sensor to collect the ambient temperature.

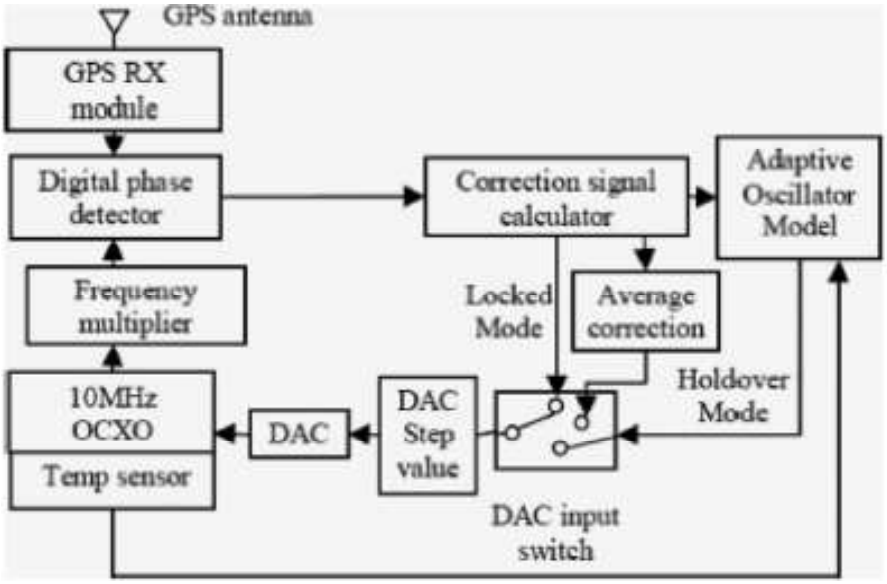

Figure 17: Detailed Block Diagram of the Timing Module System. [19]

The correction signal generated by the control loop of the timing module compensates the temperature and ageing effects of the oscillator. The authors determine the oscillator frequency stability model as shown here:

$$OSC_{stab}(k) = a * u^2(k) + b * u(k) + c + d * k \qquad [19]$$

where $OSC_{stab}(k)$ is the oscillator frequency stability (in ppb)

$u(k)$ is the ambient temperature

a and b are coefficients of the temperature

c is a nonzero initial offset

d is the ageing rate

From the oscillator stability equation, the authors determine the correction signal model which is based on an ARMAX model and solved by the recursive prediction error method (RPEM) (Details can be found in [19]). The estimated correction signal $\hat{y}(k)$ is:

$$\hat{y}(k) = -\hat{a} * u^2(k) - \hat{b} * u(k) - \hat{c} - \hat{d} * k \qquad [19]$$

where $\hat{a}$, $\hat{b}$, $\hat{c}$ and $\hat{d}$ are the estimated system parameters.

The authors plotted the CTE for a corrected OCXO using their adaptive correction method vs. an uncorrected OCXO as shown in Figure 18. The locked period is 6 hours while the holdover period is 8 hours. The corrected OCXO shows a huge improvement over an uncorrected OCXO. Note that the sharp spike in the corrected OCXO figure in the beginning, when the training starts, is due to a nonzero initial offset existing in the system model and has no effect on the CTE improvement. [19]

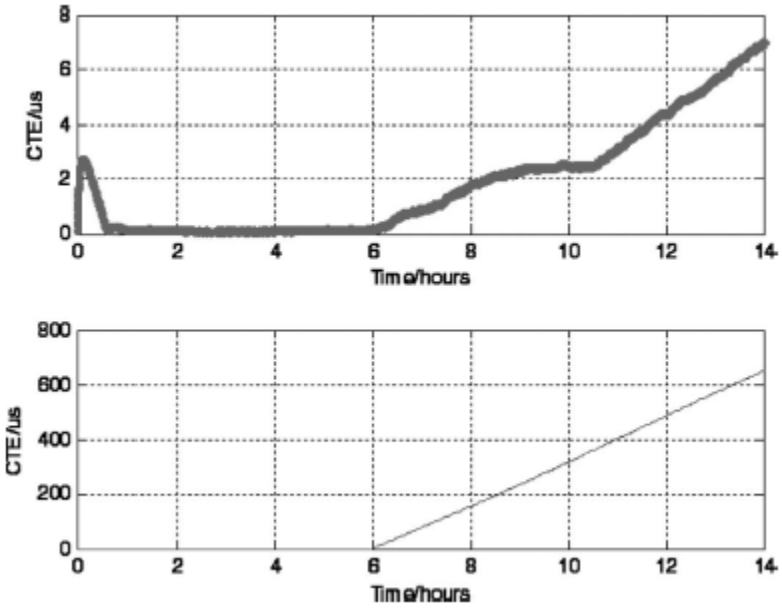

Figure 18: CTE for corrected (top) and uncorrected (bottom) OCXO during locked and holdover mode. [19]

3.4 Motivation

To conclude from the literature survey, the IEEE 1588 synchronization protocol is superior to other synchronization protocols such as NTP and IRIG. It has the flexibility to use an NTP-like basic message exchange structure or enhance itself further using the transparent clocks and/or the hardware-assisted time stamping for high precision and accuracy. Nevertheless the proposed solution only uses the basic message exchange structure of the IEEE 1588 protocol to synchronize the master and slave clocks. This means that if NTP or IRIG protocols were used instead, the results obtained would have been same. However, the use of IEEE 1588 protocol leaves an opportunity for future work to enhance it by including the transparent clocks or the hard-assisted time stamping. From the literature survey, we have seen that the IEEE 1588 performance has been

evaluated for a master-slave hierarchy without explicitly looking at the environment factors such as temperature and ageing effects, which cause drifts in the master and slave clocks. Furthermore, none of the published works attempt to correct for slave clock drifts while the slave is receiving IEEE 1588 synchronization updates. The adaptive oscillator correction method [19] looks only at the correction of the master clock drifts when its reference GPS signal is lost. The objective of the thesis is to enhance this adaptive oscillator correction method to apply on the slave clocks such that when the slave clock receives the IEEE 1588 synchronization updates, it trains an adaptive model. The resulting model can then be applied to the slave clock in two cases.

- Slave clock is waiting for its next update from the master clock.
- Slave clock is temporarily disconnected from the master clock due to some network outage.

The proposed solution can then be evaluated for various test scenarios such as varying load conditions, network congestion, outages etc. while explicitly looking at the temperature and ageing effects of the clock.

4 Chapter: Proposed Solution

In this chapter, the proposed solution for the clock synchronization problem is presented, which combines the IEEE 1588 protocol and the adaptive oscillator correction method (AOCM). It also provides details of the clock agent and algorithm implemented in the NS-2 simulator, presenting the results of temperature and ageing effects using our NS-2 model.

4.1 Combined IEEE 1588 and Adaptive Oscillator Correction Method

The proposed solution to master-slave synchronization revolves around the idea of combining the IEEE 1588 synchronization protocol (Section 3.1) and the adaptive oscillator correction method (Section 3.3). The solution suggests having a basic implementation of the IEEE 1588 protocol using the Delay Request-Response Mechanism as detailed in Section 3.1.3 to synchronize master and slave clocks in a network. The master and slave clocks here drift due to different initial offsets, different oscillator rates, temperature and ageing effects. The master clock is locked to a GPS signal for its accurate time and, using the Adaptive Correction Oscillator Model [19] (also briefly described in Section 3.3), a model is trained to account for the master clock drifts due to its rate, temperature and ageing effects. The correction signal obtained from the model is used to correct the rate of the master clock when the GPS signal is lost (i.e. in holdover mode). The solution proposes to extend this oscillator correction method to the slave clocks in the network.

For a slave clock in a network, its reference signal is the master clock from which it receives the IEEE 1588 synchronization updates at regular intervals (say every X

seconds). The solution suggests training a model for the slave clock using these IEEE 1588 synchronization updates from the master clock every X seconds in the same manner as the model for a master clock locked to a GPS signal is trained using the corrected oscillator method [19] (Section 3.3). The difference between the master clock and slave clock training models are the following:

- The master clock adaptive model is trained every 1 second, while in the case of the slave clock, it is receiving the IEEE 1588 synchronization updates every X seconds and therefore the slave clock adaptive model is trained every X seconds.
- When the master clock loses its connection to GPS signal, it goes in holdover mode and the resulting correction signal from the training model is applied to the oscillator rate (by adjusting the tuning voltage in case of a real oscillator). On the other hand, the slave clock has two kinds of holdover periods, which are termed here as micro-holdover and macro-holdover periods.
    - A micro-holdover period for a slave clock is the period during which the slave clock is waiting for its next IEEE 1588 synchronization update from its master clock. For a synchronization frequency of X seconds, each micro-holdover period is therefore equal to X seconds.
    - A macro-holdover period for a slave clock is the period during which the slave clock is temporarily disconnected from the master clock due to some network outage.

The solution suggests applying the resulting correction signal from the slave training model in both kinds of holdover periods. This is done by directly adjusting the slave oscillator rate using the correctional rate (by adjusting the tuning voltage in case of a real

oscillator). The modified diagram for the slave clock timing module system is shown in Figure 19.

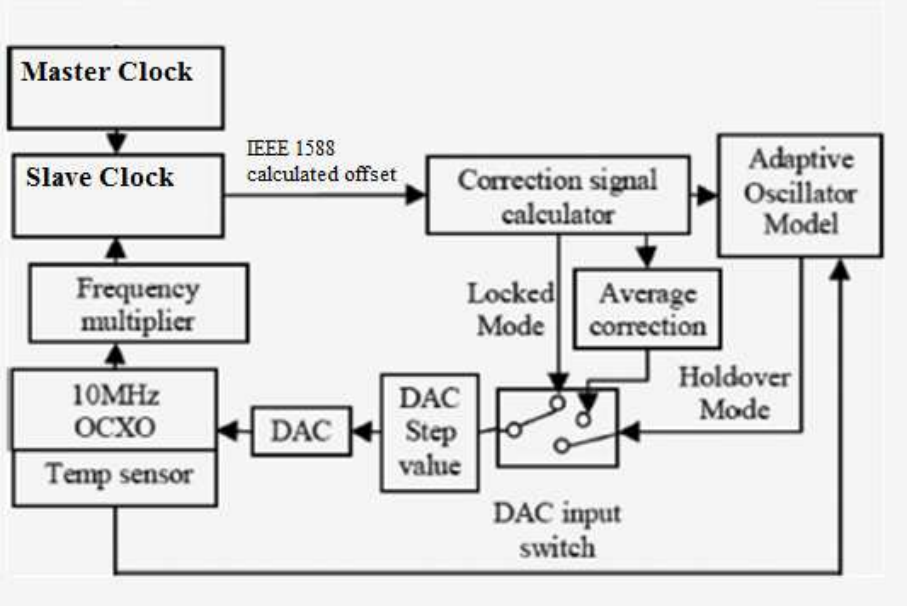

Figure 19: Detailed Block Diagram of Slave Clock Timing Module System.

The slave clock timing module in Figure 19 is based on the master clock timing module in Figure 17. Here, the slave offset from the master clock calculated using the IEEE 1588 protocol is fed to generate a correction signal as well as to train the adaptive oscillator correction model.

The extension of the adaptive oscillator correction method to the slave clock aims to correct the temperature and ageing drifts of the oscillator and hence to improve the accuracy and stability of the clock. The performance of the proposed solution is evaluated by implementing it in the NS-2 simulator [21], discussed in Chapter 5. For this purpose, a clock agent simulating a real network clock having drifts due to temperature and ageing effects is implemented in NS-2. Based on the proposed solution, the IEEE 1588 protocol

and the adaptive correction methods for the master and slave clock are also implemented in NS-2. The next sub-section discusses the implemented NS-2 clock agent model, presenting the NS-2 simulation results for the temperature and ageing effects on the clock time stamps.

4.2 NS-2 Clock Agent

4.2.1 Description

NS-2 is an open-source event-driven simulator designed specifically for research in computer communication networks. NS-2 supports simulation of TCP, routing, and multicast protocols over wired and wireless networks. NS-2 is written in the C++ programming language and the OTcl scripting language. [21] [22]

As part of the thesis work, a Clock Agent C++ Class is implemented in NS-2 to simulate a real clock. Here the environmental changes such as temperature and ageing effects causing drifts to a real clock oscillator are also considered. A clock agent can be attached to an NS-2 node[3]. The details from creating a clock agent to configuring the parameters of clock agent in a TCL script are given in Appendix A. Here are some of the features of the implemented clock agent.

- A clock agent has a time stamp and a natural rate which determines its drift in Parts Per Billion (PPB) w.r.t. to a perfect reference clock. The natural rate is configurable from an NS-2 OTcl script. The natural rate of the clock is affected by the temperature and ageing effects.

[3] "A node is a basic network component in NS-2. It acts as a computer host (e.g., a source or a destination) and a router (e.g., an intermediate node). It receives packets from an attached application or an upstream object, and forwards them to the attached links specified in the routing table (as a router) or delivers them to the ports specified in the packet header (as a host)."[22]

- The implementation provides an ability to configure a linear ageing effect and a linear or quadratic temperature effect. The temperature profile of the surroundings is also configurable independently for each clock in the system.
- An initial value at which the clock starts is configurable.
- The implementation provides a way to capture time stamps of the clock at regular intervals. The collected data can be imported into a graphing tool to visualize the accuracy of the clock.
- A clock agent can be configured to act as a master clock or a slave clock. A master clock has the ability to synchronize itself to a highly accurate GPS signal with a GPS noise of 20 ns RMS (Root Mean Square) jitter on a 1pps edge. The GPS signal is implemented in NS-2 using the simulation time with the 20 ns RMS noise reflected by using the normal random variable generator. A slave clock, on the other hand, has the ability to synchronize itself to a master clock using the IEEE 1588 synchronization protocol.
- A basic version of the IEEE 1588 synchronization protocol as discussed in Section 3.1.3 is implemented in NS-2 to facilitate the slave-master synchronization. The synchronization frequency and the assignment of a master to a slave clock are configurable. The logs are also available to capture the actual delays experienced by the IEEE 1588 packets in both directions.
- Both master and slave clocks have the self-correcting adaptive oscillator correction algorithms implemented to counter the effects of ageing and temperature effects. The master correction method as described in Section 3.3 is

implemented. The slave correction method is implemented using the proposed solution (see Section 4.1).

- The clock agent provides the ability to configure the locked and holdover periods for master and slave clocks to reflect a network outage for master-GPS and master-slave connections respectively. During a holdover period, the correction signal obtained from the trained adaptive model is applied directly on the natural rate of the clock to correct its drifts.

4.2.2 Temperature and Ageing Effects

In this section, the accuracy of a free running clock w.r.t. a perfect clock is presented using the Clock Agent implemented in NS-2. The purpose is to demonstrate the temperature and ageing effects on the clock accuracy for various simulation periods. A linear temperature effect of 4ppb/75°C with an 8-hour cycle temperature profile shown in Figure 20 and a linear ageing effect of 1ppb/day are used. The clock used has no initial drifts and any drifts happening over time in the clock rate are only due to the effects of ageing and temperature variations. The default values are used for other simulation parameters as detailed in Appendix A. The complete TCL script is given in Appendix B.

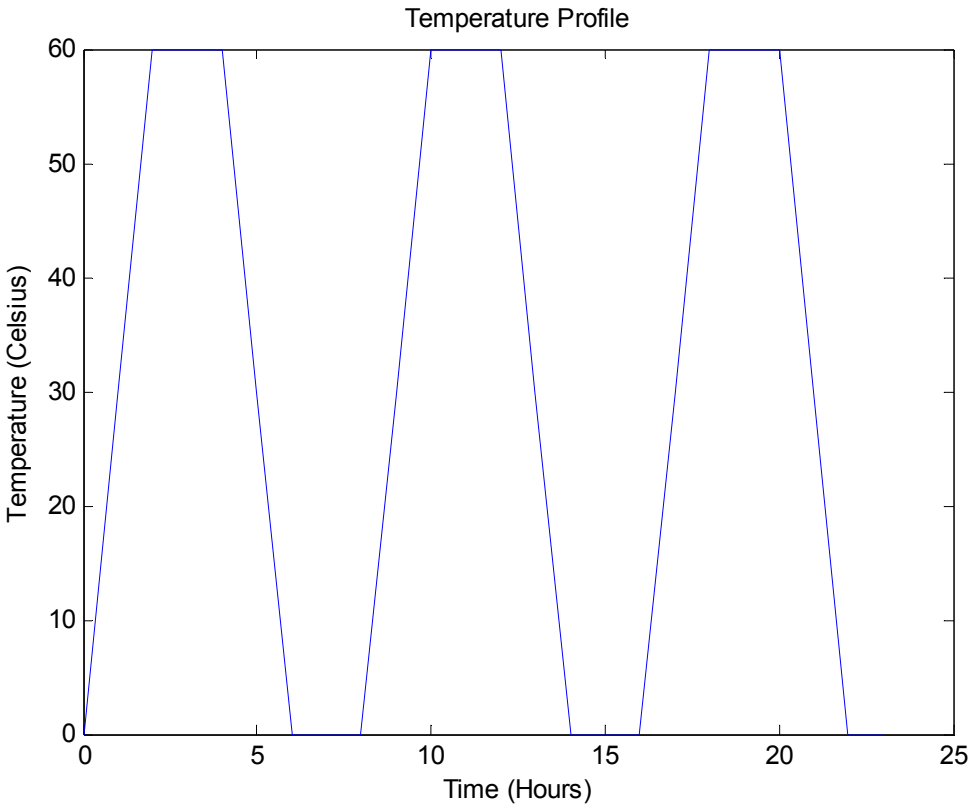

Figure 20: Temperature Profile

Figure 21, Figure 22 and Figure 23 show the clock accuracy for a simulation periods of 1 day, 1 week and 2 weeks for three scenarios: ageing only, temperature only and both ageing and temperature effects.

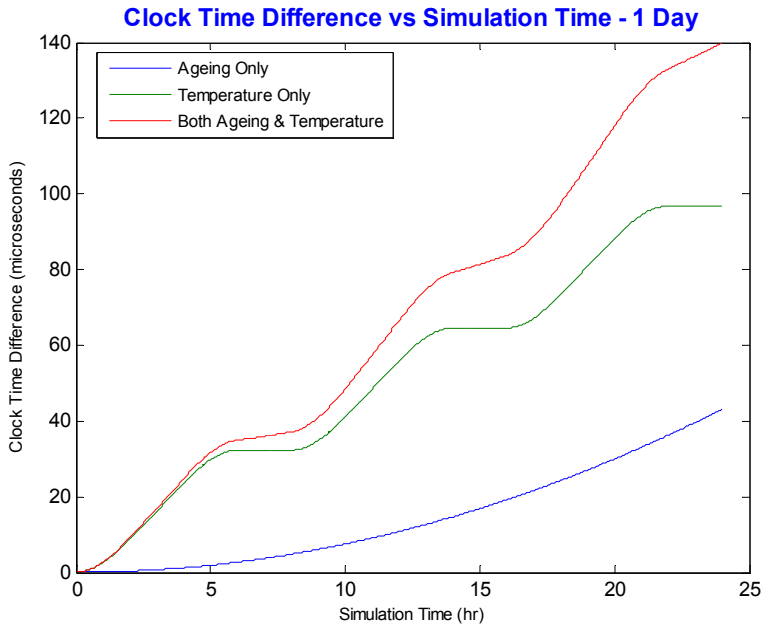

Figure 21: Clock Accuracy for Simulation Period of 1 Day - Temperature and Ageing Effects

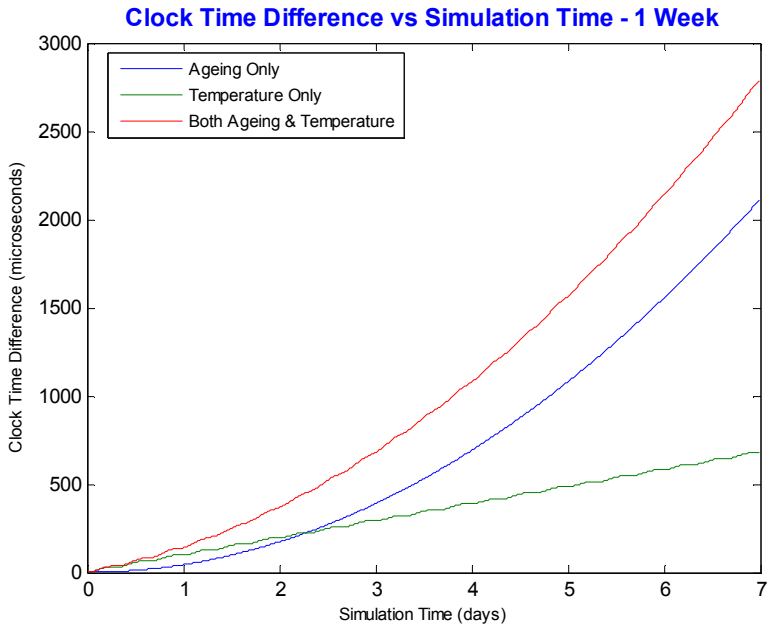

Figure 22: Clock Accuracy for Simulation Period of 1 Week - Temperature and Ageing Effects

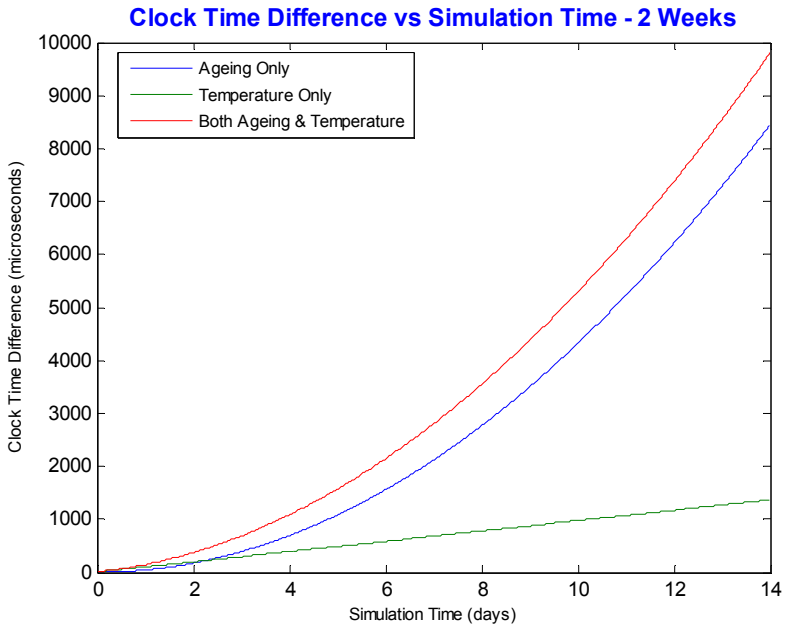

Figure 23: Clock Accuracy for Simulation Period of 2 Weeks - Temperature and Ageing Effects.

The figures show that the ageing effect is exponential as the ageing drifts accumulate over time. The temperature effect follows the pattern of the temperature profile used such that the clock drifts faster when it is hot and drifts slower when it is cold. The figures also demonstrate the cumulative effect of temperature and ageing drifts on clock accuracy. For a short period, the cummulative effect is dominated by temperature drifts, however for a longer simulation period, the ageing effect dominates the cummulative effect as the drifts caused by ageing accumulates over time. This is consistent with the discussion in Section 2.2: if the thermal environment is undergoing variations in a time period which is shorter than the time required for the oscillator to drift significantly with respect to the aging rate, then the temperature effect on the frequency stability of the oscillator will dominate. On the other hand, if the thermal environment is stable, aging will dominate the frequency stability of the oscillator.

5 Chapter: Simulation Results

In this chapter, the results from NS-2 simulations measuring synchronization accuracy using the IEEE 1588 synchronization protocol and the adaptive oscillator correction method (AOCM) are presented and discussed. First, the generic NS-2 simulation setup is described. The test cases run in NS-2 cover various factors such as temperature and ageing effects affecting the clock frequency of master and slave clocks. The network conditions such as varying network traffic and network outages are also considered and their effect on clock synchronization is studied in this chapter. Particularly the test cases are divided into three sections given below.

- Test cases with no traffic in the network. (Section 5.2)
- Test cases with traffic introduced in the network using different load profiles. (Section 5.3)
- Test cases to examine the direct effect of parameters such as clock rate, IEEE 1588 synchronization frequency, simulation time period, AOCM training period and AOCM holdover period. (Section 5.4)

5.1 Simulation Setup

The network topology shown in Figure 24 is used in all simulation test cases unless a different topology is specified in a test case. The topology consists of a master node n0 connected to a slave node n5 with n1, n2, n3 and n4 as intermediate nodes, making it a 5-hop topology.

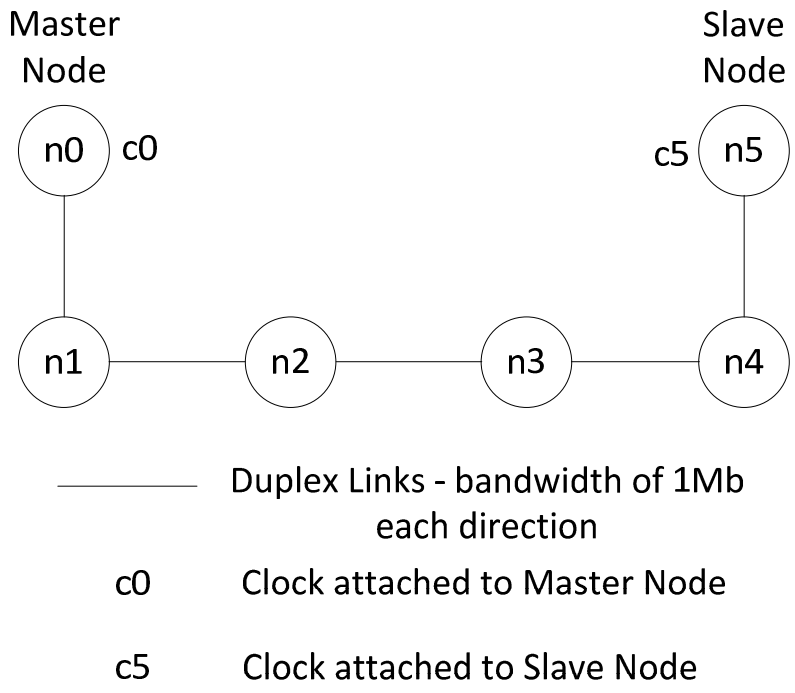

Figure 24: Network topology

By default, the following values are used for different simulation parameters for all test cases unless a different value is explicitly specified in a test case.

- Bandwidth of Duplex links = 1 Mb
- Propagation delay between two adjacent nodes = 100 ns
- Simulation run time = 16 hours
- Master clock rate = 1 (running at the same rate as ref. clock)
- Slave clock rate = 1.0000001 (100 ppb faster than ref. clock)
- IEEE 1588 synchronization frequency = 100 s
- Number of hops (between master and slave clocks) = 5 hops

The temperature and ageing effect values are derived from [19].

- Master node temperature effect when enabled = 4 ppb/75°C with quadratic term equal to -0.00031966 ppb/°C^2
- Master node ageing effect when enabled = 1 ppb/day
- Slave node temperature effect when enabled = 5 ppb/75°C with quadratic term equal to -0.00031966 ppb/°C^2
- Slave node ageing effect when enabled = 2 ppb/3days
- Adaptive Model on Master Clock (when enabled). Parameter values below are derived from [19].
    - Training period (locked mode) = 0 to 8 hours
    - Holdover period (unlocked mode) = 8 to 16 hours
    - AOCM frequency = 1s
        - Note: The AOCM frequency of 1 second applies to both training and holdover periods of the master clock.
- Adaptive Model on Slave Clock (when enabled)
    - Training period (locked mode) = 0 to 4 hours, 4.5 hours to 16 hours
    - Holdover period (unlocked mode) = 4 to 4.5 hours
    - AOCM frequency = 1s
        - Note: The AOCM frequency of 1 second applies to the holdover period of the slave clock only. For the training period of the slave clock, the AOCM step is executed right after receiving the IEEE 1588 synchronization update from the master clock i.e. the AOCM

frequency for the slave clock's training period is same as the IEEE 1588 synchronization frequency (100 seconds in the current setup).

The clocks of both master and slave nodes start ticking at a simulation time of 1s and stop when the simulation ends.

5.1.1 Traffic Model Description

The traffic model described in Appendix VI in [20] is used whenever required in a test case. This traffic model is aimed to model the traffic on networks where the majority of the traffic is data. To model this traffic, 60% of the load should be based on packets of maximum size while the remaining 40% on packets are a mix of minimum and medium size.

Hence the packet size profile can be summarized as below:

- 60% of the load must be maximum size packets (1518 bytes)
- 30% of the load must be minimum size packets (64 bytes)
- 10% of the load must be medium size packets (576 bytes)

The network load profile used is individually described in the test case description whenever needed.

5.1.2 Temperature Profile Description

The temperature profile used for calculating the temperature effects on a clock oscillator is taken from [19] and is shown in Figure 25. The temperature profile represents the temperature values observed at the clock oscillator. The range of temperature variation is 60°C over an 8-hour cycle. The temperature range is large enough to represent the real

working environment such as the cellular base stations located outdoors where the temperature could be very hot or cold depending on the region. The 8-hour cycle guarantees that the simulation results are obtained fast enough.

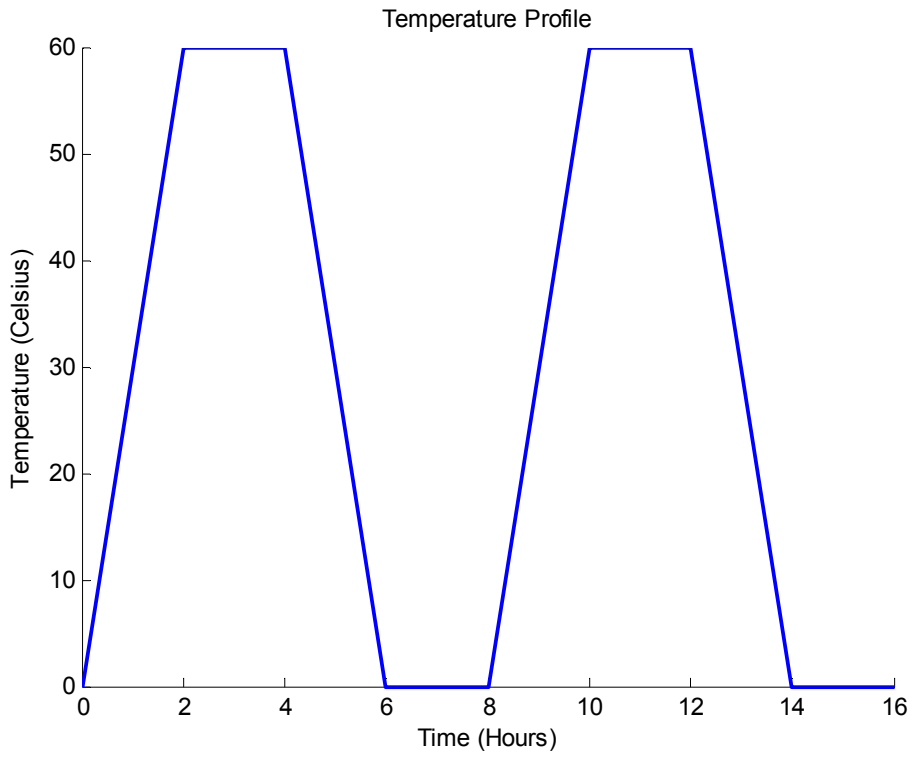

Figure 25: Temperature Profile

5.1.3 Metrics Collected

In general for each test case, the data is collected as described in sub-sections here. Each test case is run 50 times and the data collected is averaged and plotted versus the simulation time (in hours). The number of runs is chosen as 50 because it was determined from the initial experiments that 50 runs are enough to produce stable results with 95 % confidence interval.

5.1.3.1 Master Clock Synchronization

Master clock synchronization accuracy is measured with respect to the reference clock. For this, the absolute difference (in μs) between the time stamp of the master node and the reference signal is collected every simulation second for each run of a test case.

5.1.3.2 Slave Clock Synchronization

Slave clock synchronization accuracy is measured both with respect to the master clock as well as the reference clock. Since IEEE 1588 synchronization happens every 100 seconds (default synchronization window size), the slave clock drifts from the master between the synchronization events. Therefore the maximum slave drift in each synchronization window can be used to determine the slave clock accuracy. The following data is therefore collected.

- The maximum absolute difference (in μs) between the time stamps of the slave node and the master node in each synchronization window.
- The maximum absolute difference (in μs) between the time stamps of the slave node and the reference clock in each synchronization window.

It should be noted that if there is traffic in the network, the IEEE 1588 synchronization accuracy could be affected due to asymmetric latencies introduced due to traffic. The asymmetric delays contribute to the errors in clock offset calculation in the IEEE 1588 protocol as described in Section 3.1.3. Hence, for the test cases with traffic scenarios, the delays as given below are also collected to investigate their effect on slave synchronization.

- The difference between the forward delay (actual master to slave delay experienced) and the reverse delay (actual slave to master delay experienced), collected every synchronization event.

5.2 Test Cases with No Traffic

The test cases presented in this section are run with no traffic in the network using the topology shown in Figure 24. This means the delays experienced by the IEEE 1588 synchronization packets are symmetric and do not affect the IEEE 1588 synchronization accuracy. Here, only a few test cases are presented. The additional test cases are described in Sub-Appendix C.1.

5.2.1 A Real Slave Clock vs. a Perfect Master Clock with Temporary Network Outages – No Corrections in Macro-Holdover Period

5.2.1.1 Description

In this test case, we have a real slave clock connected to a perfect master clock. A perfect master clock means that the oscillator of the clock has no drifts due to temperature and ageing effects and therefore runs perfectly, at the same frequency as the reference clock. The slave clock used here is referred to as a real slave clock as it has both temperature and ageing effects enabled. At a simulation time of 4 hours, the connection between the master and slave clocks is lost, reflecting a temporary network outage. The outage prolongs for 30 minutes and then the connection is restored. The outage period is considered as the macro-holdover period for the slave clock. During the 30 minutes outage, the slave clock fails to receive the IEEE 1588 synchronization updates from its

master clock. No adaptive corrections are applied on the slave clock in macro-holdover mode, however in the locked mode, the slave clock is training a model based on the master clock updates. The test case examines the effect of outage on clock synchronization. The test case is repeated for the network outage of 14 to 14.5 hours.

5.2.1.2 Results

Figure 26 shows the master clock synchronization w.r.t. to the reference clock and the slave clock synchronization w.r.t. to the master clock and the reference clock, facing a network outage (4 to 4.5 hours) but no AOCM corrections applied during the whole simulation period.

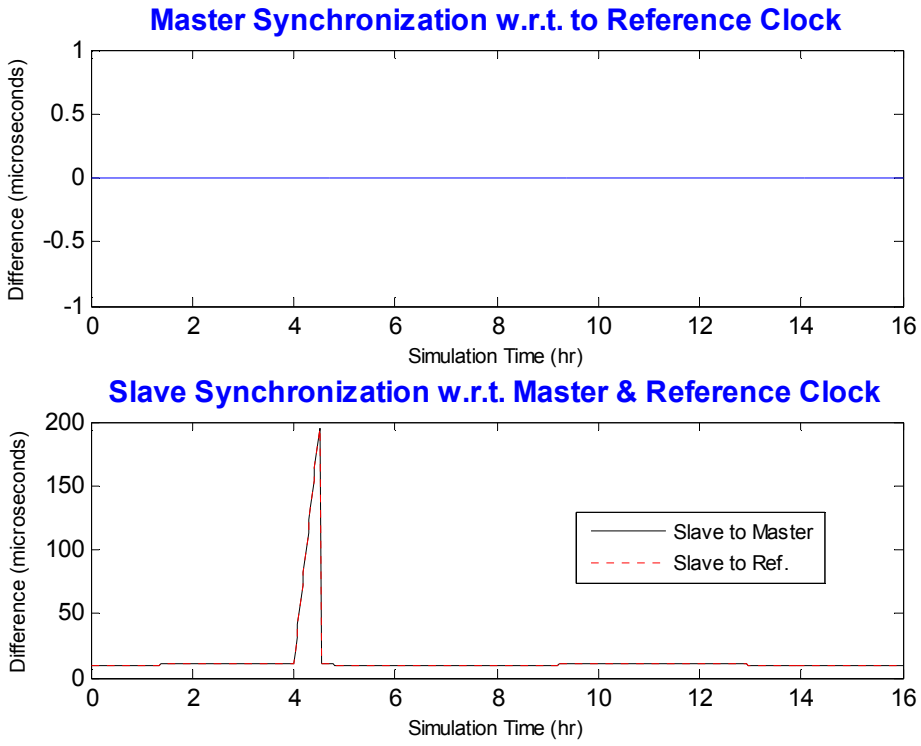

Figure 26: Clock Synchronization - A Real Slave Clock vs. a Perfect Master Clock Reflecting Network Outage (4 to 4.5 hours) with No AOCM Corrections

Figure 27 shows the slave clock synchronization when AOCM corrections are applied in the micro-holdover period. Whenever the slave clock receives the IEEE 1588 clock synchronization update (i.e. every 100 seconds), the AOCM step is executed to train the model.

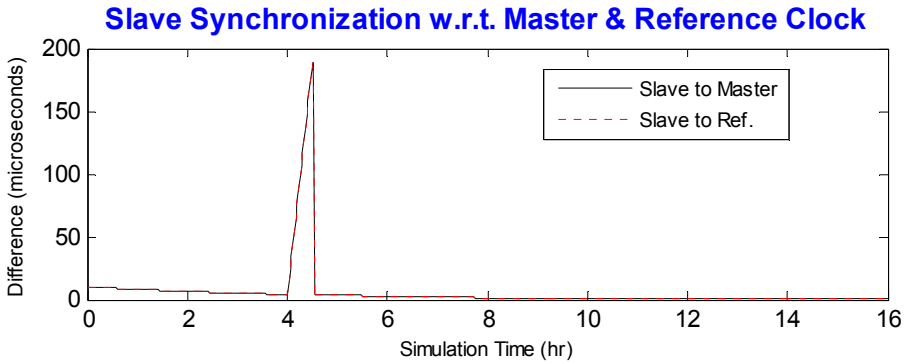

Figure 27: Clock Synchronization - A Real Slave Clock Reflecting Network Outage (4 to 4.5 hours) with AOCM Corrections Applied on Slave Clock in Micro-Holdover Period only

Figure 28 shows the slave clock synchronization for the case when the network outage happens from 14 to 14.5 hours.

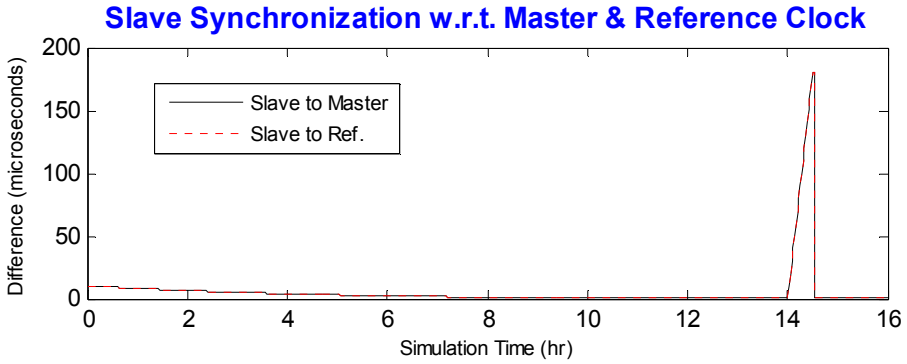

Figure 28: Clock Synchronization - A Real Slave Clock Reflecting Network Outage (14 to 14.5 hours) with AOCM Corrections Applied on Slave Clock in Micro-Holdover Period only

5.2.1.3 Discussion

From Figure 26, the master to reference clock difference is zero as the master is a perfect clock. Both the slave to master and the slave to reference differences during the locked period (0 to 4 hours and 4.5 to 16 hours) are around 10 µs due to dominant slave clock drift of 100 ppb as explained in the test case results of Sub-Appendix C.1.1. However when the AOCM corrections are applied on the slave clock during the micro-holdover period as shown in Figure 27, the slave clock differences are reduced from almost 10 µs to 4 µs for the 0 to 4 hours period and then to 1 µs for the 4.5 to 16 hours period. Here the slave clock, on receiving the update from its master clock every 100 seconds, is training an AOCM model. The AOCM correction calculated from the training model every 100 seconds is applied to the slave clock rate whenever queried for the time stamp (i.e. every 1 second). It should be noted that the slave only learns about the master time stamp every 100 seconds compared to the master clock learning about the GPS signal every 1 second (see Sub-Appendix C.1.3 for master clock test case). The slave AOCM model has less training data and is therefore less accurate compared to the master AOCM model. This means the slave needs a longer training period for producing accurate results. During the network outages (4 to 4.5 hours), the slave clock no longer receives updates from the master clock and the difference grows from 10 µs to around 190 µs which reflects the dominant slave clock drift due its rate and some drift due to the temperature/ageing effects in the 30 minutes holdover period. The results of 14 to 14.5 hours outage in Figure 28 are similar with the difference growing from 1 µs to around 180 µs.

5.2.2 A Real Slave Clock vs. a Perfect Master Clock with Temporary Network Outages – Corrective Model Enabled on Slave Clock

5.2.2.1 Description

In the second test case, the first test case (Section 5.2.1) is repeated with the modification that the adaptive corrective model trained during the locked period is applied in the slave macro-holdover period as well, to correct the time stamp of the slave during the network outage (4 to 4.5 hours).

5.2.2.2 Results

Figure 29 shows the slave clock synchronization w.r.t. to the master clock and the reference clock for a real slave vs. a perfect master scenario facing a network outage (4 to 4.5 hours) with AOCM corrections applied on slave during the outage as well.

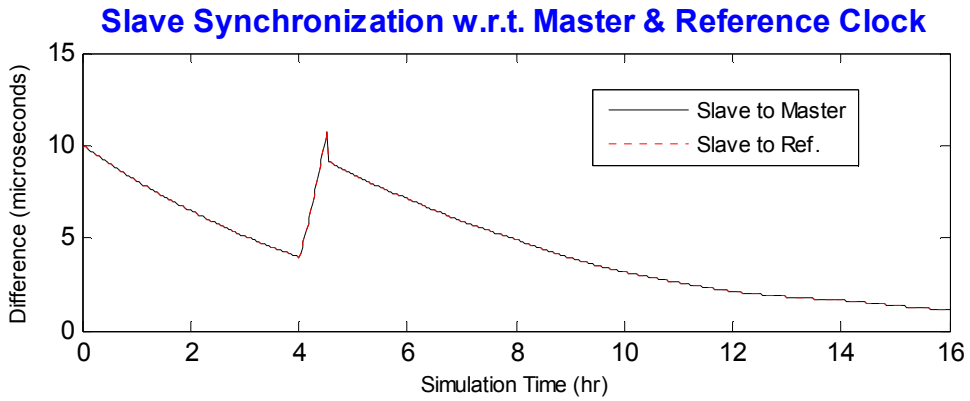

Figure 29: Clock Synchronization - A Real Slave Clock vs. a Perfect Master Clock Reflecting Network Outage (4 to 4.5 hours) with AOCM Corrections on Slave Clock

Figure 30 shows the slave clock synchronization for the case when the network outage happens from 14 to 14.5 hours.

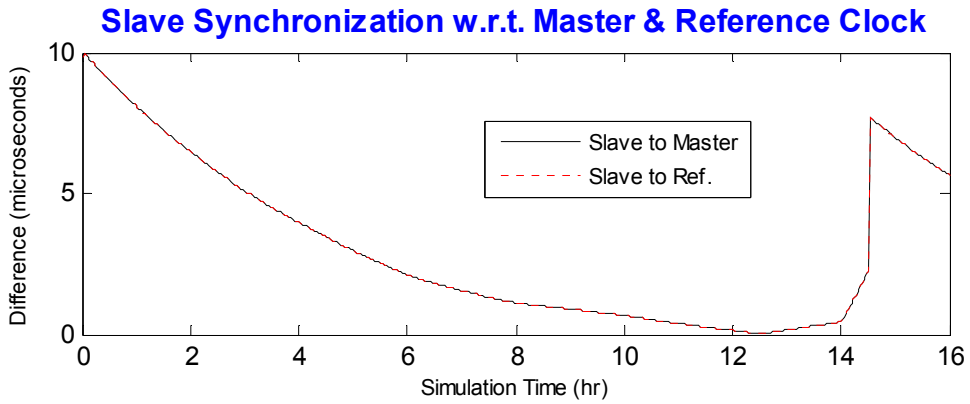

Figure 30: Clock Synchronization - A Real Slave Clock vs. a Perfect Master Clock Reflecting Network Outage (14 to 14.5 hours) with AOCM Corrections on Slave Clock

5.2.2.3 Discussion

From Figure 29, during the first locked period (0 to 4 hours), the slave to master and the slave to reference differences are the same as in the previous test case (Section 5.2.1) i.e. the AOCM model corrects the slave clock every second and the differences are reduced from almost 10 µs to 4 µs. On the other hand, during the outage period (4 to 4.5 hours), the slave differences decrease significantly compared to the previous test case from about 190 µs to 10 µs. This indicates that the slave accuracy w.r.t. the master as well as the reference clock is greatly improved using the oscillator corrected model in holdover period. In the 2nd locked period (4.5 to 16 hours), when the IEEE 1588 updates from the master clock are resumed, the slave clock continues to train its AOCM model and to improve the accuracy slightly. For the network outage of 14 to 14.5 hours as shown in

Figure 30, the slave has been comparatively trained longer and the slave drift is further improved to about 7 μs. It is observed that a longer training period for the AOCM model for the slave clock is needed to achieve more accurate results. The effect of the AOCM training period for the slave clock is studied in a later test case (Section 5.4.2).

5.2.3 Summary

The test cases in this section and Sub-Appendix C.1 have been run under no traffic with varying conditions on the master and the slave clocks. The adaptive oscillator correction model has been considered on both master and slave clocks. In a nutshell, the following points are observed.

- When no AOCM corrections are applied on the master clock in holdover mode, the slave to master clock synchronization accuracy can improve if both master and slave clocks are drifting in the same direction and can decrease if they are drifting in the opposite direction.
- The adaptive correction method on the master clock improves the master to reference clock synchronization significantly. When in holdover mode, the difference grows to about 2 μs for an 8 hour holdover period, much less compared to about 50 μs with no AOCM corrections applied for the parameters considered in the test case. Also, in the master holdover mode, the slave to reference clock accuracy would improve if the slave clock is running faster than reference clock and would decrease if the slave clock is running slower than the reference clock.
- The adaptive correction method on the slave clock improves the slave clock synchronization with respect to the master clock both in locked and holdover

mode. In locked mode, in addition to receiving the master synchronization updates every 100 seconds, the slave also trains an AOCM model every 100 seconds. In both micro and macro holdover modes, the accuracy improves up to 1 μs but the accuracy depends on the training period of the slave clock. Also in this case, the slave to reference clock synchronization becomes directly dependant on the master to reference clock accuracy.

It should be noted that here the performance of the IEEE 1588 and the oscillator corrected method is evaluated using a master-slave hierarchy with no traffic disturbance. Next it will be interesting to see how the network traffic delays the master updates to the slave clock and whether the proposed solution still improves the accuracy and stability of a slave clock in such scenarios. We know that the IEEE 1588 protocol does not perform well when the delays are asymmetric. So it can be expected that any asymmetric traffic will deteriorate the performance of the proposed solution. The next set of test cases examines this in more detail.

5.3 Test Cases with Traffic

The test cases presented in this section introduce traffic in the network using different load profiles. The network topology is shown in Figure 24 and the traffic model used is presented in Section 5.1.1. Each test case demonstrates the effect of various load profiles using real slave and master clocks with the AOCM corrections applied on the master only or both the slave and master clocks. The forward traffic (master to slave clock) is introduced from nodes n1 to n4. The reverse traffic (slave to master clock) is introduced from nodes n4 to n1. It should be noted that the load profiles used in different test cases

represent the real situation in cellular base station networks where most of the traffic arise from download traffic (i.e. from a base station to a mobile device), and hence making the forward traffic dominant over the reverse traffic. Since we have traffic in the network, the delays experienced by the IEEE 1588 synchronization packets are also observed to determine their effect on the IEEE 1588 synchronization accuracy. Since the traffic has no impact on the master clock, therefore the graphs for the master clock synchronization are not produced. Here, only a few test cases are presented. The additional test cases can be found in Sub-Appendix C.2.

5.3.1 Static Packet Load using Real Slave and Master Clocks - Corrective Model Enabled on the Master Clock only

5.3.1.1 Description

In the first test case with traffic scenario, we use real master and slave clocks having temperature and ageing effects. The adaptive oscillator correction model is enabled on the master clock only. Using the traffic model described in Section 5.1.1, a static traffic load of 80% is introduced in the forward direction (master to slave) while 20% traffic is introduced in the reverse direction (slave to master) starting at a simulation time of 3hrs and stopping at a simulation time of 13 hours. The test case examines the effect of static traffic in both directions on the slave clock synchronization.

5.3.1.2 Results

Figure 31 shows the slave clock synchronization w.r.t. to the master and reference clocks and the difference of delays experienced (forward - reverse) when a static packet load is

introduced between the real slave and master clocks with AOCM corrections applied on the master clock only.

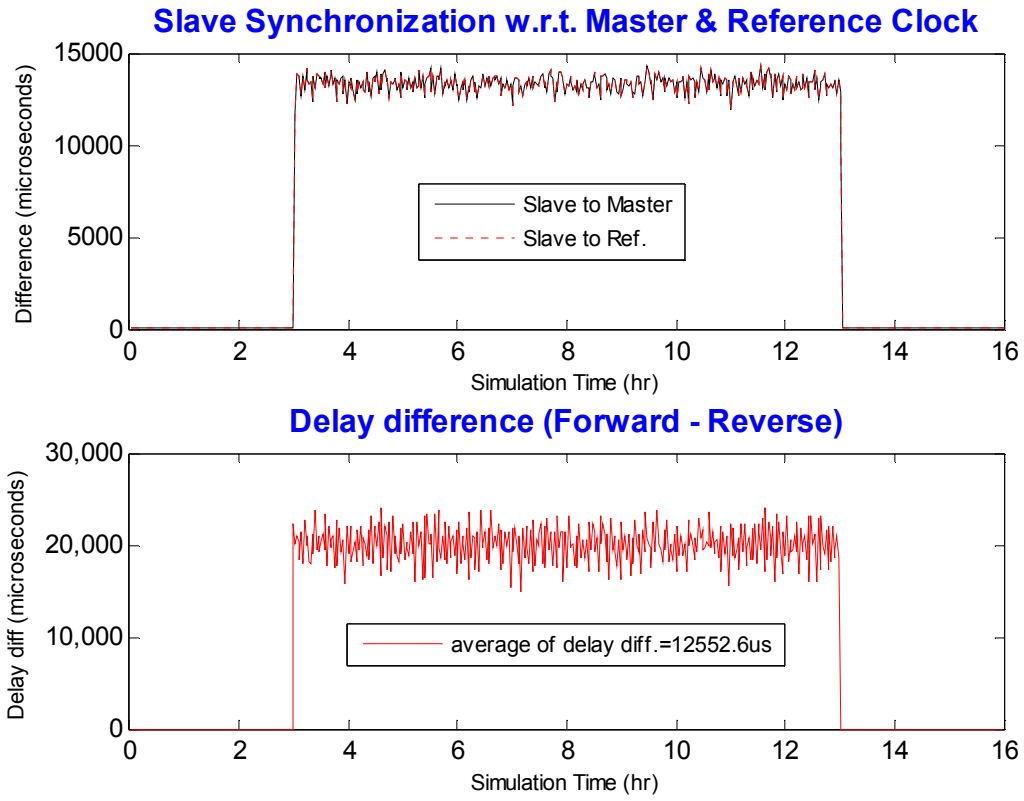

Figure 31: Clock Synchronization - Static Packet Load using the Corrective Model on the Master Clock only

5.3.1.3 Discussion

From Figure 31, the slave to master and reference clock differences are at the 14,000 µs level, which indicate that the traffic affects the slave accuracy very badly. With 80% traffic in the forward and 20% traffic in the reverse direction from a simulation time of 3 hours to 13 hours, the delays experienced by the synchronization packets are highly asymmetric and at the 20,000 µs level. The average of delay differences (forward –

reverse delay) is 12,552.6 µs which indicates the forward delays dominate due to the 80% forward traffic. Since the slave clock difference graph is the average of the absolute values while the delay graph is the average of true values in Figure 31, therefore they cannot be used to study the relationship between the slave accuracy and the asymmetric delays. The slave clock difference and delay plots for a single run is therefore produced and shown in Figure 32.

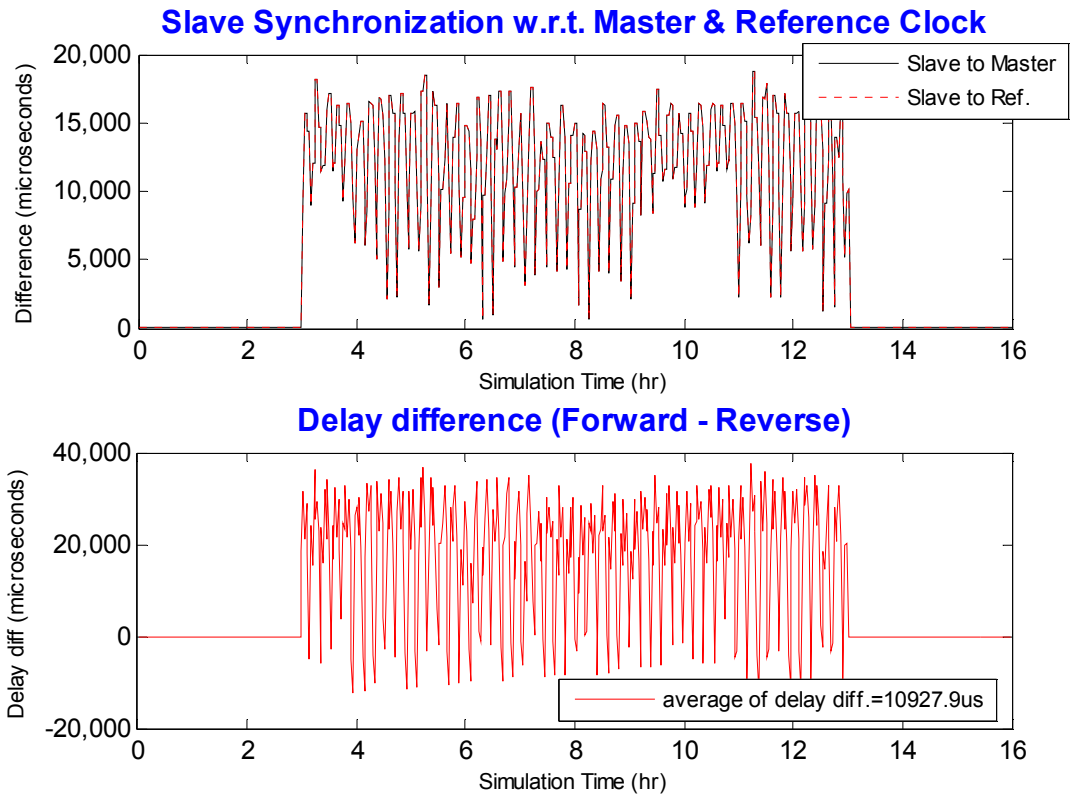

Figure 32: Clock Synchronization for Single Run - Static Packet Load using the Corrective Model on the Master Clock only

From Figure 32, the slave clock inaccuracy introduced by the traffic is in direct relationship with the asymmetric latencies appearing in the network such that the degree of inaccuracy is half of the degree of asymmetric delays. This is in accordance with the

IEEE 1588 protocol. As described in Section 3.1.3, the IEEE 1588 clock offset is calculated using the equation given below.

$$Clock\ Offset = t_2 - t_1 - \ Mean\ Propagation\ Time = \frac{[(t_2 - t_1) - (t_4 - t_3)]}{2}$$

where

$$Mean\ Propagation\ Time = \frac{[(t_2 - t_1) + (t_4 - t_3)]}{2}$$

$$= \frac{Forward\ Delay + Reverse\ Delay}{2}$$

If the delays are symmetric then the *Mean Propagation Time* is equal to the actual propagation times experienced by the synchronization packets. If the delays are asymmetric, the computed *Mean Propagation Time* differs from the actual propagation times. In this case, either the *Forward Delay* or the *Reverse Delay* is dominant, and the *Mean Propagation Time* and hence the *Clock Offset* contains an error term of up to $\frac{Forward\ Delay}{2}$ or $\frac{Reverse\ Delay}{2}$. Therefore the degree of inaccuracy introduced in the slave clock time stamp is approximately half of the dominant delays.

More specifically, the slave clock offset from the master clock can be written as:

$$Clock\ Offset = \frac{[(t_2 - t_1) - (t_4 - t_3)]}{2} = \frac{Forward\ Delay - Reverse\ Delay}{2}$$

Therefore, if we can predict or calculate the delay asymmetry correctly, we can correct the IEEE 1588 clock offset and account for errors introduced by irregular traffic latencies.

5.3.2 Sudden Large and Persistent Changes in Traffic Load using Real Slave and Master Clocks - Corrective Model Enabled on the Master Clock only

5.3.2.1 Description

In a second test case, the first test case is repeated using a different load profile. Using the load profile shown in Figure 33, the traffic is introduced starting at a simulation time of 3 hours and stopping at a simulation time of 13 hours. Here in the forward direction (master to slave) the network load is changed between 80% and 20% every hour while simultaneously in the reverse direction (slave to master), the network load is changed between 50% and 10%. The test case examines the effect of large and persistent changes in network load on the slave clock synchronization with AOCM enabled on the master clock only.

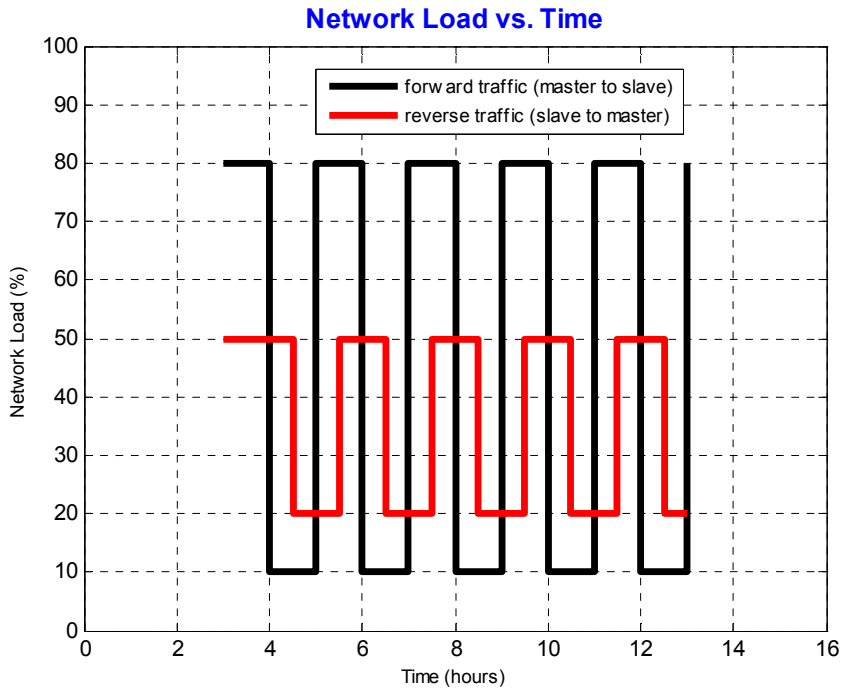

Figure 33: Load Profile demonstrating Sudden Large and Persistent Changes in the Network Load

5.3.2.2 Results

Figure 34 shows the slave clock synchronization w.r.t. to the master and reference clocks and the difference of delays experienced (forward - reverse) when traffic using load profile in Figure 33 is introduced between the real slave and master clocks with AOCM corrections applied on the master clock only.

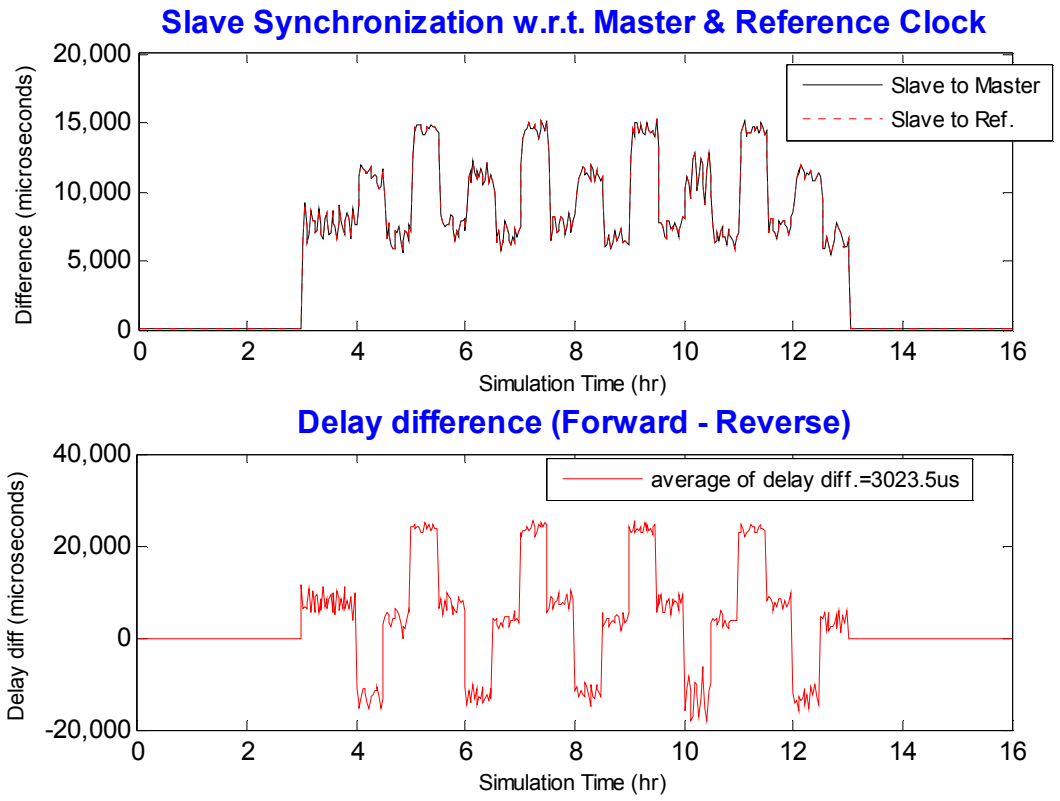

Figure 34: Clock Synchronization: Sudden Large and Persistent Changes in Network Load using the Corrective Model on the Master Clock only

5.3.2.3 Discussion

From Figure 34, the results of the slave to master and reference clock synchronization are similar to the previous test case with static packet load scenario (Section 5.3.1). The slave

differences are at the 15,000 µs level and the asymmetric delays experienced by the synchronization packets are at the 20,000 µs level. For the simulation time interval where forward traffic is dominant (e.g. 3 to 4 hours, 80% forward and 50% reverse), the effective delays are in the forward direction. For the simulation time interval where the reverse traffic is dominant (e.g. 4.5 to 5 hours, 10% forward and 20% reverse), the effective delays are in the reverse direction. The average of delay differences (forward – reverse delay) is 3,023.5 µs i.e. overall the forward delays dominate due to the dominant forward traffic in the load profile used.

5.3.3 Temporary Network Outage with Static Traffic Load using Real Slave and Master Clocks - Corrective Model Enabled on Both Clocks

5.3.3.1 Description

In a third test case, we demonstrate a temporary network outage (4 to 4.5 hours) between the master and slave clocks in the presence of static traffic load. The adaptive correction model is enabled on both master and slave clocks. Using the traffic model described in Section 5.1.1, a static traffic load of 40% is introduced in the forward direction while 30% traffic is introduced in the reverse direction starting at a simulation time of 3 hours and stopping at a simulation time of 13 hours. The test case examines the effect of network outage in the presence of static traffic in both directions on the slave clock synchronization accuracy.

5.3.3.2 Results

Figure 35 shows the slave clock synchronization w.r.t. to the master and reference clocks and the difference of delays experienced (forward - reverse) during network outage with static network load and the AOCM corrections applied on the master clock only. Figure 36 looks closely at the slave clock synchronization during the network outage.

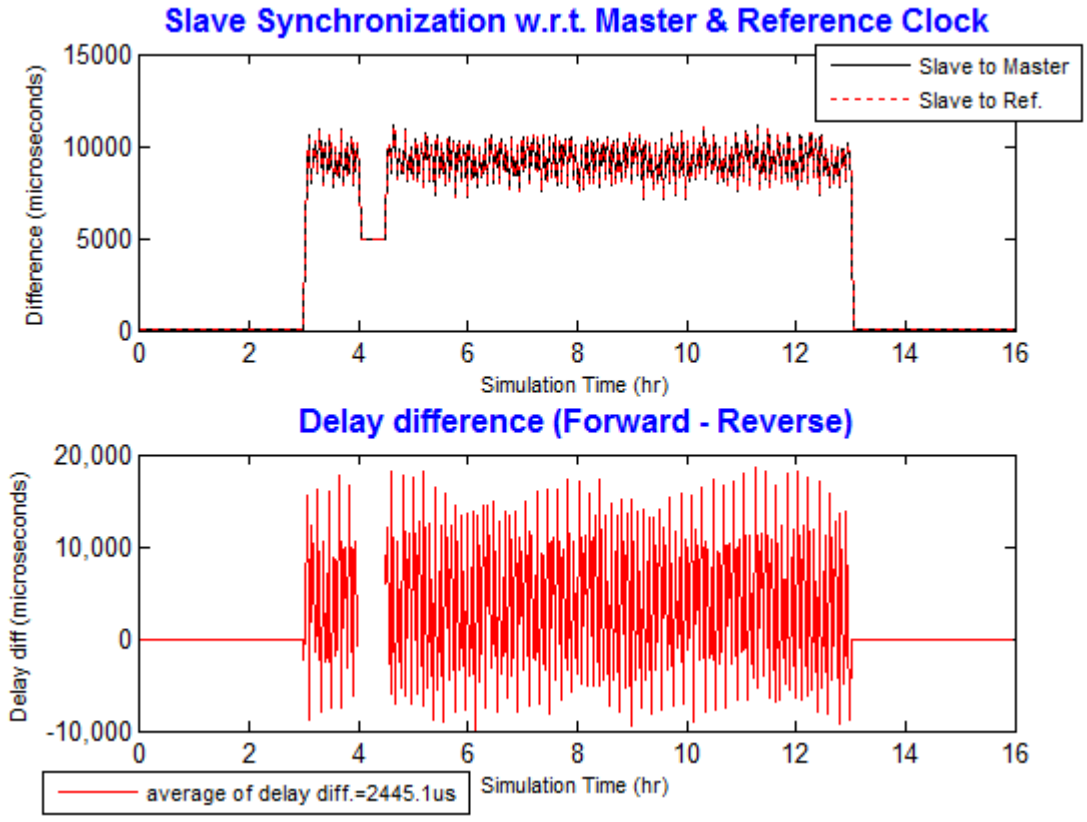

Figure 35: Clock Synchronization - Temporary Network Outage during Static Traffic with Corrections Applied on Both Clocks

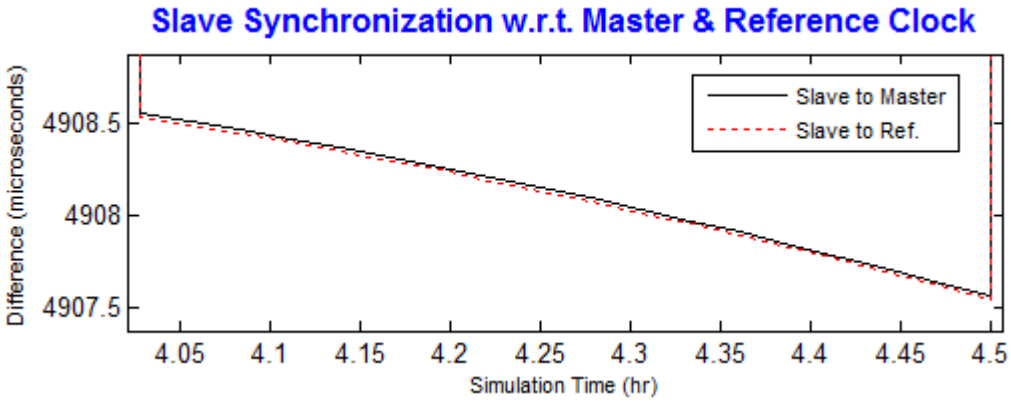

Figure 36: Clock Synchronization - Temporary Network Outage during Static Traffic with Corrections Applied on Both Clocks - Closer Look at the Holdover Period

Figure 37 and Figure 38 show the slave synchronization results for the case when AOCM correction is not applied on the slave clock.

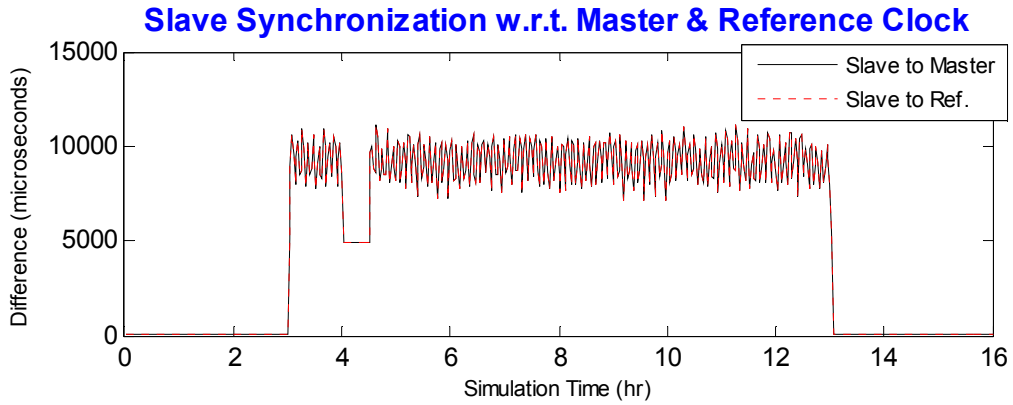

Figure 37: Clock Synchronization - Temporary Network Outage during Static Traffic with no AOCM Corrections on Slave Clock

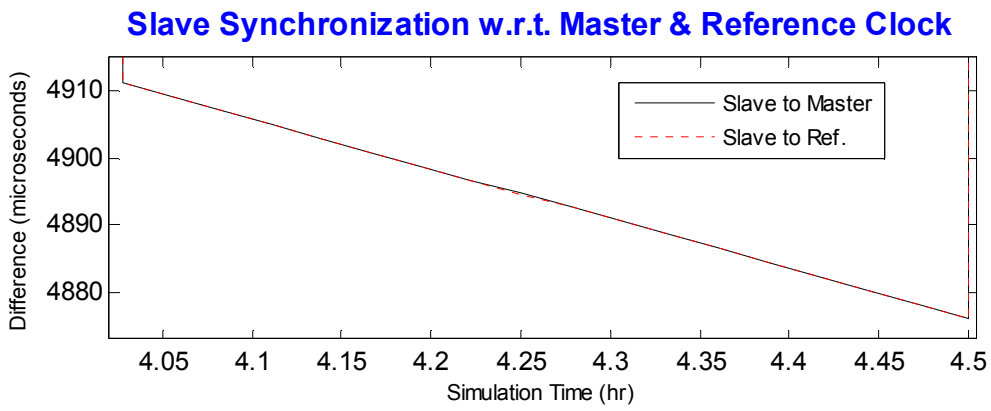

Figure 38: Clock Synchronization - Temporary Network Outage during Static Traffic with no AOCM Corrections on Slave Clock - Closer Look at the Holdover Period

5.3.3.3 Discussion

From Figure 35, the results of slave synchronization are the same as in the previous test cases except during the network outage (4 to 4.5 hours). As shown in Figure 36, the slave clock is stable and drifts up to 1 µs only for the 30 minutes outage. To investigate the sudden drop of slave clock difference to about 5000 µs, we plot all the difference values in Figure 39 instead of plotting maximum difference within a synchronization window (100 s).

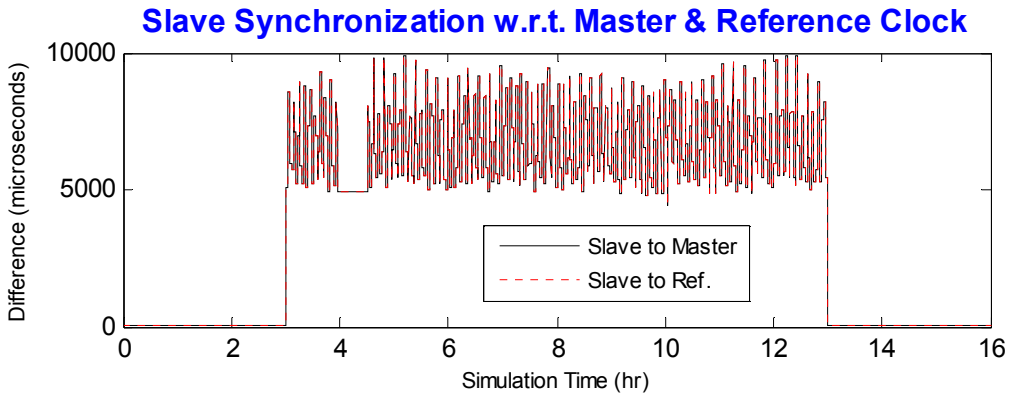

Figure 39: Clock Synchronization - Temporary Network Outage during Static Traffic with Corrections Applied on Both Clocks

It is observed that the 5000 µs difference is within the drifts experienced by the slave clock due to the asymmetric delays. This was not obvious in Figure 35 where the maximum slave difference is taken every 100 s. With the outage happening at 4 hours it just so happens that slave clock has not drifted enough to cause a slave difference of more than 5000 µs. If the outage happens at a different time e.g. from 3.9721 to 4.5 hours and the slave difference is plotted again every 1 second, it is observed that the slave stabilizes around 7500 µs as shown in Figure 40.

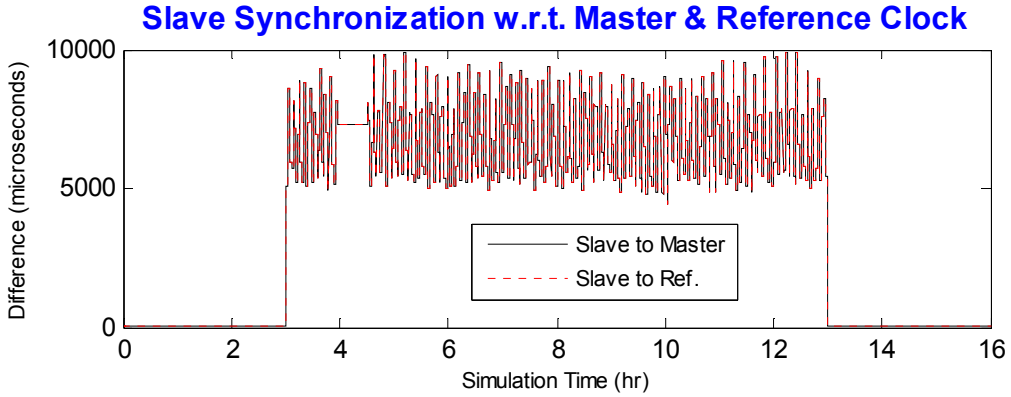

Figure 40: Clock Synchronization - Temporary Network Outage during Static Traffic with Corrections Applied on Both Clocks

To further investigate the latency values just before the network outage happens in Figure 35, the average of absolute delay differences is plotted along with the average of absolute slave clock differences, and is shown in Figure 41.

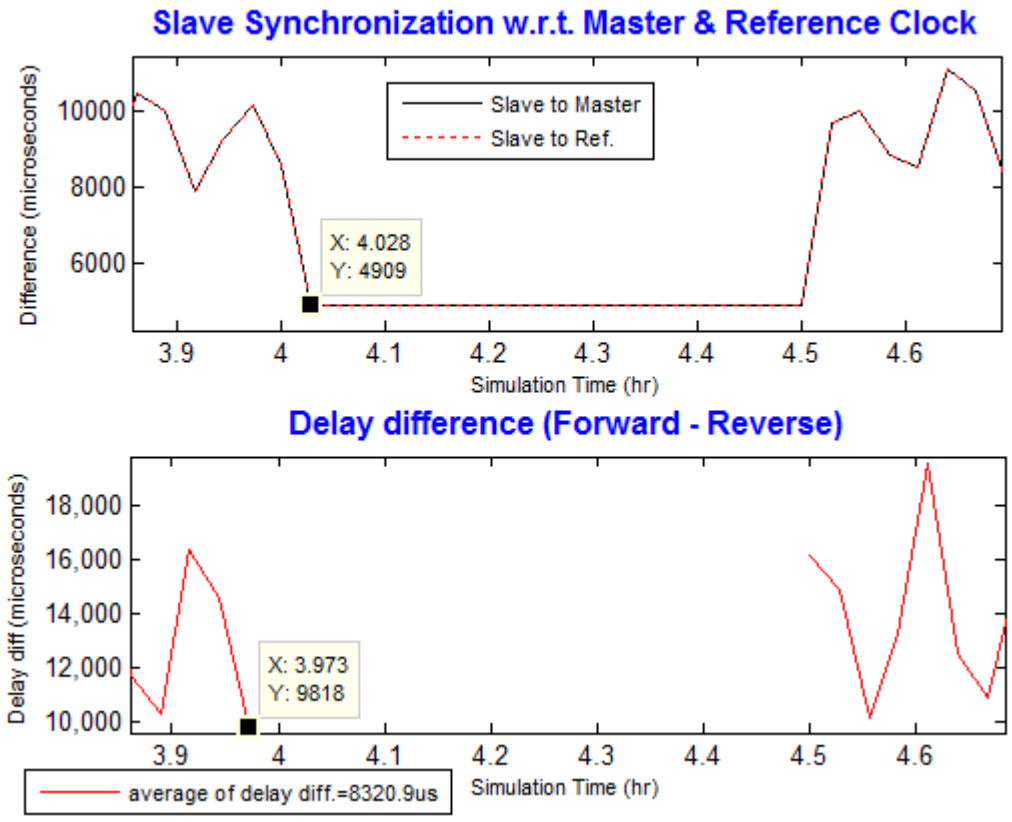

Figure 41: Using the Average of Absolute Delay Differences - Temporary Network Outage during Static Traffic with Corrections Applied on Both Clocks

From Figure 41, it is observed that the asymmetric delays just before the outage starts is 9818 µs and therefore the slave clock difference when the outage starts becomes stable around a value of 4909 µs (i.e. half of asymmetric delays, in accordance with the mathematical results in the first test case, Section 5.3.1). Hence, in general the value

where the slave gets stable could vary and depends on the asymmetric latencies just before the outage happens.

Comparing this result to the case when no AOCM correction is applied on slave clock, the slave drifts more than 30 µs during the outage period as shown in Figure 37 and Figure 38. The result is surprising as the drift in 30 minute outage should be around 180 µs as observed in a previous test case (Section 5.2.1). To investigate the further, the plots for true clock differences for multiple single runs are plotted as shown in Figure 42. It is observed that depending on the asymmetric latency just before the outage, the slave clock difference could be positive or negative and the drift for a single run is about 180 µs in the positive direction. The average absolute differences plotted earlier are not a true indication of the expected results in this case.

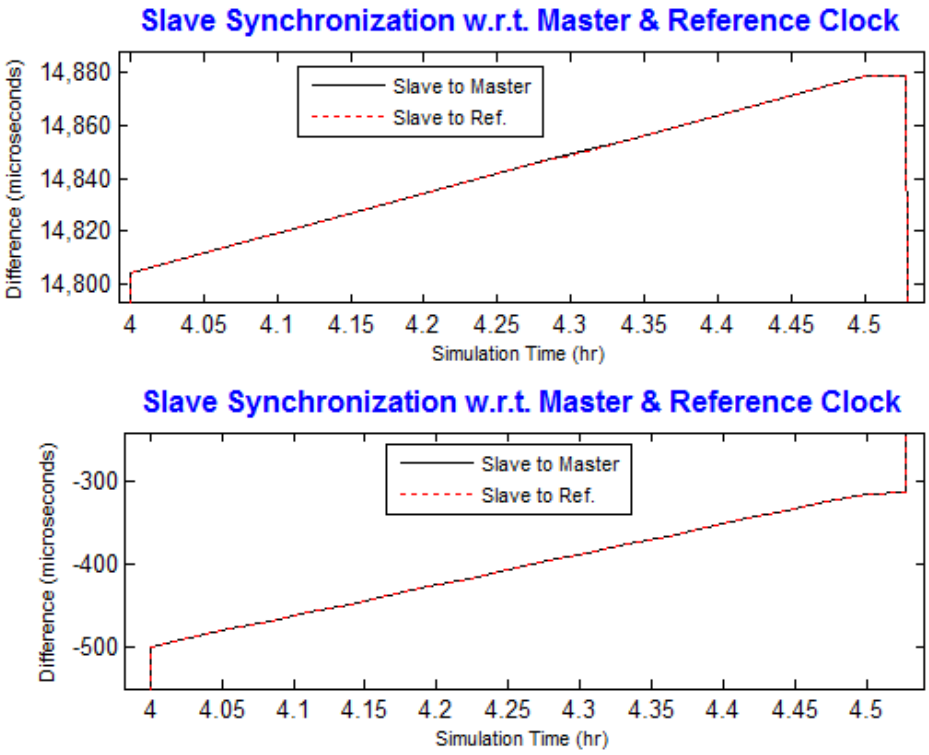

Figure 42: Individual Runs using True Clock Differences - Temporary Network Outage during Static Traffic with no AOCM Corrections on Slave Clock

Hence, the inclusion of adaptive correction method on a slave clock with traffic in the network results in an increased stability of the slave clock during the outage.

5.3.4 Summary

The test cases in this section and Sub-Appendix C.2 have been run with traffic using difference load profiles and varying conditions on the master and the slave clocks. The adaptive correction model has been considered on both master and slave clocks. In a nutshell, the following points are observed.

- The slave clock difference is one half of the asymmetric latencies experienced due to traffic. The shape of the asymmetric latencies and hence the slave clock difference follows the traffic profile used.
- When an asymmetric traffic (e.g. 40% forward, 30% reverse) is increased to 100% in both directions to cause network congestion, the slave accuracy improves because the asymmetric latencies are relatively reduced.
- When there is a network outage and the AOCM corrections are applied on the slave clock, the stability of the slave clock improves compared to the case when the AOCM correction is not applied. However the slave synchronization accuracy remains poor.

5.4 Test Cases to Examine the Effect of Miscellaneous Parameters

The test cases presented in this section study the direct effect of various parameters on the clock synchronization. The parameters considered are master clock rate, slave clock

rate, IEEE 1588 synchronization frequency, master AOCM training period and slave AOCM training period. The master and slave clocks are connected as shown earlier in Figure 24 using the parameters mentioned in the simulation setup section (Section 5.1.). Also no traffic is considered in the network. For each parameter, the test case is executed 50 times for all the values under consideration. The average values of the clock inaccuracy are calculated. In case of the slave clock, the average values for the maximum drift in each synchronization window are calculated. The values calculated are plotted using a bar chart. Here, only a few test cases are presented. The additional test cases are discussed in Sub-Appendix C.3.

5.4.1 IEEE 1588 Synchronization Frequency

5.4.1.1 Description

In a first test case, a real slave clock is connected to a perfect master clock and the IEEE 1588 synchronization frequency is varied using the values of 1, 5, 10, 100, 1000 and 10000 seconds. Here the slave clock is running at a rate of 100 ppb. The test case is run first without AOCM applied on the slave clock and is then repeated with the AOCM corrections enabled on the slave clock. The effect on the slave clock synchronization w.r.t. the master clock is observed in both cases versus the varying synchronization frequency.

5.4.1.2 Results

The average values for maximum slave drifts in each synchronization window are calculated to determine the slave clock inaccuracy w.r.t. the master clock and plotted

using a bar chart both with no AOCM and with AOCM on the slave clock as shown in Figure 43 and Figure 44 respectively.

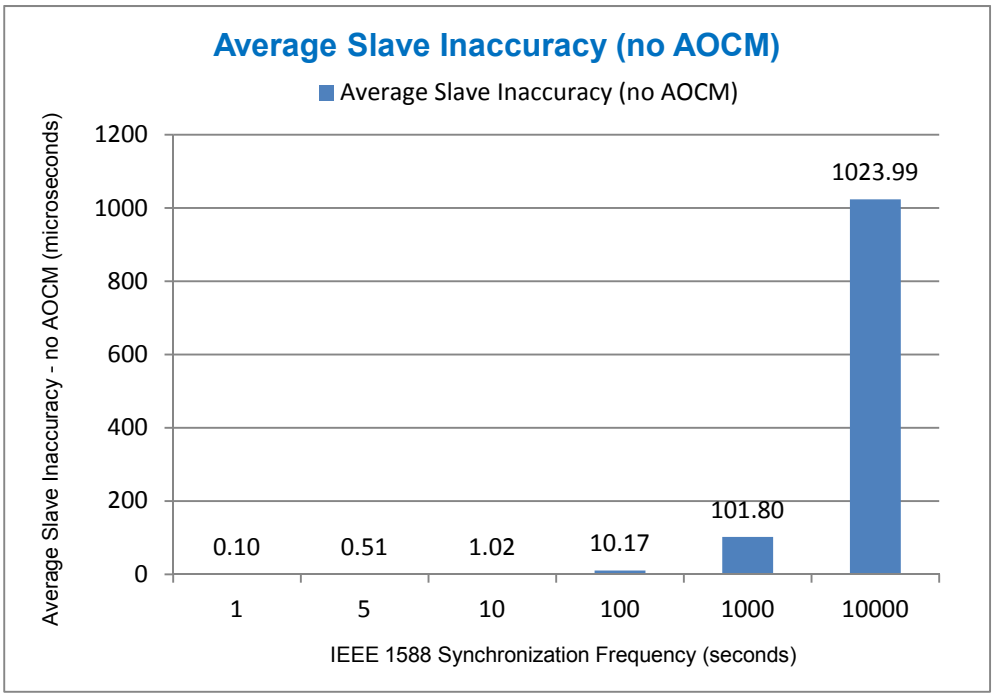

Figure 43: Slave Clock Synchronization with No AOCM - Varying IEEE 1588 Synchronization Frequency

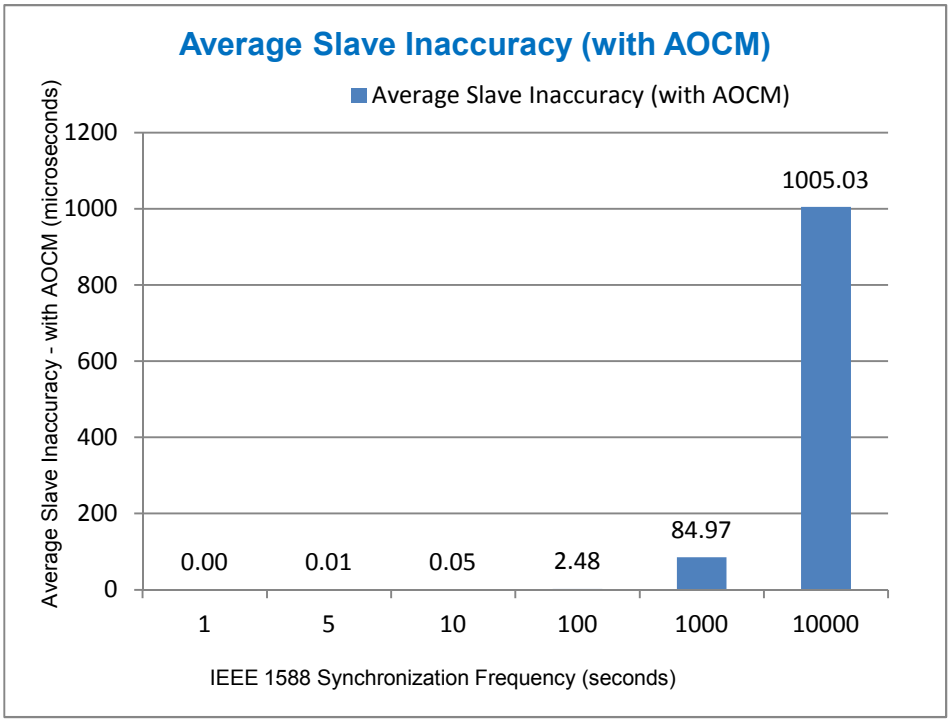

Figure 44: Slave Clock Synchronization with AOCM enabled on the Slave - Varying IEEE 1588 Synchronization Frequency

5.4.1.3 Discussion

From Figure 43 and Figure 44, the effect of varying synchronization frequency values can be seen on the slave clock accuracy both without AOCM and with AOCM applied respectively. A higher synchronization frequency (lower time value) has two advantages. Firstly, the slave clock receives updates from the master clock more frequently. Secondly, with AOCM enabled, the higher synchronization frequency means the slave clock has more updates to use in the training model (for the same training period of 8 hours) and hence improving the slave accuracy. It should be noted that if the synchronization frequency is high (i.e. 1 s), then the IEEE 1588 itself alone is enough to

provide a good clock synchronization. Nevertheless, applying the AOCM does improve the slave accuracy significantly and that can be seen for the synchronization frequency values of 1 s, 10 s and 100 s. A higher synchronization frequency also means more traffic in the network. So an optimal synchronization frequency should be found where the effect of AOCM can be realized at maximum with minimum load on the network. On the other hand, if the synchronization updates are received less frequently (i.e. 1000 s or 10000), then the improvement achieved using the AOCM is not that significant. This is due to the reason that the adaptive model does not have enough training data to achieve higher synchronization accuracy.

The effect of the slave AOCM training period is considered in the next test case (Section 5.4.2) where it will also be determined how many IEEE 1588 updates are required by the slave clock to achieve 1 μs accuracy with a confidence level of 95%.

5.4.2 Slave AOCM Training and Holdover Periods

5.4.2.1 Description

In a second test case, we have a real slave connected to a perfect master as in Figure 24 using the parameters as described earlier in the simulation setup section (Section 5.1) and the AOCM model enabled on the slave clock. The slave AOCM training period is varied while the holdover period of 100 hrs is fixed. The training periods used are 1, 2, 5, 10 and 20 hours resulting in total simulation periods of 101, 102, 105, 110 and 120 hours respectively. The effect of the training period variation on the slave clock synchronization is examined in this test case. In addition, for a training period of 20 hours, the slave inaccuracy is observed for various holdover periods of 1, 5, 10, 20, 30,

50 and 100 hours. It is also determined how many IEEE 1588 synchronization updates from master to slave clock are needed to achieve 1µs slave accuracy with 95% confidence.

5.4.2.2 Results

Figure 45 and Figure 46 show the average slave inaccuracy values for training and holdover periods respectively while Figure 47 shows the average values for various holdover periods.

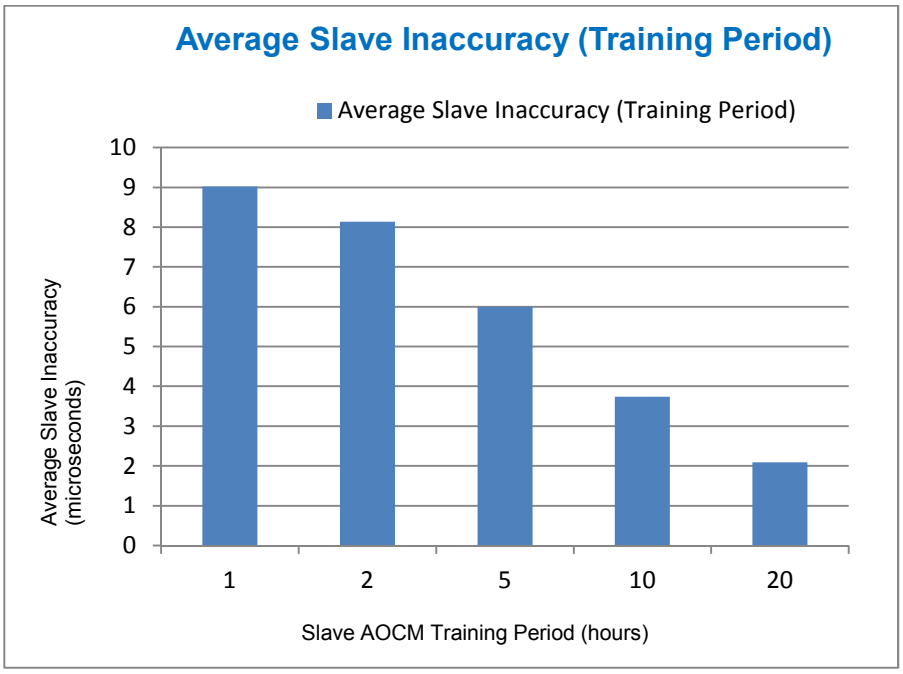

Figure 45: Slave Clock Synchronization in Training Period - Varying Slave AOCM Training Period

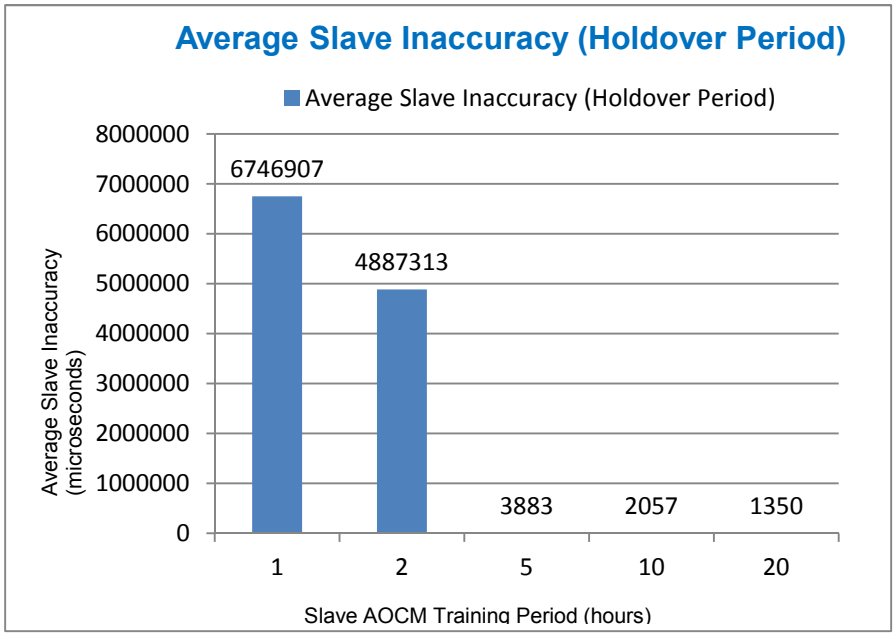

Figure 46: Slave Clock Synchronization in Holdover Period - Varying Slave AOCM Training Period

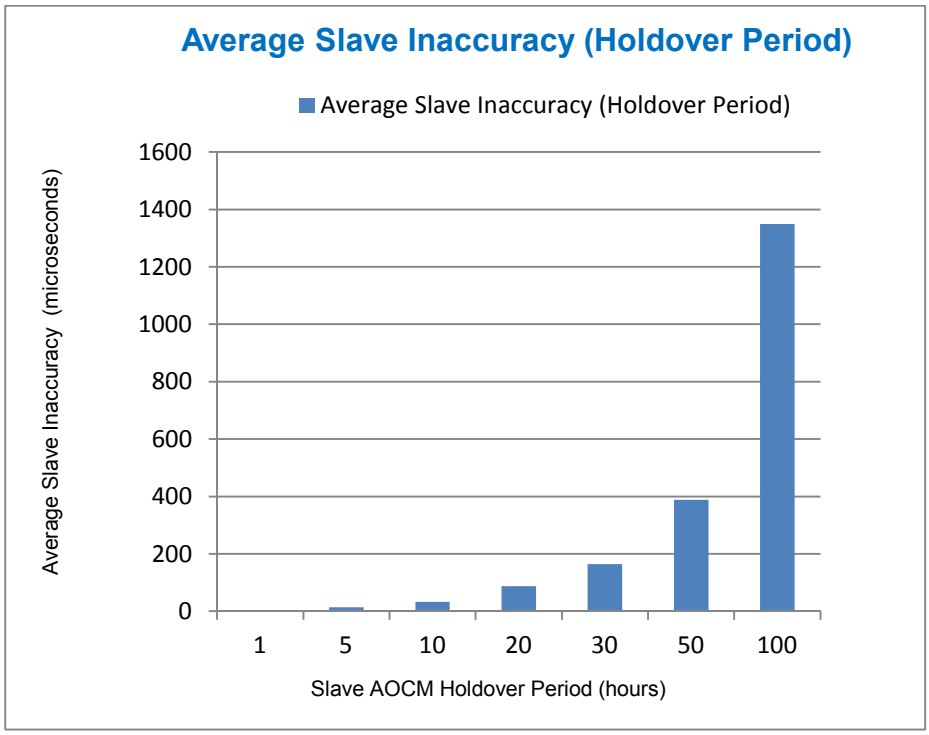

Figure 47: Slave Clock Synchronization - Varying Slave AOCM Holdover Period

5.4.2.3 Discussion

From Figure 45, it can be seen that the slave accuracy during the training period improves with increasing training periods. From Figure 46, it is observed the slave inaccuracy in holdover period is also reduced greatly with longer training periods where the holdover period is fixed at 100 hours for all cases. The longer training period ensures more samples contribute to the AOCM training model, producing better results in the holdover period.

From Figure 47, the results for varying holdover periods as expected show that the slave clock differences grow with longer holdover periods. The slave inaccuracy also has a dominant ageing effect for longer simulation periods. However the slave clock drifts up to 1350 μs and the master clock drifts up to 15 μs for the same 100 hr holdover period. The reason for this big difference is that the slave clock is running at 100 ppb but the master clock is running at the rate of the reference clock and only has the temperature and ageing effects. Hence the master clock drifts much less than the slave clock.

It was determined that after 306 IEEE synchronization updates from master to slave clock (i.e. Training period = 306x100 seconds = 8.5 hours), the slave to master accuracy is about 1 μs. To determine whether this is true in general, we collected two sets of data for this training period, and calculated the 95% confidence interval, as shown in Table 4. The confidence interval of Set 1 overlaps with the confidence interval of Set 2 such that the average of Set 1 belongs to Set 2 CI and the average of Set 2 belongs to Set 1 CI. Hence the difference in the results is statistically insignificant and we conclude that with 95% confidence that after 306 time stamp updates, the slave clock achieves an accuracy of 1 μs.

Table 4: Slave Clock Accuracy of 1 µs after 306 Synchronization Updates

	Set 1	**Set 2**
Average	1.000612247 µs	0.998811083 µs
Standard Deviation	0.01838711	0.018317667
Sample Size	50	50
Interval	0.005226663 µs	0.005206924 µs
Confidence Interval	0.995386 to 1.005839 µs	0.993604 to 1.004018 µs

The result can be more generally used to determine how much training is needed for a slave clock. If we know the clock synchronization frequency is f, then the training period should be approximately (f x 306) seconds to achieve an accuracy of 1 µs with 95% confidence. It should be noted that the number of updates (306) obtained here depends on the simulation parameters used such as the network topology, the clock rate, the temperature profile etc. Nevertheless similar calculations can be done for a different set of simulation parameters to predict the optimal number of synchronization updates. On a final point, these calculations are done using a no traffic network. For networks with traffic, the delays could be highly asymmetric depending on the traffic profile. The performance of IEEE 1588 and AOCM, in this case, will not be same as explained earlier in Section 5.3.

5.4.3 Summary

The test cases in this section and Sub-Appendix C.3 have been run to observe the effect of various parameters on the master and slave clock synchronization. In summary, the following points are observed.

- The more the master clock rates deviate (positively or negatively) from the reference clock, the worse is the master accuracy w.r.t the reference clock in holdover mode on average and vice-versa.
- The more the slave clock rates deviate (positively or negatively) from the master clock, the worse is the slave accuracy w.r.t the master clock on average and vice-versa.
- The higher the IEEE 1588 synchronization frequency, the higher is the slave accuracy w.r.t the master clock and vice-versa.
- Longer training periods both for master and slave clocks produce better accuracy results for both clocks.
- The master and slave inaccuracy increases as the holdover period increases.
- The slave to master synchronization accuracy of 1 μs is achieved after 306 synchronization updates from master to slave clock with 95 % confidence.

6 Chapter: Conclusions and Future Work

6.1 Conclusions

The conclusions of the thesis work are presented here.

- The thesis proposes a solution to master-slave synchronization problem that combines the IEEE 1588 synchronization protocol and the adaptive oscillator correction method. The purpose of the adaptive oscillator correction method is to train a clock locked to a GPS signal and apply that training model to correct the clock in holdover mode (i.e. when the GPS signal is lost), hence improving the oscillator accuracy and stability in holdover mode. The first step of the thesis is to apply the corrected oscillator method to a simulated master clock which has a GPS signal reference. The next step extends the adaptive oscillator correction method to the slave clock, using the IEEE 1588 synchronization updates it receives from its master clock. This model is applied to the slave clock during the waiting period between the synchronization updates and when the slave stops receiving updates from the master due to network outage or temporary failure of the master clock itself.

- A clock agent is implemented in the NS-2 simulator to simulate a real clock. Environmental changes are also implemented such as changes in temperature and ageing affects of the clock. These changes cause the clocks to drift.

- The proposed solution is evaluated by implementing it in NS-2. Test cases are also implemented based on an ITU document covering various network conditions and network loads.

- The NS-2 results indicate that our proposed solution improves the slave accuracy up to 10 times for no traffic networks, depending on the length of the slave clock training period.
- The solution results in traffic networks indicate that the slave accuracy is affected by the asymmetric delays such that the degree of accuracy is one half of the asymmetric latencies.
- In case of network congestion, the slave accuracy improves because the delays are less asymmetric.
- In a traffic scenario, when there is an outage, the proposed solution only improves the slave clock stability.

6.2 Future Work

Some suggestions are presented here to enhance the thesis work in the future.

- The solution uses a basic IEEE 1588 message exchange structure to calculate the slave clock offset from the master clock, and applies it directly to the time stamp of the slave clock. This in essence only corrects the phase of the slave clock. This technique can be enhanced to correct the frequency of the slave clock as well. This can be done by enhancing the IEEE 1588 clock synchronization model. One suggestion for this is to determine the frequency adjustment by continuously monitoring the calculated offsets, and computing the correctional rate from the slope of the most recent set of calculated offsets. Another proposal is to reuse the

frequency adjustment work done in the Clock Sampling Mutual Network Synchronization (CS-MNS) protocol [23].

- The IEEE 1588 protocol can also be enhanced by accounting for the asymmetric latencies observed in traffic networks. One proposal is to dynamically exchange a large number of messages between the master and slave clocks within every synchronization window and use the sample with the lowest RTT (Round Trip Time) for the slave offset calculation. The smallest measured RTT will reduce the impact of latency asymmetry. As the error is bound by ½ RTT, the maximum error is reduced as well. Another suggestion is to use the IEEE 1588 transparent clocks as mentioned in IEEE 1588 standard [9].
- The use of hardware time stamping for the synchronization updates can also help reduce the operating system latency. Hardware time stamping means time stamping the packets at the interface between the physical and data link layers. [24]
- The performance of the adaptive oscillator correction model can further be evaluated by comparing the estimated model parameters ($\hat{a}$, $\hat{b}$, $\hat{c}$ and $\hat{d}$) with the known model parameters (a, b, c and d) under varying network conditions and traffic scenarios.

Appendices

Appendix A: NS-2 Clock Agent TCL Commands

The details from creating a clock agent to configuring the parameters of clock agent in a TCL script are presented in this appendix. To quickly start using the clock agent using the default values for the parameters, refer to Sub-Appendix A.1 for Creating a Clock Agent, Sub-Appendix A.15 for Attaching a Clock Agent to a node, Sub-Appendix A.16 for Starting and Stopping a Clock Agent and Sub-Appendix A.17 for Capturing time stamp passed from ns2 C++ code to ns2 TCL code.

A.1 Creating a Clock Agent

Syntax:

To create a clock agent c0

```
set c0 [new Agent/Clock]
```

A.2 Clock rate

The parameter rate represents the natural rate of the clock with respect to the ns2 simulation time (also referred to here as reference time). For example if rate = 1 it means the clock is running at the same rate as simulation time, rate = 1.5 means the clock is running 1.5 times faster than simulation time, rate = 0.5 means the clock is running 0.5 times the simulation time. The parameter rate is adjusted if the ageing effect of the oscillator is enabled (Refer to Sub-Appendix A.12 for more info).

Syntax:

```
$c0 set rate 1          Clock running at the same rate as reference time
```

$c5 set rate 1.0000001 Clock 100 ppb (parts per billion) faster than reference time

Default:

The default value of rate is 1.

A.3 Clock initial offset

The parameter offset represents the initial offset of a clock i.e. when a clock starts ticking it starts from this value. For example if offset is set to 100, it means the clock start ticking from 100s.

Syntax:

$c0 set offset 1 Sets the offset as 1s

$c0 set offset 20 Sets the offset as 20s

Unit:

The unit of offset is seconds w.r.t the corresponding clock.

Default:

The default value of offset is 0. The offset must be greater than or equal to zero and can be a double value.

A.4 Querying time stamp value

The parameter timeToDisplayInfo represents how often the clock time stamp value is queried. The value of the time stamp is passed from ns2 C++ code to TCL where it can be outputted to screen or captured in a text file (More info in Sub-Appendix A.17). Every time the clock time stamp is queried, the simulation time elapsed between the last query and current query is calculated. The time stamp of the clock is returned based on the clock rate and elapsed time. Here the clock rate includes the “natural rate” of the clock

(i.e. the parameter rate which also includes the ageing effect, refer to Sub-Appendix A.2 and A.12) as well as the correction rate as calculated by the adaptive oscillator correction algorithm (Refer to Sub-Appendix A.14). The drift due to temperature effects is also added to the returned time stamp. (Refer to Sub-Appendix A.11).

Syntax:

$c0 set timeToDisplayInfo 1 Time stamp information is captured every 1 simulation second

Unit:

The unit of timeToDisplayInfo is simulation time seconds.

Default:

The default value of timeToDisplayInfo is 100. The parameter timeToDisplayInfo must be a positive integer i.e. multiple of seconds.

A.5 masterClock

The parameter masterClock indicates whether the clock is a master clock or a slave clock. A value of 1 means a master clock and 0 means a slave clock. If there is a slave clock in the network, we must also have a master clock which would be used to synchronize the slave clock time stamp using the IEEE 1588 synchronization protocol. The address and port of the master clock to be used by the slave clock to request 1588 messages must be set as explained in Sub-Appendix A.6. Both master and slave clocks have the self-correcting adaptive oscillator correction algorithms implemented to counter the effects of ageing and temperature effects. The master correction method as described in Section 3.3 is implemented. The slave correction method is implemented using the proposed solution

(see Section 4.1). The NS-2 TCL configurations to enable these algorithms are given in Sub-Appendix A.14.

Syntax:

```
$c0 set masterClock 1
```

Default:

The default value of masterClock is 1 i.e. a master clock.

A.6 masterAddr and masterPort

For using the IEEE 1588 synchronization protocol, a slave clock can be configured to receive updates from a master clock. The address and port of the master clock is configured at the slave clock so that it can request IEEE 1588 sync messages.

Syntax:

To set a node represented by address 0 and port 0 to be the master node of a slave node with clock c5, the syntax given below is used.

```
$c5 set masterAddr 0
$c5 set masterPort 0
```

Default:

Both masterAddr and masterPort have default values of -1.

Note:

The correct values must be set before using a slave clock. The first node created in the TCL script has the default address of 0 and the default port of 0. Therefore it is recommended to use the first node as the master node so that the parameters masterAddr and masterPort of the slave node can both be set to 0.

A.7 timeStampReqFreq

The parameter timeStampReqFreq indicates how often the slave clock requests the timestamp from the master clock for doing IEEE 1588 synchronization.

Syntax:

$c5 set timeStampReqFreq 100 Master clock sends sync messages every 100 seconds (as per its own rate)

Unit:

The unit of timeStampReqFreq is seconds w.r.t. the corresponding master clock.

Default:

The default value of timeStampReqFreq is 100. The parameter timeStampReqFreq must be a positive integer i.e. multiple of seconds. If the AOCM method is enabled, the parameter timeStampReqFreq must also be a positive integer multiple of the parameter RLSFreq (see Section A.14) .

A.8 enable1588Logs

The parameter enable1588Logs is used to enable or disable IEEE 1588 protocol logs outputted on the screen. If set to 1, the logs are enabled. If set to 0, the logs are disabled. If enabled the logs are outputted to the standard output such as console.

Syntax:

To enable logs for a master node 0 represented by clock c0 and slave node 5 represented by clock c5, the following is the syntax.

```
$c0 set enable1588Logs 1
$c5 set enable1588Logs 1
```

A snapshot of the logs outputted to the screen for the above example is given next:

```
Node: 5 ***INIT Message (packet rate) sent by slave clock @ Ref. time = 1s
Node: 0 ***INIT Message received by master clock @ Ref. time = 1.0025605s
Node: 0 ***SYNC Message sent by master clock @ Ref. time = 1.0025605s
Node: 5 ***SYNC Message received by slave clock @ Ref. time = 1.005121s
        master to slave delay: 0.00256049999999952
Node: 0 ***Delay Request Message received by master clock @ Ref. time = 1.0076815s
        slave to master delay: 0.00256049999999952
Node: 5 ***Delay Response Message received by slave clock @ Ref. time = 1.010242s
        master to slave delay: 0.00256049999999952
        offset = 5.08937558763023e-10
        mean delay = 0.00256049999999952
        old timestamp = 1.00512100050894
        new timestamp = 1.01024200050894
```

The highlighted logs are outputted from the slave side whereas the logs not highlighted are outputted from the master side. Note the INIT message logs are outputted only once in the beginning when the packet rate is sent by slave clock and then received by master clock.

Default:

The default value of enable1588Logs is 0 i.e. logs are disabled.

A.9 enable1588Delays

The parameter enable1588Delays is used to enable or disable the recording of actual forward (master to slave) and reverse (slave to master) delays into a text file observed during each IEEE 1588 synchronization event. The delays are logged into a text file called “delays_x.txt” where x is the node number corresponding to the clock. If the file "delays_x.txt" already exists it will be overwritten by the new one. The unit of the delays is simulation seconds. The parameter is valid for a slave clock only.

Syntax:

To enable delays for a slave node 5 represented by clock c5

$c5 set enable1588Delays 1

As a result a text file "delays_5.txt" will be generated. A snapshot of the file is given in Figure 48:

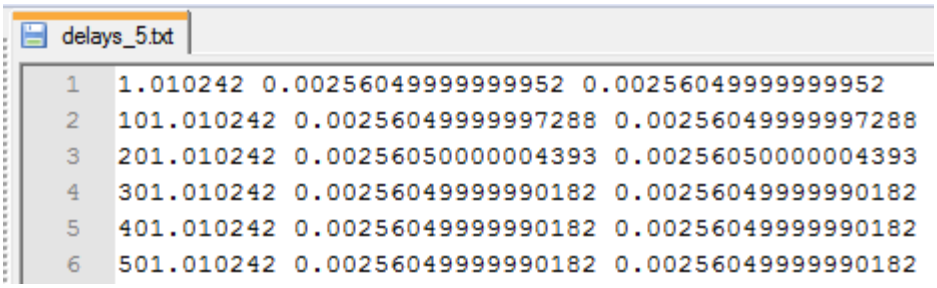

Figure 48: A snapshot of delays_5.txt file

The first column represents simulation time, the 2nd column represents forward delay and the 3rd column represents reverse delay. Each row represents one complete synchronization event.

Default:

The default value of enable1588Delays is 0 i.e. output of delays is disabled.

A.10 tempProfileName

The parameter tempProfileName is used to indicate the text file to be used for reading the temperature profile coefficients. The temperature profile indicates the values of temperature to be used over time. A value of -1 indicates no temperature profile selected, hence no temperature effects. Any other number e.g. 1 means the text file "TempProfile1.txt" would be read for temperature profile coefficients.

Syntax:

$c0 set tempProfileName 1 Text file "TempProfile1.txt" is used for temperature profile

A typical temperature profile text file is shown in Figure 49.

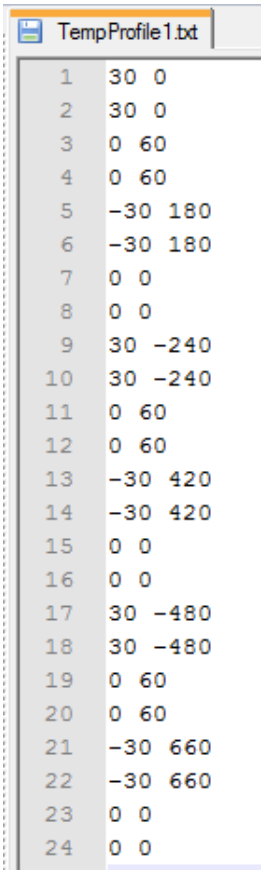

Figure 49: Temperature profile text file "TempProfile1.txt"

Each line in the text file represents coefficients a and b for each hour. Temperature and time are related using linear equation $y = ax + b$, where y is temperature and x is time. The temperature profile will reuse the coefficients from the beginning after reaching the last set of coefficients. The Figure 49 temperature profile coefficients results in a temperature profile graph shown in Figure 50.

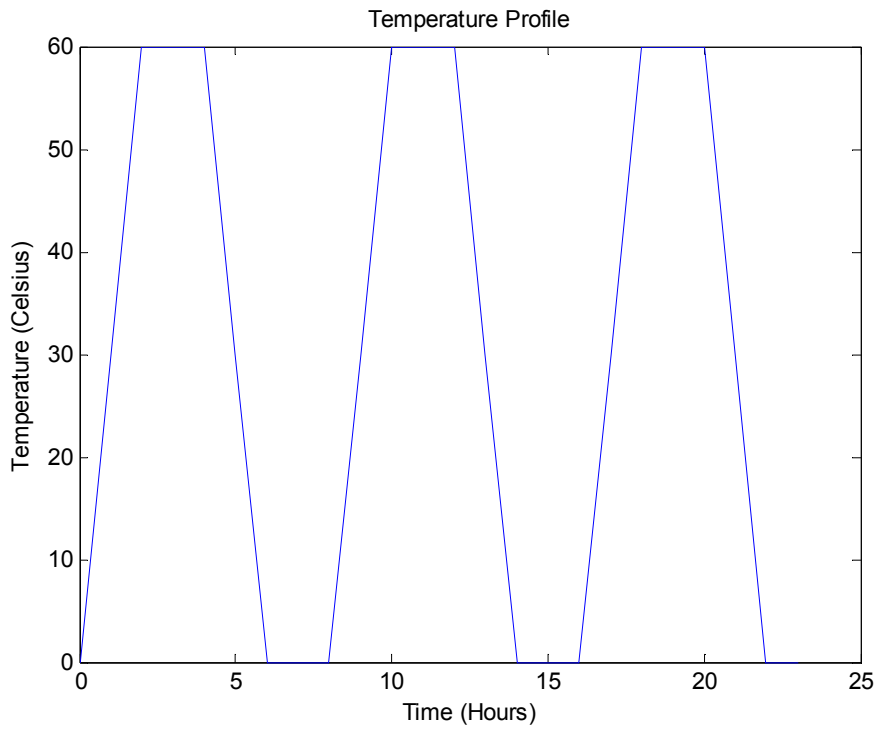

Figure 50: Temperature Profile using coefficients from "TempProfile1.txt"

Default:

The default value of tempProfileName is -1 i.e. no temperature profile provided, therefore no temperature effects on the clock oscillator.

A.11 Temperature Effect

The three parameters temp_ppb, temp_degreesC and temp_quad_term together define the temperature effects on the clock oscillator. The linear temperature effect is represented by temp_ppb and temp_degreesC e.g. if temp_ppb = 4 and temp_degreesC = 75, it means a linear temperature effect of 4ppb/75C. To model a quadratic temperature effect, a quadratic term temp_quad_term is included e.g. temp_quad_term = -0.00031966 ppb/C2

Syntax:

```
$c0 set temp_ppb 5
$c0 set temp_degreesC 80
$c0 set temp_quad_term -0.00031966
```

Whenever the time stamp is queried, the drift due to temperature is calculated for the time elapsed since last query. The elapsed interval is divided into sub-intervals, each represented by the parameter driftInterval (More info in Sub-Appendix A.13). For each sub-interval, the temperature at its middle (midTemp) is obtained using the assigned temperature profile and the drift calculated as below:

$$drift = (((temp\_ppb/temp\_degreesC) * midTemp) + (temp\_quad\_term * midTemp * midTemp)) * 1e^{-9} * driftInterval$$

Unit:

The unit of temp_ppb is ppb (parts per billion). The unit of temp_degreesC is C (Celsius). The unit of temp_quad_term is ppb/C^2.

Default:

The default values are temp_ppb = 4, temp_degreesC = 75, temp_quad_term = -0.00031966. The parameter temp_ppb must be an integer. temp_degreesC can be a double value excluding zero. temp_quad_term can be a double value.

A.12 Ageing Effect

The parameters ageing, ageing_ppb and ageing_days together define the ageing effects on the clock oscillator. To enable the ageing effect, the parameter ageing must be set to 1, and to disable ageing, it is to be set to 0. Only a linear ageing effect is considered and is represented by the parameters ageing_ppb and ageing_days e.g. if ageing_ppb = 1 and ageing_days = 1, it means a linear ageing effect of 1ppb/day.

Syntax:

```
$c0 set ageing 1
$c0 set ageing_ppb 1
$c0 set ageing_days 1
```

Whenever the time stamp is queried, the change in the natural rate of the clock due to ageing is calculated for the time elapsed since last query. The elapsed interval is subdivided into sub-intervals, each represented by the parameter driftInterval (More info in Sub-Appendix A.13). For each sub-interval, the new natural rate of the clock is obtained as follows:

$$rate = rate + ((ageing\_ppb * 1e^{-9})/(ageing\_days * 24 * 3600)) * driftInterval$$

Unit:

The unit of ageing_ppb is ppb (parts per billion). The unit of ageing_days is days.

Default:

The default values are ageing = 0, ageing_ppb = 1, ageing_days = 1

The parameter ageing_ppb must be an integer. ageing_days must be a positive number and can be a double value.

A.13 driftInterval

The parameter driftInterval represents a sub-interval used for temperature and ageing effects calculations. For simplicity, use the same value as for timeToDisplayInfo (Sub-Appendix A.4).

Syntax:

```
$c0 set driftInterval 1
```

Unit:

The unit of driftInterval is simulation seconds.

Default:

The default value of driftInterval is 1. The parameter driftInterval must be a positive integer i.e. multiple of seconds.

A.14 Adaptive Oscillator Correction Method Parameters

enableRLS, gpsSignal and RLSFreq

In the adaptive oscillator correction method (AOCM) aka RLS algorithm, to allow the clock to have a locked mode and a holdover mode feature, the parameter gpsSignal must be set to 1. The interval for the holdover period is determined by the parameters holdoverStart and holdoverEnd explained later in this appendix. The remaining interval represents a training (locked) period. To disable such a feature, set gpsSignal to 0. In addition if the purpose is to apply AOCM corrections to the clock rate in holdover mode, set enableRLS to 1, 0 to disable AOCM corrections in holdover mode. The parameter RLSFreq determines how often the AOCM algorithm is run. The AOCM algorithm can be applied to both master and slave clocks. For the master clock, the AOCM algorithm tries to synchronize it with the reference time i.e. simulation time. For the slave clock, the AOCM algorithm tries to synchronize it with the master clock time.

Syntax:

```
$c0 set gpsSignal 1
$c0 set enableRLS 1
$c0 set RLSFreq 1
```

Unit:

The unit of RLSFreq is simulation time seconds and must be a positive integer multiple of the parameter timeToDisplayInfo (Refer to Sub-Appendix A.4).

Default:

The default values are gpsSignal = 0, enableRLS = 0, RLSFreq = 100.

holdoverStart and holdoverEnd

The holdover period start and end are indicated by the parameters holdoverStart and holdoverEnd. For the master clock, a holdover period means that it no longer receives updates from GPS (or whatever reference clock it is tuned to). For a slave clock, a holdover period means it no longer receives IEEE 1588 synchronization updates from its master clock. Any period which is not a holdover period is the training period (i.e. locked mode). Only one holdover period is supported.

Syntax:

$c0 set holdoverStart 8	Holdover mode starts at 8hrs
$c0 set holdoverEnd 20	Holdover mode ends at 20hrs

Unit:

The unit of the parameters holdoverStart and holdoverEnd is hours.

Default:

The default values are holdoverStart = 10 and holdoverEnd = 52. Double values can be used here.

outputMaxCTE

To log maximum CTE (Cumulative Time Error in microseconds) for each simulation run to a text file "maxCTE_x.txt", set the parameter outputMaxCTE to 1, to disable set it to 0,

where x is the node number for the corresponding clock. If the file "maxCTE_x.txt" already exists, the CTE is appended to it otherwise it is generated. One value is appended for each simulation run.

Syntax:

To enable output of max CTE for a node 0 represented by clock c0

```
$c0 set outputMaxCTE 1
```

As a result max CTE is appended to a text file "maxCTE_0.txt". A snapshot of the file is given for 3 runs in Figure 51.

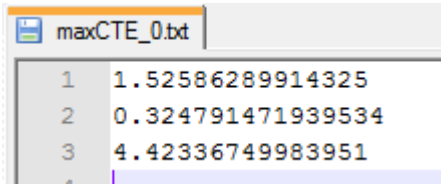

Figure 51: A snapshot of maxCTE_0.txt file

Default:

The default value of outputMaxCTE is 0

outputRLSCorrections

To output all the AOCM rate corrections for a single run to a text file "corrections_x.txt", set the parameter outputRLSCorrections to 1, to disable set it to 0. Again, x is the node number corresponding to the clock. If the file "corrections_x.txt" already exists it will be overwritten by the new one. The unit of the corrections is ppb.

Syntax:

To enable output of AOCM corrections for a node 0 represented by clock c0

```
$c0 set outputRLSCorrections 1
```

As a result a text file "corrections_0.txt" is generated. A snapshot of the file is given in Figure 52.

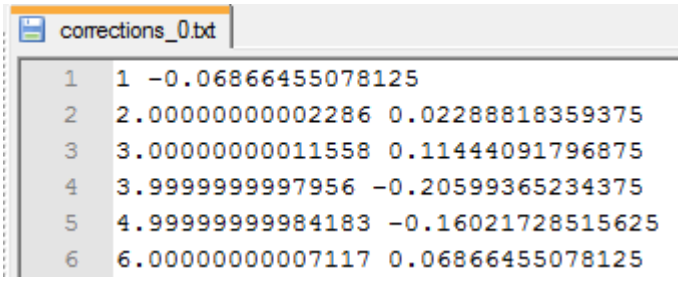

Figure 52: A snapshot of corrections_0.txt file

The first column represents the time stamp of the node and the 2nd column represents the corresponding correction applied to the rate.

Default:

The default value of outputRLSCorrections is 0

A.15 Attaching a Clock Agent to a node

To attach a given clock agent c0 to a given node n0, use the following syntax.

```
$ns attach-agent $n0 $c0
```

A.16 Starting and Stopping a Clock Agent

To start a clock agent, use

```
$ns at 1.0 "$c0 start"
```
Starts at simulation time of 1.0s

To stop a clock agent, use

```
$ns at 100.0 "$c0 stop"
```
Stop at simulation time of 100.0s

A.17 Capturing time stamp passed from ns2 C++ code to ns2 TCL code

To capture the time stamp of the clock, passed from C++ code to TCL script, a procedure timeout must be written in the TCL script. The timeout routine defined in the TCL script

is called by the C++ code. Apart from passing the clock timestamp, the natural rate of the clock and the effective rate (after applying the AOCM correction) are also passed from the C++ code. The timeout procedure is given below.

```
#Define a 'timeout' function for the class 'Agent/Clock'
Agent/Clock instproc timeout {ts rate eff_rate} {
        global ns
        $self instvar node_
        puts "Node [$node_ id] @ $ts\s (rate = $rate, effective_rate = $eff_rate, ref. =
[$ns now]s)."
}
```

The first argument ts is the time stamp, the 2nd argument rate is the natural rate of the clock and the 3rd argument eff_rate is the effective rate of the clock, all passed from ns2 C++ code. Here these values could be used to output to the screen or capture in a text file. The above routine shows how to output to the screen.

To capture the time stamps of a master clock (indicated by node 0) and a slave clock into two separate text files, the following TCL code can be used in the script.

```
set dataMaster [open master.txt w]
set dataSlave [open slave.txt w]
#Define a 'timeout' function for the class 'Agent/Clock'
Agent/Clock instproc timeout {ts rate eff_rate} {
        global ns dataMaster dataSlave
        $self instvar node_
        variable nodeID [$node_ id]
        if {$nodeID == 0} {
                puts $dataMaster "$ts [$ns now]"
        } else {
                puts $dataSlave "$ts"
        }
}
```

Appendix B: NS-2 TCL Example – Temperature and Ageing Effects

The NS-2 TCL script presented here uses four nodes n0, n1, n2 and n3 connected as shown in Figure 53. Nodes n0, n1, n2 and n3 have clocks c0, c1, c2 and c3 respectively. All the clocks used have no initial offsets in the rate i.e. initially all tick at the same rate as the reference clock (simulation time). Clock c0 is a perfect clock with no temperature and ageing effects. Clock c1 has ageing effect using the default values but no temperature drifts. Clock c2 has temperature effect using the default values but no ageing drifts. Clock c3 has both temperature and ageing effects using the default values. The TCL script measures how the clock drifts due to the temperature/ageing effects.

Figure 53: Network Topology containing four nodes

NS-2 TCL Script Code

```
set ns [new Simulator]
#Open a trace file
set graphData [open graphData.txt w]

#Define a 'finish' procedure
proc finish {} {
        global ns graphData
        $ns flush-trace
        close $graphData
        exit 0
}

#Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
```

```
set n3 [$ns node]

$ns duplex-link $n0 $n1 1Mb 100ns DropTail
$ns duplex-link $n1 $n2 1Mb 100ns DropTail
$ns duplex-link $n2 $n3 1Mb 100ns DropTail

#Define a 'timeout' function for the class 'Agent/Clock'
Agent/Clock instproc timeout {ts rate eff_rate} {
        global ns graphData
        $self instvar node_
        puts $graphData "[$node_ id] $eff_rate $ts [$ns now]"
}

#Create clock agents
set c0 [new Agent/Clock]
$c0 set offset 1
$c0 set rate 1
$c0 set timeToDisplayInfo 100
$c0 set masterClock 1
$ns attach-agent $n0 $c0

set c1 [new Agent/Clock]
$c1 set offset 1
$c1 set rate 1
$c1 set timeToDisplayInfo 100
$c1 set masterClock 1
$c1 set ageing 1
$c1 set driftInterval 1
$ns attach-agent $n1 $c1

set c2 [new Agent/Clock]
$c2 set offset 1
$c2 set rate 1
$c2 set timeToDisplayInfo 100
$c2 set masterClock 1
$c2 set tempProfileName 1
$c2 set driftInterval 1
$ns attach-agent $n2 $c2

set c3 [new Agent/Clock]
$c3 set offset 1
$c3 set rate 1
$c3 set timeToDisplayInfo 100
$c3 set masterClock 1
$c3 set tempProfileName 1
$c3 set ageing 1
```

```
$c3 set driftInterval 1
$ns attach-agent $n3 $c3

#Schedule events
$ns at 1.0 "$c0 start"
$ns at 1.0 "$c1 start"
$ns at 1.0 "$c2 start"
$ns at 1.0 "$c3 start"
$ns at 86400.05 "$c0 stop"
$ns at 86400.05 "$c1 stop"
$ns at 86400.05 "$c2 stop"
$ns at 86400.05 "$c3 stop"
$ns at 86400.1 "finish"

#Run the simulation
$ns run
```

Appendix C: Additional NS-2 Simulation Results

C.1 Additional Test Cases with No Traffic

In this sub-appendix, some more no traffic test cases not already covered in Section 5.2 are presented.

C.1.1 A Real Slave Clock vs. a Perfect Master Clock

Description

The first test case measures the IEEE 1588 synchronization accuracy of a real slave clock connected to a perfect master clock.

Results

Figure 54 shows the master clock synchronization w.r.t. to the reference clock and the slave clock synchronization w.r.t. to the master clock and the reference clock when a perfect master clock is used.

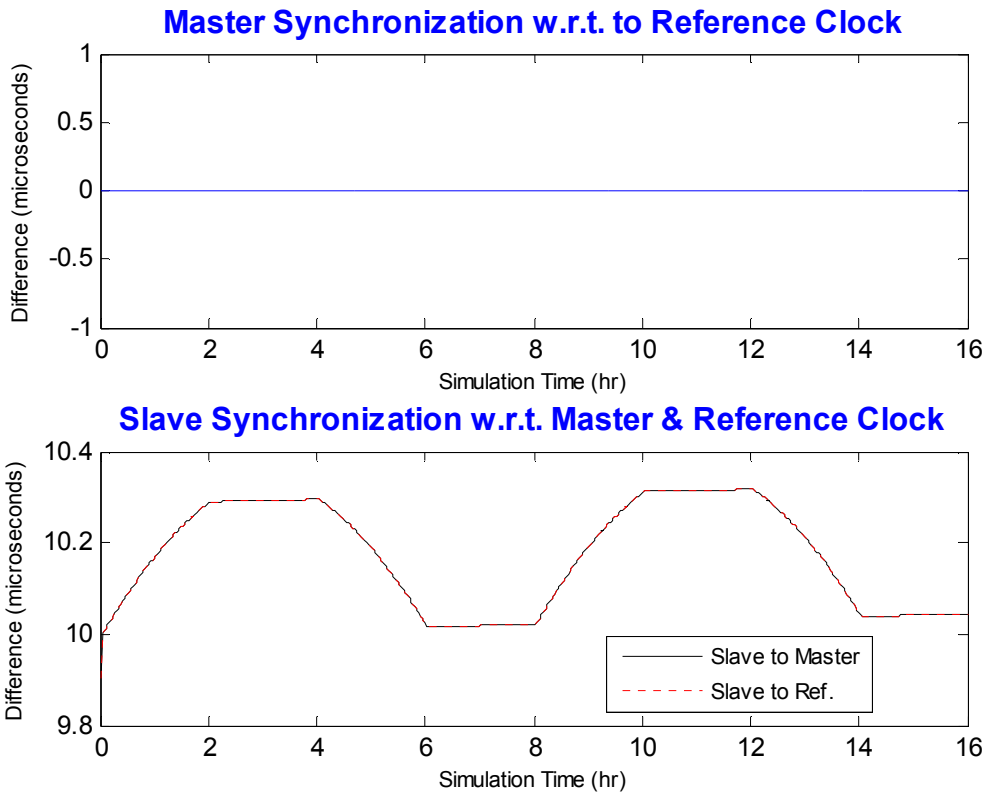

Figure 54: Clock Synchronization – A Real Slave Clock vs. a Perfect Master Clock

Discussion

From Figure 54, the master to reference signal difference is always zero because the master clock used is a perfect clock i.e. no temperature and ageing effects. In addition, the slave to master clock difference and the slave to reference clock difference are the same because the master clock is a perfect clock i.e. the same as the reference clock. This difference comprises of the drift (about 10 μs) arising due to the slave clock rate before the next synchronization happens and a small drift arising from the dominant temperature effect and ageing effect of the slave clock in a synchronization window. The overall

graph shapes look like the temperature profile used due to the dominant temperature effect.

C.1.2 Real Slave and Master Clocks with No Corrective Models Enabled

Description

In a second test case, both master and slave clocks have temperature and ageing effects. The master clock is connected to a reference signal (a highly accurate GPS signal with a GPS noise of 20 ns RMS (Root Mean Square) jitter on a 1pps edge) to mimic a clock model mentioned in Section 3.3. The GPS signal is implemented in NS-2 using the simulation time with the 20 ns RMS noise reflected by using the normal random variable generator. A master clock is said to be in locked mode when it is connected to the GPS signal. When the GPS signal is lost, no correction is applied on the master's time stamp. The test case measures the master clock accuracy as well as the IEEE 1588 synchronization accuracy of a slave clock receiving updates from its master clock.

Results

Figure 55 shows the master clock synchronization w.r.t. to the reference clock and the slave clock synchronization w.r.t. to the master clock and the reference clock when real slave and master clocks are used with no AOCM corrections applied on either.

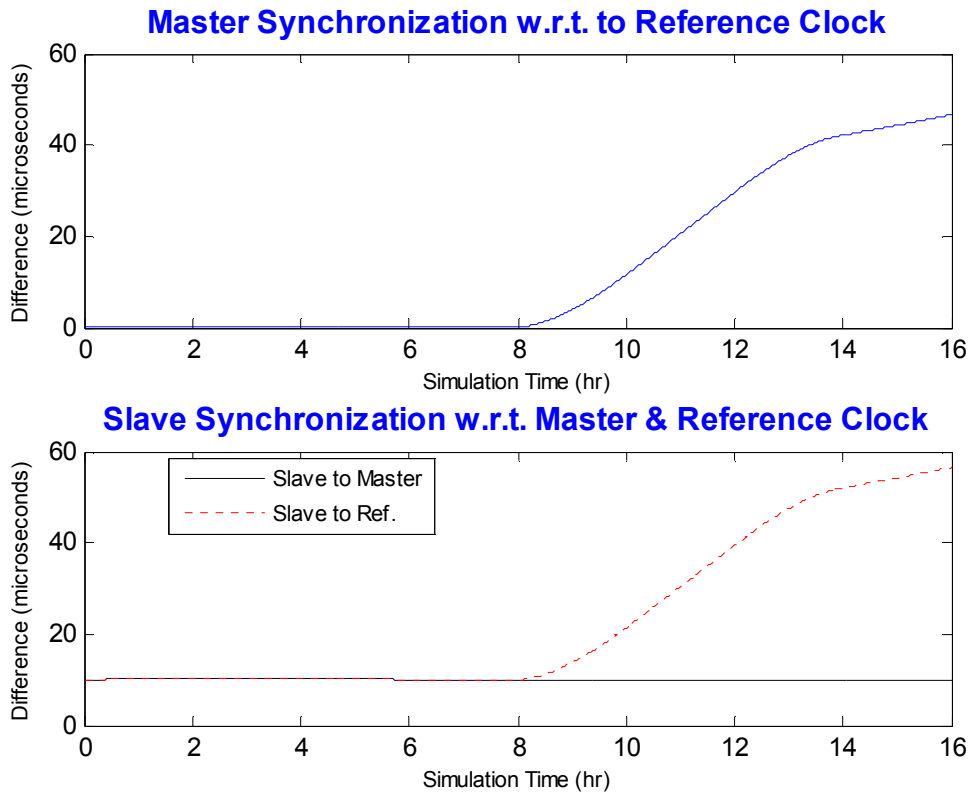

Figure 55: Clock Synchronization: No Corrective Models Enabled

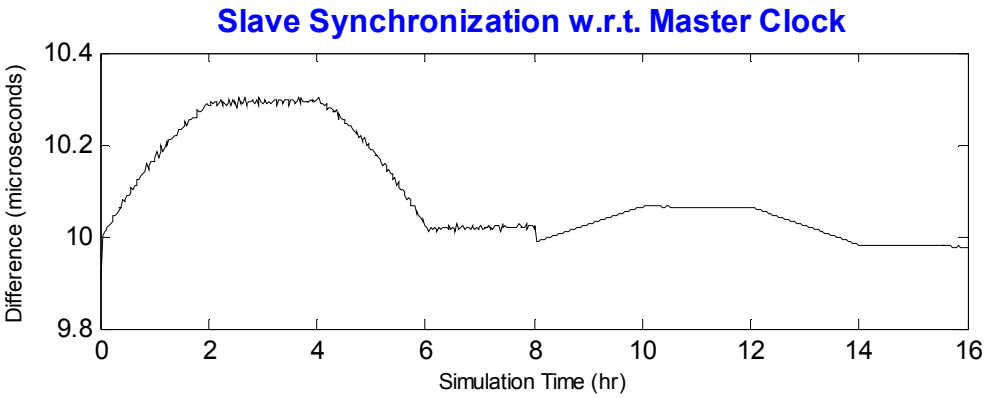

Figure 56: Closer Look at Slave to Master Difference: No Corrective Models Enabled

Discussion

From Figure 55, the master to reference clock difference is almost zero (20ns rms GPS noise) during the training period (0 to 8 hours). When the GPS signal is lost and the master clock stops receiving updates from the GPS signal, the master to reference clock difference starts growing due to the temperature and ageing effects on the master clock oscillator.

From Figure 55 and Figure 56, the slave to master clock difference when the master clock is in locked mode (0 to 8 hours) is the same as in the first test case using a perfect master clock (Section C.1.1) with a very slight variation due to the addition of GPS noise. It should be noted that when the master is in locked mode, the drifts due to temperature and ageing effects on the master clock get corrected after receiving updates from the GPS signal. So the master to slave clock difference is effectively due to the drifts due to the rate, the temperature and ageing effects of the slave clock. However when the master clock is in holdover mode (8 to 16 hours), the slave to master synchronization is observed to be improved. This is because in this case, there are drifts due to temperature and ageing effects both on master clock and slave clock which are in the same direction. The overall master to slave difference does not have much contribution from the temperature and ageing effect. The difference is therefore decreased compared to locked mode case but still stays around 10 μs due to drift from the dominant slave clock rate. This is not to say that in general that the slave to master synchronization would always improve when the master goes into holdover mode. In general it would depend on the values of the temperature and ageing effects both on master and slave clocks. If they are drifting in the

same direction, the accuracy will improve, if they are in opposite direction, the accuracy will get worse.

Finally the slave to reference signal difference during the master training period is around 10μs as expected, but in the holdover mode it grows as the master clock itself is no longer accurate w.r.t. the reference clock.

C.1.3 Real Slave and Master Clocks when AOCM is Enabled on the Master Clock

Description

In the third test case, the second test case (Section C.1.2) is repeated with the modification that the adaptive AOCM model is applied on the master clock to correct its time stamp.

Results

Figure 57 shows the master clock synchronization w.r.t. to the reference clock and the slave clock synchronization w.r.t. to the master clock and the reference clock when real slave and master clocks are used but AOCM corrections are applied on the master clock only.

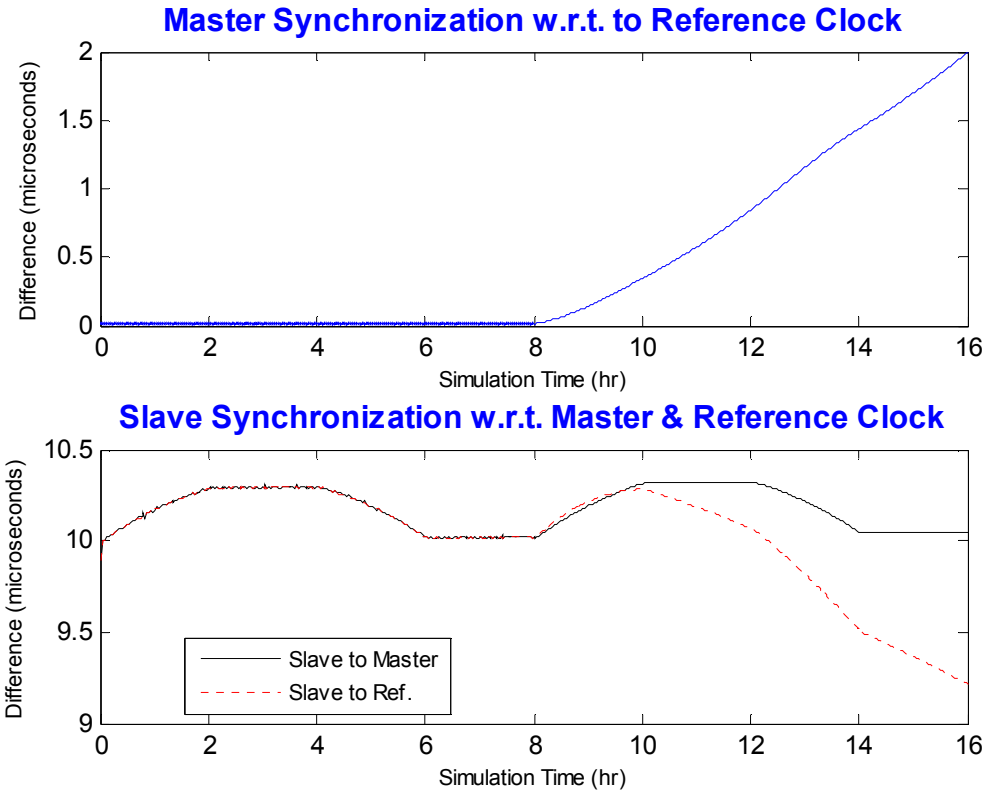

Figure 57: Clock Synchronization: AOCM Enabled on Master Clock

Discussion

From Figure 57, the master to reference difference is almost zero (20ns rms GPS noise) during the training period (0 to 8 hours). However, in holdover mode, since the adaptive corrective model is applied, the difference does not grow as fast as in the second test case where AOCM is not applied (Section C.1.2). This behavior is consistent with the results reported in [19] as summarized in Section 3.3.

Next, the slave to master clock difference is around 10 µs i.e. consists of the maximum slave clock drift in a synchronization window and drifts due to temperature and ageing

effects with the graph looking like the temperature profile used due to the dominant temperature effect.

Finally, the slave to reference clock difference in Figure 57 shows that during the training period, the difference is around 10µs as expected but in holdover mode it reduces slightly and therefore shows an improvement in accuracy. The result is surprising and upon further investigation, and plotting the average of the true differences (master – reference) instead of the average of the absolute differences as shown in Figure 58, it is revealed that in holdover mode, the master clock becomes slower with respect to the reference clock, resulting in a difference of about 1 µs at simulation end.

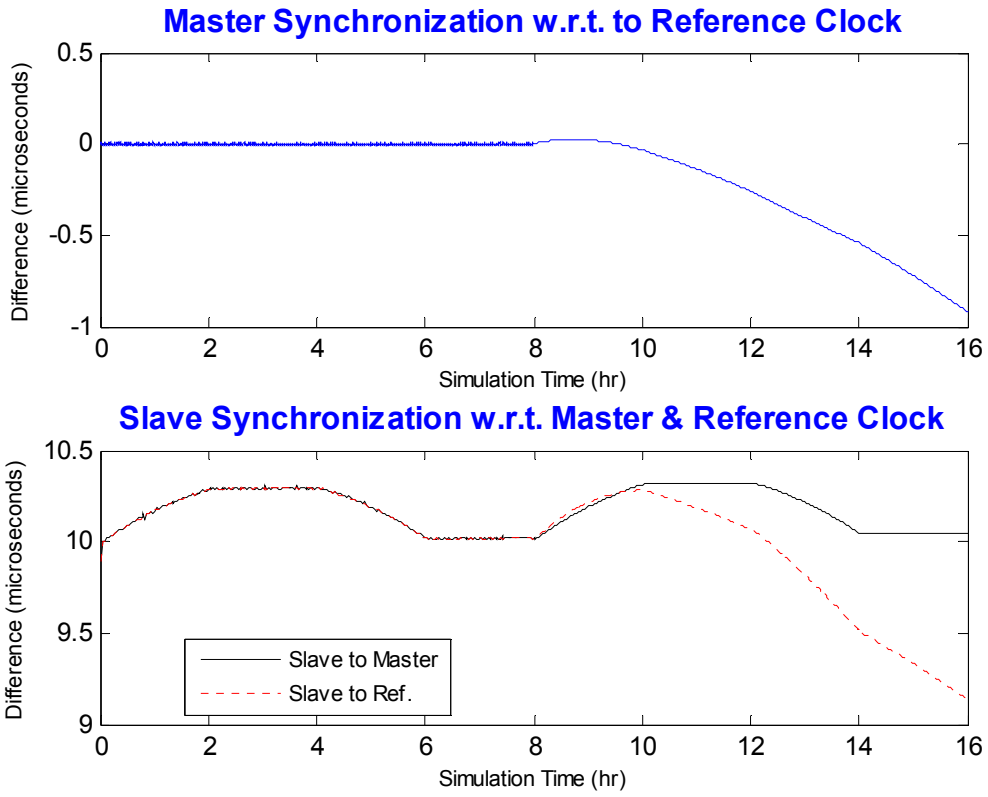

Figure 58: True Clock Differences: AOCM Enabled on Master Clock

As a result the slave clock enjoys improved accuracy w.r.t. the reference clock and the difference drops from 10 μs to 9 μs approximately at simulation end. In general, it seems the AOCM corrections on the master clock in holdover mode over-correct the master rate, making it slower. Therefore, in general the slave to reference clock accuracy would improve if the slave clock is running faster than the reference clock and would decline if the slave clock is running slower than the reference clock. The effect of slave clock rate on the synchronization accuracy is further examined in the test case in Section C.3.2.

C.1.4 Real Slave and Master Clocks with Temporary Network Outages – Corrective Model Enabled on Both Clocks

Description

In the current test case, the second test case (Section 5.2.2) is repeated with the modification that a real master clock experiencing the temperature and ageing effects is used instead of a perfect master clock. The AOCM model is applied on both master and slave clocks.

Results

Figure 59 shows the master clock synchronization w.r.t. to the reference clock and the slave clock synchronization w.r.t. to the master clock and the reference clock for real slave and master clocks scenario facing a network outage (4 to 4.5 hours) with AOCM corrections applied on both.

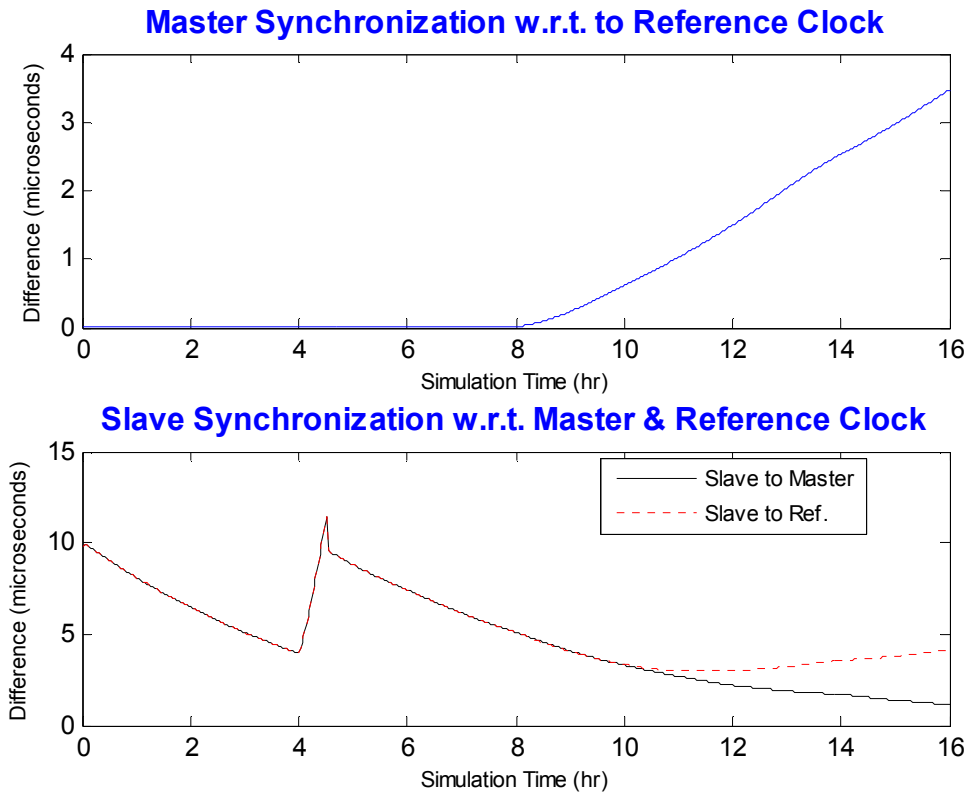

Figure 59: Clock Synchronization - Real Slave and Master Clocks Reflecting Network Outage (4 to 4.5 hours) with AOCM Corrections Enabled on Both Clocks

Figure 60 shows the slave clock synchronization for the case when the network outage happens from 14 to 14.5 hours.

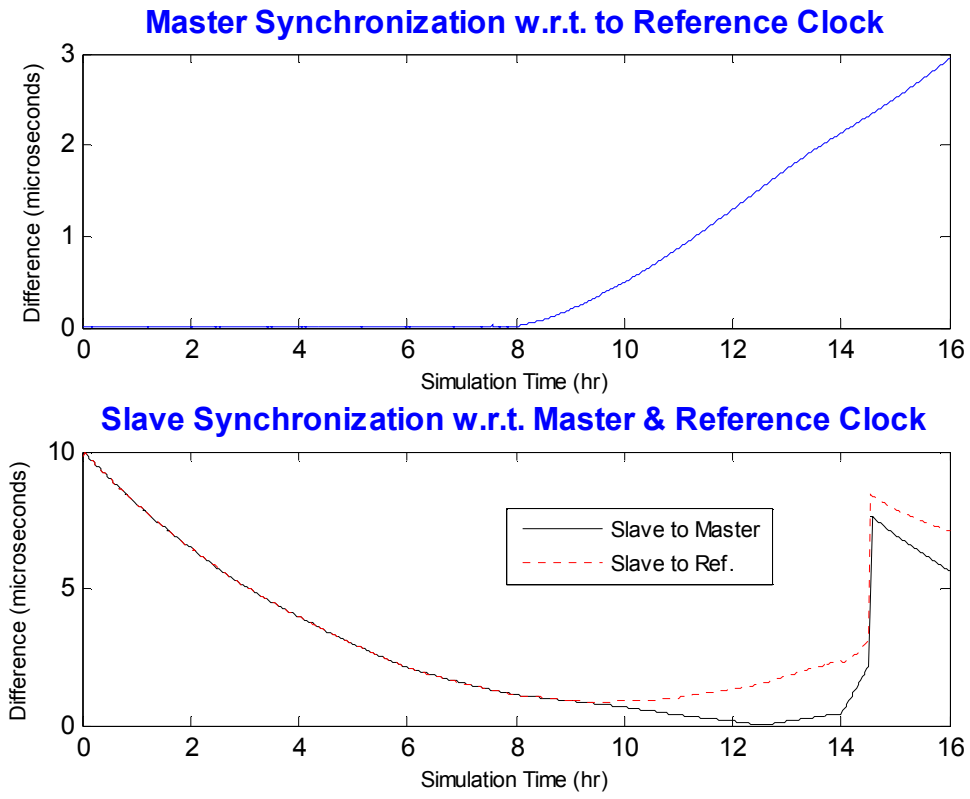

Figure 60: Clock Synchronization - Real Slave and Master Clocks Reflecting Network Outage (14 to 14.5 hours) with AOCM Corrections Enabled on Both Clocks

Discussion

From Figure 59, the master to reference clock difference grows from 0 to about 3 µs as it switches from locked mode (0 to 8 hours) to holdover mode as also observed in the third test case (Section C.1.3). The slave to master accuracy is the same as in the previous test case (Section 5.2.2). However the slave to reference difference grows when the master clock itself becomes inaccurate in its holdover period. The results for the case when the network outage happens from 14 to 14.5 hours as shown in Figure 60 are similar.

C.2 Additional Test Cases with Traffic

In this sub-appendix, some more traffic test cases not already covered in Section 5.3 are presented.

C.2.1 Temporary Network Congestion using Real Slave and Master Clocks - Corrective Model Enabled on the Master Clock only

Description

In the current test case with traffic scenario, we explore the impact of temporary network congestion between the master and slave clocks. The adaptive oscillator correction model is enabled on the master clock only. Using the traffic model described in Section 5.1.1, a static traffic load of 40% is introduced in the forward direction while 30% traffic is introduced in the reverse direction starting at a simulation time of 3 hours and stopping at a simulation time of 13 hours. At a simulation time of 7 hours, the traffic load is increased to 100% in both directions for 10 minutes and then restored to the previous values. The test case examines the effect of network traffic congestion on the slave clock synchronization accuracy.

Results

Figure 61 shows the slave clock synchronization w.r.t. to the master and reference clocks and the difference of delays experienced (forward - reverse) in network congestion scenario with AOCM corrections applied on the master clock only.

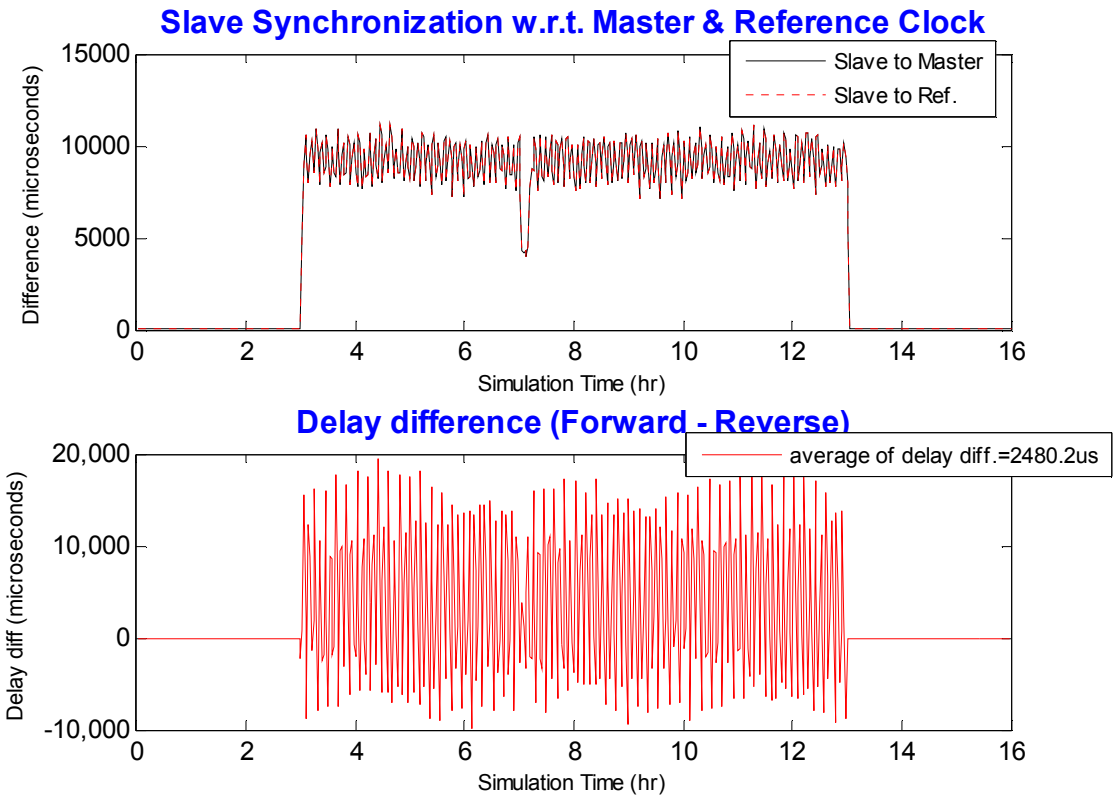

Figure 61: Clock Synchronization - Temporary Network Congestion with Corrective Model applied on Master Clock only

Discussion

From Figure 61, the results of the slave to master and reference clock synchronization are generally similar to the test case with static packet load scenario in Section 5.3.1. The slave differences are at the 10,000 µs level and the asymmetric delays experienced by the synchronization packets are at the 40,000 µs level. When the traffic is increased to 100% in both directions for 10 minutes starting at 7 hours simulation time, the delays are comparatively less asymmetric and hence the slave accuracy is fairly improved. The

average of all delay differences (forward – reverse delay) is 2480.2 μs i.e. overall the forward delays dominate due to the dominant forward traffic.

C.2.2 Slow Change in Traffic Load using Real Slave and Master Clocks - Corrective Model Enabled on the Master Clock only

Description

In a second test case, we demonstrate slow changes in network load over a long period of time between the master and slave clocks. The adaptive oscillator correction model is enabled on the master clock only. Using the traffic model described in Section 5.1.1, and the load profile shown in Figure 62, the traffic is introduced in both directions. Here in the forward direction, the load is changed smoothly from 20% to 90% and back over a 16-hour period. Simultaneously, in the reverse direction, the load is changed smoothly from 10% to 45% and back over the same 16-hour period. The test case examines the effect of slow traffic changes in the network on the master and slave clock synchronization with AOCM enabled on the master clock only.

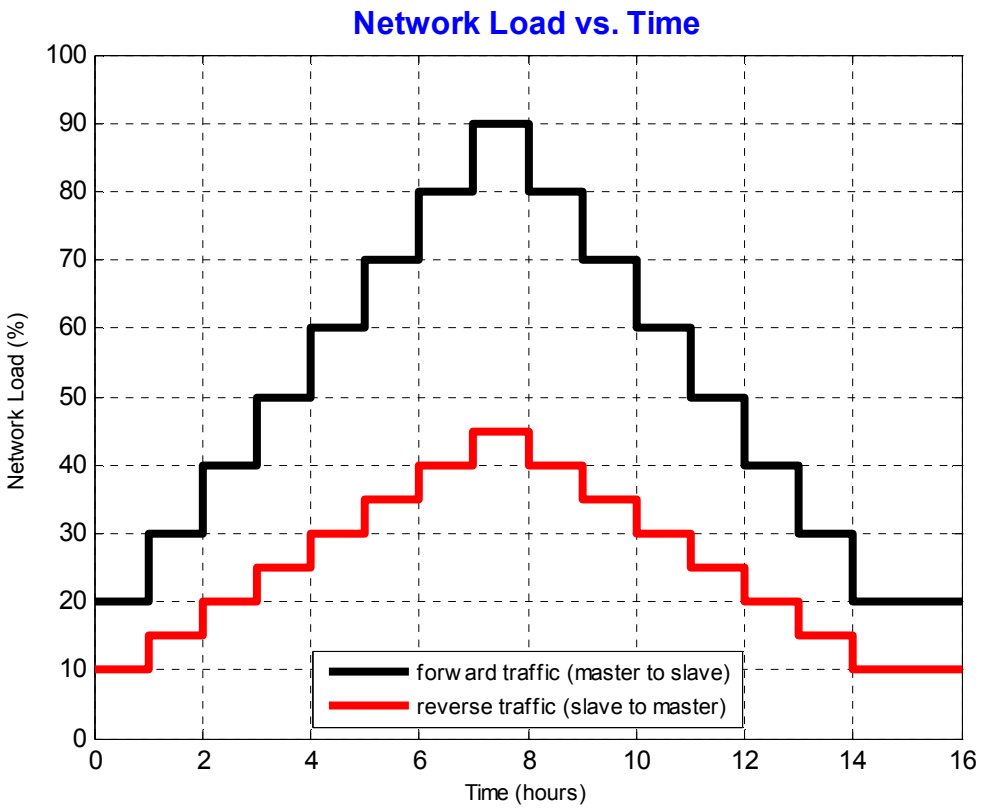

Figure 62: Load Profile demonstrating Slow Change in the Network Load

Results

Figure 63 shows the slave clock synchronization w.r.t. to the master and reference clocks and the difference of delays experienced (forward - reverse) when traffic using load profile in Figure 62 is introduced between the real slave and master clocks with AOCM corrections applied on the master clock only.

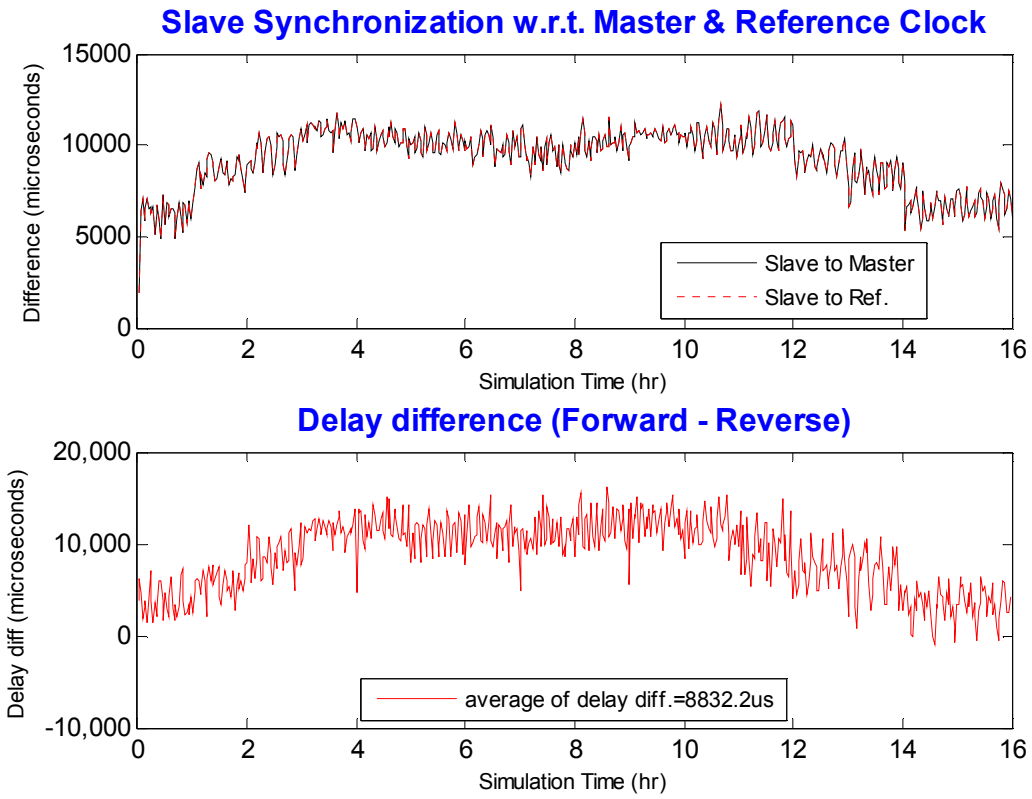

Figure 63: Clock Synchronization - Slow Changes in Network Load using the Corrective Model on the Master Clock only

Discussion

From Figure 63, the results of the slave to master and reference clock synchronization are in general similar to the previous test cases with traffic. The delays are highly asymmetric and represent the traffic load used with the average of delay differences (forward – reverse delay) being 8,832.2 µs.

C.2.3 Network Outage with Slow Change in Traffic Load using Real Slave and Master Clocks - Corrective Model Enabled on Both Clocks

Description

In a third test case, we repeat the 2nd test case (Sub-Appendix C.2.2) with the inclusion of network outage (4 to 4.5 hours) and also enabling the AOCM model on the slave clock. We examine the effect of network outage in the presence of slow traffic changes on the accuracy of slave clock synchronization.

Results

Figure 64 shows the slave clock synchronization w.r.t. to the master and reference clocks and the difference of delays experienced (forward - reverse) when traffic using the load profile in Figure 62 is used and a network outage (4 to 4.5 hours) is experienced between the real slave and master clocks with AOCM corrections applied on both the master and slave clocks. Figure 65 looks closely at the slave clock synchronization during the outage period (4 to 4.5 hours).

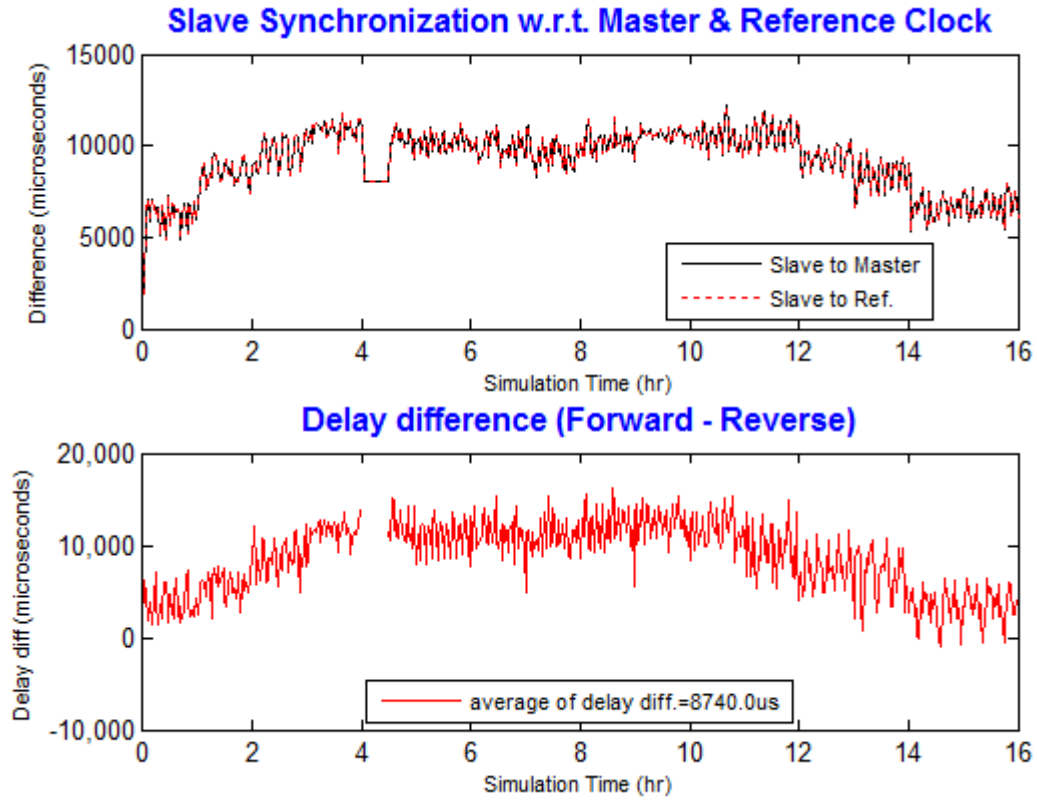

Figure 64: Clock Synchronization – Network Outage during Slow Changes in Network Load using the Corrective Model on Both Clocks

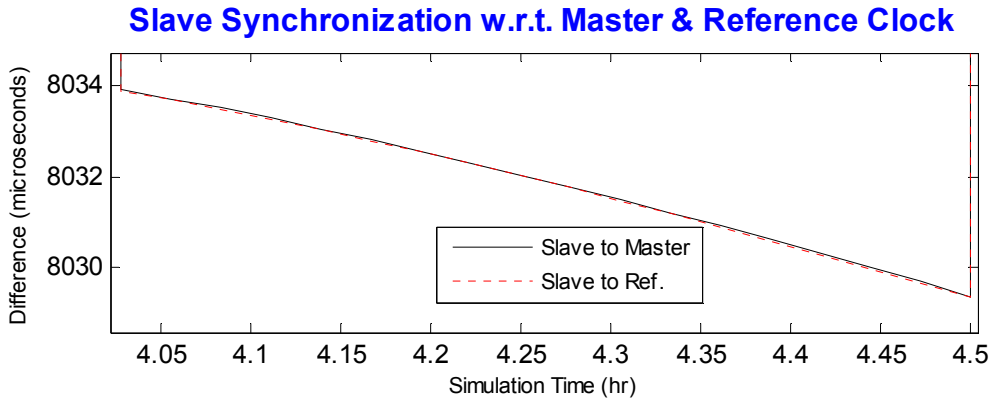

Figure 65: Clock Synchronization – Network Outage during Slow Changes in Network Load using the Corrective Model on Both Clocks - Closer Look at the Holdover Period

Figure 66 and Figure 67 show the slave synchronization results for the case when AOCM correction is not applied on the slave clock.

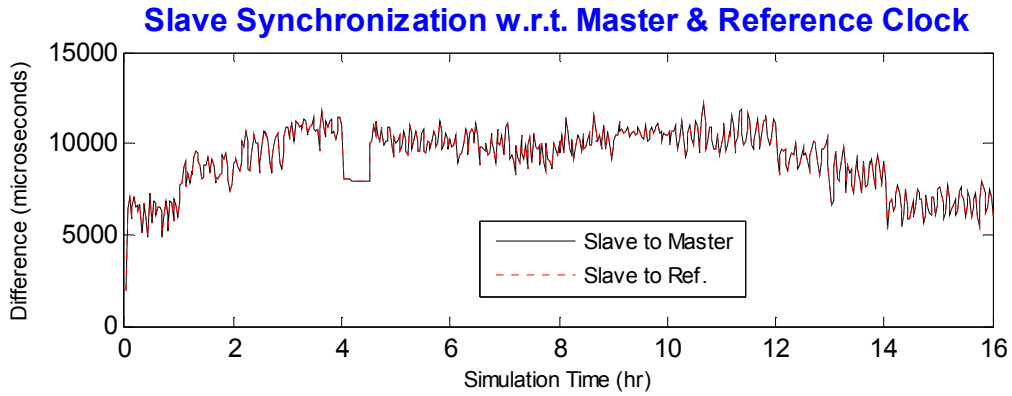

Figure 66: Clock Synchronization – Network Outage during Slow Changes in Network Load with No AOCM Model on the Slave Clock

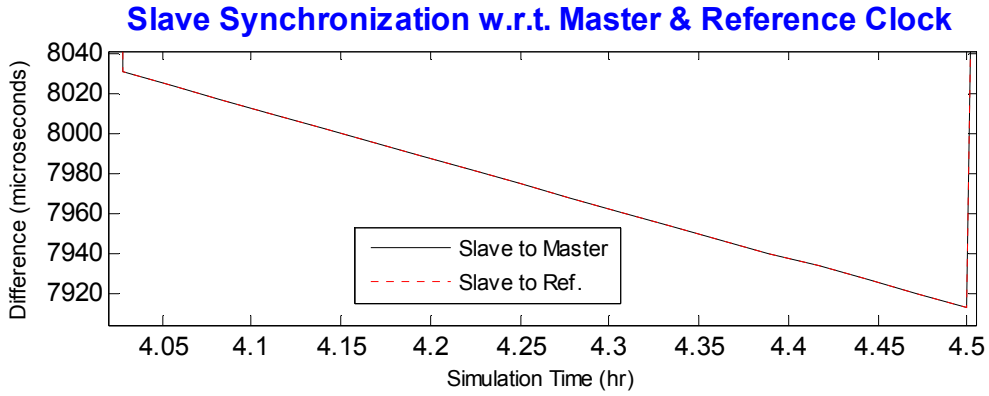

Figure 67: Clock Synchronization – Network Outage during Slow Changes in Network Load with No AOCM on the Slave Clock - Closer Look at the Holdover Period

Discussion

From Figure 64, the results of slave synchronization are the same as in the previous test case (Sub-Appendix C.2.2) while the results during the network outage (4 to 4.5 hours) are similar to the test case in Section 5.3.3 i.e. the slave clock is stable during the outage and drifts only up to 4 to 5 µs. Again plotting the average of absolute delay differences in Figure 68 shows that the slave clock stabilizes at half of the asymmetric inaccuracy level just before the network outage starts.

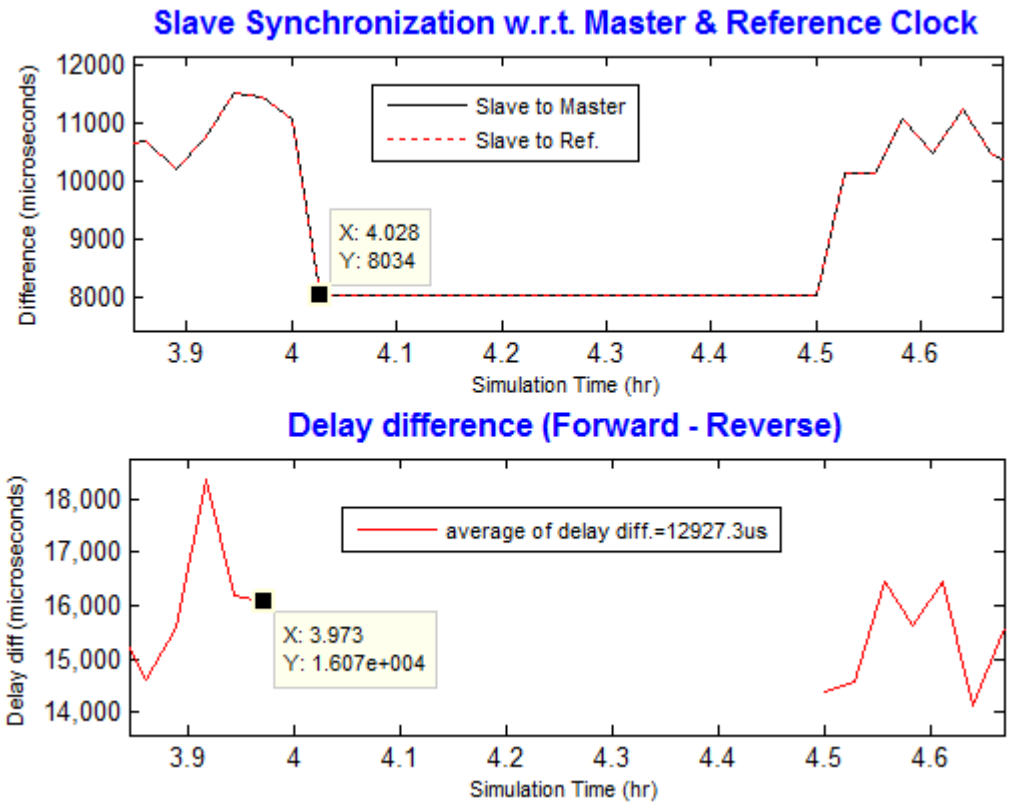

Figure 68: Using the Average of Absolute Delay Differences - Network Outage during Slow Changes in Network Load using the Corrective Model on Both Clocks

Comparing the result with the scenarios where no AOCM is applied on the slave clock, the slave drift is 120 µs as shown in Figure 67. Again for a single run, the drift is 180 µs.

C.3 Additional Test Cases to Examine the Effect of Miscellaneous Parameters

Some more test cases not covered in Section 5.4 are presented in this sub-appendix to study the direct effect of various parameters on the clock synchronization.

C.3.1 Master Clock Rate

Description

In the first test case, we have real master and slave clocks as connected earlier in Figure 24 with AOCM enabled on the master clock. The master clock rate w.r.t. the reference clock is varied and the effect on the master clock synchronization is observed. The master clock rates to be used are -100ppb, -50ppb, -20ppb, -10ppb, 10ppb, 20ppb, 50ppb, 100ppb where a minus value means a clock running slower than the reference clock by that much value and a positive value indicates a faster clock. The effect of the parameter variation on the master clock synchronization in holdover mode is examined in this test case.

Results

The average values for master clock inaccuracy w.r.t the reference clock in holdover mode are calculated and plotted using a bar chart as shown in Figure 69.

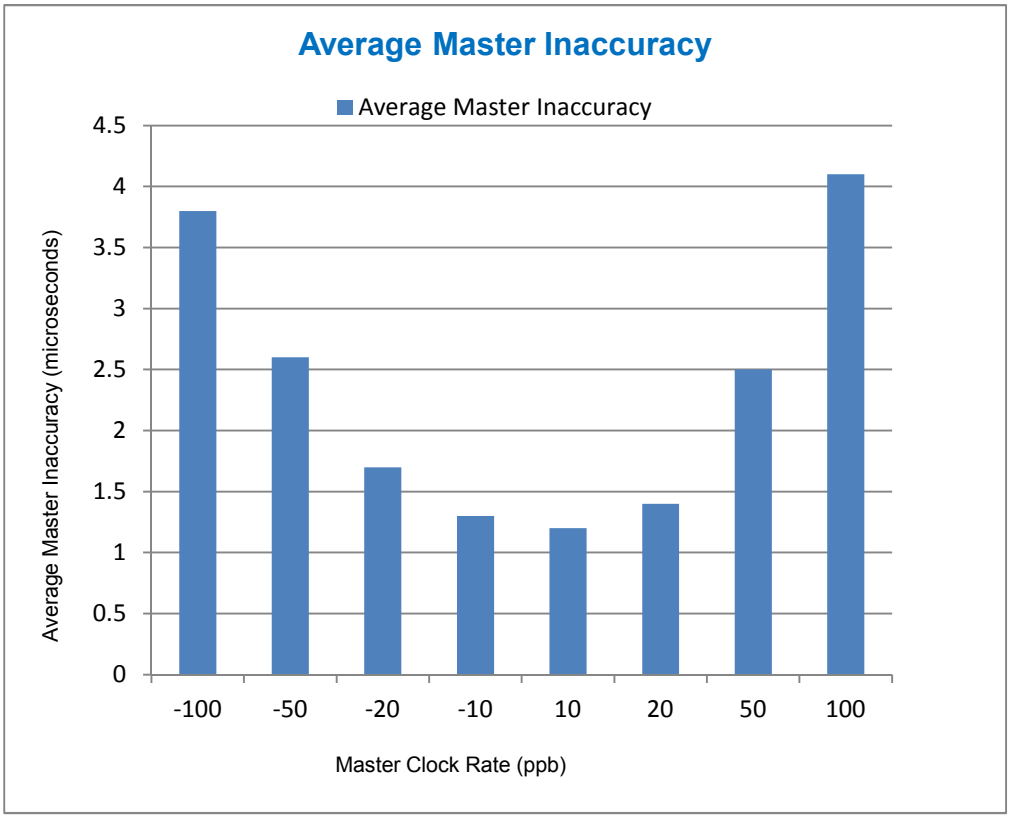

Figure 69: Master Clock Synchronization in Holdover Mode - Varying Master Clock Rate

Discussion

From Figure 69, as expected it is observed that when the master clock rate is 100 ppb faster or slower than the reference clock, the master inaccuracy is the most. On the other hand, with ±10 ppb master clock rate, the inaccuracy is the least. Hence the AOCM corrections applied on the master clock in holdover mode on average results in better results with lower inherent clock rates compared to higher rates.

C.3.2 Slave Clock Rate

Description

In a second test case, again we have real master and slave clocks as in the first test case with the AOCM enabled on the master clock. The master clock rate is the same as the reference clock while the slave clock rate is varied. The effect on the slave clock synchronization is observed. The slave clock rates to be used are -100ppb, -50ppb, -20ppb, -10ppb, 10ppb, 20ppb, 50ppb, 100ppb.

Results

The average values for maximum slave drifts in each synchronization window are calculated to determine the slave clock inaccuracy w.r.t. the master clock and plotted using a bar chart as shown in Figure 70.

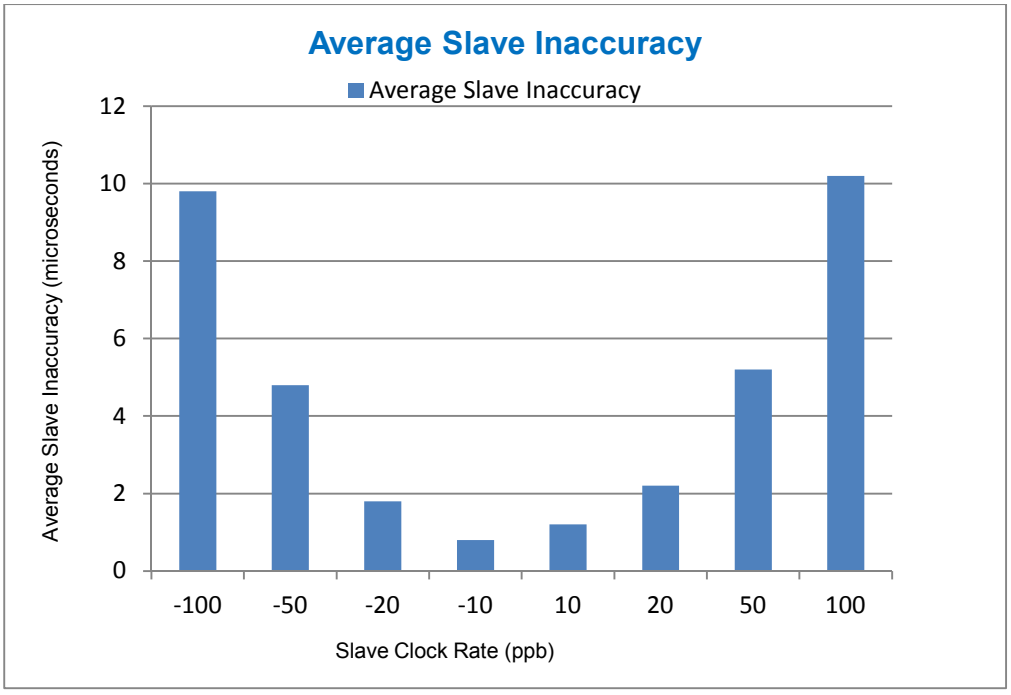

Figure 70: Slave Clock Synchronization - Varying Slave Clock Rate

Discussion

From Figure 70, as expected, it is observed that when the slave clock rate is 100 ppb faster or slower than the reference clock, the slave inaccuracy is the most. On the other hand, with ±10 ppb slave clock rate, the inaccuracy is the least. Also comparing the slave results with the previous results of the master clock, it can be seen that the master accuracy is better than the slave accuracy for different clock rates. This makes sense because in the previous test case, the master clock is locked to a GPS signal and is trained every second for 8 hours. The trained model is then used in holdover mode. In case of the slave clock, the synchronization updates are received every 100 second at which point the slave clock time stamp is updated based on the IEEE 1588 offset. Hence the clock accuracy of the slave clock suffers more as compared to the master clock for the same clock rate.

C.3.3 Master AOCM Training and Holdover Periods

Description

In a third test case, we have real master and slave clocks as connected earlier in Figure 24 with the AOCM enabled on the master clock and using the parameters as described earlier in the simulation setup section (Section 5.1). The master AOCM training period is varied while the holdover period of 100 hrs is fixed. The training periods used are 1, 2, 5, 10 and 20 hours resulting in total simulation periods of 101, 102, 105, 110 and 120 hours respectively. The effect of the training period variation on the master clock synchronization in holdover mode is examined in this test case. In addition, for a training

period of 20 hours, the master inaccuracy is observed for various holdover periods of 1, 5, 10, 20, 30, 50 and 100 hours.

Results

Figure 71 shows the master inaccuracy for various training periods while Figure 72 shows the master inaccuracy for various holdover periods.

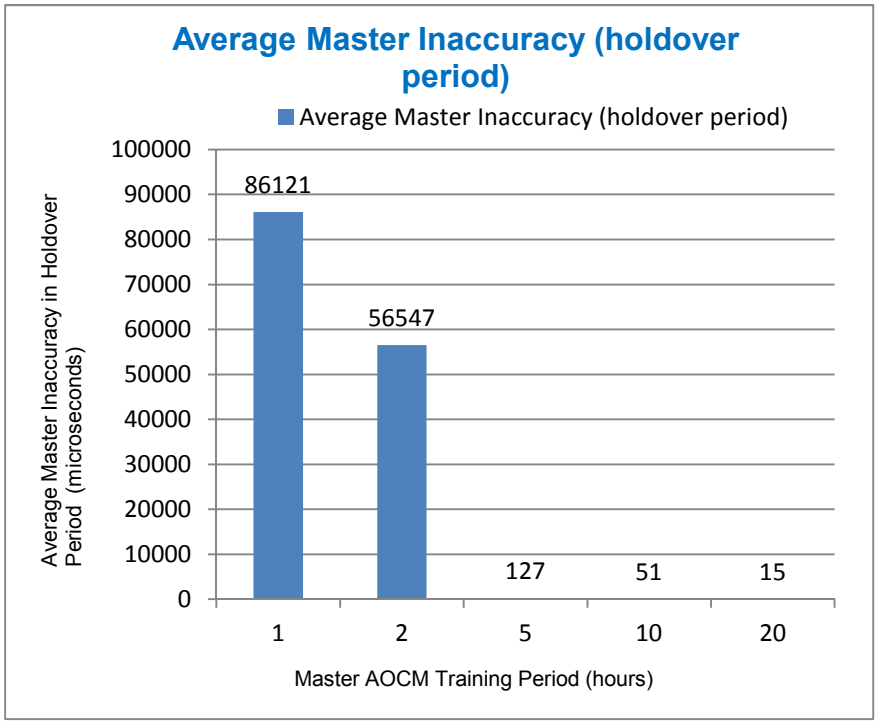

Figure 71: Master Clock Synchronization in Holdover Mode - Varying Master Training Period

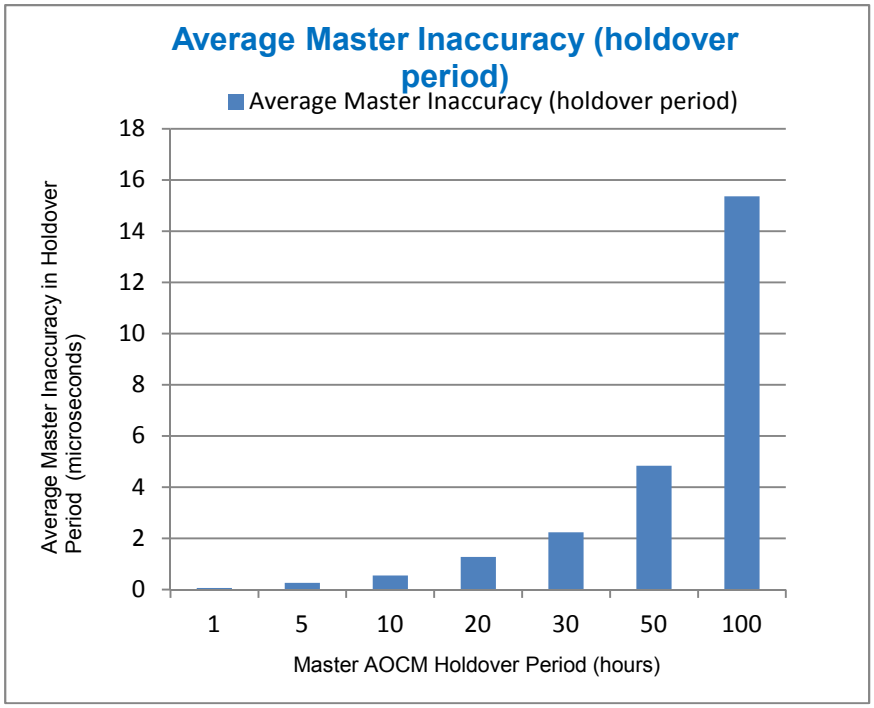

Figure 72: Master Clock Synchronization in Holdover Mode - Varying Master Holdover Period

Discussion

From Figure 71, it is observed the master inaccuracy is reduced greatly with longer training periods for the same holdover periods. The longer training period ensures more samples are contributed to the AOCM training model producing better results during the holdover period.

From Figure 72, the results for varying holdover periods, as expected, show that the master clock differences grow with longer holdover periods. The factors affecting the master clock accuracy include the master clock rate, the temperature and ageing effects. We know from the earlier discussion on the ageing effect (Section 4.2.2) that for longer simulation periods, the ageing effect gets dominant. Here also looking at the trend of the

master clock inaccuracy, it can be seen that it has big contributions from the ageing effect for longer simulation periods such as 50 to 100 hours.

References

[1] Symmetricom white paper, "The Importance of Network Time Synchronization", 2009.

[2] Symmetricom white paper, "The Five Dangers of Poor Network Timekeeping", 2009.

[3] J. Vig, "Quartz Crystal Resonators and Oscillators for Frequency Control and Timing Applications - A Tutorial", 2004 IEEE International Frequency Control Symposium Tutorials, May 2004.

[4] W. Zhou, "Time, Frequency Measurement and Control Technology", Xidian University Press, 2006. ISBN-10: 7560616720

[5] T. Pialis and K. Phang, "Analysis of Timing Jitter in Ring Oscillators Due to Power Supply Noise", Circuits and Systems, ISCAS'03, Proceedings of the 2003 International Symposium, pp. I-685 – I-688, 2003.

[6] J. Vig, "Introduction to Quartz Frequency Standards" Technical Report SLCET-TR-92-1, Army Research Laboratory, Electronics and Power Sources Directorate, 1992. [Online], available: http://www.ieee-uffc.org/frequency_control/teaching.asp?name=vigtoc (accessed Nov 20, 2011)

[7] Symmetricom white paper, "IEEE 1588 Precise Time Protocol: The new standard in time synchronization", 2009.

[8] Symmetricom white paper, "Improving Real World Synchronization Accuracy with IEEE-1588 Transparent Clocks", 2009.

[9] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, 2008.

[10] H. Weibel, “High Precision Clock Synchronization according to IEEE 1588 Implementation and Performance Issues”, in Proc. Embedded World 2005.

[11] N. M. Freris, S. R. Graham, P. R. Kumar, "Fundamental Limits on Synchronization of Affine Clocks in Networks", Automatic Control, IEEE Transactions on, On page(s): 1352 - 1364, Volume: 56 Issue: 6, June 2011.

[12] D. Köhler, “A Practical Implementation of an IEEE 1588 supporting Ethernet Switch”, Proc. IEEE Int. Symp. Precision Clock Synchronization Meas., Control Commun., pp. 134–137, Oct. 2007.

[13] J. Han, H. Chi and D-K. Jeong, “A Clock Synchronization System with IEEE 1588-2008 Adapters over Existing Gigabit Ethernet Equipment”, Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium, pp. 193 – 196, Aug. 2010.

[14] H. Weibel and D. Béchaz, “IEEE 1588 Implementation and Performance of Time Stamping Techniques”, Proceedings of the 2004 Conference on IEEE 1588, Sep. 2004.

[15] J. Han and D.-K. Jeong, “Practical Considerations in the Design and Implementation of Time Synchronization Systems Using IEEE 1588”, Communications Magazine, IEEE , vol.47, no.11, pp. 164-170, Nov 2009.

[16] N. Shenoy, J. Fischer, P. Myers and Z. Qui, “A Simulation Study – Performance of IEEE 1588 over Ethernet and IEEE 802.11b WLANs”, IEEE Conference, 2005.

[17] Crystal Oscillator [Online], available: http://en.wikipedia.org/wiki/Crystal_oscillator (accessed December 2, 2011)

[18] H. Zhou, C. Nicholls, T. Kunz and H. Schwartz, "Frequency Accuracy & Stability Dependencies of Crystal Oscillators", Carleton University, Systems and Computer Engineering, Technical Report SCE-08-12, November 2008. [Online], available: http://kunz-pc.sce.carleton.ca/thesis/CrystalOscillators.pdf (accessed December 2, 2011).

[19] H. Zhou, T. Kunz and H. Schwartz, "Adaptive Correction Method for an OCXO and Investigation of Analytical Cumulative Time Error Upper Bound", IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, vol. 58, no. 1, January 2011.

[20] ITU-T G.8261/Y.1361., April 2008, “Timing and synchronization aspects in packet networks”, [Online], available: http://www.itu.int/rec/T-REC-G.8261-200804-I (accessed June 1, 2011).

[21] NS-2 Simulator, [Online], available: http://isi.edu/nsnam/ns/

[22] T. Issariyakul and E. Hossain, “Introduction to Network Simulator NS2”, Springer Science, Business Media, LLC, 2009. ISBN: 978-0-387-71759-3.

[23] C. H. Rentel and T. Kunz, "A Mutual Network Synchronization Method for Wireless Ad Hoc and Sensor Networks", IEEE transaction on mobile computing, vol. 7, no. 5, May 2008.

[24] L. Xueqiao, C. Yuan, L. Shuang and D. Yaxing, "Implementation and Research of Hardware Time Stamping Techniques Based on IEEE1588", Communication Software and Networks (ICCSN), IEEE 3rd International Conference, pages 6 - 9, May 2011.