ABSTRACT

DISTRIBUTED SYNCHRONIZATION ALGORITHMS FOR WIRELESS SENSOR NETWORKS

by
Nicola Varanese

The ability to distribute time and frequency among a large population of interacting agents is of interest for diverse disciplines, inasmuch as it enables to carry out complex cooperative tasks. In a wireless sensor network (WSN), time/frequency synchronization allows the implementation of distributed signal processing and coding techniques, and the realization of coordinated access to the shared wireless medium. Large multi-hop WSN's constitute a new regime for network synchronization, as they call for the development of scalable, fully distributed synchronization algorithms. While most of previous research focused on synchronization at the application layer, this thesis considers synchronization at the lowest layers of the communication protocol stack of a WSN, namely the physical and the medium access control (MAC) layer. At the physical layer, the focus is on the compensation of carrier frequency offsets (CFO), while time synchronization is studied for application at the MAC layer. In both cases, the problem of realizing network-wide synchronization is approached by employing distributed clock control algorithms based on the classical concept of coupled phase and frequency locked loops (PLL and FLL). The analysis takes into account communication, signaling and energy consumption constraints arising in the novel context of multi-hop WSN's. In particular, the robustness of the algorithms is checked against packet collision events, infrequent sync updates, and errors introduced by different noise sources, such as transmission delays and clock frequency instabilities. By observing that WSN's allow for greater flexibility in the design

of the synchronization network architecture, this work examines also the relative merits of both peer-to-peer (mutually coupled - MC) and hierarchical (master-slave - MS) architectures. With both MC and MS architectures, synchronization accuracy degrades smoothly with the network size, provided that loop parameters are conveniently chosen. In particular, MS topologies guarantee faster synchronization, but they are hindered by higher noise accumulation, while MC topologies allow for an almost uniform error distribution at the price of much slower convergence. For all the considered cases, synchronization algorithms based on adaptive PLL and FLL designs are shown to provide robust and scalable network-wide time and frequency distribution in a WSN.

DISTRIBUTED SYNCHRONIZATION ALGORITHMS FOR WIRELESS SENSOR NETWORKS

by
Nicola Varanese

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Department of Electrical and Computer Engineering

January 2011

.

APPROVAL PAGE

DISTRIBUTED SYNCHRONIZATION ALGORITHMS FOR WIRELESS SENSOR NETWORKS

Nicola Varanese

Dr. Yeheskel Bar-Ness, Dissertation Co-advisor — Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. Umberto Spagnolini, Dissertation Co-advisor — Date
Full Professor of Dipartimento di Elettronica e Informazione, Politecnico di Milano

---

Dr. Luca Schenato, Committee Member — Date
Associate Professor of Department of Information Engineering, University of Padova

---

Dr. Alexander M. Haimovich, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Osvaldo Simeone, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Nicola Varanese

Degree: Doctor of Philosophy

Date: January 2011

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2011
- Doctor of Philosophy in Information Engineering,
  Politecnico di Milano, Milano, Italy, 2011
- Laurea Specialistica (M.Sc.) in Telecommunication Engineering,
  Politecnico di Milano, Milano, Italy, 2006
- Laurea (B.Sc.) in Telecommunication Engineering,
  Politecnico di Milano, Milano, Italy, 2003

Major: Electrical Engineering

Publications:

N. Varanese, U. Spagnolini, Y. Bar-Ness, "Synchronization tracking in wireless sensor networks with low duty cycles," in preparation for submission to IEEE Trans. on Communications.

N. Varanese, U. Spagnolini, Y. Bar-Ness, "On the accuracy of distributed synchronization algorithms for wireless networks," in preparation for submission to IEEE Trans. on Signal Processing.

N. Varanese, U. Spagnolini, Y. Bar-Ness, "Distributed frequency locked loops for wireless networks," submitted to IEEE Trans. on Communications (second revision round).

N. Varanese, Y. Bar-Ness, U. Spagnolini, “On the synchronization rate of distributed medium access protocols,” in proc. Conference on Information Sciences and Systems (CISS), Princeton, NJ USA, Mar. 17-19 2010.

U. Spagnolini, N .Varanese, O. Simeone, Y. Bar-Ness, “Distributed digital locked loops (D-DLL) for time/frequency locking in packet communications,” in proc. International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Cannes, France, Sept. 15-18 2008 (invited paper).

N. Varanese, O. Simeone, U. Spagnolini, Y. Bar-Ness, “Distributed frequency-locked loops for wireless networks,” in proc. International Symposium on Spread Spectrum Techniques and Applications (ISSSTA), Bologna, Italy, Aug. 25-28 2008.

N. Varanese, O. Simeone, Y. Bar-Ness, U. Spagnolini, “Achievable rates of multi-hop and cooperative MIMO Amplify-and-Forward relay systems with full CSI,” in proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Cannes, France, July 2-5 2006.

N. Varanese, O. Simeone, Y. Bar-Ness, U. Spagnolini, “A power allocation strategy for multi-antenna amplify-and-forward fading relay channels,” in proc. Conference on Information Sciences and Systems (CISS), Princeton, NJ USA, Mar. 22-24 2006.

To my parents, Vincenzo and Angela.

ACKNOWLEDGMENT

The opportunity to work within the double-degree program between NJIT and Politecnico di Milano made my doctoral studies an extremely enriching experience. I have to thank my advisors, Prof. Yeheskel Bar-Ness of NJIT and Prof. Umberto Spagnolini of Politecnico, for devising this special program and choosing me as the first student to be enrolled in it. Also, I would like to thank Prof. Bar-Ness for his wise guidance and deep research experience, which have provided this work with rigor and structure. Prof. Spagnolini nurtured me since undergraduate courses, when his engaging way of teaching motivated me to pursue higher goals. Prof. Spagnolini is definitely a research enthusiast, and he taught me how a novel problem should be approached and solved in a simple yet effective way.

I would like to acknowledge the members of my dissertation committee, Prof. Alexander Haimovich and Prof. Osvaldo Simeone of NJIT, and Prof. Luca Schenato of University of Padova. Their insightful comments greatly helped to improve the quality of this thesis. In particular, I would like to thank Prof. Simeone for introducing me to research during my M.Sc. thesis and for his generous advice throughout my doctoral studies.

Funding of my dissertation was provided by the Italian Ministry of Education, University and Research (MIUR) within the Scuola Interpolitecnica di Dottorato program, the Ross Fellowship Memorial Fund, and the Elisha Yegal Bar-Ness Fellowship.

The indispensable help of many from the administrative bodies of NJIT and Politecnico has been decisive for the success of the double-degree program. I would like to thank Dr. Ronald Kane, Ms. Clarisa Gonzalez-Lenahan, Mr. Jeffrey Grundy, Mr. Scott Kline, Dr. Graziella Lo Schiavo, Ms. Nadia Prada, Dr. Chiara Lauritano, and Dr. Danila Ferrara. A special gratitude is reserved for Ms. Marlene Toeroek, for her versatile and wholehearted help during all my stay at NJIT.

My academic days have been enriched by the company of many colleagues at the Center for Wireless Communication and Signal Processing Research (CWCSPR) of NJIT and at the Dipartimento di Elettronica e Informazione (DEI) of Politecnico. Among my fondest memories I can find many debates around research and our respective cultures, which have made my doctoral studies a unique experience. For the special friendship and support they provided me throughout these years, I would like to mention Jonathan, Marco, Daniele, Igor, Ciprian, Vlad, Amir, Riccardo and Simone.

Finally, I am infinitely indebted to my parents, Vincenzo and Angela, to whom I dedicate this work. They taught me with their example that any task is accomplished by employing the right measure of perseverance.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

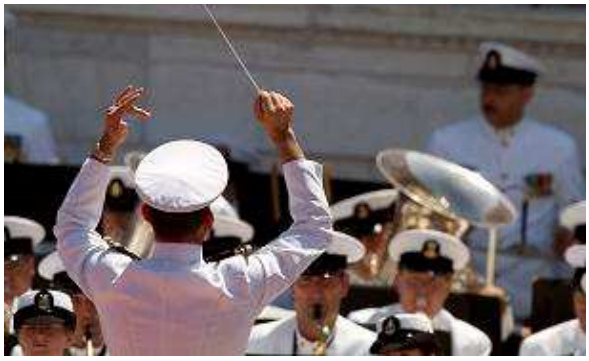
Figure **Page**

CHAPTER 1

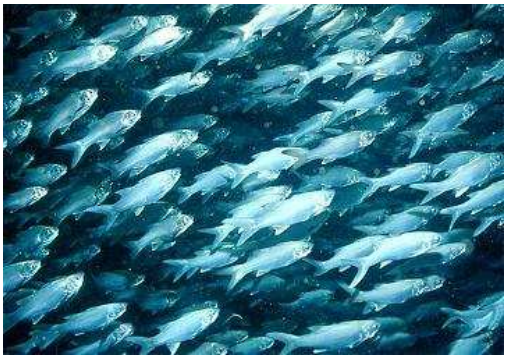
INTRODUCTION

Synchronization is the onset of *coordinated* periodic events in a population of interacting agents. Indeed, synchronization is the simplest form of coordination, and its function is to enable more complex cooperative actions.

Synchronization is part of everyday life, where the common notion of time constitutes one of the pillars of social interaction. It actually goes from the quartz clock on our wrist deep into the innermost workings of the human body. In fact, each living being possesses a number of internal clocks, the circadian rhythms, which regulate all periodic activities, from sleep/wake cycles to the secretion of specific hormones. In humans, the central ("*master*") clock resides in the cells of the suprachiasmatic nucleus (SCN) in the hypothalamus, and peripheral clocks exist in many organs such as the esophagus, lungs, liver, pancreas. Circadian rhythms are entrained by external reference signals (or zeitbergs), the most important of which is daylight. Daylight influences directly the master clock in the SCN via the signals traveling from the eyes through the retinohypothalamic tract. An animal kept in total darkness eventually assumes a free-running activity period. As an example, experiments have determined that the human free-running sleep/wake cycle amounts to around 25 hours. While looking for a treatment for sleep disorders, researchers have studied the *phase response curve* (PRC) of the human sleep/wake rhythm to artificial light stimuli. Another example of master-slave synchronization is found in musical ensembles, where independent performers need a common time and rhythm (frequency) in order to play the score in sync. The conductor in Figure 1.1.a distributes time and frequency to a large orchestra, while the drummer of a rock band is often referred to as the "clock distributor".



a)



b)

Figure 1.1 Examples of synchronization phenomena in nature and everyday life: a) the conductor of an orchestra and b) a school of fish.

Synchronization is not always induced in a master-slave fashion, but it can happen as the outcome of spontaneous collective interaction. In nature, many examples of coordinated dynamics are found, such as the movement of the school of fish in Figure 1.1.b. Fish is spurred to cooperation by the need to escape predators, and it behaves so without any leader or external forcing mechanism. Focusing on synchronization, it arises spontaneously in a variety of circumstances. In the human body, pacemaker cells trigger contractions of the heart tissues in unison. Neurons may fall in step with one another in certain diseases. Crickets distributed over a wide area may start to chirp simultaneously. Norbert Wiener in his landmark book "Cybernetics" [1] described his fascination for self-synchronization arising in populations of flashing fireflies in South-East Asia. A mathematical explanation for these phenomena has not been available until the seminal work of Winfree [2], who first posed the problem as a system of *mutually coupled* limit-cycle oscillators. Mutual coupling implies that there exist some mean (chemical, electrical, visual, acoustic) for each oscillator to influence its neighbors and vice-versa, so that they cooperatively converge to a synchronous regime. After Winfree, many researchers investigated diverse autonomous synchronization phenomena ranging from neural networks to complex social interaction.

Finally, synchronization and timekeeping have always been fundamental facilities in engineering systems. Over the centuries, timekeeping evolved from sundials to the

pendulum clock invented by Christiaan Huygens in 1657 (following an idea of Galileo Galilei). The pendulum clock was the first clock to be based on a *harmonic oscillator*. Clock accuracy improved enormously in the Twentieth century by the invention of the crystal oscillator, patented by Alexander M. Nicholson of Bell Labs in 1918, and the development of atomic clocks in the 1950's at the National Physical Laboratory in the UK. The ability of artificial clocks to interact with each other was first discovered by Huygens himself, who observed that two pendulum clocks hung on a common beam oscillate with the same frequency and 180 degrees out of phase. Actually, a striking similarity connects artificial and biological synchronization systems: every time and frequency distribution system in operation is based on the familiar notions of master-slave and mutual coupling interactions already found in natural phenomena. Also, when designing phase and frequency recovery circuits, the PRC (or S-curve, in engineering terminology [3]) is a fundamental design parameter. Synchronization in communication systems will be detailed later on, after a brief introduction to synchronization regimes and oscillator models.

1.1 Synchronization in a Nutshell

In order to introduce basic notions and definitions about synchronization regimes, consider a a simple harmonic oscillator with output

$$x(t) = A(t) \sin\left(2\pi f t + \phi\right), \tag{1.1}$$

where t is absolute time, f is the oscillator frequency and ϕ is the oscillator phase at $t = 0$ in radians. The quantity $\theta(t) = 2\pi f t + \phi$ is also called the *instantaneous phase* of the oscillator. By neglecting amplitude dynamics, $A(t) = A$, the oscillator is a system described by a scalar state variable, the instantaneous phase, $\theta(t)$. When the oscillator is isolated, the phase increments with a speed proportional to the oscillator *frequency* in rad/s $2\pi f$, $d\theta(t)/dt = 2\pi f$. More general models accounting for amplitude dynamics and chaotic behaviors are possible, but they are out of the scope of this brief introduction. The

time function (or *local time*) associated with $x(t)$ is

$$\tau(t) = \frac{f}{f_0} t + \beta, \tag{1.2}$$

where f_0 is the nominal frequency of the oscillator and $\beta = \phi/(2\pi f_0)$ is the phase in seconds. Because of inaccuracies in the manufacturing process, the actual frequency is always different from the nominal one, $f \neq f_0$. The phase β depends on the instant at which the oscillator was started. A *digital clock* is implemented by driving a counter register with a voltage $x(t)$. At the end of each period (e.g., on positive-going zero-crossings), the register value is incremented, i.e., the clock *ticks*. The value displayed by the register is therefore a *quantized* version of (1.2). In this work, quantization effects will be often neglected, and $\tau(t)$ will be considered as the clock output signal. Given the strict relationship between $\tau(t)$ and $x(t)$, there will be no distinction between clocks and oscillators in the following. From this simple definitions, the possible synchronization regimes of a couple of clocks $\tau_1(t)$, $\tau_2(t)$, are the following (refer to Figure 1.2):

a) *asynchronism*: clocks run independently with different frequencies $f_1 \neq f_2$ and different phases $\beta_1 \neq \beta_2$.

b) *frequency synchronization*: clocks differ for the phase, but they run at the same frequency f_1. Consider two subsequent time instants t_a and t_b. Frequency synchronization implies that the duration of the time interval $(t_b - t_a)$ is measured coherently by both clocks, namely

$$\tau_1(t_b) - \tau_1(t_a) = \tau_2(t_b) - \tau_2(t_a) = \frac{f_1}{f_0}(t_b - t_a). \tag{1.3}$$

c) *phase (and frequency) synchronization*: phase synchronized clocks display the same time at every instant t, $\tau_1(t) = \tau_2(t)$. Phase synchronization naturally implies frequency synchronization .

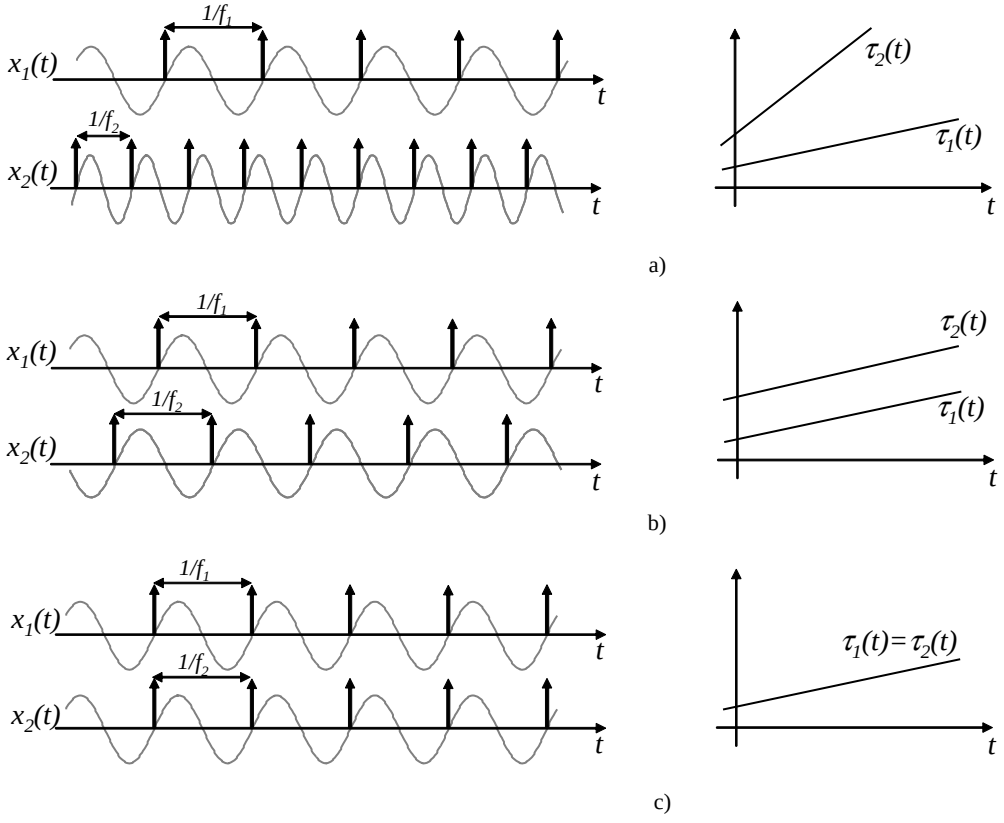


Figure 1.2 Synchronization regimes: a) asynchronous clocks; b) frequency synchronized clocks; c) phase (and frequency) synchronized clocks. Clock ticks are represented as periodic pulses.

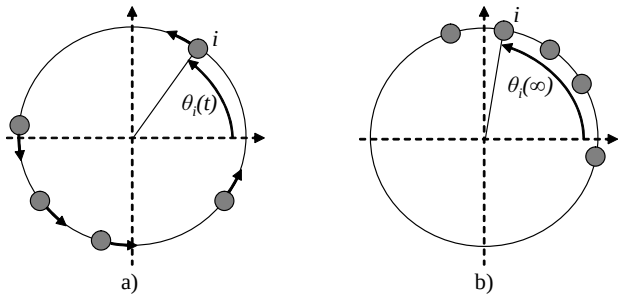


Figure 1.3 Representation of phase oscillators as spheres rotating on a circle: a) uncoupled and b) coupled case (steady-state regime).

Depending on the application, synchronization may be superfluous, frequency synchronicity may suffice, or both phase and frequency synchronization may be requested. Before treating applications, the following section introduces basic mathematical models of coupled oscillators.

1.2 Models of Coupled Oscillators

Since Winfree's work [2], researchers have striven to devise mathematical models suitable to predict the dynamics of biological oscillators. Most of the efforts have concentrated on autonomous synchronization phenomena arising from *mutual coupling* mechanisms. On the other hand, *master-slave* configurations are the typical design choice in engineering practice, whereby an accurate oscillator (the *master*) drives the phase of a lower-quality clock (the *slave*). If the coupling is well-designed, the slave oscillates synchronously (or *in-lock*) with the master at steady-state. In the following, two general models for coupled (non-chaotic) oscillators are introduced, the phase oscillator and the integrate-and-fire oscillator, which have found application both in biology and in engineering researChapter

The *phase oscillator* model has been first proposed by Kuramoto [4]. By neglecting amplitude dynamics, the state of an oscillator can be represented as the position of a sphere rotating on a circle (see Figure 1.3.a). The instantaneous angular position of the sphere, $\theta_i(t)$, corresponds to the instantaneous phase of the oscillator output. The angular speed of

the sphere, instead, is determined by the oscillator frequency in rad/s, $2\pi f_i$. When coupled, oscillators tune their frequency based on the phase offset with respect to their neighbors. Kuramoto proposed a sinusoidal sensitivity function to describe the dynamics of the generic node

$$\frac{d\theta_i(t)}{dt} = (2\pi f_i) + \frac{K}{N}\sum_{j=1}^{N} \sin\left(\theta_j(t) - \theta_i(t)\right), \tag{1.4}$$

where node j is in the set of neighbors of i, $j \in \mathcal{N}_i$, and K is the *coupling strength*. Kuramoto proved that, if the coupling K is large enough, node frequencies are pulled into a frequency synchronous regime. This means that, if the observer's viewpoint is moved on a frame rotating at the convergence frequency f^*, the spheres appear motionless on the circle, at a fixed angular distance (phase offset) from each other (Figure 1.3.b). In particular, global frequency synchronization is achieved if the coupling strength is larger than the maximum frequency offset with respect to f^*, $K/2\pi > \max_i |f_i - f^*|$.

The simple Kuramoto model (1.4) finds relevant application in engineering systems. Consider an electrical oscillator, such as the passive resonator oscillator in Figure 1.4.a, with natural frequency $f_0 = \frac{1}{2\pi\sqrt{L_p C_p}}$. The frequency of the sinusoidal current $I_0(t)$ produced by the active element may be tuned by introducing (or by *injecting*) within the circuit a sinusoidal current at frequency f_{inj}: $I_{inj}(t) = I_{inj}\sin(2\pi f_{inj} t)$. *Injection locking* was first studied by Adler [5], who derived the following equation for the dynamics of the phase of the slave oscillator

$$\frac{d\theta_0(t)}{dt} = 2\pi f_0 + \frac{I_{inj}}{I_0}\frac{\pi f_0}{Q}\sin\left(2\pi f_{inj} t - \theta_0(t)\right), \tag{1.5}$$

where I_0 and $\theta_0(t)$ are the peak amplitude and instantaneous phase of $I_0(t)$ and Q is the quality factor of the resonator. With injection coupling, the coupling strength $K = \frac{I_{inj}}{I_0}\frac{\pi f_0}{Q}$ depends on the oscillator implementation through the quality factor Q. Remarkably, Adler's equation (1.5) is the equivalent of the Kuramoto equation (1.4) for the case of master-slave coupling. As in Kuramoto's case, the slave oscillator locks with the

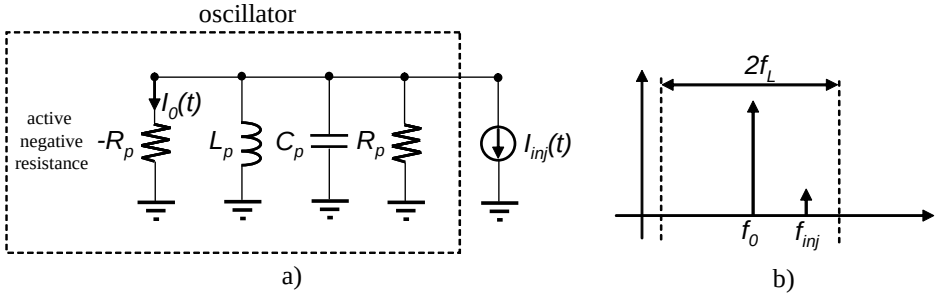


Figure 1.4 Injection locking of a passive resonator oscillator: a) equivalent parallel circuit; b) oscillation and injection frequencies f_0, f_{inj}, with respect to the locking range f_L.

injected signal if the coupling strength is sufficiently large, $K/2\pi > |f_0 - f_{inj}|$. In fact, a fundamental parameter for the analysis of injection locking is the *locking range* $f_L = \frac{I_{inj}}{I_0}\frac{f_0}{2Q}$ (see Figure 1.4.b).

When a more refined control over the slave's frequency and phase is needed, *phase-locked loops (PLL)* are to be preferred to injection coupling [6] (see Figure 1.5). A PLL is a control circuit that tunes dynamically the frequency of a voltage controlled oscillator (VCO) based on the phase difference between the master (or *reference*) oscillator and the VCO. PLL's are common in modern RF transceivers, where they are mainly employed for carrier frequency generation. The dynamic equation of a PLL with a proportional-integral (PI) controller and a sinusoidal phase difference detector is

$$\frac{d\theta_0(t)}{dt} = 2\pi f_0 + K_P \sin\left(\theta_r(t) - \theta_0(t)\right) + K_I \int_0^t \sin\left(\theta_r(\tau) - \theta_0(\tau)\right) d\tau, \tag{1.6}$$

where $\theta_r(t)$ is the reference phase, and K_P and K_I are the proportional and integral branch gains, respectively. In the PLL nomenclature, the VCO controller is commonly called *loop filter*. The loop filter can be modified according to design requirements. The number of integrators within the PLL determines its *type.* As an example, a PLL with a PI controller is a type 2 PLL, as it comprises two integrators: one in the loop filter and one in the VCO. The sinusoidal PLL (1.6) has the same dynamics of an injection locked oscillator (1.5) if $K_I = 0$, but the coupling gain K_P may be tuned by appropriately choosing the gains

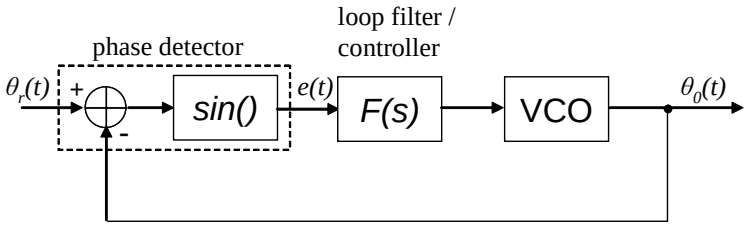


Figure 1.5 Block diagram of a phase-locked loop.

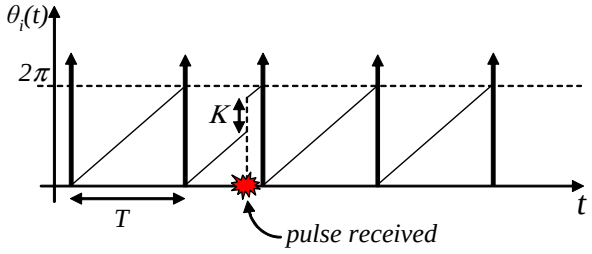


Figure 1.6 Integrate-and-Fire clock model.

of the VCO and loop filter [6]. Furthermore, the integral control allows a type 2 PLL to synchronize to any reference frequency (i.e., it provides a theoretically infinite locking range).

Notice that both injection locking and PLL techniques can be employed to realize mutually coupled networks of oscillators. As an example, systems of mutually injection-coupled oscillators have been studied to implement accurately phased antenna arrays [7].

In the phase oscillator model, oscillators are *continuously* coupled to each other through their instantaneous phases. The *integrate-and-fire (IF) oscillator* model, instead, has been proposed for those situations where oscillators are believed to be coupled by *discrete-time* events, such as the spike of a neuron or the flash of a firefly. In the simplest IF model, originally introduced in [8], oscillators are coupled with neighbors by the exchange of periodic "pulses" (*pulse coupling*). Many variations of the model have been developed over the years, see, e.g., the review in [9]. When the phase of oscillator i reaches the threshold $\theta_i(t) = 2\pi$, it "fires" a pulse to its neighbors and rewinds its phase back to $\theta_i(t) = 0$. Upon receiving a pulse from i, each of its neighbors $j \in \mathcal{N}_i$ advances its phase

with

$$\theta_j(t^+) = \theta_j(t) + K\Delta\left(\theta_j(t)\right), \tag{1.7}$$

where $\Delta(\theta)$ is the coupling function. In the simplest case, depicted in Figure 1.6, it is $\Delta(\theta) = 1$. Recall the phase model of Figure 1.3.a and let t_k be the (absolute) time instant where the i-th sphere crosses the positive horizontal axis for the k-th time. The IF model is based on the observation that synchronization information is contained in the time series $\{t_0, t_1, t_2, \ldots\}$. Remarkably, it has been shown (see, e.g., [9]) that the phase model and the IF model are equivalent in the weak coupling regime, $K \ll 1$.

The *sampled* or digital PLL (DPLL) [10] is a discrete-time controller that tracks the zero-crossings of the reference oscillator signal. Interestingly, its dynamics are the same of master-slave IF oscillators with coupling function $\Delta(\theta) = \sin(2\pi - \theta)$. The dynamic behavior of sampled and analogue PLL's is also equivalent in the weak coupling regime, $K_I < K_P \ll 1$ [6].

A third PLL type often encountered in network synchronization systems is the *software* PLL (SPLL). SPLL allows the reference node to transmit synchronization messages *asynchronously* by including the local clock value at transmission time within the message itself. The clock controller at the receiving (slave) node is then implemented in software, as inferred from the name.

1.3 Networks of Coupled Oscillators

The oscillator models of Section 1.2 were developed assuming that each oscillator can interact with all others[1] (full, or *all-to-all* connectivity). The interest in distributed synchronization mechanisms was revived in recent years by the development of suitable models to describe *complex networks* of dynamical systems [11], whereby each node is sensitive only to a subset of the network nodes (*neighbors*). Network connectivity

[1]With master-slave topologies, this assumption means that all slaves are attached to a single master node.

determines which nodes can be coupled to each other (either mutually or in master-slave fashion) and it can be represented as an undirected graph. Tools from *graph theory* have allowed to categorize the various connectivity models that have arisen in researChapter A brief overview of the most important connectivity models is hereby provided:

- *Lattice graphs*: the graph vertexes are deployed regularly as the nodes in a lattice or a grid. The two most relevant cases are the two-dimensional grid (square grid graph), and the one-dimensional grid (path graph or line network). The model may be extended by allowing nodes to be connected with all neighbors within a connectivity radius r.

- *Random graphs*: the graph is generated by some random process. Several variations exist depending on the assumptions on the edge generation process. According to the popular model first proposed by Gilbert, a link between any two nodes is established with fixed probability p.

- *Random geometric graphs*: nodes are scattered uniformly and independently at random in a bounded region (e.g., a square area). Two nodes are coupled if their geometric distance is smaller than a fixed threshold (or connectivity radius) r. It has been proved that dense random geometric graphs share important characteristics with the lattice graph defined over the same region (and with the same connectivity radius r).

- *Small-world networks*: introduced by Watts and Strogatz [11], these networks feature a small shortest-path length between any couple of nodes (*small-world effect*), as for random graphs. By contrast, lattices are also known as "large-world" networks, as the average shortest-path length between a couple of nodes is the largest among all types of graphs. Small worlds can be generated starting from a regular lattice, part of whose links are "rewired" at random so as to connect nodes that were far away from each other in the original graph. The reason of their importance relies in that

small-world graphs appear to fit many real-world networks, such as social networks of people, populations of firing neurons or chirping insects.

- *Scale-free networks*: first studied by Barabási and Albert [11], they constitute a special case of small-world networks. "Scale-free" refers to the fact that the network lacks a characteristic scale since the vast majority of its nodes has a small number of connections (or *degree*), while a small minority of nodes (called *hubs*) features a very high degree. It is conjectured that these networks cannot be created just by purely random rewiring. Barabási and Albert devised a mechanism to generate scale-free graphs called "preferential attachment". Basically, rewiring is not performed at random, but nodes connect preferably with other nodes having a high degree (a sort of rich-get-richer mechanism). Preferential attachment is not the only generative model for scale-free networks, and other algorithms have been devised recently. The general scale-free model fits many real-world networks, both natural and artificial, such as scientific collaboration networks, protein interaction networks, sexual partners among humans, and the World Wide Web.

A wireless network comprises nodes scattered in a bounded area or volume. Also, nodes are coupled with neighbors within a connectivity radius which depends on transmission power and receiver sensitivity. From these observations, lattice and random geometric graph models are often encountered for the analysis of wireless networks (see the discussion in Chapter 2). The reader interested in more details about small-world and scale-free networks is referred to the recent review in [12].

The connectivity graph describes which links can be activated in order to couple the corresponding oscillators and build the *synchronization network*. The synchronization network is forcibly a sub-graph of the connectivity graph, and it is in general a a *directed graph*. In particular, two types of synchronization networks are found in practice: *mutually coupled* (MC) and *master-slave* (MS) networks.

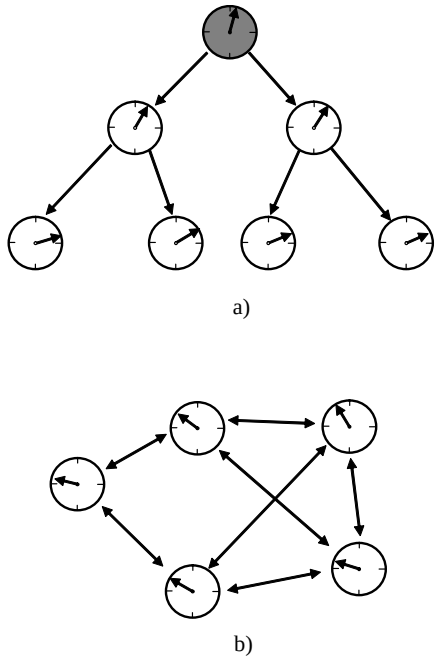


Figure 1.7 Network synchronization topologies: a) Master-Slave and b) Mutually Coupled.

In MS networks, (Figure 1.7.a) a single node is deployed with a precise reference clock (*master* node). The master node stays at the top of a hierarchical topology whereby nodes belonging to a lower layer adjust their local clocks given the synchronization information from their parents.

In MC networks, (Figure 1.7.b) nodes are arranged in a peer-to-peer (or *mutually coupled*) fashion: each node controls its local clock with the information received from all its neighbors. In this case the synchronization network often coincides with the connectivity graph. Also, mutually coupled topologies are typically employed to model most of the synchronization phenomena observed in nature.

Research on distributed synchronization aims at understanding the interplay between the synchronization network structure and the propensity of the network to synchronize. Synchronization networks are also treated analytically with tools from algebraic graph theory, see Chapter 2.

1.4 Network Synchronization in Wired Networks

Synchronization is a fundamental property in many fields of communication engineering, ranging from RF transceiver design to time distribution based on atomic clocks. This section reviews synchronization techniques employed in wired communication networks.

1.4.1 Digital Circuit-Switched Networks

In circuit-switched networks based on digital time-division multiplexing (TDM), synchronization is a crucial facility in order to allow heterogeneous lower-rate bit-streams to be conveyed on a single higher-rate transmission link. In fact, the frequency of the local clock at a given node determines the actual bit-rate at the transmission interface. Clock frequency differences between nodes in the network cause elastic buffers either to overflow or to empty periodically (*slip* events) [13]. Asynchronous digital multiplexing systems, such as those employed to implement the plesiochronous digital hierarchy (PDH) developed in the 1960's**,** employ bit justification techniques (also known as "pulse stuffing") in order to compensate for clock frequency offsets. Network-wide synchronization enables *synchronous* digital multiplexing. Synchronous multiplexing systems, such as those employed by the synchronous digital hierarchy (SDH), do not need bit justification, thereby saving valuable transmission resources.

If the nodes have clear sky visibility, universal time and frequency transfer can be realized via GPS (Figure 1.8). The GPS signal is referenced to universal time (UTC) and can be employed to provide a stable time and frequency reference to a local clock. GPS is often used as a backup synchronization facility in public switched telephone networks (PSTN) and as the primary time and frequency reference for base stations in 2G/3G wireless cellular systems. When employing a single satellite as reference (time dissemination configuration, Figure 1.8.a), the time offset estimation error depends on the uncompensated propagation delay between the satellite and the local clock, and it has been reported to amount to less than 100 ns [14]. Relevant improvements can be achieved by

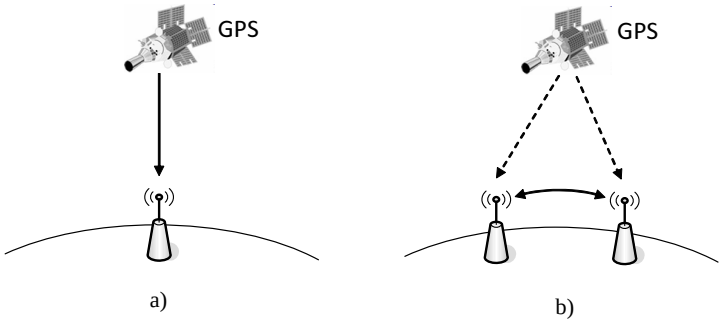


Figure 1.8 GPS time and frequency transfer: a) time dissemination and b) common view configurations.

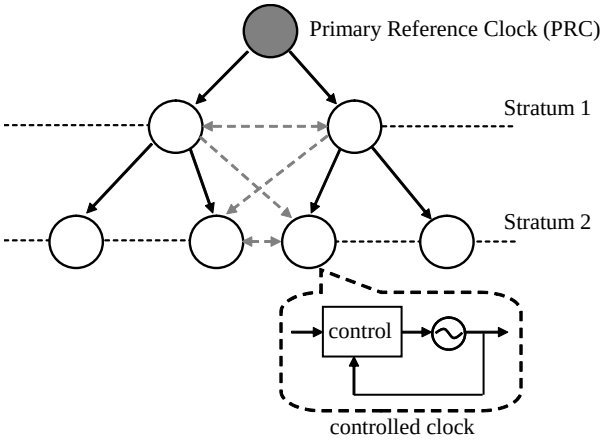


Figure 1.9 Master-slave synchronization network employed for circuit-switched communication. Backup links are represented as dashed lines.

allowing two nodes to cooperate by exchanging their measured time offsets (common view configuration, Figure 1.8.b). In this case one of the two clocks has a known relationship to UTC and acts as the synchronization master for the other. By taking the difference of the two measurements, most of the uncompensated propagation delay cancels out and the offset estimation error (between master and slave) drops below 30 ns [14].

Whenever GPS is not available at all node locations, there is need for a *distributed* network synchronization mechanism. In particular, SDH networks employ coupled PLL's to realize network time and frequency distribution [15][13]. During all the 1970's and 1980's there has been a long debate over which topology (either MS or MC) should have been employed to implement the synchronization network for synchronous multiplexing

systems. The adopted solution in SDH (and in most other cases) is a MS architecture as the one in Figure 1.9, whereby a master node (primary reference clock - PRC) occupies the top of a hierarchical topology which is divided in subsequent layers called *strata*. Nodes belonging to a lower stratum (slaves) adjust their local clocks given the synchronization information from upper layers by employing a PLL clock controller. The stability and accuracy characteristics of the clocks at each stratum are defined by international standards. In practical deployments, additional links (dashed lines in Figure 1.9) are available for backup purposes in case of outage of the main links. MC architectures were intensely investigated and found adoption in proprietary and military networks featuring stringent requirements in terms of robustness to node failures, topology changes and clock instabilities. In his seminal work [15], Lindsey and others proposed the use of a well-connected (fully connected where feasible) MC network of PLL's. According to Lindsey's design, each PLL employs as local reference a weighted average of the clock signals from neighboring nodes. This configuration proved to be more robust against clock frequency instabilities and path delays as compared to MS designs [15]. Nevertheless, the MS topology is typically preferred in practical systems as it features simpler deployment methodologies and easily predictable performance [13].

A complete review of frequency offset compensation and network synchronization techniques for TDM networks (with a discussion on their suitability for specific network topologies) can be found in [16].

1.4.2 Packet-Switched Networks

In packet-switched networks, distributed applications may require synchronization for different purposes such as data timestamping, versioning control, and time-of-day services. Also, industrial networks comprise distributed sensory and control systems that need a common time-scale in order to timestamp sensed data and coordinate actions. Differently from the case of circuit-switched networks, clocks may not extract synchronization

information from electrical signals on wire[2]. Also, a GPS receiver may be too costly or unfeasible due to the lack of clear sky visibility. In packet-switched networks such as Ethernet local area networks (LAN) and the Internet, synchronization information is exchanged by *time-stamping* packets at reception and transmission times and by inserting these *timestamps* in their payload. A timestamp is a sample of the local clock taken at packet transmission or reception time[3]. What undermines synchronization accuracy in packet-switched networks is the delay in packet delivery. On the Internet, delays are caused by queuing in intervening routers, while heavy traffic loads may trigger retransmissions in LAN's employing contention-based channel access mechanisms. Delays are in general asymmetric and time-varying because of route changes and traffic load variations. The random delay component is referred to delay jitter or packet delay variation (PDV). Synchronization algorithms usually aim at estimating and compensating the deterministic delay component and filtering out the random PDV. The two most prominent protocols are the network time protocol (NTP) [17] and the recent IEEE 1588 or precise time protocol (PTP) [18]. NTP is an application-layer protocol for TCP/IP networks, currently developed by IETF. NTP messages are transported as a payload in UDP packets, and NTP clocks are organized within a hierarchical MS structure (NTP sub-network) analogous to Figure 1.7.a. PTP targets relatively localized systems and aims at synchronizing real-time clocks for industrial automation and circuit-switching emulation. PTP messages can be transported by UDP or they can be directly inserted in the payload of Ethernet frames. PTP is also based on a MS topology. Both protocols assume that the deterministic delay component is symmetric, and they estimate it by means of pair-wise handshakes (returnable time approach) as in Figure 1.10. NTP messages can carry multiple timestamps, and the exchange follows the client-server paradigm, while PTP messages carry only

[2]An exception is the recent Synchronous Ethernet system (SyncE), which synchronizes clocks on a Ethernet LAN by feeding the physical electrical signals on the transmission line to a PLL, just as in SDH circuit-switched networks. Unfortunately, it is not possible to synchronize heterogeneous wide area networks (WAN) with SyncE, a limitation that currently affects its widespread application.

[3]The clock sampling operation is sometimes called clock *capture* when referred to hardware clocks.

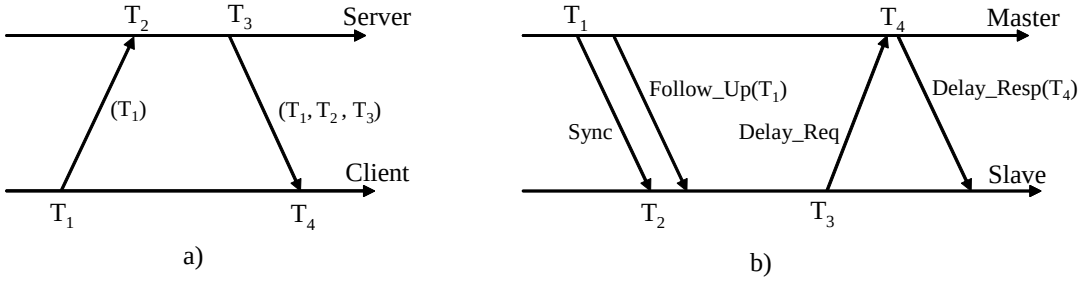


Figure 1.10 Timestamp exchange for a) NTP and b) IEEE 1588 (PTP). Timestamps carried within messages are shown between brackets.

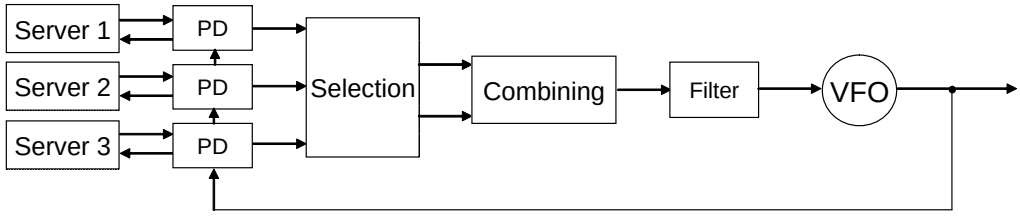


Figure 1.11 NTP clock control algorithm (VFO - variable frequency oscillator).

a single or no timestamps at all (depending on the message type), and the exchange performed in a master-slave fashion. Another important difference is that NTP controls the operating system software clock, while PTP controls the hardware clock in the network interface card (NIC). Hardware assist makes PTP timestamps more accurate than NTP timestamps as it clears out additional delays due to packet processing and OS interrupt management. In particular, the timestamp is captured right after the preamble of the PHY layer frame, before the start of the MAC layer header. The clock control law adopted by the Network Time Protocol (NTP), is based on a discrete-time *software* PLL[4] (see Figure 1.11), which is designed by exploiting the analogy with an analogue PLL mentioned before. Also, the conventional PLL design is augmented with peer selection and combining functions. In particular, the peer selection procedure provides resilience to potential attackers ("falsetickers") and individuates correct time providers ("truechimers"). The clock control algorithm for PTP is not specified in the standard and it

[4]The design of the NTP software PLL is actually complicated by the intrinsic asynchronism of time message exchanges.

is therefore implementation-dependent. In any case, selection and combining functions are not necessary as the standard requires a slave node to select a single reference master once and for all. The rationale behind this is to keep the slave as simple as possible, in contrast with NTP, where the algorithm intelligence is all in the client. In [19], the authors propose a simple clock control algorithm tailored for the implementation of PTP on a Ethernet. NTP maintains a maximum time error with respect to UTC of 10 ms over the public Internet, and less than 200 μs over a LAN in ideal conditions. PTP is reported to achieve accuracies in the sub-microsecond range over a LAN thanks to hardware timestamping support. Finally, the theoretical resolution of NTP timestamps (i.e., the smallest measurable time interval) is 232 ps, while it is 1 ns for PTP.

1.5 Network Synchronization in Wireless Networks

Motivations

In a wireless network, network-wide synchronization enables crucial features at different layers of the protocol stack:

- **Physical layer: Cooperative communication.** Cooperation among independent nodes allows to employ distributed (or *cooperative*) communication techniques at the physical (PHY) layer in order to convey a single information stream to one or multiple receivers. If perfect cooperation is possible, a group of cooperating devices can utilize all the signal processing and coding schemes originally devised for co-located antenna arrays, thereby defining a *virtual* antenna array (VAA). In particular, link reliability and resilience to channel fading can be improved by implementing distributed space-time codes (DSTC). A STC consists in a set of permutation and simple processing operations that cooperating nodes have to perform on the information stream to be transmitted. Lack of symbol-time synchronization among cooperating nodes causes inter-symbol interference (ISI) at the receiving node, thereby degrading code performance (Figure 1.12). Recent works have proposed to

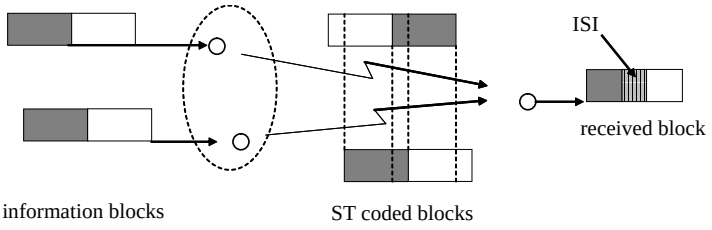


Figure 1.12 Occurrence of inter-symbol interference (ISI) when employing distributed space-time codes (DSTC) with lack of symbol-time synchronization.

enhance resilience to ISI by employing orthogonal frequency division multiplexing (space-time OFDM - ST-OFDM) or time-reversal STC (TR-STC) [20]. While time synchronization at the PHY layer may be superfluous, the opposite is true for *carrier frequency* synchronization. Carrier frequency offsets (CFO) among transmitting nodes cause relevant signal distortion on the receiving side, and need to be compensated by suitable synchronization techniques. In the simple case of Figure 1.12, one of the two cooperating nodes can serve as reference for the other, but such an approach entails a rapidly increasing overhead as the cooperating cluster gets large. A de-centralized frequency synchronization algorithm needs to be devised in order to guarantee the necessary scalability with the number of nodes.

- **MAC layer: Coordinated medium access.** In wireless sensor and ad-hoc networks, all nodes employ the wireless medium for communication. Therefore, medium access needs to be disciplined either by a distributed or a centralized medium access control (MAC) protocol. Efficient MAC protocols divide the time resource into super-frames, which are further divided into smaller time-slots (time division multiple access - TDMA). A node may contend with others for access to a particular time-slot, or it can be assigned a dedicated resource by a centralized or distributed scheduling mechanism. In both cases, *slot time* synchronization is crucial to avoid accidental packet collisions (Figure 1.13.a), or the employment of excessive guard times. The fulfillment of this requirement is particularly demanding when nodes

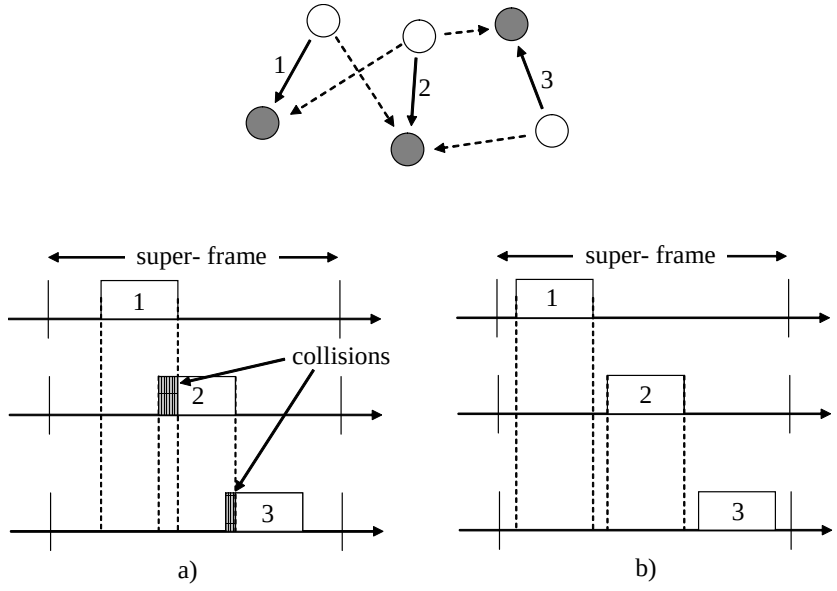


Figure 1.13 Slotted medium access access in presence of a) imperfect and b) perfect slot timing. Packets and corresponding active links are matched with the same number. Dashed lines connect each transmitter with interfered receivers.

are allowed to turn off the RF transceiver for long periods ("sleep" mode), and they are unable to update synchronization as frequently as requested by the stability of the local oscillator. Notice that, in systems employing frequency division multiple access (FDMA), carrier frequency synchronization is also relevant at the MAC layer.

- **Application layer: Distributed sensing.** An appealing application of large-scale sensor networks is the distributed sampling of space-time signals (Figure 1.14) for, e.g., geophysical prospecting and target localization. Correct reconstruction of the space-time signal requires nodes to be time synchronized sufficiently well to avoid unwanted aliasing effects.

Functional view of synchronization

In order to perform diverse tasks at different layers, a wireless sensor node is usually equipped with one or more local oscillators. The essential block diagram of a generic wireless device is depicted in Figure 1.15. It comprises a RF transceiver and a micro-

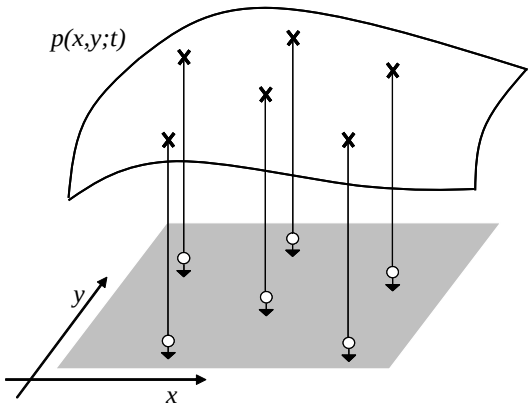


Figure 1.14 Distributed sampling of a space-time field.

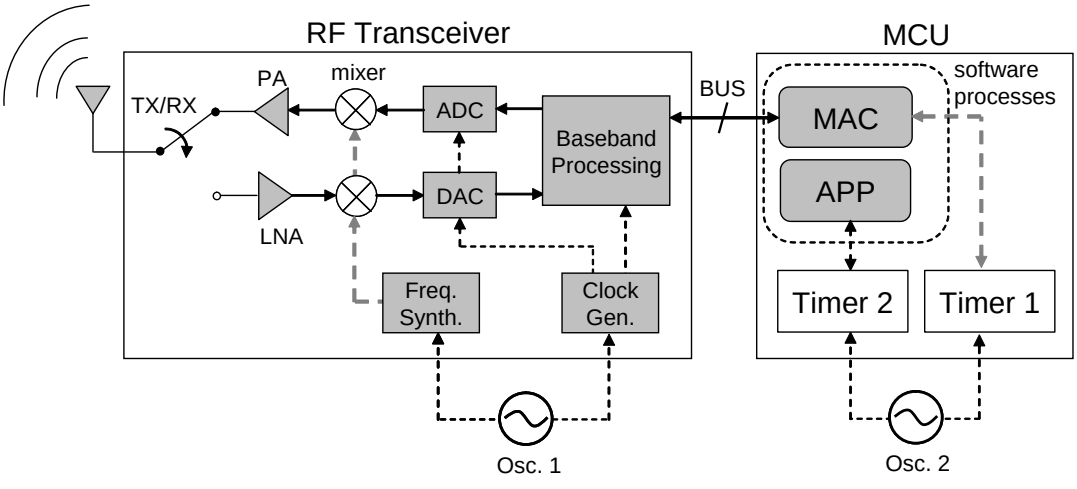


Figure 1.15 Block diagram of a generic wireless device. Synchronization signals of interest are highlighted.

controller unit (MCU) connected by a dedicated bus. In this example, oscillator 1 serves the RF transceiver by generating (through appropriate synthesizers) both the carrier signal for RF up/down-conversion and the clock employed for sampling and processing of digital baseband signals. Oscillator 2 drives two timers necessary for MCU operation. Timer 1 is employed by the medium access control (MAC) process to determine the start of access time-slots and to regulate the sleep cycle of the transceiver module. Timer 2 can be used by the operating system in order to provide a system clock to applications or other routines (e.g., sampling and actuation functions). In practice, more complicated configurations are typically encountered as, for example, the PHY layer may need to be coordinated with MAC layer timing (e.g., the symbol clock may be required to be in sync with MAC slot start time [21]).

This thesis focuses on lower layers, namely on the PHY and MAC layer. In particular, the results in this work comprise techniques to compensate CFO at the physical layer and to achieve time synchronization at the MAC layer. The clock output signals of interest for these purposes are highlighted in Figure 1.15. The next section provides a brief review on synchronization techniques for CFO compensation and time synchronization in wireless networks.

1.6 State of the Art

Conventional synchronization techniques entail several drawbacks that impair their application in WSN. The cost of a GPS receiver is excessive in terms of price and consumed energy. Furthermore, sensors are not under clear sky visibility in indoor deployments, and GPS signal reception is easily impaired even outdoor by, e.g., tree branches. Radio clocks based on short-wave terrestrial broadcast are typically not accurate enough (with a maximum time error of 1 ms) and require to increase the size of devices significantly. NTP does not satisfy complexity and accuracy constraints because of its client/server architecture and the employment of software timestamps (but it supports duty cycles as

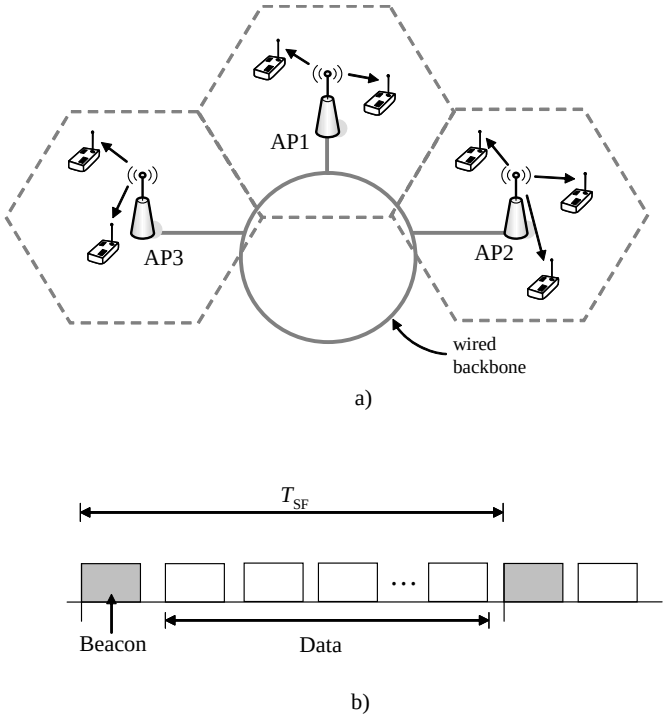


Figure 1.16 Synchronization in centralized wireless communication systems: a) network topology and b) TDMA super-frame structure.

low as one packet per day). Tentative implementations of PTP on sensor networks have been proposed (see, e.g., [22]), but there are still serious concerns about its scalability in large WSN. For these reasons, WSN constitute a new regime for network synchronization [23], and require the development of new synchronization techniques.

1.6.1 Synchronization in Wireless Communication Systems

Most wireless communication systems in operation are based on a centralized network structure, whereby an access point[5] (AP in Figure 1.16.a) manages the nodes that are within its transmission range. Multiple AP's are connected through a wired backbone infrastructure in order to guarantee network-wide connectivity. In this context, the AP is the natural reference for synchronization purposes. As far as frequency synchronization, each terminal may easily estimate its CFO from the preamble of packets received from the

[5]The denomination of the AP may change depending on the context: e.g., in a cellular network it is called "base station", while in a WSN it is called "network coordinator".

AP. Frequency synchronization between AP's is required only by cellular communication systems in order to support handovers from one cell to the other, and it is usually provided by a GPS receiver. Time synchronization is necessary whenever medium access is organized into time-slots, in particular with TDMA protocols, whereby the AP reserves each time-slot to a specific terminal. Typically, nodes obtain time synchronization through a signaling channel. As an example, in networks based on IEEE 802.11 [24] and 802.15.4 [25] the AP transmits a signaling frame (also called "beacon", see Figure 1.16.b) at the start of each super-frame. By tracking the transmission time of the beacon frame, each node maintains time synchronization with the AP. In cellular systems, propagation delays are estimated by the AP and communicated to each terminal in order to be compensated (timing advance mechanism). IEEE 802.11 wireless LAN's, in the absence of a network coordinator (Independent Basic Service Set - IBSS), prescribe each node to contend for beacon transmission at the start of a new super-frame [24]. Specifications require to adjust the local clock by employing Lamport's algorithm [26], which retains some similarities with the pulse-coupling mechanism of IF oscillators.

WSN and ad-hoc networks, instead, extend over multiple hops and cannot rely on a pre-existing wired infrastructure. Sensor networks based on ZigBee/IEEE 802.15.4 [27][25] guarantee network-wide connectivity by organizing nodes in a cluster-tree topology (see Figure 1.17.a). Each cluster has a local coordinator, and coordinators of different clusters are organized in a fixed hierarchy. The cluster coordinator acquires synchronization from its parent in the hierarchy, and in turn provides a time reference to nodes within its cluster by transmitting a beacon frame in a reserved beacon slot. The ECMA 368 standard for high-rate personal area networks [28] allows a flat mesh architecture (see Figure 1.17.b), whereby *each node* is reserved a beacon slot for signaling and synchronization purposes. Each node adjusts its local clock so as to be synchronized with the slowest clock in the network. In both cases, the beacon slot allocation procedure is a critical functionality that determines network performance, see, e.g., [29][30]. TSMP

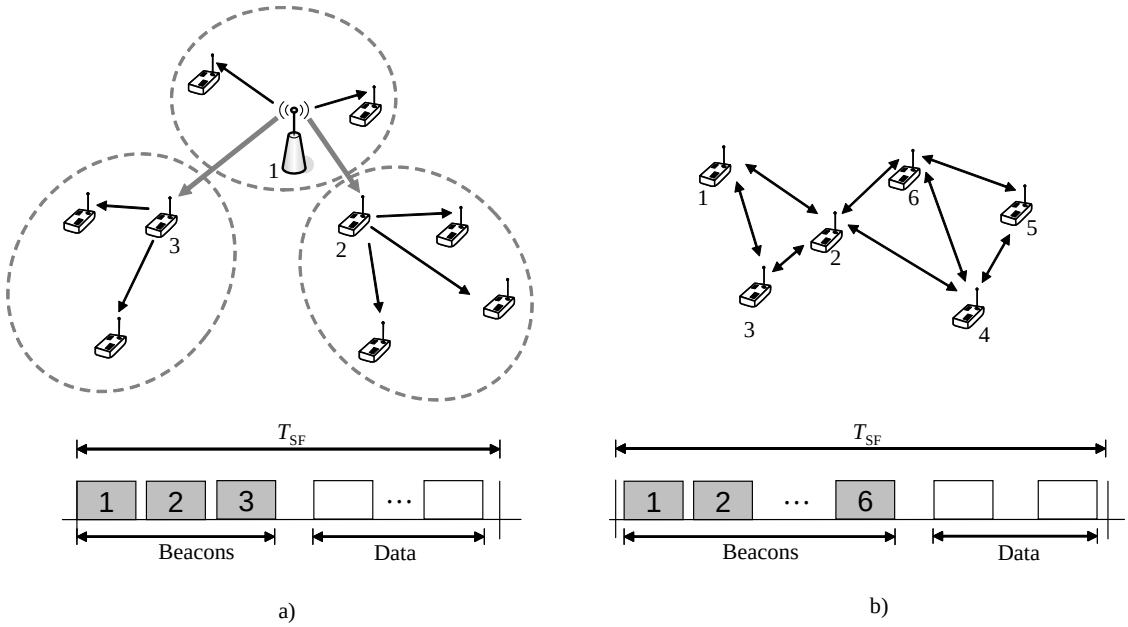


Figure 1.17 Synchronization in multihop wireless network with a) cluster-tree topology (ZigBee) and b) mesh topology (ECMA 368).

[31] is a TDMA protocol which does not make use of beacons. When receiving, a node measures the offset of packet reception time with respect to the expected slot start time. If necessary, measured offsets are piggybacked to the transmitter within the acknowledgment (ACK) frame. In order to distribute a network-wide time reference, TSMP organizes nodes in a tree structure. TSMP is at the basis of the WirelessHART [32] standard for industrial automation and IEEE 802.15.4E, the forthcoming amendment to 802.15.4 MAC specifications (also known as time-synchronized channel hopping MAC - TSCH). It is important to emphasize that all the above mentioned standards prescribe the use of accurate hardware-assisted timestamps.

1.6.2 Protocols for Distributed Synchronization

All of the works reviewed in this section focus on time synchronization. To the best of the author's knowledge, distributed CFO compensation had not been considered before. When targeting time synchronization, local time is translated to network (absolute) time by compensating the local clock signal for phase and frequency offsets. Distributed

algorithms are able to estimate the parameters of the local clock by employing observations of (local) pair-wise timing offsets. Recently, a number of works have applied Master-Slave (MS) and Mutually Coupled (MC) synchronization architectures to realize distributed synchronization in wireless networks, see, e.g., the reviews in [33][34][35]. Notably, a number of works has proposed the use of periodic RF pulses in place of timestamps for time information dissemination.

The first protocols to emerge were based on MS topologies. The timing-synch protocol for sensor networks (TPSN) [36] was the first to propose to exploit the enhanced accuracy of hardware timestamps to estimate pair-wise phase offsets. The flooding time-synchronization protocol (FTSP) [37] improved over the performance of TPSN by correcting frequency offsets through pair-wise linear regression. FTSP entails also increased communication overhead as it requires nodes to transmit multiple timestamps at a time. This feature enables the receiving node to filter out most of the residual communication delays. FTSP achieves a typical time error of a few microseconds. In order to reduce uncertainties due to transmission delays, the reference broadcast synchronization (RBS) protocol [38] employs the same approach of GPS common view configuration in Figure 1.8.b (also called receiver-receiver synchronization to distinguish it from the conventional approach or sender-receiver synchronization). In this case, the role of the GPS satellite is covered by a third cooperating node. RBS estimates pair-wise phase and frequency offsets by linear regression. Notably, RBS can be adapted to handle more general topologies. Improvements to the original RBS protocol may be found in [39]. Servetto [40] was the first to propose the use of periodic RF pulses in place of timestamps. Synchronization with RF pulses is appealing due to the possibility of exploiting the properties of the radio channel, whereby signals transmitted over the air superimpose with a significant reduction of synchronization overhead. In [40] synchronization pulses (or pilot sequences of pulses) from *multiple* parents superimpose at the receiver by the nature of the radio channel. A receiving node can estimate its clock parameters by averaging

the received pulses and by applying linear regression to the resulting observations. The algorithm may be extended to dense (i.e., well-connected) MC networks by assuming that phase offsets average out in the limit of infinite nodes, thereby giving rise to an observable reference clock.

An algorithm tailored for hybrid MS/MC networks was proposed in [41]. As in the previous protocols, each node estimates first pair-wise phase and frequency offsets with respect to its neighbors. By observing that absolute phase and frequency offsets can be written as a linear combination of pair-wise offsets, clock parameters can be computed by solving a linear system of equations with a distributed (Jacobi-like) algorithm.

The usage of MC topologies has been inspired by natural phenomena, where large populations of oscillators are entrained by wireless signals such as flashing lights (fireflies) and chirps (crickets). In an effort to mimic nature, most of the previous works in the area propose the use of periodic RF pulses for time and frequency synchronization. The IF pulse-coupling mechanism was the first to be considered for employment in WSN by [42]. Unfortunately, the basic IF model does not provide an easy way to jointly compensate both phase and frequency offsets, and it provides only frequency synchronization. For this reason, [43] considered the use of a PLL clock controller, as suggested by previous experience in the field of digital circuit-switched networks [15]. In particular, [43] adopted a discrete-time type 1 second-order PLL, which was demonstrated to correct both frequency and phase offsets, up to a residual phase mismatch which depends on loop parameters. The algorithm of [43] can be improved by employing a type 2 PLL (i.e., a PI controller), which provides full phase and frequency synchronization. Stability and steady-state accuracy of a MC network of type 2 PLL's were studied in [44][45], along with the use of timestamps instead of RF pulses. In particular, the clock control scheme in [45] is closely related to the one employed in NTP, and it could be extended to support fully asynchronous (i.e., non periodic) timing message exchanges. The algorithm developed by [46] adopts a distributed frequency offset estimation algorithm which is based on consensus strategies.

The frequency estimation algorithm is operated jointly with a PLL for full phase and frequency correction. Finally, the protocol proposed by [47] is similar to a type 2 PLL, but there the frequency of the local clock is controlled in a nonlinear fashion. The authors of [47] prove that the algorithm is robust against packet collision events, and it is therefore tailored for implementation with contention-based medium access schemes.

1.7 Outline of the Thesis and Contributions

In this thesis, network synchronization is approached by employing clock control algorithms based on the concept of phase and frequency locked loops (PLL and FLL). The performance of these rather classical methods is analyzed in the novel context of multihop wireless sensor networks (WSN). In particular, this work focuses on synchronization both at the physical and at the MAC layer. The structure of physical and MAC layer frames dramatically affects the way nodes are allowed to exchange synchronization information and this needs to be taken into account in the design and analysis of synchronization algorithm. On the other hand, WSN offer a wider flexibility in the design of the synchronization architecture. A specific architecture may be even chosen dynamically depending on the network topology. For this reason, the performance of synchronization algorithms are evaluated on both MC and MS synchronization networks. The main contributions of the present work are the following:

- At the physical layer, the focus is on *frequency* synchronization issues arising when employing distributed space-time coding techniques (Chapter 3). An algorithm based on distributed FLL (DFLL) is developed in order to compensate CFO among an ensemble of nodes. A suitably designed frequency difference detector extracts synchronization information from the preamble of PHY layer frames while the control loop adjusts the frequency of the local oscillator. Stability conditions are provided and the steady-state accuracy of a network of DFLL's is evaluated for MC

and MS topologies. The effects of CFO and frequency synchronization are analyzed on a multi-hop relay network employing distributed orthogonal space-time codes.

- At the MAC layer, the focus is on *time* synchronization issues arising with TDMA protocols. Firstly, synchronization acquisition is studied (Chapter 4) during the network set-up phase. Since MS networks achieve synchronization in finite time, the focus is on MC topologies and consider three transmission strategies for the exchange of synchronization-related information: superposition (pulse-coupling), contention and reservation-based beacon transmission. Sufficient conditions for stability of the considered access strategies are derived, and it is verified that superposition yields fastest convergence. Next (Chapter 5), the focus is on the steady-state accuracy attainable by distributed synchronization algorithms during normal network operation. The case of beacon-enabled TDMA protocols is considered, whereby syncrhonziation is based on the periodic transmission of beacon frames. The performance of distributed synchronization based on PLL clock control is compared with a distributed regression algorithm for the estimation of clock parameters and with the Cramer-Rao lower bound (CRLB) for the observation model at hand. Closed-form expressions are provided for the accuracy of PLL's over regular MC networks and for the performance of distributed regression over general networks. Simulation results show how MC topologies are more suited for networks with good connectivity, while MS architectures achieve reasonable accuracies also on poorly connected graphs. Finally, the case of beacon-less TDMA protocols is considered. These protocols are typically employed when the network operates under very low duty cycles (Chapter 6). In this case, synchronization is performed along with data communication. Also, node clocks cannot be considered stable and their frequency is randomly time-varying due to, e.g., changing environmental temperature. The frequency tracking performance of a type 2 PLL and a joint PLL/FLL (P/FLL) algorithm is investigated and compared with conventional techniques based on

(rough) temperature compensation (temperature compensated clocks - TCC). In particular, analytical expressions are derived concerning the attainable tracking accuracy over regular MC networks and it is shown by simulations that PLL and P/FLL techniques are superior to TCC.

In more detail, the outline of the thesis is as follows.

Chapter 2: This chapter contains some mathematical concepts used throughout the thesis. In particular, the clock model and the network topological models are presented from the point of view of both graph theory and linear algebra.

Part I - Synchronization at the Physical Layer

Chapter 3: The establishment of a common frequency reference in a wireless network is a critical factor in enabling any degree of node cooperation in communication and sensing functions. This chapter introduces distributed frequency-locked loops (D-FLL) to synchronize the frequencies of autonomous nodes with wireless communication capabilities. D-FLL's are connected within a synchronization network that may be organized with a peer-to-peer (MC) or hierarchical (MS) topology. The stability of the D-FLL synchronization network is investigated considering the use of a suitable frequency difference detector. The accuracy of frequency tracking is also analyzed in detail for two sample network topologies. D-FLL's prove to be a robust and accurate technique for frequency synchronization purposes that can be readily employed with any network architecture. This chapter is based on

- N. Varanese, O. Simeone, U. Spagnolini, Y. Bar-Ness, *"Distributed frequency-locked loops for wireless networks"*, in proc. International Symposium on Spread Spectrum Techniques and Applications (ISSSTA), Bologna, Italy, Aug. 25-28 2008.

- U. Spagnolini, N .Varanese, O. Simeone, Y. Bar-Ness, *"Distributed digital locked loops (D-DLL) for time/frequency locking in packet communications"*, in proc. International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Cannes, France, Sept. 15-18 2008 (invited paper).

- N. Varanese, U. Spagnolini, Y. Bar-Ness, *"Distributed frequency locked loops for wireless networks"*, submitted to IEEE Trans. on Communications (second revision round).

Part II - Synchronization at the MAC layer

Chapter 4: Network synchronization at the MAC layer can improve the performance of wireless networks by enabling the use of slotted medium access protocols, such as TDMA. Distributed synchronization is an appealing technique to achieve network synchronization in wireless *ad-hoc* and sensor networks. At the MAC layer, different signaling techniques and medium access protocols may be employed for synchronization information. This chapter evaluates the impact of the use of reservation, contention and superposition access protocols on the convergence rate of distributed synchronization based on phase-locked loops (PLL). Contention and superposition are random access protocols, and almost sure convergence can be guaranteed by studying convergence in the mean. Also, an approximation of the convergence rate of random access protocols is proposed based on numerical results. Finally, the performance of the considered protocols over a regular square grid network is compared via simulations. The results of this chapter have been extended from

- N. Varanese, Y. Bar-Ness, U. Spagnolini, *"On the synchronization rate of distributed medium access protocols"*, in proc. Conference on Information Sciences and Systems (CISS), Princeton, NJ USA, Mar. 17-19 2010.

Chapter 5: This chapter considers a TDMA medium access protocol, where synchronization is obtained by the periodic transmission of beacon frames. The study focuses on the accuracy attainable by employing distributed synchronization techniques. In particular, the focus is on two different approaches to the solution of the distributed synchronization problem. The first (closed-loop algorithm) is based on the distributed control of local clocks trough a type 2 phase-locked loop (PLL), while the second (open-loop algorithm) is based on the distributed estimation of the phase and frequency offsets of the local clocks. Both protocols are shown to be suitable for implementation over mutually-coupled (MC), master-slave (MS), and hybrid synchronization networks. The synchronization accuracy of the open-loop algorithm is derived for all topologies of interest, while for the accuracy of the closed-loop algorithm exact analytical results are available only for regular MC networks. The performance of practical algorithms is then compared with the Cramér-Rao lower bound (CRLB) for the problem at hand. Results show that distributed algorithms are inefficient with respect to the CRLB over peer-to-peer topologies, whereas they achieve the accuracy limit over MS hierarchical architectures. Finally, the performance of MC and hybrid topologies improves rapidly when increasing network connectivity, while MS proves to be the best choice in poorly connected networks. Part of the results in this chapter are contained in

- N. Varanese, U. Spagnolini, Y. Bar-Ness, *"On the accuracy of distributed synchronization algorithms for wireless networks"*, in preparation for submission to IEEE Trans. on Signal Processing.

Chapter 6: This chapter focuses on TDMA MAC protocols where nodes activate their transceiver with a very low duty-cycle. Signaling and synchronization information is exchanged along with data and ACK messages (beacon-less protocol). With low duty-cycles, local clocks are subject to relevant frequency changes driven by environmental temperature variations. This chapter analyzes the capability of adaptive clock control algorithms to track frequency instabilities. In particular, two algorithms are considered,

namely a type 2 PLL with a proportional-integral controller and a type 1 PLL equipped with an independent frequency tracking loop (P/FLL). The performance of adaptive tracking techniques is checked against a conventional approach based on improving clock accuracy by the compensation of frequency changes due to environmental temperature variations (temperature-compensated clock - TCC). Adaptive designs are shown to be competitive with respect to the employment of TCC's since they effectively track frequency variations at smaller duty-cycles, while they filter out clock noise at larger duty cycles. Also, when communication activity is extremely low, optimal loop parameters do not depend on the transceiver's duty-cycle. The results of this chapter are contained in

- N. Varanese, U. Spagnolini, Y. Bar-Ness, *"Synchronization tracking in wireless sensor networks with low duty cycles"*, in preparation for submission to IEEE Trans. on Communications.

Chapter 7: This chapter summarizes the results of the thesis and concludes by suggesting possible future extensions to the presented work.

CHAPTER 2

SYSTEM MODEL

This chapter introduces general assumptions about the clock and synchronization network models employed throughout the subsequent chapters of the thesis. Furthermore, algebraic tools are presented in order to describe complex graphs. These tools will be instrumental in the analysis of the syncrhonziation algorithms under consideration.

2.1 Oscillator and Clock models

An oscillator is a dynamical system, whose output is (in absence of noise sources) a periodic signal. A typical model for the oscillator output is

$$x(t) = A \sin\left(\phi\left(t\right)\right), \tag{2.1}$$

where $\phi\left(t\right)$ is the *instantaneous phase*. A general model for $\phi\left(t\right)$ is [13]

$$\phi\left(t\right) = \phi\left(0\right) + \left(1 + \alpha + Dt\right) 2\pi f_0 t + \upsilon\left(t\right), \tag{2.2}$$

where α is the *fractional frequency offset* (oscillator *accuracy*), D is the *frequency drift*, $\upsilon\left(t\right)$ is the *phase noise* random process and f_0 is the *nominal frequency*. In a general radio transceiver, an oscillator with $f_0 \simeq$ 10-30 MHz is employed as reference to generate the RF carrier and internal timing signals for the physical layer circuitry (and often also for MAC layer functions). In sensor networks, the transceiver may be turned off for long periods for energy saving purposes. In applications with low duty-cycle, the high-frequency oscillator is turned off along with the radio, and sleep timing is provided by a low-power low-frequency oscillator (typically a quartz crystal oscillator at $f_0 = 32$ kHz). Depending on the manufacturing process, at $t = 0$ a real oscillator outputs a signal at a frequency f slightly different from the nominal one, $f = f_0 + \Delta f$, where $\Delta f = \alpha f_0$. The accuracy α

Table 2.1 Clock Accuracies Required by Different Wireless Communication Systems (for the Air Interface), Including Aging and Dependence on Environmental Conditions. Requirements for Cellular Systems Refer to the Base Station Clock

Technology	**Clock accuracy**
3GPP GSM, W-CDMA (UMTS), LTE	$5 \cdot 10^{-8}$ (50 ppb), wide area 10^{-7} (0.1 ppm), medical/local area $2.5 \cdot 10^{-7}$ (0.25 ppm), home area (femtocell)
3GPP2 CDMA2000	$5 \cdot 10^{-8}$ (50 ppb)
IEEE 802.16 (WiMAX)	10^{-6} (1 ppm)
IEEE 802.11 (Wi-Fi)	10^{-4} (100 ppm)
IEEE 802.15.4 (ZigBee)	$4 \cdot 10^{-5}$ (40 ppm)
ECMA-368 (WiMedia)	$2 \cdot 10^{-5}$ (20 ppm)
WirelessHART	10^{-5} (10 ppm), recommended
Quartz Crystal Oscillator (XO)	$\simeq 10^{-4}$ (100 ppm) over industrial temperature range

is equal to the fractional oscillator frequency offset at $t = 0$, $\alpha = \Delta f / f_0$, and it is usually expressed in μs/s (or equivalently, in ppm with respect to the nominal frequency f_0). The frequency of an oscillator changes over time ($t > 0$) due external factors, mainly because of environmental temperature variations and aging. The drift D accounts for the oscillator aging and it is typically expressed in ppm/year. It is important to remark that specifications in current standards prescribe a minimum accuracy that takes temperature and aging effects into account. Typical required accuracies for oscillators employed in wireless networks are listed in Table 2.1. The strict accuracies required for the clocks of Base Stations in cellular systems are typically satisfied by employing a GPS receiver or by deriving a stable clock source from the backhauling circuit-switched network. Notice that systems employed for local area and sensor networks require much looser accuracies, in the $[10, 100]$ μs/s range

(i.e., $[10^{-4}, 10^{-5}]$ μs/μs), which are achievable with a quartz crystal oscillator (XO). Phase noise $\upsilon\,(t)$ is a correlated random process due to the short-term frequency instability of the oscillator, and its power spectral density (PSD) is typically close to a power-law model, see, e.g., [13].

A time clock is composed by an oscillator and a b bits counter register. The start of a new cycle of the oscillator signal triggers an increment (a clock *tick*) of the counter register. The nominal time between two ticks is the clock granularity, or clock *precision*. In the class of protocols of interest, clock granularity ranges roughly from[1] 1 μs to 100 μs. The value of the counter at time t corresponds to the local time-scale $\tau(t)$,

$$\tau(t) = \frac{\phi\,(t)}{2\pi f_0} + Q\,(t) \tag{2.3}$$

$$= \beta + (1+\alpha)\,t + Dt^2 + \delta(t), \tag{2.4}$$

where $Q\,(t)$ is the *quantization error* due to the granularity of the counter register, $\beta = \phi\,(0)\,/\,(2\pi f_0)$ is the *initial clock phase*, and $\delta(t) = Q\,(t) + \upsilon\,(t)\,/\,(2\pi f_0)$ is the clock *jitter*, which is the sum of the effects of quantization error and phase noise. Quantization error $Q\,(t)$ is typically assumed to be a stationary white random process. In the time distribution nomenclature, it is more appropriate to denote with jitter only the white component of $\delta(t)$, while the portion of the phase noise PSD closer to the oscillation frequency (typically within 10 Hz) is referred to as frequency *wander*. Frequency wander is mainly caused by external factors such as environmental temperature variations. The quantity $\frac{d\tau(t)}{dt}$ is called either timing skew or *clock frequency*. A clock is *stable* (over a pre-defined limited time interval) if wander and drift are negligible, and therefore its frequency can be considered constant. A common model for a stable clock is

$$\tau(t) = \beta + (1+\alpha)\,t + \delta(t), \tag{2.5}$$

[1]The typical length of the clock register is between $b = 24$ bits (IEEE 802.15.4) and $b = 64$ bits (IEEE 802.11). Clock granularity can be dynamically tuned in IEEE 802.15.4, while it is 1 μs in IEEE 802.11 [24] (local time (practically) never folds).

where $\delta(t)$ is a white noise process. In the following, clock frequencies are adimensional (or, equivalently, expressed in μs/μs), unless noted otherwise.

2.2 Connectivity Models for Large Sensor Fields

The deployment of a large number of sensors on a wide area is required by specific applications such as geophysical prospecting, road and industrial structure monitoring and control, environmental and proximity control. The nodes are assumed to be deployed on a rugged bi-dimensional surface. Also, it is assumed that only line-of-sight (LOS) communication is possible and that multipath propagation is negligible due to the absence of relevant obstructions to propagation between nodes. Link quality between any pair of nodes (i, j) can be therefore modeled by the Friis transmission equation

$$P_r = P_t G_i\left(\theta_i, \phi_i\right) G_j\left(\theta_j, \phi_j\right) \left|\mathbf{a}_i \cdot \mathbf{a}_j^*\right|^2 \left(\frac{\lambda}{4\pi d_{ij}}\right)^2, \tag{2.6}$$

where P_t is the transmission power (in mW), $G_i\left(\theta_i, \phi_i\right)$ is the antenna gain in the direction (θ_i, ϕ_i) from which i sees j, $G_j\left(\theta_j, \phi_j\right)$ is the antenna gain in the direction (θ_j, ϕ_j) from which j sees i, $\mathbf{a}_i, \mathbf{a}_j$ are the polarization vectors of i and j, and λ is the wavelength at the carrier frequency. Reliable communication is possible between node i and node j if the following condition on received power is satisfied

$$10 \log_{10} P_r > \bar{\gamma}_{\mathrm{dBm}} + m_{\mathrm{dBm}}, \tag{2.7}$$

where $\bar{\gamma}_{\mathrm{dBm}}$ is the receiver sensitivity and m_{dBm} is an interference and fading margin, both expressed in dBm. If it is further assumed that antennas are matched in polarization and well-pointed, (2.6) reduces to $P_r = P_0/d_{ij}^2$, where $P_0 = P_t G^2 \left(\frac{\lambda}{4\pi}\right)^2$, and (2.7) implies

$$d_{ij} < r = P_0 10^{-\frac{\bar{\gamma}_{\mathrm{dB}} + m_{\mathrm{dB}}}{10}}, \tag{2.8}$$

where r is the *transmission range*. The condition (2.8) is the basis of the so-called *geometric connectivity* model. Whenever the assumption of ideal propagation character-

istics is verified, the ring model is a satisfying description of network connectivity as it captures the fact that a given node is able to exchange synchronization information only with its nearest neighbors.

If nodes are placed outdoor on rough terrain, the ring model becomes optimistic, and it is needed to resort to the general formula (2.6).

2.3 Synchronization Network Models

A network of K nodes is considered where each node runs an independent clock $\tau_i(t)$. From the clock model of Section 2.1, two synchronization regimes can be identified, namely *time* and *frequency* synchronization. A network is time synchronous if $\tau_1 = \tau_2 = \ldots = \tau_K = \tau_0$, where τ_0 defines the network *reference time*. A network is frequency synchronous if $\alpha_1 = \alpha_2 = \ldots = \alpha_K$, or equivalently $f_1 = f_2 = \ldots = f_K = f_0$, where f_0 defines the network *reference frequency*. Clearly time synchronization implies frequency synchronization.

It is useful to model the topology of the synchronization network as a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of order K, where $\mathcal{V} = \{1, .., K\}$ is the set of nodes (graph vertexes) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of links (graph edges). Generally, it is assumed a geometric connectivity model, whereby $(i, j) \in \mathcal{E}$ if the distance between the two nodes is smaller than the transmission radius r. The transmission radius depends on the employed transmission power, receiver sensitivity and channel model. It is assumed that the first K_u nodes are provided with independent time-scales and need to be synchronized, while the remaining $K_r = K - K_u$ nodes have access to a stable universal time reference (e.g., they are deployed with a GPS receiver). The reference (or *master*) nodes are grouped in the set $\mathcal{M} \subset \mathcal{V}$, while the rest of the nodes (*slaves*) are part of the set $\mathcal{S} = \mathcal{V} \setminus \mathcal{M}$. The *neighbors* of a slave node $i \in \mathcal{S}$ are the nodes from which i *receives* synchronization information, and the *neighbor set* is $\mathcal{N}_i = \{j | (i, j) \in \mathcal{E}\}$. Note that the neighbor set $\mathcal{N}_i$ includes both master and slave nodes. Links between slave nodes are weighted by the off-diagonal elements of

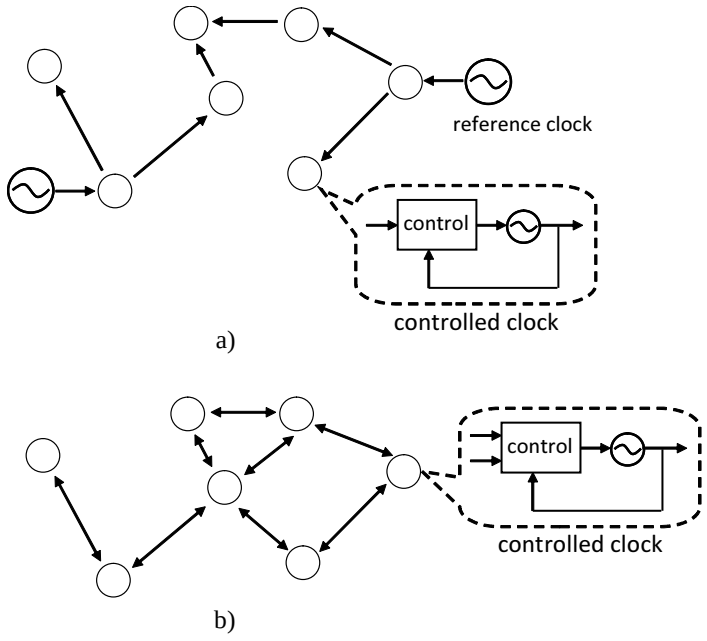


Figure 2.1 Synchronization network topologies: a) Master-Slave and b) Mutually Coupled.

the $K_u \times K_u$ adjacency matrix $[\mathbf{A}]_{i,j} = a_{ij}$ ($[\mathbf{A}]_{i,i} = 0$). If a link exists from node j to node i, $\{(i,j) \in \mathcal{E}, i,j \in \mathcal{S}\}$, the corresponding weight $a_{i,j} \neq 0$. The $K_u \times K_r$ master adjacency matrix $[\mathbf{M}]_{i,j} = m_{ij}$ is similarly defined, whose elements weight the links from a master to a slave node, $\{(i,j) \in \mathcal{E}, i \in \mathcal{S}, j \in \mathcal{M}\}$. In the following, it is assumed that master nodes have access to a stable universal time and frequency reference, so that $\beta_i = \alpha_i = 0$ and $\tau_i(t) = t$ if $i \in \mathcal{M}$.

Topologies of synchronization networks fall into three categories:

Master Slave **(MS)**: nodes are organized in a *hierarchical* structure as in Figure 2.1.a. Master nodes occupy the top layer, while slave nodes receive synchronization information according to the predefined hierarchy. The network time and frequency reference is given by the master nodes' clocks, $\tau_k = \tau_0$, $f_k = f_0$, $k \in \mathcal{M}$. MS graphs may take the form of a tree (for a single master, $|\mathcal{M}| = 1$), or a forest (for multiple masters, $|\mathcal{M}| > 1$). Also, each node may be allowed to be coupled to multiple parents. MS topologies are widely employed for synchronization in wired communication networks (e.g., digital

circuit-switched telephone networks [13] and the Internet [17]). In fact, MS networks are easy to implement and to analyze since they are a direct extension of the single master-slave case. The main drawback of a hierarchical structure is its scarce resilience to node failures: if the master node of a tree network breaks down, synchronization is irremediably lost. In the following, it is always assumed that there are no isolated nodes in MS graphs (i.e., the master(s) can reach any node in the network).

***Mutually Coupled* (MC)**: the network is deployed without any master node, $|\mathcal{M}| = 0$, and slave nodes are coupled in a *peer-to-peer* fashion as in Figure 2.1.b. MC topologies have been first proposed in the 70's by Lindsey [15], because they improve network synchronization stability and are highly robust to node failures. Only recently, MC architectures have attracted widespread interest for their applicability to wireless networks, where the availability of a decentralized and robust synchronization mechanism could be vital. In a MC setting, there is no universal reference and each node influences the synchronization process. Synchronization occurs when all nodes agree on the same reference time-scale τ_0 and frequency f_0. In general, when employing synchronization algorithms over a MC network, τ_0 and f_0 depend on the characteristics of each clock and on the underlying graph topology. In many cases, it is seen that the reference time and frequency can be suitably defined as the node-wise *instantaneous average* time and frequency $\tau_0(t) = \frac{1}{K}\sum_{i=1}^{K}\tau_i(t)$, $f_0(t) = \frac{1}{K}\sum_{i=1}^{K} f_i(t)$. In general, the definition depends on the specific network topology at hand and needs to be restated for each case. Any MC network reaches network synchronization from any initial state if the underlying directed graph $\mathcal{G}$ is *strongly connected*[2] (*sufficient* condition) [48]. In the following, MC graphs are always assumed to be strongly connected.

Hybrid: the network is deployed with some master nodes, $|\mathcal{M}| \neq 0$, but slave nodes are still coupled in a *peer-to-peer* fashion as in the MC case. This network architecture was first proposed in [49] and comprises the previous ones as special cases. It retains the

[2]A directed graph is said to be strongly connected if there exist a directed path (i.e., a collection of edges in $\mathcal{E}$) connecting *any* pair of nodes in the graph.

robustness of MC thanks to the peer-to-peer coupling while allowing for universal time and frequency distribution.

2.4 Algebraic Description of Synchronization Networks

One of the purposes of this thesis is to analyze the performance of distributed synchronziation algorithms in relation with the specific network topology. To this end, the synchronization network can be mathematically described by means of the algebraic tools hereafter introduced. First of all, let the *in-degree* of node k be defined as the sum of the weights of all incoming links, $d_i = \sum_{j=K_u+1}^{K} m_{ij} + \sum_{j=1}^{K_u} a_{ij}$. The $K_u \times K_u$ *Laplacian* matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ [50], where$\mathbf{A}$ is the adjacency matrix previously introduced in Section 2.3, and $\mathbf{D}$ is the diagonal matrix of in-degrees, $\mathbf{D} = \mathrm{diag}\,(d_1, d_2, \ldots, d_{K_u})$. If the graph has symmetric weights, $a_{ij} = a_{ji}$, $\mathbf{L}$ is symmetric. In network synchronization, it is common to make use of the *normalized Laplacian*, defined as $\mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$, where $\mathbf{D}^{-1}\mathbf{A}$ is recognized as the transition matrix of the natural random walk over the graph $\mathcal{G}$ [51]. The eigenvalue spectra of these matrices is particularly relevant for the analysis of synchronization algorithms. It is easily verified that the normalized Laplacian is similar to the symmetric matrix[3] $\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$. Therefore, both $\mathbf{L}$ and $\mathbf{D}^{-1}\mathbf{L}$ have real eigenvalues. By Gershgorin's theorem, it can be verified that the eigenvalues of both matrices are also nonnegative, and that the spectrum of the normalized Laplacian is comprised within the interval $0 \leq \lambda_k(\mathbf{D}^{-1}\mathbf{L}) \leq 2$, while for the Laplacian it is $0 \leq \lambda_k(\mathbf{D}^{-1}\mathbf{L}) \leq d_{\max}$, $d_{\max}$ being the largest node degree. Finally, if $\mathcal{G}$ is a *regular* graph, i.e., if all nodes have the same degree, $d_1 = d_2 = \ldots = d_{K_u} = d$, the normalized Laplacian is symmetric as well $\mathbf{D}^{-1}\mathbf{L} = \frac{1}{d}\mathbf{L}$.

In the case of MS networks, $\mathbf{L}$ and $\mathbf{D}^{-1}\mathbf{L}$ are full-rank and can be turned into a triangular form by permutation of columns and rows. In particular, it can be shown that the permutation operation does not change the elements on the main diagonal, and therefore

[3]Notice that the matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$ is also often referred to as the normalized Laplacian.

the eigenvalues of $\mathbf{L}$ coincide with node degrees, while the eigenvalues of $\mathbf{D}^{-1}\mathbf{L}$ are all equal to 1. In the case of tree graphs, the permutation turns $\mathbf{L}$ and $\mathbf{D}^{-1}\mathbf{L}$ directly into a Jordan block form, and the two matrices cannot be diagonalized. This property reflects the fact that MS networks arise by connecting dynamical systems in a cascade fashion.

In the case of MC networks, $\mathbf{L}$ and $\mathbf{D}^{-1}\mathbf{L}$ are singular as it holds (by construction) $\mathbf{L}\mathbf{1} = (\mathbf{D} - \mathbf{A})\,\mathbf{1} = \mathbf{0}$, where $\mathbf{1}$ is the all-ones vector. This property also implies that $\mathbf{1}$ is the right eigenvector associated with the zero eigenvalue for both $\mathbf{L}$ and $\mathbf{D}^{-1}\mathbf{L}$. If link weights are symmetric, $\mathbf{1}^T\mathbf{L} = \mathbf{0}$, i.e., $\mathbf{1}$ is also the left eigenvector for the zero eigenvalue of $\mathbf{L}$. For the normalized Laplacian, instead, it is easily verified that the left eigenvector depends on node degrees, in particular it is $\mathbf{v}^T\mathbf{D}^{-1}\mathbf{L} = \mathbf{0}$ where $[\mathbf{v}]_i = d_i / \left(\sum_i d_i\right)$. If the graph is connected, the zero eigenvalue is unique and the Laplacian can be diagonalized by eigenvalue decomposition (EVD). Let the EVD of $\mathbf{L}$ and $\mathbf{D}^{-1}\mathbf{L}$ be defined as $\mathbf{L} = \mathbf{V}\mathbf{B}\mathbf{V}^{-1}$ and $\mathbf{D}^{-1}\mathbf{L} = \mathbf{U}\mathbf{C}\mathbf{U}^{-1}$, respectively. The eigenvectors of $\mathbf{D}^{-1}\mathbf{L}$ correspond to the *generalized* eigenvectors of $\mathbf{L}$ and $\mathbf{D}$, and it is $\mathbf{U} = \mathbf{D}^{-\frac{1}{2}}\mathbf{V}$ since $\mathbf{D}$ is positive-definite [52]. The nonzero eigenvalues of the Laplacian spectra depend on graph properties, and in particular the second smallest eigenvalue $\lambda_2(\mathbf{L})$ reflects network *connectivity* (the better the network is connected the larger is $\lambda_2(\mathbf{L})$). The next section provides some insights on the properties of the spectrum of MC graphs.

2.5 The Spectrum of MC Networks

The computation of the Laplacian spectrum poses in general insurmountable difficulties, except in few special cases, such as regular graphs. The graph of an MC network is *regular* if all nodes have the same number of neighbors, or equivalently the same degree $d_1 = d_2 = \ldots = d_{K_u} = d$. The simplest example of a regular graph is the ring network. This section intends to show that the spectrum of a regular graph can indeed approximate the spectrum of a general graph with similar local connectivity properties. In particular, the focus is on a regular deployment, whereby nodes are located at unitary distance from the nearest

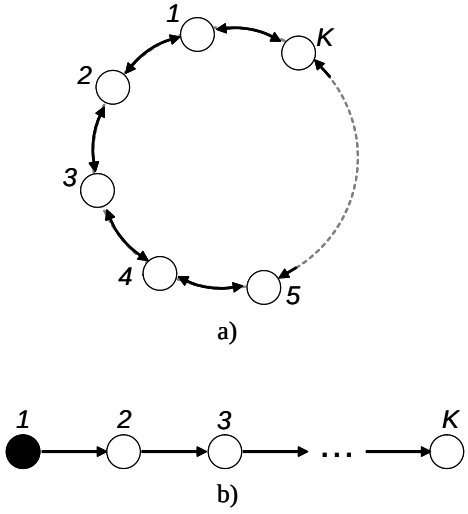


Figure 2.2 Example topologies: a) mutually coupled (MC) ring and b) master slave (MS) line networks of K nodes. The distance between nodes is normalized to unity and the transmission radius is $r = 1$ (nearest neighbor connectivity).

neighbor (unitary node density). Consider first a (regular) ring network of K nodes as in Figure 2.2.a. In this case, the Laplacian matrix $\mathbf{L}_K^{(r)}$ is a circulant Toeplitz matrix, defined by its first row $\mathbf{c}$. The eigenvalues of a circulant matrix are the DFT coefficients of $\mathbf{c}$ [53],

$$\lambda_i(\mathbf{L}_K^{(r)}) = \sum_{n=0}^{K-1} c_n e^{-j2\pi \frac{i}{K} n}. \tag{2.9}$$

If each node can reach d neighbors ($d_k = d$) with unitary edge weights, i.e., $r = d/2$ and $[\mathbf{A}]_{ij} = 1$ if $|i-j| \leq N$ and $[\mathbf{A}]_{ij} = 0$ otherwise, from (2.9) the Laplacian spectrum is found to be

$$\lambda_i(\mathbf{L}_K^{(r)}) = \left(1 + \frac{1}{d}\right) - \frac{1}{d} \frac{\sin\left(\pi\,(d+1)\,\frac{(i-1)}{K}\right)}{\sin\left(\pi \frac{(i-1)}{K}\right)} \tag{2.10}$$

for $i = 1, \ldots, K$. As expected, the smallest eigenvalue is $\lambda_1(\mathbf{L}_K^{(r)}) = 0$. The second smallest eigenvalue, which is an index of network connectivity, is

$$\lambda_2(\mathbf{L}_K^{(r)}) = \left(1 + \frac{1}{d}\right) - \frac{1}{d} \frac{\sin\left(\pi \frac{(d+1)}{K}\right)}{\sin\left(\pi \frac{1}{K}\right)}, \tag{2.11}$$

and it is proportional to node degree d and inversely proportional to network size K. If the degree is let grow with the network size, $d = \alpha K$, $\lambda_2(\mathbf{L}_K^{(r)})$ is still decreasing for increasing K, but for an infinite network size it converges to $\lambda_2(\mathbf{L}_\infty^r) = 1 - \sin(\pi\alpha)/(\pi\alpha)$. By approximating the sine as $\sin(x) \simeq x - x^3/6$, (2.1) simplifies to

$$\lambda_2(\mathbf{L}_K^{(r)}) \simeq \frac{\pi^2}{6}\left(\frac{d}{k}\right)^2\left(1+\frac{3}{d}\right). \tag{2.12}$$

Now consider a (non regular) line network of K nodes (Figure 2.2.b) employing the same transmission range r and associated Laplacian matrix $\mathbf{L}_K^{(l)}$. In this case the nodes can not have all the same degree, as the edge nodes have no neighbors on one side. It can be shown that, for growing network size, i.e., $K \to \infty$, the sequence of matrices $\mathbf{L}_K^{(l)}$ is *asymptotically equivalen*t to $\mathbf{L}_K^{(r)}$,

$$\lim_{K\to\infty} \frac{1}{\sqrt{K}} \left\| \mathbf{L}_K^{(l)} - \mathbf{L}_K^{(r)} \right\|_{\mathrm{F}} = 0, \tag{2.13}$$

where $\|\mathbf{B}\|_{\mathrm{F}}$ is the Frobenius norm of $\mathbf{B}$, and therefore their eigenvalues are *asymptotically equally distributed* [53]. This property guarantees that the eigenvalues of $\mathbf{L}_K^{(l)}$ and $\mathbf{L}_K^{(r)}$ will behave similarly in large networks, as it is illustrated in Figure 2.4, where the ordered spectra of a line and ring network are plotted for increasing K and $r = 5$. Notice that, due to the properties of the Laplacian matrix, $\lambda_1(\mathbf{L}_K^{(l)}) = \lambda_1(\mathbf{L}_K^{(r)}) = 0$ for any network size K. Clearly, the asymptotic equidistribution property does not imply that the two spectra will converge pointwise. In fact, it is always true that a ring network has a better connectivity than a line network, $\lambda_2\left(\mathbf{L}_K^{(r)}\right) > \lambda_2\left(\mathbf{L}_K^{(l)}\right)$.

If the network size is fixed and the transmission radius r is increased, instead, the Laplacian spectrum would converge in both cases to $\lambda_k\left(\mathbf{L}_K^{(l)}\right) = \lambda_k\left(\mathbf{L}_K^{(r)}\right) = 1 + \frac{1}{K+1}$, i.e., the spectrum of a network with full all-to-all connectivity. The spectrum convergence is plotted in Figure 2.4 for a line and ring network of $K = 50$ nodes. It can be seen that small eigenvalues converge more slowly than the rest.

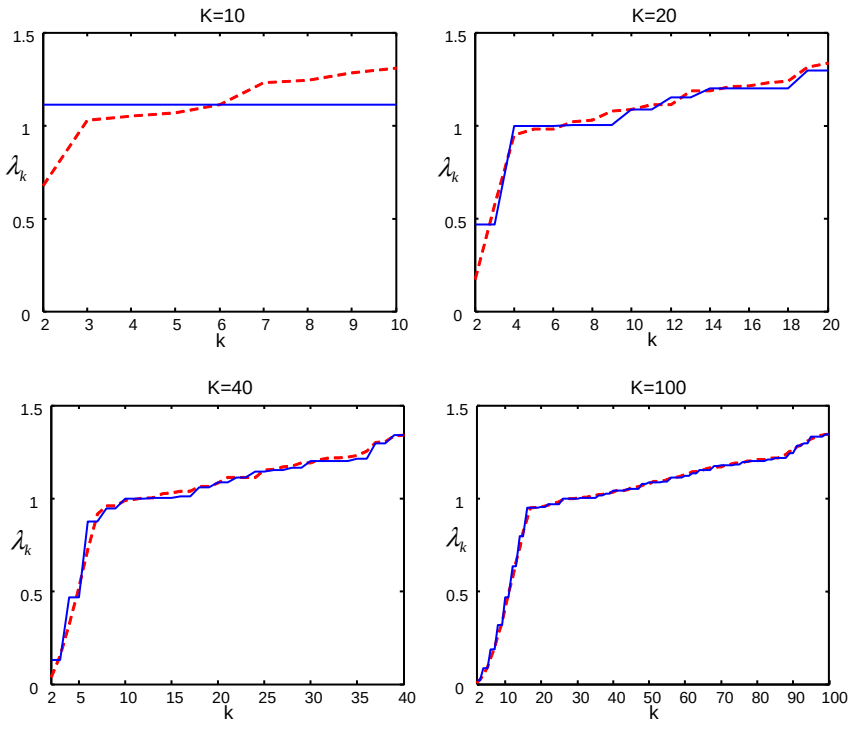


Figure 2.3 Laplacian eigenvalue spectra of a line (dashed line) and ring (solid line) network, $r = 5$.

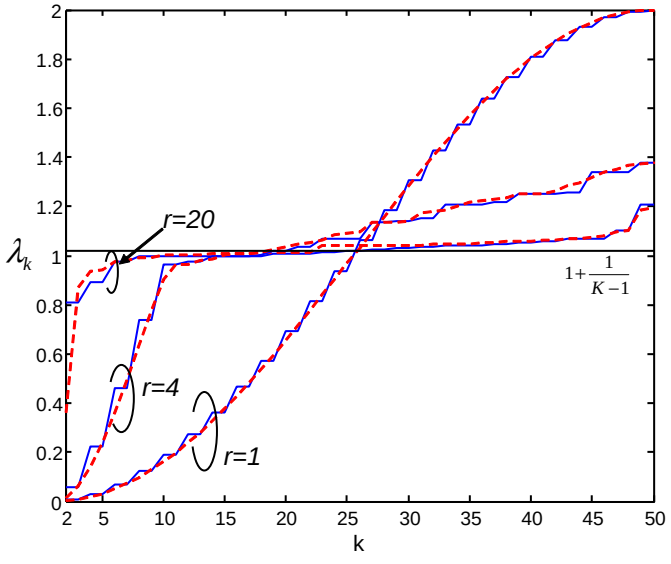


Figure 2.4 Laplacian eigenvalue spectra of a line (dashed line) and ring (solid line) network, $K = 50$.

It can be shown that the same properties hold also for the bidimensional case, whereby nodes are deployed over a planar (2D lattice) or toroidal domain (2D regular graph). Also, it is conceivable to extend the asymptotic equivalence in (2.13) from deterministic to *random* geometric graphs. In fact, it is known that, for a sufficiently large transmission radius r, a random geometric graph is regular with high probability [54]. As shown in [55], the transition matrix of a natural random walk over a random geometric and a regular deterministic graph are asymptotically equivalent as long as the random graph is connected with high probability.

Part I

Synchronization at the Physical Layer

CHAPTER 3

DISTRIBUTED CARRIER FREQUENCY SYNCHRONIZATION

At the physical layer, cooperative communication techniques require independent nodes to generate RF signals with the same carrier frequency. In the analysis of virtual antenna arrays (VAA) systems (e.g., those employing distributed space-time codes - DSTC), cooperating nodes are typically assumed to be frequency synchronous. In practice, the limited accuracy and stability of real oscillators defies this assumption, and the carrier frequency of a given transceiver is different from the nominal one and continuously time-varying. Therefore, distributed synchronization techniques are needed in order to compensate and track carrier frequency offsets (CFO) when employing cooperative communication schemes.

Most of the work in the area of network synchronization has dealt with synchronization at higher layers via transmission and reception of signaling packets carrying timestamps. Inspired by the pulse-coupling mechanism in IF oscillators, [42] first proposed to realize time synchronization at the physical layer, through the exchange of periodic RF pulses. The protocol presented in [43], while exploiting pulse-coupling principles, it employs a more refined clock controller based on a second-order type 1 DPLL, generalizing classical work on networks of analogue PLL's [15]. Despite its importance, the feasibility of distributed frequency synchronization algorithms at the physical layer has not been carefully investigated yet. Recently, [56] has studied the problem of estimating and adjusting the carrier frequencies in a two-transmitters-one-receiver system employing distributed Alamouti space-time code. The algorithm cannot easily scale to a higher number of nodes as it relies on the assignment of specific pilot sequences to each node.

This chapter proposes a distributed approach to frequency synchronization in a wireless network which is based on frequency-locking principles: each node controls the

frequency of its clock by a frequency-locked loop (FLL) while the clock phase is arbitrary. A signaling channel is considered, whereby nodes exchange synchronization information with neighbors by transmission of unmodulated pilot tones. There is no need to provide the nodes with different pilot sequences. The local frequency correction is based on a weighted sum of frequency offsets as a result of the combination of the received signals operated by a suitably designed frequency difference detector (FDD). It is shown that a network of Distributed FLL's (D-FLL's) is flexible enough to acquire frequency synchronization with both MS or MC architectures (with a proper setting of FLL parameters). In particular, for a sufficiently high SNR, the algorithm stability is not impaired by either pilot length or phase noise. After acquisition, the network of multiple D-FLL's tracks frequency instabilities and channel noise by acting as a frequency-selective filter. It is proved that MC architecture is equivalent to the parallel connection of FLL's, and that provides a smooth distribution of synchronization error among nodes. On the other hand, MS architectures correspond to a series connection of FLL's and imply error accumulation along the chain.

The chapter is organized as follows. Section II presents a FDD design that is able to compute a frequency correction from the superposition of pilot tones. Section III analyzes the conditions under which a D-FLL network equipped with the proposed FDD is able to acquire frequency synchronization. Steady-state tracking accuracy is analyzed in Section IV for general topologies by taking into account both channel noise and the frequency instability of local oscillators. The results are specialized in Section IV-A for a MC ring network, and in Section IV-B for a MS line network. The frequency acquisition rate of a MC network equipped with the proposed FDD is compared in Section V with an alternative detector design by simulations. The accuracy of frequency synchronization with MC and MS architectures is compared for a line network. Finally, Section VI concludes the chapter.

3.1 System Model

3.1.1 Distributed Frequency-Locked Loops (D-FLL)

Each node employs a frequency-locked loop (FLL) to control its clock frequency dynamically. A FLL is a discrete-time dynamic system whose state variable is the local frequency $f_k[n]$. A D-FLL is designed so that $f_k[n]$ is controlled by the weighted sum of frequencies of neighboring nodes $\sum_{i\neq k} p_{k,i} f_i[n]$, where $\sum_{i\neq k} p_{k,i} = 1$. A common and practical choice for the weights is $p_{k,i} = a_{k,i}/d_k$, where $a_{k,i}$ is the weight for the edge (k,i) in the topology graph $\mathcal{G}$ and d_k is the degree of node k: $d_k = \sum_{i\neq k} a_{k,i}$. The local frequency error $\bar{e}_k[n] = \left(\sum_{i\neq k} p_{k,i} f_i[n]\right) - f_k[n]$ follows from the frequency difference detector (FDD):

$$\bar{e}_k[n] = \frac{1}{d_k} \sum_{i\neq k} a_{k,i} \left(f_i[n] - f_k[n]\right). \tag{3.1}$$

The frequency of the k-th node is updated with the linear correction

$$f_k[n+1] = f_k[n] + \epsilon \bar{e}_k[n], \tag{3.2}$$

the parameter ϵ is the *loop gain*. The update rule (3.2) is a first-order discrete-time FLL as any memory in the loop is not necessary for frequency locking [6]. At equilibrium, $f_k[n+1] = f_k[n]$ and the error is $\bar{e}_k[n] = 0$. The frequency synchronization network under analysis is composed of multiple *distributed* FLL's (D-FLL's) whose connections are described by the topology graph $\mathcal{G}$. In the following, D-FLL's are employed to achieve frequency synchronization in a network of nodes with radio communication capabilities. Recall that frequency is a parameter embedded in a pilot tone transmitted by each node according to a synchronization protocol as described below.

3.1.2 Synchronization Protocol

It is assumed that a common network time-scale is maintained by implementing a suitable network time synchronization algorithm, see, e.g., [42][43] and the survey [35]. Therefore, the time axis can be divided into frames of duration T_F seconds as in Figure 3.1. Each

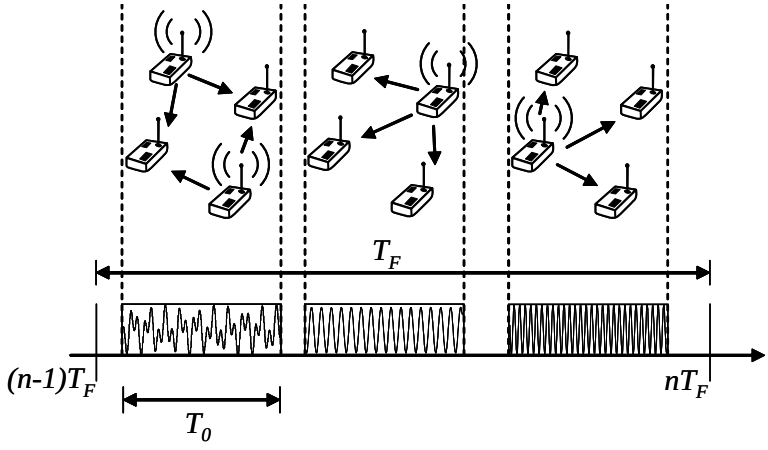


Figure 3.1 Frame structure and network operation.

frame corresponds to one D-FLL iteration, and comprises a certain number of time-slots each of duration T_0 seconds. In every time-slot, each node may either transmit, receive or sleep (i.e., turn off its transceiver for energy saving purposes). When receiving during a time-slot, a node computes the frequency error with respect to transmitting neighbors. At the end of the frame, the frequency offsets are combined as in (3.1) in order to obtain the frequency correction $\epsilon\bar{e}_k[n]$ in (3.2) used for the next frame. To simplify, nodes are assumed to be provided with *full duplex* capabilities and to have their transceiver always turned on. Given this assumption, each node both transmits and receives at the same time, and there is no need for more than one time-slot per frame (i.e., $T_F = T_0$). This assumption simplifies the analysis without affecting in any way applicability and conclusions to more general medium access protocols (see, e.g., [57]).

The base-band model of the synchronization signal transmitted by node k during the n-th frame is

$$x_k(t;n) = e^{j(2\pi f_k[n]t+\phi_k[n])}, \tag{3.3}$$

where $\phi_k[n]$ is the carrier phase (the time-slot length is assumed large enough to neglect the modulating pulse). When employing BPSK modulation, the signal (3.3) corresponds to a long sequence of 1's, i.e., all nodes employ the same pilot sequence. Notice that

the local frequency $f_k[n]$ is controlled by the D-FLL, while the phase $\phi_k[n]$ is a random process due to the oscillator phase noise. In the following, $\phi_k[n]$ is assumed to be stationary and uniformly distributed in $[-\pi, \pi)$. The wireless channel between any pair of nodes (i, k) is modeled as a time-invariant complex scalar gain $h_{k,i} = |h_{k,i}|\, e^{j\psi_{k,i}}$. The channel amplitude incorporates the transmission power P_T (the same for all nodes) and it is inversely proportional to the geometric distance between nodes $D_{k,i}$: $|h_{k,i}|^2 = P_T D_{k,i}^{-\alpha}$, reciprocity holds with $|h_{k,i}| = |h_{i,k}|$. From (3.3), it is clear that the local carrier frequency $f_k[n]$ is a parameter embedded in the pilot signal $x_k(t; n)$. In the next section, a suitable frequency difference detector is designed in order to compute the desired error signal $\bar{e}_k[n]$ of (3.1) from the received signal samples.

3.2 Design of the Frequency Difference Detector

The FDD has to approximate the frequency error estimate (3.1) in order to implement D-FLL's under the assumptions on network and node operation of Section 3.1. At node k, the received RF signal is down-converted to base-band by mixing with the current frequency $f_k[n]$ and sampled at frequency $1/T_s$. As a consequence, the received sampled base-band signal in the n-th time-slot is

$$y_k\left(mT_s\right) = \sum_{i \neq k} |h_{k,i}| e^{j\left(2\pi(f_i - f_k)mT_s + \phi_{k,i}\right)} + z_k\left(mT_s\right), \tag{3.4}$$

for $m = 0, \ldots, L-1$, where the total number of samples within a time-slot is $L = T_0/T_s$, $z_k\left(mT_s\right) \sim \mathcal{CN}\left(0, N_0/T_s\right)$ are additive white Gaussian noise (AWGN) samples, and $\phi_{k,i}$ is the overall phase offset between the i-th and k-th node. Due to the uniform distribution of ϕ_i, $\phi_{k,i}$ is also uniform on $[-\pi, \pi)$.

The error to be used by the D-FLL is (Appendix 3.A)

$$\bar{e}_k = \frac{1}{4\pi T_s \tilde{r}_y\left(0\right)} \mathrm{Im}\left\{\tilde{r}_y\left(T_s\right) - \tilde{r}_y\left(-T_s\right)\right\}. \tag{3.5}$$

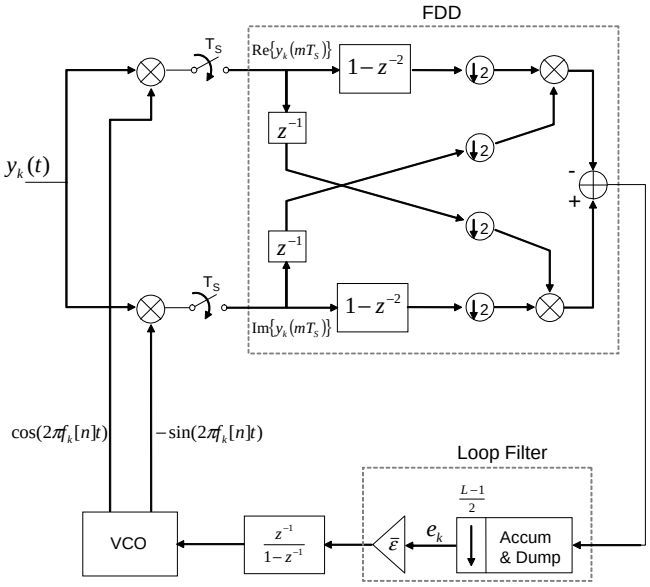


Figure 3.2 Block diagram for a FLL employing a Digital Balanced Quadricorrelator (DBQC) as detector; $y_k(t) = \sum_{i \neq k} |h_{k,i}| \cos(2\pi f_i[n]t + \phi_{k,i}) + z_k(t)$ is the RF received signal, while the normalized loop gain $\bar{\epsilon} = \epsilon / (2\pi(L-1)T_s\tilde{r}_y[0])$. VCO stands for Voltage-Controlled Oscillator (VCO).

where $\tilde{r}_y(lT_s)$ is the (unbiased) sample autocorrelation of $y_k(mT_s)$. For L odd ($L \geq 3$), (3.5) reduces to

$$\bar{e}_k = \frac{1}{2\pi(L-1)T_s\tilde{r}_y(0)} \sum_{i=0}^{\frac{L-3}{2}} \mathrm{Im}\left\{ \Big(y_k((2i+2)T_s) - y_k((2i)T_s) \Big) y_k^*((2i+1)T_s) \right\}, \tag{3.6}$$

with $\tilde{r}_y(0) = \frac{2}{L-1} \sum_{i=0}^{\frac{L-3}{2}} |y_k((2i+1)T_s)|^2$. Notice that the detector (3.6) processes the aggregated received signal samples $y_k(mT_s)$ directly, and it does not separate the different contributions in any way. By substituting $y_k(mT_s) = \mathrm{Re}\{y_k(mT_s)\} + j\mathrm{Im}\{y_k(mT_s)\}$ in (3.6), it is possible to see that the FDD (3.5) may be implemented as in Figure 3.2. In particular, the scheme in Figure 3.2 is a *Digital* Balanced Quadricorrelator (DBQC). Analog BQC [58] have been employed in the past for frequency offset estimation with analog and digital modulations (in the latter case when timing synchronization was not available). The frequency error (3.1) could also be evaluated by using DFT-based spectral methods. DFT-based techniques achieve coarse frequency synchronization, as their frequency resolution is limited by the number of available samples [59], as shown

by simulation results in Section 3.5. Other techniques, such as Fitz and L&R frequency estimators [58], could be employed as FDD, at the price of a higher implementation complexity.

In the next section, it is shown under which conditions the frequency correction computed by the DBQC detector induces frequency synchronization in a network of D-FLL's.

3.3 Frequency Acquisition

The transient phase from start-up to network synchronization (up to a prescribed accuracy) is usually dubbed frequency *acquisition*. The D-FLL network is said to be *asymptotically stable* if, in the absence of noise, $z_k\left(mT_s\right) = 0$, it can reach the synchronization state as $n \to \infty$, $f_1[\infty] = f_2[\infty] = \ldots = f_K[\infty] = f_0$, for any initial condition $f_k[0]$. Given the signal model (3.4) and the frequency difference detector (3.5), this section studies the stability of frequency acquisition in a D-FLL network. Let the *frequency spread* be defined as the maximum pair-wise frequency offset, $\Delta f = \max_{i,j} |f_i - f_j|$. At start-up, the frequency spread may be quite large (typically up to $0.1/T_s$), and it is progressively decreased by the action of D-FLL's. By plugging the input signal expression (3.4) in (3.6), after some algebra the error (3.5) becomes

$$\bar{e}_k[n] = \frac{1}{2\pi T_s d_k[n]} \sum_{i \neq k} a_{k,i}[n] \sin\Big(2\pi \left(f_i[n] - f_k[n]\right) T_s\Big), \tag{3.7}$$

where the weight $a_{k,i}[n] = |h_{k,i}|^2 + b_{k,i}[n]$ for the edge (k,i) depend on deterministic attenuation $|h_{k,i}|^2$ and the random term $b_{k,i}[n]$,

$$b_{k,i}[n] = \frac{|h_{k,i}|}{L} \sum_{l \neq k,i} \left[|h_{k,l}| \sum_{m=0}^{\frac{L-3}{2}} \cos\Big(2\pi \left(f_i[n] - f_l[n]\right)(2m+1) T_s + \phi_{k,i}[n] - \phi_{k,l}[n]\Big) \right]. \tag{3.8}$$

The randomness of $b_{k,i}[n]$ is due to the random phase offsets $\phi_{k,i}[n]$, $\phi_{k,l}[n]$. In particular, $E\left[b_{k,i}[n]\right] = 0$, and $b_{k,i}[n] \to 0$ as the number of samples $L \to \infty$. As before, the node

degree $d_k[n]$ is defined as $d_k[n] = \sum_{i \neq k} a_{k,i}[n]$. Comparing (3.1) with (3.7)-(3.8), it can be seen that the actual detector characteristic (or S-curve [58]) is nonlinear and that the edge weights contain a random component due to the superposition of multiple signals at the detector input. In the following, it is shown that, under appropriate conditions, the detector nonlinearity and the randomness of the edge weights do not impair the stability of the frequency acquisition process.

As discussed in Section 2.3, Master-Slave (MS) synchronization networks are equivalent to the connection of D-FLL's in a hierarchical fashion. In a chain network, for example, each node adjusts its frequency so as to follow the frequency of the preceding node. Intuitively, if each FLL is stable, the whole network will be stable and each node will ultimately acquire the frequency of the master node(s). In the following, the focus will be on the stability of Mutually Coupled (MC) networks, whose analysis is more involved. Furthermore, the stability of a single FLL (and thus of a whole MS network) may be studied by specializing the following considerations. Two limiting cases are considered in which the stability of frequency acquisition in MC networks can be characterized.

Large observation window: If $L \to \infty$ (or $T_0 \to \infty$), each D-FLL is a *deterministic nonlinear dynamical system* of the form

$$f_k[n+1] = f_k[n] + \epsilon \frac{1}{2\pi T_s d_k} \sum_{i \neq k} a_{k,i} \sin\Big(2\pi \left(f_i[n] - f_k[n]\right) T_s\Big), \quad k = 1, \ldots, K, \quad (3.9)$$

with $a_{i,k} = |h_{k,i}|^2$. The periodicity of the S-curve could cause spurious effects, possibly leading a node to lock on a frequency that is outside the system bandwidth (*false lock* event). The maximum frequency spread $\Delta f_{\max}$ that guarantees no false locks in the network is defined as the *locking range*. In [60] it was noticed that, for a MC topology with all-to-all connectivity, a sufficient condition to avoid false locks in (3.9) is $f_i[0] \in (\gamma - 1/4T_s, \gamma + 1/4T_s)$, where γ is an arbitrary frequency. Therefore, the locking range of the DBQC detector for an all-to-all topology is $\Delta f_{\max} = 1/2T_s$. In a classical single-master/single-slave system Fitz and L&R detectors have maximum locking range

$\Delta f_{\max} = 1/2T_s$ and $1/T_s$, respectively [58]. Other results on stability of a system analogous to (3.9) can be found in [61][60].

Large sample rate: For large sample rate as compared to the frequency spread, $1/T_s \gg \Delta f$, (3.7) can be linearized as

$$f_k[n+1] = f_k[n] + \frac{\epsilon}{d_k[n]} \sum_{i \neq k} a_{k,i}[n] \left(f_i[n] - f_k[n] \right), \quad k = 1, \ldots, K. \tag{3.10}$$

Notice that without any assumption on the number of available samples L, the weights $a_{k,i}[n]$ are random and time-varying as in (3.7). Let the *normalized graph Laplacian* be defined as $\mathbf{L}[n] = \mathbf{I} - \mathbf{D}[n]^{-1}\mathbf{A}[n]$, where the adjacency matrix $\mathbf{A}[n]$ was defined in Section 2.3, and $\mathbf{D}[n] = \operatorname{diag}\left(d_1[n], d_2[n], \ldots, d_K[n]\right)$ is the diagonal degree matrix. By defining the vector containing the frequencies of all nodes as $\mathbf{f}[n] = [f_1[n], ..., f_K[n]]^T$, (3.10) becomes

$$\mathbf{f}[n+1] = \mathbf{f}[n] - \epsilon \mathbf{L}[n]\mathbf{f}[n] = \mathbf{W}_\epsilon[n]\mathbf{f}[n]. \tag{3.11}$$

The system matrix $\mathbf{W}_\epsilon[n] = \mathbf{I} - \epsilon\mathbf{L}[n]$ is a i.i.d random matrix process with unitary row sums $\mathbf{W}_\epsilon[n] \cdot \mathbf{1} = \mathbf{1}$. This property guarantees that the synchronization state $\mathbf{f}[n] = f_0\mathbf{1}$ is a fixed point of the random iteration (3.11). The system (3.11) is a *random linear* dynamical system that has been widely investigated in stochastic control theory [62]. Given the reference frequency for a MC topology $f_0[n] = \frac{1}{K}\mathbf{1}^T\mathbf{f}[n]$, the vector of synchronization errors with respect to the reference $f_0[n]$ is $\mathbf{\Delta}[n] = (\mathbf{I} - \mathbf{J})\,\mathbf{f}[n]$, where $\mathbf{J} = \frac{1}{K}\mathbf{1}\mathbf{1}^T$. It can be shown that the dynamic of $\mathbf{\Delta}[n]$ is ruled by the recursion

$$\mathbf{\Delta}[n] = \mathbf{P}_\epsilon[n-1]\mathbf{\Delta}[n-1], \tag{3.12}$$

where $\mathbf{P}_\epsilon[n] = \mathbf{W}_\epsilon[n] - \mathbf{J}\mathbf{W}_\epsilon[n]$. A suitable Lyapunov potential function for the system (3.12) has been derived in [62], proving that a sufficient condition for almost sure stability (i.e., convergence with probability 1) is that the eigenvalues of the matrix

$$\mathbf{\Pi}_\epsilon = E\left[\mathbf{P}_\epsilon[n] \otimes \mathbf{P}_\epsilon[n]\right] \tag{3.13}$$

be inside the unit circle, where $\otimes$ denotes the Kronecker product and the average is taken over the distribution of the phase offsets $\phi_{k,i}[n]$. Therefore, if the loop gain ϵ is chosen so that the largest eigenvalue satisfies $\lambda_1(\mathbf{\Pi}_\epsilon) < 1$, a.s. stability is guaranteed. This result is important as it means that a proper choice of the loop gain ϵ implies that the randomness of the weights $a_{k,i}[n]$ does not limit the attainable synchronization accuracy. It has been shown in [54][63] that the constraint $\lambda_1(\mathbf{\Pi}_\epsilon) < 1$ is also necessary and sufficient for convergence in the mean square sense, and that the mean square convergence rate of (3.12) depends on $\lambda_1(\mathbf{\Pi}_\epsilon)$.

3.4 Frequency Tracking

In steady-state, if the system is stable and noiseless as assumed in Section 3.3, node frequencies are kept perfectly synchronized by the action of D-FLL's so that $\mathbf{f}[n] = f_0\mathbf{1}$. In practice, noise induces frequencies to fluctuate randomly around the synchronization state. The *tracking* performance of a synchronization algorithm refers to its steady-state accuracy. This section studies D-FLL frequency tracking by accounting for two noise sources, namely channel noise introduced in Section 3.2 and the *frequency noise* of the local oscillator.

The channel noise $z_k(mT_s)$ in (3.4) undermines the accuracy of the FDD frequency estimate. In particular, the output of the FDD (3.7) during tracking may be written as

$$\bar{e}_k[n] = \frac{1}{d_k}\sum_{i\neq k} a_{k,i}\left(f_i[n] - f_k[n]\right) + w_k[n], \tag{3.14}$$

where $w_k[n]$ is the frequency estimation error due to channel noise (or *FDD noise*). In (3.14), as customary in PLL tracking analysis [58], the frequency spread is assumed to be small enough, namely $\Delta f \ll 1/T_s$, in order to neglect the detector nonlinearity. Also, a small loop gain ϵ is typically employed during tracking operation in order to reduce frequency fluctuations. Under these conditions, the random coupling term (3.8) is averaged out, and it is possible to assume the weights in (3.14) to be deterministic, namely $a_{k,i} = |h_{k,i}|^2$. The normalization factor $\tilde{r}_y[0]$ in (3.6) is substituted by the received signal power

$d_k = \sum_{i \neq k} |h_{k,i}|^2$, which can be estimated accurately by using pilot signals received over multiple frames. Given the detector (3.6), the variance of the noise term $w_k[n]$ in (3.14) can be approximated as (Appendix B)

$$\mathrm{Var}\,(w_k) \simeq \frac{1}{4\pi^2 T_s^2\,(L-1)^2}\left(\frac{N_0}{d_k T_s}\right) + \frac{1}{8\pi^2 T_s^2\,(L-1)}\left(\frac{N_0}{d_k T_s}\right)^2. \tag{3.15}$$

Not surprisingly, the noise variance is inversely proportional to the number of neighbors of node k through d_k. Also, by defining the SNR at node k as $SNR_k = d_k T_s/N_0$, it is seen from (3.15) that the noise variance is $O\,(1/L)$ at low SNR, while it is $O\,(1/L^2)$ at high SNR. If all nodes have the same degree $d_k = d$ (i.e., the underlying graph is *regular*), the FDD noise variance is independent of the node: $\mathrm{Var}\,(w_k) = \sigma_w^2$. When studying loop tracking, $w_k[n]$ is assumed to be a i.i.d. Gaussian process with variance (3.15).

Consider now the noise due to local oscillator instability. The frequency of a free-running oscillator is a random process which is usually modeled as $f_k(t) = f_c + \Delta f_k + D_k t + v_k(t)$, where f_c is the nominal oscillator frequency, Δf_k is a deterministic frequency offset, D_k is the linear frequency drift in [Hz/s]. Frequency noise $v_k(t)$ is a correlated random process due to the short-term frequency instability of the oscillator, and its power spectral density is typically close to a power-law model, see, e.g., [13]. When the local oscillator is controlled by a discrete-time FLL with updating time aligned to the frame timing T_F, frequency noise can be modeled as a discrete-time random process $v_k[n] = v_k(nT_F)$ added at the output of the NCO. If the duration of a frame T_F is much larger than the coherence time of $v_k(t)$, $v_k[n]$ can be modeled as a i.i.d. process with a Gaussian distribution, $v_k[n] \sim \mathcal{N}(0, \sigma_v^2)$ for all nodes.

Given the previous discussion, a D-FLL with noise sources may be represented as the block diagram in Figure 3.3, where the weights $p_{k,i} = a_{k,i}/d_k$, $\sum_{i \neq k} p_{k,i} = 1$ (see Section 3.1.1). The Z-transform of the *loop transfer function* from the input $\sum_{i \neq k} p_{k,i} f_i[n]$ to the output $f_k[n]$ is

$$H(z) = \frac{\epsilon z^{-1}}{1 - (1-\epsilon)\,z^{-1}}, \tag{3.16}$$

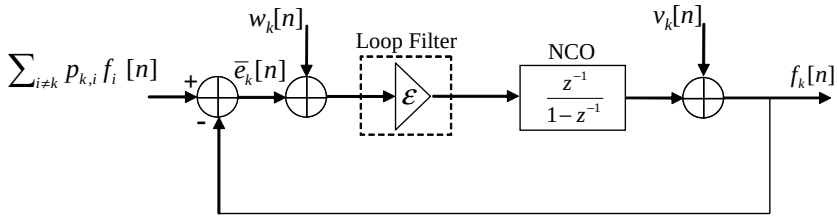


Figure 3.3 A D-FLL model with all noise sources.

Stability of (3.16) implies $\epsilon \in (0, 2)$, and $H(z)$ is low-pass if $\epsilon \in (0, 1)$, while it is high-pass for $\epsilon \in [1, 2)$ without any attenuation, thus yielding indefinite noise accumulation inside the loop. Therefore, $\epsilon \in (0, 1)$ is always a preferred choice. The *error transfer function* from the input $\sum_{i\neq k} p_{k,i} f_i[n]$ to the error $\bar{e}_k[n]$ is

$$G(z) = \frac{1 - z^{-1}}{1 - (1 - \epsilon)\, z^{-1}} = 1 - H(z). \tag{3.17}$$

This is also the transfer function from $v_k[n]$ to the output $f_k[n]$. In general $G(z)$ has a high-pass characteristic. The Z-transform of the frequency $f_k[n]$ is

$$F_k\,(z) = H\,(z) \left(\frac{1}{d_k} \sum_{i\neq k} a_{k,i} F_i\,(z) \right) + N_k(z), \tag{3.18}$$

where $N_k(z) = H\,(z)\, W_k(z) + G(z) V_k(z)$ is the equivalent noise process. Recall that if node k is a master node ($k \in \mathcal{M}$) its frequency is assumed stable and known, i.e., $f_k[n] = f_0$, and therefore $N_k(z) = 0$. Network dynamics may be described concisely in vector notation as

$$\mathbf{F}\,(z) = H\,(z)\, \mathbf{D}^{-1} \mathbf{A} \mathbf{F}\,(z) + \mathbf{N}(z), \tag{3.19}$$

where the diagonal degree matrix $\mathbf{D} = \operatorname{diag}(d_1, d_2, \ldots, d_K)$, and the adjacency matrix $\mathbf{A}$ depend on synchronization topology, and $\mathbf{F}\,(z) = [F_1\,(z)\,, F_2\,(z)\,, \ldots, F_N\,(z)]^T$. From (3.19), the matrix of cross-spectra for the vector process $\mathbf{f}\,[n]$ is

$$\mathbf{S_f}\,(z) = E\left[\mathbf{F}\,(z)\, \mathbf{F}^H\,(z)\right] = \left(\mathbf{I} - H\,(z)\, \mathbf{D}^{-1}\mathbf{A}\right)^{-1} \mathbf{S_n}\,(z) \left(\mathbf{I} - H\,(z)\, \mathbf{D}^{-1}\mathbf{A}\right)^{-H}, \tag{3.20}$$

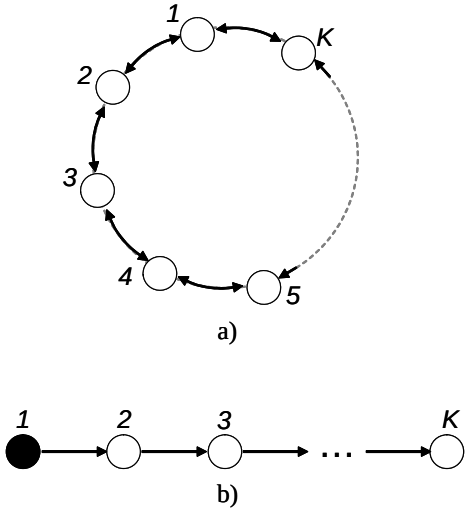


Figure 3.4 Exemplary topologies: a) mutually coupled (MC) ring and b) master slave (MS) line networks of K nodes. The distance between nodes is normalized to unity and the transmission radius is $r = 1$ (nearest neighbor connectivity).

where $\mathbf{S_n}(z)$ is a diagonal matrix with $[\mathbf{S_n}(z)]_{k,k} = 0$ if $k \in \mathcal{M}$ (i.e., k is a master node), and $[\mathbf{S_n}(z)]_{k,k} = \sigma_w^2 |H(z)|^2 + \sigma_v^2 |G(z)|^2$ if $k \notin \mathcal{M}$. Similar results can be found in [64] for a network of continuous-time PLL's. Correlation properties for any topology and loop filter depend on (3.20). This is specialized in the next subsections for Mutually Coupled (MC) ring network and a Master Slave (MS) line network (see Figure 3.4).

3.4.1 Mutually Coupled (MC) Ring Topology

In the case of MC topologies as the one in Figure 1.7.b, the analysis of the spectral matrix $\mathbf{S_f}(z)$ is quite complex due to the bi-directional connections between nodes. Nevertheless, if the network is *connected*, the *normalized adjacency matrix* $\mathbf{D}^{-1}\mathbf{A}$ may be factorized by eigenvalue decomposition, viz. $\mathbf{D}^{-1}\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$, and the vector equation (3.19) can be diagonalized

$$\mathbf{\Phi}(z) = (\mathbf{I} - H(z)\,\mathbf{\Lambda})^{-1}\,\mathbf{\Pi}(z), \tag{3.21}$$

where $\mathbf{\Phi}(z) = \mathbf{U}^{-1}\mathbf{F}(z)$ and $\mathbf{\Pi}(z) = \mathbf{U}^{-1}\mathbf{N}(z) = H(z)\,\mathbf{U}^{-1}\mathbf{W}(z) + G(z)\mathbf{U}^{-1}\mathbf{V}(z)$. By defining $\boldsymbol{\nu}[n] = \mathbf{U}^{-1}\mathbf{v}[n]$ and $\boldsymbol{\omega}[n] = \mathbf{U}^{-1}\mathbf{w}[n]$, it is possible to compute the transfer

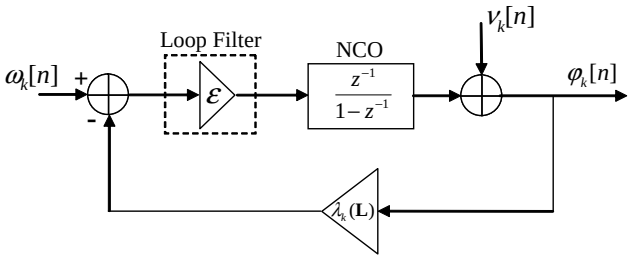


Figure 3.5 Decomposition of MC network of D-FLL's.

function for each noise component of (3.21) separately. Therefore, the dynamics of (3.21) may be described by a set of *parallel* dynamical systems as the one in Figure 3.5. In fact, the system in Figure 3.5 is similar to the D-FLL in Figure 3.3, the only difference being the scalar *feedback gain* $\lambda_k(\mathbf{L}) = 1 - \lambda_k(\mathbf{D}^{-1}\mathbf{A})$, where $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$ is the normalized Laplacian matrix. The transfer functions from FDD noise $\omega_k[n]$ and frequency noise $\nu_k[n]$ components, to the output frequency $\varphi_k[n]$ are, respectively,

$$H_k\left(z; \lambda_k\left(\mathbf{L}\right)\right) = \frac{\epsilon z^{-1}}{1 - \left(1 - \epsilon\lambda_k\left(\mathbf{L}\right)\right) z^{-1}}, \tag{3.22}$$

and

$$G_k\left(z; \lambda_k\left(\mathbf{L}\right)\right) = \frac{1 - z^{-1}}{1 - \left(1 - \epsilon\lambda_k\left(\mathbf{L}\right)\right) z^{-1}}. \tag{3.23}$$

In general, $G_k\left(z; \lambda_k\left(\mathbf{L}\right)\right) \neq 1 - H_k\left(z; \lambda_k\left(\mathbf{L}\right)\right)$. It is worthwhile to remark that for a network with full all-to-all connectivity and unitary edge weights, $a_{k,i} = 1$, the analysis simplifies as the Laplacian spectrum is $\lambda_k\left(\mathbf{L}\right) = 1 + \frac{1}{K-1}$ for $k = 2, \ldots, K$, and $\lambda_k\left(\mathbf{L}\right) \simeq 1$ for sufficiently large K. In this specific (though often impractical) case, the network effectively corresponds to K *parallel* D-FLL's.

The decomposition is particularly useful when the matrix $\mathbf{D}^{-1}\mathbf{A}$ is symmetric and the eigenvector matrix is unitary: $\mathbf{U}\mathbf{U}^H = \mathbf{I}$, with $\mathbf{U}^{-1} = \mathbf{U}^H$. This is the case of regular networks as the ring network in Figure 3.4.a, where each node has the same degree $d = 2$. Recall the definition of reference frequency for MC topologies $f_0[n] = \frac{1}{K}\mathbf{1}^T\mathbf{f}[n]$, which was introduced in Section 3.3. Since $\mathbf{D}^{-1}\mathbf{A}\mathbf{1} = \mathbf{1}$, the frequency component $\varphi_1[n] =$

$\frac{1}{\sqrt{K}}\mathbf{1}^T\mathbf{f}[n]$ is proportional to the reference frequency, $\varphi_1[n] = \sqrt{K}f_0[n]$. The associated feedback gain is $\lambda_1(\mathbf{L}) = 1 - \lambda_1(\mathbf{D}^{-1}\mathbf{A}) = 0$, and thus this component undergoes a random walk due to FDD noise, as pointed out by [65]. This fact does not constitute a serious drawback for common D-FLL applications as long as the loop gain ϵ is small enough. In order to gain insights, let the *network* mean square synchronization error be defined as

$$\xi^2 = \frac{1}{K}\sum_{k=1}^{K} E\left[\Delta_k^2[n]\right] = \frac{1}{K}E\left[\|\mathbf{\Delta}[n]\|^2\right], \tag{3.24}$$

where the synchronization error vector $\mathbf{\Delta}[n] = (\mathbf{I} - \mathbf{J})\,\mathbf{f}[n]$, with $\mathbf{J} = 1/K\mathbf{1}\mathbf{1}^T$. Since $\mathbf{U}$ is orthogonal, (3.24) may be computed as $\xi^2 = \frac{1}{K}\sum_{k=2}^{K} E\left[\varphi_k^2[n]\right]$. Standard noise analysis applied to the transfer functions (3.22)-(3.23) reveals that the synchronization error is

$$\xi^2 = \frac{1}{K}\sum_{k=2}^{K}\left[\frac{\sigma_w^2\epsilon^2}{1-(1-\epsilon\lambda_k(\mathbf{L}))^2} + \sigma_v^2\left(1 + \frac{\epsilon^2\lambda_k^2(\mathbf{L})}{1-(1-\epsilon\lambda_k(\mathbf{L}))^2}\right)\right], \tag{3.25}$$

where the Laplacian spectrum $\lambda_k(\mathbf{L})$ of a ring network with nearest-neighbor connectivity as the one in Figure 3.4.a is $\lambda_k(\mathbf{L}) = 2\left(1 - \cos\left(2\pi\frac{(k-1)}{K}\right)\right)$. The network error (3.25) depends on all the spectral components of the Laplacian except $\lambda_1(\mathbf{L}) = 0$. The error (3.25) diverges as the loop gain $\epsilon \to 2/\max_k\{\lambda_k(\mathbf{L})\}$, but the choice $\epsilon \in (0,1)$ guarantees finite output noise as $0 \leq \lambda_k(\mathbf{L}) \leq 2$ for any MC topology (see, e.g., [66]). When ϵ is small enough ($\epsilon \to 0$), error (3.25) can be approximated as

$$\xi^2 \simeq \frac{K-1}{K}\sigma_v^2 + \frac{\epsilon}{2K}\sum_{k=2}^{K}\left[\sigma_w^2\frac{1}{\lambda_k(\mathbf{L})} + \sigma_v^2\lambda_k(\mathbf{L})\right], \tag{3.26}$$

thus showing that it is possible to reduce the network mean square synchronization error to any desired value by controlling the loop gain ϵ. The first term within brackets in (3.26) (caused by FDD noise) is inversely proportional to $\lambda_k(\mathbf{L})$, while the second term within brackets (caused by frequency noise) is directly proportional to $\lambda_k(\mathbf{L})$. For a given loop gain ϵ, minimum synchronization error ξ^2 is achieved when each eigenvalue of the Laplacian spectrum $\lambda_k(\mathbf{L}) = \sigma_w/\sigma_v$. If $\sigma_w^2 = \sigma_v^2$, minimizing the synchronization error ξ^2

requires $\lambda_k(\mathbf{L}) = 1$. This can be achieved by a dense synchronization network deployed with full all-to-all connectivity. In general, increasing connectivity is beneficial for tracking performance as spectral components associated to small eigenvalues dominate in (3.26).

As a final remark, the analysis carried out here can be considered a raw approximation for more practical MC topologies. In fact, it is possible to show that grid networks on planar and toroidal domains behave similarly as the network size grows, namely for $K \to \infty$ (see the theory on *asymptotic equivalence* of matrices in [53]). Recent works (see, e.g., [54]) have also pointed out similarities between the spectra of dense random geometric networks and grid networks.

3.4.2 Master Slave (MS) Line Topology

In the case of MS topologies as the one in Figure 1.7.a, it is not possible to exploit the spectral techniques employed for the MC case. In fact, the normalized adjacency matrix $\mathbf{D}^{-1}\mathbf{A}$ cannot be diagonalized for hierarchical topologies. Nevertheless, it is still possible to follow the general analysis from (3.20). This section considers the regular MS line network in Figure 3.4.b, where the node degree is $d = 1$. The set of master nodes is $\mathcal{M} = \{1\}$, and therefore the reference frequency is $f_0 = f_1$. While an MC network is equivalent to parallel D-FLL's, the analysis for this topology corresponds to a *cascade* of D-FLL's. The error analysis follows the same lines of jitter accumulation in chains of PLL repeaters (see, e.g., [13][67]). The nodes are labelled with their distance in hops from the master node, so that node k in Figure 3.4.b has hop distance $\ell = k - 1$. In the general case of tree or forest-like topologies, ℓ is also dubbed the *layer* or *stratum index*, as it indicates the node position within the clock hierarchy with respect to the closest master [13][17]. The synchronization *tree depth* is defined as the maximum layer index, and it is here $P = K-1$. The variance of the synchronization error at hop distance ℓ, $\Delta_\ell[n] = f_\ell[n] - f_1$, can be computed from its power spectral density (PSD) $S_\ell(f)$ as $E\left[\Delta_\ell^2[n]\right] = \int_0^1 S_\ell(f)df$, where f is the normalized frequency in cycles/sample. Given the transfer functions $H(z)$ and

$G(z)$ previously defined in Section 3.4, the error PSD at layer ℓ may be expressed as (see, e.g., [13])

$$S_\ell(f) = \left(\sigma_w^2 \left|H(f)\right|^2 + \sigma_v^2 \left|G(f)\right|^2\right) \left(\frac{1 - \left|H(f)\right|^{2\ell}}{1 - \left|H(f)\right|^2}\right), \tag{3.27}$$

where the equality $z = \exp(2\pi f)$ has been employed. To integrate (3.27), let the *incremental noise* PSD $s_\ell(f) = S_\ell(f) - S_{\ell-1}(f)$ be defined as the difference between the noise PSD at layer ℓ, and the noise PSD at layer $\ell - 1$, where the noise PSD of the master node (layer $\ell = 0$) is $S_0(f) = 0$. The synchronization error variance can be expressed as the sum

$$E\left[\Delta_\ell^2[n]\right] = \sum_{i=1}^{\ell} \delta_i^2, \tag{3.28}$$

where δ_i^2 is the variance of the incremental noise and it is computed by the integral $\delta_i^2 = \int_0^1 s_i(f)df$. Given the expression of $H(z)$ and $G(z)$ in (3.16)-(3.17), after some algebra $s_\ell(f)$ may be written as

$$s_\ell(f) = \left(\sigma_w^2 + \frac{\sigma_v^2}{\epsilon^2}\left(4\sin^2(\pi f)\right)\right)\left[\frac{\epsilon^2}{(1 + (1-\epsilon)^2) - 2(1-\epsilon)\cos(2\pi f)}\right]^\ell. \tag{3.29}$$

The integral of (3.29) may be computed analytically (see eq. 2.554.3 of the table in [68]) in the case of noiseless local oscillators ($\sigma_v^2 = 0$) . However, the final expression is involved and does not allow any insight on the behavior of the system. In order to derive a useful approximation for δ_ℓ^2, two properties of (3.27) are exploited, namely that $S_\ell(0) = \ell\sigma_w^2$, and that, for $\ell \to \infty$ and $f > 0$, $S_\ell(f)$ can be approximated as

$$S_{\ell\to\infty}(f) = \left(\sigma_w^2 \left|H(f)\right|^2 + \sigma_v^2 \left|G(f)\right|^2\right) \frac{1}{1 - \left|H(f)\right|^2}. \tag{3.30}$$

It is then possible to obtain a raw approximation of the integral of (3.29) by computing the area of the shaded region in Figure 3.6. In particular, it can be proved that, for $\ell > \frac{\sigma_v^2}{\sigma_w^2(1-\epsilon)}$

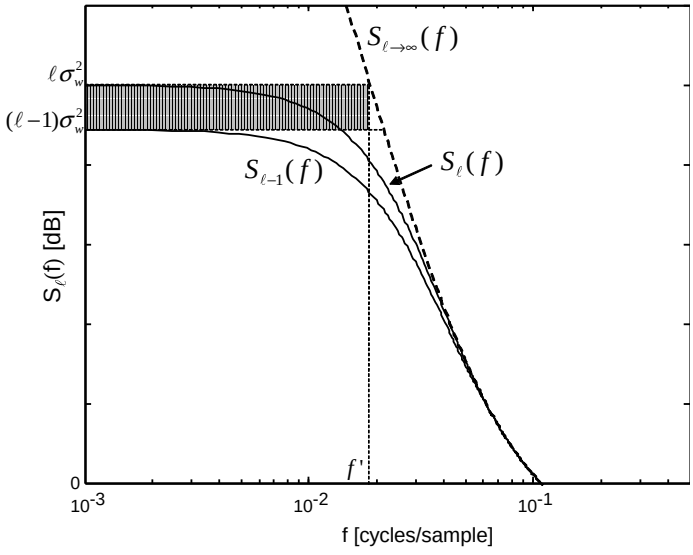


Figure 3.6 PSD of synchronization error at layer ℓ and $\ell - 1$ compared with the limit $S_{\ell\to\infty}(f)$ in (3.30), $\epsilon = 0.3$, $\sigma_w^2 = 1$, $\sigma_v^2 = 0.5$.

and $\epsilon \in (0, 1)$, the synchronization error is

$$\delta_\ell^2 \simeq \frac{\epsilon\sigma_w^2}{\pi\sqrt{\ell\,(1-\epsilon) - \frac{\sigma_v^2}{\sigma_w^2}}}, \tag{3.31}$$

thus showing that δ_ℓ^2 can be made arbitrarily small by reducing the loop gain ϵ, as expected. The incremental noise also decreases as the layer index grows large, $\delta_\ell^2 = O\left(1/\sqrt{\ell}\right)$ as $\ell \to \infty$. The approximation (3.31) is compared with the exact integral of (3.29) in Figure 3.7, for different values of the loop gain ϵ and ideal oscillators ($\sigma_v^2 = 0$). The approximation is tighter for small ϵ and large ℓ. The reduction of the incremental noise variance is clearly due to the low-pass filtering properties of the FLL. In fact, if $\epsilon = 1$ the FLL is an all-pass filter, and the incremental noise is constant over ℓ.

For the line network of Figure 3.4.b, the network mean square error (3.24) in terms of δ_ℓ^2 becomes

$$\xi^2 = \frac{1}{K}\sum_{\ell=1}^{K-1}(K-\ell)\,\delta_\ell^2. \tag{3.32}$$

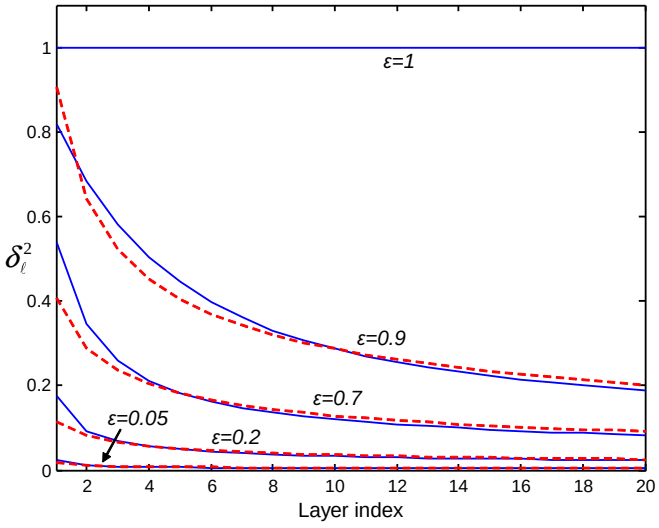


Figure 3.7 Incremental noise variance δ_ℓ^2 (solid line) compared with the approximation (3.31) (dashed line) versus the layer index ℓ, for $\sigma_w^2 = 1$ and $\sigma_v^2 = 0$.

In that network, each node communicates only with the next nearest neighbor along the line, the tree depth is equal to $P = K - 1$, and each tree layer comprises a single node. By increasing the transmission range, the master node could reach more than one neighbor. On the other hand, if each node along the line did the same, the tree depth would be smaller than $K-1$. Let the synchronization layer or stratum $\mathcal{L}_\ell$ be defined as the set comprising the nodes that are ℓ hops far from the master node [13][17]. Assuming that each node selects the neighbor belonging to the highest layer as its parent, the synchronization error variance is the same for nodes belonging to the same layer. The relationship between $E\left[\Delta_\ell^2[n]\right]$ and δ_ℓ^2 is again as in (3.28). Therefore, (3.32) can easily generalized to

$$\xi^2 = \frac{1}{K} \sum_{\ell=1}^{P} |\mathcal{L}_\ell| \, (P - \ell + 1) \, \delta_\ell^2, \tag{3.33}$$

where P is the number of tree layers. Since, as before, each layer accumulates noise from previous ones, the network synchronization error decreases when the number of layers P is reduced by increasing the transmission radius. The minimum network error is achieved when the master node is able to reach all the other nodes in the network, i.e., $P = 1$ and $\xi^2 = \frac{K-1}{K}\delta_1^2$.

3.5 Simulation Results

This section first validates the discussion on acquisition stability in Section 3.3 by considering a MC topology where $K = 4$ nodes are grouped in two clusters as in Figure 3.8.a. This topology is particularly relevant for cooperative communications applications,

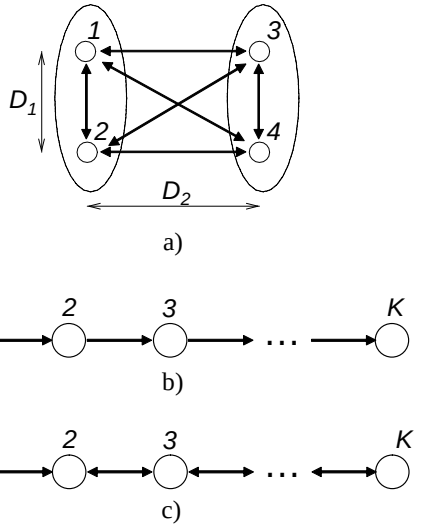


Figure 3.8 Simple network topologies considered in Section 3.5: a) MC network of $K = 4$ nodes with full connectivity; b)-c) MS and MC line networks of K nodes with $r = 1$, the distance between nodes is normalized to unity.

as seen in [69]. The transmission radius is such that each node is connected with all the others ($r \gg D_2$, all-to-all network). In the following, all frequencies are normalized over the sampling frequency $1/T_s$, or equivalently $T_s = 1$. For this scenario, the starting point is chosen as $\mathbf{f}[0] = f_c + [0.15, 0.05, -0.05, -0.15]^T$, where f_c is the nominal carrier frequency of the communication system. This setup guarantees no occurrence of false locks for the deterministic system (3.9) (that is for a number of samples $L \rightarrow \infty$) . It is worth to recall that, on the other hand, the convergence of the random linear system (3.10) does not depend on the initial conditions. It has been observed that, due to the random nature of the connections, false locks do occur for finite L, but rapidly decreasing in probability as L grows. For the sake of clarity, the following simulation results do not include the instances where false locks occur. The proposed algorithm is compared with a DFT-based algorithm,

which estimates the first moment of the power spectrum of the received signal through the DFT of the L samples available. In Figure 3.9 the root mean square (RMS) network

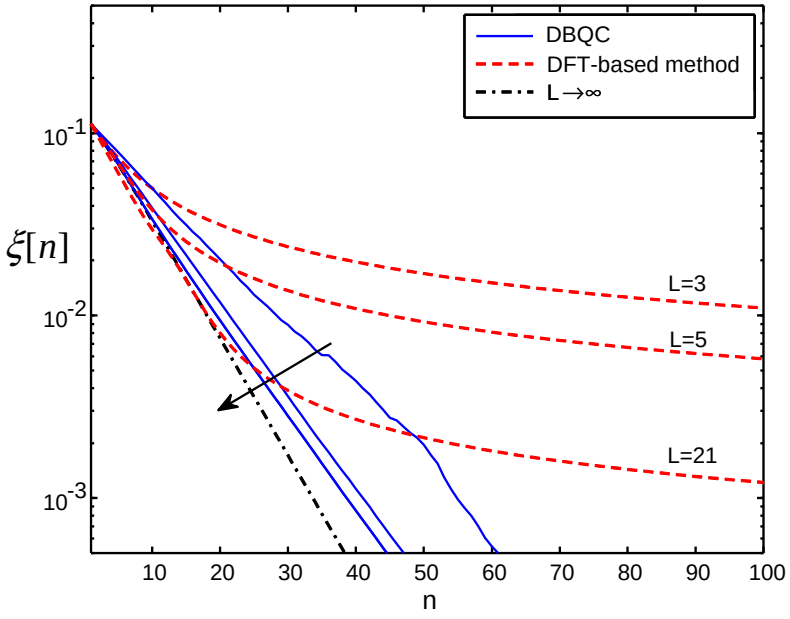


Figure 3.9 RMS network synchronization error $\xi[n]$ for different algorithms ($D_2/D_1 = 1.2$, $\epsilon = 0.15$, $L = 3, 5, 21$).

synchronization error $\xi[n]$ is plotted versus the iteration index n for different values of $L = 3, 5, 21$. In Figure 3.8.a it is $D_2/D_1 = 1.2$, path-loss exponent is $\alpha = 2$, loop gain is $\epsilon = 0.15$, and noise sources have been neglected as in Section 3.3, $\sigma_w^2 = \sigma_v^2 = 0$. Regarding instances where a false lock occurs, with $L = 3$ the probability of a false lock is 1.48%, and never occurred for $L = 5$ and 10000 independent runs. The eigenvalues of the matrix $\mathbf{\Pi}_\epsilon$ are all inside the unit circle for all values of L, and the nonlinear system is able to achieve consensus w.p.1 when no false lock occurs, where greater L improves convergence speed. The DFT-based algorithm is inevitably limited by the few frequency samples available. Despite an intrinsic resilience to false locks, the convergence rate of this algorithm dramatically reduces after few iterations, and, for a practical (small) number of iterations, it can achieve synchronization only to a finite accuracy.

Next, the focus is on the tracking performance of MC and MS synchronization topologies. The following simulations are relative to linear topologies comprising $K = 25$

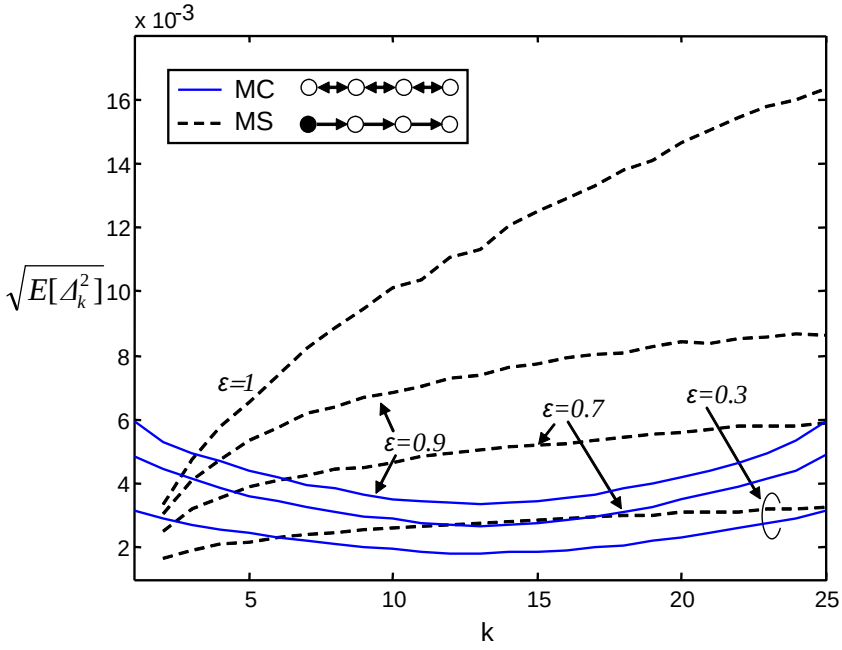


Figure 3.10 Steady-state RMS synchronization error for a line network of $K = 25$ nodes with MC and MS topology ($r = 1$).

nodes, as the ones in Figure 3.8.b-c for MS and MC, respectively. D-FLL's are modeled as a discrete-time linear system driven by noise as in Section 3.4. Initial frequencies are uniformly distributed within the interval $[-0.1, 0.1]$. It is assumed that a transmission power $P_T = p_0$ is needed to guarantee nearest neighbor connectivity, or, equivalently, unitary transmission radius, $r = 1$. The frequency noise variance is $\sigma_v^2 = 10^{-6}$, while the FDD noise variance is obtained from (3.15) with $p_0 T_s/N_0 = 10dB$ and $L = 26$ samples. In Figure 3.10 the steady-state RMS synchronization error $\sqrt{E\left[\Delta_k^2\right]}$ is plotted versus node index k for both synchronization topologies with nearest neighbor connectivity, $r = 1$. As expected, both MC and MS achieve higher synchronization accuracy as the loop gain ϵ is reduced. As predicted by the findings in Section 3.4.2, it is seen that a small loop gain reduces noise accumulation with MS topology. An interesting feature of MC topologies that was not predicted by the analysis in Section 3.4.1, is the smooth error distribution within the network, especially for small loop gains. Noise accumulation for MC occurs at the edges of the linear network, while synchronization error is minimum at the center. It is

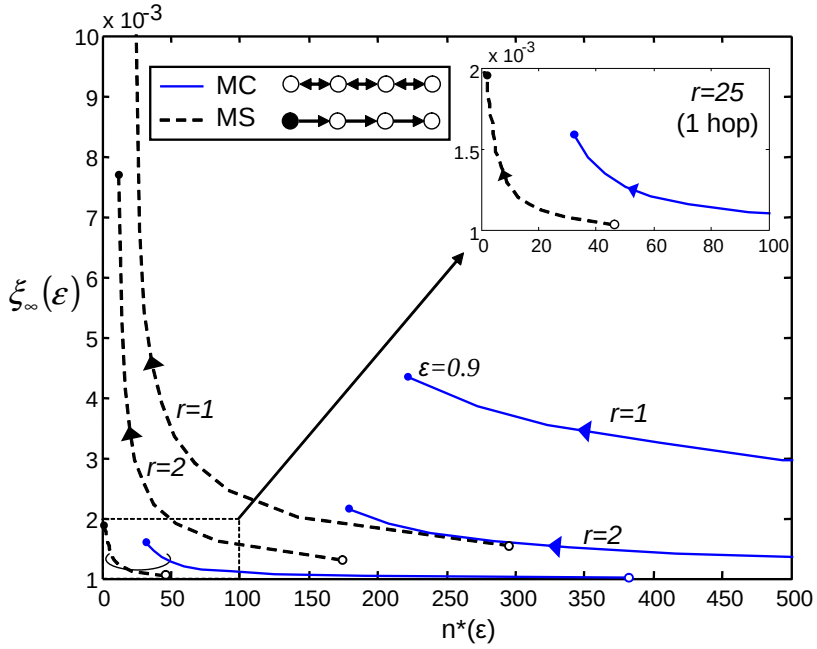


Figure 3.11 Steady-state RMS network synchronization error $\xi_\infty(\epsilon)$ versus convergence time $n^*(\epsilon)$ for a line network of $K = 25$ nodes with MC and MS topology. Loop gain ϵ varies from $\epsilon = 0.1$ (empty dot) to $\epsilon = 1$ (filled dot).

envisaged that the inherent robustness to noise accumulation of MC topologies could make this architecture attractive for large networks.

The lower noise sensitivity of MC architectures is traded with larger convergence times, as each node is sensitive to the ensemble of his neighbors. This effect is shown in Figure 3.11 in terms of steady-state RMS network synchronization error $\xi_\infty(\epsilon)$ versus convergence time necessary to reach steady-state $n^*(\epsilon)$ for varying loop gain $\epsilon = [0.1, 1]$ and increasing transmission range r. The convergence time is defined as the iteration index $n^*(\epsilon)$ such that $|\xi[n^*(\epsilon)] - \xi_\infty(\epsilon)| / \xi_\infty(\epsilon) < 0.1$. It is assumed that a transmission range r may be achieved by employing a transmission power $P_T = p_0 r^\alpha$. As shown previously in Figure 3.10, Figure 3.11 confirms that higher accuracy may be achieved by reducing the loop gain ϵ. Nevertheless, Figure 3.11 shows that improving the accuracy of network synchronization entails a cost in terms of convergence time. In addition, given a desired accuracy ξ_∞, MS topologies are uniformly faster in convergence than MC topologies, for any ϵ. This means that the smooth error distribution provided by the MC architecture is

achieved at the price of a slower convergence. As expected, for a given required accuracy, the convergence speed of both architectures can be improved by increasing connectivity through a higher transmission range r. For $r = 25$, each node is able reach any other node with 1-hop (all-to-all MC, single-layer MS topology), and MS still outperforms MC. In fact, the detector of Section 3.2 weights the contribution of received signals based on their relative power. This means that the contribution of nearest neighbors is dominant with respect to other nodes, irrespectively of the transmission radius. If the employed detector weights contributions from all neighbors in the same way (as in MAC layer synchronization), 1-hop MC and MS topologies are equivalent.

3.6 An Application: Multi-hop relay networks

As remarked before, one of the possible applications of distributed carrier frequency synchronization is the implementation of cooperative communication techniques at the physical layer. This section presents some results regarding the impact of carrier frequency offsets on a multi-hop relay network as the one depicted in Figure 3.12. Similar networks

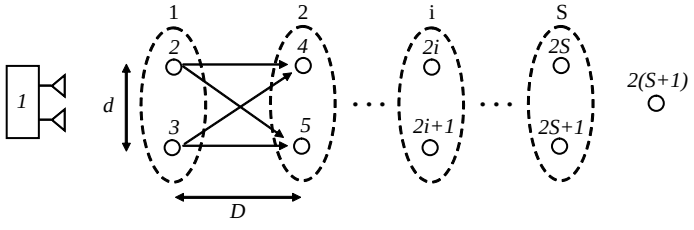


Figure 3.12 Multi-hop relay network with S relaying stages.

have been considered in recent works on distributed space-time codes, see, e.g., [70][71]. In this setup, a two-antenna source node wishes to communicate to a single-antenna destination node. Since the destination is out of the transmission range of the source, multi-hop communication is realized by S relaying stages, each made up of two relay nodes (the total number of nodes is thus $K = 2(S + 1)$). As in Section 3.1.2, it is assumed that each transmitted packet contains a preamble signal for synchronization purposes. To

elaborate, during the first frame the source node transmits a packet (preamble and data) to the first stage of relays, which process it and forward it to the next stage during the second frame. In the i-th frame, the i-th relay stage process the signal received from the $(i-1)$-th stage, until the message finally gets to the destination, at the end of the $(S+1)$-th frame.

Targeting a scenario with frequency (and phase) offsets, it is considered the use of differential space-time block codes (DSTBC) at each relaying stage (see Appendix 3.C). DSTBC does not require channel estimation at the receiving side (at the price of about 3 dB loss in equivalent SNR as compared to coherent STBC) [20]. Also, the probability of symbol error in the presence of carrier frequency offsets can be shown to be independent of the block (packet) length. According to the communication protocol, the $(i-1)$-th relay stage employs a DSTBC to forward the message to the i-th stage, where each node independently decodes (decode and forward relaying, DF) and re-encodes for transmission in the subsequent slot. Different synchronization strategies can be devised for this scenario.

a - Open-loop: each node adjusts its frequency in a *memory-less* (one-shot, or open-loop) fashion before decoding the data payload of the transmitted packet. Namely, the carrier frequency offset of the nodes in the receiving stage is computed upon reception of the preamble signal employing (3.6). This scheme correspond to the current practice, and it essentially assumes that previous stages have already achieved a good level of synchronization. However, for small L, the one-shot frequency estimate is affected by a relevant error already at stage 1, which inevitably propagates to the following stages in the subsequent steps.

b - Closed-loop A: similarly to the open-loop technique above, only nodes in the receiving stage update their local offsets. However, a running frequency estimate is performed according to the D-FLL algorithm (3.2)-(3.6) (or equivalently Figure 3.2). Again, here each node updates the local frequency correction only when it needs to decode the transmitted data. The local frequency is updated by combining the new estimate

provided by the detector (3.6) with previous estimates, thus allowing for a progressive refinement of the local correction.

c - Closed-loop B: this scheme extends closed-loop B by considering that the D-FLL algorithm shows faster convergence times in well-connected networks. The simplest way to improve network connectivity (for synchronization purposes) is to let all nodes listen to each synchronization signal transmitted in the network, whether or not they need to decode the data payload. Therefore, differently from the previous techniques, *all* nodes that are currently not busy in transmission (except the source) update their running frequency estimate according to Figure 3.2.

Notice that all the aforementioned synchronization algorithms are based on a MS architecture, where the frequency of the source node is the network reference. In the following, it is assumed that each transmitted symbol has unitary power and that each link is affected by additive white Gaussian noise with variance N_0. The SNR is defined as $SNR = 1/N_0$. The path loss exponent is $\alpha = 3$. The preamble signal is always assumed to be transmitted with a 5 dB power boost with respect to the data payload. The employed modulation is BPSK. The carrier frequency of node k at startup, $f_k[0]$, is assumed to be uniformly distributed in the interval $[-f_{0,\max}, f_{0,\max}]/T_s$. The ratio between the inter-stage distance and the intra-stage distance is $D/d = 1.2$. Also, trading off accuracy versus convergence speed, the loop gain is set $\epsilon = 0.35$. Finally, the training length is $L = 11$ samples for both open and closed-loop techniques.

Figure 3.13 shows the degradation in the end-to-end BER due to increasing frequency offsets among the nodes in the network, in the case where no frequency offset correction takes place. In this case $S = 5$ stages. In ideal conditions ($f_{0,\max} = 0$), the use of DSTBC provides a diversity gain of 2, while a maximum spread $f_{0,\max} = 0.04$ is sufficient to nullify the diversity gain of the space-time scheme, raising the slope of the curve from 2 to approximately 1 (for this range of SNR values).

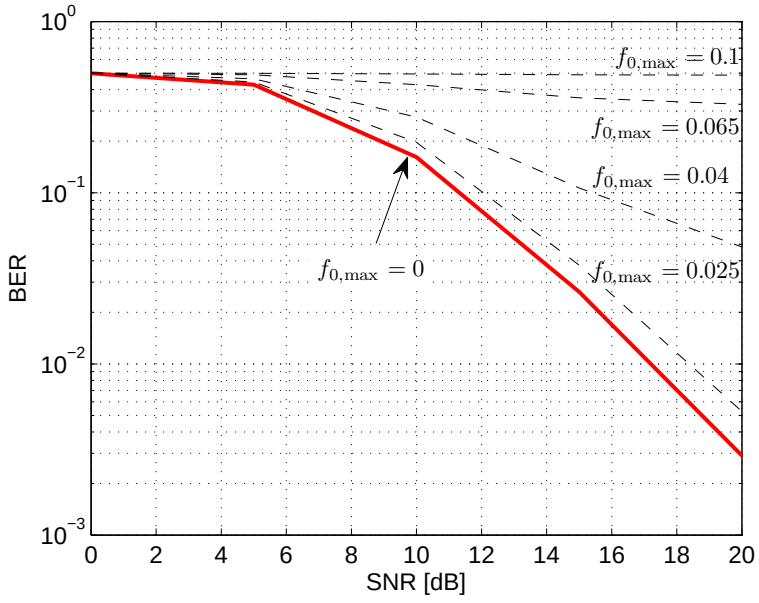


Figure 3.13 End-to-end BER after $S = 5$ stages for the network in Figure 3.12 without frequency offset compensation ($\epsilon = 0$, $D/d = 1.2$).

In the following simulation results, the end-to-end transmission of p packets (corresponding to $(S + 1)p$ frames) is considered. The performance measures of interest are the network RMS synchronization error $\xi[p]$ and the end-to-end BER associated with each packet. Figure 3.14 compares the speed of convergence of the three algorithms discussed in the previous section, in terms of the RMS synchronization error $\xi[p]$, ($S = 3$ stages, $f_{0,\max} = 0.15$ and $SNR = 15dB$). The open loop approach is limited by the very few samples employed ($L = 11$), whereas both closed-loop algorithms exploit the filtering properties of the D-FLL to achieve much better accuracy. The algorithm B converges faster, but at the price of a higher noise floor. However, this impairment is immaterial to BER performance (see below). Algorithm B was expected to be faster because it better exploits network connectivity. Nevertheless, its accuracy is impaired by the fact that nodes adjust their frequencies even when receiving noisy pilot signals from distant nodes. Scheme A, on the contrary, requires nodes to listen only to their immediate neighbors.

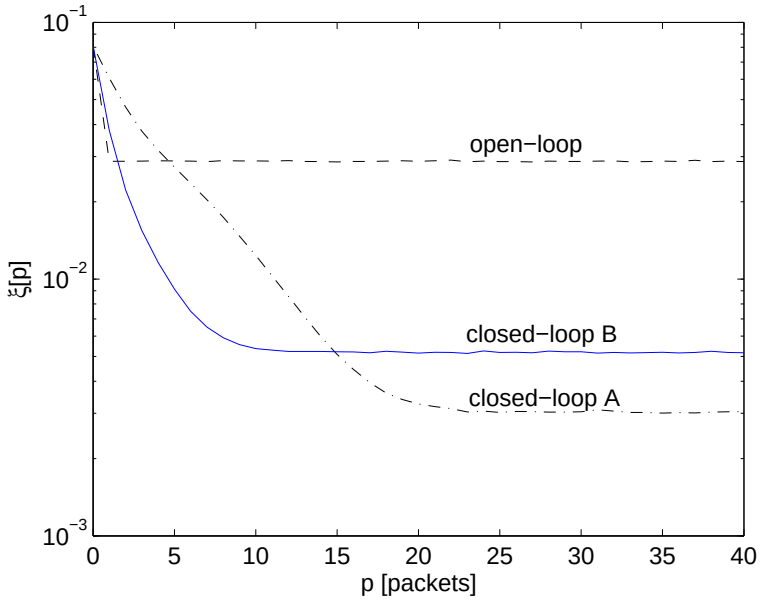


Figure 3.14 Root mean square (RMS) synchronization error $\xi[p]$ for different algorithms ($f_{0,\max} = 0.15$, $S = 3$, $D/d = 1.2$, $\epsilon = 0.35$, $L = 11$, $SNR = 15dB$).

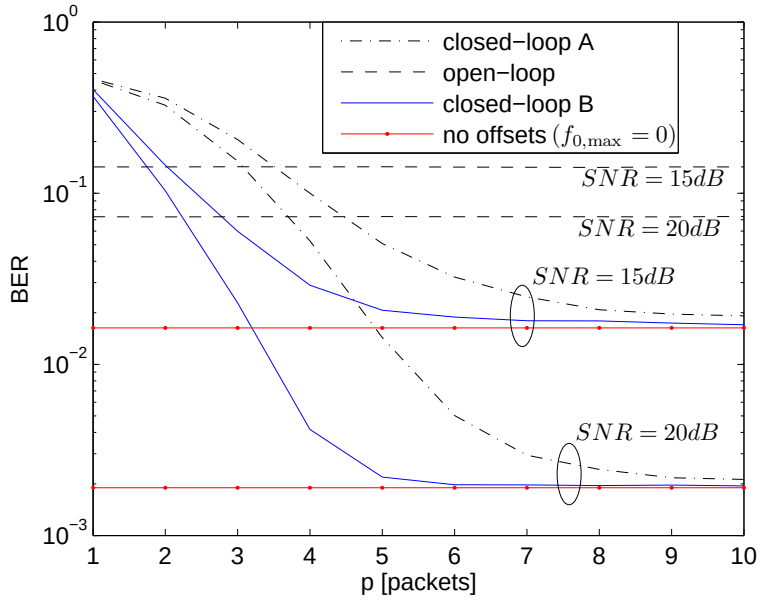


Figure 3.15 End-to-end BER for different algorithms ($f_{0,\max} = 0.15$, $S = 3$, $D/d = 1.2$, $\epsilon = 0.35$, $L = 11$, $SNR = 15, 20dB$).

Finally, Figure 3.15 verifies the impact of the performance in Figure 3.14 on the end-to-end BER as a function of the packet index p ($S = 3$ stages and $f_{0,\max} = 0.15$). Despite of a higher error floor, the algorithm B needs only $p = 4$ packets to get close to the synchronous system performance for both SNR values, while the scheme A requires at least $p = 7$ packets. The open loop technique would need a longer preamble sequence to improve its performance.

3.7 Conclusions

This chapter presented frequency synchronization techniques for wireless networks based on the concept of Distributed Frequency Locked Loops (D-FLL). Both hierarchical (master slave - MS) and peer-to-peer (mutually coupled - MC) architectures have been considered for the synchronization network. A design for the frequency difference detector (FDD) has been proposed whereby all nodes transmit the same pilot signal for synchronization purposes, and the local frequency correction is computed by processing the received superposition of pilot signals from neighboring nodes. It was shown that the stability of frequency acquisition with a D-FLL network equipped with the proposed FDD can be guaranteed by a suitable choice of loop and detector parameters. Analysis of steady-state accuracy of frequency tracking accounts for both channel noise and oscillator frequency instabilities. A general analysis, valid for any network topology, has been carried out and then specialized for a MC ring network and a MS line network. While the MC network is equivalent to a parallel connection of FLL's, the MS network corresponds to a cascade connection of FLL's. In both cases, synchronization accuracy may be improved by reducing the loop bandwidth and/or by improving connectivity with a higher transmission range. Finally, by the aid of simulation results, it was shown that MC provides a smooth error distribution over the network at the price of a slower convergence time as compared to MS. On the other hand, noise accumulation with hierarchical architectures may be mitigated by reducing the loop bandwidth.

Appendix 3.A: Derivation of the FDD

First, consider the continuous-time counterpart of (3.4), $y_k(t)$, and let the frame duration be $T_F \to \infty$ (or $L \to \infty$). With $a_{k,i} = |h_{k,i}|^2$, it can be seen that, for $|f_i - f_k| < \frac{1}{2T_s}$,

$$\bar{e}_k = \frac{e_k}{\sum_{i \neq k} a_{k,i}} = \frac{\int_{-1/2T_s}^{1/2T_s} f \cdot Y_k(f) df}{\int_{-1/2T_s}^{1/2T_s} \left(Y_k(f) - N_0\right) df}, \tag{3.34}$$

where $Y_k(f)$ is the power spectral density of $y_k(t)$ and $e_k = \sum_{i \neq k} a_{k,i} \left(f_i - f_k\right)$. Recalling that $Y_k(f)$ is the Fourier transform of the autocorrelation $r_y(t) = E\left[y_k(t+\tau) y_k^*(\tau)\right]$, it is possible to write the error e_k in (3.34) as

$$e_k = \frac{1}{2\pi j} \int_{-1/2T_s}^{1/2T_s} j2\pi f Y_k(f) df = \frac{1}{2\pi j} \frac{dr_y(t)}{dt}\bigg|_{t=0}. \tag{3.35}$$

By employing a first-order finite difference in lieu of the derivative, (3.35) is approximated by

$$e_k \simeq \frac{1}{2\pi j} \frac{r_y(T_s) - r_y(-T_s)}{2T_s}. \tag{3.36}$$

By neglecting the noise at the denominator of (3.34) ($N_0 = 0$), and employing the (unbiased) *sample* autocorrelation of the sampled signal $y_k\left(mT_s\right)$, $\tilde{r}_y\left(mT_s\right) = \frac{1}{L-m} \sum_{i=0}^{L-1-m} y_k\left(\left(i+m\right)T_s\right) y_k^*\left(iT_s\right)$, the normalized error $\bar{e}_k$ is computed with

$$\bar{e}_k = \frac{1}{4\pi T_s} \frac{\mathrm{Im}\left\{\tilde{r}_y\left(T_s\right) - \tilde{r}_y\left(-T_s\right)\right\}}{\tilde{r}_y\left(0\right)}, \tag{3.37}$$

which is the proposed FDD[1]. As a final remark, an alternative estimator for (3.34) was proposed in [72] along the same lines of the derivation presented here.

[1]By neglecting the noise in the normalization of (3.34), the actual FDD output is $\bar{e}_k[n] = \alpha \left[\frac{e_k}{\sum_{i \neq k} a_{k,i}}\right]$, where the *detector gain* $\alpha \leq 1$. This does not affect loop stability, which is the main concern in this chapter. Notice that the detector gain would still be close to unity for high SNR, $\sum_{i \neq k} |h_{k,i}|^2 \gg N_0/T_s$.

Appendix 3.B: Derivation of the FDD Error Variance (3.15)

This appendix contains the derivation of the approximation of the FDD error variance in (3.15). The input to the FDD (3.4) may be reformulated as

$$y_k\left(mT_s\right) = s_k\left(mT_s\right) + z_k\left(mT_s\right), \tag{3.38}$$

for $m = 0, \ldots, L-1$, where the signal is $s_k\left(mT_s\right) = \sum_{i \neq k} |h_{k,i}| \exp\left(j\left(2\pi(f_i - f_k)mT_s + \phi_{k,i}\right)\right)$ and the noise process $z_k\left(mT_s\right)$ is i.i.d. and Gaussian, $z_k\left(mT_s\right) \sim \mathcal{CN}\left(0, N_0/T_s\right)$. The *degree* of node k is defined as the power of the signal $s_k\left(mT_s\right)$ when $L \to \infty$, $d_k = \sum_{i \neq k} |h_{k,i}|^2$. Recall the expression of the FDD in (3.6)

$$\bar{e}_k = \frac{1}{2\pi\left(L-1\right)T_s d_k} \mathrm{Im}\left\{ \sum_{i=0}^{\frac{L-3}{2}} \left[\left(y_k\left(\left(2i+2\right)T_s\right) - y_k\left(\left(2i\right)T_s\right)\right) y_k^*\left(\left(2i+1\right)T_s\right) \right] \right\}, \tag{3.39}$$

where the normalization factor $\tilde{r}_y(0)$ in (3.6) is substituted by the received signal power $d_k = \sum_{i \neq k} |h_{k,i}|^2$. Given the model (3.38), the detector (3.39) can be expressed as the sum of three terms, namely

$$\bar{e}_k = \frac{1}{2\pi\left(L-1\right)T_s d_k} \mathrm{Im}\left\{u_{\mathrm{SS},k} + u_{\mathrm{NS},k} + u_{\mathrm{NN},k}\right\}, \tag{3.40}$$

where the term $u_{\mathrm{SS},k}$ is related to signal-signal interaction, $u_{\mathrm{NS},k}$ to noise-signal interaction, and $u_{\mathrm{NN},k}$ to noise-noise interaction. The impact of the term $u_{\mathrm{SS},k}$ on loop dynamics has already been analyzed in detail in Section 3.3. In the following, the focus is on the analysis of the noise terms $u_{\mathrm{NS},k}$ and $u_{\mathrm{NN},k}$, which are the cause of frequency estimation errors.

Let us start by focusing on noise-signal interaction, $u_{\mathrm{NS},k}$. By plugging (3.38) in (3.39) and collecting the terms where the signal is multiplied by noise, it holds that

$$u_{\mathrm{NS},k} = \sum_{i=0}^{\frac{L-3}{2}} \Big[\left(s_k\left(2i+2\right) - s_k\left(2i\right)\right) z_k^*\left(2i+1\right) + \left(z_k\left(2i+2\right) - z_k\left(2i\right)\right) s_k^*\left(2i+1\right) \Big], \tag{3.41}$$

where the sampling period T_s has been neglected for notational convenience (or equivalently $T_s = 1$). The term $u_{\text{NS},k}$ is a random variable with zero mean and variance to be determined. The expression (3.47) may be rewritten as

$$u_{\text{NS},k} = \sum_{i=0}^{\frac{L-3}{2}} \left[\left(s_k\left(2i+2\right) - s_k\left(2i\right) \right) z_k^*\left(2i+1\right) \right] - \sum_{i=0}^{\frac{L-5}{2}} \left[\left(s_k\left(2i+3\right) - s_k\left(2i+1\right) \right)^* z_k\left(2i+2\right) \right] + \\ + \left(s_k^*\left(L-2\right) z_k\left(L-1\right) - s_k^*\left(1\right) z_k\left(0\right) \right). \tag{3.42}$$

Notice that only the imaginary part of (3.42), $\text{Im}\left\{u_{\text{NS},k}\right\}$, contributes to the detector output (3.40). By exploiting the independence of noise samples, the variance of $\text{Im}\left\{u_{\text{NS},k}\right\}$ may be evaluated as

$$E\left[\left(\text{Im}\left\{u_{\text{NS},k}\right\}\right)^2\right] = \frac{N_0}{2T_s} \sum_{i=0}^{L-3} \left| s_k\left(i+2\right) - s_k\left(i\right) \right|^2 + \frac{N_0}{2T_s} \left(\left| s_k\left(L-2\right) \right|^2 + \left| s_k\left(1\right) \right|^2 \right). \tag{3.43}$$

The last term of (3.43) is roughly proportional to the energy of $s_k\left(m\right)$, i.e., it is proportional to the node degree d_k; the first term, instead, is proportional to the energy of the signal

$$s_k\left(m+2\right) - s_k\left(m\right) = 2j \sum_{i \neq k} \left| h_{k,i} \right| \sin\left(2\pi (f_i - f_k) T_s\right) e^{j\left(2\pi (f_i - f_k)(m+1)T_s + \phi_{k,i}\right)}, \tag{3.44}$$

where $m = 0, \ldots, L-1$. Consequently, (3.43) is approximated as

$$E\left[\left(\text{Im}\left\{u_{\text{NS},k}\right\}\right)^2\right] \simeq 2\frac{N_0}{T_s} \left(L-2\right) \sum_{i \neq k} \left| h_{k,i} \right|^2 \sin^2\left(2\pi (f_i - f_k) T_s\right) + N_0 d_k. \tag{3.45}$$

When frequency offsets are small, (3.45) is further simplified to yield

$$E\left[\left(\text{Im}\left\{u_{\text{NS},k}\right\}\right)^2\right] \simeq 2\frac{N_0}{T_s} \left(2\pi T_s\right)^2 \left(L-2\right) \sum_{i \neq k} \left| h_{k,i} \right|^2 \left(f_i - f_k\right)^2 + \frac{N_0}{T_s} d_k. \tag{3.46}$$

We are left with the analysis of noise-noise interaction, $u_{\mathrm{NN},k}$. This term corresponds to the output of the FDD when there is only noise at the input, namely

$$u_{\mathrm{NN},k} = \sum_{i=0}^{\frac{L-3}{2}} \Big[\left(z_k\left(2i+2\right) - z_k\left(2i\right) \right) z_k^*\left(2i+1\right) \Big]. \tag{3.47}$$

It can be observed that $u_{\mathrm{NN},k}$ is zero mean and uncorrelated with $u_{\mathrm{NS},k}$ in (3.41). The FDD (3.40) employs only the imaginary part of (3.47), $\mathrm{Im}\left\{u_{\mathrm{NN},k}\right\}$, whose variance may be computed by exploiting the independence of noise samples as

$$E\left[\left(\mathrm{Im}\left\{u_{\mathrm{NN},k}\right\}\right)^2\right] = \frac{L-1}{2}\left(\frac{N_0}{T_s}\right)^2. \tag{3.48}$$

Let us now go back to the expression of the frequency detector (3.40). Given the previous derivations, the variance of the FDD output is

$$\mathrm{Var}\left(\bar{e}_k\right) = \frac{E\left[\left(\mathrm{Im}\left\{u_{\mathrm{NS},k}\right\}\right)^2\right] + E\left[\left(\mathrm{Im}\left\{u_{\mathrm{NN},k}\right\}\right)^2\right]}{\left(2\pi\left(L-1\right)T_s d_k\right)^2}. \tag{3.49}$$

By plugging (3.46)-(3.48) in (3.49), it can be obtained

$$\begin{aligned}\mathrm{Var}\left(\bar{e}_k\right) \simeq & \frac{2\left(L-2\right)}{\left(L-1\right)^2}\frac{\sum_{i\neq k}|h_{k,i}|^2(f_i-f_k)^2}{\sum_{i\neq k}|h_{k,i}|^2}\left(\frac{N_0}{d_k T_s}\right) + \frac{1}{4\pi^2 T_s^2\left(L-1\right)^2}\left(\frac{N_0}{d_k T_s}\right) + \\ & \frac{1}{8\pi^2 T_s^2\left(L-1\right)}\left(\frac{N_0}{d_k T_s}\right)^2\end{aligned} \tag{3.50}$$

In the tracking regime, the frequency spread is small, $\Delta f \ll 1/T_s$, and therefore the first term in (3.50) may be neglected, so as to yield (3.15).

Appendix 3.C: DSTBC with frequency offsets

Without loss of generality, here we consider the l-th space-time codeword transmitted during the n-th frame from the i-th stage to the $(i+1)$-th stage (nodes $2i$ and $2i+1$ are transmitting, see Figure 3.12). Also, the focus is on the processing at node $2i+2$ within the $(i+1)$-th stage, as the two receiving nodes decode independently of each other.

Let the input alphabet $\mathcal{X} = \{\mathbf{X}_l\}$ be a finite set of 2×2 unitary matrices. The differentially encoded Space-Time codeword actually transmitted over the channel is $\mathbf{W}_l = \mathbf{W}_{l-1}\mathbf{X}_l$, with $\mathbf{W}_0 = \mathbf{I}$. As suggested in [20], $\mathbf{X}_l$ is chosen as a *normalized* Alamouti code matrix, such that $\mathbf{X}_l^H \mathbf{X}_l = \mathbf{I}$. Assuming a synchronous system, the 1×2 received vector signal over two consecutive symbol periods is

$$\mathbf{y}_l = \mathbf{h}\mathbf{W}_l + \mathbf{n}_l = \mathbf{y}_{l-1}\mathbf{X}_l - \mathbf{n}_{l-1}\mathbf{X}_l + \mathbf{n}_l, \tag{3.51}$$

where $\mathbf{h} = [h_{2i+2,2i}[n], h_{2i+2,2i+1}[n]]$ is the channel between the transmitting nodes $(2i, 2i+1)$ and the receiving node $2i+2$ (constant over the whole frame period), and the additive noise $\mathbf{n}_l \sim \mathcal{CN}(\mathbf{0}, N_0\mathbf{I})$. From (3.51), as in differential modulation for point-to-point channels, the signal vector received at time $l-1$ is the effective channel at time l, and the information-bearing signal is corrupted by two noise terms.

In case of different frequency offsets at the two transmitting nodes, the received vector signal (3.51) can be written as

$$\mathbf{y}_l = \mathbf{h} \begin{bmatrix} w_{1,l} e^{j\omega_1 2lT_s} & -w_{2,l}^* e^{j\omega_1 (2l+1)T_s} \\ w_{2,l} e^{j\omega_2 2lT_s} & w_{1,l}^* e^{j\omega_2 (2l+1)T_s} \end{bmatrix} + \mathbf{n}_l, \tag{3.52}$$

where $\omega_1 = 2\pi(f_{2i}[n] - f_{2i+2}[n])$ and $\omega_2 = 2\pi(f_{2i+1}[n] - f_{2i+2}[n])$ are the offsets between the two transmitting nodes and the receiving node. Due to the different carrier frequencies, the effective Space-Time codeword is no more orthogonal, generating Inter-Symbol-Interference at the output of the detector.

Part II

Synchronization at the MAC Layer

Preface to Part II

In the first part of the thesis, it was shown how a network of coupled FLL's can be employed to provide network-wide carrier frequency synchronization, which is a fundamental facility whenever there is need to implement cooperative communication schemes at the physical layer. The study was divided into the analysis of frequency acquisition (transient dynamics) and the analysis of frequency tracking in presence of diverse noise sources (steady-state accuracy). The second part of the thesis is devoted to *time* synchronization at the medium access control (MAC) layer. MAC layer synchronization is required to allow the use of efficient medium access techniques for data communication, such as time division multiple access (TDMA) and slotted contention-based access (e.g., slotted-Aloha). When implemented at the higher layers of the communication protocol stack, a synchronization algorithm needs to cope with the specific constraints imposed by protocol design. At the MAC layer, by design, synchronization information can be conveyed either by a sync sequence (i.e., a waveform with tight autocorrelation) or a timestamp (i.e., a sample of the hardware clock). In case timestamps are employed, sync packets (called beacons) can be transmitted employing either a reservation or a contention-based access protocol. Also, synchronization can be updated with very low frequency when the transceiver is operated with a small duty-cycle, e.g., for energy saving purposes. Infrequent updates make it more difficult for the nodes to maintain a tight network-wide synchronization because of the frequency instability of hardware clocks. As in Part I, Part II analyzes both synchronization acquisition and tracking, keeping into account all the aforementioned aspects of MAC layer synchronization. In particular, Chapter 4 discusses the time needed to achieve synchronization when employing different medium access strategies for the transmission of sync information. Chapter 5 focuses on the tracking accuracy attained by distributed synchronization algorithms during normal network operation (i.e., in steady-state). Finally, Chapter 6 analyzes the effects of clock frequency instability on synchronization accuracy when nodes are operated with low duty cycles.

CHAPTER 4

SYNCHRONIZATION RATE OF MEDIUM ACCESS PROTOCOLS

If the network is organized in a layered hierarchical structure (MS architecture), synchronization information propagates subsequently from one layer to the other, and network-wide synchronization can be achieved within a finite number of steps that depends on the number of layers (see, e.g., [37][25]). This chapter investigates the more difficult case whereby the network is organized with a peer-to-peer architecture (MC architecture), and synchronization has to be realized by a completely *distributed* process.

As detailed in Chapter 2, distributed synchronization algorithms for coordinated medium access in MC networks have been proposed in several research works [73] [46] [47] and standards [28] [24]. In this chapter, the general framework of phase-locked loops (PLL) is applied to network time synchronization at the MAC layer. The time required to achieve network synchronization is investigated when employing different medium access strategies for sync information. In fact, each node may convey its local time to its neighbors either by transmitting beacon frames carrying a timestamp, or by broadcasting a frame synchronization sequence on a dedicated signaling channel. Beacon frames require the use of a *reservation* or *contention* access scheme, in order to avoid or reduce the occurrence of beacon collision events. On the other hand, the same frame synchronization sequence may be transmitted simultaneously by many nodes, since the *superposition* of sequences over the radio channel allows the use of a PLL design similar to the one introduced in [43]. Reservation access exploits spatial resource reuse to enable each node to communicate with all its neighbors, and convergence may be studied by adapting results on consensus algorithms, see, e.g., [48][66][74]. When contention or superposition access schemes are used, instead, only a randomly chosen subset of nodes is active at the same time, thus

resulting in interaction among a random set of nodes. The focus of this chapter is to evaluate the convergence of *random interactions*.

The convergence of PLL's with random interactions may be studied with tools employed in the analysis of random consensus algorithms. Gossip algorithms [54] are a class of random consensus strategies where only one node is transmitting at a time. Convergence of Gossip algorithms in mean and mean square sense were studied in [54][75], and extensions to more general random interactions can be found in [76][63][77]. The approaches for analysis pursued in [63] introduced simple conditions for almost sure convergence of random consensus, and [78] studied the convergence rate of distributed Gossip through the Lyapunov exponent of the corresponding random linear dynamical system.

This chapter evaluates how random interactions induced by the use of contention or superposition access schemes affect the convergence of PLL's, by comparing the convergence rate of these protocols to the rate achieved by reservation access. In particular, the analysis focuses on convergence in average, which was proved [63] to constitute a necessary and sufficient condition for almost sure convergence. It is also shown, by simulations, that the actual convergence rate (Lyapunov exponent) can be approximated by the rate of convergence in the mean for the network model under analysis. Finally, the three considered access schemes are compared by simulations, showing that superposition is the most efficient random access protocol for synchronization purposes.

4.1 System Model

Consider a wireless network of K nodes employing packet-based communication and a slotted medium access protocol (Figure 4.1). Time slots are organized in superframes, where the initial slots are dedicated to signaling purposes (beacon slots), and the remaining slots are used for data communication. Signaling is needed mainly for data slot reservation (in case TDMA is employed) and for time synchronization. In particular, synchronization

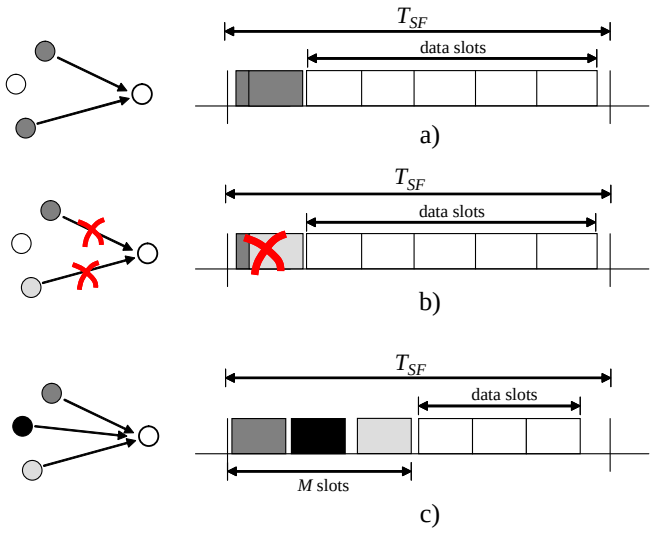


Figure 4.1 Superframe structure for different medium access protocols: a) superposition, b) contention, c) reservation.

is the first task to be carried out during the set-up of the network. Also, network synchronization can be lost after nodes have turned off their transceivers for a long time ("sleep" mode), because of the instability of hardware clocks. Notice that during network set-up, data slots are unusable because of the lack of accurate network-wide synchronization. Nevertheless, data slots are not used for synchronization, and therefore their employment is immaterial to the analysis. Nodes are assumed to employ a PLL to achieve network synchronization in a completely de-centralized fashion.

Within a signaling slot, synchronization information may be conveyed either by a beacon frame carrying a timestamp, or by a frame synchronization sequence. Beacon frames are MAC-layer packets that need to be demodulated and decoded upon reception. Therefore, a node can not receive and decode correctly multiple beacon frames at the same time, and a *reservation* or *contention* access protocol is mandatory for beacon frame transmission. A frame synchronization sequence, instead, is a pre-defined sequence of modulated bits, designed to have a narrow auto-correlation. Upon reception, time information is extracted by correlating the received signal with the known sequence and locating the maximum. When multiple sequences are received simultaneously, the output

of the correlator shows multiple peaks. As discussed in [43][79] and in Chapter 3, network synchronization can be achieved in the case of signal *superposition* by using a PLL with a proper detector design. Of note, reservation and contention access protocols may also be used with frame synchronization sequences. This chapter focuses on the impact of different medium access protocols on the convergence rate of PLL's.

Given the previous discussion, the following medium access protocols are considered for the signaling slots (see Figure 4.1):

a) *Superposition:* each superframe contains only one signaling slot. At the start of the superframe, each node will choose *randomly* whether to broadcast its time information or listen to its neighbor's transmissions.

b) *Contention:* each superframe contains only one signaling slot. Each node contends for the signaling slot by using some contention access protocol. If there are no collisions, all the neighbor of the transmitting node may receive its time information. A similar approach is employed in the Independent Basic Service Set (IBSS) mode of IEEE 802.11 [24].

c) *Reservation:* each node is assigned a signaling time slot to broadcast time information to its neighbors. A decentralized procedure may be used to allocate signaling slots in order to avoid beacon collisions. Slot allocation is assumed to be fixed and the number of signaling slots in each superframe depends on the spatial reuse factor M [80]. A similar approach was proposed for the ECMA 368 standard [28] and ZigBee networks with cluster-tree topology [29]. This approach is feasible when beacon scheduling is known beforehand (e.g., when the radios are waken up after a prolonged sleep period). Each time slot needs to comprise suitable guard times in order to avoid collisions despite residual clock skews.

A node updates its local clock each time it receives time information within a signaling slot. Convergence time is expressed as the *number of (signaling) slots* necessary to reach

network synchronization up to a prescribed accuracy with respect to the initial conditions. Since a slot corresponds to an iteration of the synchronization protocol, in the following the terms slot and iteration is used interchangeably.

Type 1 Phase-Locked Loop (PLL)

Network time synchronization can be achieved in a completely distributed fashion by deploying each node with a phase-locked loop (PLL). This subsection reviews the basic principles underlying coupled PLL's. In the following, clocks are assumed to be all frequency-synchronous, i.e., for simplicity, it holds that $\alpha_1 = \alpha_2 = \ldots = \alpha_K = 1$. According to the model introduced in Section 2.1, under the assumption of frequency synchronization, the local clock at node i reads $\tau_i(t) = t + \beta_i$, and a simple type 1 PLL is enough in order to achieve network-wide time (phase) synchronization. Let $\tau_i[n]$ be the time displayed by node i's clock at the time the n-th signaling packet is transmitted. At the n-th signaling slot, the output of the time error detector of the local PLL at node i is a linear combination of the time offsets between node i and its neighbors

$$e_i[n] = \frac{1}{\sum_{j\in\bar{\mathcal{N}}_i} a_{ij}} \sum_{j\in\bar{\mathcal{N}}_i} a_{ij}\left(\tau_j[n] - \tau_i[n]\right), \tag{4.1}$$

where $\bar{\mathcal{N}}_i$ is the set of neighbors of i. In the case of reservation and contention access, the timestamp $\tau_j[n]$ is captured at the transmitting side right before beacon transmission, and then inserted in the beacon payload. In the case of superposition access, instead, the error detector needs to be designed in order to produce directly an estimate of the weighted combination of pair-wise offsets (4.1). A suitable detector design to this end was already proposed in [43] (see also Chapter 3 for the case of carrier frequency synchronization). The weights a_{ij} can be arbitrarily chosen [54], but here it is assumed $a_{ij} \in \{0, 1\}$. In practice, the output of the error detector (4.1) is corrupted by additive noise due to, e.g., transmission delays and channel noise. In this chapter the influence of additive noise is neglected, and the focus is on stability analysis (time sync acquisition). In fact, if the system is stable,

additive noise determines only its steady-state accuracy, which will be the subject of the next two chapters. Based on the error $e_i[n]$, the i-th node corrects the local time according to

$$\tau_i^+[n] = \tau_i[n] + \epsilon e_i[n], \tag{4.2}$$

where the parameter ϵ is defined as the *loop gain*. As usual in the analysis of coupled oscillators, the focus is on phase dynamics and the clock phase is defined as $x_i(t) = \tau_i(t) - t$. Also, let $x_i[n]$ be the phase at local time $\tau_i[n]$. From (4.2), it can be seen that phase dynamics obey the following recursion

$$x_i[n+1] = x_i[n] + \frac{\epsilon}{\sum_{j\in\bar{\mathcal{N}}_i} a_{ij}} \sum_{j\in\bar{\mathcal{N}}_i} a_{ij}\left(x_j[n] - x_i[n]\right), \tag{4.3}$$

The update rule (4.3) describes a type 1 discrete-time PLL where the controlled variable is the local clock phase $x_i[n]$. Typically, the system is designed to act as a low-pass filter on $e_i[n]$, and therefore $\epsilon \in (0,1)$. As pointed out in [43][69], the update rule (4.3) can also be seen as an instance of linear consensus algorithms, see, e.g., [48][66]. By reaching *network synchronization*, it is intended that all the K nodes converge to the same value $x_1[\infty] = x_2[\infty] = \ldots = x_K[\infty] = x^*$.

4.2 Reservation Access Protocol

This section focuses on the convergence rate for reservation access protocols. The following is an application of results in [43][81][74], but its inclusion here is necessary for the discussion of Section 4.3. In the case of *reservation* access, a given link is active only during the time slot reserved for its transmissions. The active links at the n-th slot (iteration) may be described by the *directed* graph $\mathcal{G}_r[n] = (\mathcal{V}, \mathcal{E}_r[n])$ and the associated adjacency matrix $\mathbf{A}_r[n]$, where the subscript r stands for "reservation". Notice that the graph $\mathcal{G}_r[n]$ is a subgraph of the connectivity graph $\bar{\mathcal{G}} = (\mathcal{V}, \bar{\mathcal{E}})$, which defines the links that can be activated during any time slot (see Section 2.3). The elements of the adjacency

matrix $[\mathbf{A}_r[n]]_{ij} = a_{ij}^{(r)}[n]$ may be defined as

$$a_{ij}^{(r)}[n] = \begin{cases} 1 & i \neq j, (i,j) \in \mathcal{E}_r[n] \\ 0 & \text{otherwise} \end{cases}. \tag{4.4}$$

If node i does not receive any information from its neighbors during the n-th superframe, i.e., $\sum_j a_{ij}^{(r)}[n] = 0$, it will not update its local time:

$$x_i[n+1] = x_i[n]. \tag{4.5}$$

Otherwise, if $\sum_j a_{ij}^{(r)}[n] \neq 0$, it will update its local time as

$$x_i[n+1] = (1-\epsilon)\,\tau_i[n] + \frac{\epsilon}{\sum_j a_{ij}^{(r)}[n]} \sum_j a_{ij}^{(r)}[n] x_j[n]. \tag{4.6}$$

The vector $\mathbf{x}[n] = [x_1[n], ..., x_K[n]]^T$ contains the local time of all nodes at the n-th iteration. Equations (4.5)-(4.6) can be expressed compactly as the vector equation [48][43]

$$\mathbf{x}[n+1] = \mathbf{W}_r[n]\mathbf{x}[n]. \tag{4.7}$$

If $\epsilon \in (0,1)$, the matrix $\mathbf{W}_r[n]$ is nonnengative and row stochastic since, by construction, $\mathbf{W}_r[n]\mathbf{1} = \mathbf{1}$ for all values of the loop gain.

The slot allocation procedure assures that all links in $\bar{\mathcal{E}}$ are active with a period of M slots, i.e., the graph sequence satisfies

$$\begin{aligned} &\bigcup_{l=1}^{M} \mathcal{G}_r[kM+l] = \bar{\mathcal{G}}, \\ &\mathcal{G}_r[kM+l] = \mathcal{G}_r[nM+l], \quad k \neq m \end{aligned} \tag{4.8}$$

Therefore, for a given slot allocation, the matrix process $\mathbf{W}_r[n]$ is *deterministic* and *periodic*. By exploiting the periodicity of the system matrix, (4.7) can be expressed alternatively as

$$\mathbf{x}[p+1] = \bar{\mathbf{W}}_r \mathbf{x}[p] \tag{4.9}$$

where p is the superframe index (i.e., the time slot index $n = pM$) and $\bar{\mathbf{W}}_r = \prod_{l=1}^{M} \mathbf{W}_r[kM+l]$. Again, the matrix $\bar{\mathbf{W}}_r$ is nonnegative and row stochastic if $\epsilon \in (0,1)$. As shown in [48], a synchronization (consensus) point $\mathbf{x}^* = x^*\mathbf{1}$ is globally asymptotically stable for all initial states $\mathbf{x}[0]$ provided that the graph $\bar{\mathcal{G}}$ is connected.

In order to study the convergence rate of (4.9) to network synchronization, the synchronization error vector $\boldsymbol{\Delta}[n]$ is defined as deviations of the components of $\boldsymbol{\tau}[n]$ from their *instantaneous* average $\mathbf{J}\mathbf{x}[n]$, i.e., $\boldsymbol{\Delta}[n] = (\mathbf{I}-\mathbf{J})\mathbf{x}[n]$, where $\mathbf{J} = \frac{1}{K}\mathbf{1}\mathbf{1}^T$. It is apparent that when the network is synchronized, that is when $\mathbf{x}[n] = x^*\mathbf{1}$, the synchronization error is $\boldsymbol{\Delta}[n] = \mathbf{0}$.

When communication occurs on a fixed graph, the convergence of a MC network of PLL's to network synchronization is *asymptotic* [15], i.e., $\|\boldsymbol{\Delta}[n]\| \to 0$ as $n \to \infty$, for all values of the loop gain ϵ. The only exception is the fully-connected case (one-hop or all-to-all network), where nodes can synchronize to a common time with a single step by taking the average of all other nodes' local time. When the coupling graph is time-varying, instead, convergence may occur in *finite time*, i.e., $\|\boldsymbol{\Delta}[n]\| = 0$ for some $n < \infty$ and ϵ. It has been observed by simulation that finite time convergence occurs typically when $\epsilon = 1$, and in fully connected networks even when $\epsilon < 1$. Nevertheless, full connectivity is impractical in large networks, and choosing $\epsilon = 1$ leads to instability for some beacon slot allocations[1]. Therefore, in the following the focus is on asymptotic convergence. When convergence is asymptotic, the convergence time in terms of number of iterations can be defined as

$$T_r = \frac{\delta}{R_r}, \tag{4.10}$$

where the factor $\delta = \log_{10}\left(\|\boldsymbol{\Delta}[0]\| \,/\, \|\boldsymbol{\Delta}[T]\|\right)$ depends on the synchronization accuracy requirements with respect to initial conditions, and the convergence rate R_r can be found

[1]Setting $\epsilon = 1$ leads to oscillatory steady-state dynamics whenever the sequence $\mathcal{G}_r[n]$ routes sync information over cycles.

to be (see also [63])

$$R_r = -\sup_{\mathbf{\Delta}[0]} \lim_{p\to+\infty} \frac{1}{M} \log_{10} \|\mathbf{\Delta}[p]\|^{\frac{1}{p}}, \tag{4.11}$$

It can be shown after some algebra that

$$R_r = -\frac{1}{M} \log_{10} \rho\left(\bar{\mathbf{W}}_r - \mathbf{J}\right) = -\frac{1}{M} \log_{10} \lambda_2\left(\bar{\mathbf{W}}_r\right), \tag{4.12}$$

where the second equality follows from the fact that the spectral radius $\rho\left(\bar{\mathbf{W}}_r - \mathbf{J}\right)$ coincides with $\lambda_2\left(\bar{\mathbf{W}}_r\right)$, the second largest eigenvalue of $\bar{\mathbf{W}}_r$. It can be seen from (4.12) that the use of reservation access entails a cost given by the spatial reuse factor M, which reduces the actual convergence rate. Also, different slot allocations will provide different convergence rates.

4.3 Contention and Superposition Access Protocols

In the case of *contention* or *superposition* access, the active links at the n-th iteration are chosen at random within $\bar{\mathcal{E}}$. As before, the active links at the n-th slot (iteration) are described by the *directed* graph $\mathcal{G}_m[n] = (\mathcal{V}, \mathcal{E}_m[n])$ and the associated adjacency matrix $\mathbf{A}_m[n]$, where the subscript indicates whether it refers to contention ($m = c$) or superposition ($m = s$). The vector difference equation that describes the dynamics of the PLL's with contention or superposition access can be written as

$$\mathbf{x}[n+1] = \mathbf{W}_m[n]\mathbf{x}[n], \tag{4.13}$$

where $\mathbf{W}_m[n]$ is now a *random* matrix process. Contention and superposition make for different properties of the directed graph $\mathcal{G}_m[n]$:

- *Contention:* A simple contention protocol is assumed, which is inspired by the one employed in [24]. At the n-th iteration, each node chooses to transmit with the same probability p_c. Let $\mathcal{A}_c[n]$ be the set of transmitting nodes. When node j transmits ($j \in \mathcal{A}_c[n]$), its time information may be received by all receiving

neighbors (half-duplex constraint), i.e., $\left\{i | i \notin \mathcal{A}_c[n], i \in \bar{\mathcal{N}}_j\right\}$. It is assumed that a transmission over a directed edge (i, j) does not incur a collision whenever no other node in the neighborhood of i is transmitting[2], i.e., $\bar{\mathcal{N}}_i \cap \mathcal{A}_c[n] = j$. This can be summarized as

$$\mathcal{E}_c[n] = \left\{(i, j) | j \in \mathcal{A}_c[n], i \notin \mathcal{A}_c[n], \bar{\mathcal{N}}_i \cap \mathcal{A}_c[n] = j\right\}. \tag{4.14}$$

Notice that, in this case, each node will receive time information from *at most one* of its neighbors at a time.

- *Superposition:* At the n-th iteration, each node chooses to transmit at random with the same probability p_s. The active edges are

$$\mathcal{E}_s[n] = \left\{(i, j) | j \in \mathcal{A}_s[n], i \notin \mathcal{A}_s[n], i \in \bar{\mathcal{N}}_j\right\}. \tag{4.15}$$

Therefore, a receiving node $i \notin \mathcal{A}_s[n]$ will receive time information from *all* its transmitting neighbors.

Since, for both protocols, $\mathcal{G}_m[n]$ is the outcome of a *random* construction (i.e., it is a random subgraph of $\bar{\mathcal{G}}$), (4.13) is a random linear dynamical system (while (4.9) was a deterministic system). Given that $\mathcal{G}_m[n]$ is constructed in a i.i.d. fashion at each iteration, the sequence of system matrices $\{\mathbf{W}_m[n]\}_{n=0}^{+\infty}$ is a (stationary) random matrix process. Similarly to the deterministic case, if $\epsilon \in (0, 1)$ $\mathbf{W}_m[n]$ is nonnegative and row stochastic, i.e., $\mathbf{W}_m[n]\mathbf{1} = \mathbf{1}$. Since $x^*\mathbf{1}$ (for some $x^* \in \mathbb{R}$) is a fixed point of (4.13), whenever the algorithm reaches a network synchronization state, it would not leave it.

In the following, conditions for almost sure convergence of (4.13) will be provided, along with a lower bound to the convergence rate of (4.13) for both contention and superposition protocols.

[2]More complex collision conditions based on the interference graph or on the signal to interference and noise ratio (SINR) may be considered; nevertheless, here this simple criterion is chosen for its wide adoption in many practical algorithms and standards, see, e.g., [28][25].

4.3.1 Almost Sure Convergence

Let $\bar{\mathbf{A}}$ be the adjacency matrix associated with the connectivity graph $\bar{\mathcal{G}}$. Also, let the Laplacian matrix associated to $\bar{\mathbf{A}}$ be defined as $\bar{\mathbf{L}} = \bar{\mathbf{D}} - \bar{\mathbf{A}}$, where $\bar{\mathbf{D}} = \mathrm{diag}\,(d_1, d_2, \ldots, d_K)$ and $d_i = \sum_j [\bar{\mathbf{A}}]_{i,j}$. The following result from [63] provides a necessary and sufficient condition for almost sure (a.s.) convergence of (4.13) to network synchronization.

Theorem ([63])**.** *The random dynamical system* (4.13) *converges a.s. to network synchronization iff the average system*

$$\mathbf{x}[n+1] = E\left[\mathbf{W}_m[n]\right]\mathbf{x}[n]. \tag{4.16}$$

converges to network synchronization.

Therefore, the study of a.s. convergence boils down to the study of convergence of the average system (4.16).

Lemma. *The average system matrix for contention and superposition access is given by (the index n is dropped for notational convenience)*

$$E\left[\mathbf{W}_m\right] = \left(\mathbf{I} - \epsilon \mathbf{D}_m\left(p_m\right)\right) + \epsilon \mathbf{D}_m\left(p_m\right)\bar{\mathbf{D}}^{-1}\bar{\mathbf{A}}, \tag{4.17}$$

where $\mathbf{D}_m\left(p_m\right)$ is a diagonal matrix. For contention

$$\left[\mathbf{D}_c\left(p_c\right)\right]_{ii} = 1 - p_c - \left(1 - p_c\right)\left(1 - d_i p_c \left(1 - p_c\right)^{d_i+1}\right), \tag{4.18}$$

while for superposition

$$\left[\mathbf{D}_s\left(p_s\right)\right]_{ii} = 1 - p_s - \left(1 - p_s\right)^{d_i+1}. \tag{4.19}$$

For $p_m \in (0,1)$, $\epsilon \in (0,1)$, $E\left[\mathbf{W}_m\right]$ satisfies the following:

$$E\left[\mathbf{W}_m\right]\mathbf{1} = \mathbf{1},\ \rho(E\left[\mathbf{W}_m\right] - \mathbf{J}) < 1, \tag{4.20}$$

and in both cases the average system (4.16) *converges to network synchronization. The convergence rate of the average system* (4.16) *is given by*

$$\begin{aligned} R_m^{\text{ave}} &= -\log_{10}\rho(E\left[\mathbf{W}_m\right]-\mathbf{J}) \\ &= -\log_{10}\lambda_2\left(E\left[\mathbf{W}_m\right]\right). \end{aligned} \tag{4.21}$$

In the following, it will be shown that the rate of convergence in average is also strictly connected to the actual convergence rate of (4.13).

4.3.2 Convergence Rate

As in Section 4.2, the error vector is defined as $\boldsymbol{\Delta}[n]=(\mathbf{I}-\mathbf{J})\boldsymbol{\tau}[n]$. In case of contention or superposition access, the dynamics of $\boldsymbol{\Delta}[n]$ are ruled by the following recursion

$$\begin{aligned} \boldsymbol{\Delta}[n] &= (\mathbf{W}_m[n]-\mathbf{1}\mathbf{v}_m^T[n])\boldsymbol{\Delta}[n-1] \\ &= \mathbf{P}_m[n]\boldsymbol{\Delta}[n-1], \end{aligned} \tag{4.22}$$

where $\mathbf{v}_m^T[n]=\frac{1}{N}\mathbf{1}^T\mathbf{W}_m[n]$. It can be seen that (4.22) is again a random dynamical system, and it can be shown that its rate of convergence is determined by a *deterministic* constant γ_m (Lyapunov exponent [82][78])

$$R_m=-\sup_{\boldsymbol{\Delta}[0]}\lim_{n\to+\infty}\log_{10}\|\boldsymbol{\Delta}[n]\|^{\frac{1}{n}}=-\log\gamma_m. \tag{4.23}$$

The Lyapunov exponent γ_m cannot be computed in closed form except in some cases. Nevertheless, as proved in [78], it can be lower bounded with

$$-\log\gamma_m\geq-\frac{1}{2}\log_{10}\rho\left(E\left[\mathbf{P}_m\otimes\mathbf{P}_m\right]\right), \tag{4.24}$$

where $\otimes$ denotes the Kronecker product. As a consequence, the convergence time may be upper bounded as

$$T_m\leq\frac{\delta}{-\frac{1}{2}\log_{10}\rho\left(E\left[\mathbf{P}_m\otimes\mathbf{P}_m\right]\right)}. \tag{4.25}$$

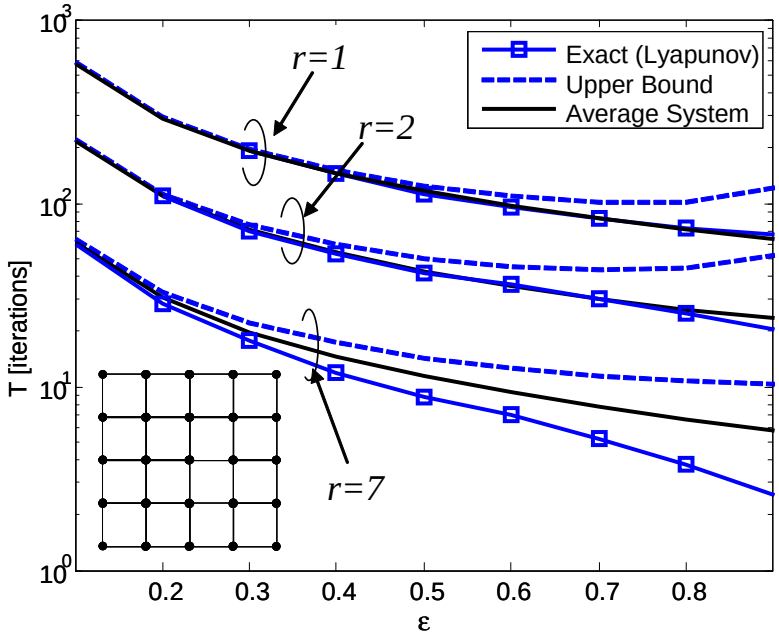


Figure 4.2 Comparison of convergence time computed by the Lyapunov exponent γ_c with respect to possible approximations versus the loop gain ϵ and for increasing transmission range r. Square grid network of 5×5 nodes at unit distance, contention access protocol.

Unfortunately, the bound (4.24) is very challenging to compute for the algorithms at hand. Figure 4.2 refers to contention access and a square grid network of $K = 25$ nodes deployed at unit distance from each other. The convergence time computed by the Lyapunov exponent γ_c is compared with the upper bound (4.25) and the convergence time of the average system $T_c^{\text{ave}} = (\delta / -\log_{10} \lambda_2 (E[\mathbf{W}_c]))$, as a function of the loop gain ϵ and increasing transmission range r (the Lyapunov exponent is computed by using the methods in [78]). The accuracy chosen for convergence is $\delta = 1$ (i.e., $\|\mathbf{\Delta}[T]\| = \|\mathbf{\Delta}[0]\| / 10$) and the transmission probability p_c is optimized so that to maximize the probability of a given edge to be active. It can be seen that, for this sample network, the convergence rate of the average system is the closest approximation for the Lyapunov exponent. Similar results hold for superposition access and for other topologies, such as a random geometric network of 25 nodes deployed in a unit square. From this numerical analysis, it is concluded that the convergence rate R_m^{ave} is a reasonable approximation of the actual rate of convergence

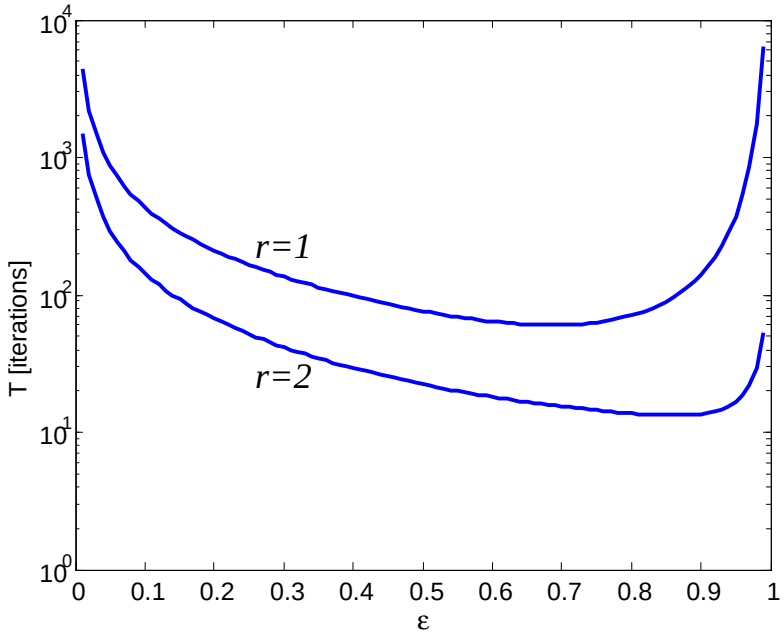


Figure 4.3 Average convergence time of reservation access protocol versus the loop gain ϵ for increasing transmission range r. Regular square grid network of 5×5 nodes.

(4.23) for the considered topology. In the next section, the rate of convergence R_m^{ave} will be used to choose the loop gain ϵ for contention and superposition access protocols.

4.4 Simulation Results

This section presents simulation results in order to compare the convergence rate of the access protocols introduced in Section 4.1. A regular square grid network of 5×5 nodes is considered, where the distance between nodes on the grid is 1 m. The same collision conditions as those in Section 4.3 are considered for both the reservation and contention access protocols. For a given transmission radius r, the transmission probability for contention and superposition access, p_c and p_s, are optimized so as to maximize the probability of a generic edge to be active. Required accuracy is chosen as $\delta = 1$.

Looking for the value of the loop gain ϵ that maximizes the convergence rate for each of the considered protocols, Figure 4.3 depicts the *average* convergence time for the reservation access protocol of Section 4.2 as a function of the loop gain ϵ for increasing

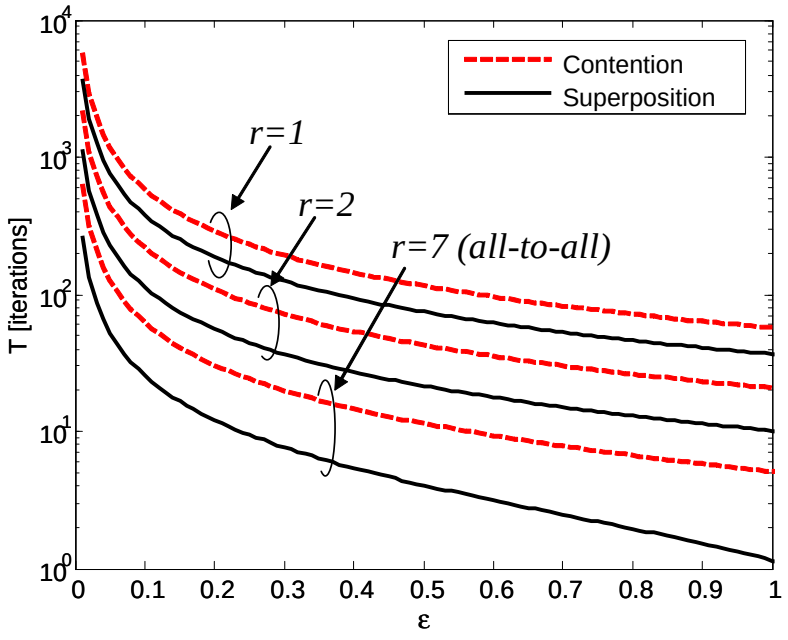


Figure 4.4 Convergence time of random access protocols versus the loop gain ϵ for increasing transmission range r. Regular square grid network of 5×5 nodes.

transmission radius r. Notice that, for a given network topology, the convergence time (4.10) depends on the specific signaling slot allocation. In [57], the average convergence time was computed by averaging the actual convergence time (4.10) only for the fastest slot allocations. Differently from [57], Figure 4.3 depicts the average convergence time of a larger set of feasible allocations[3]. Interestingly, $\epsilon = 1$ is not a good choice, since it results in PLL instability with some allocations. From Figure 4.3, the optimal loop gain ϵ is 0.7 and 0.9 for $r = 1, 2$, respectively. If $r = 7$ (not shown), each node is able to communicate with any other node in the network (all-to-all coupling), and the reservation algorithm allocates a different slot to every node (i.e., $M = 25$). In this case, if $\epsilon > 0.5$, all nodes are perfectly synchronized (i.e., $\|\mathbf{\Delta}[n]\| = 0$) within M iterations and the definition of asymptotic convergence time (4.10) may not be applied.

Following the discussion in Section 4.3.2, Figure 4.4 depicts the convergence time in average as an approximation of the actual convergence time of random access protocols

[3]In particular, feasible allocations are found by employing a random vertex coloring algorithm [83].

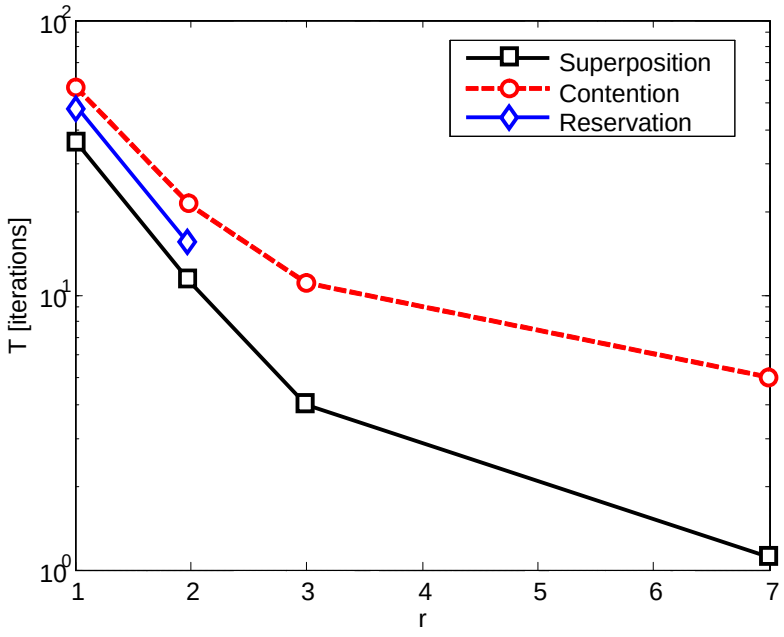


Figure 4.5 Convergence time for different access protocols versus the transmission range r. Regular square grid network of 5×5 nodes.

for the network under analysis. It can be seen that the convergence time is a decreasing function of the loop gain for random access protocols, and therefore the best ϵ is close to unity.

Finally, Figure 4.5 depicts the convergence time as a function of the transmission range r for all the considered medium access protocols. The loop gain for random access protocols is chosen as $\epsilon = 0.95$. It can be seen that superposition access is able to exploit signaling resources more efficiently than contention access. With respect to the reservation protocol, a special remark needs to be done regarding connectivity. In fact, when the network is sufficiently well connected, it has been observed that it is in general easy to find feasible allocations allowing convergence in finite time. In particular, for the specific network considered here, when $r > 2$ all the considered slot allocations were able to guarantee finite-time convergence. The curve for reservation access is therefore truncated at $r = 2$ in Figure 4.5 since, when finite-time convergence occurs, the definition (4.10) is not applicable.

4.5 Conclusions

This chapter considered the application of distributed synchronization based on phase-locked loops (PLL) at the MAC layer. The focus was on the impact of the medium access protocol employed for signaling purposes on the convergence rate of PLL's. Three access protocol were considered: reservation, contention and superposition. While the reservation protocol allocates a dedicated signaling slot for each node, contention and superposition protocols are both random access schemes. Convergence in the mean has been studied for contention and superposition, as it is a necessary and sufficient condition for almost sure convergence to network synchronization. Also, it was shown by simulations that, for a sample square grid network, the rate of convergence in average is a good approximation to the actual convergence rate of these schemes. Finally, the considered protocols have been compared to each other with respect to the convergence speed, showing that superposition is the most efficient random access protocol for synchronization purposes.

CHAPTER 5

ACCURACY OF DISTRIBUTED SYNCHRONIZATION

A TDMA protocol reserves dedicated resources for each link and requires tight network synchronization in order to avoid unwanted collisions and manage interference. Synchronization in a TDMA network can be achieved by employing a MS protocol such as FTSP or by a MC scheme as detailed in Chapter 4. After the network setup phase, each node keeps in sync with the network by tracking the periodic transmission of signaling packets (beacon frames) at frame start. In This chapter focuses on this regime and analyzes the steady-state accuracy of distributed synchronization algorithms. Clocks are characterized by two parameters: timing offset (or *phase* offset) and timing skew (or *frequency* offset). Synchronization algorithms aim at correcting phase and frequency offsets so that all clocks in the network display the same time. In particular, distributed synchronization algorithms may be categorized into open-loop and closed-loop techniques, depending on how the clock correction operation is performed.

Open-loop algorithms estimate the parameters of the local clock employing observations of pair-wise timing offsets. Local time is translated to network (absolute) time by compensating each clock sample (or *timestamp*) for phase and frequency offsets. The most popular synchronization protocol based on the MS architecture is the reference broadcast synchronization (RBS) protocol [38]. RBS corrects phase and frequency offsets in a MS network by applying linear regression techniques. Improvements to the original RBS protocol may be found in [39]. The algorithm proposed in [41], instead, applies to hybrid networks. This technique is based on writing the clock parameter estimation problem as a linear system of equations, which is then solved by a distributed iterative algorithm.

Closed-loop synchronization algorithms control the local clock directly by dynamically tuning the local clock frequency. The local controller adapts the frequency correction

applied to the clock according to the observed pair-wise time offsets. Most of the algorithms following this approach are close relatives of distributed phase-locked loops (PLL) systems widely employed in synchronous digital circuit-switched networks [15][13]. The clock control law adopted by the Network Time Protocol (NTP) [17], is based on a discrete-time PLL, and it is tailored for MS topologies. Recently, [45] has studied the stability of a clock control algorithm similar to NTP on a MC network. Finally, the algorithm developed in [46] follows a hybrid approach by enhancing a closed-loop phase correction with an open-loop frequency estimator.

This chapter focuses on synchronization at the MAC layer by considering a TDMA (i.e., collision-less) protocol for medium access. In addition to MC and MS topologies, a hybrid network architecture is considered, whereby master nodes are deployed in a peer-to-peer coupled network. A stable clock model is assumed, whereby local time needs to be compensated only for phase and frequency offsets. Analysis considers the synchronization accuracy attained by an open-loop algorithm based on distributed linear regression and a closed-loop algorithm based on distributed PLL's. The performance of practical distributed algorithms is also compared with the Cramér-Rao lower bound (CRLB) for the problem at hand. Results show that peer-to-peer topologies are inefficient with respect to the CRLB, whereas the MS hierarchical architecture is able to achieve the accuracy limit. Nevertheless, the performance of MC and hybrid topologies improves rapidly when increasing network connectivity, while MS proves to be optimal in poorly connected networks.

5.1 System Model

5.1.1 Beacon-enabled TDMA MAC Protocol

This chapter analyzes the asymptotic (steady-state) accuracy of distributed algorithms for time synchronization at the MAC layer. Since the focus on the steady-state regime, nodes are assumed to have already acquired raw network-wide time synchronization. Given this

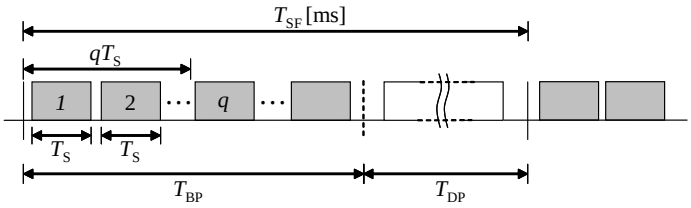


Figure 5.1 Super-Frame structure for a general beacon-enabled TDMA MAC protocol.

assumption, a beacon-enabled TDMA MAC protocol is considered, whereby the time axis is divided into super-frames of duration T_{SF} seconds as in Figure 5.1. Each superframe in divided into a beacon period (BP) and a data period (DP), that are further divided into time-slots. Time-slots within the DP are employed for data transmission, while time-slots within the BP (or *beacon slots*) are employed for transmission of signaling information (beacon *frames*). No assumption is made on channel access during the DP, as that portion of the super-frame is irrelevant for the synchronization function. On the other hand, beacon frames are assumed to be transmitted on a reservation basis, whereby a distributed or centralized scheduling algorithm assigns a beacon slot to each node. A given beacon slot may be reused by non-interfering nodes and the number of beacon slots (i.e., the length of the BP) needed for collision-less transmission depends on the transmission range r (i.e., on node density). Each slot has a fixed duration T_S and it is associated to an integer index q, so that its nominal time offset from super-frame start is qT_S seconds. When the local clock of node j marks the start of the beacon slot reserved for it, j broadcasts its beacon frame to its neighbors (accounting for appropriate guard times). Upon receiving a beacon frame at time t, node i associates it with a timestamp $\tau_i(t)$, obtained by reading its local clock. It is assumed the use of accurate *hardware timestamps*: the timestamp is captured from the local clock upon reception of the first symbol of the start-of-frame delimiter (SFD) sequence at the physical layer interface. It is important to remark that the exchange of synchronization information through periodic beacon transmissions is strictly *unidirectional*, i.e., nodes do not communicate each other the observed timestamps. As detailed in the following, this factor has a major impact on the attainable accuracy of distributed synchronization

algorithms. On the other hand, beacon frames can include other signaling information, e.g., the estimated accuracy of the local oscillator. All nodes need to have their radios turned on during the BP, while they are allowed to sleep during the DP. This communication and synchronization model is motivated by recent solutions proposed for general multi-hop mesh networks [28] and applies (with little modifications) also to sensor networks with a cluster-tree architecture, such as ZigBee networks [29].

It has to be noted that the synchronization algorithms studied here could be adapted with minor changes to contention-based medium access protocols. In that case, synchronization error analysis would need to account for asynchronous transmissions and beacon collisions, thus making the TDMA scenario presented here somewhat optimistic. The next chapter will focus on synchronization in non-beacon enabled networks, where time information is exchanged along with data and ACK packets (as in the time-synchronized mesh protocol - TSMP [31]). Finally, the proposed algorithms are not confined to the MAC layer, as they could be implemented (with minor changes) at every layer of the protocol stack, in particular at the application layer. At higher layers, large network delays need to be continuously compensated by handshaking procedures such as those specified by IEEE 1588/PTP or NTP time transfer protocols. The interested reader could refer to [39] for a review of handshaking procedures suited for wireless networks that can be employed alongside the proposed algorithms.

5.1.2 Observation Model

In order to develop an observation model suitable for the development of synchronization algorithms, consider two sample nodes, j and i. Assume that they do not employ any synchronization technique and let their clocks running freely. Node j broadcasts a beacon periodically, whenever its local clock strikes $\tau_j\left(t_j[n]\right) = nT_{SF} + q_jT_S$, where $t_j[n]$ is the *absolute* beacon transmission time (see Figure 5.2). Alternatively, beacon transmission is triggered when node j's clock increases by an amount equal to T_{SF} with respect to the last

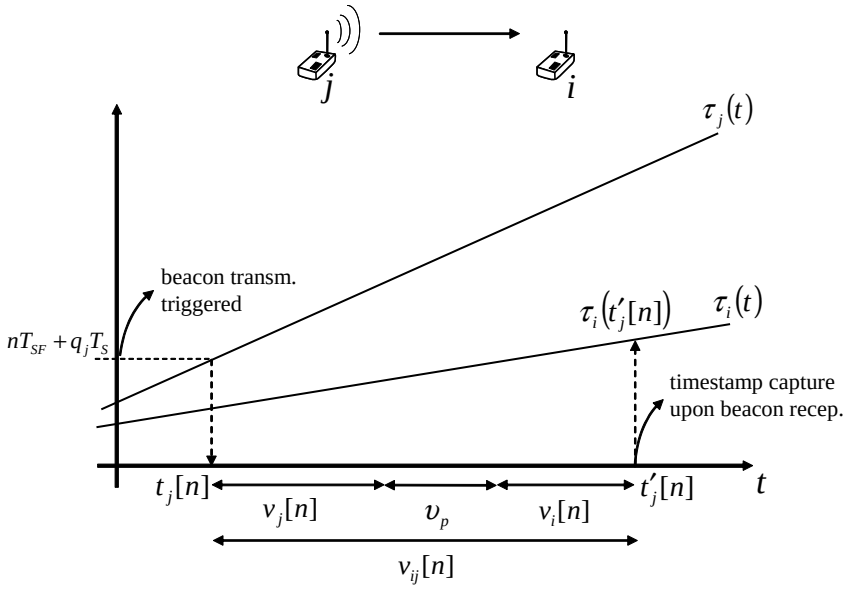


Figure 5.2 Timestamping procedure at a receiving node and main sources of delivery delays. Clock skews are exaggerated for clarity.

beacon transmission time,

$$\tau_j(t_j[n]) - \tau_j(t_j[n-1]) = T_{SF}, \tag{5.1}$$

where $\tau_j(t_j[0]) = q_j T_S$. The recursive definition of the beacon transmission time (5.1) is particularly important for the design of *iterative* synchronization algorithms. The reception of a beacon frame triggers a timestamp $\tau_i(t'_j[n]) = \tau_i(t_j[n] + v_{ij}[n])$ at node i. The reception timestamp $\tau_i(t_n + v_{ij}[n])$ depends on the delivery time delay $v_{ij}[n]$ (or *channel jitter*[1]). The delivery time is due to the accumulation of both deterministic and random delay components. In general, delivery time is the sum of send time and medium access time $v_j[n]$ (random), channel propagation (time-of-flight) and transmission time ν_p (deterministic), and finally receive time $v_i[n]$ (random) (see [37] for a detailed description). For distances below 300 m, the propagation time is negligible (it is less than 1 μs). In case hardware-assisted timestamps are implemented (as prescribed by IEEE 802.15.4 [25] and IEEE 802.11 [24] specifications), random delay components can be neglected. The

[1]The random component of the delivery time delay is also called packet delay variation (PDV) in the literature on synchronization on packet-switched networks.

residual jitter boils down to the errors in compensating deterministic delays and residual circuit delays at both transmitting and receiving side.

In the following, clocks are assumed to be perfectly stable within a super-frame (frequency drift and phase noise are negligible). If quantization noise is neglected as well, the clock at node i may be modeled as

$$\tau_i(t) = (1 + \alpha_i)\, t + \beta_i, \tag{5.2}$$

where β_i is the initial phase and the clock frequency is $1 + \alpha_i$. From (5.2), the epoch t can be recovered from local time at node j as $t(\tau_j) = \frac{1}{1+\alpha_j}(\tau_j - \beta_j)$. By plugging the transmission epoch $t_n(\tau_j)$ in the expression for the reception timestamp $\tau_i\left(t_j[n] + v_{ij}[n]\right)$, it is obtained

$$\tau_i\left(t_j[n] + v_{ij}(t_n)\right) = \frac{1+\alpha_i}{1+\alpha_j}\tau_j(t_j[n]) + \left(\beta_i - \frac{1+\alpha_i}{1+\alpha_j}\beta_j\right) + (1+\alpha_i)\, v_{ij}[n]. \tag{5.3}$$

The observation model (5.3) has been studied in many works on synchronization (see, e.g., [46]). From the super-frame model of Section 5.1.1, the local time at node j is known[2] to be $\tau_j(t_j[n]) = nT_{SF} + q_j T_S$. In the following it is considered $\tau_j(t_j[n]) \simeq nT_{SF}$, as this approximation is irrelevant for accuracy analysis. The model may be simplified by noting that, since $|\alpha_k| \ll 1$ for $\forall k$, $\frac{1+\alpha_i}{1+\alpha_j} \simeq 1 + (\alpha_i - \alpha_j)$, and (5.3) can be approximated as

$$\tau_i\left(t_j[n] + v_{ij}[n]\right) \simeq nT_{SF} + (\alpha_i - \alpha_j)\left(nT_{SF} - \beta_j\right) + (\beta_i - \beta_j) + v_{ij}[n]. \tag{5.4}$$

The local time at node j, nT_{SF}, is a known constant and does not add any useful information by itself. Therefore, it is possible to focus on pair-wise *time offsets*, $o_{ij}[n] = nT_{SF} - \tau_i\left(t_j[n] + v_{ij}[n]\right)$. By defining $w_{ij}[n] := -v_{ij}[n]$, the model for time offset observations

[2]Other protocols, such as IEEE 1588 and NTP, do not require to perform time information exchanges with a fixed and known periodicity. In those cases the packet needs to be timestamped *also at transmitting side*, since $\tau_j(t_j[n])$ is unknown to the receiver. The transmission timestamp $\tau_j(t_n)$ is then inserted either in the packet payload or in a following signaling packet, thereby increasing synchronization overhead.

may be written as

$$o_{ij}[n] = (\alpha_j - \alpha_i)\,(nT_{SF} - \beta_j) + (\beta_j - \beta_i) + w_{ij}[n]. \tag{5.5}$$

If initial time offsets are bounded within a finite interval (e.g., $\beta_j \in [0, T_{SF}]$), $nT_{SF} \gg \beta_j$ for sufficiently large n, and the observation model (5.5) may be approximated with the following

$$o_{ij}[n] \simeq (\alpha_j - \alpha_i)\, nT_{SF} + (\beta_j - \beta_i) + w_{ij}[n]. \tag{5.6}$$

In practical systems, the reception timestamp $\tau_i\left(t_n + v_{ij}(t_n)\right)$ is quantized. Overall, considering residual channel jitter and timestamp quantization, the random noise $w_{ij}[n]$ is assumed to be a i.i.d. random process with variance $E\left[\left(w_{ij}[n]\right)^2\right] = \sigma_{ij}^2$. Notice that the average noise is non-zero, $E\left[w_{ij}[n]\right] \neq 0$, due to residual deterministic delays on the link between node j and node i. In the following, it is assumed that any residual deterministic delay may be accurately measured and compensated beforehand[3], so that $E\left[w_{ij}[n]\right] = 0$. Typically, the observation noise is assumed to follow either a Gaussian or an exponential distribution [39]. If the transmitting node j is a master node deployed with an accurate time reference (e.g., GPS), its clock is assumed be matched perfectly to absolute time, i.e., $\tau_j(t) = t$ and $\alpha_j = \beta_j = 0$.

When a synchronization algorithm is employed, the local clock $\tau_j(t)$ is compensated for the estimated phase and frequency offsets, and the resulting *corrected* clock $s_i\left(t\right)$ is employed as the timer for MAC layer tasks. To clarify this point, consider the simple case of beacon transmission from node j to node i in Figure 5.2. Similarly to (5.1), node j evaluates the time interval between successive beacon transmissions by employing the corrected clock,

$$s_j(t_j[n]) - s_j(t_j[n-1]) = T_{SF}. \tag{5.7}$$

[3]Deterministic delays may be estimated through a preliminary network calibration procedure consisting in series of timing handshakes between neighbors.

When a beacon is received, node i captures the local clock value $\tau_i(t'_j[n])$ and corrects it to obtain the corrected timestamp $s_i(t'_j[n])$. Finally, the corrected time offset $o_{ij}[n] = nT_{SF} - s_i\left(t'_j[n]\right)$ retains the same expression of (5.6), but it depends on the *residual* phase and frequency offsets after the clock correction operation.

In order to simplify the treatment in the following sections, a simplified notation is introduced: with reference to events within the n-th super-frame, $\tau_{ij}[n] = \tau_i(t'_j[n])$ and $s_{ij}[n] = s_i(t'_j[n])$ indicate the beacon transmission time from node j according to node i's local and corrected clocks, while $\tau_{ii}[n] = \tau_i(t_i[n])$ and $s_{ii}[n] = s_i(t_i[n])$ are the beacon transmission time from node i according to its local and corrected clocks.

5.2 Distributed Clock Control: Closed-loop Synchronization

This section analyzes a synchronization algorithm that is based on the distributed control of the clock ensemble. Consider a simple master-slave system. Given the observation (5.6), the internal model principle suggests that a PI controller is needed to drive to zero the static error with respect to the master. In the synchronization nomenclature, a PI control law corresponds to a type 2 phase-locked loop (PLL) [6][15]. In the following, it will be shown that a type 2 PLL is able to compensate both frequency and phase offsets also in a general synchronization network[4].

When a synchronization algorithm is employed, beacon transmissions are triggered according to the *corrected* clock $s_i[n]$. A PLL applies a *linear correction* to the local clock, so that $s_{ii}[n] = \tau_{ii}[n] + p_i[n]$, where $p_i[n]$ is the local correction term. From the discussion in Section 5.1.2, it holds that $s_{ii}[n] - s_{ii}[n-1] = T_{SF}$, and it can be seen by simple manipulations that the transmission time of the n-th beacon frame with respect to the *local*

[4]Notably, a type 2 PLL is employed also by NTPv3 [17]. NTPv3 implements an *asynchronous* correction mechanism for a (software) system clock in order to provide an accurate time-of-day service. The presented algorithm, instead, entails *periodic* adjustments of the MAC layer clock employed to coordinate medium access.

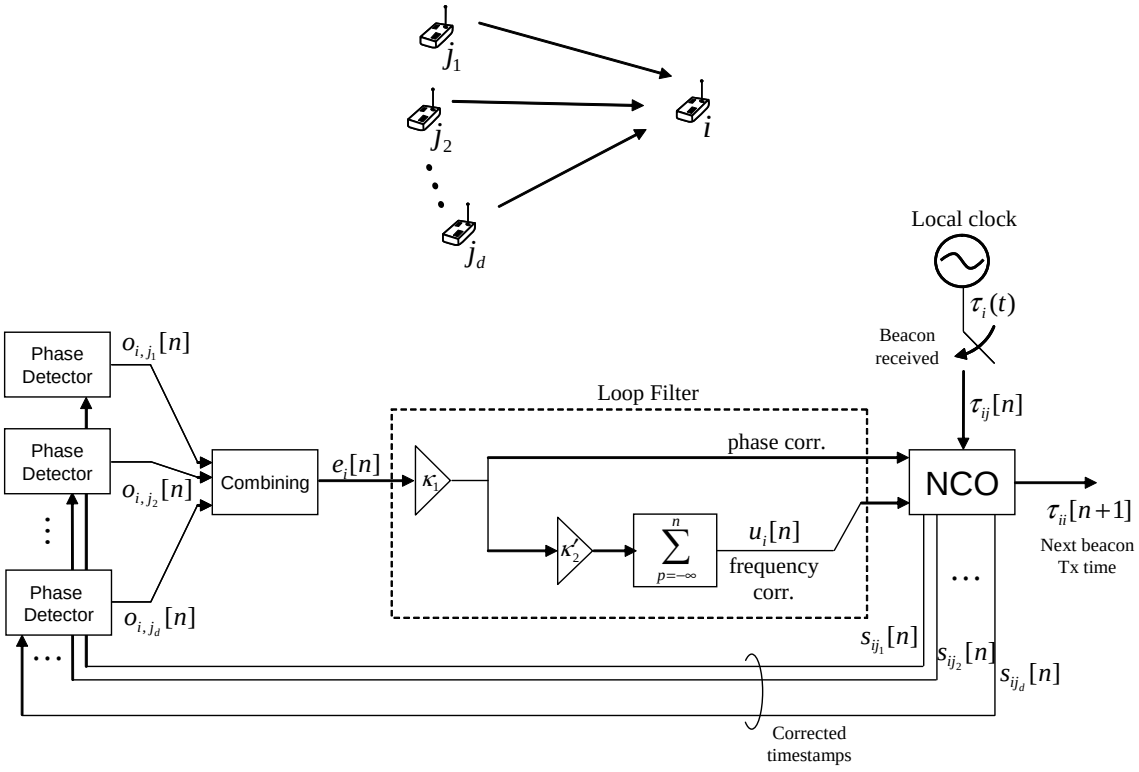


Figure 5.3 Block diagram of the PLL synchronization algorithm at node i. The algorithm is modeled as a discrete-time system operating with sample period T_{SF}.

clock satisfies

$$\tau_{ii}[n] - \tau_{ii}[n-1] = T_{SF} - (p_i[n] - p_i[n-1])\,. \tag{5.8}$$

By comparing (5.8) with (5.1), it is seen that the PLL changes dynamically (*closed-loop* synchronization) the *frequency* of beacon transmission events as in a variable frequency oscillator (VFO). It is important to stress that (5.8) allows to determine the corrected beacon transmission times with respect to the local clock by employing the conventional compare function of hardware clock registers [21]. No direct correction of the local clock is requested (i.e., no timing advance or periodic tick removal/addition). When a PI (type 2) loop filter is utilized, the clock correction is computed as (see Figure 5.3)

$$p_i[n] = p_i[n-1] + \kappa_1 e_i[n-1] + T_{SF} u_i[n-1], \tag{5.9}$$

where the *phase correction* term $\epsilon e_i[n]$ is provided by the proportional branch of the loop filter, while the *frequency correction* term $T_{SF}u_i[n]$ provided by the integral branch is

$$u_i[n] = u_i[n-1] + \kappa_2'\epsilon e_i[n-1]. \tag{5.10}$$

Notice that the dimension of $u_i[n]$ is [μs/μs], the integral gain κ_2' is expressed in [μs^{-1}] and the gain κ_1 is dimensionless. The error $e_i[n]$ is computed by the combining block as a linear combination of pair-wise time offsets[5]

$$e_i[n] = \frac{1}{d_i}\left[\sum_{j=K_u+1}^{K} m_{ij}o_{ij}[n] + \sum_{j=1}^{K_u} a_{ij}o_{ij}[n]\right], \tag{5.11}$$

where the *in-degree* of node k, d_k, is defined as the sum of the weights of all *incoming* links at node k, $d_i = \sum_{j=K_u+1}^{K} m_{ij} + \sum_{j=1}^{K_u} a_{ij}$ and the weights m_{ij}, a_{ij} are subject to design choice. In particular, each node may arbitrarily decide whether to listen or not to each of its neighbors (i.e., $a_{ij} \in \{0,1\}$, $m_{ij} \in \{0,1\}$). Notice that pair-wise time offsets $o_{ij}[n]$ need to be computed with respect to the corrected clock. For this purpose, when a beacon is received from node j during the n-th super-frame, node i captures the local clock value $\tau_{ij}[n]$ and computes the corrected timestamp $s_{ij}[n]$ as $s_{ij}[n] = \tau_{ij}[n] + p_{ij}[n]$, where $p_{ij}[n]$ is the timestamp correction term. It will be shown in the next section that a reasonable choice for timestamp correction is $p_{ij}[n] = p_i[n] + u_i[n-1]\,(q_j - q_i)\,T_S$. The corrected reception timestamp $s_{ij}[n]$ and the next beacon transmission time $\tau_{ii}[n]$ in (5.8) may be computed by the NCO circuit in Figure 5.4. The start time of any time-slot within the DP of the n-th super-frame may be evaluated with respect to the local clock as $\tau_{ii}\,[n;q] = \tau_{ii}[n] + \left(1 - u_i[n] - \kappa_1\frac{e_i[n]}{T_{SF}}\right)(q - q_i)\,T_S$, where q is the index of the time-slot of interest.

A necessary condition for the clock control algorithm to work properly is that $\tau_{ii}[n] > \tau_{ii}[n-1]$, or, equivalently, that the corrected clock $s_{ii}[n]$ is monotonically increasing (time

[5]In general, the phase detector can include additional memory and filtering. Phase detectors may also implement outlier detection for security purposes [17].

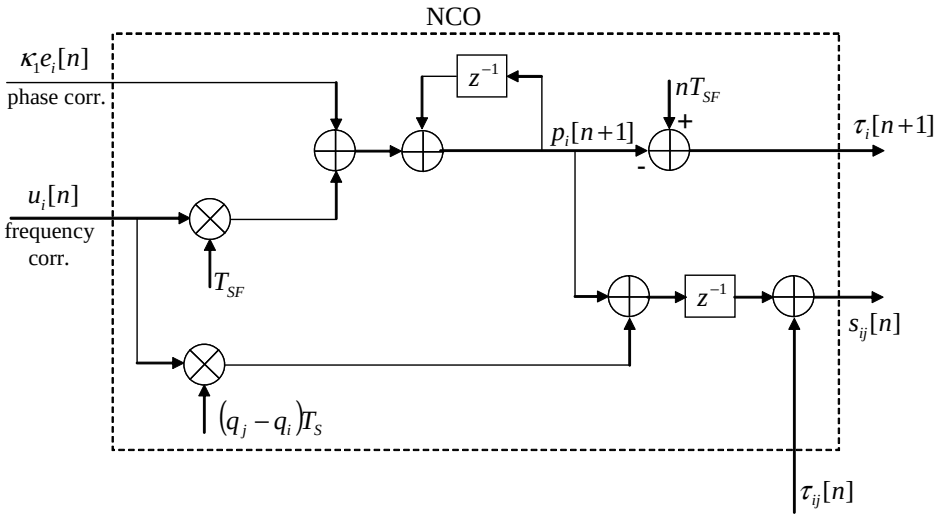


Figure 5.4 NCO structure for the type 2 PLL. For simplicity, only one branch of the circuit for reception timestamp correction is depicted.

never runs backwards). From (5.9), it is possible to express (5.8) in terms of phase and frequency corrections as

$$\tau_{ii}[n] - \tau_{ii}[n-1] = T_{SF} - \kappa_1 e_i[n-1] - T_{SF} u_i[n-1]. \tag{5.12}$$

The monotonicity condition requires that $|\kappa_1 e_i[n-1] + T_{SF} u_i[n-1]| < T_{SF}$. If loop parameters are chosen wisely, frequency correction is typically small, $|u_i[n-1]| \ll 1$ and most of the overall correction is provided by the phase term. Phase correction is a weighted average of corrected time offsets, which can be expressed as

$$\begin{aligned} o_{ij}[n] &= nT_{SF} + q_j T_S - (\tau_{ij}[n] + p_{ij}[n]) \\ &= nT_{SF} + q_j T_S - \Big[nT_{SF} + q_i T_S + (\tau_{ij}[n] - \tau_{ii}[n]) + p_{ij}[n] - p_i[n] \Big] \\ &= (\tau_{ii}[n] - \tau_{ij}[n]) - (1 - u_i[n-1])\,(q_i - q_j)\,T_S. \end{aligned} \tag{5.13}$$

Under normal operating conditions, $|\tau_{ii}[n] - \tau_{ij}[n]| < T_{SF}$, and it holds $|o_{ij}[n]| < T_{SF}$, therefore allowing to meet the monotonicity constraint. Notice that the second term in (5.13) takes into account the fact that beacons from j and i are transmitted in different time-slots.

In the next section, an analytical model for the clock control algorithm in Figure 5.3 is derived in order to derive stability conditions and synchronization accuracy for a network of coupled PLL's.

5.2.1 PLL Stability Analysis

This section investigates conditions necessary to ensure the stability of a network of coupled PLL's. Stability is in fact a required preliminary assumption in order to study the accuracy of synchronization at steady-state. From (5.12) and (5.2), the *absolute* time interval between two successive beacon transmissions from node i is

$$t_i[n] - t_i[n-1] = \frac{1}{1+\alpha_i}\left(T_{SF} - \kappa_1 e_i[n-1] - T_{SF} u_i[n-1]\right). \tag{5.14}$$

Recall that, since master nodes have access to an accurate time reference, it is $\tau_i[n] = t_i[n]$ and $t_i[n] - t_i[n-1] = T_{SF}$ if $i \in \mathcal{M}$. Since additive noise does not affect stability, observations are here considered noiseless, i.e., with reference to Figure 5.2, it is assumed $t'_j[n] = t_j[n]$ (or equivalently $w_{ij}[n] = 0$ in (5.6)). Measured time offsets can therefore be expressed as a function of absolute time by observing that $\tau_{ij}[n] - \tau_{ii}[n] = (1+\alpha_i)\left(t_j[n] - t_i[n]\right)$ in (5.13), and therefore

$$\begin{aligned} o_{ij}[n] &= (1+\alpha_i)\left[(t_i[n] - q_i T_S) - (t_j[n] - q_j T_S)\right] - (\alpha_i + u_i[n-1])(q_j - q_i) T_S. \\ &\simeq (1+\alpha_i)\left[(t_i[n] - q_i T_S) - (t_j[n] - q_j T_S)\right] \end{aligned} \tag{5.15}$$

where the last approximation holds assuming that, in steady-state, frequency correction is sufficiently accurate with respect to noise and clock granularity so that $(\alpha_i + u_i[n-1])(q_j - q_i) T_S \simeq 0$ for every q_j, q_i. The phase and frequency estimation errors of node i with respect to absolute time are defined as

$$x_i[n] = t_i[n] - q_i T_S - n T_{SF} \tag{5.16}$$

$$y_i[n] = -(u_i[n] + \alpha_i)\frac{T_{SF}}{1+\alpha_i}. \tag{5.17}$$

From (5.10)-(5.15) the dynamic equation for phase and frequency errors can be written as

$$x_i[n] = x_i[n-1] + \frac{\kappa_1}{d_i} e_i[n-1] + y_i[n-1], \tag{5.18}$$

$$y_i[n] = y_i[n-1] + \kappa_2 \frac{\kappa_1}{d_i} e_i[n-1], \tag{5.19}$$

where $\kappa_2 = T_{SF}\kappa_2'$ is now a dimension-less constant and the initial condition for the frequency error is $y_i[0] = -\frac{\alpha_i}{1+\alpha_i} T_{SF}$. The error $e_i[n]$ in (5.18)-(5.19) reads

$$e_i[n] = -\sum_{j=K_u+1}^{K} m_{ij} x_i[n] + \sum_{j=1}^{K_u} a_{ij} \left(x_j[n] - x_i[n] \right). \tag{5.20}$$

Recall the definition of the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}$ is the diagonal matrix of node degrees, $\mathbf{D} = \mathrm{diag}\,(d_1, d_2, \ldots, d_{K_u})$. By introducing the vectors $\mathbf{x} = [x_1, x_2, \ldots, x_{K_u}]^T$, $\mathbf{y} = [y_1, y_2, \ldots, y_{K_u}]^T$, the equations (5.18)-(5.20) may be cast in vector form as

$$\begin{bmatrix} \mathbf{x}\,[n] \\ \mathbf{y}\,[n] \end{bmatrix} = \begin{bmatrix} \mathbf{I} - \kappa_1 \mathbf{D}^{-1}\mathbf{L} & \mathbf{I} \\ -\kappa_1\kappa_2 \mathbf{D}^{-1}\mathbf{L} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}\,[n-1] \\ \mathbf{y}\,[n-1] \end{bmatrix}. \tag{5.21}$$

A dynamic system equivalent to (5.21) was studied in [44] for MC networks. Here a general treatment is pursued, which is valid for any topology, by following on the lines of stability analysis for formation control [84]. In the case of MS and hybrid networks, the Laplacian $\mathbf{L}$ is nonsingular, and the equilibrium point of (5.21) is $\mathbf{x}^* = \mathbf{y}^* = \mathbf{0}$. In the case of MC networks, instead, $\mathbf{L}$ is singular as $\mathbf{L1} = \mathbf{0}$. This property reflects the fact that the universal time t is unobservable in a MC network, and phase and frequencies are estimated with respect to a virtual reference $\tau_0(t) = \alpha_0 t + \beta_0$, where the constants α_0, β_0 depend on network topology and synchronization algorithm adopted. In fact, it can be shown (see also [44]) that $\lim_{n\to\infty} \mathbf{x}[n] = \mathbf{1}\mathbf{v}^T\mathbf{x}\,[0] - n\mathbf{y}^*$, where $\mathbf{y}^* = \mathbf{1}\mathbf{v}^T\mathbf{y}\,[0]$ and $[\mathbf{v}]_i = d_i / \left(\sum_i d_i\right)$.

The Schur decomposition of the *normalized* Laplacian matrix can be expressed as $\mathbf{D}^{-1}\mathbf{L} = \mathbf{Q}\mathbf{T}\mathbf{Q}^H$, where $\mathbf{Q}$ is a unitary matrix and $\mathbf{T}$ is an upper triangular matrix with diagonal entries $[\mathbf{T}]_{ii} = \lambda_i(\mathbf{D}^{-1}\mathbf{L})$ [85]. By letting $\tilde{\mathbf{x}}[n] = \mathbf{Q}^H\mathbf{x}[n]$, $\tilde{\mathbf{y}}[n] = \mathbf{Q}^H\mathbf{y}[n]$,

(5.21) can be rewritten as

$$\begin{bmatrix} \tilde{\mathbf{x}}[n] \\ \tilde{\mathbf{y}}[n] \end{bmatrix} = \begin{bmatrix} \mathbf{I} - \kappa_1 \mathbf{T} & \mathbf{I} \\ -\kappa_1\kappa_2 \mathbf{T} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}[n-1] \\ \tilde{\mathbf{y}}[n-1] \end{bmatrix}. \tag{5.22}$$

The blocks comprised in the equivalent system matrix in (5.22) are now either diagonal or triangular, and therefore the system (5.22) is stable depending on the stability of the K_u 2×2 subsystems defined over the diagonal elements of the blocks,

$$\begin{bmatrix} \tilde{x}_i[n] \\ \tilde{y}_i[n] \end{bmatrix} = \begin{bmatrix} 1 - \kappa_1\mu_i & 1 \\ -\kappa_1\kappa_2\mu_i & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_i[n-1] \\ \tilde{y}_i[n-1] \end{bmatrix}, \tag{5.23}$$

where $\mu_i = \lambda_i(\mathbf{D}^{-1}\mathbf{L})$ for notational convenience. The system in (5.23) is the same as a type 2 discrete-time phase-locked loop (DPLL) with loop gain $\kappa_1\mu_i$. If the eigenvalues μ_i are all real, (5.23) is known to be stable if $0 < \kappa_2 < 1$ and $0 < \kappa_1\mu_i < 4/(2-\kappa_2)$ [6]. The normalized Laplacian $\mathbf{D}^{-1}\mathbf{L}$ of a MS network may be triangularized by simple permutation, and its eigenvalues are all equal to 1, $\mu_i = 1$ for $i = 1, \ldots K$. Therefore, the stability criteria valid for a single MS link are sufficient to guarantee the stability of a whole MS network[6]. In the case of MC networks, $\mathbf{L}$ is symmetric, and therefore its eigenvalues are all real and nonnegative. The eigenvalues of $\mathbf{D}^{-1}\mathbf{L}$ correspond to the *generalized* eigenvalues of $\mathbf{L}$ and $\mathbf{D}$ [52]; since $\mathbf{D}$ is positive-definite, they are also real. Finally, by Gershgorin's theorem [85], μ_i is comprised within $\mu_i \in [0, 2]$, and therefore a sufficient condition for stability of (5.23) is

$$\begin{cases} 0 < \kappa_2 < 1 \\ 0 < \kappa_1 < \frac{2}{2-\kappa_2} \\ \mu_i > 0 \qquad i = 2, \ldots, K \end{cases}. \tag{5.24}$$

where the last condition takes into account that $\mu_1 = 0$ from the unobservability of absolute time. The fact that all the other eigenvalues need to be non-zero implies that the network

[6]A necessary requirement for network synchronization with a MS architecture is that the topology graph contains a forest rooted at the master nodes, which implies $\mu_i > 0$.

is connected . If the graph is regular (e.g., a ring network), the normalized Laplacian is symmetric and the matrix $\mathbf{T}$ is diagonal. In this way, the system (5.21) is effectively decoupled into K parallel 2×2 systems. Of note, MC networks with full connectivity have $\mu_{i>1} = 1 + \frac{1}{K-1}$, i.e., $\mu_{i>1} \simeq 1$ for large K. For a hybrid network, $\mathbf{L}$ is still symmetric, but $\mu_1 > 0$ needs to be added to the sufficient conditions (5.24) in order to achieve network synchronization.

The loop gain κ_1 and the integrator gain κ_2 govern the loop dynamic response. A common practice in discrete-time PLL design is to employ a time-continuous approximation of (5.18)-(5.19) in order to choose their values [6]. The approximation is accurate for small loop gain κ_1 ($\kappa_1 < 0.2$ according to [6]), which is a typical choice to achieve accurate tracking. In particular, a type 2 PLL may be approximated by a continuous-time (analogue) PLL with natural frequency ω_n and damping factor ζ

$$\omega_n = \frac{1}{T_{SF}} \sqrt{\kappa_1 \kappa_2} \tag{5.25}$$

$$\zeta = \frac{1}{2} \sqrt{\frac{\kappa_1}{\kappa_2}}. \tag{5.26}$$

In general, it is recommended to choose $\kappa_2 < \kappa_1$ in order to have $\zeta > 0.5$ and avoid large oscillations in the dynamic response. Typical choices are $\kappa_2 = \kappa_1/4$ or $\kappa_2 = \kappa_1/2$, which imply $\zeta = 1$ or $\zeta = 0.707$, respectively. When PLL's are coupled in a synchronization network, their dynamic response depends on the specific architecture adopted. Wide oscillations are observed with both MS and MC topologies even with $\zeta > 1$, but large damping factors entail longer convergence times. Of note, it is recommended in the literature to employ $\zeta \geq 4.4$ with MS topologies (see, e.g., [6]). From simulation results here omitted, it has been determined that $\zeta = 5$ is a reasonable choice for all the architectures considered in this work.

5.2.2 PLL Accuracy Analysis

Consider now the case where the measured offsets $o_{ij}[n]$ in (5.15) are corrupted by i.i.d. noise with variance $E\left[(w_{ij}[n])^2\right] = \sigma_w^2$. After the selection and combining block, the noise term added to the error $e_i[n]$ in (5.20) is

$$w_i[n] = \frac{1}{d_i}\left[\sum_{j=K_u+1}^{K} m_{ij}w_{ij}[n] + \sum_{j=1}^{K_u} a_{ij}w_{ij}[n]\right]. \tag{5.27}$$

Let the noise vector $\mathbf{w}[n]$ be defined as $\mathbf{w}[n] = [w_1[n], w_2[n], \ldots, w_{K_u}[n]]$. From (5.27) the noise covariance is $E\left[\mathbf{w}\mathbf{w}^T\right] = \sigma_w^2\mathbf{D}^{-1}$. The update (5.21) may be rewritten as

$$\mathbf{z}\left[n\right] = \begin{bmatrix} \mathbf{I} - \kappa_1\mathbf{D}^{-1}\mathbf{L} & \mathbf{I} \\ -\kappa_1\kappa_2\mathbf{D}^{-1}\mathbf{L} & \mathbf{I} \end{bmatrix} \mathbf{z}\left[n-1\right] + \kappa_1 \begin{bmatrix} \mathbf{w}[n-1] \\ \kappa_2\mathbf{w}[n-1] \end{bmatrix}. \tag{5.28}$$

where $\mathbf{z}\left[n\right] = \left[\mathbf{x}\left[n\right], \mathbf{y}\left[n\right]\right]^T$. Let $\mathbf{P}$ be the iteration matrix in (5.28). From (5.28), the covariance $\mathbf{C}\left[n\right] = E\left[\mathbf{z}\left[n\right]\mathbf{z}^T\left[n\right]\right]$ satisfies the recursion

$$\mathbf{C}\left[n\right] = \mathbf{P}\mathbf{C}\left[n-1\right]\mathbf{P}^T + \mathbf{C}_w, \tag{5.29}$$

where the covariance matrix of the random additive term in (5.28) is

$$\mathbf{C}_w = \kappa_1^2\sigma_w^2 \begin{bmatrix} 1 & \kappa_2 \\ \kappa_2 & \kappa_2^2 \end{bmatrix} \otimes \mathbf{D}^{-1}. \tag{5.30}$$

If the iteration $\mathbf{P}$ is stable, the steady-state covariance $\mathbf{C}_{ss} = \lim_{n\to\infty}\mathbf{C}[n]$ satisfies the discrete-time Lyapunov equation

$$\mathbf{C}_{ss} = \mathbf{P}\mathbf{C}_{ss}\mathbf{P}^T + \mathbf{C}_w, \tag{5.31}$$

which has a unique solution that may be expressed as

$$\mathbf{C}_{ss} = \sum_{n=0}^{\infty}\left(\mathbf{P}^T\right)^n\mathbf{C}_w\mathbf{P}^n. \tag{5.32}$$

In the case of MS and hybrid networks, the steady-state average MSE for phase and frequency is defined as $\xi_p^2 = \lim_{n\to\infty} \frac{1}{K} \sum_{i=1}^{K_u} E\left[x_i^2[n]\right]$, $\xi_f^2 = \lim_{n\to\infty} \frac{1}{K} \sum_{i=1}^{K_u} E\left[y_i^2[n]\right]$, respectively. These quantities are computed from the covariance matrix in (5.32) as

$$\xi_p^2 = \frac{1}{K} \sum_{i=1}^{K_u} \left[\mathbf{C}_{ss}\right]_{ii} \tag{5.33}$$

$$\xi_f^2 = \frac{1}{K} \sum_{i=K_u+1}^{2K_u} \left[\mathbf{C}_{ss}\right]_{ii}. \tag{5.34}$$

In the case of MC networks, absolute time is unobservable due to the lack of a master node. Phase and frequency errors are therefore defined with respect to a *virtual* reference, $x_0[n]$, $y_0[n]$, given by the weighted average of absolute phases and frequencies, e.g., $x_0[n] = \sum_{j=1}^{K} v_i x_i[n]$ for phase. The weight for the contribution of node i, v_i, can be conveniently chosen according to the specific case under analysis. In particular, in order to perform noise analysis for a MC networks of PLL's with dynamics (5.28), the following change of variables is introduced

$$\mathbf{s}[n] = \begin{bmatrix} \left(\mathbf{I}_K - \mathbf{1}\mathbf{v}^T\right)\mathbf{x}[n] \\ \left(\mathbf{I}_K - \mathbf{1}\mathbf{v}^T\right)\mathbf{y}[n] \end{bmatrix}, \tag{5.35}$$

where the node weights are $\left[\mathbf{v}\right]_i = v_i = d_i / \left(\sum_i d_i\right)$. It can be shown that the Lyapunov equation (5.31) is rewritten for the transformed state (5.35) as

$$\mathbf{C}_{ss} = \mathbf{P}'\mathbf{C}_{ss}\mathbf{P}'^T + \mathbf{C}'_w, \tag{5.36}$$

where $\mathbf{P}' = \mathbf{G}\mathbf{P}\mathbf{G}$, $\mathbf{C}'_w = \mathbf{G}\mathbf{C}_w\mathbf{G}$, and $\mathbf{G} = \mathbf{I}_2 \otimes \left(\mathbf{I}_K - \mathbf{1}\mathbf{v}^T\right)$. The steady-state average MSE for phase and frequency, ξ_p^2, ξ_f^2, is then computed from (5.36) analogously to (5.33)-(5.34).

In the special case of *regular* MC networks, the steady-state accuracy can be computed in closed form as detailed in the following section.

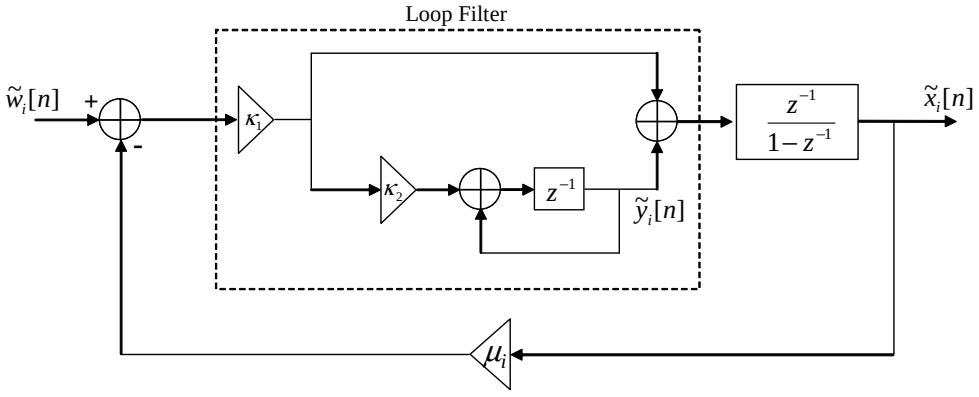


Figure 5.5 Decomposition of a regular MC network of PLL's.

5.2.3 A Special Case: Regular MC Networks

In the case of regular MC networks, the node degree is the same for all nodes, $d_i = d$, and the steady-state phase and frequency MSE may be computed analytically. In fact, the normalized Laplacian $\mathbf{D}^{-1}\mathbf{L} = \frac{1}{d}\mathbf{L}$ is symmetric and (5.28) may be diagonalized into K parallel dynamic systems as the one in Figure 5.5. The input noise $\tilde{w}_i[n]$ is i.i.d. along the index i, $E\left[\left(\tilde{w}_i[n]\right)^2\right] = \frac{\sigma_w^2}{d}$. The transfer function from $\tilde{w}_i[n]$ to $\tilde{x}_i[n]$ is

$$H_p\left(z;\mu_i\right) = \kappa_1 z^{-1}\frac{1-\left(1-\kappa_2\right)z^{-1}}{\left(1-z^{-1}\right)^2+\kappa_1\mu_i z^{-1}\left[1-\left(1-\kappa_2\right)z^{-1}\right]}. \tag{5.37}$$

The output phase noise power for the i-th spectral component is computed by integrating the power spectral density of (5.37), $S_p\left(z;\mu_i\right) = H_p\left(z;\mu_i\right)H_p\left(z^{-1};\mu_i\right)$, as in

$$2b_L^{(p)}\left(\mu_i\right) = \frac{1}{2\pi j}\oint_{|z|=1} S_p\left(z;\mu_i\right)z^{-1}dz. \tag{5.38}$$

The term $b_L^{(p)}\left(\mu_i\right)$ is defined as the one-sided *noise equivalent bandwidth* of (5.37) [6]. Notice that (5.38) is a normalized bandwidth, and it is therefore dimension-less. By employing the residue theorem [86], (5.38) yields

$$2b_L^{(p)}\left(\mu_i\right) = \frac{\kappa_1}{2\mu_i}\frac{1+\frac{\kappa_2}{\kappa_1\mu_i}-\frac{\kappa_2}{2}\left(3-\kappa_2\right)}{1-\kappa_2-\frac{\kappa_1\mu_i}{4}\left(2-\kappa_2+\kappa_2^2\right)}. \tag{5.39}$$

Therefore, the average steady-state phase error is

$$\xi_p^2 = \frac{1}{K}\sum_{i=2}^{K}\frac{\sigma_w^2}{d}2b_L^{(p)}\left(\mu_i\right) \tag{5.40}$$

$$\simeq \frac{\kappa_1}{2}\frac{\sigma_w^2}{K}\sum_{i=2}^{K}\left[\frac{1}{d\mu_i}+\frac{1}{4\zeta^2}\frac{1}{d\mu_i^2}\right], \tag{5.41}$$

where the last approximation follows from $2b_L^{(p)}\left(\mu_i\right) \simeq \frac{\kappa_1}{2\mu_i}\left(1+\frac{1}{4\zeta^2\mu_i}\right)$ for small loop gain κ_1, $\kappa_2 < \kappa_1 \ll 1$, and $\kappa_2 = \frac{\kappa_1}{4\zeta^2}$ as in (5.26).

The steady-state frequency MSE may be computed by following a similar procedure. The transfer function from from $\tilde{w}_i[n]$ to $\tilde{y}_i[n]$ is

$$H_f\left(z;\lambda_i\right) = \kappa_1\kappa_2 z^{-1}\frac{1-z^{-1}}{\left(1-z^{-1}\right)^2+\kappa_1\mu_i z^{-1}\left[1-\left(1-\kappa_2\right)z^{-1}\right]}. \tag{5.42}$$

Again by applying the residue theorem, the noise equivalent bandwidth of (5.42) reads

$$2b_L^{(f)}\left(\mu_i\right) = \frac{1}{\mu_i^2}\frac{2\kappa_2}{\left(1-\kappa_2\right)\left[1+\frac{2(2-\kappa_1\mu_i)}{\kappa_1\kappa_2\mu_i}\right]}. \tag{5.43}$$

Therefore, the steady-state average MSE is

$$\xi_f^2 = \frac{1}{K}\sum_{i=2}^{K}\frac{\sigma_w^2}{d}2b_L^{(f)}\left(\mu_i\right) \tag{5.44}$$

$$\simeq \frac{\kappa_1^3}{32\zeta^4}\frac{\sigma_w^2}{K}\sum_{i=2}^{K}\frac{1}{d\mu_i^2}. \tag{5.45}$$

where the last approximation holds for small loop gain κ_1, $\kappa_2 < \kappa_1 \ll 1$. Notice that both phase and frequency accuracy in (5.41)-(5.45) show the same dependence from network topology through the normalized Laplacian spectrum $\mu_i = \lambda_i\left(\mathbf{L}\right)/d$.

5.3 Distributed Estimation of Clock Parameters: Open-loop Synchronization

From the simple model (5.2), each clock is defined by two parameters: the frequency offset α_i and the phase offset β_i. If node i has available an estimate of these two, $\hat{\mathbf{x}}_i^T = \left[\hat{\alpha}_i, \hat{\beta}_i\right]$,

it can correct the local clock $\tau_i(t)$ as in (*time translation* operation[7])

$$s_i(\tau_i) = \frac{1}{1+\hat{\alpha}_i}\left(\tau_i - \hat{\beta}_i\right). \tag{5.46}$$

If the estimates coincide with the actual values, $\hat{\alpha}_i = \alpha_i$, $\hat{\beta}_i = \beta_i$, then time displayed by the corrected clock coincides with absolute time, $s_i(\tau_i) = t_i(\tau_i)$. As for the PLL, beacon transmissions can be triggered without adjusting the local clock directly, but just waiting for the local clock to strike

$$\tau_{ii}[n] = (1+\hat{\alpha}_i)\, nT_{SF} + \hat{\beta}_i. \tag{5.47}$$

Also, as in (5.12), subsequent beacon transmissions can be triggered via the recursion

$$\tau_{ii}[n+1] - \tau_{ii}[n] = (1+\hat{\alpha}_i)\, T_{SF}. \tag{5.48}$$

In this section an algorithm is devised in order to estimate clock parameters in a completely distributed fashion. Notice how, from the observation model (5.6), the problem can be posed as a *distributed linear regression*. In general, distributed estimation problems are solved in two steps, namely a training phase and a fusion phase. During the training phase, nodes acquire the *local* observations necessary for estimation, while during the fusion phase each node cooperates with its neighbors in order to compute the *global* estimate of the parameters of interest. If clocks are indefinitely stable, their frequency never changes, and clock parameters can be estimated once and for all during the initial network calibration by a single training and fusion cycle. In practice, training and fusion phases would repeat periodically throughout normal network operation (see Figure 5.6) in order to update current estimates and track slow clock frequency variations due to environmental factors. The assumption of clock stability formulated in Section 5.1.2 implies that frequency changes are so slow that clock parameters can be regarded as constant within a single

[7]By employing the same useful approximations adopted in Section 5.1.2, the corrected clock can be alternatively computed with $s_i(\tau_i) \simeq \tau_i - \hat{\alpha}_i\tau_i - (1-\hat{\alpha}_i)\,\hat{\beta}_i$.

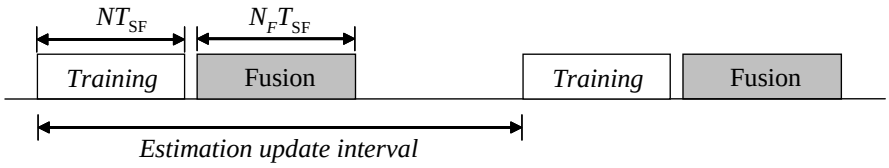


Figure 5.6 Alternating training and fusion phases for the distributed estimation of clock parameters.

update interval. If estimation is performed independently in subsequent training/fusion cycles (*open-loop* synchronization), synchronization accuracy depends only on the length of the training phase. For simplicity of exposition, in the following the focus is on the initial network calibration, whereby nodes transmit only beacon frames leaving their clocks free-running, and each super-frame comprises suitable inter-slot guard times to account for phase and frequency offsets.

During the training phase, each node i collects a set of N time offset observations $\{o_{ij}[n]\}_{n=0}^{N-1}$ for each incoming link $(i,j) \in \mathcal{E}$. Notice that the set of available measures depends on network connectivity through the edge set $\mathcal{E}$. Since each super-frame allows to collect only a single time offset for each link, the training phase comprises N subsequent super-frames. During the fusion phase, nodes employ a distributed procedure to estimate local clock parameters. This phase comprises several super-frames, say N_F, depending on the fusion algorithm adopted and network architecture. The exact value of N_F is immaterial for the present discussion since it does not affect synchronization accuracy. In the following, a fusion algorithm will be devised in order to allow a given node i to estimate its own clock parameters without exchanging measured offsets with its neighbors, but only by broadcasting its current estimate $\hat{\mathbf{x}}_i^T = \left[\hat{\alpha}_i, \hat{\beta}_i\right]$. The value of the current estimate can be inserted in the payload of the beacon frame, thereby complying with the communication model of Section 5.1.1.

Suitable distributed algorithms can be devised by considering *synthetic* observations, computed as the weighted sum of the time offsets observed by node i in the n-th super-

frame of the training phase, namely

$$o_i[n] = \sum_{j \in \mathcal{N}_i \cap \mathcal{M}} m_{ij} o_{ij}[n] + \sum_{j \in \mathcal{N}_i \cap \mathcal{S}} a_{ij} o_{ij}[n], \tag{5.49}$$

where the weights a_{ij}, m_{ij} are subject to design choice. It can be easily proved that (5.49) is a sufficient statistics for the problem at hand. Still, processing the original observations typically implies a degradation of the estimation efficiency. As it will be shown in Section 5.6, this is also the case here. Given the model (5.6), (5.49) may be expressed in terms of the parameters of interest as

$$o_i[n] = -\sum_{j=K_u+1}^{K} m_{ij} \left[\alpha_i n T_{SF} + \beta_i\right] + \sum_{j=1}^{K_u} a_{ij} \left[\left(\alpha_j - \alpha_i\right) n T_{SF} + \left(\beta_j - \beta_i\right)\right] + w_i[n], \tag{5.50}$$

for $n = 0, \ldots, (N-1)\, T_{SF}$, where the noise $w_i[n] = \sum_{j=K_u+1}^{K} m_{ij} w_{ij}[n] + \sum_{j=1}^{K_u} a_{ij} w_{ij}[n]$. Recall the definition of the in-degree of node k, $d_i = \sum_{j=K_u+1}^{K} m_{ij} + \sum_{j=1}^{K_u} a_{ij}$. The noise variance is proportional to the node degree, $E\left[\left(w_i[n]\right)^2\right] = d_i \sigma_w^2$. As a preliminary step towards the development of a distributed fusion scheme, the next section introduces a centralized procedure that estimates phase and frequency offsets from the synthetic observations (5.50).

5.3.1 Centralized Estimation

In this section it is assumed to employ a centralized processor which has somehow obtained timing offsets observations (5.50) from all nodes. The analysis of the centralized algorithm is instrumental in order to derive the *distributed* algorithm of Section 5.3.2.

The $N \times 1$ vector comprising the observations of node i is $\mathbf{o}_i = [o_i[0], o_i[1], \ldots, o_i[N-1]]^T$, the $N \times 1$ observation noise vector is $\mathbf{w}_i = [w_i[0], w_i[1], \ldots, w_i[N-1]]^T$, and the $2K_u \times 1$ parameter vector is $\mathbf{x} = \left[\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_{K_u}^T\right]^T$, $\mathbf{x}_i^T = [\alpha_i, \beta_i]$. From (5.50), it can be seen that

$$\mathbf{o}_i = \left[\left(-d_i \mathbf{e}_i^T + \mathbf{a}_i^T\right) \otimes \mathbf{R}\right] \mathbf{x} + \mathbf{w}_i, \tag{5.51}$$

where $\mathbf{e}_i$ is the i-th column of the $K_u \times K_u$ identity matrix, $\mathbf{a}_i^T$ is the i-th row of the $K_u \times K_u$ adjacency matrix $\mathbf{A}$, and the $N \times 2$ matrix $\mathbf{R}$ is

$$\mathbf{R} = \begin{bmatrix} 0 & 1 \\ T_{SF} & 1 \\ \vdots & \vdots \\ (N-1)\,T_{SF} & 1 \end{bmatrix}. \tag{5.52}$$

By defining the $NK_u \times 1$ network observation vector $\mathbf{o} = \left[\mathbf{o}_1^T, \mathbf{o}_2^T, \ldots, \mathbf{o}_{K_u}^T\right]^T$, the observations can be written in compact form as the following linear system

$$\mathbf{o} = \mathbf{H}\mathbf{x} + \mathbf{w}, \tag{5.53}$$

where the $NK_u \times 2K_u$ system matrix is $\mathbf{H} = -\mathbf{L} \otimes \mathbf{R}$ and the covariance matrix of the random noise $\mathbf{w}$ is $\mathbf{C}_w = \mathbf{D} \otimes \mathbf{I}_N \sigma_w^2$.

In the case of MS and hybrid architectures introduced in Section 2.3, if the network graph comprises a forest rooted at the master nodes, the Laplacian $\mathbf{L}$ is nonsingular, and therefore $\mathbf{H}$ has full column-rank. From the Gauss-Markov theorem, the best linear unbiased estimator (BLUE) for the vector parameter $\mathbf{x}$ given the linear model (5.53) is $\hat{\mathbf{x}} = \mathbf{W}\mathbf{o}$, where [87]

$$\mathbf{W} = \left(\mathbf{H}^T \mathbf{C}_w^{-1} \mathbf{H}\right)^{-1} \mathbf{H}^T \mathbf{C}_w^{-1}. \tag{5.54}$$

By exploiting the mixed-product property of the Kronecker product [85], $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$, and the $2K_u \times NK_u$ matrix $\mathbf{W}$ is written as

$$\mathbf{W} = -\mathbf{L}^{-1} \otimes \mathbf{R}^{\dagger}, \tag{5.55}$$

where $\mathbf{R}^{\dagger}$ is the $2 \times N$ pseudoinverse (Moore-Penrose inverse) of $\mathbf{R}$, $\mathbf{R}^{\dagger} = \left(\mathbf{R}^T\mathbf{R}\right)^{-1}\mathbf{R}^T$ [85]. The $2K_u \times 2K_u$ covariance matrix of the estimate is

$$\mathrm{Cov}\left(\hat{\mathbf{x}}\right) = \sigma_w^2 \left(\mathbf{L}^T \mathbf{D}^{-1} \mathbf{L}\right)^{-1} \otimes \left(\mathbf{R}^T\mathbf{R}\right)^{-1}. \tag{5.56}$$

Let the network phase estimation error ξ_p^2 be defined as the average MSE, $\xi_p^2 = \frac{1}{K}\sum_{i=1}^{K_u} E\left[\left(\hat{\beta}_i - \beta_i\right)^2\right]$; a similar definition holds for the network frequency estimation error ξ_f^2. From (5.56), it holds that

$$\xi_p^2 = \frac{2\left(2N-1\right)}{N\left(N+1\right)}\frac{\sigma_w^2}{K}\sum_{i=1}^{K_u}\frac{1}{\lambda_i\left(\mathbf{L}^T\mathbf{D}^{-1}\mathbf{L}\right)} \tag{5.57}$$

$$\xi_f^2 = \frac{12}{T_{SF}^2 N\left(N^2-1\right)}\frac{\sigma_w^2}{K}\sum_{i=1}^{K_u}\frac{1}{\lambda_i\left(\mathbf{L}^T\mathbf{D}^{-1}\mathbf{L}\right)}, \tag{5.58}$$

where it has been employed the property of the trace of an invertible matrix $\mathbf{E}$, $\mathrm{tr}\left(\mathbf{E}^{-1}\right) = \sum_{i=1}^{K_u}\lambda_i^{-1}\left(\mathbf{E}\right)$, and $\lambda_i\left(\mathbf{E}\right)$ are the ordered eigenvalues of $\mathbf{E}$, $\lambda_1\left(\mathbf{E}\right) \leq \lambda_2\left(\mathbf{E}\right) \leq \ldots \leq \lambda_{K_u}\left(\mathbf{E}\right)$.

In the case of MC architectures, there are no master nodes and the Laplacian $\mathbf{L}$ is a symmetric singular matrix, $\mathbf{L1} = \mathbf{0}$. Since universal time can not be observed in a MC network, phase and frequencies are estimated with respect to a virtual reference $\tau_0(t) = \alpha_0 t + \beta_0$, where the reference phase and frequency are $\alpha_0 = \frac{1}{K}\sum_{j=1}^{K}\alpha_i$, $\beta_0 = \frac{1}{K}\sum_{j=1}^{K}\beta_i$. The new parameter vector $\mathbf{y} = \left[\mathbf{y}_1^T, \mathbf{y}_2^T, \ldots, \mathbf{y}_K^T\right]^T$, $\mathbf{y}_i^T = [\alpha_i - \alpha_0, \beta_i - \beta_0]$, can be written as a linear combination of absolute phases and frequencies, $\mathbf{y} = \left[\left(\mathbf{I}_K - \mathbf{J}\right)\otimes\mathbf{I}_2\right]\mathbf{x}$, where $\mathbf{J} = \frac{1}{K}\mathbf{1}\mathbf{1}^T$. It is shown in Appendix 5.A that the BLUE estimator for the transformed parameters is $\hat{\mathbf{y}} = \mathbf{W}\mathbf{o}$, where

$$\mathbf{W} = -\left(\mathbf{L}^T\mathbf{D}^{-1}\mathbf{L}\right)^{\dagger}\mathbf{L}^T\mathbf{D}^{-1}\otimes\mathbf{R}^{\dagger}. \tag{5.59}$$

The covariance matrix of the estimate is

$$\mathrm{Cov}\left(\hat{\mathbf{y}}\right) = \sigma_w^2\left(\mathbf{L}^T\mathbf{D}^{-1}\mathbf{L}\right)^{\dagger}\otimes\left(\mathbf{R}^T\mathbf{R}\right)^{-1}. \tag{5.60}$$

From (5.60) and the definition of Moore-Penrose inverse [85], the average MSE reads

$$\xi_p^2 = \frac{2\,(2N-1)}{N\,(N+1)}\frac{\sigma_w^2}{K}\sum_{i=2}^{K}\frac{1}{\lambda_i\left(\mathbf{L}^T\mathbf{D}^{-1}\mathbf{L}\right)} \tag{5.61}$$

$$\xi_f^2 = \frac{12}{T_{SF}^2 N\,(N^2-1)}\frac{\sigma_w^2}{K}\sum_{i=2}^{K}\frac{1}{\lambda_i\left(\mathbf{L}^T\mathbf{D}^{-1}\mathbf{L}\right)}. \tag{5.62}$$

Notice that sum starts from $i = 2$ as $\lambda_1\left(\mathbf{L}^T\mathbf{D}^{-1}\mathbf{L}\right) = 0$ from the singularity of the Laplacian $\mathbf{L}$.

The next section introduces a distributed algorithm that achieves the same accuracy as the centralized estimator on certain topologies.

5.3.2 Distributed Estimation

Clock parameter estimation may be performed in a *distributed* fashion by the use of iterative methods for the inversion of the linear system

$$-\left(\mathbf{L}\otimes\mathbf{R}\right)\hat{\mathbf{x}} = \mathbf{o}. \tag{5.63}$$

Notice that the system (5.63) is inconsistent as the observation $\mathbf{o}$ is corrupted by noise. In order to find a solution to (5.63), a distributed iterative procedure can be derived in the likes of the block Jacobi algorithm [88]. By defining $\tilde{\mathbf{A}} = \mathbf{A}\otimes\mathbf{R}$ and the block-diagonal matrix $\tilde{\mathbf{D}} = \mathbf{D}\otimes\mathbf{R}$, the system matrix in (5.63) can be split as $\left(\mathbf{L}\otimes\mathbf{R}\right) = \left(\tilde{\mathbf{D}} - \tilde{\mathbf{A}}\right)$. If $d_i \neq 0$ (i.e., if there are no isolated nodes), $\tilde{\mathbf{D}}$ has full column-rank, and the pseudoinverse of $\tilde{\mathbf{D}}$ reads

$$\begin{aligned}\tilde{\mathbf{D}}^{\dagger} &= \left(\mathbf{D}\otimes\mathbf{R}^T\cdot\mathbf{D}\otimes\mathbf{R}\right)^{-1}\mathbf{D}\otimes\mathbf{R}^T\\ &= \mathbf{D}^{-1}\otimes\mathbf{R}^{\dagger},\end{aligned} \tag{5.64}$$

which is again block-diagonal. As it holds that $\tilde{\mathbf{D}}^{\dagger}\tilde{\mathbf{D}} = \mathbf{I}$, (5.63) may be cast into the following fixed-point equation

$$\hat{\mathbf{x}} = \tilde{\mathbf{D}}^{\dagger}\tilde{\mathbf{A}}\hat{\mathbf{x}} - \mathbf{c}, \tag{5.65}$$

where $\mathbf{c} = \tilde{\mathbf{D}}^{\dagger}\mathbf{o}$. The fixed-point of (5.65) can be computed recursively by the following iterative procedure

$$\hat{\mathbf{x}}\left[p+1\right] = \left(1-\epsilon\right)\hat{\mathbf{x}}\left[p\right] + \epsilon\left[\tilde{\mathbf{D}}^{\dagger}\tilde{\mathbf{A}}\hat{\mathbf{x}}\left[p\right] - \mathbf{c}\right], \tag{5.66}$$

where $\hat{\mathbf{x}}\left[p\right]$ is the vector comprising phase and frequency estimates at step p, and ϵ is a *relaxation* parameter. By substituting (5.64) in (5.66), the estimate update is rewritten as

$$\hat{\mathbf{x}}\left[p+1\right] = \left[\left(\mathbf{I}_{K_u} - \epsilon\mathbf{D}^{-1}\mathbf{L}\right)\otimes\mathbf{I}_2\right]\hat{\mathbf{x}}\left[p\right] - \epsilon\mathbf{c}. \tag{5.67}$$

In the case of MS and hybrid architectures, the recursion (5.67) converges to the BLUE (5.56) if the spectral radius of the iteration matrix is strictly smaller than unity [50], $\rho\left(\mathbf{I} - \epsilon\mathbf{D}^{-1}\mathbf{L}\right) < 1$. As already observed in Section 5.2.1, Gershgorin's theorem assures that the eigenvalues of $\mathbf{D}^{-1}\mathbf{L}$ are comprised within the interval $(0, 2]$. Therefore, a sufficient condition to ensure the convergence of the algorithm is $0 < \epsilon < 1$. Notably, the algorithm is still convergent with $\epsilon = 1$, since the iteration matrix would be $\mathbf{D}^{-1}\mathbf{A}$ and $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is an M-matrix[8]. In the case of MC architectures, the recursion (5.67) converges to the BLUE (5.59) only on regular networks (e.g., ring networks). In general, the accuracy of the distributed algorithm is very close to the optimum (5.60) on graphs with almost regular degree distribution (e.g, the lattice and line graphs considered in Section 5.6). Appendix 5.A is devoted to the detailed convergence analysis of (5.67) in the case of MC architectures.

Direct inspection reveals that the recursion (5.67) allows for a distributed implementation. In particular, since $\tilde{\mathbf{D}}$ is block-diagonal, $\mathbf{c}$ is partitioned as $\mathbf{c} = \left[\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{K_u}\right]$, where each 2×1 component $\mathbf{c}_i = \left[c_{\alpha,i}, c_{\beta,i}\right]^T$ can be computed *locally* by node i by applying linear regression to local observations, $\left[c_{\alpha,i}, c_{\beta,i}\right]^T = \frac{1}{d_i}\mathbf{R}^{\dagger}\mathbf{o}_i$. Also, from (5.67), it is apparent that frequency and phase updates are independent. In conclusion, the phase

[8]Several conditions can ensure that a given matrix is an M-matrix. In this case, it can be exploited the fact that $\mathbf{L}$ is nonsingular with nonpositive off-diagonal entries and its eigenvalues are real and positive [50].

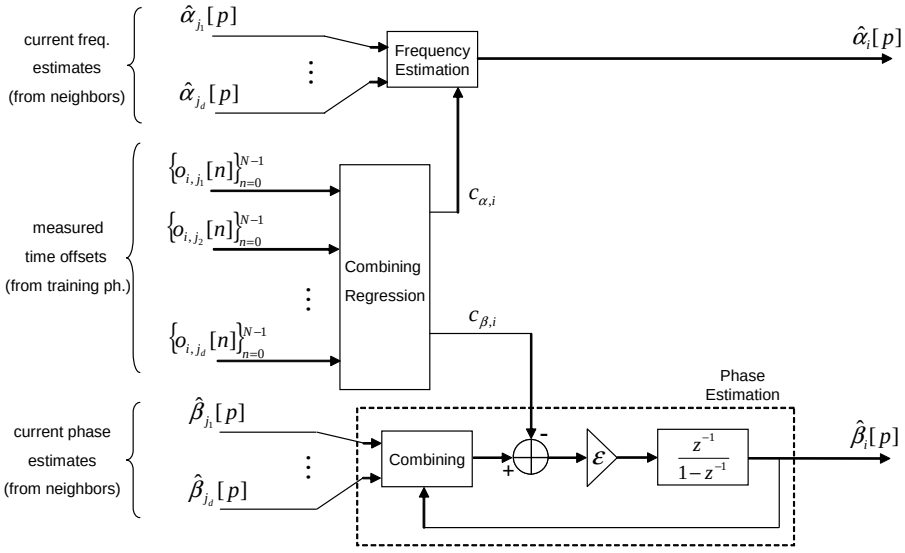


Figure 5.7 Block diagram of the implementation of the distributed open-loop algorithm at node i. Only the structure of the phase estimation block is detailed.

update at node i is (frequency update is analogous)

$$\hat{\beta}_i[p+1] = \hat{\beta}_i[p] + \frac{\epsilon}{d_i}\left[\sum_{j\in\mathcal{N}_i\cap\mathcal{S}} a_{ij}\left(\hat{\beta}_j[p] - \hat{\beta}_i[p]\right) - \sum_{j\in\mathcal{N}_i\cap\mathcal{M}} m_{ij}\hat{\beta}_i[p]\right] - \epsilon c_{\beta,i}. \quad (5.68)$$

The iteration (5.68) is implemented as a distributed *parallel* algorithm, whereby each node updates its local phase and frequency estimates given its current estimate and the current estimates of its neighbors. The block diagram of the algorithm is depicted in Figure 5.7. The combining function in Figure 5.7 refers to the fact that the link weights a_{ij}, m_{ij} in (5.49) are subject to design choice. In practice, after the training phase, each node i computes $\mathbf{c}_i$ from local observations, and during the subsequent fusion phase it runs the recursion (5.68) to compute the global estimate of its phase offset. Each super-frame within the fusion phase corresponds to one iteration of the recursion, and current estimates $\alpha_i[p]$, $\beta_i[p]$ are advertised within the beacon frames broadcast by node i.

The authors of [41] employ a local update which corresponds to (5.68) specialized with $\epsilon = 1$. Nevertheless, the "spatial smoothing" algorithm proposed in [41] assumed a different communication model, whereby neighbors exchange local observations $o_{ij}[n]$ by bi-directional communication. Furthermore, convergence of the algorithm is proved only

for hybrid topologies. Finally, Appendix 5.B discusses an alternative algorithm tailored for MS networks.

5.4 Cramér-Rao Lower Bound (CRLB)

This section deals with the computation of the Cramér-Rao lower bound (CRLB) for the estimation of clock frequencies and phases with the observation model (5.6). The CRLB is a lower bound on the accuracy of any unbiased estimator. Parameters are organized in the $2K_u \times 1$ vector $\mathbf{x} = \left[\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_{K_u}^T\right]^T$, where $\mathbf{x}_i^T = [\alpha_i, \beta_i]$. The CRLB for an unbiased estimator $\hat{\mathbf{x}}$ is defined as

$$\mathrm{Cov}(\hat{\mathbf{x}}) = E\left[(\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^T\right] \geq \mathbf{F}^{-1}, \tag{5.69}$$

where $\mathbf{F}$ is the $2K_u \times 2K_u$ Fisher information matrix (FIM). The CRLB can be computed only for an ideal estimator that has somehow obtained the observations taken on all links. In particular, it is assumed that N time offset observations $\{o_{ij}[n]\}_{n=0}^{N-1}$ are taken over each link $(i,j) \in \mathcal{E}$. Let $\mathbf{o}$ be the the $NK^2 \times 1$ observation vector. In the following, the CRLB is computed for the case of Gaussian observation noise, whereby $\mathbf{o} \sim \mathcal{N}(\bar{\mathbf{o}}(\mathbf{x}), \mathbf{C})$. The elements of the FIM are [87]

$$[\mathbf{F}]_{ij} = \left[\frac{\partial \bar{\mathbf{o}}(\mathbf{x})}{\partial x_i}\right]^T \mathbf{C}^{-1} \left[\frac{\partial \bar{\mathbf{o}}(\mathbf{x})}{\partial x_j}\right], \tag{5.70}$$

where x_i is the i-th component of the parameter vector, $x_i = [\mathbf{x}]_i$. In the case of closed-loop synchronization, the covariance of observations $\mathbf{C}$ depends on loop parameters and network topology. In this case, in fact, time offsets measured on a given link are observed at a distant node after being filtered by the PLL's of the intevening nodes. The CRLB of closed-loop algorithms is computed numerically for the topologies of interest in Section 5.6. The following discussion focuses on the case of open-loop synchronization, whereby observations are i.i.d. over time and links, $\mathbf{C} = \sigma_w^2 \mathbf{I}$, and the CRLB may be computed in

closed form. The elements of the FIM are [87]

$$[\mathbf{F}]_{ij} = \frac{1}{\sigma_{ij}^2} \sum_{(l,m)\in\mathcal{E}} \sum_{n=0}^{N-1} \frac{\partial o_{lm}[n]}{\partial x_i} \frac{\partial o_{lm}[n]}{\partial x_j}. \tag{5.71}$$

The link weights are assumed to be unitary whenever a link exists between two slave nodes, $a_{ij} = 1$ if $(i,j) \in \mathcal{E}$, or between a master and a slave node, $m_{ik} = 1$ if $(i,k) \in \mathcal{E}$. Given the structure of the parameter vector $\mathbf{x}$ and the observation model (5.6), the FIM $\mathbf{F}$ is a $K_u \times K_u$ block matrix

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} & \cdots & \mathbf{F}_{1K_u} \\ \mathbf{F}_{21} & \mathbf{F}_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ \mathbf{F}_{K_u1} & \cdots & \cdots & \mathbf{F}_{K_uK_u} \end{bmatrix}, \tag{5.72}$$

where the 2×2 submatrix $\mathbf{F}_{ij}$ is

$$\mathbf{F}_{ij} = -\frac{1}{\sigma_w^2} (a_{ij} + a_{ji}) \left(\mathbf{R}^T\mathbf{R}\right) \qquad \text{for } j \neq i \tag{5.73}$$

$$\mathbf{F}_{ii} = \frac{1}{\sigma_w^2} \left[\sum_{j=K_u+1}^{K} m_{ij} + \sum_{j=1}^{K_u} (a_{ij} + a_{ji}) \right] \left(\mathbf{R}^T\mathbf{R}\right) \qquad \text{for } j = i, \tag{5.74}$$

The $N \times 2$ matrix $\mathbf{R}$ was defined in (5.52), and the 2×2 matrix $\mathbf{R}^T\mathbf{R}$ is the FIM for a linear regression problem [87]. The *out-degree* of node k is defined as the sum of weights of all *outgoing* links, $d_{o,k} = \sum_{j=1}^{K_u} a_{ji}$, and the $K_u \times K_u$ *complementary* Laplacian as $\mathbf{L}_c = \mathbf{D}_o - \mathbf{A}^T$, where the diagonal matrix $\mathbf{D}_o = \mathbf{diag}\,(d_{o,1}, d_{o,2}, \ldots, d_{o,K_u})$. Notice that the complementary Laplacian is always singular as its rows sum to zero, $\mathbf{L}_c\mathbf{1} = \mathbf{0}$. Given (5.72)-(5.74) the FIM can be written in a convenient compact form as

$$\mathbf{F} = \frac{1}{\sigma_w^2} (\mathbf{L} + \mathbf{L}_c) \otimes \left(\mathbf{R}^T\mathbf{R}\right), \tag{5.75}$$

where $\otimes$ denotes the Kronecker product.

In the case of MS and hybrid networks, the matrix $(\mathbf{L} + \mathbf{L}_c)$ is invertible, and it holds that $\mathbf{F}^{-1} = \sigma_w^2 (\mathbf{L} + \mathbf{L}_c)^{-1} \otimes \left(\mathbf{R}^T\mathbf{R}\right)^{-1}$, by the properties of the Kronecker product [85].

From (5.69) and (5.75), a lower bound on the average MSE (defined in Section 5.3) can be written as

$$\xi_p^2 \geq \frac{2\left(2N-1\right)}{N\left(N+1\right)}\frac{\sigma_w^2}{K}\sum_{i=1}^{K_u}\frac{1}{\lambda_i\left(\mathbf{L}+\mathbf{L}_c\right)} \tag{5.76}$$

$$\xi_f^2 \geq \frac{12}{T_{SF}^2 N\left(N^2-1\right)}\frac{\sigma_w^2}{K}\sum_{i=1}^{K_u}\frac{1}{\lambda_i\left(\mathbf{L}+\mathbf{L}_c\right)}. \tag{5.77}$$

The CRLB is achieved by the open-loop algorithm on MS topologies where each node is constrained to have only a *single* parent (see Appendix 5.B). In all other cases, the open-loop algorithm does not achieve the bound. This was expected, as the open-loop algorithm was derived after processing (i.e., degrading) the original observations. If nodes are allowed to exchange observed time offsets with their neighbors, distributed algorithms capable of achieving the CRLB can be devised by exploiting the linearity of the observation model (5.6) (see [89] for the case of hybrid and MS networks). Nevertheless, as already remarked in Section 5.1.1, nodes are not allowed to communicate timestamps to each other when synchronization is based on periodic beacon transmissions, which is the case considered here.

In the case of MC networks, there are no master nodes, $\mathbf{M}=\mathbf{0}$, and the Laplacian matrix is singular since its rows sum to zero, $\mathbf{L1}=\mathbf{0}$. Also, as all links are bi-directional, the adjacency matrix is symmetric and $\mathbf{L}_c=\mathbf{L}$. This means that the matrix $\left(\mathbf{L}+\mathbf{L}_c\right)=2\mathbf{L}$ is singular as well and the FIM (5.75) is not invertible. Again, this fact reflects the impossibility of estimating absolute phase and frequency offsets in a MC network. Estimation problems with singular FIM have been studied in several works on estimation theory, see, e.g., [90]. The CRLB is computed by considering a suitable transformation of the original parameter set. In particular, consider the vector $\mathbf{y}=\left[\left(\mathbf{I}_K-\mathbf{J}\right)\otimes\mathbf{I}_2\right]\mathbf{x}$ as in Section 5.3.1. The CRLB for this problem is [90]

$$\mathrm{Cov}(\hat{\mathbf{y}})=E\left[\left(\hat{\mathbf{y}}-\mathbf{y}\right)\left(\hat{\mathbf{y}}-\mathbf{y}\right)^T\right]\geq\mathbf{G}\mathbf{F}^{\dagger}\mathbf{G}, \tag{5.78}$$

where $\mathbf{G} = (\mathbf{I}_K - \mathbf{J}) \otimes \mathbf{I}_2$ and $\mathbf{F}^\dagger$ is the Moore-Penrose pseudoinverse of the FIM in (5.75). Since the Laplacian $\mathbf{L}$ is symmetric and $(\mathbf{I}_K - \mathbf{J})$ is a projection matrix, (5.78) can be simplified by employing the properties of the Kronecker product in [91],

$$\mathrm{Cov}(\hat{\mathbf{y}}) \geq \frac{\sigma_w^2}{2}\mathbf{L}^\dagger \otimes \left(\mathbf{R}^T\mathbf{R}\right)^{-1}. \tag{5.79}$$

From (5.79), the bounds on the average MSE for phase and frequency estimates read

$$\xi_p^2 \geq \frac{2\left(2N-1\right)}{N\left(N+1\right)}\frac{\sigma_w^2}{K}\sum_{i=2}^{K}\frac{1}{2\lambda_i\left(\mathbf{L}\right)} \tag{5.80}$$

$$\xi_f^2 \geq \frac{12}{T_{SF}^2 N\left(N^2-1\right)}\frac{\sigma_w^2}{K}\sum_{i=2}^{K}\frac{1}{2\lambda_i\left(\mathbf{L}\right)}. \tag{5.81}$$

Notice that, differently from (5.76)-(5.77), the summation in (5.80)-(5.81) starts from $i = 2$ as the smallest eigenvalue of the Laplacian matrix is null in this case, $\lambda_1\left(\mathbf{L}\right) = 0$.

5.5 Discussion

Comparing the presented algorithms among themselves and the CRLB is not straightforward as the closed-loop estimator employs observations over infinite past, while the open-loop estimator and the CRLB are derived assuming finite-length observations. In [58], the length of the *effective* closed-loop observation window N_{eff} is computed by opening the loop and evaluating the accuracy of an open-loop estimator employing the same observations as the PLL. Consider a single MS link: the open-loop counterpart to the type 2 PLL is simple linear regression. Therefore, N_{eff} is related with the PLL (phase) noise bandwidth (5.39) as

$$\frac{2\left(2N_{eff}-1\right)}{N_{eff}\left(N_{eff}+1\right)} = 2b_L^{(p)}\left(1\right), \tag{5.82}$$

where the left-hand side is recognized as the accuracy of intercept estimation achieved by linear regression. Approximating (5.82) for large N_{eff}, it is $N_{eff} = 2/b_L^{(p)}\left(1\right)$.

Consider the case of a *regular* MC network (e.g., a ring network). When the network is poorly connected (i.e., when the number of nodes K is large with respect to node degree d) the smallest nonzero eigenvalue is small, $\mu_2 = \lambda_2\left(\mathbf{D}^{-1}\mathbf{L}\right) \ll 1$, and the accuracy (5.41) may be roughly approximated in terms of the effective observation window N_{eff} as

$$\xi_p^2 \simeq \frac{4\sigma_w^2}{N_{eff}} \frac{1}{1+4\zeta^2} \frac{d}{\lambda_2^2\left(\mathbf{L}\right)}, \tag{5.83}$$

where $\lambda_2\left(\mathbf{D}^{-1}\mathbf{L}\right) = \lambda_2\left(\mathbf{L}\right)/d$ since $d_i = d$ for a regular network. By the same arguments and by the symmetry of the Laplacian matrix for MC networks, $\mathbf{L} = \mathbf{L}^T$, the accuracy of the open-loop algorithm (5.62) may be approximated as

$$\xi_p^2 \simeq \frac{4\sigma_w^2}{N_{eff}} \frac{d}{\lambda_2^2\left(\mathbf{L}\right)}. \tag{5.84}$$

By comparing (5.83) with (5.84), it is seen that the network MSE of the closed-loop algorithm is smaller by a factor of $\left(1+4\zeta^2\right) = 1 + \kappa_1/\kappa_2$. This effect is clearly due to the noise filtering action of the PLL loop filter. The approximation of the open-loop CRLB (5.80) for poorly connected networks reads

$$\xi_p^2 \geq \frac{4\sigma_w^2}{N_{eff}} \frac{1}{2\lambda_2\left(\mathbf{L}\right)}. \tag{5.85}$$

From (5.83)-(5.85), distributed algorithms appear inefficient with respect to the open-loop CRLB, as their synchronization error is inversely proportional to the *square* of the smallest Laplacian eigenvalue $\lambda_2\left(\mathbf{L}\right)$. The noise filtering property of the closed-loop scheme partially compensates the cost of distributed synchronization.

A similar approximation may be carried out in the case of MS and hybrid networks. In particular, the open-loop MSE (5.57) is

$$\xi_p^2 \simeq \frac{4\sigma_w^2}{N_{eff}} \frac{K_u}{K} \frac{1}{\lambda_1\left(\mathbf{L}^T\mathbf{D}^{-1}\mathbf{L}\right)}, \tag{5.86}$$

while the open-loop CRLB (5.76) is

$$\xi_p^2 \geq \frac{4\sigma_w^2}{N_{eff}} \frac{K_u}{K} \frac{1}{\lambda_1 \left(\mathbf{L} + \mathbf{L}_c \right)}. \tag{5.87}$$

Unfortunately, closed-loop performance may not be derived in closed form for these topologies. By comparing (5.86) with (5.87), it is seen that the distributed algorithm is efficient only if the spectrum of $\mathbf{L}^T \mathbf{D}^{-1} \mathbf{L}$ is similar to the spectrum of $\mathbf{L} + \mathbf{L}_c$. As it will be shown in the next section, this is the case only with the MS architecture (see also Appendix 5.B).

Finally, it is important to stress how, besides performance considerations, several practical observations lean in favor of closed-loop synchronization algorithms. In fact, in order to achieve accurate synchronization, the open-loop algorithm needs to employ a large number of observations N. A large N entails three main drawbacks. Firstly, N samples require N super-frames to be acquired (long training phase) and a lot of memory to be allocated before processing. Secondly, communication resources need to be reserved for communication of current estimates (during the fusion phase). These considerations make the open-loop algorithm less appealing for implementation at the MAC layer.

5.6 Simulation Results

This section presents simulation results in order to compare the accuracy of open and closed-loop synchronization algorithms of Section 5.2-5.3 with the CRLB of a centralized procedure computed in Section 5.4. In order to compare the accuracy of distributed algorithms with the CRLB, it is assumed that the timestamp observation noise follows a Gaussian distribution. In all of the following simulations, the RMS jitter $\sigma_w = 10$ μs, $T_{SF} = 250$ ms, $T_S = 5$ ms, $\zeta = 5$. Also, only results regarding phase synchronization accuracy will be shown, as similar behavior has been observed with frequency accuracy. First, it is interesting to check the impact of the topologies introduced in Section 2.3 on the distribution of synchronization error in the network. Notably, the distribution shape

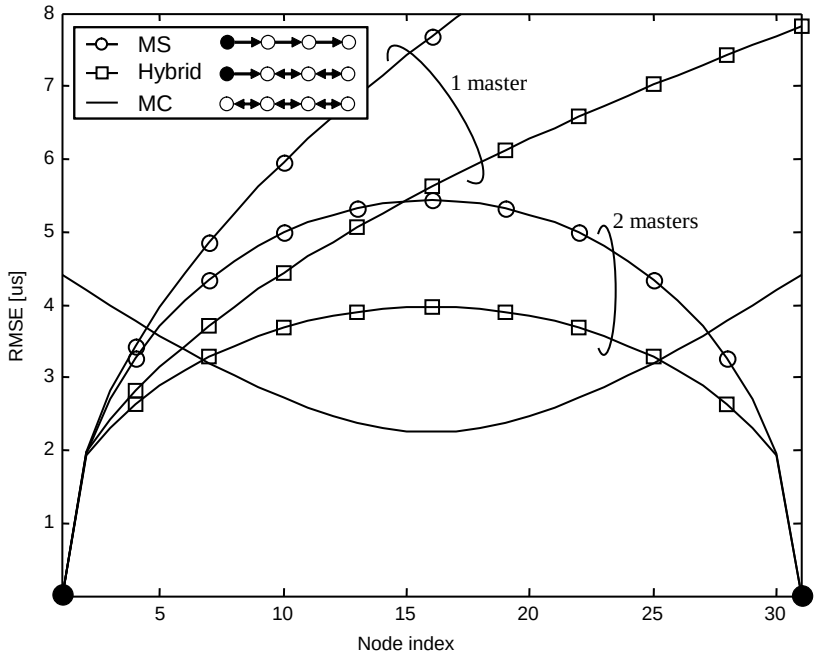


Figure 5.8 CRLB for a line network of $K = 31$ nodes ($N = 100$, $\sigma_w = 10$ μs).

does not change significantly whether practical algorithms or the CRLB are considered. Figure 5.8 shows the CRLB for a line network of $K = 31$ nodes with nearest-neighbor connectivity. The limit accuracy of the three architectures is computed with the open-loop formula (5.75). With the MC architecture, the synchronization error depends on the connectivity of the whole network and on the distance of each node from the network border. With MS and hybrid architectures, instead, the error depends solely on the distance of each node from the master nodes. The consequence is that the accuracy with MC topology degrades smoothly from the center through the borders of the network. With MS and hybrid topologies, instead, a smooth accuracy distribution is obtained only by deploying two master nodes at network edges. The CRLB for a grid network of $K = 31 \times 31$ nodes with MS and MC architectures is depicted in Figure 5.9. Analogously to the line network case, the error with MC architecture concentrates at the borders of the network. A smooth error distribution is achieved also with MS architecture and 4 master nodes deployed at corner points as in Figure 5.9.a. The symmetry of level curves

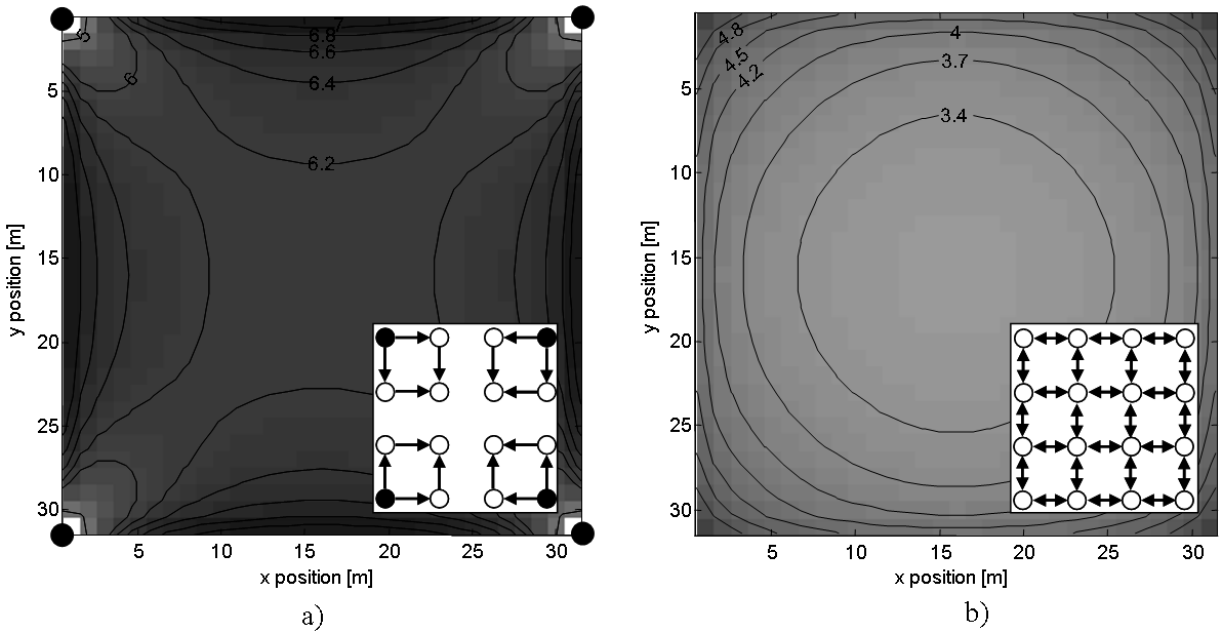


Figure 5.9 CRLB (in μs) for a grid network of $K = 31 \times 31$ nodes with a) MS and b) MC architecture ($N = 10$, $\sigma_w = 10$ μs).

is significantly different in the two cases, though. Finally, MC architecture provides an overall higher accuracy as it exploits network connectivity more efficiently.

The performance of the open-loop and closed-loop algorithms introduced in Section 5.2-5.3 is compared with the CRLB in Figure 5.10 for a line network with MC topology. The open-loop accuracy is computed with (5.60), which constitutes a lower bound for the accuracy achieved by the distributed algorithm introduced in Section 5.3.2. As expected from the discussion in Section 5.5, the closed-loop algorithm clearly outperforms the open-loop algorithm, and both are quite far from the accuracy limit of a centralized estimator. The approximations (5.83)-(5.84) provide a raw prediction of accuracy in both open and closed-loop cases. For the same line network, the performance of proposed algorithms with MS and hybrid topologies is depicted in Figure 5.11. Again, closed-loop outperforms the open-loop approach. Both schemes achieve the CRLB with MS architecture, while the open-loop performance is pretty far from the CRLB when

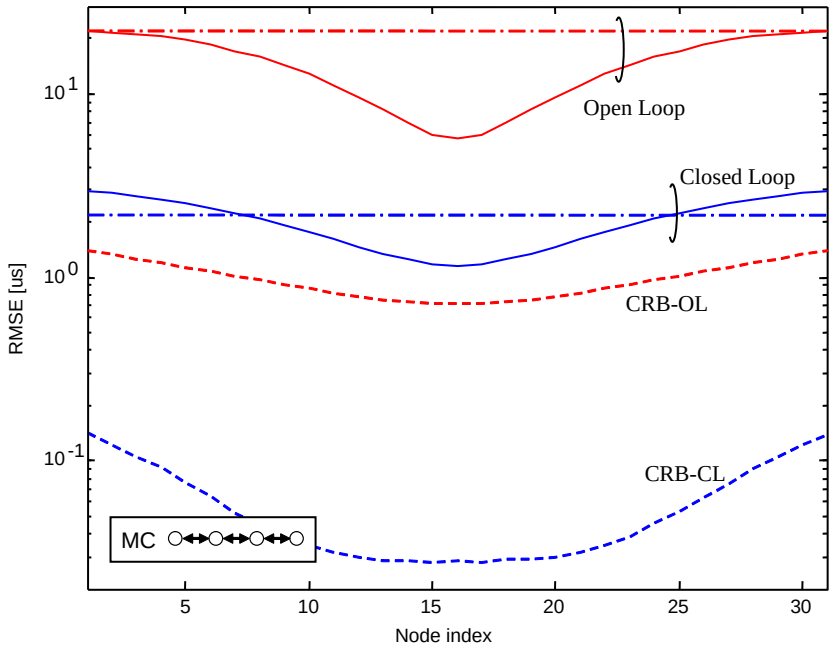


Figure 5.10 Performance of open and closed-loop algorithms compared with CRLB for a line network of $K = 31$ nodes, MC architecture ($N = 1000$, $\sigma_w = 10$ μs). The dash-dotted lines are the approximations (5.83)-(5.84).

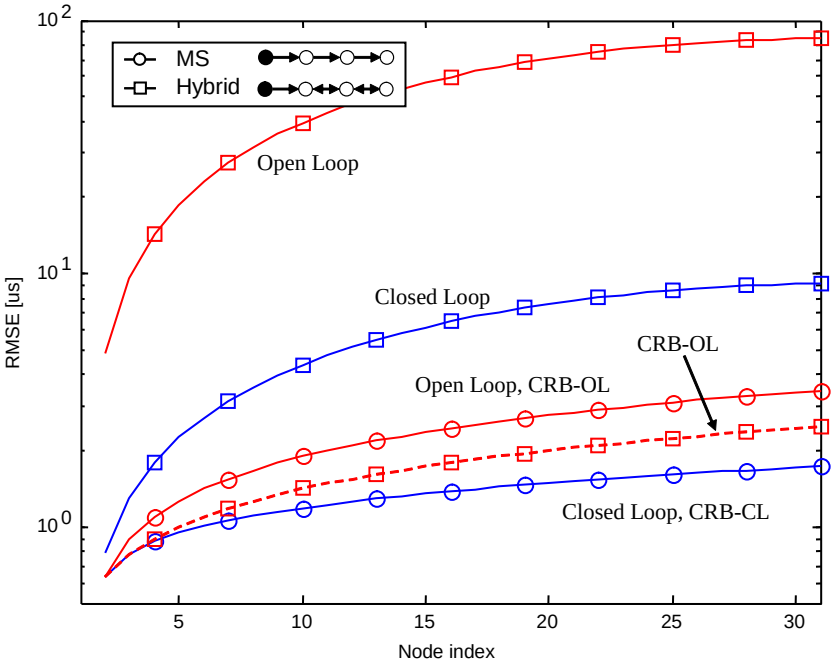


Figure 5.11 Performance of open and closed-loop algorithms compared with CRLB for a line network of $K = 31$ nodes, MS and hybrid architectures ($N = 1000$, $\sigma_w = 10$ μs).

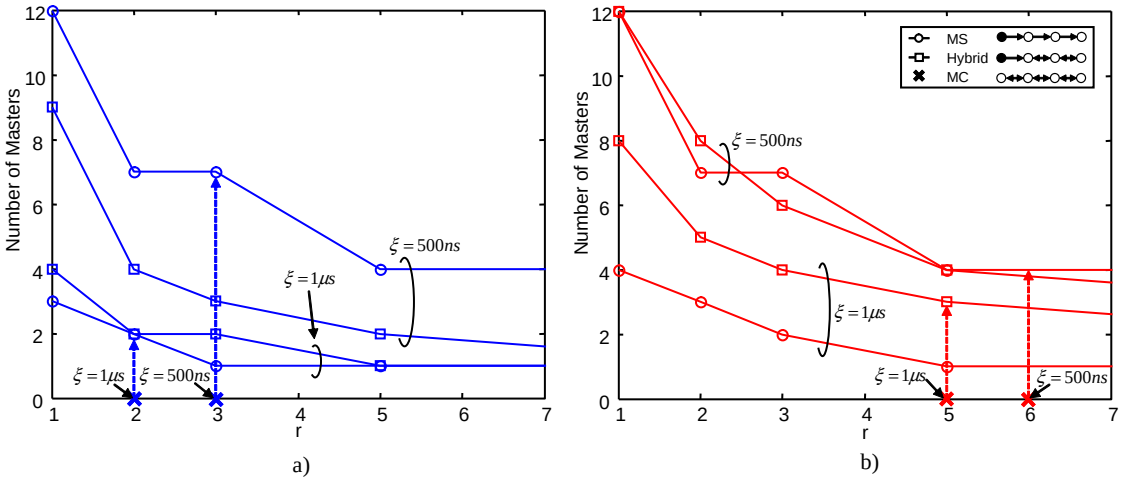


Figure 5.12 Trade-off between number of master nodes and transmission range r to achieve a required RMS network accuracy ξ_p; a) closed-loop and b) open-loop algorithm on a line network of $K = 31$ nodes ($N = 1000$, $\sigma_w = 10$ μs).

employing the hybrid architecture. In general, it can be concluded that distributed algorithms fail to be efficient when the coupling is peer-to-peer.

From Figure 5.10-5.11, it is observed how MS outperforms hybrid and MC architectures when employing distributed algorithms on a line network with nearest neighbor connectivity ($r = 1$). In the following it will be shown how MC and hybrid topologies turn convenient when the network is well connected. Higher connectivity provides a reduction of the RMS network synchronization error ξ_p at the expense of a larger transmission range r. When employing MS and hybrid topologies, a given RMS error ξ_p may be achieved also by deploying more master nodes without increasing r. The trade-off between transmission range r and the number of deployed master nodes is depicted in Figure 5.12 for a line network of $K = 31$ nodes. For the closed-loop algorithm in Figure 5.12.a, $\xi_p = 1$ μs is achieved by hybrid and MS topologies with roughly the same number of master nodes for all values of r. The hybrid topology is definitely more efficient than MS if the target RMS error is $\xi_p = 500$ ns. The MC architecture largely benefits from increasing r. In particular, it achieves $\xi_p = 500$ ns already with $r = 3$, while hybrid and MS need to be deployed with 3 and 7 master nodes, respectively. It is concluded that topologies employing peer-to-peer

coupling are more efficient in exploiting good network connectivity. For the open-loop algorithm in Figure 5.12.b, it is observed that MS and hybrid architectures are convenient only for a high target accuracy.

5.7 Conclusions

TDMA medium access protocols require accurate time synchronization in order to avoid packet collision events. In this chapter, one such protocol has been considered, where synchronization is obtained by the periodic transmission of beacon frames. The study has focused on the accuracy attainable by distributed synchronization techniques. Two algorithms for distributed synchronization have been developed. The first (closed-loop algorithm) is based on the distributed control of local clocks trough a type 2 phase-locked loop (PLL), while the second (open-loop algorithm) is based on the distributed estimation of the phase and frequency offsets of the local clocks. The two protocols are both suitable for implementation over general synchronization networks; in particular, the chapter considers mutually-coupled (MC), master-slave (MS), and hybrid networks. The synchronization accuracy of the open-loop algorithm is derived for all topologies of interest, while the accuracy of the closed-loop algorithm exact analytical results are available only for regular MC networks. The performance of practical algorithms is then compared with the Cramér-Rao lower bound (CRLB) for the problem at hand. Results show that distributed algorithms are inefficient with respect to the CRLB over peer-to-peer topologies, whereas they achieve the accuracy limit over MS hierarchical architectures. Finally, the performance of MC and hybrid topologies improves rapidly when increasing network connectivity, while MS proves to be the best choice in poorly connected networks.

Appendix 5.A: Open-loop synchronization for MC networks

In the case of MC networks, the Laplacian matrix $\mathbf{L}$ is singular, and it is not possible to employ (5.55). In order to simplify the analysis, consider the normalized system

$$\begin{aligned}\left(\mathbf{D}^{-\frac{1}{2}} \otimes \mathbf{I}\right) \mathbf{o} &= \left(\mathbf{D}^{-\frac{1}{2}} \otimes \mathbf{I}\right) \mathbf{H}\mathbf{x} + \left(\mathbf{D}^{-\frac{1}{2}} \otimes \mathbf{I}\right) \mathbf{w} \\ &= \left(-\mathbf{D}^{-\frac{1}{2}}\mathbf{L} \otimes \mathbf{R}\right) \mathbf{x} + \tilde{\mathbf{w}},\end{aligned} \tag{5.88}$$

where $\mathrm{Cov}\left(\tilde{\mathbf{w}}\right) = \mathbf{I}$. The BLUE of $\mathbf{y}$ corresponds to the least-square solution of (5.88), which is found by employing the Moore-Penrose inverse,

$$\begin{aligned}\hat{\mathbf{y}} &= \left(-\mathbf{D}^{-\frac{1}{2}}\mathbf{L} \otimes \mathbf{R}\right)^{\dagger} \left(\mathbf{D}^{-\frac{1}{2}} \otimes \mathbf{I}\right) \mathbf{o} \\ &= -\left(\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\right)^{\dagger} \mathbf{D}^{-\frac{1}{2}} \otimes \mathbf{R}^{\dagger}\mathbf{o},\end{aligned} \tag{5.89}$$

where we exploited the compatibility of the Kronecker product with the Moore-Penrose inverse, $\left(\mathbf{A} \otimes \mathbf{B}\right)^{\dagger} = \mathbf{A}^{\dagger} \otimes \mathbf{B}^{\dagger}$ [91]. Notice that $\mathbf{D}$ cannot be simplified in (5.89) as the reverse-order law fails (in general) for the pseudoinverse, $\left(\mathbf{AB}\right)^{\dagger} \neq \mathbf{B}^{\dagger}\mathbf{A}^{\dagger}$ [50]. The final result in (5.59) is obtained from (5.89) by the property $\mathbf{A}^{\dagger} = \left(\mathbf{A}^{T}\mathbf{A}\right)^{\dagger} \mathbf{A}^{T}$ [50]. By direct substitution, it can be verified that (5.89) is the sum of $\mathbf{y}$ and a random term with covariance (5.60).

We now prove the convergence of the distributed fusion algorithm presented in Section 5.3.2 on MC networks. As noted before, phases and frequencies are updated independently in (5.67). Therefore, in order to simplify the treatment, we consider the following simplified recursion

$$\mathbf{z}\left[p+1\right] = \left(\mathbf{I} - \epsilon\mathbf{D}^{-1}\mathbf{L}\right) \mathbf{z}\left[p\right] - \epsilon\mathbf{b}, \tag{5.90}$$

where $\mathbf{z}\left[p\right]$ is a vector comprising either phase or frequency estimates, and $\mathbf{b}$ is defined accordingly. Notice that the normalized Laplacian $\mathbf{D}^{-1}\mathbf{L}$ is singular since $\mathbf{D}^{-1}\mathbf{L}\mathbf{1} = \mathbf{0}$. If the network is connected, the null eigenvalue has multiplicity 1, and all other eigenvalues

are real and positive by the *generalized* eigenvalue theorem [52]. Therefore, the system matrix in (5.90) can be diagonalized by eigenvalue decomposition as

$$\left(\mathbf{I}_K - \epsilon \mathbf{D}^{-1}\mathbf{L}\right) = \mathbf{U} \begin{bmatrix} 1 & 0 \\ 0 & \mathbf{I}_{K-1} - \epsilon \mathbf{\Lambda} \end{bmatrix} \mathbf{U}^{-1}, \tag{5.91}$$

where $\mathbf{\Lambda} = \operatorname{diag}\left(\lambda_2\left(\mathbf{D}^{-1}\mathbf{L}\right), \ldots, \lambda_K\left(\mathbf{D}^{-1}\mathbf{L}\right)\right)$. If the relaxation coefficient is bounded as $0 < \epsilon < 2/\lambda_K\left(\mathbf{D}^{-1}\mathbf{L}\right)$, then it is $\rho\left(\mathbf{I} - \epsilon \mathbf{D}^{-1}\mathbf{L}\right) < 1$, and $\left(\mathbf{I} - \epsilon \mathbf{D}^{-1}\mathbf{L}\right)$ is a *semiconvergent* matrix according to the definition in [92]. Notice that, by Gershgorin's theorem, it is $\lambda_K\left(\mathbf{D}^{-1}\mathbf{L}\right) \leq 2$, and therefore $0 < \epsilon < 1$ is a sufficient condition for semiconvergence. Semiconvergence implies that the limit of the powers of $\left(\mathbf{I} - \epsilon \mathbf{D}^{-1}\mathbf{L}\right)$ converges to a rank-1 matrix, namely[9]

$$\lim_{p\to\infty} \left(\mathbf{I} - \epsilon \mathbf{D}^{-1}\mathbf{L}\right)^p = \mathbf{1}\mathbf{v}^T, \tag{5.92}$$

where $[\mathbf{v}]_i = d_i / \left(\sum_i d_i\right)$. Semiconvergence allows also to compute the following limit

$$\begin{aligned} \lim_{p\to\infty} \sum_{k=0}^{p} \left(\mathbf{I} - \epsilon \mathbf{D}^{-1}\mathbf{L}\right)^k - p\mathbf{1}\mathbf{v}^T &= \mathbf{U} \left(\sum_{k=0}^{p} \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I} - \epsilon \mathbf{\Lambda} \end{bmatrix}^k \right) \mathbf{U}^{-1} \\ &= \frac{1}{\epsilon} \mathbf{U} \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{\Lambda}^{-1} \end{bmatrix} \mathbf{U}^{-1} \\ &= \frac{1}{\epsilon} \left(\mathbf{D}^{-1}\mathbf{L}\right)^D, \end{aligned} \tag{5.93}$$

where $\mathbf{A}^D$ is the Drazin generalized inverse of $\mathbf{A}$ [50]. From the properties of the Drazin inverse, it holds that $\left(\mathbf{D}^{-1}\mathbf{L}\right)^D \left(\mathbf{D}^{-1}\mathbf{L}\right) = \mathbf{I} - \mathbf{1}\mathbf{v}^T$. Now, the recursion (5.90) can be solved by direct substitution,

$$\mathbf{z}\left[p+1\right] = \left(\mathbf{I} - \epsilon \mathbf{D}^{-1}\mathbf{L}\right)^{p+1} \mathbf{z}\left[0\right] - \sum_{k=0}^{p} \left(\mathbf{I} - \epsilon \mathbf{D}^{-1}\mathbf{L}\right)^k \epsilon \mathbf{b}. \tag{5.94}$$

[9] In general, the powers of a square semiconvergent matrix $\mathbf{A}$ converge to the projector onto $N\left(\mathbf{I} - \mathbf{A}\right)$ along $R\left(\mathbf{I} - \mathbf{A}\right)$ [50].

Similarly to the PLL analysis in Section 5.2.2, estimates of phase or frequency offsets are here defined with respect to a weighted average, namely $\mathbf{s}\,[p] = \left(\mathbf{I} - \mathbf{1}\mathbf{v}^T\right)\mathbf{z}\,[p]$. By the substitution of (5.92)-(5.93) in (5.94), it can be seen that

$$\lim_{p\to\infty} \mathbf{s}\,[p] = -\left(\mathbf{D}^{-1}\mathbf{L}\right)^D \mathbf{b}. \tag{5.95}$$

Consider now the original recursion (5.67), and define the running parameter estimate as $\hat{\mathbf{y}}[p] = \left[\left(\mathbf{I} - \mathbf{1}\mathbf{v}^T\right) \otimes \mathbf{I}_2\right] \hat{\mathbf{x}}[p]$. By employing (5.95), it can be verified that $\hat{\mathbf{y}}[p]$ converges to a noisy estimate of the transformed parameters $\mathbf{y} = \left[\left(\mathbf{I} - \mathbf{1}\mathbf{v}^T\right) \otimes \mathbf{I}_2\right] \mathbf{x}$. After some manipulations, the covariance of the estimation error is found to be

$$\mathrm{Cov}\left(\hat{\mathbf{y}}[\infty]\right) = \sigma_w^2 \mathbf{D}^{-\frac{1}{2}} \left(\mathbf{D}^{-\frac{1}{2}}\mathbf{L}^T\mathbf{D}^{-1}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\right)^{\dagger} \mathbf{D}^{-\frac{1}{2}} \otimes \left(\mathbf{R}^T\mathbf{R}\right)^{-1}. \tag{5.96}$$

In deriving (5.96), we have exploited the similarity of $\mathbf{D}^{-1}\mathbf{L}$ with the symmetric matrix $\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$, and the fact that the Drazin and Moore-Penrose inverses coincide for symmetric matrices. In general, the estimation error of the distributed algorithm (5.96) is very close to the optimum (5.60) for graphs with almost regular degree distribution. The distributed and centralized estimates achieve the same accuracy when $\mathbf{D} = d\mathbf{I}$, i.e., on regular networks.

Appendix 5.B: Open-loop synchronization for MS networks: CRLB and an alternative algorithm

Let us consider first a MS chain network. In this case, the CRLB assumes a particularly simple form. Let node 1 be the master node, and the remaining $K_u = K - 1$ nodes be slave

nodes. If we have nearest-neighbor connectivity, the $K_u \times K_u$ Laplacian matrix is

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix}, \tag{5.97}$$

which is is tridiagonal and full-rank. The inverse of the FIM is

$$\mathbf{F}^{-1} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \cdots & K_u \end{bmatrix} \otimes \left(\mathbf{R}^T\mathbf{R}\right)^{-1}, \tag{5.98}$$

From (5.98), the bound for the phase and frequency estimates of node i is

$$\mathrm{Cov}\left(\begin{bmatrix} \hat{\alpha}_i \\ \hat{\beta}_i \end{bmatrix}\right) \geq (i-1)\,\sigma_w^2\left(\mathbf{R}^T\mathbf{R}\right)^{-1}. \tag{5.99}$$

For a large number of samples, $N \to \infty$, $\mathrm{Var}\left(\hat{\beta}_i\right) \simeq (i-1)\,\frac{\sigma_w^2}{N}$ and $\mathrm{Var}\left(\hat{\alpha}_i\right) \simeq (i-1)\,\frac{12\sigma_w^2}{T_{SF}^2 N^3}$. From (5.103) the noise (jitter) accumulation due to the hierarchical network structure is apparent.

For this network, a simple iterative algorithm may be devised by direct inspection of the inversion matrix $\mathbf{W}$ in (5.55). The Laplacian matrix for a MS chain network is (5.97), which is lower triangular and invertible. The inversion matrix is $\mathbf{W} = \mathbf{L}^{-1} \otimes \mathbf{W}_R$, where the inverse of the Laplacian is the lower triangular matrix

$$\mathbf{L}^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 1 & \cdots & 1 & 1 \end{bmatrix}. \tag{5.100}$$

From (5.100), we can express the ML estimate $\hat{\mathbf{x}}_i^T = \left[\hat{\alpha}_i, \hat{\beta}_i\right]$ in closed form a

$$\begin{bmatrix} \hat{\alpha}_i \\ \hat{\beta}_i \end{bmatrix} = -\sum_{k=2}^{i} \mathbf{R}^{\dagger}\mathbf{o}_k. \tag{5.101}$$

Notice that in the MS chain each node observes only time offset with respect to the preceding node, that is $o_k[n] = (\alpha_{k-1} - \alpha_k)\, nT_{SF} + (\beta_{k-1} - \beta_k) + w_k[n]$. Therefore, the 2×1 vector $\mathbf{R}^{\dagger}\mathbf{o}_k$ is the standard ML regression of the frequency and phase offsets, $\Delta\alpha_k = (\alpha_{k-1} - \alpha_k)$ and $\Delta\beta_{i-1} = (\beta_{i-1} - \beta_i)$. Given these considerations, we may rewrite (5.101) as

$$\begin{bmatrix} \hat{\alpha}_i \\ \hat{\beta}_i \end{bmatrix} = -\sum_{k=2}^{i} \begin{bmatrix} \hat{\Delta\alpha}_k \\ \hat{\Delta\beta}_k \end{bmatrix}. \tag{5.102}$$

The estimator (5.102) may be implemented by a distributed *sequential* algorithm that takes exactly K steps to complete. In the first step, node 1 computes $\hat{\alpha}_1 = -\hat{\Delta\alpha}_1$, $\hat{\beta}_1 = -\hat{\Delta\beta}_1$ and sends a packet to node 2 with their values. In the second step, node 2 computes $\hat{\Delta\alpha}_2$, $\hat{\Delta\beta}_2$ and sends a packet to node 3 with $\hat{\alpha}_2 = \hat{\alpha}_1 - \hat{\Delta\alpha}_2$, $\hat{\beta}_2 = \hat{\beta}_1 - \hat{\Delta\beta}_2$, and so on. The attained accuracy is as in (5.56). In particular, it holds that

$$\mathrm{Cov}\left(\begin{bmatrix} \hat{\alpha}_i \\ \hat{\beta}_i \end{bmatrix}\right) = (i-1)\,\sigma_w^2 \left(\mathbf{R}^T\mathbf{R}\right)^{-1}, \tag{5.103}$$

which coincides with the CRLB in (5.99). This is not surprising, as the synthetic observation (5.49) coincides with the original observation (5.6) for a MS chain network.

Now, the Laplacian of *any* MS network may be cast into a lower triangular form as (5.97) through appropriate node indexing. As the inverse of a lower triangular matrix is also lower triangular [85], analogous sequential protocols may be derived for *any* MS network. If applied on a tree network (where each node can have only a single parent), this protocol is identical to the operation of the FTSP protocol [37].

CHAPTER 6

ACCURATE SYNCHRONIZATION WITH LOW DUTY CYCLES

The lifetime of a Wireless Sensor Networks (WSN) is defined with respect to the specific task the network is designed for (e.g., monitoring, surveillance, sensing) [93] and depends, barring malfunctioning devices, mainly on the energy consumption of the participating nodes. Most of the energy consumed by a sensor node is drained by the RF transceiver module, while the energy required by the MCU and sensor units is less relevant. For this reason, most MAC layer protocols (including TDMA protocols) enable nodes to shut down their RF circuitry ("sleep" mode) when not transmitting or receiving, thereby allowing for relevant energy savings. Also, the energy efficiency of a MAC protocol is degraded when a packet collisions occur and when nodes are required to keep their radio on for a long time when nobody is transmitting (idle listening). With TDMA protocols, data is transmitted according to a pre-defined schedule and collisions and idle listening are inherently minimized. The drawback of TDMA is the requirement of tight network-wide time synchronization, which is typically regarded as a factor of energy consumption. In fact, if the MAC protocol is beacon-enabled, a node needs to be awake during the whole beacon transmission interval. Also, synchronization need to be refreshed frequently because of clock drifts and inaccuracies, thereby forcing nodes to wake up just to re-synchronize with the network. For these reasons, TDMA protocols targeting energy efficiency do not employ beaconing, and synchronization information is carried along with data and ACK frames. Recent advances in clock design have also improved the accuracy of clocks without increasing hardware costs significantly. This chapter deals with comparing the performance of accurate clocks against the one achieved employing regular clocks and distributed synchronization algorithms.

6.1 System Model

6.1.1 *Non*-Beacon Enabled TDMA MAC Protocol

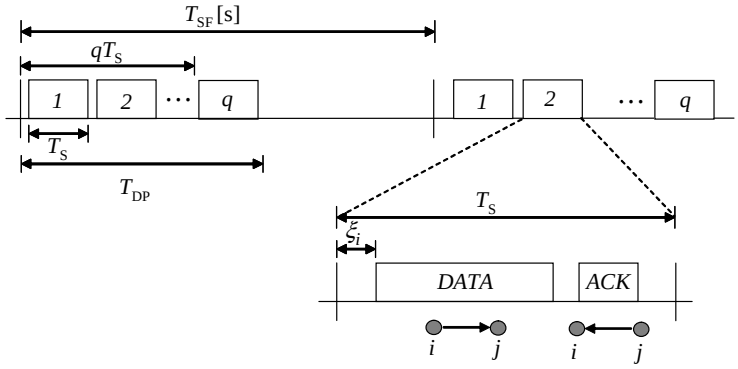


Figure 6.1 Super-Frame structure for a *non*-beacon-enabled TDMA MAC protocol.

A *non*-beacon-enabled TDMA MAC protocol is considered, whereby the time axis is divided into super-frames of duration T_{SF} seconds as in Figure 6.1. There is no beacon period (BP), and the data period (DP) may - in principle - occupy the whole super-frame. As before, the DP is then divided into time-slots, which are employed for data transmission by a *reservation*-based medium access protocol. Each time-slot is allocated to a given link (i, j), and comprises both data and ACK/NACK transmission. Each slot has a fixed duration T_S and it is associated to an integer index q, so that its nominal time offset from super-frame start is qT_S. It is assumed the existence of an efficient link scheduling algorithm, whereby each slot may be reused by non-interfering links. During the time-slot reserved for link (i, j), the data frame is transmitted with an offset ξ_i from the ideal slot start time due to the synchronization error at the transmitting node i. When receiving the data frame, node j measures the clock offset with respect to i and (if necessary) sends this measure back within the acknowledgment (ACK) message to node i. The maximum allowed synchronization error $\xi_{\max}$ depends on reception and transmission guard times (for a detailed discussion, see [31]).

Nodes have their radio turned on during the DP slots where they are either transmitting or receiving, while they are allowed to turn their radio off (“sleep” mode)

for the rest of the time. In low duty-cycle networks, the DP can be much shorter than the super-frame, which may range from few seconds to (in principle) a whole day. The duty-cycle D of the TDMA protocol is the ratio between the active period and the sleep period. If a node transmits at least one data frame in each super-frame, the duty cycle is $D = T_S/T_{SF}$. As an example, the duty cycle for $T_{SF} = 40$ s and $T_S = 10$ ms is $D = 0.025\%$. If the MAC protocol is well configured, the duty cycle of the radio transceiver matches the frequency with which the upper layers require information exchange among the nodes. The duty-cycle for most sensor networks applications is well below 1% [94].

The sleep schedule is usually maintained by a sleep timer driven by a low-power crystal oscillator (XO) (typically a 32 kHz quartz crystal). XO's are in general stable in the short-term, but very unstable in the long-term, and phase synchronization is lost after few seconds in sleep mode. For this reason, TDMA protocols are said not to be suited for low duty-cycles because of their stringent synchronization requirements. In order to maintain clock phase synchronization, nodes may have to turn their radio on just to perform synchronization tasks. Alternatively, large guard times can be arranged to account for timing mismatch, but this would cause nodes to waste precious energy while waiting for incoming transmissions (idle listening). The next section introduces a clock model capable of taking into account the effects of frequency instability.

6.1.2 Unstable Clock Model

In the following, it is assumed that each node broadcasts a single packet to all its neighbors during each super-frame. The transmission time of the $(n-1)$-th packet according to the local clock is $\tau_i[n-1]$. When the clock is free-running, the n-th packet is transmitted when the local clock reaches $\tau_i[n-1] + T_{SF}$, or

$$\tau_i[n] - \tau_i[n-1] = T_{SF}. \tag{6.1}$$

If the frequency $\alpha_i[n]$ changes from one super-frame to another, the clock is *unstable*. Packet transmission times (6.1) can be expressed in terms of absolute time $t_i[n]$ as

$$t_i[n] - t_i[n-1] = \frac{1}{1+\alpha_i[n-1]} T_{SF} \simeq (1 - \alpha_i[n-1])\, T_{SF}, \tag{6.2}$$

where the last approximation holds since frequency offset is always small, $\alpha_i[n] \ll 1$. Clock instability becomes particularly visible for large super-frame lengths T_{SF}. The main cause of long-term frequency fluctuations (or frequency *wander*) in XO's are environmental temperature variations, mechanical shocks and exposure to electric and magnetic fields. Focusing on temperature, the typical frequency-temperature characteristic for a tuning fork XO is [95]

$$\alpha_i\,(T) = \bar{\alpha}_i - k_S\,(T - T_o)^2 + k_D \frac{dT}{dt}, \tag{6.3}$$

where $\bar{\alpha}_i$ is the manufacturing accuracy (adimensional or, equivalently, in μs/μs), T is the external temperature in °C, k_S is the *static* temperature coefficient in °C^{-2}, while k_D is the *dynamic* temperature coefficient in s/°C. In static conditions, the characteristic has a parabolic shape, while in dynamic conditions it deviates from the static curve by an amount that depends on the rate of change in temperature (see Figure 6.2.a). Typical values for the static temperature coefficient k_S are between $3 \cdot 10^{-8}$ and $4.2 \cdot 10^{-8}$ °C^{-2}, thus yielding an offset around -160 ppm at -40 °C (with $T_o = 25$ °C).

The frequency dynamic is modeled as the sum of two random processes, namely

$$\alpha_i[n] = \bar{\alpha}_i + v_i[n] + \psi_i[n], \tag{6.4}$$

where $v_i[n]$ is a white noise process (white frequency modulation - WFM), while $\psi_i[n]$ is a random walk process (random walk frequency modulation - RWFM) with initial value $\psi_i[0] = 0$. The WFM component models quick frequency changes due to noise within the clock circuitry, while the RWFM component models frequency wander. The frequency *change* $(\alpha_i[n] - \alpha_i[n-1])\, T_{SF}$ corresponds to the *period jitter* of the clock over the time

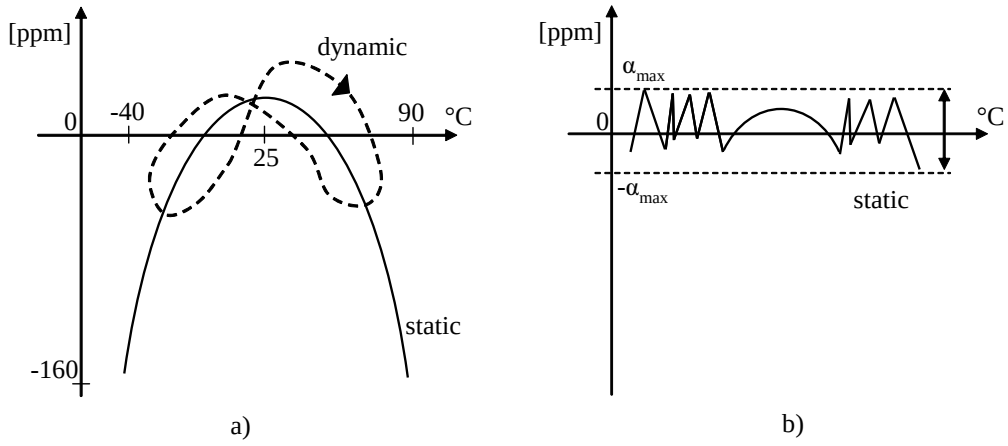


Figure 6.2 Frequency-temperature characteristic for a tuning fork crystal oscillator (XO) a) without and b) with temperature compensation. Dynamic characteristic is depicted assuming a sinusoidal temperature variation over time (see [95] for experimental results).

interval T_{SF} [96]. From (6.2)-(6.4), the local clock process may be modeled as a double-integrator system, namely

$$\begin{bmatrix} t_i[n] \\ \psi_i[n] \end{bmatrix} = \begin{bmatrix} 1 & -T_{SF} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_i[n-1] \\ \psi_i[n-1] \end{bmatrix} + \begin{bmatrix} -T_{SF} v_i[n-1] \\ \eta_i[n-1] \end{bmatrix} + \begin{bmatrix} (1-\bar{\alpha}_i)\, T_{SF} \\ 0 \end{bmatrix}, \tag{6.5}$$

where $\eta_i[n]$ is the RWFM step. Noise sources are assumed to be Gaussian i.i.d random variables, namely

$$\begin{bmatrix} v_i[n] \\ \eta_i[n] \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\eta^2 \end{bmatrix} \right). \tag{6.6}$$

The power of the WFM and RWFM components is related to the super-frame length as [96]

$$\begin{cases} \sigma_v^2 = q_\nu T_{SF}^{-1} \\ \sigma_\eta^2 = q_\eta T_{SF} \end{cases}. \tag{6.7}$$

The severity of the RWFM q_η depends on the intensity of the thermal and mechanical stress the clock undergoes within a super-frame.

In order to overcome frequency variations, possible solutions are either to improve the stability of the clock while keeping its cost as low as possible, or to enhance the

synchronization algorithm with frequency tracking capabilities. The next section deals with methods to improve clock quality by temperature compensation techniques.

6.2 Temperature-Compensated Clocks with Phase Synchronization

Conventional synchronization algorithms control only the clock phase, and the local time drifts away when synchronization is stopped during sleep mode because of frequency offsets. If the super-frame is too long with respect to the local oscillator accuracy, nodes will be required to wake up just to update (or "keep alive") their clock phase by exchanging synchronization information. Assuming that nodes are perfectly synchronized to the network reference time-scale at the end of a given super-frame, the clock *keep-alive interval* (or re-synchronization interval) is easily computed for MS networks as [31]

$$T_{KA} = \frac{\xi_{\max} - \bar{\delta}}{2\alpha_{\max}}, \tag{6.8}$$

where $\xi_{\max}$ is the maximum allowed phase offset, $\bar{\delta}$ is the phase synchronization accuracy (e.g., clock precision), and $\alpha_{\max}$ is the maximum frequency offset (in s/s). Synchronization requirements do not impair energy efficiency if nodes do not have to wake up just to re-synchronize their clocks, i.e., $T_{KA} \geq T_{SF}$. Given this requirement and (6.8), the maximum phase mismatch is related with T_{SF} as

$$\xi_{\max} = 2\alpha_{\max} T_{SF} + \bar{\delta}. \tag{6.9}$$

Since T_{SF} depends on the application duty cycle, energy efficiency can be achieved either by improving clock accuracy or by reducing spectral efficiency by allowing larger guard times to cope with large errors $\xi_{\max}$. The current trend in industry is to improve the clock accuracy $\alpha_{\max}$. As an example, if $\xi_{\max} = 1$ ms, $\bar{\delta} = 10$ μs and $T_{SF} = 40$ s, the required clock accuracy[1] is $\alpha_{\max} \leq 25$ ppm. With low duty cycles, most of the frequency offset

[1]Requirements of current standards are $\alpha_{\max} \leq 40$ ppm for IEEE 802.15.4, and $\alpha_{\max} \leq 100$ ppm for IEEE 802.11. Notice that these requirements include manufacturing accuracy and all environmental and aging effects [25].

between two nodes is due to environmental temperature variations, and higher accuracy can be obtained by the employment of a temperature-compensated clock (TCC). Temperature compensation is realized by keeping a look-up table where temperature is matched with the relative induced frequency offset. A control circuit adjusts the frequency of the clock given the temperature measured by a sensor and the corresponding offset in the table. The table can be compiled by a calibration procedure at the end of the manufacturing process. Digital compensation techniques constitutes the simplest (and cheapest) way to implement temperature compensation [97]. In particular, digital compensation is realized by adding or subtracting ticks to the clock register in order to compensate for the offset triggered by the measured temperature. Even when employing a TCC, frequency offsets cannot be compensated perfectly (see Figure 6.2.b and [97]): temperature is measured only periodically, and the dynamic frequency-temperature characteristic is not known. In the following, it is assumed a maximum offset $\alpha_{\max} = 10$ ppm after compensation, accounting for both static and dynamic compensation errors. This has been reported [31] as a satisfying stability for super-frame lengths below 1 minute and maximum error $\xi_{\max} = 1$ ms (with slot duration $T_s = 10$ ms). Notice that, for longer super-frame lengths, this is a somewhat optimistic assumption since $\alpha_{\max}$ still depends on T_{SF} in practice.

In the following, it is investigated whether enhancing the synchronization algorithm with *frequency tracking* capabilities constitutes a valid alternative to temperature compensation.

6.3 Type 2 Phase-Locked Loop (PLL)

If it is feasible to control the packet timestamping process, network synchronization may be achieved by employing suitably-designed phase-locked loops (PLL) [6][15]. A type 2 PLL as the one in Figure 6.3 is capable of tracking not only phase, but also frequency drifts. If the algorithm could track frequency changes perfectly, the maximum phase offset $\xi_{\max}$ would depend only on the maximum frequency *change* within a super-frame, and not on

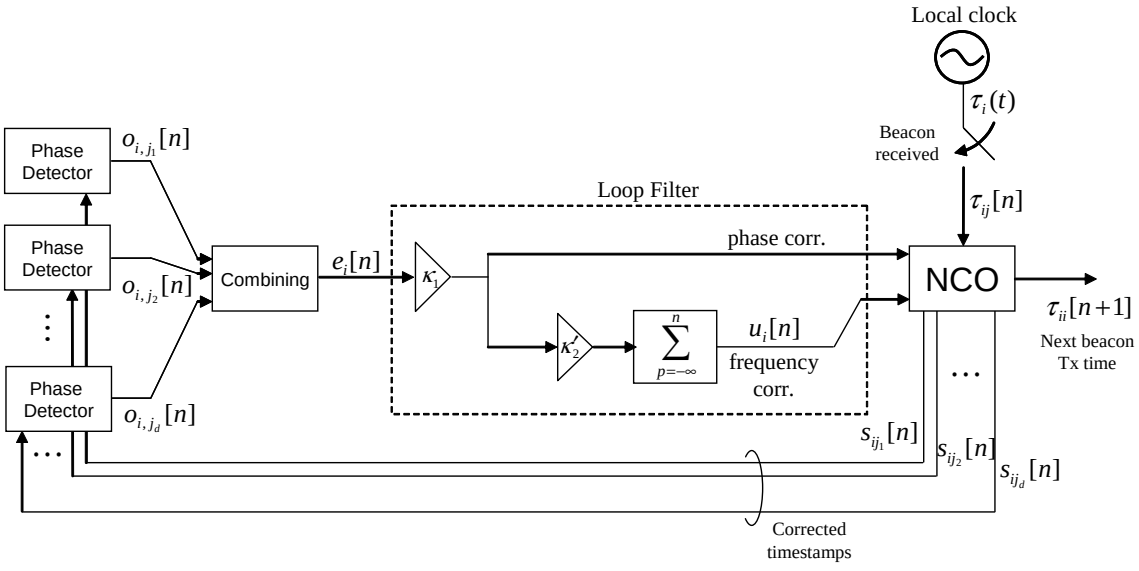


Figure 6.3 Block diagram of the PLL synchronization algorithm. The algorithm is modeled as a discrete-time system operating with sample period T_{SF}.

the overall frequency offset as in the case of TCC. Notably, for a simple master-slave link, the type 2 PLL shares the same structure of the optimal Kalman filter[2] [98]. As detailed in Chapter 5, when the clock is controlled by a type 2 PLL, the *absolute* time interval between two successive packet transmissions reads

$$\begin{aligned} t_i[n] - t_i[n-1] &= \frac{1}{1+\alpha_i[n-1]} \left(T_{SF} - \kappa_1 e_i[n-1] - T_{SF} u_i[n-1]\right) \\ &\simeq \left(1 - \alpha_i[n-1]\right) T_{SF} - \frac{\kappa_1 e_i[n-1] + T_{SF} u_i[n-1]}{1+\alpha_i[n-1]}, \end{aligned} \tag{6.10}$$

where the approximation holds since $\alpha_i[n] \ll 1$ and $e_i[n-1]$ and $u_i[n-1]$ are the phase and frequency corrections, respectively. In practice, node i computes the phase offset o_{ij} with respect to a transmitting neighbor j during the time-slot reserved for the link (i,j). The offset o_{ij} is then employed to compute phase and frequency corrections as in (5.10)-(5.11). In order to simplify the treatment and derive general results, in the following the specific link schedule is neglected, and it is ideally assumed that nodes transmit and

[2]The problem of synchronizing networked *unstable* clocks can be posed in the Kalman filtering framework. Unfortunately, the optimal Kalman filter cannot be implemented in a distributed fashion.

receive simultaneously in a unique time-slot at the beginning of the super-frame. Therefore, $t_i[n]$ identifies the absolute time at which node i transmits its packet at the beginning of the super-frame. Recall that, if node i is a master node, it has access to an accurate and *stable* time reference, and it is $t_i[n] = t_o[n]$ and $t_o[n] = nT_{SF}$. In the following, the tracking accuracy of synchronization algorithms is evaluated by computing the phase MSE at steady-state,

$$\xi = \lim_{n\to\infty} \sqrt{\frac{1}{K}\sum_{i=1}^{K_u} E\left[\left(t_i[n] - t_o[n]\right)^2\right]}, \tag{6.11}$$

where $t_o[n]$ is the network reference time-scale. In the case of MC topologies, it is $t_o[n] = \frac{1}{K}\sum_{i=1}^{K} t_i[n]$. As in Chapter 5, the phase and frequency estimation errors of node i are defined with respect to absolute time as

$$x_i[n] = t_i[n] - nT_{SF} \tag{6.12}$$

$$y_i[n] = -\left(u_i[n] + \alpha_i[n]\right) T_{SF}. \tag{6.13}$$

From (5.10)(6.10)(5.15) the dynamic equation for phase and frequency errors may be written as

$$x_i[n] = x_i[n-1] + \frac{\kappa_1}{d_i} e_i[n-1] + y_i[n-1] \tag{6.14}$$

$$y_i[n] = y_i[n-1] + \kappa_2 \frac{\kappa_1}{d_i} e_i[n-1] - T_{SF}\left(\alpha_i[n] - \alpha_i[n-1]\right). \tag{6.15}$$

The error $e_i[n]$ in (6.14)-(6.15) reads

$$e_i[n] = -\sum_{j=K_u+1}^{K} m_{ij} x_i[n] + \sum_{j=1}^{K_u} a_{ij}\left(x_j[n] - x_i[n]\right), \tag{6.16}$$

where the weights m_{ij}, a_{ij} are subject to design choice. By defining the vectors $\mathbf{x} = [x_1, x_2, \ldots, x_{K_u}]^T$, $\mathbf{y} = [y_1, y_2, \ldots, y_{K_u}]^T$, the equations (6.14)-(6.15) may be cast in vector

form as

$$\begin{bmatrix} \mathbf{x}\,[n] \\ \mathbf{y}\,[n] \end{bmatrix} = \begin{bmatrix} \mathbf{I} - \kappa_1 \mathbf{D}^{-1}\mathbf{L} & \mathbf{I} \\ -\kappa_1\kappa_2 \mathbf{D}^{-1}\mathbf{L} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}\,[n-1] \\ \mathbf{y}\,[n-1] \end{bmatrix} - T_{SF} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\alpha}[n] - \boldsymbol{\alpha}[n-1] \end{bmatrix}. \quad (6.17)$$

Recall the conditions required for the system (6.17) to be stable,

$$\begin{cases} 0 < \kappa_2 < 1 \\ 0 < \kappa_1 < \frac{2}{2-\kappa_2} \\ \mu_i > 0 \qquad i = 2, \ldots, K \\ \mu_1 > 0 \qquad \text{MS and hybrid} \end{cases}, \quad (6.18)$$

where $\mu_i = \lambda_i(\mathbf{L})$. When the system is stable, noise analysis follows the same procedure of Section 5.2.2, which treated the case of stable clocks. In fact, it is possible to write the Lyapunov equation for the state covariance matrix and compute the steady-state covariance for the input covariance (6.6).

In the case of regular MC networks, the node degree is the same for all nodes, $d_i = d$, and the steady-state phase error variance may be computed in closed form. The eigenvalue decomposition of the normalized Laplacian matrix is $d^{-1}\mathbf{L} = \mathbf{Q}\mathbf{T}\mathbf{Q}^H$, whereby $\mathbf{Q}$ is a unitary matrix and $\mathbf{T}$ is a diagonal matrix with entries $[\mathbf{T}]_{ii} = \lambda_i(\mathbf{L})/d$ [85]. By letting $\tilde{\mathbf{x}}[n] = \mathbf{Q}^H\mathbf{x}[n]$, $\tilde{\mathbf{y}}[n] = \mathbf{Q}^H\mathbf{y}[n]$, $\tilde{\boldsymbol{\alpha}}[n] = \mathbf{Q}^H\boldsymbol{\alpha}[n]$, (6.17) can be rewritten as K parallel dynamic systems as the one in Figure 6.4. The update equation of a the i-th component is

$$\begin{bmatrix} \tilde{x}_i\,[n] \\ \tilde{y}_i\,[n] \end{bmatrix} = \begin{bmatrix} 1 - \kappa_1\mu_i & 1 \\ -\kappa_1\kappa_2\mu_i & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_i\,[n-1] \\ \tilde{y}_i\,[n-1] \end{bmatrix} - T_{SF} \begin{bmatrix} 0 \\ \tilde{\alpha}_i[n] - \tilde{\alpha}_i[n-1] \end{bmatrix}. \quad (6.19)$$

By employing the spectral analysis techniques of Section 5.2.3, the overall output error due to observation noise and oscillator instability is

$$\xi^2 = \frac{1}{K}\sum_{i=2}^{K} p_i\left(\kappa_1, \kappa_2, \mu_i\right) \left[\frac{\sigma_w^2}{d\mu_i^2}\left(\kappa_1\mu_i + \kappa_2 - \frac{\kappa_1\mu_i\kappa_2}{2}\left(3 - \kappa_2\right)\right) + \right.$$
$$\left. \frac{\sigma_v^2}{\kappa_1\mu_i} + \sigma_\eta^2\left(\frac{1}{\left(\kappa_1\mu_i\right)^2\kappa_2} - \frac{1-\kappa_2}{2\left(\kappa_1\mu_i\right)\kappa_2}\right)\right]. \quad (6.20)$$

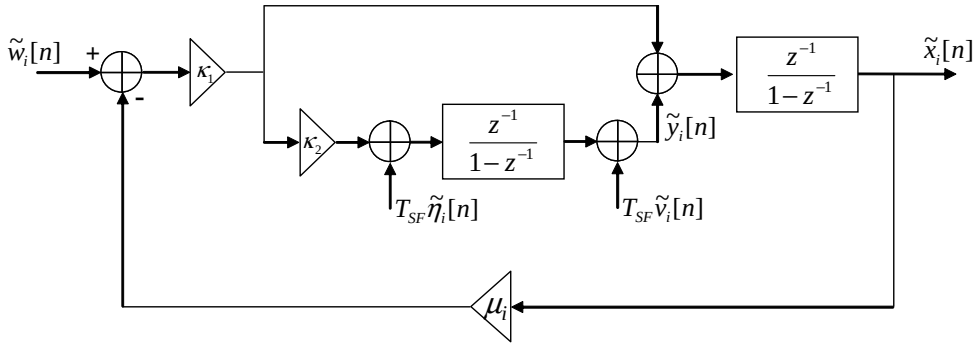


Figure 6.4 Decomposition of a regular MC network of PLL's accounting for all noise sources.

where

$$p_i\left(\kappa_1, \kappa_2, \mu_i\right) = \frac{2}{4 - 4\kappa_2 - \kappa_1\mu_i\left(2 - 3\kappa_2 + \kappa_2^2\right)}. \tag{6.21}$$

Notice that the weight $p_i\left(\kappa_1, \kappa_2, \mu_i\right)$ in (6.21) diverges at the boundary of the stability region. The network MSE (6.20) is therefore a function of loop parameters, network topology and the variances of noise sources. For small loop gain, κ_1, $\kappa_2 < \kappa_1 \ll 1$, the network steady-state phase error is approximated as

$$\xi^2 \simeq \frac{1}{K}\sum_{i=2}^{K}\frac{1}{2\mu_i}\left[\left(\kappa_1 + \frac{\kappa_2}{\mu_i}\right)\frac{\sigma_w^2}{d} + \frac{\sigma_v^2}{\kappa_1} + \frac{\sigma_\eta^2}{\kappa_2\kappa_1\mu_i}\right]. \tag{6.22}$$

From (6.22), it is clear that reducing the gains κ_1, κ_2 is beneficial in order to reduce the channel noise, but it degrades the PLL tracking capabilities. This implies that a large bandwidth is needed in order to filter out the local clock noise, in line with classical PLL theory [6]. It is expected that the optimal parameters will grow in proportion with the super-frame length T_{SF}, since local noise sources become dominant in the low duty-cycle regime. The next section introduces an alternative synchronization algorithm based on tracking frequency dynamics by means of a type 1 frequency locked loop (FLL).

6.4 Phase/Frequency-Locked Loop (P/FLL)

Frequency variations may be tracked by the use of a suitably designed frequency-locked loop (FLL), at the price of increased synchronization overhead. The FLL updates the frequency correction with

$$u_i[n] = u_i[n-1] + \kappa_F e_i^{(f)}[n], \tag{6.23}$$

where the frequency error is a linear combination of pair-wise frequency offsets,

$$e_i^{(f)}[n] = \frac{1}{d_i}\left[\sum_{j=K_u+1}^{K} m_{ij} f_{ij}[n] + \sum_{j=1}^{K_u} a_{ij} f_{ij}[n]\right]. \tag{6.24}$$

Pair-wise frequency offsets are computed by the frequency detector block from the first-order difference of pair-wise phase offsets and phase corrections (see Appendix 6.A for a proof),

$$f_{ij}[n] = (o_{ij}[n] - o_{ij}[n-1]) - \kappa_P \left(e_j[n-1] - e_i[n-1]\right). \tag{6.25}$$

From (6.25), implementing a frequency detector requires each node j to communicate its last phase correction $e_j[n-1]$ to its neighbors[3]. When using a FLL to recover frequency synchronization, a simple type 1 PLL is sufficient to retrieve phase synchronization. The block diagram of the P/FLL algorithm is depicted in Figure 6.5. After the change of variables (6.12)-(6.13), the dynamic equation of the P/FLL can be written compactly as (see Appendix 6.A)

$$\begin{bmatrix} \mathbf{x}[n] \\ \mathbf{y}[n] \end{bmatrix} = \begin{bmatrix} \mathbf{I} - \kappa_P \mathbf{D}^{-1}\mathbf{L} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} - \kappa_F \mathbf{D}^{-1}\mathbf{L} \end{bmatrix} \begin{bmatrix} \mathbf{x}[n-1] \\ \mathbf{y}[n-1] \end{bmatrix} - T_{SF} \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\alpha}[n] - \boldsymbol{\alpha}[n-1] \end{bmatrix}. \tag{6.26}$$

[3]The necessity to communicate $e_j[n-1]$ to the neighbors makes the FLL not implementable with plain pulse-coupling techniques as [43].

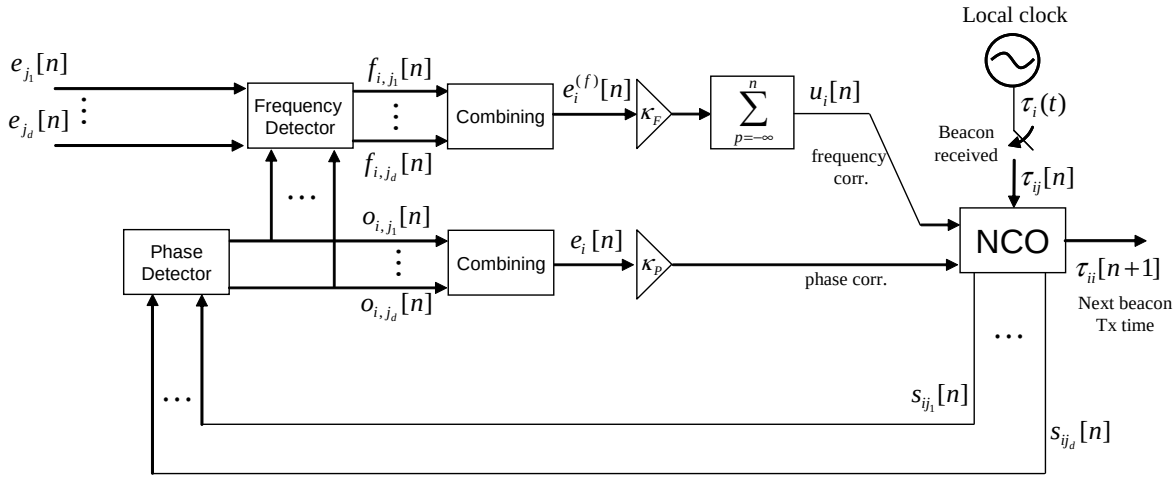


Figure 6.5 Block diagram of the P/FLL.

If the PLL and the FLL are both stable, then the overall update (6.26) is stable. Therefore, sufficient stability conditions read

$$\begin{cases} 0 < \kappa_P < 1 \\ 0 < \kappa_F < 1 \\ \quad \mu_i > 0 \qquad i = 2, \ldots, K \\ \quad \mu_1 > 0 \qquad \text{MS and hybrid} \end{cases} . \tag{6.27}$$

It is worthwhile to emphasize how, differently from the type 2 PLL case, stability conditions of frequency and phase tracking loops are decoupled in the P/FLL. If the system is stable, it is easy to check that converges to $\mathbf{x}[n] = \mathbf{1}\left(\mathbf{v}^T\mathbf{x}[0]\right) + n\mathbf{1}\left(\mathbf{v}^T\mathbf{y}[0]\right)$, where $\mathbf{v}$ is the left eigenvector of the normalized Laplacian matrix, $\mathbf{v}^T\mathbf{D}^{-1}\mathbf{L} = \mathbf{0}$. For general topologies, noise analysis is again carried out by solving the steady-state Lyapunov equation (see Section 5.2.2).

As before, in the case of regular MC networks, the system (6.26) may be diagonalized by eigenvalue decomposition so as to obtain K uncoupled systems as the one in Figure 6.4.

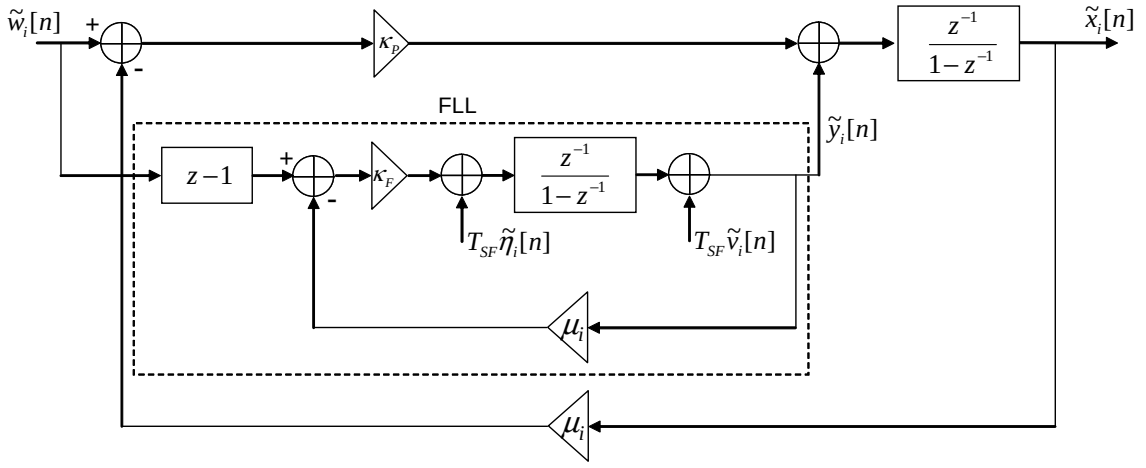


Figure 6.6 Decomposition of a regular MC network of P/FLL's.

The update equation of the i-th system is

$$\begin{bmatrix} \tilde{x}_i[n] \\ \tilde{y}_i[n] \end{bmatrix} = \begin{bmatrix} 1-\kappa_P\mu_i & 1 \\ 0 & 1-\kappa_F\mu_i \end{bmatrix} \begin{bmatrix} \tilde{x}_i[n-1] \\ \tilde{y}_i[n-1] \end{bmatrix} - T_{SF} \begin{bmatrix} 0 \\ \tilde{\alpha}_i[n]-\tilde{\alpha}_i[n-1] \end{bmatrix}. \tag{6.28}$$

It is convenient to introduce the equivalent loop parameters $\gamma = \kappa_P\kappa_F$ and $\rho = \kappa_P + \kappa_F$. It can be shown that the network MSE reads

$$\xi^2 = \frac{1}{K}\sum_{i=2}^{K} p_i(\gamma,\rho,\mu_i)\left[\frac{\sigma_w^2}{d}(\gamma\mu_i(2+\gamma\mu_i)+\rho(2\rho-3\gamma\mu_i))+2\sigma_v^2+\frac{\sigma_\eta^2}{\mu_i^2\gamma}(2-\rho-\gamma\mu_i)\right], \tag{6.29}$$

where

$$p_i(\gamma,\rho,\mu_i) = \frac{1}{\mu_i(\rho-\gamma\mu_i)(4-2\rho+\gamma\mu_i)}. \tag{6.30}$$

As for the type 2 PLL, the weights $p_i(\gamma,\rho,\mu_i)$ tend to infinity close to the boundary of the stability region. For small loop gains, $\kappa_P \ll 1$, $\kappa_F \ll 1$, the network steady-state phase error is approximated as

$$\xi^2 \simeq \frac{1}{K}\sum_{i=2}^{K}\frac{1}{2\mu_i}\left[\left(\rho+\frac{\gamma}{\rho}\right)\frac{\sigma_w^2}{d}+\frac{\sigma_v^2}{\rho}+\frac{\sigma_\eta^2}{\mu_i^2\gamma\rho}\right]. \tag{6.31}$$

Equation (6.31) shows that, as expected, small loop gains reduce channel noise but emphasize the local oscillator noise. The next section deals with the optimization of the network MSE for both the type 2 PLL and P/FLL algorithms.

6.5 Optimization of Loop Parameters

For both the PLL and the P/FLL algorithm, the network MSE ξ^2 of a regular MC network is a function of the loop parameters, network connectivity and noise variances. In particular, it may be shown that ξ^2 tends to infinity at the boundary of the stability region, and it is convex in terms of the loop parameters and coupling coefficients. Convexity implies that the optimization of coupling coefficients a_{ij}, m_{ij}, and loop parameters may be carried out by efficient numerical methods [99]. It is conjectured that these properties hold for all network topologies. Nevertheless, even for the regular MC case, the global optimization over edge weights and loop parameters can be carried out only by a centralized controller which is aware of the complete network topology. Notably, the NTP protocol comprises a distributed gain adaptation algorithm which is based on the *measured* pair-wise synchronization error [17]. The optimality of this heuristic algorithm is questionable, though. From simulation results here omitted, it can be conjectured that it holds for the optimal P/FLL parameters that $\kappa_p = \kappa_F$. This implies that the P/FLL dynamics may be regulated by a single parameter, while two parameters are needed for the type 2 PLL. A suboptimal approach to optimize PLL dynamics (adopted by NTP) is to keep the loop damping factor $\zeta = \sqrt{\kappa_1/4\kappa_2}$ fixed (i.e., to choose $\kappa_2 = \kappa_1/4\zeta^2$) and search for the optimal κ_1. The merits of the optimal and suboptimal approach are delved in the next section, which studies the performance of the PLL and P/FLL algorithms for varying loop parameters.

6.6 Simulation Results

This section deals with the evaluation of the performance achieved by PLL and P/FLL algorithms in terms of network phase root MSE (RMSE) ξ with respect to the super-frame

length T_{SF}. The results for the algorithms of interest are compared with two reference curves, namely the frequency change variance (i.e., the power of the *untracked* clock noise component) and the phase accuracy of a *noiseless* TCC (6.9) with $\alpha_{\max} = 10$ ppm and $\bar{\delta} = 40$ μs. The TCC reference is quite optimistic for two reasons. Firstly, a TCC is indeed affected by period jitter, due to its inability to correct frequency offsets exactly in dynamic conditions. Secondly, TCC's are typically employed with synchronization protocols in the likes of FTSP [37], which are tailored for MS topologies. Since this class of protocols features phase jitter accumulation down the hierarchy, this curve is exact only for a single master-slave link, and it is a lower bound for general multihop networks. The untracked noise constitutes a limit to the performance of the proposed protocols, as it is impossible to predict the next step of the frequency random walk. In particular, it has been chosen $\sigma_\eta = 3.2 \cdot 10^{-9}\sqrt{T_{SF}}$, which corresponds to a RWFM step with a standard deviation of 0.2 ppm for $T_{SF} = 1$ hour[4]. In order to highlight the effects of RWFM, it is assumed that the WFM component of clock noise is not present, $\sigma_v = 0$. The following simulations refer to a line network of $K = 10$ nodes with nearest-neighbor connectivity. The super-frame length varies from 100 ms to 10^5 s, i.e., well over one day, and the channel noise is $\sigma_w = 10$ μs. Figure 6.7 depicts the performance for a MC topology and *constant* loop parameters, $\kappa_1 = 0.1$, $\zeta = 5$, $\kappa_P = \kappa_F = 0.1$. Both proposed algorithms outperform TCC accuracy for small T_{SF}, mainly because of their phase filtering capability. As expected, the performance degrades rapidly with longer super-frames.

The performance of the PLL and P/FLL schemes over the same MC line network, but with adaptive loop gains, is depicted in Figure 6.8. Loop gains are adapted so as to minimize the network MSE in the equivalent *ring* network of 10 nodes. This solution is clearly suboptimal when applied to a line network, but it is expected to perform reasonably close to optimal from the asymptotic equivalence of ring and line topologies (see Section

[4]According to the clock model (6.3), with $k_S = 4{\cdot}10^{-8}$°C^{-2}, $k_D = 0$ and $T_o = 25$°C, a temperature change of ± 1°C from a starting temperature of 20°C roughly corresponds to a frequency change of ± 0.4 ppm.

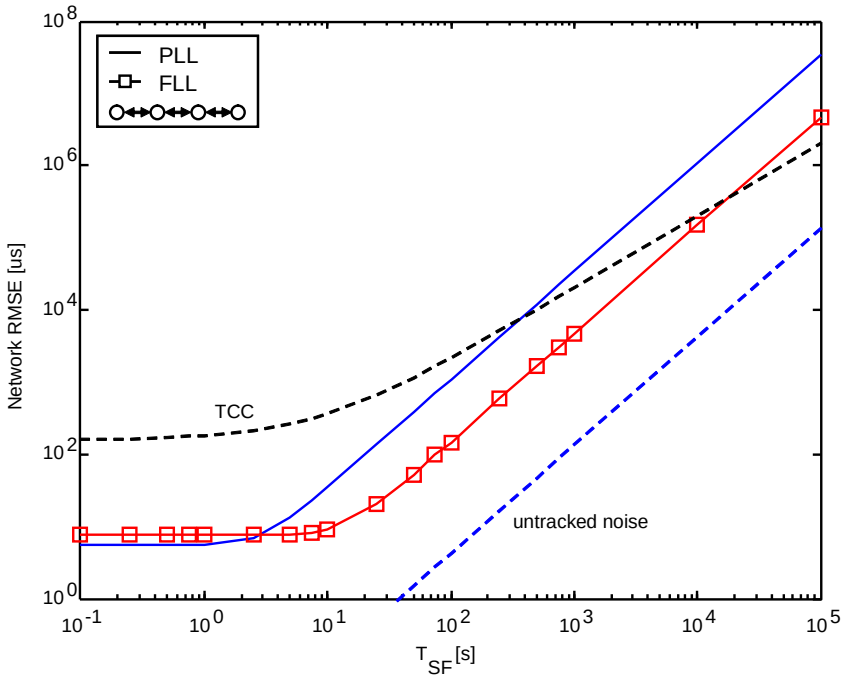


Figure 6.7 Network RMSE for a line network of 10 nodes with MC topology, PLL and P/FLL parameters are kept fixed:$\kappa_1 = 0.1$, $\zeta = 5$, $\kappa_P = \kappa_F = 0.1$.

2.5). Both the PLL and the P/FLL outperform the TCC over the whole range of values for T_{SF}. In particular, their performance is identical when the PLL is optimized over both κ_1 and κ_2, and the damping changes with T_{SF} as $\zeta = \sqrt{\kappa_1/4\kappa_2} = \zeta_{\text{opt}}(T_{SF})$. When keeping the damping factor ζ fixed, $\zeta = 2, 5, 10$, the PLL performance is inevitably degraded, but it is still more accurate with respect to a TCC. The optimal loop parameters as a function of super-frame length T_{SF} are depicted in Figure 6.9. For short super-frames ($T_{SF} < 100$ s), loop gains grow proportionally with T_{SF}. When the untracked noise contribution is dominant ($T_{SF} > 100$ s), the optimal loop parameters are close to the stability boundary, and their values do not change for larger values of T_{SF}. Notice how the optimal PLL damping factor $\zeta_{\text{opt}}(T_{SF})$ is almost constant at small and large T_{SF}, and it is always $\zeta_{\text{opt}}(T_{SF}) > 0.707$.

Figure 6.10 reports the network RMSE for a MS topology and adaptive loop gains. Since there is no analytical expression for the phase error in a general MS network, the gains are here adapted so as to optimize a single master-slave link. Again, both adaptive

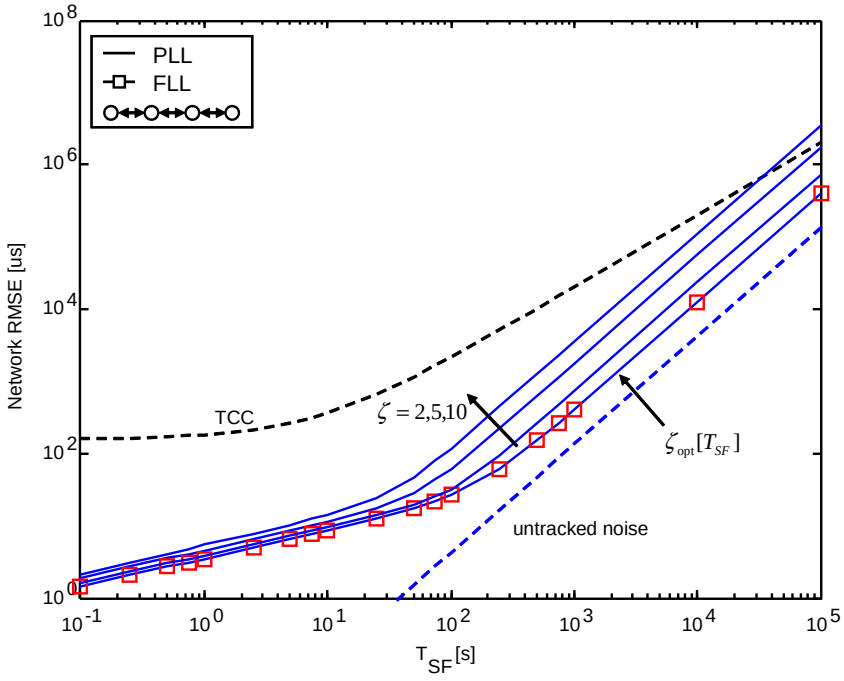


Figure 6.8 Network RMSE for a line network of 10 nodes with MC topology, PLL and P/FLL parameters are optimized.

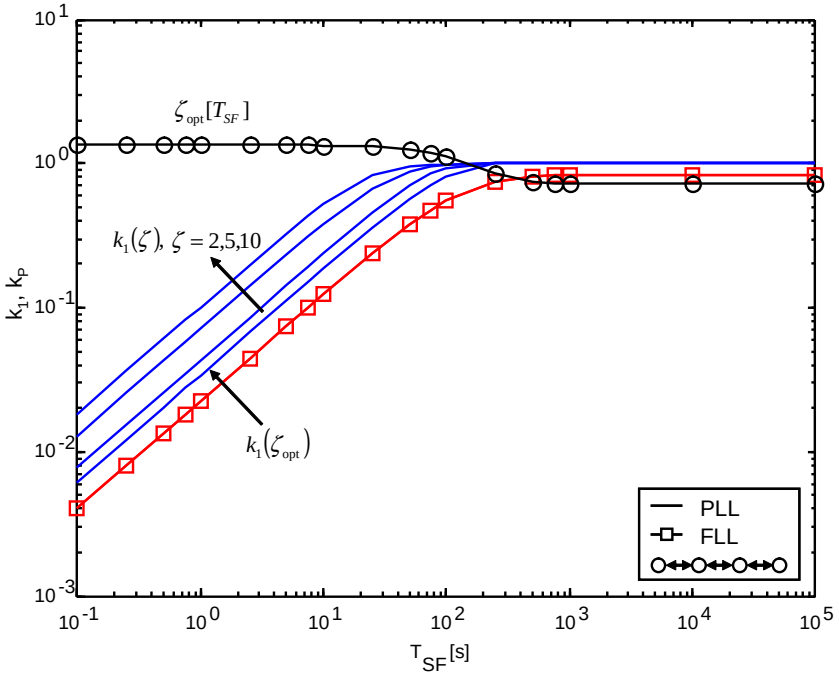


Figure 6.9 Optimal loop parameters for a line network of 10 nodes with MC topology

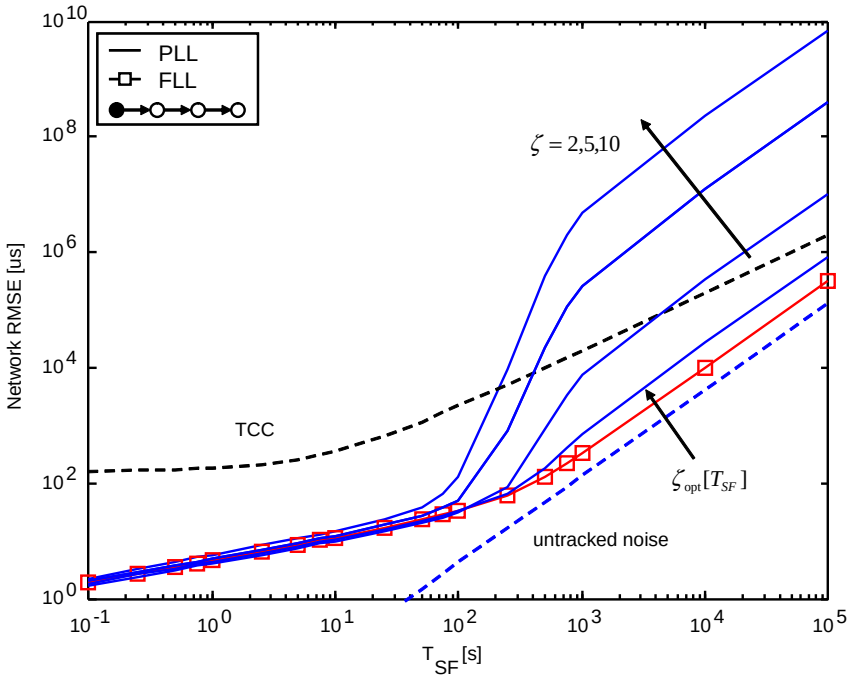


Figure 6.10 Network RMSE for a line network of 10 nodes with MS topology, PLL and P/FLL parameters are optimized.

algorithms outperform TCC, but the P/FLL algorithm performs slightly better than the PLL algorithm. Also, the PLL RMSE is largely degraded when employing a constant damping factor $\zeta > 1$. Finally, the optimal parameter values are plotted in Figure 6.10 with respect to the super-frame length T_{SF}. The behavior of the curves is similar to the MC case in Figure 6.9. Of note, the optimal PLL damping for large T_{SF} is $\zeta_{\text{opt}}\left(T_{SF}\right) = 1$, with loop gain $\kappa_1 > 1$.

6.7 Conclusions

In networks with low duty-cycles, nodes are kept in sleep mode most of the time, and their clocks are subject to relevant frequency changes driven by environmental temperature variations. This chapter has analyzed the capability of adaptive clock control algorithms to track frequency instabilities in a network employing a beacon-less TDMA MAC protocol, where the duty-cycle depends on the super-frame length T_{SF}. In particular, the two algorithms that have been considered are a type 2 PLL with a

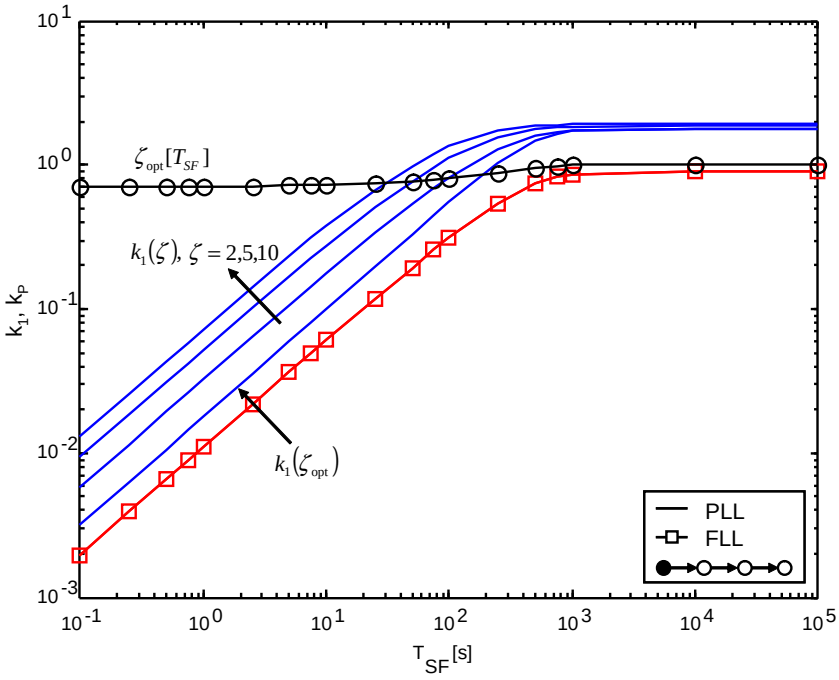


Figure 6.11 Optimal loop parameters for a line network of 10 nodes with MS topology.

proportional-integral controller and a P/FLL equipped with a frequency tracking loop. The performance of adaptive schemes has been checked against a conventional approach based on improving clock accuracy by the compensation of frequency changes due to environmental temperature variations (temperature-compensated clock - TCC). Adaptive designs have shown to be competitive with respect to the employment of TCC's since they effectively track frequency variations at smaller duty-cycles, while they filter out clock noise at larger duty cycles. In particular, at very small duty-cycles ($T_{SF} > 1000$ s), the optimal loop parameters do not depend on the super-frame length T_{SF}.

Appendix 6.A: Design of the Frequency Difference Detector

Recall the phase offset model (5.15) introduced in Section 5.2,

$$o_{ij}[n] = (1 + \alpha_i[n])\,(t_i[n] - t_j[n])\,. \tag{6.32}$$

If the frequency changes slowly with time, $\alpha_i[n] \simeq \alpha_i[n-1]$, the first-order difference of pair-wise phase offsets reads

$$\begin{aligned} o_{ij}[n] - o_{ij}[n-1] &= (1+\alpha_i[n]) \left[(t_i[n] - t_i[n-1]) - (t_j[n] - t_j[n-1]) \right] \\ &\simeq (1+\alpha_i[n])\, T_{SF} \left(\frac{1-u_i[n-1]}{1+\alpha_i[n-1]} - \frac{1-u_j[n-1]}{1+\alpha_j[n-1]} \right) + \\ &\quad + \kappa_P \left(\frac{1+\alpha_i[n]}{1+\alpha_j[n-1]} e_j[n-1] - e_i[n-1] \right), \end{aligned} \tag{6.33}$$

where the update law (6.10) has been employed. The pair-wise frequency difference $f_{ij}[n]$ is computed by the FDD with

$$\begin{aligned} f_{ij}[n] &= (o_{ij}[n] - o_{ij}[n-1]) - \kappa_P \left(e_j[n-1] - e_i[n-1] \right). \\ &= (1+\alpha_i[n])\, T_{SF} \left[\frac{1-u_i[n-1]}{1+\alpha_i[n-1]} - \frac{1-u_j[n-1]}{1+\alpha_j[n-1]} \right] + \kappa_P \left[\frac{\alpha_i[n] - \alpha_j[n-1]}{1+\alpha_j[n-1]} e_j[n-1] \right] \\ &\simeq (1+\alpha_i[n])\, T_{SF} \left[(u_j[n-1] + \alpha_j[n-1]) - (u_i[n-1] + \alpha_i[n-1]) \right], \end{aligned} \tag{6.34}$$

where the last approximation holds since $\alpha_i[n], \alpha_j[n] \ll 1$ for every n. After the change of variables (6.12)-(6.13), it is $f_{ij}[n] = (1+\alpha_i[n])\,(y_i[n-1] - y_j[n-1])$, from which (6.26) is readily obtained.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

Synchronization is a fundamental facility for communication networks. In wireless sensor networks (WSN), in particular, it is instrumental in order to enable cooperation at the physical layer, to coordinate medium access at the link layer, and to schedule sampling and control tasks at the application layer. In the past, synchronization techniques have been specialized for the specific task (carrier frequency, symbol/frame time sync) and protocol layer (physical, link, application) they were intended for. Nevertheless, in most cases practical algorithms are based on the familiar phase locking principles that drive natural synchronization phenomena. With reference to the case of wireless networks, previous works on network synchronization focused either on higher layers (neglecting complexity constraints) or on a specific communication system (Wi-Fi, cellular networks). The aim of this thesis was to develop low-complexity synchronization algorithms targeted at application at the lower layers of the protocol stack of a WSN, namely at the physical and MAC layers. This work considered communication models incorporating typical design aspects of WSN protocols, and developed algorithms based on the classical concept of coupled phase and frequency locked loops (PLL and FLL). PLL and FLL techniques have shown not only to constitute low-complexity solutions, but they also provide enhanced robustness against the impairments of wireless communication, i.e., channel noise, interference and packet collision events. The proposed algorithms proved to be versatile enough to adapt to both peer-to-peer (mutually coupled - MC) and hierarchical (master-slave - MS) networks, thereby providing absolute freedom in the design of the underlying communication architecture. Finally, the capability of PLL and FLL algorithms to track clock frequency instabilities have been checked and shown to outperform current industrial solutions.

From the results presented throughout the thesis, synchronization algorithms based on phase-locking principles appear to be fundamental tools to realize simple, accurate and scalable distributed synchronization in a large wireless network. Several topics of interest for future research are listed below.

- At the *physical* layer, the study focused exclusively on distributed carrier frequency synchronization, realized exploiting frequency-locking principles. Even in a simple point-to-point link, multi-path propagation complicates *symbol time* synchronization, which is typically implemented jointly with channel equalization [3]. Nevertheless, multipath signal distortion is mitigated when employing ultra-wideband (UWB) signaling. UWB modulation allows to discriminate signals received from different propagation paths[1], and to accurately estimate clock time offsets and propagation delays up to the nanosecond scale. Distributed time synchronization in a UWB network requires to be performed jointly with *distributed localization*. This problem is the focus of active industrial and academic research efforts. In fact, UWB modulations have been recently introduced in the first amendment to IEEE 802.15.4 [100] (called IEEE 802.15.4a-2007). At the time of this writing, the first 802.15.4a chips with a real-time location system (RTLS) are being commercialized. A long-standing issue is the feasibility of distributed carrier *phase synchronization* [101]. Phase synchronization is instrumental for the implementation of advanced cooperative functions, such as distributed beamforming. Unfortunately, the practicality of distributed phase synchronization is undermined by the necessity to compensate propagation delays and to accurately track rapid phase variations.

- At the MAC layer, *time* synchronization has been analyzed by simulations employing synthetic models for the impairments due to the communication protocol and various noise sources. The actual tracking capabilities of PLL and FLL techniques should be

[1]The reader can think about impulse radio UWB (IR-UWB) modulations, but the same observations hold true also for other UWB types, e.g., OFDM-UWB modulations.

checked experimentally on real-world hardware employing different communication protocols. Also, the performance of iterative synchronization algorithms is heavily dependent on the specific choice of loop parameters. In order to ease practical implementation, simple heuristics have to be designed in order to choose values for the loop parameters. When employing contention access protocols, transmission probabilities determine the frequency of occurrence of retransmissions and therefore the tracking capabilities of the synchronization network. This problem is particularly relevant in the design and setup of ad-hoc networks based on IEEE 802.11, where signaling information (including sync-related information) is transmitted by using a contention-based access protocol. Also, optimal loop parameters should be chosen in order to accurately balance accuracy versus tracking performance as a function of the synchronization update interval. With respect to this topic, it is worth to point out that interesting heuristics have already been incorporated in the design of NTPv4 [102].

REFERENCES

[1] N. Wiener, *Cybernetics or Control and Communication in the Animal and the Machine*. The MIT Press, 1965.

[2] A. Winfree, "Biological rhythms and the behavior of populations of coupled oscillators," *Journal of theoretical biology*, vol. 16, no. 1, pp. 15–42, 1967.

[3] H. Meyr, M. Moeneclaey, and S. Fechtel, *Digital communication receivers: synchronization, channel estimation, and signal processing*. John Wiley & Sons, Inc. New York, NY, USA, 1997.

[4] Y. Kuramoto, *Chemical oscillations, waves, and turbulence*. Springer, 1984.

[5] R. Adler, "A study of locking phenomena in oscillators," *Proceedings of the IRE*, vol. 34, pp. 351–357, Jun. 1946.

[6] F. M. Gardner, *Phaselock techniques*. Wiley Interscience, 2005.

[7] A. Banai, F. Farzaneh, and S. Ayazian, "Investigation of the locking bandwidth in linear and circular arrays of mutually coupled oscillators, intended for microwave power combining," *IEE Proc. Microwave Antennas and Propagat.*, vol. 152, no. 6, p. 441, 2005.

[8] C. Peskin, *Mathematical aspects of heart physiology*. Courant Institute of Mathematical Sciences, New York University, 1975.

[9] E. Izhikevich, "Weakly pulse-coupled oscillators, FM interactions, synchronization, and oscillatory associative memory," *IEEE Trans. Neural Netw.*, vol. 10, pp. 508–526, Mar. 1999.

[10] W. Lindsey and C. Chie, "A survey of digital phase-locked loops," *Proceedings of the IEEE*, vol. 69, Apr. 1981.

[11] A. Arenas, A. Díaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou, "Synchronization in complex networks," *Physics Reports*, vol. 469, no. 3, pp. 93–153, 2008.

[12] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. Hwang, "Complex networks: Structure and dynamics," *Physics Reports*, vol. 424, no. 4-5, pp. 175–308, 2006.

[13] S. Bregni, *Synchronization of Digital Telecommunications Networks*. Wiley, 2002.

[14] W. Lewandowski and C. Thomas, "GPS time transfer," *Proceedings of the IEEE*, vol. 79, pp. 991–1000, Jul. 1991.

[15] W. C. Lindsey, F. Ghazvinian, W. C. Hagmann, and K. Dessouky, "Network synchronization," *Proc. IEEE*, vol. 73, pp. 1445–1467, Oct. 1985.

[16] E. Harrington, "Synchronization techniques for various switching network topologies," *IEEE Trans. Commun.*, vol. 26, pp. 925–932, Jun. 1978.

[17] D. Mills, *Computer network time synchronization: the network time protocol*. CRC Press, 2006.

[18] *Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std. 1588-2008, 2008.

[19] P. Ferrari, A. Flammini, D. Marioli, and A. Taroni, "IEEE 1588-based synchronization system for a displacement sensor network," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 2, pp. 254 –260, feb. 2008.

[20] E. G. Larsson and P. Stoica, *Space-time block coding for wireless communications*. Cambridge Univeristy Press, 2003.

[21] CC2530 IEEE 802.15.4 System-on-Chip Data Sheet, Texas Instruments Inc., rev. Apr. 2009.

[22] H. Cho, J. Jung, B. Cho, Y. Jin, S. Lee, and Y. Baek, "Precision time synchronization using ieee 1588 for wireless sensor networks," *Computational Science and Engineering, 2009. CSE '09. International Conference on*, vol. 2, pp. 579 –586, aug. 2009.

[23] J. Elson and K. Römer, "Wireless sensor networks: A new regime for time synchronization," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 149–154, 2003.

[24] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11-2007, 2007.

[25] *Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)*, IEEE Std. 802.15.4-2006, 2006.

[26] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[27] *ZigBee Specifications*, ZigBee Alliance Std. version 1.0, 2005.

[28] *High Rate Ultre Wideband (PHY) and (MAC) Standard*, ECMA Std. 368, 2008.

[29] A. Koubâa, A. Cunha, M. Alves, and E. Tovar, "TDBS: a time division beacon scheduling mechanism for ZigBee cluster-tree wireless sensor networks," *Real-Time Systems*, vol. 40, no. 3, pp. 321–354, 2008.

[30] V. Vishnevsky, A. Lyakhov, A. Safonov, S. Mo, and A. Gelman, "Study of Beaconing in Multihop Wireless PAN with Distributed Control," *IEEE transactions on mobile computing*, pp. 113–126, Jan. 2008.

[31] K. Pister and L. Doherty, "TSMP: time synchronized mesh protocol," in *Proceedings of the IASTED International Symposium on Distributed Sensor Networks*, Orlando, FL, Nov. 2008.

[32] *WirelessHART Communication Specification (HART 7.1)*, HART Foundation Std., 2009.

[33] B. Sadler and A. Swami, "Synchronization in sensor networks: an overview," in *Military Communications Conference, 2006. MILCOM 2006*, 2006, pp. 1–6.

[34] B. Sundararaman, U. Buy, and A. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, 2005.

[35] K. Romer, P. Blum, and L. Meier, "Time synchronization and calibration in wireless sensor networks," in *Handbook of Sensor Networks: Algorithms and Architectures*. John Wiley & Sons, 2005, pp. 199–237.

[36] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, p. 149.

[37] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 39–49.

[38] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.

[39] E. Serpedin and Q. Chaudhari, *Synchronization in wireless sensor networks: parameter estimation, performance benchmarks and protocols*. Cambridge University Press, 2009.

[40] A. Hu and S. Servetto, "On the scalability of cooperative time synchronization in pulse-connected networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2725–2748, 2006.

[41] A. Giridhar and P. Kumar, "Distributed clock synchronization over wireless networks: Algorithms and analysis," in *45th IEEE Conference on Decision and Control, 2006*, 2006, pp. 4915–4920.

[42] Y. W. Hong and A. Scaglione, "A scalable synchronization protocol for large scale sensor networks and its applications," *IEEE J. Select. Areas Commun.*, vol. 23, pp. 1085–1099, May 2005.

[43] O. Simeone and U. Spagnolini, "Distributed synchronization for wireless sensor networks with couple discrete-time oscillators," in *Eurasip Journal on Wireless Commun. and Networking*, vol. 2007, Jul. 2007, pp. 3153–3167.

[44] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "A PI consensus controller for networked clocks synchronization," in *Proc. of 17th IFAC World Congress, Seoul (Korea)*, 2008.

[45] R. Carli and S. Zampieri, "Networked clock synchronization based on second order linear consensus algorithms," Center for Control Dynamical Systems and Computation, Univ. of California, Santa Barbara, Tech. Rep., 2010.

[46] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," in *Proc. of the 46th IEEE Conference on Decision and Control*, vol. 2007, Dec. 2007.

[47] C. Rentel and T. Kunz, "A clock-sampling mutual network time-synchronization algorithm for wireless ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 7, pp. 633–646, May 2008.

[48] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, pp. 1520–1533, Sep. 2004.

[49] D. Mitra, "Network synchronization-Analysis of a hybrid of master-slave and mutual synchronization," *IEEE Trans. Commun.*, vol. 28, pp. 1245–1259, Aug. 1980.

[50] C. Meyer, *Matrix analysis and applied linear algebra*. Society for Industrial Mathematics, 2000.

[51] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *Siam Review*, pp. 667–689, 2004.

[52] A. Laub, *Matrix analysis for scientists and engineers*. Society for Industrial Mathematics, 2005.

[53] R. Gray, *Toeplitz and circulant matrices: A review*. Now Publishers, 2006.

[54] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[55] S. Rai, "The spectrum of a random geometric graph is concentrated," *Journal of Theoretical Probability*, vol. 20, no. 2, pp. 119–132, 2007.

[56] P. Parker, P. Mitran, D. Bliss, and V. Tarokh, "On bounds and algorithms for frequency synchronization for collaborative communication systems," *IEEE Trans. on Signal Proc.*, vol. 56, no. 8, pp. 3742–3752, 2008.

[57] N. Varanese, Y. Bar-Ness, and U. Spagnolini, "On the Synchronization Rate of Distributed Medium Access Protocols ," in *Proc. of 44th Annual Conference on Information Sciences and Systems (CISS)*, Princeton, NJ USA, Mar. 2010.

[58] U. Mengali and A. N. D'Andrea, *Synchronization Techniques for Digital Receivers*. Springer, 1997.

[59] K. L. Fernando, V. J. Matthews, and E. B. Clarck, "Mean frequency estimation of narrowband signals," *IEEE Signal Process. Lett.*, vol. 11, pp. 175–178, Feb. 2004.

[60] B. I. Triplett, D. J. Klein, and K. A. Morgansen, "Discrete time Kuramoto models with delay," in *Workshop on Networked Embedded Sensing and Control*, vol. 2005, Oct. 2005.

[61] E. Mallada and A. Tang, "Synchronization of phase-coupled oscillators with arbitrary topology," in *Proc. of IEEE ITA Workshop 2007*, La Jolla, CA USA, Feb. 2010.

[62] H. J. Kushner, *Introduction to stochastic control*. Holt, Rinehart and Winston, 1971.

[63] F. Fagnani and S. Zampieri, "Randomized consensus algorithms over large scale networks," in *Proc. of IEEE ITA Workshop 2007*, La Jolla, CA USA, Jan. 2007.

[64] W. R. Braun, "Short term frequency instability effects in networks of coupled oscillators," *IEEE Trans. Commun.*, vol. COM-28, pp. 1269–1275, Aug. 1980.

[65] L.Xiao, S. Boyd, and S. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, pp. 33–46, Jan. 2007.

[66] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, pp. 215–233, Jan. 2007.

[67] E. Lee and D. Messerschmitt, *Digital communication*. Springer, 1994.

[68] I. Gradshteyn, I. Ryzhik, A. Jeffrey, and D. Zwillinger, *Table of integrals, series and products*. Academic press, 1994.

[69] U. Spagnolini, N. Varanese, O. Simeone, and Y. Bar-Ness, "Distributed digital looked loops for time/frequency locking in packet-based wireless communication," in *Proc. IEEE PIMRC 2008*, Sep. 2008.

[70] S. Borade, L. Zheng, and R. Gallager, "Amplify-and-forward in wireless relay networks: rate, diversity, and network size," *IEEE Trans. Inf. Theory*, vol. 53, pp. 3302–3318, Oct. 2007.

[71] M. Vajapeyam and U. Mitra, "Performance analysis of distributed space-time coded protocols for wireless multi-hop communications," *IEEE Trans. Commun.*, vol. 9, pp. 122–133, Jan. 2010.

[72] K. Miller and M. Rochwarger, "A covariance approach to spectral moment estimation," *IEEE Trans. Inf. Theory*, vol. 5, pp. 588–596, Sep. 1972.

[73] O. Simeone, U. Spagnolini, G. Scutari, and Y. Bar-Ness, "Physical-layer distributed synchronization in wireless networks and applications," *Physical Communication*, pp. 67–83, Mar. 2008.

[74] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Trans. Automat. Contr.*, vol. 50, pp. 169–182, Feb. 2005.

[75] T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Trans. Signal Processing*, vol. 57, no. 7, pp. 2748–2761, 2009.

[76] S. Kar and J. Moura, "Sensor Networks With Random Links: Topology Design for Distributed Consensus," *IEEE Trans. Signal Processing*, vol. 56, no. 7, Part 2, pp. 3315–3326, 2008.

[77] A. Tahbaz-Salehi and A. Jadbabaie, "On consensus over random networks," in *Proceedings of the 44th Annual Allerton Conference on Communication, Control and Computing*, 2006.

[78] P. Denantes, F. Benezit, P. Thiran, and M. Vetterli, "Which Distributed Averaging Algorithm Should I Choose for my Sensor Network?" in *IEEE INFOCOM 2008*, Phoenix, AZ USA, Apr. 2008.

[79] N. Varanese, O. Simeone, Y. Bar-Ness, and U. Spagnolini, "Distributed frequency-locked loops for wireless networks," in *Proc. IEEE ISSSTA 2008*, Bologna, Italy, Aug. 2008.

[80] R. Nelson and L. Kleinrock, "Spatial-TDMA: A collision-free multihop channel access protocol," *IEEE Trans. Commun.*, pp. 934–944, Sep. 1985.

[81] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[82] L. Arnold, *Random dynamical systems*. Springer, 1998.

[83] R. Diestel, *Graph Theory*. Springer-Verlag, 2006.

[84] J. Fax and R. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[85] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, 1990.

[86] L. Ahlfors, *Complex analysis*. McGraw-Hill, 1979.

[87] S. Kay, *Fundamentals of statistical signal processing: estimation theory*. Prentice Hall, 1993.

[88] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation*. Englewood Cliffs, NJ, 1999.

[89] P. Barooah, N. da Silva, and J. Hespanha, "Distributed optimal estimation from relative measurements for localization and time synchronization," in *Distributed Computing in Sensor Systems*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 4026, pp. 266–281.

[90] P. Stoica and T. Marzetta, "Parameter estimation problems with singular information matrices," *IEEE Trans. on Signal Processing*, vol. 49, no. 1, pp. 87–90, 2001.

[91] A. Langville and W. Stewart, "The Kronecker product and stochastic automata networks," *Journal of computational and applied mathematics*, vol. 167, no. 2, pp. 429–447, 2004.

[92] A. Berman and R. Plemmons, *Nonnegative matrices in the mathematical sciences*. Society for Industrial Mathematics, 1994.

[93] Y. Chen and Q. Zhao, "On the lifetime of wireless sensor networks," *IEEE Commun. Lett.*, pp. 976–978, Nov. 2005.

[94] A. Bachir, M. Dohler, T. Watteyne, and K. Leung, "MAC essentials for wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 12, no. 2, pp. 222 –248, 2010.

[95] F. Walls and J.-J. Gagnepain, "Environmental sensitivities of quartz oscillators," *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 39, no. 2, pp. 241–249, mar. 1992.

[96] D. Lee, "Analysis of jitter in phase-locked loops," *IEEE Trans. Circuits Syst. II*, vol. 49, pp. 704–711, Nov. 2002.

[97] K. Pister, Y. Zats, R. Conant, and N. Treuhaft, "Low-powered autonomous radio node with temperature sensor and crystal," U.S. Patent US 2005/0 213 612 A1, Sep. 29, 2005.

[98] A. Patapoutian, "On phase-locked loops and Kalman filters," *IEEE Trans. Commun.*, vol. 47, pp. 670–672, May 1999.

[99] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[100] *Wireless MAC and PHY Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs): Amendment 1: Add Alternate PHY*, IEEE Std. 802.15.4a-2007, 2007.

[101] R. Mudumbai, G. Barriac, and U. Madhow, "On the feasibility of distributed beamforming in wireless networks," *Wireless Communications, IEEE Transactions on*, vol. 6, no. May, pp. 1754 –1763, 2007.

[102] D. Mills, "Network time protocol version 4 reference and implementation guide," University of Delaware, Tech. Rep., June 2006.