TIME SYNCHRONIZATION IN LARGE-SCALE NETWORKS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

An-swol Clement Hu

January 2007

TIME SYNCHRONIZATION IN LARGE-SCALE NETWORKS

An-swol Clement Hu, Ph.D.

Cornell University 2007

Network time synchronization is an important aspect of sensor network operation. It is often achieved by synchronizing the clock of each node in the network to the clock of some reference node. However, it is well known that synchronization error accumulates over multiple hops. This *scalability problem* presents a challenge for large-scale, multi-hop sensor networks with a large number of nodes distributed over wide areas.

In this thesis we develop the use of spatial averaging as an approach to mitigating the effects of the scalability problem. We first develop a cooperative synchronization technique using spatial averaging that can achieve "perfect" synchronization in the limit of an infinitely dense network. We show that it is possible to maintain a perfect timing signal with equispaced zero-crossings that occur at integer values of the reference time. Second, we study the benefits of cooperative time synchronization using spatial averaging in networks of finite density. We present a protocol that uses spatial averaging to reduce error accumulation in large-scale networks and show that synchronization performance can be significantly improved by increasing network density.

BIOGRAPHICAL SKETCH

An-swol Hu grew up in the California Bay Area and received his B.S. in Electrical Engineering from Stanford University, Stanford, California in June 2002. Since that July, he has been a graduate student at Cornell University, Ithaca, New York and a member of the Communication Networks research group. His research interests center around applying statistical and probabilistic principles to the study of communication networks. He received his Ph.D. from Cornell University in January 2007.

To my family and Sherry.

ACKNOWLEDGEMENTS

First of all, I would like to thank my adviser, Professor Sergio Servetto, for his many years of guidance. His tireless passion for doing research has been an inspiration. I would also like to thank the members of my committee, Professor Lang Tong, Professor Toby Berger, and Professor Richard Rand, for their valuable insight that helped direct this work.

I want to thank Ron, Will, Christina, Megan, Filip, and Emilia; their friendship made the good times at Cornell great. I will fondly remember the board games, the movies, the ski trips, and the cooking adventures. Also, the members of the Communication Networks research group, Mingbo, Georgios, Ron, and Christina, made the experience of working in Rhodes Hall a rewarding one.

I want to thank my parents, my brother, and my family for their constant encouragement. From thousands of miles away their love helped make this work possible.

Last, but not least, I want to thank Sherry for standing by me all these years, for believing in me, and for supporting me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Time Synchronization and High Density Networks

The problem of time synchronization is the task of giving all elements of the system a common time scale on which to operate. This common time scale is usually achieved by periodically synchronizing the clock at each element to a reference time source so that the local time seen by each element of the system is approximately the same. Time synchronization plays a vital role since it allows the entire system to cooperate and function as a cohesive group. This is particularly important for a variety of tasks such as synchronizing event detection, data fusion, and coordinating wake and sleep cycles. Due to its importance, the problem of time synchronization has been around for a long time [1], but a different type of network is presenting new challenges and opportunities for network synchronization techniques.

Modern ad hoc sensor networks represent a type of network that often does not have any infrastructure. This means that nodes can be deployed over some area and then are required to organize and synchronize using only network resources. This is very different than networks such as the internet where an infrastructure can be established using resources outside the network. For example, every node in the internet may only be a few hops away from a time server and these time servers can be synchronized using out-of-network resources such as GPS signals or WWVB radio broadcasts [2]. However, even though ad hoc sensor networks present a new synchronization challenge, they may also have characteristics that are beneficial.

Recent advances in ad hoc sensor networks are beginning to enable the deployment of large-scale networks with high node density. For example, a hardware simulation-and-deployment platform for wireless sensor networks capable of simulating networks that have on the order of 100,000 nodes was recently developed [3]. As well, for many years the Smart Dust project sought to build cubic-millimeter motes for a wide range of applications [4]. Also, there is work in progress on the drastic miniaturization of power sources [5]. These developments, and many others, indicate that future networks may have extremely large numbers of nodes deployed over wide areas. The question we consider is whether or not the density of future networks can be used to address synchronization issues that plague existing techniques. In particular, we study the scalability problem in the context of high density networks.

1.2 Synchronization and the Scalability Problem

Many techniques have been proposed for synchronizing sensor networks [6, 7, 8, 9, 10]. These techniques rely on nodes exchanging packets with timing information. Using the exchanged timing information, each node can then estimate clock offset and maybe clock skew. However, all of these traditional synchronization techniques suffer from an inherent scalability problem—synchronization error accumulates over multiple hops. Consider the situation illustrated in Fig. 1.1. Let us assume that we wish to establish a global time scale for this network where node 1 contains the reference time. Nodes 2 and 3 can estimate their synchronization parameters, i.e. clock skew and offset, directly from node 1. This means that nodes 2 and 3 will have an estimate of the clock of node 1, but recall that the estimates will have errors. Node 4, on the other hand, is outside the broadcast domain of node

1 so it must estimate its clock skew and offset relative to node 1 through the use of timing information from nodes 2 and 3. However, since the timing information provided by nodes 2 and 3 regarding the clock of node 1 will have errors, the skew and offset estimates made by node 4 will be further corrupted by timing errors.

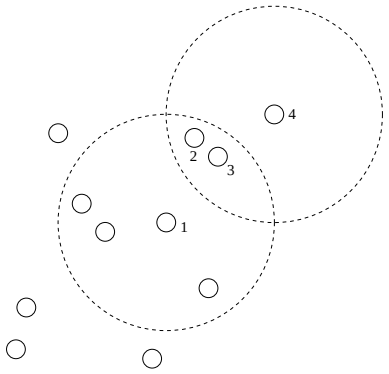

Figure 1.1: In the above figure, assume node 1 has the reference time and all nodes want to be synchronized to this clock. Nodes 2 and 3 are in the broadcast domain of node 1. However, node 4 must be synchronized through nodes 2 and 3.

This accumulation of error over multiple hops poses a problem as sensor networks are deployed over larger and larger areas. The number of hops required to communicate across the network increases; thus, the synchronization error across the network increases. One possible way to avoid the scalability problem is to use a few nodes with powerful radios to limit the number of hops required to communicate timing information across the network. However, this technique does not address the fundamental scalability problem of errors accumulating over multiple hops. In addition, this technique places the burden of time synchronization on a few special nodes and is bad for fault tolerance. It would be desirable to develop synchronization techniques for large-scale, multi-hop networks that do not rely on special nodes for synchronization. It is our belief that high density networks may provide a new alternative to mitigating the scalability problem.

1.3 Motivation for Cooperation

The scalability problem is an inherent part of all synchronization algorithms since estimation errors will inevitably accumulate over multiple hops. In order to reduce the effect of this problem, we need to find ways to reduce the synchronization error at each hop. There are two primary ways to accomplish this. The first is to collect more timing information. With more timing data, nodes can generally make a better estimate of clock skew and clock offset. For example, RBS [6] and FTSP [10] both let nodes collect many timing data points before estimating clock skew and clock offset. A timing data point provides a node with the time at a reference clock at a specific time in its local time scale. With more data points, synchronization error will decrease. This technique is essentially doing a *time* average to estimate clock skew and clock offset. However, increasing the number of data points is not necessarily practical since it would significantly increase the amount of network traffic and the time needed to synchronize the network.

The second primary approach is to improve the quality of the timing data point. For example, TPSN [8] and FTSP use a MAC layer time stamping technique that is more accurate than that employed in RBS. However, we believe that there is a fundamentally new technique for improving data point quality that has not been considered before. This new idea is to use spatial averaging to improve data point quality. Even though we may not have a large number of timing data points that occur at different points in time, in a high density network we may have a large number of surrounding nodes. Instead of only doing a time average to estimate the clock skew and clock offset, perhaps we can also do a *spatial* average to improve these estimates. For example, for a given node i, many surrounding neighbors will have timing information about the global clock since we want to synchronize the

entire network. Therefore, it seems reasonable that the information from many surrounding nodes should somehow be able to help improve the skew and offset estimates of node i.

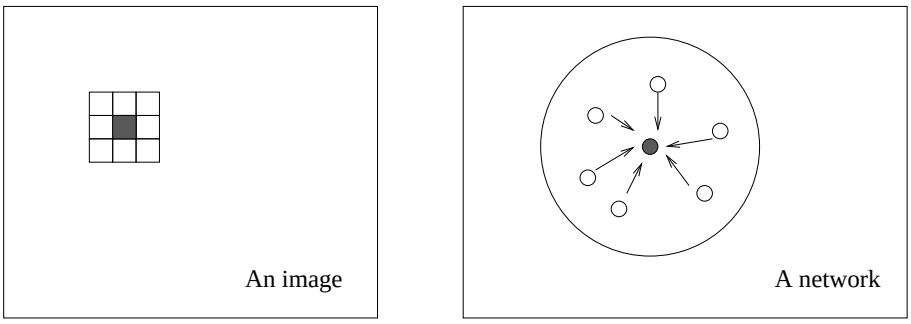

Figure 1.2: When conventional spatial averaging ideas, such as those employed in image processing applications, are applied to networks, communication becomes a bottleneck.

However, it is important to realize that our spatial averaging idea is much more involved than simply collecting more timing information from many surrounding nodes. Existing techniques often do collect information from many neighbors since a particular node may obtain each timing data point from a different neighbor. However, this existing approach is essentially a direct extension of conventional spatial averaging concepts. Consider spatial averaging in image processing applications where, for a given pixel, it is possible to use surrounding pixels to de-noise the image through spatial averaging. If such a concept is used in a network, the first thing that we notice is that timing information needs to be collected by each node (Fig. 1.2). This means that communication becomes a bottleneck. If node density is increased and if a given node wants to utilize the timing information in more neighbors, then the amount of network traffic will grow. This increase in communication traffic is clearly not desirable.

Our goal is to develop a new concept for spatial averaging in which it is possi-

ble to decouple network density and the network traffic used for synchronization. Such a technique would make it possible to improve synchronization performance through increased node density without placing additional communication burden on the network.

1.4 Pulses for Spatial Averaging

The question then becomes how we can actually implement this idea of spatial averaging. Packet exchanges used in existing synchronization protocols are well suited for time averaging, but are not easily adapted for spatial averaging. As a result, we turn to studies of synchronization in other fields for motivation.

Emergent synchronization properties in large populations have been the object of intense study in the applied mathematics ([12, 13]), physics ([14, 15, 16, 17, 18, 19, 20]), and neural networks ([21, 22]) literature. These studies were motivated by a number of examples observed in nature:

- In certain parts of south-east Asia, thousands of male fireflies congregate in trees and flash in synchrony at night [23].
- Pacemaker cells of the heart on average cause 80 contractions a minute during a person's lifetime [24].
- The insulin-secreting cells of the pancreas [25].

For further information and examples, see [26, 27, 28, 29], and the references therein.

A number of models have been proposed to explain the emergence of synchrony, but perhaps one of the most successful and well known is the model of *pulse-coupled*

oscillators by Mirollo and Strogatz [26], based on dynamical systems theory. Consider a function $f : [0,1] \to [0,1]$ that is smooth, monotone increasing, concave down (i.e., $f' > 0$ and $f'' < 0$), and is such that $f(0) = 0$ and $f(1) = 1$. Consider also a phase variable ϕ such that $\partial\phi/\partial t = \frac{1}{T}$, where T is the period of a cycle. Then, each element in a group of N oscillators is described by a state variable $x_i \in [0,1]$ and a phase variable $\phi_i \in [0,1]$ as follows:

- In isolation, $x_i(t) = f(\phi_i(t))$.
- If $\phi_i(t) = 0$ then $x_i(t) = 0$, and if $\phi_i(t) = 1$ then $x_i(t) = 1$.
- When $x_i(t_0) = 1$ for any of the i's and some time t_0, then for all other $1 \leq j \leq N$, $j \neq i$

$$\phi_j(t_0^+) = \begin{cases} f^{-1}(x_j(\phi_j(t_0)) + \varepsilon_i), & x_j(\phi_j(t_0)) + \varepsilon_i \leq 1 \\ 1, & x_j(\phi_j(t_0)) + \varepsilon_i > 1, \end{cases}$$

  where t_0^+ denotes an infinitesimal amount of time after t_0. That is, oscillator i reaching the end of a cycle causes the state of all other oscillators to increase by the amount ε_i, and the phase variable to change accordingly.

The state variable x_i can be thought of as a voltage. Charge is accumulated over time according to the nonlinearity f and it discharges once it reaches full charge, resetting the charging process. Upon discharging, it causes all other charges to increase by a fixed amount of ε_i, up to the discharge point. For this model, it is proved in [26] that for all N and for almost all initial conditions, the system eventually becomes synchronized.

In [30], this convergence to synchrony result was extended to networks that were not fully connected. Kuramoto [31] further generalizes the convergence result in [26] for identical oscillators subject to small noise and small coupling in

asymptotically large populations. Senn and Urbanczik [32] were then able to establish conditions on the heterogeneity of N non-leaky integrate-and-fire neurons that guarantee a convergence to a fully synchronous state.

The convergence to a synchronous state is clearly desirable for synchronization in networks and, thus, recently there has been much work in applying these mathematical models of natural phenomena to engineered networks. In [33], theoretical and simulation results suggested that such a technique could be adapted to communication and sensor networks. Experimental validation for the ideas of [26] was obtained in [34] where the authors implemented the Reachback Firefly Algorithm (RFA) on TinyOS-based motes. They were able to successfully show that a network of nodes started in arbitrary initial conditions would eventually become synchronized.

What we notice about these pulse-coupled oscillator models is that the behavior of a node is influenced by many neighbors. This characteristic seems to suggest that pulses are a mechanism through which spatial averaging may be used. However, the problem with these emergent synchronization results is that the fundamental theory assumes all nodes are nearly the same and, thus, these convergence techniques are not ideal for nodes with arbitrary clock skew. For example, results from [33] and [34] show that the convergence results may partially hold when nodes have approximately the same firing period, but the authors of [34] explain that clock skew will degrade synchronization performance.

As a result, we draw ideas from both traditional networking techniques and pulse-coupled oscillator models. As in existing synchronization protocols, each node will still use the time averaging of a sequence of timing data points to estimate its skew and offset relative to the reference time scale. However, spatial averaging

through the use of pulses will be used to improve each timing data point received by the node. Recall that in traditional techniques, each timing data point is constructed with information from only *one* neighboring node. In our technique that incorporates spatial averaging, the timing data point will be constructed with timing information from *many* neighboring nodes. This means that the timing data point will be significantly less noisy, thus improving the overall skew and offset estimates. Since our technique uses information from many neighboring nodes, we call the method *cooperative time synchronization*.

As a final note, different types of cooperation have been considered for other aspects of wireless network operation. In extracting information from a deployed network, nodes may be able to cooperatively transmit information [35],[36],[37],[38]. As well, cooperation can be effectively employed in broadcasting information to the deployed network [39],[40].

1.5 Spatial Averaging Example

To see how it is actually possible to achieve spatial averaging using pulses, let us consider an example. Assume each node in a network has a sequence of m timing data points that occur at integer values of the reference time. This means that each node has a vector of m time values in its local clock that are known to be occurring at consecutive integer values of the reference time. Using this vector of times, each node can estimate the next integer value of the reference time in its own time scale. Each node will then transmit a pulse at that estimated time. Any node j in the network will then see the following aggregate signal generated by

pulses sent from nodes surrounding node j:

$$A_{j,N}(t) = \sum_{i=1}^{N} A_i K_{j,i} p(t - \tau_0 - T_i)$$

where

- $p(t)$: transmitted pulse
- τ_0: the desired pulse transmit time (integer value of reference time)
- T_i: random timing error
- $K_{j,i}$: amplitude loss in the signal transmitted by the ith node
- A_i: scaling constant

Is it possible to use this aggregate signal $A_{j,N}(t)$ to provide improved timing data?

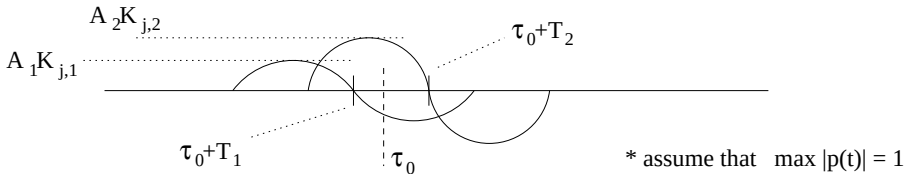

Figure 1.3: Two pulses are transmitted and the pulse transmission times (zero-crossings) do not coincide with the ideal transmission time τ_0 due to timing errors. The aggregate may provide a zero-crossing closer to τ_0.

If we further assume that $p(t)$ is odd shaped and the zero-crossing is the pulse transmission time, then an aggregate signal with two pulses will be the sum of the shifted pulses shown in Fig. 1.3. We see that the pulse transmission times do not coincide with the ideal transmission time τ_0 due to timing errors, but it seems very likely that the aggregate signal will make a zero-crossing much closer to τ_0. If this is indeed the case, then perhaps the aggregate zero-crossing will provide much better information about the location of τ_0.

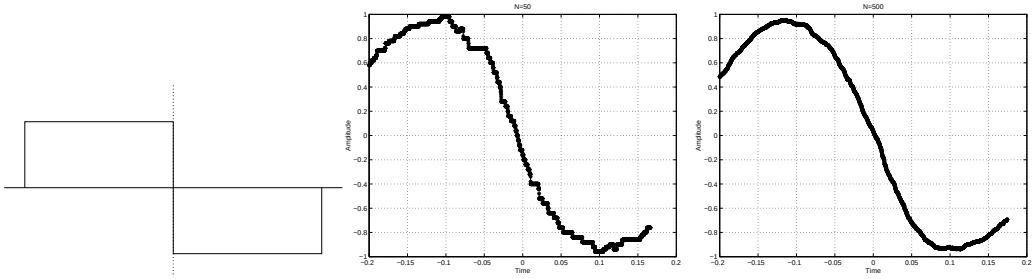

Figure 1.4: Left: The basic transmitted pulse. Middle: The aggregate signal with more pulses makes a zero-crossing close to the ideal transmission time at the center of the x-axis. Right: The aggregate signal with even more pulses makes an even better zero-crossing.

It turns out that under certain conditions, the aggregate signal does provide a zero-crossing that is much closer to τ_0. As shown in Fig. 1.4, if each node transmits an odd-shaped pulse and the zero-crossing is the transmission time, then with increasing numbers of pulses, the aggregate signal makes a zero-crossing closer and closer to τ_0. Therefore, we see that by taking the aggregate signal zero-crossing we can get a better approximation of the ideal transmission time. Using a sequence of these observations would then allow us to make an improved estimate of clock skew. Thus, we see that it is possible to construct aggregate signals that provide improved timing information and use a collection of these observations to estimate synchronization parameters.

1.6 Summary of Contributions

This thesis develops a comprehensive understanding of the use of spatial averaging in time synchronization. We consider all aspects of the system from proposing a general system model to developing a synchronization protocol for cooperative time synchronization using spatial averaging.

In Chapter 2 we begin by establishing a general system model for large-scale, high density networks. The random pathloss and delay models are designed with high density networks specifically in mind. We also show that the model is in fact a generalization of the popular Mirollo and Strogatz model for pulse-coupled oscillators.

Using the proposed general system model, we begin the study of cooperative time synchronization in Chapter 3 by considering spatial averaging in asymptotically dense networks. Since spatial averaging should allow for better synchronization when nodes have a larger number of neighbors, in an infinitely dense network, some sort of perfect synchronization should be possible. In Chapter 3 we show that perfect synchronization is indeed possible in the limiting case. We find that it is possible for the network to cooperatively generate a time synchronization signal that specifies the integer values of the reference time which can be seen by every single node in the network. Thus all nodes will have a common sequence of synchronization events.

The asymptotic synchronization technique is modified for networks with a finite number of nodes and in Chapter 4 we study the synchronization performance through the use of simulations. We find that synchronization can not be maintained simply by using the asymptotic technique in finite sized networks. However, introducing minimal feedback will allow the network to remain synchronized.

In Chapter 5, we provide a characterization of the performance improvement achievable by using spatial averaging in networks with a finite number of nodes. In that chapter, nodes are synchronized by estimating their skew and offset relative to the reference clock. Through the use of analysis and simulations, we show that the mean squared error of the skew and offset estimates can be significantly decreased

by increasing the number of cooperating nodes.

Finally, in Chapter 6, we discuss the generality of cooperative time synchronization and stress the fact that cooperative time synchronization through the use of spatial averaging provides an alternative technique for improving synchronization performance. It allows for a new trade-off between synchronization performance and node density and this new ability will provide added flexibility in designing future networks. Future research directions are also considered in Chapter 6.

CHAPTER 2

GENERAL SYSTEM MODEL

2.1 Introduction

The first step in studying the use of spatial averaging to improve time synchronization is the development of a system model for modeling large-scale, high density networks. There are four key components of the model: the clock, the pathloss model, the propagation delay model, and the pulse-connection function. Existing pulse-coupled oscillator models do not have a concept of a clock, but since we want to allow our nodes to be able to use many pulse arrivals over time, a clock is necessary. Our clock model comes from models used in networking applications. The signal magnitude is essential for our study of cooperative time synchronization since we are using observations from the aggregate signal. It is reasonable to expect that the aggregate signal magnitude affects the observed statistic, so we model pathloss. Propagation delay is of interest since in networks where the nodes have large transmission ranges, propagation delay becomes non-negligible. We want to study the affects of propagation delay on spatial averaging so we need a way to quantify the delay. Lastly, the pulse-connection function allows us to design the behavior of the nodes.

In Section 2.2 we first develop a clock model that describes the clock at any node in terms of a reference clock. Next, we develop two signal propagation models, one that only captures signal magnitude (Section 2.3) while the other describes both signal magnitude and propagation delay (Section 2.4). The propagation models are designed to work especially well for high density networks. The pulse-connection function is described in Section 2.5. In Section 2.6, we show that our model is

a generalization of the Mirollo and Strogatz pulse-coupled oscillator model. This means that our model is a generalization of existing models and is not only of interest for the study of our cooperative synchronization technique; the model allows for the study of a larger class of synchronization problems.

2.2 Clock Model

We consider a network with N nodes uniformly distributed over a fixed finite area. The behavior of each node i is governed by a clock c_i that counts up from 0. The introduction of c_i is important since it provides a consistent time scale for node i. By maintaining a table of pulse-arrival times, node i can utilize the arrival times of many pulses over an extended period of time.

The clock of one particular node in the network will serve as the reference time and to this clock we wish to synchronize all other nodes. We will call the node with the reference clock node 1 and the clocks of other nodes are defined relative to the clock of node 1. We never adjust the frequency or offset of the local clock c_i because we wish to maintain a consistent time scale for node i.

The clock of node 1, c_1, will be defined as $c_1(t) = t$ where $t \in [0, \infty)$. Taking c_1 to be the reference clock, we now define the clock of any other arbitrary node i, c_i. We define c_i as

$$c_i(t) = \alpha_i(t - \bar{\Delta}_i) + \Psi_i(t), \tag{2.1}$$

where

- $\bar{\Delta}_i$ is an unknown offset between the start times of c_i and c_1.
- $\alpha_i > 0$ is a constant and for each i, $\alpha_i \in [\alpha_{low}, \alpha_{up}]$ where $\alpha_{up}, \alpha_{low} > 0$ are finite. This bound on α_i means that the frequency offsets between any two

nodes can not be arbitrarily large.

- $\Psi_i(t)$ is a stochastic process modeling random timing jitter.

It is important to note that node 1 does not have to be special in any way; its clock is simply a reference time with which to define the clocks of the other nodes. This means that our clock model actually describes the relative relationship of all the clocks in the network by using an arbitrary node's clock as a reference.

2.3 Pathloss Only Model

2.3.1 A Random Model for Pathloss

For the study of cooperative time synchronization, nodes cooperatively generate signals for synchronization. Thus, we will be particularly interested in the aggregate signal magnitude at a receiving node and not so much in the particular signal contribution from each surrounding node. With this in mind, we can develop a random model for pathloss that, for dense networks, gives the appropriate aggregate signal magnitude at any receiving node j. Such a model is ideal for our situation since we are studying high density networks.

We start with a general pathloss model $K(d)$, where $0 \leq K(d) \leq 1$ for all distances $d \geq 0$, that is non-increasing and continuous. $K(d)$ is a fraction of the transmitted magnitude seen at distance d from the transmitter. For example, if the receiver node j is at distance d from node i, and node i transmits a signal of magnitude A, then node j will hear a signal of magnitude $AK(d)$. We derive $K(d)$ from a power pathloss model since any pathloss model captures the *average* received power at a given distance from the transmitter. This average received power is perfect for modeling received signal magnitudes in our problem setup

since we are considering dense networks. Due to the large number of nodes at any given distance d from the receiver, using the average received magnitude at distance d as the contribution from each node at that distance will give a good modeling of the amplitude of the aggregate waveform.

The random pathloss variable K_j will be derived from $K(d)$. To understand how K_j and $K(d)$ are related, we give an intuitive explanation of the meaning of K_j as follows: the $\Pr[K_j \in (k, k+\Delta)]$ is the fraction of nodes at distances d from node j such that $K(d) \in (k, k+\Delta)$, where Δ is a small constant. This means that, roughly speaking, for any given scaling factor $K_j = k$, $f_{K_j}(k)\Delta$ is the fraction of received signals with magnitude scaled by approximately k, where $f_{K_j}(k)$ is the probability density function of K_j. Thus, if we scale the transmit magnitude A from every node i by an independent K_j, then as the number of nodes, N, gets large, node j will see $Nf_{K_j}(k)\Delta$ signals of approximate magnitude Ak, and this holds for all k in the range of K_j. This is because taking a large number of independent samples from a distribution results in a good approximation of the distribution.

Thus, this intuition tells us that scaling the magnitude of the signal transmitted from every node i by an independent sample of the random variable K_j gives an aggregate signal at node j that is the same magnitude as a signal generated using $K(d)$ directly. Even though the signals from two nodes at the same distance from a receiver have correlated magnitudes, we do not care about the signal magnitude from any particular node. We only care that, for any given scaling factor k, an appropriate fraction of the signals received at node j are scaled by k. For a receiving node j, we choose therefore to work with the random variable K_j instead of directly with $K(d)$ because, for the goals of this thesis, doing so has two major advantages:

(a) we can obtain desirable limit results by placing very minimal restrictions on the distribution of the K_j's (and hence on $K(d)$) and (b) we can apply the strong law of large numbers to carry out our analysis.

2.3.2 Definition of K_j

From the above intuition we can define the cumulative distribution function of K_j as

$$F_{K_j}(k) = \Pr(K_j \leq k) = \begin{cases} 0 & k \in (-\infty, 0) \\ \frac{A_T - A(j,\bar{r}(k))}{A_T} = 1 - \frac{A(j,\bar{r}(k))}{A_T} & k \in [0,1] \\ 1 & k \in (1, \infty) \end{cases} \tag{2.2}$$

where

- A_T is the total area of the network,
- $A(j, a)$ is the area of the network contained in a circle of radius a centered at node j,
- $\bar{r}(k) = \sup\{d : K(d) > k\}$.

From the above discussion we see that the distribution of K_j is only a function of node j, the receiving node. We illustrate the relationship among node j, $K(d)$, $\bar{r}(k)$, and $F_{K_j}(k)$ in Fig 2.1. We sometimes write $K_{j,i}$ with i used to index each node surrounding node j. For a given j, $K_{j,i}$'s are independent and identically distributed (i.i.d.) with a cumulative distribution function given by (2.2) for all i.

We assume that K_j has the following properties:

- K_j is independent from $\Psi_l(t)$ for all j, l, and t.

- $0 \leq K_j \leq 1$, $0 < E(K_j) \leq 1$, and $\mathrm{Var}(K_j) \leq 1$.

The requirements on the random variable K_j places restrictions on the model $K(d)$. Any function $K(d)$ that yields a K_j with the above requirements can be used to model pathloss.

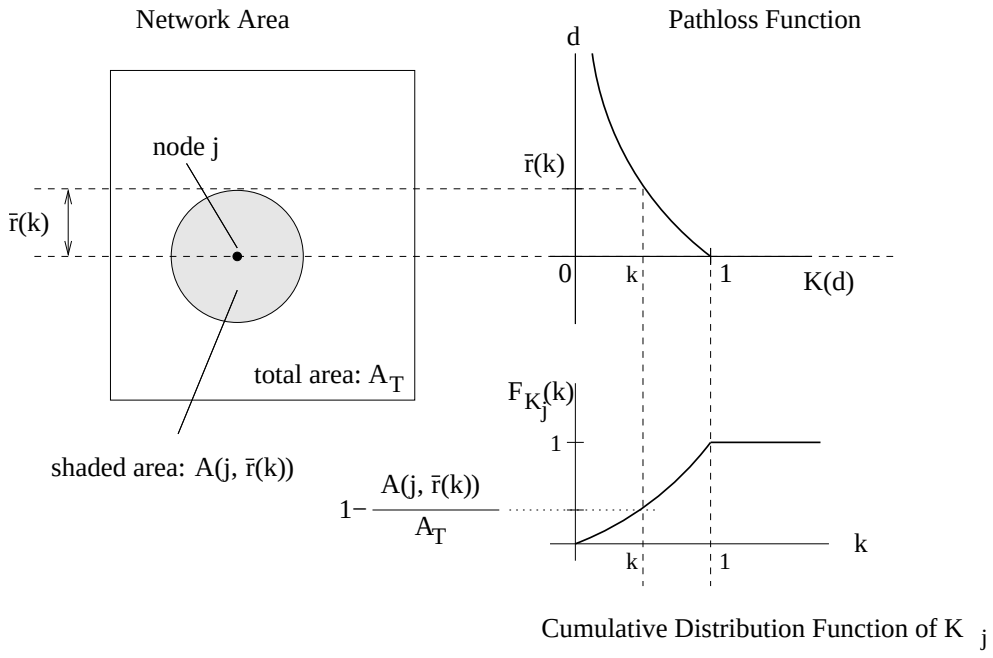

Figure 2.1: An illustration of the cumulative distribution function $F_{K_j}(k)$ is shown in the bottom-right figure. For a given scaling value $k \in [0, 1]$, $F_{K_j}(k)$ is defined to be $1 - (A(j, \bar{r}(k))/A_T)$, where the relationship between $\bar{r}(k)$ and k is shown in the top-right figure. The area $A(j, \bar{r}(k))$ and its relation to node j is shown in the top-left figure.

2.4 Delay and Pathloss Model

In this section we develop a more complex model to simultaneously model propagation delay and pathloss. This leads to the joint development of the delay random variable D_j and a corresponding pathloss random variable K_j.

2.4.1 Correlation Between Delay and Pathloss

Since we want to develop a model for both pathloss and time delay, we start by keeping the pathloss function $K(d)$ defined in Section 2.3. The general delay model assumes a function $\delta(d)$ that models the time delay as a function of distance. $\delta(d)$ describes the time, in terms of c_1, that it takes for a signal to propagate a distance d. For example, if node i and node j are distance d_0 apart, then a pulse sent by node i at time $c_1 = 0$ will be seen at node j at time $c_1 = \delta(d_0)$. We make the reasonable assumption that $\delta(d)$ is continuous and strictly monotonically increasing for $d \geq 0$.

As with the pathloss only model, we want to define a delay random variable D_j for each receiving node j. Recall that this means for every node j there is a random variable D_j associated with it since, in general, each node j will see different delays. There is a correlation between the delay random variable D_j and the pathloss random variable K_j. This correlation arises for two main reasons. First, since in Section 2.3 we define $K(d)$ to be monotonically decreasing and continuous, it is possible for $K(d) = 0$ for $d \in [R, \infty)$, $R > 0$. This might be the case for a multi-hop network. In this situation, there will be a set of nodes whose transmissions will never reach node j (i.e. infinite delay) even though according to $\delta(d)$ these nodes should contribute a signal with finite delay. Second, a small K_j value would represent a signal from a far away node. As a result, the corresponding D_j value should be large. Therefore, keeping these two points in mind, we proceed to develop a model for both pathloss and propagation delay.

2.4.2 Definition of D_j and K_j

We define the cumulative distribution function of D_j as

$$F_{D_j}(x) = \Pr(D_j \leq x) = \begin{cases} 0 & x \in (-\infty, 0) \\ \frac{A(j,r'(x))}{A_T} & x \in [0, \delta(R)] \\ a(x - \delta(R)) + \frac{A(j,R)}{A_T} & x \in (\delta(R), \delta(R + \Delta R)] \\ 1 & x \in (\delta(R + \Delta R), \infty) \end{cases} \tag{2.3}$$

where $r'(x) = \sup\{r : \delta(r) \leq x\}$, $\Delta R > 0$ is a constant, $R = \sup\{d : K(d) > 0\}$, and

$$a = \frac{1 - \frac{A(j,R)}{A_T}}{\delta(R + \Delta R) - \delta(R)}.$$

Recall that $A(j, a)$, defined in Section 2.3, is the area of the network contained in a circle of radius a centered at node j and A_T is the total area of the network. Note that R can be infinite.

Using the delay random variable D_j with the cumulative distribution function in (2.3), we define K_j as

$$K_j = K(\delta^{-1}(D_j)), \tag{2.4}$$

where $K(\cdot)$ is the deterministic pathloss function from Section 2.3 and $\delta^{-1} : [0, \infty) \to [0, \infty)$ is the inverse function of the deterministic delay function $\delta(\cdot)$. Note that $\delta^{-1}(\cdot)$ exists since $\delta(\cdot)$ is continuous and strictly monotonically increasing on $[0, \infty)$.

2.4.3 Intuition Behind D_j and K_j

To understand the distribution of D_j, we need to consider the definition of K_j as well. Recall that a signal arriving with delay D_j is scaled by the pathloss

random variable K_j. Let us consider the cumulative distribution in two pieces, $x \in [0, \delta(R)]$ and $x \in (\delta(R), \infty)$. The case for $x \in (-\infty, 0)$ is trivial. First, for $x \in [0, \delta(R)]$, the probability that D_j takes a value less than or equal to x is simply the fraction of the network area around node j such that the nodes are at distances d with $\delta(d) \leq x$. The intuition is the same as that for the development of K_j in Section 2.3. Second, for $x \in (\delta(R), \infty)$, the situation is more complex. Note that a transmitted signal from a node at distance $d \in (R, \infty)$ from j will arrive at node j with infinite delay since $K(d) = 0$ for $d \in (R, \infty)$. Since any delay values in $x \in (\delta(R), \infty)$ correspond to distances $d = \delta^{-1}(x) \in (R, \infty)$, the corresponding scaling value will be zero because K_j and D_j are related by (2.4). As a result, it does not matter what delay values we assign to the fraction of the network area outside a circle of radius R centered at node j as long as their delay value x is such that $\delta^{-1}(x) \in (R, \infty)$. Thus, we can arbitrarily choose a constant ΔR value and construct a piecewise linear portion of the cumulative distribution function of D_j on $x \in (\delta(R), \infty)$. The probability that $D_j \in (\delta(R), \infty)$ will be the fraction of the network area outside a circle of radius R around node j. And since $D_j \in (\delta(R), \infty)$ will have a corresponding K_j value that is zero, this fraction of nodes will not contribute to the aggregate waveform at node j. It is clear that the correlated D_j and K_j random variables work together to accurately model a signal arriving with both pathloss and propagation delay. An illustration of how $K(d)$, $\delta(d)$, node j, and $F_{D_j}(x)$ are related can be found in Fig. 2.2.

We require that D_j is bounded, has finite expectation, and has finite variance for all j. Note that $D_j \geq 0$ by the requirement that $\delta(d) \geq 0$. As well, since the cumulative distribution in (2.3) is continuous, and often absolutely continuous, we assume that D_j has a probability density function $f_{D_j}(x)$. When we write $D_{j,i}$,

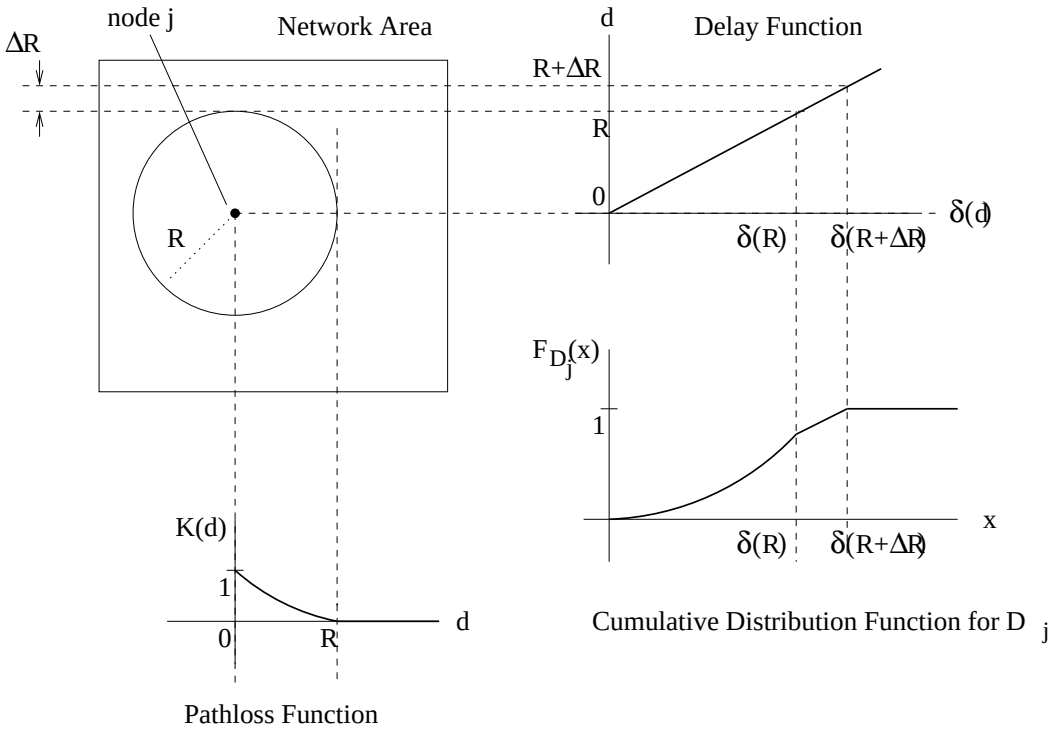

Figure 2.2: From the top-left and bottom-left figures, we can see how $K(d)$ determines the set of nodes surrounding node j that will contribute to the aggregate waveform at node j. This contributing set of nodes is related to $F_{D_j}(x)$ through $\delta(d)$ and this is illustrated in the top-right and bottom-right figures.

the i indexes each node surrounding node j. Thus, the $D_{j,i}$'s are independent and identically distributed in i for a given j and have a cumulative distribution given by (2.3). Using the K_j and D_j developed in this section to simultaneously model pathloss and propagation delay, respectively, we will be able to closely approximate the received aggregate waveform at any node j as N becomes large.

To summarize, we see that our choice of the pathloss and delay random variables will depend on what we want to model. If we only consider pathloss and not propagation delay, then we will use the random variable K_j defined in Section 2.3. If we account for both pathloss and delay, then we will use the delay random

variable D_j in this section (Section 2.4) and the pathloss random variable K_j defined by (2.4).

2.5 Synchronization Pulses and the Pulse-Connection Function

The exchange of pulses is the method through which the network will maintain time synchronization. Each node i will periodically transmit a scaled pulse $A_i p(t)$, where A_i is a constant and $p(t)$, in general, can be any pulse. We call the interval of time during which a synchronization pulse is transmitted a *synchronization phase*.

What each node does with a set of pulse arrival observations is determined by the pulse-connection function $X_{n,i}^{c_i}$ for node i. The pulse-connection function is a function that determines the time, in the time scale of c_i, when node i will send its nth pulse. It can be a function of the current value of $c_i(t)$ and past pulse arrival times. This function basically determines how any node i reacts to the arrival of pulses.

2.6 An Example: Pulse-Coupled Oscillators

The system model that we presented thus far is powerful because it is very general. It is in fact a generalization of existing models and to show this we specialize it to the pulse-coupled oscillator model proposed by Mirollo and Strogatz [26]. Therefore, the results presented in that paper will hold under the simplified version of our model. We see that our more general model allows the study of a larger class of synchronization problems in networks that communicate via pulses.

2.6.1 Model Parameters for Pulse-Coupled Oscillators

In setting up the system model, Mirollo and Strogatz make four key assumptions:

- Pathloss Model: The first assumption is that there is all-to-all coupling among all N oscillators. This means that each oscillator's transmission can be heard by all other oscillators. Thus, for our model we ignore pathloss, i.e. $K(d) = 1$, to allow any node's transmission to be heard by each of the other $N - 1$ nodes.

- Delay Model: The second assumption is that there is instantaneous coupling. This assumption is the same as setting $\delta(d) = 0$. In such a situation we would use our pathloss only model.

- Synchronization Pulses: The third key assumption made in [26] is that there is non-uniform coupling, meaning that each of the N oscillators fire with strengths $\epsilon_1, \ldots, \epsilon_N$. We modify the parameters in our model by making node i transmit with magnitude $A_i = \epsilon_i$. They also assume that any two pulses transmitted at different times will be seen by an oscillator as two separate pulses. In our model, we may choose any pulse $p(t)$ that has an arbitrarily short duration and each node will detect the pulse arrival time and pulse magnitude.

- Clock Model: The fourth important assumption made by Mirollo and Strogatz is that the oscillators are identical but they start in arbitrary initial conditions. We simplify our clock model in (2.1) by eliminating any timing jitter, i.e. $\Psi_i(t) = 0$, and making the clocks identical by setting $\alpha_i = 1$ for $i = 1, \ldots, N$. We leave $\bar{\Delta}_i$ in the model to account for the arbitrary initial

conditions. We also assume that the phase variable in the pulse-coupled oscillator model increases at the same rate as our clock. That is, the time it takes the phase variable to go from zero to one and the time it takes our clock to count from one integer value to the next are the same.

Now that we have identical system models, what remains is to modify our model to mimic the coupling action detailed in [26]. This is accomplished by defining a proper pulse-connection function $X_{n,i}^{c_i}$.

2.6.2 Choice of Pulse-Connection Function

To match the coupling action in [26], we choose $X_{n,i}^{c_i}(z_{k,i}^{c_i}, z_{k-1,i}^{c_i}, \ldots, z_{1,i}^{c_i}, x_{n-1,i}^{c_i})$ that is a function of pulse receive times and also the time of node i's $(n-1)$th pulse transmission time. $z_{k,i}^{c_i}$ is the time in terms of c_i that node i receives its kth pulse since its last pulse transmission at $x_{n-1,i}^{c_i}$. In this case, $X_{n,i}^{c_i}$ will be a function that updates node i's nth pulse transmission time each time node i receives a pulse. Let $X_{n,i}^{c_i}(k) \stackrel{\Delta}{=} X_{n,i}^{c_i}(z_{k,i}^{c_i}, z_{k-1,i}^{c_i}, \ldots, z_{1,i}^{c_i}, x_{n-1,i}^{c_i})$ where it is node i's nth pulse transmission time after observing k pulses since its last pulse transmission. Node i will transmit its pulse as soon as $X_{n,i}^{c_i} \leq c_i(t)$ where $c_i(t)$ is node i's current time. As soon as the node transmits a pulse at $X_{n,i}^{c_i}$ the function will reset and become $X_{n+1,i}^{c_i}(0) = x_{n,i}^{c_i} + 1$. The node is now ready to receive pulses and at its first received pulse, the next transmission time will become $X_{n+1,i}^{c_i}(1)$. $X_{n,i}^{c_i}$ will thus be defined as

$$X_{n,i}^{c_i}(k) \;=\; X_{n,i}^{c_i}(k-1) - [f^{-1}(\epsilon_j + f(z_{k,i}^{c_i} - x_{n-1,i}^{c_i})) - (z_{k,i}^{c_i} - x_{n-1,i}^{c_i})] \quad (2.5)$$

for $k > 0$ and

$$X_{n,i}^{c_i}(0) \;=\; x_{n-1,i}^{c_i} + 1 \quad (2.6)$$

where the pulse received at $z_{k,i}^{c_i}$ is a pulse of magnitude ϵ_j and the function $f : [0,1] \rightarrow [0,1]$ is the smooth, monotonic increasing, and concave down function defined in [26].

Equations (2.5) and (2.6) fundamentally say that each time node i receives a pulse, node i's next transmission time will be adjusted. This is in line with the behavior of the coupling model described by Mirollo and Strogatz since each time an oscillator receives a pulse, its state variable is pulled up by ϵ thus adjusting the time at which the oscillator will next fire. To see how equations (2.5) and (2.6) relate to the coupling model in [26], let us consider an example with two pulse coupled oscillators. Consider two oscillators A and B illustrated in Fig. 2.3. In Fig. 2.3(a), we have that oscillator A is at phase q and oscillator B is just about

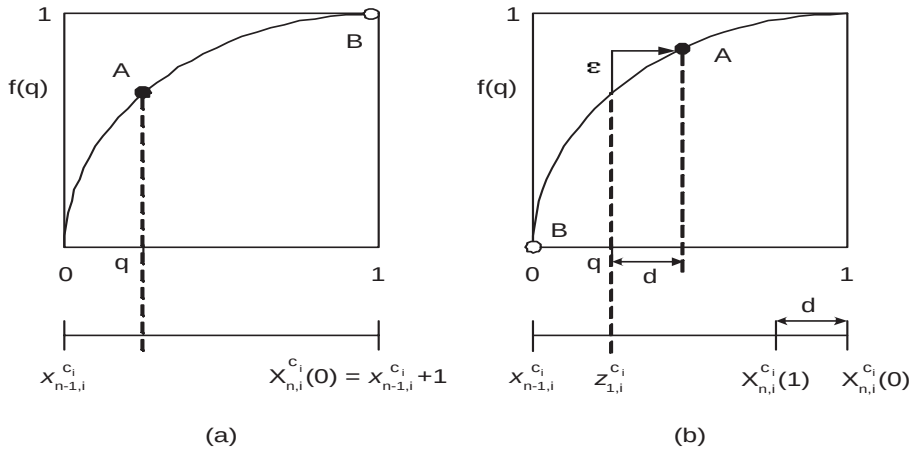

Figure 2.3: We illustrate the connection between the pulse-coupled oscillator coupling model and our clock model. In (a), oscillator B is just about to fire and oscillator A has phase q. In (b), oscillator B fires and increases the phase of oscillator A by d. This d increase in phase effectively decreases the time at which A will next fire. We capture this time decrease by decreasing the firing time of our node by an amount d. Thus, oscillator A and our node will fire at the same time.

to fire. Below the pulse-coupled oscillator model we have a time axis for node i

corresponding to our clock model going from time $x_{n-1,i}^{c_i}$ to $x_{n-1,i}^{c_i} + 1$. Our time axis for node i models the behavior of oscillator A, so we want node i to behave in the same way as oscillator A under the influence of oscillator B. If oscillator B did not exist, then the phase variable q will match our clock in that q reaches 1 at the same time our clock reaches $X_{n,i}^{c_i}(0) = x_{n-1,i}^{c_i} + 1$ and oscillator A will fire at the same time node i fires. In Fig. 2.3(b), oscillator B has fired and has pulled the state variable of oscillator A up by ϵ. This coupling has effectively pushed the phase of oscillator A to $q + d$ and decreased the time before A fires. In fact, the time until oscillator A fires again is decreased by d. We can capture this coupling in our model since we can calculate the lost time d. The time at which oscillator B fires is $z_{1,i}^{c_i}$ and it is clear that $d = f^{-1}(\epsilon + f(z_{1,i}^{c_i} - x_{n-1,i}^{c_i})) - (z_{1,i}^{c_i} - x_{n-1,i}^{c_i})$. Thus, if the time that oscillator A will fire again is decreased by time d due to the pulse of B, then we adjust our node firing time by decreasing the firing time to $X_{n,i}^{c_i}(1) = x_{n-1,i}^{c_i} + 1 - d$. This is exactly the expression in (2.5) for $k = 1$. This relationship between our chosen pulse-connection function and the pulse-coupled oscillator coupling model can be easily extended to N oscillators.

We can see, therefore, that the pulse-coupled oscillator model proposed by Mirollo and Strogatz in [26] is a special case of our model. Our model generalizes this pulse-coupled oscillator model by considering timing jitter, pulses of finite width, propagation delay, non-identical clocks, and an ability to accommodate arbitrary coupling functions.

2.7 Conclusion

In this chapter we have set up a model that will be used for studying spatial averaging in dense networks. The clock model will serve as the basis for all the

networks that we study. The random variables K_j and D_j for pathloss and delay, respectively, will be used extensively in Chapter 3 to study spatial averaging in asymptotically dense networks. The deterministic function $K(d)$ will be central to studying spatial averaging in finite sized networks in Chapter 5.

CHAPTER 3

COOPERATION IN ASYMPTOTICALLY DENSE NETWORKS

3.1 Introduction

Intuitively, spatial averaging takes advantage of the large number of neighboring nodes in order to improve synchronization performance; more neighbors should yield better performance. As a result, it would make sense that in the limit of an infinite number of neighbors, the network would be able to achieve some sort of "perfect" synchronization. In this chapter, we show that an asymptotically dense network can achieve perfect synchronization in the sense that every single node, regardless of distance from the reference clock, can see a sequence of equispaced zero-crossings that occur at integer values of the reference time. The sequence of zero-crossings acts as a perfect sequence of synchronization events.

Just as we could specialize our general model to the pulse-coupled oscillator model of Mirollo and Strogatz in Section 2.6, we start this chapter in Section 3.2 by specializing the model for our proposed synchronization technique. We begin under the assumption of no propagation delay and develop the synchronization technique in two steps. Step 1, we determine the desired properties of the aggregate waveform. In Section 3.3, we specify the model for $A_{j,N}^{c_1}(t)$, the received waveform at any node j. With this signal reception model, in Section 3.4, we prove that given certain characteristics of the model, $A_{j,N}^{c_1}(t)$ has very useful limiting properties. Step 2, we actually engineer the waveform with the desired properties. We design estimators (i.e., the pulse connection function) in Section 3.5 that give $A_{j,N}^{c_1}(t)$ the desired properties and show how $A_{j,N}^{c_1}(t)$ can be effectively used for synchronization. Synchronization with propagation delay is considered separately in Section 3.6.

3.2 System Setup

For our synchronization technique, we specialize the general model by making the following assumptions on α_i and $\Psi_i(t)$ for $i = 1 \ldots N$:

- A characterization of $\{\alpha_i\}$ is given by a known function $f_\alpha(s)$ with $s \in [\alpha_{low}, \alpha_{up}]$ that gives the percentage of nodes with any given α value. Thus, the fraction of nodes with α values in the range s_0 to s_1 can be found by integrating $f_\alpha(s)$ from s_0 to s_1. We assume that $|f_\alpha(s)| < G_\alpha$, for some constant G_α. We keep this function constant as we increase the number of nodes in the network ($N \to \infty$). Given any circle of radius R that intersects the network, the nodes within that circle will have α_i's that are characterized by $f_\alpha(s)$. R is the maximum d such that $K(d) > 0$. This means that the set of nodes that any node j will hear from will have its α_i's characterized by a known function. Note that R can be infinite, and in that case, any node j hears from all nodes in the network. Fundamentally, $f_\alpha(s)$ means that as we increase node density, the new nodes have α parameters that are well distributed in a predictable manner.

- $\Psi_i(t)$ is a zero mean Gaussian process with samples $\Psi_i(t_0) \sim \mathcal{N}(0, \sigma^2)$, for any t_0, and independent and identically distributed samples for any set of times $[t_0, \ldots, t_k]$, k a positive integer. We assume $\sigma^2 < \infty$ and note that σ^2 is defined in terms of the clock of node i. We assume that $\Psi_i(t)$ is Gaussian since the RMS (root mean square) jitter is characterized by the Gaussian distribution [41].

We maintain the full generality of the pathloss model from Section 2.3. Propagation delay using the model in Section 2.4 is considered in Section 3.6. Note that

throughout this work we assume no transmission delay or time-stamping error. This means that a pulse is transmitted at exactly the time the node intends to transmit it. We make this assumption since there will be no delay in message construction or access time [6] because our nodes broadcast the same simple pulse without worrying about collisions. Also, when a node receives a pulse it can determine its clock reading without delay since any time stamping error is small and can be absorbed into the random jitter.

3.3 Signal Reception Model

For our proposed synchronization technique, the aggregate waveform seen by node j at any time t is

$$A_{j,N}^{c_1}(t) = \sum_{i=1}^{N} \frac{A_{max} K_{j,i}}{N} p(t - \tau_o - T_i), \tag{3.1}$$

where $A_{j,N}^{c_1}(t)$ is the waveform seen at node j written in the time scale of c_1 and $A_i = A_{max}/N$ for all i. A_{max} is the maximum transmit magnitude of a node and we scale the transmit magnitude by $1/N$ for two reasons. First, as nodes become smaller, their transmission magnitudes will likely decrease. Second, the $1/N$ factor keeps the total network power bounded as we let N go to infinity. T_i is the random timing error suffered by the ith node, which encompasses the random clock jitter and estimation error. This model says that each node i's pulse transmission occurs at the ideal transmit time τ_0 plus some random error T_i.

There are two comments about (3.1) that we want to make. First, note that even though we sum the transmissions from all N nodes in (3.1), we do not assume that node j can hear all nodes in the network. Recall from the pathloss model that if we have a multi-hop network, then there will be a nonzero probability that

$K_{j,i} = 0$. Thus, node j will not hear from the nodes whose transmissions have zero magnitude. Second, it may be possible that the nodes assume there are $\bar{N} = vN$ nodes in the network while the actual number of functioning nodes is N. In this case, each node will transmit with signal magnitude $A_i = A_{max}/(vN)$ and (3.1) will have a factor of $1/v$. Other than for this factor, however, the theoretical results that follow are not affected.

To model the quality of the reception of $A_{j,N}^{c_1}(t)$ by node j, we model the reception of a signal by defining a threshold γ. γ is the received signal threshold required for nodes to perfectly resolve the pulse arrival time. If the maximum received signal magnitude is less than γ then the node does not make any observations and ignores the received signal waveform. We assume that $\gamma \ll A_{max}$.

In this chapter we will assume that $p(t)$ takes on the shape

$$p(t) = \begin{cases} q(t) & -\tau_{nz} < t < 0 \\ 0 & t = 0, t \leq -\tau_{nz}, t \geq \tau_{nz} \\ -q(-t) & 0 < t < \tau_{nz} \end{cases} \tag{3.2}$$

where $\tau_{nz} > 0$ is expressed in terms of c_1. We assume $q(t) > 0$ for $t \in (-\tau_{nz}, 0)$, $q(t) \neq 0$ only on $t \in (-\tau_{nz}, 0)$, $\sup_t |q(t)| = 1$, and $q(t)$ is uniformly continuous on $(-\tau_{nz}, 0)$. Thus, we see that $p(t)$ has at most three jump discontinuities (at $t = 0, -\tau_{nz}, \tau_{nz}$). τ_{nz} should be chosen large compared to $\max_i \sigma_i^2$, i.e. $\sigma_i^2 << \tau_{nz}$, where σ_i^2 is the value of σ^2 translated from the time scale of c_i to c_1. This way, over each synchronization phase, with high probability a zero-crossing will occur. For each node, the duration in terms of c_1 of a synchronization phase will be $2\tau_{nz}$. Note that we assume τ_{nz} is a value that is constant in any consistent time scale. This means that even though nodes have different clocks, identical pulses are transmitted by all nodes. We define a pulse to be transmitted at time t if

the pulse makes a zero-crossing at time t. Similarly, we define the *pulse receive (arrival) time* for a node as the time when the observed waveform first makes a zero-crossing. A *zero-crossing* is defined for signals that have a positive amplitude and then transition to a negative amplitude. It is the time that the signal first reaches zero.

For the exchange of synchronization pulses, we assume that nodes can transmit pulses and receive signals at the same time. This simplifying assumption is not required for the ideas presented here to hold, but simplifies the presentation. We mention a way to relax this assumption in Section 3.5.4.

In (3.1) and in the discussions above, we have focused on characterizing the aggregate waveform for any one synchronization phase. That is, (3.1) is the waveform seen by any node j for the synchronization phase centered around node 1's transmission at $t = \tau_0$, τ_0 a positive integer. We can, however, describe a synchronization pulse train in the following form,

$$\bar{A}^{c_1}_{j,N}(t) = \sum_{u=1}^{\infty} \sum_{i=1}^{N} \frac{A_{max} K_{j,i}}{N} p(t - \tau_u - T_{i,u}), \tag{3.3}$$

where τ_u is the integer value of t at the uth synchronization phase, and $T_{i,u}$ is the error suffered by the ith node in the uth synchronization phase. We seek to create this pulse train with equispaced zero-crossings and use each zero-crossing as a synchronization event. An illustration of such a pulse train is shown in Fig. 3.1. For simplicity, however, most of the theoretical work is carried out on one synchronization phase.

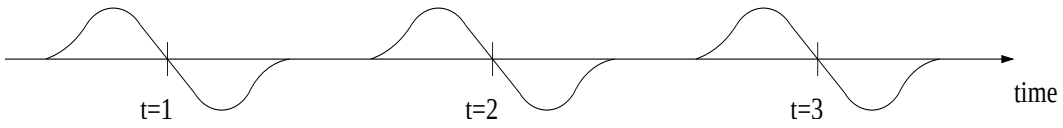

Figure 3.1: An illustration of a pulse train with equispaced zero-crossings. The pulse at each integer value of t is an instance of $A_{j,\infty}(t) = \lim_{N\to\infty} A_{j,N}^{c_1}(t)$ so we see three instances of $A_{j,\infty}(t)$ in the above figure with zero-crossings at $t = 1, 2, 3$. We can control the zero-crossings of $A_{j,\infty}(t)$ and choose to place it on an integer value of t. As a result, we can use these zero-crossings as synchronization events since they can be detected simultaneously by all nodes in the network.

3.4 Desired Structural Properties of the Received Signal

In this section, we characterize the properties of T_i that give us desirable properties in the aggregate waveform. From (3.1), the aggregate waveform seen at each node j in the network has the form

$$A_N(t) = \frac{1}{N}\sum_{i=1}^{N} A_{max} K_i p(t - \tau_0 - T_i) \tag{3.4}$$

We have dropped the j and c_1 for notational simplicity since in this section we deal solely with the received waveform at a node j in the time scale of c_1. As we let the number of nodes grow unbounded ($N \to \infty$), the properties of this limit waveform can be characterized by Theorem 1. These properties will be essential for asymptotic cooperative time synchronization. As a note, in Theorem 1 we present the case for Gaussian distributed T_i but similar results hold for arbitrary zero-mean, symmetrically distributed T_i with finite variance.

Theorem 1 *Let $p(t)$ be as defined in equation (3.2) and $T_i \sim \mathcal{N}(0, \frac{\bar{\sigma}^2}{\alpha_i^2})$ with $\bar{\sigma}^2 > 0$ a constant and $\frac{\bar{\sigma}^2}{\alpha_i^2} < B < \infty$ for all i, B a constant. Also, let K_i be defined as in Section 2.3 and be independent from T_i for all i. Then, $\lim_{N\to\infty} A_N(t) = A_\infty(t)$ has the properties*

- $A_\infty(\tau_0) = 0$,
- $A_\infty(t) > 0$ *for* $t \in (\tau_0 - \tau, \tau_0)$, *and* $A_\infty(t) < 0$ *for* $t \in (\tau_0, \tau_0 + \tau)$ *for some* $\tau < \tau_{nz}$.
- $A_\infty(t)$ *is odd around* $t = \tau_0$, *i.e.* $A_\infty(\tau_0 + \xi) = -A_\infty(\tau_0 - \xi)$ *for* $\xi \geq 0$
- $A_\infty(t)$ *is continuous.* $\triangle$

The properties outlined in Theorem 1 will be key to the synchronization mechanism we describe. We see that $\tau_0 + T_i$ is the transmission time of the ith node, so when we develop the pulse-connection function it will be important to make sure that every node's timing error satisfies the conditions on T_i. The specific value of $\bar{\sigma}^2$ will be determined by our choice of the pulse-connection function. Before we prove Theorem 1 in Section 3.4.2 we develop and motivate a few important related lemmas.

3.4.1 Polarity and Continuity of $A_\infty(t)$

At time $t = \tau_1 \neq \tau_0$, we have that

$$A_N(\tau_1) \;=\; \sum_{i=1}^{N} \frac{A_{max} K_i}{N} p(\tau_1 - \tau_0 - T_i) \;=\; \sum_{i=1}^{N} \frac{1}{N} \bar{M}_i(\tau_1),$$

where $\bar{M}_i(\tau_1) \stackrel{\Delta}{=} A_{max} K_i p(\tau_1 - \tau_0 - T_i)$. We have the mean of $\bar{M}_i(\tau_1)$ being

$$E(\bar{M}_i(\tau_1)) = A_{max} E(K_i) \int p(\tau_1 - \tau_0 - \psi) f_{T_i}(\psi) d\psi,$$

where $f_{T_i}(\psi)$ is the Gaussian pdf

$$f_{T_i}(\psi) = \frac{1}{\frac{\bar{\sigma}}{\alpha_i}\sqrt{2\pi}} \exp\left\{ -\frac{(\psi)^2}{2\frac{\bar{\sigma}^2}{\alpha_i^2}} \right\}.$$

It is clear that the $\bar{M}_i(\tau_1)$'s, for different i's, do not have the same mean and do not have the same variance since the two quantities depend on the α_i value. Since the α_i's are characterized by $f_\alpha(s)$ (defined in Section 3.2), we write the Gaussian distribution for T as

$$f_T(\psi, s) = \frac{1}{\frac{\bar{\sigma}}{s}\sqrt{2\pi}} \exp\left\{ - \frac{(\psi)^2}{2\frac{\bar{\sigma}^2}{s^2}} \right\}.$$

and $\bar{M}_i(\tau_1)$ is in fact a function of s as well, denoted $\bar{M}_i(\tau_1, s)$. Using $f_T(\psi, s)$ and $\bar{M}_i(\tau_1, s)$, the notation makes it clear that we can average over the α_i's that are characterized by $f_\alpha(s)$. We use the results of Lemmas 1 and 2 to prove the polarity result for $A_\infty(t)$ in Section 3.4.2.

Lemma 1 *Given the sequence of independent random variables $\bar{M}_i(\tau_1)$ with $\tau_1 < \tau_0$, $E(\bar{M}_i(\tau_1)) = \mu_i$, and $Var(\bar{M}_i(\tau_1)) = \sigma_i^2$. Then, for all i,*

$$\gamma_2 > \mu_i > \gamma_1 > 0 \tag{3.5}$$

$$\sigma_i^2 < \gamma_3 < \infty, \tag{3.6}$$

for some constants γ_1, γ_2, and γ_3 and

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \bar{M}_i(\tau_1) = \eta(\tau_1) > 0$$

almost surely, where

$$\begin{aligned} \eta(\tau_1) &= \int_{\alpha_{low}}^{\alpha_{up}} E(\bar{M}_i(\tau_1, s)) f_\alpha(s) ds \\ &= A_{max} E(K_i) \int_{\alpha_{low}}^{\alpha_{up}} \int_{-\infty}^{\infty} p(\tau_1 - \tau_0 - \psi) f_T(\psi, s) d\psi f_\alpha(s) ds. \qquad \triangle \end{aligned}$$

Lemma 2 *Given the sequence of independent random variables* $\bar{M}_i(\tau_1)$ *with* $\tau_1 > \tau_0$, $E(\bar{M}_i(\tau_1)) = \mu_i$, *and* $Var(\bar{M}_i(\tau_1)) = \sigma_i^2$. *Then, for all* i,

$$\gamma_2 < \mu_i < \gamma_1 < 0$$

$$\sigma_i^2 < \gamma_3 < \infty,$$

for some constants γ_1, γ_2, *and* γ_3 *and*

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \bar{M}_i(\tau_1) = \eta(\tau_1) < 0$$

almost surely, where

$$\eta(\tau_1) = \int_{\alpha_{low}}^{\alpha_{up}} E(\bar{M}_i(\tau_1, s)) f_\alpha(s) ds. \qquad \triangle$$

The results of Lemma 1 and Lemma 2 are intuitive since given that $p(t)$ is odd and the Gaussian error distribution is symmetric, it makes sense for $A_\infty(t)$ to have properties similar to an odd waveform. Since the proofs of the two lemmas are very similar, we only prove Lemma 1. The proof can be found in Appendix A.1.

Knowing only the polarity of $A_\infty(t)$ is not entirely satisfying since we would also expect that the limiting waveform be continuous. The proof of Lemma 3 is left for Appendix A.2.

Lemma 3 *Using* $p(t)$ *in (3.2),*

$$A_\infty(t) \;=\; \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} A_{max} K_i p(t - \tau_0 - T_i) \;=\; \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \bar{M}_i(t) \;=\; \eta(t)$$

is a continuous function of t, *where*

$$\begin{aligned} \eta(t) &= \int_{\alpha_{low}}^{\alpha_{up}} E(\bar{M}_i(t, s)) f_\alpha(s) ds \\ &= A_{max} E(K_i) \int_{\alpha_{low}}^{\alpha_{up}} \int_{-\infty}^{\infty} p(t - \tau_0 - \psi) f_T(\psi, s) d\psi f_\alpha(s) ds. \qquad \triangle \end{aligned}$$

3.4.2 Proof of Theorem 1

We can proceed in a straightforward manner to show that $A_\infty(\tau_0) = 0$. For $t = \tau_o$,

$$A_N(\tau_0) \;=\; \sum_{i=1}^{N} \frac{A_{max}K_i}{N} p(\tau_0 - \tau_0 - T_i) \;=\; \frac{1}{N}\sum_{i=1}^{N} A_{max}K_i p(-T_i) \;=\; \frac{1}{N}\sum_{i=1}^{N} M_i,$$

where $M_i \triangleq -A_{max}K_i p(T_i)$.

Since our goal is to apply some form of the strong law of large numbers, we first examine the mean of M_i. We have that $E(M_i) = -A_{max}E(K_i)E(p(T_i))$. Furthermore,

$$E(p(T_i)) = \int_{-\infty}^{\infty} p(\psi) f_{T_i}(\psi) d\psi = 0,$$

since $p(\psi)$ is odd and $f_{T_i}(\psi)$ is even because it is zero-mean Gaussian. Thus, $E(M_i) = 0$.

We next consider the variance of M_i:

$$\begin{aligned} \mathrm{Var}(M_i) &= E(M_i^2) - E^2(M_i) = A_{max}^2 E(K_i^2 p^2(T_i)) \\ &= A_{max}^2 E(K_i^2) E(p^2(T_i)) < A_{max}^2 < \infty, \end{aligned}$$

where we have used the fact that $E(K_i^2) \leq 1$ and $|p(t)| \leq 1$.

From the preceding discussion we see that the M_i's are a sequence of zero mean, finite (but possibly different) variance random variables. From Stark and Woods [42], we know that if $\sum_{i=1}^{\infty} \mathrm{Var}(M_i)/i^2 < \infty$, then we have strong convergence of the M_i's:

$$\frac{1}{N}\sum_{i=1}^{N} M_i \to E(M_i),$$

with probability-1 as $N \to \infty$. But it is easy to see that

$$\sum_{i=1}^{\infty} \frac{\mathrm{Var}(M_i)}{i^2} < \sum_{i=1}^{\infty} \frac{A_{max}^2}{i^2} = A_{max}^2 \frac{\pi^2}{6} < \infty,$$

so the condition is satisfied. As a result,

$$A_N(\tau_0) = \frac{1}{N}\sum_{i=1}^{N} M_i \to 0,$$

as $N \to \infty$.

We have that $A_\infty(t)$ is continuous from Lemma 3. Thus, next we need to show that $A_\infty(t) > 0$ for $t \in (\tau_0 - \tau, \tau_0)$, and $A_\infty(t) < 0$ for $t \in (\tau_0, \tau_0 + \tau)$ for some $\tau < \tau_{nz}$. We show the case for $t = \tau_1 \in (\tau_0 - \tau, \tau_0)$ by simply applying Lemma 1. Since Lemma 1 holds for all $\tau_1 < \tau_0$, there clearly exists a τ such $A_\infty(t) > 0$ for $t \in (\tau_0 - \tau, \tau_0)$. The case for $t \in (\tau_0, \tau_0 + \tau)$ comes similarly from Lemma 2.

Lastly, it remains to be shown that $A_\infty(t)$ is odd around $t = \tau_0$. This, however, is evident from the form of $\eta(t)$. Since $f_T(\psi, s)$ is even in ψ about 0 and $p(\psi)$ is odd about 0, it is clear that $\int_\infty^\infty p(t - \tau_0 - \psi) f_T(\psi, s) d\psi$ as a function of t is odd about τ_0. Thus, $\eta(t)$ is odd around τ_0. This then completes the proof for Theorem 1. $\triangle$

3.5 Time Synchronization in Asymptotically Dense Networks

3.5.1 The Use of Estimators in Time Synchronization

In Section 3.4 we determined the structural properties of $A^{c_1}_{j,N}(t)$ so that the limiting waveform will have specific desired properties. Now the task is to actually construct such an aggregate waveform and use it for synchronization. If we can show that as $N \to \infty$ we can recover deterministic parameters that allow for time synchronization, then this result would provide a rigorous theoretical foundation showing that spatial averaging leads to perfect synchronization in the limit

of infinitely dense networks. To simplify the study, we focus on the steady-state time synchronization properties of asymptotically dense networks. In particular, we develop a cooperative technique that constructs a sequence of equispaced zero-crossings seen by all nodes which allows the network to maintain time synchronization indefinitely given that the nodes start with a collection of equispaced zero-crossings. Starting with a few equispaced zero-crossings allows us to avoid the complexities of starting up the synchronization process but still allows us to show that spatial averaging can be used to average out timing errors. If we are able to maintain a sequence of equispaced zero-crossing indefinitely using cooperative time synchronization, then it means that spatial averaging can average out all uncertainties in the system as we let node density grow unbounded. This recovery of deterministic parameters is our desired result. Here, we overview the estimators needed for cooperative time synchronization.

Let $t_{n,i}^{c_k}$ be the time, with respect to clock c_k, that the ith node sees its nth pulse. In dealing with the steady-state properties, we start by assuming that each node i in the network has observed a sequence of m pulse arrival times, $t_{n-1,i}^{c_i}, \ldots, t_{n-m,i}^{c_i}$, that occur at integer values of t, m is an integer. Recall that $t_{n-1,i}^{c_i}, \ldots, t_{n-m,i}^{c_i}$ is defined as a set of m pulse arrival times in the time scale of c_i. Therefore, even though $t_{n-1,i}^{c_i}, \ldots, t_{n-m,i}^{c_i}$ occur at integer values of t (the time scale of c_1), these values are not necessarily integers since they are in the time scale of c_i. Note also that in our model the pulse arrival time is a zero-crossing location. Using these m pulse arrival times, each node i has two distinct, yet closely related tasks. The first task is time synchronization. To achieve time synchronization, node i wants to use these m pulse arrival times to make an estimate of the next integer value of t. This estimator can then be extended to estimate arbitrary times in the future which

gives node i the ability to synchronize to node 1. The second task is that node i needs to transmit a pulse so that the sum of all pulses from the N nodes in the network will create an aggregate waveform that, in the limit as $N \to \infty$, will give a zero-crossing at the next integer value of t. This second task is very significant because if the aggregate waveform gives the exact location of the next integer value of t, then each node i in the network can use this new zero-crossing along with $t^{c_i}_{n-1,i}, \ldots, t^{c_i}_{n-m+1,i}$ to form a set of m zero-crossing locations. This new set can then be used to predict the next zero-crossing location as well as node i's next pulse transmission time. Recall that determining the pulse transmission time is the job of the pulse-connection function $X^{c_i}_{n,i}$. With such a setup, synchronization would be maintained indefinitely. The zero-crossings that always occur at integer values of t would provide node i with a sequence of synchronization events and also illustrate how cooperation is averaging out all random errors.

The waveform properties detailed in Theorem 1 play a central role in accomplishing the nodes' task of cooperatively generating an aggregate waveform with a zero-crossing at the next integer value of t. From (3.4), if the arrival time of any pulse at a node j is a random variable of the form $\tau_0 + T_i$, where τ_0 is the next integer value of t and T_i is zero-mean Gaussian (or in general any symmetric random variable with zero-mean and finite variance), then Theorem 1 tells us that the aggregate waveform will make a zero-crossing at the next integer value of t. This idea is illustrated in Fig. 3.2.

Thus, for achieving time synchronization in an asymptotically dense network we need to address two issues. First, we need to develop an estimator for the next integer value of t given a sequence of m pulse arrival times that occur at integer values of t. We will call this estimator the *time synchronization estimator*,

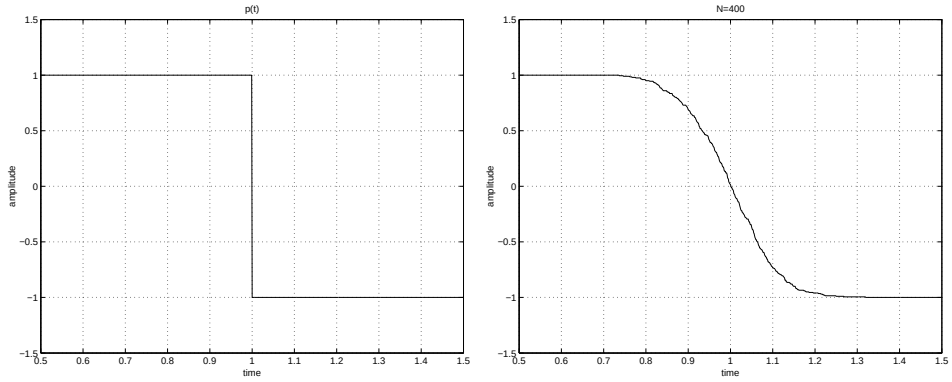

Figure 3.2: Theorem 1 is key in explaining the intuition behind spatial averaging. The pulse $A_{max}p(t-\tau_0)$ is shown in the left figure, with $\tau_0 = 1$ and $A_{max} = 1$. On the right we have a realization of $A_N(t)$ ($N = 400$), and we assume that $K_{j,i} = 1$ (no path loss) and $T_i \sim \mathcal{N}(0, 0.01)$ for all i. As expected from Theorem 1, we notice that the zero-crossing of the simulated waveform is almost exactly at $t = 1$.

denoted as $V_{n,i}^{c_i}$. It uses $t_{n-1,i}^{c_i}, \ldots, t_{n-m,i}^{c_i}$ to determine when the next integer value of t occurs, in the time scale of c_i. Two, we need to develop the pulse-connection function $X_{n,i}^{c_i}$ such that node i's transmitted pulse will arrive at a node j with the random properties described in Theorem 1.

3.5.2 Time Synchronization Estimator

Let us explicitly model the time at an integer value of t in terms of the clock of node i. Assume τ_0 is an integer value of t and at this time, node i will observe its nth pulse. Thus, from (2.1) we have that

$$t_{n,i}^{c_i} = \alpha_i(\tau_0 - \bar{\Delta}_i) + \Psi_i(\tau_0). \tag{3.7}$$

The equation makes use of the clock model of node i (2.1) to tell us the time at clock c_i when node 1 is at τ_0, where τ_0 is an integer in the time scale of c_1. We are also starting with the assumption that the zero-crossing occurring at an integer

value of t is observed by node i at this time.

For the time synchronization estimator, node i will seek to estimate the next integer value of t in the time scale of c_i given $t^{c_i}_{n-1,i}, \ldots, t^{c_i}_{n-m,i}$. Recall that we assume $\Psi_i(t)$ is a zero mean Gaussian process with independent and identically distributed samples $\Psi_i(t) \sim \mathcal{N}(0, \sigma^2)$, for any t. Since we're assuming that the zero-crossings at node i occur at consecutive integer values of t, from (3.7) we see that $\mathbf{T} = [t^{c_i}_{n-m,i}, \ldots, t^{c_i}_{n-1,i}]^T$ is a jointly Gaussian random vector. The random variable $t^{c_i}_{n-m,i}$ is Gaussian with $t^{c_i}_{n-m,i} \sim \mathcal{N}(\alpha_i(\tau_0 - m - \bar{\Delta}_i), \sigma^2)$ for some set of parameters $\vartheta = [\alpha_i, \tau_0 - m, \bar{\Delta}_i]$. We notice that

$$E_\vartheta(t^{c_i}_{n-m+1,i}) = \alpha_i(\tau_0 - m + 1 - \bar{\Delta}_i) = \alpha_i(\tau_0 - m - \bar{\Delta}_i) + \alpha_i.$$

Since each noise sample is independent, we see that the distribution of $\mathbf{T}$ parameterized by ϑ can be written as $\mathbf{T} \sim \mathcal{N}(\mathbf{M}, \Sigma)$ where

$$\mathbf{M} = \begin{bmatrix} \alpha_i(\tau_0 - m - \bar{\Delta}_i) \\ \alpha_i(\tau_0 - m - \bar{\Delta}_i) + \alpha_i \\ \alpha_i(\tau_0 - m - \bar{\Delta}_i) + 2\alpha_i \\ \vdots \\ \alpha_i(\tau_0 - m - \bar{\Delta}_i) + (m-1)\alpha_i \end{bmatrix}$$

and $\Sigma = \sigma^2 \mathbf{I}$.

As a result, for any m consecutive observations, we can simplify notation by using the model

$$\mathbf{Y} = \mathbf{H}\theta + \mathbf{W}, \tag{3.8}$$

where $\mathbf{Y} = [Y_1 \quad Y_2 \ldots Y_m]^T = [t^{c_i}_{n-m,i} \quad t^{c_i}_{n-m+1,i} \ldots t^{c_i}_{n-1,i}]^T$ and

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} \alpha_i(\tau_0 - m - \bar{\Delta}_i) \\ \alpha_i \end{bmatrix}$$

with

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & 1 & 2 & \dots & m-1 \end{bmatrix}^T$$

and $\mathbf{W} = [W_1 \dots W_m]^T$. Since $\Psi_i(t)$ is a Gaussian noise process, $\mathbf{W} \sim \mathcal{N}(0, \Sigma)$ with $\Sigma = \sigma^2 \mathbf{I}$.

With the linear model in (3.8), node i can make an estimate of the next integer value of t by making a uniformly minimum variance unbiased (UMVU) estimate of $\theta_1 + m\theta_2$. From [45] we know the maximum likelihood (ML) estimate of θ, $\hat{\theta}_{ML}$, is given by

$$\hat{\theta}_{ML} = (\mathbf{H}^T \Sigma^{-1} \mathbf{H})^{-1} \mathbf{H}^T \Sigma^{-1} \mathbf{Y} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{Y}.$$

This estimate achieves the Cramer Rao lower bound, hence is efficient. The Fisher information matrix is $I(\theta) = \frac{\mathbf{H}^T \mathbf{H}}{\sigma^2}$ and $\hat{\theta}_{ML} \sim \mathcal{N}(\theta, \sigma^2 (\mathbf{H}^T \mathbf{H})^{-1})$. This means that $\hat{\theta}_{ML}$ is UMVU.

Again from [45], the invariance of the ML estimate tells us that the ML estimate for $\phi = g(\theta) = \theta_1 + m\theta_2$ is $\hat{\phi}_{ML} = \hat{\theta}_{1ML} + m\hat{\theta}_{2ML}$. First, it is clear that $\hat{\phi}_{ML} = \mathbf{C}\hat{\theta}_{ML}$, where $\mathbf{C} = [1 \quad m]$. As a result, we first see that $E_\theta(\hat{\phi}_{ML}) = \mathbf{C} E_\theta(\hat{\theta}_{ML}) = \theta_1 + m\theta_2$ so $\hat{\phi}_{ML}$ is unbiased. Next, to see that $\hat{\phi}_{ML}$ is also minimum variance we compare its variance to the lower bound. First we have

$$\mathrm{Var}_\theta(\hat{\phi}_{ML}) = \mathbf{C}\sigma^2 (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{C}^T = \frac{2\sigma^2(2m+1)}{m(m-1)}.$$

The extension of the Cramer Rao lower bound in [45] to a function of parameters tells us that

$$E_\theta(\|\hat{g} - g(\theta)\|^2) \geq \mathbf{G}(\theta) \mathbf{I}^{-1}(\theta) \mathbf{G}^T(\theta)$$

with $\mathbf{G}(\theta) = (\nabla_\theta g(\theta))^T$. In this case, $\mathbf{G}(\theta) = [1 \quad m]$ so the lower bound to the

mean squared error is

$$\mathbf{G}(\theta)\mathbf{I}^{-1}(\theta)\mathbf{G}^{T}(\theta) = \frac{2\sigma^2(2m+1)}{m(m-1)}.$$

As a result, we see that $\hat{\phi}_{ML}$ is UMVU. Since $\hat{\phi}_{ML}$ is the desired estimate of where the next pulse arrival time will be, it is the time synchronization estimator. Thus,

$$V_{n,i}^{c_i}(\mathbf{Y}) = \mathbf{C}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}. \tag{3.9}$$

Note that

$$V_{n,i}^{c_i}(\mathbf{Y}) = \hat{\phi}_{ML} \sim \mathcal{N}\Big(\phi, \frac{2\sigma^2(2m+1)}{m(m-1)}\Big). \tag{3.10}$$

has a variance that goes to zero as $m \to \infty$.

3.5.3 Time Synchronization with No Propagation Delay

We now need to develop the pulse-connection function so that the conditions for T_i in Theorem 1 are satisfied. Recall we are developing the synchronization technique under the assumption of no propagation delay, i.e. $\delta(d) = 0$. Given a sequence of m pulse arrival times, the time synchronization estimator $V_{n,i}^{c_i}$ given in (3.9) gives each node the ability to predict the next integer value of t. What remains to be considered is the second part of the synchronization process: developing a pulse-connection function $X_{n,i}^{c_i}$ such that the aggregate waveform seen by a node j will have the properties described in Theorem 1.

Let us first consider the distribution of $V_{n,i}^{c_i}$. From (3.10), we have that

$$V_{n,i}^{c_i}(\mathbf{Y}) \sim \mathcal{N}\left(\alpha_i(\tau_0 - m - \bar{\Delta}_i) + m\alpha_i, \frac{2\sigma^2(2m+1)}{m(m-1)}\right).$$

Using (2.1), we can translate $V_{n,i}^{c_i}(\mathbf{Y})$ into the time scale of c_1 as

$$V_{n,i}^{c_i}(\mathbf{Y}) = \alpha_i(V_{n,i}^{c_1}(\mathbf{Y}) - \bar{\Delta}_i) + \Psi_i$$

which gives

$$V_{n,i}^{c_1}(\mathbf{Y}) = \frac{(V_{n,i}^{c_i}(\mathbf{Y}) - \Psi_i)}{\alpha_i} + \bar{\Delta}_i.$$

This means that

$$V_{n,i}^{c_1}(\mathbf{Y}) \sim \mathcal{N}\bigg(\tau_0, \frac{\sigma^2}{\alpha_i^2}\bigg(1 + \frac{2(2m+1)}{m(m-1)}\bigg)\bigg). \tag{3.11}$$

Under our assumption of $\delta(d) = 0$, any transmission by node i will be instantaneously seen by any node j. As a result, the random variable $V_{n,i}^{c_1}(\mathbf{Y})$ will be seen as the pulse arrival time at node j, in the time scale of c_1.

Due to the assumption of no propagation delay, defining $X_{n,i}^{c_1}(\mathbf{Y}) \triangleq V_{n,i}^{c_1}(\mathbf{Y})$ will give us the desired properties in the aggregate waveform. To see this, let us compare the distribution of $X_{n,i}^{c_1}(\mathbf{Y})$ in (3.11) to the assumptions of Theorem 1. Since τ_0 is the ideal crossing time in the time scale of c_1, we have

$$X_{n,i}^{c_1}(\mathbf{Y}) = \tau_0 + T_i.$$

Therefore, we see that

$$\mathrm{Var}(T_i) = \frac{\sigma^2}{\alpha_i^2}\bigg(1 + \frac{2(2m+1)}{m(m-1)}\bigg) = \frac{\bar{\sigma}^2}{\alpha_i^2},$$

where $\bar{\sigma}^2$ from Theorem 1 is

$$\bar{\sigma}^2 = \sigma^2\bigg(1 + \frac{2(2m+1)}{m(m-1)}\bigg).$$

We have shown that using the pulse connection function $X_{n,i}^{c_1}(\mathbf{Y}) \triangleq V_{n,i}^{c_1}(\mathbf{Y})$ satisfies the conditions of Theorem 1. Thus, all the results of the theorem apply.

As a result, we have established a time synchronization estimator $V_{n,i}^{c_1}(\mathbf{Y})$ and a pulse-connection function $X_{n,i}^{c_1}(\mathbf{Y})$. In the case of $\delta(d) = 0$, we have that $X_{n,i}^{c_1}(\mathbf{Y}) \triangleq V_{n,i}^{c_1}(\mathbf{Y})$, or in the time scale of c_i, $X_{n,i}^{c_i}(\mathbf{Y}) \triangleq V_{n,i}^{c_i}(\mathbf{Y})$. When each

node in the network uses the pulse-connection function $X_{n,i}^{c_i}(\mathbf{Y})$ we have a resulting aggregate waveform that has a zero-crossing at the next integer value of t as $N \to \infty$. This fact follows from applying Theorem 1. Thus, we have an asymptotic steady-state time synchronization method that can maintain a sequence of equispaced zero-crossings occurring at integer values of t. An interesting feature of this synchronization technique is that no node needs to know any information about its location or its surrounding neighbors.

3.5.4 No Simultaneous Transmission and Reception

Before ending this section, let us comment on the assumption of simultaneous transmission and reception. One way to relax this assumption is to divide the network into two disjoint sets of nodes, say the odd numbered nodes and the even numbered nodes, where each set is still uniformly distributed over the area. Then, the odd nodes and the even nodes will take turns transmitting and receiving. For example, the odd numbered nodes can transmit pulses at odd values of t and the even numbered nodes will listen. The even numbered nodes will then transmit pulses at the even values of t and the odd numbered nodes will listen. With such a scheme, nodes do not transmit and receive pulses simultaneously, but can still take advantage of spatial averaging. The odd numbered nodes will see an aggregate waveform generated by a subset of the even numbered nodes and the even numbered nodes will receive a waveform cooperatively generated by the odd numbered nodes. Let us take a more detailed look at this scheme.

In Fig. 3.3 we assume that τ_0 is an even integer value of t and use $m = 3$. Each even numbered node will use the aggregate signals occurring at $\tau_0 - 5$, $\tau_0 - 3$, and

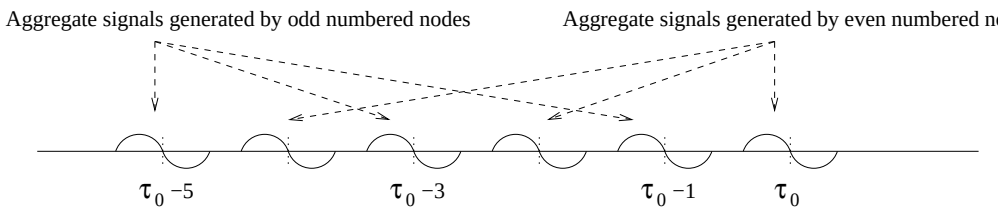

Figure 3.3: In the above figure, we assume τ_0 is an even integer value of t and $m = 3$. Therefore, each even numbered node will turn on its receiver to receive the aggregate signal arriving at times $\tau_0 - 5$, $\tau_0 - 3$, and $\tau_0 - 1$. Using these three received times, it can then estimate the time of τ_0. Thus, the aggregate signal occurring at τ_0 is cooperatively generated by the even numbered nodes and is received by the odd numbered nodes.

$\tau_0 - 1$ to estimate τ_0 and cooperatively generate the aggregate signal at τ_0. The odd numbered nodes will then use the aggregate signals occurring at $\tau_0 - 4$, $\tau_0 - 2$, and τ_0 to generate the aggregate signal at $\tau_0 + 1$. Therefore, the odd and even numbered nodes can take turns transmitting and receiving signals and nodes never need to simultaneously transmit and receive.

Of course, such a setup would require a modification of the estimators used by the nodes. For an arbitrary ideal transmission time τ_0, nodes will receive a vector of m observations $\mathbf{Y}$ with $\mathbf{Y}[l+1] = \alpha_i(\tau_0 + 1 - 2(m-l) - \bar{\Delta}_i) + \Psi_i$ for $l = 0, 1, \ldots, m-1$. With such a mechanism, the $\mathbf{H}$ matrix in equation (3.8) would change to

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 0 & 2 & 4 & \ldots & 2(m-1) \end{bmatrix}^T$$

and θ becomes

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} \alpha_i(\tau_0 + 1 - 2m - \bar{\Delta}_i) \\ \alpha_i \end{bmatrix}.$$

To estimate the location τ_0 in the time scale of c_i, we can proceed as in Sec-

tion 3.5.2:

$$\hat{\theta}_{ML} = (\mathbf{H}^T\Sigma^{-1}\mathbf{H})^{-1}\mathbf{H}^T\Sigma^{-1}\mathbf{Y} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T Y$$

will be distributed $\hat{\theta}_{ML} \sim \mathcal{N}(\theta, \sigma^2(\mathbf{H}^T\mathbf{H})^{-1})$ and $\hat{\theta}_{ML}$ is UMVU. This leads to the UMVU estimate $\hat{\phi}_{ML} = \mathbf{C}\hat{\theta}_{ML}$, where $\mathbf{C} = [1 \quad 2m-1]$, and $E(\hat{\phi}_{ML}) = \mathbf{C}E(\hat{\theta}_{ML}) = \theta_1 + (2m-1)\theta_2$. In this case, the variance of $\hat{\phi}_{ML}$ will be $\mathrm{Var}_\theta(\hat{\phi}_{ML}) = \mathbf{C}\sigma^2(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{C}^T$, and thus we have that

$$V_{n,i}^{c_i}(\mathbf{Y}) = \hat{\phi}_{ML} \sim \mathcal{N}\left(\alpha_i(\tau_0 + 1 - 2m - \bar{\Delta}_i) + (2m-1)\alpha_i, \frac{\sigma^2(2m+1)(2m-1)}{m(m-1)(m+1)}\right).$$

Converted to the time scale of c_1 we have

$$V_{n,i}^{c_1}(\mathbf{Y}) \sim \mathcal{N}\left(\tau_0, \frac{\sigma^2}{\alpha_i^2}\left(1 + \frac{(2m+1)(2m-1)}{m(m-1)(m+1)}\right)\right). \tag{3.12}$$

Comparing equations (3.11) and (3.12), we see that they have the same form. As a result, we can again set $X_{n,i}^{c_i}(\mathbf{Y}) \triangleq V_{n,i}^{c_i}(\mathbf{Y})$ and achieve cooperative time synchronization.

3.6 Time Synchronization with Propagation Delay

We now extend the ideas of cooperative time synchronization to the situation where signals suffer, not only from pathloss, but also propagation delay. It turns out that the effect of propagation delay can also be addressed using the concept we have been using throughout this thesis — spatial averaging.

In this section, we use the pathloss and propagation delay model detailed in Section 2.4. We introduce a time delay function $\delta(d)$. For generality, we explicitly model a multi-hop network where we have a $K(d)$ function that is zero for d greater than some distance R, i.e. $K(d) = 0$ for $d > R$. Such a model implies that the aggregate signal seen at any node j is influenced only by the set of nodes inside

a circle of radius R centered at node j. With this we can effectively divide the network into two disjoint sets, a set of *interior nodes* and a set of *boundary nodes*. An interior node j is defined to be a node whose distance from the nearest network boundary is greater than or equal to R. A boundary node is thus defined to be a node that is a distance less than R away from the nearest network boundary.

We make this distinction since the synchronization technique for each set of nodes is different. Please note that if a pathloss function where $K(d) = 0$ for $d > R$ is unreasonable, then we simply choose R to be infinite and consider all nodes in the network to be boundary nodes.

Using the propagation delay model, $D_{j,i}$ will obviously modify the general received aggregate waveform seen at any node j. In fact, equation (3.1) will now be written as

$$A_{j,N}^{c_1}(t) = \sum_{i=1}^{N} \frac{A_{max} K_{j,i}}{N} p(t - \tau_o - T_i - D_{j,i}). \tag{3.13}$$

For N large, this model will give an accurate characterization of the aggregate waveform seen at node j. From equation (3.13), we see that propagation delay introduces asymmetries in the aggregate signal. We will try to fix the asymmetries and then use spatial averaging.

3.6.1 Conceptual Motivation

From equation (3.13), it is clear that the aggregate waveform will not have a zero-crossing at τ_0 for every node j because of the presence of the $D_{j,i}$ random variables. Therefore, to average out propagation delay, the idea we employ is to have each node introduce a *random* artificial time shift that counteracts the effect of the time delay random variable. More precisely, we want to introduce

another random variable D_{fix} such that $D_{fix} + D_j$ will have zero mean and a symmetric distribution. At the same time, we assume each node knows $K(\cdot)$ and $\delta(\cdot)$ and will also introduce an artificial scaling factor $K_{fix} = K(\delta^{-1}(-D_{fix}))$ to simplify the analysis of the aggregate waveform. This means that instead of using the scaling factor $A_i = A_{max}/N$, each node i will scale its transmitted pulse by $A_i = A_{max}K_{fix}/N$. For the motivation in this section, let us assume that node j is an interior node.

To find the distribution of D_{fix}, we consider the following. D_j has density $f_{D_j}(x)$ and let $f_{D_{fix}}(x)$ be the density of D_{fix}. Since D_j and D_{fix} are independent, we know that the density of $D_T = D_{fix} + D_j$, $f_{D_T}(x)$, will be the convolution of $f_{D_j}(x)$ and $f_{D_{fix}}(x)$. Therefore, by the properties of the convolution function, if we set $f_{D_{fix}}(x) \stackrel{\Delta}{=} f_{D_j}(-x)$, then we have that $f_{D_T}(x)$ is symmetric, i.e. $f_{D_T}(x) = f_{D_T}(-x)$. As well, since D_j has finite expectation, it is easy to see that $E(D_T) = 0$.

Given a sequence of m zero-crossings that we know to be occurring at integers of t, we can still use $V_{n,i}^{c_1}(\mathbf{Y})$ (from (3.9) in the time scale of node 1) as the time synchronization estimator. However, with propagation delay, the pulse-connection function will now be $X_{n,i}^{c_1}(\mathbf{Y}) = V_{n,i}^{c_1}(\mathbf{Y}) + D_{fix,i} = \tau_o + T_i + D_{fix,i}$. With $D_{fix,i}$ and $K_{fix,i}$ included, we can rewrite equation (3.13) as

$$A_{j,N}^{c_1}(t) = \sum_{i=1}^{N} \frac{A_{max}K_{fix,i}K_{j,i}}{N} p(t - \tau_o - T_i - D_{fix,i} - D_{j,i}). \tag{3.14}$$

Note that each node takes an independent sample of D_{fix} so $D_{fix,i}$ are i.i.d. for all i. It is important to see that since D_j has the same distribution for *all* interior nodes j, equation (3.14) holds for every node j that is an interior node. This means that for the network to cooperatively generate the waveform in (3.14) each transmit node i needs to have the following additional knowledge: (1) the distribution of D_{fix} whose density is $f_{D_{fix}}(x) \stackrel{\Delta}{=} f_{D_j}(-x)$, where j is an interior node, and (2)

the functions $K(\cdot)$ and $\delta(\cdot)$ to generate K_{fix}. With this knowledge, we can use equation (3.14) to study the aggregate waveform seen at any interior node j. In fact, we find that the aggregate waveform has limiting properties that are similar to those outlined in Theorem 1. These properties are described in Theorem 2.

Theorem 2 *Let $p(t)$ be as defined in equation (3.2) and $T_i \sim \mathcal{N}(0, \frac{\bar{\sigma}^2}{\alpha_i^2})$ with $\bar{\sigma}^2 > 0$ a constant and $\frac{\bar{\sigma}^2}{\alpha_i^2} < B < \infty$ for all i, B a constant. $K_{j,i}$ and $D_{j,i}$ are defined as in Section 2.4 and $D_{fix,i}$ with density $f_{D_{fix}}(x) \stackrel{\Delta}{=} f_{D_j}(-x)$ is independent from $D_{j,i}$. $K_{fix,i} = K(\delta^{-1}(-D_{fix,i}))$ and let $D_{j,i}$, $D_{fix,i}$, and T_i be mutually independent for all i. Then, for any interior node j with $A_{j,N}^{c_1}(t)$ as defined in (3.14), $\lim_{N\to\infty} A_{j,N}^{c_1}(t) = A_{j,\infty}^{c_1}(t)$ has the properties*

- *$A_{j,\infty}^{c_1}(t)$ is odd around $t = \tau_0$, i.e. $A_{j,\infty}^{c_1}(\tau_0 + \xi) = -A_{j,\infty}^{c_1}(\tau_0 - \xi)$ for $\xi \geq 0$,*
- *$A_{j,\infty}^{c_1}(\tau_0) = 0$.* $\triangle$

The proof of Theorem 2 is left for Appendix A.3.

From the arguments so far, it seems that time synchronization with delay, at least for interior nodes, can be solved simply by modifying the pulse-connection function $X_{n,i}^{c_1}(\mathbf{Y})$ and changing the scaling factor to $A_i = A_{max}K_{fix}/N$. Theorem 2 tells us that the limiting aggregate waveform makes a zero-crossing at the next integer value of t and the waveform is odd. Thus, we can use this zero-crossing as a synchronization event and maintain synchronization in a manner identical to the technique used in the situation without propagation delay. This, however, unfortunately is not the case. In order to implement the above concept, we need to find the random variable, $D_{fix}^{c_i}$, in the time scale of c_i, that corresponds to D_{fix}

such that

$$\begin{aligned}(V_{n,i}^{c_i}(\mathbf{Y}) + D_{fix}^{c_i})^{c_1} &= \frac{V_{n,i}^{c_i}(\mathbf{Y}) + D_{fix}^{c_i} - \Psi_i}{\alpha_i} + \bar{\Delta}_i \\ &= V_{n,i}^{c_1}(\mathbf{Y}) + \frac{D_{fix}^{c_i}}{\alpha_i} \\ &= V_{n,i}^{c_1}(\mathbf{Y}) + D_{fix}.\end{aligned}$$

This means that we need $D_{fix}^{c_i}/\alpha_i = D_{fix}$. However, each node i cannot find $D_{fix}^{c_i}$ that satisfies this since it does not know its α_i.

3.6.2 Time Synchronization of Interior Nodes

Since the ith node does not know its own value of α_i, to do time synchronization with propagation delay we can have each node estimate its α_i value. However, this estimate will not be perfect and we may no longer have the symmetric limiting aggregate waveform described by Theorem 2. This means that the center zero-crossing might occur some ϵ away from τ_0, τ_0 an integer value of t. However, steady-state time synchronization can be maintained if the network can use a sequence of m equispaced zero-crossings that occur at $t = \tau_0 - m + \epsilon, \tau_0 - m + 1 + \epsilon, \tau_0 - m + 2 + \epsilon, \ldots, \tau_0 - 1 + \epsilon$, where τ_0 is an integer value of t, to cooperatively generate a limiting aggregate waveform that has a zero-crossing at $\tau_0 + \epsilon$. In such a situation, the network will be able to construct a sequence of equispaced zero-crossings and maintain the occurrence of these zero-crossings indefinitely. The idea is the same as in the case without propagation delay, but the only difference here would be that the zero-crossings do not occur at integer values of t. Let us give a more formal description of this idea.

Using notation from Section 3.5.2, we start with the assumption that each

interior node i has a sequence of m observations that has the form

$$\alpha_i(\tau_0 - m + l + \epsilon - \bar{\Delta}_i) + \Psi_i, \tag{3.15}$$

where $l = 0, 1, \ldots, m-1$ and ϵ is known. To develop the time synchronization estimator $V_{n,i}^{c_i}(\mathbf{Y})$ and the pulse-connection function $X_{n,i}^{c_i}(\mathbf{Y})$, we consider the observations made by each node. If we assume that each node knows the value of ϵ, the vector of observations can be written as in (3.8)

$$\mathbf{Y} = \bar{\mathbf{H}}\theta + \mathbf{W},$$

where the matrix $\bar{\mathbf{H}}$ in this case is

$$\bar{\mathbf{H}} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ \epsilon & 1+\epsilon & 2+\epsilon & \ldots & m-1+\epsilon \end{bmatrix}^T.$$

Using this model, we can follow the development in Section 3.5.2 to find the the time synchronization estimator

$$V_{n,i}^{c_i}(\mathbf{Y}, \epsilon) = \mathbf{C}(\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T\mathbf{Y}, \tag{3.16}$$

where $\mathbf{C} = [1 \quad m]$. This estimator will give each node the ability to estimate the next integer value of t. Note that the variance of the time synchronization estimator is

$$\mathrm{Var}_\theta(V_{n,i}^{c_i}(\mathbf{Y}, \epsilon)) = \mathbf{C}\sigma^2(\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\mathbf{C}^T = \sigma^2\left(\frac{2(2m+1)}{m(m-1)} + \frac{12\epsilon(\epsilon-1-m)}{(m-1)m(m+1)}\right). \tag{3.17}$$

Using the time synchronization estimator, we can choose the pulse-connection function as

$$X_{n,i}^{c_i}(\mathbf{Y}) = V_{n,i}^{c_i}(\mathbf{Y}, \epsilon) + \hat{\alpha}_i D_{fix} = V_{n,i}^{c_i}(\mathbf{Y}, \epsilon) + D_{fix}^{c_i}, \tag{3.18}$$

where each time node i makes the estimate $V_{n,i}^{c_i}(\mathbf{Y}, \epsilon)$ it also estimates $\hat{\alpha}_i$ as

$$\hat{\alpha}_i = \bar{\mathbf{C}}(\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T\mathbf{Y},$$

$\bar{\mathbf{C}} = [0 \quad 1]$. We find that $\hat{\alpha}_i \sim \mathcal{N}(\alpha_i, 12\sigma^2/((m-1)m(m+1)))$. Since, from Section 3.6.1, we know we want $D_{fix}^{c_i}/\alpha_i = D_{fix}$, we have set $D_{fix}^{c_i} \triangleq \hat{\alpha}_i D_{fix}$. Notice that since $D_{fix}^{c_i}$ is simply a realization of D_{fix} multiplied by node i's estimate of α_i, node i can use the realization of D_{fix} and find $K_{fix} = K(\delta^{-1}(-D_{fix}))$.

With our choice of $X_{n,i}^{c_i}(\mathbf{Y})$ in (3.18), we see that

$$(V_{n,i}^{c_i}(\mathbf{Y}, \epsilon) + D_{fix}^{c_i})^{c_1} \quad = \quad V_{n,i}^{c_1}(\mathbf{Y}, \epsilon) + Z_i D_{fix} \quad = \quad \tau_0 + T_i + Z_i D_{fix},$$

where $Z_i \sim \mathcal{N}(1, 12\sigma^2/(\alpha_i^2(m-1)m(m+1)))$, and $\tau_0 + T_i = V_{n,i}^{c_1}(\mathbf{Y}, \epsilon)$. Because of the random factor Z_i, we see that $D_T = Z_i D_{fix} + D_j$ is no longer a symmetric distribution. As a result, the limiting aggregate waveform

$$A_{j,\infty}^{c_1}(t) = \lim_{N\to\infty} A_{j,N}^{c_1}(t) = \lim_{N\to\infty} \sum_{i=1}^{N} \frac{A_{max} K_{fix,i} K_{j,i}}{N} p(t - \tau_o - T_i - Z_i D_{fix,i} - D_{j,i}) \tag{3.19}$$

may not have a zero-crossing at $t = \tau_0$.

Thus, if we can find an ϵ such that each node i using a set of observations of the form (3.15) allows the network to cooperatively generate the waveform in (3.19) that has its zero-crossing occurring at $t = \tau_0 + \epsilon$ (in the time scale of c_1), then we have steady-state time synchronization. This is because the network would be able to use a sequence of m observations to generate the next observation that gives the same information as any of the previous observations. Thus, by always taking the m most recent observations, the process can continue forever and maintain synchronization. Each node i would need to know distribution of D_{fix}, the value of ϵ, and the functions $K(\cdot)$ and $\delta(\cdot)$. Therefore, we find that steady-state time

synchronization of the interior nodes is possible under certain conditions. As a note, no interior node needs to know any location information.

3.6.3 Time Synchronization of Boundary Nodes

Before we consider the synchronization of boundary nodes, we note that the key requirement for each boundary node i is to have a pulse-connection function given in equation (3.18). The reason that this must be the pulse-connection for every boundary node i is because the analysis for the interior nodes assumes that the aggregate waveform seen by any interior node j is created by pulse transmissions occurring at a time determined by (3.18). Since the aggregate waveform seen by some interior nodes are created by pulse transmissions from boundary nodes, each boundary node must have the appropriate pulse-connection function. This requirement, however, proves to be extremely problematic and reveals a limitation of the elegant technique of averaging out timing delay when we come to boundaries of the network.

The problem comes because $D_{fix} + D_{j,i}$ already does not have a symmetric distribution if j is a boundary node. Recall that $f_{D_{fix}}(x) = f_{D_j}(-x)$ when j is an interior node and $f_{D_j}(x) = f_{D_l}(x)$ when j and l are both interior nodes. However, $f_{D_j}(x) \neq f_{D_l}(x)$ when j is an interior node and l is a boundary node. As a result, $D_{fix} + D_{j,i}$ is no longer symmetric if j is a boundary node. In fact, it is clear that the distribution of $D_{fix} + D_{j,i}$ is a function of node j's location near the boundary. Because of this additional asymmetry, let us assume for a moment that the sequence of zero-crossings observed by boundary node i occur ϵ_i away from an integer value of t. That is, if every node in the network, including the boundary nodes, transmitted a sequence of pulses where each pulse was sent according to

(3.18), then boundary node i would observe the sequence of observations

$$\alpha_i(\tau_0 - m + l + \epsilon_i - \bar{\Delta}_i) + \Psi_i, \tag{3.20}$$

where $l = 0, 1, \ldots, m-1$ and ϵ_i is known.

This boundary node i could then use the time synchronization estimator given by (3.16) but where the matrix $\bar{\mathbf{H}}$ is now replaced with $\bar{\mathbf{H}}_i$

$$\bar{\mathbf{H}}_i = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ \epsilon_i & 1+\epsilon_i & 2+\epsilon_i & \ldots & m-1+\epsilon_i \end{bmatrix}^T.$$

Thus, for this boundary node i we have

$$V_{n,i}^{c_i}(\mathbf{Y}, \epsilon_i) = \mathbf{C}(\bar{\mathbf{H}}_i^T \bar{\mathbf{H}}_i)^{-1} \bar{\mathbf{H}}_i^T \mathbf{Y}.$$

In this case, however, the variance of the time synchronization estimator depends on ϵ_i

$$\mathrm{Var}_\theta(V_{n,i}^{c_i}(\mathbf{Y}, \epsilon_i)) = \sigma^2 \left(\frac{2(2m+1)}{m(m-1)} + \frac{12\epsilon_i(\epsilon_i - 1 - m)}{(m-1)m(m+1)} \right).$$

The fact that the variance depends on ϵ_i is the root of the problem. The pulse-connection function

$$X_{n,i}^{c_i}(\mathbf{Y}) = V_{n,i}^{c_i}(\mathbf{Y}, \epsilon_i) + \hat{\alpha}_i D_{fix},$$

is *not* the same as that given by (3.18).

To correct for this, we can make the strong assumption that each boundary node i knows is own α_i. We address the reasoning behind this assumption in Section 3.6.4. If we use this assumption, then each boundary node i can get an observation sequence of the form (3.15) simply by adding $\alpha_i(\epsilon - \epsilon_i)$ to each of the m observations of the form given in (3.20), where we assume that node i knows both ϵ and ϵ_i. With such an observation sequence, boundary node i will

have the time synchronization estimator (3.16) and, more importantly, the pulse-connection function (3.18). Thus, maintaining time synchronization for the case of propagation delay would be possible.

What we have then is that boundary node synchronization would require only the boundary nodes to know their α_i parameters. With this strong assumption only for the boundary nodes, the network is effectively synchronized. Even though the boundary nodes do not see the same zero-crossing as the interior nodes, they can calculate this time and thus have all the required synchronization information.

3.6.4 The Boundary Node Assumption

The assumption that each boundary node i knows α_i is a strong assumption. Even though the fraction of nodes that are boundary nodes is small for large, multi-hop networks, we believe that the assumption is still very artificial. The main reason we make the assumption is that it allows us to give an elegant presentation of the main concept of this chapter which is to use spatial averaging to average out errors in the network. Throughout this chapter we have used spatial averaging to average out inherent errors present in the nodes. We were able to average out random timing jitter that is present in each node and provide the network with a sequence of zero-crossings that can serve as synchronization events. We then applied this technique to averaging out the errors introduced by time delay. To this end we were partially successful in that the interior nodes can average out these errors assuming the boundary nodes have additional information. But this is of interest since the goal of this chapter is to understand the theory of spatial averaging for synchronization and discover its fundamental advantages and limitations.

3.7 Conclusion

In this chapter, the asymptotic study has allowed us to understand the fundamental capabilities and limitations of spatial averaging. We have shown that in the limit of an asymptotically dense network, there is a cooperative synchronization technique that can construct a sequence of equispaced zero-crossings that occur at integer values of the reference time when there is negligible propagation delay. Since every single node in the network, regardless of the node's distance from the reference clock, can see the same sequence of zero-crossings, we find that the zero-crossings effectively function as a common synchronization signal for the entire network. Therefore, we see that the use of spatial averaging can indeed maintain perfect synchronization in an asymptotically dense network and spatial averaging is ideally suited for networks with negligible propagation delay. However, spatial averaging has limitations due to the asymmetries introduced by propagation delay.

CHAPTER 4

ASYMPTOTIC SYNC TECHNIQUE IN FINITE DENSITY NETWORKS

4.1 Introduction

Even though the results in Chapter 3 are for asymptotically dense networks, they may still approximately hold for networks that have high, but still finite, node densities. In this chapter, we explore how well a synchronization technique based on the asymptotic synchronization method of Chapter 3 can synchronize a network with a finite number of nodes. In Section 4.2 we lay out the system model and in Section 4.3 we describe a synchronization protocol based on the asymptotic synchronization technique of the previous chapter. The implementation of the simulator is described in Section 4.4 and the simulation results are presented in Section 4.5.

4.2 System Setup

As in Chapter 3, each node will periodically transmit a scaled version of the pulse $p(t)$ to achieve and maintain synchronization. Using $p(t)$ from (3.2) with

$$q(t) = \begin{cases} 1 & t \in (-\tau_{nz}, 0) \\ 0 & \text{otherwise} \end{cases}$$

we get that $p(t)$ takes on the shape

$$p(t) = \begin{cases} 1 & -\tau_{nz} < t < 0 \\ 0 & t = 0, t \leq -\tau_{nz}, t \geq \tau_{nz} \\ -1 & 0 < t < \tau_{nz} \end{cases} \tag{4.1}$$

for some $\tau_{nz} > 0$ and τ_{nz} is expressed in terms of c_1.

Similar to (3.1), the aggregate waveform seen by node j is

$$A_{j,\eta_k}^{c_1}(t) = \sum_{i=1}^{\eta_k} \frac{A_{max} K_{j,i}}{\eta_k} p(t - \tau_o - T_i), \tag{4.2}$$

where $A_{j,\eta_k}^{c_1}(t)$ is the waveform seen at node j written in the time scale of c_1. T_i is the random timing jitter suffered by the ith node. The only difference is the use of η_k instead of N. The reason for η_k is that not all N nodes in the network will initially be transmitting when the synchronization process starts. At first, node 1 will be the only node broadcasting synchronization pulses. Following that, the nodes in the broadcast domain of node 1 will join in transmitting pulses. Eventually, all N nodes in the network will be broadcasting and we will have the situation in (3.1). However, during the transient startup processes, the number of transmitting nodes changes. Details of the protocol are described in Section 4.3.

To model the quality of the reception of $A_{j,\eta_k}^{c_1}(t)$ by node j, we model the reception of a signal by defining a threshold γ. γ is the minimum received maximum signal magnitude required for nodes to perfectly resolve the pulse arrival time. If the maximum received signal magnitude is less than γ then the node does not make any observations and ignores the received signal waveform. We assume that $\gamma << A_{max}$, where A_{max} is the maximum transmit magnitude of a node.

As a reminder, the expression in (4.2) is the synchronization waveform for one synchronization phase. The goal of the synchronization protocol will be to construct a synchronization pulse train similar to (3.3) that has the form

$$\bar{A}_{j,\eta_k}^{c_1}(t) = \sum_{q=1}^{\infty} \sum_{i=1}^{\eta_{k,q}} \frac{A_{max} K_{j,i}}{\eta_{k,q}} p(t - \tau_q - T_{i,q}),$$

where $\eta_{k,q}$ is the number of contributing nodes at the qth synchronization pulse, τ_q is the integer value of t that is the ideal transmission time of the qth synchronization

pulse, and $T_{i,q}$ is the jitter suffered by the qth synchronization pulse of ith node.

4.3 Synchronization Protocol

We consider a network of N nodes, uniformly distributed over a fixed area. We extend the synchronization process for asymptotically dense networks from Section 3.5 for a network with a finite number of nodes and node 1 at the center of the network.

Synchronization will be achieved in the following manner. Node 1 will start transmitting pulses at some integer value of the reference time and continue to transmit pulses at integer values of t. After the initial m pulses, the set of nodes in the broadcast domain of 1, not including node 1, will make an estimate using the time synchronization estimator $V_{n,i}^{c_i}$ (3.9) of the location of the $(m+1)$th pulse and transmit at that time. The set of nodes in the broadcast domain of node 1 will be called R_1. The nodes in R_1 will then use its most recent m observations to estimate the time of pulse $m+2$. The R_1 nodes will continue in this manner. The nodes that can hear the aggregate transmissions from R_1 and node 1, the R_2 nodes, will begin their own predictions and transmissions after observing m pulses. This propagation will then continue until all nodes in the network hear signals. Fig 4.1 illustrates this propagation.

When a new set of nodes first begin to transmit, say R_j, they send out a packet of information following their first pulse that has the integer value of t at which the pulse occurred. If we call this value v, then when nodes R_{j+1} start transmitting, they will know that the pulse they just sent occurred at about $t = v + m$. The R_{j+1} nodes will send this value out in a packet after the first pulse. This will let all nodes not only see a common sequence of zero-crossings, but also tell the nodes

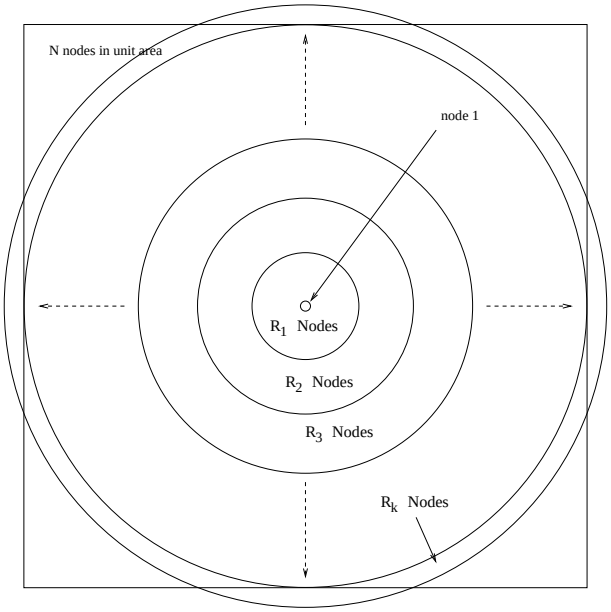

Figure 4.1: The above figure illustrates the propagation of the synchronization pulses starting from node 1 at the center of the network with N nodes uniformly distributed over the area. The R_1 nodes hear the pulses from node 1 and the R_2 nodes hear the aggregate signal from node 1 and the nodes in R_1. This propagation continues beyond the R_k nodes until all nodes in the finite area can hear synchronization pulses.

at what time each zero-crossing occurred at. The one other piece of information contained in the packet is a count, q, of how many hops out from node 1 the synchronization has gone. For example, node 1 will send $q = 1$. After m pulses, node 1 will internally increment its value, $q + 1 = 2$. It will then continue to increment q every m pulses. The R_1 nodes will increment their own internal value of q every m pulses they see. When the R_1 nodes start sending their pulses, they will have $q = 2$ and this is the value they send with the data packet they broadcast to the R_2 nodes. The importance of q is so that nodes can approximately scale their signal amplitudes to conserve power by keeping the aggregate amplitude controlled. All transmitting nodes will scale their transmit amplitude by $1/\eta_{q-1}$.

η_{q-1} is the number of nodes transmitting when the number of hops out from node 1 is q. In this chapter we assume η_q is known for all q, but in reality it will be estimated using the transmission range of the set of nodes $\bigcup_{i=0}^{q-1} R_i$ and the density of the network. This scaling by η_q occurs only after the initial m transmissions by node 1. Node 1 initiates the synchronization mechanism by making m pulse transmissions at magnitude A_{max}. After that, node 1 will scale its transmissions along with all other transmitting nodes. Since we consider finite area networks, the nodes would know at most how many hops are required to cover the entire network. We say that all nodes are transmitting when $q = q_T$, for some q_T known by all nodes. Note that the scaling by η_{q-1} is part of the transient start-up process that starts the synchronization processes. After this initial setup processes, the amplitude that the nodes need to transmit will be scaled by $1/N$. Since the scaling factor of $1/N$ goes to zero as $N \to \infty$, we get that the more nodes we have in the network (for a fixed area), each node will need to consume less power.

It is important to note that the packet distribution overhead is only for the initial startup phase. After all nodes in the network have been been synchronized, the network only needs to emit synchronization pulses to maintain synchronization. In terms of energy consumption, the fact that only synchronization pulses are required means very little energy will be consumed to maintain synchronization. A pseudo-code description of the synchronization protocol is given in Table 4.1. Note that the algorithm does not apply to node 1. Node 1 will initiate the synchronization phase with m transmissions at magnitude A_{max} and then scale its transmissions in the same way other nodes do. However, node 1 never adjusts its transmissions and simply transmits a pulse at the appropriately scaled magnitude at every integer value of t.

Table 4.1: The synchronization algorithm for each node i, $i \neq 1$

```
TimeSync (observation length m > 1)


observe pulse arrival time;
if (first observed pulse)
  { receive packet and set v, q values; };
while (pulse arrived)
  { if (m or more arrival times in memory)
      { keep only m most recent & discard
          all other arrival times;
        use last m arrivals to estimate
          next pulse transmission time t_p;
        if (q < q_T)
          { transmit pulse (A_max p(t))/η_{q-1} at time t_p; };
        else {transmit pulse (A_max p(t))/N at time t_p; };
        if (first pulse transmitted)
          { transmit packet with v+1, q; };
      };
    observe pulse arrival time;
    if (# pulses received is multiple of m)
      { set q = q+1; };
    set v = v+1;
  };
```

The mechanism so far is nearly identical to the process outlined in Section 3.5 for asymptotically dense networks. In applying it to finite sized ($N < \infty$) networks, we introduce a small amount of feedback into the system to prevent small errors from accumulating. Node 1 is the only node in the network that can observe the aggregate waveform and have access to the reference clock. We define a tolerance factor, ϱ, such that if node 1's observed zero-crossing is more than ϱ from the ideal zero-crossing, then it informs all nodes in the network to adjust their estimate. ϱ is defined as

$$\varrho = \frac{\text{maximum allowed distance between ideal and observed zero-crossing}}{\text{time between synchronization pulses}},$$

where all times are defined in terms of c_1. It is clear that ϱ is defined in the design of the system so each node knows its value.

When node 1 notices that ϱ has been exceeded, it sends a one bit feedback to all nodes. That bit will tell nodes whether the observed zero-crossing occurred before or after the ideal zero-crossing location. If the observed occurred before the ideal, then each node will delay all m of its observations by an adjustment factor. If the observed zero-crossing occurred after the ideal, each node will shift all of its observations earlier in time by an adjustment factor. This means that if, for example, the observed zero-crossing occurred early, then by having each node delay its set of m observations, the next estimate made by each node will occur later. Since all nodes are making a later estimate, the next aggregate waveform zero-crossing should be delayed as well, bringing it closer to the ideal zero-crossing time. For each node i the adjustment factor is calculated as

node i adjustment factor

$= \varrho \times |\text{difference between most recent two observation times}|.$

Note that these calculations are all done by node i in terms of its own time scale.

It is important to stress that this added feedback does not in any way affect the asymptotic properties of the synchronization mechanism. In an asymptotically dense network, the extra feedback and correction mechanism will simply not be needed. It is added only to make the asymptotic synchronization mechanism robust for networks of finite size.

4.4 Simulator Implementation

As mentioned, we study the synchronization problem in the asymptotically dense regime since it closely approximates the behavior of networks with large, but still finite, densities. As a result, an obvious question is how well the limiting regime actually approximates finite density networks. In an effort to answer this question, we implement a simulator to study the performance of our synchronization mechanism on finite density networks.

The time synchronization simulator is implemented in MATLAB and initializes by running the function `NodeGen` to generate the set of N nodes uniformly distributed over a circular area. N is calculated as $N = \text{density} \times \text{area}$. `NodeGen` generates an $N \times 5$ matrix with any node i's x and y coordinate, its distance from node 1 (assumed to be at the center of the circle), and the parameters α_i and $\bar{\Delta}_i$ for its clock. α_i and $\bar{\Delta}_i$ are generated as samples from a Gaussian distribution. In the simulations performed we always use the following:

$$\text{area} = 30 \qquad \alpha_i \sim \mathcal{N}(0, 0.01) \qquad \bar{\Delta}_i \sim \mathcal{N}(0, 0.1).$$

After `NodeGen`, the simulator executes as illustrated in Fig. 4.2, and each function is described below.

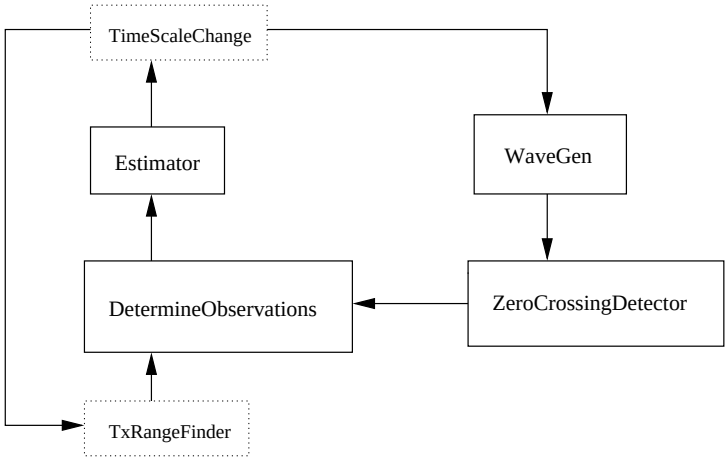

Figure 4.2: Block diagram illustrating the structure of the simulator. Data flow follows the arrows and the most important functions are the ones surrounded by a solid box. `WaveGen` generates the aggregate waveform seen by all nodes. `ZeroCrossingDetector` finds the zero-crossing of the aggregate waveform and `DetermineObservations` finds each node's observation of the zero-crossing. `Estimator` takes each node's observations and makes the estimate of where that node should next make a pulse transmission. The information is then passed back to `WaveGen`. The functions `TimeScaleChange` and `TxRangeFinder` in the dotted boxes are key supporting functions.

`Estimator`: This function takes from `DetermineObservations` the $N \times m$ matrix which contains each node i's observations in terms of c_i (recall that m is the number of observations each node makes before it starts estimating the next zero-crossing time). If node i has m observations then the function makes the estimate of the next integer value of t. `Estimator` outputs an $N \times 1$ vector that contains the zero-crossing estimates for the nodes that are making pulse transmissions and a place holder value for nodes that are not.

`TimeScaleChange`: Since node i's estimate in the output of `Estimator` is in the time scale of c_i, `TimeScaleChange` changes the value into the time scale of c_1. This is done, for the ith node using $t_{out} = (t_{in} - \Psi_i)/\alpha_i + \bar{\Delta}_i$, where t_{out} is in

the $N \times 1$ output of `TimeScaleChange` and t_{in} comes from `Estimator`. Ψ_i is the Gaussian random jitter of c_i and we use $\text{Var}(\Psi_i) = 0.01$ for all i.

`WaveGen`: This function generates the waveform using A_{max} and the pulse $p(t)$ described in (4.1). To simplify waveform generation, we assume that there is no amplitude loss in the waveform. Amplitude loss is accounted for in calculating the transmission range however. Node transmission times come from the output vector of `TimeScaleChange`. For all simulations we assume:

$$\tau_{nz} = 0.2 \qquad A_{max} = 8 \qquad K_{j,i} = 1.$$

`ZeroCrossingDetector`: `ZeroCrossingDetector` takes the waveform generated by `WaveGen` and finds the zero-crossing.

`DetermineObservations`: This function takes the zero-crossing value, which is in the time scale of c_1, and translates it into the time of c_i for each node i that is making observations. This conversion is done using (2.1). `DetermineObservations` knows which nodes are making observations since it obtains the current transmission range from `TxRangeFinder`. This function also implements the feedback/adjustment mechanism. For simulations we set the tolerance factor to be $\varrho = 0.05$.

`TxRangeFinder`: This function finds the transmission range of the nodes that are currently transmitting. It assumes $K(d)$ to be

$$K(d) = \begin{cases} 1 & d < \epsilon \\ \sqrt{\frac{\epsilon^\beta}{d^\beta}} & d \geq \epsilon \end{cases}$$

It approximates the range of the aggregate waveform generated by nodes in a circular area by partitioning the circle into vertical sections of width 0.01. Each section is then assumed to generate an equivalent waveform originating on the

x-axis. These equivalent waveforms are then summed to determine the distance from node 1 at which the aggregate amplitude first drops below γ. For simulations we use

$$\beta = 2 \qquad \epsilon = 0.1 \qquad \gamma = 0.2.$$

It uses the output of `TimeScaleChange` to determine the transmitting nodes and thus the circular area of transmitting nodes.

The parameters m and density are not specified above because they are varied throughout the simulations.

4.5 Simulation Results

Before presenting the results of the simulations, we first describe how we measure the performance of the synchronization mechanism. Recall that ideally we would want all nodes to transmit a synchronization pulse at the exact same time. This means that in the ideal situation, when we translate each node i's estimate of the next zero-crossing location into the time scale of node 1 it should be the next integer value of t. The output of `TimeScaleChange` is used to measure the performance of the synchronization mechanism since it gives each node i's estimate in the time scale of c_1. Thus, ideally all values would be the next integer value of t. In reality, this is not the case and we use a measure, which we call the *average squared distance* or ASD, to quantify the average distance of the nodes' estimates from the ideal integer time of c_1. The ASD is calculated as follows:

$$ASD = \frac{1}{\bar{N}} \sum_{i=1}^{\bar{N}} (\hat{t}_i^{c_1} - t_0)^2,$$

where $\bar{N} \leq N$ is the number of nodes currently making estimates, t_0 is the next integer value of t, and $\hat{t}_i^{c_1}$ is the ith estimating node's estimate of t_0 in the time

scale of c_1. Note that the ASD measures the performance of the time synchronization estimator of all nodes in network since the goal of the time synchronization estimator is to predict the next integer value of t.

The first simulation result that we present in Fig. 4.3 serves as motivation for

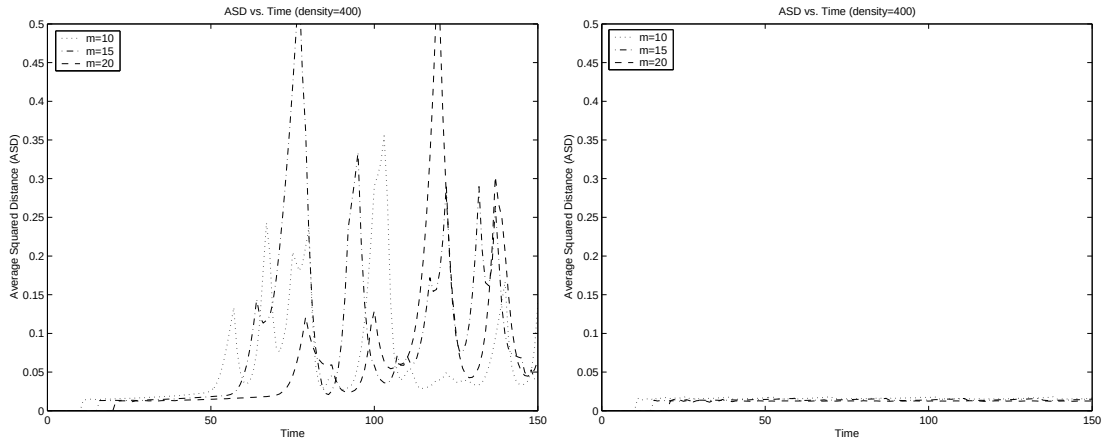

Figure 4.3: Left: A plot of ASD versus time for the synchronization mechanism without feedback. The results were averaged over 10 runs. We see that synchronization is held for a period of time but not indefinitely. Right: A plot of ASD versus time for the synchronization mechanism with feedback. We note that ASD is bounded and synchronization can be maintained indefinitely.

the modified synchronization mechanism that includes feedback. We see in the first figure of Fig. 4.3 that for $m = 10, 15, 20$, synchronization is maintain over a period of time. If fact, for $m = 20$, synchronization is maintained for over seventy time units. On average, as seen in the first figure of Fig. 4.3, the larger the m value the longer synchronization can be maintained. However, in all cases synchronization is eventually lost. This is due to the fact that small errors in the aggregate waveform zero-crossing location accumulate. For example, if an observed zero-crossing arrives late, then the next aggregate waveform zero-crossing may arrive late as well since all nodes are making an estimate using the delayed zero-crossing. Thus, these

errors accumulate and eventually the aggregate waveform zero-crossing may drift significantly away from the integer values of t. Note that the peaks and troughs in the ASD plot are a result of the simulator. The simulator looks for the zero-crossing in an interval around each integer value of t. As a result, when the zero-crossing drifts so much that the nodes can longer observe the zero-crossing in the preset interval, then huge errors will result. If no zero-crossing is observed, the nodes are set to simply increment their last observed pulse arrival time by one and use that as a new observation. Thus, the zero-crossing may reappear and drive down the ASD. Nonetheless, as soon as the ASD starts spiking, the zero-crossing has effectively drifted significantly away from the integer values of the reference time.

By introducing feedback we can correct this drifting zero-crossing. An illustration of ASD versus time for the mechanism with feedback is presented in the second figure of Fig. 4.3. There we run the simulation once and notice that in all cases the ASD is bounded and synchronization is maintained indefinitely. In Fig. 4.4 we zoom in on the second figure of Fig. 4.3 and notice the "sawtooth" waveform for $m = 10$ and $m = 15$. Each "tooth" coincides to one time that the feedback triggered by node 1 adjusted each node's observations. In fact, the simulator tells us that for $m = 10$ there were six corrections, $m = 15$ had four, and $m = 20$ so far did not require any corrections to the node observations.

Another key property of the synchronization mechanism with feedback is that it performs well for a wide range of network sizes. In Fig. 4.5 we plot the ASD versus time for network sizes varying from $N = 300$ (density=10, area=30) to $N = 18300$ (density=610, area=30). In steady-state all nodes are transmitting and we notice that the ASD curve for the 300 node network is at most 0.0005 greater than the ASD curve for the 18300 node network. This means that on

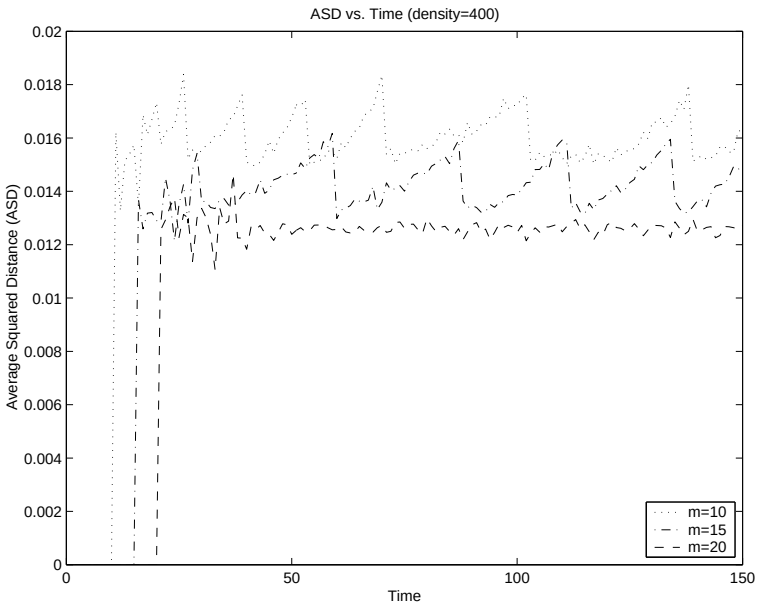

Figure 4.4: A plot of ASD versus time for the synchronization mechanism with feedback. We notice a "sawtooth" waveform for $m = 10, 15$ and each "tooth" occurs at a time where a correction was made.

average the ASD varies only by at most 0.0005 for network sizes in this range and thus the mechanism is well suited for network sizes as small as a few hundred nodes. Of course, as expected, the mechanism must make more active corrections based on feedback from the network. In fact, we find that the average number of corrections made for the 150 time units of the simulation was 18.2 for $N = 300$ and 2.9 for $N = 18300$. As a result, even though the mechanism performs well for networks of only a few hundred nodes, it does require more active adjustments on the part of the mechanism. Such a result is in line with our comment at the end of Section 4.3 since in the limit as $N \to \infty$ the feedback and correction mechanism will not be needed.

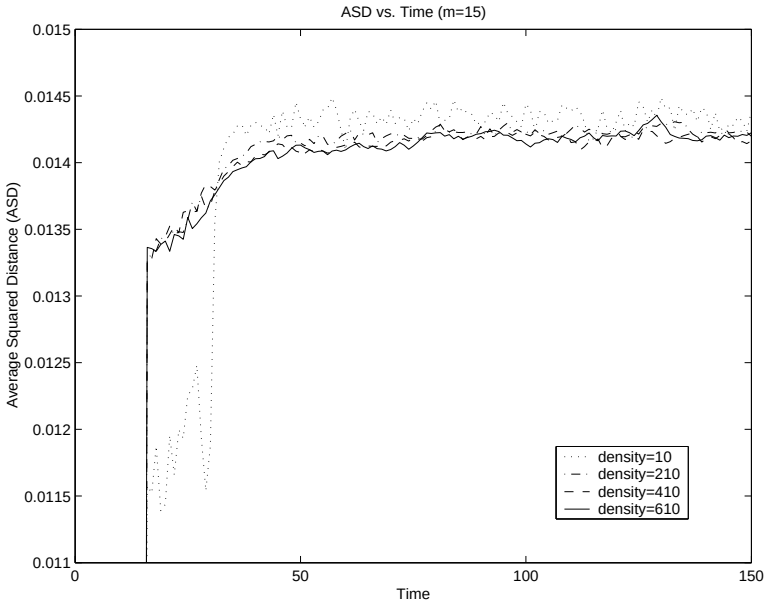

Figure 4.5: A plot of ASD versus time for the synchronization mechanism with feedback for different network sizes. Each plot was averaged over 500 runs. In steady-state we see that the mechanism performs well for a wide range of network sizes since the difference in ASD for a network of $N = 300$ nodes and a network of 18300 nodes is at most 0.0005.

4.6 Conclusion

Through the use of simulations, in this chapter we have shown that a synchronization technique based solely on the asymptotic technique of Chapter 3 can not maintain synchronization indefinitely. However, a minor modification of the technique that allows the reference node to communicate a single bit of information to the rest of the nodes in the network allows the network to remain synchronized.

The problem with applying an asymptotic technique to networks with a finite number of nodes is that we do not have any sort of analytical characterization of how well the finite sized network is synchronized. The only sort of characterization we have so far comes from simulation results and the tolerance factor ϱ from the synchronization protocol. Also, most existing network synchronization techniques

synchronize the network by having each node estimate its clock skew and clock offset relative to a reference clock. Thus, in the next chapter we address these issues by characterizing the improvement in skew and offset estimates when spatial averaging is used.

CHAPTER 5

COOPERATION IN FINITE DENSITY NETWORKS

5.1 Introduction

In Chapter 3 we analytically studied the asymptotic properties of cooperation using spatial averaging. Since these analytical results do no provide any information about how spatial averaging will perform in finite density networks, in Chapter 4 we provided some simulation results to understand how the asymptotic synchronization technique performs in networks of finite density. However, no attempt is made to analytically characterize synchronization performance.

In this chapter, we develop a cooperative synchronization protocol to estimate clock skew and clock offset in networks of finite density and we seek to analytically characterize how synchronization performance improves as a function of the amount of cooperation. We set up the system model in Section 5.2 and describe the synchronization protocol in Section 5.3. To analyze the performance of the protocol, we first start by studying its performance in a Type I basic cooperative network. Type I network analysis is provided in Section 5.4 and simulations are presented in Section 5.5. Using our understanding from a Type I network, we proceed to study Type II general networks. Type II analysis is in Section 5.6 and simulations are in Section 5.7. Lastly, we compare the performance of cooperative and non-cooperative synchronization techniques in Section 5.8.

5.2 System Setup

To develop the system model, we need to consider the new challenges we face in studying finite density networks. We first realize that for finite density networks,

the distribution of the random observation (i.e. zero-crossing) needs to be characterized. In the infinite density situation the observation was deterministic, but it is now random in the finite density case. Second, we actually need to characterize the distribution of functions of the observations since the observations are used to make estimates. Third, we see that the zero-crossing is extremely difficult to characterize. As a result, for our finite density network study, we will develop a simpler observation and spatial averaging model.

For pulse transmissions, each node i in the network can transmit short pulses $p(t)$ for time synchronization. These are short duration pulses, i.e. ultra wideband pulses, and for our purposes we consider them to be delta functions $\delta(t)$. The particular choice of $p(t)$ is not important. For the purposes of studying cooperative time synchronization, we assume a node receiving the pulse can uniquely determine a pulse arrival time, pulses sent from different nodes do not overlap, and a node seeing multiple pulses can identify the different pulse arrival times.

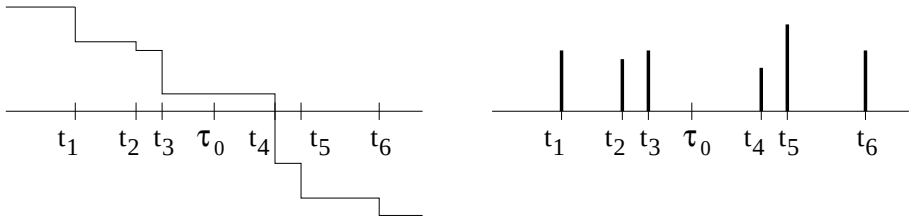

Figure 5.1: The zero-crossing observation (left) and the sample mean of pulse clusters (right) are different implementations of the spatial averaging concept.

The observation will now be the sample mean of clusters of pulses instead of a zero-crossing (Fig. 5.1). The important thing to realize, however, is that the two techniques are simply different approaches to the same spatial averaging concept. In both cases, the key property of the pulse transmission time is that it is symmetric about the ideal transmission time τ_0. With an increasing number

of pulse transmissions, we know that the zero-crossing converges to τ_0. Similarly, with more pulse transmissions, we know that the sample mean converges to the true mean, which is τ_0. Thus, we see that the underlying idea of spatial averaging is exactly the same for the two different aggregate signals. However, using the sample mean instead of the zero-crossing provides us with a more analytically manageable setup.

The remainder of the system setup is similar to what we considered before. We assume a network with a large number of nodes uniformly distributed over a wide area. No assumptions are made on the type of nodes. This can be a homogeneous network of identical nodes or a network composed of a variety of different types of sensor nodes. Since the network is deployed over a large area, this is a multi-hop network and a large number of hops are required to send information across the network. The nodes do not have the capability to communicate with devices outside the network, i.e. nodes do not have GPS receivers.

The important assumption that we do maintain, however, is that the network has high node density. That is, each sensor node can receive signals from a large number of neighbors. It is important to realize that this assumption does not significantly limit the applicability of cooperative time synchronization. For any given network, we can always artificially construct a dense network by deploying a large number of nodes whose sole purpose is to aid in time synchronization.

We again assume that node 1 contains the reference clock and every node in the network is to be synchronized to this clock. We use the general clock model described in Section 2.2.

Lastly, we simplify the signal propagation model by assuming that each node has a transmission range of R. This comes from using $K(d)$ in the pathloss only

model (Section 2.3) and assuming that a received signal must have an amplitude greater than or equal to some threshold γ. Since all nodes transmit the same pulse, transmission range R is therefore the distance at which the signal amplitude drops below the required threshold. This means that a node j must be within a distance R from a transmitting node i in order to hear pulses from node i. Note that the assumption of a circular transmission region is made only to simplify the illustration of spatial averaging. The synchronization protocol proposed in Section 5.3 does not require this assumption and most of the results in this work will hold under more realistic conditions [43, 44]. Since we are dealing with sensor nodes who have short transmission distances, we further assume that propagation delay is negligible. We make this assumption since from [10] we know that propagation delay is less than $1\mu s$ for distances up to 300 meters.

5.3 Synchronization Protocol

To start synchronization, the reference node, node 1, will send a sequence of m pulses that are d seconds apart. Since we assume the nodes have impulse radio transmitters, each pulse is extremely narrow in time. The values of d and m are parameters of the protocol that are established before deploying the network so the values are known by all nodes in the network. Therefore, in the time scale of node 1 the pulses are transmitted at times $\tau_0, \tau_0 + d, \ldots, \tau_0 + d(m-1)$, where τ_0 is the time at which the synchronization process started. Let node 1 be the only element of the R_0 nodes.

The nodes that are in the broadcast domain of node 1 will hear this sequence of m pulses. We call these nodes the R_1 nodes and each node $i \in R_1$, $i \geq 1$, will be denoted by node $1i$. The vector of pulse arrival times observed by node $1i$

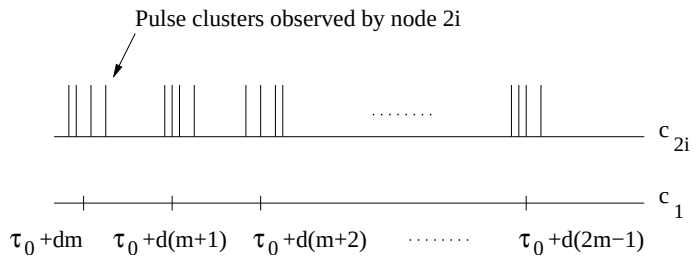

Figure 5.2: A node $2i$ in R_2 has clock c_{2i}. This node will see clusters of pulse arrivals that are transmitted from a group of nodes in the set R_1. These clusters arrive at node $2i$ around times $\tau_0 + dm, \tau_0 + d(m+1), \ldots, \tau_0 + d(2m-1)$ in the time scale of node 1, c_1.

will be denoted $\mathbf{Y}_{1i}$. Each node $1i$ will be able to estimate its clock skew since it knows that node 1 transmitted these pulses d seconds apart. Each node $1i$ will also predict, in its own time scale, when times $\tau_0+dm, \tau_0+d(m+1), \ldots, \tau_0+d(2m-1)$ will occur in the time scale of node 1 and transmit m pulses, one at each predicted time. This means that each node $1i$ will transmit a pulse approximately at times $\tau_0+dm, \tau_0+d(m+1), \ldots, \tau_0+d(2m-1)$ in the time scale of node 1. When the R_1 nodes each transmit their sequence of m pulses, the nodes that can hear a subset of the R_1 nodes, the R_2 nodes, will observe clusters of pulses around the times $\tau_0 + dm, \tau_0 + d(m+1), \ldots, \tau_0 + d(2m-1)$ since each node $2i$ can hear many R_1 nodes (Fig. 5.2). In fact, we make sure each node $2i$ can hear a cluster by requiring the node to observe at least $\bar{N}$ pulses in each cluster. If a node $2i$ sees less than $\bar{N}$ pulses in a cluster, then it will not make observations. Each node $2i$, a node $i \in R_2$, will note the arrival time of each pulse in the kth cluster, $k = 1, \ldots, m$, and take the sample mean of these times to be its kth observation. Node $2i$'s vector of observations will be denoted as $\mathbf{Y}_{2i}$. Using these m observations, any node $2i$ will be able to estimate its clock skew since it knows that these observations should

be occurring d seconds apart. As well, it will be able to predict in its local time scale when times $\tau_0 + d(2m), \tau_0 + d(2m+1), \ldots, \tau_0 + d(3m-1)$ will occur in the time scale of the reference time. Node $2i$ will then transmit a pulse at each of those predicted times. This processes will repeat until all nodes in the network have an estimate of their clock skew. Notice that the R_1 nodes are not required to observe $\bar{N}$ pulses in each cluster since they will always only receive a sequence of m pulses from node 1. Node 1 can simply broadcast a special packet to its surrounding nodes to identify the R_1 nodes. An illustration of the process can be found in Fig. 5.3 and note that nodes will remain silent for the remainder of the synchronization process after transmitting their m pulses.

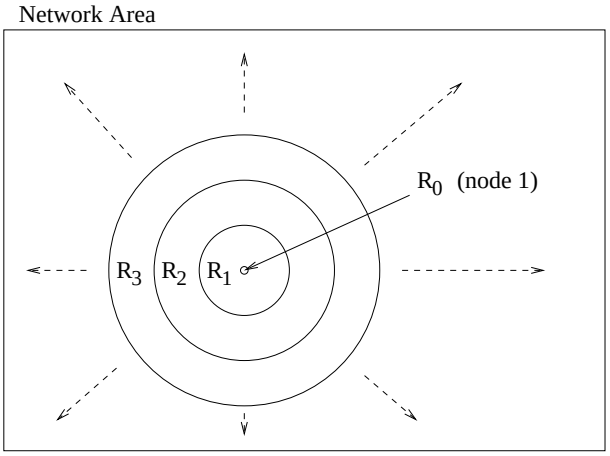

Figure 5.3: For increasing i, the R_i nodes are progressively farther and farther away from reference node 1, the R_0 node. Each node in the set R_i receives its timing information from a group of nodes in set R_{i-1}.

The cooperation occurs when a node ki in R_k, $k > 1$, can take a sample mean of a cluster of pulse arrivals. Recall that each cluster will have at least $\bar{N}$ pulses. Since each pulse transmission will be corrupted by random estimation errors from a node in the set R_{k-1}, by taking the average of a number of pulse

arrivals, some of the random error can be averaged out. This in turn will allow for better synchronization performance.

So far, each node in the network has the ability to estimate its clock skew relative to the reference clock. To obtain the clock offset, the R_k nodes will broadcast a packet of information to the R_{k+1} nodes, $k \geq 0$. This packet will contain the value of τ_0 and a number q denoting the number of hops out from node 1. For example, node 1 will transmit the value of τ_0 and $q = 0$ to the R_1 nodes. The R_1 nodes will then send τ_0 and $q = 1$ to the R_2 nodes. In general, the R_k nodes will send τ_0 and $q = k$ to the R_{k+1} nodes. Any node ki will then know that its first observation approximately occurred at time $\tau_0 + dmq$ in the time scale of the reference time, where the value of q is the one received from set R_{k-1}.

We now describe how any node ki can estimate its clock skew, clock offset, and its m pulse transmission times. From (2.1), we know that there is a linear relationship between the reference clock c_1 and the clock of node ki, c_{ki}. Node ki will have a set of m observations denoted by the $m \times 1$ vector $\mathbf{Y}_{ki}$, where the elements of the vector are ordered from the earliest observation time to the latest observation time. Node ki will estimate its clock skew as

$$\hat{\alpha}_{ki} = \bar{\mathbf{C}}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}_{ki} \tag{5.1}$$

and clock offset as

$$\hat{\Delta}_{ki} = \tilde{\mathbf{C}}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}_{ki} - (\tau_0 + dm(k-1)), \tag{5.2}$$

where $\bar{\mathbf{C}} = [0 \quad 1]$, $\tilde{\mathbf{C}} = [1 \quad 0]$ and

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & d & 2d & \dots & (m-1)d \end{bmatrix}^T \tag{5.3}$$

Table 5.1: The synchronization protocol for each node ki, $k > 1$

Cooperative Time Sync

wait for pulse arrivals, at least $\bar{N}$
  per cluster;
while (number of arrival clusters < m) {
  record arrival time of all pulses;
  listen for packet with τ_0 and q values;
};
for each (pulse arrival cluster j) {
  $\mathbf{Y}_{ki}[j] =$ sample mean of cluster;
};
skew $\hat{\alpha}_{ki} = \bar{\mathbf{C}}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}_{ki}$;
offset $\hat{\Delta}_{ki} = \tilde{\mathbf{C}}(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}_{ki} - (\tau_0 + dmq)$
for (l from 0 to $m-1$) {
  transmission time $X^{c_{ki}}_{l+1,ki}(\mathbf{Y}_{ki}) = \mathbf{C}_l(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}_{ki}$;
  transmit pulse at $X^{c_{ki}}_{l+1,ki}(\mathbf{Y}_{ki})$;
};
while (transmitting pulses) {
  send a packet with values τ_0 and $q+1$;
};

Note that in the calculation of the clock offset $\hat{\Delta}_{ki}$, the term $\tau_0 + dm(k-1)$ is the time in the time scale of c_1 that node ki should receive its first pulse. Node ki has

used the τ_0 and $q = k - 1$ parameters sent to it from the R_{k-1} nodes. Node ki will also estimate its own m pulse transmission times using the pulse-connection function

$$X_{l+1,ki}^{c_{ki}}(\mathbf{Y}_{ki}) = \mathbf{C}_l(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}_{ki}, \tag{5.4}$$

where $\mathbf{C}_l = [1 \quad d(m+l)]$, for $l = 0, 1, \ldots, m-1$. $X_{l+1,ki}^{c_{ki}}(\mathbf{Y}_{ki})$ is the transmission time of node ki's $(l+1)$th pulse. A pseudo-code description is given in Table 5.1.

Note that the pulse-connection function in (5.4) is essentially the same as the one used in the infinite density case with no propagation delay (3.9). They both make unbiased estimates of the ideal pulse transmission time. The only real difference is that a node using (3.9) estimates only the next pulse transmission time while a node using (5.4) estimates the next m pulse transmission times.

5.4 Type I Node Deployment — Analysis

The analysis of the protocol described in Section 5.3 is difficult since, in general, different nodes see different observations. As shown in Fig. 5.4, node $(k+1)i$ and node $(k+1)j$ see different nodes from R_k and they see different numbers of nodes. This means that their observations are different, but may potentially be correlated. As a result, the complexity in characterizing the observations made by different nodes in the network make analyzing the protocol difficult. We, therefore, breakdown the analysis by analyzing the protocol in two types of networks: Type I basic cooperative networks and Type II general networks. This section and Section 5.5 focus on Type I networks while Section 5.6 and Section 5.7 focus on Type II networks.

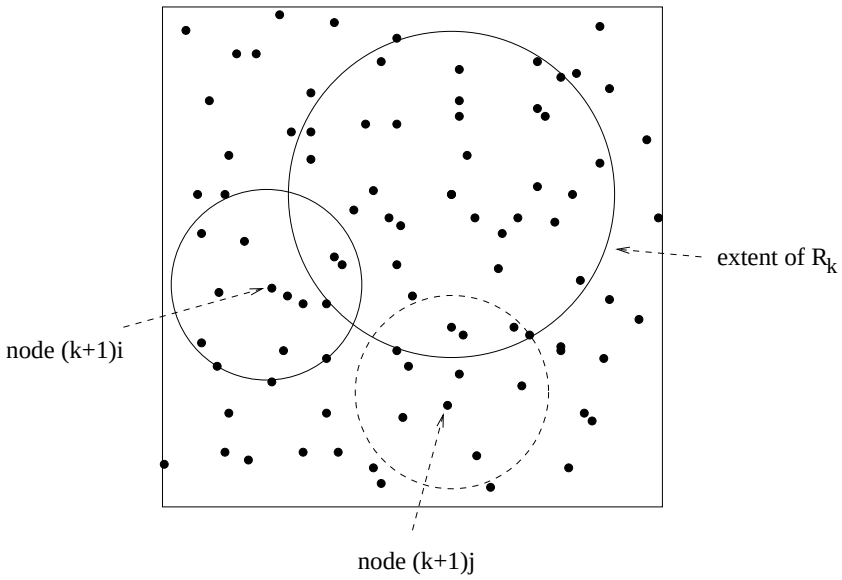

Figure 5.4: Node $(k+1)i$ and node $(k+1)j$ have different observations that are possibly correlated.

5.4.1 Network Setup

The most basic and fundamental deployment of nodes that effectively employs cooperative time synchronization is the case where *all* $\bar{N}$ nodes at any given hop contribute to the signals observed at the next hop. This Type I deployment is illustrated in the top of Fig. 5.5 where each set of nodes R_i, $i \geq 1$, have $\bar{N}$ nodes. We see that every single node in R_i is in the broadcast domain of every node in R_{i-1}. Note that if we let $\bar{N} = 1$, then the network becomes a non-cooperative network where timing information is communicated from one node to the next (bottom of Fig. 5.5). Thus, the Type I network is a generalization of the simple non-cooperative situation.

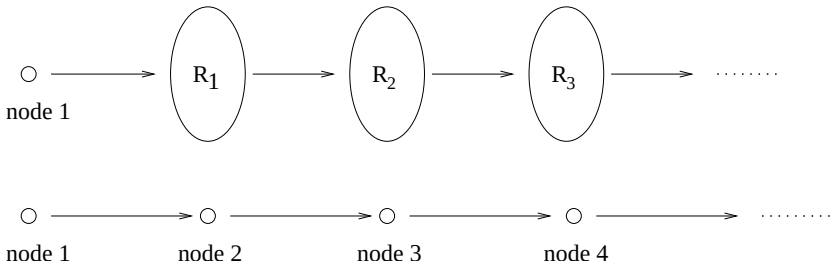

Figure 5.5: Top: A Type I network deployment. Node 1 (R_0 node) initiates synchronization and the $\bar{N}$ nodes in set R_i, $i \geq 1$, convey timing information to set R_{i+1}. Bottom: A non-cooperative network.

5.4.2 Analysis

Due to the scalability problem, we would expect synchronization error to grow as timing information from node 1 (the R_0 node) is communicated to a node in the R_k set of nodes, $k > 0$. Therefore, it is of particular interest to quantify how the mean squared error of the skew and offset estimates change as a function of the hop number k. Looking at the structure of the basic cooperative network in the top of Fig. 5.5, we notice that the skew and offset estimates at a node ki must be dependent only on the estimates made by the nodes in R_{k-1}. Therefore, to understand the synchronization error growth over multiple hops, we need the distribution of the estimates made by the nodes in R_k as a function of the distribution of the estimates made by the nodes in R_{k-1}. Theorem 3, below, provides us with this characterization.

In the statement of the theorem we use e_l to be the column vector of all zeros

except for a one in the lth position and

$$\bar{\mathbf{H}} = \begin{bmatrix} \mathbf{H} & 0 & \dots & 0 \\ 0 & \mathbf{H} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{H} \end{bmatrix},$$

where $\mathbf{H}$ is from (5.3). Also, α_{ki} and $\bar{\Delta}_{ki}$ are the clock model parameters from (2.1) for node ki.

Theorem 3 *Assume a Type I basic cooperative network and the synchronization protocol from Section 5.3.*

1. *Given the distribution of the $2\bar{N} \times 1$ vector of estimates made by the R_{k-1} nodes,*

$$\hat{\bar{\theta}}_{k-1} \sim \mathcal{N}(\bar{\mu}_{k-1}, \bar{\Sigma}_{k-1}),$$

the distribution of the $2\bar{N} \times 1$ vector of estimates made by the R_k nodes,

$$\hat{\bar{\theta}}_k \sim \mathcal{N}(\bar{\mu}_k, \bar{\Sigma}_k),$$

is found as follows: $\hat{\bar{\theta}}_k$ *has mean vector*

$$\begin{aligned}
\bar{\mu}_k &= E(\hat{\bar{\theta}}_k) \\
&= E(\mathbf{A}_k \hat{\bar{\theta}}_{k-1} + \mathbf{B}_k) \\
&= \mathbf{A}_k \bar{\mu}_{k-1} + \mathbf{B}_k \\
&= \begin{bmatrix} \alpha_{k1}(\tau_0 + (k-1)md - \bar{\Delta}_{k1}) \\ \alpha_{k1} \\ \alpha_{k2}(\tau_0 + (k-1)md - \bar{\Delta}_{k2}) \\ \alpha_{k2} \\ \vdots \\ \alpha_{k\bar{N}}(\tau_0 + (k-1)md - \bar{\Delta}_{k\bar{N}}) \\ \alpha_{k\bar{N}} \end{bmatrix}
\end{aligned} \tag{5.5}$$

and covariance matrix

$$\bar{\Sigma}_k = Cov(\hat{\bar{\theta}}_k) = \Sigma_{m_k} + \mathbf{A}_k \bar{\Sigma}_{k-1} \mathbf{A}_k^T \tag{5.6}$$

for

$$\Sigma_{m_k} = (\bar{\mathbf{H}}^T \bar{\mathbf{H}})^{-1} \bar{\mathbf{H}}^T \Sigma_{\bar{\mathbf{W}}_k} ((\bar{\mathbf{H}}^T \bar{\mathbf{H}})^{-1} \bar{\mathbf{H}}^T)^T$$

$$\Sigma_{\bar{\mathbf{W}}_k} = \mathbf{Q}_k \Sigma_{\tilde{\Psi}_{k-1}} \mathbf{Q}_k^T + \sigma^2 \mathbf{I}_{\bar{N}m}$$

$$\mathbf{Q}_k = \begin{bmatrix} \alpha_{k1}\mathbf{I}_m \\ \alpha_{k2}\mathbf{I}_m \\ \vdots \\ \alpha_{k\bar{N}}\mathbf{I}_m \end{bmatrix}$$

$$\Sigma_{\tilde{\Psi}_{k-1}} = \frac{\sigma^2}{\bar{N}^2} \sum_{i=1}^{\bar{N}} \frac{1}{\alpha_{(k-1)i}^2} \mathbf{I}_m$$

where

$$\mathbf{A}_k = \mathbf{D}_k \begin{bmatrix} \frac{1}{\alpha_{(k-1)1}} & \frac{dm}{\alpha_{(k-1)1}} & 0 & 0 & \dots & 0 & 0 \\ 0 & \frac{1}{\alpha_{(k-1)1}} & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \frac{1}{\alpha_{(k-1)2}} & \frac{dm}{\alpha_{(k-1)2}} & \dots & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\alpha_{(k-1)2}} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \frac{1}{\alpha_{(k-1)\bar{N}}} & \frac{dm}{\alpha_{(k-1)\bar{N}}} \\ 0 & 0 & 0 & 0 & \dots & 0 & \frac{1}{\alpha_{(k-1)\bar{N}}} \end{bmatrix}$$

$$\mathbf{D}_k = \frac{1}{\bar{N}} \begin{bmatrix} \alpha_{k1} & 0 & \alpha_{k1} & 0 & \dots & \alpha_{k1} & 0 \\ 0 & \alpha_{k1} & 0 & \alpha_{k1} & \dots & 0 & \alpha_{k1} \\ \alpha_{k2} & 0 & \alpha_{k2} & 0 & \dots & \alpha_{k2} & 0 \\ 0 & \alpha_{k2} & 0 & \alpha_{k2} & \dots & 0 & \alpha_{k2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha_{k\bar{N}} & 0 & \alpha_{k\bar{N}} & 0 & \dots & \alpha_{k\bar{N}} & 0 \\ 0 & \alpha_{k\bar{N}} & 0 & \alpha_{k\bar{N}} & \dots & 0 & \alpha_{k\bar{N}} \end{bmatrix}$$

$$\mathbf{B}_k = \mathbf{D}_k \begin{bmatrix} \bar{\Delta}_{(k-1)1} \\ 0 \\ \bar{\Delta}_{(k-1)2} \\ 0 \\ \vdots \\ \bar{\Delta}_{(k-1)\bar{N}} \\ 0 \end{bmatrix} - \begin{bmatrix} \alpha_{k1}\bar{\Delta}_{k1} \\ 0 \\ \alpha_{k2}\bar{\Delta}_{k2} \\ 0 \\ \vdots \\ \alpha_{k\bar{N}}\bar{\Delta}_{k\bar{N}} \\ 0 \end{bmatrix}.$$

The initial conditions are

$$\bar{\mu}_1 = \begin{bmatrix} \alpha_{11}(\tau_0 - \bar{\Delta}_{11}) \\ \alpha_{11} \\ \alpha_{12}(\tau_0 - \bar{\Delta}_{12}) \\ \alpha_{12} \\ \vdots \\ \alpha_{1\bar{N}}(\tau_0 - \bar{\Delta}_{1\bar{N}}) \\ \alpha_{1\bar{N}} \end{bmatrix}$$

and $\bar{\Sigma}_1 = \sigma^2(\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}$.

2. *The skew estimate and offset estimate for node ki can be found as*

$$\hat{\alpha}_{ki} = e^T_{2(i-1)+2}\hat{\bar{\theta}}_k \tag{5.7}$$

and

$$\hat{\Delta}_{ki} = e^T_{2(i-1)+1}\hat{\bar{\theta}}_k - (\tau_0 + dm(k-1)) \tag{5.8}$$

for $i = 1, 2, \ldots, \bar{N}$ *and* $k \geq 1$. $\triangle$

The proof of Theorem 3 is found in Appendix B.1. Note that since the estimates of skew and offset are unbiased, the variance is the mean squared error. Also, since the distribution of $\hat{\bar{\theta}}_k$ is available, the distribution of $\hat{\alpha}_{ki}$ and $\hat{\Delta}_{ki}$ can be found. In fact, the variance of $\hat{\alpha}_{ki}$ can be found in element $(2(i-1)+2,\, 2(i-1)+2)$ of $\bar{\Sigma}_k$ in (5.6) and the variance of $\hat{\Delta}_{ki}$ can be found in element $(2(i-1)+1,\, 2(i-1)+1)$. The mean of $\hat{\alpha}_{ki}$ is the $(2(i-1)+2)$th element of $\bar{\mu}_k$ in (5.5) and the mean of $\hat{\Delta}_{ki}$ can be found as the $(2(i-1)+1)$th element minus the offset $\tau_0 + dm(k-1)$.

From the statement of Theorem 3, we see that the distribution of the estimates made by the R_k nodes, $\hat{\bar{\theta}}_k$, is completely determined from the distribution of $\hat{\bar{\theta}}_{k-1}$.

This recursive nature comes from the fact that the parameters estimated by the R_k nodes is only dependent on the estimates made by the R_{k-1} nodes. The relationship between $\hat{\bar{\theta}}_{k-1}$ and $\hat{\bar{\theta}}_k$ can be intuitively understood in two steps. First, $\hat{\bar{\theta}}_{k-1}$ is the vector of synchronization parameters estimated by the nodes in R_{k-1}. Therefore, these estimates will establish the synchronization parameters for the R_k nodes since the R_{k-1} nodes communicate timing information to the R_k nodes. The synchronization parameters for R_k are found as

$$\bar{\theta}_k = \mathbf{A}_k \hat{\bar{\theta}}_{k-1} + \mathbf{B}_k.$$

Second, the R_k nodes will use the timing information from the R_{k-1} nodes to make an unbiased estimate of the parameters $\bar{\theta}_k$, which gives us $\hat{\bar{\theta}}_k$. In fact, the two terms that are added to give us $\bar{\Sigma}_k$ in equation (5.6) represent the two steps. $\mathbf{A}_k \bar{\Sigma}_{k-1} \mathbf{A}_k^T$ is the error from step 1, the error propagated to R_k from R_{k-1}. Σ_{m_k} is the error from step 2, the estimation error at R_k.

Since any node ki's skew and offset estimates are found as affine transforms of $\hat{\bar{\theta}}_k$ in (5.7) and (5.8), respectively, we see that any estimation errors made by the R_{k-1} nodes will be propagated to the R_k nodes' estimates of clock skew and clock offset. However, the intuitive understanding of cooperative time synchronization comes from realizing that the matrix $\mathbf{A}_k$ takes an "average" over $\hat{\bar{\theta}}_{k-1}$ thus mitigating the errors made by any particular node $(k-1)i$. As a result, the synchronization parameters communicated to the R_k nodes will be less noisy and, therefore, the skew and offset estimates made by a node ki will have less error. We would, thus, expect the variance of the estimates to decrease with increasing $\bar{N}$. Notice that our Type I network analysis does not explicitly utilize the circular transmission region with radius R.

5.5 Type I Node Deployment — Simulations

In Fig. 5.6 we illustrate the MATLAB simulation results for two 20 hop networks, one with $\bar{N} = 2$ and the other with $\bar{N} = 4$. The following parameters were used:

$$\begin{aligned} R &= 1 \\ d &= 5 \\ m &= 4 \\ \sigma &= 0.01 \end{aligned}$$

For each network, a set of $N = 20\bar{N}+1$ nodes were first placed in a Type I network deployment. Each node's skew parameter was then generated using $\alpha_i = |X_i|$ for $X_i \sim \mathcal{N}(1, 0.005)$, independently for each node i. Node 1 was assumed to have $\alpha_1 = 1$. The cooperative time synchronization protocol was then run 5000 times using the deployed network. At each hop, the 5000 skew and offset estimates of one chosen node were used to generate the simulated skew and offset estimate variance curves shown in Fig. 5.6. The theoretical variance value of the chosen node at each hop was computed using the recursive expression found in (5.6). In Fig. 5.7, we illustrate a similar plot using $\bar{N} = 3$ and $\bar{N} = 6$ with $\sigma = 0.05$.

In Fig. 5.6, we first clearly see that the simulated skew and offset variance values nicely match the predicted theoretical variance values. As well, the expected decrease in skew and offset variance as $\bar{N}$ increases from 2 to 4 is immediately noticeable. In fact, in both the skew variance and offset variance curves, we have an approximate halving of the variance values as we double $\bar{N}$ from 2 to 4. Also expected, is that the variance values at each hop depend on the particular values of α_i, $i = 1, \ldots, N$. This dependence on the α_i values result in the jagged skew and offset variance curves seen in Fig. 5.6. The $\bar{N} = 2$ network had α_i values

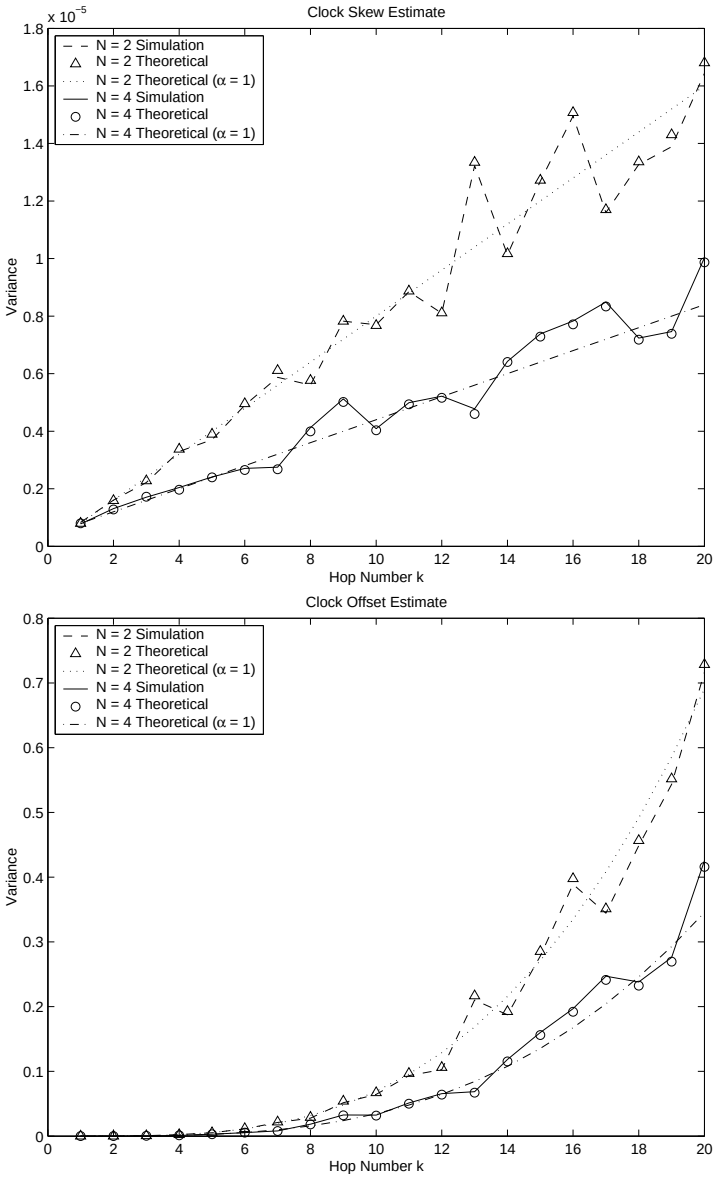

Figure 5.6: $\mathrm{Var}(\hat{\alpha}_{k1})$ is plotted in the top figure and $\mathrm{Var}(\hat{\Delta}_{k1})$ is plotted in the bottom figure as a function of k.

ranging from 0.9073 to 1.1342, while the $\bar{N} = 4$ network had skew values ranging from 0.8339 to 1.1669. Similar observations also hold for Fig. 5.7 where the $\bar{N} = 3$ network had α_i values ranging from 0.8465 to 1.1544, while the $\bar{N} = 6$ network had skew values ranging from 0.8310 to 1.1636.

The problem with having the variance curves depend on the actual skew values is that the exact performance of cooperative time synchronization is dependent on the network realization. However, we find that for α_i values that are close to and centered around 1, the variance curves follow the trend established by the theoretical variance curves for $\alpha_i = 1$, all i. This can be seen in Fig. 5.6 where we have also plotted the theoretical curves using $\alpha_i = 1$ for all i for $\bar{N} = 2$ and $\bar{N} = 4$. Again in Fig. 5.7, we see that the simulated skew and offset variance curves follow the trend established by the theoretical variance curves for $\alpha_i = 1$, all i, for both $\bar{N} = 3$ and $\bar{N} = 6$. As a result, the situation where $\alpha_i = 1$, all i, can be used to study the the performance improvement of cooperative time synchronization without dealing specifically with the skew values of individual nodes.

Therefore, to get a better understanding of how cooperative time synchronization improves synchronization performance, let us simplify the recursive expression in (5.6) for the special case where $\alpha_i = 1$ for all i and find a non-recursive expression for skew and offset variance. The first thing to note is that under the assumption of $\alpha_i = 1$ for all i, $\mathbf{A}_k = \mathbf{A}$ and $\Sigma_{m_k} = \Sigma_m$ are no longer dependent on k. Therefore, writing out the recursive expression for $\bar{\Sigma}_k$ (5.6), we have

$$\bar{\Sigma}_k = \sum_{i=0}^{k-2} \mathbf{A}^i \Sigma_m (\mathbf{A}^T)^i + \mathbf{A}^{k-1} \bar{\Sigma}_1 (\mathbf{A}^T)^{k-1}. \tag{5.9}$$

Using (5.9), Corollary 1 gives us the non-recursive expression for skew and offset variance.

Corollary 1 *For a basic cooperative network with $\alpha_i = 1$, all i, $\hat{\alpha}_{ki}$ and $\hat{\Delta}_{ki}$ have the following mean and variance:*

$$E(\hat{\alpha}_{ki}) = 1$$

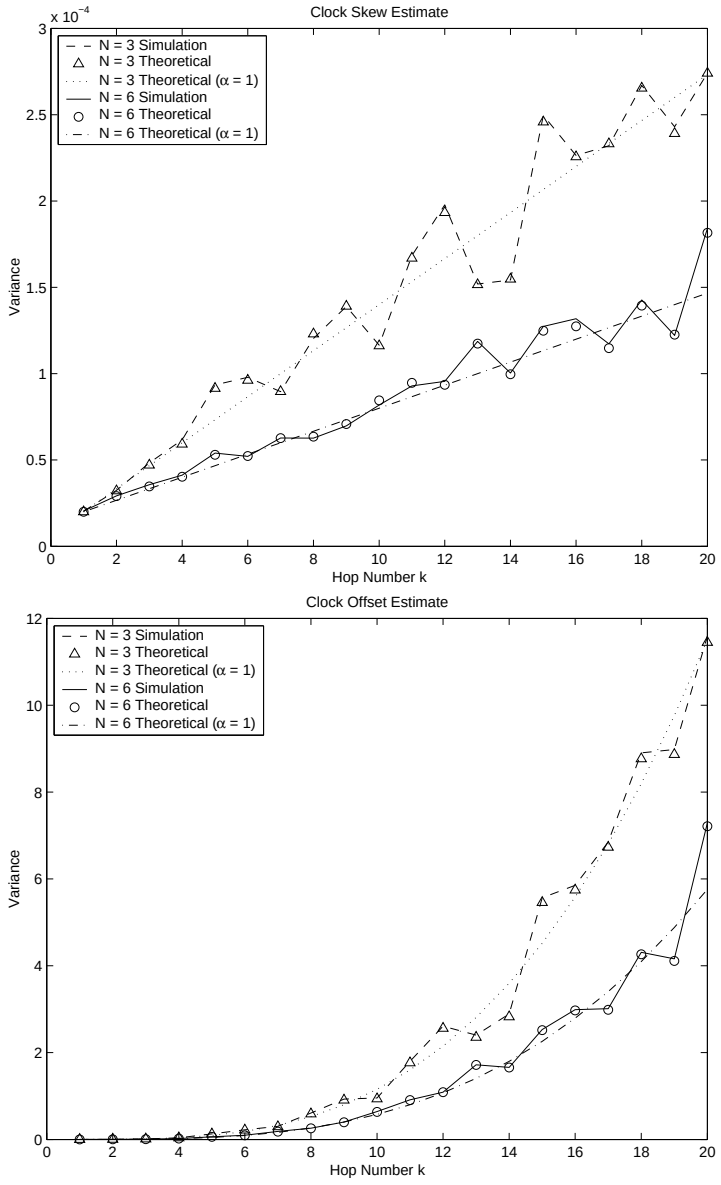

Figure 5.7: $\mathrm{Var}(\hat{\alpha}_{k1})$ and $\mathrm{Var}(\hat{\Delta}_{k1})$ are plotted. In this figure, $\sigma = 0.05$.

$$E(\hat{\Delta}_{ki}) = -\bar{\Delta}_{ki}$$

$$Var(\hat{\alpha}_{ki}) = \frac{12\sigma^2}{d^2(m-1)m(m+1)}\left(1 + \frac{2(k-1)}{\bar{N}}\right) \tag{5.10}$$

$$Var(\hat{\Delta}_{ki}) = \frac{2\sigma^2(2m-1)}{m(m+1)} + \frac{\sigma^2}{\bar{N}}\Bigg[\frac{4(k-1)(2m-1)}{m(m+1)} + (k-1)^2\Bigg(-\frac{12}{(m+1)} + \frac{12m}{(m-1)(m+1)}\Bigg) + \frac{1}{3}(k-2)(k-1)(2k-3)\frac{12m}{(m-1)(m+1)}\Bigg] \tag{5.11}$$

where k is a positive integer. $\triangle$

The proof of Corollary 1 is found in Appendix B.2. Note that the skew and offset variance expressions are only a function of k and not i. The theoretical skew and offset variance of the ith node at the kth hop (node ki) can be found in elements $(2(i-1)+2,\, 2(i-1)+2)$ and $(2(i-1)+1,\, 2(i-1)+1)$, respectively, of $\bar{\Sigma}_k$ in (5.9). However, the skew variance values in elements $(2(i-1)+2,\, 2(i-1)+2)$, $i = 1, \ldots, \bar{N}$, are all equal and the offset variance values in elements $(2(i-1)+1,\, 2(i-1)+1)$, $i = 1, \ldots, \bar{N}$, are also equal when we assume that $\alpha_i = 1$ for all i. As a result, we can consider the skew and offset variance at a hop k without specifying a particular node. Notice also that, besides the sign change in the mean of the offset estimate, the skew and offset estimates are unbiased estimates of the clock parameters of node ki.

The theoretical skew and offset variance curves from (5.10) and (5.11), respectively, are compared to simulated skew and offset variance curves in Fig. 5.8. The same parameters used in Fig. 5.6 were used to generate Fig. 5.8 except we assume that $\alpha_i = 1$ for all i. We have plotted the skew estimate variance and offset variance as a function of hop number for 15 hops and the plots are generated using 1000 runs. This is done for $\bar{N} = 1, 2, 4, 8$ and it is immediately clear that as $\bar{N}$ increases, the variance values of the skew and offset estimates decrease. Also, we see that every time $\bar{N}$ is doubled, the variance values are approximately halved. This comes from the $1/\bar{N}$ factor in both (5.10) and (5.11) and is expected since

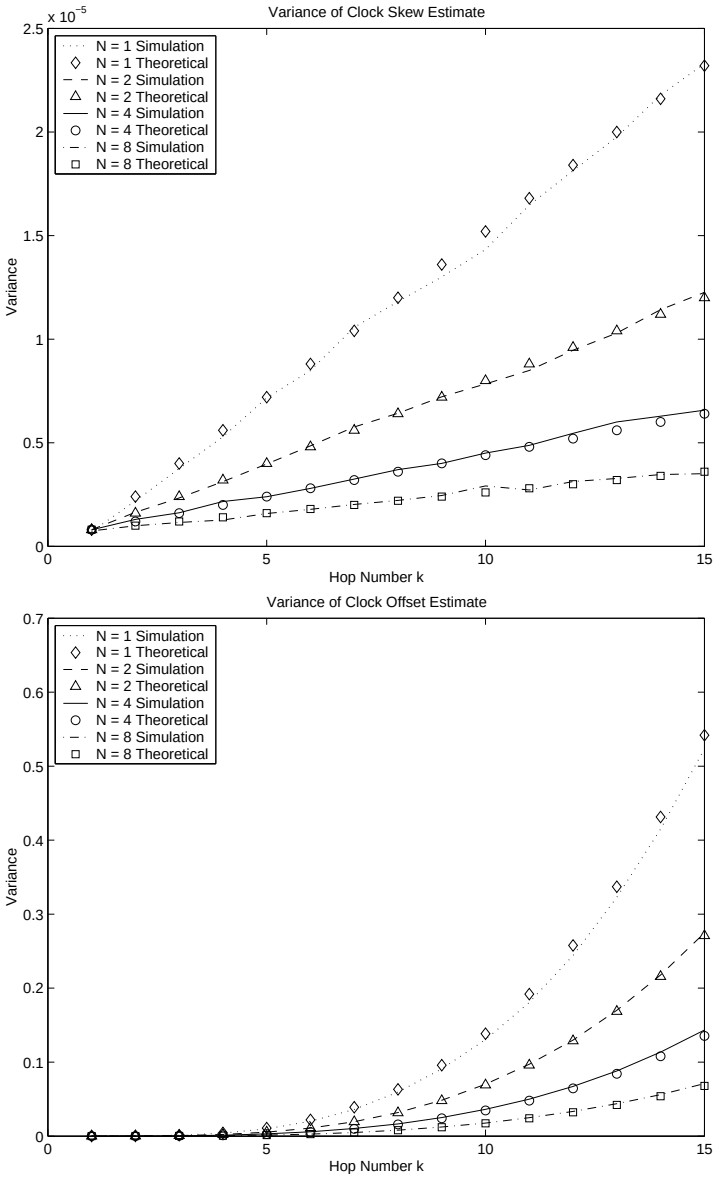

Figure 5.8: $\mathrm{Var}(\hat{\alpha}_{ki})$ and $\mathrm{Var}(\hat{\Delta}_{ki})$ are plotted as a function of k under the assumption of $\alpha_i = 1$ for all i. Variance values for skew and offset estimates decrease approximately as $1/\bar{N}$.

every node takes the sample mean of $\bar{N}$ pulses to be an observation. The variance of the observation error decreases like $1/\bar{N}$ because it is a sample mean and, thus, it is not surprising that the skew and offset variance values also approximately decrease like $1/\bar{N}$.

5.6 Type II Node Deployment — Analysis

5.6.1 Network Setup

Nodes will not generally be clustered together as in a basic cooperative network, but be deployed in a more random manner. As a result, to study general network deployments, we will consider a Type II situation where nodes are uniformly deployed with density ρ over a circular region of radius LR with node 1 at the center. In such a setup, at any hop k, $k \geq 2$, a node ki in the R_k nodes will see *at least* $\bar{N}$ nodes from the R_{k-1} set of nodes. However, the exact number of observed nodes will depend on node ki's location in the region occupied by the R_k nodes.

An illustration of a Type II deployment is shown in Fig. 5.9. We note that the R_0 node (node 1) is placed at the center of the disk and the R_1 nodes occupy a circular region of radius R. However, the region occupied by the R_k nodes for $k \geq 2$ is a ring centered around node 1 with a ring thickness of $d_{max,k}$. For increasing k, the distance from node 1 to the inner circular boundary of the region occupied by the R_k set of nodes increases.

5.6.2 Analysis

To study a Type II network, we could carry out an analysis similar to the one we did for the Type I basic cooperative network. Assuming we know the location of all nodes for a given network deployment over the circular region of radius LR, we would be able to determine the neighbors of each node and then readily extend the Type I analysis to this Type II network. The primary change that would occur in the analysis is the determination of the affine transform

$$\hat{\bar{\theta}}_k \mapsto \bar{\theta}_{k+1} = \mathbf{A}_{k+1}\hat{\bar{\theta}}_k + \mathbf{B}_{k+1}.$$

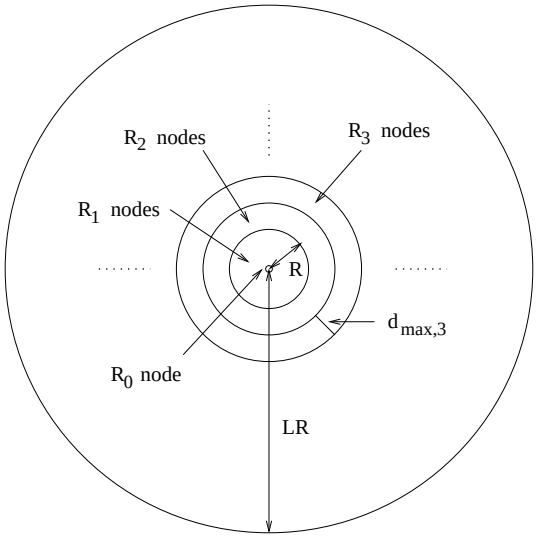

Figure 5.9: A Type II network deployment. Nodes are deployed with uniform density ρ and node 1 is at the center of the network.

However, there are two issues that arise in determining the transform matrix $\mathbf{A}_{k+1}$ and vector $\mathbf{B}_{k+1}$.

First, since R_k and R_{k+1} will most likely have different numbers of nodes, we immediately see that $\mathbf{A}_{k+1}$ will be a $2|R_{k+1}| \times 2|R_k|$ matrix and $\mathbf{B}_{k+1}$ will be a $2|R_{k+1}| \times 1$ vector, where $|R_k|$ is the cardinality of set R_k. This means that the length of vector $\hat{\hat{\theta}}_k$ will change with every hop.

Second, for any node $(k+1)i$ in R_{k+1}, the set of cooperating nodes in R_k will be different. Thus, $\mathbf{A}_k$ will also reflect this difference. Therefore, every time we move from hop k to $k+1$, the correlation structure of $\hat{\hat{\theta}}_k$ will change.

Together, these two points suggest that even though it is possible to carry out the full analysis, the complexity would make the resulting expressions depend on the particular network realization and not provide significant insight into the problem. In fact, it would be nearly impossible to visualize the result without carrying out a numerical evaluation. Since our goal is to comprehend the impact of spatial

averaging on general networks, we choose to proceed directly with simulations and compare the results with our analytical expressions for Type I networks.

In the following analysis, we develop a basic understanding of what we would expect to see in the simulation results that are presented in Section 5.7. We assume that the number of nodes in any given area of the Type II network is proportional to the area. The reason is that for uniformly deployed nodes with density ρ, averaging over all network realizations will give us that the average number of nodes in an area $\mathcal{A}$ is $\mathcal{A}\rho$. Note that even though the analysis and simulation results for Type II networks use the assumption of a circular transmission range of R, the simulation results in Section 5.7 still provide valid insight when realistic transmission regions [43, 44] are assumed since the figures illustrate synchronization error as a function of *hop* number. Therefore, regardless of the shape of the transmission region, a node at hop k will have received the appropriate synchronization information and, thus, our simulation results reflect its synchronization performance.

Estimation of $\bar{L}$

Our first consideration is to estimate the number of hops, $\bar{L}$, required to communicate timing information from node 1 to the edge of the network a distance LR away. In order to do this, we need a way to quantify $d_{max,k}$. In Fig. 5.10, we illustrate $d_{max,2}$ and see that $d_{max,2}$ is determined by having the intersection of the two radius R circles contain an average of $\bar{N}$ nodes. This is because if we increase $d_{max,2}$, then nodes at this increased distance will not see $\bar{N}$ nodes on average and, thus, not be considered an R_2 node. However, $d_{max,k} > d_{max,2}$, for $k > 2$, because the ring occupied by the R_k nodes increases in size for increasing k. We can see this in Fig. 5.11 and realize that $d_{max,k}$ can be slightly larger since the circles to not

need to intersect as much to have an area that contains an average of $\bar{N}$ nodes. As a result, we choose to be conservative and let $d_{max} \triangleq d_{max,2}$ approximate $d_{max,k}$ for all k. This means that our estimate of $\bar{L}$ using d_{max} will be greater than or equal to the number of hops required to reach a distance of LR when the differences in $d_{max,k}$ are considered.

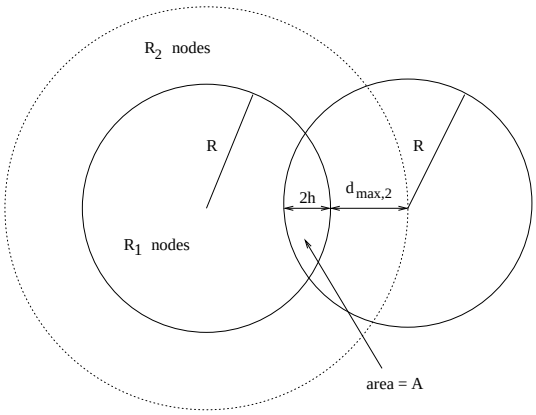

Figure 5.10: An illustration of $d_{max,2}$.

Let A be the area of the intersection of the two radius R circles in Fig. 5.10 and we have from [48] that

$$A = 2\left(R^2 \cos^{-1}\left(\frac{R-h}{R}\right) - (R-h)\sqrt{2Rh-h^2}\right). \tag{5.12}$$

Since A contains $\bar{N}$ nodes, we have that

$$A = \bar{N}/\rho. \tag{5.13}$$

From (5.12) and (5.13) we can numerically determine h thus giving us

$$d_{max} = R - 2h. \tag{5.14}$$

As a result, we need $\bar{L}$ to satisfy

$$R + (\bar{L}-1)d_{max} \geq LR$$

which means that

$$\bar{L} = \left\lceil \frac{R(L-1)}{R-2h} + 1 \right\rceil . \tag{5.15}$$

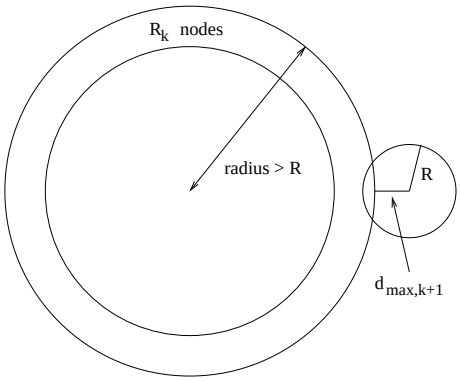

Figure 5.11: $d_{max,k} > d_{max,2}$ for $k > 2$ since the ring occupied by the R_k nodes increases in size with increasing k. Thus, less overlap between the two circles is needed to have an average of $\bar{N}$ nodes in the intersection.

Comparison to Type I Networks

We will compare the Type II network simulation results to the Type I analytical results. This comparison will allow us to carry over the intuition regarding spatial averaging that we have developed for the basic cooperative network. However, Type I and Type II networks differ primarily in that Type I networks assume all nodes will observe $\bar{N}$ neighbors from the previous hop while any node in a Type II network will only see *at least* $\bar{N}$ nodes. Thus, if we want to compare Type I and Type II plots, we need to establish some meaningful choices of the number of cooperating nodes for use with expressions (5.10) and (5.11).

Looking at (a) of Fig. 5.12, we see that if a node ki in the region occupied by the R_k nodes is at the circular boundary farthest from node 1 (outer circular boundary), then it will likely hear only $\bar{N}$ nodes from R_{k-1}. That is, there are

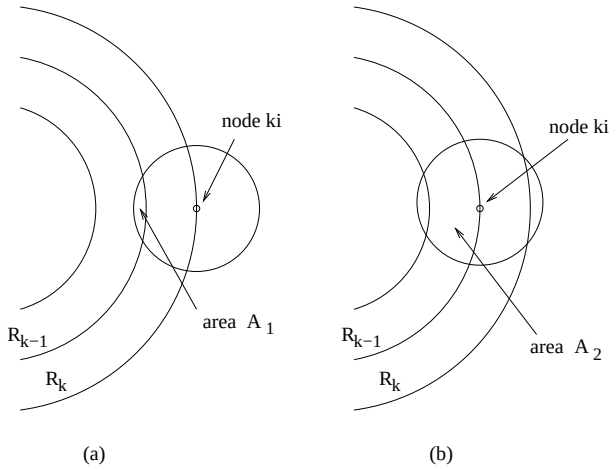

Figure 5.12: (a) Node ki at the outer circular boundary of the R_k set of nodes. (b) Node ki at the inner circular boundary of the R_k set.

$\bar{N} = A_1\rho$ nodes in area A_1. Recall that $\bar{N}$ is the minimum number of R_{k-1} nodes any node ki will hear. However, looking at (b) in Fig. 5.12, a node ki at the circular boundary closest to node 1 (inner circular boundary) in the R_k region will hear many more nodes. In fact, a node ki at the boundary between R_{k-1} and R_k will hear the largest average number of nodes $\bar{N}_{max}(k) = A_2\rho$. Since $\bar{N}$ and $\bar{N}_{max}(k)$ is the range of the number of cooperating nodes seen by a node in R_k, it would make sense to plot Type I expressions (5.10) and (5.11) using these two values. However, $\bar{N}_{max}(k)$ varies with k. In Fig. 5.13 we illustrate the regions occupied by the R_k nodes for $k = 1$, $k > 1$, and $k >> 1$ overlayed on top of each other and in each situation, we see that the set of nodes in R_k seen by a node at the boundary between the R_k nodes and the R_{k+1} nodes is different for changing values of k. However, it is clear that the area of intersection always falls inside a semicircle of radius R. As a result, we will approximate $\bar{N}_{max} = \max_{k:k\geq 2} \bar{N}_{max}(k)$, by upper bounding the maximum area of intersection with the area of the semicircle. This

means that

$$\bar{N}_{max} \approx \rho \frac{\pi R^2}{2}. \tag{5.16}$$

Thus, in comparing Type II and Type I results, we will use $\bar{N}$ and $\bar{N}_{max}$ in (5.16) with both (5.10) and (5.11)

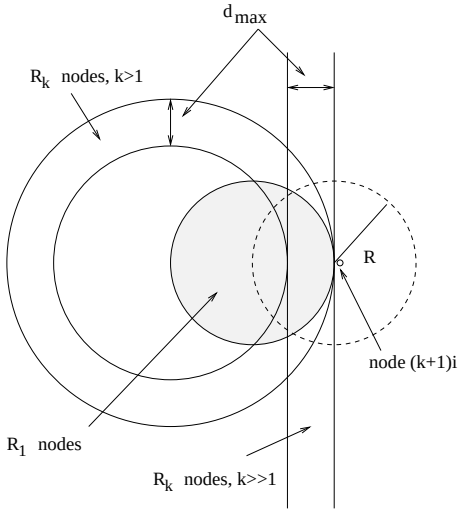

Figure 5.13: The regions occupied by the R_k nodes for $k = 1$, $k > 1$, and $k >> 1$ overlayed on top of each other. The region of nodes seen by a node at the inner circular boundary of R_{k+1} changes with k.

Using $\bar{N}$ with (5.10) and (5.11) will provide a curve that tends to be higher than the Type II simulated curves for two main reasons. First, since $\bar{N}$ is the minimum number of nodes in R_{k-1} that a node ki in R_k will hear and we know that a larger number of cooperating nodes will result in decreased estimation variance, the variance values computed using $\bar{N}$ will tend to be higher. Second, even if a node ki in R_k hears $\bar{N}$ nodes from R_{k-1}, each of those $\bar{N}$ nodes did not necessarily only hear $\bar{N}$ nodes from R_{k-2}. Thus, the skew and offset estimates made by each of those $\bar{N}$ nodes in R_{k-1} whose transmissions are being heard by node ki may have a variance that is less than predicted by (5.10) and (5.11) using $\bar{N}$. The

improved skew and offset estimates made by the nodes in R_{k-1} will thus lead to a lower estimation variance for node ki even though node ki hears only $\bar{N}$ from R_{k-1}.

Using $\bar{N}_{max}$ with (5.10) and (5.11) will provide a curve that tends to be lower than the Type II simulated curves for two similar reasons. First, since $\bar{N}_{max}$ is the average number of nodes heard by a node at the inner circular boundary of R_k, $k \geq 2$, and all other nodes in R_k will on average hear fewer nodes, a Type I curve using $\bar{N}_{max}$ will tend to yield lower values. Second, not all nodes in R_{k-1} make their estimates using a signal cooperatively generated by $\bar{N}_{max}$ nodes. In fact, most nodes in R_{k-1} observe fewer than $\bar{N}_{max}$ nodes. As a result, the lower quality estimates made by some of the R_{k-1} nodes will cause the estimation variance of the R_k nodes that hear $\bar{N}_{max}$ from R_{k-1} to be greater than predicted by (5.10) and (5.11) using $\bar{N}_{max}$.

Synchronization Performance and Node Density

The third issue we want to address in analyzing a Type II network deployment is how to decrease synchronization error when we know that the number of hops $\bar{L}$ required to communicate timing information from node 1 to the edge of the network a distance LR away is determined by $\bar{N}$. Given a fixed R, we can start with some $\bar{N}$ and ρ. Using (5.12), (5.13), and (5.14), we can determine the value of d_{max} and, hence, from (5.15) the number of hops $\bar{L}$ required to send timing information from node 1 to the edge of the network. In order to decrease synchronization error at a distance LR from node 1, we need to increase $\bar{N}$. However, only increasing $\bar{N}$ will decrease d_{max} and increase $\bar{L}$. Therefore, we need to increase both $\bar{N}$ and ρ. From (5.12) and (5.13), we see that if $\bar{N}/\rho$ is kept constant, then h will be constant.

If h is constant, then so is d_{max}. As a result, by increasing node density, we can increase the minimum number of cooperating nodes $\bar{N}$ and therefore decrease synchronization error.

5.7 Type II Node Deployment — Simulations

In the following simulation results, we have assumed that all nodes in the network have no clock skew, i.e. $\alpha_i = 1$ for all i. From Section 5.5 we know that general α_i values result in variance curves that follow the trends established by curves generated using $\alpha_i = 1$. As a result, using $\alpha_i = 1$ for all i allows us to study the benefits of spatial averaging without considering effects that are dependent on the particular network realization.

5.7.1 Comparison to Type I Results

To being the study of cooperative time synchronization in general networks, we deploy a network for Simulation 1 with the parameters in Table 5.2. The simulation results are displayed in Fig. 5.14. In each run, a new network of nodes was uniformly deployed over a circular area of radius $LR = 5$ and the MATLAB simulator implemented the cooperative time synchronization protocol. Besides plotting the Type I comparison curves described in Section 5.6.2, we also plot the sample variance of the best performing node and the worse performing node. In each run, the node in R_k that sees the fewest number of nodes from R_{k-1} is considered the worse performing node while the node in R_k that sees the largest number of nodes from R_{k-1} is the best performing node. For the lth run, the fewest number of nodes seen by a node in R_k is denoted $X_{min}^{(l)}(k)$ while the largest number of nodes seen by a node in R_k is denoted $X_{max}^{(l)}(k)$. The skew and offset estimate of the best

Table 5.2: Simulation 1 Parameters

ρ	19.10
$\bar{N}$	4
R	1
L	5
d	2
m	4
σ	0.01
Number of Runs	5000

and worst performing node at each hop is recorded and the sample variance over the 5000 runs is plotted.

The top figure in Fig. 5.14 illustrates the sample skew variance curves of the worst and best synchronized node along with the Type I curves for comparison. The bottom figure in Fig. 5.14 illustrates the clock offset estimate sample variance. Note that using equation (5.15) and the parameters in Table 5.2, we find that $\bar{L} = 7$. From the simulations, we also see that 7 hops are required to traverse the network. In fact, only 7.32% of the networks required more than 7 hops to reach all nodes in the network.

As predicted in Section 5.6.2, we clearly see in Fig. 5.14 that the worst case variance and the best case variance are sandwiched between the Type I comparison curves. Also, as expected, the skew and offset variances do not closely follow the upper and lower Type I comparison curves. The worst case skew and offset variance follow the upper comparison curve for the first 2 hops and then begin do

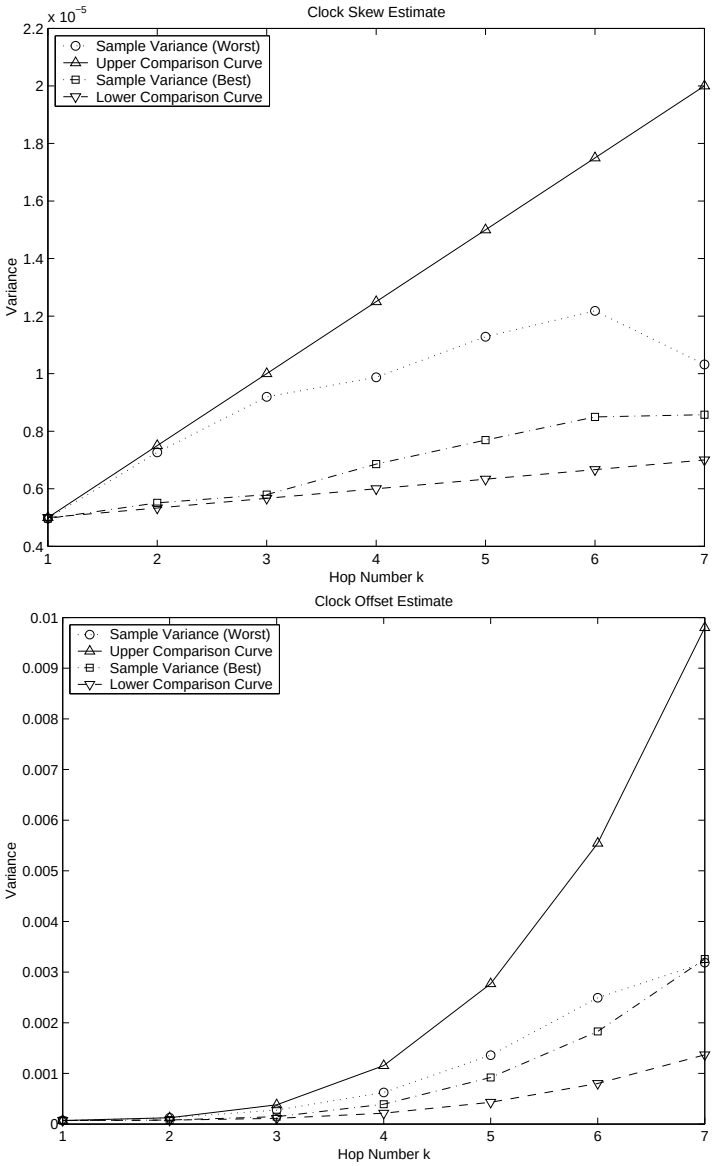

Figure 5.14: Simulation 1. Top: Sample variance for the skew estimate of a Type II network along with Type I comparison curves. Bottom: Sample variance of the offset estimate along with Type I comparison curves.

deviate from the curve. As mentioned in Section 5.6.2, this is because the nodes contributing to the worst performing node may have received signals from more than $\bar{N}$ nodes. Similarly, the best case skew and offset variance follow the lower comparison curve for the first 2 hops before deviating. This is because many of

the nodes contributing signals to the best performing node made their estimates using a signal cooperatively generated by less than $\bar{N}_{max}$ nodes. Also of interest is the steep decrease in the worst case skew and offset variance at hop $k = 7$. This is due to the fact that on average, the distance from the outer circular boundary of the R_6 region to the network boundary is much less than d_{max}. As a result, the R_7 region is smaller and $X_{min}(7)$ will be larger than $\bar{N}$. Table 5.3 shows the $X_{min}(k) = \frac{1}{5000}\sum_{l=1}^{5000} X_{min}^{(l)}(k)$ and $X_{max}(k) = \frac{1}{5000}\sum_{l=1}^{5000} X_{max}^{(l)}(k)$ values and we see that $X_{min}(6) = \bar{N} = 4$, but $X_{min}(7)$ is nearly twice $X_{min}(6)$.

Table 5.3: $X_{min}(k)$ and $X_{max}(k)$ for Fig. 5.14

k	$X_{min}(k)$	$X_{max}(k)$
1	1	1
2	4.00	27.56
3	4.00	29.36
4	4.00	31.86
5	4.00	33.50
6	4.00	34.60
7	7.77	35.32

We also note that $X_{max}(k)$ increases from 27.56 for $k = 2$ to 35.32 for $k = 7$. Using (5.16), however, we find that $\bar{N}_{max} = 30$. The reason $X_{max}(k)$ increases with each hop and does not equal $\bar{N}_{max}$ is because $X_{max}(k)$ is a different statistic. $\bar{N}_{max}$ approximates the average number of R_{k-1} nodes seen by a node ki at the inner circular boundary of R_k. However, $X_{max}^{(l)}(k)$ is *the* largest number of nodes seen by any node ki in R_k for the lth network realization. Therefore, $X_{max}^{(l)}(k)$ is actually

an ordered statistic since it takes the largest number of nodes seen by a node at hop k. $X_{max}(k)$ is thus the mean of the ordered statistic. Therefore, we would not expect $\bar{N}_{max}$ and $X_{max}(k)$ to be the same. Also, $X_{max}(k)$ increases with k since as the circumference of the circular ring occupied by R_k increases, there are more nodes at the boundary between R_k and R_{k-1}. Since there are more nodes at the boundary, there are also more opportunities to find the largest number of nodes seen by a node ki. Thus, the maximum number of nodes would tend to be larger. Note that, not considering the effects at the network boundary, $X_{min}(k) = \bar{N}$ because the definition of the protocol specifies the minimum to be $\bar{N}$ and it is nearly always the actual minimum number of observed nodes.

Table 5.4: Simulation 2 Parameters

ρ	7.47
$\bar{N}$	2
R	1
L	9
d	2
m	3
σ	0.01
Number of Runs	5000

To see that a similar behavior holds for different parameters and larger networks, we consider Simulation 2 (Fig. 5.15) with parameters shown in Table 5.4. As in Fig. 5.14, we see in Fig. 5.15 that the worst case skew and offset variance values follow the upper comparison curve for the first two hops before deviating away.

Similarly, the best case skew and offset variance values deviate from the lower comparison curve after two hops. However, the sample variance values clearly fall between the comparison curves over the 12 hops required to synchronize the larger network.

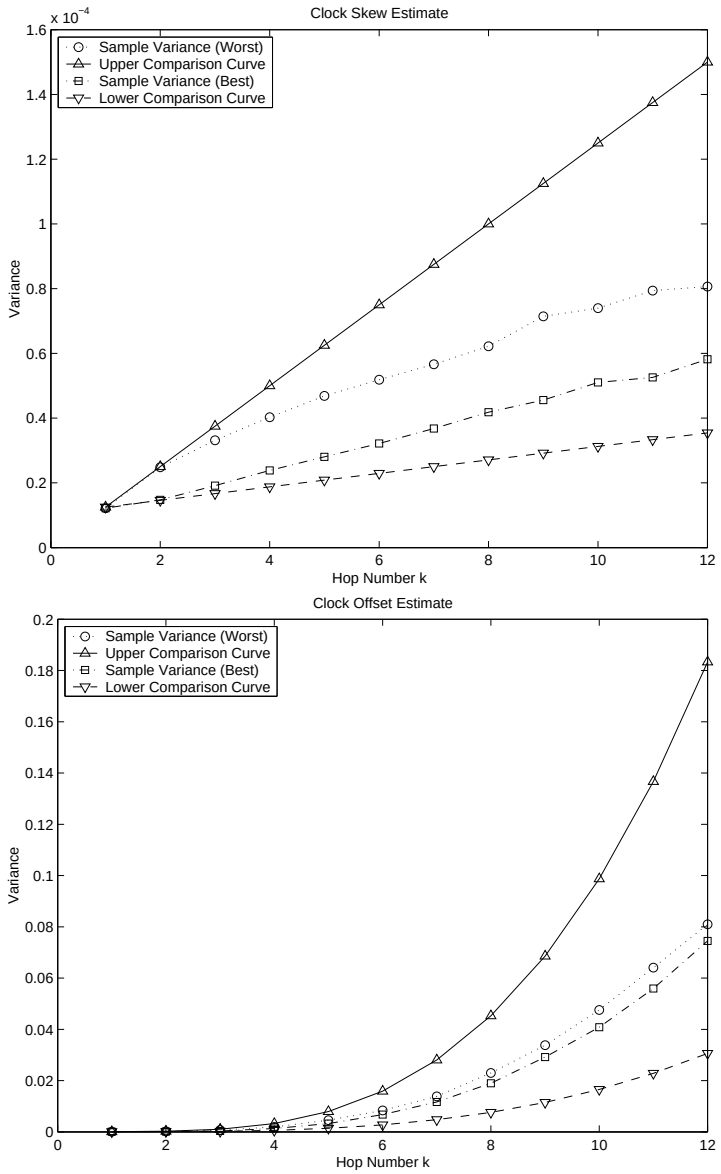

Figure 5.15: Simulation 2. Top: Comparison curves with the skew variance for a larger Type II network. Bottom: Comparison curves with the offset variance.

One thing to note about Simulation 2 is that using its parameters in Table 5.4

with equation (5.15), we find that $\bar{L} = \lceil 12.36 \rceil = 13$. However, from the figure we have only plotted 12 hops. The reason for this discrepancy is that in the 5000 runs used to generate the sample variance values of Fig. 5.15, 4999 of the runs required 13 or more hops to traverse the network while one run needed only 12 hops. As a result, even though we have plotted the results using the fewest number of hops, we see that $\bar{L}$ still gives a very good estimate of the number of hops required to cover the network.

5.7.2 Synchronization Performance and Node Density

Table 5.5: Simulation 1b Parameters

ρ	23.87
$\bar{N}$	6
R	1
L	5
d	2
m	4
σ	0.01
Number of Runs	5000

Next, we want to improve synchronization performance by increasing node density. Starting with the parameters for Simulation 1, we increase the minimum number of cooperating nodes to $\bar{N} = 6$ while keeping $\bar{N}/\rho = 0.25$ constant. Therefore, for Simulation 1b (Table 5.5), $\rho = 23.87$ and we plot the simulation results in Fig. 5.16. Comparing Fig. 5.14 and Fig. 5.16, it is clear that Fig. 5.16 yields

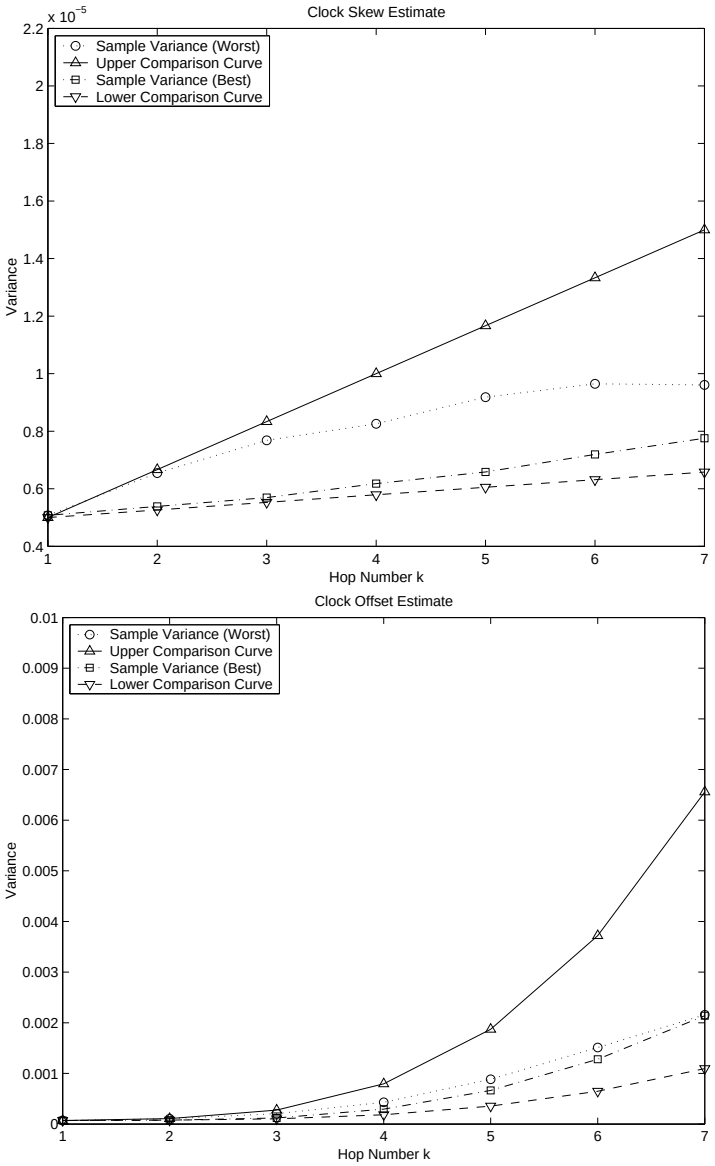

Figure 5.16: Simulation 1b. The Type I comparison curves and the Type II sample skew and offset variance curves are lower as compared to Fig. 5.14 when $\bar{N}$ and ρ are increased. More cooperation yields improved synchronization performance.

improved skew and offset variances, thus showing that increased node density and larger $\bar{N}$ values indeed improve synchronization performance in Type II networks. In Table 5.6 we show $X_{min}(k)$ and $X_{max}(k)$ for Fig. 5.16. Note that there is only a slight decrease in the worst case skew and offset variance curves at hop $k = 7$

since in this simulation, we have that $X_{min}(7) = 6.57$ is only slightly larger than $X_{min}(6) = \bar{N} = 6$.

Table 5.6: $X_{min}(k)$ and $X_{max}(k)$ for Fig. 5.16

k	$X_{min}(k)$	$X_{max}(k)$
1	1	1
2	6.00	34.01
3	6.00	34.64
4	6.00	37.64
5	6.00	39.50
6	6.00	40.80
7	6.57	41.70

Similarly, for Simulation 2b (Table 5.7) we have increased the density ρ and the minimum number of cooperating nodes $\bar{N}$ of Simulation 2. In Fig. 5.17, we see that by increasing $\bar{N}$ to 3 from 2 and increasing the density ρ to 11.20, we can significantly decrease the skew and offset variance. The variance of the worst and best synchronized nodes are still between the upper and lower Type I comparison curves, but it is immediately clear that the upper comparison curve is significantly lower than in Fig. 5.15 and the lower comparison curve has also decreased.

Another very effective way to visualize how increasing ρ and $\bar{N}$ can decrease skew and offset variance is to choose one *test node* in the network and consider how its skew and offset variance decreases as the network density and number of cooperating nodes are increased. In Simulation 3 (Table 5.8), we placed a test node at distance $LR = 2.2$ from node 1 and simulated its skew and offset variance

Table 5.7: Simulation 2b Parameters

ρ	11.20
$\bar{N}$	3
R	1
L	9
d	2
m	3
σ	0.01
Number of Runs	4000

as we increased ρ and $\bar{N}$. $\bar{N}$ took on values ranging from 1 to 10 and we adjusted ρ accordingly to keep $\bar{N}/\rho = 0.15$ fixed. The results are plotted in Fig. 5.18 and we clearly see that as $\bar{N}$ increases along with ρ, the skew and offset variance of this test node decreases. Also, from Section 5.6.2, we know that since we keep $\bar{N}/\rho$ constant, the number of hops required to reach the test node stays the same as we increase $\bar{N}$. Therefore, since the test node is at $\bar{L} = 3$ for every value of $\bar{N}$, we have also plotted the upper and lower Type I comparison curves for the skew and offset variance at hop $k = 3$ to illustrate how the comparison curves change in relation to the simulated variance curves. In Fig. 5.18, the simulated skew and offset variance curves of the test node fall between the Type I upper and lower comparison curves.

It is clear that by keeping the ratio $\bar{N}/\rho$ constant while increasing $\bar{N}$ and ρ allows us to reduce the synchronization error at each hop while keeping the number of hops required to synchronize the network, $\bar{L}$, constant. The variance of the skew

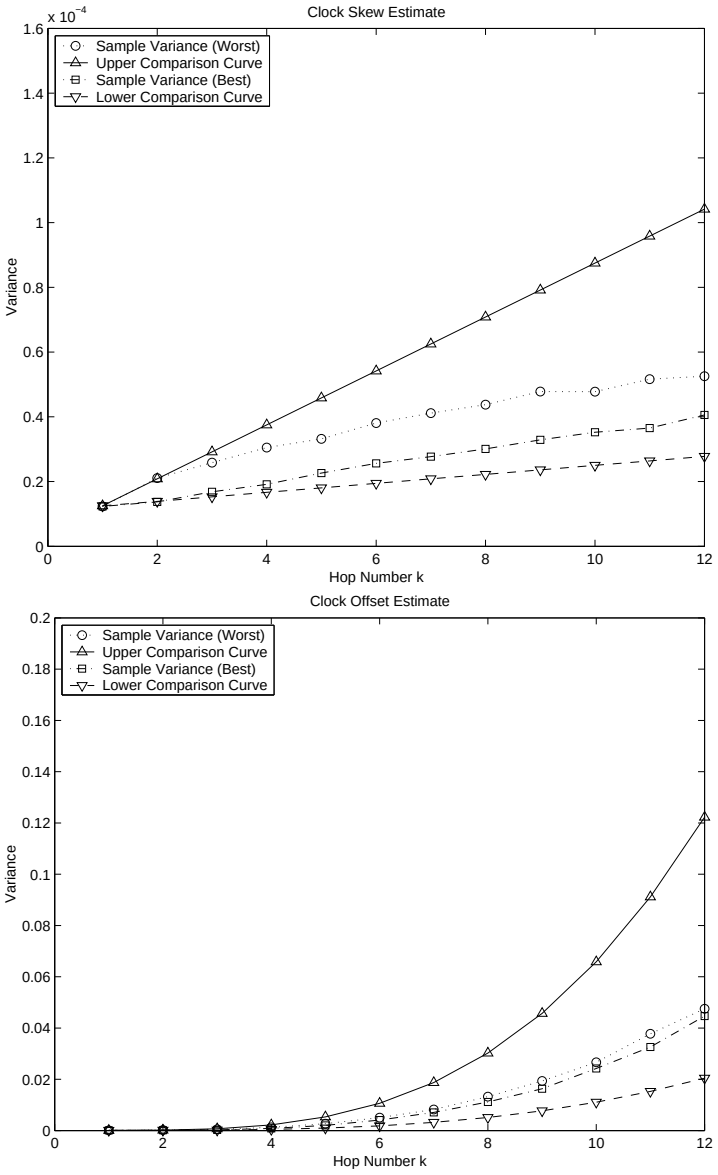

Figure 5.17: Simulation 2b. Compared to Fig. 5.15, the skew and offset variance values are decreased due to an increase in $\bar{N}$ and ρ.

and offset estimates is decreased by increasing the minimum number of cooperating nodes.

Furthermore, from the simulations in this section, we find that the upper and lower Type I comparison curves provide a good reference for the performance of Type II networks. We have established that the best and worst case variance values

Table 5.8: Simulation 3 Parameters

$\bar{N}/\rho$	0.15
$\bar{N}$	[1 2 4 6 8 10]
R	1
L	2.2
d	1
m	2
σ	0.01
Number of Runs	5000

for the Type II skew and offset estimates fall between the upper and lower Type I comparison curves. As the density of the network and $\bar{N}$ are both increased, the comparison curves will shift downwards and become closer together. Thus, we would expect the variance of the Type II network estimates to change similarly with increasing $\bar{N}$ and ρ.

5.7.3 Changing $\bar{N}$ With Fixed Node Density

We have seen from Section 5.7.2 that increasing node density ρ and increasing the minimum number of cooperating nodes $\bar{N}$ can yield significant synchronization performance. However, another question is what happens when ρ is kept constant but $\bar{N}$ is increased.

The question we consider in this section is what happens to the skew and offset estimate variance of a test node at the edge of the network (distance LR away from node 1) as we increase $\bar{N}$. This situation is unclear because it involves a complex

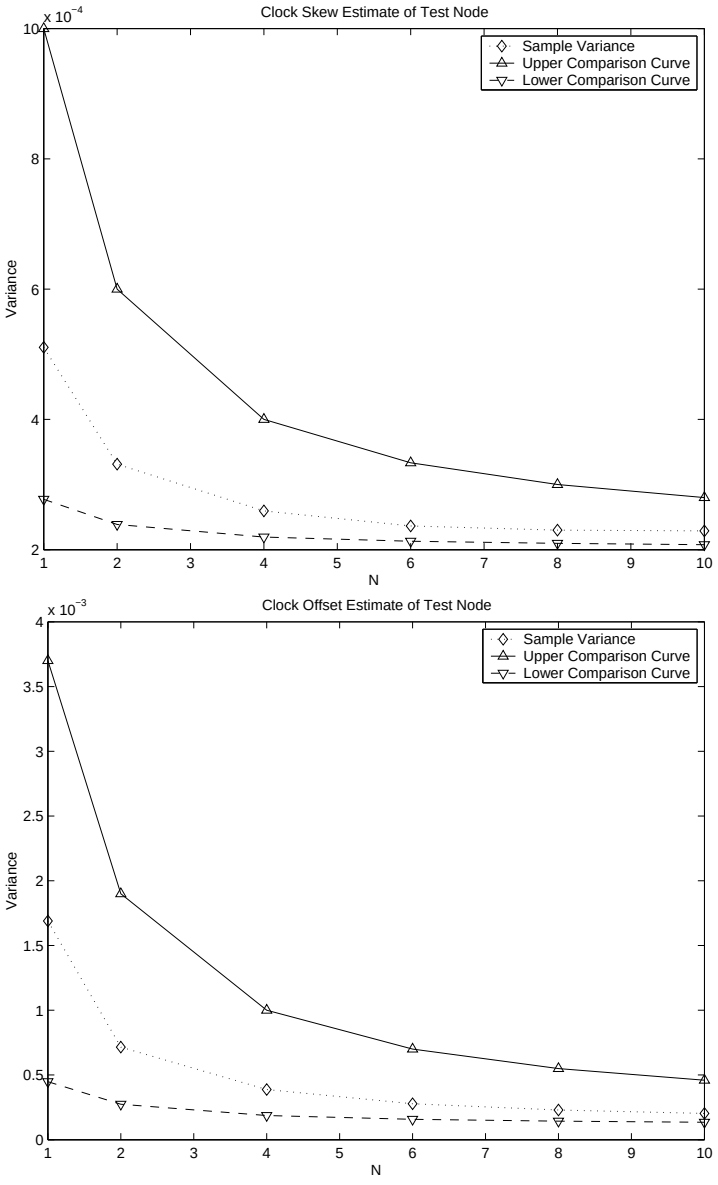

Figure 5.18: Simulation 3. Variance of the skew and offset estimates of the test node fall between the Type I comparison curves and decrease with increasing $\bar{N}$ and ρ.

interaction among three factors:

(a) Different locations inside R_k.

(b) Increasing $\bar{N}$.

(c) Increasing number of hops, k.

Factor (a) tends to increase the skew and offset variance. As $\bar{N}$ is increased and ρ is fixed, from (5.13) we see that A will increase. This will result in a larger h value and, hence, a decrease in d_{max} from (5.14). This means that each R_k ring will decrease in size and a test node that is in a fixed location will move closer to the outer circular boundary of R_k. By moving towards the outer circular boundary, however, the test node is seeing on average fewer nodes from R_{k-1} and will therefore tend to have a larger skew and offset variance.

Factor (b) tends to decrease the skew and offset variance since increasing $\bar{N}$ means more cooperation and less estimation error. As a result, even if the test node remains in set R_k, factors (a) and (b) make it unclear how the skew and offset variance will change.

Factor (c), on the other hand, also tends to increase the skew and offset variance. As we increase $\bar{N}$, it is possible to decrease d_{max} enough so that the test node goes from being in the set R_k to being in the set R_{k+1}. Since synchronization error increases with each hop, there tends to be an increase in synchronization error.

Table 5.9: Simulation 4 Parameters

ρ	44.21
$\bar{N}$	[2 4 6 8 10 12 14 16 18]
R	1
L	2.4
d	2
m	4
σ	0.01
Number of Runs	6000

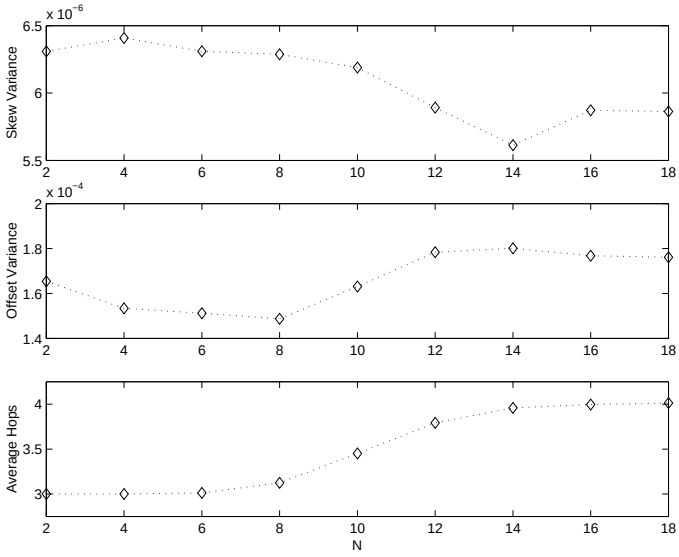

Figure 5.19: Simulation 4. The skew variance (top) and the offset variance (middle) of a test node at distance LR from node 1 is plotted as a function of changing $\bar{N}$. The average number of hops required to reach the test node (bottom) increases from 3 hops to 4 as $\bar{N}$ increases.

In Simulation 4 (Table 5.9), we simulated the variance of the skew and offset estimate of a test node located a distance $LR = 2.4$ away from node 1. We fixed the network density at $\rho = 44.21$. We increase $\bar{N}$ from 2 to 18 in increments of 2 nodes. The top plot in Fig. 5.19 shows how the skew variance changes with varying $\bar{N}$ while the middle plot illustrates how the offset variance varies with $\bar{N}$. The bottom figure shows the average number of hops required to reach the test node at distance $LR = 2.4$. It is clear in the last plot that the number of hops to reach the edge of the network increases from 3 hops to 4 hops as $\bar{N}$ increases.

Looking at the plot of the skew variance in Fig. 5.19, we first note that factor (c) does not impact the skew variance. As the number of hops required to reach the chosen node increases from 3 to 4 at $\bar{N}$ values 8 to 12, we see that the skew variance actually decreases. This means that the skew variance is generally driven

by factor (b) where increasing $\bar{N}$ values decrease the skew variance. However, we also see that when the test node is close to the inner circular boundary of an R_k region, there is an increase due to factor (a). We see that there is an increase in skew variance from $\bar{N} = 2$ to $\bar{N} = 4$ when the test node is near the inner circular boundary of R_3. Similarly, we see an increase from $\bar{N} = 14$ to $\bar{N} = 16$ when the node is near the inner circular boundary of R_4. Nonetheless, generally as $\bar{N}$ increases, the skew variance decreases.

The behavior for the offset variance plot at the middle of Fig. 5.19 is very different. The first thing we note is that factor (c) plays a dominant role. When $\bar{N}$ is increased from 8 to 12 there is a very noticeable increase in the offset variance, which coincides with the increase from 3 hops to 4 hops required to reach the test node at distance $LR = 2.4$. Factor (b), on the other hand, dominates factor (a) when the node is in the same region R_k. From $\bar{N} = 2$ to $\bar{N} = 8$, while the node is in R_3, we see a consistent decrease in offset variance due to the increase in $\bar{N}$. Similarly, from $\bar{N} = 12$ to $\bar{N} = 18$ while the test node is in R_4, there is a consistent decrease in offset variance. However, note that the improvement in offset variance due to factor (b) from $\bar{N} = 2$ to $\bar{N} = 8$ is completely wiped out by the variance increase due to factor (c) when $\bar{N}$ increases from 8 to 12. Therefore, there is no overall decrease in offset variance.

It makes sense for factor (c) to have more of an impact on the offset variance since we know that the offset variance grows at a rate faster than linear while the skew variance grows linearly with the number of hops. Thus, an extra hop will have a larger impact on the offset variance than on the skew variance. The question then becomes, if we increase the rate at which the skew variance grows, will we still see the general trend of decreasing skew variance with increasing $\bar{N}$

and fixed ρ?

Table 5.10: Simulation 5 Parameters

ρ	44.21
$\bar{N}$	[2 4 6 8 10 12 14 16 18]
R	1
L	2.4
d	1
m	2
σ	0.01
Number of Runs	8994

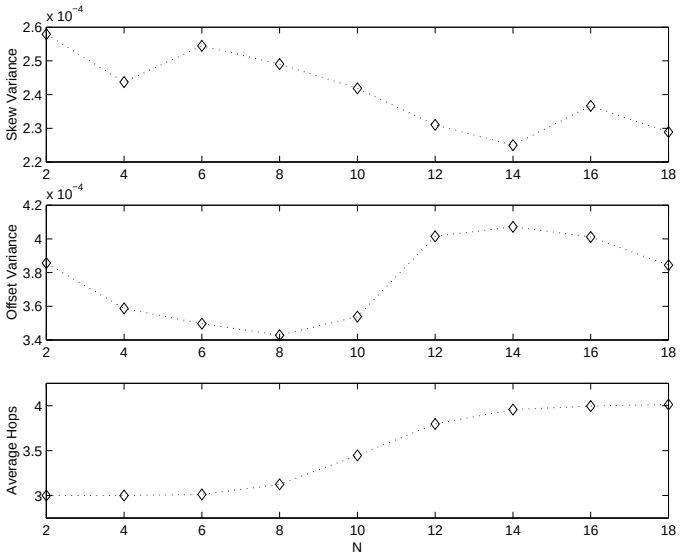

Figure 5.20: Simulation 5. The skew variance (top), offset variance (middle), and average hop number required to reach a distance $LR = 2.4$ (bottom) is shown for changing $\bar{N}$. In this figure, $d = 1$ and $m = 2$.

For Simulation 5 (Table 5.10), we keep all parameters the same except we set

$d = 1$ and $m = 2$ to increase the rate at which the skew variance grows with hop number. In Fig. 5.20, again with the skew variance we see that an increase in hop number, factor (c), does not affect the variance values. Skew variance decreases when the transition from 3 hops to 4 hops occurs. Thus, again we see that factor (b) is what primarily drives down the skew variance as $\bar{N}$ increases. However, factor (a) also impacts the variance by slightly increasing the variance value at certain points while the test node is within a single R_k region. Nonetheless, the general trend of decreasing skew variance when $\bar{N}$ increases still holds.

Also, a behavior similar to Fig. 5.19 is seen in the offset variance curve in Fig. 5.20. Factor (a) has very little affect on the offset variance while the test node is either in R_3 or R_4. In fact, factor (b) generally drives down the offset variance while the node is in R_3 from $\bar{N} = 2$ to $\bar{N} = 8$ and while the node is in R_4 from $\bar{N} = 12$ to $\bar{N} = 18$ with only minor increases in the variance due to factor (a). The increase in offset variance is again driven by factor (c), an increase in hop number. We see, however, that there is still no overall increase or decrease in the offset variance since the affects of the different factors cancel each other out.

So we see that increasing $\bar{N}$ while keeping a constant ρ yields different behavior for the skew and offset variances. While increasing $\bar{N}$ does not yield any general improvements in the offset variance, it does give a general decrease in the skew variance. However, it is important to realize that even though some performance gains can be achieved simply by increasing $\bar{N}$ and not increasing node density ρ at the same time, this technique is severely limited by how much $\bar{N}$ can be increased.

As $\bar{N}$ is increased, it becomes harder for the network to synchronize since each node is required to see more neighbors. At a certain point, the synchronization protocol will terminate before synchronizing all nodes in the network since not all

Table 5.11: Simulation 6 Parameters

ρ	44.21
$\bar{N}$	[16 18 20 22 24 26 28 30]
R	1
L	2.4
d	2
m	4
σ	0.01
Number of Runs	600

nodes are able to hear the required $\bar{N}$ neighbors. For example, in Simulation 6 (Table 5.11), we take the same network as in Simulation 4 and consider how often the cooperative synchronization protocol fails to synchronize all the nodes in the network for larger values of $\bar{N}$ and a fixed node density ρ. We see in Fig. 5.21 that for 600 randomly deployed Type II networks, at $\bar{N} = 16$ all networks were completely synchronized while at $\bar{N} = 18$ only 0.33% of the networks failed to have all nodes synchronize. However, for even larger $\bar{N}$ values, the percentage of networks whose nodes are not all synchronized increases quickly and by $\bar{N} = 24$, 98.5% of the networks are not fully synchronized. Of course, the largest $\bar{N}$ value that can be chosen will depend on the requirements of the network. If, on average, only 90% of the nodes in the network need to be synchronized, then we see that $\bar{N} = 18$ will achieve this requirement 100% of the time and $\bar{N} = 20$ will still meet this requirement 99% of the time. Also, in general, the higher the density ρ, the larger the $\bar{N}$ value can be while still meeting synchronization percentage requirements.

Nonetheless, $\bar{N}$ and ρ both need to be increased in order to consistently improve synchronization performance and ensure that the required percentage of the nodes in the network are synchronized.

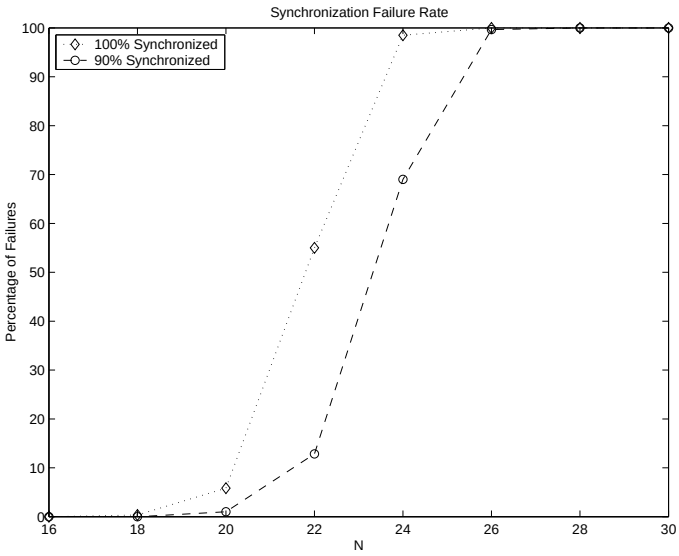

Figure 5.21: Simulation 6. With fixed node density ρ, increasing $\bar{N}$ will eventually lead to networks where the required percentage of nodes are not always synchronized.

5.7.4 Summary of Simulation Results

From the simulation results of this section, we find that given parameters R, d, and m and density ρ, the first task is to choose the number of cooperating nodes $\bar{N}$. From Section 5.7.3, we find that $\bar{N}$ should be chosen as large as possible while still guaranteeing that the desired percentage of the network is always synchronized. The reason for this is that as $\bar{N}$ increases, the general trend is for the skew variance of any specific node to decrease. Thus by choosing $\bar{N}$ as large as possible, the skew variance of any node in the network tends to be minimized.

To further improve synchronization performance of the network, node density ρ and the minimum number of cooperating nodes $\bar{N}$ can be increased while keeping

$\bar{N}/\rho$ constant. From Section 5.7.2, we clearly see that the skew and offset variance of all nodes in the network will decrease significantly with increasing ρ and $\bar{N}$.

5.8 A Comparison to Non-Cooperative Synchronization

Traditional, non-cooperative network synchronization techniques generally communicate timing information from node to node. That is, each timing data point obtained by any given node comes from one neighboring node. This is in contrast to our cooperative synchronization technique where each timing data point is constructed using information from many neighboring nodes. Before we compare cooperative and non-cooperative synchronization, let us look at three existing synchronization techniques from the view point of scalability.

5.8.1 Traditional Synchronization Techniques

Reference Broadcast Synchronization (RBS)

Reference Broadcast Synchronization [6] eliminates transmitter side uncertainties by synchronizing a set of receivers with one another. For example, consider three sensor nodes. An arbitrary node first broadcasts a reference packet to the surrounding nodes. The two receiving nodes use this packet arrival as a synchronization event and exchange this timing information with each other. Each receiving node now has the reference packet arrival time in its own time scale and the time scale of the other receiving node. With this pair of times, the node can calculate the clock offset between its local clock and the clock of the other node. This idea can be extended to use a sequence of reference packets. After the receiving nodes exchange all the timing information, they will each have a collection of data points. Each

data point is a pair of times telling the node when a particular reference packet arrived at the two receiving nodes. A least-squares linear regression is performed to estimate clock skew and clock offset. As a result, the receiving nodes can be synchronized to each other. RBS is also extended to networks with multiple hops so nodes that do not see a common reference packet can still be synchronized. This is done by using using a series of time scale conversions.

Whenever two nodes are synchronized, estimates of clock skew and clock offset are made. These estimates have inherent errors and the authors of [6] find that the synchronization error at each hop is Gaussian. Since the error at each hop is independent, they expect that the variance will grown linearly with the number of hops. The experimental results that are presented support this claim.

Timing-sync Protocol for Sensor Networks (TPSN)

The Timing-sync Protocol for Sensor Networks [8] operates in two phases. The first phase is the level discovery phase where the network is organized into a hierarchial structure. Each node in the network is assigned a level and it can communicate with at least one node in the level below it. The root node is assigned level 0. The second phase is the synchronization phase where a node in level i will initiate a handshake with a node in level $i-1$. All levels will carry-out the handshake and it involves two packet exchanges with time stamping occurring at the medium access control (MAC) layer. The packet exchanges only estimate clock offset. Due to the improved time stamping technique, synchronization performance can be better than that of RBS.

As expected, synchronization error will also increase with each hop. In the case of TPSN, the error comes from two sources. The first source of error is

the error in estimating clock offset. The second source of error stems from clock skew. Since TPSN does not explicitly estimate clock skew, after the estimate of clock offset between any two nodes, the skew of the clocks will immediately cause the nodes to loose synchronization. The experimental results in [8] do not show synchronization error increasing significantly with hop number. Ganeriwal et al. attribute this to the magnitude and sign of the random error averaging out over many realizations of the synchronization protocol. The scalability problem is more clearly seen in the standard deviation or variance of the synchronization error, but these metrics were not carefully studied in [8]. Nonetheless, the authors do explain that synchronization error does increase with each hop. Their simulation results in [11] show that synchronization error is a non-decreasing function of hop distance.

Flooding Time Synchronization Protocol (FTSP)

The Flooding Time Synchronization Protocol [10] achieves synchronization with single packet broadcasts. For example, one-hop synchronization can be achieved by having a root node broadcast timing information to surrounding nodes. The timing packets that are broadcast are time stamped using an improved medium access control (MAC) layer time stamping technique. The nodes that receive this sequence of timing packets can use these to estimate both clock skew and clock offset relative to the root node. These synchronized nodes then proceed to broadcast timing packets to nodes beyond the broadcast domain of the root node. This process can continue for larger multi-hop networks. FTSP improves upon TPSN by explicitly estimating clock skew, improving upon the time stamping technique, and removing the need for a hierarchial structure.

FTSP attempts to compensate for or eliminate the send, access, interrupt handling, encoding, decoding, and receive time errors, but does not account for propagation delay. Even with addressing all these sources of error, synchronization error is still seen to grow with the number of hops. Experimental work in [10] finds that the average per-hop synchronization error was in the range of one microsecond.

5.8.2 Analytical Comparison

We have established that cooperative time synchronization using spatial averaging can effectively decrease synchronization error as the network density and number of cooperating nodes are increased. We also know that traditional synchronization techniques are non-cooperative and that generally each timing data point received by a node comes from one neighboring node. Let us now compare cooperative time synchronization to a particular non-cooperative case.

We notice that in increasing node density, we have increased the total power usage in the network since we have added more nodes. As a result, it is not necessarily surprising that better synchronization performance can be obtained. Let us, therefore, briefly consider the synchronization performance of cooperative time synchronization and synchronization without cooperation when total network power remains constant.

In Section 5.4.1 we mentioned that a Type I basic cooperative network is a generalization of a non-cooperative network. Thus, using these two networks, we will compare cooperative time synchronization to a particular case of non-cooperative synchronization. Without cooperation, let us look at $L+1$ nodes in a line starting with node 1 on the left and node $L+1$ on the right (top of Fig. 5.22). The skew and offset estimate variance at node $L+1$ can easily be found using

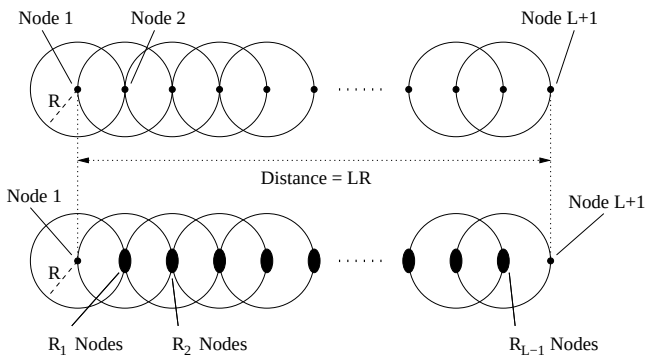

Figure 5.22: The top network is an illustration of the non-cooperative network while the bottom network is a basic cooperative network.

(5.10) and (5.11) with $\bar{N} = 1$ and $k = L$. Node 1 and node $L + 1$ are a distance LR apart and we can compare this to a basic cooperative network where each node i, $i = 2, \ldots, L$ in the no cooperation network is replaced by a set of $\bar{N}$ nodes in the set R_{i-1} (bottom of Fig. 5.22). Therefore, in both cases it takes L hops to send information from node 1 to node $L + 1$, but in the no cooperation case the intermediate hops are single nodes while in the cooperative case the intermediate hops are clusters of nodes.

In the case of cooperation, each intermediate cluster of nodes has $\bar{N}$ nodes transmitting synchronization pulses. As a result, if we assume the power usage of each node is normalized to one, then the power used by each set of nodes R_j, $j = 1, \ldots, L-1$, is $\bar{N}$. To give the non-cooperative network the same total network power, each intermediate node is assigned power $\bar{N}$ instead of 1. This means that nodes 2 through L in the non-cooperative network will now transmit $\bar{N}m$ pulses instead of just m pulses. To keep the time required to synchronize the network from growing with $\bar{N}$, each node will now send a pulses every $d/\bar{N}$ seconds instead of one every d seconds. Note that these changes to the non-cooperative network do not apply to node 1, which still transmits m pulses d seconds apart.

Let us now consider the skew and offset estimate variance of node $L+1$ in the cooperative network and the non-cooperative network. First, for the cooperative network, it is clear that node $L+1$ is actually a node in the set R_L. As a result, its skew and offset estimate variances are found in (5.10) and (5.11), respectively, for $k = L$. For the non-cooperative network, we start with expressions (5.10) and (5.11) with $\bar{N} = 1$ and $k = L$. Then we replace m with $\bar{N}m$ and d with $d/\bar{N}$ to get

$$\begin{aligned} \mathrm{Var}(\hat{\alpha}_L) &= \frac{12\sigma^2}{d^2(m-1)m(m+1)} \\ &+ \frac{12\sigma^2}{\left(\frac{d}{\bar{N}}\right)^2(\bar{N}m-1)\bar{N}m(\bar{N}m+1)}(2(L-1)) \end{aligned} \tag{5.17}$$

$$\begin{aligned} \mathrm{Var}(\hat{\Delta}_L) = &\frac{2\sigma^2(2m-1)}{m(m+1)} \\ &+\sigma^2\Bigg[\frac{4(L-1)(2\bar{N}m-1)}{\bar{N}m(\bar{N}m+1)} \\ &+(L-1)^2\Bigg(-\frac{12}{(\bar{N}m+1)}+\frac{12\bar{N}m}{(\bar{N}m-1)(\bar{N}m+1)}\Bigg) \\ &+\frac{1}{3}(L-2)(L-1)(2L-3)\frac{12\bar{N}m}{(\bar{N}m-1)(\bar{N}m+1)}\Bigg] \end{aligned} \tag{5.18}$$

Note that the first term of $\mathrm{Var}(\hat{\alpha}_L)$ and $\mathrm{Var}(\hat{\Delta}_L)$, the term that is the variance at hop 1, still uses d and m instead of $d/\bar{N}$ and $\bar{N}m$ because node 1 sends m pulses d seconds apart.

From (5.10), we find that the rate of skew variance growth (the slope) for the cooperative case can be simplified to

$$2\frac{12\sigma^2}{d^2m(m^2-1)\bar{N}}. \tag{5.19}$$

Similarly, from (5.17) we find that the rate of skew variance growth for the non-cooperative case simplifies to

$$2\frac{12\sigma^2}{d^2m(\bar{N}m^2-\frac{1}{\bar{N}})}. \tag{5.20}$$

Looking at the ratio of (5.20) to (5.19), we get the ratio of non-cooperative to cooperative error growth rate

$$\eta = \frac{\bar{N}m^2 - \bar{N}}{\bar{N}m^2 - \frac{1}{\bar{N}}} = \frac{m^2 - 1}{m^2 - \frac{1}{\bar{N}^2}}. \tag{5.21}$$

We notice from (5.21) that for $\bar{N} \geq 1$, $\eta \leq 1$, which means that the skew error growth rate for cooperative time synchronization is slightly higher than that of non-cooperative synchronization for the same network power.

For the offset variance, we see that the dominant term in both (5.11) with $k = L$ and (5.18) is the L^3 term. As a result, the dominant offset error term for cooperative synchronization simplifies to

$$\frac{2}{3}\sigma^2 \frac{12m}{\bar{N}(m^2 - 1)}. \tag{5.22}$$

Similarly, the dominant offset error term for non-cooperative synchronization simplifies to

$$\frac{2}{3}\sigma^2 \frac{12m}{\bar{N}m^2 - \frac{1}{\bar{N}}}. \tag{5.23}$$

Comparing, the two terms, we easily see that (5.22) is greater than or equal to (5.23) when $\bar{N} \geq 1$.

From this analysis, we see that cooperative time synchronization performs nearly as well as non-cooperative synchronization when total network power is kept constant. Spatial averaging is only slightly less efficient at averaging out synchronization error than collecting more timing data points. Fig. 5.23 illustrates the skew and offset variance as a function of hop number for cooperation and no cooperation. It is clear that cooperative synchronization has a higher variance than non-cooperative synchronization for hops $k > 1$ when the two networks have the same power. Even though cooperation using spatial averaging yields slightly

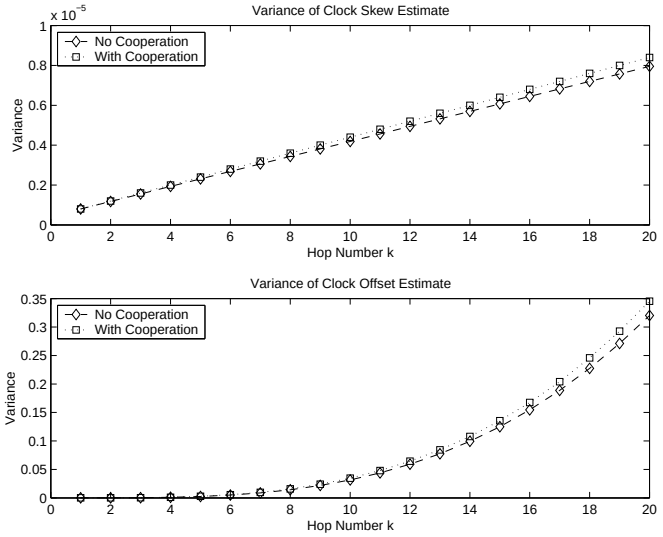

Figure 5.23: Skew and offset variance for cooperative synchronization and non-cooperative synchronization when network power is kept constant. The following parameters were used: $m = 4$, $d = 5$, $\sigma = 0.01$, and $\bar{N} = 4$.

more error accumulation per hop than no cooperation, in the limit as $\bar{N} \to \infty$, both synchronization procedures yield zero error accumulation.

In a more general setup where cooperating nodes may be uniformly distributed over an area and every receiving node hears at least $\bar{N}$ nodes, we can clearly expect cooperative time synchronization to again yield a higher skew and offset variance that non-cooperative synchronization. In fact, the performance gap between cooperative synchronization and non-cooperative synchronization when the cooperative network is uniformly distributed will be larger since the available power is spread out over a larger area when cooperation is used and this additional power can not be used as efficiently to communicate timing information across the network.

Therefore, what we have is that cooperative time synchronization using spatial averaging can be only slightly less efficient than increasing the power in nodes and using non-cooperative time synchronization. Thus, cooperation provides a very

powerful alternative to reducing synchronization error that has two key benefits over non-cooperative techniques. First, since cooperative time synchronization using spatial averaging does not require increasing the power available to the nodes, it provides an alternative to improving synchronization performance when it is not practical or cost effective to build more powerful nodes. Instead of deploying more powerful nodes, network designers can simply deploy a higher density network. Cooperation through spatial averaging therefore provides a fundamentally new trade-off between density and performance. Second, since node transmissions are tightly clustered and there are always only m clusters per hop, as the number of nodes in the network increases, network traffic does not significantly increase. This is different than in the non-cooperative case where the number of timing data points increases. By using an aggregate signal, spatial averaging is able to decouple synchronization performance and the amount of network traffic.

As a final comment, if the signals cooperatively transmitted by the nodes were able to coherently add up in the channel, then for bounded network power, cooperation will improve synchronization performance as node density increases. In such a case, cooperative time synchronization would be superior to no cooperation. This is the case for asymptotic time synchronization in Chapter 3.

5.9 Conclusion

In this chapter we proposed a cooperative time synchronization protocol using spatial averaging and quantified the synchronization performance improvement for networks with finite node density. Due to the complexity of directly analyzing the performance of the protocol for finite density networks, we approached the analysis in two parts. First, we analyzed the synchronization protocol in a Type I

basic cooperative network. Type I networks assume that all $\bar{N}$ nodes in the set R_k are in the broadcast domain of all $\bar{N}$ nodes in R_{k-1}. With the Type I network, we were able to analytically quantify the synchronization performance of the synchronization protocol. Second, we analyzed the synchronization protocol in Type II general networks using simulations and the analytical results developed for Type I networks. We find that increasing node density allows us to decrease synchronization error in general networks and that comparison curves based on Type I results allow us to characterize the synchronization performance improvement.

CHAPTER 6

CONCLUSION

6.1 Forms of Cooperation

It is important to stress the fact that spatial averaging is in reality a very general concept. May different approaches can be taken to take advantage of the properties of spatial averaging.

In Chapter 5, we focus on using the sample mean of a cluster of pulses as a timing data point. This means that the statistic we are employing is the sample mean of the transmission times of neighboring nodes. One benefit of using this particular statistic is that we can analytically quantify the synchronization improvement for a basic cooperative network over many hops. We are able to show that in such a Type I deployment, the variance of the skew and offset estimates decrease approximately as $1/\bar{N}$. These analytical results were then used to understand the synchronization performance in Type II general networks.

However, the sample mean is not the only statistic that can be used to implement the idea of spatial averaging. In Chapter 3 and Chapter 4, timing data points are constructed using a statistic that is closer to a median. Nodes cooperatively transmit odd-shaped pulses with a single zero-crossing in the middle. Each timing data point observed by a receiving node is the zero-crossing of the aggregate waveform. Since the aggregation of the signals is not the same as taking the sample mean of the pulse transmission times of surrounding nodes, the static used for spatial averaging is clearly very different. Due to the complexity of the pulse transmission model, the actual statistic is not exactly the median, but much closer to the median than a sample mean. With such a statistic, we are still able

to illustrate the asymptotic benefits of spatial averaging by showing that all error in the aggregate waveform zero-crossing location can be eliminated. The difficulty with this particular statistic is that an analytically tractable study for networks with a finite number of nodes is extremely difficult.

As a result, from this thesis we see that there can be different approaches to spatial averaging. The advantages can be realized through the use of a true sample mean or a statistic that more closely resembles the median. In fact, any statistic that reduces the error in the data can be used to realize the advantages of spatial averaging. Each chosen statistic may yield different rates of improvement for increasing numbers of cooperating nodes, but spatial averaging improvements will be realized. Hopefully this generality of spatial averaging allows the further development and implementation of practical cooperative time synchronization protocols.

6.2 A New Trade-Off

The fundamental benefit of spatial averaging is that it provides a new trade-off between synchronization performance and node density. Without spatial averaging, synchronization performance could be improved only by collecting more timing data points or improving the data point quality through improved time stamping. However, utilizing spatial averaging allows the density of the network to influence synchronization performance. Using spatial averaging provides a new dimension over which to average out synchronization errors.

This provides much added flexibility in the deployment of sensor networks. As individual sensors become cheaper and are massed produced, large numbers of cheap sensors will become readily available. However, this also means that the

ability to customize the capability of the sensors will also be more limited. As a result, if a network can not be adequately synchronized given the existing resources of the sensors, it may be extremely expensive to provide additional resources to each sensor. However, with the use of spatial averaging, the performance of the network could be improved simply by deploying additional sensors. Thus, cooperative time synchronization through the use of spatial averaging allows network designers to directly benefit from the economies of scale involved in mass producing large numbers of cheap nodes. Spatial averaging allows network synchronization performance to improve with higher node density.

6.3 Future Research Directions

It is important to note that the concept of spatial averaging is very general and the cooperative techniques proposed in this thesis are but a few ways in which to take advantage of it. Our protocols show that techniques using spatial averaging can be designed. Even though the proposed protocols have certain limitations, such as requiring access to the physical layer, they allow us to successfully illustrate the performance improvement achievable using spatial averaging. Future work should focus on other approaches to spatial averaging. For example, it would be desirable to develop a cooperative technique using spatial averaging that achieves performance gains while needing only access to the data link or network layer.

The trade-off introduced by the use of spatial averaging should also be further considered. In this thesis we showed that it is possible to improve synchronization performance by increasing node density. It would be extremely beneficial to be able to improve the performance of other network characteristics simply by increasing node density. Such a *scalable network* would be a network whose performance can

be enhanced by increasing the number of nodes in the network. This would mean that future networks could take advantage of the large numbers of inexpensive nodes that we expect to be readily available in the coming years.

APPENDIX A

LIMIT WAVEFORM PROPERTIES

A.1 Proof of Lemma 1

Here, we prove Lemma 1 in Section 3.4.1. To show (3.5), we consider

$$\begin{aligned} E(A_{max}K_i p(\tau_1 - \tau_0 - T_i)) &= A_{max}E(K_i)E(p(\tau_1 - \tau_0 - T_i)) \\ &= A_{max}E(K_i)\int p(\tau_1 - \tau_0 - \psi)f_{T_i}(\psi)d\psi \\ &= -A_{max}E(K_i)\int p(\psi - (\tau_1 - \tau_0))f_{T_i}(\psi)d\psi \end{aligned}$$

Since $\tau_1 < \tau_0$, we have that $\tau_1 - \tau_0 < 0$ implying that $p(\psi)$ is shifted to the left and the zero-crossing of $p(\psi)$ occurs at a negative value. $p(\psi)$ is odd about its zero-crossing and $f_{T_i}(\psi)$ is symmetric about zero and strictly monotonically increasing on $(-\infty, 0]$ for all positive finite variance values. Thus, it is clear that $\int p(\psi - (\tau_1 - \tau_0))f_{T_i}(\psi)d\psi < 0$ which makes $E(A_{max}K_i p(\tau_1 - \tau_0 - T_i)) > 0$.

Now, the expectation will vary with the variance of T_i and the variance will range from a positive upper bound of $\bar{\sigma}^2/\alpha_{low}^2 < B$ to a positive lower bound of $\bar{\sigma}^2/\alpha_{up}^2$, where recall that $\bar{\sigma}^2$ is a value determined by our choice of the pulse connection function. If we consider $\int p(\psi - (\tau_1 - \tau_0))f_{T_i}(\psi)d\psi$ to be a function of the variance of T_i, then we see that it is bounded and continuous on the compact domain $[\bar{\sigma}^2/\alpha_{up}^2, \bar{\sigma}^2/\alpha_{low}^2]$. Since we showed in the previous paragraph that $E(A_{max}K_i p(\tau_1 - \tau_0 - T_i)) > 0$ whenever T_i has a nonzero finite variance, clearly $E(A_{max}K_i p(\tau_1 - \tau_0 - T_i)) > 0$ when $\mathrm{Var}(T_i) \in [\bar{\sigma}^2/\alpha_{up}^2, \bar{\sigma}^2/\alpha_{low}^2]$. Thus, it is clear that γ_1 and γ_2 exist and (3.5) is shown.

To show (3.6), we consider

$$\begin{aligned}
&\mathrm{Var}(A_{max}K_i p(\tau_1 - \tau_0 - T_i)) \\
&= E(A_{max}^2 K_i^2 p^2(\tau_1 - \tau_0 - T_i)) - E^2(A_{max}K_i p(\tau_1 - \tau_0 - T_i)) \\
&\leq A_{max}^2 E(K_i^2) E(p^2(\tau_1 - \tau_0 - T_i)) \\
&\leq A_{max}^2 E(K_i^2) \\
&\leq A_{max}^2
\end{aligned}$$

where the second to last inequality follows from the fact that $E(p^2(\tau_1 - \tau_0 - T_i))$ is upper bounded by 1. The last inequality follows since $E(K_i^2) \leq 1$ by the fact that $0 \leq K_i \leq 1$. Thus, we have shown (3.6).

Next we define $S_n = \bar{M}_1(\tau_1) + \cdots + \bar{M}_n(\tau_1)$ and $m_n = E(S_n) = \mu_1 + \ldots + \mu_n$. From [46] we have the following theorem

Theorem 4 *The convergence of the series*

$$\sum \frac{\sigma_i^2}{i^2}$$

implies that the strong law of large numbers will apply to the sequence of independent random variables $\bar{M}_i(\tau_1)$. That is, again from [46], for every pair $\epsilon > 0$, $\delta > 0$, there corresponds an N such that

$$Pr\left\{ \frac{|S_n - m_n|}{n} < \epsilon; \quad n = N, N+1, \ldots, N+r \right\} > 1 - \delta$$

for all $r > 0$. $\triangle$

We have shown (3.6) so we have $\sigma_i^2 < \gamma_3 < \infty$. Thus

$$\lim_{N\to\infty} \sum_{i=1}^{N} \frac{\sigma_i^2}{i^2} \leq \lim_{N\to\infty} \sum_{i=1}^{N} \frac{\gamma_3}{i^2} = \gamma_3 \frac{\pi^2}{6}.$$

and we have convergence by the direct comparison test. Therefore, we can apply Theorem 4 and get that for any pair $\epsilon > 0$, $\delta > 0$, we can find an N such that

$$\Pr\left\{\left|\frac{S_n}{n} - \frac{m_n}{n}\right| < \epsilon; \quad n = N, N+1, \ldots, N+r\right\} > 1 - \delta \tag{A.1}$$

for all $r > 0$.

By (3.5) we have that $\gamma_2 > \mu_i > \gamma_1 > 0$. Thus, we can clearly see that

$$\frac{m_n}{n} > \gamma_1.$$

Furthermore, since we keep the function $f_\alpha(s)$ constant as we increase the number of nodes in the network we get that m_n/n converges to a constant $\eta(\tau_1)$ given by

$$\begin{aligned}\eta(\tau_1) &= A_{max}E(K_i)\int_{\alpha_{low}}^{\alpha_{up}}\int_{-\infty}^{\infty} p(\tau_1 - \tau_0 - \psi)f_T(\psi, s)d\psi f_\alpha(s)ds \\ &= \int_{\alpha_{low}}^{\alpha_{up}} E(\bar{M}_i(\tau_1, s))f_\alpha(s)ds.\end{aligned}$$

The above expression comes from the fact that since each $\mu_i = E(\bar{M}_i(\tau_1))$ is a function of α_i, m_n/n will converge to the average of the μ_i over $f_\alpha(s)$, the function that characterizes the set of α_i's. Therefore, given any ϵ, we can find an N' such that

$$\left|\frac{m_n}{n} - \eta(\tau_1)\right| < \epsilon \tag{A.2}$$

for all $n > N'$. Note that since $(m_n/n) > \gamma_1$, we have that $\eta(\tau_1) \geq \gamma_1$. Since

$$\left|\frac{S_n}{n} - \eta(\tau_1)\right| < \left|\frac{S_n}{n} - \frac{m_n}{n}\right| + \left|\frac{m_n}{n} - \eta(\tau_1)\right|,$$

using (A.1) and (A.2) we have

$$\Pr\left\{\left|\frac{S_n}{n} - \eta(\tau_1)\right| < 2\epsilon; \quad n = N'', N''+1, \ldots, N''+r\right\} > 1 - \delta.$$

for all $r > 0$, where $N'' = \max\{N, N'\}$. Thus, we have

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} \bar{M}_i(\tau_1) = \eta(\tau_1) > 0$$

almost surely. This completes the proof of Lemma 1. $\triangle$

A.2 Proof of Lemma 3

Here we prove Lemma 3 in Section 3.4.1. First, we start by finding an analytical expression for $|A_\infty(t) - A_\infty(t_o)|$. From the proof of Lemma 1 we have that

$$A_\infty(t) = A_{max} E(K_i) \int_{\alpha_{low}}^{\alpha_{up}} \int_{-\infty}^{\infty} p(t - \tau_0 - \psi) f_T(\psi, s) d\psi f_\alpha(s) ds.$$

Therefore, $|A_\infty(t) - A_\infty(t_o)|$ can be written as

$$\begin{aligned}
&|A_\infty(t) - A_\infty(t_o)| \\
&= |A_{max} E(K_i) \int_{\alpha_{low}}^{\alpha_{up}} \int_{-\infty}^{\infty} [p(t - \tau_o - \psi) - p(t_o - \tau_o - \psi)] f_T(\psi, s) f_\alpha(s) d\psi ds| \\
&\leq A_{max} \int_{\alpha_{low}}^{\alpha_{up}} \int_{-\infty}^{\infty} |p(t - \tau_o - \psi) - p(t_o - \tau_o - \psi)| f_T(\psi, s) f_\alpha(s) d\psi ds \\
&= A_{max} \int_{\alpha_{low}}^{\alpha_{up}} \int_{-\tau_{nz}+t_o-\tau_0-|t-t_o|}^{\tau_{nz}+t_o-\tau_0+|t-t_o|} |p(t - \tau_o - \psi) - p(t_o - \tau_o - \psi)| \\
&\qquad \times f_T(\psi, s) f_\alpha(s) d\psi ds,
\end{aligned}$$

where $E(K_i) \leq 1$. The change in the limits of integration in the last equality comes from the fact that $p(t - \tau_o - \psi) - p(t_o - \tau_o - \psi) = 0$ outside of $\psi \in [-\tau_{nz} + t_o - \tau_0 - |t - t_o|, \tau_{nz} + t_o - \tau_0 + |t - t_o|]$. This is the maximum interval over which $p(t - \tau_o - \psi) - p(t_o - \tau_o - \psi)$ can be non-zero. There is no need to take the absolute value of $f_T(\psi, s)$ and $f_\alpha(s)$ since they are always non-negative.

Our second step is to bound the inner integral. Before doing so, we first show that the inside integral is in fact Riemann integrable. For any given t and t_o, the inside integral is taken over a closed interval. Over a closed interval, we know from Strichartz [47] that any bounded function that is continuous except at a finite number of points is Riemann integrable. Furthermore, also from [47] we know that the sums and products of continuous functions are continuous. As well, if a function is continuous then the absolute value of that function is also continuous.

$p(t)$ has at most $D = 3$ locations at which it is discontinuous and over any open interval not containing a discontinuity, $p(t)$ in (3.2) is uniformly continuous since $q(t)$ is uniformly continuous. $f_T(\psi, s)$ has $D' = 0$ discontinuities in ψ for a given s since it is Gaussian for any s. And since $s \in [\alpha_{low}, \alpha_{up}]$, $|f_T(\psi, s)| \leq G_T$ for all ψ and s (G_T occurring when $\psi = 0$ and $s = \alpha_{up}$). Thus, since $p(t)$ and $f_T(\psi, s)$ are continuous except at a finite number of points, we see that for given s, t, and t_0

$$|p(t - \tau_o - \psi) - p(t_o - \tau_o - \psi)| f_T(\psi, s)$$

is also continuous in ψ except at a finite number of points (at most $D' + 2D$ points). This function is also bounded since the product of two bounded functions is bounded. As a result, we see that the integral is Riemann integrable over any closed interval.

We now proceed to bound from above the value of this integral by first bounding the maximum value of the integral assuming no discontinuities and then introducing another term that bounds the maximum area contributed by the discontinuities. If we ignore the discontinuities and assume $p(t)$ is uniformly continuous, for any $m_1 > 0$ there exists a $n > 0$ such that

$$|t - t_o| < \frac{1}{n} \Rightarrow |p(t) - p(t_o)| < \frac{1}{m_1},$$

for all t and t_o. As a result, $p(t - \tau_o - \psi) - p(t_o - \tau_o - \psi)$ can be made as small as desired by choosing the proper n thus giving us $p(t-\tau_o-\psi)-p(t_o-\tau_o-\psi) < 1/m_1$ for all ψ for an appropriate choice of n.

Furthermore, we note that $|p(t - \tau_o - \psi)| f_T(\psi, s) \leq G_T$ because $|p(t)| \leq 1$ and $|f_T(\psi, s)| \leq G_T$. The maximum possible jump at a discontinuity in the function $|p(t - \tau_o - \psi) - p(t_o - \tau_o - \psi)| f_T(\psi, s)$ is thus $2G_T$ and for any $|t - t_o|$, the maximum area contributed by each discontinuity is $2G_T|t - t_o|$. As a result, for

all $D' + 2D$ discontinuities, the maximum area contribution will be no more than $2G_T|t - t_o|(D' + 2D)$.

We can, therefore, bound the inner integral as

$$\begin{aligned}
&\int_{-\tau_{nz}+t_o-\tau_0-|t-t_o|}^{\tau_{nz}+t_o-\tau_0+|t-t_o|} |p(t-\tau_o-\psi) - p(t_o-\tau_o-\psi)| f_T(\psi, s) d\psi \\
&\leq \int_{-\tau_{nz}+t_o-\tau_0-|t-t_o|}^{\tau_{nz}+t_o-\tau_0+|t-t_o|} \frac{G_T}{m_1} d\psi + 2G_T|t-t_o|(D'+2D) \\
&= \frac{G_T}{m_1}(2\tau_{nz} + 2|t-t_o|) + 2G_T|t-t_o|(D'+2D) \\
&= 2\frac{G_T}{m_1}\tau_{nz} + 2\frac{G_T}{m_1}|t-t_o| + 2G_T|t-t_o|(D'+2D),
\end{aligned}$$

where $|t - t_o| < 1/n$.

What we have is that if $|t - t_0| < 1/n$ then

$$\begin{aligned}
&|A_\infty(t) - A_\infty(t_o)| \\
&\leq A_{max} \int_{\alpha_{low}}^{\alpha_{up}} \left(2\frac{G_T}{m_1}\tau_{nz} + 2\frac{G_T}{m_1}|t-t_o| + 2G_T|t-t_o|(D'+2D) \right) f_\alpha(s) ds \\
&\leq A_{max} G_\alpha (\alpha_{up} - \alpha_{low}) \left(2\frac{G_T}{m_1}\tau_{nz} + 2\frac{G_T}{m_1}|t-t_o| + 2G_T|t-t_o|(D'+2D) \right)
\end{aligned}$$

since $|f_\alpha(s)| < G_\alpha$ (defined in Section 3.2). We define $\bar{A}$ as

$$\bar{A} = A_{max} G_\alpha (\alpha_{up} - \alpha_{low}).$$

Now, for the third step of our proof we make

$$\begin{aligned}
&|A_\infty(t) - A_\infty(t_o)| \\
&\leq \bar{A} \left(2\frac{G_T}{m_1}\tau_{nz} + 2\frac{G_T}{m_1}|t-t_o| + 2G_T|t-t_o|(D'+2D) \right) \\
&< \frac{1}{m},
\end{aligned}$$

for any choice of $m > 0$. We do this by making each of the three terms less than $1/(3m)$.

For the first term we want

$$\frac{2\bar{A}G_T\tau_{nz}}{m_1} < \frac{1}{3m}.$$

We solve and get

$$m_1 > 6m\bar{A}G_T\tau_{nz}.$$

Since for any value of $m_1 > 0$ we can find an $n > 0$, this condition can be satisfied.

For the third term we want

$$2\bar{A}G_T(D' + 2D)|t - t_o| < \frac{1}{3m}.$$

This gives us

$$|t - t_o| < \frac{1}{6\bar{A}G_T(D' + 2D)m}.$$

Since the only requirement is $|t - t_o| < 1/n$ for n chosen by any given $m_1 > 0$, we can always choose $|t - t_o|$ as small as desired. Thus, this condition can be satisfied.

With the second term we want the condition

$$\frac{2\bar{A}G_T}{m_1}|t - t_o| < \frac{1}{3m}$$

which means that

$$\frac{|t - t_o|}{m_1} < \frac{1}{6m\bar{A}G_T}.$$

Again, this condition can be satisfied since we can choose m_1 as large as we want and $|t - t_o|$ as small as we want as long as $|t - t_o| < 1/n$ for a given m_1.

Thus, for any $m > 0$, we first choose $m_1 > 6m\bar{A}G_T\tau_{nz}$. Then, we find an $n' > 0$ such that $|t - t_o| < 1/n'$ implies that $|p(t) - p(t_o)| < 1/m_1$ for all t and t_o if we remove the discontinuities in $p(t)$. Then, if necessary, n' is increased to n so that $|t - t_o| < 1/n$ implies that $|t - t_o| < 1/(6\bar{A}G_T(D' + 2D)m)$ and $|t - t_o|/m_1 < 1/(6m\bar{A}G_T)$. If no increase is necessary, then $n = n'$. With this

choice of $n > 0$, $|A_\infty(t) - A_\infty(t_o)| < 1/m$. As a result, for any m, we can find an n such that $|t - t_o| < 1/n$ implies that $|A_\infty(t) - A_\infty(t_o)| < 1/m$. Thus, $A_\infty(t)$ is continuous.

This completes the proof for Lemma 3. $\triangle$

A.3 Proof of Theorem 2

Here we prove Theorem 2 in Section 3.6.1. Let us start by writing (3.14) as

$$A_{j,N}^{c_1}(t) \;=\; \sum_{i=1}^{N} \frac{A_{max} K_{fix,i} K_{j,i}}{N} p(t - \tau_o - T_i - D_{fix,i} - D_{j,i}) \;=\; \sum_{i=1}^{N} \frac{1}{N} \tilde{M}_i(t,s),$$

where $\tilde{M}_i(t,s) \stackrel{\Delta}{=} A_{max} K_{fix,i} K_{j,i} p(t - \tau_o - T_i - D_{fix,i} - D_{j,i})$. Recall that the dependence on s comes from the fact that the density of T_i is a function of α_i which is characterized by $f_\alpha(s)$. This notation is analogous to the notation used in Section 3.4.1. Following the steps in the proof of Lemma 1 (Appendix A.1), we can quickly show that the limiting aggregate waveform at node j will take on the form

$$\eta(t) = \int_{\alpha_{low}}^{\alpha_{up}} E(\tilde{M}_i(t,s)) f_\alpha(s) ds, \tag{A.3}$$

where

$$\begin{aligned} &E(\tilde{M}_i(t,s)) \\ &\quad = A_{max} \int_{-\infty}^{\infty} \int_{-\infty}^{0} \int_{0}^{\infty} g(-y) g(x) p(t - \tau_0 - \psi - y - x) \\ &\qquad \times f_{D_j}(x) f_{D_{fix}}(y) f_T(\psi, s) dx dy d\psi, \end{aligned}$$

with $g(\cdot) = K(\delta^{-1}(\cdot))$. Therefore, we can prove Theorem 2 in two steps:

- To show that $\eta(t)$ is odd about τ_0, we need to show that $E(\tilde{M}_i(t,s))$ is odd in t about τ_0, i.e. $E(\tilde{M}_i(\tau_0 + \xi, s)) = -E(\tilde{M}_i(\tau_0 - \xi, s))$ for $\xi \geq 0$.

- To show a zero-crossing at τ_0, show that $E(\tilde{M}_i(\tau_0, s)) = 0$.

These two steps come directly from the form of $\eta(t)$ in (A.3).

We first show that $E(\tilde{M}_i(\tau_0 + \xi, s)) = -E(\tilde{M}_i(\tau_0 - \xi, s))$ for $\xi \geq 0$. Using the fact that $K_{fix} = K(\delta^{-1}(-D_{fix})) = g(-D_{fix})$ and $K_{j,i} = g(D_{j,i})$, we have the following:

$$
\begin{aligned}
& E(\tilde{M}_i(\tau_0 + \xi, s)) \\
&= E\big(A_{max} g(-D_{fix}) g(D_{j,i}) p(\xi - [T_i + D_{fix} + D_{j,i}])\big) \\
&\overset{(a)}{=} -E\big(A_{max} g(-D_{fix}) g(D_{j,i}) p(-\xi + [T_i + D_{fix} + D_{j,i}])\big) \\
&= -A_{max} \int_{-\infty}^{\infty} \int_{-\infty}^{0} \int_{0}^{\infty} g(-y) g(x) p(-\xi + [\psi + y + x]) \\
&\qquad \times f_{D_j}(x) f_{D_{fix}}(y) f_T(\psi, s) dx dy d\psi \\
&\overset{(b)}{=} A_{max} \int_{\infty}^{-\infty} \int_{\infty}^{0} \int_{0}^{-\infty} g(z) g(-u) p(-\xi - [w + z + u]) \\
&\qquad \times f_{D_j}(-u) f_{D_{fix}}(-z) f_T(-w, s) du dz dw \\
&\overset{(c)}{=} -A_{max} \int_{-\infty}^{\infty} \int_{-\infty}^{0} \int_{0}^{\infty} g(-u) g(z) p(-\xi - [w + u + z]) \\
&\qquad \times f_{D_j}(z) f_{D_{fix}}(u) f_T(w, s) dz du dw \\
&= -E\big(A_{max} g(-D_{fix}) g(D_{j,i}) p(-\xi - [T_i + D_{fix} + D_{j,i}])\big) \\
&= -E(\tilde{M}_i(\tau_0 - \xi, s)),
\end{aligned}
$$

where (a) follows because $p(t) = -p(-t)$ and at (b) we did a change of variables with $u = -x$, $w = -\psi$, and $z = -y$. (c) follows from $f_T(x, s) = f_T(-x, s)$ and $f_{D_j}(x) = f_{D_{fix}}(-x)$. We thus have $E(\tilde{M}_i(\tau_0 + \xi, s)) = -E(\tilde{M}_i(\tau_0 - \xi, s))$ for $\xi \geq 0$.

$E(\tilde{M}_i(\tau_0, s)) = 0$ can now be shown as follows. Using the just proven fact that $E(\tilde{M}_i(\tau_0 + \xi, s)) = -E(\tilde{M}_i(\tau_0 - \xi, s))$ for $\xi \geq 0$, setting $\xi = 0$ gives us $E(\tilde{M}_i(\tau_0, s)) = -E(\tilde{M}_i(\tau_0, s))$. This implies that $E(\tilde{M}_i(\tau_0, s)) = 0$.

This completes the proof for Theorem 2. $\triangle$

APPENDIX B

FINITE NODE DENSITY NETWORKS

B.1 Proof of Theorem 3

Node 1 begins the synchronization processes by transmitting a sequence of pulses at times $\tau_0 + ld$, for $l = 0, \ldots, m-1$. For simplicity, assume that τ_0 and d are integer values. Note that since node 1 transmits these pulses in its own time scale c_1 (the reference time), the pulses will occur at integer values of t. Using the clock model in (2.1), any node $1i$, $i = 1, \ldots, \bar{N}$, in the R_1 set of nodes will get a vector of observations $\mathbf{Y}_{1i}$, where $\mathbf{Y}_{1i}[1] = \alpha_{1i}(\tau_0 - \bar{\Delta}_{1i}) + \Psi_{1i,1}$ and the $(l+1)$th element of $\mathbf{Y}_{1i}$ is $\mathbf{Y}_{1i}[l+1] = \alpha_{1i}(\tau_0 - \bar{\Delta}_{1i}) + ld\alpha_{1i} + \Psi_{1i,l+1}$. This can also be written as

$$\mathbf{Y}_{1i} = \mathbf{H}\theta_{1i} + \mathbf{W}_{1i}, \tag{B.1}$$

where

$$\theta_{1i} = \begin{bmatrix} \theta_{1i,1} \\ \theta_{1i,2} \end{bmatrix} = \begin{bmatrix} \alpha_{1i}(\tau_0 - \bar{\Delta}_{1i}) \\ \alpha_{1i} \end{bmatrix}$$

with $\mathbf{H}$ as in (5.3) and $\mathbf{W}_{1i} = [W_{1i,1}, \ldots, W_{1i,m}]^T$. Since $\Psi_{1i,l+1}$ is an independent Gaussian random variable for each l, $\mathbf{W}_{1i} \sim \mathcal{N}(0, \Sigma_{1i})$ with $\Sigma_{1i} = \sigma^2 \mathbf{I}_m$. As mentioned, this set of observations is for any node $1i$ in the set of R_1 nodes.

Since we have $\bar{N}$ R_1 nodes, we can write the vector of observations made by all R_1 nodes as

$$\bar{\mathbf{Y}}_1 = \bar{\mathbf{H}}\bar{\theta}_1 + \bar{\mathbf{W}}_1 \tag{B.2}$$

where

$$\bar{\mathbf{Y}}_1 = \begin{bmatrix} \mathbf{Y}_{11} \\ \mathbf{Y}_{12} \\ \vdots \\ \mathbf{Y}_{1\bar{N}} \end{bmatrix}, \quad \bar{\theta}_1 = \begin{bmatrix} \theta_{11} \\ \theta_{12} \\ \vdots \\ \theta_{1\bar{N}} \end{bmatrix}, \quad \bar{\mathbf{W}}_1 = \begin{bmatrix} \mathbf{W}_{11} \\ \mathbf{W}_{12} \\ \vdots \\ \mathbf{W}_{1\bar{N}} \end{bmatrix}$$

and

$$\bar{\mathbf{H}} = \begin{bmatrix} \mathbf{H} & 0 & \dots & 0 \\ 0 & \mathbf{H} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{H} \end{bmatrix}$$

Note that $\bar{\mathbf{W}}_1 \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_{\bar{N}m})$. This way we have $\bar{\mathbf{Y}}_1$ as the vector of observations made by all R_1 nodes and we can make a UMVU (uniformly minimum variance unbiased) estimate of $\bar{\theta}_1$ by taking

$$\hat{\bar{\theta}}_1 = (\bar{\mathbf{H}}^T \bar{\mathbf{H}})^{-1} \bar{\mathbf{H}}^T \bar{\mathbf{Y}}_1 \sim \mathcal{N}\left(\bar{\mu}_1, \bar{\Sigma}_1\right),$$

where

$$\bar{\mu}_1 = \bar{\theta}_1, \quad \bar{\Sigma}_1 = \sigma^2 (\bar{\mathbf{H}}^T \bar{\mathbf{H}})^{-1}. \tag{B.3}$$

It is easy to see that

$$(\bar{\mathbf{H}}^T \bar{\mathbf{H}})^{-1} = \begin{bmatrix} (\mathbf{H}^T \mathbf{H})^{-1} & 0 & \dots & 0 \\ 0 & (\mathbf{H}^T \mathbf{H})^{-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & (\mathbf{H}^T \mathbf{H})^{-1} \end{bmatrix} \tag{B.4}$$

and

$$(\mathbf{H}^T \mathbf{H})^{-1} = \begin{bmatrix} \frac{2(2m-1)}{m(m+1)} & \frac{-6}{dm(m+1)} \\ \frac{-6}{dm(m+1)} & \frac{12}{d^2(m-1)m(m+1)} \end{bmatrix}. \tag{B.5}$$

This establishes the initial conditions for the theorem. $\hat{\bar{\theta}}_1$ is a $2\bar{N}\times 1$ column vector where the subvector made up of the $(2(i-1)+1)$th and $(2(i-1)+2)$th elements, $i = 1,\ldots,\bar{N}$, is $\hat{\theta}_{1i} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}_{1i}$. Therefore, any node $1i$'s skew estimate (5.1) and offset estimate (5.2) can be found from $\hat{\bar{\theta}}_1$ as

$$\hat{\alpha}_{1i} = e^T_{2(i-1)+2}\hat{\bar{\theta}}_1 \tag{B.6}$$

and

$$\hat{\Delta}_{1i} = e^T_{2(i-1)+1}\hat{\bar{\theta}}_1 - \tau_0 \tag{B.7}$$

where e_l is the column vector of all zeros except for a one in the lth position.

Each node $1i$ can now make an estimate of the next appropriate integer value of t, in this case $t = \tau_0 + md$, by making a minimum variance unbiased estimate of $\theta_{1i,1} + md\theta_{1i,2} = \alpha_{1i}(\tau_0 - \bar{\Delta}_{1i}) + md\alpha_{1i}$. This can be done with the estimator

$$\hat{\tau}_{1i} = \hat{\theta}_{1i,1} + md\hat{\theta}_{1i,2} = \mathbf{C}_0\hat{\theta}_{1i}$$

where $\mathbf{C}_0 = [1 \quad md]$. This will then be node $1i$'s estimate of the next appropriate integer value of t in its own time scale c_{1i}.

From (5.4), every node $1i$ will then transmit a sequence of m pulses occurring, in the time scale of c_{1i}, at $X^{c_{1i}}_{l+1,1i}(\mathbf{Y}_{1i}) = \hat{\tau}_{1i} + ld\hat{\theta}_{1i,2}$, for $l = 0,\ldots,m-1$. Using the clock model (2.1), we find that in the time scale of c_1 these pules occur at

$$\begin{aligned}(\hat{\tau}_{1i} + ld\hat{\theta}_{1i,2})_{c_1} &= \frac{\hat{\tau}_{1i} + ld\hat{\theta}_{1i,2} - \Psi_{1i,l+1}}{\alpha_{1i}} + \bar{\Delta}_{1i} \\ &= \frac{\hat{\tau}_{1i}}{\alpha_{1i}} + \bar{\Delta}_{1i} + ld\frac{\hat{\theta}_{1i,2}}{\alpha_{1i}} - \frac{\Psi_{1i,l+1}}{\alpha_{1i}}.\end{aligned}$$

Any node $2j$ in the R_2 set of nodes that can hear node $1i$ will thus get a sequence of pulses

$$\tilde{\mathbf{Y}}_{2j}[l+1] = \alpha_{2j}\left(\left(\frac{\hat{\tau}_{1i}}{\alpha_{1i}} + \bar{\Delta}_{1i} + ld\frac{\hat{\theta}_{1i,2}}{\alpha_{1i}} - \frac{\Psi_{1i,l+1}}{\alpha_{1i}}\right) - \bar{\Delta}_{2j}\right) + \Psi_{2j,l+1},$$

where $l = 0, \ldots, m-1$.

In this Type I network deployment every node $2j$ hears the same set of $\bar{N}$ nodes and takes the sample mean of each cluster of pulses for its observation, we can express the actual vector of observations made by node $2j$ as

$$\mathbf{Y}_{2j}[l+1] = \sum_{i=1}^{\bar{N}} \frac{\alpha_{2j}}{\bar{N}} \left(\left(\frac{\hat{\tau}_{1i}}{\alpha_{1i}} + \bar{\Delta}_{1i} + ld\frac{\hat{\theta}_{1i,2}}{\alpha_{1i}} - \frac{\Psi_{1i,l+1}}{\alpha_{1i}} \right) - \bar{\Delta}_{2j} \right) + \Psi_{2j,l+1},$$

where $l = 0, \ldots, m-1$. Note that since these pulse arrivals are clustered, we assume that for a given cluster, each pulse arrival is corrupted by the same jitter. Thus, receiver side jitter $\Psi_{2j,l+1}$ is an independent sample for every l, but takes the same value for each i. This models the fact that clock errors occurring in a small time window are highly correlated while errors farther apart in time are independent. We can rewrite this simply as $\mathbf{Y}_{2j}[l+1] = \alpha_{2j}((\tau_1 + ld\tilde{\alpha}_1 - \tilde{\Psi}_{1,l+1}) - \bar{\Delta}_{2j}) + \Psi_{2j,l+1}$, where

$$\tau_1 \triangleq \frac{1}{\bar{N}} \sum_{i=1}^{\bar{N}} \frac{\hat{\tau}_{1i}}{\alpha_{1i}} + \bar{\Delta}_{1i} \qquad \tilde{\alpha}_1 \triangleq \frac{1}{\bar{N}} \sum_{i=1}^{\bar{N}} \frac{\hat{\theta}_{1i,2}}{\alpha_{1i}} \qquad \tilde{\Psi}_{1,l+1} \triangleq \frac{1}{\bar{N}} \sum_{i=1}^{\bar{N}} \frac{\Psi_{1i,l+1}}{\alpha_{1i}}.$$

Since every node $2j$ will see the same $\bar{N}$ nodes, this means that every node $2j$ will have the same τ_1 and $\tilde{\alpha}_1$. Therefore, τ_1 and $\tilde{\alpha}_1$ are now fixed, and it can be easily found that

$$\begin{bmatrix} \tilde{\Psi}_{1,1} \\ \tilde{\Psi}_{1,2} \\ \vdots \\ \tilde{\Psi}_{1,m} \end{bmatrix} \sim \mathcal{N}\left(0, \Sigma_{\tilde{\Psi}_1}\right)$$

where

$$\Sigma_{\tilde{\Psi}_1} = \frac{\sigma^2}{\bar{N}^2} \sum_{i=1}^{\bar{N}} \frac{1}{\alpha_{1i}^2} \mathbf{I}_m.$$

Node $2j$'s vector of observations can also be written in a linear form similar to

(B.1), $\mathbf{Y}_{2j} = \mathbf{H}\theta_{2j} + \mathbf{W}_{2j}$, where

$$\theta_{2j} = \begin{bmatrix} \theta_{2j,1} \\ \theta_{2j,2} \end{bmatrix} = \begin{bmatrix} \alpha_{2j}(\tau_1 - \bar{\Delta}_{2j}) \\ \alpha_{2j}\tilde{\alpha}_1 \end{bmatrix}$$

with $\mathbf{H}$ as in (5.3) and $\mathbf{W}_{2j} = [W_{2j,1} \dots W_{2j,m}]^T$.

$$\mathbf{W}_{2j} = \alpha_{2j} \begin{bmatrix} \tilde{\Psi}_{1,1} \\ \vdots \\ \tilde{\Psi}_{1,m} \end{bmatrix} + \begin{bmatrix} \Psi_{2j,1} \\ \vdots \\ \Psi_{2j,m} \end{bmatrix} \sim \mathcal{N}(0, \Sigma_{2j})$$

with

$$\Sigma_{2j} = \sigma^2\big(1 + \frac{\alpha_{2j}^2}{\bar{N}^2}\sum_{i=1}^{\bar{N}}\frac{1}{\alpha_{1i}^2}\big)\mathbf{I}_m.$$

The vector of observations made by all R_2 nodes can be written in a manner similar to (B.2),

$$\bar{\mathbf{Y}}_2 = \bar{\mathbf{H}}\bar{\theta}_2 + \bar{\mathbf{W}}_2$$

where

$$\bar{\mathbf{Y}}_2 = \begin{bmatrix} \mathbf{Y}_{21} \\ \mathbf{Y}_{22} \\ \vdots \\ \mathbf{Y}_{2\bar{N}} \end{bmatrix}, \quad \bar{\theta}_2 = \begin{bmatrix} \theta_{21} \\ \theta_{22} \\ \vdots \\ \theta_{2\bar{N}} \end{bmatrix}, \quad \mathbf{Q}_2 = \begin{bmatrix} \alpha_{21}\mathbf{I}_m \\ \alpha_{22}\mathbf{I}_m \\ \vdots \\ \alpha_{2\bar{N}}\mathbf{I}_m \end{bmatrix}$$

$$\bar{\mathbf{W}}_2 = \begin{bmatrix} \mathbf{W}_{21} \\ \mathbf{W}_{22} \\ \vdots \\ \mathbf{W}_{2\bar{N}} \end{bmatrix} = \mathbf{Q}_2 \begin{bmatrix} \tilde{\Psi}_{1,1} \\ \vdots \\ \tilde{\Psi}_{1,m} \end{bmatrix} + \begin{bmatrix} \Psi_{21,1} \\ \vdots \\ \Psi_{21,m} \\ \vdots \\ \Psi_{2\bar{N},1} \\ \vdots \\ \Psi_{2\bar{N},m} \end{bmatrix}$$

This means that $\bar{\mathbf{W}}_2 \sim \mathcal{N}(0, \Sigma_{\bar{\mathbf{W}}_2})$, where

$$\Sigma_{\bar{\mathbf{W}}_2} = \mathbf{Q}_2 \Sigma_{\tilde{\Psi}_1} \mathbf{Q}_2^T + \sigma^2 \mathbf{I}_{\bar{N}m}.$$

The R_2 nodes will estimate $\bar{\theta}_2$ as

$$\begin{aligned} \hat{\bar{\theta}}_2 &= (\bar{\mathbf{H}}^T \bar{\mathbf{H}})^{-1} \bar{\mathbf{H}}^T \bar{\mathbf{Y}}_2 \\ &\sim \mathcal{N}\left(\bar{\theta}_2, (\bar{\mathbf{H}}^T \bar{\mathbf{H}})^{-1} \bar{\mathbf{H}}^T \Sigma_{\bar{\mathbf{W}}_2} ((\bar{\mathbf{H}}^T \bar{\mathbf{H}})^{-1} \bar{\mathbf{H}}^T)^T\right). \end{aligned} \tag{B.8}$$

However, for analysis, this does not give us the complete distribution of $\hat{\bar{\theta}}_2$ since $\bar{\theta}_2$ is a function of $\hat{\bar{\theta}}_1$. Therefore, we first consider how θ_{2j} is a function of $\hat{\bar{\theta}}_1$. We find that

$$\begin{aligned} \theta_{2j} &= \begin{bmatrix} \alpha_{2j}(\tau_1 - \bar{\Delta}_{2j}) \\ \alpha_{2j}\tilde{\alpha}_1 \end{bmatrix} \\ &= \begin{bmatrix} \alpha_{2j}(\frac{1}{\bar{N}} \sum_{i=1}^{\bar{N}} \frac{\hat{\tau}_{1i}}{\alpha_{1i}} + \bar{\Delta}_{1i} - \bar{\Delta}_{2j}) \\ \alpha_{2j} \frac{1}{\bar{N}} \sum_{i=1}^{\bar{N}} \frac{\hat{\theta}_{1i,2}}{\alpha_{1i}} \end{bmatrix} \\ &= \begin{bmatrix} \alpha_{2j}(\frac{1}{\bar{N}} \sum_{i=1}^{\bar{N}} \frac{\hat{\theta}_{1i,1} + md\hat{\theta}_{1i,2}}{\alpha_{1i}} + \bar{\Delta}_{1i} - \bar{\Delta}_{2j}) \\ \alpha_{2j} \frac{1}{\bar{N}} \sum_{i=1}^{\bar{N}} \frac{\hat{\theta}_{1i,2}}{\alpha_{1i}} \end{bmatrix} \\ &= \frac{\alpha_{2j}}{\bar{N}} \sum_{i=1}^{\bar{N}} \left(\begin{bmatrix} \frac{1}{\alpha_{1i}} & \frac{dm}{\alpha_{1i}} \\ 0 & \frac{1}{\alpha_{1i}} \end{bmatrix} \begin{bmatrix} \hat{\theta}_{1i,1} \\ \hat{\theta}_{1i,2} \end{bmatrix} + \begin{bmatrix} \bar{\Delta}_{1i} \\ 0 \end{bmatrix} \right) - \alpha_{2j} \begin{bmatrix} \bar{\Delta}_{2j} \\ 0 \end{bmatrix} \end{aligned} \tag{B.9}$$

Using (B.9) we have

$$\bar{\theta}_2 = \mathbf{A}_2 \hat{\bar{\theta}}_1 + \mathbf{B}_2, \tag{B.10}$$

where

$$\mathbf{A}_2 = \mathbf{D}_2 \begin{bmatrix} \frac{1}{\alpha_{11}} & \frac{dm}{\alpha_{11}} & 0 & 0 & \dots & 0 & 0 \\ 0 & \frac{1}{\alpha_{11}} & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \frac{1}{\alpha_{12}} & \frac{dm}{\alpha_{12}} & \dots & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\alpha_{12}} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \frac{1}{\alpha_{1\bar{N}}} & \frac{dm}{\alpha_{1\bar{N}}} \\ 0 & 0 & 0 & 0 & \dots & 0 & \frac{1}{\alpha_{1\bar{N}}} \end{bmatrix},$$

$$\mathbf{B}_2 = \mathbf{D}_2 \begin{bmatrix} \bar{\Delta}_{11} \\ 0 \\ \bar{\Delta}_{12} \\ 0 \\ \vdots \\ \bar{\Delta}_{1\bar{N}} \\ 0 \end{bmatrix} - \begin{bmatrix} \alpha_{21}\bar{\Delta}_{21} \\ 0 \\ \alpha_{22}\bar{\Delta}_{22} \\ 0 \\ \vdots \\ \alpha_{2\bar{N}}\bar{\Delta}_{2\bar{N}} \\ 0 \end{bmatrix},$$

for

$$\mathbf{D}_2 = \frac{1}{\bar{N}} \begin{bmatrix} \alpha_{21} & 0 & \alpha_{21} & 0 & \dots & \alpha_{21} & 0 \\ 0 & \alpha_{21} & 0 & \alpha_{21} & \dots & 0 & \alpha_{21} \\ \alpha_{22} & 0 & \alpha_{22} & 0 & \dots & \alpha_{22} & 0 \\ 0 & \alpha_{22} & 0 & \alpha_{22} & \dots & 0 & \alpha_{22} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha_{2\bar{N}} & 0 & \alpha_{2\bar{N}} & 0 & \dots & \alpha_{2\bar{N}} & 0 \\ 0 & \alpha_{2\bar{N}} & 0 & \alpha_{2\bar{N}} & \dots & 0 & \alpha_{2\bar{N}} \end{bmatrix}.$$

Using (B.8) and (B.10), the distribution of $\hat{\bar{\theta}}_2$ can now be found.

$$\begin{aligned}
\bar{\mu}_2 &= E(\hat{\bar{\theta}}_2) \\
&= E(E(\hat{\bar{\theta}}_2|\hat{\bar{\theta}}_1)) \\
&= E(\bar{\theta}_2) \\
&= E(\mathbf{A}_2\hat{\bar{\theta}}_1 + \mathbf{B}_2) \\
&= \begin{bmatrix} \alpha_{21}(\tau_0 + md - \bar{\Delta}_{21}) \\ \alpha_{21} \\ \alpha_{22}(\tau_0 + md - \bar{\Delta}_{22}) \\ \alpha_{22} \\ \vdots \\ \alpha_{2\bar{N}}(\tau_0 + md - \bar{\Delta}_{2\bar{N}}) \\ \alpha_{2\bar{N}} \end{bmatrix}
\end{aligned} \tag{B.11}$$

Using the decomposition

$$\mathrm{Cov}(\hat{\bar{\theta}}_2) = E(\mathrm{Cov}(\hat{\bar{\theta}}_2|\hat{\bar{\theta}}_1)) + \mathrm{Cov}(E(\hat{\bar{\theta}}_2|\hat{\bar{\theta}}_1)),$$

we have from (B.8) and (B.10)

$$\Sigma_{m_2} = E(\mathrm{Cov}(\hat{\bar{\theta}}_2|\hat{\bar{\theta}}_1)) = (\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T\Sigma_{\bar{\mathbf{W}}_2}((\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T)^T$$

$$\mathrm{Cov}(E(\hat{\bar{\theta}}_2|\hat{\bar{\theta}}_1)) = \mathrm{Cov}(\bar{\theta}_2) = \mathbf{A}_2\bar{\Sigma}_1\mathbf{A}_2^T,$$

giving us

$$\bar{\Sigma}_2 = \mathrm{Cov}(\hat{\bar{\theta}}_2) = \Sigma_{m_2} + \mathbf{A}_2\bar{\Sigma}_1\mathbf{A}_2^T. \tag{B.12}$$

Thus, the distribution of $\hat{\bar{\theta}}_2$ is

$$\hat{\bar{\theta}}_2 \sim \mathcal{N}(\bar{\mu}_2, \bar{\Sigma}_2).$$

$\hat{\bar{\theta}}_2$ is again a $2\bar{N}\times 1$ column vector where the subvector made up of the $(2(i-1)+1)$th and $(2(i-1)+2)$th elements, $i = 1, \ldots, \bar{N}$, is $\hat{\theta}_{2i} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{Y}_{2i}$. Therefore, as in (B.6) and (B.7), any node $2i$'s skew estimate (5.1) and offset estimate (5.2) can be found from $\hat{\bar{\theta}}_2$ as

$$\hat{\alpha}_{2i} = e^T_{2(i-1)+2}\hat{\bar{\theta}}_2 \tag{B.13}$$

and

$$\hat{\Delta}_{2i} = e^T_{2(i-1)+1}\hat{\bar{\theta}}_2 - (\tau_0 + dm). \tag{B.14}$$

Each node $2i$ will now be able to transmit a sequence of m pulses occurring, in the time scale of c_{2i}, at $X^{c_{2i}}_{l+1,2i}(\mathbf{Y}_{2i}) = \hat{\tau}_{2i} + ld\hat{\theta}_{2i,2}$, for $l = 0, \ldots, m-1$, where $\hat{\tau}_{2i} = \hat{\theta}_{2i,1} + md\hat{\theta}_{2i,2}$. Repeating the same process we carried out for the observations of any node $2j$ with any node $3j$, we can find that

$$\hat{\bar{\theta}}_3 \sim \mathcal{N}(\bar{\mu}_3, \bar{\Sigma}_3)$$

where similar to (B.11) we have

$$\begin{aligned}
\bar{\mu}_3 &= E(\hat{\bar{\theta}}_3) \\
&= E(\mathbf{A}_3\hat{\bar{\theta}}_2 + \mathbf{B}_3) \\
&= \begin{bmatrix} \alpha_{31}(\tau_0 + 2md - \bar{\Delta}_{31}) \\ \alpha_{31} \\ \alpha_{32}(\tau_0 + 2md - \bar{\Delta}_{32}) \\ \alpha_{32} \\ \vdots \\ \alpha_{3\bar{N}}(\tau_0 + 2md - \bar{\Delta}_{3\bar{N}}) \\ \alpha_{3\bar{N}} \end{bmatrix}
\end{aligned} \tag{B.15}$$

and similar to (B.12) we have

$$\bar{\Sigma}_3 = \mathrm{Cov}(\hat{\bar{\theta}}_3) = \Sigma_{m_3} + \mathbf{A}_3\bar{\Sigma}_2\mathbf{A}_3^T. \tag{B.16}$$

for

$$\Sigma_{m_3} = E(\mathrm{Cov}(\hat{\bar{\theta}}_3|\hat{\bar{\theta}}_2)) = (\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T\Sigma_{\bar{\mathbf{W}}_3}((\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T)^T$$

$$\mathrm{Cov}(E(\hat{\bar{\theta}}_3|\hat{\bar{\theta}}_2)) = \mathrm{Cov}(\bar{\theta}_3) = \mathbf{A}_3\bar{\Sigma}_2\mathbf{A}_3^T.$$

In this case

$$\Sigma_{\bar{\mathbf{W}}_3} = \mathbf{Q}_3\Sigma_{\tilde{\Psi}_2}\mathbf{Q}_3^T + \sigma^2\mathbf{I}_{\bar{N}m}$$

for

$$\mathbf{Q}_3 = \begin{bmatrix} \alpha_{31}\mathbf{I}_m \\ \alpha_{32}\mathbf{I}_m \\ \vdots \\ \alpha_{3\bar{N}}\mathbf{I}_m \end{bmatrix}$$

$$\Sigma_{\tilde{\Psi}_2} = \frac{\sigma^2}{\bar{N}^2}\sum_{i=1}^{\bar{N}}\frac{1}{\alpha_{2i}^2}\mathbf{I}_m$$

and

$$\mathbf{A}_3 = \mathbf{D}_3 \begin{bmatrix} \frac{1}{\alpha_{21}} & \frac{dm}{\alpha_{21}} & 0 & 0 & \dots & 0 & 0 \\ 0 & \frac{1}{\alpha_{21}} & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \frac{1}{\alpha_{22}} & \frac{dm}{\alpha_{22}} & \dots & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{\alpha_{22}} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \frac{1}{\alpha_{2\bar{N}}} & \frac{dm}{\alpha_{2\bar{N}}} \\ 0 & 0 & 0 & 0 & \dots & 0 & \frac{1}{\alpha_{2\bar{N}}} \end{bmatrix}$$

where

$$\mathbf{D}_3 = \frac{1}{\bar{N}} \begin{bmatrix} \alpha_{31} & 0 & \alpha_{31} & 0 & \dots & \alpha_{31} & 0 \\ 0 & \alpha_{31} & 0 & \alpha_{31} & \dots & 0 & \alpha_{31} \\ \alpha_{32} & 0 & \alpha_{32} & 0 & \dots & \alpha_{32} & 0 \\ 0 & \alpha_{32} & 0 & \alpha_{32} & \dots & 0 & \alpha_{32} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha_{3\bar{N}} & 0 & \alpha_{3\bar{N}} & 0 & \dots & \alpha_{3\bar{N}} & 0 \\ 0 & \alpha_{3\bar{N}} & 0 & \alpha_{3\bar{N}} & \dots & 0 & \alpha_{3\bar{N}} \end{bmatrix}.$$

Continuing this procedure, we can find the distribution of $\hat{\bar{\theta}}_k$ for the R_k nodes as

$$\hat{\bar{\theta}}_k \sim \mathcal{N}(\bar{\mu}_k, \bar{\Sigma}_k)$$

where similar to (B.15) we have

$$\begin{aligned} \bar{\mu}_k &= E(\hat{\bar{\theta}}_k) \\ &= E(\mathbf{A}_k \hat{\bar{\theta}}_{k-1} + \mathbf{B}_k) \\ &= \begin{bmatrix} \alpha_{k1}(\tau_0 + (k-1)md - \bar{\Delta}_{k1}) \\ \alpha_{k1} \\ \alpha_{k2}(\tau_0 + (k-1)md - \bar{\Delta}_{k2}) \\ \alpha_{k2} \\ \vdots \\ \alpha_{k\bar{N}}(\tau_0 + (k-1)md - \bar{\Delta}_{k\bar{N}}) \\ \alpha_{k\bar{N}} \end{bmatrix} \end{aligned}$$

and similar to (B.16) we have

$$\bar{\Sigma}_k = \mathrm{Cov}(\hat{\bar{\theta}}_k) = \Sigma_{m_k} + \mathbf{A}_k \bar{\Sigma}_{k-1} \mathbf{A}_k^T.$$

for

$$\Sigma_{m_k} = E(\mathrm{Cov}(\hat{\bar{\theta}}_k|\hat{\bar{\theta}}_{k-1})) = (\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T\Sigma_{\bar{\mathbf{W}}_k}((\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T)^T$$

$$\mathrm{Cov}(E(\hat{\bar{\theta}}_k|\hat{\bar{\theta}}_{k-1})) = \mathrm{Cov}(\bar{\theta}_k) = \mathbf{A}_k\bar{\Sigma}_{k-1}\mathbf{A}_k^T.$$

In this case

$$\Sigma_{\bar{\mathbf{W}}_k} = \mathbf{Q}_k\Sigma_{\tilde{\Psi}_{k-1}}\mathbf{Q}_k^T + \sigma^2\mathbf{I}_{\bar{N}m} \tag{B.17}$$

for

$$\mathbf{Q}_k = \begin{bmatrix} \alpha_{k1}\mathbf{I}_m \\ \alpha_{k2}\mathbf{I}_m \\ \vdots \\ \alpha_{k\bar{N}}\mathbf{I}_m \end{bmatrix}$$

$$\Sigma_{\tilde{\Psi}_{k-1}} = \frac{\sigma^2}{\bar{N}^2}\sum_{i=1}^{\bar{N}}\frac{1}{\alpha_{(k-1)i}^2}\mathbf{I}_m$$

and

$$\mathbf{A}_k = \mathbf{D}_k \begin{bmatrix} \frac{1}{\alpha_{(k-1)1}} & \frac{dm}{\alpha_{(k-1)1}} & \dots & 0 & 0 \\ 0 & \frac{1}{\alpha_{(k-1)1}} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \frac{1}{\alpha_{(k-1)\bar{N}}} & \frac{dm}{\alpha_{(k-1)\bar{N}}} \\ 0 & 0 & \dots & 0 & \frac{1}{\alpha_{(k-1)\bar{N}}} \end{bmatrix}$$

$$\mathbf{B}_k = \mathbf{D}_k \begin{bmatrix} \bar{\Delta}_{(k-1)1} \\ 0 \\ \bar{\Delta}_{(k-1)2} \\ 0 \\ \vdots \\ \bar{\Delta}_{(k-1)\bar{N}} \\ 0 \end{bmatrix} - \begin{bmatrix} \alpha_{k1}\bar{\Delta}_{k1} \\ 0 \\ \alpha_{k2}\bar{\Delta}_{k2} \\ 0 \\ \vdots \\ \alpha_{k\bar{N}}\bar{\Delta}_{k\bar{N}} \\ 0 \end{bmatrix}.$$

where

$$\mathbf{D}_k = \frac{1}{\bar{N}} \begin{bmatrix} \alpha_{k1} & 0 & \alpha_{k1} & 0 & \dots & \alpha_{k1} & 0 \\ 0 & \alpha_{k1} & 0 & \alpha_{k1} & \dots & 0 & \alpha_{k1} \\ \alpha_{k2} & 0 & \alpha_{k2} & 0 & \dots & \alpha_{k2} & 0 \\ 0 & \alpha_{k2} & 0 & \alpha_{k2} & \dots & 0 & \alpha_{k2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha_{k\bar{N}} & 0 & \alpha_{k\bar{N}} & 0 & \dots & \alpha_{k\bar{N}} & 0 \\ 0 & \alpha_{k\bar{N}} & 0 & \alpha_{k\bar{N}} & \dots & 0 & \alpha_{k\bar{N}} \end{bmatrix}$$

As in (B.13) and (B.14), any node ki's skew estimate (5.1) and offset estimate (5.2) can be found from $\hat{\bar{\theta}}_k$ as

$$\hat{\alpha}_{ki} = e^T_{2(i-1)+2}\hat{\bar{\theta}}_k$$

and

$$\hat{\Delta}_{ki} = e^T_{2(i-1)+1}\hat{\bar{\theta}}_k - (\tau_0 + dm(k-1)).$$

This concludes the proof of Theorem 3. $\triangle$

B.2 Proof of Corollary 1

First, let us consider the mean of node ki's skew and offset estimates. Under the assumption of $\alpha_i = 1$ for all i, from (5.5), (5.7), and (5.8) we have that

$$E(\hat{\alpha}_{ki}) = e_{2(i-1)+2}^T E(\hat{\bar{\theta}}_k) = \alpha_{ki} = 1$$

$$E(\hat{\Delta}_{ki}) = e_{2(i-1)+1}^T E(\hat{\bar{\theta}}_k) - (\tau_0 + dm(k-1)) = -\bar{\Delta}_{ki}.$$

Using (B.4) and (B.5), $\bar{\Sigma}_1$ in (B.3) is known. Using $\alpha_i = 1$ for all i, from (B.17) we have that $\Sigma_{\bar{\mathbf{W}}_k} = \Sigma_{\bar{\mathbf{W}}}$ becomes the $\bar{N}m \times \bar{N}m$ matrix

$$\Sigma_{\bar{\mathbf{W}}} = \sigma^2 \begin{bmatrix} (1+\frac{1}{\bar{N}})\mathbf{I}_m & \frac{1}{\bar{N}}\mathbf{I}_m & \dots & \frac{1}{\bar{N}}\mathbf{I}_m \\ \frac{1}{\bar{N}}\mathbf{I}_m & (1+\frac{1}{\bar{N}})\mathbf{I}_m & \dots & \frac{1}{\bar{N}}\mathbf{I}_m \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\bar{N}}\mathbf{I}_m & \frac{1}{\bar{N}}\mathbf{I}_m & \dots & (1+\frac{1}{\bar{N}})\mathbf{I}_m \end{bmatrix}, \tag{B.18}$$

which gives $\Sigma_m = (\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T\Sigma_{\bar{\mathbf{W}}}((\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T)^T$. It can also be easily found that $\mathbf{A}^l$ is the $2\bar{N} \times 2\bar{N}$ matrix

$$\mathbf{A}^l = \frac{1}{\bar{N}} \begin{bmatrix} 1 & lmd & 1 & lmd & \dots & 1 & lmd \\ 0 & 1 & 0 & 1 & \dots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & lmd & 1 & lmd & \dots & 1 & lmd \\ 0 & 1 & 0 & 1 & \dots & 0 & 1 \end{bmatrix} \tag{B.19}$$

for integer values of $l \geq 1$ and we use $\mathbf{A}^0 = \mathbf{I}_{2\bar{N}}$.

Using (B.18), we find Σ_m to be the $2\bar{N} \times 2\bar{N}$ matrix

$$
\begin{aligned}
&(\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T\Sigma_{\bar{\mathbf{W}}}((\bar{\mathbf{H}}^T\bar{\mathbf{H}})^{-1}\bar{\mathbf{H}}^T)^T \\
&\quad = \sigma^2 \begin{bmatrix} a(\mathbf{H}^T\mathbf{H})^{-1} & b(\mathbf{H}^T\mathbf{H})^{-1} & \dots & b(\mathbf{H}^T\mathbf{H})^{-1} \\ b(\mathbf{H}^T\mathbf{H})^{-1} & a(\mathbf{H}^T\mathbf{H})^{-1} & \dots & b(\mathbf{H}^T\mathbf{H})^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ b(\mathbf{H}^T\mathbf{H})^{-1} & b(\mathbf{H}^T\mathbf{H})^{-1} & \dots & a(\mathbf{H}^T\mathbf{H})^{-1} \end{bmatrix} \\
&\quad = \Sigma_m
\end{aligned}
$$

where

$$a = 1 + \frac{1}{\bar{N}} \tag{B.20}$$

$$b = \frac{1}{\bar{N}} \tag{B.21}$$

Letting $(\mathbf{H}^T\mathbf{H})^{-1}$ from (B.5) be

$$(\mathbf{H}^T\mathbf{H})^{-1} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}$$

and using (B.19), we can find the (1,1) element and (2,2) element of $\mathbf{A}^l\Sigma_m(\mathbf{A}^l)^T$ to be

$$
\begin{aligned}
&\mathbf{A}^l\Sigma_m(\mathbf{A}^l)^T(1,1) \\
&\quad = \frac{\sigma^2}{\bar{N}}\left[\left(a + (\bar{N}-1)b\right)\left(s_{11} + lmd(s_{12}+s_{21}) + (lmd)^2 s_{22}\right)\right] \qquad \text{(B.22)} \\
&\mathbf{A}^l\Sigma_m(\mathbf{A}^l)^T(2,2) \\
&\quad = \frac{\sigma^2}{\bar{N}}\left[\left(a + (\bar{N}-1)b\right)s_{22}\right] \qquad \text{(B.23)}
\end{aligned}
$$

for $l \geq 1$. Likewise, we can find the (1,1) element and (2,2) element of $\mathbf{A}^l\bar{\Sigma}_1(\mathbf{A}^l)^T$ to be

$$\mathbf{A}^l\bar{\Sigma}_1(\mathbf{A}^l)^T(1,1) = \frac{\sigma^2}{\bar{N}}\left[s_{11} + lmd(s_{12}+s_{21}) + (lmd)^2 s_{22}\right] \tag{B.24}$$

$$\mathbf{A}^l\bar{\Sigma}_1(\mathbf{A}^l)^T(2,2) = \frac{\sigma^2}{\bar{N}}s_{22} \tag{B.25}$$

for $l \geq 1$, where $\bar{\Sigma}_1$ is from (B.3).

Therefore, using (B.20) and (B.21) with (B.22) and (B.24) gives us element (1,1) of $\bar{\Sigma}_k$ from (5.9) to be

$$\begin{aligned}\bar{\Sigma}_k(1,1) &= \sigma^2 s_{11} + \frac{\sigma^2}{\bar{N}}\Bigg[2(k-1)s_{11} \\ &\quad +(k-1)^2\Big(md(s_{12}+s_{21}) + (md)^2 s_{22}\Big) \\ &\quad +\frac{1}{3}(k-2)(k-1)(2k-3)(md)^2 s_{22}\Bigg] \end{aligned} \tag{B.26}$$

Similarly, using (B.20) and (B.21) with (B.23) and (B.25) gives us element (2,2) of $\bar{\Sigma}_k$ from (5.9) to be

$$\bar{\Sigma}_k(2,2) = \sigma^2 s_{22}\left(1 + \frac{2(k-1)}{\bar{N}}\right) \tag{B.27}$$

Simplifying (B.27) and (B.26) yields equations (5.10) and (5.11), respectively. This concludes the proof of Corollary 1. $\triangle$

REFERENCES

[1] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Comm. ACM*, 21(4):558–565, 1978.

[2] J. Elson and K. Romer. Wireless Sensor Networks: A New Regime for Time Synchronization. In *Proc. of the First Workshop on Hot Topics in Networks*, Princeton, NJ, October 2002.

[3] C. Kelly IV, V. Ekanayake, and R. Manohar. SNAP: A Sensor Network Asynchronous Processor. In *Proc. 9th Int. Symp. Async. Circ. Syst.*, Vancouver, BC, 2003.

[4] B. Warneke, M. Last, B. Liebowitz, and K. S. J. Pister. Smart Dust: Communicating with a Cubic-Millimeter Computer. *IEEE Computer Mag.*, 34(1):44–51, 2001.

[5] H. Li, A. Lal, J. Blanchard, and D. Henderson. Self-Reciprocating Radioisotope-Powered Cantilever. *J. Applied Phys.*, 92(2):1122–1127, 2002.

[6] J. Elson, L. Girod, and D. Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. In *Proc. 5th Symp. Op. Syst. Design Implementation (OSDI)*, Boston, MA, 2002.

[7] M. L. Sichitiu and C. Veerarittiphan. Simple, Accurate Time Synchronization for Wireless Sensor Networks. In *Proc. IEEE Wireless Communication and Networking Conference (WCNC 2003)*, New Orleans, LA, March 2003.

[8] S. Ganeriwal, R. Kumar and M. B. Srivastava. Timing-Sync Protocol for Sensor Networks. In *Proc. First ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, CA, November 2003.

[9] J. van Greunen and J. Rabaey. Lightweight Time Synchronization for Sensor Networks. In *Proc. 2nd ACM International Workshop on Wireless Sensor Networks and Applications (WSNA 2003)*, San Diego, CA, September 2003.

[10] M. Maroti, B. Kusy, G. Simon and A. Ledeczi. The Flooding Time Synchronization Protocol. In *Proc. 2nd International Conference on Embedded Networked Sensor Systems*, Baltimore, MD, November 2004.

[11] S. Ganeriwal, R. Kumar, S. Adlakha and M. B. Srivastava. Network-wide Time Synchronization in Sensor Networks. NESL Technical Report, 2003.

[12] R. Mathar and J. Mattfeldt. Pulse-Coupled Decentral Synchronization. *SIAM Journal on Applied Mathematics*, 56(4):1094-1106, 1996.

[13] C. Vanvreeswijk and L. F. Abbott. Self-Sustained Firing in Populations of Integrate-and-Fire Neurons. *SIAM Journal on Applied Mathematics*, 53(1):253-264, 1993.

[14] C. Chen. Threshold Effects on Synchronization of Pulse-Coupled Oscillators. *Physical Review E*, 49(4):2668-2672, 1994.

[15] A. Corral, C. J. Pérez, A. Díaz-Guilera and A. Arenas. Self-Organized Criticality and Synchronization in a Lattice Model of Integrate-and-Fire Oscillators. *Physical Review Letters*, 74(1):118-121, 1995.

[16] A. Díaz-Guilera, C. J. Pérez and A. Arenas. Mechanism of Synchronization and Pattern Formation in a Lattice of Pulse-Coupled Oscillators. *Physical Review E*, 57(4):3820-3828, 1998.

[17] U. Ernst, K. Pawelzik and T. Geisel. Delay-Induced Multistable Synchronization of Biological Oscillators. *Physical Review E*, 57(2):2150-2162, 1998.

[18] W. Gerstner. Rapid Phase Locking in Systems of Pulse-Coupled Oscillators with Delays. *Physical Review Letters*, 76(10):1755-1758, 1996.

[19] X. Guardiola, A. Díaz-Guilera, M. Llas and C. J. Pérez. Synchronization, Diversity, and Topology of Networks of Integrate and Fire Oscillators. *Physical Review E*, 62(4):5565-5570, 2000.

[20] A. Herz and J. J. Hopfield. Earthquake Cycles and Neural Reverberations: Collective Oscillations in Systems with Pulse-Coupled Threshold Elements. *Physical Review Letters*, 75(6):1222-1225, 1995.

[21] E. M. Izhikevich. Weakly Pulse-Coupled Oscillators, FM Interactions, Synchronization, and Oscillatory Associative Memory. *IEEE Trans. Neural Networks*, 10(3):508-526, 1999.

[22] L. S. Smith, D. E. Cairns and A. Nschwitz. Synchronization of Integrate-and-Fire Neurons with Delayed Inhibitory Lateral Connections. In *Proc. International Conference on Artificial Neural Networks (ICANN)*, 1994.

[23] J. Buck and E. Buck. Synchronous Fireflies. *Scientific American*, 234:74-85, 1976.

[24] J. Jalife. Mutual Entrainment and Electrical Coupling as Mechanisms for Synchronous Firing of Rabbit Sinoatrial Pacemaker Cells. *J. Physiol.*, 356:221-243, 1984.

[25] A. Sherman, J. Rinzel and J. Keizer. Emergence of Organized Bursting in Clusters of Pancreatic Beta-Cells by Channel Sharing. *Biophys. J.*, 54:411-425, 1988.

[26] R. E. Mirollo and S. H. Strogatz. Synchronization of Pulse-Coupled Biological Oscillators. *SIAM J. Appl. Math.*, 50(6):1645–1662, 1990.

[27] S. Strogatz. *Sync: The Emerging Science of Spontaneous Order*. Theia, 2003.

[28] M. K. McClintock Menstrual Synchrony and Suppression. *Nature*, 229:244-245, 1971.

[29] T. J. Walker. Acoustic Synchrony: Two Mechanisms in the Snowy Tree Cricket. *Science*, 166:891-894, 1969.

[30] D. Lucarelli and I. Wang. Decentralized Synchronization Protocols with Nearest Neighbor Communication. In *Proc. SenSys'04*, Baltimore, Maryland, 2004.

[31] Y. Kuramoto. Collective Synchronization of Pulse-Coupled Oscillators and Excitable Units. *Physica D*, 50:15-30, 1991.

[32] W. Senn and R. Urbanczik. Similar Non-Leaky Integrate-and-Fire Neurons with Instantaneous Coupling Always Synchronize. *SIAM J. Appl. Math.*, 61(4):1143–1155, 2000.

[33] Y. Hong and A. Scaglione. A Scalable Synchronization Protocol for Large Scale Sensor Networks and its Applications. *IEEE Journal on Selected Areas in Communications (JSAC)*, 23(5):1085-1099, May 2005.

[34] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal. Firefly-Inspired Sensor Network Synchronicity with Realistic Radio Effects. In *Proc. SenSys'05*, San Diego, CA, November 2005.

[35] B. Barriac, R. Mudumbai and U. Madhow. Distributed Beamforming for Information Transfer in Sensor Networks. In *Proc. International Symposium on Information Processing in Sensor Networks (IPSN)*, Berkeley, CA, 2004.

[36] H. Ochiai, P.Mitran, H. V. Poor and V. Tarokh. Collaborative Beamforming for Distributed Wireless Ad Hoc Sensor Networks. *IEEE Transactions on Signal Processing*, 53(11):4110-4124, 2005.

[37] A. Hu and S. D. Servetto. dFSK: Distributed Frequency Shift Keying Modulation in Dense Sensor Networks. In *Proc. IEEE Int. Conf. Communications (ICC)*, Paris, France, 2004.

[38] A. Scaglione and Y. W. Hong. Opportunistic Large Arrays: Cooperative Transmission in Wireless Multihop Ad Hoc Networks to Reach Far Distances. *IEEE Transactions on Signal Processing*, 51(8):2082-2092, August 2003.

[39] I. Maric and R. D. Yates. Cooperative Multihop Broadcast for Wireless Networks. *IEEE J. Selected Areas in Communications*, 22(6):1080-1088, August 2004.

[40] B. Sirkeci-Mergen, A. Scaglione and G. Mergen. Asymptotic Analysis of Multistage Cooperative Broadcast in Wireless Networks. *IEEE Transactions on Information Theory*, 52(6):2531-2550, June 2006.

[41] N. Roberts. Phase Noise and Jitter: A Primer for Digital Designers. http://www.eedesign.com/showArticle.jhtml?articleID=16501598, 2003.

[42] H. Stark and J. Woods. *Probability, Random Processes, and Estimation Theory for Engineers*. Prentice Hall, Inc., 2nd edition, 1994.

[43] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin and S. Wicker. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks. In *Technical Report UCLA/CSD-TR 02-0013*, Computer Science Department, UCLA, July 2002.

[44] J. Zhao and R. Govindan. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *Proc. 1st International Conf. on Embedded Networked Sensor Systems*, Los Angeles, CA, 2003.

[45] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. PTR Prentice Hall, Inc., 1993.

[46] W. Feller. *An Introduction to Probability Theory and its Applications*. John Wiley & Sons, Inc., 1968.

[47] R. S. Strichartz. *The Way of Analysis*. Jones and Bartlett Publishers, 2000.

[48] E. W. Weisstein. Circular Segment. From *MathWorld*–A Wolfram Web Resource. http://mathworld.wolfram.com/CircularSegment.html