A Study of Distributed Pulse-Based Synchronization for Device-to-Device Communication

David Tétreault-La Roche

Electrical and Computer Engineering
McGill University
Montréal

August 2016

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Enginnering.

Abstract

The benefits of device-to-device (D2D) communication have garnered interest in both academic and industrial circles. Time synchronization is a key aspect of D2D schemes, particularly in decentralized networks where no reference time is available. Distributed phase-locked loops (DPLL) is a common synchronization algorithm that is suited for decentralized situations. In this work, we study the implementation of a DPLL algorithm in the context of fifth generation (5G) wireless networks, where we include in our analysis several limitations and practical aspects of D2D communication, such as transmission delays, wideband multipath propagation, and single-carrier frequency demodulation multiple access (SC-FDMA) modulation. We propose practical methods to compensate their effects, and introduce new performance metrics to evaluate the merits of the synchronization algorithm. Through simulations at the physical layer, which capture the effects of analog-digital conversions, we demonstrate that time synchronization in a decentralized setting is possible under the constraints specified by the 3rd Generation Partnership Project (3GPP) for D2D applications.

Sommaire

Les bénéfices de la communication dispositif-à-dispositif (D2D) suscitent beaucoup d'intérêts dans les domaines académiques et industriels. La synchronization temporelle est un aspect clé de la communication D2D, particulièrement dans les réseaux complètement décentralisés où une référence de temps commune n'est pas disponible. L'algorithme de synchronisation nommé boucles à phases asservies distribuées (DPLL) est un algorithme bien adapté aux situations décentralisées. Dans ce mémoire, nous étudions l'implémentation d'un algorithme DPLL dans le contexte de réseaux sans-fil de cinquième génération (5G). Nous incluons dans notre analyse plusieurs contraintes et aspects pratiques de la communication D2D, tels que le délai de transmission, la propagation par trajets multiples, et la modulation à porteuse unique SC-FDMA. De plus, nous proposons des méthodes pour réduire l'impact de ces aspects sur l'algorithme de synchronisation. À travers nos simulations au niveau physique, qui capturent les effects des conversions analogique-numérique nous démontrons que la synchronisation temporelle dans un réseau décentralisé est possible sous les contraintes spécifées par le *3rd Generation Partnership Project (3GPP)* pour les applications D2D.

Acknowledgements

This work was supported by a grant from MITACS and InterDigital Canada under the MITACS Accelerate program.

I would like to thank my internship supervisor Benoit Pelletier from InterDigital, who provided valuable input on this project, and helped me navigate the complex 3GPP technical spefications to find those that applied to this project.

Finally, this thesis would not have been possible without the guidance of my thesis supervisor and co-supervisor, Benoit Champagne and Ioannis Psaromiligkos. Their fair criticism and enlightening technical discussions were instrumental in the realization of this thesis.

TABLE OF CONTENTS

List of Abbreviations

3GPP 3rd Generation Partnership Project

5G fifth-generation

ADC analog-to-digital

BR bias removal

CFO carrier frequency offset

CP cyclic prefix

CS cyclic suffix

D2D device-to-device

DAC digital-to-analog

DC drift compensation

DPLL distributed phase-locked loops

ISI inter-symbol interference

LOS line-of-sight

PAPR peak-to-average power ratio

RF radio frequency

SC-FDMA single-carrier frequency demodulation multiple access

SFO SC-FDMA frequency offset

SIR signal-interference ratio

TO time offset

ZC Zadoff-Chu

Notation table

α_i	Node i's clock skew
α_{ij}	Coupling strength between nodes i and j
$a_{ij,p}$	Coupling strength between nodes i and j for path p
β	Root raised cosine rolloff factor
$\underline{\beta}_j$	Estimated slope of the $\theta_j[\nu]$
$\overline{\beta}$	Average of β_j over all j
B	Number of samples to calculate the stability criterion β_j
$c[m]$	SCFDMA symbol in the time domain
C	Communication ratio for the whole network
C_{ij}	Link quality between nodes i and j
ϵ	Correction factor
f_i	Node i's carrier frequency
Δf_{ij}	CFO between nodes i and j
F_s	Sampling rate
γ	Weighting parameter in the estimator $\hat{q}_j$
$\Gamma_j[\nu]$	Filtered version of $\Delta t_j[\nu]$
h_{ij}	Flat fading channel coefficient between nodes i and j
h_γ	Drift correction filter
i	Node index, generally reserved for transmitting nodes
j	Node index, generally reserved for receiving nodes
κ_{ij}	Discrete equivalent of Δf_{ij}
k	Sampling index
$\Delta\lambda_{ij}$	Normalized CFO between nodes i and j
l_{ij}	Estimated location ZC sequences contained transmitted between nodes i and j
$l_{ij}^{\pm}$	Estimated location l_{ij} for the leading (+) or trailing (-) ZC sequences contained in $s[n]$
$l_{ij,p}^{\pm}$	Estimated $l_{ij}^{\pm}$ specific for the path p
L	Factor between the lengths of the input and output of an SCFDMA modulation
$m_{\pm}$	Offsets of the ZC sequences contained in $s[n]$
M	Number of nodes/devices
ν	Time index
N	Length of ZC sequences
$p(t)$	Shaping pulse
$\rho_{ij,p}$	Amplitude of path p between nodes i and j
$\hat{q}_j$	Estimation of the average reception time of $s[n]$ broadcasted by multiple nodes

$\hat{q}_{j\pm}$	Estimation of the average reception time of the leading (+) or trailing (-) ZC sequences
q_{ij}	Discrete equivalent of $\Delta\theta_{ij}$
$q_{ij,p}$	Transmission delay introduced by path p in multipath propagation
$\overline{q_{P,j}}$	Weighted average of the path delays $q_{ij,p}$
Q	Length of the DC filter
$r_{xy}[l]$	Circular cross-correlation between sequences x and y
$R_{xy}[l]$	Linear cross-correlation between sequences x and y
$s[n]$	Synchronization sequence
$\sigma_{Q,i}[\nu]$	Standard deviation of the last Q correction terms $\Delta t_j[\nu]$
$\sigma_{\max}$	Threshold standard deviation to differentiate between the transient and drifting regime
$\theta_i[\nu]$	Node i's clock phase at index ν
$\overline{\Delta\theta}_j[\nu]$	Weighted average of all the phase offsets between node j and other nodes
$\Delta\theta_{ij}[\nu]$	Phase offset between nodes i and j
$\overline{\tau}_j$	Weighted average of the LOS propagation delays $\tau_{ij,0}$
$\tau_{ij,p}$	Delay of path p between nodes i and j
$\Delta t_j[\nu]$	Clock correction term for node j
$\widehat{\Delta t}_j[\nu]$	Estimation of the clock correction term $\Delta t_j[\nu]$
$t_i(t)$	Node i's physical clock
$t_i[\nu]$	Node i's sampled clock
$t'_{ij}[\nu]$	Reception time of node i's synchronization signal at node j
T_0	Clock period
T_s	Sampling period
u	Root index of a ZC sequence
$x(t)$	Pulse-shaped version of $s[n]$
$x_i(t)$	Synchronization signal broadcasted by node i on the ν^{th}clock tick
$\tilde{x}_i(t)$	Signal broadcasted by node i
$y_{ij}(t)$	ν^{th}synchronization signal received at node j from node i
$y_j[k]$	ν^{th}discrete-time signal received by node j
$w_j[w]$	Discrete thermal noise at node j
$z_\pm$	Leading (+) or trailing (-) ZC sequences in $s[n]$
$\mathbb{N}$	Set of natural numbers
$\mathbb{Z}$	Set of integer numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{C}$	Set of complex numbers

Chapter 1

Introduction

The previous years have seen an explosion in the number of wireless devices connected to mobile networks. The introduction of powerful multimedia applications have dramatically increased the wireless broadband demand per device. Combined, these factors have led to an exponential growth in traffic for wireless communications [1]. Just in Western Europe, cellular traffic is expected to increase seventyfold by 2020, when compared to the 2010 figure [2]. Other predictions are much more aggressive and estimate a thousandfold traffic increase by 2020 in densely populated areas [1].

Current wireless networks make use of a centralized paradigm, where all communications transit through an access point. The introduction of device-to-device (D2D) communication could help offload some traffic from the access point by allowing devices within range to interact directly. This approach also boasts other advantages, such as reduced power consumption and lower infrastructure costs [3]. However, time synchronization between the concerned devices is an essential requirement for D2D communication in a practical setting.

There exists different approaches to acquire synchronization between devices. In this chapter, we first review those approaches and determine which method is ideally suited for decentralized applications, after which we outline the contributions of this thesis to the topic.

1.1 Litterature Survey

Synchronization involves acquiring information about the differences between the clock of the communicating nodes in a network, which can be done via the exchange of synchronization messages. By analyzing these messages, the nodes can estimate the time offset (TO) that exists between their respective clocks.

Synchronization approaches can be divided into two main categories: packet coupling and pulse coupling. The former involves the transmission of timestamps between multiple synchronizing nodes, from which optimal clock parameters can be computed via various probability estimation [4] or optimization [4, 5] techniques.

Certain packet coupling algorithms take advantage of a master-slave relationship between two nodes in a network. These algorithms have a long history in digital communications, and are generally very robust under reasonable conditions [4, 6-8]. By exchanging timestamps with the master node, a slave node can estimate the clock parameters of the master node.

The construction of a master-slave hierarchy in a fully distributed manner is possible with a distributed spanning tree algorithm [6]. Combined with a master-slave synchronization method, the use of a spanning tree allows an arbitrary topology of nodes to achieve synchronization without running into cyclical problems. However, the behaviour of this approach under dynamic topology and large-scale implementations is still unexplored.

Alternative packet coupling methods let go of the master-slave algorithms and rely instead on each node randomly broadcasting their timestamps. This approach is specified in the 802.11 IEEE standard [9] under the form of a channel contention algorithm, where each node has equal probability to win the contention and broadcast its timing message. Recent work using this mechanism has been demonstrated to be robust against fast changing topologies [7].

An important caveat of packet coupling methods is that, when a device is powered on, its clock may be completely desynchronized with the rest of the network, making synchronous communication methods unusable. Since packet coupling requires the transmission of digital information, an asynchronous communication paradigm must be im-

plemented to utilize packet coupling methods as a first-line synchronization scheme. This concurrent implementation is usually avoided due to its hardware complexity.

Contrasting with packet coupling methods, pulse coupling approaches are a physical-layer solution to the synchronization problem, and have been extensively studied in the last decade [6, 10-13]. For pulse coupling methods, every node in a network broadcasts a short signal, or pulse, at regular intervals. By analyzing the arrival time and shape of the received pulses, a node can correct its own clock towards a common average.

In general, packet coupling methods can achieve a high synchronization precision, while pulse coupling are limited in precision by the sampling rate of the device [4]. Packet coupling embeds the timing information in both the arrival time of the packet and the information contained in the packet. Instead, pulse coupling algorithms rely solely on the arrival time of the synchronization signals, and do not require the transmission of any message proper. This avoids spending computational resources to symbol detection, and avoids having to implement asynchronous communication system alongside synchronous systems. Pulse coupling therefore trades precision for reduced complexity, making it ideal as a first-line synchronization method, where coarse synchronization is a sufficient requirement.

Certain pulse coupling algorithms often assign a unique pulse to each node in order to individually track the synchronization of that particular node. This work instead considers a unique pulse approach, where all nodes broadcast the same synchronizing signal. This prevents the receiving nodes from identifying the transmitting node, but removes the constraint of mapping a unique identifier to all nodes in the system, granting better scalability [11].

Distributed pulse coupling algorithms can be further divided into two main algorithms: *integrate and fire,* and *distributed phase-locked loops (DPLL)*. In both approaches, each node in the network periodically broadcasts a synchronizing pulse to its neighbours, but the approaches process differently the received pulses.

The *integrate and fire* approach has each node increment its own clock by a small amount every time a pulse is received; nodes that are late with respect to the average end up being dragged earlier in time [6]. While heavily studied [6, 12, 13], this approach is not well suited

for wireless networks, where reliability is an important requirement. A single corrupted node can easily disrupt the synchronization process by broadcasting pulses at a high rate, triggering many clock updates for its neighbours.

In the DPLL approach, nodes compute the average reception time of all the received pulses. Each node then compares the average reception time to its own pulse broadcast time, and adjust its clock accordingly. Doing so, the node's clock and pulse will move towards the average reception time of the other pulses. A single corrupted node in a DPLL scheme will have a modest impact on the whole process, as broadcasting pulses merely shift the average, instead of triggering a network-wide clock update [10]. This inherent robustness, along with the scalability of pulse coupling approaches, makes DPLL an algorithm of choice for distributed synchronization [10].

In all cases, the synchronization signals are transmitted over the air via radio frequency (RF) carrier waves. However, the transmitting and receiving nodes rarely use precisely the same carrier frequency for modulation and demodulation, due to physical limitations. As a result, a carrier frequency offset (CFO) exists between the nodes. This CFO distorts the transmitted signals, and has a significant effect on the accuracy of modern synchronization algorithms [14, 15]. Specifically, the estimation of the Time Offset (TO) is closely affected by the presence of a CFO.

Ideally, communicating nodes should account for both the TO and CFO between themselves. Recent work on the topic successfully decoupled the estimation of TO and CFO by using identical, well crafted synchronization messages [11, 14, 16]. The combination of DPLL with this TO/CFO decoupling was conceived in [11], leading to a distributed synchronization algorithm robust against CFO. However, [11] was limited to cases where transmission delays and wideband multipath effects are negligible. Other works on DPLL [10, 17] did not analyze in depth the impact of these effects, let alone attempt to compensate for them.

Finally, the current wireless communication standards [18] specify that D2D services should use a single-carrier frequency demodulation multiple access (SC-FDMA) modulation scheme. The interactions between DPLL and SC-FDMA has not been analyzed in previous work.

1.2 Thesis Contributions and Organization

In this thesis, we present a discrete-time software implementation of a DPLL algorithm with TO/CFO decoupling under realistic conditions. In comparison to previous work [10, 11, 17], we include in our analysis several limitations and practical aspects of wireless D2D communication:

- Significant transmission delays between devices;
- Wideband multipath propagation;
- Use of SC-FDMA modulation.

In particular, transmission delays and multipath propagation have a strong negative impact on the quality of the synchronization. We propose modifications to the DPLL algorithm to compensate for both of these effects and introduce new performance metrics to evaluate the merit of those modifications. We further adapt the algorithm to function properly under SC-FDMA modulation. Through our realistic simulations, which include the effects of analog-digital conversions, we demonstrate that time synchronization in a decentralized D2D setting is achievable under the simulation parameters specified by 3rd Generation Partnership Project (3GPP) [18] for future fifth-generation (5G) networks.

The organization of this thesis goes as follows. Chapter II describes the system model of the distributed synchronization problem, while Chapter III develops the background theory for DPLL and TO/CFO decoupling. Chapter IV discusses in detail the impact of various practical limitations on DPLL. Chapter V presents the proposed improvements to the algorithm, along with the details of our software implementation. Finally, Chapter VI introduces new performance metrics, describes the simulation methodology, and discusses the results of our numerical simulations. Chapter VII concludes this paper by summarizing our findings and discuss further avenues of research.

Chapter 2

System Model

In this chapter, we present various core concepts that are critical to understand and apply the DPLL algorithm. We first see a discrete clock model, after which we present the transceiver and receiver models used for multipath radio channels. Finally, we formulate the synchronization problem we are aiming to solve.

2.1 Discrete Clock Model

We consider a network of M nodes indexed with $i \in \{1, 2, ..., M\}$, where a node can be any device with wireless communication capabilities. Individual nodes have limited *a priori* information about the rest of the network. Each node can communicate with the other nodes via a wideband multipath fading radio channel.

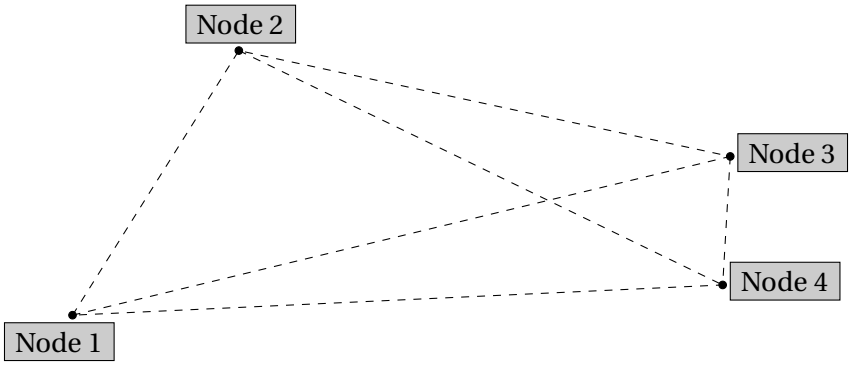

Figure 2.1: Example network of $M = 4$ nodes with their associated links.

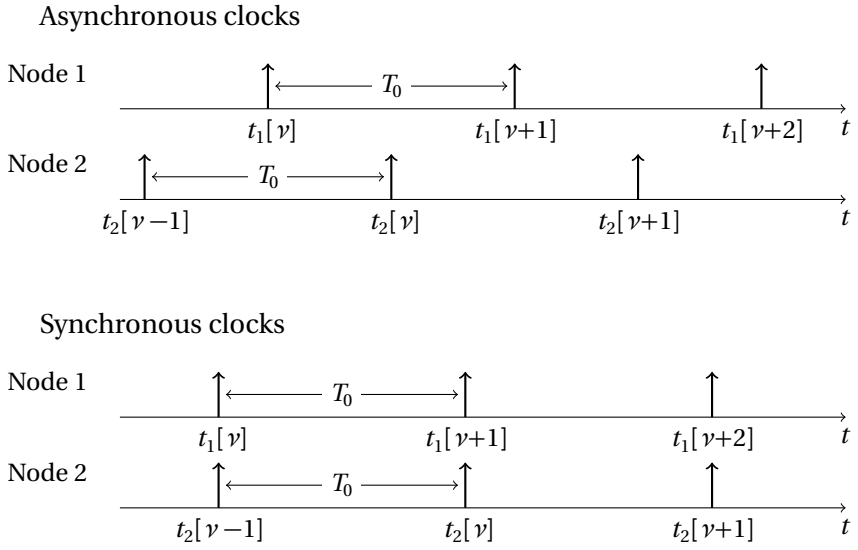

Figure 2.2: Graphical representation of synchronous and asynchronous discrete clocks. Each vertical arrow represent a clock tick at the associated time index.

We assume that each node i has a physical clock t_i, which can be modeled as a continuous linear function of universal time t:

$$t_i(t) = \alpha_i t + \theta_i \tag{2.1}$$

where θ_i and α_i are respectively the clock phase and the clock skew [4]. This thesis focuses on frequency-synchronized clocks, i.e., $\alpha_i = 1$, $\forall i$. Note that the DPLL algorithm can be adapted to frequency-desynchronized clocks reasonably easily, as it has an effect on the algorithm similar to that of a transmission delay [10], which can be compensate for.

For the purpose of synchronization, each node maintains a discrete logical clock $t_i[\nu]$, which is obtained by sampling (2.1):

$$t_i[\nu] = t_i(\nu T_0) = \nu T_0 + \theta_i \tag{2.2}$$

where $\nu \in \mathbb{Z}$ is the tick index and T_0 the clock period.

A graphical representation of a pair of discrete clocks in different synchronicity states is depicted in Fig. 2.2. In the asynchronous case, the clock ticks are obviously misaligned,

while in the synchronous case, the clock ticks are aligned but the tick indices may be different. For the purpose of this paper, we consider clocks i and j to be synchronized if there exists an $l \in \mathbb{Z}$ such that $t_i[\nu] = t_j[\nu + l]$. This condition can be rewritten using the modulo of the clock phases: $(\theta_i)_{T_0} = (\theta_j)_{T_0}$. We can therefore assume without loss of generality that θ_i is constrained within the interval $[0, T_0)$.

2.2 Signal Transmission

We assume that each node broadcasts an identical synchronization sequence $s[n] \in \mathbb{C}$ of length N at each local clock tick. Prior to broadcasting, this sequence is mapped into an analog baseband signal by means of digital-to-analog (DAC) conversion at the rate $F_s = \frac{1}{T_s}$, where T_s denotes the sampling period, and follows the inequality $N T_s \ll T_0$. That is, each symbol $s[n]$ is used to scale a properly shifted replica of a band-limited pulse shape $p(t)$. The analog equivalent of $s[n]$ can be expressed as:

$$x(t) = \sum_{n=0}^{N-1} s[n]\, p(t - n T_s) \tag{2.3}$$

A common choice for $p(t)$ used in this paper is the root raised cosine pulse [19]:

$$p(t) = \begin{cases} \operatorname{sinc}\left(\frac{1}{2\beta}\right), & |t| = \frac{1}{2\beta T_s} \\ \operatorname{sinc}\left(\frac{t}{T_s}\right) \frac{\cos\left(\frac{\pi\beta t}{T_s}\right)}{1 - \frac{4\beta^2 t^2}{T_s^2}}, & \text{otherwise} \end{cases} \tag{2.4}$$

where β is the rolloff factor of the raised cosine, related the to the excess bandwidth.

We can now write the analog baseband synchronization signal transmitted by node i on its ν^{th} clock tick:

$$x_i(t) = x(t - t_i[\nu]) \tag{2.5}$$

Node i transmits carrier wave modulated by its baseband signal. This is done by upconverting (2.5) into the RF spectrum, which is then broadcasted to neighbouring nodes:

$$\widetilde{x}_i(t) = x_i(t)e^{j2\pi f_i t} \tag{2.6}$$

with f_i being node i's carrier frequency.

2.3 Received Signal

We assume that the signal transmission from node i to node j occurs over a wideband multipath channel with impulse response:

$$h_{ij}(t) = \sum_p \rho_{ij,p}\,\delta(t-\tau_{ij,p}) \tag{2.7}$$

where $\rho_{ij,p} \in \mathbb{C}$ and $\tau_{ij,p} > 0$ are the amplitudes and delays introduced by the p^{th} resolvable path. The delay $\tau_{ij,0}$ corresponds a direct path transmission, which may have a corresponding amplitude $\rho_{ij,0} = 0$ if there is no line-of-sight (LOS). For convenience, we also let $\tau_{ij,p+1} > \tau_{ij,p}$ for all p.

Since each node broadcasts $s[n]$ at every local clock tick, a node j receives a synchronization signal $x_i(t)$ from each node $i \neq j$ once per clock period T_0. After reception, the receiving node downconverts the overall signal from the RF spectrum to baseband. The ν^{th} synchronization signal received at node j from node i can be written as $y_{ij}(t)$:

$$\begin{aligned} y_{ij}(t) &= \sum_p \rho_{ij,p}\, x_i(t-\tau_{ij,p})e^{j2\pi f_i(t-\tau_{ij,p})}e^{-j2\pi f_j t} \\ &= \sum_p \rho_{ij,p}\, x(t-\tau_{ij,p}-t_i[\nu])e^{j2\pi\Delta f_{ij} t} \end{aligned} \tag{2.8}$$

where $\Delta f_{ij} = f_i - f_j$ is the CFO between nodes i and j. Note that in (2.8), the complex factor $e^{-j2\pi f_i \tau_{ij,p}}$ is absorbed into $\rho_{ij,p}$ with no loss of generality. The CFO Δf_{ij} can be caused by innate differences between the on-board oscillators of the devices, or external effects such as Doppler shift [14, 15].

Over a period T_0, the received signal at node j after downconversion is the superposition of the broadcasted synchronization signals of all nodes in the network. After an analog-to-digital (ADC) conversion done with rate F_s, which includes band-limited filtering, the ν^{th} discrete-time signal received at node j can be written as:

$$\begin{aligned} y_j[k] &= \sum_{i,i\neq j} y_{ij}(kT_s + t_j[\nu]) + w_j[k] \\ &= \sum_{i,i\neq j}\sum_{p} \rho_{ij,p}\, x(kT_s - \tau_{ij,p} - \Delta\theta_{ij}) e^{j\frac{2\pi}{N}\Delta\lambda_{ij}k} + w_j[k] \end{aligned} \tag{2.9}$$

where $\Delta\theta_{ij} = \theta_i - \theta_j$ is the phase difference between transmitting node i and receiving node j, $w_j[k]$ denotes the antenna noise term after sampling, and $\Delta\lambda_{ij} = N\Delta f_{ij}/F_s$ corresponds to the CFO normalized with the sequence length and sampling rate. Note that the added term $t_j[\nu]$ term in the argument of y_{ij} allows $y_j[k]$ to be expressed in the frame of reference of the local time of node j. The exact bounds on k are not rigid, as long as k runs over the equivalent of T_0 samples. Note that the sample with $k = 0$ corresponds in time to the local clock tick. Using the bounds $-\frac{F_s T_0}{2} \leq k < \frac{F_s T_0}{2}$ allows the local clock tick to be positioned in the centre of the sampling window, where we make the assumption that $F_s T_0 \in \mathbb{Z}$. This reduces edge effects that may arise from further processing of $y_j[k]$.

2.4 Problem Formulation

A synchronization algorithm would allow a node to adjust its local clock phase θ_j after calculating some correction from $y_j[k]$. Accordingly, the clock model should be rewritten as:

$$t_j[\nu] = \nu T_0 + \theta_j[\nu] \tag{2.10}$$

Given the received signal found in (2.9), our goal is to devise an effective synchronization algorithm that would allow all clocks in the network to achieve similar values of $\theta_j[\nu]$, i.e.: $\lim_{\nu\to\infty} \theta_j[\nu] \approx \theta_0$, a constant value, for all nodes j. For the purpose of practical applications in future 5G networks, this algorithm should be robust against the effects of CFO, transmission delay, and multipath propagation. Furthermore, its behaviour through SC-

FDMA modulation and analog-digital conversion should be well understood and numerically verified.

Chapter 3

Background

The DPLL algorithm allows multiple nodes to achieve synchronization by broadcasting a synchronization signal at each local clock tick [10]. However, DPLL requires an accurate estimation of the reception time of the signals. Both the DPLL algorithm and reception time estimation are discussed below. To simplify this discussion, we temporarily assume that the nodes communicate over flat fading channels. Multipath channels are reintroduced in the next chapter.

3.1 Distributed Phase-Locked-Loops

The DPLL algorithm [10] allows nodes in a network to achieve synchronization by updating their clock phases based on the synchronization signals received from their neighbours. To do this update, the algorithm calculates the offset of the local clock with respect to that of the other nodes. Node j computes the weighted average of the time differences between its own clock and the time of reception of the other nodes' synchronization signals, as expressed by:

$$\Delta t_j[\nu] = \sum_{i,i\neq j} \alpha_{ij}\,(t'_{ij}[\nu] - t_j[\nu]) \tag{3.1}$$

where $t'_{ij}[\nu]$ represents the reception time of node i's pulse at node j, and the coupling strength $\alpha_{ij} > 0$ relates how strongly the nodes i and j interact with one another, with the restriction that $\sum_{i,i\neq j} \alpha_{ij} = 1$.

In the absence of transmission delays, which is the underlying assumption in [10, 11, 17], $t'_{ij}[\nu]$ corresponds to the emission time of node i's pulse. In such a case, $t'_{ij}[\nu] = t_i[\nu]$. Therefore, since $\sum_{i,i\neq j} \alpha_{ij} = 1$, the quantity $\Delta t_j[\nu]$ is a weighted average of the phase offsets between the nodes:

$$\Delta t_j[\nu] = \sum_{i,i\neq j} \alpha_{ij} \Delta\theta_{ij}[\nu] \tag{3.2}$$

The quantity $\Delta t_j[\nu]$ is used to schedule the next clock tick:

$$t_j[\nu+1] = t_j[\nu] + T_0 + \epsilon \Delta t_j[\nu] \tag{3.3}$$

which is equivalent to updating the clock phase [10]:

$$\theta_j[\nu+1] = \theta_j[\nu] + \epsilon \Delta t_j[\nu] \tag{3.4}$$

where $0 < \epsilon < 1$ scales the correction: a high ϵ yields a faster convergence time, but a low ϵ gives a better synchronization precision. It can be shown that, using this approach, all clocks will eventually converge to the same phase in the absence of transmission delays [10, 17]. In practice, node j does not have access to θ_j, as it is the offset with respect to universal time, an arbitrary quantity. An implementation of DPLL must therefore use the update rule in (3.3).

In the presence of transmission delays, the arrival time of the synchronization pulse does not correspond to its transmission time: $t'_{ij}[\nu] \neq t_i[\nu]$. This has a significant effect on the evolution of the synchronization: this topic is developed in the next chapter in Sec. 4.3

3.2 Time of Reception Estimation: Two Users

We now turn our attention to the two-user problem, as it is a simpler problem that we use as a building block for the multi-user problem outlined in the next section. Follow-

ing [11, 14], a simplified model is used here for the transmission/reception of the synchronization signals. Specifically, instead of considering the transmission of the analog signal $x(t)$, we consider an equivalent discrete-time model that involves the transmission of a discrete sequence $s[n]$. We assume here that the discrete index n corresponds to a length of continuous time T_s.

We define $\tilde{s}_{ij}[n]$ as the sequence received by node j from node i when transmitted over a flat fading channel $h_{ij} \in \mathbb{C}$, in the absence of transmission delay. The equation that describes $\tilde{s}_{ij}[n]$ shares many similarities with (2.8):

$$\tilde{s}_{ij}[n] = h_{ij}\, s[n - q_{ij}] e^{j\frac{2\pi}{N}\kappa_{ij} n} \tag{3.5}$$

where $q_{ij}, \kappa_{ij} \in \mathbb{Z}$ are the discrete phase offset and the discrete carrier frequency offset between nodes j and i, respectively [11]. They are the discrete homologues of $\Delta\theta_{ij}[\nu]$ and $\Delta\lambda_{ij}$. In this case, q_{ij} can be directly related to the continuous phase offset via the following equation:

$$q_{ij} \approx \frac{1}{T_s}\Delta\theta_{ij} \tag{3.6}$$

We are interesting in estimating q_{ij} from the received signal $\tilde{s}_{ij}[n]$.

A common choice for $s[n]$ are the Zadoff-Chu (ZC) sequences, due to their attractive cyclical correlation properties. The odd-length ZC sequence with non-zero integer root index u is defined as [20]:

$$z_u[n] = e^{-j\pi u n(n+1)/N} \tag{3.7}$$

where the odd natural number N is the sequence period and $n \in [0, N)$ is the symbol index.

We define the cyclical cross-correlation at lag l between two arbitrary complex sequences $x[n]$ and $y[n]$ as:

$$r_{xy}[l] = \sum_{n=0}^{N-1} y[n] x^*[(n+l)_N] \tag{3.8}$$

Two important features of the ZC sequences will be used in this text. The first one is their perfect cyclical autocorrelation ZCs [20], which can be obtained by cross-correlating $z[n]$ with itself:

$$r_{zz}[l] = N\delta[l] \tag{3.9}$$

The other property of ZC sequences [20] is that sequences with different root indices u and v weakly interact in a cross-correlation, when compared to the autocorrelation (3.9):

$$|r_{z_u z_v}[l]| = \sqrt{N}\,,\ \forall l \tag{3.10}$$

where u and v are relative primes of one another.

Let $\tilde{z}_{u,ij}[n]$ be a ZC sequence that underwent transmission from node i to j, which is obtained from $z[n]$ in a similar way as (3.5). The cross-correlation between $\tilde{z}_{u,ij}[n]$ and $z_u[n]$ can be used to estimate q_{ij}. The maximum magnitude $|h_{ij}|N$ of this correlation occurs at the lag:

$$\ell_{ij} \triangleq \arg\max_l |r_{\tilde{z}z}[l]| = q_{ij} - \frac{\kappa_{ij}}{u} \tag{3.11}$$

where ℓ_{ij} is the lag of the peak of the cross correlation. This equation stems from the properties of ZC sequences [16, 20]; a full development can be found in Appendix A.

However, ℓ_{ij} is not a perfect estimation of q_{ij}, due to its dependency on the CFO κ_{ij}. An approach to estimate q_{ij} in the presence of CFO was introduced in [14], and consists of building a synchronization sequence by concatenating two ZC sequences with opposite u parameter:

$$s[n] = \begin{cases} z_{-u}[n+N] & -N \le n < 0 \\ z_u[n] & 0 \le n < N \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

For the remainder of this discussion, we will use z_- and z_+ as shorthands for z_{-u} and z_u, respectively.

Before we continue, it should be mentioned that a major caveat to this approach is that the properties of ZC sequences are defined through circular correlations, but in practical applications, only linear correlations are usable. The linear cross-correlation is defined for bounded sequences $x[n]$ and $y[n]$ as:

$$R_{xy}[l] = \sum_{n\in\mathbb{Z}} y[n]x^*[n+l] \tag{3.13}$$

In our case, the linear correlation of $s[n]$ with any ZC sequence does not correspond to a circular correlation. This point is not raised in [11]. Nevertheless, we noticed through numerical methods that the properties of ZC sequences carry well enough when using a non-circular cross correlations; we further expand on the topic in Sec. 4.1. For the remainder of this chapter, we use the linear cross-correlation $R_{xy}[l]$ and assume that the properties of ZC sequences carry over to linear correlations.

The magnitude of $R_{\tilde{s}z_\pm}[l]$, the cross-correlation of $\tilde{s}_{ij}[n]$ with either z_-, z_+, yields a clearly defined peak at ℓ^-_{ij}, ℓ^+_{ij} respectively:

$$\ell^\pm_{ij} = q_{ij} \mp \frac{\kappa_{ij}}{u} + m_\pm \tag{3.14}$$

where $m_+ = \frac{N-1}{2}$ and $m_- = -\frac{N-1}{2}-1$ are the offsets of $z_\pm$ with respect to the center of $s[n]$. Such peaks are clearly identifiable from the rest of the correlation, due to the weak interaction between z_+ and z_-, as shown in (3.10).

Taking the average between ℓ^-_{ij} and ℓ^+_{ij} yields a biased estimate of q_{ij}:

$$\begin{aligned}\frac{\ell^-_{ij}+\ell^+_{ij}}{2} &= \frac{1}{2}\Big(\big(q_{ij}+\frac{\kappa_{ij}}{u}+m_-\big)+\big(q_{ij}-\frac{\kappa_{ij}}{u}+m_+\big)\Big)\\ &= q_{ij}-\frac{1}{2}\end{aligned} \tag{3.15}$$

This method allows the accurate estimation of the reception time q_{ij} even in the presence of CFO.

3.3 Time of Reception Estimation: Multi-User

In a multi-user setting, node j receives all the transmitted synchronization signals $\tilde{s}_{ij}[n]$ from the neighbouring nodes i:

$$y_j[n] = \sum_{i,i\neq j} h_{ij}\, s[n-q_{ij}]\, e^{j\frac{2\pi}{N}\kappa_{ij}n} + w_j[n] \tag{3.16}$$

Cross-correlating this received signal with z_+ and z_- yields multiple peaks, one for each node i that transmitted a synchronization signal. These peaks are located at lags $\ell_{ij}^{\pm}$. We can obtain an estimate $q_{j\pm}$, $\hat{q}_{j\pm}$, as the weighted average of the lags l of the cross-correlation:

$$\hat{q}_{j\pm} \triangleq \frac{\sum_l l\, |R_{y_j z_\pm}[l]|^\gamma}{\sum_l |R_{y_j z_\pm}[l]|^\gamma} \tag{3.17}$$

where γ is a weighting parameter. Since the terms that correspond to the peaks located at $\ell_{ij}^{\pm}$ dominate the rest of the terms, we can approximate (3.17) by considering only those terms. This simplifies $\hat{q}_{j\pm}$ to the weighted average of the peaks generated by each transmitting node i [11]:

$$\begin{aligned}
\hat{q}_{j\pm} &\approx \frac{\sum_l l\, |h_{ij} N\, \delta[l-\ell_{ij}^{\pm}]|^\gamma}{\sum_l |h_{ij} N\, \delta[l-\ell_{ij}^{\pm}]|^\gamma} \\
&= \frac{\sum_{i,i\neq j} \ell_{ij}^{\pm}\, h_{ij}^{\gamma}}{\sum_{i,i\neq j} h_{ij}^{\gamma}} \\
&= \sum_{i,i\neq j} \alpha_{ij}\Big(q_{ij} \mp \frac{\kappa_{ij}}{u} + m_\pm\Big)
\end{aligned} \tag{3.18}$$

where $\alpha_{ij} = \frac{|h_{ij}|^\gamma}{\sum_{i,i\neq j}|h_{ij}|^\gamma}$ are the weights of this average. Similar to (3.15), by taking a biased average between the two estimations $\hat{q}_{j^-}$ and $\hat{q}_{j+}$, it is possible to recover a unbiased weighted average of all the q_{ij}:

$$\hat{q}_j = \frac{1}{2}(\hat{q}_{i^-} + \hat{q}_{i+}) + \frac{1}{2} \approx \sum_{i,i\neq j} \alpha_{ij} q_{ij} \tag{3.19}$$

An estimation for the continuous variable $\Delta t_j[\nu]$ can be obtained from the discrete estimate $\hat{q}_j$ by using (3.6):

$$\widehat{\Delta t}_j[\nu] = T_s \hat{q}_j \tag{3.20}$$

Applying the clock-update equation found in (3.4) with this estimation of $\Delta t_j[\nu]$ leads to converging clock phases even in the presence of strong CFO [11]. A big advantage of this approach is its scalability: it does not increase in complexity as the number of nodes increases, since computing (3.17) does not depend on M.

Chapter 4

Practical Issues

The previously described DPLL algorithm works well for dense networks [10, 11], in the special case where transmission delays and wideband multipath propagation are absent. However, the use of DPLL in the presence of delays and multipath can prevent nodes from achieving time synchronization. In this chapter, before addressing those aspects, we first investigate if linear correlations and analog-digital conversion affect the performance of the synchronization algorithm.

4.1 Linear vs Circular Cross-Correlations

As discussed in Sec. 3.2, "the time of reception" estimation specified properties of ZC sequences, which in theory hold only for circular correlation, while the correlation used in practice would be linear. We noticed through numerical studies that the properties of ZCs apply reasonably well with linear correlation.

Through numerical methods, we observed that the equation for ℓ_{ij} found in (3.11) holds well for $\kappa < N/4$ when using linear correlations, which is well below the CFO encountered in practice [14]. Furthermore, compared to circular correlations, linear correlations of ZCs with different root indices presented in (3.10) exhibit amplitude oscillations, but maintain the upper bound of $\sqrt{N}$. An typical example of the linear cross correlation between $s[n]$ and $z_+[n]$ can be found in Fig. 4.1. In all cases, the cross-correlation will have a similar shape to the one pictured. The use of linear correlation introduces artifacts, but the overall

shape of the cross-correlation is very clear: the autocorrelation peak at $l = \ell_{ij}^{+}$ is clearly identifiable from the remaining values of the $|R_{\tilde{s}_{ij}z_+}[l]|$. Due to that, the approximation used in Sec. 3.2, where we only considered the dominant peaks, is still reasonable for linear correlations. This is especially true for larger values of N and γ.

Note that the graph presented in Fig. 4.1 is representative of the overall shape $|R_{\tilde{s}_{ij}z_+}[l]|$ will have with varying values for N, γ, or κ_{ij}

The artifacts could still have a negative effect on the synchronization algorithm. The cross-correlation bias-removal method discussed in section Sec. 5.1 attempts to lessen this effect.

4.2 Analog-digital Conversions

As described in Chapter 2, node i passes the sequence $s[n]$ through a DAC to obtain the signal $x(t)$, which is then broadcasted. A receiver first samples the received analog sig-

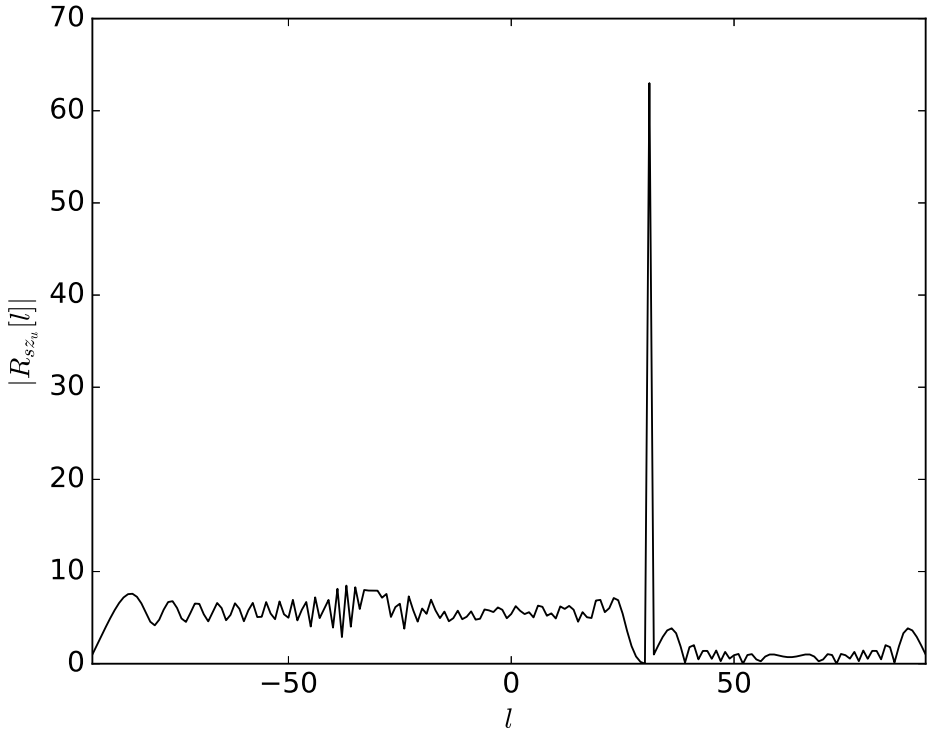

Figure 4.1: Example cross-correlation of $s[n]$ with $z_u[n]$, with $N = 63$. The main peak is located at $l = \pm 31$ and has a height of N

nal into a discrete sequence. Ideally, this sampling should be synchronized with the main peak of the shaping pulse, but in the presence of a timing offset, the sampling may be misaligned. This leads to low signal-interference ratio (SIR), due to signal loss and increased inter-symbol interference (ISI) [19].

In the presence of ISI, the output of an ideal ADC is a linear combination of $s[n]$ with time-shifted copies of itself [19], i.e. a smeared version of $s[n]$. Note that we are not concerned with $s[n]$ itself but instead with its cross-correlation with $z_{\pm}[n]$. Since cross-correlation is a linear operator, the smearing of the input transfers to the output: misaligned sampling causes some broadening of the peaks in $|R_{sz\pm}[l]|$. However, the the overall algorithm is largely unaffected by this effect due to the weighted-average nature of the time of reception estimation.

We numerically tested that the algorithm described in the previous chapter can be used with discrete time signals sampled from the received analog signals, with minor negative impact on the quality of the synchronization.

4.3 Effect of Transmission Delay

Recall that $\tau_{ij,0}$ is the transmission delay between nodes i and j. Equation (3.2) is no longer accurate, as the reception time of the synchronization signal does not correspond to its transmission time anymore. In this case, we have $t'_{ij}[\nu] = t_i[\nu] + \tau_{ij,0}$. Equation (3.2) becomes [10]:

$$\begin{aligned}
\Delta t_j[\nu] &= \sum_{i,i\neq j} \alpha_{ij}\left(t_i[\nu] + \tau_{ij,0} - t_j[\nu]\right) \\
&= \underbrace{\sum_{i,i\neq j} \alpha_{ij}\Delta\theta_{ij}[\nu]}_{} + \underbrace{\sum_{i,i\neq j} \alpha_{ij}\tau_{ij,0}}_{} \\
&= \qquad \overline{\Delta\theta_j}[\nu] \quad + \qquad \overline{\tau}_j
\end{aligned} \tag{4.1}$$

Here, $\overline{\tau}_j$ depends only on the coupling strength α_{ij} and the topology of the network, which we both assume to be static in time.

When we apply the DPLL algorithm in the presence of transmission delays, two regimes can be identified: the transient regime ($\overline{\tau}_j \ll \overline{\Delta\theta_j}[\nu]$) where the algorithm progresses as if there is no transmission, and the drift regime ($\overline{\tau}_j \approx \overline{\Delta\theta_j}[\nu]$), where the phases of the nodes constantly drift forward in time. These regimes are illustrated in Fig. 4.2.

It can clearly be seen from Fig. 4.2 that the phases of the clocks not only keep increasing, but do so at different rates. Over time, the phases keep moving away from one another, and thus the synchronization is not maintained. In the next chapter, we discuss a method to compensate for this drift.

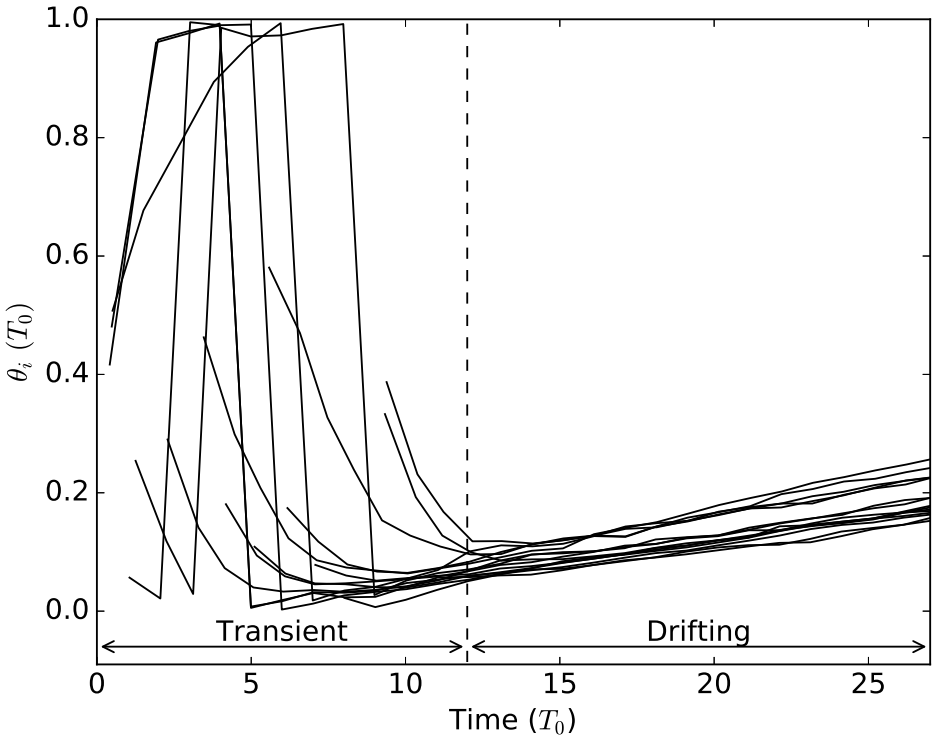

Figure 4.2: Evolution of the phases $\theta_j[\nu]$ of multiple interacting nodes j. Each node applies the DPLL algorithm described in Sec. 3.1. The two main regimes are roughly highlighted. See Chapter 6 for simulation setup.

4.4 Effect of Wideband Multipath Propagation

Channels used in practical applications are not flat fading. Instead, consider the discrete multipath channel response:

$$h_{ij}[n] = \sum_p \rho_{ij,p}\, \delta[n - q_{ij,p}] \tag{4.2}$$

where $\rho_{ij,p} \in \mathbb{C}$ and $q_{ij,p} \in \mathbb{Z}$ are the amplitude and delay introduced by the p^{th} path between nodes j and i, and $\delta[n]$ corresponds to the Kronecker delta function.

The estimation method proposed in (3.17) approximated the cross-correlation $R_{y_j z_\pm}[l]$ by only considering the peaks as significant. In the presence of multipath propagation, a single peak from node i contributes P peaks in $R_{y_j z_\pm}[l]$. We denote those peaks' location in the cross-correlation as $\ell^\pm_{ij,p}$, which are related to the delay-less lag $\ell^\pm_{ij}$ defined in (3.14) with the following equation:

$$\ell^\pm_{ij,p} = \ell^\pm_{ij} + q_{ij,p} \tag{4.3}$$

Akin to (3.17), the weighted average of the cross-correlation can be approximated by only considering the peaks of the correlation:

$$\begin{aligned}
\hat{q}_{j\pm} &= \frac{\sum_l l\, |R_{y_j z_\pm}[l]|^\gamma}{\sum_l |R_{y_j z_\pm}[l]|^\gamma} \\
&\approx \frac{\sum_l l\, |\rho_{ij,p} N\, \delta[l - \ell^\pm_{ij,p}]|^\gamma}{\sum_l |\rho_{ij,p} N\, \delta[l - \ell^\pm_{ij,p}]|^\gamma} \\
&= \frac{\sum\limits_{p,i,i\neq j} (\ell^\pm_{ij} + q_{ij,p}) |\rho_{ij,p}|^\gamma}{\sum\limits_{p,i,i\neq j} |\rho_{ij,p}|^\gamma} \\
&= \sum_{\substack{i \\ i\neq j}} \alpha_{ij} \Big(q_{ij} \mp \frac{\kappa_{ij}}{u} + m_\pm \Big) + \sum_{\substack{p,i \\ i\neq j}} a_{ij,p}\, q_{ij,p}
\end{aligned} \tag{4.4}$$

where $\alpha_{ij} = \frac{\sum_p |\rho_{ij,p}|^\gamma}{\sum\limits_{\substack{p,i \\ i\neq j}} |\rho_{ij,p}|^\gamma}$ and $a_{ij,p} = \frac{|\rho_{ij,p}|^\gamma}{\sum\limits_{\substack{p,i \\ i\neq j}} |\rho_{ij,p}|^\gamma}$ can be understood as pair-specific and path-specific weights.

Note that the second term in (4.4) depends on the paths and topology of the network, which we assume to be static in time. Taking a biased average of $\hat{q}_{i+}$ and $\hat{q}_{i-}$ yields:

$$\begin{aligned}\hat{q}_j &= \frac{1}{2}(\hat{q}_{i+} + \hat{q}_{i-}) + \frac{1}{2} \\ &\approx \sum_{i,i\neq j} \alpha_{ij}\, q_{ij} + \underbrace{\sum_{\substack{p,i \\ i\neq j}} a_{ij,p}\, q_{ij,p}}_{\overline{q_{P,j}}} \\ &= \sum_{i,i\neq j} \alpha_{ij}\, q_{ij} + \qquad \overline{q_{P,j}}\end{aligned} \tag{4.5}$$

This is similar to the result in (3.17), with an additional bias term $\overline{q_{P,j}}$. The continuous estimation $\widehat{\Delta t}_j[\nu]$ defined in (3.20) now becomes:

$$\begin{aligned}\widehat{\Delta t}_j[\nu] &= T_s \hat{q}_j \\ &\approx \sum_{i,i\neq j} \alpha_{ij}\, \Delta\theta_{ij} + T_s \overline{q_{P,j}} \\ &= \overline{\Delta\theta}_j + T_s \overline{q_{P,j}}\end{aligned} \tag{4.6}$$

Since $\overline{q_{P,j}}$ is static in time, this equation has the same form as (4.1). We conclude that the impact of multipath propagation is no different from that of simple transmission delay: it should create a drift in the phases of the clocks. Therefore, the effects of multipath propagation should be properly corrected by the drift compensation algorithm described in the next chapter.

Chapter 5

Algorithm Implementation

Some of the aspects presented in the previous chapter have an important impact on the DPLL algorithm. We present here two additions to DPLL: the cross-correlation bias removal method and the drift Compensation algorithm. The former removes the artifacts discussed in Sec. 4.1, while the latter compensates for both the transmission delay and multipath propagation that exists between the synchronizing nodes. In addition, we explore an adaptation of the algorithm to SC-FDMA modulation. Finally, we discuss in details the inner workings of our software implementation.

5.1 Cross-Correlation Bias Removal

The artifacts located at the bottom of Fig. 4.1 stem from the imperfect correlation properties of ZC sequences in linear correlation. Removing said artifacts from $R_{y_j z_\pm}[l]$ may improve the convergence of the algorithm, which can be done by clipping to zero the values below some threshold. This clipped cross-correlation $|\tilde{R}_{y_j z_\pm}[l]|$ can be written as:

$$|\tilde{R}_{y_j z_\pm}[l]| = \begin{cases} |R_{y_j z_\pm}[l]| & |R_{y_j z_\pm}[l]| > \sqrt{A_{\max}} \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

where $A_{max} = \max_l(|R_{y_j z_\pm}[l]|)$. This approach is motivated from the observation that artifacts are proportional to $\sqrt{N}$, while the peaks are proportional to N. In practice, the

threshold should be set to a value slightly larger than $\sqrt{A_{\max}}$, due to the oscillation seen at the edges of the processing artifacts. This threshold still scales proportionally with $\sqrt{N}$.

5.2 Drift Compensation

The drift regime is due to the bias term $\overline{\tau}_j$ found in (4.1), which would be zero in the absence of transmission delays. Averaging multiple consecutive values of $\Delta t_j[\nu]$ yields a rough estimate of $\overline{\tau}_j$. To compensate for the drift, let us define a filtered correction $\Delta\Gamma_j[\nu]$:

$$\begin{aligned}\Delta\Gamma_j[\nu] &= \Delta t_j[\nu] - \frac{1}{Q}\sum_{k=\nu-Q}^{\nu} \Delta t_j[k] \\ &= \Delta t_j[\nu] - \overline{\tau}_j - \frac{1}{Q}\sum_{k=\nu-Q}^{\nu} \overline{\Delta\theta}_j[k]\end{aligned} \tag{5.2}$$

which is a FIR filter of length Q with impulse response:

$$h_\Gamma[n] = \begin{cases} \frac{Q-1}{Q} & n = 0 \\ \frac{-1}{Q} & 0 < n < Q \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

Essentially, this filter estimates the bias from the last Q values of $\Delta t_j[\nu]$, and removes it from the current value.

Clearly, (5.2) is far from ideal, as it only accurately removes the bias $\overline{\tau}_j$ if $\overline{\Delta\theta}_j[\nu]$ vanishes after averaging, which is rarely the case in either regime. In particular, if this filter is applied during the transient regime, it fails to properly correct for $\overline{\tau}_j$ since $\overline{\tau}_j \ll \overline{\Delta\theta_j}[\nu]$: the contribution of $\overline{\Delta\theta}_j[\nu]$ dwarfs the contribution from $\overline{\tau}_j$. This compensation should only be employed during the drift regime, where $\overline{\tau}_j \approx \overline{\Delta\theta_j}[\nu]$. Therefore, the current regime must be determined before applying (5.2).

We observe that the correction terms $\Delta t_j[\nu]$ has a significantly smaller variance in the drift regime, compared to the transient regime. This is due to the reduction in magnitude of $\overline{\Delta\theta}_j$ as iterations progress. Let $\sigma_{Q,i}[\nu]$ be the standard deviation of $\Delta t_j[\nu]$ over the last

Q iterations

$$\sigma_{Q,i}[\nu] = \sqrt{\frac{1}{Q}\sum_{l=\nu-Q}^{\nu}\left(\Delta t_j[l] - \mu_Q\right)^2} \tag{5.4}$$

where $\mu_Q = \frac{1}{Q}\sum_{k=\nu-Q}^{\nu}\Delta t_j[k]$ corresponds to the arithmetic mean of $\Delta t_j[k]$ over the last Q estimates. We consider a node to be in the drift regime if $\sigma_{Q,i}[\nu]$ is below some threshold $\sigma_{\max}$. An appropriate threshold can be estimated empirically.

Therefore, if a node has been in the drift regime for Q steps, the filtered correction $\Delta\Gamma_j[\nu]$ is used in place of the regular correction $\Delta t_j[\nu]$ in the clock update equation in (3.4). In practice, this filter is instead applied on the estimated correction $\widehat{\Delta t}_j[\nu]$, as the quantity $\Delta t_j[\nu]$ is unavailable during synchronization. The overall Drift Compensation algorithm is presented in Alg. 1. Our numerical simulations presented in the next chapter show that this algorithm successfully prevents the nodes' phase from drifting.

Require: drift_counter = 0 (Initialization)

```
function DRIFT_COMPENSATION_FILTER
    Compute σ_{Q,i}[ν] from (5.4)
    if σ_{Q,i}[ν] < σ_max
        if drift_counter < Q
            drift_counter ← drift_counter + 1
            return Δt̂_j[ν]
        else
            Compute ΔΓ̂_j[ν] from (5.2)
            return ΔΓ̂_j[ν]
    else
        drift_counter = 0
        return Δt̂_j[ν]
```

Algorithm 1: Drift compensation algorithm, applied after every calculation of $\widehat{\Delta t}_j[\nu]$. Note that the algorithm only modifies the correction term $\Delta t_j[\nu]$ if it detects that the node has been in the drift regime for Q samples or more.

Note that (5.3) corresponds to a simple FIR highpass filter. Since we are trying to remove a static term from (4.1), a more sophisticated highpass filter with a deep notch at frequency 0 could be thought of as a better solution than (5.3). However, our testing showed that popular choices for highpass filters (Butterworth, elliptic, FIR Remez, etc) did not produce viable results. This is likely caused by the large magnitude of the taps of such filters coupled with the strong feedback that exists between the nodes as they apply the DPLL algorithm.

5.3 SC-FDMA Modulation

SC-FDMA modulation is commonly used today in wireless uplink transmission due to its attractive peak-to-average power ratio (PAPR), compared to multicarrier approaches. A block diagram of SC-FDMA modulation can be found in Fig. 5.1. The length-N_s sequence $s[n]$ is modulated to the length-N_c sequence $c[m]$ through the following transformation:

$$c[m] = \text{IDFT}_{N_c} \begin{pmatrix} 0_{1\times a_i} \\ S_k \\ 0_{1\times b_i} \end{pmatrix} \tag{5.5}$$

where $S_k = \text{DFT}_{N_s}(s[n])$, a_i is the mapping offset, and $b_i = N_c - N_s - a_i$ is the rest of the zero padding. The mapping offset is a value assigned to different users, and allows them to transmit their signals on a different set of subcarriers. For the remainder of this discussion, we assume that both N_c and a_i are integer multiples of N_s.

During initial synchronization in a D2D network, the mapping offset a_i is unknown at the receiving node j. Therefore $s[n]$ cannot be retrieved through conventional SC-FDMA

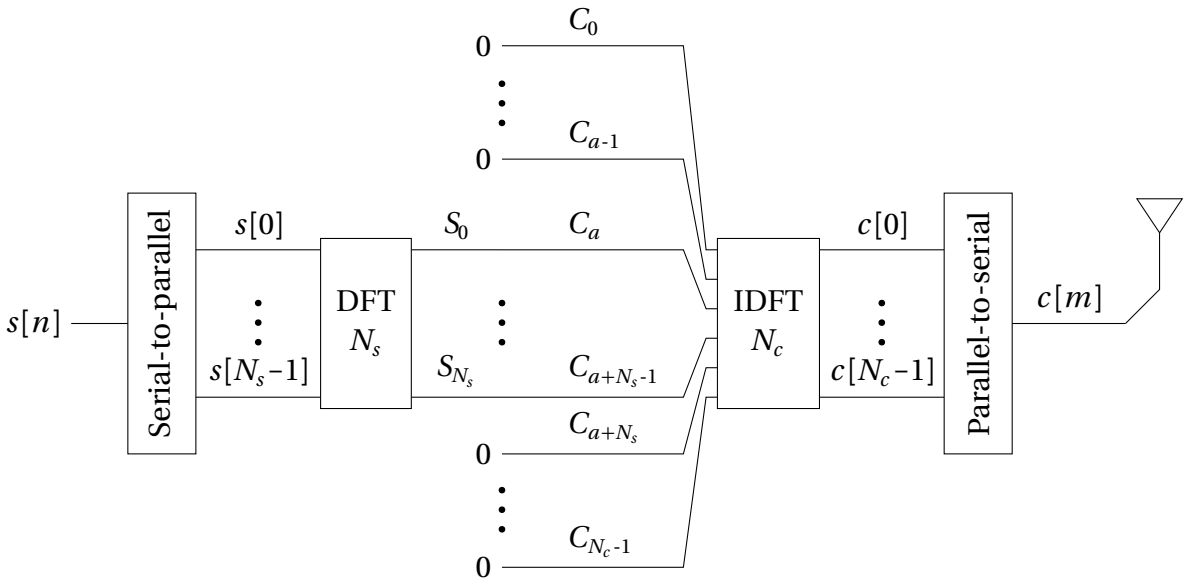

Figure 5.1: SC-FDMA modulation block diagram. Here, $S_k = \text{DFT}_{N_s}(s[n])$ and $C_k = \text{DFT}_{N_c}(c[m])$ are the discrete Fourier coefficients of the input and output sequences.

demodulation. To recover $s[n]$, we first look at the expanded form of $c[m]$:

$$c[m] = \frac{1}{N_c} e^{\pi j \frac{m}{N_c}(2a_i+N_s-1)} \sum_{n=0}^{N_s-1} s[n]\,\mathrm{sincl}(n-\tfrac{m}{L})\, e^{-j\pi n(1-\frac{1}{N_s})} \tag{5.6}$$

$$\textit{where } \mathrm{sincl}(n) = \begin{cases} N_s & n=0 \\ \frac{\sin(\pi n)}{\sin(\frac{\pi}{N_s} n)} & \text{otherwise} \end{cases}$$

where L is such that $N_c = LN_s$, $L \in \mathbb{N}_{\sim}$. The full mathematical development can be found in Appendix 2. From this expansion, $c[m]$ can be thought of as an interpolated, frequency-shifted version of $s[n]$, with interpolating function the sinc-like function $\mathrm{sincl}(m)$. On the receiving side, downsampling by a factor of L can directly recover an exact replica of $s[n]$, barring for the complex exponential factors. These exponentials contribute to DPLL in an equivalent manner to a CFO; since the DPLL algorithm we use is CFO-robust, the presence of these exponentials is inconsequential. This systematic frequency offset will be henceforth called SC-FDMA frequency offset (SFO).

Akin to Sec. 4.2, downsampling an interpolated sequence with some misalignment can be though of as a problem, but this is not a concern for our purpose. This is because we not concerned with the individual values of $s[n]$, but instead with its cross-correlation with $z_{\pm}[n]$. As previously discussed, misalignment in the downsampling leads to a broadening of the cross-correlation peaks, which does not significantly affect the time of reception estimation due to its weighted average nature.

In the multi-user case, an added L-decimation at the receiver yields a signal of the same form as (3.16), that is, a linear superposition of the different synchronization signals, with added inter-symbol interference and SFO.

Overall, the CFO-robust DPLL algorithm described in the previous chapters can be easily adapted to the presence of SC-FDMA modulation.

5.4 Software Implementation

Up to this point, we saw many aspects that affect the DPLL algorithm in practice. Previous work do not consider them at all [10, 11, 17]. In this section, we present a software implementation that incorporates these aspects. We first discuss the implementation details of the DPLL algorithm executed at each node, and then its integration to the simulator as a whole.

It should be noted that we implemented the DPLL algorithm with the assumption that nodes could broadcast and receive signals at the same time (full-duplex). Of course, this is not possible in practice with today's radio equipment, which is half-duplex. We want to emphasize that the use of a full-duplex scheme in our software implementation, while impractical, was a conscious choice to limit the scope of this work, as we wanted to limit our focus on the aspects described on in the previous chapters.

The half-duplex limitation has been the object of decades of research, with many solutions appearing over time, such as time duplexing, frequency duplexing or echo cancellation. Regarding DPLL, we are confident that the algorithm can be adapted to half-duplex schemes using one of the various existing solutions. This is further corroborated by recent work on the topic, which successfully achieved synchronization in a distributed setting using DPLL in conjunction with a random broadcasting scheme for the synchonization signals [21]. Ultimately, a full-duplex implementation allowed us to obtain results that are independent of a particular half-duplex solution.

5.4.1 Local Algorithm Overview

Each node in the system applies the following synchronization algorithm locally, independently of other nodes. The synchronization process is divided into time windows that are roughly T_0 wide, called adjustment windows, which are depicted in Fig. 5.2. Each adjustment window is positioned such that the local clock tick is approximately located halfway inside the window; this is done to minimize edge effects when estimating the correction $\Delta t_j[\nu]$. In each adjustment window, the node broadcasts its synchronization signal once while listening for incoming signals from other nodes. The incoming signals are aggregated

into $y_j[k]$, from which the quantity $\Delta t_j[\nu]$ is estimated. The clock phase is then updated with this estimate of $\Delta t_j[\nu]$. In turn, this new clock phase then affects the time of transmission of the next synchronization signal; therefore, it also affects the location and length of the next adjustment window.

Alg. 2 presents a rough overview of the DPLL implementation locally executed at each node, with the proposed modifications shown as options. After being powered on, the node applies the DPLL algorithm and corrects its local clock, but only begins broadcasting after D clock updates. This prevents nodes that are powering on from corrupting the ongoing synchronization process of already synchronizing nodes.

In practice, node j does not have access not $\theta_j[\nu]$, which is why the *Wait* actions do not directly depend on $\theta_j[\nu]$. This approach is equivalent to the clock update rule specified in (3.3). To track the behaviour of the system, the following is added as line 21 of Alg. 2:

$$\theta_j[\nu] \leftarrow (\theta_j[\nu] + \text{correction})_{T_o} \tag{5.7}$$

The variable $\theta_j[\nu]$ should be understood as a bookkeeping variable, and has no effect on the synchronization itself.

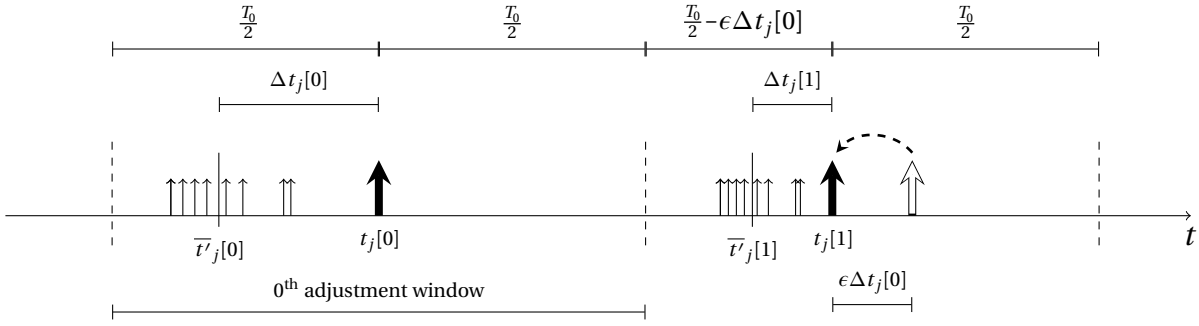

Figure 5.2: Graphical representation of the time of reception and transmission of the synchronization signals. Here, node j's signal is represented with the thick arrow, and corresponds to $t_j[\nu]$. The various reception times $t'_{ij}[\nu]$ are represented with the smaller arrows, and their average by a solid line labelled with $\overline{t'}_j[\nu] = \sum_{i,i\neq j} \alpha_{ij} t'_{ij}[\nu]$. The different adjustment windows corresponds to the area between the dashed lines. Each successive correction approaches the local clock tick closer to the average received clock tick from the other nodes in the network.

In our simulations, all actions happen instantaneously; only *Wait* commands make the simulated universal time move forward. A physical implementation would necessarily have non-zero processing time to many of the proposed steps. To simulate this aspect, we assumed that processing would last at most $T_0/10$ time; this is represented on line 19 of Alg. 2. To understand why the thresholding value is $-2T_0/5$, note that the earliest possible correction is $-T_0/2$. which would make the next clock tick happen instantaneously. Any correction inferior to $-T_0/2$ must therefore be deffered to the next period. Adding $T_0/10$ to that, and we obtain $-2T_0/5$.

Prior to broadcasting, node i interpolates the synchronization sequence $s[n]$ with a root-raised cosine pulse, oversampled by a factor of η. SC-FDMA modulation is then applied, if applicable. This pseudo-analog signal is then broadcasted to node j. The signal is processed to account for the multipath and the CFO that exists between nodes i and j. The TO is not directly applied, but is instead a consequence of the fact that the broadcast

1: $\nu = 0$, broadcast_counter $= 0$, correction $= 0$
2: **while** True
3: $\quad \nu \leftarrow \nu + 1$
4: $\quad$ Begin aggregating $y_j[k]$
5: $\quad$ Wait $T_0/2 +$ correction
6: $\quad$ **if** broadcast_counter $< D$
7: $\quad\quad$ broadcast_counter $\leftarrow$ broadcast_counter $+ 1$
8: $\quad$ **else**
9: $\quad\quad$ Broadcast the synchronization signal.
10: $\quad$ Wait $T_0/2$
11: $\quad$ Stop aggregating $y_j[k]$
12: $\quad$ *Option:* Apply L-Decimation on $y_j[k]$ if SC-FDMA modulation is used
13: $\quad$ *Option:* Apply Bias removal on $y_j[k]$. *See (5.1)*
14: $\quad$ Compute $\widehat{\Delta t}_j[\nu]$ from $y_j[k]$
15: $\quad$ **if** *Option:* Drift Compensation
16: $\quad\quad$ correction $\leftarrow \epsilon \cdot$DRIFT_COMPENSATION_FILTER() *See Alg. 1*
17: $\quad$ **else**
18: $\quad\quad$ correction $\leftarrow \epsilon \cdot \widehat{\Delta t}_j[\nu]$
19: $\quad$ **if** correction $< -2T_0/5$
20: $\quad\quad$ correction $\leftarrow$ correction $+ T_0$

Algorithm 2: Summary of the algorithm applied locally by each node in the network, where all actions other than *Wait* happen instantaneously.

time corresponds to the current clock time of node i. This broadcast process is repeated for all nodes.

5.4.2 Simulator Overview

The simulation first randomly initializes the clock phases, antenna noise samples, and multipath parameters in the network. Each node j is then turned on at universal time $t_{j,\text{start}} = \theta_j[0] + l_j T_o$, where l_j is a randomly picked integer. It is at that time $t_{j,\text{start}}$ that the nodes begin executing Alg. 2.

To truly simulate the DPLL algorithm, a software implementation of Alg. 2 must be executed concurrently at each node. We highlight here how this concurrent execution was implemented. We define *events* as the principal sets of actions executed by the local algorithm, namely, the *broadcast* event and the Δt_j *estimation* event. Put another way, an event can be understood as the sequence of actions between two *Wait* commands in Alg. 2. The execution of an event leads to the creation of a new event. More specifically, a *broadcast* event creates an *estimation* event, which in turn creates a *broadcast* event. These events are inserted in the event list, such that the event list is always in chronological order. This can be summarized as:

1: Initialize event list with a *Broadcast* event for each node, at times $t_{j,\text{start}} + \frac{T_0}{2}$.
2: **while** $t < t_{\text{final}}$
3: Increase universal time t to the next scheduled *Event* in the event list.
4: Remove *Event* from the event list
5: *NewEvent* $\leftarrow$ Execute *Event*.
6: Insert *NewEvent* at the appropriate chronological location in the event list.

where t_{final} is the simulation stoppage time specified by the user. It should be understood that each node always has at least one event in the event list. This approach truly emulates the passage of time within the simulation while being reasonably performant.

Chapter 6

Numerical Simulations

The CFO-resistant DPLL algorithm was implemented and tested using numerical methods. In this chapter, we first introduce new metrics to evaluate the performance of the synchronization algorithm. We then elaborate on the precise methodology used in our simulations. Finally, we present our results, which highlight the impact of the various modifications to the DPLL algorithm proposed in the previous chapter.

6.1 Performance Metrics

Multiple metrics can be used to evaluate the convergence of a distributed synchronization algorithm. A commonly used metric is the standard deviation of the phases $\theta_j[\nu]$. However, it does not directly reflect each node's capacity to communicate after synchronization. For this task, we shall define a communication ratio as the fraction of the number of usable links in the network. Furthermore, two other metrics will be introduced to monitor the presence of a drift in the phases of the nodes.

6.1.1 Communication Criterion

The utilization of cyclic prefixes (CPs) and cyclic suffixes (CSs) enables wireless devices to tolerate some forward or backward lag between the expected and actual reception time of a signal. Let O_{ij} be the overall time offset between receiving node j and transmitting node

i:

$$O_{ij} = \Delta\theta_{ij} + \tau_{ij,b} \tag{6.1}$$

where $\Delta\theta_{ij}$ is the previously defined phase offset and $\tau_{ij,b}$ is the delay of the path that would be used for communication, where we define this b^{th} path as having the highest magnitude across all paths p: $|\rho_{ij,b}| = \max_p |\rho_{ij,p}|$.

Furthermore, let t_p and t_s be the respective duration of the cyclic prefix and suffix. A pair of nodes can be considered able to communicate if their overall time offsets are within the bounds prescribed by t_p and t_s. Let the link quality C_{ij} be the bidirectional usability of the link between nodes i and j:

$$C_{ij} = \begin{cases} 1 & -t_s \leq O_{ij}, O_{ji} < t_p \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

The communication ratio C can then be used to evaluate the synchronization state of the whole network. It is defined as the ratio between the sum of all C_{ij} over the total number of potential links:

$$C = \frac{\sum_{i<j} C_{ij}}{\frac{1}{2}(M^2 - M)} \tag{6.3}$$

Note that by definition, $0 \leq C \leq 1$, where a value of 1 indicates that all the links in the network can be used. This quantity represents the fraction of all usable link pairs in the system, and can therefore be used to evaluate the overall quality of the synchronization.

6.1.2 Stability Criterion

An easy way to determine the stability of the network's synchronization is to look at the evolution of the clock phase $\theta_j[\nu]$ after the system reached the drift regime. As an example, if we look at Fig. 4.2, we are interested in the values of $\theta_j[\nu]$ once all nodes have entered the drift regime. Let β_j be the estimated slope of $\theta_j[\nu]$ over the last B samples of $\theta_j[\nu]$, counting down from the end of the simulation. We consider $\overline{\beta}$ and σ_β^2 to be the arithmetic mean and variance of the slopes across all nodes. Ideally, this slope should be zero for all nodes. If however drift exists, having the same drift rate can be sufficient for communication, as the

phase offset would remain the same over time. Thus, the drift can be characterized by the statistics of β:

- $|\overline{\beta}|$: The absolute value of $\overline{\beta}$ represents the significance of the overall network drift. A large value points at a systematic error in the synchronization algorithm.
- σ^2_β: Represents the spread of the drift across all nodes. A low value corresponds to a network were most clocks are drifting in the same way, whereas a high value implies that each node in the network is drifting in a different manner.

The slope β_j is in units of $\frac{\text{time}}{\text{time}}$; we will express it in milliseconds per seconds.

6.2 Methodology

The simulations described here were done using Python and its scientific computing libraries Numpy/Scipy, under the technical specifications for LTE Advanced D2D proximity services [18]. These include the pathloss model, probability of LOS between nodes, shadowing, multipath parameters, thermal noise power, transmission power, spatial distribution, and others. It should be noted that [18] stipulate an average node velocity of $3m/s$,

Deployment parameters	
Number of nodes M	40
Channel model	According to TR 36.843 [18, Table A.2.1.2]
Carrier frequency F_c	2 GHz
Sampling rate F_s	30.72 MHz
Node velocity	-
Spatial distribution	Uniform over a 500m x 500m square
Range of f_{ij} (CFO) [14]	Uniform over $[-20, 20]$ kHz
Noise power	-101 dBm
Antenna noise figure	9 dB
Antenna gain	0 dBi
Transmission power	23 dBm
Shadowing standard deviation	7 dB
Shadowing correlation	i.i.d.

Table 6.1: List of deployment parameters. All the parameters presented here were obtained from [18] unless otherwise specified. The exception to this rule is the node velocity, which was assumed to be zero.

Algorithm parameters	
Clock period T_0	3.26 ms
Start time factor l_j	Uniform $\in [0, 15]$
Correction scaling ϵ	0.5
DC filter length Q	6
Weighting parameter γ	2
Broadcast delay D	4
SC-FDMA length L	8
SC-FDMA offset a_i	Uniform $\in [0, L)$
ZC length N	31
Stoppage time t_{final}	$140\,T_0$

Table 6.2: Simulations parameters pertaining to the DPLL algorithm implemented at each node. It includes the parameters for the modifications described in the previous chapter.

but was omitted for this work, as we considered static networks. Otherwise, the parameters specified in [18] were respected and properly implemented. Table 6.1 contains all the deployment parameters used for our simulations.

The algorithm presented in this work also involved several parameters. The values for these parameters are presented in Table 6.2, which were empirically determined to be good compromises between computational load, realism, and quality of synchronization. It should be noted that our implementation generally reached a drift regime in approximately 25 iterations, where each iteration lasts approximately T_0 time. This led us to choosing a stoppage time of $t_{\text{final}} = 140\,T_0$, to ensure the simulation exited the converging regime.

Unless otherwise specified, all results presented in the next section used the parameters specified in Table 6.1 and Table 6.2.

6.3 Results

In this section, we investigate the impact of our algorithm modifications using numerical simulations. We consider five configurations, where each configuration has a different subset of options, activated or not. 1500 simulations were executed per configuration. Each simulation had different randomized initialization parameters. At the end of the simula-

DC	BR	SC	C_{avg}	C_{std}	$\lvert\overline{\beta}\rvert_{\text{avg}}$	$\sigma^2_{\beta,\text{avg}}$
-	-	-	0.378	0.102	0.323	0.0134
✓	-	-	0.641	0.108	0.00657	5.65e-05
✓	✓	-	0.608	0.109	0.00722	5.98e-05
✓	-	✓	0.63	0.106	0.00712	9.17e-05
✓	✓	✓	0.621	0.106	0.00664	0.000239

Table 6.3: Convergence metrics for various configurations of the synchronization algorithm. The configuration options are DC for Drift Correction, BR for Bias Removal, and SC for SC-FDMA modulation. Each configuration was simulated 1500 times.

tions, the performance metrics discussed earlier in this chapter were computed. The result of the simulations were then averaged per configuration, which can be found in Table 6.3.

The drift compensation (DC) algorithm had a very positive impact on both the drift magnitude $|\overline{\beta}|$ and the drift spread σ^2_{β}, when compared to the unmodified algorithm described in [11]. It also had significantly higher synchronization quality, as demonstrated by the increase of the average communication ratio C_{avg} and the reduction in its standard deviation. Due to its ability to prevent clock drift, both the bias removal and SC-FDMA modulation were studied in simulations were DC was concurrently enabled. The second line in Table 6.3 will henceforth be referred as the base case.

The application of bias removal (BR) had a clear negative effect on the performance metrics when compared to the base case. The average communication ratio diminished while the average drift magnitude increased. The processing artifacts shown in Fig. 4.1 are not as detrimental as we originally hypothesized. It should be noted that BR removes the effect noise term $w[k]$ from the cross-correlation under most circumstances. We noticed, for simulations without BR, that setting the noise power to $-\infty$dB led to a poorer synchronization performance. It is possible that the presence of noise helps achieving a better synchronization, and would explain the poorer performance of the synchronization algorithm with BR enabled.

SC-FDMA had a slight impact on the synchronization quality, with a small reduction in both C_{avg} and its standard deviation C_{std}, when compared to the base case. In spite of this,

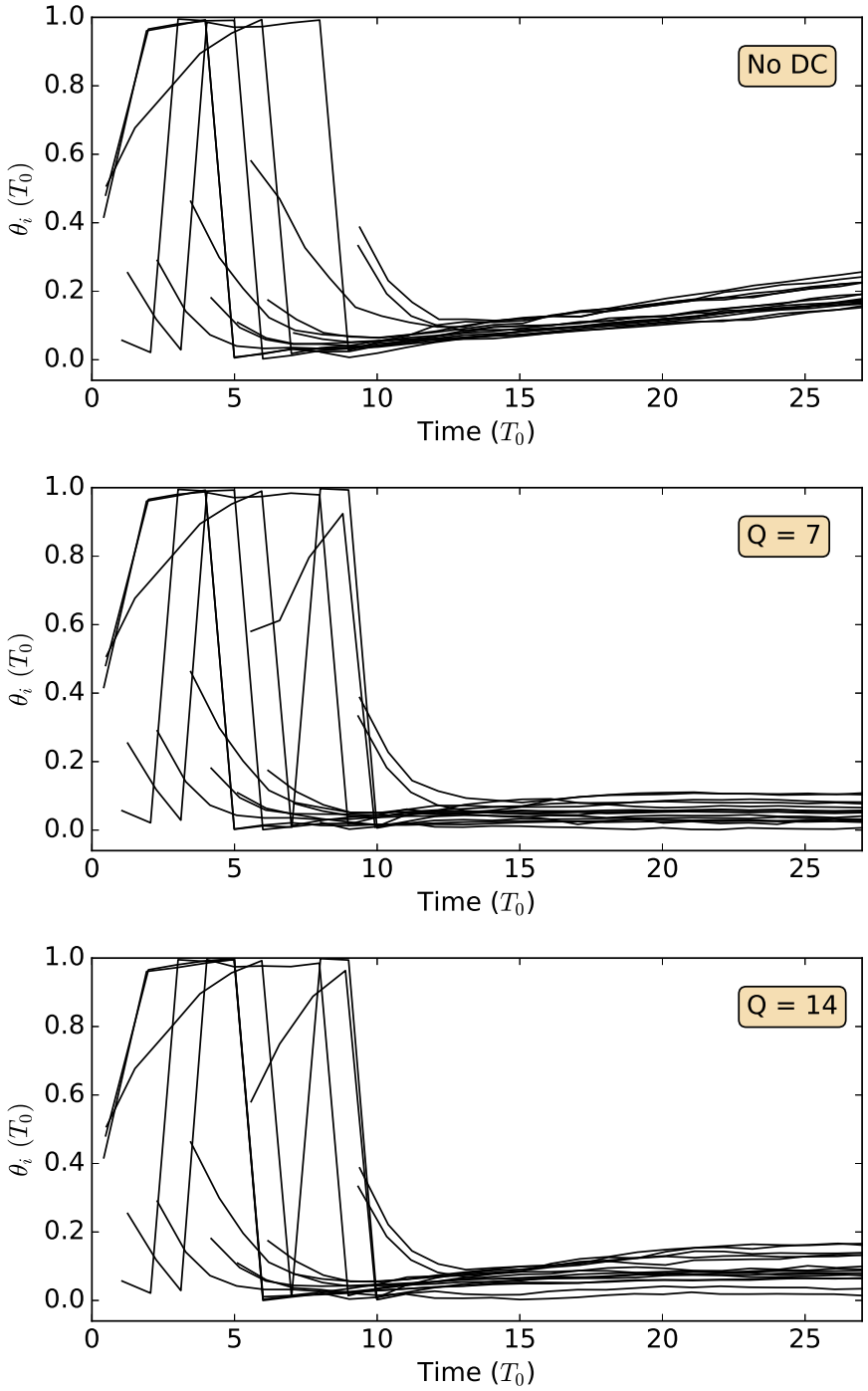

Figure 6.1: Example time evolution of $\theta_j[n]$. The top graph has no DC applied. The bottom graphs has the DC enabled, with the middle graph having a filter length of $Q = 7$. The bottom has a DC filter length of $Q = 14$. Unlike the rest of the results presented in this section, the simulations presented in these graphs had a clock period of $T_0 = 120\mu s$, in order to give a better visual effect for the reader.

the SC-FDMA modulation does not substantially impair the network's capacity to synchronize, and can therefore be used safely in conjunction with the DPLL algorithm.

To highlight the dramatic impact of the DC algorithm, we re-ran the synchronization simulation depicted Fig. 4.2, but this time with the DC algorithm enabled; this is presented on Fig. 6.1. Note that the simulations depicted in Fig. 6.1 used the same initialization values, including the same noise samples. Clearly, the DC was successful in stopping the drift and stabilizing the clock phases: the mean slope $|\overline{\beta}|$ for the $Q = 7$ simulation is $0.08\frac{ms}{s}$, while the mean slope for the $Q = 14$ simulation is $0.07\frac{ms}{s}$. This is a clear improvement to the mean slope of $0.52\frac{ms}{s}$ found in Fig. 4.2. It should be explicit from the figure that having a longer filter length Q leads to a delayed compensation of the drift, which in turn often results in a poorer synchronization quality, as the clock phases $\theta_i[\nu]$, as the clock phases are more spread out. We determined empirically through many simulations that a filter length of $Q = 6$ used for the results presented in Table 6.3 yielded both good stability and good synchronization quality.

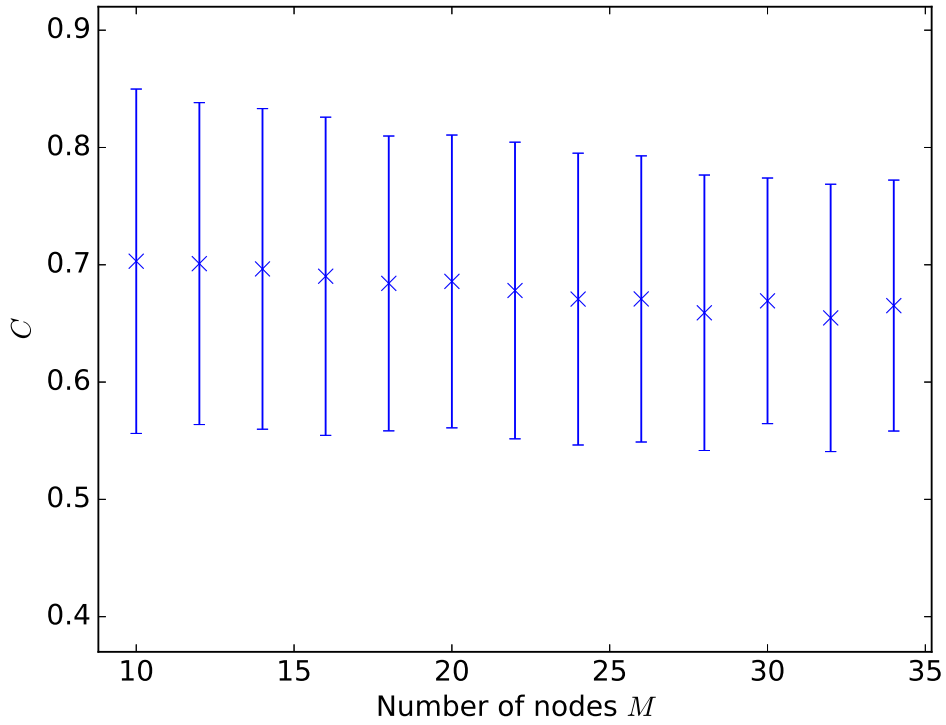

Figure 6.2: Communication ratio as a function of number of nodes. Each datapoint represents the average of 150 simulations, with associated standard deviation.

We now test the behaviour of the algorithm under different settings. The simulations presented until the end of this chapter all used the DPLL algorithm where both the DC algorithm and SC-FDMA modulation were applied.

We first change the density of the node distribution. Fig. 6.2 show the communication ratio C for the DPLL algorithm as a function of number of node. Changing the number of node did not have a significant impact on quality of synchronization. Synchronization quality seems to be mostly independent from the density of nodes in the network.

We limited our previous simulations to happen in a 500m x 500m square area, as prescribed by 3GPP. However, real-world applications may have a larger area requirement. Presented on Fig. 6.3 is the communication ratio C for the DPLL algorithm as a function of area size, where $M = 35$ nodes populated the network. It is clear that larger areas lead to poorer synchronization performances, which can be explained by the increased path loss between the nodes.

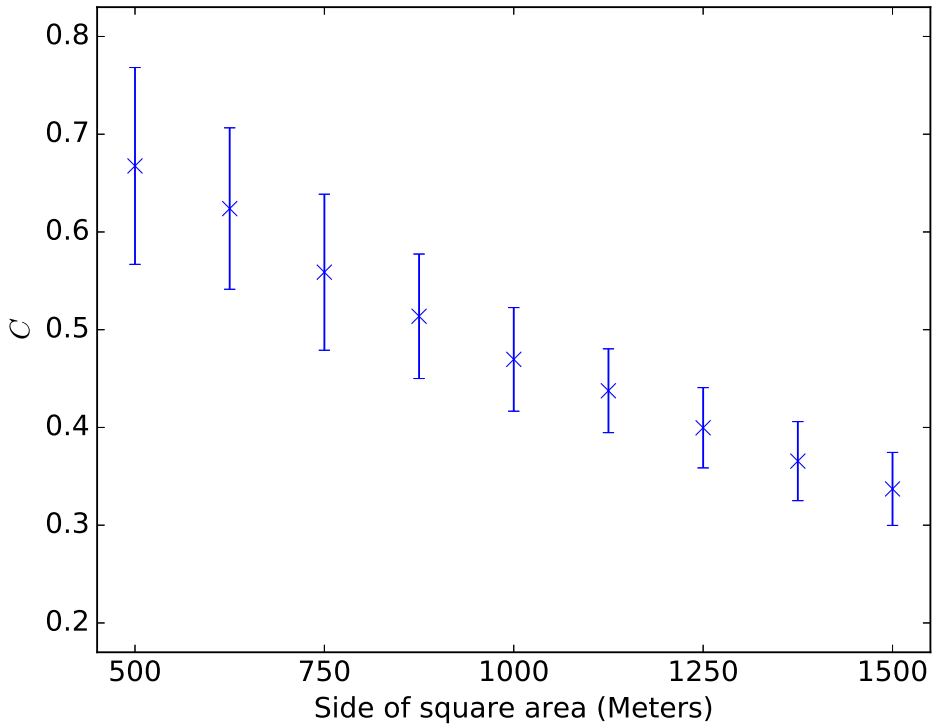

Figure 6.3: Communication ratio as a function of area size. Each datapoint represents the average of 150 simulations, with associated standard deviation.

Finally, we varied the length of the ZC sequences used in the algorithm. Having a longer ZC sequence yields higher peak in the cross-correlation $R_{yz_{\pm}}[l]$. This effect is evident at shorter sequence lengths, which have a worse C_{ag}. We would expect a continuing increase in C_{avg} as but is not observed at higher N, as depicted in Fig. 6.4. A average reduction of 10 points in communication ratio can be observed between the best ($N = 25$) and the longest ($N = 260$) tested ZC lengths. We are unsure as to what precisely causes this decrease.

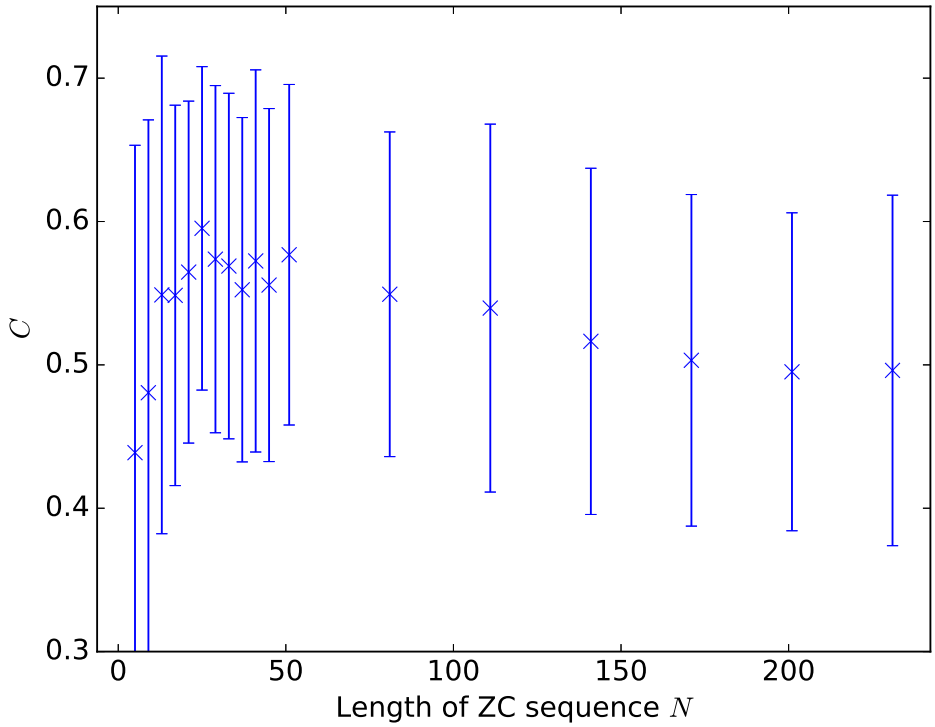

Figure 6.4: Communication ratio as a function the length of the ZC sequence used. Each datapoint representes the average of 150 simulations, with associated standard deviation.

Chapter 7

Conclusion

7.1 Summary

This work examined several limitations and aspects that affect a fully distributed synchronization algorithm under constraints specified by 3GPP for D2D applications. We first presented a comprehensive summary of existing synchronization algorithms before outlining a CFO resistant version the DPLL algorithm proposed in [11]. Unlike previous work on the topic, we considered in our analysis the effects of transmission delay, multipath propagation, and SC-FDMA modulation. These aspects had a significant impact on the synchronization algorithm. Accordingly, we designed modifications to the original algorithm to compensate for them.

A software simulator was then created expressly for the purpose of testing the proposed modifications. It properly imitates the concurrent nature of a distributed algorithm, where many devices perform actions at the same time. Furthermore, we attempted to make the simulator as realistic as possible by including the effects of analog-digital conversions normally present in real-world hardware. In the same vein, parameters such as channel model or noise power closely reflected the current requirements set by 3GPP for D2D proximity services [18].

We demonstrated that some of our modifications to the CFO-resistant DPLL algorithm proposed in [11] significantly increased the quality and stability of the network's synchronization in realistic scenarios. In particular, the effects of transmission delays and multi-

path propagation were properly compensated by the DC algorithm. We further demonstrated that the synchronization algorithm could easily be adapted for SC-FDMA modulation at the transmitter. However, our proposed BR algorithm did not result in a better synchronization. To enable this comparison of algorithms, we devised new performance metrics to evaluate the quality of a network's synchronization.

7.2 Future Work

There are many different avenues on which the topics presented in this work could be extended. We noticed that having fewer nodes in the network often led to better synchronization. Therefore, in larger networks, it may be beneficial to limit the number of broadcasting nodes; non-broadcasting nodes, or quiet nodes, would execute the rest synchronization algorithm as normal. However, the decision of which nodes are to be quiet is a non-trivial problem, especially considering that the decision process should ideally be implementable in a distributed manner.

The analysis and results presented in this work considered static D2D networks. Many real-world applications are likely to have very fluid networks, where nodes can move or be turned off at any time. Modifying the synchronization algorithm to which such conditions would be a significant step for future research.

Throughout our analysis, we made no mention of the time required for a network to achieve the drift regime. In fact, we avoided the topic of convergence time altogether, as we desired to maintain the focus on the quality of a network's synchronization. While our implementation could reach a drift regime quickly enough, certain applications may require careful analysis of the convergence time of this algorithm. Particularly, in the more fluid networks described above, having a better convergence time is likely to lead to better tracking in the event of large network changes.

Finally, we assumed in our analysis that the nodes in the network did not move. This allowed us to have static channel coefficient. Practical applications would require testing the merits of this algorithm in the presence of mobile nodes.

Bibliography

[1] Nokia Siemens Networks. (2011) 2020: Beyond 4G Radio Evolution for the Gigabig Experience. [Online]. Available: http://ch.networks.nokia.com/file/15036/2020-beyond-4g-radio-evolution-for-the-gigabit-experience

[2] A. Osseiran, V. Braun, T. Hidekazu, P. Marsch, H. Schotten, H. Tullberg, M. Uusitalo, and M. Schellman, "The Foundation of the Mobile and Wireless Communications System for 2020 and Beyond: Challenges, Enablers and Technology Solutions," in *Vehicular Technology Conference (VTC Spring)*, June 2013, pp. 1–5.

[3] X. Chen, L. Chen, M. Zeng, X. Zhang, and D. Yang, "Downlink Resource Allocation for Device-to-Device Communication Underlaying Cellular Networks," in *Symp. on Personal Indoor and Mobile Radio Commun. (PIMRC)*, Sept 2012, pp. 232–237.

[4] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, "Clock Synchronization of Wireless Sensor Networks," *IEEE Signal Processing Mag.*, vol. 28, no. 1, pp. 124–138, Jan 2011.

[5] A. Bletsas, "Evaluation of Kalman Filtering for Network Time Keeping," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 52, no. 9, pp. 1452–1460, Sept 2005.

[6] S.-L. Chao, H.-Y. Lee, C.-C. Chou, and H.-Y. Wei, "Bio-Inspired Proximity Discovery and Synchronization for D2D Communications," *IEEE Commun. Lett.*, vol. 17, no. 12, pp. 2300–2303, December 2013.

[7] W. Sun, M. Gholami, E. Strom, and F. Brannstrom, "Distributed Clock Synchronization with Application of D2D Communication Without Infrastructure," in *Proc. 2013 Globecom Workshops*, Dec 2013, pp. 561–566.

[8] W. Sun, E. Strom, F. Brannstrom, and M. Gholami, "Random Broadcast Based Distributed Consensus Clock Synchronization for Mobile Networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 6, pp. 3378–3389, June 2015.

[9] D. Goodman and R. Myers, "3G Cellular Standards and Patents," in *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, vol. 1, June 2005, pp. 415–420 vol.1.

[10] O. Simeone and U. Spagnolini, "Distributed Synchronization in Wireless Networks," *IEEE Signal Processing Mag.*, vol. 25, no. 5, pp. 81–97, 2008.

[11] M. Alvarez, B. Azari, and U. Spagnolini, "Time and Frequency Self-Synchronization in Dense Cooperative Network," in *48th Asilomar Conf. on Signals, Syst. and Compt.*, Nov 2014, pp. 1811–1815.

[12] Y.-W. Hong and A. Scaglione, "A Scalable Synchronization Protocol for Large Scale Sensor Networks and its Applications," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 5, pp. 1085–1099, May 2005.

[13] A. Tyrrell, G. Auer, and C. Bettstetter, "Emergent Slot Synchronization in Wireless Networks," *IEEE Trans. Mobile Computing*, vol. 9, no. 5, pp. 719–732, May 2010.

[14] M. Gul, X. Ma, and S. Lee, "Timing and Frequency Synchronization for OFDM Downlink Transmissions Using Zadoff-Chu Sequences," *IEEE Trans. Wireless Commun.*, vol. 14, no. 3, pp. 1716–1729, March 2015.

[15] W. Yang, M. Hua, J. Zou, J. Hu, J. Zhang, and M. Wang, "On the Frequency Offset Effect on Zadoff-Chu Sequence Timing Performance," in *Procs. 2014 World Telecommun. Congr.*, June 2014, pp. 1–5.

[16] J. Zhang, C. Shen, G. Deng, and Y. Wang, "Timing and Frequency Synchronization for Cooperative Relay Networks," in *Fall 2013 Vehicular Technology Conf.*, Sept 2013, pp. 1–5.

[17] M. Cremasehi, O. Simeone, and U. Spagnolini, "Distributed timing synchronization for sensor networks with coupled discrete-time oscillators," in *2006 Ann. IEEE Commun. Soc. Sensor, Ad Hoc Commun., and Networks*, vol. 2, Sept 2006, pp. 690–694.

[18] "Technical Specification Group Radio Access Network; Study on LTE Device to Device Proximity Services," 3rd Generation Partnership Project (3GPP), TR 36.843, Mar. 2014, Sections A.2.1.1 - A.2.1.2. [Online]. Available: www.3gpp.org/ftp/Specs/archive/36_series/36.843/36843-c01.zip

[19] J. Proakis and M. Salehi, *Digital Communications*, ser. McGraw-Hill International Edition. McGraw-Hill, 2008.

[20] D. Chu, "Polyphase codes with good periodic correlation properties," *IEEE Trans. Inf. Theory*, vol. 18, no. 4, pp. 531–532, Jul 1972.

[21] M. A. Alvarez and U. Spagnolini, "Half-duplex scheduling in distributed synchronization," *IEEE Int. Conf. on Communs. 2015*, pp. 6240–6245, June 2015.

Appendix A

Circular Cross-correlation of a Zadoff-Chu Sequence under CFO

Let $z[n]$ be an odd-length zadoff-chu sequence, and let $z'[n]$ be the same sequence affected by some arbitrary carrier frequency offset $\Delta\lambda$:

$$z[n] = e^{-j\pi un(n+1)/N}, \quad N \text{ odd} \tag{A.1}$$

$$z'[n] = z[n]e^{j\frac{2\pi}{N}\Delta\lambda n} \tag{A.2}$$

The cyclical cross-correlation between the two sequences can be written as:

$$r_{zz'}[l] = \sum_{n=0}^{N-1} z[(n+l)_N](z'[n])^* \tag{A.3}$$

Since $z[n]$ is periodic with period N, the modulo operator can be omitted. Substituting in the definitions of both sequences, we obtain:

$$r_{zz'}[l] = \sum_{n=0}^{N-1} e^{-j\pi u(n+l)(n+l+1)/N} e^{j\pi un(n+1)/N} e^{-j\frac{2\pi}{N}\Delta\lambda n} \tag{A.4}$$

$$= \sum_{n=0}^{N-1} e^{\gamma_u(n,l)} \tag{A.5}$$

where $\gamma_u(n,l)$ can be simplified as:

$$\gamma_u(n,l) = \frac{j\pi}{N}\big[-u(n+l)(n+l+1)+un(n+1)-2\Delta\lambda n\big] \tag{A.6}$$

$$= -\frac{j\pi}{N}\big[ul(l+1)+2n(ul+\Delta\lambda)\big] \tag{A.7}$$

Substituting $\gamma_u(n,l)$ back into (A.5), we obtain:

$$r_{zz'}[l] = \sum_{n=0}^{N-1} e^{-\frac{\pi ju}{N}l(l+1)} e^{-\frac{2n\pi j}{N}(ul+\Delta\lambda)} \tag{A.8}$$

$$= z[l]\sum_{n=0}^{N-1}(e^{-\frac{2\pi j}{N}(ul+\Delta\lambda)})^n \tag{A.9}$$

$$= z[l]\frac{1-e^{-2\pi j(ul+\Delta\lambda)}}{1-e^{-\frac{2\pi j}{N}(ul+\Delta\lambda)}} \qquad , \frac{ul+\Delta\lambda}{N} \notin \mathbb{Z} \tag{A.10}$$

$$= z[l]\frac{e^{-\pi j(ul+\Delta\lambda)}\sin(\pi(ul+\Delta\lambda))}{e^{-\frac{\pi j}{N}(ul+\Delta\lambda)}\sin(\frac{\pi}{N}(ul+\Delta\lambda))} \tag{A.11}$$

$$= z[l]e^{-\pi j\alpha(1-\frac{1}{N})}\mathrm{sinc}(\alpha) \tag{A.12}$$

where $\alpha = ul+\Delta\lambda$ and $\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\sin\frac{\pi x}{N}}$

Note the restriction on using the geometric series in (A.10), where $\frac{ul+\Delta\lambda}{N} \notin \mathbb{Z}$. In such cases, the exponential simplifies to 1, meaning that $r_{zz'}[\frac{kN-\Delta\lambda}{u}] = Nz[l]$, $k \in \mathbb{Z}$

Appendix B

SC-FDMA Modulation: Expanded Form

The length-N sequence $s[n]$ is SC-FDMA modulated to the length-M sequence $c[m]$ with the following transformation:

$$c[m] = \text{IDFT}_M \begin{pmatrix} 0_{1\times a} \\ S_k \\ 0_{1\times b} \end{pmatrix} \tag{B.1}$$

$$c[m] = \text{IDFT}_M(C_k) \tag{B.2}$$

where $S_k = \text{DFT}_N(s[n])$, $C_k = \text{DFT}_M(s[m])$, a is the mapping offset, and $b = M - N - a$ is the rest of the zero padding. For the remainder of this discussion, we assume that both M and a are integer multiples of N, namely: $M = LN$, $L \in \mathbb{N}$ and $a = IN$, $I \in \mathbb{N}$. We will also assume that $a < (L-1)M$.

Expanding the IDFT in (B.1), we obtain:

$$c[m] = \frac{1}{M}\sum_{k=0}^{M-1} C_k\, e^{2\pi jkm} \tag{B.3}$$

$$= \frac{1}{M}\sum_{k=a}^{a+N-1} S_k\, e^{2\pi jkm} \tag{B.4}$$

$$= \frac{1}{M}\sum_{k=a}^{a+N-1}\sum_{n=0}^{N-1} s[n]\, e^{-2\pi jk\frac{n}{N}} e^{2\pi jk\frac{m}{M}} \tag{B.5}$$

$$= \frac{1}{M}\sum_{n=0}^{N-1} s[n] \sum_{k=a}^{a+N-1} W^k \tag{B.6}$$

where $W = e^{-2\pi j(\frac{n}{N}-\frac{m}{M})}$. It can be simplified to:

$$W = e^{-2\pi j(\frac{n}{N}-\frac{m}{M})} \tag{B.7}$$

$$= e^{-2\pi j\frac{nL-m}{NL}} \tag{B.8}$$

$$= e^{-2\pi j\frac{p}{N}} \tag{B.9}$$

where $p = \frac{nL-m}{L} = n - \frac{m}{L}$ is a simplification done to lighten the notation in the following equations.

The sum over k in (B.6) can be interpreted as a geometric series:

$$\sum_{k=a}^{a+N-1} W^k = W^a \frac{1-W^N}{1-W} \tag{B.10}$$

$$= W^a \frac{1-e^{-2\pi jp}}{1-e^{-2\pi jp/N}} \tag{B.11}$$

$$= e^{-2\pi aj\frac{p}{N}} e^{-\pi jp(1-\frac{1}{N})} \frac{\sin(\pi p)}{\sin(\pi p/N)} \tag{B.12}$$

The complex exponentials can be simplified to:

$$= \left(e^{-2\pi a j \frac{p}{N}}\right)\left(e^{-\pi j p(1-\frac{1}{N})}\right) \tag{B.13}$$

$$= \left(e^{-2\pi a j(\frac{n}{N}-\frac{m}{M})}\right)\left(e^{-\pi j(n-\frac{m}{L})(1-\frac{1}{N})}\right) \tag{B.14}$$

$$= \left(e^{-2\pi j \frac{LNn}{N}}\right)\left(e^{2\pi j \frac{am}{M}}\right)\left(e^{-\pi j n(1-\frac{1}{N})}\right)\left(e^{\pi j \frac{m}{L}(1-\frac{1}{N})}\right) \tag{B.15}$$

$$= \left(e^{\pi j \frac{m}{M}(2a+N-1)}\right)\left(e^{-\pi j n(1-\frac{1}{N})}\right) \tag{B.16}$$

Pluging everything back into $c[m]$, we obtain:

$$c[m] = \frac{1}{M} e^{\pi j \frac{m}{M}(2a+N-1)} \sum_{n=0}^{N-1} s[n] \frac{\sin(\pi(n-\frac{m}{L}))}{\sin(\frac{\pi}{N}(n-\frac{m}{L}))} e^{-j\pi n(1-\frac{1}{N})} \tag{B.17}$$

This can be interpreted as an interpolation formula with the sinc-like function $\text{sincl}(x) = \frac{\sin(x)}{\sin(x/N)}$, along with some carrier frequency offset.