Delay Asymmetry Correction Model for IEEE 1588 Synchronization Protocol

By

Md. Arifur Rahman

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Carleton University
Ottawa, Ontario

Abstract

The thesis proposes a delay asymmetry correction (DAC) model to enhance the IEEE 1588 synchronization protocol. The purpose of this work is to mitigate the effects of unpredictable packet delay variations (PDV), which cause asymmetric link delays on timing packets, in order to improve the synchronization accuracy of the slave clock with respect to the master clock. This is done by computing the time difference between the master and the slave clock in the presence of traffic in a network. The NS-2 results indicate that the proposed solution improves the slave accuracy by measuring the correct offset value in a slave clock for asymmetric communication link delays. The solution results show that the slave clock is able to achieve high synchronization accuracy in the presence of various bi-directional traffic loads, network congestions, and temporary network outage. Furthermore, when there is a routing path change due to the failure in the network, the solution also improves the accuracy of the slave clock with respect to the master clock. However, the proposed solution does not perform well when it is incorporated with the AOCM model.

Acknowledgements

This thesis would have been impossible without the support and the encouragement I received from several people.

First and foremost I am deeply grateful 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. Haward Schwartz and Dr. Mark Wyville (Ericsson, Canada) for their support and invaluable feedback during the thesis research period.

I am also thankful to my colleagues M. Raisul Alam, M. Zulhasnine, and Y. Chang for sharing their experiences and thoughts on various aspects of the scientific research.

Finally, I would like to thank my family 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
BMC	Best Master Clock
CAPEX	Capital Expenses
DACM	Delay Asymmetry Correction Model
E2E TC	End-to-End Transparent Clock
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineering
IETF	Internet Engineering Task Force
ITU	International Telecommunication Union
IRIG	Inter-Range Instrumentation Group
LTE	Long Term Evolution
MTU	Maximum Transfer Unit
NGN	Next Generation Network
NS	Network Simulator
NTP	Network Time Protocol
PDV	Packet Delay Variation
PTP	Precision Time Protocol
PPB	Parts Per Billion
PPM	Parts Per Million
PPS	Pulse Per Second
QoS	Quality of Service
RAN	Radio Access Network

RMS	Root Mean Square
RTE	Residence Time Error
RTT	Round Trip Time
TCL	Tool Command Language
TDM	Time Division Multiplexing
UDP	User Datagram Protocol
UTC	Coordinated Universal Time

1 Chapter: Introduction

1.1 Overview

A network consists of computers, routers, switches and other devices, all of which rely on clocks. For a successful communication between these devices, clocks play a significant role in maintaining the same global time across network devices. Now the question arises how these clocks maintain global time with each other. The short answer is synchronizing them. In essence, clock synchronization is setting the time on two or more clocks to be identical. Every node on the network must count time in the same way and every node has to agree when time "zero" occurs [1]. Clocks are typically made from inexpensive oscillator circuits, or battery backed quartz crystals. Each of these clocks tends to drift due to inherent instabilities in the source of oscillator, in addition to environmental factors such as temperature, aging, manufacturer imprecision, air pressure, mechanical pressure etc. [5]. If the network clocks are not synchronized, unexpected things may start to happen: data could be lost, processes could fail, security could be compromised, legal implications could be faced and most importantly the organizations could lose credibility with customers and their business partners [2]. Synchronization requires communications between individual clocks to check whether their deviation is tolerable and whether the clock needs to be corrected. It takes time to go through the process of correcting time and maintaining the accuracy of time relative to another clock. The correction mechanism to synchronize individual clock is a challenging task and is a limiting factor in how accurately two clocks maintain a common time. IEEE 1588v2, known as Precision Time Protocol (PTP) [17], is an industry standard clock synchronization protocol that enables to transfer timing information over packet switched

networks. It is widely used both in wire-line and wireless network environments due to its low cost implementations in networked measurement and control systems.

The objective of the thesis is to study different approaches that can be used to synchronize clocks in the network. To understand the problems explicitly, various factors need to be investigated which deteriorate the synchronization accuracy in a network. We propose a solution to achieve high synchronization accuracy. This also involves understanding why the clocks drift, what are the factors affecting the synchronization accuracy, the stability of the clock oscillators, and what can be done to counter those factors.

1.2 Contributions

The contributions of this thesis include the following:

- A Delay Asymmetry Correction (DAC) Model is proposed for the clock synchronization problem. The proposed work aims to enhance the IEEE 1588 protocol by computing the time difference between the master and the slave clock in the presence of traffic in a network. The purpose of this work is to mitigate the effects of unpredictable packet delay variations (PDV), which cause asymmetric link delays on timing packets in order to improve the synchronization accuracy of the slave clock with respect to the master clock. The initiative revolves around the idea of incorporating the DAC model with the conventional IEEE 1588 synchronization protocol.
- The proposed solution further extends to coordinate multiple master clocks through a single slave clock, which may be connected through multiple networks. The rationale of the extension is to support multiple master clocks instead of

selecting a grandmaster clock. The proposed solution ensures that the slave clock receives at least one good offset sample from one of these master clocks within defined update frequency interval for maintaining high synchronization accuracy between the slave and the master clocks.

- The proposed solution also integrates the Adaptive Oscillator Correction Method (AOCM) [4], which is modified to support the proposed DAC model. AOCM is locked to both the master clock and the slave clock that aims to compensate for the temperature and ageing effects of the oscillator and hence, to improve the accuracy and stability of the slave clock during holdover mode.
- The proposed solution is implemented in Network Simulator 2 (NS-2.34), and relies on a two stage filtering methods.
- NS-2 test cases are implemented according to an ITU-T document [3] covering various network loads and network conditions. The test cases are run to evaluate the proposed solution and the results are analyzed.
- The NS-2 results indicate that the proposed solution improves the slave accuracy by measuring the correct offset value in a slave clock for asymmetric communication link delays. The solution results show that the slave clock is able to achieve high accuracy in the presence of various bi-directional traffic loads, network congestions, and temporarily network outage. Furthermore, when there is a change in the routing path due to the failure in the network, the solution also improves the accuracy of the slave clock with respect to the master clock.

However, the slave accuracy deteriorates when the AOCM model is incorporated with the DAC model.

1.3 Outline of the Thesis

The chapters of this thesis are organized in the following manner:

Chapter 2 provides background information about the elements affecting the clock accuracy and stability. It reviews different protocols that are commonly used to improve clock accuracy. It also describes the IEEE 1588 protocol including the network factors affecting clock synchronization.

Chapter 3 reviews the state-of-the-art literature related to the performance of IEEE 1588 clock synchronization to account for the delay asymmetry and packet delay variation effects in terms of clock accuracy.

Chapter 4 presents the proposed solution to the clock synchronization problem using the IEEE 1588 protocol. It also discusses the details of the proposed algorithm implemented in NS-2.

Chapter 5 presents the simulation results using the proposed solution from Chapter 4. The test cases are derived from an ITU-T standard document [3].

Chapter 6 scrutinizes the direct effects of various parameters on the slave clock synchronization and provides the analytical explanation.

Chapter 7 concludes the thesis and provides the directions for possible future work.

2 Chapter: Background Information

In this chapter, background information related to clock synchronization is presented. The first section of this chapter introduces crystal oscillators, a frequently deployed clock source. This section also discusses the factors affecting the stability and accuracy of a clock oscillator. The second section focuses on different network protocols that can be used to improve clock accuracy, signifying the importance of the IEEE 1588 synchronization protocol in terms of its accuracy. The third section presents the details of the IEEE 1588 synchronization protocol. Section four reviews the elements affecting the clock synchronization accuracy in a network. Finally this chapter is concluded with a brief overview of a clock agent implemented in NS2.

2.1 Crystal Oscillators

Oscillators are an important component of radio frequency and digital devices. The time keeping elements like digital clock, radio, computer, cell phones as well as test and measurement equipment, such as counters, signal generators and oscilloscope are driven by an oscillator stabilized by a crystal resonator. A crystal oscillator is an electric circuit used for generating an electrical signal with a very precise frequency determined by the piezoelectric material. The piezoelectric material converts the electrical impulses to mechanical stress which is subject to the very high mechanical resonances of the crystal and vice versa. The frequency is commonly used to keep track of time to provide a stable clock signal. The clock accuracy and stability depend on the underlying crystal oscillator as frequency source and frequency control components [5]. Furthermore, the degree of frequency stability and the required level of quality of crystal oscillators differ according to the application. The major factors affecting the accuracy of the oscillator are

temperature and aging. Oscillators are inherently not perfect even if there are no temperature and aging effects. An oscillator frequency deviates from its nominal frequency by a given amount, measured in PPM (Part-Per-Million) or PPB (Part-Per-Billion), and the deviation is more pronounced for cheaper oscillators.

Temperature is a significant factor which affects the stability of crystal oscillators. Different angels of crystal cuts produce different frequency-temperature characteristic. In general, it is observed that oscillator frequency stability exhibits a cubic dependency on temperature [6]. The frequency drifts typically are -100ppm to +100ppm for the temperature range of -60ºC to +100°C. The term drift refers to several phenomena where a clock does not run at the exact same speed compared to another clock.

Aging is referred as the change of the crystal oscillator frequencies according to the operational time. In general, the aging effect is not linear. However, the aging effect for a short period of time such as 24 h can be considered as linear effect on frequency stability [5, 6]. The frequency drifts due to aging are up to 30ppm for a 25 day period.

2.2 Network Synchronization

Network synchronization ensures that all terminal devices in a network maintain the notion of global time from one source. To achieve such high accuracy clocks, various timekeeping techniques have been studied over the past few decades.

The pioneering work in this area was reading remote clocks in networks, proposed by Dutch scientist Flaviu Cristian in 1989, which deals with unbounded random message delays [7]. The proposed method was useful to improve the precision of both internal and external synchronization algorithms. In this method, each client sends a message to the remote server asking for the current time. Upon arrival of a response, the

sender calculates the round trip time (RTT) as the difference between the transmission and reception times. One of the key features of this algorithm relies on a series of measurements so that at least one measurement encountered the smallest amount of traffic. Logically, the chosen measurement is the one with least RTT.

Another approach was the distributed clock synchronization algorithm, known as Berkley algorithm, developed by Gusella and Zatti at the University of California, Berkley in 1989, discussed in [8]. The method describes the upper and lower bounds on the accuracy of the time synchronization based on a master-slave hierarchy in a local area network. By implementing the algorithm, the results show that it may achieve better synchronization accuracy at a lower cost comparing with other synchronization algorithms till then.

Network Time Protocol (NTP) is one of the most prominent time synchronization methods over four generations of NTP to the present, proposed by Mills and the Internet Engineering Task Force (IETF) group [9]. Initially NTP was designed to distribute time information in a network for systematic dissemination of national standard time throughout the Internet via wire or radio. The latest version of NTP, NTPv4 [10], is widely used to synchronize the local clocks of a computer system over packet-switched networks with variable latency to the order of less than a millisecond or two relative to Coordinated Universal Time (UTC). The main attractive features of NTP are its scalability, robustness to failure, self-configuration in a large multi-hop networks and ubiquitous development. NTP is a layered client-server architecture based on UDP (User Datagram Protocol) messages, which synchronize the clocks in a hierarchical manner. The NTP subnet model introduced a number of widely accessible primary time servers,

which are also synchronized by using wire or radio to national standards. The objective of the NTPv4 protocol is to convey timekeeping information from these primary servers to secondary time servers and clients via both private networks and the public Internet. One of the major problems with NTP is that it does not consider network latencies and buffering for precise timing applications. On a NTP-based LAN network, the clock accuracy reduces to 1 to 2 milliseconds due to latency and jitter added by the network devices. The clock accuracy further drops to 1 to 20 milliseconds in NTP-based WAN networks [11]. Furthermore, the latest implementation of NTPv4 does not meet the higher precision requirements for next generation synchronization architecture such as LTE (Long Term Evolution) or 4G wireless backhaul network [12, 13, and 14]. For instance, the deployments of LTE or 4G systems require a time/phase synchronization accuracy within 1μs and frequency synchronization accuracy within 50ppb to 16ppb [15].

One of the emerging alternatives to NTP is the Precision Time Protocol (PTP), defined by IEEE Standard 1588, which was published first in November 2002 [16]. This protocol was originally developed by Agilent Technologies between 1990 and 1998 for testing and industrial automation applications. PTP offers high accuracy and the cost effectiveness of NTP by using existing Ethernet network supporting multicast communications. The protocol makes it possible to achieve synchronization accuracy in the order of sub-microsecond range. A new version of IEEE 1588 was published in 2008 with advanced features and improved performance [17]. The IEEE 1588 protocol overcomes most of the NTP latency and jitter issues through hardware time stamping at the physical layer of the network. It is important to note that the standard performs very well when the network delays are symmetric. As a matter of fact, though, packet network

delays are not symmetric and the one way latency in each direction may be different. In order to minimize the impact of asymmetry, PTP introduced specialized elements known as IEEE 1588 boundary clocks and transparent switches with added functionalities to preserve accuracy [17]. With the addition of boundary clocks or transparent switches, the protocol is able to achieve accuracy in the range of 20 to 100 nanoseconds in a LAN network and less than 10 microseconds in a WAN environment [18]. A comparison of NTP, IRIG (Inter-Range Instrumentation Group) time code and IEEE 1588 is provided in Figure 1 [11, 19]. IRIG provides synchronization over a dedicated cable to carry timing information between IRIG clocks.

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 1: Comparison of the Synchronization Accuracy of NTP, IRIG and IEEE 1588 [11]

IEEE 1588-2008 PTP also provides an evolutionary step towards the deployment of the next generation synchronization architecture [15]. Next generation network (NGN) infrastructure combining traditional TDM (Time Division Multiplexing) core networks and packet-based IP backhaul networks based on Ethernet is seen as the future in the telecommunication networks. In addition, there are situations where GPS is not an appropriate synchronization source for traditional synchronization methods particularly for fast-moving objects and in-building Pico-cell applications. PTP is an excellent candidate for GPS backup and as a redundant synchronization source for CDMA/LTE Macro-cells. The PTP system allows for setting/maintaining time/phase synchronization

of Radio Access Networks (RAN) such as CDMA, LTE and in-building Pico-cell synchronization. Moreover, IEEE 1588-2008 has been widely studied in various fields such as power distribution networks, wireless sensor network, telecommunications networks, and military applications [20]. The detailed functionalities of IEEE 1588-2008 including the node, system and necessary communication properties to support PTP will be discussed in the next section.

2.3 IEEE 1588: Precision Time Protocol (PTP)

The clocks in a PTP system are structured into a master-slave hierarchy based on time keeping capability and the traceability of its time source. IEEE 1588 PTP aims to achieve sub-microsecond synchronization of real-time clocks in components of a networked distributed measurement and control system [17]. The basic principle of PTP is that the most precise clock in the network has the ability to synchronize all other clocks. The most precise clock is the master clock, which determines the reference time for the entire system; also referred to as Grand Master Clock. The master serves as the reference clock for one or more slave clocks. The process of selecting the master in the network is performed using PTP's Best Master Clock (BMC) algorithm, which is applied by each participating nodes at specific intervals [17]. A master node transmits UDP multicast messages to the slaves at configurable intervals, while slaves respond to the master via unicast messages. Every slave uses the timing information to adjust their local clock with reference to their master.

The synchronization between the master and the slave clock relies on the exchange of timing messages. The standard [17] defines two types of timing messages: event and PTP general messages. Event messages require an accurate timestamp while

sending and receiving the event messages. On the other hand, general messages do not require accurate timestamps. The set of event messages consists of

- *Sync*
- *Delay_Req*
- *Pdelay_Req and*
- *Pdelay_Resp*

The set of general messages include the followings:

- *Announce*
- *Follow_Up*
- *Delay_Resp*
- *Pdelay_Resp_Follow_Up,*
- *management and*
- *Signaling*.

2.3.1 Synchronization Mechanisms

IEEE 1588 [17] defines two mechanisms for synchronization in a master-slave hierarchy. The first method is called the Delay Request-Response mechanism which uses the PTP event messages. This approach relies on mean delay knowledge, i.e. half of the round trip delay, in order to correct the time difference between two clocks. The second method is the peer delay mechanism which uses Pdelay_Req, Pdelay_Resp, and if required Pdelay_Resp_Follow_Up messages. The details of the former approach are discussed as follows.

The Delay Request-Response Mechanism is the basic synchronization mechanism in the IEEE 1588 standard to synchronize clocks in a master-slave hierarchy. The basic principle of this method relies on measuring the time difference between the master and the slave clock, which is then used to synchronize a slave to the master clock. Figure 2 shows the basic pattern of synchronization message exchange to synchronize a slave clock to the master clock.

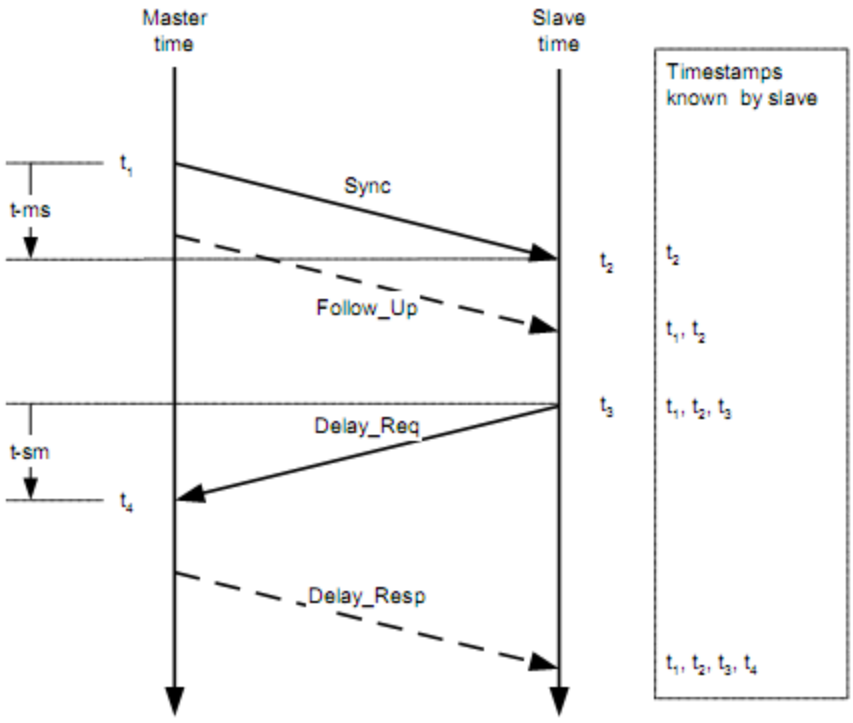

Figure 2: Basic Synchronization Message Exchange in Delay Request-Response Mechanism [17]

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

- The master sends a *Sync* message to the slave and notes the transmission time 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 transmission time 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.

At the end of this exchange of timing messages, the slave clock possesses all four timestamps t_1, t_2, t_3, and t_4. All these timestamps can be used to compute the offset of the slave clock with respect to the master clock and the mean propagation time of the messages between two clocks, which is the mean of t-ms and t-sm in Figure 2. The mean propagation time and the offset can be calculated using following equations:

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

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

$$Clock\ Offset(O_s) = t_2 - t_1 - Mean\ Propagation\ Time$$

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

The clock offset is directly applied to the time stamp of the slave clock [5], as follows:

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

The computation of an offset and the propagation time assumes that the master-to-slave and the slave-to-master propagation times are equal. Any asymmetry in propagation time introduces an error into the computed value of the clock offset. The computed mean propagation time also differs from the actual propagation times due to the asymmetry [17].

2.4 IEEE 1588: Factors Affecting Synchronization Performance

IEEE 1588 [17] devices are capable enough for achieving highly accurate clock synchronization. Nonetheless, it is important to recognize the key factors that directly affect the network synchronization performance.

2.4.1 Asymmetric Delay

Asymmetric delay is one of the major sources of error of transferring time from one clock to another. In PTP packet based networks, timing packets are exchanged between the master and the slave clock for the purpose of computing the time difference. If the packet exchange delay on the master-to-slave direction and the slave-to-master direction are identical, the offset can be computed precisely since the delays will cancel each other. Unfortunately, the path delay does vary between the master-to-slave direction and the slave-to-master direction, which is referred to as asymmetric delay. The synchronization accuracy is affected by the asymmetric delays in such a way that the degree of accuracy is one half of the asymmetric latencies. To make it worse, asymmetric delays may introduce by the packet delay variation (PDV), which is very difficult to characterize. Moreover, changes in the routing path on a longer time scale will also introduce differences in latency. These fluctuations will further aggravate the asymmetry problem.

Packet delay variation (PDV) or delay jitter in timing packets is another significant factor that deteriorates the synchronization accuracy severely in the IEEE 1588 system. PDV occurs as a result of queuing delays experienced by the timing packets at switching hubs such as routers, switches, cables and other hardware that exists between clocks. Queuing delays are very much dependent on the traffic load, topology of the

network and the path reconfiguration of the network. Packets received on a port are placed temporarily in a queue until they can be sent out. The whole process happens extremely fast unless the forwarding port is busy transmitting data packets. As a result, the fluctuation of queuing delays introduces PDV leading to timing errors in the path between two clocks. Considering the quality of service (QoS) and setting the timing packets to be the highest priority are effective in reducing queuing delay. However, queuing delay due to a single packet that has already started to transmit is unavoidable. In a switching hub with a Fast Ethernet interface, queuing delay varies by up to 122.4 μs if only one packet with a maximum transfer unit (MTU) size of 1518 bytes is in the buffer at the instance of arrival of a timing packet [21]. Another source of timing fluctuation is the equipment needed to translate between communication protocols in the networks [18, 22].

2.4.2 Timing Packet Rate

An increase in the rate of timing packets generally increases the synchronization performance but may reduce the effective throughput of a system [23]. More frequent transmission of timing packets might increase the accuracy of synchronization. But it also increases the overhead in the network, particularly when more network traffic exists. On the other hand, shortening the transmission interval leads to diminishing returns. So, the transmission frequency of timing packet is a considerable factor in order to balance between the desired accuracy of synchronization and the achievable throughput.

2.4.3 Process of Time Stamping

The time transfer latency problem is associated with processing of timing packets by the operating systems. How and where the timestamps are generated are key factors to

affect the time synchronization accuracy. Unlike hardware based time stamping, if the packet exchanges are instantaneous there will be no delay and the time difference can be computed accurately. On the other hand, a software based PTP slave program operates at the application layer of the server. A software based PTP computes the time difference in order to adjust the local clock. When a PTP message traverses the protocol stack in a node, it is queued in the buffer to process, which is a fluctuating value and will affect the synchronization accuracy. To achieve the best possible results, the IEEE 1588 time stamps should be generated in hardware or the timestamp should be captured as close as possible to the hardware or the NIC card [24].

2.4.4 Oscillator Quality

The synchronization accuracy also depends on the drift rate of the local oscillator. Temperature and aging are the major driving factors for oscillator drift as discussed in Section 2.1. Oscillator drift can be mitigated by using higher quality oscillators or by driving time from a more accurate source, such as GPS. However, higher quality oscillators and GPS may not be cost effective in terms of capital expenses, CAPEX. The PTP system must support a higher packet transmission rate in order to alleviate the drift of the local oscillator.

2.5 NS-2 Clock Model

Network Simulator 2 (NS-2) is an open source network simulation tool [25]. NS-2 is one of the prominent tools particularly designed for conducting research in computer communications networks. The primary use of NS-2 is to simulate various types of network protocols such as TCP, UDP, multicast protocols, routing algorithms, traffic source behavior, queue management mechanisms, etc. over wired and wireless

environments. It is an object-oriented, discrete event driven simulator written in C++ and the Otcl language [36]. We choose NS-2 because it facilitates to study network effects elaborately such as network congestion, temporary network outage, network path reconfigurations and many more, which are very difficult to model in other simulation tools, for instance MatLab.

Furthermore, other simulation tools do not have a clock model which is affected by temperature and aging effects. As described in Section 1 of this chapter, a clock oscillator starts to drift due to the environmental changes such as temperature and aging effect. Keeping this in mind, a clock agent is implemented by a prior student in NS2 via a C++ class hierarchy. Our proposed work extends the current model of such a clock agent. The details of the implementation of such a clock agent can be found in [4]. The major features of the implemented clock agent are summarized as follows:

- A basic version of IEEE 1588 synchronization protocol as discussed in Section 2.3.1.1 is implemented in NS-2 using a master-slave hierarchy.
- A clock agent is implemented with an initial value to act as a master or slave clock. The 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. Similarly, a slave clock has the ability to synchronize itself with the master clock using the IEEE 1588 protocol [17].
- The clock agent provides a way to capture the time stamps and natural rates of the clock at a regular interval. The natural rate determines the drifts of the clock w.r.t. to a reference clock, which is affected by temperature and aging effects.

- The implementation also provides the self-correcting Adaptive Oscillator Correction Model (AOCM) for both the master and the slave clock to account for the linear aging effects and linear or quadratic temperature effects, adopted from [26].

The implementation also considers locked and holdover periods for both the master and the slave clocks to reflect a network outage for master-GPS and master-slave connections respectively.

3 Chapter: Literature Review

This chapter provides a survey of research literature related to the clock synchronization models and performance of IEEE 1588 clock synchronization. Section 1 will focus on the fundamental limitations for synchronizing clocks over wired and wireless networks. Section 2 will review the performance of IEEE 1588 using only software assisted time stamping. The performance of IEEE 1588 using only hardware assisted time stamping will be discussed in Section 3. Finally, this chapter will be concluded with a motivation for our work in this area.

3.1 Fundamental Limitations of Clock Synchronization

Freris *et al.* [27], and Nikolaos *et al.* [28] analyzed the feasibility of clock synchronization and also characterized the fundamental limitations for synchronizing clocks over wired and wireless network. To study the problem, the authors assumed a simple clock model called affine clock which ran at a constant speed, not necessary with identical speed. Each clock is characterized by its relative speed i.e. skew with respect to a reference clock (i.e. master clock) as well as its offset. In addition, another model is considered for delays as the sum of a transmitter-dependent delay, receiver-dependent delay, and a known electromagnetic propagation delay. The authors only considered noiseless communication, where latencies are deterministic and time-invariant but unknown between any two nodes. Given these assumptions, the uncertainty set is characterized. The authors have shown the following impossibility results:

I. Using one master and one slave clock, it is shown that the determination of unknown clock offsets and link delays, particularly one way latency, is impossible even for the ideal scenario where neither offset nor skews drift with time.

II. When a single master clock coordinates with multiple slaves or a single slave clock coordinates with multiple master clocks, the study showed that it is possible to estimate all nodal skews and round trip times correctly between every pair of nodes. But delays and offsets of the nodes cannot be determined exactly by characterizing the uncertainty set because of (n-1) unknown offsets, where n is the total number of nodes. The authors remarked that the delays can be estimated up to one unknown offset for each node except the reference clock (i.e. master clock).

III. Even considering causality, i.e. packets cannot be received before they are transmitted, the same results are established as above. It does not reduce the region of uncertainty which is unbounded.

As a remark, the authors mentioned that if there is a known asymmetry in the delays that can be characterized, a unique solution exists. Also, the authors did not consider any temperature and ageing effects in this study.

3.2 Performance of IEEE 1588: Software Assisted Time Stamping

This section provides a literature survey related to the software assisted time stamping to assess the performance of the IEEE 1588 protocol.

3.2.1 IEEE 1588 Synchronization using a Queuing Estimation Method

T. Murakami *et al* [21] proposed a queuing estimation method by using probing packets to improve the synchronization accuracy with the IEEE 1588 protocol. Synchronization accuracy suffers due to the packet delay variation (PDV), which occurs at the queuing buffer in a switch/ router. The objective of the proposed method is to

estimate the occurrence of delay jitter in timing packets and filtering out the packets with delay jitter. To implement such a technique, a modified message sequence is added to the conventional IEEE 1588 protocol. According to the proposed mechanism, it is assumed that the inter packet gap is T_i. Occurrence of delay jitter at the i-th packet in the group ($2 \leq i \leq Np$) is estimated with the following criteria:

1. $If\ \left(T_{arr\,(i-1)} - T_{dep\,(i-1)}\right) > \partial T\ or\ \left(T_{arr\,(i-1)} - T_{dep\,(i-1)}\right) < -\partial T\ ;$

$$delay\ jitter\ occured\ in\ the\ packet.$$

2. $Otherwise, delay\ jitter\ did\ not\ occur\ in\ the\ packet$

The threshold ∂T is selected based on clock resolution and accuracy of the clock. Based on the estimation, the packet that has the smallest transmission delay is used for time adjustment. The authors performed experiments using the OPNET discrete event simulator. To configure the network topology, one master and one slave clock is considered and two switches are employed between the master and the slave clock. As result, it is shown that the accuracy of the slave clock is remarkably improved by using the proposed method. It also reduced the adverse impact of delay jitter on synchronization accuracy. However, the proposed method is effective only if at least some timing packets with no delay jitter are able to reach to the slave nodes periodically. In addition, this work does not deal with temperature and ageing effects.

3.2.2 IEEE 1588 Synchronization in a Congested Network: Packet Delay Distribution Estimation Method

T. Murakami *et al* [32] proposed a packet distribution estimation method to improve clock synchronization with IEEE 1588. In this work, the authors also adopted

the queuing estimation method [21] to suppress synchronization error in a heavily congested network. The proposed work relies on switching between two packet selection schemes, which depend on the defined threshold and window size of a target bin. A packet with no queuing delay is used to adjust the clock. The authors used the OPNET discrete event simulator to evaluate the performance of their packet filter techniques. To explain the network scenario, one master and one slave clock is used and four intermediate switches are employed between the master and the slave clock. The performance of the slave clock is evaluated in the presence of symmetric constant cross traffic, as described in ITU-T G.8261 (traffic model 2) [3]. The estimated result reflects the actual delay distribution which is effective for selecting the mode of distribution. The findings of this experiment are summarizes as follows:

I. When the mean traffic load is 50%, the slave clock is synchronized with the master clock by using the queuing delay estimation method. When the mean traffic load reached 80%, the slave continued its synchronization by applying the packet delay distribution filtering mechanism.

II. The proposed mechanism improved the synchronization accuracy in the presence of traffic. When symmetric cross traffic is applied, the fluctuation in timing error is within 5 μs. On the other hand, the slave maintains synchronization within 15 μs, when asymmetric cross traffic with sudden large changes is applied. The authors mentioned that the clock offset error is around 100 μs if the delay asymmetry correction is not applied. In the simulation, initial frequency offset in the slave clock is 1 ppm.

However, similar to [21], the proposed mechanism did not consider temperature and ageing effect that also have adverse impact on clock synchronization accuracy.

3.2.3 IEEE 1588 Synchronization: Asymmetric Communication Link in Packet Transport Network

S. Lv *et al.* [33] presented an enhanced IEEE 1588 time synchronization method to estimate the correct offset between the master and the slave clock. Considering the asymmetric latency, the proposed algorithm is based on an additional procedure named *OffsetCorrection* for computing the offset. The proposed *OffsetCorrection* mechanism relies on an additional message exchange added to the conventional IEEE 1588 procedure. The performance of the proposed method is evaluated by simulations with several assumptions: initial offset between master and slave clock is 50 s, the end-to-end delay of master-to-slave direction is 25 ms. Considering the asymmetric delay, the ratios of end-to-end delay of master-to-slave direction and slave-to-master direction range from 1:1 to 8:1. The result shows that the error between the estimated and the ideal value changes from 0.3 μs to 2.8 μs. The proposed method also shows better performance with the transparent clock mechanism described in [17]. However, the authors did not consider any standard traffic profile to evaluate the performance of the proposed method. In addition, they also did not consider any temperature and ageing effects.

3.2.4 IEEE 1588 Synchronization using Fixed Delay Ratio

Z. Du *et al.* [34] proposed an enhanced end-to-end transparent clock (E2E TC) mechanism with a fixed delay ratio in order to correct the time of the slave clock with respect to the master clock. A transparent clock provides an accurate distribution of the PTP protocol across multiport network components such as bridges, routers and switches.

The E2E TC does not act as a master or slave, but forwards all messages just as a normal switch. The proposed mechanism uses two rounds of message exchange. The rationale of introducing the additional message is to capture the difference between the ideal and real values of the residence time referred as Residence Time Error (RTE). The residence time in a transit node is a random variable uniformly distributed in the range of 200 ns to 1 ms. The proposed approach relies on several assumptions, details will be found in [34]. The performance of the proposed method is evaluated by computer simulations and compared with the traditional approach of the IEEE 1588v2 and the E2E TC scheme defined in the IEEE 1588 standard. The bias error of the slave clock is used to compare the above mentioned algorithms. The bias error is calculated as the difference between the calculated offset and the actual offset. The initial offset and the frequency accuracy of the slave clock are assumed to be 50ms and 100ppm respectively. The network is composed of five E2E TC devices between the master and the slave clock. The fixed delay ratio is set 1.2 to 3, and the upper limits of RTE are set to 100 ns, 1 μs, and 10 μs. The results are summarized as follows:

I. When the fixed short term frequency stability of the TC clocks is 10 ppb, the maximum bias error of the conventional IEEE 1588 method is about 1ms, and the value of the TC scheme is about 100 ns. In contrast, the proposed mechanism exhibits less than 10 ns bias error.

II. When the short term frequency stability of the TC clocks varies, the maximum bias error of the proposed approach and TC scheme are about 100 ns, whereas the conventional method displays 1 ms bias error.

However, the proposed method is not compatible with the original standard to ensure the coexistence and smooth evolution. Furthermore, replacing the legacy transit nodes (routers and switches) with a TC device is not a realistic approach in terms of capital expenses, CAPEX.

3.2.5 IEEE 1588 Synchronization: Dynamically Changing Asymmetric Wireless Links

S. Lee *et al.* [35] proposed an enhanced IEEE 1588 synchronization algorithm to compensate the offset error due to the dynamically changing data rate of wireless links. The data rate changes dynamically due to the movement of the mobile stations, the shared pipe characteristics of the conventional packet oriented wireless technologies, and the asymmetric duplexing method. The proposed mechanism relies on calculating the asymmetric ratio by introducing several new parameters of a dynamically changing wireless link between the master and the slave clock. In general, conventional cellular base stations allocate the data rate for uplink and downlink based on the available bandwidth, the air condition, and its associated buffers status. In the proposed method, the mobile station stores the data rate for each IEEE 1588 message traversing the wireless switch. The performance of the proposed mechanism is evaluated using computer simulation. The initial offset between the master and the slave clock is 25.2 ns.

The result shows that the bias error of the proposed method is reduced to 1 ps, whereas the bias error of the conventional IEEE 1588 [17] is roughly 100 μs. The bias error is calculated as the difference between the calculated offset and the actual offset.

The authors did not consider network congestion and path reconfiguration in the network to evaluate the performance of the proposed approach.

3.2.6 Combined IEEE 1588 and Adaptive Oscillator Correction Model

W. Ahmed *et al*. [4] proposed a solution that combines the traditional IEEE1588 protocol and the adaptive oscillator correction model (AOCM). The solution suggests a basic implementation of the IEEE 1588 protocol using the Delay Request-Response mechanism to synchronize the slave clock with respect to the master clock in a network. The solution also extends the AOCM model to train a slave clock locked to the master clock and apply that training model to correct the slave clock during the network outage or temporarily network failure. The extension of the AOCM model to the slave clock also intends to correct the temperature and ageing drifts of the oscillator and thus to improve the accuracy and stability of the clock. Here, using the IEEE 1588 protocol, the calculated offset in a slave clock with respect to the master clock is used to generate a correction signal as well as to train the AOCM model.

To implement such an approach, the IEEE 1588 protocol and the AOCM model for the master and the slave clock are implemented in NS-2. A clock agent is implemented in NS-2 to simulate a real clock having drifts due to temperature and aging effects [4]. To evaluate the performance of this work, test cases from an ITU-T document covering various network conditions and network loads are implemented [3]. The experimental findings are summarized as follows:

I. A slave to master synchronization accuracy of 1 μs is achieved after 306 synchronization updates from master to slave clock with 95 % confidence. The results indicate that 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. It also shows that the higher the IEEE 1588 synchronization frequency, the higher is the slave accuracy w.r.t the master clock and vice-versa.

II. The proposed solution improves the slave clock accuracy up to 10 times in the absence of traffic in the networks, depending on the length of the slave clock training period.

III. Considering traffic in the network, the results show that the slave accuracy is affected by the asymmetric latencies such that the degree of accuracy is half of the asymmetric latencies.

IV. When an asymmetric traffic such as 40% load in the master-to-slave direction and 30% load in the slave-to-master direction is increased to 100% in both directions to cause network congestion, the slave accuracy relatively improves because of relatively less asymmetric latencies.

V. 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. The results indicate that the slave accuracy decreases as the network outage period increases.

However, the author did not take the initiative to correct the asymmetry latency issue in terms of the slave clock accuracy.

3.3 Performance of IEEE 1588: Hardware Assisted Time Stamping

This section provides a literature survey related to the hardware assisted time stamping to evaluate the performance of the IEEE 1588 protocol.

3.3.1 IEEE 1588v2 Synchronization using Multicast Mechanism in a Packet Network

L. Xie *et al*. [29] proposed a multicast mechanism supported by the hardware over a packet network in order to improve the synchronization accuracy between the master and the slave clock. The proposed method focused on factors such as the location of the time stamps, frequency difference, delay asymmetry and error accumulation, which affects the synchronization accuracy. To implement such an approach, the authors considered all transparent clocks, which are used in the packet network and the master clock provides a stable reference time. The findings of the experiment are summarized as follows:

I. The slave clock frequency is synchronized to the master clock within 100 ppb after 3 synchronization intervals and the convergence period of the frequency synchronization is roughly 9 s. The frequency stability of the slave clock is about 10 ppb after the convergence period. The result also shows that the PTP starts up and performs initial clock reset after the first 21 synchronization intervals.

II. The slave clock time is synchronized to the master within 100 ns after 9 s and the convergence period of the time synchronization is roughly 9 s as well. The time offset between the master and slave clock remains within 50 ns after the

convergence period. This is mainly caused due to the frequency difference and delay measurement errors.

The authors did not consider other major factors such as temperature and ageing effects, which affects the synchronization accuracy. The authors also did not mention about the traffic load and the ratio of asymmetric link delays to evaluate the performance of the proposed concept.

3.3.2 IEEE 1588v2 Clock Synchronization using Controlled Packets

B. Mochizuki *et al.* [30] proposed a new mechanism in order to eliminate PDV effects on timing packets. The proposed method relies on a multiplexing process between the IEEE 1588 SYNC packet and the background traffic. The authors assumed that the background traffic source can be back pressured [31] to generate the idle time (i.e. headroom) so that the timing packet does not incur any queuing delay in the network. The idea is to store the background packets in the local buffer by the multiplexing state machine integrated with the master clock and transmit them in a regular way when no SYNC packets need to be transmitted. As the scheduled time of SYNC packet approaches, the state machine starts the gap timer. As soon as the gap timer expires, it suspends background traffic transmission and sends the SYNC packet. The gap timer is calculated by each SYNC packet to make sure the channel is free. The multiplexing state machine is implemented in FPGA. The experimental finding shows that packet delays over a 16 hop network range lie between 28.7 μs and 30 μs with a standard deviation of 170 ns. If the SYNC packet does not have any coordination with the background traffic,

packet delay ranges from 98.5 μs to 274.4 μs with a mean delay of 215.5 μs, and a standard deviation of 30.5 μs.

Since real time traffic is unpredictable, there is a possibility that more than 2 background packets are transmitted simultaneously when multiple packets are delayed. The authors did not focus on such situation. Moreover, the performance of the proposed method is evaluated with one traffic source only, which does not model a realistic network scenario.

3.4 Motivation

To conclude from the literature survey, the IEEE 1588 protocol plays a significant role to provide a high degree of accuracy and precision both in wired and wireless networks. Unlike NTP, it has the flexibility either by performing a regular message exchange or enhances itself further using the transparent clocks and/or the hardware-assisted time stamping for high precision and accuracy. From the literature survey, we have observed that various filtering techniques have been investigated to improve the slave accuracy with respect to the master clock in a network. A major focus dealt with asymmetric latencies and PDV, which deteriorates the synchronization accuracy severely. In order to overcome the degradation of synchronization performance, additional messages are appended with the conventional IEEE 1588, which may induce additional overhead. The performance of the IEEE 1588 protocol has been evaluated for a master-slave hierarchy without explicitly looking at the variation of the network conditions such as the variation of bi-directional traffic loads or network congestions, which lead to timing errors between the master and slave clocks. In addition, none of the published work attempted to correct the time difference when routing changes, caused by failure in the network and temporary network outage as well. Furthermore, none of the authors

demonstrate how a single slave clock will coordinate with multiple master clocks connected through multiple networks. This leaves an opportunity for future work to enhance the performance of the slave clock with respect to the master clock under a high traffic load in a network. Combined IEEE 1588 and AOCM method [4] provides the self correcting mechanism for both the master and the slave clock to reflect a network outage or temporary network failure for master-to-GPS and master-to-slave connections respectively. This work also considered the temperature and aging effects and investigated the slave clock performance covering various network loads. However, the slave accuracy is not good in the existence of traffic in the network. The objective of this thesis is to enhance the IEEE 1588 protocol by compensating the adverse effect of unpredictable packet delay variation on timing packets. The proposed solution further extends to support multiple master clocks connected through multiple networks having PDV or asymmetric latency effects as well. The resulting model also incorporates the adaptive oscillator correction method [4] on slave clocks such that when the slave clock receives IEEE 1588 synchronization updates, it trains an adaptive model. Finally, the proposed solution can then be evaluated for various test cases according to the variation of bi-directional traffic loads, network congestion, routing changes and temporary network outage.

4 Chapter: Proposed Work

In this chapter, we propose an enhanced IEEE 1588 clock synchronization algorithm, which combines the conventional IEEE 1588 protocol with a Delay Asymmetry Correction (DAC) Model. The goal of this proposed work is to achieve high accuracy by determining the correct offset value in a slave clock for asymmetric communication link delays. This chapter mainly focuses on the details of the proposed DAC model with flowcharts.

4.1 Overview of the Proposed Solution

The proposed work aims to enhance the IEEE 1588 protocol by computing the time difference between the master and the slave clock in a network. The focal point of this proposed work is to mitigate the effect of unpredictable packet delay variation on timing packets, which deteriorates the synchronization accuracy. The difference between the delays in the forward direction (master-to-slave) and the reverse (slave-to-master) direction depends on the characteristics or states of the networks. More specifically, asymmetric delays are introduced by the traffic loads, network topology, routing path reconfiguration and many more, which are very difficult to characterize in general. As mentioned earlier in Chapter 2, PDV is caused at the queuing buffer in switching hubs such as routers, switches, and other hardware that exists between clocks. The fluctuation of queuing delay depends on the processing time of the current data packet when the timing packet arrives at the "busy" router/switch. Hence, the variations in latencies are simply random both in the forward and the reverse direction. The timing error for the legacy IEEE 1588 algorithm will increase with the degree of asymmetric latency between the forward direction and the reverse direction. Therefore, it is very evident that the

asymmetric latencies affect the synchronization accuracy severely between two clocks in a packet network. The solution is based on the idea of incorporating the proposed delay asymmetry correction (DAC) model with the traditional IEEE 1588 synchronization protocol. The proposed solution exchanges the basic time stamped messages of IEEE 1588 protocol between the master clock and the slave clock using the Delay Request-Response Mechanism as discussed in Section 2.3.1. However, based on a metric we will introduce, offsets that are deemed to have been acquired using a message exchange that experienced very asymmetric latencies will be filtered out. The proposed solution further extends to support multiple master clocks with a single slave clock. In a real world scenario, a single master clock might coordinate with multiple slaves or a single slave clock might coordinate with multiple master clocks using multiple network connections. Keeping this in mind, a new equation is developed so that all master clocks initiate IEEE 1588 message exchange systematically. Hence the slave clock may be able to avoid not getting updates for a long period of time. Thus, it is possible to achieve high synchronization between the master and the slave clocks in a network.

Furthermore, the synchronization accuracy suffers due to different initial offsets, different oscillator rates, as well as the environmental factors such as the temperature and ageing effects which cause drifts both in the master and the slave clocks. Considering all of these factors, the proposed solution also integrates the Adaptive Correction Oscillator Method (AOCM) [4] to both the master clock and the slave clock. The purpose is to compensate the temperature and ageing effects of the local oscillator and hence to improve the accuracy and the stability of the clock during holdover mode. The master clock is locked to a GPS signal for its reference time and the slave clock is also locked

with the master clock in order to receive the IEEE 1588 synchronization updates from master clocks at regular intervals to keep the slave accuracy high. The adaptive model is trained to compensate the drifts due to its rate, temperature and ageing effects. When the GPS signal is lost or the slave clock is temporarily disconnected from the master clock due to some network outage, the timing module enters into the holdover mode and then the synchronization accuracy depends on the quality of the local oscillator [4]. In this situation, the correction signal obtained from the AOCM trained model is used to correct the rate of the master oscillator, when the master clock loses its connection to the GPS signal. Furthermore, the slave clock has two kinds of holdover periods, which are termed as micro-holdover period and macro holdover period. A micro-holdover period for a slave clock is the time interval during which the slave clock is waiting for its next IEEE1588 synchronization update from its master clock. A macro-holdover period for a slave clock is the time interval during which the slave clock is temporarily disconnected from the master clock due to some network outage. The correction signal obtained from the AOCM trained model is used in both cases to adjust the slave oscillator.

In a nutshell, the proposed solution aims to achieve high synchronization accuracy by applying our proposed DAC model with the legacy IEEE1588 protocol considering the PDV across a network in the presence of various traffic loads. The aforementioned method reduces the offset error, which is caused by asymmetric latencies of the communication links.

4.2 Delay Asymmetry Correction (DAC) Model

The DAC model is used to determine good samples in order to update the slave clock correctly under heavy traffic load in the network. The proposed DAC model is using a two stage filtering method. In the first stage, we use the delay asymmetry ratio between the master and the slave clock differences, called R, which we also refer to as 'R' test. It is worth mentioning that the term 'R' test mentioned in the thesis does not refer to the statistical test of the same name. The second stage relies on updating the samples which pass the first test. The rationale of this DAC model is to ensure that only good samples are used to update the slave clock so that the synchronization accuracy remains high. Moreover, this process not only involves filtering out bad samples. We also save a notion of good updates, so that we can apply these saved values when we have no samples from the IEEE 1588 protocol. Otherwise the slave clock would run uncorrected for potentially long periods of time. The flowchart of the proposed DAC model is as follows:

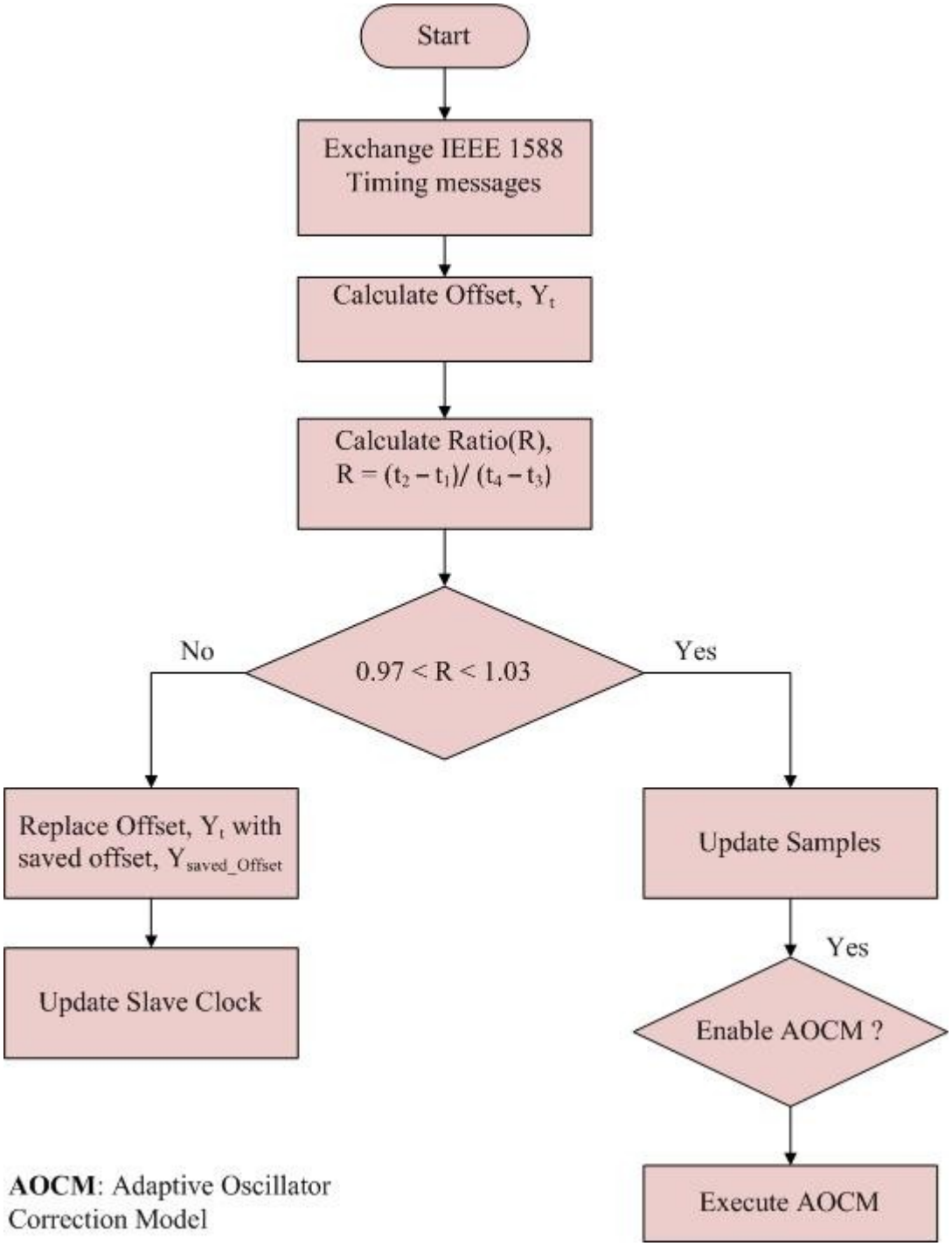

Figure 3: Flowchart of the Proposed DAC Model

As an explanation of the above flowchart, first, the master clock and the slave clock will exchange the basic timing messages based on the delay request-response mechanism

defined in the IEEE 1588 Std. [17]. The timing message sequences are shown in the following Figure 4.

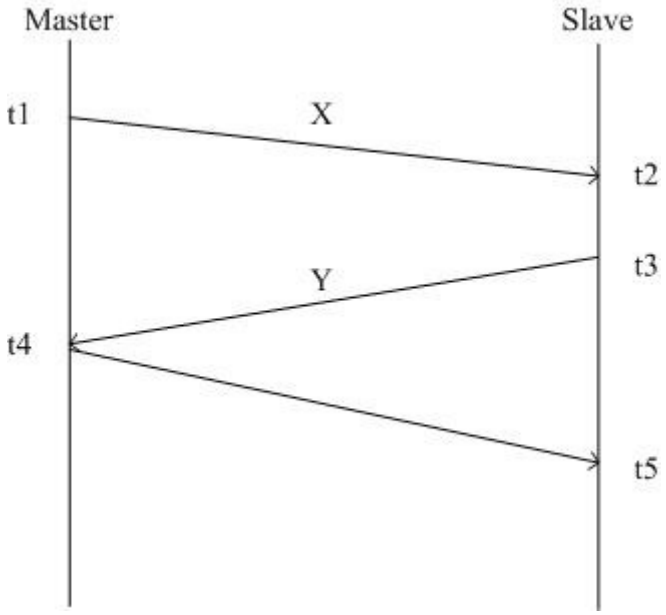

Figure 4: IEEE 1588 Timing Diagram

Let X be the transmission time from the master to the slave clock and Y be the transmission time from the slave to the master clock. X ≠ Y, when latencies are asymmetric. The master to the slave time difference, t_{m2s} is defined as the time difference between the receiving timestamp of the slave clock (i.e.t_2) and the sending timestamp of the master clock (i.e.t_1). Similarly, t_{s2m} corresponds to the time difference between the receiving time stamp of the master clock (i.e. t_4) and the sending timestamp of the slave clock (i.e. t_3), t_{md} is the mean propagation delay between the master and the slave, the offset is represented by t_{offset} . t_{m2s} and t_{s2m} are defined as follows:

$$t_{m2s} = (t_2 - t_1) \quad \ldots\ldots\ldots (1)$$

$$t_{s2m} = (t_4 - t_3) \quad \ldots\ldots\ldots (2)$$

The offset is calculated using IEEE 1588 message sequences as follows:

Offset, $t_{offset} = t_2 - t_1 - Mean\ Propagation\ time$

$= [(t_2 - t_1) - (t_4 - t_3)]/2$ …….. (3)

Where, Mean Propagation time, $t_{md} = [(t_2 - t_1) + (t_4 - t_3)]/2$ ……. (4)

Since the nature of the network traffic is unpredictable and may shift over time, the dominant delay might vary either in the forward direction (master-to-slave) or in the reverse direction (slave-to-master). If the clocks were perfectly synchronized, we could actually measure X and Y, but the timestamps are based on clocks that are not perfectly synchronized. Therefore, we can only estimate X and Y using the delay asymmetry ratio, R. The ratio, R, represents the estimated delay asymmetry ratio between the master to the slave clock difference (i.e. t_{m2s}) and the slave to the master clock difference (i.e t_{s2m}).

So, the estimated delay asymmetry ratio, $R = t_{m2s} \ / \ t_{s2m}$ ……. (5)

$= (t_2 - t_1)/(t_4 - t_3)$

= (Offset + X) / (- Offset + Y) …… (6)

Equation 6 includes both the offset and the actual transmission time of timing packets. It should be noted that the estimated delay asymmetry ratio, R, will provide an indication of asymmetric latencies, in particular when the clock offset is small relative to the actual transmission delay.

However, a boundary condition, $0.97 < R < 1.03$, is introduced as part of estimating good samples. To explain the feasibility of boundary condition, the cumulative distribution function (CDF) with respect to the ratio (R) is depicted in Figure 5. Here, the slave clock is drifting 100ppb faster than the master clock and the synchronization frequency is 1 second. The data of Figure 5 are collected for 24 hours of simulation time

when 80% static traffic load is introduced in the forward direction and 20% static traffic load in the reverse direction.

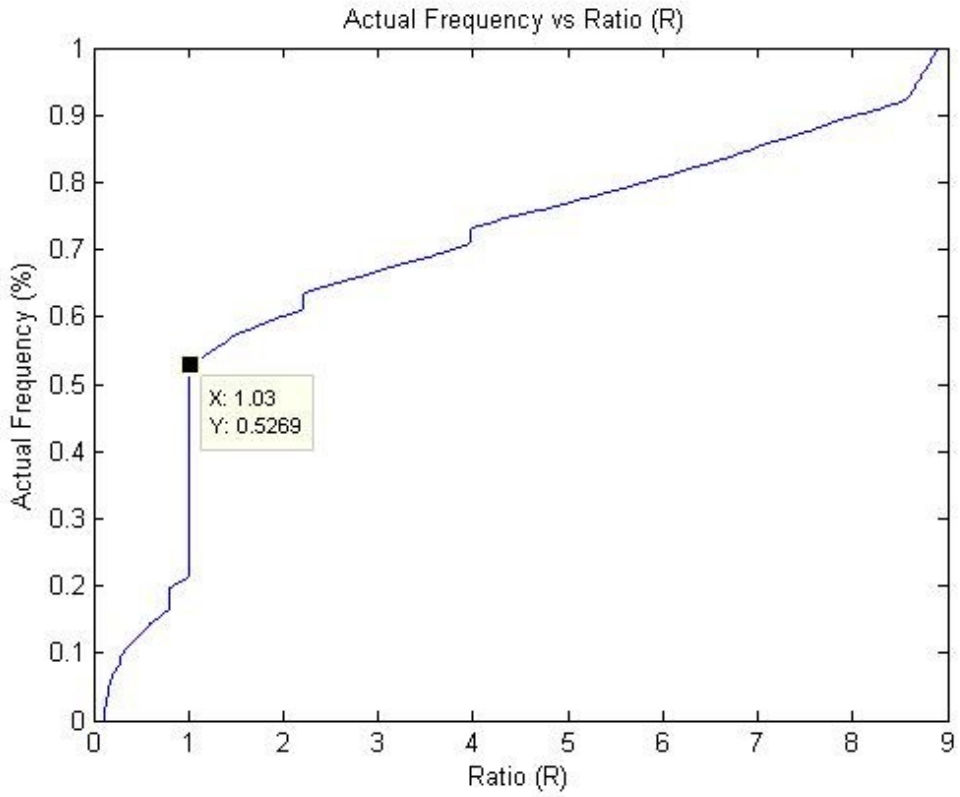

Figure 5: CDF w.r.t the Ratio (R)

From Figure 5, the distribution shows that about 21% samples fall below a ratio of 0.97 and about 53% samples fall below a ratio of 1.03. Thus, approximately 32% of all samples lie within the defined boundary $0.97 < R < 1.03$, when latencies are deemed to have been symmetric between master-to-slave and slave-to-master directions. The remaining 68% samples experienced significant delay differences between the master clock and the slave clock due to the presence of traffic in the network. We believe that it is possible to achieve high synchronization accuracy of the slave clock in the presence of bursty traffic by using 32% of all samples within 3% variance. Since the distribution is

plotted under high traffic load, there is a good possibility to receive more samples under less traffic loads.

The calculated offset values which pass the 'R' test, are fed into a 2nd stage filter, named as update sample filter. As the name suggests, the 2nd stage filter is implemented to ensure that only good samples are used to update the slave clock to keep the synchronization high. It has been observed that some clock packets passed the 'R' test with higher or lower offset values than the anticipated offset. So, those timing packets are considered as outlier samples. The slave accuracy could be severely affected when it is updating with an outlier. Table 1 shows exemplary one set of data which passed the 'R' test. These data are also collected with 80% traffic load in the forward direction and 20% traffic load in the reverse direction. The slave clock is drifting 100ppb faster than the master clock and the synchronization frequency is 100s. Thus, the maximum clock difference would be 10 μs, assuming the slave clock could be perfectly synchronized in each synchronization interval. Total simulation time is 24 hours and 8640 samples are generated during that period of time.

Table 1: Statistical Data Collected from 'R' Test only

Simulation time (s)	Ratio (R)	Calculated Offset (μs)	Synchronization Intervals	Saved offset = Offset/Sync Interval(μs)	Clock Difference (μs)
1	1.00	0.0	1	0	0.0
201	1.02	20	2	10	0.0
701	1.02	10	5	2	0.0
19101	0.98	-8.14	184	-0.044	1482.1

It is evident from the ratio column that the calculated offsets listed in Table 1 passed the 'R' test. Only 3 samples passed the 'R' test out of 8640 samples. We are not considering the first sample as it is used to set the time of the slave clock, i.e., correct for initial clock differences between the master and the slave. In addition, the accumulated offset may not be the anticipated offset in the very first second. However, the second sample passed the 'R' test with an offset value of 10 μs after 2 synchronization intervals. The number of synchronization interval is counted since the last successful update. The slave clock is updated according to that sample value and then a saved offset value is calculated. Similarly, the slave clock is updated with the third sample. It is noticeable that the third sample passed the 'R' test after 5 additional synchronization intervals with an offset value of 10 μs. The slave clock is updated and the calculated saved offset is 2 μs. The saved offset should be close to the actual offset in order to update the slave clock correctly when highly asymmetric latencies exist. Otherwise, clock difference will accumulate over time. The slave clock is updated with that saved offset for the next 183 intervals. As a result, the clock difference is increased to about 1464 μs. The fourth sample appears after 184 intervals with an offset value of -8.14 μs, which is an outlier. The slave clock is updated with that sample and the saved offset is calculated according to that sample. Afterwards, none of the samples passes the 'R' test, even if a good sample appears after a long interval as the calculation of 'R' becomes distorted at some point. 'R' is an approximation of the undefined asymmetry ratio and subject to significant distortion when the offset is roughly on the same order of magnitude as the one-way latencies (i.e. X or Y). Hence, the saved offset is used to update the slave clock and the slave clock difference accumulates until the end of the simulation. Thus, it is necessary to restrain

those outliers before updating the slave clock. Therefore, an update sample filter is implemented to suppress outliers and smooth the sequence of incoming samples. Finally, the filtered samples are fed into AOCM to train its model in order to reduce the current drift of the slave clock according to the correction value obtained from the AOCM model. It is also essential to provide good samples to train the AOCM model. Otherwise, the clock accuracy will suffer due to training this correction model with noisy or poor samples. Thus, these two filtering mechanisms ensure that AOCM trains the model with good samples only.

However, as soon as the slave clock realizes that the latencies are highly asymmetric due to the existence of traffic flows and the calculated offset does not pass the 'R' test, the recently calculated offset value is replaced with a previously stored offset value. Such a stored offset value has previously passed both filtering stages. Finally, the slave clock adjusts the time according to the saved offset value. The detail flowchart of the update sample filter is illustrated in Figure 6:

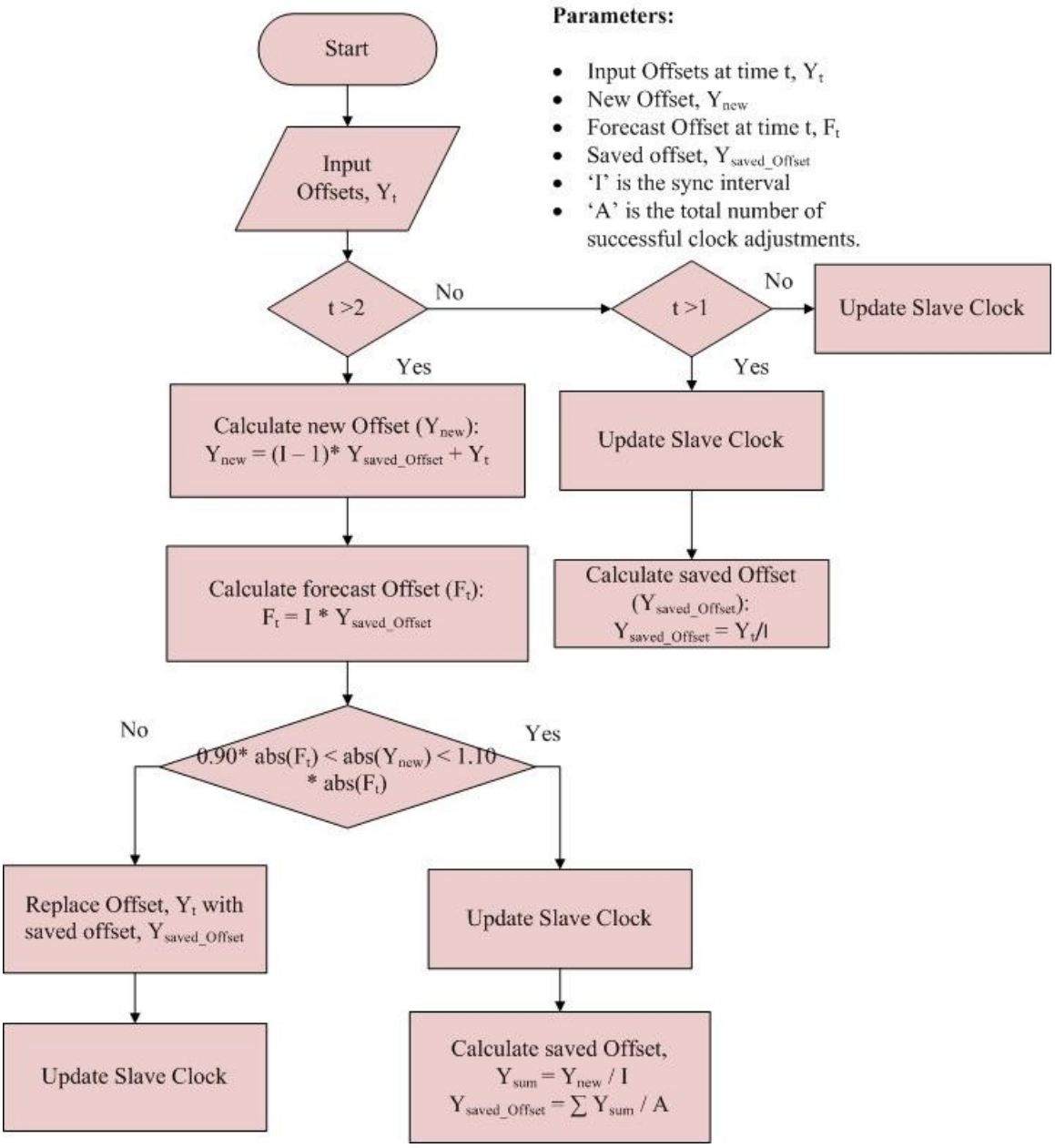

Figure 6: Flowchart of the Update Sample Filter

As mentioned earlier, the offset samples which pass the 'R' test are fed into the update sample filter as input offsets, Y_t. Three new equations, namely new offset (Y_{new}), forecast offset (F_t), and another boundary condition are introduced in order to restrain the outliers. According to the input offset, the total amount of slave clock adjustment to be applied is the new offset, Y_{new}. The new offset is computed as the multiplication of the sync

intervals with the saved offset value, which is then added with the recent input offset. The new offset is calculated as follows:

$$Y_{new} = (I - 1) * Y_{saved\_Offset} + Y_t \ldots\ldots\ldots\ldots (7)$$

Following the input offset, the forecast offset, F_t, is calculated for the upcoming offset. The forecast value is computed as the multiplication of the sync interval with the saved offset value, which is formulated as follows:

$$F_t = I * Y_{saved\_Offset} \quad \ldots\ldots\ldots\ldots (8)$$

Since the computation of the forecast process entirely depends on the saved offset value, it is essential to make sure that the saved offset is a good value. To set the forecasting procedure in motion, a boundary condition is introduced, which is defined as follows:

$$abs(F_t) * 0.90 < abs(Y_{new}) < abs(F_t) * 1.10 \ldots\ldots\ldots (9)$$

The boundary condition defined in Equation 9 compares the newly calculated offset value with the forecast value in order to suppress outliers. If the newly calculated offset value passes the boundary condition, the offset to be applied for the slave clock adjustment is Y_t. We are also keeping a track record of all successful clock adjustments for calculating the saved offset as well as spreading out the adjustments over time. The saved offset value is defined as the average of all previous saved offsets with the total numbers of successful clock adjustments, computed as follows:

$$Y_{saved\_Offset} = \sum Y_{sum} / A \ldots\ldots\ldots (10)$$

Where A is the total number of successful adjustments and $Y_{sum} = Y_{new} / I$; I is the sync interval.

However, if the newly calculated offset value does not pass the boundary condition, the offset, Y_t, is replaced with the saved offset, Y_{saved_Offset}. Finally, the slave clock is

updated with the saved offset value, Y_{saved_Offset}. In a nutshell, both the 'R' test and update sample filter are used to determine only good samples to update the slave clock correctly. These two filters also ensure that the synchronization accuracy between the master and the slave clock remains high.

The next section is going to focus on the extension of the proposed mechanism, through which a single slave clock supports multiple master clocks connected through multiple networks having asymmetric latencies due to the presence of traffic in the networks.

4.3 Supports for Multiple Master Clocks in NS-2

As mentioned earlier, a single slave clock might coordinate with multiple master clocks connected through multiple networks. The solution suggests that all master clocks will start to exchange IEEE 1588 messages with the slave clock according to the following equation:

$$1 + m * \left(\frac{X}{N}\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (11)$$

Where X is the synchronization frequency, N is the total number of master clocks, and $m = 0,1,2, \ldots . . N - 1.$

This process enables the slave clock to update its time more frequently. In other words, the slave clock does not need to rely on the saved offset for a longer period of time. It is worth mentioning that the synchronization frequency for the slave clock is the same as the IEEE 1588 update frequency (i.e. X seconds) received from the master clock. For instance, a slave clock is connected with two master clocks (Master 0 and Master 1) in a

network. Figure 7 illustrates the timing messages sequence between the slave clock and two master clocks.

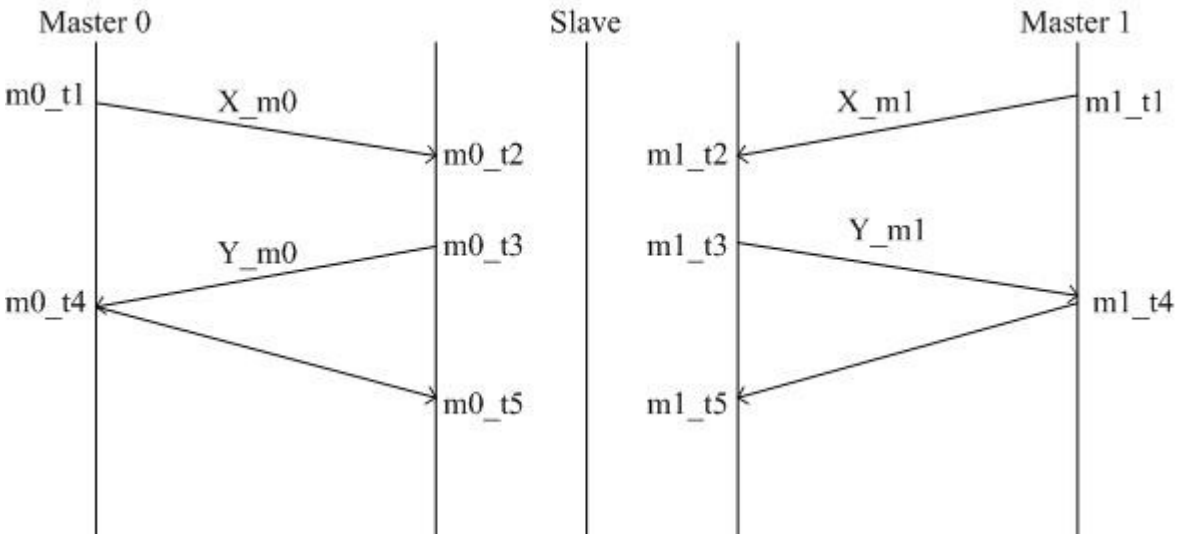

Figure 7: Timing Diagram for Multiple Master Clocks using IEEE 1588 Messages

Let the synchronization frequency be X seconds. If master (0) starts to exchange IEEE 1588 messages at time m0_t1, master (1) will initiate to exchange IEEE 1588 messages at time $(m_0\_t_1 + X/2)$ (i.e. m1_t1) seconds. Hence, the slave clock will be able to receive an update message every X/2 seconds instead of X seconds. Since the network condition is random, there is a possibility that the states of the network between the master clock and the slave clocks may be different. In this situation, we expect to receive at least one good offset sample from one of these master clocks within the defined update frequency interval in order to update the slave clock correctly. Thus, we do not need to rely on saved offset values for a long period of time to update the slave clock. Consequently, we expect to have a relatively good saved value.

The offset is calculated using the IEEE 1588 timing diagram 9 as follows:

Offset w.r.t Master (1):

$$m_1\_t_{offset} = (m_1\_t_2 - m_1\_t_1) - (m_1\_t_4 - m_1\_t_3)]/2 \quad \text{--------- (12)}$$

Offset w.r.t Master (2):

$$m_2\_t_{offset} = (m_2\_t_2 - m_2\_t_1) - (m_2\_t_4 - m_2\_t_3)]/2 \quad \text{---------- (13)}$$

The delay asymmetry correction mechanism will follow the proposed DAC model as in the case of a single master clock.

The performance of the proposed solution is evaluated by implementing it in the NS-2 simulator [25]. For this purpose, a clock agent implemented in NS-2 [4] is used, which is able to simulate a real network clock having drifts due to temperature and ageing effects. A clock agent can be attached to an NS-2 node. The details for creating a clock agent with different parameters configuration in a TCL (Tool Command Language) script are discussed in [4]. According to the proposed solution, the DAC model is also implemented in NS-2 in conjunction with IEEE 1588 protocol. In addition, it integrated the AOCM model for the master and slave clocks implemented in NS-2. Finally, NS-2 test cases are implemented according to the ITU-T document [3] covering various network disturbance loads and network conditions. The results of these experiments are discussed in the next chapter.

4.4 Summary

This chapter proposes a Delay Asymmetry Correction (DAC) Model to enhance the traditional IEEE 1588 synchronization protocol for asymmetric communication links. The proposed solutions are summarized as follows:

- The DAC model is proposed to achieve high synchronization accuracy by determining the correct offset value in a slave clock for asymmetric communication link delays. The initiative revolves around the idea of incorporating the DAC model with the conventional IEEE 1588 synchronization protocol. The proposed DAC model relies on two consecutive filtering methods named as the 'R' test and Update sample filter, which make sure that only good samples are used to update the slave clock.
- The filtering process of the proposed work does not only filter out bad samples, it also saves a notion of good updates for calculating a saved offset value. The latter value is used when the slave clock does not receive (good) samples from the IEEE 1588 protocol.
- Furthermore, the proposed solution further extends to support multiple master clocks updating a single slave clock. The rational of the extension is to support multiple master clocks instead of selecting a grand master clock. To do so, a new equation is developed. The solution suggests that the master clocks will initiate IEEE 1588 message exchange according to that equation for achieving high synchronization accuracy between the slave and the master clocks.
- The proposed solution also integrates the Adaptive Oscillator Correction Model (AOCM) in order to compensate temperature and aging effects of the oscillator and hence, to improve the stability of the slave clock during holdover mode.

5 Chapter: Simulation Results

In this chapter, the performance of the proposed DAC model is evaluated. The results are obtained using NS-2 simulations by measuring the synchronization accuracy of the IEEE 1588 protocol combined with the proposed DAC model. First, the generic NS-2 simulation set up is discussed. The test cases in this chapter are designed according to the ITU-T G.8261 document [3], covering various network loads and network conditions and their effect on the slave clock synchronization. While we tested the proposed scheme for all test cases in this document, this chapter only presents a subset of the results, with the reminder, showing similar characteristics, summarized in the appendix. The test cases are divided into the following categories:

- A test case with no traffic is discussed in Section 5.2.
- Test cases with data traffic are introduced in the network using different load profiles discussed in Section 5.3.

Furthermore, in this chapter, we do not consider temperature and aging effects, nor do we enable AOCM on the slave clock.

5.1 Simulation Setup

The network topology shown in the following Figure 8 is used in all simulations test cases unless a different topology is specified in a test case. The network topology consists of a master node n0 connected to a slave node n3 with two intermediate nodes n1 and n2, making it a 3 hop topology. Two traffic sources are also introduced. One of the traffic sources is node n4, which sends traffic to node n5. Another traffic source is node n5, with traffic destined to node n4.

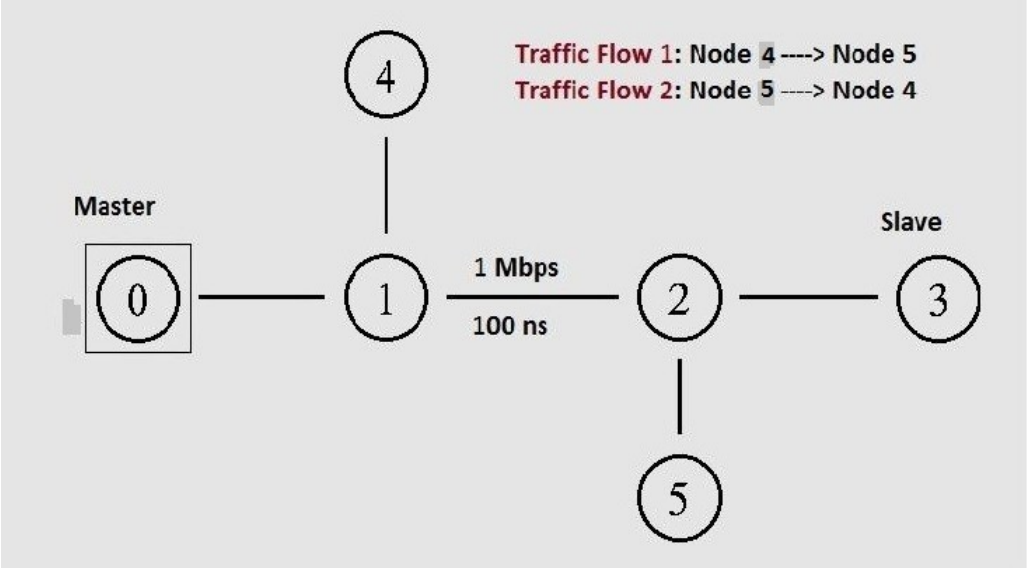

Figure 8: Network Topology

By default, the following values are used for different simulations 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 = 24 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 = 1 s
- Number of hops (between master and slave clocks) = 3 hops
- Slave node temperature effect when enabled = 5ppb/75°C with quadratic term equal to -0.00031966ppb/$°C^2$
- Slave node aging effect when enabled = 3ppb/3days
- Adaptive Model on Slave Clock (when enabled)

o Training period (locked mode) = AOCM will train its model only with those values which passes both the 'R' test and the update sample filter.

o Holdover period (unlocked mode) = Not considered in current setup.

o AOCM frequency = 1s

Note: The AOCM frequency of 1 second applies to the holdover period of the slave clock only. 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 Models Description

There are two traffic models described in Appendix VI in ITU-T G.8261 [3]. One of the traffic models is data centric traffic model, where the majority of the traffic is data. Another traffic model is voice centric, where the majority of the traffic is voice. First, we will use data centric traffic model to evaluate the performance of the slave clock synchronization accuracy w.r.t the master clock of the IEEE 1588 protocol. The results with voice centric traffic model are discussed in the Appendix C. Both of these traffic models are discussed in the subsequent subsections.

5.1.1.1 Data Centric Traffic Model

In data centric traffic model, 60% of the load must be based on packets of maximum size while the remaining 40% of 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)

5.1.1.2 Voice Centric Traffic Model

Voice centric traffic model is based on the typical traffic characteristics of the wireless access networks. According to 3GPP, the access traffic is composed of conversational (voice), streaming (audio-video), interactive (e.g., http) and background (sms, e-mail) traffic. It is known that in wireless network, 80% to 90% of the traffic is conversational voice traffic, with the average call lasting from 1 minute to 2 minutes. To model this traffic, 80% of the load should be based on packets of fixed small size constant bit rate (CBR), and 20% based on packets with a mix of medium and maximum size.

Hence the packet size profile can be summarized as below:

- 80% of the load must be minimum size packets (64 bytes)
- 15% of the load must be maximum size packets (1518 bytes)
- 5% of the load must be medium size packets (576 bytes)

In both these traffic models, maximum size packets (1518 bytes) will occur in bursts lasting between 0.1 s and 3 s. For each burst event, the burst length will be selected randomly using an identically independent uniformly distributed random generator. The network load of the traffic profile is individually described in the test case description whenever required.

5.1.2 Metrics Collected

In general, for each test case, the key performance data collected is as the maximum clock difference in each synchronization window, which is plotted versus simulation time (in hours). In addition, statistics for several other parameters are collected for each test case and provided in a separate table. Each test case is run 50 times unless it

is explicitly mentioned and total simulation time is 24 hours. To show the worst-case behavior, rather than averaging clock differences per synchronization window, we plot the maximal value for each synchronization window across all 50 repetitions.

5.1.3 Slave Clock Synchronization

The slave clock synchronization accuracy is measured with respect to the master clock in the network. We assume that the master clock is perfectly synchronized with the reference clock, i.e. GPS. Since the IEEE 1588 synchronization happens every 1 second (default synchronization window), the slave clock drifts from the master between the configured synchronization interval. Therefore, the maximum slave drift in each synchronization window can be used to determine the slave clock accuracy. Hence, the absolute difference and maximum of the maximum difference (in ns) in each synchronization window between the time stamps of the slave node and the master node is considered.

5.2 Test Case with No Traffic

The test case presented in this section is run with no traffic in the network using the topology shown in Figure 8. From the following Figure 9, the result shows that the delays experienced by the IEEE 1588 synchronization packets are symmetric and do not affect the IEEE 1588 synchronization accuracy. As the slave clock is drifting by 100 ppb, we could expect that during a synchronization window of 1 second the slave would deviate from the master or reference clock by 100 ns and each synchronization step perfectly synchronizes the slave, which indeed is what Figure 9 shows.

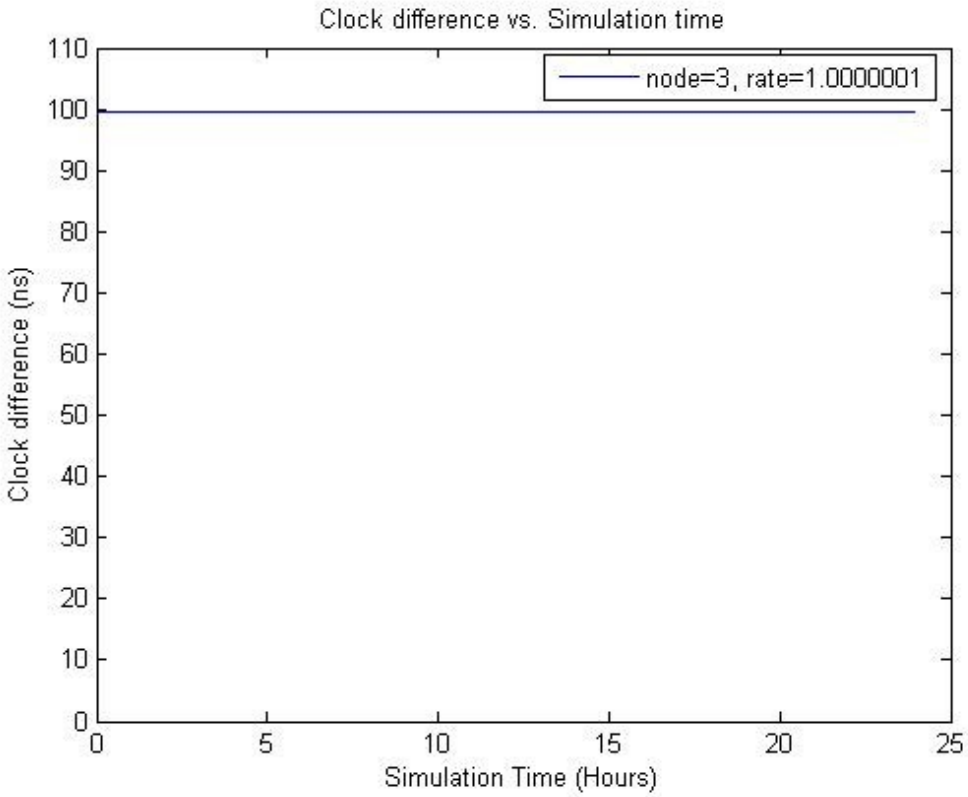

Figure 9: Slave Clock Accuracy w.r.t the Master clock- No Traffic in the Network

5.3 Test Cases with Traffic

The test cases presented in this section are run by introducing data traffic using different traffic load profiles. The network topology shown in the Figure 8 and the traffic model described in Section 5.1.1 are used. First, the slave clock accuracy is measured with a single simulation run without imposing the proposed DAC model. Later on, the slave clock accuracy is measured considering the proposed DAC model, using 50 repetitions. Each test case demonstrates the effects of various load profiles on the slave clock accuracy. The traffic in the forward direction (master-to-slave) is introduced at node 4 and destined to node 5. Similarly, the traffic in the reverse direction (slave-to-master) is introduced at node 5 and destined to node 4. Statistics for several parameters

are provided in a separate table in order to examine the effects of the various filtering stages. As mentioned, we are not considering any temperature and aging effects in order to evaluate the performance of the slave clock here. Only a few test cases are presented. The additional test cases for the data centric traffic model are described in Appendix B.

5.3.1 Static Packet Load– with the IEEE 1588 Message Sequences only

5.3.1.1 Description

In the first test case with traffic scenario, the traffic model described in Section 5.1.1, a static network load of 80% in the forward direction (master-to-slave) and 20% in the reverse direction (slave-to-master) are introduced for 24 hours, starting at a simulation time of 1 second and stopping at time 24 hours of the simulation time. The test case examines the performance of the slave clock synchronization accuracy using IEEE 1588 message sequences only with a static network load in both directions.

5.3.1.2 Results

Figure 10 shows the slave clock difference with respect to the master clock before applying the proposed DAC model and the difference of delays experienced (forward - reverse) when a static packet load is introduced between the slave and the master clock.

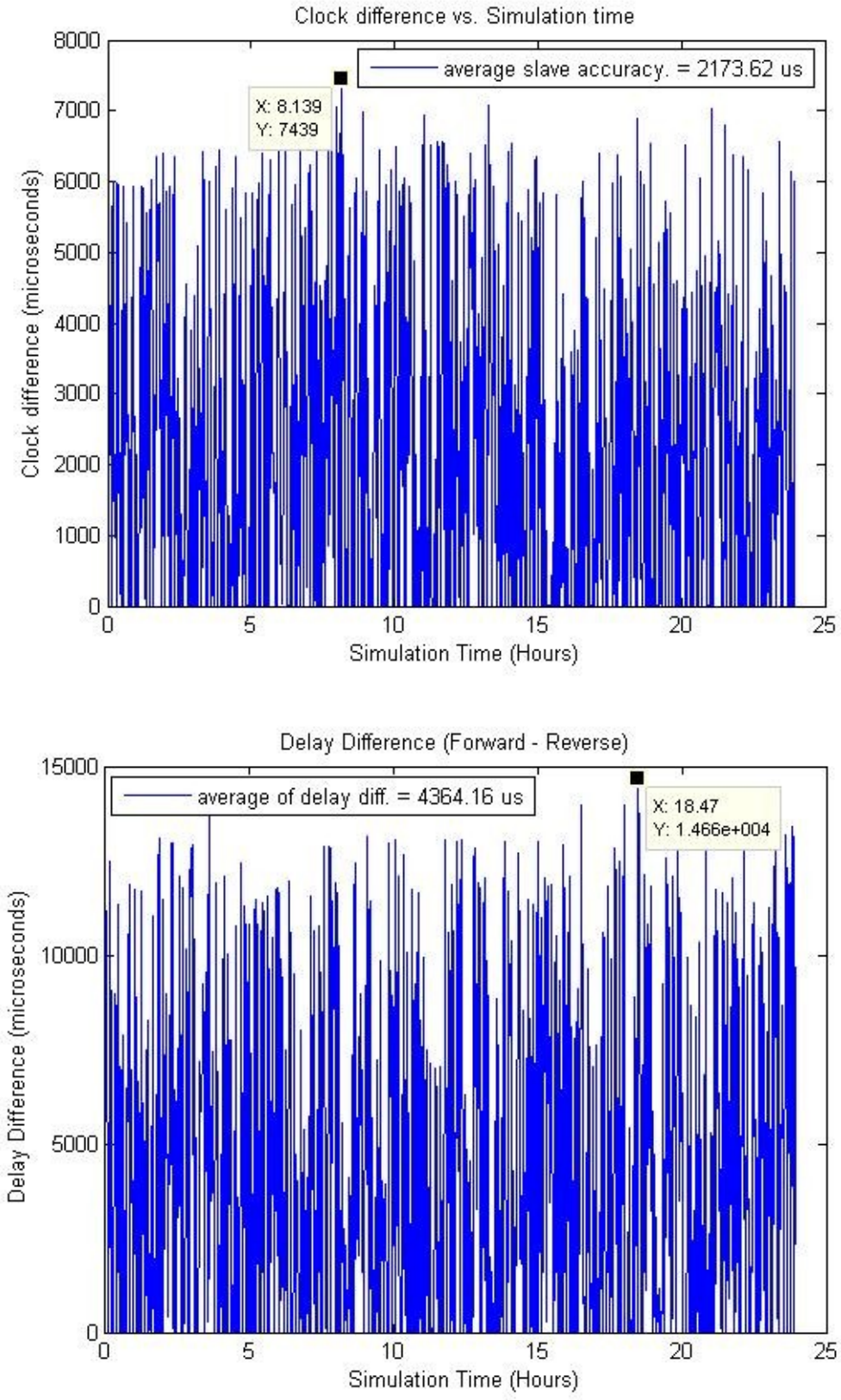

Figure 10: Slave Clock Synchronization using the IEEE 1588 Message Sequences only

5.3.1.3 Discussion

From Figure 10, the slave to the master clock difference is as high as 7439 μs and the average slave accuracy is about 2173 μs before applying the DAC model, which indicates that the slave synchronization accuracy deteriorates significantly in the presence of traffic in the network. With 80% static traffic load in the forward direction and 20% static load in the reverse direction, the delay difference (forward - reverse) can be as high as 14660 μs and the average delay difference is about 4364 μs level. The resultant difference implies that the delays experienced by the IEEE 1588 synchronization packets are highly asymmetric. Thus, it is evident that the slave clock inaccuracy introduced by the traffic is in direct relationship with the asymmetric latencies in such a way that the degree of inaccuracy is half of the asymmetric delays.

5.3.2 Static Packet Load – with the Proposed DAC Model

5.3.2.1 Description

In this test case, the first test case with a static packet load is repeated, this time filtering IEEE updates with the proposed DAC model. The NS-2 TCL script of this test case is described in Appendix A.

5.3.2.2 Results

Figure 11 shows the slave clock synchronization w.r.t the master clock, when a static packet load is introduced between the slave and the master clock with DAC model applied on the slave clock. Statistical data are provided in Table 2.

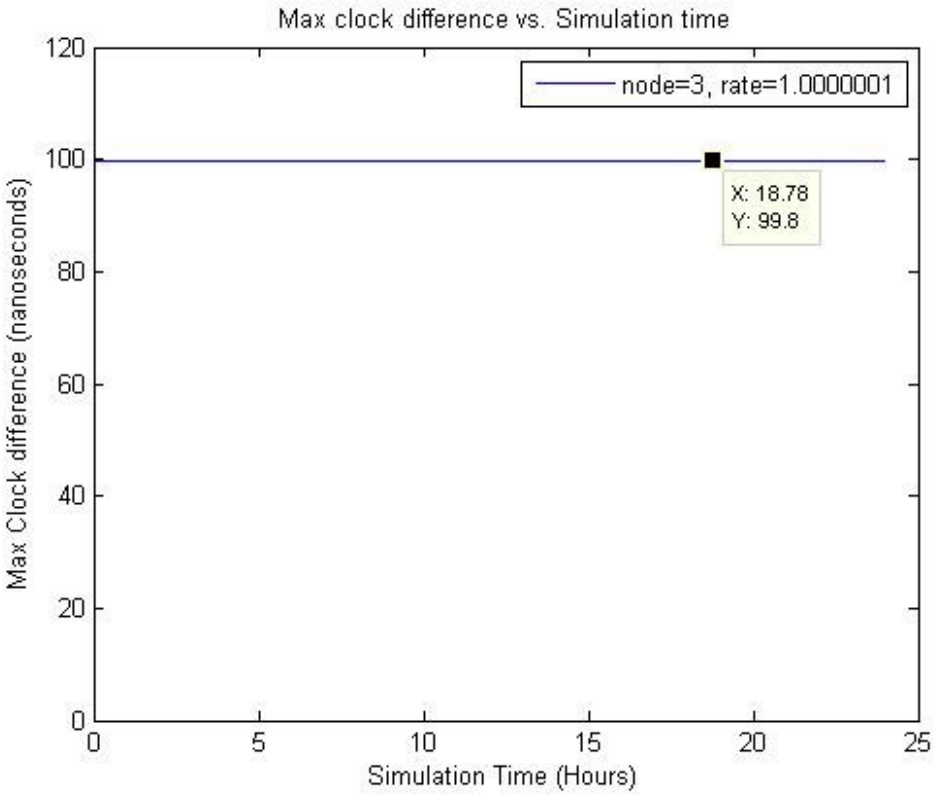

Figure 11: Slave Clock Synchronization with Static Packet Load-DAC Model Applied on the Slave Clock

Table 2: Statistical Data for Static Packet Load

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	22664 (26.23%)
Avg. no. of samples passed the update sample filter	21895 (25.34%)
Max. of the max. interval between successive successful updates (seconds)	81.87 s
Avg. of the avg. clock accuracy	99.71 ns
Max. Of the max. clock difference (ns) with DAC model	99.80 ns

5.3.2.3 Discussion

From Figure 11, the result shows that the maximum of the maximum slave clock difference is about 99.80 nanoseconds level. The resultant difference implies that the slave clock accuracy improved significantly in the presence of heavy traffic load in the network, much lower than the one shown in Figure 10. It is worth mentioning that we are not comparing the performance of the proposed DAC model with other solutions except IEEE 1588 message sequence only as shown in Figure 10. Because if we get the slave synchronization accuracy to 100 ns, that is as good as we can make it considering given parameters (i.e. 1 second synchronization interval, 100 ppb faster drift of the slave clock w.r.t the master clock). Thus, every time we get a result close to 100 ns, we got the best possible performance of the proposed DAC model. However, with 80% static traffic load in the forward direction and 20% static traffic load in the reverse direction for 24 hours simulation period, more than 73% samples are highly asymmetric. Here, saved offset plays a crucial role to keep the synchronization accuracy high between the two clocks. According to the Equation 10 mentioned in the delay asymmetry correction mechanism, the saved offset is calculated as the average of all previously calculated saved offset values to spread out the adjustments over time. When the slave clock realizes that the latencies are highly asymmetric due to the existence of data traffic, it replaces the recently calculated offset value with the saved offset value. Finally, the slave clock is updated with the saved offset value. Therefore, the slave clock synchronization accuracy remains high between the slave and the master clock. From the statistics provided in Table 2, the average of the average slave clock accuracy is about 99.71 nanoseconds, which also indicates the improvement of the slave clock accuracy w.r.t the master clock.

Moreover, about 27% samples passed the ‘R’ test and about 26% samples are used directly to update the slave clock. It is evident from the statistics that the synchronization accuracy is improved significantly despite of having more than 73% samples collected in highly delay-asymmetric conditions, and having an additional 1% samples fail to pass the second test. In addition, the maximum of the maximum interval between successive successful update is about 82 seconds. It indicates that the slave clock is updated frequently with the calculated offsets, which passed both the filters. As a result, the slave synchronization accuracy remains high until the end of the simulation time. Since we are considering the maximum of the maximum slave clock difference in each synchronization window in Figure 11 and the statistics are collected as the average of absolute values, therefore to understand more clearly, the slave clock difference w.r.t the master clock is produced again in Figure 12 for a single simulation run. Statistical data are provided in Table 3.

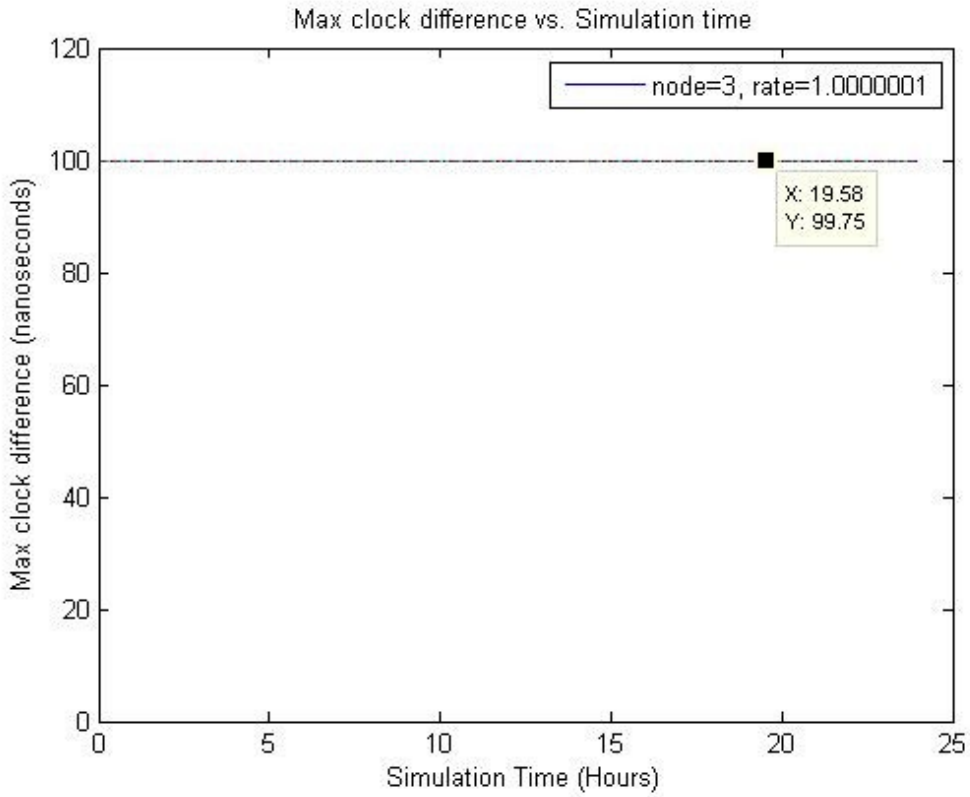

Figure 12: Slave Clock Synchronization with Static Packet Load -Single Run

Table 3: Statistical Data for Static Packet Load - Single Run

Total no. of samples	86400
Total no. of samples passed the ratio (R) test	23480 (27.18%)
Total no. of samples passed the update sample filter	22591 (26.15%)
Maximum. interval between successive successful updates (seconds)	30 s
Last successful update received at (Simulation time)	86401 s (24.00 Hours)
Average slave accuracy	99.69 ns
Max. clock difference with DAC model	99.75 ns

5.3.3 Slave Clock Synchronization with Sudden Large and Persistent Changes in Traffic Load

5.3.3.1 Description

In this test case, network load is introduced, which varies with time. Using the load profile shown in Figure 13, the traffic is introduced at a simulation time of 1second and stopped at a simulation time of 24 hours. Here in the **forward** direction (master-to-slave), the network load is changing between 80% and 20% every hour, while in the **reverse** direction (slave-to-master), the network load is changing between 50% and 10%. The test case examines the effects of large and persistent changes in network load on the slave clock synchronization with DAC model enabled on the slave clock.

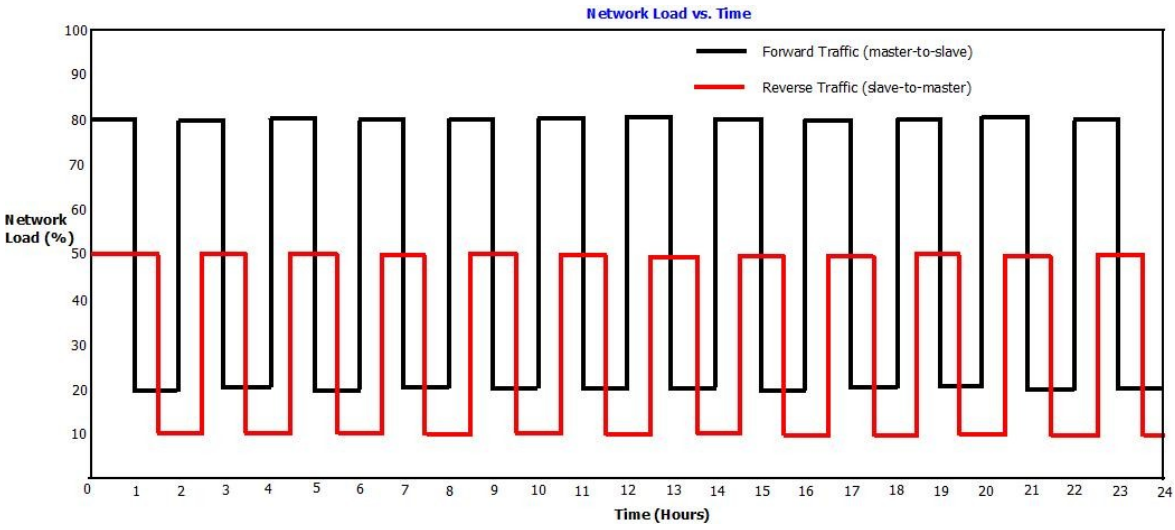

Figure 13: Load Profile Demonstrating Sudden Large and Persistent Changes in Traffic Load [3]

5.3.3.2 Results

Figure 14 shows the slave clock synchronization accuracy with respect to the master clock with the DAC model applied on the slave clock following the network load profile described in Section 5.3.3.1. Statistical data are provided in Table 4.

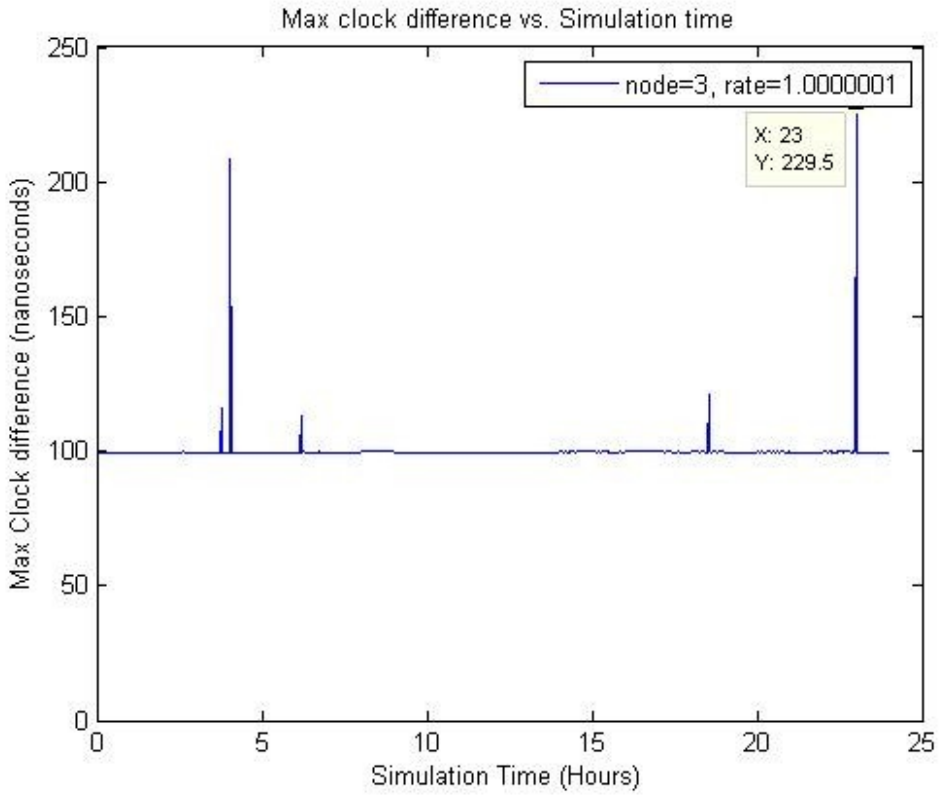

Figure 14: Slave Clock Synchronization with Sudden Large and Persistent Changes in Traffic Load

Table 4: Statistical Data for Sudden Large and Persistent Changes in Traffic Load

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	33472 (38.74%)
Avg. no. of samples passed the update sample filter	32882 (38.06%)
Max. of the max. interval between successive successful updates (seconds)	59
Avg. of the avg. clock accuracy	99.98 ns
Max. Of the max. clock difference (ns) with DAC model	229.5 ns

5.3.3.3 Discussion

From Figure 14, the maximum of the maximum clock difference between the slave and the master clock is about 230 nanoseconds. The slave clock accuracy is increased to 230

nanoseconds due to updating the slave clock with a relatively small saved offset value for a long period of time. As a result, the difference is accumulated over time. A similar effect is observed at times 4 hours, 7 hours, and 18 hours of the simulation time. Since the saved offset is calculated as the average offset correction over time, the slave clock may be under-corrected or over-corrected due to updating the clock with a relatively small or large saved offset value compared to the anticipated offset. In this case, the slave accuracy may increase or decrease depending on the applied value. However, from the statistics provided in Table 4, the average of the average slave accuracy is about 100 nanoseconds, which implies that the slave clock is able to maintain high accuracy with persistent change in the network load in both directions. About 39% samples passed the 'R' test and about 38% samples are used directly to update the slave clock. It reflects that about 1% samples did not pass the 2^{nd} stage filter and about 61% samples experienced highly asymmetric latencies. In both of these cases, the saved offset values are used to update the slave clock. In addition, the maximum of the maximum interval between successive successful update is 59 seconds, which indicates that the slave clock is updated frequently with the calculated offset, which passed both the filters.

5.3.4 Summary

The test cases in this chapter have been run using different traffic load profiles and varying network conditions applied on the slave clock. In a nutshell, the results are summarized as follows:

- The slave clock synchronization accuracy deteriorates severely without considering the DAC model. The slave accuracy is one half of the asymmetric latencies experienced due to the presence of traffic in the network.

- In the case of data centric traffic, the slave clock synchronization accuracy w.r.t the master clock is improved significantly for all the test cases with the DAC model applied on the slave clock. Both the 'R' test and the update sample filter ensure that only "good" offset values under high traffic load are used to update the slave clock. When a large percentage of samples experiences highly asymmetric latencies, saved offset plays a vital role to keep the synchronization accuracy high.
- In the case of network congestions, and temporarily network outage are presented in Appendix B, the slave clock achieved high synchronization accuracy w.r.t the master clock.
- In the case of routing path reconfiguration due to the failure in the network also presented in Appendix B, the slave clock shows high synchronization accuracy at the nanoseconds level for the data centric traffic model.

6 Chapter: Sensitivity Analysis

In this chapter, we will scrutinize the direct effects of various parameters on the slave clock synchronization. Section 6.1 describes the traffic profile, which is used in all test cases in this chapter for examining the effects of the slave clock performance. Section 6.2 focuses on the performance of the slave clock accuracy, when it is coordinating with multiple master clocks. Section 6.3 also examines the effects on the slave clock accuracy with different parameters, such as synchronization frequencies, drifts and initial offsets. Finally this chapter is concluded by examining the temperature and the aging effects on the slave clock and AOCM is used to compensate the temperature and ageing effects of the local oscillator. The slave and the master clocks are connected as shown earlier in Figure 8 using the parameters mentioned in the simulation set up section (Section 5.1), unless a different topology and parameters are specified in a test case. Total simulation time is 24 hours for all the test cases. The results are collected with a single simulation run for all the test cases.

6.1 Traffic Profile

The data centric traffic model described in Section 5.1.1 is considered in this chapter. All the test cases in this chapter assume 40% of network load in the **forward** direction (master-to-slave) and 30% network load in the **reverse** direction (slave-to-master). At 7 hours of simulation time, network load is increased to **100%** in both directions for 100 s, and then restored to the previous levels.

6.2 Effect on a Slave Clock with Multiple Master Clocks

6.2.1 Description

The network topology shown in Figure 15 is used to evaluate the performance of the slave clock accuracy w.r.t the multiple master clocks.

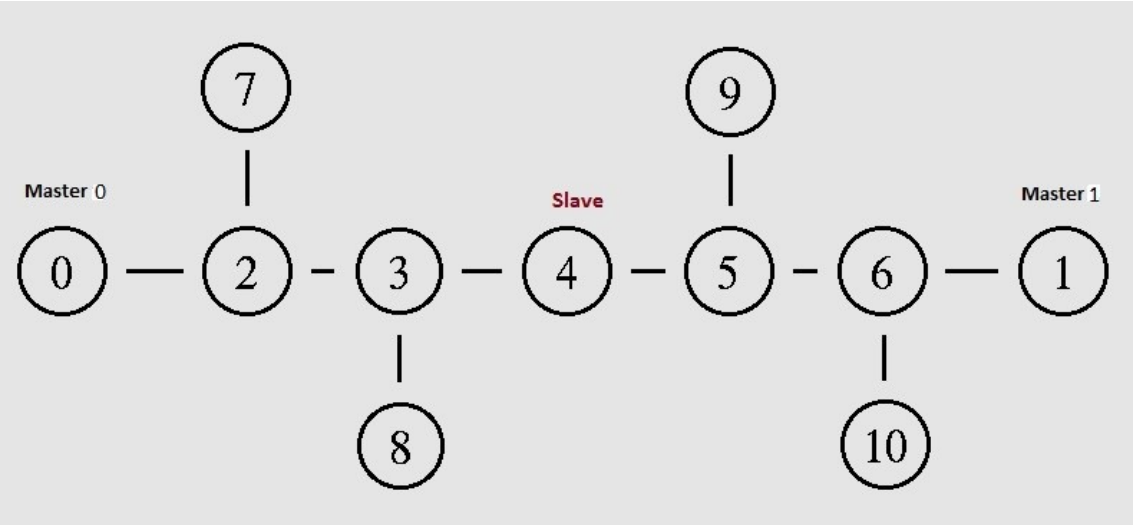

Figure 15: Network Topology using Multiple Master Clocks

The network topology consists of two master nodes n0 and n1 connected to a slave node n4 with two intermediate nodes in each domain. Two traffic sources are introduced in each domain. In the master-0 and the slave domain, one of the traffic sources is node n7, which sends traffic to node n8 and vice versa. Similarly, node n9 and n10 are considered as the traffic sources for the master-1 and the slave domain. Node n9 sends traffic to node 10 and vice versa. The parameters mentioned in the simulation set up section (Section 5.1) are used in this test case, except for the IEEE 1588 synchronization frequency. 10 seconds IEEE 1588 synchronization frequency is considered in this test case. Both the master clocks initiate the IEEE 1588 message exchange according to the Equation 11,

described in Section 4.3. The data are collected as the average values of the slave clock accuracy w.r.t each master clock.

6.2.2 Results

Figure 16 shows the slave clock synchronization accuracy w.r.t both the master clocks. 40% of network load is introduced in the **forward** direction and 30% load in the **reverse** direction with the increment of network load to **100%** in both directions for 100 s, and then restored. In addition, statistics for several parameters are collected for each domain and provided in Table 5.

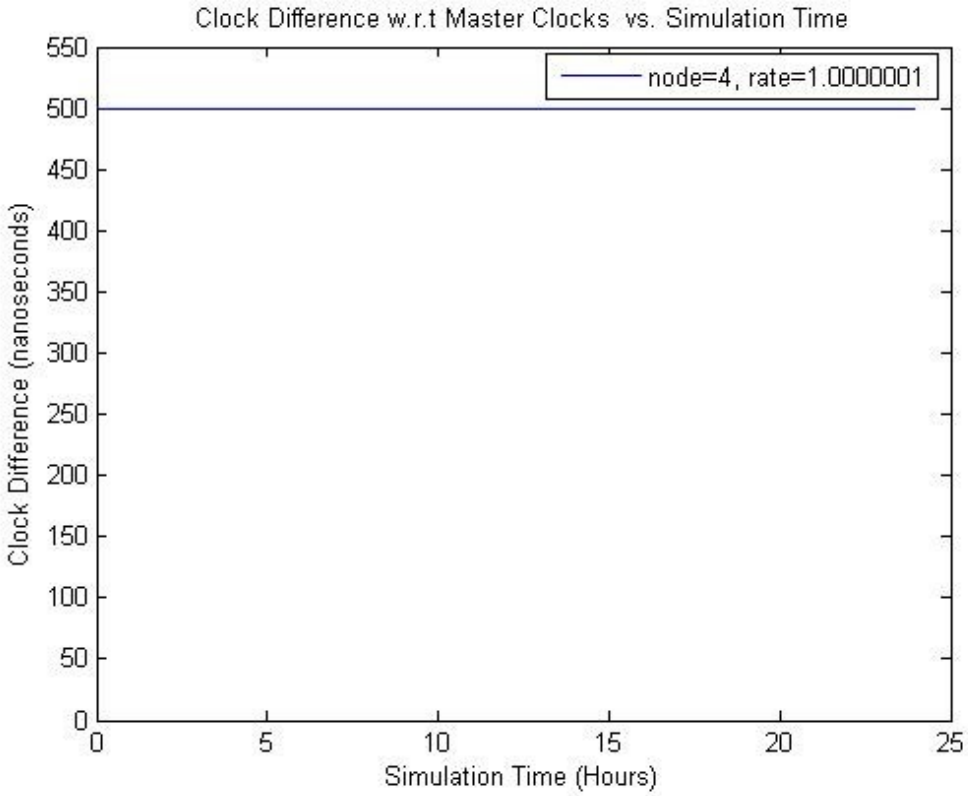

Figure 16: Slave Clock Synchronization Accuracy w.r.t both the Master Clocks

Table 5: Statistical Data with Multiple Master Clocks

	Master 0	Master 1
Total no. of samples	8640	8640
Total no. of samples passed the ratio (R) test	8640 (100%)	6020 (69.68%)
Total no. of samples passed the update sample filter	8639 (99.99%)	6015 (69.62%)
Maximum interval between successive successful updates (seconds)	10 s	80 s
Last successful update received at simulation tine	86401s (24 Hours)	86396s (23.99 Hours)
The average slave accuracy	500 ns	496 ns
Maximum clock difference (ns) with DAC model	500 ns	500 ns

6.2.3 Discussion

From Figure 16, the maximum slave clock synchronization accuracy w.r.t both the master clocks is 500 ns. The resultant difference implies that the slave clock accuracy is improved 2 times having two master clocks. Since the slave clock is drifting 100 ppb faster than both the master clocks and the synchronization frequency is 10 s, the maximum offset would be 1 μs. From the statistics provided in Table 5, the average slave clock accuracy is 500 ns, which indicates that the slave clock accuracy remains high until end of the simulation time. It is noticeable that almost 100% samples passed both the 'R' test and the update sample filter in the master-0 domain, while about 70% samples are used directly to update the slave clock correctly w.r.t the master-1 domain. In addition,

the maximum intervals between successive successful updates are 10 s for master-0 domain and 80 s for master-1 domain, which also indicates that the slave clock is updated frequently with the calculated offsets. Therefore, it is evident that the slave clock received at least one good sample from one of these master clocks within the defined update frequency interval in order to update the slave clock correctly. Thus, the slave clock does not rely on the saved offset value for a long period of time.

6.3 Effects at Different Parameters

In this section, the effects on the slave clock synchronization w.r.t the master clock are observed using different parameters, such as synchronization frequencies, drifts and initial offsets on the slave clock. The data are collected as the maximum value for each synchronization frequency. It should be noted that all results are collected after stabilization period of 900 s, as described in [3]. In addition, all results are plotted with logarithmic scales for both axes, unless it is explicitly mentioned.

6.3.1 IEEE 1588 Synchronization Frequency

6.3.1.1 Description

In this test case, a slave clock is connected with a master clock as shown in Figure 8 and the IEEE 1588 synchronization frequency is varied using the values of 1s, 5s, 10s, 50s, 100s, and 1000s. Here the slave clock is drifting at a fixed rate of 100ppb.

6.3.1.2 Result

Figure 17 shows the maximum slave synchronization accuracy w.r.t the master clock at different IEEE 1588 synchronization frequencies.

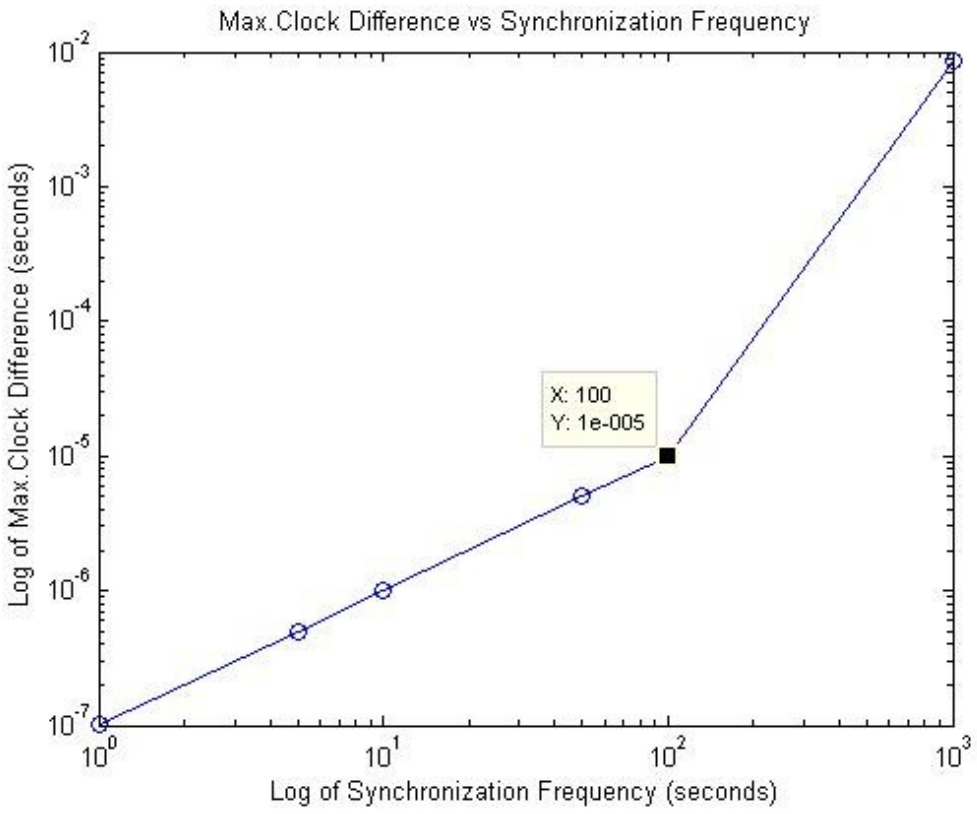

Figure 17: Effects on Slave Clock Synchronization-Varying IEEE 1588 Synchronization Frequency

6.3.1.3 Discussion

From Figure 17, the effects of varying synchronization frequency values can be seen on the slave clock accuracy. The result shows that a higher synchronization frequency (lower time value) provides better clock synchronization. The higher synchronization frequency increases the possibility of receiving updates from the master clock more frequently and hence improves the slave accuracy. The slave clock achieves high synchronization accuracy with the synchronization frequency values of 1 s, 5 s, 10 s, 50 s, and 100 s. If the synchronization updates are received less frequently (i.e. 1000 s or more), the slave clock synchronization accuracy severely depends on the existing traffic load in the network. According to the DAC model, the longer the slave clock waits for

receiving synchronization updates from the master clock, the fewer the number of samples that pass the 'R' test. Thus, the slave clock will be updated with the saved offset value and the difference will be accumulated over time as well. Moreover, the calculation of 'R' will be distorted if the clock difference is roughly on the same order of the magnitude of the one way latency, which is the case for the 1000 s synchronization frequency.

6.3.2 Slave Clock Rates

6.3.2.1 Description

In this test case, we are considering that the slave clock is drifting faster than the master clock. The slave clock drifting rate is varied using the values of 1 ppb, 10 ppb, 100 ppb, 1 ppm, 10 ppm, and 100 ppm. Here the slave clock synchronization frequency is 1s.

6.3.2.2 Result

Figure 18 shows the maximum slave synchronization accuracy w.r.t the master clock at different drifting rates.

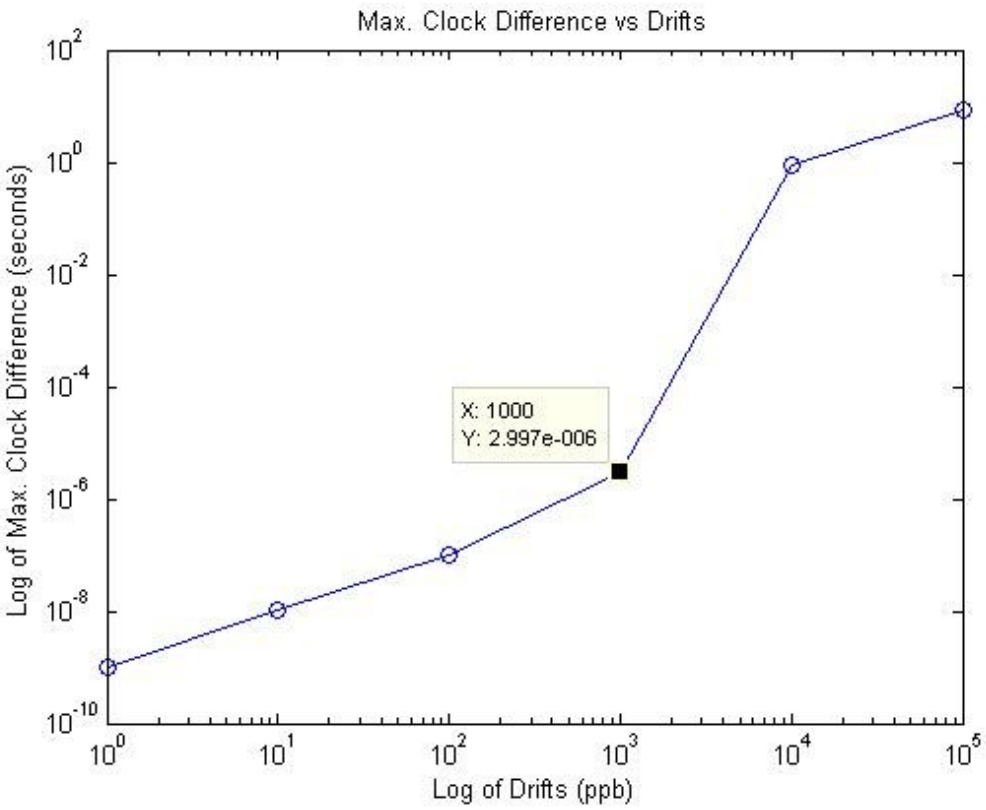

Figure 18: Effects on Slave Clock Synchronization-Varying Slave Clock Rates

6.3.2.3 Discussion

From Figure 18, the effects of varying the slave clock drifting rate can be seen on the slave clock accuracy. The result shows that a lower drifting rate values provide better clock synchronization. According to the rates considered in this test case, the maximum clock difference would be 1 ns, 10 ns, 100 ns, 1μs, 10 μs, and 100 μs per synchronization period for the drifting rate of 1 ppb, 10 ppb, 100 ppb, 1 ppm, 10 ppm, and 100 ppm respectively. The slave clock achieves high synchronization accuracy with the drifting rate values of 1 ppb, 10 ppb, 100 ppb, and 1 ppm. It is noticeable that the slave clock maximum difference is increased to 3 μs for 100 ppb drifting rate. The rationale behind this is that the slave clock is updated with a saved offset value for a long period of time

and the difference is accumulated over time. When the slave clock drifting rate is higher than 1ppm (i.e. 10 ppm, 100 ppm), none of the offsets passed the 'R' test and the calculation of 'R' is distorted at some point because the offset is in the same order of the magnitude as the one way latency. Hence, the slave clock is updated with default saved offset value until the end of the simulation time. Therefore, the clock difference is accumulated until 24 hours of the simulation time. It has been observed that almost similar results are obtained while the slave clock is drifting slower than the master clock with the values mentioned in this test case.

6.3.3 Slave Clock Initial Offsets

6.3.3.1 Description

In this test case, the slave clock initial offset is varied using the values of 1 s, 5 s, 10 s, 20 s, 30 s, 40 s, 50 s, 60 s, 70 s, 80 s, 90 s, and 100 s. Here the slave clock synchronization frequency is fixed at 1s and the slave clock is drifting 100ppb faster than the master clock. Therefore, the maximum slave clock difference would be 100 ns after the stabilization period of 900 s.

6.3.3.2 Result

Figure 19 shows the maximum values for slave clock synchronization at different initial offsets.

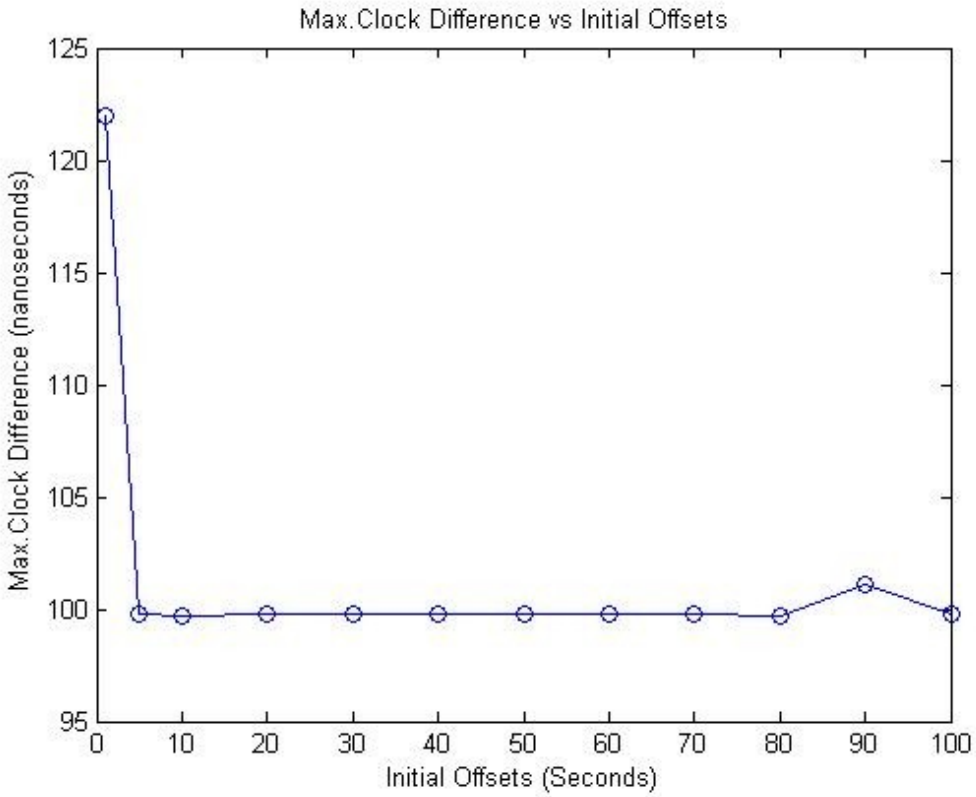

Figure 19: Effects on Slave Clock Synchronization-Varying Slave Initial Offsets

6.3.3.3 Discussion

From Figure 19, the effects of varying initial offset values can be seen on the slave clock accuracy. The result shows that different initial offsets do not affect on the slave synchronization accuracy. The maximum clock difference between the slave and the master clock is bounded within 100 ns for all the values except for 1 second initial offset. The slave clock difference is increased to about 123 ns when it is off by 1 s. This is because we update the slave clock with a saved offset value for a long period of time. Therefore, we can state that the slave clock is able to achieve high synchronization accuracy with different initial offset values.

6.4 Temperature and Aging Effect on a Slave Clock with AOCM

In this section, first, we are examining the temperature and the aging effects on the slave clock using different network conditions. Later on, AOCM is introduced to mitigate both the temperature and the aging effects on the slave clock. We have a slave clock having both temperature and aging effects connected to a master clock. The oscillator of the master clock has no drifts due to the temperature and aging effects and therefore runs perfectly. A linear temperature effect of 5ppb/75°C with quadratic term equal to -0.00031966ppb/°C^2 with an 8 hour cyclical temperature profile shown in Figure 20 and a linear aging effect of 3ppb/day are used on the slave clock [4]. Other parameters are used from the simulation set up section (Section 5.1).

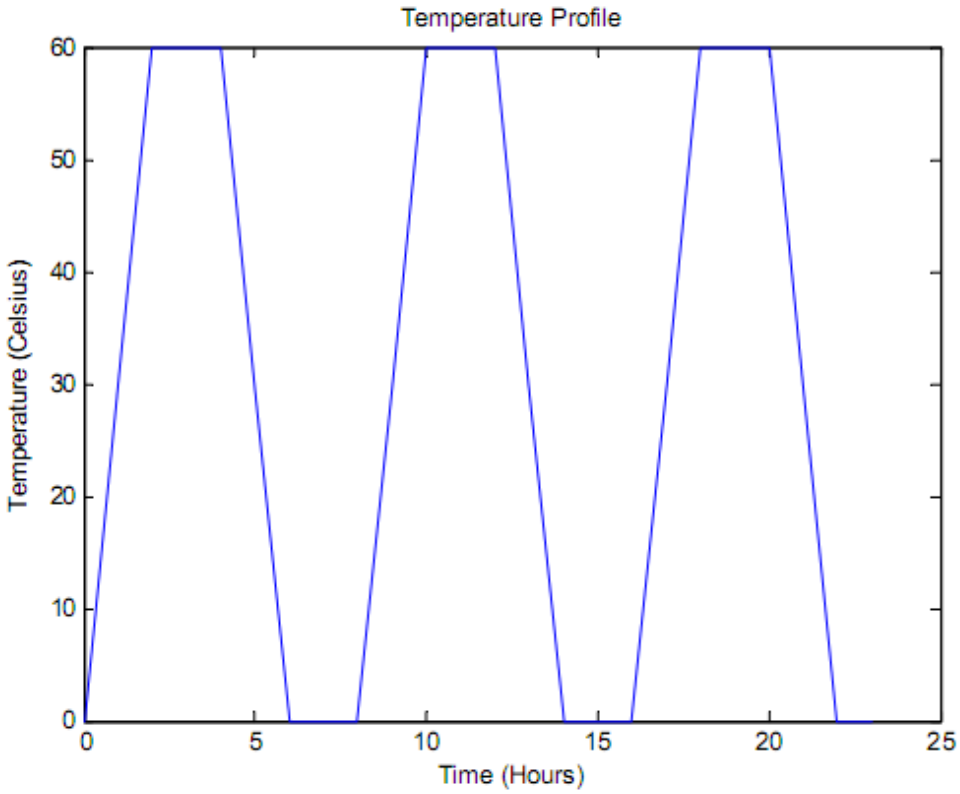

Figure 20: Temperature Profile [4]

The temperature profile represents the temperature values (i.e. coefficients) observed at the clock oscillator [4, 26]. The range of the temperature variation is 60°C over an 8 hour cycle. We are using 3 cycles of temperature variations to study the effect of temperature over shorter simulation time periods. Thus, we artificially condensed the daily temperature cycle. The temperature range is large enough to represent a real working environment. For instance, if a BTS located in a desert, the average temperature might be very hot in day time. The 8 hour cycle guarantees that the simulation results are obtained fast enough.

6.4.1 Temperature and Aging Effect on a Slave Clock

6.4.1.1 Description

In this test case, the accuracy of a slave clock with the temperature and the aging is presented. The purpose is to examine the temperature and aging drifts on the slave clock accuracy under various network conditions.

6.4.1.2 Results

Figure 21 shows the temperature and the aging effects on the slave clock accuracy when the slave and the master clocks are running at the same rate. Figure 22 also shows the temperature and the aging effects when the slave clock is drifting 100 ppb faster than the master clock. In both of these cases, there is no traffic in the network and the DAC model is enabled on the slave clock. Figure 23 shows the temperature and the aging effects on the slave clock with the traffic profile described in Section 6.1, and the DAC model is enabled on the slave clock. Here the slave clock is drifting 100 ppb faster than the master clock.

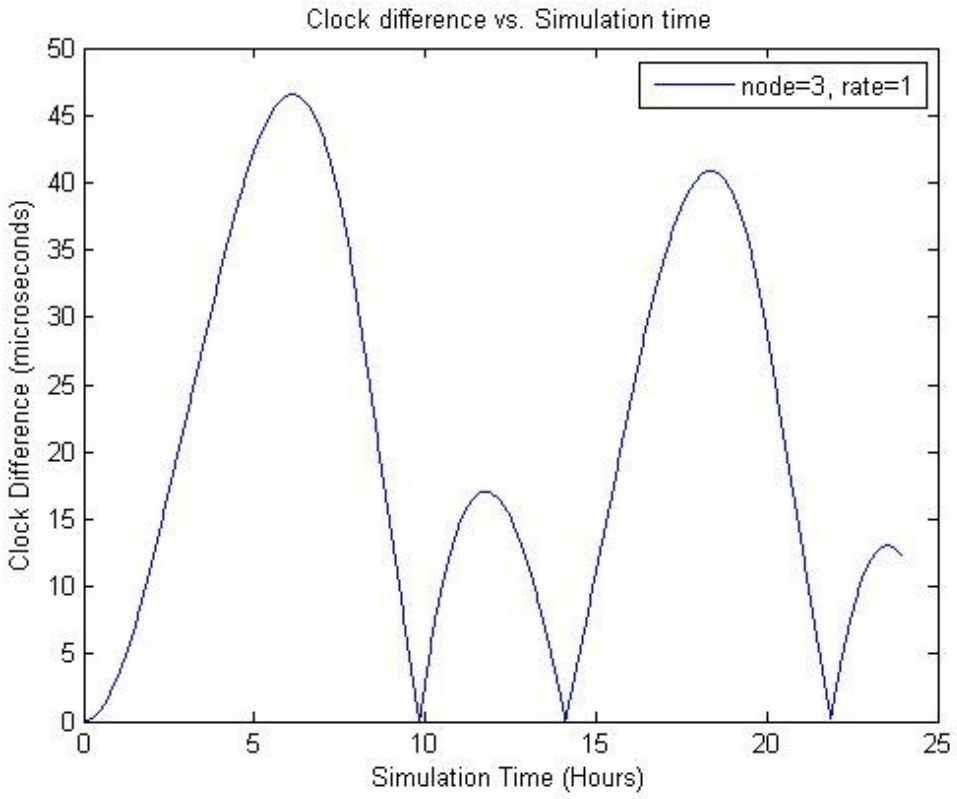

Figure 21: Slave Clock Accuracy with the Same Drifting Rate w.r.t the Master Clock – Temperature and Aging Effects

Table 6: Statistical Data with the Temperature and Aging Effects with the Same Drifting Rate

Total no. of samples	86400
No. of samples passed the ratio (R) test	26261
No. of samples passed the update sample filter	2
Max. difference (ns) with DAC model	47 µs

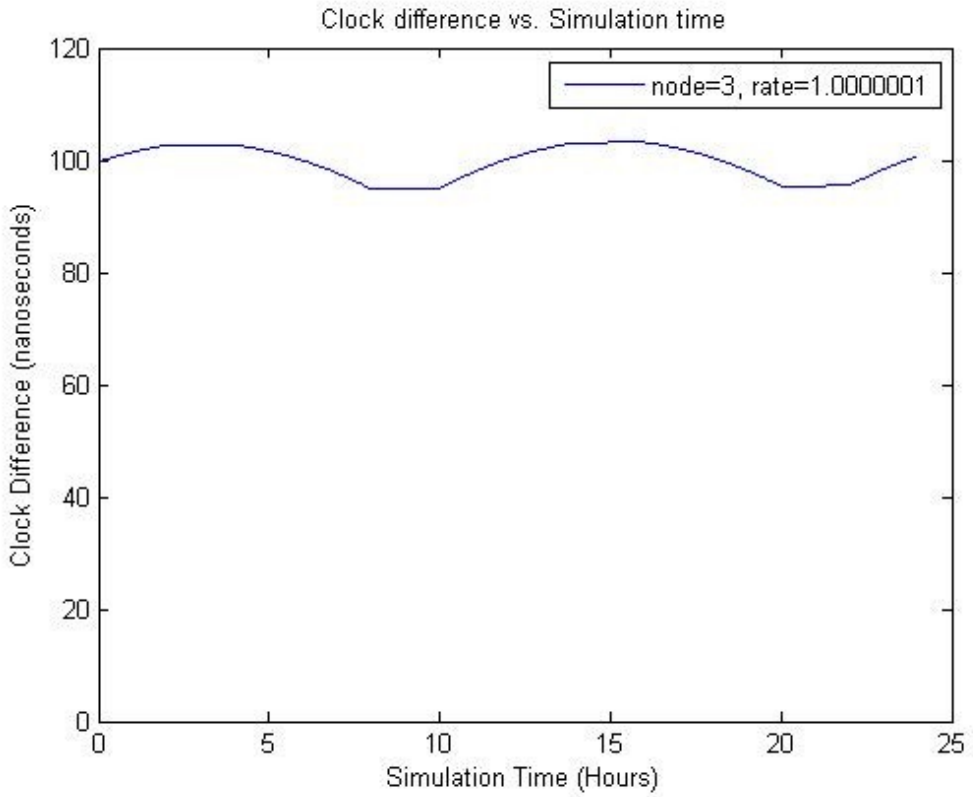

Figure 22: Both Temperature and Aging Effects on the Slave Clock with 100 ppb faster Drift

Table 7: Statistical Data with the Temperature and Aging Effects with 100 ppb Faster Drift

Total no. of samples per run	86400
No. of samples passed the ratio (R) test	86400 (100%)
No. of samples passed the update sample filter	86400 (100%)
Max. interval between successive successful updates (seconds)	1 s
Max. Of the max. clock difference (ns) with DAC model	105 ns

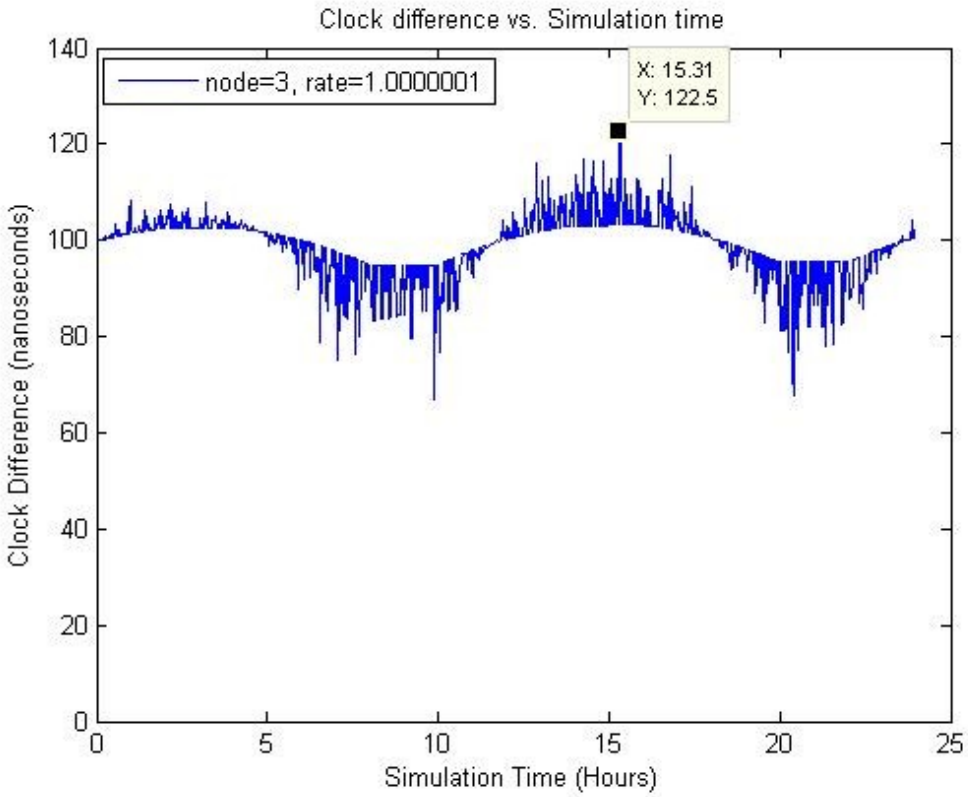

Figure 23: Both Temperature and Aging Effects on the Slave Clock with Traffic Profile and 100 ppb Faster Drift

Table 8: Statistical Data for the Temperature and Aging Effects with Traffic Profile and 100 ppb Faster Drift

Total no. of samples per run	86400
No. of samples passed the ratio (R) test	41225 (47.71%)
No. of samples passed the update sample filter	40654 (47.05%)
Max. interval between successive successful updates (seconds)	65 s
Max. Of the max. clock difference (ns) with DAC model	122 ns

6.4.1.3 Discussion

The figures show that the temperature and the aging effects on the slave clock accuracy using different network conditions. In all of the results, the temperature effect follows the pattern of the temperature profile shown in Figure 20: the clock drifts faster when it is hot and drifts slower when it is cold. The figures also demonstrate the cumulative effect of the temperature and the aging drifts on the slave clock accuracy. However, from Figure 21, the slave accuracy increased to 47 μs, when the slave and the master clock are drifting at the same rate. The resultant difference is increased due to updating the slave clock with a relatively small saved offset value for a long period of time. Since there is no traffic in the network and the slave clock is drifting at the same rate with the master clock, the maximum slave drift would be 5 ppb with a linear temperature effect of 5ppb/75°C with quadratic term equal to -0.00031966ppb/°C^2 with 8 hour cycle temperature profile shown in Figure 20 and a linear aging effect of 3ppb/day. If the temperature fluctuations are on the same order of the magnitude of the offset, the proposed scheme, particularly the forecast procedure does not work well. It is worth mentioning that the proposed scheme works well when offset is the dominant source of the slave clock drift due to the inherent frequency offset of a crystal oscillator. From the statistics provided in Table 6, only 2 samples passed the 2^{nd} stage filter, which indicates the poor performance of the forecasting due to the larger temperature variation than the slave clock drift. A similar result is observed when the traffic is introduced in the network and the slave and the master clocks are drifting at the same rate. From Figure 22, the maximum slave accuracy is 105 ns when the slave clock is drifting 100 ppb faster than the master clock. The resultant difference is fluctuating from 95 ns to 105 ns within an 8 hour cycle due to the

temperature variations used in the temperature profile. The resultant difference implies that the slave clock achieved high accuracy when the slave clock drift is the dominant factor, rather than the temperature variations. From the statistics provided in Table 7, all of the samples passed both filters and are used directly to update the slave clock. In addition, the maximum interval between successive successful updates is 1 s. It implies that the slave clock is updated with the calculated offset value during every synchronization event in order to achieve the high synchronization accuracy. An almost similar result is obtained when the traffic profile is introduced, shown in Figure 23. The maximum clock difference is about 121 ns. The resultant difference is the cumulative effect of temperature, aging drifts and the traffic load in the network. The saved offset value is used to update the slave clock for a long period of time when the latencies are highly asymmetric. From the statistics provided in Table 8, about 48% samples passed the 'R' test and about 47% samples passed the 2nd test filter. In addition, the maximum interval between two successive successful updates is 65 seconds, which indicates that the slave clock is updated frequently with the calculated offset values.

6.4.2 Effect of AOCM on Slave Clock

6.4.2.1 Description

In this test case, adaptive corrections are applied on the slave clock in locked mode only. In locked mode, AOCM will train its model only with those values which passed both the 'R' test and the update sample filter. In this test case, we are not considering a holdover mode such as temporary network outage between the master and the slave clock.

6.4.2.2 Results

Figure 24 shows the slave clock synchronization w.r.t the master clock with AOCM applied on the slave clock. Statistics are provided in Table 9.

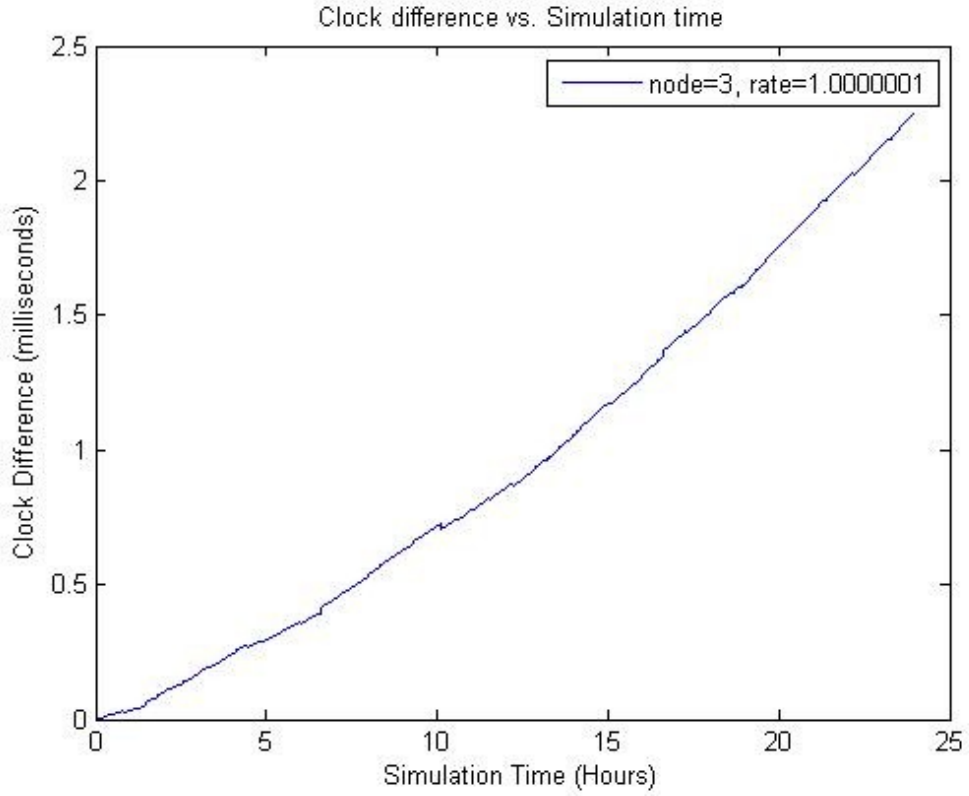

Figure 24: Slave Clock Synchronization with AOCM Corrections in Locked Mode only

Table 9: Statistical data with AOCM

Total no. of samples	86400
Total no. of samples passed the ratio (R) test	766 (0.89%)
Total no. of samples passed the update sample filter	90 (0.10%)
Maximum. interval between successive successful updates (seconds)	4030 s (1.12 Hours)
Last successful update received at (Simulation time)	85422 s (23.73 Hours)
Max. clock difference with DAC model	2.4 ms

6.4.2.3 Discussion

From Figure 24, the slave to the master clock difference is about 2.4 milliseconds with AOCM correction applied on the slave clock. The resultant difference increases with time due to updating the slave clock with a small saved offset value for a long period of time. The AOCM does both training and correction when a sample passes both the 'R' test and the update sample filter. Hence, the AOCM correction is applied to the slave clock rate in order to compensate the drift of the slave clock, caused by temperature and aging effects. The proposed model cannot forecast well due to making correction on the slave clock rate while it trains the model. From the statistics provided in Table 9, less than 1% sample passed the 'R' test. It implies that more than 99% samples are highly asymmetric and the slave clock is updated with the saved offset value. Therefore, the difference is accumulated over time, which is reflected in Figure 24.

6.5 Summary

The test cases in this chapter examine the effects of various parameters on the slave clock synchronization accuracy w.r.t the master clock. The experimental findings are summarized as follows:

- The slave clock accuracy improved 2 times when it is coordinating with two master clocks in a network. Having multiple master clocks, the slave clock does not need to rely on the saved offset value for a long period time in order to update itself.
- The higher (lower time) the IEEE 1588 synchronization frequency, the higher is the slave synchronization accuracy w.r.t the master clock. If the IEEE 1588

synchronization frequency is more than 100s, the slave synchronization accuracy is severely affected.

- The more the slave clock rates deviates (either positively or negatively) from the master clock, the worse is the slave clock accuracy w.r.t the master clock on average. The slave clock synchronization accuracy deteriorates when the slave rate deviates (positively or negatively) by more than 1ppm.
- Varying initial offset values do not affect the slave synchronization accuracy w.r.t the master clock.
- The slave accuracy deteriorates when the temperature variation is large compared to the drift. The slave accuracy improved significantly when the slave clock drift is the dominant factor, rather than the temperature variations.
- The proposed model cannot forecast well due to making corrections on the slave clock rate while it trains the AOCM model.

7 Chapter: Conclusions and Future Work

7.1 Conclusions

The conclusions of the thesis work are presented as follows:

- A Delay Asymmetry Correction (DAC) Model is proposed for the clock synchronization problem. The proposed work enhances the IEEE 1588 protocol by computing the time difference between the master and the slave clock in the presence of traffic in a network. The first step of the thesis is to mitigate the effects of unpredictable packet delay variations (PDV), which cause asymmetric link delays on timing packets between the slave and the master clocks. The next step extends to coordinate multiple master clocks through a single slave clock, which may be connected through multiple networks. Finally the DAC model integrates the AOCM model for compensating both the temperature and ageing effects of the oscillator.
- The proposed solution is evaluated by implementing it in the NS-2 simulator. Test cases are also implemented based on an ITU-T document covering various network loads and network conditions. The NS-2 results indicate that the proposed solution improves the slave accuracy by measuring the correct offset value in a slave clock for asymmetric communication link delays.
- The solution results show that the slave clock is able to achieve high accuracy in the presence of various bi-directional traffic loads and network conditions. In the case of data centric traffic model, the slave clock synchronization accuracy w.r.t the master clock is improved significantly for all the test cases with the DAC model applied on the slave clock.

- In the case of voice centric traffic model, sometimes the slave accuracy is affected severely under high traffic load, particularly for the case of 80% load in the forward direction and 20% load in the reverse direction. In general, the key problem is the lower amount of bursty traffic (i.e. 15%) in the network. But the slave accuracy improved significantly when the amount of bursty traffic is increased to 50% of the network load. A similar result is observed with multiple master clocks in the network.
- The solution results also show that the slave clock achieved high synchronization accuracy in the presence of network congestions, and temporarily network outage for both traffic models. Furthermore, when there is a routing path change due to a failure in the network, the solution also improves the accuracy of the slave clock with respect to the master clock. Both the 'R' test and the update sample filter ensure that only "good" offset values under high traffic load are used to update the slave clock.
- When a high percentage of samples are collected in highly delay-asymmetric conditions, the saved offset plays a vital role to keep the synchronization accuracy high.
- The slave accuracy is increased by a factor of two when having two master clocks in the network. The slave clock received at least one good update from one of these master clocks in each synchronization period.
- The higher (lower time) the IEEE 1588 synchronization frequency, the higher is the slave synchronization accuracy w.r.t the master clock.

- The more the slave clock rates deviate (either positively or negatively) from the master clock, the worse is the slave clock accuracy w.r.t the master clock on average.
- Varying initial offset values does not affect the slave synchronization accuracy w.r.t the master clock.
- The slave accuracy deteriorates when the temperature variation is large compare to the drift. The slave accuracy improved significantly when the drift is the dominant factor.
- The proposed solution cannot forecast well when it is integrated with the AOCM model due to making corrections on the slave clock rate while it trains the model.

7.2 Future Work

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

- The solution uses a basic IEEE 1588 message exchange structure by incorporating two filtering methods to ensure that only good samples are used to update the slave clock in order to achieve high synchronization accuracy. One of the major limitations of this work exists in the calculation of 'R' test. The calculation of 'R' is an approximation of the undefined asymmetry ratio and subject to significant distortion, when the offset is roughly on the same order of magnitude as the one-way latencies. This issue can be solved by enhancing the IEEE 1588 synchronization protocol. One suggestion for this is to append additional messages for calculating the exact ratio so that the DAC model initializes with good sample.

- The solution does not perform well when AOCM corrections are applied on the slave clock. The problem with the DAC model is that it cannot forecast well when correcting the slave clock rate while it trains the model. The forecasting procedure is very sensitive and depends on the parameters that are used. This issue can be solved by changing the forecasting procedure in the DAC model so that it can inter-operate with temperature and aging effects and with the AOCM model as well. This can be done by continuously monitoring the calculated offsets and computing the correctional rate from the slope of the most recent set of calculated offsets. Another solution is to introduce Holt's linear exponential smoothing (LES) technique or autoregressive integrated moving average (ARIMA) methods, which are very effective for forecasting based on the recent trend of time series data.

Appendices

Appendix A : NS-2 TCL Examples

The NS-2 TCL scripts presented in this appendix use a master node n0 connected with a slave node n3 with two intermediate nodes n1 and n2, resulting in the 3 hops topology as shown in Figure 8. Two traffic sources are also introduced. One of the traffic sources is node n4, which sends traffic to node n5 and vice versa. All the clocks used have no initial offsets in the rate i.e. initially all tick at the same time as the reference clock (simulation time), but they are not drifting at the same rate. Clock m1 is a master clock agent connected with node n0 and clock s1 is a slave clock agent connected with node n3. The slave clock s1 is drifting 100 ppb faster than the master clock m1. No temperature and aging effects are enabled in the clock agents. The detail descriptions of the clock parameters can be found in [4]. The traffic profile described in Section 5.3.1.1 is implemented in the scripts. Node n4 generates 80% static packet load, which sends traffic to node n5 (i.e. Master-to-Slave). Similarly, node n5 generates 20% static packet load, which sends traffic to node n4 (Slave-to-Master).

A.1 NS-2 TCL Script Code for Data Centric Traffic Model

In this test case, a TCL script for data centric traffic model, which is described in Section 5.1.1.1, is presented with the static traffic load described in Section 5.3.1.1.

NS-2 TCL Script

```
set ns [new Simulator]

#Open a trace file
set tracefd [open clktest1d.tr w]
$ns trace-all $tracefd
set nf [open clktest1d.nam w]
```

```
$ns namtrace-all $nf
set graphData [open graphData.txt w]

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

#Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

#Link configurations
$ns duplex-link      $n0 $n1      1Mb   100ns  DropTail
$ns duplex-link      $n1 $n2      1Mb   100ns  DropTail
$ns duplex-link      $n2 $n3      1Mb   100ns  DropTail
$ns duplex-link      $n1 $n4      1Mb   100ns  DropTail
$ns duplex-link      $n2 $n5      1Mb   100ns  DropTail

#Define a 'timeout' function for the class 'Agent/Clock'
Agent/Clock instproc timeout {ts rate eff_rate} {
# NOTE: rate is natural rate of clock and eff_rate is effective rate (after applying RLS
correction) of the clock
        global ns graphData
        $self instvar node_
        variable nodeID [$node_ id]
        puts $graphData "[$node_ id]  $eff_rate $ts [$ns now] "
}


# Generate bursts lasting between 0.1 s to 3 s
set rng1 [new RNG]
$rng1 seed [lindex $argv 0]
set burstDuration [new RandomVariable/Uniform]
$burstDuration set min_ 0.1
$burstDuration set max_ 3
$burstDuration use-rng $rng1
```

 # *** 80% "Static" packet load in the Forward Direction (Master-to-Slave)*****

 # 60% of the load based on maximum sized packet (1518 bytes) generated at node n4, which sends traffic to node n5

```
#Create a UDP agent and attach it to node n4
set udp4a [new Agent/UDP]
$udp4a set packetSize_ 1518
$ns attach-agent $n4 $udp4a

# Create a CBR traffic source, which will generate bursty traffic and attach it to udp4a
set cbr4a [new Application/Traffic/CBR]
$cbr4a set packetSize_ 1518
$cbr4a set rate_ 0.96Mb
$cbr4a attach-agent $udp4a

#Create a Null agent (a traffic sink) and attach it to node n5
set null5 [new Agent/Null]
$ns attach-agent $n5 $null5

#Connect the traffic source with the traffic sink
$ns connect $udp4a $null5
```

 # 30% of the load based on small sized packet (64 bytes) generated at node n4, which sends traffic to node n5

```
#Create a UDP agent and attach it to node n4
set udp4b [new Agent/UDP]
$udp4b set packetSize_ 64
$ns attach-agent $n4 $udp4b

# Create a CBR traffic source and attach it to udp4a
set cbr4b [new Application/Traffic/CBR]
$cbr4b set packetSize_ 64
$cbr4b set rate_ 0.24Mb
$cbr4b attach-agent $udp4b

#Connect the traffic source with the traffic sink
$ns connect $udp4b $null5
```

 # 10% of the load based on medium sized packet (576 bytes) generated at node n4, which sends traffic to node n5

```
#Create a UDP agent and attach it to node n4
```

```
set udp4c [new Agent/UDP]
$udp4c set packetSize_ 576
$ns attach-agent $n4 $udp4c

# Create a CBR traffic source and attach it to udp4c
set cbr4c [new Application/Traffic/CBR]
$cbr4c set packetSize_ 576
$cbr4c set rate_ 0.08Mb
$cbr4c attach-agent $udp4c

#Connect the traffic source with the traffic sink
$ns connect $udp4c $null5

# ***20% "Static" packet load in the Reverse Direction (Slave-to- Master)****

# 60% of the load based on maximum sized packet (1518 bytes) generated at node n5, which sends traffic to node n4

#Create a UDP agent and attach it to node n5
set udp5a [new Agent/UDP]
$udp5a set packetSize_ 1518
$ns attach-agent $n5 $udp5a

# Create a CBR traffic source, which will generate bursty traffic and attach it to udp5a
set cbr5a [new Application/Traffic/CBR]
$cbr5a set packetSize_ 1518
$cbr5a set rate_ 0.48Mb
$cbr5a attach-agent $udp5a

#Create a Null agent (a traffic sink) and attach it to node n4
set null4 [new Agent/Null]
$ns attach-agent $n4 $null4

#Connect the traffic source with the traffic sink
$ns connect $udp5a $null4

# 30% of the load based on small sized packet (64 bytes) generated at node n5, which sends traffic to node n4

#Create a UDP agent and attach it to node n5
set udp5b [new Agent/UDP]
$udp5b set packetSize_ 64
$ns attach-agent $n5 $udp5b

# Create a CBR traffic source and attach it to udp5b
set cbr5b [new Application/Traffic/CBR]
```

```
$cbr5b set packetSize_ 64
$cbr5b set rate_ 0.06Mb
$cbr5b attach-agent $udp5b

#Connect the traffic source with the traffic sink
$ns connect $udp5b $null4
```

 # 10% of the load based on medium sized packet (576 bytes) generated at node n5, which sends traffic to node n4

```
#Create a UDP agent and attach it to node n5
set udp5c [new Agent/UDP]
$udp5c set packetSize_ 576
$ns attach-agent $n5 $udp5c

# Create a CBR traffic source and attach it to udp5c
set cbr5c [new Application/Traffic/CBR]
$cbr5c set packetSize_ 576
$cbr5c set rate_ 0.02Mb
$cbr5c attach-agent $udp5c

#Connect the traffic source with the traffic sink
$ns connect $udp5c $null4

#Create master clock agent
set m1 [new Agent/Clock]
$m1 set offset 1
$m1 set rate 1
$m1 set timeToDisplayInfo 1
$m1 set masterClock 1
$m1 set gpsSignal 0
$m1 set enableRLS 0
$m1 set enableLockedRLS 0
$m1 set RLSFreq 0
$m1 set tempProfileName -1
$m1 set ageing 0
$m1 set driftInterval 1
$m1 set enable1588Logs 1
$ns attach-agent $n0 $m1

# Create a slave clock agent
set s1 [new Agent/Clock]
$s1 set offset 1
#slave clock rate is 100 ppb (100 ns) faster than the master clock
$s1 set rate 1.000000100
$s1 set timeToDisplayInfo 1
```

```
$s1 set masterClock 0
$s1 set gpsSignal 0
$s1 set enableLockedRLS 0
$s1 set enableRLS 0
$s1 set RLSFreq 0
$s1 set tempProfileName -1
$s1 set driftInterval 1
$s1 set timeStampReqFreq 1
$s1 set masterAddr 0
$s1 set masterPort 0
$s1 set enable1588Logs 1
$s1 set enable1588Delays 0
$ns attach-agent $n3 $s1
```

#***Schedule events: Traffic in the Forward Direction (Master-to-Slave)*****

```
set rng [new RNG]
$rng seed [lindex $argv 0]
set u [new RandomVariable/Uniform]
$u set min_ 0
$u set max_ 1
$u use-rng $rng

#Schedule events for bursty traffic as cbr4a agents
set simTime4a_ 1
while {$simTime4a_  < 86400.00 } {
        set burstTime4a_ [$burstDuration value]
        set startTime [expr [expr ([$u value]) + {$simTime4a_ }]]
        $ns at  $startTime "$cbr4a start"
        set simTime4a_ [expr $startTime + $burstTime4a_ ]
        #puts " CBR Traffic Stops at $simTime4a_"
        $ns at $simTime4a_ "$cbr4a stop"
        set simTime4a_ [expr {$simTime4a_ + $burstTime4a_}]
}

#Schedule events for cbr4b and cbr4c agents will continue until 24 hours
$ns at [expr [expr ([$u value]) + 1.0]] "$cbr4b start"
$ns at 86400.00 "$cbr4b stop"
$ns at [expr [expr ([$u value]) + 1.0]] "$cbr4c start"
$ns at 86400.00 "$cbr4c stop"
```

#***Schedule events: Traffic in the Reverse Direction (Slave-to-master)*****

```
#Schedule events for bursty traffic as cbr5a agents
set simTime5a_ 1
while {$simTime5a_ < 86400.00 } {
        set burstTime5a_ [$burstDuration value]
```

```
        #puts [format " Burst Time at Node n5 is %-4.3f  " $burstTime5a_]
        set startTime [expr [expr ([$u value]) + {$simTime5a_ }]]
        $ns at  $startTime "$cbr5a start"
        set simTime5a_ [expr $startTime + $burstTime5a_ ]
        $ns at $simTime5a_ "$cbr5a stop"
        set simTime5a_ [expr {$simTime5a_ + $burstTime5a_}]
}

#Schedule events for cbr5b and cbr5c agents will continue until 24 hours
$ns at [expr [expr ([$u value]) + 1.0]] "$cbr5b start"
$ns at 86400.00 "$cbr5b stop"
$ns at [expr [expr ([$u value]) + 1.0]] "$cbr5c start"
$ns at 86400.00 "$cbr5c stop"

#Schedule events for clock agents
$ns at 1.0 "$m1 start"
$ns at 1.0 "$s1 start"

$ns at 86401.05 "$m1 stop"
$ns at 86401.05 "$s1 stop"
$ns at 86401.1 "finish"

#Run the simulation
$ns run
```

A.2 NS-2 TCL Script Code for Voice Centric Traffic Model

In this test case, a TCL script for voice centric traffic model, which is described in Section 5.1.1.2, is presented with the static traffic load described in Section 5.3.1.1. The results of this test case are discussed in Appendix C.

NS-2 TCL Script

```
set ns [new Simulator]

#Open a trace file
set tracefd [open clktest1v.tr w]
$ns trace-all $tracefd
set nf [open clktest1v.nam w]
$ns namtrace-all $nf
set graphData [open graphData.txt w]
```

```
#Define a 'finish' procedure
proc finish {} {
        global ns graphData nf tracefd
        $ns flush-trace
        close $nf
        close $tracefd
        close $graphData
        #exec nam clktest1v.nam &
        exit 0
}

#Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

#Link configurations
$ns duplex-link      $n0 $n1      1Mb   100ns  DropTail
$ns duplex-link      $n1 $n2      1Mb   100ns  DropTail
$ns duplex-link      $n2 $n3      1Mb   100ns  DropTail
$ns duplex-link      $n1 $n4      1Mb   100ns  DropTail
$ns duplex-link      $n2 $n5      1Mb   100ns  DropTail

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

# Generate bursts lasting between 0.1 s to 3 s
set rng1 [new RNG]
$rng1 seed [lindex $argv 0]
set burstDuration [new RandomVariable/Uniform]
$burstDuration set min_ 0.1
$burstDuration set max_ 3
$burstDuration use-rng $rng1

# ****** 80% traffic load in the forward direction (Master-to-slave) ******

# 15% of the load based on maximum sized packet (1518 bytes) generated at node
n4, which sends traffic to node n5
```

```
#Create a UDP agent and attach it to node n4
set udp4a [new Agent/UDP]
$udp4a set packetSize_ 1518
$ns attach-agent $n4 $udp4a

# Create a CBR traffic source, which will generate bursty traffic and attach it to udp4a
set cbr4a [new Application/Traffic/CBR]
$cbr4a set packetSize_ 1518
$cbr4a set rate_ 0.24Mb
$cbr4a attach-agent $udp4a

#Create a Null agent (a traffic sink) and attach it to node n5
set null5 [new Agent/Null]
$ns attach-agent $n5 $null5

#Connect the traffic source with the traffic sink
$ns connect $udp4a $null5
```

 # 80% of the load based on minimum sized packet (64 bytes) generated at node n4, which sends traffic to node n5

```
#Create a UDP agent and attach it to node n4
set udp4b [new Agent/UDP]
$udp4b set packetSize_ 64
$ns attach-agent $n4 $udp4b

# Create a CBR traffic source, which will generate bursty traffic and attach it to udp4a
set cbr4b [new Application/Traffic/CBR]
$cbr4b set packetSize_ 64
$cbr4b set rate_ 0.64Mb
$cbr4b attach-agent $udp4b

#Connect the traffic source with the traffic sink
$ns connect $udp4b $null5
```

 # 5% of the load based on medium sized packet (576 bytes) generated at node n4, which sends traffic to node n5

```
#Create a UDP agent and attach it to node n4
set udp4c [new Agent/UDP]
$udp4c set packetSize_ 576
$ns attach-agent $n4 $udp4c

# Create a CBR traffic source and attach it to udp4c
set cbr4c [new Application/Traffic/CBR]
$cbr4c set packetSize_ 576
```

```
$cbr4c set rate_ 0.04Mb
$cbr4c attach-agent $udp4c

#Connect the traffic source with the traffic sink
$ns connect $udp4c $null5

# ***20% "Static" packet load in the reverse direction ( Slave-to-Master)***

# 15% of the load based on maximum sized packet (1518 bytes) generated at node n5, which sends traffic to node n4

#Create a UDP agent and attach it to node n5
set udp5a [new Agent/UDP]
$udp5a set packetSize_ 1518
$ns attach-agent $n5 $udp5a

# Create a CBR traffic source, which will generate bursty traffic and attach it to udp5a
set cbr5a [new Application/Traffic/CBR]
$cbr5a set packetSize_ 1518
$cbr5a set rate_ 0.06Mb
$cbr5a attach-agent $udp5a

#Create a Null agent (a traffic sink) and attach it to node n4
set null4 [new Agent/Null]
$ns attach-agent $n4 $null4

#Connect the traffic source with the traffic sink
$ns connect $udp5a $null4

# 80% of the load based on minimum sized packet (64 bytes) generated at node n5, which sends traffic to node n4

#Create a UDP agent and attach it to node n5
set udp5b [new Agent/UDP]
$udp5b set packetSize_ 64
$ns attach-agent $n5 $udp5b

# Create a CBR traffic source and attach it to udp5b
set cbr5b [new Application/Traffic/CBR]
$cbr5b set packetSize_ 64
$cbr5b set rate_ 0.16Mb
$cbr5b attach-agent $udp5b

#Connect the traffic source with the traffic sink
$ns connect $udp5b $null4
```

 # 5% of the load based on medium sized packet (576 bytes) generated at node n5, which sends traffic to node n4

```
#Create a UDP agent and attach it to node n5
set udp5c [new Agent/UDP]
$udp5c set packetSize_ 576
$ns attach-agent $n5 $udp5c

# Create a CBR traffic source and attach it to udp5c
set cbr5c [new Application/Traffic/CBR]
$cbr5c set packetSize_ 576
$cbr5c set rate_ 0.01Mb
$cbr5c attach-agent $udp5c

#Connect the traffic source with the traffic sink
$ns connect $udp5c $null4

#Create master clock agent
set m1 [new Agent/Clock]
$m1 set offset 1
$m1 set rate 1
$m1 set timeToDisplayInfo 1
$m1 set masterClock 1
$m1 set gpsSignal 0
$m1 set enableRLS 0
$m1 set enableLockedRLS 0
$m1 set RLSFreq 1
$m1 set tempProfileName -1
$m1 set ageing 0
$m1 set driftInterval 1
$m1 set enable1588Logs 1
$ns attach-agent $n0 $m1

#  Create slave clock agent
set s1 [new Agent/Clock]
$s1 set offset 1
#slave clock rate is drifting100 ppb (100 ns) faster than the master clock
$s1 set rate 1.000000100
$s1 set timeToDisplayInfo 1
$s1 set masterClock 0
$s1 set gpsSignal 0
$s1 set enableLockedRLS 0
$s1 set enableRLS 0
$s1 set RLSFreq 0
$s1 set tempProfileName -1
$s1 set driftInterval 1
```

```
$s1 set timeStampReqFreq 1
$s1 set masterAddr 0
$s1 set masterPort 0
$s1 set enable1588Logs 1
$s1 set enable1588Delays 0
$ns attach-agent $n3 $s1

#*****Schedule events: Traffic in the Forward Direction (Master-to-Slave)*****
set rng [new RNG]
$rng seed [lindex $argv 0]
set u [new RandomVariable/Uniform]
$u set min_ 0
$u set max_ 1
$u use-rng $rng

#Schedule events for bursty traffic as cbr4a agents
set simTime4a_ 1
while {$simTime4a_  < 86400.00 } {
        set burstTime4a_ [$burstDuration value]
        set startTime [expr [expr ([$u value]) + {$simTime4a_ }]]
        $ns at  $startTime "$cbr4a start"
        set simTime4a_ [expr $startTime + $burstTime4a_ ]
        #puts " CBR Traffic Stops at $simTime4a_"
        $ns at $simTime4a_ "$cbr4a stop"
        set simTime4a_ [expr {$simTime4a_ + $burstTime4a_}]
}

#Schedule events for cbr4b and cbr4c agents will continue until 24 hours
$ns at [expr [expr ([$u value]) + 1.0]] "$cbr4b start"
$ns at 86400.00 "$cbr4b stop"
$ns at [expr [expr ([$u value]) + 1.0]] "$cbr4c start"
$ns at 86400.00 "$cbr4c stop"

#*****Schedule events: Traffic in the Reverse Direction (Slave-to-master)*****

#Schedule events for bursty traffic as cbr5a agents
set simTime5a_ 1
while {$simTime5a_ < 86400.00 } {
        set burstTime5a_ [$burstDuration value]
        #puts [format " Burst Time at Node n5 is %-4.3f  " $burstTime5a_]
        set startTime [expr [expr ([$u value]) + {$simTime5a_ }]]
        $ns at  $startTime "$cbr5a start"
         set simTime5a_ [expr $startTime + $burstTime5a_ ]
        $ns at $simTime5a_ "$cbr5a stop"
        set simTime5a_ [expr {$simTime5a_ + $burstTime5a_}]
}
```

```
#Schedule events for cbr5b and cbr5c agents will continue until 24 hours
$ns at [expr [expr ([$u value]) + 1.0]] "$cbr5b start"
$ns at 86400.00 "$cbr5b stop"
$ns at [expr [expr ([$u value]) + 1.0]] "$cbr5c start"
$ns at 86400.00 "$cbr5c stop"

#Schedule events for clock agents
$ns at 1.0 "$m1 start"
$ns at 1.0 "$s1 start"
$ns at 86401.05 "$m1 stop"
$ns at 86401.05 "$s1 stop"
$ns at 86401.1 "finish"

#Run the simulation
$ns run
```

Appendix B : Additional Test Cases Results for Data Centric Traffic Model

In this appendix, additional test cases results with data centric traffic model, which are not covered in Chapter 5, are presented.

B.1 Slave Clock Synchronization with the Slow Change in Network Load over an Extremely Long Timescale

Description

In this test case, we demonstrate a slow change in network load over an extremely long timescale between the master and the slave clock. Using the traffic model described in Section 5.1.1.1, the load profile is depicted in Figure 25. The traffic load is introduced at a simulation time of 1 second and stopping at a simulation time of 24 hours. Here in the **forward** direction (master-to-slave), the network load is smoothly changing from 20% to 80% with 1% increments every 12 minutes and back over a 24 hours period. In the **reverse** direction, the network load is smoothly changing from 10% to 55% with 1% increments every 16 minutes and back over a 24 hours period. The test case examines the effects of slow changes in network load over a long period of time on the slave clock synchronization with a DAC model enabled on the slave clock.

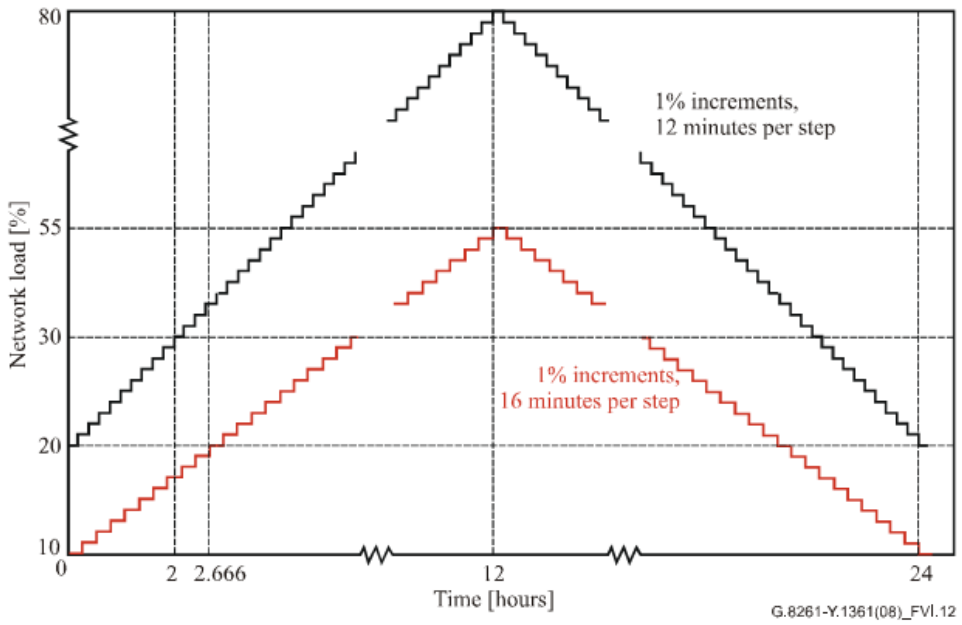

Figure 25: Load Profile Demonstrating Slow Changes in Network Load over an Extremely Long Time Scale [3]

Results

Figure 26 shows the slave clock synchronization accuracy with respect to the master clock with the DAC model applied on the slave clock following the load profile described in Appendix B.1.1. Statistical data are provided in Table 10.

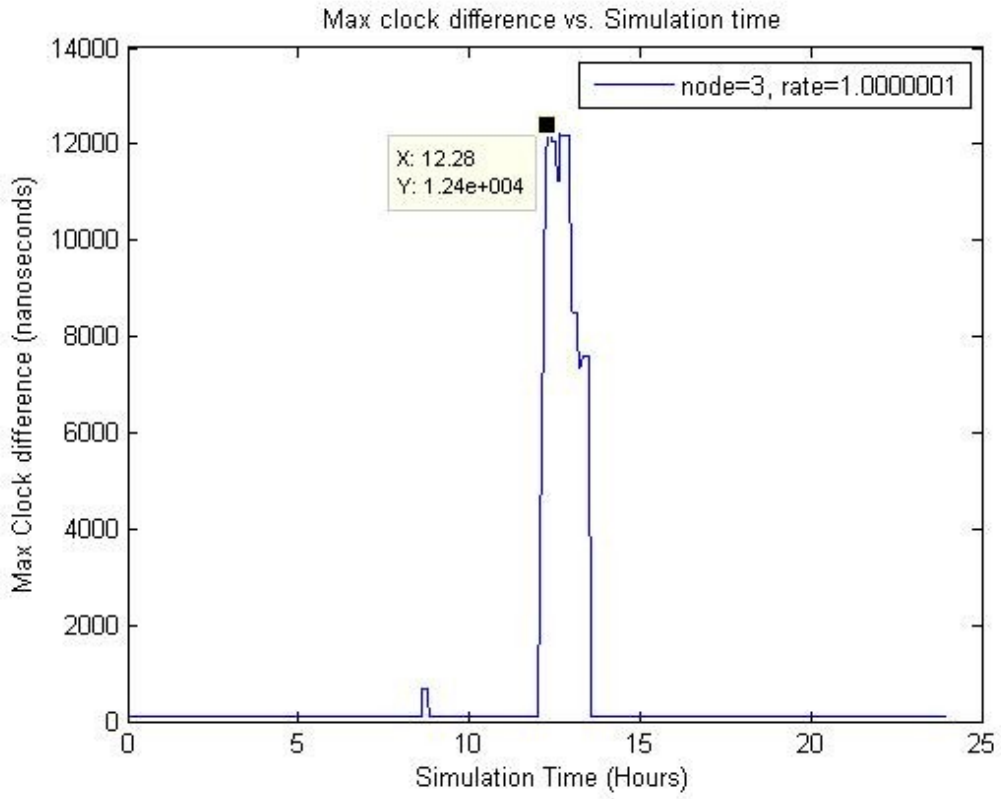

Figure 26: Slave Clock Synchronization with Slow Changes in Network Load - using the DAC Model

Table 10: Statistical Data for Slow Changes in Network Load

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	42965 (49.73%)
Avg. no. of samples passed the update sample filter	42308 (48.97%)
Max. of the max. interval between successive successful updates (seconds)	2367 s (0.66 hours)
Avg. of the avg. clock accuracy	193 ns
Max. Of the max. clock difference (ns) with DAC model	12.4 μs

Discussion

From Figure 26, the maximum of the maximum clock difference between the slave clock and the master clock is 12.4 μs with the slow changes in the network load. The resultant clock difference is bounded within 100 ns until 9 hours of the simulation time. At 9 hours of the simulation time, the slave clock is under-corrected with a relatively small saved offset value for a long period of time. Hence, the clock difference is accumulated over time and increased to about 1.8 μs. A similar effect is observed at 12.3 hours of the simulation time. The synchronization packets experienced highly asymmetric latencies due to 80% load in the forward direction and 55% load in the reverse direction. Hence, the slave clock is under-corrected with the saved offset value for a long period of time. As a result, the clock difference is accumulated over time and increased to about 12.4 μs. This effect continues until 14 hours of the simulation time. As soon as the slave clock is updated with a calculated offset value that passed both the filters, the clock difference reduced to 100 ns and continues until end of the simulation time. From the statistics provided in Table 10, about 50% samples passed the 'R' test and about 49% samples are used directly to update the slave clock. In addition, the average of the average slave accuracy is about 193 ns. It indicated that the slave clock achieved high accuracy with the slow change in the network load over an extremely long period of time scale.

B.2 Slave Clock Synchronization with the Temporary Network Outages and Restoration

Description

In this test case, we demonstrate a temporary network outage and restoration between the slave and the master clock. Using the traffic model described in Section 5.1.1.1, the

traffic load is introduced at a simulation time of 1 second and stopped at a simulation time of 24 hours. Here 40% of network load is introduced in the **forward** direction (master-to-slave) and 30% load in the **reverse** direction (slave-to-master). It also demonstrates temporary network outages for 100 seconds, where the slave clock is disconnected from the master clock. Network connection is removed at time 12 hours and continued until 12.028 hours of the simulation time, and then the network connection is restored. The test case examines the ability of the slave clock to survive network outage and recover on restoration.

Results

Figure 27 shows the slave clock synchronization accuracy with respect to the master clock with the DAC model applied on the slave clock. Statistical data are provided in Table 11.

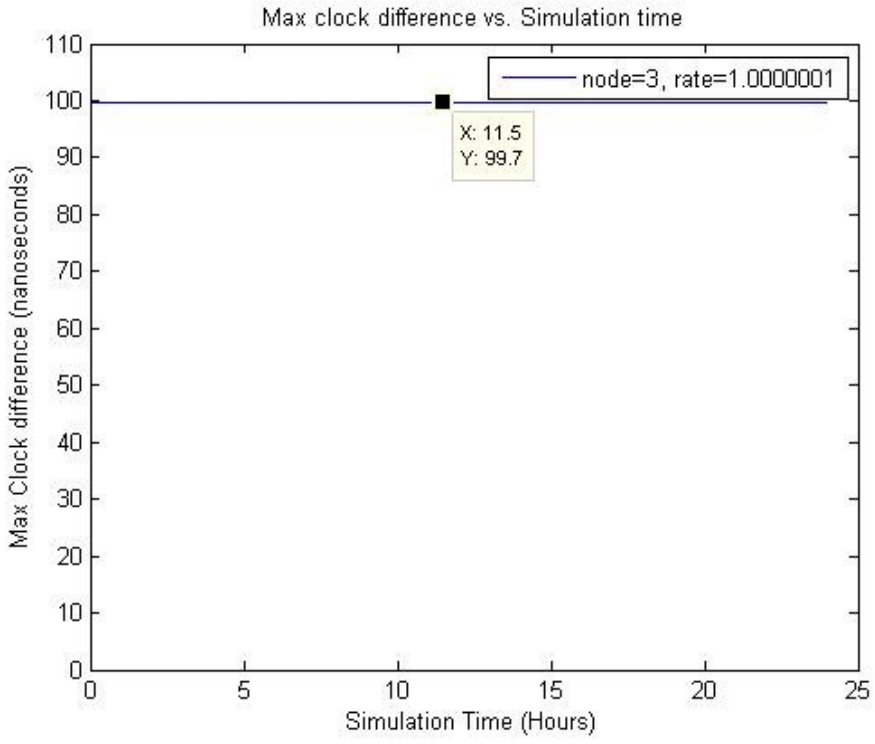

Figure 27: Slave Clock Synchronization with the Temporary Network Outage and Restoration

Table 11: Statistical Data for Temporary Network Outage and Restoration

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	40192 (46.52%)
Avg. no. of samples passed the update sample filter	39441(45.65%)
Max. of the max. interval between successive successful updates (seconds)	119 s
Avg. of the avg. clock accuracy	99.70 ns
Max. Of the max. clock difference (ns) with DAC model	100 ns

Discussion

From Figure 27, the maximum of the maximum clock difference between the slave and the master clock is about 100 ns. The resultant clock difference implies that the slave clock achieved high accuracy w.r.t the master clock with temporary network outage and restoration. The slave clock is temporarily disconnected from the master clock for 100 s, starts at a simulation time of 12 hours and continues until 12.028 hours. Hence, the slave clock did not receive any updates from the master clock during that period of time, and the slave clock is updated with the saved offset value while it was disconnected. Therefore, the slave accuracy remains high w.r.t the master clock until the end of the simulation time. From the statistics provided in Table 11, about 47% samples passed the 'R' test and about 46% samples are used directly to update the slave clock. In addition, the maximum of the maximum interval between successive successful update is 119 s. It reflects that the slave clock was disconnected from the master clock for 100 seconds and updated with the saved offset value during that period of time. Moreover, the average of

the average slave clock accuracy is about 99.70 ns. It indicates that the slave accuracy remains high w.r.t the master clock until the end of the simulation time

B.3 Slave Clock Synchronization with the Temporary Network Congestion and Restoration

Description

In this test case, we demonstrate temporary network congestion and restoration between the slave and the master clock. Using the traffic model described in Section 5.1.1.1, the traffic load is introduced at a simulation time of 1 second and stopped at a simulation time of 24 hours. Here 40% of network load is introduced in the **forward** direction (master-to-slave) and 30% load in the **reverse** direction (slave-to-master). At 7 hours of the simulation time, network load is increased to **100%** in both directions for 100 s, then restored. The test case examines the ability of the slave clock to survive temporary congestion in a packet network.

Results

Figure 28 shows the slave clock difference with respect to the master clock with the DAC model applied on the slave clock. Statistical data are provided in Table 12.

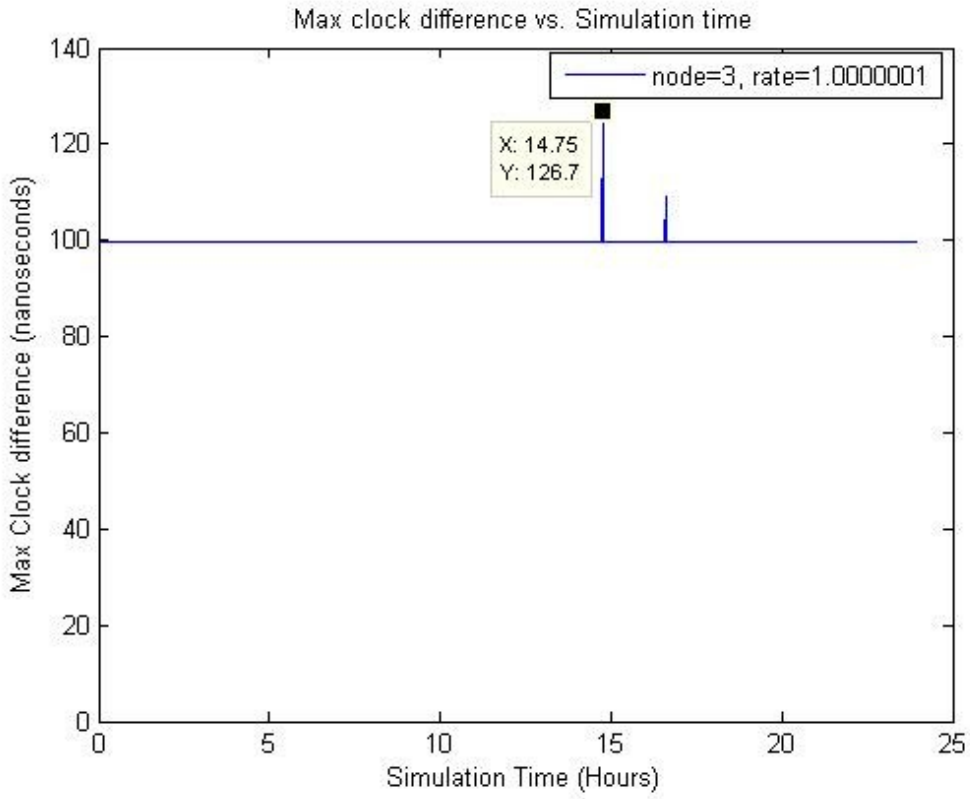

Figure 28: Slave Clock Synchronization with the Temporary Network Congestion and Restoration

Table 12: Statistical Data for Temporary Network Congestion and Restoration

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	41184(46.67%)
Avg. no. of samples passed the update sample filter	39647(45.89%)
Max. of the max. interval between successive successful updates (seconds)	22 s
Avg. of the avg. clock accuracy	99.70 ns
Max. Of the max. clock difference (ns) with DAC model	127 ns

Discussion

From Figure 28, the maximum of the maximum clock difference between the slave and the master clock is about 127 ns. The resultant clock difference implies that the slave accuracy w.r.t the master clock is high with temporary network congestion and recovers on restoration. The clock difference is bounded to 100 ns until 14.75 hours of the simulation time. At time 14.75 hours of the simulation time, the clock packets experienced highly asymmetric latencies and the slave clock is updated with a small saved offset value for a long period of time. Thus, the difference between the anticipated offset value and the saved offset value is accumulated over time and is increased to about 127 ns. A similar effect is observed at time 17 hours of the simulation time. From the statistics provided in Table 12, about 47% samples passed the 'R' test and about 46% samples are used to update the slave clock. In addition, the maximum of the maximum interval between successive successful update is 22 s and the average of the average slave accuracy is about 99.7 ns. It is evident that the slave clock is updated frequently with the calculated offset, which passed both the filters and hence, the slave clock is maintaining high accuracy w.r.t the master clock.

B.4 Slave Clock Synchronization - Re-route Network Traffic to Bypass One Switch

Description

In this test case, we demonstrate re-routing traffic (in both directions) to bypass **one** switch between the slave and the master clock. Using the traffic model described in Section 5.1.1.1, the traffic load is introduced at a simulation time of 1 second and stopped at a simulation time of 24 hours. Here 40% of the network load is introduced in the

forward direction (master-to-slave) and 30% load in the **reverse** direction (slave-to-master). Network traffic is re-routed for 5 minutes to bypass **one** switch, starting at 10 hours of the simulation time and continues until 10.083 hours, and then restored to the original path. The test case examines the effects of re-routing network traffic to bypass **one** switch, which causes a step change in the packet network delay.

The network topology shown in Figure 29 is used to evaluate the performance of the slave clock accuracy w.r.t the master clock.

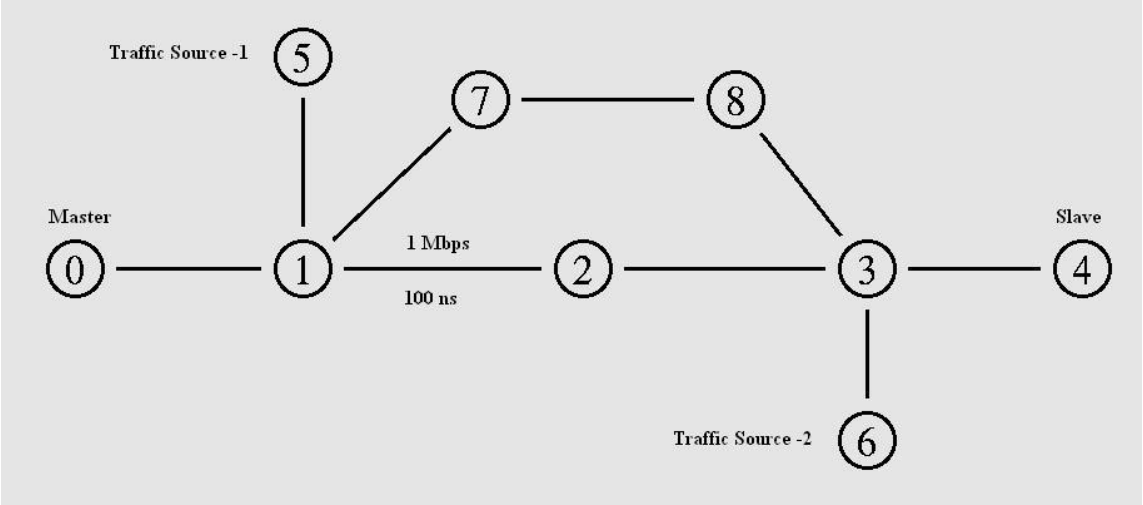

Figure 29: Network Topology for Re-routing Traffic to Bypass One Switch

The network topology consists of a master node (n0) connected to a slave node (n4) with three intermediate nodes n1, n2, and n3. Two traffic sources are introduced. One of the traffic sources is node n5, which sends traffic to node n6. Another traffic source is node n6, with traffic destined to node n5. By default, network traffic follows the route n5 → n1→ n2 → n3 → n6 and vice versa. At 10 hours of the simulation time, network traffic is re-routed from n5 → n1→ n7 → n8 → n3 → n6 and vice versa for 5 minutes and then restored to the original path. The parameters mentioned in the simulation set up section

(Section 5.1) are used in this test case. In addition, Distance Vector (DV) routing protocol is used for dynamically re-routing network traffic.

Results

Figure 30 shows the slave clock synchronization accuracy with respect to the master clock with the DAC model applied on the slave clock. Statistical data are provided in Table 13.

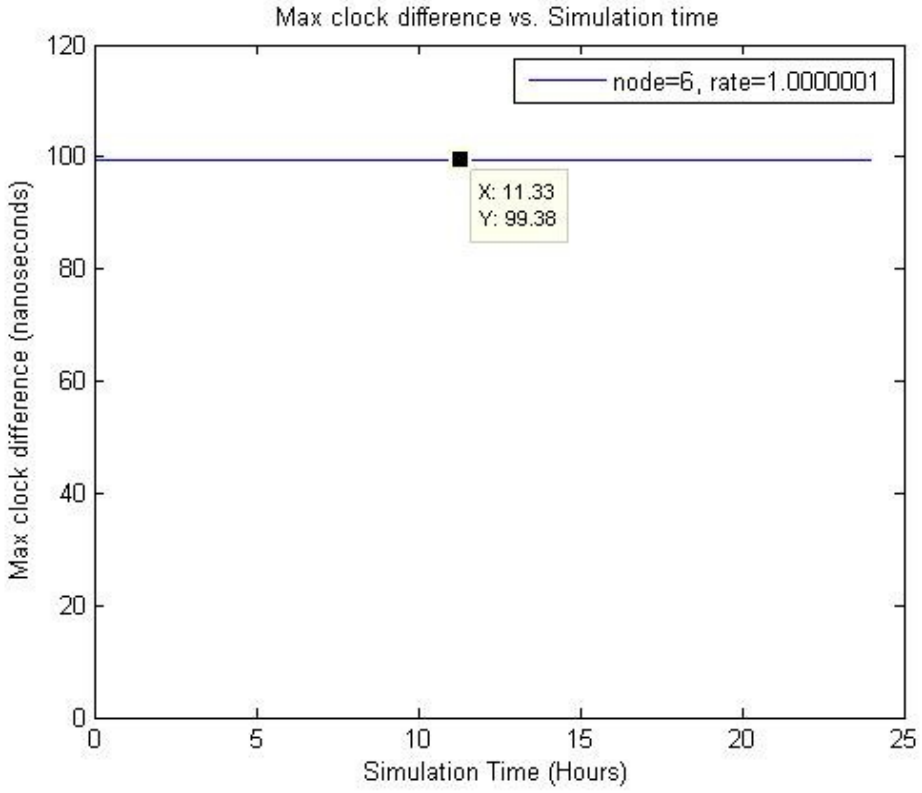

Figure 30: Slave Clock Synchronization- Re-route Network Traffic to Bypass One Switch

Table 13: Statistical Data for Re-routing Network Traffic to Bypass One Switch

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	24321 (28.15%)
Avg. no. of samples passed the update sample filter	21886 (25.33%)
Max. of the max. interval between successive successful updates (seconds)	41s
Avg. of the avg. clock accuracy	99.95 ns
Max. Of the max. clock difference (ns) with DAC model	100 ns

Discussion

From Figure 30, the maximum of the maximum clock difference between the slave and the master clock is 100 ns. The resultant clock difference implies that the slave accuracy w.r.t the master is high with re-routing network traffic to bypass one switch. From the statistics provided in Table 13, about 28% samples passed the 'R' test and about 25% samples are used directly to update the slave clock. In addition, the maximum interval between successive successful updates is 41 seconds. It indicates that the slave clock is updated frequently with the calculated offset value. Moreover, the average of the slave accuracy is 99.55 ns. It implies that the saved offset value is very close to the anticipated offset value. As a result, the slave accuracy remains high until end of the simulation time.

B.5 Slave Clock Synchronization - Re-route Network Traffic to Bypass Three Switches

Description

In this test case, we demonstrate re-routing traffic (in both directions) to bypass **three** switches between the slave and the master clock. Using the traffic model described in Section 5.1.1.1, the traffic load described in the previous Appendix B.4 is introduced at a simulation time of 1 second and stopped at a simulation time of 24 hours. The test case examines the effects of re-routing network traffic to bypass **three** switches, which causes a step change in the packet network delay.

The network topology shown in Figure 31 is used to evaluate the performance of the slave clock accuracy w.r.t the master clock.

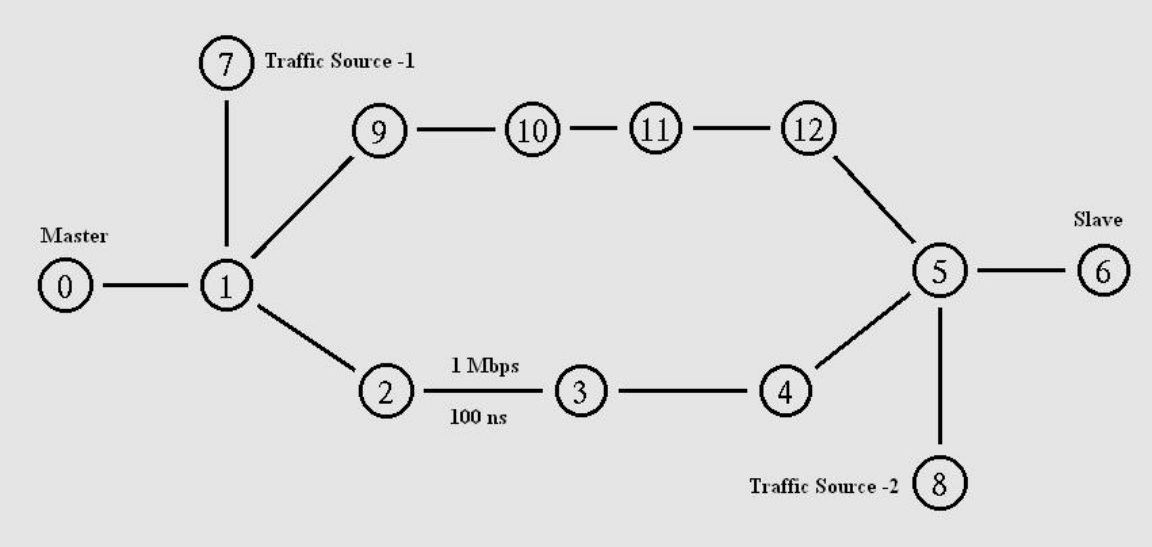

Figure 31: Network Topology for Re-routing Traffic to Bypass Three Switches

The network topology consists of a master node (n0) connected to a slave node (n6) with five intermediate nodes n1, n2, n3, n4, and n5. Two traffic sources are also introduced. One of the traffic sources is node n7, which sends traffic to node n8. Another traffic

source is node n8, with traffic destined to node n7. By default, network traffic follows the route n7 → n1→ n2 → n3 → n4 →n5 → n6 and vice versa. At 10 hours of the simulation time, network traffic is re-routed from n7 → n1→ n9 → n10 → n11→ n12 → n5 → n6 and vice versa for 5 minutes and then restored to the original path. The parameters mentioned in the simulation set up section (Section 5.1) are used in this test case. In addition, Distance Vector (DV) routing protocol is used for dynamically re-routing network traffic.

Results

Figure 32 shows the slave clock synchronization accuracy with respect to the master clock with the DAC model applied on the slave clock. Statistical data are provided in Table 14.

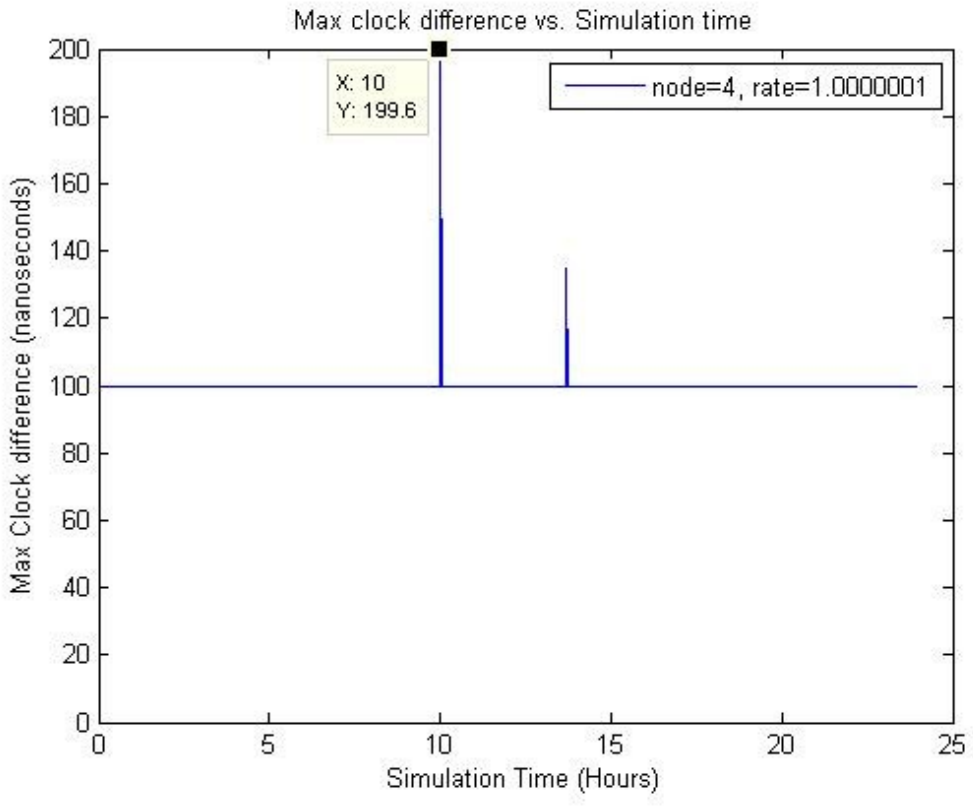

Figure 32: Slave Clock Synchronization- Re-route Network Traffic to Bypass Three Switches

Table 14: Statistical Data for Re-routing Network Traffic to Bypass Three Switches

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	24549 (28.41%)
Avg. no. of samples passed the update sample filter	15657 (18.12%)
Max. of the max. interval between successive successful updates (seconds)	49s
Avg. of the avg. clock accuracy	99.65 ns
Max. Of the max. clock difference (ns) with DAC model	200 ns

Discussion

From Figure 32, the maximum of the maximum clock difference is about 200 ns. The slave clock accuracy is increased to 200 ns as it is updated with a saved offset value, while the network traffic is re-routed through a new link. The slave clock accuracy is bounded within 100 ns before changing the routing path. When the original path (i.e. n7 → n1→ n2 → n3 → n4 → n5 → n6 and vice versa) is disconnected due to the failures in the network, DV routing protocol takes some time to calculate and set up a new path. Hence, the clock packets experienced highly asymmetric delays and saved offset values are used to update the slave clock. Thus, the difference between the anticipated offset value and the saved offset value is accumulated over time and increased to 200 ns. As soon as the slave clock is updated with the calculated offset value which passed both the filters, the clock difference is reduced to 100 ns. At 14 hours of the simulation time, the slave clock is updated with a relatively small saved offset value for a long period of time, the clock difference increased to about 130 ns. From the statistics provided in Table 14, about 28% samples passed the 'R' test and about 18% samples are used directly to update

the slave clock. In addition, the average of the average slave accuracy is 99.65 ns, which indicates that the slave clock achieved high accuracy with re-routing network traffic to bypass three switches. The maximum of the maximum interval between successive successful update is 41 s, which reflects that the slave clock is updated frequently.

Appendix C : Test Cases Results with Voice Centric Traffic Model

In this appendix, test cases with voice centric traffic model, which is described in Section 5.1.1.2, are presented. The network topology shown in Figure 8 is used to evaluate the performance of the slave clock, unless a different topology is specified in a test case.

C.1 Slave Clock Synchronization with Static Packet Load for Voice Centric Traffic Model

Description

In this test case, the first test case with a static packet load described in Section 5.3.1.1 is repeated using the voice centric traffic model. The performance of the slave clock accuracy with respect to the master clock is evaluated with static network load in both directions.

Results

Figure 33 shows the slave clock synchronization accuracy with respect to the master clock when a static packet load is introduced between the slave and the master clock with DAC model applied on the slave clock. Statistical data are provided in Table 15.

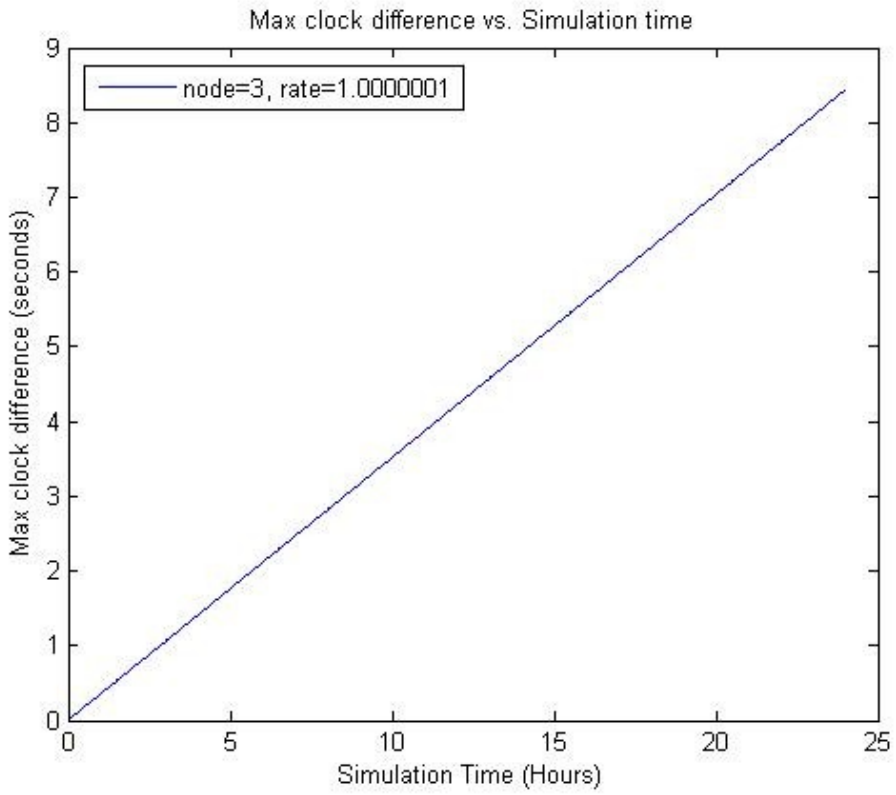

Figure 33: Slave Clock Synchronization with Static Packet Load-using Voice Centric Traffic Model

Table 15: Statistical Data for Static Packet Load – using Voice Centric Traffic Model

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	21427 (24.80%)
Avg. no. of samples passed the update sample filter	13180 (15.25%)
Max. of the max. interval between successive successful updates (seconds)	6463 s
Avg. of the avg. clock accuracy	10 µs
Max. Of the max. clock difference (ns) with DAC model	8.8 s

Discussion

From Figure 33, the maximum of the maximum clock difference between the slave and the master clock is about 8.8 seconds. The clock difference is increased to 8.8 seconds because the algorithm initializes the DAC model with a very large offset value (i.e. 97.7 μs), compared to the anticipated offset value (i.e. 100 ns). Hence, the calculated saved offset is 97.7 μs as well. Afterward, none of the samples passed both tests. Hence the slave clock is updated with the saved offset value and the calculation of 'R' is distorted at some point. The calculation of 'R' is subject to significant distortion when the offset is roughly on the same order of magnitude as the one-way latency because 'R' is an approximation of undefined asymmetry ratio. In this situation, none of the samples pass the 'R' test even if a good sample appears after a long interval. It is worth mentioning that the slave accuracy deteriorated due to having less bursty traffic in the voice centric traffic model (i.e. 15%), compared to the data centric traffic model (i.e. 60%) described in Section 5.1.1.1. In addition, the resultant difference is the worst case scenario, which does not happen all the time. We are collecting the data as the maximum of the maximum clock difference in each synchronization window for 50 simulations run, so the maximum of the maximum value appeared in Figure 33. It has been observed that the DAC model is initialized with large offset value for 2 simulations runs out of the 50 runs only. In another runs, none of the offset passed the 'R' test and the slave is updated with default offset value (i.e. 0). Therefore, the accumulated clock difference is 8.6 milliseconds. The remaining 46 simulations run are as expected. About 25% samples passed the 'R' test and about 15% samples are used directly to update the slave clock, shown in Table 15. As a result, the average of the average slave accuracy is 10 μs. An almost similar result is

observed for the test case with sudden large and persistent changes in network load described in Section 5.3.3.1 using the voice centric traffic profile. We believe that the DAC model will perform better if the amount of bursty traffic is relatively high. Thus, an additional traffic profile is presented in Appendix D, where various amounts of bursty traffic will be introduced to determine the minimum percentage of bursty traffic through which the slave clock is able maintain the high accuracy. Another solution is to introduce multiple master clocks. In this case, the slave clock may receive at least one good sample from one of these master clocks in order to update the slave clock. We will repeat this test case with multiple master clocks and with an additional traffic model for evaluating the slave synchronization accuracy, presented in Appendix C.5 and Appendix D respectively.

C.2 Slave Clock Synchronization with the Slow Change in Network Load for Voice Centric Traffic Model

Description

In this test case, slow change in network load over an extremely long timescale as described in Appendix B.1 is repeated considering the voice centric traffic model. The performance of the slave clock accuracy with respect to the master clock is evaluated with a slow change in network load in both directions.

Results

Figure 34 shows the slave clock synchronization accuracy with respect to the master clock, when a slow change in network load is introduced between the slave and the master clock with DAC model applied on the slave clock. Statistical data are provided in Table 16.

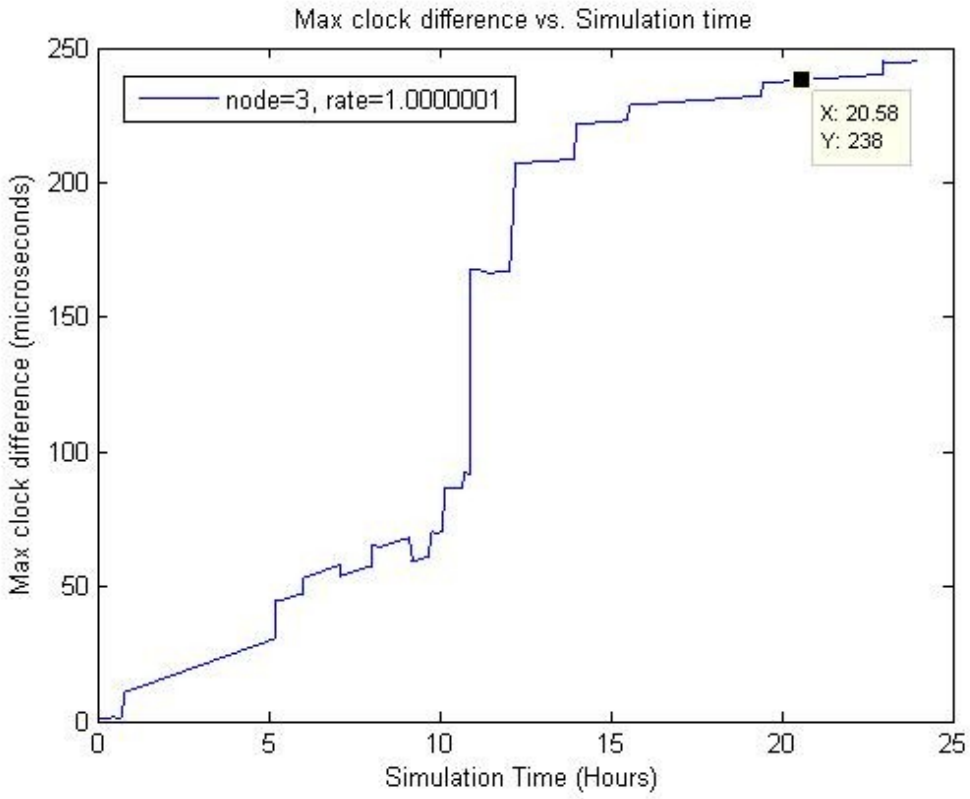

Figure 34: Slave Clock Synchronization with Slow Changes in Network Load -using Voice Centric Traffic Model

Table 16: Statistical Data for Slow Changes in Network Load – using Voice Centric Traffic Model

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	1773 (2.05%)
Avg. no. of samples passed the update sample filter	1672 (1.94%)
Max. of the max. interval between successive successful updates (seconds)	15900 s (4.42 hours)
Avg. of the avg. clock accuracy	25 μs
Max. Of the max. clock difference (ns) with DAC model	244 μs

Discussion

From Figure 34, the maximum of the maximum clock difference between the slave clock and the master clock is about 244 μs. The resultant difference implies that the slave clock is updated with a relatively small saved offset value for a long period of time. At the very beginning of the simulation, the slave clock is updated with the calculated offset values when we have a 20% load in the forward direction and 10% load in the reverse direction. As the network load is increased by 1% in both directions, fewer samples passed the 'R' test. So, the saved offset is used to update the slave clock. Hence the difference between the saved offset value and the anticipated offset value is accumulated over time. The slave clock accuracy further deteriorated and could not recover when the synchronization packets experienced highly asymmetric latencies due to 80% load in the forward direction and 55% load in the reverse direction. Similar to the previous test case, the resultant clock difference is the worst result out of 50 simulations run. However, from the statistics provided in Table 16, about 2% samples passed the 'R' test and less than 2% samples passed the 2nd stage filter. It is evident that the DAC model relies on the saved offset value for a long period of time since about 98% samples are highly asymmetric. Therefore, the average of the average slave clock accuracy is 25 μs.

C.3 Slave Clock Synchronization with the Temporary Network Outages and Restoration using Voice Centric Traffic Model

Description

In this test case, the test case with temporary network outage and restoration described in Appendix B.2 is repeated considering the voice centric traffic model. The performance of

the slave clock accuracy with respect to the master clock is evaluated with temporary network outage and restoration.

Results

Figure 35 shows the slave clock synchronization accuracy with respect to the master clock, when temporary network outage and restoration is introduced between the slave and the master clock with DAC model applied on the slave clock. Statistical data are provided in Table 17.

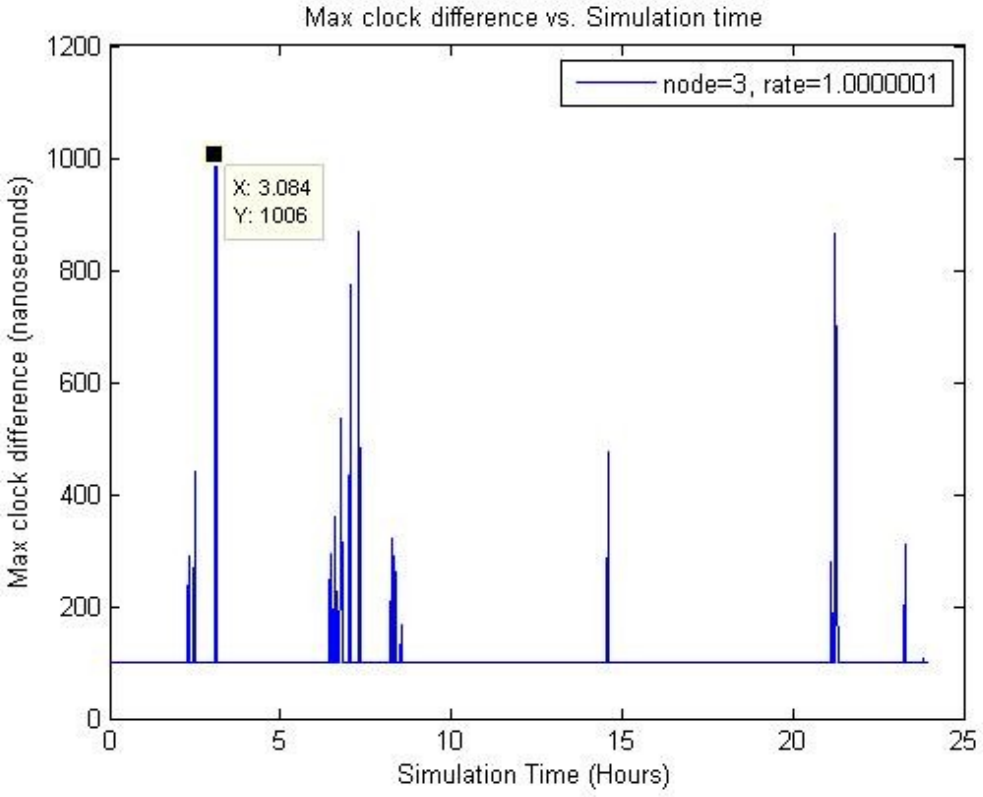

Figure 35: Slave Clock Synchronization with Temporary Network Outage and Restoration - using Voice Centric Traffic Model

Table 17: Statistical Data for Temporary Network Outage and Restoration – using Voice Centric Traffic Model

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	40736 (47.15%)
Avg. no. of samples passed the update sample filter	39342 (45.54%)
Max. of the max. interval between successive successful updates (seconds)	187 s
Avg. of the avg. clock accuracy	101 ns
Max. Of the max. clock difference (ns) with DAC model	1.06 μs

Discussion

From Figure 35, the maximum of the maximum clock difference between the slave and the master clock is about 1 μs. The resultant difference is increased to 1 μs due to updating the slave clock with a relatively small offset value for a long period of time. As a result, the difference accumulates over time. The slave accuracy is bounded within 100 ns until 3 hours of the simulation time. At 3 hours of the simulation time, synchronization packets experienced highly asymmetric latencies and the saved offset value is used for a long period of time for updating the slave clock. Thus, the accumulated clock difference is 450 ns. A similar effect is observed at 7 hours, 8 hours, 9 hours, 15 hours, 21 hours, and 23 hours of the simulation time. It should be noted that these effects appeared from different simulations runs. In all of these cases, the slave clock is recovered as soon as it is updated with the calculated offset, which passed both the filters. From the statistics provided in Table 17, about 47% samples passed the 'R' test and about 45% samples passed the update sample filter. In addition, the average of the average slave accuracy is

101 ns. It indicates that the slave clock achieved high accuracy when it did not receive synchronization updates from the master clock due to temporary network outage. The maximum of the interval between successive successful update is 187 seconds. It implies that the slave clock is updated frequently.

C.4 Slave Clock Synchronization with Temporary Network Congestion and Restoration using Voice Centric Traffic Model

Description

In this test case, the test case with temporary network congestion and restoration described in Appendix B.3 is repeated considering the voice centric traffic model. The performance of the slave clock accuracy with respect to the master clock is evaluated with temporary network congestion and restoration.

Results

Figure 36 shows the slave clock synchronization accuracy with respect to the master clock, when temporary network congestion and restoration is introduced between the slave and the master clock with DAC model applied on the slave clock. Statistical data are provided in Table 18.

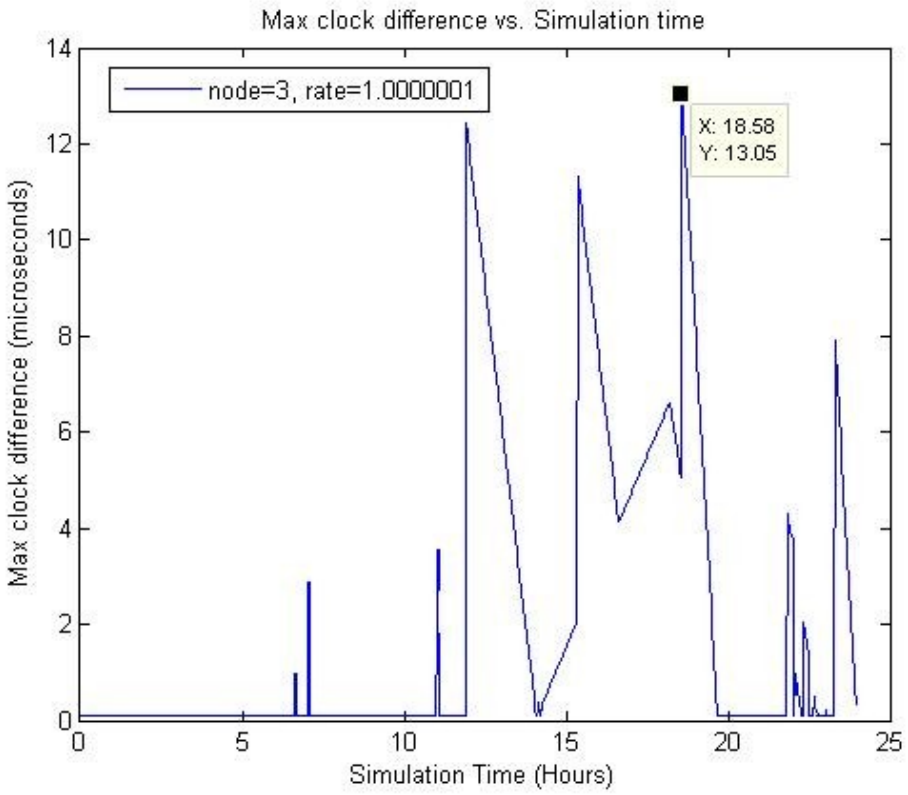

Figure 36: Slave Clock Synchronization with Temporary Network Congestion and Restoration - using Voice Centric Traffic Model

Table 18: Statistical Data for Temporary Network Congestion and Restoration – using Voice Centric Traffic Model

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	36040.1 (41.7%)
Avg. no. of samples passed the update sample filter	34120 (39.5%)
Max. of the max. interval between successive successful updates (seconds)	12455 s (3.46 hours)
Avg. of the avg. clock accuracy	146.57 ns
Max. Of the max. clock difference (ns) with DAC model	13.05 µs

Discussion

From Figure 36, the maximum of the maximum clock difference between the slave and the master clock is about 13 μs. The resultant difference is increased to 13 μs due to updating the slave clock with a relatively small offset value for a long period of time. The clock difference is bounded within 100 ns until 7 hours of the simulation time with 40% network load in the forward direction and 30% load in the reverse direction. At 7 hours of the simulation time, when asymmetric traffic load (e.g. 40% forward, 30% reverse) is increased to 100% for 5 minutes in both directions to cause temporary network congestion, the clock difference is increased to 3 μs, much lower compare to the maximum of the maximum value, 13 μs. The rationale behind this is that the asymmetric latencies are relatively reduced during that period of time when the network experiences 100% load in both directions. From the statistics provided in Table 18, about 41% samples passed the 'R' test and about 39% samples passed the update sample filter. In addition, the average of the average slave accuracy is 147 ns. It indicates that the slave clock achieved high accuracy with temporary congestion in the network. The maximum of the interval between successive successful update is about 3.46 hours. It implies that few samples are used directly to update the slave clock for one of the simulation runs.

C.5 Slave Clock Synchronization with Re-routing Network Traffic to Bypass One Switch using Voice Centric Traffic Model

Description

In this test case, the test case with re-routing network traffic to bypass one switch described in Appendix B.4 is repeated considering the voice centric traffic model. The

network topology shown in Figure 29 is used to evaluate the performance of the slave clock accuracy with respect to the master clock.

Results

Figure 37 shows the slave clock synchronization accuracy w.r.t the master clock, when re-routing network traffic to bypass one switch is introduced between the slave and the master clock with DAC model applied on the slave clock. Statistical data are provided in Table 19.

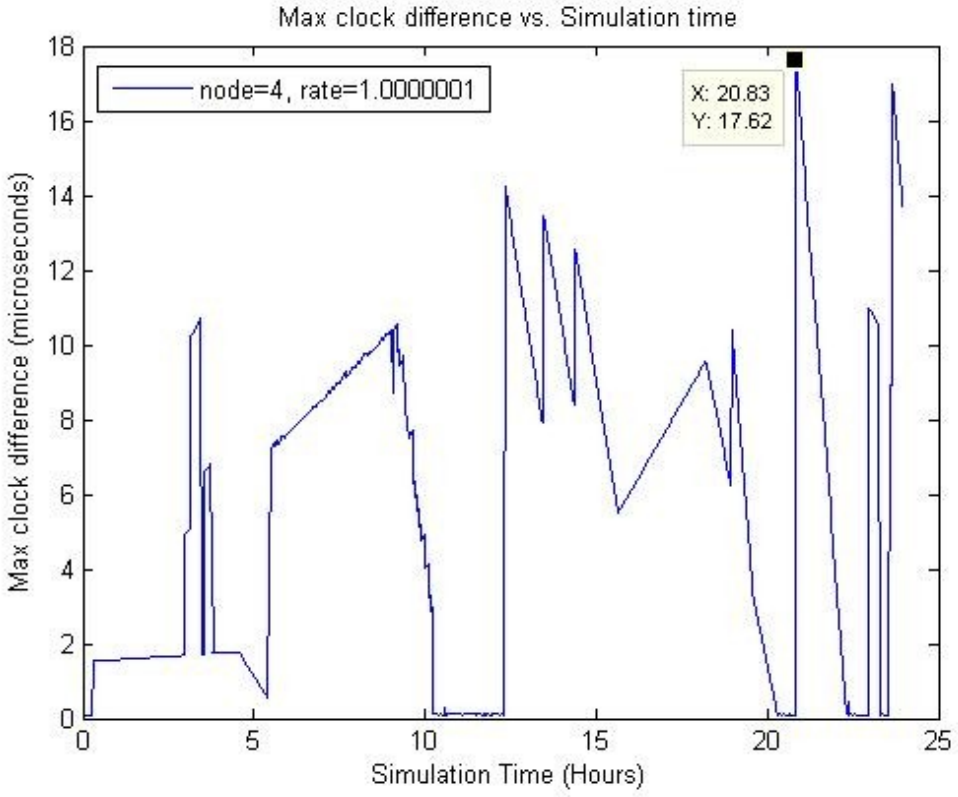

Figure 37: Slave Clock Synchronization with Re-routing Network Traffic to Bypass One Switch – using Voice Centric Traffic Model

Table 19: Statistical Data for Re-routing Network Traffic to Bypass One Switch – using Voice Centric Traffic Model

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	24630 (28.5%)
Avg. no. of samples passed the update sample filter	23460 (27.2%)
Max. of the max. interval between successive successful updates (seconds)	511 s
Avg. of the avg. clock accuracy	240 ns
Max. Of the max. clock difference (ns) with DAC model	17.6 μs

Discussion

From Figure 37, the maximum of the maximum clock difference between the slave and the master is about 18 μs. The resultant clock difference is increased to 18 μs due to updating the slave clock with a relatively small saved offset value for an extended period of time. At 10 hours of the simulation time, network traffic is re-routed for 5 minutes. Hence the clock packets experienced severe delays due to changing and setting up a new routing path. From the statistics provided in Table 19, about 28% samples passed the 'R' test and about 27% samples are used directly to update the slave clock. It has been observed that a large percentage of samples (i.e. about 42%) passed the 'R' test before re-directing network traffic and few amount of samples passed the 'R' test after changing the routing path. As a result, the slave clock is updated with the saved offset value and the difference between the anticipated offset value and the saved offset value is accumulated over time. The average of the average slave accuracy is 240 ns, which reflects that the slave clock is able to maintain high accuracy when network traffic is re-

routed to bypass one switch. An almost similar result is observed when network traffic is re-routed bypassing three switches.

C.6 Slave Clock Synchronization with Static Packet Load using Multiple Master Clocks for Voice Centric Traffic Model

From Appendix C.1.3, we observed that the slave accuracy deteriorates with high traffic load using the voice centric traffic model, particularly when the network experiences 80% static traffic in the forward direction and 20% static traffic in the reverse direction. So, we will examine the performance of the slave clock having multiple master clocks with high traffic load in the network.

Description

In this test case, the test case with static packet load described in Section 5.3.1.1 is repeated using the voice centric traffic model. The network topology shown in Figure 15 is used to evaluate the performance of the slave clock accuracy with respect to multiple master clocks. The parameters mentioned in the simulation set up section (Section 5.1) are used in this test case, except the IEEE 1588 synchronization frequency. 2 seconds IEEE 1588 synchronization frequency is considered in this test case. Both the master clocks initiate the IEEE 1588 message exchange according to the Equation 11, mentioned in Section 4.3. The data are collected as the maximum of the maximum values of the slave clock accuracy with respect to each master clock.

Results

Figure 38 shows the slave clock synchronization accuracy w.r.t both the master clocks, when the network has an 80% static packet load in the forward direction and a 20% load

in the reverse direction. The DAC model is applied on the slave clock. Statistical data are provided in Table 20.

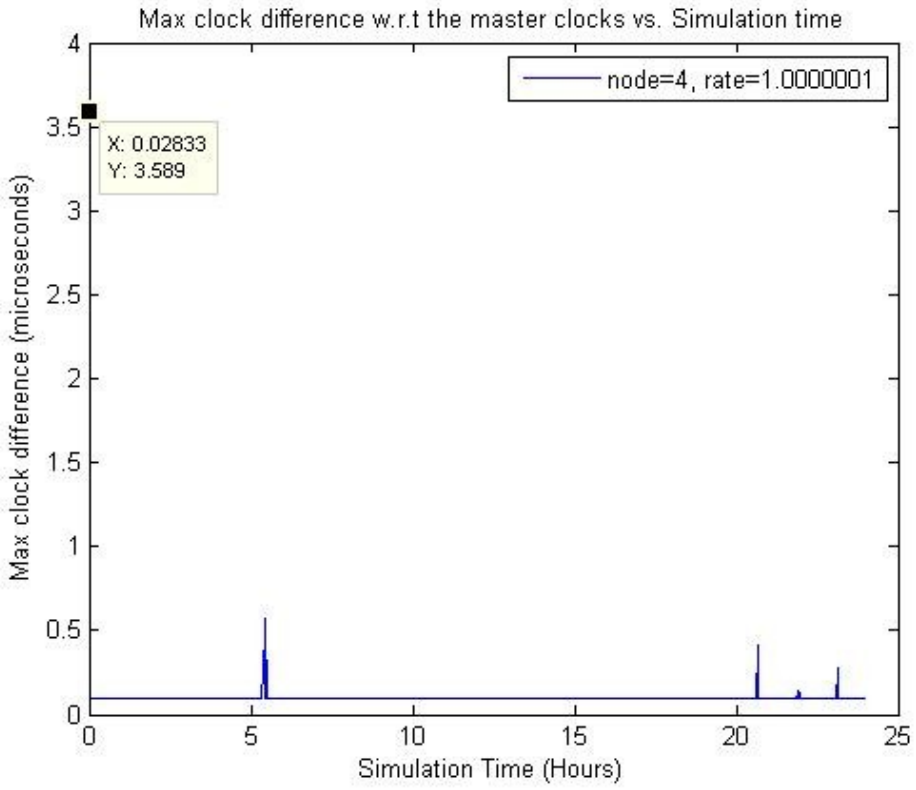

Figure 38: Slave Clock Synchronization with Static Packet Load w.r.t both the Master Clocks -using Voice Centric Traffic Model

Table 20: Statistical Data for Static Packet Load using Multiple Master Clocks (Voice Centric Traffic Model)

	Master 0	Master 1
Total no. of samples per run	43200	43200
Average no. of samples passed the ratio (R) test	43200 (100%)	7961 (18.43%)
Average no. of samples passed the update sample filter	43158 (99.90%)	7620 (17.64%)
Max. of the max. interval between successive successful updates (seconds)	2 s	338 s
Avg. of the avg. clock accuracy	100 ns	102 ns
Max. of the max. clock difference (ns) with DAC model	100 ns	3.62 µs

Discussion

From Figure 38, the maximum of the maximum slave clock accuracy w.r.t both the master clocks is about 3.6 µs. The resultant clock difference is increased to 3.6 µs because of initializing the DAC model after a long period of time, which is referred as stabilization period. Hence the slave clock is updated with the default saved offset value (i.e. 0) and the difference is accumulated over time according to the slave clock drifting rate. The slave accuracy is bounded within 100 ns until 5.5 hours of the simulation time. At time 5.5 hours, the synchronization packets experienced highly asymmetric delays and the slave clock is updated with the saved offset value for a long period of time. Thus the slave clock difference w.r.t the master clocks is increased to 500 ns. A similar effect is observed at 21 hours, 22 hours and 23 hours of the simulation time. From the statistics provided in Table 20, about 100% samples passed both the 'R' test and the update sample

filter w.r.t the first master clock (i.e. master 0). About 18.5% samples passed the 'R' test and about 17.5% samples are used directly to update the slave clock w.r.t the second master clock (i.e. master 1). Hence it is evident that the slave clock received at least one good sample from one of these master clocks (here first master clock) within the defined update interval in order to adjust the slave clock correctly. In addition, the maximum intervals between successive successful updates are 2 s w.r.t the master-0 and 338 s w.r.t the master-1. It indicates that the slave clock is updated frequently with the calculated offsets. Thus, the slave clock does not rely on the saved offset value for a long period of time. Moreover, the average of the average slave accuracy is about 100 ns w.r.t both the master clocks. It implies that the slave clock achieved high accuracy having two master clocks in the network and resolved the initialization problem stated in Appendix C.1.3 for the voice centric traffic model by providing more diversity. In addition, the slave accuracy is improved 2 times having two master clocks with static packet load in both directions.

Appendix D : Test Case Result with an Additional Traffic Model – 3

In this appendix, test cases with an additional traffic model are presented. The slave accuracy suffers when only a small fraction of bursty traffic exists in the network, as was the case with the voice-centric traffic model. So, an additional traffic model is introduced to determine the minimum percentage of bursty traffic through which the slave clock is able maintain the high accuracy all the time.

D.1 Network Traffic Model - 3 Descriptions

In this traffic model, we introduce three different scenarios. First, 70% of the network load is based on small size (64 bytes) CBR packets and 30% of the load is based on maximum size (1518 bytes) packets. Second, 60% of the network load is based on CBR packets and 40% of the load is based on maximum size packets. Finally, 50% of the network load is based on CBR packets and 50% of the load is based on maximum size packets. It should be noted that the maximum size packets occur in bursts lasting between 0.1 s to 3 s.

D.2 Slave Clock Synchronization with Static Packet Load for Traffic Model – 3

Description

In this test case, the first test case with a static packet described in Section 5.3.1.1 is repeated using traffic model-3. The performance of the slave clock is evaluated using varying percentage of bursty traffic with static packet loads in both directions.

Results

Figure 39 shows the slave clock synchronization accuracy w.r.t the master clock when 70% of the network load is based on CBR packets and 30% of the load is bursty traffic. Figure 40 also shows the slave clock synchronization accuracy w.r.t the master clock when 60% of the load is based on CBR packets and 40% of the load is bursty traffic. Figure 41 also shows the slave clock synchronization accuracy w.r.t the master clock when 50% of the load is based on CBR packets and 50% of the network load is bursty traffic. In all of these cases, a static packet load is introduced between the slave and the master clock with DAC model applied on the slave clock. Statistical data are provided in Table 21, Table 22, and Table 23 respectively.

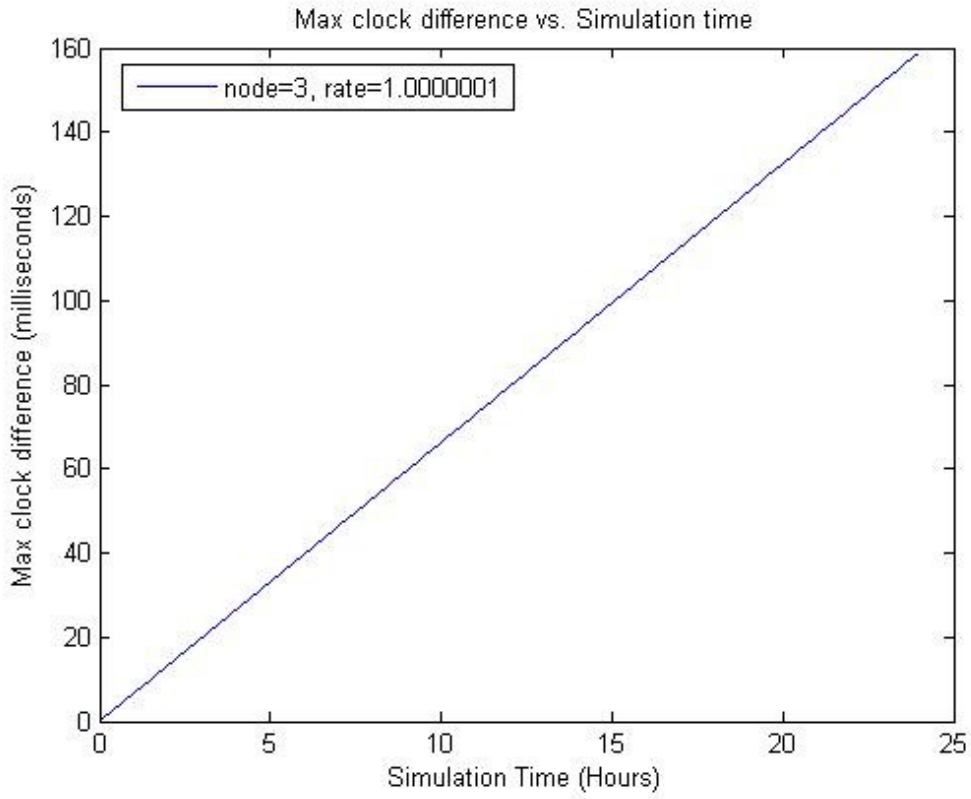

Figure 39: Slave Clock Synchronization with Static Packet Load -using 30% Bursty Traffic

Table 21: Statistical Data for Static Packet Load (Traffic Model-3) - using 30% Bursty Traffic

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	18600 (21.53%)
Avg. no. of samples passed the update sample filter	15493 (17.93%)
Max. of the max. interval between successive successful updates (seconds)	1421 s
Avg. of the avg. clock accuracy	3.12 ms
Max. Of the max. clock difference (ns) with DAC model	160 ms

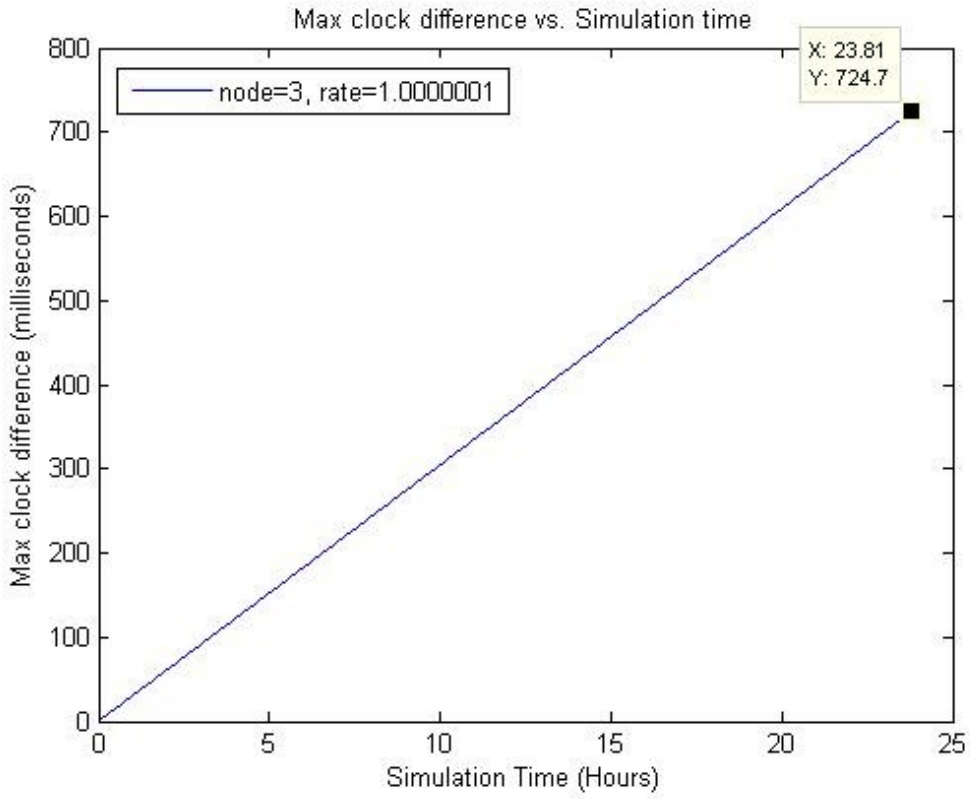

Figure 40: Slave Clock Synchronization with Static Packet Load -using 40% Bursty Traffic

Table 22: Statistical Data for Static Packet Load (Traffic Model-3) - using 40% Bursty Traffic

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	21852 (25.29%)
Avg. no. of samples passed the update sample filter	17936 (20.76%)
Max. of the max. interval between successive successful updates (seconds)	4817 s (1.34 hrs)
Avg. of the avg. clock accuracy	10.04 ms
Max. Of the max. clock difference (ns) with DAC model	725 ms

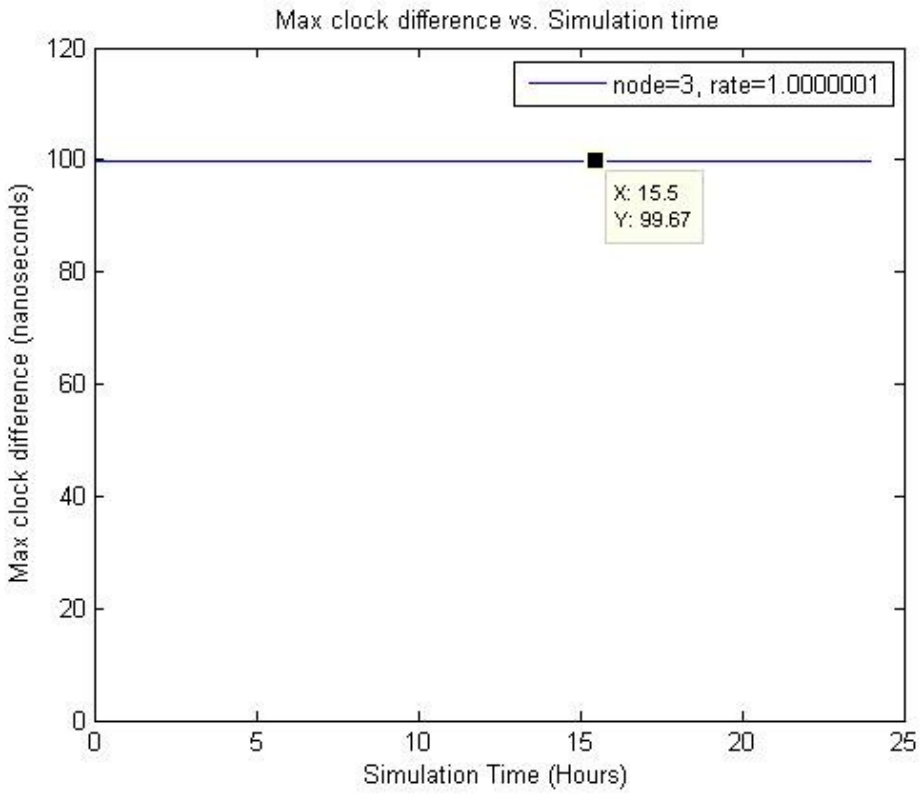

Figure 41: Slave Clock Synchronization with Static Packet Load -using 50% Bursty Traffic

Table 23: Statistical Data for Static Packet Load (Traffic Model-3) - using 50% Bursty Traffic

Total no. of samples per run	86400
Avg. no. of samples passed the ratio (R) test	23321 (26.99%)
Avg. no. of samples passed the update sample filter	21429 (24.8%)
Max. of the max. interval between successive successful updates (seconds)	31s
Avg. of the avg. clock accuracy	99.95 ns
Max. Of the max. clock difference (ns) with DAC model	100 ns

Discussion

The figures show that the maximum of the maximum clock difference between the slave and the master clock using different percentage of bursty traffic with static packet loads in both directions. In Figure 39, the maximum of the maximum clock difference is about 160 ms when 30% of the network load is bursty traffic. The resultant difference is increased, compared to Figure 41 because the algorithm initializes the DAC model with a large offset value, compared to the anticipated offset value. A similar effect is observed in Figure 40 when 40% of the network load is bursty traffic. In both of these cases, the slave clock is updated with a relatively small saved offset value until the end of the simulation time. Thus, the difference between the saved offset value and the anticipated offset value is accumulated over time. It is worth mentioning that the resultant differences are the worst case scenarios, which do not happen all the time. In both of these cases, it has been observed that the DAC model is initialized with large offset value for 2 simulations runs out of 50 runs. The remaining 48 simulations runs are as expected. However, when the amount of bursty traffic is increased to 50% of the network load, the maximum of the maximum clock difference is about 100 ns. The resultant clock difference implies that the slave clock achieved high accuracy when half of the network load is bursty traffic. From the statistics provided in Table 21, Table 22, and Table 23, as the percentage of bursty traffic increases, the average percentage of samples also increases, which passed both tests. When the network load has 30% bursty traffic, the minimum number of samples passed both the ‘R’ test and the update sample filter are about 22% and about 18% respectively on average. In the case of 50% bursty traffic of the network load, about 27% samples passed the ‘R’ test and about 25% samples are used

directly to update the slave clock. In addition, the maximum interval between successive successful updates is 31 seconds for 50% bursty traffic of the network load, much less compared to the 30% and 40% bursty traffic of the network load. It indicates that the maximum interval between successive successful updates decreases with the increments of the percentage of bursty traffic of the network load. Moreover, the average of the average slave accuracy is 99.95 ns when half of the network load is bursty traffic, much less compared to the 30% and 40% bursty traffic of the network load. It implies that the slave accuracy remains high until end of the simulation time when 50% of the network load is bursty traffic. Finally, we can state that if the amount of bursty traffic is 50% of the applied load, then the slave clock is able to achieve high synchronization accuracy w.r.t the master clock.

References

[1] R. Ratzel, R Greenstreet, “Toward Higher Precision,” *Comm. of the ACM*, Vol. 55, No.10, pp. 38-47, August 27, 2012.

[2] Symmetricom white paper, “The Importance of Network Time Synchronization,” 2009.

[3] 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 September, 2012).

[4] W. Ahmed, “Clock Synchronization: Combining IEEE 1588 and Adaptive Oscillator Correction Method,” Department of Engineering, Electrical and Computer, Carleton University, 2012. [Online], available: http://kunz-pc.sce.carleton.ca/thesis/WaheedThesis.pdf (accessed April 30, 2013).

[5] 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).

[6] W. Zhou, “Time, Frequency Measurement and Control Technology,” *Xidian University Press*, ISBN-10: 75606167202006.

[7] F.Christian, “Probabilistic Clock Synchronization,” *Journal of Distributed Computing*, Vol. 3146-158, pp. 1989.

[8] R. Gusella, and S. Zatti, "The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX4.3 BSD," *IEEE Transactions on Software Engineering*, Vol.15, No. 7, July 1989.

[9] D.L. Mills, "Measured Performance of the Network Time Protocol in the Internet System," Network *Working Group Report*, RFC-1128, University of Delaware, October 1989.

[10] D.L. Mills, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905. June 2010.

[11] Symmetricom white paper, "IEEE 1588 Precise Time Protocol: The New Standard in Time Synchronization," 2009.

[12] A. Karvelas, "Synchronization in Packet-based Mobile Backhaul Networks," Ethernet Academy Article, September, 2009.

[13] J. Ferrant, and S. Ruffuni, "Evolution of the Standards for Packet Network Synchronization," *IEEE Comm. Magazine*, Vol.49, pp.132-138, February, 2011.

[14] A. Magee, "Synchronization in Next Generation Mobile Backhaul Networks," IEEE Comm. Magazine, October, 2010.

[15] Symmetricom white paper, "New Needs for Synchronization Testing in Next Generation Networks," April, 2012.

[16] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, 2002

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

[18] Symmetricom white paper, “The Role of Grandmaster, Boundary and Ordinary Clocks in IEEE 1588 Precision Time Protocol (PTP) for Frequency Synchronization Over Packet Networks,” 2012.

[19] Symmetricom white paper, “Improving Real World Synchronization Accuracy with IEEE-1588 Transparent Clocks,” 2009.

[20] D. Tonks, “IEEE 1588 in Telecommunication Applications,” *IEEE 1588 conference*, September, 2005.

[21] T.Murakami, and Y. Horiuchi, “Improvement of Synchronization Accuracy in IEEE 1588 using a Queuing Estimation Method,” *Int. Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, ISPCS 2009, Brescia, Pages: 1-5, 2009.

[22] Symmetricom white paper, “Using IEEE 1588 Transparent Clocks to Improve System Time Synchronization Accuracy,” November, 2009.

[23] S. Kumar, and M. Kumar, “Synchronization in Packet Switched Networks: A survey of Existing Techniques,” Tejas Networks white paper.

[24] L. Xie, Y. Wu, and J. Wang, “Efficient Time Synchronization of 1588v2 Technology in Packet Network,” *Comm. Software and Networks (ICCSN),* May 2011.

[25] Network Simulator (NS-2.34),[Online], available: http://isi.edu/nsnam/ns/

[26] 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 Ultrasonic’s, Ferroelectrics, and Frequency Control*, vol. 58, no. 1, January 2011.

[27] N. M. Freris, S. R. Graham, and P. R. Kumar, "Fundamental Limits on Synchronization of Affine Clocks in Networks," *Automatic Control, IEEE Transactions on*, Vol. 56, pp. 1352 - 1364, June 2011.

[28] M. Nikolas, N.M. Freris, S.R. Graham, and P.R. Kumar, "Fundamental Limits on Synchronizing Clocks over Networks," *Journal of IEEE Transactions on Automatic Control*, Vol. 56, issue: 6, pp. 1352 – 1364, June 2011, USA.

[29] L. Xie, Y. Wu, and J. Wang, "Efficient Time Synchronization of IEEE 1588v2 Technology in Packet Network," *Comm. Software and Networks (ICCSN)*, *IEEE* 3rd Int. Conference, May, 2011.

[30] B. Mochizuki, and I. Hadzic, "Improving IEEE 1588v2 Clock Performance through Controlled Packet Departures," *IEEE Communications Letters*, Vol.14, No. 5, pp. 459-461, May 2010.

[31] "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specification – annex 31b, MAC Control PAUSE Operation," IEEE Std. 802.3-2005, pp. 763-772, Dec. 2005.

[32] T. Murakami, Y. Horiuchi, and K. Nishimura, "A packet Filtering Mechanism with a Packet Delay Distribution Estimation Function for IEEE 1588 Time Synchronization in a Congested Network," *Int. IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS*), pp. 114 – 119, September, 2011, Munich.

[33] S. Lv, Y. Lu, and Y. Ji, "An Enhanced IEEE 1588 Time Synchronization for Asymmetric Communication Link in Packet Transport Network," IEEE Comm. Letters, Vol. 14, No.8, pp. 764-766, August 2010.

[34] Z. Du, Y. Lu, and Y. Ji, "An Enhanced End-to-End Transparent Clock Mechanism with a Fixed Delay Ratio," IEEE Comm. Letters, Vol.15, No.8, pp.872-874, August 2011.

[35] S. Lee, S. Lee, and C. Hong, "An Accuracy Enhanced IEEE 1588 Synchronization Protocol for Dynamically Changing and Asymmetric Wireless Links," *IEEE Comm. Letters*, Vol.16, No.2, pp.190-192, February 2012.

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