Bias Estimation in Asymmetric Packet-based Networks

by

MohammadJavad Hajikhani

A Thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Applied Science
in

Electrical and Computer Engineering
Carleton University
Ottawa, Ontario, Canada
August 2013

Abstract

In the context of the IEEE 1588 Precision Time Protocol (PTP), estimating the delay's bias is a problem that appears in both one-way (using transparent devices) or two-way message exchange mechanisms. For estimating the offset via the two-way message exchange mechanism, it is usually being assumed that the expected value of delays in forward and reverse directions are equal with each other. However this is not a realistic assumption for packet-based wide area networks, where delays in down-link and up-link directions may have a significant difference. In this work we propose a solution to estimate the random delay's bias and improve the synchronization accuracy of IEEE 1588. Our method is easy to implement and is compatible with the current version of the protocol. We compared our results with no bias correction and the Boot-strap method. In addition to the improvement in synchronization accuracy, our method allows us to update the slave clock recursively. The proposed method works well even in the presence of large frequency offsets and can also be implemented by using different filters.

Acknowledgments

I would like to express my deepest appreciation to my supervisor, Prof. Thomas Kunz for his supports throughout my masters program. Without his guidance and persistent help this work would not have been possible.

I would also like to express my sincere gratitude to my co-supervisor, Prof. Howard Schwartz whose contribution in stimulating suggestions and encouragement, helped me a great deal.

I would also like to thank Ericsson, which this work was not possible without their supports and fund.

Last but not least, I would like to thank my family who have supported me always in my life. I will be grateful forever for your love.

Table of Contents

List of Acronyms

Acronyms	Definition
IEEE	Institute of Electrical and Electronics Engineers
IRIG	Inter-Range Instrumentation Group
NTP	Network Time Protocol
PTP	Precision Time Protocol
PDV	Packet Delay Variation

List of Symbols

Symbols	Definition
$C(t)$	slave time, when the master time is t
d_{ms}	fixed part of the delay from a master to a slave node
d_{sm}	fixed part of the delay from a slave to a master node
$\theta(n)$	phase offset between a master and slave node at the n_{th} synchronization interval
$f(n)$	frequency offset between a master and slave node at the n_{th} synchronization interval
$\theta_m(n)$	estimated phase offset at the receiver, after the n_{th} synchronization interval
$f_m(n)$	estimated frequency offset at the receiver, after the n_{th} synchronization interval
α, β	shaping and scale parameter of a Gamma distribution respectively

Chapter 1

Introduction

1.1 Overview

Computer clocks in servers, workstations and network devices are not inherently accurate. These devices usually have inexpensive clocks which may drift as much as a second per day, so if we let them to run by their own, after a short amount of time we would not have any meaningful synchronization among the network devices.

Keeping devices synchronized in a network is a crucial task. Many aspects of the network rely on this synchronicity. When the clocks in a network fall out of synchronization many problems happen. Operations for example for automated tasks, network consolidated tasks and interdependent application tasks may fail. Data is lost. Security is compromised and organizations lose their credibility [**?**].

The IEEE 1588 Precision Time Protocol (PTP) specifies a clock synchronization protocol to synchronize nodes in a distributed network [**?**]. The synchronization process consists of two tasks. First it synchronizes the nodes to one common time by estimating their offset with respect to a reference time (phase offset). The second task is to correct the nodes' clock frequency drift relative to a reference frequency (frequency offset). Synchronization happens by periodic exchanging of timing information between nodes in a network and a master node. The master clock periodically

sends its local time to the slave clock as a Sync message. The arrival time of the Sync message is recorded at the slave node. Then, the slave node sends its local time as a Delay-Req signal to the master. The arrival time of the Delay-Req message is recorded by the master node and is sent back as a Delay-Resp message to the slave node [**?**]. The offset in the slave local clock is adjusted based on these collected values. This method works well only under the assumption of symmetrical delays, which means that the down-link (from master to slave) and up-link (from slave to master) delays are equal. Although this assumption is not true in practice, for some applications it is possible to assume that the expected value of delays in the forward and reverse direction are equal with each other. This assumption reduces the synchronization process to an unbiased estimation problem. Unfortunately, in packet-based wide area networks such as metro Ethernet, it happens frequently that delays in one direction are dominant and down-link and up-link delays are not equal even statistically. As a result we deal with a biased estimation problem. In this work we study the effect of this bias in the synchronization accuracy. The objective of this thesis is to estimate the bias and correct it from the slave clock.

1.2 Outline

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

Chapter 2 introduces background information about the IEEE 1588 PTP. It provides a brief comparison between the IEEE 1588 and other network synchronization protocols. then it elaborates on the details of the IEEE 1588 and the synchronization process in this protocol.

Chapter 3 presents an analytic discussion on the sources of inaccuracy in the synchronization process. In this chapter the synchronization process is modeled as a filtering problem. Bias-aware filters are introduced and the effectiveness of these

filters in the context of this synchronization problem is discussed.

Chapter 4 introduces a number of related works that have been presented in order to solve the bias problem. In this chapter the packet filtering techniques and direct estimation methods are introduced as two main approaches for solving the bias problem in the synchronization process. The next section of this chapter discusses about the Packet Delay Variation (PDV) in networks. In this section a proper Probability Distribution Function (PDF) for modeling PDVs in networks is introduced.

Chapter 5 presents our solutions for solving the bias problem.

Chapter 6 presents the simulation results. In this chapter our proposed solutions are evaluated under different circumstances.

Chapter 7 concludes the thesis and propose a number of potential future works.

1.3 Contributions

The contributions of this thesis are as follows:

- Two new methods for estimation and correction of the delay bias are proposed. Our methods are compatible with the current version of IEEE 1588 PTP. These methods allow us to update the slave clock recursively.

- The simulation results show that our proposed solutions can improve the synchronization accuracy of the IEEE 1588 PTP.

- We evaluate the performance of our methods in the presence of frequency offsets. Simulation results show that the proposed methods work well even when there is a large frequency offset between the master and slave clocks.

- If any knowledge about the network traffic load is available at the slave clock, unlike many different methods, our methods are flexible enough to use this knowledge in order to improve the synchronization accuracy.

- We evaluate the performance of our methods when the traffic load changes with time. In reality, traffic load in a network does not usually change very fast, rather it changes gradually. Nevertheless we consider the extreme case that the traffic load changes suddenly and significantly. We present a solution in order to make our methods more adaptable with time changing traffic loads.

Chapter 2

Background Information

In this chapter, we first provide a brief introduction to crystal oscillators, a frequently used source of timing information. Then we define some technical terms in the context of synchronization. Finally, we introduce different network synchronization protocols. Our focus is on the IEEE 1588 PTP and we will signify the importance of this protocol in a distributed network.

2.1 Crystal Oscillators

A crystal oscillator is an electronic circuit device which generates periodic signals by using the mechanical resonance of the piezoelectric material. The frequency of these oscillations is relatively accurate and can be used to keep track of time. Crystal oscillators are usually inexpensive and very small. For these reasons, crystal oscillators are used widely as a heart of clocks in many devices. As a result the performance of the clocks is severely dependent on the crystal oscillator which is used in them.

2.2 Oscillator Stability and Accuracy

Crystal oscillators are like the heart of a clock. Any inaccuracy or instability of an oscillator cause problems for the clock. The offset of an oscillator from a target frequency is a measure of the oscillator's accuracy. The inaccuracy of clocks in a network causes that, over a period of time, their clocks drift with respect to the reference time (for example a reference GPS clock). Unfortunately this drift is not avoidable and inaccuracy happens in clock oscillators due to reasons such as temperature, ageing[1] and retrace[2].

Variations in the frequency of a crystal oscillator around its natural frequency over a period of time is measured as the frequency stability of a crystal oscillator. Noise is the most important factor which causes instability. The source of this noise can be in the reference signal (if the clock is using any reference signal) or in the electronic circuit itself. In Fig. 1 a comparison between frequency accuracy and stability is shown [**?**].

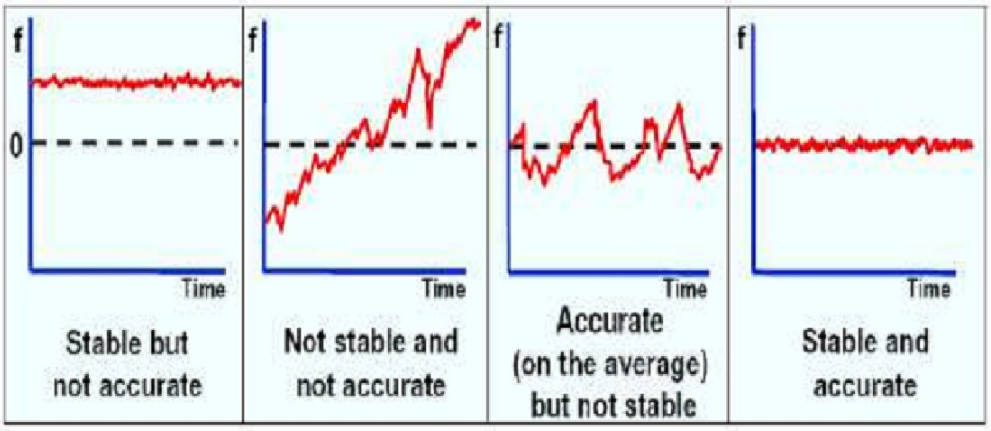

Figure 1: Accuracy and stability of a clock oscillator [3]

[1]The crystal resonator frequency changes according to the operational time and this phenomenon is called ageing [**?**]

[2]When power is removed from an oscillator for several hours, and then reapplied again, the frequency of this oscillator stabilizes at a slightly different value. this frequency variation error is called retrace error [**?**]

2.3 Network Synchronization Protocols

By increasing the data rate in distributed networks, there is a growing need to use more accurate synchronization protocols. Until relatively recent, Network Time Protocol (NTP) and IRIG time code have been the most common network synchronization protocols. The more recently developed IEEE 1588 Precision Time Protocol (PTP) introduced a new synchronization protocol that is able to increase the synchronization accuracy significantly.

NTP has been one of the most popular synchronization protocols. This protocol works well over LANs and WANs and is relatively inexpensive to implement [**?**]. NTP has evolved over the last thirty years as documented in RFC 5905 [**?**]. The accuracy expectations of this protocol vary depending on the motherboard oscillator but the protocol should be able to achieve a clock synchronization accuracy of 1-2 milliseconds on LAN or 1-20 milliseconds on a WAN [**?**]. The IRIG has improved this accuracy to 1-10 microseconds, but this improved accuracy is achieved by adding coaxial timing cables between the clocks. This means more expenses and also more physical infrastructure of the facility [**?**]. As a result IRIG is used just for some special purposes such as military, aerospace or power system devices. IEEE 1588 PTP on the other hand improves the accuracy of IRIG and also has the cost effectiveness of NTP. This standard specifies a clock synchronization protocol to synchronize nodes in a distributed network. PTP is a packet-based timing mechanism. This protocol improves the accuracy by hardware time stamping. PTP is applicable to packet-based networks using regular hubs and switches. In a LAN Ethernet network, it can provide synchronization accuracy to the sub microseconds and with the addition of IEEE 1588 boundary clocks or transparent devices 20-100 nanoseconds synchronization accuracy can be achieved [**?**]. In the next section we will study the IEEE 1588 PTP in more detail.

2.4 IEEE 1588 Precision Time Protocol

The first version of the PTP was introduced in 2002 as an IEEE 1588-2002 standard entitled "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems" [**?**]. In 2008 a revised standard, IEEE 1588-2008 was released which is also known as PTP Version 2 [**?**]. PTP is applicable to systems communicating by local area networks supporting multi-cast messaging including, but not limited to, Ethernet [**?**]. This protocol is designed to synchronize clocks across packet-based networks and due to the remarkable interest in using packet-based timing mechanism in industry [**?**], there is a growing interest in this protocol. Before proceeding further, in the next section we define a number of technical terms from [**?**] that are necessary for our work.

2.4.1 Definitions

accuracy: The mean of the time or frequency error between the clock under test and a perfect reference clock.

stability: A measure of how the mean of the time or frequency error varies with respect to variables such as time, temperature, etc.

precision: A measure of the deviation of the error from the mean.

domain: A number of clocks that are synchronized with each other, but are not necessarily synchronized with other clocks in another domain.

event PTP message: Time-critical messages that require an accurate time stamping in the transmission or reception.

general PTP message: PTP messages whose their information is important but that do not need accurate time stamping.

grand master: the main source of time for clock synchronization.

master clock: A clock to which all the other clocks in a domain synchronize.

phase offset: The difference between a slave local time and its master time.

frequency offset: The difference between a slave clock frequency and its master clock frequency.

2.4.2 PTP message classes

Sync: An event message which is sent from a master to its slave periodically and contains the local time of the master.

Follow_up: To improve the accuracy, the exact time that a Sync message is sent is measured and a follow_up message with the exact time information is immediately sent to a slave. Follow_up is a general message.

Delay_Req: An event message that is sent from a slave to its master and contains the local time of the slave.

Delay_Resp: An event message that is sent from a master to a slave, and contains the reception time of the Delay_Req signal.

2.4.3 PTP devices

Ordinary clock: A clock that has a single PTP port and in a domain can act as either a master clock or as a slave clock.

Boundary Clock: A clock with multiple ports and that can serve as a master or gets synchronized with another clock (acts as a slave clock).

End-to-end transparent clock: These devices forward all the messages just like a normal switch, router, or repeater. However, for a PTP event message, they measure the residence time of the packet, which is the time that it takes for the packet to traverse inside the transparent clock. This measured value is put inside the PTP message.

Peer-to-peer transparent clock: Peer-to-peer transparent clocks differ from the end-to-end transparent clock in the way that they provide corrections for the link propagation delay.

2.4.4 PTP systems

A PTP system is a distributed, networked system, consisting of a combination of PTP and non-PTP devices [**?**]. Ordinary clocks, boundary clocks, end-to-end transparent clocks and peer-to-peer transparent clocks are considered as PTP devices and bridges, routers and devices such as computers, printers, etc are examples of non-PTP devices.

This protocol is a distributed protocol that organizes the system into a master-slave hierarchy and the grand master is the time reference for the entire system. The synchronization is achieved by exchanging timing messages between clocks. This protocol divides the network into logical scopes called domains and all the synchronization process is implemented independently for each of these domains.

2.4.5 Communication path asymmetry

PTP requires that the transmission time of certain messages be measured between the master and slave clock and between the slave and master clock [**?**]. Typically,

these times are not equal. The PTP characterizes the transmission time as:

- mean Path Delay
- delay Asymmetry

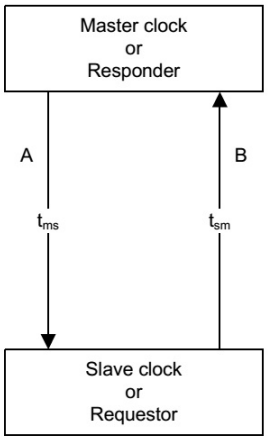

Figure 2: Propagation asymmetry [2]

The *mean Path Delay* is computed as:

$$mean\ Path\ Delay = \frac{t_{ms} + t_{sm}}{2} \tag{2.4.1}$$

where t_{ms} and t_{sm} are the time that a packet takes to travel from the master to the slave and from the slave to the master respectively. The value of delay asymmetry should be known in order to compute the actual transmission time. The protocol does not specify a method for measuring the delay asymmetry. However, if the delay asymmetry is known, then the transmission time can be computed as follow:

$$t_{ms} = mean\ Path\ Delay + delayAsymmetry \tag{2.4.2}$$

$$t_{sm} = mean\ Path\ Delay - delayAsymmetry \tag{2.4.3}$$

2.4.6 Synchronization process

As it has already been mentioned, synchronization is implemented by periodic exchanging of timing information between a master node as a reference, and a slave node which tries to get synchronized with the master. The message exchange process is as follow:

a) The master sends a Sync message which includes its local time to the slave at time t_1.

b) In order to mitigate the operating system latency, the exact time that the Sync is sent is marked, and a Follow_up message with the exact time information is sent immediately [**?**].

c) The slave sends its local time as a Delay_Req message and notes it as t_3.

d) The master receives the Delay_Req at t_4, and sends a Delay_Resp to the slave, containing the value of t_4.

Fig. 3 shows the above process.

At the end of each synchronization interval, the slave node collects a set of four numbers. The mean path delay and the phase offset between the master and slave can be computed as follows [**?**]:

$$mean\ Path\ Delay = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \tag{2.4.4}$$

$$Phase\ Offset = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \tag{2.4.5}$$

The protocol does not specify how these values should be used. For example, the calculated phase offset can be subtracted directly from the slave clock, or can be passed through a filter to eliminate noise. In the later case the output of the filter can be used to correct the slave clock. The effectiveness of these equations to synchronize

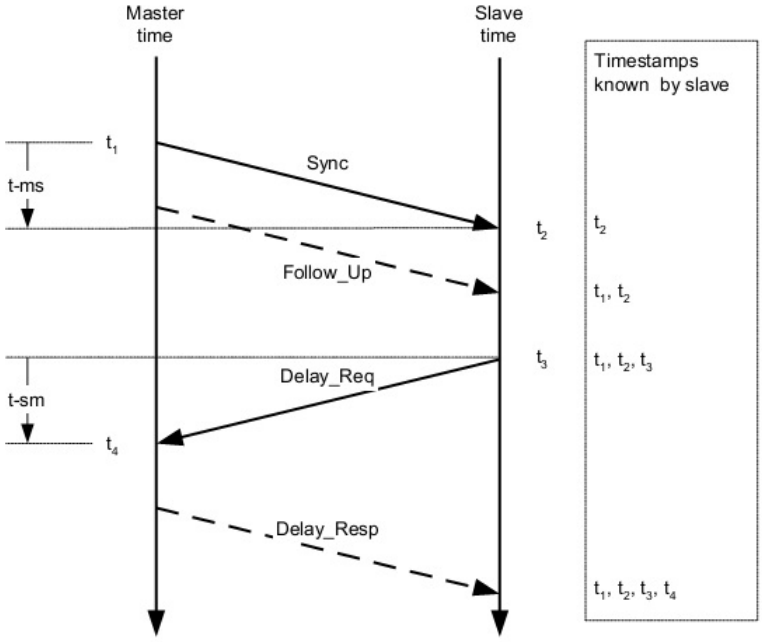

Figure 3: Basic synchronization message exchange [2]

a slave clock is a function of several factors. One factor is the frequency offset between the master and slave clocks. Although this problem can degrade the results, frequency offset can usually be estimated and compensated separately. The other problem arises from asymmetric delays between the master and slave. The accuracy of (2.4.4) and (2.4.5) is immensely dependent on the assumption that the delay from the master to the slave is the same as the delay from the slave to the master. But this assumption, in general, cannot be justified, neither in a deterministic nor a statistical view. A more detailed discussion about this problem is presented in the next chapter.

Chapter 3

Problem Statement

In this chapter, we first present an analytic discussion on the sources of inaccuracy in a synchronization process. Then we explain bias-aware filters as a natural approach to this problem and we will discuss on the effectiveness of these filters for this synchronization problem.

3.1 Sources of inaccuracy

When the slave and master clocks do not agree with each other, the slave clock can be considered as an unknown function of the master clock. So when the master clock shows a time like t, the slave clock time would be $C(t)$. By recalling from Section 2.4.6 of the last chapter and noticing that the synchronization process happens in a periodic way, the four collected times after the nth synchronization interval can be written as [**?**]:

$$t_1 = t_n$$

$$t_2 = C(t_n + d_{ms} + x_n)$$

$$t_3 = C(t'_n - d_{sm} - y_n)$$

$$t_4 = t'_n$$

where $d_{ms} + x_n$ and $d_{sm} + y_n$ represent the time it takes for the Sync message to travel from the master to the slave and the time it takes for the Delay_Req message to travel from the slave to the master, respectively. The terms d_{ms} and d_{sm} are the fixed parts and x_n and y_n are the random parts of the down-link and up-link delays respectively. From Section 7.4.2 of [**?**], messages from master to slave and slave to master traverse the same path in order to minimize the asymmetry. As a result it can be assumed that the fixed parts of the delays in the down-link and up-link directions are equal. This is one of the assumptions of this work. If for any reason the fixed parts of the delays in the down-link and up-link directions differ significantly, then we deal with an additional problem which is out of the scope of this work. We replace d_{ms} and d_{sm} with the term d. By substituting the above expressions in (2.4.5) we have:

$$\theta_m(n) = \frac{[C(t_n + d + x_n) - t_n] - [t'_n - C(t'_n - d - y_n)]}{2} \tag{3.1.1}$$

where $\theta_m(n)$ is the computed phase offset at the slave side. Master and slave clocks are perfectly synchronized if $C(t) = t$, but if we assume that the phase offset at the n_{th} synchronization interval is $\theta(n)$, we would have:

$$C(t_n) = t_n + \theta(n)$$

For short enough intervals (the size depends on how large the frequency offset is) it is possible to assume that the phase offset is unchanged, so:

$$C(t_n + \Delta) = t_n + \Delta + \theta(n)$$

Consequently, the four collected times can be written as:

$$t_1 = t_n$$

$$t_2 = t_n + d + x_n + \theta(n)$$

$$t_3 = t'_n - d - y_n + \theta(n)$$

$$t_4 = t'_n$$

By substituting the above equations in (3.1.1), we have:

$$\theta_m(n) = \theta(n) + \frac{x_n - y_n}{2} \tag{3.1.2}$$

From (3.1.2) it is clear that (3.1.1) is a good estimation of $\theta(n)$ only if x_n and y_n are equal. Although this assumption is, generally, not true in practice, for some applications it is possible to assume that the expected value of delays in forward and reverse directions are equal with each other. This assumption would reduce the synchronization process to an unbiased estimation problem. Unfortunately, in packet-based wide area networks such as metro Ethernet, it happens frequently that delays in one direction are dominant and down-link and up-link delays are not equal even statistically. Inequality in the expected values of these delays causes a bias in the convergence point of the estimated offset that should be removed.

3.2 Dealing with the frequency offset

The frequency offset can usually be estimated by the following equation [**?**]:

$$f_m(n) = \frac{\theta_m(n) - \theta_m(n-1)}{t_n - t_{n-1}} \tag{3.2.1}$$

The denominator of this equation represents the time difference between two successive synchronization intervals. The calculated value from (3.2.1) can be passed into a filter (for example a Kalman filter) and the output of the filter would be the estimated value of the frequency offset.

One interesting characteristic of (3.2.1) is that because it is the subtraction of two successive estimated phase offsets, it does not face the same bias problem due to asymmetric delays. The reason is that the term related to asymmetric delays in (3.1.2) appears both in $\theta_m(n)$ and $\theta_m(n-1)$, and they statistically cancel each other out. Consequently, we end up with an unbiased estimator of the frequency offset.

3.3 Dealing with the phase offset

Estimating the phase offset is a more difficult problem. As it has already been mentioned, the difficulty arises from the inequality in the expected values of the down-link and up-link delays. As a result of this inequality, when the calculated values from (3.1.1) are passed into a filter, the output of the filter would converge to a wrong point. The convergence point is off of the exact value by an error equal to $\frac{E(x)-E(y)}{2}$. Here $E(x)$ and $E(y)$ represent the expected values of the down-link and up-link delays respectively. To obtain a better insight into these delays, in the next few paragraphs, we introduce three major sources of delay in networks.

Delays from a master to a slave and from a slave to a master are random variables that should be modeled by a Probability Distribution Function (PDF). The changing values of the end to end delays from one packet to another usually is referred to as the Packet Delay Variation (PDV). The delay asymmetry in the synchronization process mainly comes from packet delay variations that occur in the networks. Delay and the

related delay variations can be divided into three different components [**?**].

Transmission delay: The transmission delay is the time that it takes for a signal to pass between two end points of a transmission link. This delay is a function of the velocity of the signal and the physical length of the link. Variations in this part of the delay may happen due to any change in the physical characteristics of the transmission link, like a change in the length of the wire or other reasons that may affect the speed of signal. The PDV related to the transmission delay is in the range of sub-microseconds [**?**].

Processing delay: This delay is due to the packet processing time in the network elements and is dependent on the network hardware and software. The PDV related to the processing time is in the order of 1-10 microseconds [**?**].

Queuing delay: The buffering or queuing delay is the total time that packets need to wait in different network elements before getting processed. This time is dependent on the network traffic load and also the priority of the packets. The queuing delay by far is the most important source of PDV and asymmetric delays in networks. The PDV resulting from queuing delays can be in the order of 10-10000 microseconds [**?**].

The total delay is the summation of all these three delays. In packet-based networks, specially due to different traffic loads in the down-link and up-link directions and consequently, different queuing delays, PDVs in these two directions might have a significant difference.

As it has already been mentioned, the calculated value from (3.1.1) can be passed into a filter and the output of the filter would be the estimated value of the phase offset. There are some standard methods for dealing with bias using filters, known as bias-aware filters. To study the effectiveness of these methods, we implement bias-aware Kalman filters as an example. This example will illustrate that the bias-aware

filters cannot provide a solution to the bias estimation problem.

3.4 Kalman filter

A **stochastic time-variant linear system** is described by the following equations [**?**]:

$$x_k = A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + w_{k-1} \tag{3.4.1}$$

$$z_k = H_k x_k + v_k \tag{3.4.2}$$

These two equations are called the **state** and **measurement** equations respectively. Dimension and description of the parameters are as follows [**?**]:

x_k $n \times 1$ $-$ *State vector*

u_k $l \times 1$ $-$ *Input/control vector*

w_k $n \times 1$ $-$ *Process noise*

z_k $m \times 1$ $-$ *Observation vector*

v_k $m \times 1$ $-$ *Measurement noise vector*

A_k $n \times n$ $-$ *State transition matrix*

B_k $n \times 1$ $-$ *Input/control matrix*

H_k $m \times n$ $-$ *Observation matrix*

Q_k $n \times n$ $-$ *Process noise covariance matrix*

R_k $m \times m$ $-$ *Measurement noise covariance matrix*

w_k and v_k captures uncertainties in the model and it is assumed that they are zero mean with known covariance matrices and also are uncorrelated, so:

$E[w_k] = 0 \quad E[w_k w_k^T] = 0 \quad E[w_k w_j^T] = 0 \; for \; k \neq j$

$E[v_k] = 0 \quad E[v_k v_k^T] = 0 \quad E[v_k v_j^T] = 0 \; for \; k \neq j$

and $E[w_k v_j^T] = 0$ for all values of k and j

Note that although in the basic form of a Kalman filter it is assumed that the mean value of the noises are zero, this assumption is modified for bias-aware Kalman filters which will be discussed in the next section.

The terms z_k, u_k, A_k, B_k and H_k are known parameters at the receiver. The purpose of the filtering problem is to estimate the state vector (x_k) from the observation vector (z_k).

In 1960 Kalman introduced the optimal estimator for this problem, under the assumption that the process and measurement noises follow the Gaussian distribution. A Kalman filter works in two steps: the model forecast step, also known as predictor, and the data assimilation step, also called corrector. In the Kalman filtering process, two other matrices appear. The P matrix is a measurement of the error covariance (between the estimated value and the actual state vector), and the K matrix is called the Kalman weight or Kalman gain. The process starts with an initial assumption on the value of P and the state vector. The process continues as follows:

Predictor:

$$x_k^f = A_{k-1} x_{k-1}^a + B_{k-1} u_{k-1} \tag{3.4.3}$$

$$P_k^f = A_{k-1} P_{k-1} A_{k-1}^T + Q_{k-1} \tag{3.4.4}$$

Corrector:

$$K_k = P_k^f H_k^T (H_k P_k^f H_k^T + R_k)^{-1} \tag{3.4.5}$$

$$x_k^a = x_k^f + K_k (z_k - H_k x_k^f) \tag{3.4.6}$$

$$P_k = (I - K_k H_k) P_k^f \tag{3.4.7}$$

The terms x_0^a, P_0 are used to start the process and allow us to evaluate (3.4.3) and (3.4.4) initially. By the reception of z_k, the value of x_k^as, P_k and K_k are updated, and

then the values of x_k^f andP_k^f from (3.4.3) and (3.4.4) are readjusted. At each iteration, x_k^a is our estimation of the state vector. The following block diagram from [**?**] shows the whole process:

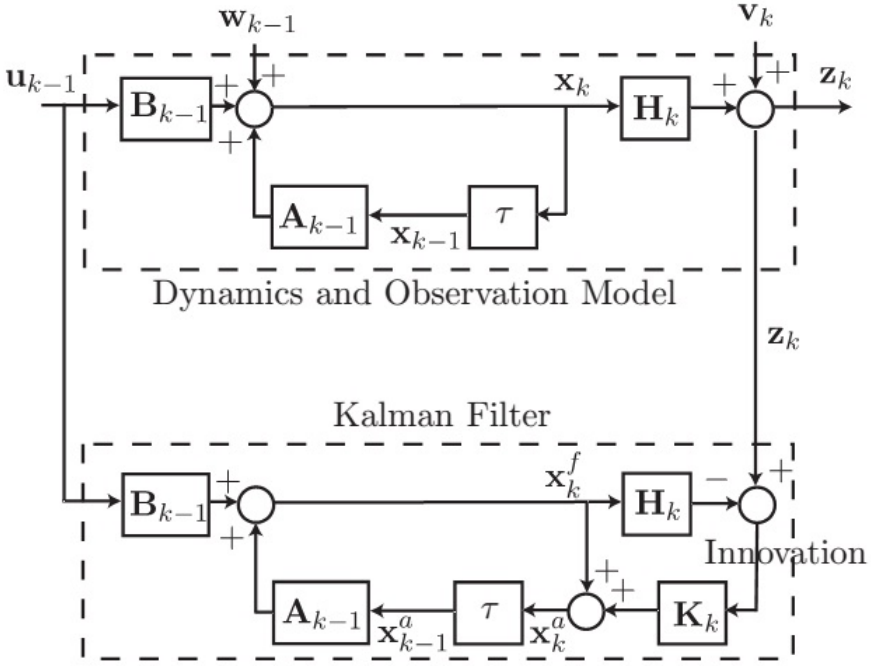

Figure 4: The block diagram for a Kalman filter [10]

In the next section, we introduce a modified version of this filter, known as the bias-aware Kalman filters.

3.4.1 Bias-aware Kalman filters

In general, the mean value of the process and measurement noises may not be zero. This non zero mean value can be considered as a bias term in the state and measurement equations. For this purpose, these equations should be modified as follows:

$$x_k = A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + C_{k-1}b + w_{k-1} \tag{3.4.8}$$

$$z_k = H_k x_k + D_{k-1} b + v_k \tag{3.4.9}$$

Here b is a bias that appears in the state and measurement equations. In [**?**], a simple solution to this problem has been introduced. For removing the effect of bias from the equations, we can simply change the state vector as follow:

$$\begin{bmatrix} x_k \\ b \end{bmatrix} = \begin{bmatrix} A_{k-1} & C_{k-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ b \end{bmatrix} + B_{k-1} u_{k-1} + w_{k-1}$$

and:

$$z_k = \begin{bmatrix} H_k & D_k \end{bmatrix} \begin{bmatrix} x_k \\ b \end{bmatrix} + v_k$$

so the new state and measurement equations can be written as :

$$x'_k = A'_{k-1} x'_{k-1} + B_{k-1} u_{k-1} + w_{k-1} \tag{3.4.10}$$

$$z_k = H'_k x'_k + v_k \tag{3.4.11}$$

where:

$$x'_k = \begin{bmatrix} x_k \\ b \end{bmatrix} \quad , A'_{k-1} = \begin{bmatrix} A_{k-1} & C_{k-1} \\ 0 & 1 \end{bmatrix} \quad \text{,and} \quad H'_k = \begin{bmatrix} H_k & D_k \end{bmatrix}$$

And an algorithm identical to the original Kalman filter can be followed to deal

with the above equations. To examine this method, we provide a numerical example. Suppose that the state and measurement equations are:

$$\begin{cases} x_k = Ax_{k-1} + w_{k-1} \\ z_k = x_k + b + v_k \end{cases}$$

w_k and v_k, are zero mean Gaussian random variables with variances of 0.001 and 0.01 respectively. The value of the bias term b is assumed to be 10. The initial state variable, x_0, is assumed to be 1. We evaluate the effectiveness of this method for different values of A.

First let us assume that $A = 4$. Fig. 5 shows the error in the estimated value of the state variable, versus the number of iterations. So if at the k_{th} iteration, we denote the estimated value by x_k^a, the error is defined to be $x_k - x_k^a$.

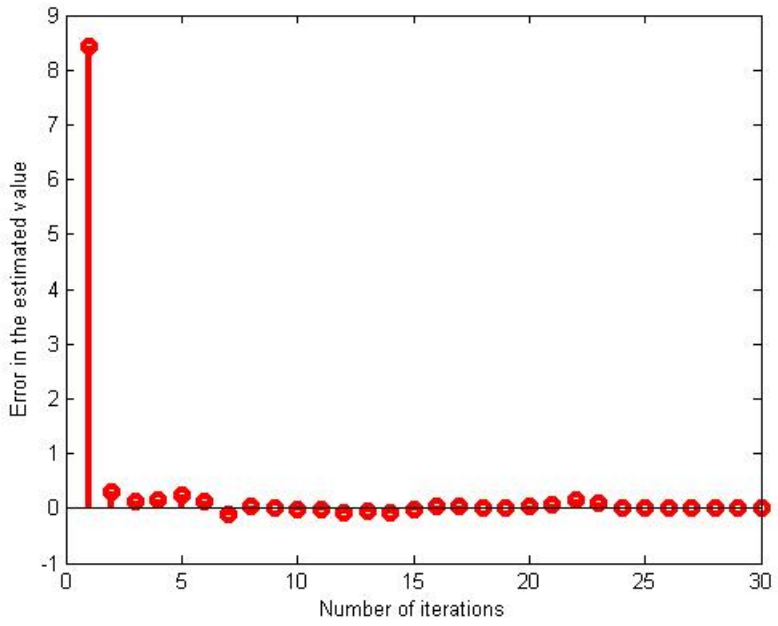

Figure 5: Kalman filter's error for A=4

Fig. 5 shows that after a few iterations, the output of the Kalman filter can provide a good estimation of the state variable. Note that after a number of iterations, the

output neither gets better nor worse, but fluctuates with an amplitude which is a function of the random properties of the process and measurement noises.

Fig. 6 shows the results for $A = 2$, $A = 1.1$ and $A = 1$. It can be observed that as the value of A gets closer to one, the speed of convergence gets slower. Finally, for $A = 1$, convergence cannot be achieved.

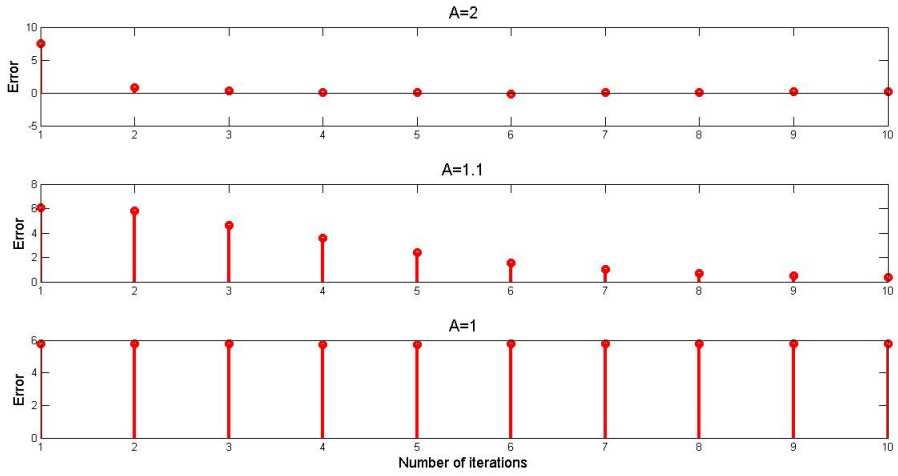

Figure 6: Kalman filter's error for different values of A

To understand why convergence cannot be achieved for A=1, we examine the state and measurement equations in our example. For simplicity, assume that the process noise is zero. The equations reduce to:

$$\begin{cases} x_k = x_{k-1} \\ z_k = x_k + b + v_k \end{cases}$$

As can be seen, the state variable is always a constant number, and the measured value is the summation of the state variable, the bias, and a noise, which makes it impossible to distinguish between the bias and the state variable.

3.4.2 Modeling the synchronization process as a filtering problem

The synchronization process which was described in Sections 3.1 and 3.2 can be modeled as a filtering problem. We consider $u_\theta(n)$ and $u_f(n)$ as the corrections that we apply to the phase and frequency of the slave clock respectively at the nth synchronization interval. Then the new phase and frequency offsets at the $(n+1)th$ synchronization interval (before the next correction), can be written as follow [**?**]:

$$\theta(n) = \theta(n-1) - u_\theta(n-1) + [f(n-1) - u_f(n-1)].\Delta T + \omega_\theta(n-1) \qquad (3.4.12)$$

and

$$f(n) = f(n-1) - u_f(n-1) + \omega_f(n-1) \qquad (3.4.13)$$

The terms $\omega_\theta(n)$ and $\omega_f(n)$ are the process noises for the phase offset and frequency offset respectively. The terms $u_\theta(n)$ and $u_f(n)$ are the estimated values for the phase and frequency offsets. ΔT is the time difference between two successive synchronization intervals and is the same as the denominator of (3.2.1). The above equation means that the next phase offset would be the previous one, minus the last estimated phase offset, plus the drift that happens between two successive corrections, plus a process noise. For the frequency offset, the next offset is the previous one, minus the last estimated frequency offset plus a process noise. We can combine (3.4.12) and (3.4.13) into one state equation:

$$\begin{bmatrix} \theta(n) \\ f(n) \end{bmatrix} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta(n-1) \\ f(n-1) \end{bmatrix} + \begin{bmatrix} -1 & -\Delta T \\ 0 & -1 \end{bmatrix} \begin{bmatrix} u_\theta(n-1) \\ u_f(n-1) \end{bmatrix} + \begin{bmatrix} \omega_\theta(n-1) \\ \omega_f(n-1) \end{bmatrix}$$

so:

$$\underline{x}(n) = A\underline{x}(n-1) + B\underline{u}(n-1) + \underline{w}(n-1) \tag{3.4.14}$$

The measurement equation can be written directly from (3.1.1) and (3.2.1). In fact each estimation can be written as a summation of the actual offset plus a measurement noise, so:

$\theta_m(n) = \theta(n) + \nu_{\theta_m(n)}$

$f_m(n) = f(n) + \nu_{f_m(n)}$

Here $\nu_{\theta_m(n)}$ and $\nu_{f_m(n)}$ are the measurement noises for the phase offset and the frequency offset respectively. The measurement noise for the phase offset from (3.1.2) is determined by $\frac{x_n - y_n}{2}$. As we already explained, the mean value of the down-link and up-link delays are not necessarily equal and as a result the mean value of the measurement noise for the phase offset in general is not zero. In contrast, since in the numerator of (3.2.1) there is a subtraction of two successive phase offsets, in terms of the bias, they cancel each other out and therefore (3.2.1) provides an unbiased estimator for the frequency offset. By rewriting the above equation in matrix form:

$$\begin{bmatrix} \theta_m(n) \\ f_m(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta(n) \\ f(n) \end{bmatrix} + \begin{bmatrix} \nu_{\theta_m(n)} \\ \nu_{f_m(n)} \end{bmatrix}$$

Finally, the measurement equation can be written as:

$$\underline{z}(n) = H\underline{x}(n) + \underline{v}(n) \tag{3.4.15}$$

This filtering problem can be solved by different filters, including but not limited to the Kalman filter. A summary of this filtering mechanism is as follow. At the n_{th} synchronization interval, the slave node receives $\underline{z}(n)$. This vector is passed into

the filter. The output of the filter is the vector $\underline{u}(n)$. The vector $\underline{u}(n)$ is subtracted from the slave clock in order to correct the phase and frequency offsets. Using a filter in this way to estimate and correct the offset, in the simulations section, is referred to as *Basic 1588*. The non zero mean value of the measurement noise for the phase offset creates a bias in the estimated phase offset by the filter. Using bias-aware filters like the one we studied in the last section may seem as a solution for estimating the bias. However, these methods cannot help us here. To see the reason, we simplify the filtering problem as follow. Let us assume that the process noise and frequency offset are zero. We also assume that $u_\theta = 0$. As it has already been explained in Section 3.1, when the expected value of delays in up-link and down-link directions are not equal to each other, there would be a bias in the estimated value of the phase offset. This bias is hidden in the mean value of the measurement noise term in (3.4.15). We can rewrite it as a summation of a bias, plus a zero mean measurement noise. In that case, the state and measurement equations for the phase offset would be:

$\theta(n) = \theta(n-1)$

$\theta_m(n) = \theta(n) + b + \nu'_{\theta_m(n)}$

We observe that the state and measurement equations are exactly the same as the example that we discussed in the last section, for the case that $A = 1$. As we showed for that example, estimating the bias and state variable just by using these equations is not possible. These equations have been imposed on us by the nature of the problem and are not avoidable. It means that for the purpose of synchronization, we need to find another solution to estimate the bias.

In the next chapter, we will review the literature with regard to proposed solutions for improving the accuracy of this synchronization protocol with a focus on methods for dealing with bias.

Chapter 4

Related Work

There are basically two main approaches for solving the bias problem in the synchronization process. The first approach is by packet filtering techniques. This group of solutions do not try to estimate the bias directly, rather they just try to filter packets in a way to reduce the effect of bias. The second approach to this problem is to estimate the bias directly or to provide an unbiased estimator of the phase offset. The Boot-strap method is an example of this group of solutions. The Boot-strap method provides the order statistic-based best linear unbiased estimator (o-BLUE) of the phase offset under the assumption that the PDVs have exponential or Pareto distributions. In this chapter we will study a number of these methods. Moreover, for developing a good solution to the bias problem, a better knowledge about the distribution of PDVs is very important. As a result, we also will study the statistical characteristics of the PDVs in multi-hop networks.

4.1 Packet filtering techniques

Packet filtering techniques try to find some "good" packets based on a reasonable condition. The definition of a good packet is subjective and varies from one paper to another. In a number of papers such as [**?**] and [**?**], after collecting a number of

received messages (this number is determined by the window size) the good packet is the one with the least delay. Such algorithms are usually referred to as *sample − minimum* filters. The mean value, maximum and mode are other criteria that can be used to filter packets. These criteria also may be used in a hybrid way [**?**].

In order to mitigate deterioration of synchronization accuracy due to PDVs, the authors of [**?**] proposed using a train of probing packets after the Sync and Delay-Req messages. Fig. 7 shows the changes that need to be done in the IEEE 1588 PTP, in order to implement this method.

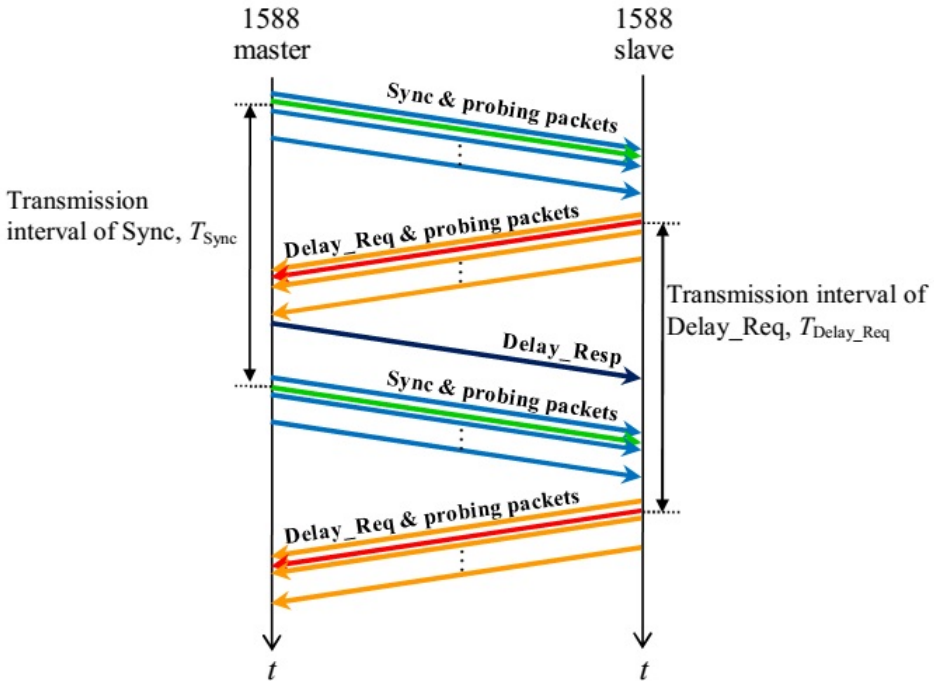

Figure 7: Message sequence between master and slave[15]

The aim of using probing packets is to estimate the occurrence of delay in timing packets and filter out the packets with large value of delays [**?**].

Fig. 8 shows the sequence of timing packets transmitted from the master to the slave. In this sequence of packets, there is one Sync message and n_p probing packets. Here SW stands for switch. The occurrence of large delays can be detected based on

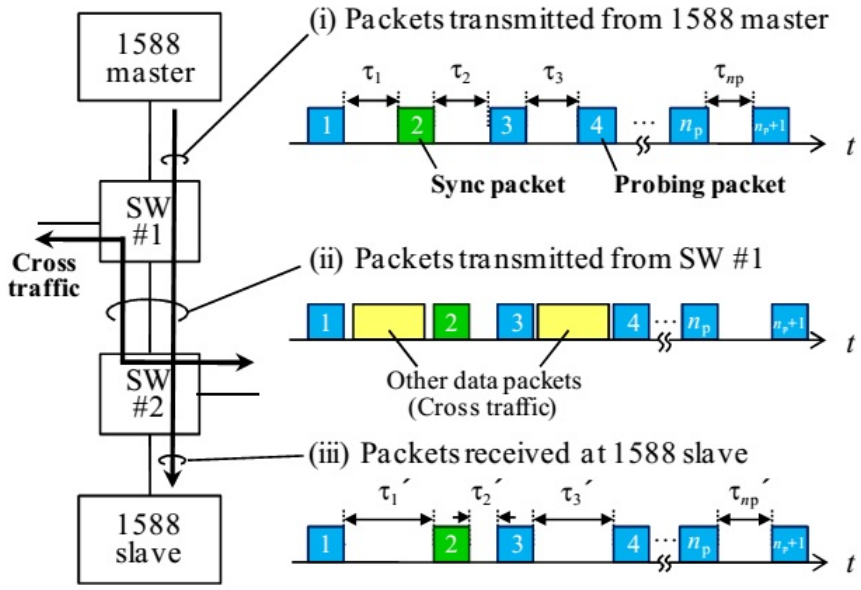

Figure 8: Transmission of Sync and probing packets [15]

the inter-frame gaps between received packets. We denote the gaps between the k_{th} and $(k+1)_{th}$ packets by τ_k. After experiencing the delay in the network, the value of the gap at the receiver will change to τ'_k. The criteria of filtering in [**?**] is defined as:

i) If $(\tau'_{k-1} - \tau_{k-1}) > \Delta\tau$ or $(\tau'_k - \tau_k) < \Delta\tau$, delay occurred in the packet.

ii) Otherwise delay did not occur.

The threshold $\Delta\tau$ should be selected based on the accuracy that is needed at the slave clock. Finally, only the packets that pass this condition can be used for synchronization. As can be observed, there is a trade off between the accuracy of this method and the number of packets that will pass through this filtering condition. There is also a possibility that the delay happens before the very first probing packet. In this case all the transmitted packets would experience this delay in the same way and it is not possible to detect it from the gap between the packets.

Although using packet filtering methods can improve the synchronization accuracy, none of these methods actually estimates the value of bias (or provides and unbiased estimator of the phase offset). One other problem with this group of solutions is that we need to filter a large number of packets. Dropping packets means loosing information and also causes more delay before correction (or alternatively requires more network traffic overhead to collect a window's worth of data within a single correction interval).

4.2 Direct estimation techniques

Direct estimation techniques try to find an estimator for the bias or an unbiased estimator for the phase offset. Using transparent devices is an example of this group of solutions. As it was defined in Chapter 2, End to End transparent clocks are able to measure the residence time of a PTP event message and put this value into a specific field of the message. The measured time can be used at the receiver to calculate the delay of the message. However, due to reasons such as the rate mismatch of the transparent clocks with respect to the master clock, the calculated residence time might be the subject of an unacceptably large error. This effect gets escalated specially over a multi-hop network. Moreover, although using the transparent devices has been predicted in IEEE 1588 v2, it requires changing the old infrastructures of the network with transparent devices which may not be cost-effective.

As it was mentioned, the calculated residence time over a multi-hop network, using the transparent devices, may not be very accurate. The authors of [**?**] proposed a solution to solve this problem. In [**?**], it is assumed that instead of one Sync message, we have two different kinds of Sync messages. They are denoted by Sync 1 and Sync 2. It is also been assumed that transparent devices somehow have the ability to give the Sync 2 message a delay k times larger than Sync 1. Knowing that the delay for

Sync 2 is k times larger than Sync 1 provides an additional independent equation, that can be used for finding the value of the bias and phase offset. Fig. 9 shows the message exchange mechanism in this method.

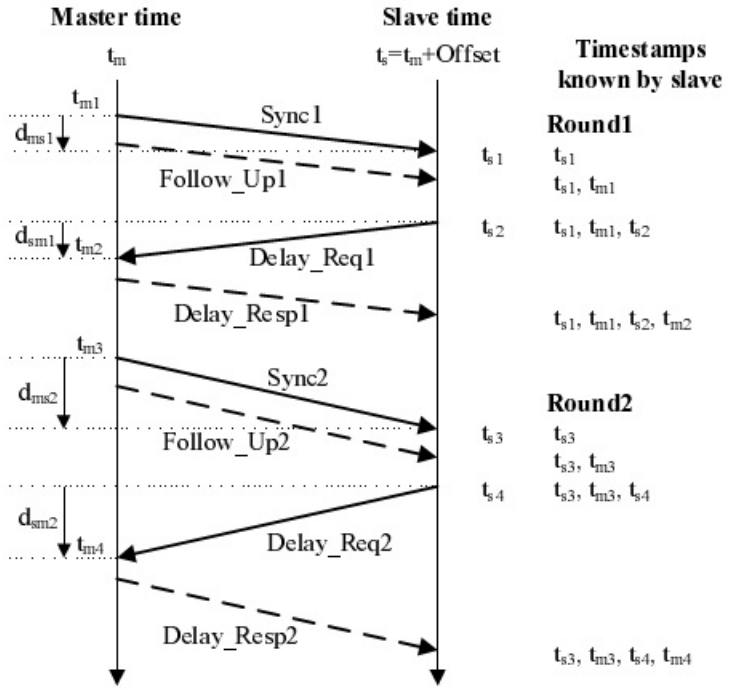

Figure 9: Enhanced PTP messaging timing diagram [16]

This method has two problematic requirements. First, using two different kinds of Sync messages is not compatible with the current version of the IEEE 1588 protocol. Moreover, the current transparent devices do not have this ability to give a delay k times larger to one of the Sync messages with respect to the other one and producing such devices may not be cost-effective. Consequently, these two problems make this method impractical.

In [**?**] and [**?**], Jesk *et al.* used the Boot-strap method to find the order statistic-based best linear unbiased estimator of the clock offset under the assumption that the Packet Delay Variations (PDV) have exponential or Pareto distributions. Following

the notation of [?], let x and y denote the master to slave and slave to master travel times corresponding to a timing message exchange. An iteration of n timing message exchanges yields the random samples $\{x_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$. Let $\{x_{(i)}\}_{i=1}^n$ and $\{y_{(i)}\}_{i=1}^n$ denote the order statistic of these samples. The Boot-strap estimator is from a class of estimators in the form of $\tilde{\theta} = \sum_{i=1}^n w_i[x_{(i)} - y_{(i)}]$ for a suitable choice of $\{w_i\}_{i=1}^n$ which have the property $\sum_{i=1}^n w_i = 1/2$. If we denote the phase offset by θ, $\tilde{\theta}$ is an unbiased estimator of the phase offset. In [?] it was shown that if x and y have an exponential distribution, then the best linear unbiased estimator of the phase offset is:

$$\tilde{\theta} = \frac{1}{2}\left[\frac{nT_1 - T_2/n}{n-1} - \frac{nS_1 - S_2/n}{n-1}\right] \tag{4.2.1}$$

where $(T_1, T_2, S_1, S_2) = [min_{1\le i\le n} x_i, \sum_{i=1}^n x_i, min_{1\le i\le n} y_i, \sum_{i=1}^n y_i]$. However, as will be discussed in the next section, more recent papers such as [?] and [?] show that the Gamma distribution is a better choice for modeling the PDVs in networks. The non-exponential distribution of the PDVs will cause an error in the estimated phase offset. The authors of the Boot-strap method also have not provided any solution for estimation and correction of the frequency offset.

The results of [?] and [?], modeling the network delay by Gamma distributions, is the main motivation and one key assumption of our work. As a result, we study the characteristics of the PDVs in a network in the last remaining section of this chapter. In the next chapter, we will explain how this results can help us to develop a better solution for the bias problem.

4.3 Packet Delay Variation in Networks

In [?] and [?], it was shown that the Packet Delay Variation (PDV) in multi-hop Ethernet networks and for strict priority queuing can be modeled by a Gamma distribution. The parameters of the PDF are a function of the traffic load and also the number of hops. The PDF of a Gamma distribution is:

$$f_X(x) = \begin{cases} \frac{x^{\alpha-1}}{\Gamma(\alpha)\beta^{\alpha}} e^{\frac{-x}{\beta}} & x > 0 \\ 0 & x < 0 \end{cases} \tag{4.3.1}$$

The terms α and β are called the shaping and scale parameters respectively. The exponential distribution is a special case of the Gamma distribution for the case that the shaping parameter is equal to one. As it was already mentioned, in [?], Jesk et al. used a Boot-strap solution to provide an unbiased estimator of the phase offset. This method was developed under the assumption that the PDV can be modeled by an exponential distribution. However, the results of [?] and [?] show that the shaping parameter is not always close to one. For example for 80% traffic load, over a 5 hops network, the shaping parameter in [?] is determined as 11. This difference causes an error in the estimated value of the phase offset for the Boot-strap method.

The experimental results in [?] are collected for the IEEE 1588v2 Sync messages in a setup composed of five carrier-class switches for different traffic loads. The Sync messages were sent at a rate of 16 packets per second in a strict priority queuing class. Fig. 10 shows how by increasing the traffic load from 20% to 80%, packet delay increases.

The authors of [?] and [?] modeled the PDVs by a Gamma distribution [1]. Figures 10 to 13 show the PDVs for different traffic loads and also the Gamma distribution

[1] In [?] and [?] they use the Erlang distribution which is a special case of the Gamma distribution for integer values of the shaping parameter

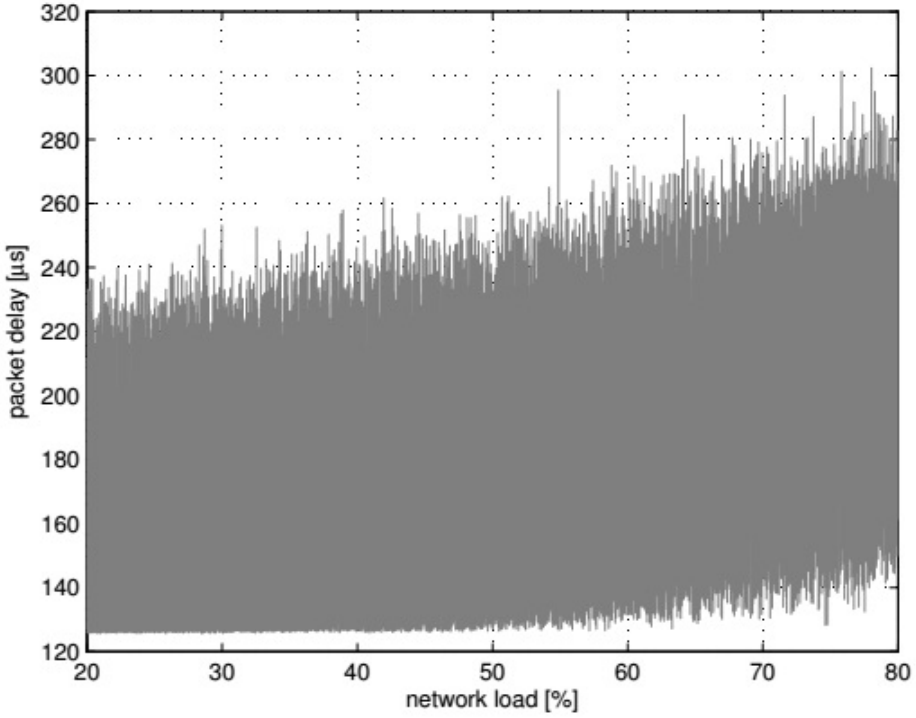

Figure 10: Packet delay for different traffic loads [19]

that fits with the experimental results. Although in [**?**] and [**?**] the distributions have not been defined explicitly, we regenerated the delays in Figures 10 to 13 using the following MATLAB codes:

for 20 percent traffic load:

$delay = 133 * 10^{-6} + 10^{-6} * gamrnd(2, 6.5);$

for 40 percent traffic load:

$delay = 133 * 10^{-6} + 10^{-6} * gamrnd(6, 6.5);$

for 60 percent traffic load:

$delay = 133 * 10^{-6} + 10^{-6} * gamrnd(8, 6.5);$

for 80 percent traffic load:

$delay = 133 * 10^{-6} + 10^{-6} * gamrnd(11, 6.5);$

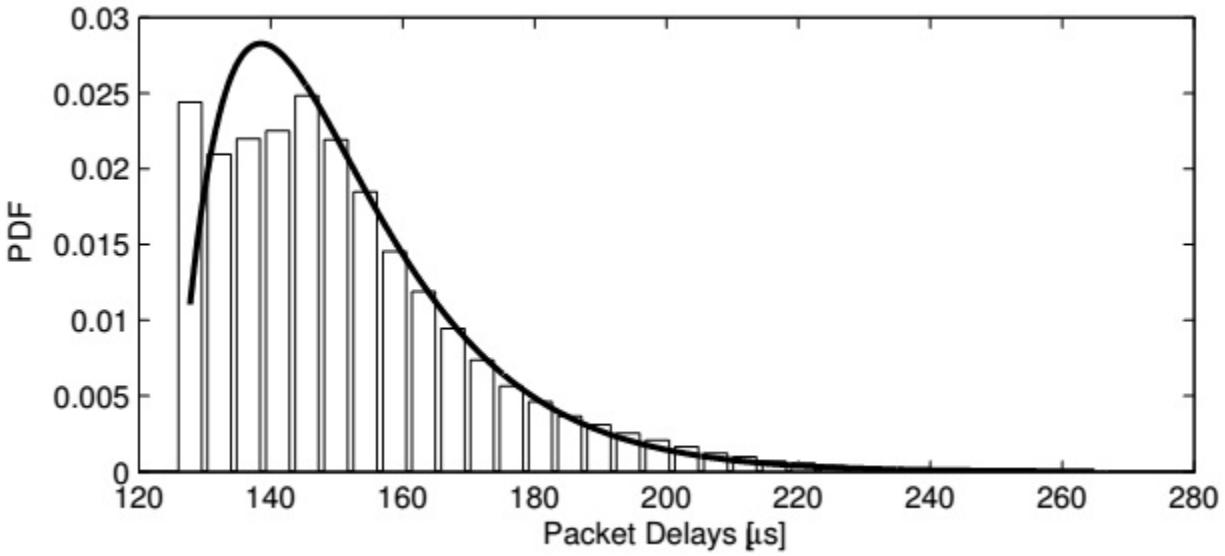

Figure 11: 5 hops 20% traffic load, Gamma($\alpha = 2$) [14]

We observe that PDVs in a network do not follow the exponential distribution and for higher traffic loads, they are not even close to this distribution. As has been illustrated in Figures 11 to 14, as the traffic load increases from 20% to 80%, the value of α increase from 2 to 11.

This result, modeling the network delay by Gamma distributions, is the main motivation and one key assumption of this work. In the next chapter we introduce a solution to estimate the bias and we compare our results with the case with no bias correction and also with the Boot-strap method.

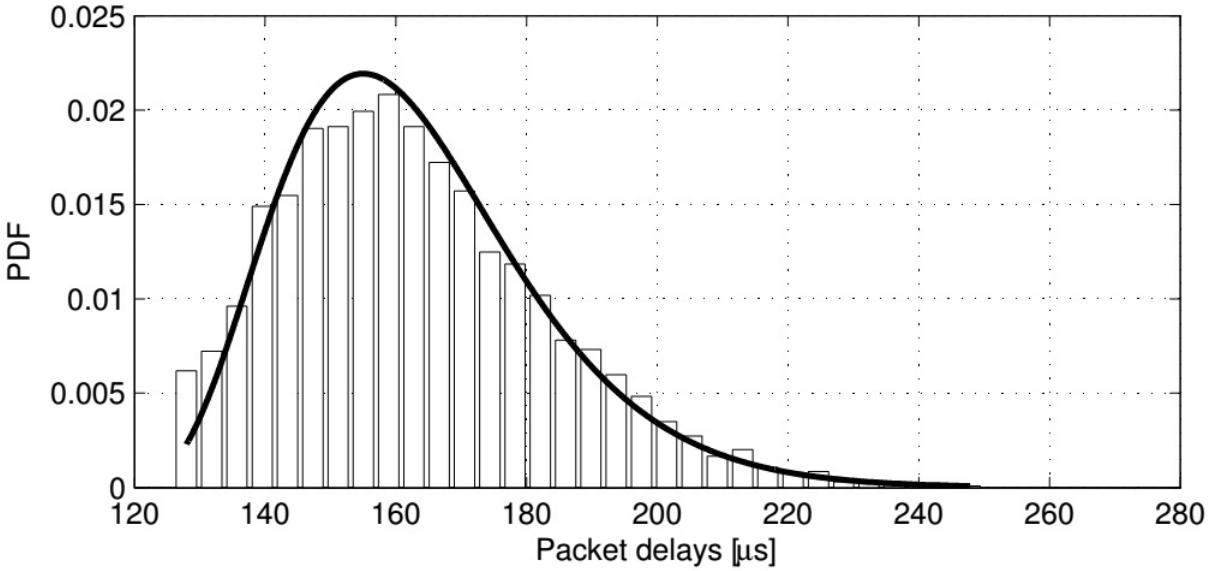

Figure 12: 5 hops 40% traffic load, Gamma($\alpha = 6$) [19]

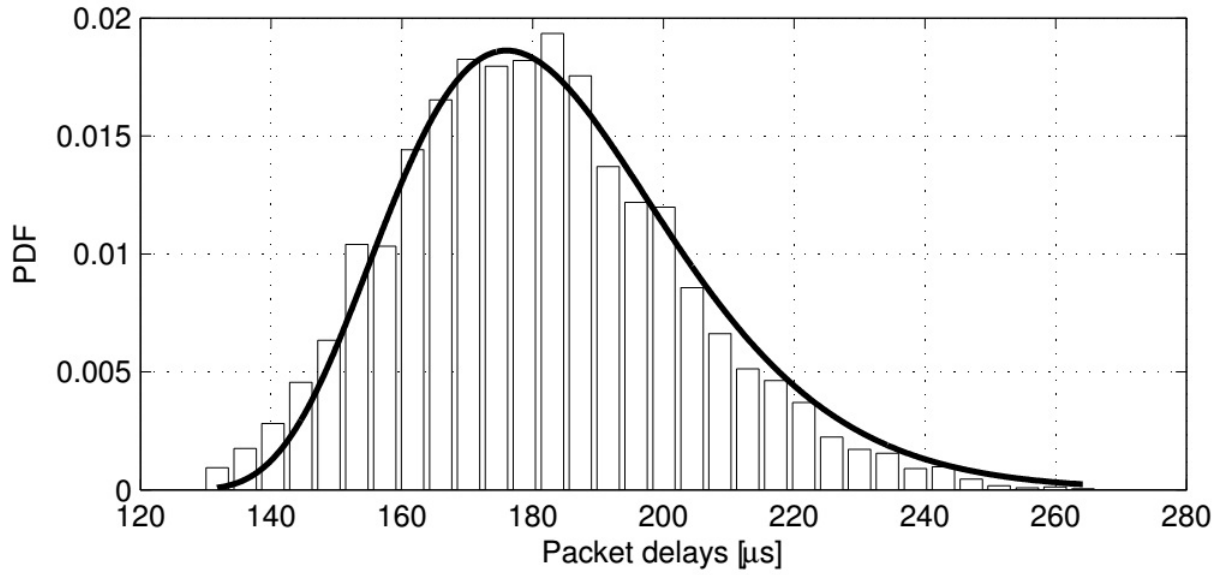

Figure 13: 5 hops 60% traffic load, Gamma($\alpha = 8$) [19]

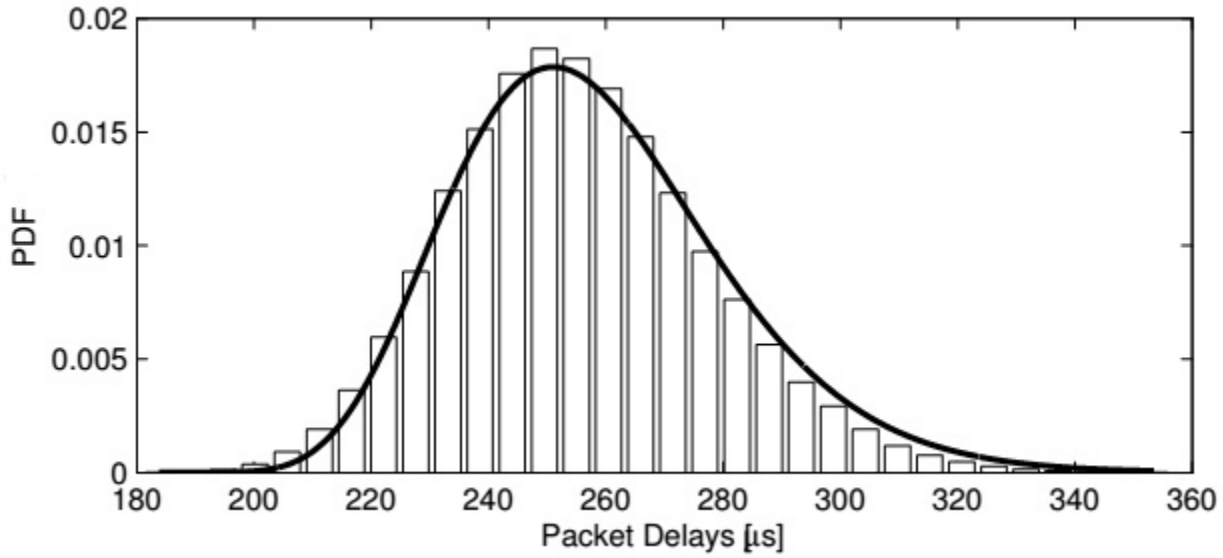

Figure 14: 5 hops 80% traffic load, Gamma($\alpha = 11$) [14]

Chapter 5

Proposed Solution

In this chapter we propose a solution to estimate the delay bias and improve the synchronization accuracy of IEEE 1588 PTP. Our method is easy to implement and is compatible with the current version of the protocol. As discussed in Section **??**, we know that the synchronization process can be modeled as a filtering problem. We refer to the method of using a filter discussed in Section **??** to estimate and correct the offset as, the *Basic 1588* method. In this chapter we introduce a method to remove the bias from the output of the filter. We call our method *Bias Correcting 1588* (BC 1588). This method allows us to update the slave clock recursively. It also works well even in the presence of large frequency offsets and can be implemented via different filters.

5.1 Bias correcting 1588 (BC 1588)

From Section **??** we know that the synchronization process happens by exchanging timing packets between master and slave. A master sends the n_{th} Sync message which includes its local time at t_n. A slave receives this message at $C(t_n + d + x_n)$. The term $C(t)$ represents the local time of the slave clock and $d + x_n$ is the time that a packet takes to travel from master to slave or the down-link delay where d is the fixed part

and x_n is the random part of the delay. Then the slave node sends its local time as a Delay-Req message at $C(t'_n - d - y_n)$ and the master receives this message at t'_n, where $d + y_n$ is the up-link delay and similar to the down-link delay d is the fixed part and y_n is the random part of the delay. Note that following Section 3.1 we are assuming that the fixed parts of the delays for the down-link and up-link directions are equal with each other. For the sake of simplicity, we first assume that the frequency offset is zero. So the phase offset is a constant number. From Section 4.3, we know that the random delays from a master to slave and from a slave to master can be modeled by a Gamma distribution. We denote the subtraction of a master time from the slave time at the n_{th} synchronization interval by $\delta_{DL}(n)$. Similarly, for the up-link direction we denote the subtraction of a slave time from the master time by $\delta_{UP}(n)$. By using the results of Section **??**, for the down-link direction we have:

$$\delta_{DL}(n) = C(t_n + d + x_n) - t_n = \theta + d + x_n \tag{5.1.1}$$

and for the up-link:

$$\delta_{UP}(n) = t'_n - C(t'_n - y_n) = -\theta + d + y_n \tag{5.1.2}$$

The term θ is the phase offset. From Chapter 3 we know that the bias term is determined as $(E(X) - E(Y))/2$. The terms $E(X)$ and $E(Y)$ represent the expected values of the random delays for the down-link and up-link directions respectively. As a result, in order to estimate the bias we need to estimate the expected values of the random delays. Our method estimates the expected value of the down-link and up-link random delays separately and in symmetric fashion, so at this point let us just consider the down-link delay.

If the random variable X represents the random part of the down-link delay, then its PDF is determined by (4.3.1). The CDF of the Gamma distribution is:

$$F_X(x) = \begin{cases} \frac{\gamma(\alpha, \frac{x}{\beta})}{\Gamma(\alpha)} & x > 0 \\ 0 & x < 0 \end{cases} \tag{5.1.3}$$

We recall that α and β are called the shaping and scale parameters respectively. The term $\gamma(\alpha, \frac{x}{\beta})$ is an incomplete Gamma function and is determined by:

$$\gamma(\alpha, \frac{x}{\beta}) = \int_0^{\frac{x}{\beta}} y^{\alpha-1} e^{-y} dy \tag{5.1.4}$$

The expected value of the Gamma distribution is determined by the multiplication of α and β, so:

$$E(X) = \alpha\beta \tag{5.1.5}$$

It needs to be emphasized that the values of α and β are unknown, so we are not able to calculate the expected value of the delay using (5.1.5).

Now assume that instead of one Gamma random variable, we consider the minimum of two successive random variables. In the context of IEEE 1588, this means that when the receiver receives the $(2i-1)_{th}$ Sync message, it waits for the next one, then takes the following minimum:

$$\begin{aligned} \delta'_{DL}(i) &= min[(C(t_{2i-1} + d + x_{2i-1}) - t_{2i-1}), (C(t_{2i} + d + x_{2i}) - t_{2i})] \\ &= min[(\theta + d + x_{2i-1}), (\theta + d + x_{2i})] \\ &= \theta + d + min(x_{2i-1}, x_{2i}) \end{aligned} \tag{5.1.6}$$

Here DL stands for down-link.

It is known from stochastic text books (for example [**?**]) that if $W = min(X, Y)$, where X and Y are independent random variables, then the distribution of W is:

$$f_W(w) = f_X(w) + f_Y(w) - [f_X(w)F_Y(w) + f_Y(w)F_X(w)] \tag{5.1.7}$$

Here f and F represent the PDF and CDF of the random variables respectively. If we assume that X and Y are from a Gamma distribution, then by using (4.3.1) and (5.1.3), the distribution of W is determined by:

$$f_W(w) = \frac{2w^{\alpha-1}e^{\frac{-w}{\beta}}}{\Gamma(\alpha)\beta^{\alpha}}\left[1 - \frac{\gamma(\alpha, \frac{w}{\beta})}{\Gamma(\alpha)}\right]. \tag{5.1.8}$$

For finding the mean value of W, we expand $\gamma(\alpha, \frac{w}{\beta})$ by the following series [**?**]:

$$\gamma(\alpha, \frac{w}{\beta}) = (\frac{w}{\beta})^{\alpha}\Gamma(\alpha)e^{\frac{-w}{\beta}}\sum_{0}^{\infty}\frac{(\frac{w}{\beta})^k}{\Gamma(\alpha+k+1)} \tag{5.1.9}$$

By substituting (5.1.9) in (5.1.8), we can rewrite (5.1.8) as:

$$f_W(w) = \frac{2w^{\alpha-1}e^{\frac{-w}{\beta}}}{\Gamma(\alpha)\beta^{\alpha}} - \sum_{0}^{\infty}[\frac{e^{\frac{-w}{(\beta/2)}}w^{2\alpha+k-1}}{(\beta/2)^{2\alpha+k}\Gamma(2\alpha+k)}\frac{\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)}] \tag{5.1.10}$$

Now by taking the mean value, we finally have:

$$E(W) = 2\alpha\beta - \sum_{0}^{\infty}\frac{(2\alpha+k)(\beta/2)\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)} \tag{5.1.11}$$

So the expected value of W can be determined by (5.1.11).

Let us assume that the slave node just uses the sequence of data that comes from (5.1.1) to estimate the value of θ. The slave node passes the data into a filter (for example a Kalman filter). Since the expected value of the random delay is not zero (in this case it is the mean value of the Gamma distribution plus the fixed part of the delay), the convergence point of the filter is:

$$\Delta_{DL} = \theta + d + \alpha\beta \tag{5.1.12}$$

Now assume that the slave clock uses (5.1.6) to estimate the value of θ. So it passes the result of (5.1.6) into a filter. As a result, the new convergence point would be:

$$\Delta'_{DL} = \theta + d + E(W) \tag{5.1.13}$$

where $E(W)$ is determined by (5.1.11). By subtracting (5.1.12) from (5.1.13):

$$\Delta_{DL} - \Delta'_{DL} = \alpha\beta - E(W)$$

$$\Delta_{DL} - \Delta'_{DL} = \alpha\beta - \left[2\alpha\beta - \sum_{0}^{\infty} \frac{(2\alpha+k)(\beta/2)\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)}\right]$$

$$\Delta_{DL} - \Delta'_{DL} = -\alpha\beta + \frac{\beta}{2}\underbrace{\sum_{0}^{\infty} \frac{(2\alpha+k)\Gamma(2\alpha+k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha+k+1)}}_{f(\alpha)}$$

$$\Delta_{DL} - \Delta'_{DL} = -\alpha\beta + \frac{\beta}{2}f(\alpha) \tag{5.1.14}$$

where $f(\alpha)$ is:

$$f(\alpha) = \sum_{0}^{\infty} \frac{(2\alpha + k)\Gamma(2\alpha + k)}{2^{2\alpha+k-1}\Gamma(\alpha)\Gamma(\alpha + k + 1)} \tag{5.1.15}$$

We can rewrite (5.1.14) as:

$$\boxed{\alpha\beta = \frac{\Delta_{DL} - \Delta'_{DL}}{\frac{f(\alpha)}{2\alpha} - 1}} \tag{5.1.16}$$

Equation (5.1.16) provides an explicit equation for estimating the mean value of the noise, which is the bias that we were trying to determine. The problem with this equation is the unknown value of α.

To proceed, we first focus on $f(\alpha)$ to study its characteristics. Fig. 15 shows $f(\alpha)/\alpha$ versus α.

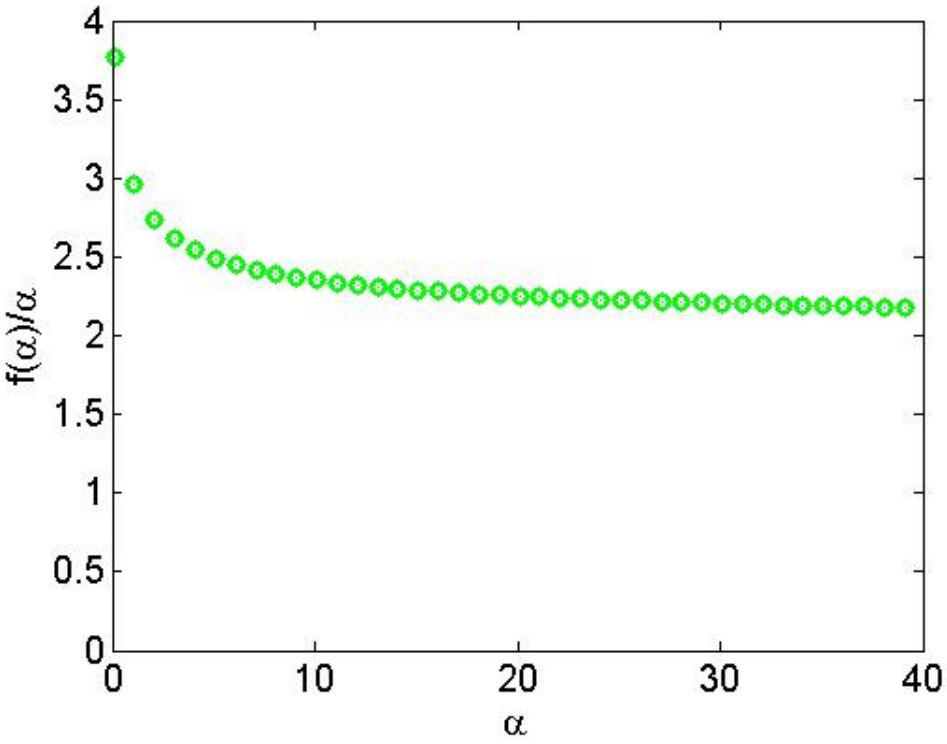

Figure 15: $f(\alpha)/\alpha$ versus α

As we observe in Fig. 15, as α decreases, $f(\alpha)/\alpha$ changes more rapidly. For

large values of α, this ratio gets closer to a constant number. By trial and error, we determine that:

$$\frac{f(\alpha)}{\alpha} \simeq 2(1 + \frac{0.56}{\sqrt{\alpha + 0.3}}) \tag{5.1.17}$$

Equation (5.1.17) provides an equation much simpler than (5.1.15) for calculating the denominator of (5.1.16).

Assume, that instead of the exact value of α, we know a neighborhood of it. This assumption is in fact a realistic assumption, since in many practical problems we may not be able to know the exact value of the delay parameters, but it is possible to have an initial guess. For example, in the busy hour, when the traffic is at its maximum, or during the night, when the traffic is at its minimum, we may have different expectations for the traffic load. The shaping parameter should be adjusted based on this knowledge about the traffic pattern. In the simulation results section, we will evaluate the robustness of our method against incorrect assumptions. We observe, however, that even under an incorrect assumption about the shaping parameter's value, we obtain a good estimation of the bias.

For estimating the frequency offset, we use the same method that was explained in Section 3.2 with a little modification. Since the bias correction happens after receiving two timing packets, equation (3.2.1) needs to be changed slightly as follow:

$$f_m(n) = \frac{\theta_m(2n) - \theta_m(2n-1)}{T_{2n} - T_{2n-1}} \tag{5.1.18}$$

In this scheme the estimation of the frequency offset is updated after two successive synchronization intervals.

In the context of IEEE 1588, bias correction from Equation (5.1.16) needs to be done for the down-link and up-link separately. To see how to implement this method, a summary of our algorithm is given by Algorithm 1.

Algorithm 1 :BC 1588

$\frac{f(\alpha)}{\alpha} = 2(1 + \frac{0.56}{\sqrt{\alpha+0.3}})$ $\%$*by knowing an approximation of* α

$\%$*four auxiliary variables to keep the summations*

$sum_{DL} \leftarrow 0,\ \ sum'_{DL} \leftarrow 0,\ \ sum_{UL} \leftarrow 0,\ \ sum'_{UL} \leftarrow 0$

For $n = 1, 2, ...$

$\delta_{DL}(n) = C(t_n + x_n) - t_n$

$\delta_{UL}(n) = t'_n - C(t'_n - y_n)$

$\theta_m(n) = \frac{\delta_{DL}(n) - \delta_{UL}(n)}{2}$

If n is even

$k = n/2$

$\delta'_{DL}(k) = min(\delta_{DL}(n), \delta_{DL}(n-1))$

$\delta'_{UL}(k) = min(\delta_{UL}(n), \delta_{UL}(n-1))$

$sum_{DL} = sum_{DL} + \delta_{DL}(n) + \delta_{DL}(n-1)$

$sum'_{DL} = sum'_{DL} + \delta'_{DL}(k)$

$sum_{UL} = sum_{UL} + \delta_{UL}(n) + \delta_{UL}(n-1)$

$sum'_{UL} = sum'_{UL} + \delta'_{UL}(k)$

$\Delta_{DL} = sum_{DL}/n$

$\Delta'_{DL} = sum'_{DL}/k$

$\Delta_{UL} = sum_{UL}/n$

$\Delta'_{UL} = sum'_{UL}/k$

$\%expected\ value\ of\ the\ down\text{-}link\ delay$

$E_{DL} = \frac{\Delta_{DL} - \Delta'_{DL}}{\frac{f(\alpha)}{2\alpha} - 1}$

$\%expected\ value\ of\ the\ up\text{-}link\ direction$

$E_{UL} = \frac{\Delta_{UL} - \Delta'_{UL}}{\frac{f(\alpha)}{2\alpha} - 1}$

$\%bias$

$B = \frac{E_{DL} - E_{UL}}{2}$

$f_m(k) = \frac{\theta_m(n) - \theta_m(n-1)}{t_n - t_{n-1}}$

$\%\theta_m(n)\ and\ f_m(k)\ would\ be\ passed\ into\ the\ filter$

$[u_\theta(k), u_f(k)] = Kalman(\theta_m(n), f_m(k))$

$C(t) = C(t) - u_\theta(k)$

$\%C(t)\ is\ the\ slave\ time$

$f(t) = f(t) - u_f(k)$

$\%f(t)\ is\ the\ slave\ frequency$

$\%The\ total\ bias\ is\ subtracted\ from\ the\ slave\ clock$

$C_{BC}(t) = C(t) - B$

end If

end For

Although using an approximated value for α can provide a good bias estimation,

the next section provides an extension to this method which enables us to estimate the value of α.

5.2 Estimating the shaping parameter

In order to estimate the shaping parameter, in addition to (5.1.16) we need to have one more independent equation. This equation can be found from the variance of the noise. So we first present a method for estimating the variance of the noise.

The bias estimator that we developed in Section 5.1 updates the slave clock not at each iteration, but after two iterations. This characteristic in fact provides for us a simple solution for estimating the variance of the noise. Fig. 16 shows two typical signal exchanges between a master and a slave for the down-link direction.

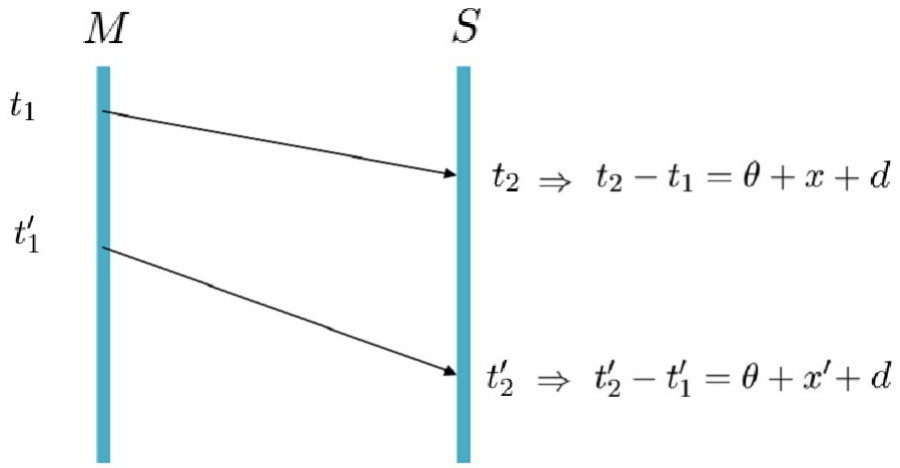

Figure 16: Two successive signal exchange for the down-link direction

We compute the following expected value:

$$\begin{aligned} E[((t_2 - t_1) - (t'_2 - t'_1))^2] &= E[((\theta + x + d) - (\theta + x' + d))^2] \\ &= E[(x - x')^2] \\ &= E(x^2) + E(x'^{\,2}) - 2E(xx') \\ \xrightarrow{x \text{ and } x' \text{ are i.i.d} \sim X} &= 2(E(X^2) - E^2(X)) = 2\sigma^2 \end{aligned}$$

As a result, the above expected value can give us an estimate of the down-link delay variance. A similar approach is then taken for the up-link delay. Note that because we correct the slave clock after two iterations, the phase offset (θ) in these two iterations is almost constant, so they cancel each other out.

By knowing the variance of the noise, we arrive at that additional equation that we were looking for. Similar to the last section, since we estimate and correct the bias for the down-link and up-link directions separately and in symmetric fashion, we just explain the equations for the down-link direction. We denote the estimated variance for the down-link delay as σ^2_{DL}. If we assume that the packet delay variation follows the Gamma distribution, we know:

$$\alpha\beta^2 = \sigma^2_{DL} \tag{5.2.1}$$

where $\alpha\beta^2$ is the expression for the variance of a Gamma distribution. By combining (5.2.1), (5.1.16) and (5.1.17), we end up with the following closed form equation for estimating α:

$$\alpha = \frac{0.3}{\left(\frac{0.56\sigma}{\Delta_{DL} - \Delta'_{DL}}\right)^2 - 1} \tag{5.2.2}$$

All the variables on the right hand side of this equation are known. The terms σ,

Δ_{DL} and Δ'_{DL} can be estimated recursively. So the BC 1588 method that we explained in the last section should be modified as follows. After two successive synchronization intervals, the estimated values for σ, Δ_{DL} and Δ'_{DL} are updated. These new estimated values are plugged into (5.2.2) to update the estimate of α. Then the new estimated value for α is substituted in (5.1.16) in order to give us a new estimation of the bias. This process is done for the down-link and up-link directions separately.

So far it seems that this solution provides an accurate estimation of the bias. However, in practice it does not happen. We will study the simulation results in the next chapter, but in order to show the problem with this method, consider the following figure. Fig. 17 shows a typical synchronization error, when we estimate the value of α by the above method. Here we have considered 20 percent down-link and 80 percent up-link traffic loads. The blue line close to zero shows the result for Basic 1588 (i.e., in the absence of bias estimation) and the range of the error in this case is in the order of tens of micro seconds. The green line shows the result when we use the bias correction method and estimate the value of α using (5.2.2).

As we observe the result for this method is much worse than the case that we have no bias correction. We also observe some large spikes in the figure. To understand the reason behind this we return to equation (5.2.2). Unfortunately the denominator of this equation gets very close to zero. To see this more clearly, from (5.1.17) we have:

$$\frac{f(\alpha)}{2\alpha} - 1 = \frac{0.56}{\sqrt{\alpha + 0.3}}$$

also from (5.2.1):

$$\sigma = \sqrt{\alpha}\beta$$

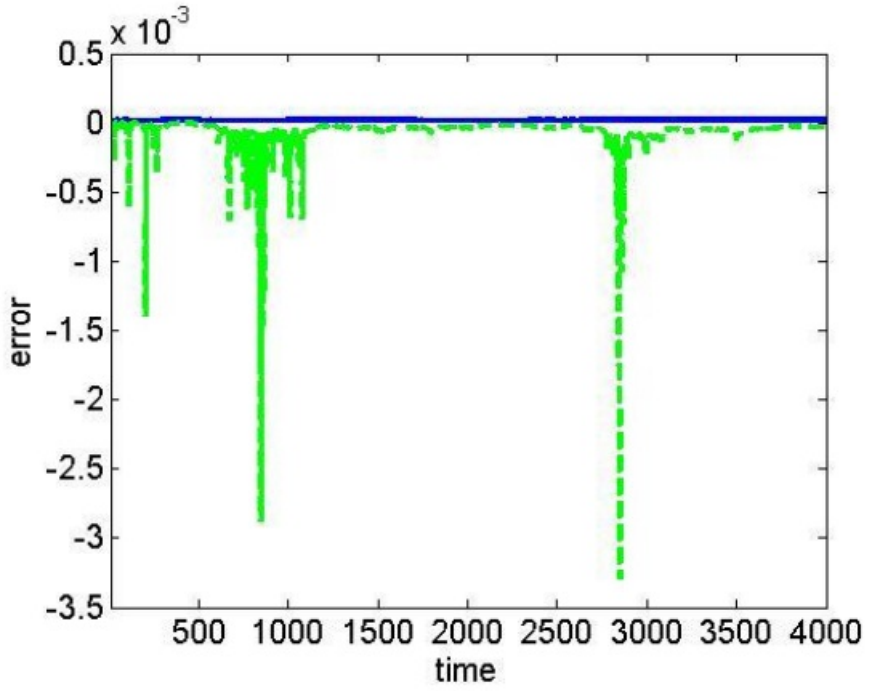

Figure 17: Error for 20 percent down-link and 80 percent up-link traffic loads

and finally from (5.1.16):

$$\Delta_{DL} - \Delta'_{DL} = \alpha\beta(\frac{f(\alpha)}{2\alpha} - 1)$$

By substituting these three equations into the denominator of (5.2.2), we have:

$$\left(\frac{0.56\sigma}{\Delta_{DL} - \Delta'_{DL}}\right)^2 - 1 = \frac{\alpha + 0.3}{\alpha} - 1$$

As we see, if we remove 0.3 from the above equation, the denominator of (5.2.2) gets zero. This observation also implies that if the shaping parameter (α) is a small number, then 0.3 can have a stronger effect and we should be able to have a more successful estimation.

In order to solve this problem, we apply two additional mechanisms to the algorithm. The first mechanism is to define a reasonable neighborhood for α. As we saw in Fig. 17, when the denominator of (5.2.2) gets very close to zero, due to computational problems, we get very bad estimates for α. By defining a neighborhood for α, we can filter out all the bad estimated values of α before using them for estimating the bias. This neighborhood should not be necessarily very tight and is just for preventing very poor estimates of α. For example let us denote the estimated value of α from (5.2.2) as $\hat{\alpha}_{DL}$. If the value of $\hat{\alpha}_{DL}$ is more than a lower boundary α^l_{DL} and less than an upper boundary α^u_{DL}, then it would be used for estimating the bias. The second mechanism that we use is to take the average of the estimated values for α. Taking the average smooths the results and will reduce the effect of bad estimated values. The first few estimated values may not be accurate enough, so we start to take the average of estimated values after a predetermined number of iterations. A summary of these two additional mechanisms is as follows. First we update our estimates of α using (5.2.2). If the estimated value falls between α^l_{DL} and α^u_{DL}, it can be used for estimation of bias. For a few initial estimated αs, we use them directly to update the bias (in the algorithm bellow, we consider the ten first iterations). After a predetermined number of iterations we start to average the estimates. Algorithm 2 summarizes the whole method:

Algorithm 2 :Advanced BC 1588

$\frac{f(\alpha)}{\alpha} = 2(1 + \frac{0.56}{\sqrt{\alpha+0.3}})$ *%by knowing an approximation of* α

%eight auxiliary variables to keep the summations

$sum_{DL} \leftarrow 0$, $sum'_{DL} \leftarrow 0$, $sum_{UL} \leftarrow 0$, $sum'_{UL} \leftarrow 0$, $sum^{\sigma}_{UL} \leftarrow 0$, $sum^{\sigma}_{UL} \leftarrow 0$,

$sum^{\alpha}_{DL} \leftarrow 0$, $sum^{\alpha}_{UL} \leftarrow 0$

$i_{DL} \leftarrow 1$, $i_{UL} \leftarrow 1$ *%counters*

For $n = 1, 2, ...$

$$\delta_{DL}(n) = C(t_n + x_n) - t_n$$
$$\delta_{UL}(n) = t'_n - C(t'_n - y_n)$$
$$\theta_m(n) = \frac{\delta_{DL}(n) - \delta_{UL}(n)}{2}$$

If n is even

$$k = n/2$$
$$\delta'_{DL}(k) = min(\delta_{DL}(n), \delta_{DL}(n-1))$$
$$\delta'_{UL}(k) = min(\delta_{UL}(n), \delta_{UL}(n-1))$$

$$sum_{DL} = sum_{DL} + \delta_{DL}(n) + \delta_{DL}(n-1)$$
$$sum'_{DL} = sum'_{DL} + \delta'_{DL}(k)$$
$$sum_{UL} = sum_{UL} + \delta_{UL}(n) + \delta_{UL}(n-1)$$
$$sum'_{UL} = sum'_{UL} + \delta'_{UL}(k)$$
$$sum^{\sigma}_{DL} = sum^{\sigma}_{DL} + (\delta_{DL}(n) - \delta_{DL}(n-1))^2$$
$$sum^{\sigma}_{UL} = sum^{\sigma}_{UL} + (\delta_{UL}(n) - \delta_{UL}(n-1))^2$$

$$\Delta_{DL} = sum_{DL}/n$$
$$\Delta'_{DL} = sum'_{DL}/k$$
$$\Delta_{UL} = sum_{UL}/n$$
$$\Delta'_{UL} = sum'_{UL}/k$$
$$\sigma^2_{DL} = sum^{\sigma}_{DL}/2k$$
$$\sigma^2_{UL} = sum^{\sigma}_{UL}/2k$$

%estimate the value of α

$$\hat{\alpha}_{DL} = \frac{0.3}{\left(\frac{0.56\sigma_{DL}}{\Delta_{DL}-\Delta'_{DL}}\right)^2 - 1}$$
$$\hat{\alpha}_{UL} = \frac{0.3}{\left(\frac{0.56\sigma_{UP}}{\Delta_{UL}-\Delta'_{UL}}\right)^2 - 1}$$

%the estimated α *should be in the proper neighborhood*

If $\alpha^l_{DL} < \hat{\alpha}_{DL} < \alpha^u_{DL}$

  If $i_{DL} > 10$

    $sum^{\alpha}_{DL} = sum^{\alpha}_{DL} + \hat{\alpha}_{DL}$

    $\alpha = sum^{\alpha}_{DL}/(i_{DL} - 10)$

  else

    $\alpha = \hat{\alpha}_{DL}$

  end If

  $i_{DL} = i_{DL} + 1$

end If

If $\alpha^l_{UL} < \hat{\alpha}_{UL} < \alpha^u_{UL}$

  If $i_{UL} > 10$

    $sum^{\alpha}_{UL} = sum^{\alpha}_{UL} + \hat{\alpha}_{UL}$

    $\alpha = sum^{\alpha}_{UL}/(i_{UL} - 10)$

  else

    $\alpha = \hat{\alpha}_{UL}$

  end If

  $i_{UL} = i_{UL} + 1$

end If

$E_{DL} = \frac{\Delta_{DL}-\Delta'_{DL}}{\frac{f(\alpha)}{2\alpha}-1}$ *%expected value for the down-link delay*

$E_{UL} = \frac{\Delta_{UL}-\Delta'_{UL}}{\frac{f(\alpha)}{2\alpha}-1}$ *%expected value for the up-link delay*

$B = \frac{E_{DL}-E_{UL}}{2}$ $\quad \%total\ bias$

$f_m(k) = \frac{\theta_m(n)-\theta_m(n-1)}{t_n-t_{n-1}}$

$\%\theta_m(n)\ and\ f_m(k)\ would\ be\ passed\ into\ the\ filter$
$[u_\theta(k), u_f(k)] = Kalman(\theta_m(n), f_m(k))$

$C(t) = C(t) - u_\theta(k)$ $\quad \%C(t)\ is\ the\ slave\ time$

$f(t) = f(t) - u_f(k)$ $\quad \%f(t)\ is\ the\ slave\ frequency$

$\%\,The\ total\ bias\ is\ subtracted\ from\ the\ slave\ clock$
$C_{BC}(t) = C(t) - B$

end If

end For

When any information about the network traffic load is available, the neighborhood that we defined for α in order to prevent those large spikes can also be used as a tool for improving the synchronization accuracy. To see how this is possible, consider the first version of our method where we used an approximate value of α for estimating the bias. Instead of an approximate value, assume we just know that the traffic load should be bounded by a possible range for α. Based on this knowledge we can consider a tighter neighborhood for the value of α. In the next chapter we see that this strategy can improve the synchronization accuracy.

Chapter 6

Simulation Results

To evaluate our method, we generated PDVs using the results of [**?**] and [**?**]. In the first section of this chapter, we use Algorithm 1 (BC 1588) from the last chapter to estimate the bias. This algorithm works using an approximated value for α. Then the performance of Algorithm 2 (Advanced BC 1588) will be evaluated. Using Algorithm 2, we can jointly estimate the value of the shaping parameter and the bias.

We compare the performance of our methods with Basic 1588 (see Section 3.4.2) and the Boot-strap method. The authors of the Boot-strap method did not provide any algorithm for estimating the frequency offset, so we first assume that the frequency offset is zero and compare the performance of our method with the Boot-strap method. Then we take the frequency offset into consideration and evaluate the performance of our method in the presence of a frequency offset. Finally, as was described in the algorithm section, we use a filter to process the data. In the following simulations, a Kalman filter is used for this purpose.

6.1 Bias estimation using an approximated α

For estimating the bias using Algorithm 1, we need to have a coarse approximation of α. In a fixed network, the value of α is different for different traffic loads. We first

assume that we know the exact value of the shaping parameter, then the next step is to examine the robustness of our method against incorrect assumptions.

In Fig. 18 we assume that we have a 20% traffic load from master to slave and 60% traffic load from slave to master. PDVs are generated based on the results of [?] and [?], so the values of α for 20 and 60 percent traffic loads are 2 and 8 respectively. The synchronization interval is assumed to be 1 sec and we assume that the frequency offset is zero. The phase offset, however, is 1 sec. The time axis starts from t=15 sec. In this figure we used the exact values of α to estimate the bias.

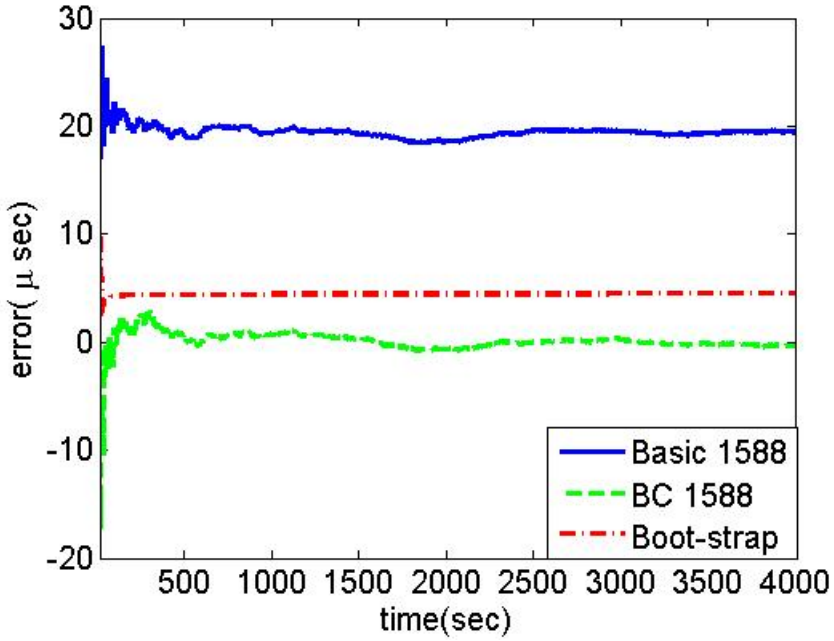

Figure 18: Error in the estimated phase offset for 20 percent down-link and 60 percent up-link traffic loads.

From the result for Basic 1588 it can be seen that the existence of the bias in the estimated phase offset degrades the synchronization accuracy. It also can be observed that because the PDVs are following the Gamma distribution instead of the exponential distribution, the Boot-strap method is not very accurate. Our method, however, can provide a more accurate synchronization.

Of course it is not realistic to assume that we know the exact value of the shaping parameter. So in the above example, let us assume that instead of the exact value of the traffic loads, we just know that traffic load from the master to slave is somewhere between 20 and 40 percent (from [?] the value of α for 40 percent traffic load is 6). By considering the middle point between these values, we would have $\alpha = 4$. Fig. 19 shows that using this approximation (rather than the correct value for α) causes our results to be worse, but our method still works better than the Boot-strap.

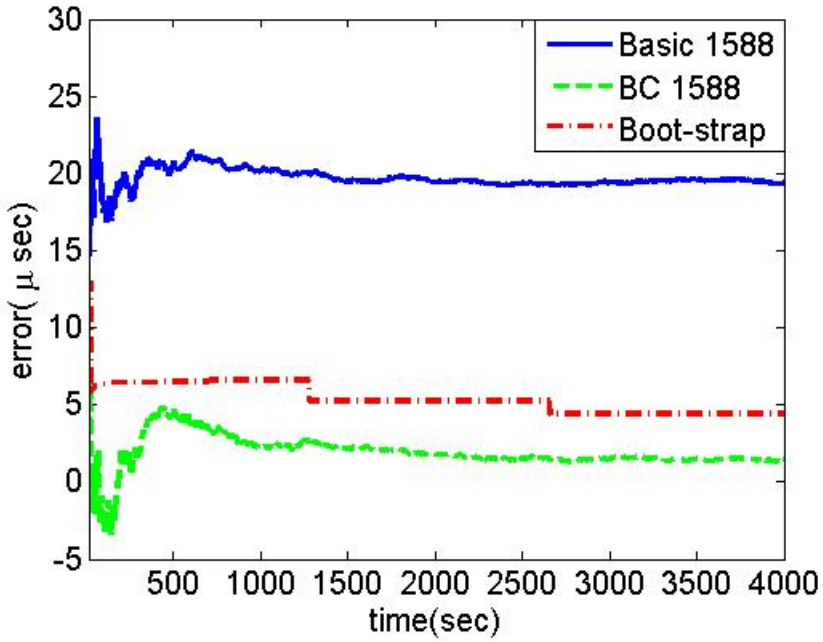

Figure 19: Error in the estimated phase offset for 20 percent down-link and 60 percent up-link traffic loads, with an incorrect assumption about the shaping parameter value.

In Fig. 20, the up-link and down-link traffic loads are assumed to be 80 and 20 percents respectively. The value of α for 80 percent traffic load from [?] is 11. By increasing the traffic load, and as a result, increasing the value of α, the PDV becomes less exponential. Consequently, as can be seen in Fig. 20, the Boot-strap method suffers from a larger error. In Fig. 21, we evaluate the performance of BC 1588 for different assumptions/approximations of α. For this purpose, we keep all the initial

settings the same as in Fig. 20. However, this time, in addition to using the exact value of α (for the up-link direction), we also evaluate the performance of BC 1588 using a number of approximated values for the shaping parameter. We observe that when we use the exact value of α ($\alpha = 11$), the error converges to zero. As we get far from the exact shaping parameter the error increases. By comparing Fig. 21 and Fig. 20, it can be seen that for a wide range of incorrect assumptions/approximations of α, BC 1588 still provides better results than the Boot-strap method. For example, using $\alpha = 7$ for estimating the bias while the actual α is 11 means that while the actual traffic load is 80%, the receiver thinks that it is around 50%. Nevertheless, even for $\alpha = 7$, BC 1588 is still slightly better than the Boot-strap method.

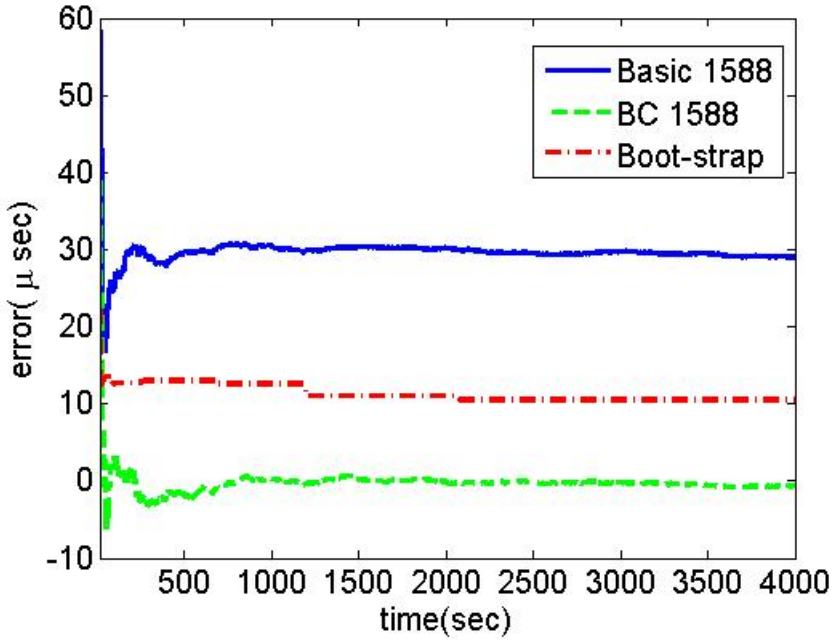

Figure 20: Error in the estimated phase offset for 20 percent down-link and 80 percent up-link traffic loads.

In previous figures we assumed that the frequency offset is zero. In practice, this assumption is clearly wrong and a slave clock drifts from a master clock gradually. Unfortunately, in [**?**] no mechanism has been provided for estimating the frequency

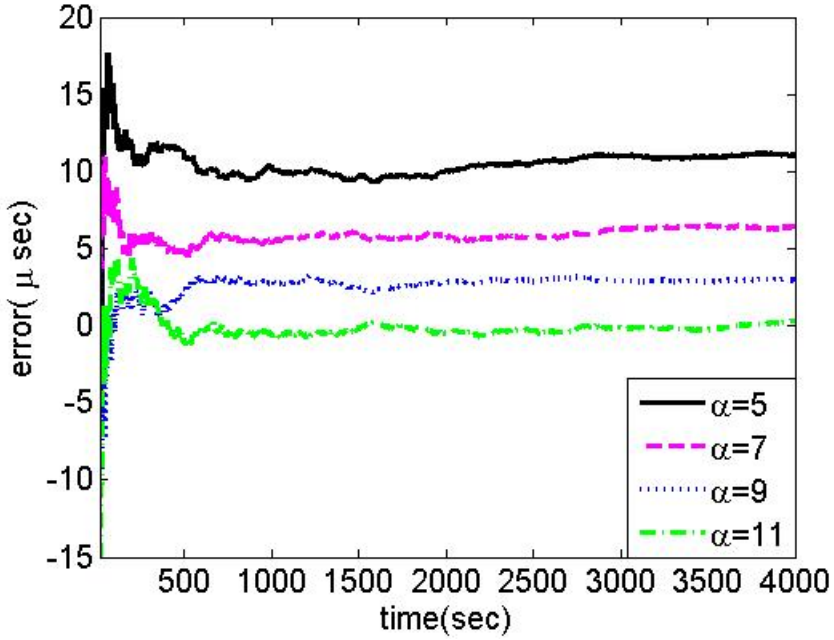

Figure 21: Performance evaluation of BC 1588 for 20 percent down-link and 80 percent up-link traffic loads, and different assumptions about the shaping parameter.

offset. As a result, in the presence of a frequency offset we just compare our results with Basic 1588. In Fig. 22, all the initial conditions are the same as in Fig. 20, except that the frequency offset of the slave clock is 10^{-6} per second. Therefore the slave time drifts from the master time with a rate of 10^{-6} second per second. Comparing the results in Fig. 22 with the results in Fig. 20, we observe that the existence of a frequency offset does not degrade the results of our method. In fact, in dealing with the frequency offset there is no difference between BC 1588 and Basic 1588. This is because BC 1588 and Basic 1588 estimate the frequency offset using (3.2.1) and (5.1.18) respectively, both of which are based on the same principle.

In Fig. 23 we evaluate the performance of BC 1588 in the presence of different frequency offsets. All the initial settings except of the variant frequency offsets are the same as in Fig. 22. As we observe, after a few initial iterations, the frequency offset is estimated and removed from the slave time. After correcting for the frequency offset,

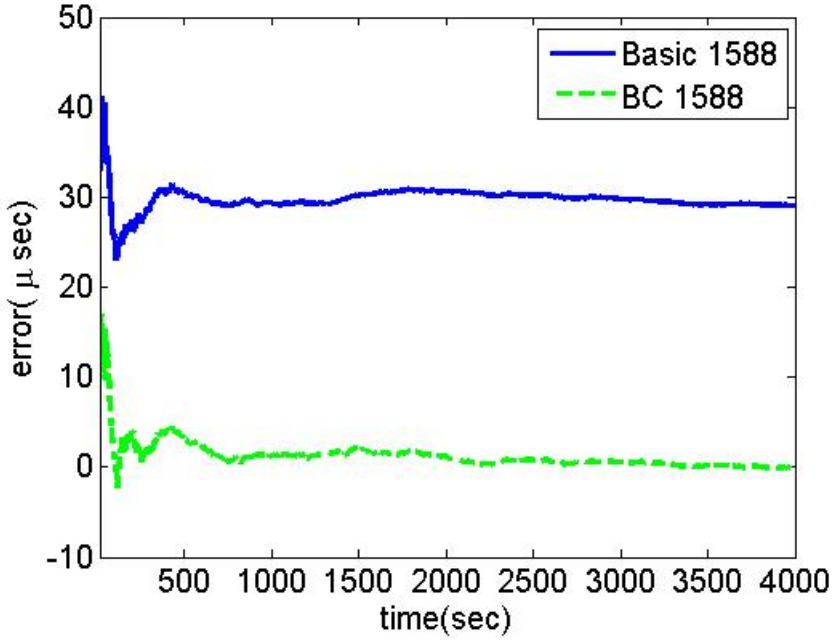

Figure 22: Error in the estimated phase offset for 20 percent down-link and 80 percent up-link traffic loads, with a frequency offset of 10^{-6}.

no significant difference in their level of error is observed. So we see that BC 1588 works well even if there is a large frequency offset between the master and slave clock.

6.2 Joint estimation of the bias and the shaping parameter

In this section we estimate the bias using Algorithm 2. We first assume that α^l_{DL}=α^l_{UL}=1 and α^u_{DL}=α^u_{UL}=15. From [?] and [?] the value of α for 20 percent and 80 percent traffic loads in a 5 hops network is 2 and 11 respectively. So the neighborhood that we consider for the shaping parameter is loose and is just based on a very approximate knowledge about the range of α.

In Fig. 24 we assume that we have a 20% traffic load from master to slave and 60% traffic load from slave to master. The initial settings are the same as in Fig. 18.

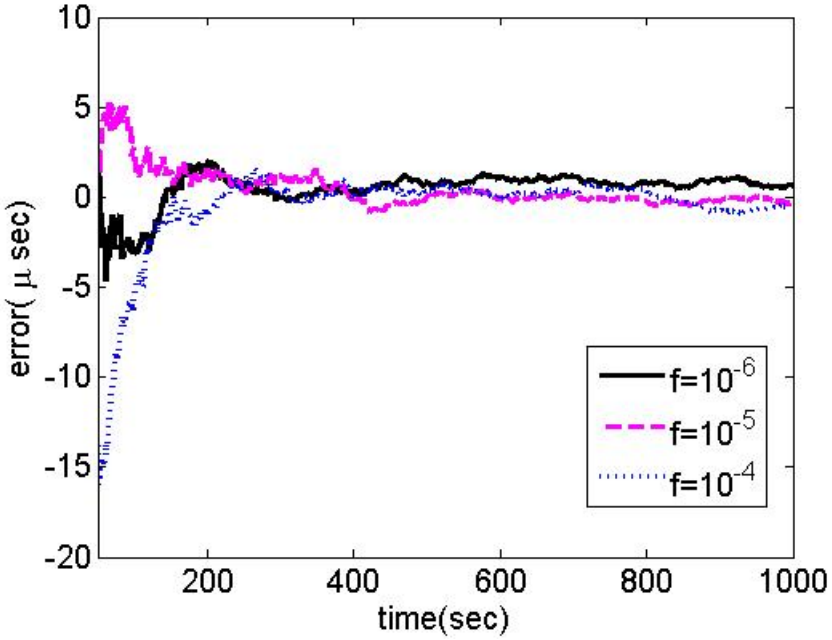

Figure 23: Performance evaluation of BC 1588 for 20 percent down-link and 80 percent up-link traffic loads and different frequency offsets.

Advanced BC 1588 estimates the shaping parameters along with the bias. In Fig. 24, the estimated shaping parameters for the down-link and up-link directions are 2.4 and 9.4 respectively. In BC 1588, instead of estimating the shaping parameter, we use an approximate value. Using a fixed value for α results in more predictable simulation output, with the performance varying very little from simulation run to simulation run. On the other hand, when we use Advanced BC 1588, it is possible that we sometimes get better or worse results than Fig. 24, but this figure is a typical output. We also observe that although Advanced BC 1588 still has better results than the Boot-strap method, it suffers from a noticeable level of error.

The initial settings in Fig. 25 are the same as in Fig. 20. So the shaping parameters for the down-link and up-link directions are 2 and 11 respectively. The estimated shaping parameters from Advanced BC 1588 for the down-link and up-link directions are 2.1 and 9.2 respectively.

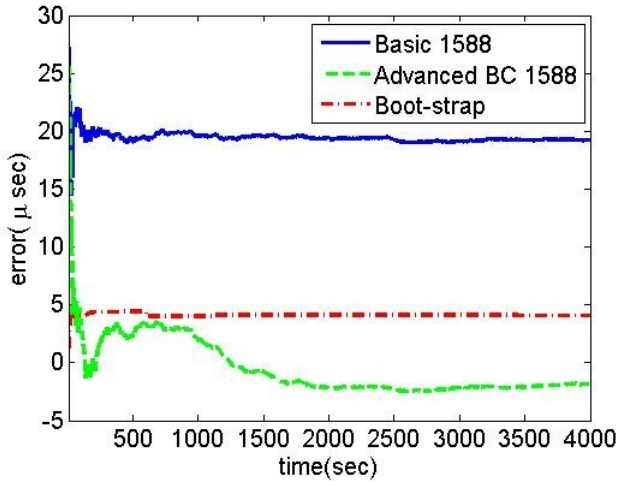

Figure 24: Error in the estimated phase offset for 20 percent down-link and 60 percent up-link traffic loads.

In both Fig. 24 and Fig. 25 we observe that for the initial iterations Advanced BC 1588 does not have a good performance. The reason is that this method, in addition to the bias, estimates the variances and the shaping parameters of the down-link and up-link delays. As a result it takes some time/iterations for the method to obtain good estimates for each of these variables.

So far we assumed that the frequency offset is zero. In Fig. 26 we set all the initial settings the same as in Fig. 25, except that the frequency offset of the slave clock is 10^{-6} per second. Similar to the last section, because the authors of the Boot-strap method have not provided any method for estimating the frequency offset, we just compare our method to Basic 1588.

Comparing the results in Fig. 26 with the results in Fig. 25, we observe that the existence of a frequency offset does not degrade the results of our method. The estimated shaping parameters for the down-link and up-link delays in Fig. 26, are 3.2 and 11.2 respectively.

At the end of the last chapter we mentioned that when information about the network traffic load is available, our method is flexible enough to provide better results.

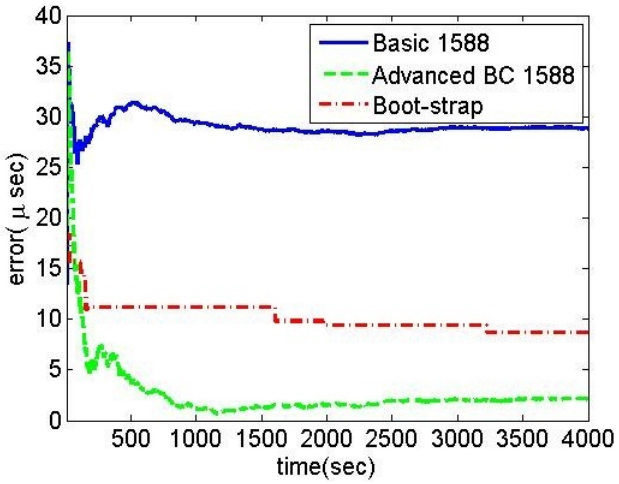

Figure 25: Error in the estimated phase offset for 20 percent down-link and 80 percent up-link traffic loads.

For this purpose we can readjust the values of α^l_{DL}, α^u_{DL}, α^l_{UL} and α^u_{UL} based on the available information. In Fig. 27 we considered 20 percent down-link and 40 percent for the up-link traffic loads. The value of α for the 40 percent traffic load from [**?**] is 6. Let us assume that the receiver just knows that the down-link traffic load is somewhere between 10 to 40 percents and the up-link traffic load is somewhere between 10 to 60 percents. So $\alpha^l_{DL} = \alpha^l_{UL} = 1$, $\alpha^u_{DL} = 6$ and $\alpha^u_{UL} = 8$. We observe that although these are not very tight neighborhoods, such boundaries can improve the synchronization accuracy noticeably. The estimated values for the shaping parameters are 2.3 and 5.5 for the down-link and up-link directions respectively. A tighter neighborhood can also make the output more predictable. It means that we can be sure that the level of error because of a wrong estimation never exceeds a specific level.

To make the comparison easier, in Fig. 28 we compare the performance of Advanced BC 1588 with a looser neighborhood for α with Advanced BC 1588 with a tighter neighborhood. Consider a 20 percent down-link and 60 percent up-link traffic load. For the looser neighborhood, we consider the same neighborhood as in Fig. 25.

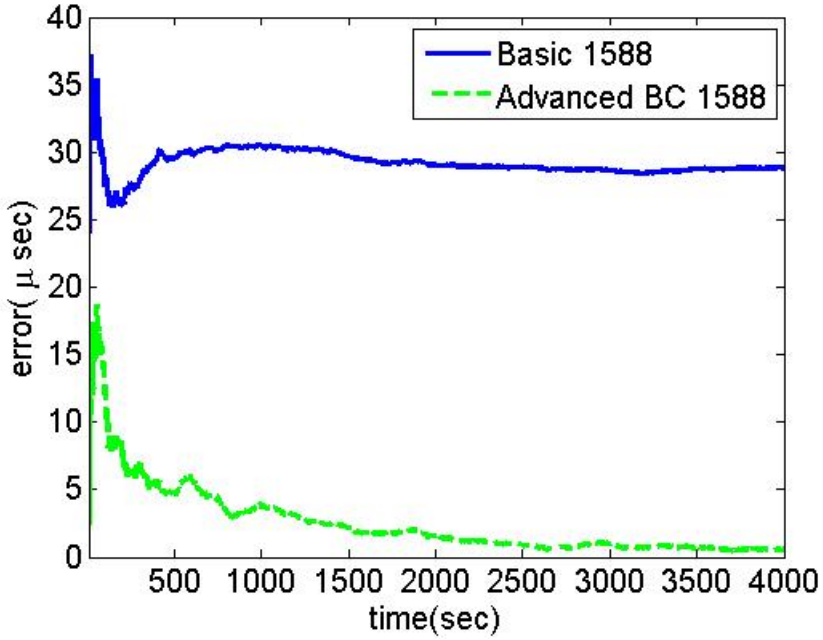

Figure 26: Error in the estimated phase offset for 20 percent down-link and 80 percent up-link traffic loads, with a frequency offset of 10^{-6}.

So $\alpha^l_{DL} = 1$, $\alpha^u_{DL} = 15$, $\alpha^l_{UL} = 1$ and $\alpha^u_{UL} = 15$. For the tighter neighborhood, assume that the receiver just knows that the down-link traffic load is somewhere between 10 to 40 percents and the up-link traffic load is between 40 and 80 percents. So $\alpha^l_{DL} = 1$, $\alpha^u_{DL} = 6$, $\alpha^l_{UL} = 6$ and $\alpha^u_{UL} = 11$.

The estimated values for the shaping parameters are 2.4 and 8.5 for the down-link and up-link directions respectively. We observe that this coarse knowledge about the traffic loads can improve the synchronization accuracy noticeably. This flexibility is one of the advantages of our method that is observed in both BC 1588 and Advanced BC 1588 (for the BC 1588 we use an approximated value for α so if we have a better knowledge about the traffic loads, we would have a better approximation).

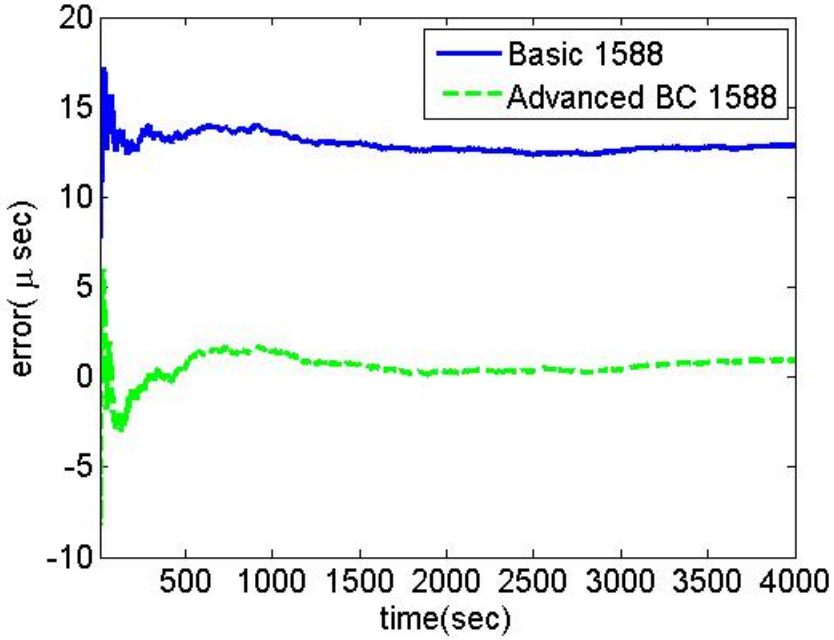

Figure 27: Error in the estimated phase offset for 20 percent down-link and 40 percent up-link traffic loads, with a frequency offset of 10^{-6}.

6.3 Non-Gamma distributed PDVs

Although the results of [**?**] and [**?**] show that the PDVs can be modeled by Gamma distributions, in this section we answer what would happen if the PDVs do not follow the Gamma distribution.

A rule of thumb is that even if the PDVs do not follow the Gamma distribution, as long as there are some values for the shaping and scale parameters to make the Gamma distribution similar to the non-Gamma distributed PDVs, our method still can provide good results. However, if the PDVs are very different from the Gamma distribution, then we may have a large amount of error. This problem is more serious specifically for Advanced BC 1588. The reason is that for this method we also estimate the shaping parameters. In order to estimate the shaping parameters, we first estimate the variance of the delays. When the estimated variances do not follow the Gamma distribution, then the estimated values for the shaping parameters would

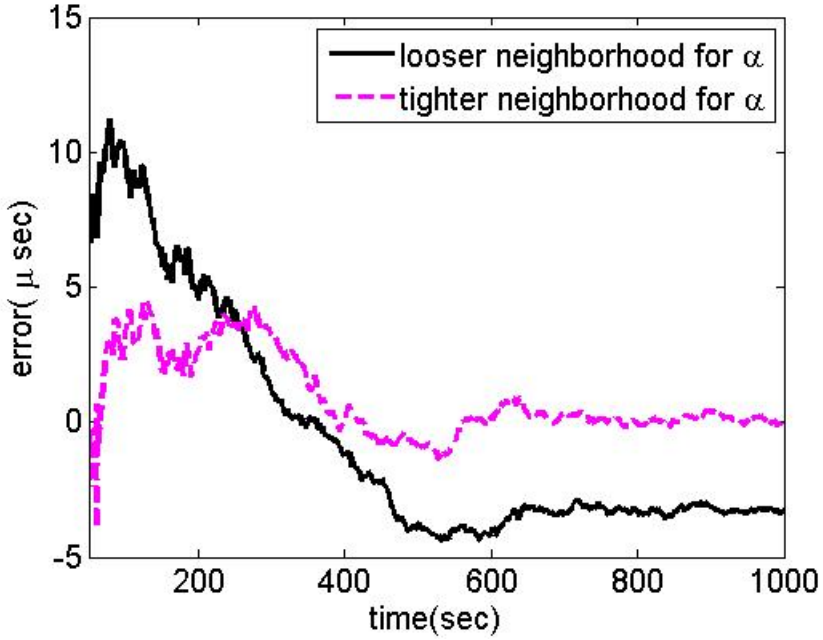

Figure 28: Error in the estimated phase offset for 20 percent down-link and 60 percent up-link traffic loads, with a frequency offset of 10^{-6}.

not be accurate anymore. Estimating the bias using a poor estimation of the shaping parameters will result in a bad estimate of the bias.

Fig. 29 illustrates the result of using the Weibull distribution to generate the PDV for the down-link delay. The delay is generated as follow:

$delay_{DL} = 133 * 10^{-6} + 10^{-6} * Weibull(0.5, 6.5)$

Here Weibull(0.5,6.5) is a Weibull random number generator with the scale parameter 0.5 and shape parameter 6.5. The up-link delay is generated using the results of [**?**] for 80 percent traffic load. We assumed that $\alpha^l_{DL} = 1$, $\alpha^u_{DL} = 6$, $\alpha^l_{UL} = 6$ and $\alpha^u_{UL} = 11$. As we observe in Fig. 29, although the distribution of the delay for the down-link direction does not follow the Gamma distribution, Advanced BC 1588 can estimate the bias successfully.

To make the comparison easier, in Fig.30 we compare the performance of Advanced BC 1588 when the down-link delay is generated using Weibull and uniform

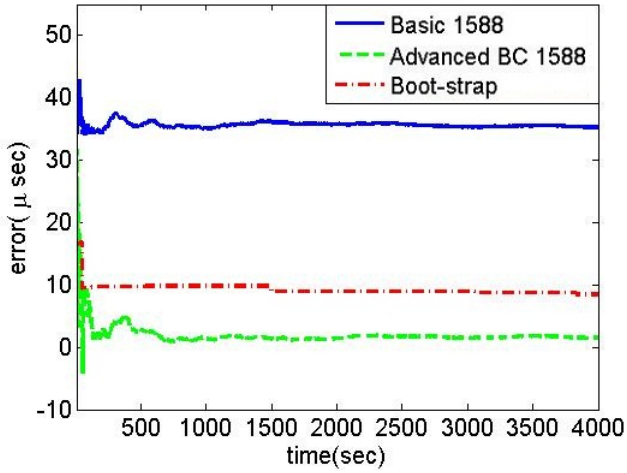

Figure 29: Error in the estimated phase offset for the Weibull distributed down-link and Gamma distributed up-link delays.

distributions. The up-link delay is the same as in Fig.29. The uniform distribution is generated as follow:

$delay_{DL} = 133 * 10^{-6} + 10^{-6} * (10 * U(0,1))$

Here $U(0,1)$ generates random numbers between 0 and 1 from uniform distribution. We observe that changing the distribution from Weibull to uniform degrades the result. This was predictable because there is less similarity between the Gamma and uniform distributions in comparison with the Weibull and Gamma distribution.

6.4 Time changing traffic loads

So far we have assumed that we have a constant traffic load in the network. Clearly this assumption is not valid and traffic loads and as a result PDVs change with time. In this section we evaluate the performance of our method when traffic loads change with time. In reality, changing the traffic load does not usually happen very fast rather it happens gradually. However, in this section we consider the extreme case. We assume that the traffic load changes instantly and significantly. In Fig. 31 we

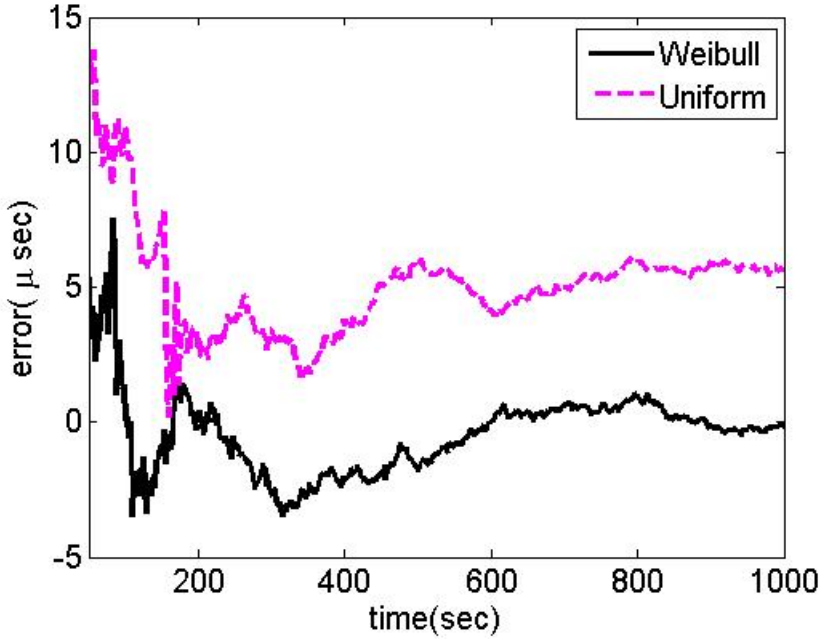

Figure 30: Comparison between the error in the estimated phase offset for the Weibull and uniform distributed down-link and Gamma distributed up-link delays.

assume that we have 20 percent traffic load from the master to the slave. The traffic load from the slave to the master initially is assumed to be 60 percent. At t=500 sec, the up-link traffic load suddenly changes from 60 to 80 percent. Then, at t=3000 the up-link traffic load changes from 80 to 40 percent. The values of α for the 40, 60 and 80 percent traffic loads from [**?**] and [**?**] are 6, 8 and 11 respectively. We assumed that $\alpha^l_{DL} = 1$, $\alpha^u_{DL} = 6$, $\alpha^l_{UL} = 6$ and $\alpha^u_{UL} = 11$.

Fig. 31 shows that when the traffic load changes, our method looses its accuracy. The reason is that all the estimated variables until the time that PDV changes, are based on the previous traffic load. When the traffic load changes the previous estimated values act as a wrong memory. It means that, until after a long time when the effect of the wrong memory fades, we cannot obtain an accurate estimation. The relatively good results of Basic 1588 is partly accidental. In other words, in this example, the wrong memory of the filter helps Basic 1588 method to obtain better

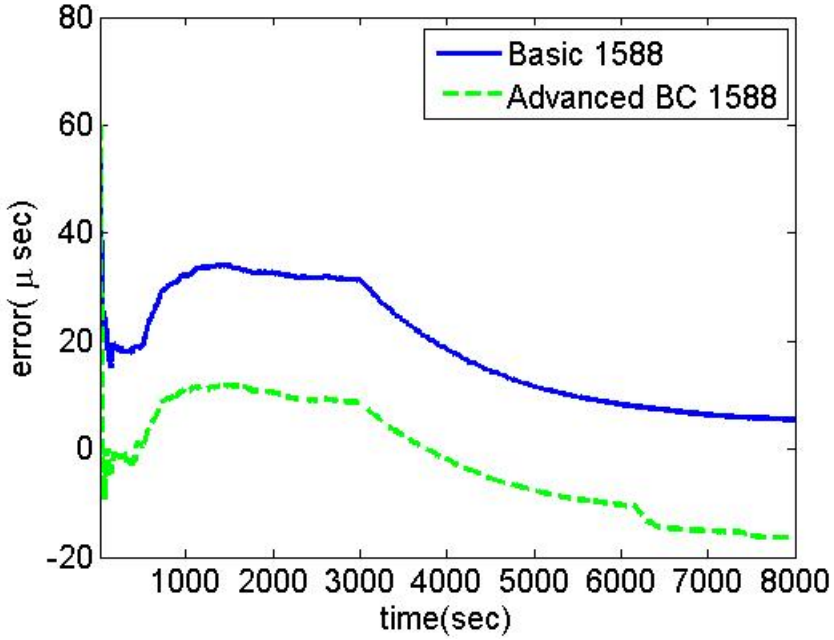

Figure 31: Error in the estimated phase offset. The down-link traffic load is 20 percent. The up-link traffic load changes from 60 to 80 percent at t=500 sec and changes form 80 to 40 percent at t=3000 sec.

results. The other reason is that when the traffic load for the up-link delay drops down to 40 percent at t=3000 sec, given the 20 percent down-link traffic load, we have relatively symmetric network. One other characteristic that is observed in Fig. 31 is that changing the traffic load, affects BC 1588 pretty much similar to Advanced BC 1588. In Basic 1588 there is no bias correction and we are just updating the slave clock using a Kalman filter. Comparing Basic 1588 and Advanced BC 1588 in Fig. 31 shows that regarding the time changing PDVs, the filter that we are using to update the slave clock is more important than the bias correction part.

One simple solution to this problem is to restart the synchronization process periodically. The process needs to be restarted often enough to catch significant changes in network traffic load. For example, in [**?**], traffic loads change never faster than every 12 minutes, and sometimes at an even slower frequency. All the initial settings in Fig. 32 are the same as in Fig. 31. In Fig. 32 we restart Basic 1588

and Advanced BC 1588 with a period of 1000 sec. It means that when the time is a multiple of 1000, we restart all the summations and matrices that are involved in Kalman filtering and bias estimation processes. The red line in Fig. 32 shows the zero line, in order to make the comparison easier.

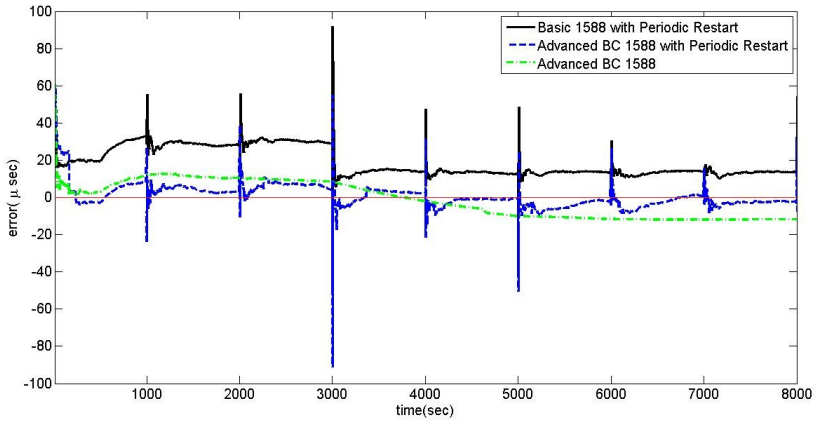

Figure 32: Error in the estimated phase offset. The down-link traffic load is 20 percent. The up-link traffic load changes from 60 to 80 percent at t=500 sec and changes form 80 to 40 percent at t=3000 sec. The period of restart is 1000

Whenever the process is restarted, previous estimated values are assigned to be zero. As a result when traffic load changes with time, in comparison with the case that the process is not restarted, there is less outdated and potentially misleading memory in the estimation equations. We observe that Advanced BC 1588 with Periodic Restart, between two successive restart points, has a better performance than Advanced BC 1588. However, a serious problem with this approach are the random spikes that happen, for the first initial iterations, whenever the synchronization process is restarted.

In order to mitigate the effect of random spikes, we propose the following solution. Suppose that whenever the synchronization process is restarted, a temporary slave clock for a predetermined interval starts to work. The temporary slave clock, before the synchronization process is restarted for the slave clock, sets its time and frequency

to the slave clock. Whenever the synchronization process is restarted, the temporary slave clock acts as the slave clock for a predetermined interval. Meanwhile, the slave clock is updating its estimates and is passing the initial spikes. When the interval is finished, the time again is read from the slave clock. In Fig. 33 all the initial settings are the same as in the Fig. 32. The predetermined interval in Fig. 33 is assumed to be 1/10 of the restart period so it is 100.

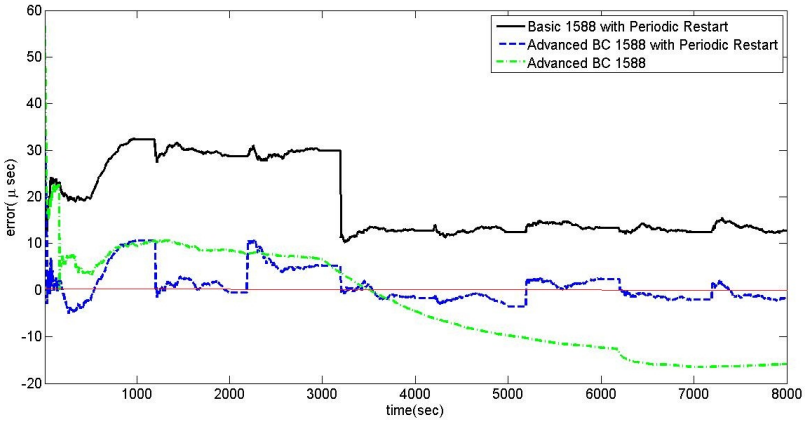

Figure 33: Error in the estimated phase offset. The down-link traffic load is 20 percent. The up-link traffic load changes from 60 to 80 percent at t=500 sec and changes form 80 to 40 percent at t=3000 sec. The period of restart is 1000. The interval for working the temporary clock is 100 sec

As we observe in Fig. 33, using this strategy mitigates the effect of random spikes significantly. Consequently, Advanced BC 1588 with Periodic Restarts, provides good results without severely getting affected by changing traffic loads.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis we presented two methods for estimation and correction of the delay's bias. Both of these methods allow us to correct the slave clock recursively. They are compatible with the current version of the IEEE 1588 PTP. They also can be applied to different kinds of filters such as the Kalman filter.

For BC 1588, we just require a coarse approximation about the value of the shaping parameters. It was shown that even when this approximation is incorrect, at least for a noticeable range of error, we still can have a good estimation of the bias. Advanced BC 1588 on the other hand, in addition to determining the bias is able to estimate the shaping parameters too. As a result, unlike BC 1588, it does not require an approximation about the shaping parameters. Because Advanced BC 1588 estimates all the parameters itself it takes a larger numbers of iterations to provide a good estimation than BC 1588. More importantly, for Advanced BC 1588 it is very important that the PDVs can be modeled by a Gamma distribution or a distribution close to it. Both Advanced BC 1588 and BC 1588 are flexible methods. It means that if any knowledge about the network traffic load is available, these methods can use it to improve their performance. This is an important characteristic because in

reality a coarse knowledge about the network traffic load is usually available at the slave node. Methods such as the Boot-strap method are not able to take advantage of this knowledge. For our methods on the other hand, as the knowledge about the traffic load is more accurate, they can improve their results.

7.2 Future Works

- Our methods are developed based on the assumption that PDVs are following the Gamma distribution. We also showed that when PDVs are not Gamma distributed but are close to this distribution, our methods still are able to provide good results. However, as the PDVs get less Gamma distributed, the performance of our methods degrade. As a future work, these methods can be extended to non-Gamma distributed PDVs. For this purpose two suggestions can be considered. First, to use a more complicated distribution than the Gamma distribution in order to model the PDVs. This suggestion does not seem very promising for two reasons. First, finding a closed form equation for bias, similar to the equations we found for the Gamma distribution, may not be possible. Moreover, even if we can find a closed form equation for the bias, the PDVs still may not follow the new distribution. The other suggestion is to consider a combination of several distributions. For example, instead of one Gamma distribution, we can consider several Gamma distributions. This can be happen by putting the observed delay values into buckets, assuming that delays of different values are caused by a different underlying PDV distribution. This approach will make all the equations more complicated, but still seems more promising than the first suggestion.

- In this work we assumed that the fixed part of the delays in the forward and reverse directions are equal with each other. As we already mentioned, this

assumption usually is true. However, there might be cases that we cannot make this assumption. The fixed part of the delay is a constant number that needs to be estimated separately and with another mechanism. Knowing the number of hops between the master and slave nodes and the processing time of the devices located in the path can be used to estimate the fixed part of the delay.

- Increasing the speed of estimation is an important topic for future work. Suppose that the PDVs change with time with a fairly fast speed. In that case the period of restart that we defined for the Advanced BC 1588 should get smaller. But we know that reducing the restarting period is not desirable because whenever we reset the process there are some random fluctuations. In the last chapter we explained how using a temporary slave clock can mitigate the effect of random fluctuations. Nevertheless, a larger window size allows the method to have a better estimation. One of the advantages of our methods is that they can be implemented using different filters. Trying other kinds of filters might be helpful in order to overcome fast-changing PDVs.

- As we just mentioned, different filters can be used to implement our method. Given the fact that PDVs do not follow the Gaussian distribution, using other filters such as particle filters may provide better results than the Kalman filter and further improve the synchronization accuracy.

List of References

[1] "The five dangers of poor network timekeeping," *Symmetricom white paper*, 2009.

[2] "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems," 2008.

[3] 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*, Nov 2008.

[4] "IEEE 1588 precise time protocol: The new standard in time synchronization," *Symmetricom white paper*, 2009.

[5] D. Mills, U. Delaware, and J. Martin, "Network time protocol version 4: Protocol and algorithm," *Request for Comments RFC 5905, Internet Engineering Task Force*, June 2010.

[6] "Standard for a precision clock synchronization protocol for networked measurement and control systems," 2002.

[7] J. Ferrant, M. Gilson, S. Jobert, and M. Mayer, "development of the first ieee1588 telecom profile to address mobile backhaul needs," *IEEE Communications Magazine*, vol. 48, pp. 118–126, 2010.

[8] G. Giorgi and C. Narduzzi, "Performance analysis of kalman-filter-based clock synchronization in ieee 1588 networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, pp. 2902 – 2909, Aug 2011.

[9] D. T. Bui, A. Dupas, and M. L. Pallec, "Packet delay variation management," *International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, pp. 1–6, 2009.

[10] G. A. Terejanu, "Discrete kalman filter tutorial," *University at Buffalo, Department of Computer Science and Engineering, NY 14260*.

[11] B. Friedland, "Treatment of bias in recursive filtering," *IEEE Transactions on Automatic Control*, vol. 14, pp. 359 – 367, 1969.

[12] T. Murakami and Y. Horiuchi, "Improvement of synchronization accuracy in ieee 1588 using a queuing estimation method," *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS09)*, pp. 12–16, Oct. 2009.

[13] S. Johannessen, "Time synchronization in a local area network," *IEEE Control Systems Magazine*, vol. 24, pp. 61–69, 2004.

[14] I. Hadzic and D. R. Morgan, "Adaptive packet selection for clock recovery," *ISPCS 2010*, pp. 42–47, 2010.

[15] T. Murakami and Y. Horiuchi, "Improvement of synchronization accuracy in ieee 1588 using a queuing estimation method," *ISPCS 2009*, pp. 1–5, 2009.

[16] Z. Du, Y. Lu, and Y. Ji, "An enhanced end-to-end transparent clock mechanism with a fixed delay ratio," *IEEE Communications Letters*, vol. 15, pp. 872–874, 2011.

[17] D. R. Jesk and A. Sampath, "Estimation of clock offset using bootstrap bias-correction techniques," *Technometrics*, vol. 45, pp. 256–261, 2003.

[18] D. R. Jesk and A. Chakravartty, "Effectiveness of bootstrap bias correction in the context of clock offset estimator," *Technometrics*, vol. 48, pp. 530–538, 2006.

[19] I. Hadzic and D. R. Morgan, "On packet selection criteria for clock recovery," *ISPCS 2009*, pp. 1–6, 2009.

[20] A. Papoulis and S. U. Pillai, *Probability, random variables and stochastic processes*. McGraw-Hil, forth ed.

[21] N. M. Temme, "Uniform asymptotic expansions of the incomplete gamma functions and the incomplete beta function," *Mathematics of Computation*, vol. 29, pp. 1109–1114, 1975.

[22] I.-T. G.8261/Y.1361, "Timing and synchronization aspects in packet networks," April 2008.