ONE WAY ACTIVE DELAY MEASUREMENT WITH ERROR BOUNDS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

TAYFUN EYLEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2015

Approval of the thesis:

ONE WAY ACTIVE DELAY MEASUREMENT WITH ERROR BOUNDS

submitted by **TAYFUN EYLEN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Gönül Turhan Sayan _____
Head of Department, **Electrical and Electronics Engineering**

Assoc. Prof. Dr. Cüneyt Fehmi Bazlamaçcı _____
Supervisor, **Electrical and Electronics Eng. Dept., METU**

Examining Committee Members:

Prof. Dr. Uğur Halıcı
Electrical and Electronics Engineering Department, METU _____

Doç. Dr. Cüneyt F. Bazlamaçcı
Electrical and Electronics Engineering Department, METU _____

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering Department, METU _____

Doç. Dr. İlkay Ulusoy
Electrical and Electronics Engineering Department, METU _____

Prof. Dr. Ezhan Karaşan
Electrical and Electronics Eng. Dept., Bilkent Uni. _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: TAYFUN EYLEN

Signature :

ABSTRACT

ONE WAY ACTIVE DELAY MEASUREMENT WITH ERROR BOUNDS

Eylen, Tayfun
M.S., Department of Electrical and Electronics Engineering
Supervisor : Assoc. Prof. Dr. Cüneyt Fehmi Bazlamaçcı

September 2015, 52 pages

This thesis deals with the problem of measuring the delay of a packet in a network with an associated error bound but without having a need for clock synchronization and for any form of bidirectional messaging between the sender and receiver. A novel lightweight technique is proposed that aims keeping the actual error made in the delay estimation very low while providing simultaneously a good error bound for each individual estimated packet delay. One way delay measurement without clock synchronization and messaging cannot guarantee an error bound on delay estimations in general, however we show that this is possible by using periodic probe packets and appropriate assumptions that are compliant with the physical conditions of the environments within which the sender and receiver operates. Although we calculate an error bound for all our delay estimates, the main purpose is to have a much smaller actual error in these delay estimates in comparison to the computed error bound and to other methods existing in the literature. The proposed method is evaluated against a recently reported technique of the same category and is shown to be much superior overall.

Keywords: One Way Delay Measurement, Delay Estimation, Error Bound

ÖZ

HATA SINIRLI AKTİF TEK YÖNLÜ GECİKME ÖLÇÜMÜ

Eylen, Tayfun
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Cüneyt Fehmi Bazlamaçcı

Eylül 2015 , 52 sayfa

Bu tez alıcı ve gönderici arasında çift taraflı mesajlaşma ve saat eşzamanlaması yapılmaksızın belirli bir hata sınırı içerisinde paketlerin gecikmelerinin ölçümü problemiyle uğraşmaktadır. Çevrim sırasında sürekli olarak maksimum hata miktarını ve düşük bir hata oranını verebilen düşük karmaşıklıkta bir teknik ortaya konmuştur. Herhangi bir varsayım olmaksızın tek yönlü gecikme sistemlerinde alıcı ve gönderici arasında saat eşzamanlaması yapılmadan belirli bir hata sınırı ortaya konamamaktadır. Ancak, belirli varsayımlar ve periyodik paketler yardımı ile ortaya konulan yöntemde hata sınırı verilebilmektedir. Oluşturulan sistemde belirli bir maksimum hata ortaya konarken bir yandan da litaratürde bulunan diğer yapılarla kıyaslanabilecek ve daha iyi olabilecek seviyede düşük hata hedeflenmektedir. Önerilen yapı literatürde bulunan ve aynı kategoride olan güncel bir yapıyla kıyaslanmış ve çok daha iyi sonuçlar verebildiği görülmüştür.

Anahtar Kelimeler: Tek Yönlü Aktif Gecikme Ölçümü, Gecikme Tahmini, Hata Sınırı

To my family

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my advisor Assoc. Prof. Cüneyt F. Bazlamaçcı for his guidance and motivation throughout the study.

I also thank Turkish Scientific and Technological Research Council (TÜBİTAK) for their financial support during my study.

Last but not least, I would like to thank my family for their love and support throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

OWDM	One Way Delay Measurement
GPS	Global Position System
NTP	Network Timing Protocol
OWADME	One Way Active Delay Measurement With Error Bounds
RTT	Round Trip Delay
p	Packet Period
d_k	Delay of The Packet With Sequence Number k,
s	Calculated Skew Value
N	The Maximum Period In Which At Least One Nondelayed Packet Exists
β_j	Exit Angle of Packet j
α_k	Entry Angle of Packet k
m	Size of The Candidate Set
M_{min}	Minimum Size of The Candidate Set
N'	Minimum Time Difference Between Two Nondelayed Packet Intervals
S_n	Maximum Number of Consecutive Nondelayed Packets Within a Nondelayed Packet Interval
f_{NDP}	Nondelayed Packet Time Function

CHAPTER 1

INTRODUCTION

One way delay measurement (OWDM) is the task of measuring the delay of a packet in a network between two end points. In the literature, there exist many methods for OWDM, which employ various tools and techniques to obtain accurate measurements to a certain extent. Some of the existing methods try synchronizing the sender and receiver clocks. By employing a time attribute for each packet, the measurement of delay becomes an easy task when clocks are synchronized. The challenge in this approach is in the synchronization of the sender and receiver clocks. Use of global position system (GPS) or messaging between the sender and receiver are two of the methods that are used to synchronize the communicating parties. Other than using direct synchronization, there are also techniques that try to measure the delay of a packet by using other properties such as periodicity in packet arrivals.

Many classifications for OWDM are possible as in [1]. To show where we stand, we first divide all existing methods into two main categories: 1) with or 2) without clock synchronization. In fact, all methods for OWDM can be regarded as an explicit or an implicit synchronization scheme. The techniques in without-clock-synchronization category aim to measure the delay of a packet without altering the frequency or offset of a node's clock. Many classifications view OWDM as a clock synchronization technique. Hence, techniques with and without clock synchronization can also be named as hard (explicit) and soft (implicit) clock synchronization methods. GPS [2], NTP (network timing protocol) [3] and the IEEE 1588 [4] standard are among the methods that are with clock synchronization. Sync & Sense [5] and Vakili & Grégoire [6] methods are two examples without clock synchronization. Another classification

can be made as: 1) with or 2) without messaging between the sender and receiver. NTP and the IEEE 1588 standard are examples with messaging and Sync & Sense is an example without messaging. The present thesis proposes a novel technique for OWDM that falls into the category of without clock synchronization and without messaging between the sender and receiver. Since Sync & Sense and our proposal are in the same category, both are compared and evaluated against each other in detail in the present work.

1.1 Aim of The Thesis

In any network that employs OWDM, the difference between the measured delay and the actual delay, i.e., the error in measurement, is among the major concerns. In this work, we aim to minimize the error in delay measurement while providing simultaneously an error bound for each individual estimated packet delay. We propose a novel technique that uses periodic probe packets to estimate the queuing delay of each individual packet in a network. OWDM without clock synchronization and without messaging can normally not guarantee any error bound; however, we show that using appropriate assumptions about clock characteristics and congestion, one can provide a guaranteed error bound for the queuing delay estimate of each packet and that such a bound is also comparable with the actual error. The assumptions about clocks are chosen in compliance with the physical conditions in the sender and receiver environments and the assumption about congestion is also easily applicable in return for a decrease in utilization. While providing an upper bound for possible errors in our delay estimates, we aim to have a much smaller actual error compared with the calculated bound.

1.2 Content of The Thesis

The rest of the thesis is organized as follows.

In section 2, a brief review of existing OWDM techniques is given.

In section 3, the proposed one-way active delay measurement with error bounds

(OWADME) approach is presented in detail.

Section 4 includes a complexity analysis of the given algorithm.

In section 5, the results of our extensive simulation work, which is carried out to evaluate the performance of OWADME and to compare it with the most recent proposal of the same category, is presented.

In section 6, active measurement method is discussed and traffic cost with respect to measurement precision is analyzed.

Section 7 summarizes the work and concludes the thesis by including new directions for possible future works.

CHAPTER 2

BACKGROUND INFORMATION AND LITERATURE OVERVIEW

We start by stating the following key definitions used in delay measurement literature [7, 6, 8, 9, 10]:

Clock offset is the difference between two clock values (the unit is seconds representing time).

Clock skew is the time derivative of offset (the unit is parts per million), which is also regarded as the frequency difference between two clocks.

Clock drift is the time derivative of skew (the unit is parts per million per second).

2.1 General Classification and Delay Components

The one way delay of a packet is the difference between the time the first bit of the packet is sent and the time the last bit of the same packet is received. In some applications, a packet can be time stamped at the sender when the first bit is sent and this time stamp information can be sent along each packet. The packet can also be time stamped at the receiver when the last bit is received. Delay measurement in such a case is as easy as subtracting the arrival time stamp value from the departure one if both the sender and receiver clocks are kept synchronous. Reference [10] categorizes clock synchronization techniques for OWDM in a different way. It first makes a distinction between external-server-based synchronization techniques and end-to-end synchronization techniques. External-server-based techniques require a centralized

time source and use of GPS [2], IEEE 1588 [4] and NTP [3] are examples. End-to-end techniques realize synchronization with the help of network measurements. There are two categories for end-to-end synchronization: online and offline. Offline techniques cannot be applicable in real-time and they mostly deal with line fitting problems for obtained measurements. Online techniques try to predict current clock skew using past measurements. Our method OWADME falls into end-to-end and online category according to this classification.

When a packet is transmitted from a sender to a receiver, the packet is delayed due to various reasons in the communications channel. The delay of a packet is composed of processing delay, queuing delay, transmission delay and propagation delay.

Processing delay is the time interval in which the packet header is processed. It depends on processing power of the sender and receiver sides and the algorithms used in corresponding applications. Most of the time, it is very small compared to other delay components.

Transmission delay is the required time for putting the whole packet onto the wire. It depends on the data transmission rate of the link. There may be small fluctuations on the data rate but most of the time it can be accepted to be a constant.

Propagation delay is the time a bit spends on the wire (or main transmission medium) during its travel from the sender to receiver side. It can be computed using the distance and the speed of the wave in the corresponding medium. For one hop links, for which the ambient operating conditions do not change abruptly, it can be accepted as a constant and small value.

Queuing delay is the most variable delay segment compared to others. For congested cases, it can be the highest.

Without synchronization attempt or without having a two-way messaging scheme, the delay components that are constant cannot be observed. Hence the main concern in this thesis is to measure the variable queuing delay. Other delays such as transmission delay are assumed to be constant therefore, any fluctuations on these will result in additional errors. In the present thesis, such error cases are not considered; hence, the proposed method is regarded as one-way queuing delay estimation.

2.2 Well Known OWDM Methods

GPS is a space-based satellite navigation system that provides location and time information. In order to have the time information, a node needs to connect to at least four satellites. If a node fulfills all GPS requirements, its clock can be synchronized with a maximum error of 1 μs and this synchronization can be repeated every second. Using a GPS receiver at every node in a network, all nodes may be kept synchronized to each other within a certain error margin and by also reserving a time stamp field at each packet, one can perform OWDM easily. However, there is a downside to use GPS for OWDM. One is the cost of employing a GPS receiver at each node. For mobile or indoor use cases, connection with at least four satellites may be a problem. In certain military applications, GPS independence may be a strict requirement.

NTP [3] is a network protocol, which also aims to synchronize the clocks within a network but without a need for any additional equipment such as GPS. It creates a virtual tree to distribute the most reliable clock in the network. Accuracy of NTP depends on the size of the network. For local area networks it can synchronize clocks with a maximum error of 1 ms. For wide area networks, however, $10 - 100$ ms measurement errors are possible.

IEEE 1588 standard [4] is another networking protocol that aims node synchronization. Messaging between network nodes makes it possible to synchronize each node. There is a self-constructed master-slave relation between each node. Synchronization information is distributed with a 2 s period. There exists some additional equipment for the IEEE 1588 standard that can be used to increase the accuracy of synchronization. Without additional help $10 - 100$ μs errors can be achieved. With the use of extra equipment maximum error bound can be decreased to and below 1 μs.

2.3 OWDM Methods Without Hard Synchronization

There are other techniques that do not need external-server-based synchronization. In such techniques, some other parameters are used for estimating the delay. For example, round-trip-time (RTT) is used for OWDM for a symmetric network topology, but

most of the time assuming a symmetric topology is not possible. In other methods, error in delay can be decreased by interchanging and using RTTs of multiple nodes [11, 9]. These methods eliminate the need for a symmetric network topology. One way delay can be found by applying a least squares error like method to multiple RTTs [6]. Multiple RTTs method has a much higher error rate in comparison to GPS based technique and the IEEE 1588 standard.

In [12], a OWDM scheme is constructed to increase the performance of network control systems. For more accurate results, the system combines an end-to-end estimation method and an online monitoring mechanism. Using a beacon packet in a loop, RTT values are obtained consecutively. These RTT values are derived in such a way that by only knowing the first packet's forward delay, network status and hence current OWD can be estimated. The first packet's forward delay is predicted by online monitoring.

Another approach that eliminates the need for hard clock synchronization or external-server-based synchronization is to use periodic packets. If the sender side emits packets with a constant period and the receiver knows this period, OWDM can be performed using a simple assumption, which states that there exist some packets without any delay in a certain time window. Using this assumption and the packet period, arrival time of each packet can be calculated following the reception of the first zero-delay packet. However, the existence of *skew* and *drift* complicates such measurement techniques. With some additional assumptions, delay estimation can be made with a low actual error level comparable to methods that perform hard clock synchronization. Sync & Sense [5] is the most recent and main example of this category of methods. In the present thesis, a similar approach is adopted, but a much better technique, both in terms of performance and attributes, is developed, analyzed and evaluated thoroughly against Sync & Sense.

As categorized in [10], online and offline skew estimation methods exist for end-to-end synchronization. Online skew estimation methods are similar to offline methods. However, online methods need to predict the current skew value in real time. To achieve this, the number of samples is reduced by using a skew estimation technique [13] and named estimation of skew with reduced samples. For any pair of successive

packets, if the interarrival time between packets is larger than an expected value, the latter packet is excluded from the measurement set. This methodology is similar to the filtering mechanism we propose in the present thesis. In [13], a constant ratio of 1.5% specifies the boundary of the interarrival time. On the other hand, our proposal uses maximum and minimum skew values and calculates the boundary of the interarrival times. Moreover, application of an additional boundary using drift is also helpful in reducing the number of samples. We believe that considering skew as a constant is a weak point in [13] and fix this in our approach.

The approach in [8] is also similar to the technique we propose in the present thesis. In order to estimate next packet's one way delay, a skew value is predicted. For this purpose, the least delayed packet within a specified period is searched. The search mechanism uses the interarrival times of packets. Following the selection process, line fitting is performed and a constant skew value is obtained. One downside of [8] is its acceptance of skew as a constant. Another one is that it has no assumption to ensure that the least delayed packet within the specified period to have zero delay. As a result, it is possible for the line fitting scheme to produce an incorrect prediction. There are other methods that employ line fitting schemes for skew estimation (for example [16]).

2.4 OWDM Method Using Physical Conditions

A relatively recent study on OWDM relates temperature to clock skew [14] and shows that both are highly correlated. Observing this correlation, the authors present an environment-aware clock synchronization (EACS) scheme. Using EACS, they aim to reduce the number of two-way messages between the sender and the receiver, and for this purpose, they follow the change in temperature. For small changes, they compute corresponding skew value using linear equations and an information-aided multimodel Kalman filter (AMKF). For relatively bigger changes, they apply a resynchronization process involving two-way messaging. Reduction in the number of such resynchronization processes is shown to reduce the energy consumption for wireless sensor networks. Another method, which reduces the number of synchronizations by following the temperature changes similarly, is presented in [15].

Reference[14] uses two temperature models, one constant and one variable, for those situations where temperature changes very slowly. In constant one, there exists a pre-set skew value corresponding to each range of temperatures, whereas in the variable one, a linear approximation is used to reduce complexity while providing a sufficient level of accuracy. Reference [14] assumes that clock skew is mainly affected by temperature. Other environmental effects such as humidity and shock are considered to be measurement noise. The experimental results in [14] vindicate the above models and assumptions, but for non-stable environmental conditions, the effect of humidity and shock can be substantial and may hinder the use of findings in [14].

Table 2.1: Comparison of OWDM techniques

Refs/Attibutes	Hard Synch	Soft Synch	2-way Comm.	Periodic Packets	External Tool Dependance	Skew Estimation	Drift Estimation	Error Bound	All-time Availability	RTT Usage
GPS [2]	x				x					
NTP [3]	x		x						x	
IEEE 1588 [4]	x		x		x				x	
Yang [14]	x		x			x		x	x	
Vakili [6]		x	x						x	x
Xu [12]		x	x						x	x
Sync & Sense [5]		x		x		x			x	
Aoki [8]		x				x			x	
Kim [11]		x	x			x			x	x
OWADME (this thesis)		x		x		x	x	x	x	

CHAPTER 3

ONE WAY ACTIVE DELAY MEASUREMENT WITH ERROR BOUNDS (OWADME)

The proposed method is named OWADME to highlight that it uses active probe packets for measuring packet delay, and it also provides additional information about the maximum error possible associated with each delay estimate made. Our method developed according to the requirements given in Table 2.1 does not involve any hard synchronization or two-way messaging between the sender and receiver. The only requirement at the receiver side is to keep track of arrival times of incoming packets. Hence, it is unique in its specification and attributes and is applicable in many situations, such as sensor networks, due to its possibility of very lightweight implementation.

Packet period is assumed to be known as an $apriori$ parameter. The key assumption in this approach is the existence and hence the arrival of some nondelayed packets from time to time. If a nondelayed packet is received and the packet period is known, delay of those packets that follow the nondelayed one can be found easily by using the difference between the expected and the observed arrival times (Fig. 3.1). But in this type of measurement, an error may accumulate and might grow without bound due to possible clock skew and drift. The difference between the frequencies of two clocks is named *clock skew* and the change in clock skew over time is named *clock drift* as was stated earlier. For example, if skew is 100 ppm, there occurs a 1 s error after 10,000 s and the past nondelayed packet becomes useless. Therefore, a new nondelayed packet is needed within a large period of time. To be able to sustain such a measurement system, the system must use, for example, a packet dropping

mechanism to create some nondelayed packets in the network. Following the arrival of nondelayed packets, the receiver identifies them and computes the delay of all other packets in the network using the approach described in this section.

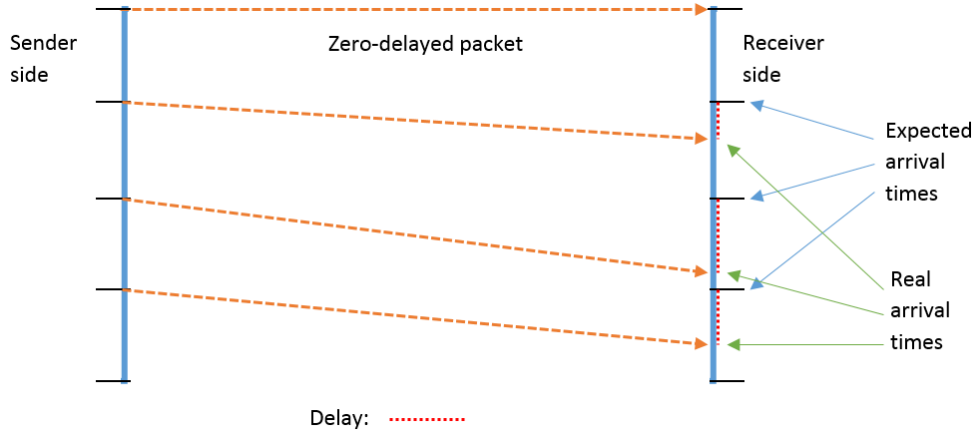

Figure 3.1: Communication between the sender and receiver

Our methodology can be viewed as four subsequent tasks that are repeatedly performed each time a packet arrives at the receiver:

1. detection of nondelayed packets within a past window of received packets (filtering)
2. estimation of clock skew using nondelayed packets (skew estimation)
3. estimation of individual packet delay (delay measurement)
4. calculation of an error bound associated with the delay estimate made (error analysis)

Using the filtering scheme, a set composed of candidate nondelayed packets that are received within a past window of time is constructed. This set is named candidate set in the rest of the thesis. Some of the packets in the candidate set may of course have some nonzero but small delays. Then clock skew is estimated using the packets in the candidate set. With a current estimate of clock skew, delay of each incoming packet can be calculated easily. Afterwards, an error analysis is possible providing an associated error bound for each estimated packet delay.

3.1 Filtering

This subsection presents the task of constructing the candidate set, which is composed of candidate nondelayed packets that are received within a past time window.

The only information the receiver has about a packet is its arrival time. If all periodic packets arrive without any delay, arrival times form a function (nondelayed packet time function - f_{NDP}) that shows the behavior of sender side clock with respect to the receiver side clock (dotted red line in Fig. 3.2.b). To be able to determine the exact delay for new packets, this behavior should be taken into account. In our filtering mechanism, the receiver side aims to identify such nondelayed packets by eliminating the delayed ones in order to observe the behavior of clocks. Two main assumptions are used in filtering out the delayed packets:

- existence of a bound on clock skew
- existence of a bound on clock drift

Our filtering effort incorporates the comparison of each packet arrival time to others using these two assumptions. If an arrival time cannot be on the zero-delay packet arrival time function, it is excluded from the candidate set. In our approach, filtering with respect to skew bound and drift bound are employed consecutively.

3.1.1 Filtering with respect to skew bound

Our first assumption for filtering is that there exists a bound on clock skew. This assumption is derived from the physical conditions of the environment in which the two clocks operate. If two identical clocks having identical physical properties are used in two different operating environments having different temperatures, the skew that occurs between the two clocks can be computed. The function for calculating skew is also assumed to be differentiable. Therefore, bounded temperature difference results in bounded skew. If the maximum of all probable differences between the frequencies of the two clocks is denoted by d_{max}, a positive and non-zero skew denoted as c can be computed by using only d_{max}. c represents an upper bound for all probable skew

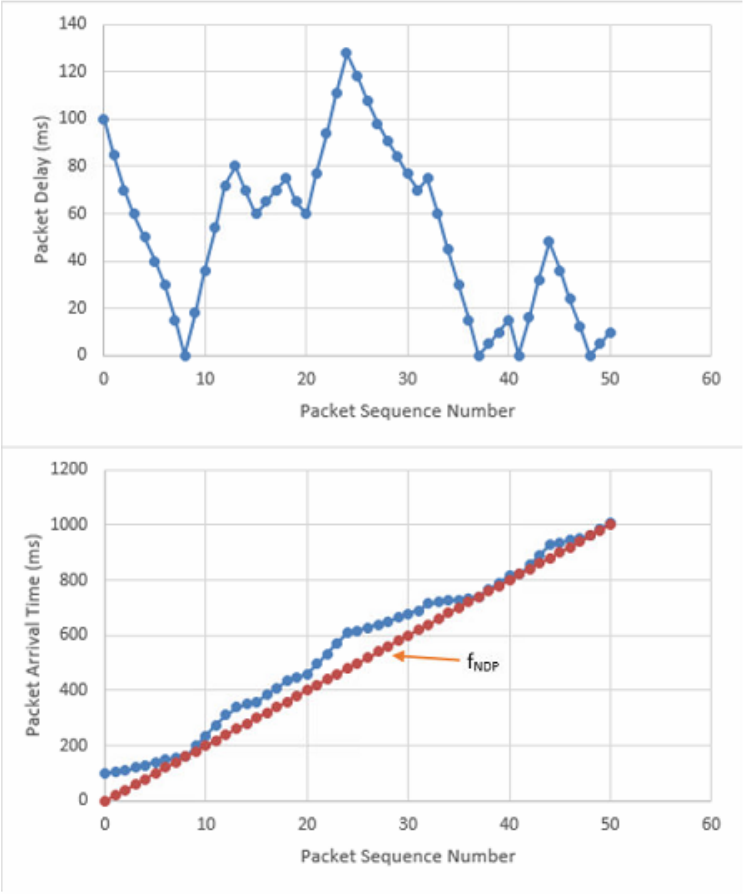

Figure 3.2: (a) Typical pattern of the one-way network delay (b) Corresponding plot of arriving packets with respect to packet sequence number [5]

values. If the period of packets is p with respect to the receiver side, the sender side might in fact send packets with an actual period p_a where $p - c < p_a < p + c$.

For two packets having sequence numbers i and j, let t_i and t_j denote their arrival times, respectively. If

$$t_j - t_i > (j - i) \times (p + c) \tag{3.1}$$

one can safely conclude that the latter packet j has a nonzero delay. Similarly, if

$$t_j - t_i < (j - i) \times (p - c) \tag{3.2}$$

then the preceding packet i has a nonzero delay. Therefore, any packet proven to have a nonzero delay may be discarded from the candidate set that holds nondelayed packets only.

Following eliminations of the above type, the candidate set of nondelayed packets has all of the certainly nondelayed packets plus some other ones that are subjected to a relatively small delay.

3.1.2 Filtering with respect to drift bound

For filtering with respect to drift bound, a similar assumption to filtering with respect to the skew case is used. The assumption is that there exists a bound on clock drift. Similar to skew, physical conditions affect drift. A bound on the rate of change in physical conditions determines the bound on skew values. Similarly, a bound on the rate of change in physical conditions determine the bound on drift values. However, rather than using linear inequalities as we did in skew filtering, we prefer using second-order piecewise inequalities in our filtering mechanism while utilizing our assumption about the existence of a bound on drift.

For a more detailed exposition, we start by considering the relation between any pair of packets. After applying filtering with respect to skew bound, the remaining packets satisfy the relation Eq. 3, which impose a time bound for a packet to be considered as nondelayed and hence to be in the candidate set. As a result of having a bound on skew, a packet in the nondelayed packet set imposes a time bound for the latter packets for them to be considered as nondelayed. Hence, for any two sequence numbers i and j, if the corresponding packets p_i and p_j are in candidate set that is already filtered with respect to skew then

$$t_i + (j - i) \times (p + c) \geq t_j \geq t_i + (j - i) \times (p - c) \tag{3.3}$$

holds true for $j > i$.

The above inequality states the linear relationship between any pair of packets in the candidate set obtained after using skew filtering. Drift filtering adds second order parameters in addition and complicates the inequality slightly. Let the function f denote the nondelayed packet arrival times. Then f should satisfy the following conditions:

$$skew_{min} < d(f)/dt < skew_{max} \tag{3.4}$$

$$drift_{min} < d^2(f)/dt^2 < drift_{max} \tag{3.5}$$

In other words, first and second order derivatives of f with respect to time should be bounded. Moreover, if a packet p_i is a nondelayed packet having sequence number i and arrival time t_i, there exist an f_i such that

$$f_i(i) = t_i, f_i(j) \leq t_j, \forall p_j \tag{3.6}$$

The equality above states that p_i is a nondelayed packet and hence, it is on f_i as illustrated in Fig. 3.3. It is also possible to find an approximation of the nondelayed packet function f_i by considering a set of packets received within a time window only. For a particular p_i, if a function that satisfies the above constraints can be found, we can treat p_i as a nondelayed packet among the current window of received packets. With the arrival of a new packet, this may change, and hence, the validity of p_i to be in the candidate set should be checked each time a new packet passes skew filtering.

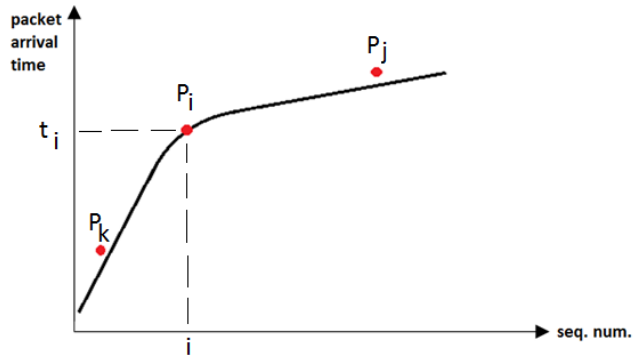

Figure 3.3: A function of possible nondelayed packet arrival times corresponding to p_i

The existence of function f_i satisfying the above constraints can be checked by using entry and exit angle concepts. We define the *entry angle* of packet p_i to be the minimum angle with respect to x-axis (sequence number) of the function on f_i at i, which satisfies the above constraints for all packets in the window but received earlier than i. Similarly, we define the *exit angle* of packet p_i to be the maximum angle with respect to x-axis of the function on f_i at i, which satisfies the above constraints for all packets in the window but received after i (see Fig. 3.4).

If a packet p_i is to remain in the candidate set, its exit angle should be larger than or equal to its entry angle. When a new packet arrives and passes skew filtering, all

packets in the candidate set should be checked using this inequality. However, since there exists no preceding neighbor for the very first packet in the window, it cannot be checked as described. Similarly, filtering with respect to drift cannot be applied to the last packet in the time window. Therefore, the error bound to be calculated for the last packet's delay estimate may be higher in comparison to error bounds of other estimates.

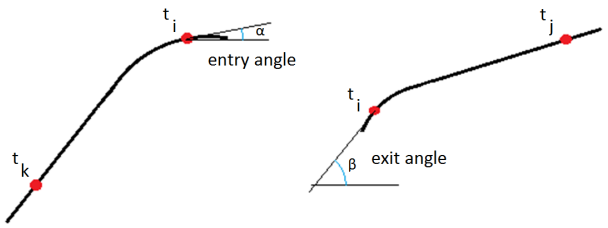

Figure 3.4: Entry and exit angles of packet p_i

3.2 Skew Estimation

The main purpose of skew estimation is to form f_{NDP} (nondelayed packet time function). With the help of our filtering mechanism, the size of the candidate set is reduced. This minimized set includes all the nondelayed packets and some delayed ones. A packet in the candidate set cannot be labeled as nondelayed or delayed with full confidence. As a result, we treat all packets in the set as nondelayed and use them in the construction of f_{NDP}.

It is worth noting that f_{NDP} is not a first order line if drift is nonzero. The rate of change in skew, i.e. drift, is a small value in general. As a result, most methods in the literature make oversimplifications and accept f_{NDP} as a line to have lower computational cost in exchange for higher error. Our technique treats f_{NDP} as a varying function and we try to estimate the most up-to-date skew value. To increase comprehensibility, we exaggerate the rate of change in skew in figures 3.3 and 3.4. Other then these, f_{NDP} is shown as a nearly straight line throughout the rest of the text.

Even after employing our candidate set size reduction technique, there still exists an

infinite number of possible f_{NDP}'s but within some bounds, thanks to our assumptions. To be able to measure the delay of a packet, one of these possible f_{NDP}'s should be selected. We name this process as skew estimation. The skew estimation should not be a random choice within the existing bounds but should be a decision based on closeness and being up to date.

Any pair of packets in the candidate set implies a possible skew value and we prefer to use a weighted averaging scheme using all packets in the candidate set. We determine the weight of a possible skew line (value) passing through a given pair of packets using some simple rules that are described in the following paragraphs.

There may be more than one consecutive nondelayed packet even in a very small time duration within our time window of processing (see Fig. 3.5). We therefore reduce the size of the candidate set further by merging some of the packets before applying our skew estimation rules.

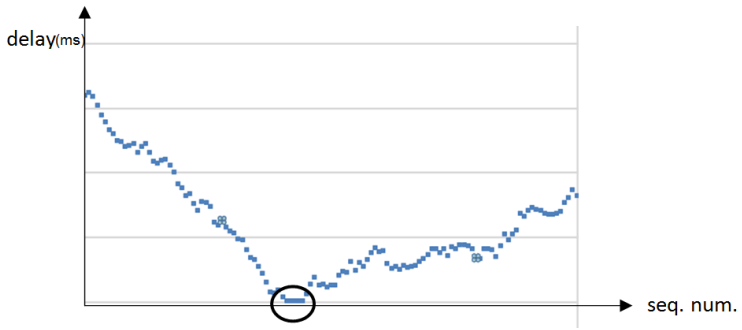

Figure 3.5: Packet merging for skew estimation

If a number of consecutive packets having very small delays exist in the candidate set and weighted pairs of such packets are to be used in skew estimation, a negative bias will be observed. For such pairs if the time difference (difference in sequence numbers) is small compared to other pairs, the error in the estimation will be larger. To eliminate such a bias, we merge those packets that have small delay and are sufficiently close to each other into a single packet to which we also associate its average delay and the number of packets used in its formation. Such a merging is a heuristic process, which can alternatively be phrased as follows: if the time difference, the difference in sequence numbers between a pair of received packets, is smaller than

the radius or merging, r, then the two packets are eligible to be merged. In this manner, the choice and tuning of r defines what *sufficiently close to each other* means. Following this merging process, the candidate set will now be composed of relatively distant pairs.

For a correct skew estimation, pairs used in computing the final average skew line (value) should be up to date to a certain extent. Therefore, we measure the time difference between a pair and the most recently acquired packet. In other words, for each packet pair (i, j), time difference $t_{ij}^k = t_k - (t_j + t_i)/2$ is computed where k is the most recent sequence number, hence k^{th} iteration.

We then use t_{ij}^k attributes in an inversely proportional way in forming the weights of possible skew lines (values). As a result, the most valuable pair becomes the last packet and the one before it. Since our proposed filtering with respect to drift mechanism does not work for the last packet also, we prefer to ignore the most recently received packet in our skew estimation process.

In finding the weights, we also employ the number of packets merged in forming a new packet in the candidate set. Due to our filtering mechanism, closer packet arrival times tend to have less error in comparison to distant ones. The weight of any pair associated with a merged packet is scaled using the number of times it is merged in its formation. For any pair with sequence numbers i and j, weight of the corresponding skew value is found as

$$w_{ij}^k = (1/t_{ij}^k) \times n_i \times n_j \tag{3.7}$$

where t_{ij}^k is the time difference corresponding to (i, j) pair and the newest packet k and n_l is the number of times merging is applied in the formation of packet l.

Using the above weights, an overall average skew value is then calculated at the k^{th} iteration as follows:

$$s^k = (\sum_i \sum_j (s_{ij}^k \times w_{ij}^k))/(\sum_i \sum_j w_{ij}^k) \tag{3.8}$$

3.3 Delay Measurement

Following the skew estimation process, we search for the closest packet to the computed skew value. For this, we draw an auxiliary line with an arbitrary starting point and with a slope as the calculated skew. Delays of all packets with respect to this auxiliary line are then calculated and the least delayed one is selected. The delay of the least delayed packet with respect to the auxiliary line can be negative. For such cases, the delay having the largest magnitude should be selected. Assuming that the sequence number of this selected packet (packet closest to auxiliary line) is x, the delay of a newly arrived packet k can then be found as follows:

$$d_k = t_k - (t_x + (k - x) \times (p + s)) \tag{3.9}$$

where

d_k: delay of new packet k,

k: sequence number of new packet,

x: sequence number of selected packet,

t_k: arrival time of new packet k,

t_x: arrival time of selected packet x,

p: packet transmission period,

s: calculated skew value.

3.4 Error Analysis

The delay calculated using the above OWDM technique contains an uncertainty because the receiver does not know a packet's exact departure time from the sender. Even though some packets are assumed to have zero delay, we still cannot determine a packet to be certainly a nondelayed one at the receiver side without any additional messaging. However, if it is known that at least one nondelayed packet exists within a predetermined period of time then the necessary conditions to assign a maximum

error bound for the predicted delay can be obtained. The above assumption, which simply states that there is a period in which at least one nondelayed packet exists, is necessary for our error analysis proposal presented in the following. To satisfy this assumption, one can use a mechanism to control the traffic on the sender side to be able to empty the sender's queue and intermediate routers' queues (in multihop transmission) within the specified period.

It is worth noting that our scheme itself does not attempt to control congestion but it rather assumes that there exists a mechanism such that within a predetermined period of time at least one packet is guaranteed to be delivered with zero delay. This is not a must for our proposed delay measurement technique, but for a necessary condition to be able to give an error bound for the queuing delay we measure. There exists many congestion control algorithms proposed in the literature, which aims to achieve a certain tradeoff between high resource utilization and low-delay requirements. There are methods designed for one-hop or multihop connections. These methods may or may not need two-way messages.

The technique proposed in this thesis is directly applicable to a one-hop link because the assumption that we have stated above can easily be satisfied, for example, by accessing the sender's outgoing queue and dropping all waiting packets at the end of each such period. This of course brings utilization overhead, a penalty to pay in return for a guaranteed error bound on the delay measurements to be made. In cases where accessing sender's queue is not possible, an alternative to satisfy the above assumption may be to use a periodic input rate adjustment scheme to be able to empty the sender's outgoing queue. Such an approach also reduces utilization in return for an error bound on delay measurements. How one satisfies the required assumption in the best possible way is out of the scope of the present work.

If a suitable multi-hop mechanism that satisfies the above stated assumption is designed in an end-to-end fashion, maybe in return for a loss in overall utilization, our idea can also be extended to multihop cases. Otherwise, our proposal can be applied to all one-hop links in a multihop path separately to find the total delay and the error in an additive manner. This approach may deteriorate the tightness of the error-bound and may reduce utilization further in order to allocate a delay field in each packet

transmitted.

In our algorithm, a skew value is estimated and a packet that fits the estimated skew value best is selected to measure the delay. In fact, there is a range of possible skew values. In our error analysis, we consider all possible skew values and all suitable packets that fit these skew values and observe a range of delay values for an arriving packet. With the above assumption, if the difference between the arrival time of the oldest packet in the set and the current time is larger than the specified period, it is guaranteed that the range of delays includes the actual delay. The specified period is named N and the size of the set is calibrated to have at least one packet that have a time difference larger than N.

For a newly arrived packet, the range of possible delay values may be found by using packets that are in the candidate set, but which arrived within a past time window of N. Among all possible packets in the specified range, it is sufficient to select the oldest and the newest ones to obtain an error bound for the estimated delay. Fig. 3.6 illustrates the concept.

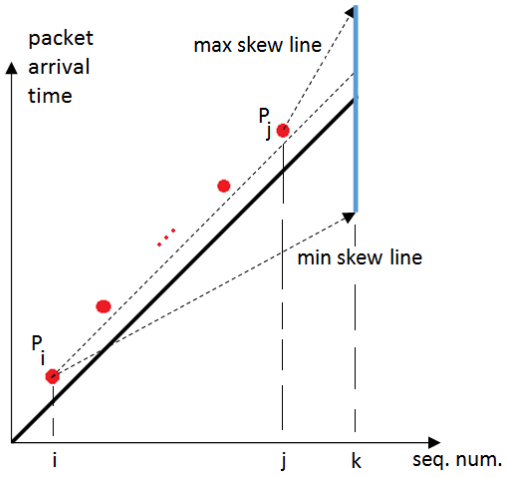

Figure 3.6: Range of possible skew lines (error bound)

Dashed lines in Fig. 3.6 represent $skew_{max}$, no skew and $skew_{min}$ lines. i represents the oldest packet within the past period of N and j represent the newest packet in the candidate set. Let k be the sequence number of the new packet, then the actual skew

value $skew_k$ at time k can be characterized as:

$$t_i + (k-i)(skew_{min} + p) < skew_k < t_j + (k-j)(skew_{max} + p) \tag{3.10}$$

Since $k = j + 1$

$$t_i + (j-i+1)(skew_{min} + p) < skew_k < t_j + (skew_{max} + p) \tag{3.11}$$

From the above range of possible nondelayed packet locations at k, a corresponding delay range for the new packet k can be deduced easily as illustrated in Fig. 3.6.

Algorithms 3.1-3.7 presents the whole idea described in the present section in a compact manner.

If the probe packet period were unknown, it would be impossible to obtain an error bound for the delay of each packet received. Knowledge of the packet period and existence of bounds on clock skew and drift enable us to implement the filtering mechanisms we propose. If a fixed packet period does not exist, the problem reduces to prediction under uncertainty, which we do not aim for in the present work at all.

Clock skew and jitter are similar concepts. In fact, jitter can be regarded as the general form of clock skew. Jitter can be caused by electromagnetic interference (EMI) and crosstalk of other signals and the frequencies of a pair of clocks can change abruptly. We keep such abrupt frequency differences out of the scope of the present work and limit our problem by using two main assumptions on skew and drift. If EMI and crosstalk can be defined as continuous functions similar to temperature and humidity, jitter can be included in our technique. Otherwise, the existence of jitter adds an error to our delay measurement and error bound but to also other similar techniques that exist in the literature.

Algorithm 3.1: Overall Algorithm (OWADME)

1 Wait for a new packet
2 Apply skew filtering
3 Apply drift filtering
4 Apply skew estimation
5 Find the closest packet to estimated skew line (value)
6 Compute the delay for the new packet
7 Compute the error bound for the new delay estimate
8 **goto** step 1.

Algorithm 3.2: Skew Filtering Algorithm

1 arrival of a new packet k
2 $j \leftarrow$ *the most recent packet in the candidate set*
3 **if** $t_k - t_j > (k - j) \times (p + c)$ **then**
4 delete p_k and exit
5 **end**
6 **while** $t_k - t_j < (k - j) \times (p - c)$ **do**
7 delete p_j
8 $j \leftarrow$ *the most recent packet in the candidate set*
9 **end**

Algorithm 3.3: Drift Filtering Algorithm

1 arrival of a packet k that passed skew filtering
2 $j \leftarrow$ *the most recent packet in the candidate set*
3 calculate entry angle α_k
4 calculate exit angle β_j
5 **while** $\alpha_k > \beta_j$ **do**
6 delete p_j
7 $j \leftarrow$ *the most recent packet in the candidate set*
8 calculate exit angle β_j
9 **end**

Algorithm 3.4: Skew Estimation Algorithm

1 arrival of a packet k that passed drift filtering
2 $p_{first} \leftarrow k$
3 $m \leftarrow 2$
4 $p_{second} \leftarrow m^{th}$ *most recent packet in candidate set*
5 $l \leftarrow a * period$ where $a \leq 10$
6 **while** $(t_{pfirst} - t_{psecond} < l) \wedge (m <$ *size of candidate set*$)$ **do**
7 merge p_{first} and p_{second} by creating p_m
8 $p_{first} \leftarrow p_m$
9 $m \leftarrow m + 1$
10 $p_{second} \leftarrow m^{th}$ *most recent packet in candidate set*
11 **end**
12 **if** $(t_{pfirst} - t_{psecond} \geq l) \wedge (m + 1 <$ *size of candidate set*$)$ **then**
13 $p_{first} \leftarrow p_{second}$
14 $m \leftarrow m + 1$
15 $p_{second} \leftarrow m^{th}$ *most recent packet in candidate set*
16 **goto** step 5
17 **end**
18 calculate skew and weight value for all pairs
19 calculate the weighted average s

Algorithm 3.5: Closest Packet Search Algorithm

$s \leftarrow$ *weighted averaged skew value*
$p \leftarrow$ *oldest packet in candidate set*
$d \leftarrow 0$
$m \leftarrow 2$
$p_m \leftarrow m^{th}$ *most elderly packet in candidate set*
while $m <$ *size of candidate set* **do**
    $s_p \leftarrow$ *sequence number of packet* p
    $s_m \leftarrow$ *sequence number of packet* p_m
    if $d \geq t_{pm} - t_p - (s_m - s_p) * (period + s)$ **then**
        $p \leftarrow p_m$
    end
end
return p

Algorithm 3.6: Delay Computation Algorithm

arrival of a packet k
$s \leftarrow$ *weighted averaged skew value*
$p \leftarrow$ *closest packet*
$s_k \leftarrow$ *sequence number of packet* k
$s_p \leftarrow$ *sequence number of packet* p
return $t_k - t_p - (s_k - s_p) * (period + s)$

Algorithm 3.7: Error Bound Computation Algorithm

arrival of a packet k
$p_n \leftarrow$ *oldest packet within past period of* N
$s_k \leftarrow$ *sequence number of packet* k
$s_n \leftarrow$ *sequence number of packet* p_n
max error $\leftarrow t_k - t_{pn} - (s_k - s_n) * (period + maxskew)$
min error $\leftarrow t_k - t_{pn} - (s_k - s_n) * (period + minskew)$
return $maxerror$ and $minerror$

CHAPTER 4

COMPLEXITY ANALYSIS

OWADME is required to be fast. It should be completed in at most one period p. The present section discusses its complexity in detail.

4.1 Complexity of Filtering with respect to Skew

Filtering with respect to skew is the process of checking whether a received packet is a candidate to be a zero-delay packet or not. This check is made using the newly arrived packet and the previously received one (the most recent packet in the candidate set). Let i and k denote the sequence numbers of the last packet in the candidate set and the new arrival, respectively. Then the following inequalities are checked in turn:

1. $t_k > t_i + (k - i) \times (p + skew_{max})$
2. $t_k < t_i + (k - i) \times (p + skew_{max})$ and $t_k > t_i + (k - i) \times (p + skew_{min})$
3. $t_k < t_i + (k - i) \times (p + skew_{min})$

If the first inequality holds then the new packet k certainly has a delay and it should be kept out of the candidate set. If the second inequality holds then the new packet k is potentially a zero-delay packet and hence it should be kept in the candidate set. If the third inequality holds then the oldest packet i in the candidate set certainly has a delay but the new packet k may or may not have delay. Hence packet i is dismissed from the candidate set and the new packet k is inserted instead. All these three tasks are O(1) but if packet i in the candidate set is dismissed then this check should be

repeated for the preceding packet in the candidate set. Such a check may continue until all packets in the candidate set except the new packet k are dismissed. Hence for a new arrival step 3 may be repeated for m times, where m is the cardinality of the candidate set during the corresponding iteration.

But for m consecutive packets, the complexity of the process becomes $O(m)$ because every dismissing step reduces the size of the candidate set.

4.2 Complexity of Filtering with respect to Drift

After filtering with respect to skew step, filtering with respect to drift process follows. In this type of filtering, entry and exit angles are to be found. For an entry angle measurement there is no need to use all packets of the candidate set if it is not necessary. For a new packet that passes through skew filtering, the measurement of an entry angle starts with the preceding packet. If this angle is larger than the preceding packet's entry angle, measurement is stopped and the function is assumed to have the entry angle of the new packet. If it is not, it means that the exit angle of the preceding packet cannot be larger than the entry angle of its own. So, the preceding packet is dismissed and the process continues using the packet before the preceding one until all packets in the candidate set are handled.

The algorithm explained above has a complexity similar to filtering with respect to skew mechanism. For a new packet, it has a worst case complexity of m. However, every dismissal process reduces the size of the set and trigger another check therefore for m subsequent packets the complexity becomes $O(m)$.

4.3 Skew, Delay and Error Bound Estimation Complexity

In skew estimation step, every possible pair in the candidate set are considered in $O(m^2)$ time. Then weighted average of these skew values is computed as in Eq. 8 and the closest packet to the estimated skew line (value) is selected. The latter two tasks run in $O(m)$ time. Finally, the oldest and the newest packets from the period N are selected for error analysis with a time complexity of $O(1)$.

4.4 Total Complexity of OWADME

The worst case time complexity of the proposed algorithm OWADME, which is performed each time a new packet arrives, is $O(m^2)$. m is dependent on the characteristics of the packet traffic but is expected to be small in practice. For the case where the sender's queue is empty, all of the received packets go into the candidate set. However in order to sustain the proposed algorithm, packets within the prespecified period of N is sufficient, meaning that m is bounded by N. On the other hand, it might be difficult to determine N for different types of packet streams. Let us consider some parameters that may be helpful in making remarks about m.

$$M_{min} = (N/N') \times S_n \tag{4.1}$$

where

M_{min}: Minimum size of the candidate set,

N': Minimum time difference between two nondelayed packet intervals,

S_n: Maximum number of consecutive nondelayed packets within a nondelayed packet interval.

The above formulation states that the candidate set must have all possible nondelayed packets within the period N. In our simulations m is kept low. However, with different traffic assumptions additional methods may be needed to ensure that m is sufficiently small. If m tends to be a large number, which is not suitable for OWADME to work online, we propose to employ one of the following two approaches in practice:

1. Reduction of S_n: If the sender's queue is empty, multiple and consecutive nondelayed packets are received by the receiver. These consecutive packets increase the size of the candidate set. For those cases where m is not sufficiently small for OWADME to work online, a merging process can be utilized similar to that we propose for our skew estimation process.

2. Variable N: N value used in OWADME helps us to specify an upper bound for delay measurement error. The assumption made in error analysis states that there is at least one nondelayed packet within period N, which means that it is

the largest possible interval between any successive nondelayed packets. Because of not having two-way messaging between the sender and receiver, it should be a predefined constant. However, observed intervals between successive nondelayed packets are mostly expected to be much smaller compared to N. The candidate set needs to have all recent nondelayed packets that are received within the last period N. Hence, small nondelayed packet intervals with respect to N increases the size of the candidate set. To reduce the size, a variable N can be used by detecting the current interval and predicting the next one, which requires another assumption on the variation of the nondelayed packet interval. Otherwise error bound reliability will be compromised. We consider use of variable N as future work.

CHAPTER 5

COMPUTATIONAL RESULTS

The present thesis and the proposed method OWADME are based on three main assumptions:

1. existence of a bound on clock skew
2. existence of a bound on clock drift
3. existence of at least one sync (zero-delay) packet in a prespecified period

The first and second assumptions are used for the filtering mechanism and state that skew and drift can change only within some boundary. These bounds derive from the physical conditions of the environment that OWDM method operates. The third assumption allows us to compute maximum error bounds for each packet.

In the present section we evaluate OWADME in detail. For this purpose a simulation study is conducted, which uses a synthetically generated packet stream transmitted between a sender and receiver under certain background traffic. In the following subsections, first our test case and traffic generation method is presented, then the effects of filtering with respect to both skew and drift are analyzed and finally OWADME is evaluated against Sync & Sense [5].

5.1 Case and Traffic Generation

For the generation of a skew pattern to be used as a reference in our tests, a random drift value is chosen within a specified boundary. This value is assumed to be valid

between two successive actions in the simulations. A delta skew is then computed by integrating the drift value and total skew is computed additively after every action in the simulation.

To realize our third assumption, we need a kind of congestion avoidance algorithm. Selection of a good congestion avoidance algorithm is out of the scope of the present thesis hence we implemented a very simple one that shapes the traffic directly.

In our simulations, we added background traffic to be able to simulate random delays for our periodic probe packets. The background traffic is generated using Poisson distribution with an adjustable rate. We employed three different rates to create and then resolve congestion. The first rate corresponds to a high load case where the link is overloaded, i.e. more background packets are injected than the link capacity. As a result, congestion builds up and the sender queue size starts to increase. The second rate corresponds to low load case. To empty the sender queue, we use a background traffic generation rate, which is slightly less than the link capacity. In this case, we use a non-zero rate to be realistic to a certain extent because a fast reduction in delay is much easier to detect and make nondelayed packets more observable. The third rate used to generate a background traffic pattern is the silence rate. After applying the low load rate we may still not observe zero delays for the periodic packets. In order to guarantee the existence of nondelayed periodic packets, the traffic rate is finally reduced to less than half the capacity.

We apply all three traffic rates in a round robin fashion during a simulation run. At the first stage high load is applied and the sender queue size increases. When the queue size reaches to a certain threshold low load is applied as a second stage and queue size starts decreasing. When queue size approaches zero, the silence rate is applied as the third stage to definitely obtain some nondelayed periodic packets. The silence rate is applied for a very brief period of time. As a consequence, a multitude of nondelayed packets may be observed in addition to packets with very small delay. Some of these packets will pass through our filtering mechanism, which makes the filtering task more challenging. The three stage traffic generation described above is repeated until the end of the simulation.

As was stated earlier, our third assumption states that there exists a prespecified pe-

riod in which at least one nondelayed packet is observed. However, due to our use of Poisson distribution in the tests, we cannot guarantee such a specific value in the simulations. After a couple of trial and error steps, we set an experimentally determined value depending on our rate choices and the associated sender queue sizes. Namely, before running our tests for OWADME, we observed the described traffic structure and fixed this prespecified period N. As future work, a variable congestion avoidance period may be considered at the receiver side.

The choice of r in Algorithm 4 depends on the actual traffic pattern and has an effect on performance. We heuristically optimized r using the traffic pattern described above. We observed that the actual error made in delay measurements increases as a result of over merging when $r > 20$. When $r \leq 5$, potential packets that can be merged stay unmerged and the error increases similarly as a result of under merging. In the following simulations, r is chosen to be 10.

5.2 Effect of Filtering

OWADME steps are composed of filtering, skew estimation, delay measurement and error analysis. Delay measurement and error analysis steps use the same set of filtered packets (candidate set) and these steps are independent of each other. To perform better in both, filtering mechanism is required to filter out all of the delayed packets and leave only the nondelayed ones in the candidate set. To evaluate the proposed filtering approach, we observe the ratio of the number of nondelayed packets to candidate set size and name it as success ratio. This ratio is provided twice, first after filtering with respect to skew and then after filtering with respect to drift. Each run is repeated for a sufficient number of times and the results are averaged such that the results are compliant with 95% confidence interval and a sufficient gap so that compared entities do not overlap. The number of zero-delay packets (sync packets) generated throughout a simulation depends on the duration of the silence period. Fig. 5.1.a shows the number of sync packets generated throughout a simulation with respect to silence duration and Fig. 5.1.b gives the success ratio with respect to silence duration for filtering using skew bound only and for filtering using both skew and drift bounds together.

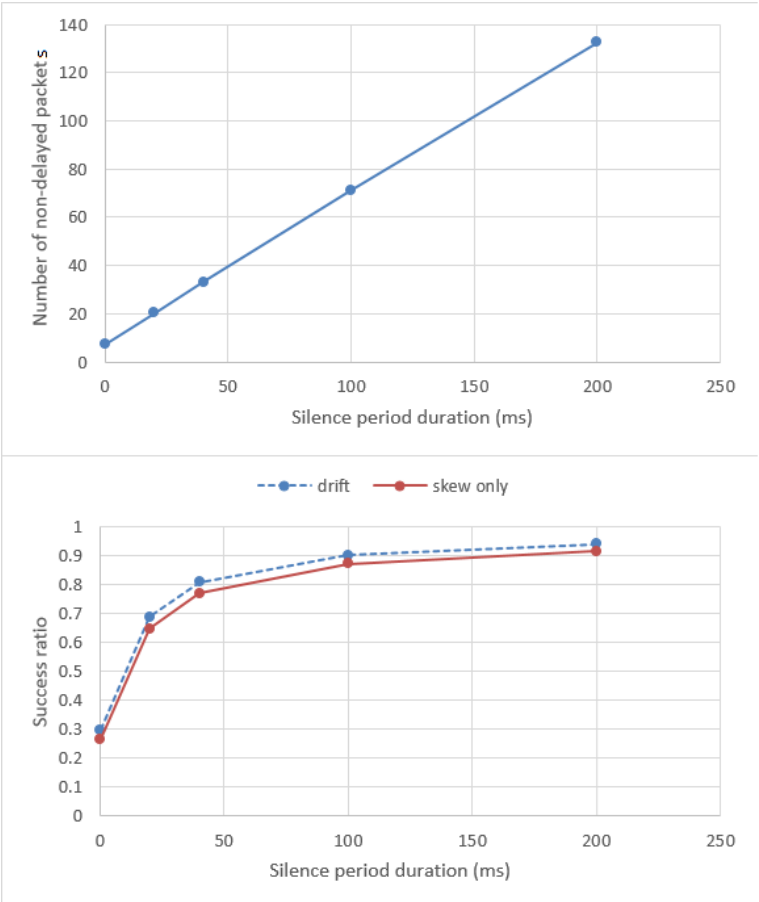

Figure 5.1: (a) Number of sync packets generated throughout simulation with respect to silence period duration (ms) (b) Success ratio as a function of silence period duration (ms)

It is observed that increasing the duration of the silence period increases the number of detected sync packets linearly. In our filtering mechanism, every nondelayed packet narrows the area of subsequent packets through which they can pass. This makes it harder for non-sync (delayed) packets to pass through the filter. In other words, with a larger number of nondelayed actual packets, elimination of the delayed ones becomes much easier and hence the ratio of the number of sync packets to candidate set size increases.

Fig. 5.1.b illustrates the effect of using filtering with respect to drift after filtering with respect to skew. It is observed that *skew* and *skew+drift* filtered success ratios are close to and within 2 to 5% proximity of each other. As a result, we believe that applying a third filter that uses a bound on the derivative of drift may not prove to be

useful in reducing the size of the candidate set further.

5.3 Overall Evaluation and Comparison with the Literature

Sync & Sense [5] is the most prominent candidate in the literature that can be compared with OWADME. So we have implemented and tested Sync & Sense using our simulation setup under the same traffic structure to be able to compare the two fairly.

Similar to our third assumption, Sync & Sense also needs nondelayed packets to estimate f_{NDP}. However, in Sync & Sense, synchronization can be made only for negative skew in reference to sender side. If synchronization can be achieved at some point during the simulation for positive skew cases, the reset mechanism embedded in Sync & Sense cannot be applied meaning that a cumulative error may occur.

In Sync & Sense, VOIP packets having a period of 40 ms are used within a multi-node multi-hop network structure [5]. However, the traffic between the nodes (except the last one) of this test network is very light and never creates congestion in their multi-hop structure. To be able to create a sufficient amount of congestion, only the last node's traffic rate was kept very close to ongoing traffic rate in the original paper. As a result, Sync & Sense traffic conditions can be repeated using only one hop communication. Therefore in the experiments, we used a single hop network and a period of 20 ms for periodic packets.

Sync & Sense can provide delay measurements whenever it is synchronized. But whenever a threshold error margin is exceeded, a reset process is applied and hence no output can be observed between such a reset and the following synchronization event. We therefore use two metrics to compare Sync & Sense and OWADME, which are the output provisioning ratio (opr) and the error made in delay estimation (err). The output provisioning ratio (opr) is the ratio of the amount of time that Sync & Sense is capable of providing a delay calculation to the overall simulation time. This ratio depends on the value of the error margin parameter used in the Sync & Sense algorithm. In order to make our comparisons reliable, we simulated the Sync & Sense with various error margin values starting from 0.5 to 2 ms with 0.25 ms step sizes. Three example cases are presented in the following subsections to explain the

similarities and differences between Sync & Sense and the proposed scheme. For these cases, the error margin parameter is selected in such a way that Sync & Sense produces high output provisioning ratio and low error.

5.3.1 Initial positive skew case

Fig. 5.2 presents results of Sync & Sense and OWADME simulations and provides a performance comparison when initial skew is assumed to be positive. The following parameters are used in the simulation scenario:

- Initial skew value: 200 ppm.
- Min and max skew values: -600ppm and 600ppm with respect to the receiver side.
- Max drift bound: 5ppm/sec.
- Simulation duration: 90 s (starting at 10 s and finishing at 100 s)

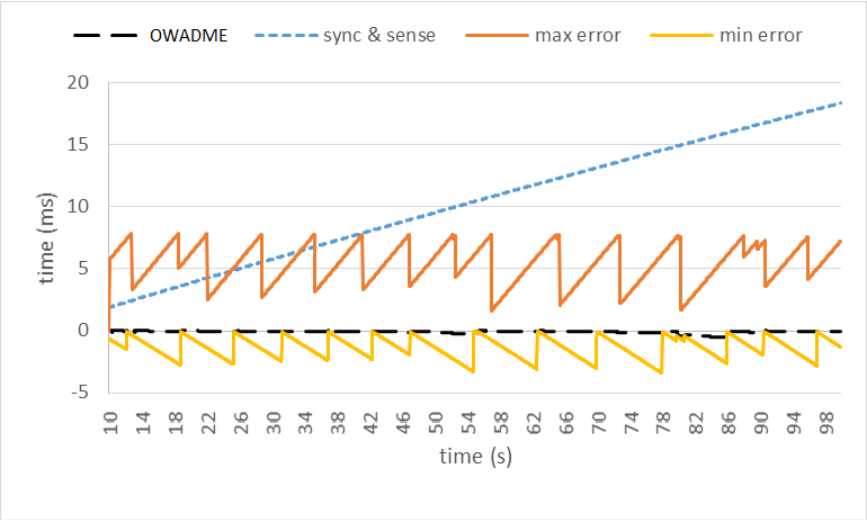

Figure 5.2: Sync & Sense and OWADME performance comparison and error bounds (positive initial skew case (ms))

x-axis corresponds to simulation time and y-axis shows error and error bound in ms precision. The blue (square dot) and black (long dash) lines are errors made by Sync

& Sense and OWADME, respectively. The orange and yellow lines indicate the maximum error boundary calculated by OWADME. As stated earlier, Sync & Sense is observed not having ability to use its reset process for positive skew cases. As a consequence, if the synchronization is achieved once, there is no reset opportunity anymore and the error accumulates. Period is constant in Sync & Sense hence with 200ppm initial skew value the error accumulates up to 20ms within about 100 s.

5.3.2 Initial zero skew case

Fig. 5.3 presents similar results for the same set of parameters, except initial skew, which is assumed to be zero.

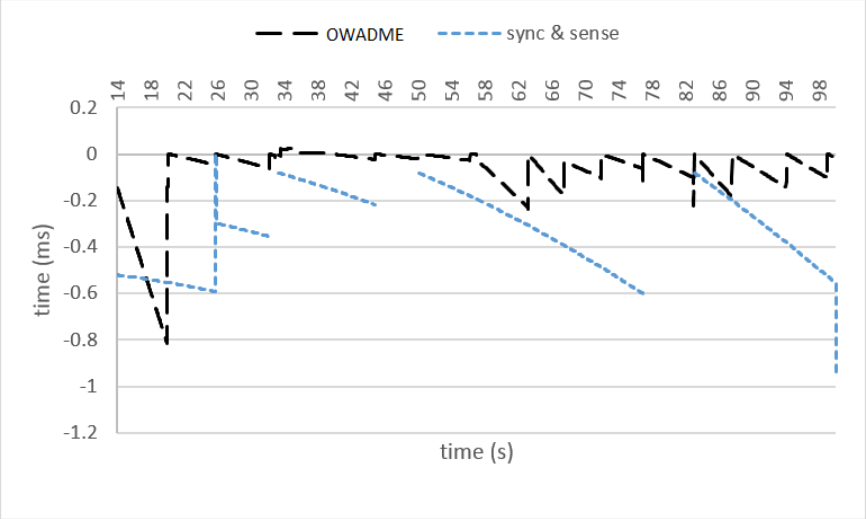

Figure 5.3: Sync & Sense and OWADME performance comparison (zero initial skew case (ms)

It is observed in Fig. 5.3 that OWADME provides better results in general. In Sync & Sense, the receiver side treats the period of packets as a constant term and hence skew value is very close to zero in this case. The error does not increase and the results are more comparable in this initial zero skew case. Due to the synchronization failures, Sync & Sense could not provide any output during some intervals (two largest intervals being (44.9, 50.0) and (76.9, 83.0)) for a total of approximately 12 s out of 90 s making an *opr* of 0.87.

The error bound computed by OWADME is presented separately in Fig. 5.4. With zero initial skew value, the error bounds calculated by OWADME are not comparable with Sync & Sense. It is observed that any error is mostly less than 0.2 ms and error bounds range between 2 and 10 ms for OWADME.

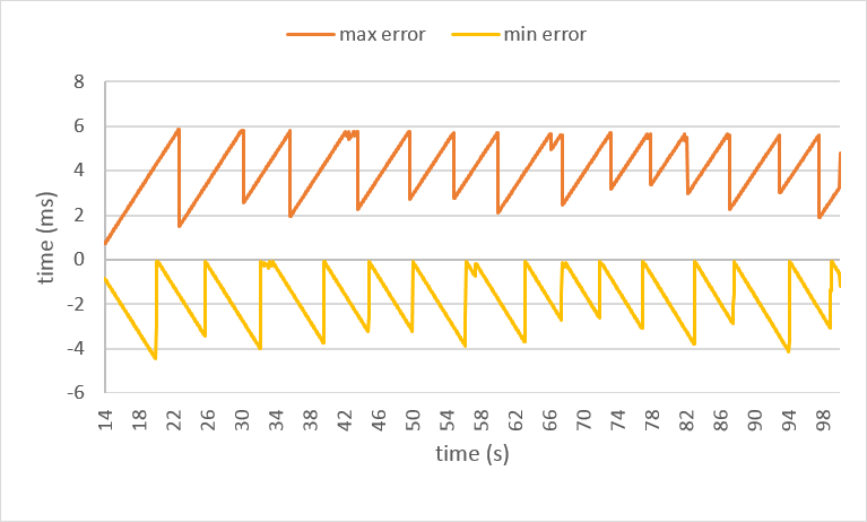

Figure 5.4: Maximum error bound of OWADME (zero initial skew case (ms))

5.3.3 Negative initial skew case

Fig. 5.5 presents similar results for the same set of parameters, except initial skew, which is assumed to be -200 ppm.

It is observed in Fig. 5.5 that OWADME provides much better results than Sync & Sense. Due to the synchronization failures, Sync & Sense could not provide any output during certain intervals (two largest intervals being (52.3, 60.0) and (89.1, 96.4)) for a total of approximately 16 s out of 90 s making an *opr* of 0.82.

Measurement error made by Sync & Sense and error bounds calculated by OWADME are given on the same graph in Fig. 5.6.

Orange and yellow lines correspond to an error bound of OWADME. Blue lines represent the measurement error made by Sync & Sense. For initial skew values as low as -200 ppm, even the error bounds computed by OWADME are comparable with

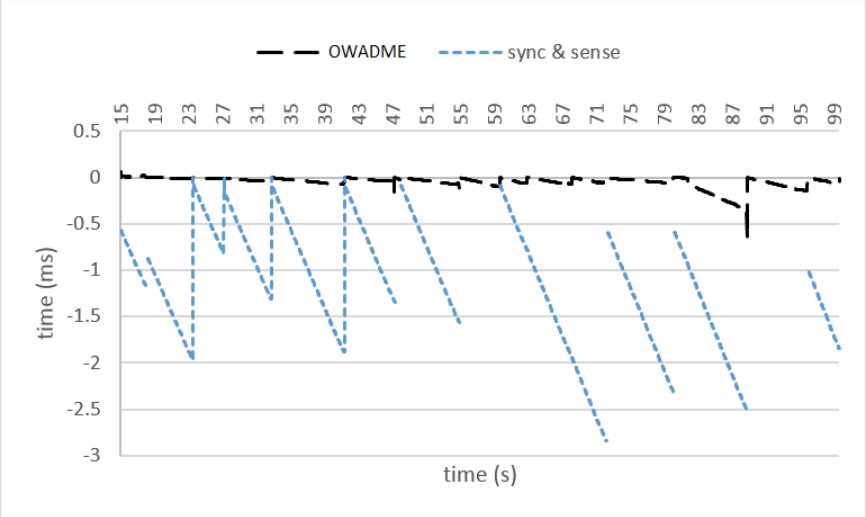

Figure 5.5: Sync & Sense and OWADME performance comparison (negative initial skew case (ms)

Sync & Sense results. The error of OWADME on the other hand is incomparably better (Fig. 5.5).

5.4 Overall comparison with Sync & Sense

Fig. 5.7 finally provides an overall comparison of Sync & Sense and OWADME. In Fig. 5.7, x-axis shows the initial skew values. Five cases are considered in our experiments, i.e. -300, -200, -100, 0 and 100 ppm. Due to very low output provisioning ratio, initial skew values higher than 100 cases are not reported. For all skew values many simulations are performed such that the results are compliant with 95% confidence interval and a sufficient gap so that compared entities do not overlap. Minimum and maximum errors made in each set of experiments are marked with the same color. y-axis in the upper graph is the average error at its corresponding initial skew value in ms. y-axis in the lower graph is the output provisioning ratio. Purple, red, green, light green, orange, gray and black lines represent Sync & Sense error margin values of 0.5ms, 0.75ms, 1ms, 1.25ms, 1.5ms, 1.75ms and 2ms, respectively. The blue line represents the error made by our proposal OWADME.

It is observed in Fig. 5.7 that the average error in Sync & Sense decreases when

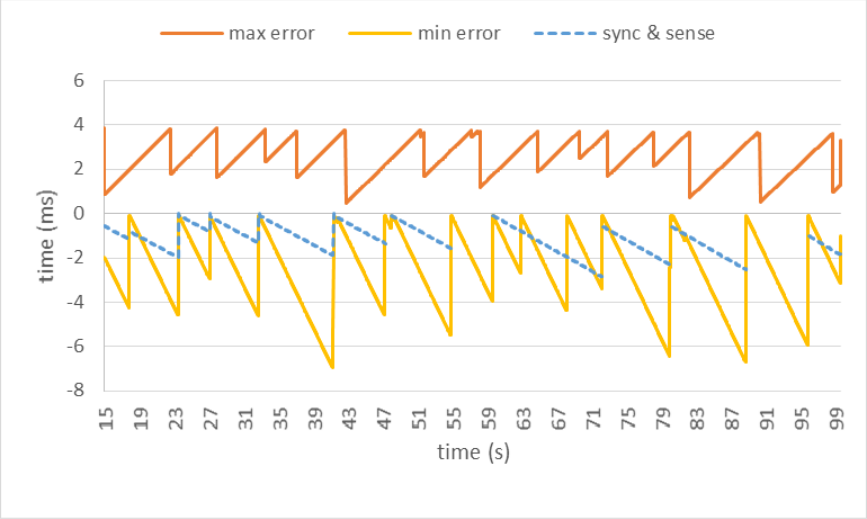

Figure 5.6: Sync & Sense measurement error and OWADME maximum error bound (negative initial skew case (ms))

initial skew value gets closer to zero. This is because of using a constant period at the receiver side. On the other hand, OWADME performance is very stable for all initial skew values. The reason for this is that OWADME measures the skew and adjusts its period value using these measurements in its delay estimations.

Another major result one can deduce in Fig. 5.7 is that use of lower error margins in Sync & Sense causes higher error rates and lower output provisioning ratios in high initial skew values.

To better explain this phenomenon we should restate the purpose of using an error margin parameter in Sync & Sense. In Sync & Sense every packet is compared with its predecessors. If the time difference between two consecutive packets has a lower period than the assumed constant period minus the error margin, synchronization state is cancelled and the system starts waiting a new zero-delay packet. If a smaller error margin value and a high initial skew value is used, every nondelayed packet causes loss of synchronization. As a result, output provisioning ratio decreases and sync intervals start to have relatively short life spans.

Use of a small error margin in Sync & Sense produces large error and low output provisioning ratio. Increasing error margin decreases the error and increases output provisioning ratio. But error margin cannot be increased endlessly and error perfor-

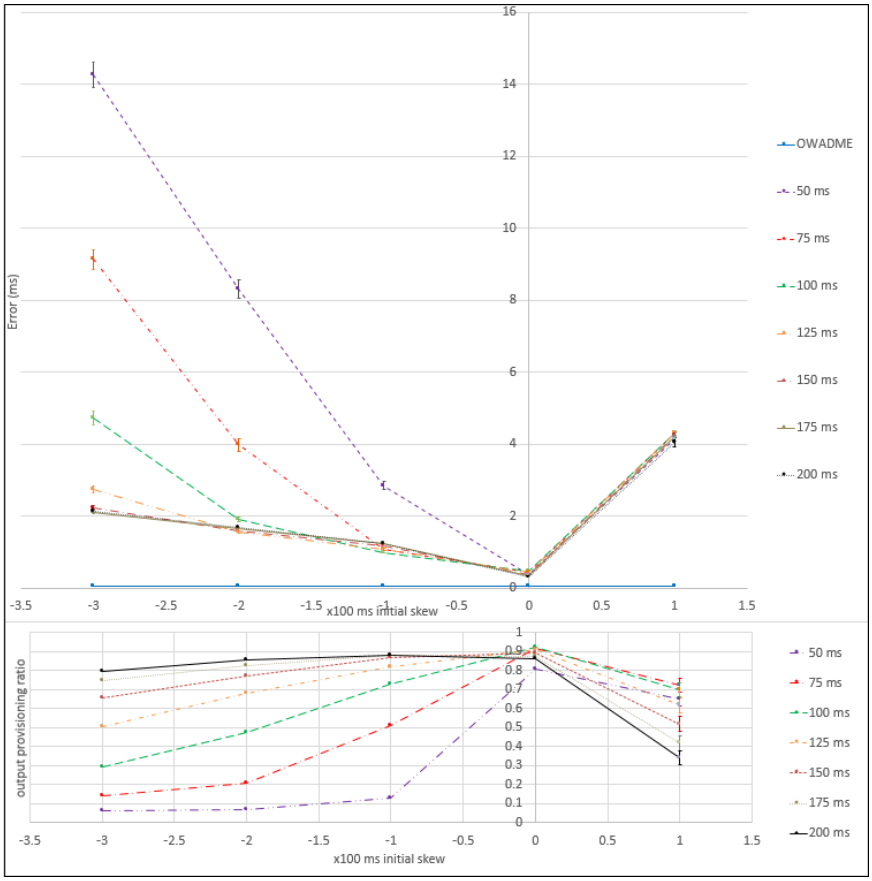

Figure 5.7: Measurement errors made by Sync & Sense and OWADME for different error margin parameters and initial skew values (ms)

mance saturates as output provisioning ratio approaches to 1. If higher error margins are tested after reaching maximum output provisioning ratio, we may expect to observe higher error rates because when the output provisioning ratio is already maximum, one cannot get longer sync intervals and hence increasing error margin further causes higher error rates only.

On the other hand OWADME's error rate is nearly the same for all initial skew values. Hence independent of initial skew, it is equal to approximately 70 μs in average. OWADME outperforms Sync & Sense error performance also with an output provisioning ratio of one in all possible cases.

CHAPTER 6

PROBE PACKETS

In this thesis, we characterize delay measurement to be active, which means that we inject some additional packets to the network for our purpose. These packets are called as probe packets throughout the thesis except for this chapter. In this chapter however, we assume that there are other aperiodic data packets, which are different than the periodic probe packets and which we want to measure their delay.

Previous chapters explain how our one way delay measurement mechanism works on periodic probe packets, in other words how we measure the delay of probe packets. In this chapter, we explain how to measure the delay of an arbitrary and aperiodic packet by using the delay of probe packets. Two key concepts used for this purpose are as follows:

Distance is an *h* bit field allocated in every aperiodic packet and is used to indicate the index of the time slot (interval) the packet is sent. Distance is coded in binary hence h=log m where m is the number of time slots in a period.

Resolution is the duration of each time slot in seconds within a period.

Let us assume that at time t_{probe} a probe packet has arrived and its delay is measured to be $t_{probedelay}$ by our previously described scheme. For a packet which comes after the probe packet, having a distance of x means that the packet is sent from the sender side at $(x/(2^h)) \times p$ seconds later than the probe packet is sent. As a result, if the packet and the probe packet have the same delay, the packet arrives at $t_{probe} + (x/(2^h)) \times p$. However, delays of the packet and the probe packet may be different. Therefore, the difference between the arrival times can be compared with the distance number to find

the correct interval of packet delay. As a result, delay of an arbitrary and aperiodic packet can be stated as;

$$t_{probedelay}-(x/(2^h)\times p)+(t_p-t_{probe}) > delay > t_{probedelay}-((x+1)/(2^h)\times p)+(t_p-t_{probe}) \tag{6.1}$$

where;

$t_{probedelay}$: delay of the probe packet,

t_p: arrival time of packet,

p: probe packet period,

t_{probe}: arrival time of the probe packet.

Distance information is an h bit number which states an interval, not a certain time spot. Therefore, there exists an uncertainty in the specified time interval which corresponds to our resolution. So, there occurs a tradeoff between the network overhead and the resolution of the measured delay. As an example, if h is 4 and probe packet period is $80ms$, our resolution is $5ms$. For this case network overhead consists of 4 bits for each packet and probe packet size of $80ms$. To be more clear, resolution and network cost can be stated as;

$$\textit{resolution} = p/(2^h) \tag{6.2}$$

$$\textit{traffic overhead} = (ps + (p/pd) \times h) \textit{ per period } p \tag{6.3}$$

where;

pd: packet density or period,

ps: probe packet size.

For the above equation, resolution is assumed to be much greater than maximum delay measurement error. For those cases they are comparable, resolution definition

will be sum of both.

6.1 Sequence Number

The sequence number is used for preventing problems caused by loss of a probe packet. Without any sequence number, if a probe packet is lost, our scheme adds one period to successive packets' delay. Therefore, one probe packet period error cannot be recovered for period N and all the packets in this period N have at least one probe packet period error. By utilizing a sequence number in each periodic probe packet, a recovery in one probe packet period can be realized by comparing the sequence number of successive probe packets.

6.2 Without probe packets

For a network with periodic packets, above defined sequence number can be inserted to all packets. By using periodicity of the actual user data packets, our delay measurement scheme can be realized easily. Successive packet drop probability defines the size of sequence number. However for any network it can be kept small compared to the size of any header of probe packets. For this case, all the traffic overhead arises from the sequence number (ss).

$$\textit{traffic overhead} = ss \textit{ per period } pd \tag{6.4}$$

Therefore, maximum delay measurement error becomes equal to the resolution.

Having periodic data packets is a special case. Period or density of packets (pd) is assumed to be a network dependent value and is accepted as a constant for a given network. Resolution depends on probe packet period p. Moreover, p should be much smaller compared to N. Assume that N is very large and to be able to minimize the overhead p can be selected as a much larger value than pd. For this case, overheads of two cases for a p period;

$$\textit{Case 1 overhead} = (ps + (p/pd) \times h) \tag{6.5}$$

$$\textit{Case 2 overhead} = (p/pd) \times ss \tag{6.6}$$

$$\textit{Case 1 resolution} = (p/(2^h) \times p) \tag{6.7}$$

$$\textit{Case 2 resolution} = \textit{ maximum error} \tag{6.8}$$

Therefore, case 1 is superior for;

$$(p/pd) \times (ss - h) > hs + ss \tag{6.9}$$

For cases where h (distance number size) is greater than ss (sequence number size), using periodicity of data packets is always a better option. For other cases, if resolution meets the design requirements above equations should be considered to have less traffic overhead.

If a network does not contain periodic data packets or packets do not have any space for sequence numbers, probe packet insertion is the only option. In that case, tradeoff between traffic overhead and resolution should carefully be analyzed.

CHAPTER 7

CONCLUSION AND FUTURE WORK

The present work deals with the problem of one way delay measurement without any hard clock synchronization and two-way messaging. The method presented in this thesis utilizes assumptions derived by considering the physical conditions that may affect clock speed in associated network nodes. With the help of these assumptions most of the delayed packets at the receiver side are easily revealed to be so and a candidate set of packets is formed by including probable nondelayed packets only. All possible pairs in this candidate set, which implies a skew value, are then considered to calculate a weighted average skew value. For calculation of the weights, certain attributes such as proximity and recency of the packets in each pair in the candidate set are utilized. The closest packet to the obtained average skew line (value) is then used as a reference point to perform the delay measurement. Moreover, as a consequence of the assumptions made, an error bound is also computed for every reported delay.

To the best of knowledge of the authors OWADME is the first method of its kind, which adaptively modifies its skew measurement and the error bound simultaneously for every delay measurement result it produces.

OWADME is compared with a recent one way delay measurement technique called as Synch & Sense [5]. Simulation results show that our proposal is superior to the existing work of the same category and comparable to other OWDM schemes that utilizes two-way messaging.

Similar to [5], the network topology implemented for our evaluation purposes has one bottleneck link. For the case that includes multiple bottleneck links, a congestion

avoidance algorithm should be implemented as future work to make the prevailing assumptions of the paper to be applicable to commonly encountered network types in practice.

For the cases a better congestion avoidance algorithm may not be possible to implement, a variable N can be considered. As stated above, period N is the duration in which at least one nondelayed packet arrives. However, changing N takes away the certainty in the computed error boundary. Hence a trade off exists between having an error bound and having a congestion avoidance algorithm that is sufficient for the assumption of the proposed scheme. With other additional assumptions and observation of nondelayed packets, period N could be managed; but, in this case use of variable N may increase the complexity. To sustain the proposed algorithm, a better merging mechanism may also be implemented as an alternative.

REFERENCES

[1] L. De Vito, S. Rapuano and L. Tomaciello "One-way delay measurement: state of the art," *IEEE Trans. on Instrumentation and Measurement*, 57/12, pp. 2742-2750 (2008).

[2] (2004). MikroTik RouterOS V2.8. Reference Manuel, GPS Synchronization. [Online]. Available: http://www.mikrotik.com/docs /ros/2.8/system/gps, accessed Aug. 26, 2015.

[3] D.L. Mills "Network Time Protocol (Version 3) Specification, Implementation and Analysis", IETF RFC 1305 (1992).

[4] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Std. 1588-2088, (2008).

[5] B. Ngamwongwattana and R. Thompson "Sync & Sense: VoIP measurement methodology for assessing one-way delay without clock synchronization," *IEEE Trans. on Instrumentation and Measurement*, 59/5, pp. 1318-1326 (2010).

[6] A. Vakili and J.C. Grégoire "Accurate one-way delay estimation: limitations and improvements," *IEEE Trans. on Instrumentation and Measurement*, 61/9, pp. 2428-2435 (2012).

[7] J. Wang, J. Yang, G. Xie, Z. Li and M. Zhou, "On-line estimating skew in one-way delay measurement," in *Proc. of the 4th Int. Conf. on Parallel and Dist. Comp., Applications and Technologies (PDCAT'2003)*, pp. 430–436 (2003).

[8] M. Aoki, E. Oki and R. Rojas-Cessa "Measurement scheme for one-way delay variation with detection and removal of clock skew," in *ETRI Journal*, Vol. 32, pp. 854-862, (2010).

[9] J.H. Choi and C. Yoo "One-way delay estimation and its application," *Computer Communications*, Vol. 28, pp. 819-828 (2005).

[10] M. Shin, M. Park, D. Oh, B. Kim and J. Lee "Clock synchronization for one-way delay measurement: a survey," in *Advanced Communication and Networking, Communications in Computer and Information Science*, Vol. 199, pp. 1-10 (2011).

[11] D. Kim and J. Lee "One way delay estimation without clock synchronization," *IEICE Electronics Express*, 4/23, pp. 717-723 (2007).

[12] W. Xu, Z. Zhou and Q. Liu "Hybrid one-way delay estimation for networked control system," *Advances in Engineering Software*, 41/5, pp. 705-711 (2010).

[13] Y. Tobe, H. Aida, Y. Tamura and H. Tokuda "Detection of change in one-way delay for analyzing the path status," in *Proc. of the Passive and Active Measurement Workshop (PAM 2000)*, pp. 61-68 (2000).

[14] Z. Yang, L. Cai, Y. Liu and J. Pan "Environment-aware clock skew estimation and synchronization for wireless sensor networks," in *Proc. of IEEE Infocom* , pp. 1017-1025 (2012).

[15] T. Schmid, Z. Charbiwala, R. Shea and M. Srivastava, "Temperature compensated time synchronization," *IEEE Embedded Systems Letters*, 1/2, pp. 37-41, (2009).

[16] H. Khlifi and J.C. Grégoire "Low-complexity offline and online clock skew estimation and removal," *Computer Networks*, 50/11, pp. 1872-1884 (2006).