On the Use of Distributed Synchronization in 5G Device-to-Device Networks

David Tétreault-La Roche[1], Benoit Champagne[1], Ioannis Psaromiligkos[1], Benoit Pelletier[2]
[1]Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada
[2]InterDigital Canada, Montréal, QC, Canada
david.tetreault-laroche@mail.mcgill.ca, benoit.champagne@mcgill.ca,
ioannis.psaromiligkos@mcgill.ca, benoit.pelletier@interdigital.com

***Abstract*—Time synchronization is a key aspect of device-to-device (D2D) schemes, particularly in decentralized networks where no reference time is available. Distributed phase-locked loops (DPLL) is a synchronization algorithm well suited for decentralized situations. In this work, we study DPLL in the context of 5G networks, where we include in our analysis several practical aspects of D2D communication, such as propagation delays, multipath propagation, and the use of single-carrier frequency division multiple access (SC-FDMA). We propose practical methods to compensate for 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 3GPP for D2D applications.**

I. Introduction

Current cellular networks make use of a centralized paradigm, where all communications transit through an access point. D2D communication was introduced to offload some traffic from the access point by allowing devices within range to interact directly. This approach also has other advantages, such as reduced power consumption and lower infrastructure costs [1], and accordingly, it is recognized as a key enabling technology for fifth-generation (5G) networks. However, time synchronization between the devices is an essential requirement for D2D communication.

To achieve synchronization, interacting nodes in a wireless network attempt to estimate the time offset (TO) between their respective clocks via the exchange of synchronization messages transmitted by means of radio frequency (RF) carrier waves. However, a carrier frequency offset (CFO) nearly always exists between the nodes, which has a detrimental effect on the TO estimation [2, 3]. Ideally, communicating nodes should account for both the TO and CFO between themselves. Recent works successfully decoupled the estimation of TO and CFO by using well crafted synchronization messages [2, 4, 5].

Certain D2D applications require a decentralized network, where time reference sources such as GPS or base stations are unavailable. Distributed synchronization algorithms have long been studied, giving rise to various methods, such as the firefly [6], timestamps exchange [7, 8], and distributed phase-locked loops (DPLL) algorithms [4, 9]. DPLL can be implemented at the physical layer and scales well with an increasing number of devices [10], making it a pertinent choice for decentralized D2D applications.

This work was supported by InterDigital Canada Ltée and MITACS Canada under the MITACS Accelerate program.

In this paper, we present a DPLL-type algorithm that takes into account several practical aspects of D2D communication under 5G. In contrast to previous work on the topic [4, 9, 10], we explicitly consider: *(i)* Propagation delays between devices, *(ii)* Wideband multipath propagation, *(iii)* Use of SC-FDMA. In particular, propagation delays and multipath propagation have a strong negative impact on DPLL's performance, as the algorithm depends on the time of reception of synchronization signals. We propose a modification to DPLL that compensates for the effects of both phenomena. Furthermore, we adapt DPLL to function properly under SC-FDMA, which is a requirement specified by the 3rd Generation Partnership Project (3GPP) for D2D applications and proximity services [11]. Finally, we introduce new performance metrics to evaluate the merit of those modifications. Through our physical-layer simulations, we demonstrate that time synchronization in a decentralized D2D setting is achievable under the constraints specified by 3GPP [11] for future 5G networks.

The paper is structured as follows. We present the system model in Section II and the background on DPLL in Section III. The proposed modifications to the DPLL algorithm are discussed in Section IV. The new performance metrics along with results are presented in Section V. Finally, Section VI concludes the paper.

II. System Model

We consider a network of M wireless devices, or nodes, indexed with $i \in \{1, 2, ..., M\}$. Each node can communicate with the other nodes via wideband multipath fading radio channels. All nodes use the same synchronization algorithm, but apart from that, they have no *a priori* information about the rest of the nodes.

A. Discrete Clock Model

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

$$t_i(t) = \alpha_i t + \theta_i \tag{1}$$

where θ_i and α_i are the clock phase and the clock skew [8], respectively. This paper focuses on skew-synchronized clocks, i.e., $\alpha_i = 1,\ \forall i$. For the purpose of synchronization, each node maintains a discrete logical clock $t_i[\nu]$, which is obtained by sampling (1) with a period T_0:

$$t_i[\nu] = t_i(\nu T_0) = \nu T_0 + \theta_i \tag{2}$$

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. 1. We consider

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

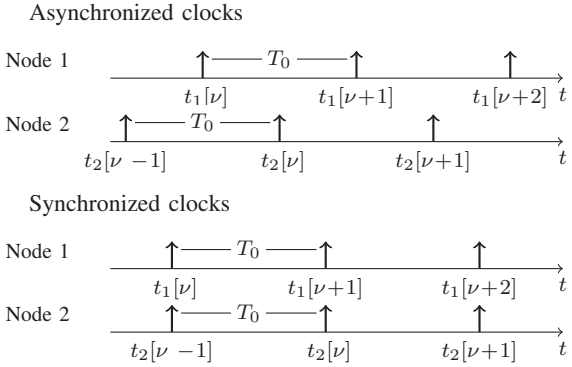

Fig. 1: Graphical representation of synchronous and asynchronous discrete clocks. Each vertical arrow represents a clock tick at the associated time index.

two clocks i and j to be synchronized if their successive clock ticks are aligned with one another (note that the tick indices may be different). That is, 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 then assume without loss of generality that θ_i lies within $[0, T_0)$.

B. Signal Transmission

Nodes broadcast an identical synchronization sequence $s[n] \in \mathbb{C}$ of length N at their respective clock ticks. Prior to broadcasting, this sequence is mapped into an analog baseband signal by means of digital-to-analog (DAC) conversion at the rate $1/T_s$ with T_s being the sampling period (we assume that $NT_s \ll T_0$). The analog equivalent of $s[n]$ can be expressed as:

$$x(t) = \sum_{n=0}^{N-1} s[n]\, p(t - nT_s) \tag{3}$$

A common choice for $p(t)$, and the one used in this paper, is the root raised cosine pulse [12].

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

$$x_i(t) = x(t - t_i[\nu]) \tag{4}$$

This signal is then upconverted to node i's carrier frequency f_i prior to being broadcasted:

$$\widetilde{x}_i(t) = x_i(t) e^{\mathrm{j}2\pi f_i t} \tag{5}$$

C. Received Signal

Signal transmission from node i to node j occurs over a wideband multipath channel with impulse response:

$$h_{ij}(t) = \sum_p \rho_{ij,p}\, \delta(t - \tau_{ij,p}) \tag{6}$$

where $\rho_{ij,p} \in \mathbb{C}$, $\tau_{ij,p} > 0$ are the complex amplitude and propagation delay, respectively, associated with the p^{th} resolvable path, and $\delta(t)$ is the Dirac delta function.

Since all nodes broadcast $\widetilde{x}_i(t)$ at every local clock tick, node j receives a superposition of the synchronization signals from all nodes $i \neq j$ once per clock period T_0. After reception, node j downconverts this signal from RF to baseband. The downconverted signal over a period T_0 can be written as:

$$\begin{aligned} y_j(t) &= \sum_{i,i\neq j} \sum_p \rho_{ij,p}\, x_i(t - \tau_{ij,p}) e^{\mathrm{j}2\pi f_i (t-\tau_{ij,p})} e^{-\mathrm{j}2\pi f_j t} + w_j(t) \\ &= \sum_{i,i\neq j} \sum_p \rho_{ij,p}\, x(t - \tau_{ij,p} - t_i[\nu]) e^{\mathrm{j}2\pi \Delta f_{ij} t} + w_j(t) \end{aligned} \tag{7}$$

where $\Delta f_{ij} = f_i - f_j$ is the CFO between nodes i and j, and $w_j(t)$ is an additive noise term. Note that in (7), the complex factor $e^{-\mathrm{j}2\pi f_i \tau_{ij,p}}$ is absorbed into $\rho_{ij,p}$ with no loss of generality. Finally, $y_j(t)$ is sampled at a rate $1/T_s$ with respect to the local clock to obtain:

$$\begin{aligned} y_j[k] &= y_j(kT_s + t_j[\nu]) \\ &= \sum_{i,i\neq j} \sum_p \rho_{ij,p}\, x(kT_s - \tau_{ij,p} - \Delta\theta_{ij}) e^{\mathrm{j}\frac{2\pi}{N}\Delta\lambda_{ij} k} + w_j[k] \end{aligned} \tag{8}$$

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 noise term after sampling and bandpass filtering, and $\Delta\lambda_{ij} = N\Delta f_{ij}/F_s$ corresponds to the normalized CFO. The exact bounds on k are not rigid, as long as k runs over the equivalent of $\frac{T_0}{T_s}$ samples, assuming that $F_s T_0 \in \mathbb{Z}$. In this sampling scheme, $k = 0$ 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. This reduces edge effects that may arise from processing $y_j[k]$.

D. Problem Statement

Given the received signals in (8), 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., $\lim_{\nu\to\infty} \theta_j[\nu] \approx \theta_0$, where θ_0 is a common phase, for all nodes j. For the purpose of practical applications in future 5G networks, this algorithm should be robust against the effects of CFO, propagation delay, and multipath propagation. In addition, its behaviour under SC-FDMA and analog-digital conversions should be well understood due to 5G requirements [11].

III. Background

The DPLL algorithm allows multiple nodes to achieve synchronization by broadcasting a synchronization signal at each 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 with no propagation delays; multipath channels and propagation delays are reintroduced later.

A. Distributed Phase-Locked-Loops

Node i broadcasts its ν^{th} synchronization signal on clock tick $t_i[\nu]$; this signal is received by node j at time $t'_{ij}[\nu]$. Node j computes the weighted average of the time differences

between its clock and the time of reception of the other nodes' synchronization signals, as expressed by:

$$\Delta t_j[\nu] = \sum_{i,i\neq j} \alpha_{ij}\left(t'_{ij}[\nu] - t_j[\nu]\right) \tag{9}$$

where $\alpha_{ij} > 0$ are the weights of this average, with the restriction that $\sum_{i,i\neq j}\alpha_{ij} = 1$. In the absence of propagation delays, which is the underlying assumption in [4, 9, 10], $t'_{ij}[\nu]$ corresponds to the emission time of node i's pulse. In such a case, $t'_{ij}[\nu] = t_i[\nu]$, and the quantity $\Delta t_j[\nu]$ becomes a weighted average of the nodes' phase offsets:

$$\Delta t_j[\nu] = \sum_{i,i\neq j} \alpha_{ij}\,\Delta\theta_{ij}[\nu] \tag{10}$$

It is used to correct the next clock tick:

$$t_j[\nu+1] = t_j[\nu] + T_0 + \epsilon\Delta t_j[\nu] \tag{11}$$

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

$$\theta_j[\nu+1] = \theta_j[\nu] + \epsilon\Delta t_j[\nu] \tag{12}$$

where $0 < \epsilon \leq 1$ scales the correction. Using this approach, all clocks will eventually converge to a common phase value θ_0 in the absence of propagation delays [9, 10]. In practice, node j does not have access to θ_j, as it is the offset with respect to the unknown universal time. An implementation of DPLL must therefore use the update rule in (11).

B. Time of Reception Estimation

Following [2, 4], a simplified model is used here for the transmission/reception of the synchronization signals. Specifically, instead of considering the transmission of the analog signal $x(t)$ in (3), we consider an equivalent discrete-time model that involves the transmission of a discrete sequence $s[n]$, where n is the discrete time index at the rate F_s.

We define $\tilde{s}_{ij}[n]$ as the sequence received by node j when node i transmits $s[n]$ over a flat fading channel $h_{ij} \in \mathbb{C}$. The equation that describes $\tilde{s}_{ij}[n]$ shares many similarities with (7):

$$\tilde{s}_{ij}[n] = h_{ij}\, s[n - q_{ij}]e^{\mathrm{j}\frac{2\pi}{N}\kappa_{ij}n} \tag{13}$$

where $q_{ij}, \kappa_{ij} \in \mathbb{Z}$ are the discrete phase and carrier frequency offsets between nodes j and i, respectively [4]. They are the discrete homologues of $\Delta\theta_{ij}$ and $\Delta\lambda_{ij}$ in (8). In this case, q_{ij} is directly related to the continuous phase offset via the following equation:

$$q_{ij} \approx \frac{1}{T_s}\Delta\theta_{ij} \tag{14}$$

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

Zadoff-Chu (ZC) sequences are often used for synchronization purposes due to their attractive cyclical correlation properties. The odd length-N ZC sequence with non-zero integer root index u is defined as [13]:

$$z_u[n] = e^{-\mathrm{j}\pi un(n+1)/N} \quad , N \text{ odd} \tag{15}$$

An approach to estimate q_{ij} in the presence of CFO was introduced in [2]. It uses a length-$2N$ synchronization sequence built by concatenating two different length-N ZC sequences:

$$s[n] = \begin{cases} z_{-u}[n+N] & -N \leq n < 0 \\ z_u[n] & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

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 can be performed. The linear correlation for bounded sequences $x[n]$ and $y[n]$ is defined as:

$$R_{xy}[l] = \sum_{n\in\mathbb{Z}} y[n]x^*[n+l] \tag{17}$$

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 [4]. Nevertheless, we verified numerically that the properties of ZC sequences carry well when using linear correlations. For that reason, for the remainder of this paper, we use the linear cross-correlation 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_- or z_+, yields a clearly defined peak at ℓ^-_{ij} or ℓ^+_{ij}, respectively [3]:

$$\ell^\pm_{ij} = q_{ij} \mp \frac{\kappa_{ij}}{u} + m_\pm \tag{18}$$

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 $s[n]$. This equation stems from the properties of ZC sequences and their interaction with CFO in cross-correlation [3, 4, 13]. Such peaks are clearly identifiable due to the weak interaction between z_+ and z_- [2].

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

$$y_j[n] = \sum_{i,i\neq j} h_{ij}s[n - q_{ij}]e^{\mathrm{j}\frac{2\pi}{N}\kappa_{ij}n} + w_j[n] \tag{19}$$

Cross-correlating this signal with z_+ and z_- yields multiple peaks located at lags $\ell^\pm_{ij}$, one peak for each transmitting node i. Let $\hat{q}_{j\pm}$ be the estimate of $q_{j\pm}$ defined as the weighted average of the lags l using the magnitude of the cross-correlation $R_{y_j z_\pm}[l]$ as the weighting sequence:

$$\hat{q}_{j\pm} \triangleq \frac{\sum_l l\,|R_{y_j z_\pm}[l]|^\gamma}{\sum_l |R_{y_j z_\pm}[l]|^\gamma} \tag{20}$$

where $\gamma \geq 1$. Since the terms that correspond to the peaks located at $\ell^\pm_{ij}$ dominate the rest of the terms, we can approximate (20) by considering only those terms [4]:

$$\begin{aligned}\hat{q}_{j\pm} &\approx \frac{\sum_l l\,|h_{ij}N\,\delta[l - \ell^\pm_{ij}]|^\gamma}{\sum_l |h_{ij}N\,\delta[l - \ell^\pm_{ij}]|^\gamma} \\ &= \sum_{i,i\neq j}\alpha_{ij}\Big(q_{ij} \mp \frac{\kappa_{ij}}{u} + m_\pm\Big)\end{aligned} \tag{21}$$

where α_{ij} are the weights of this average, defined as:

$$\alpha_{ij} = \frac{|h_{ij}|^\gamma}{\sum_{i,i\neq j} |h_{ij}|^\gamma} \tag{22}$$

By taking a biased average between the two estimations $\hat{q}_{j\text{-}}$ and $\hat{q}_{j+}$, it is possible to recover an unbiased weighted average of q_{ij}:

$$\hat{q}_j = \frac{1}{2}(\hat{q}_{i\text{-}} + \hat{q}_{i+}) + \frac{1}{2} \approx \sum_{i,i\neq j} \alpha_{ij} q_{ij} \tag{23}$$

An estimation $\widehat{\Delta t}_j[\nu] = T_s \hat{q}_j$ can then be obtained from the discrete estimate $\hat{q}_j$ by using (14). Applying the clock-update equation found in (12) with this estimation of $\Delta t_j[\nu]$ leads to converging clock phases in the presence of strong CFO [4].

IV. Proposed DPLL Algorithm Modifications

The previously described DPLL algorithm works well for dense networks [4, 10], where propagation delays and wide-band multipath propagation can safely be ignored. When they are significant enough, nodes may be prevented from achieving time synchronization. In this section, we investigate how these aspects, including analog-digital conversions, affect the performance of the DPLL algorithm, and propose appropriate modifications. We also examine the viability of DPLL when used in conjunction with SC-FDMA.

A. Analog-to-digital Conversion

As described in Sec. II, node i passes the sequence $s[n]$ through a DAC to obtain the signal $x_i(t)$, which is then up-converted and broadcasted. The receiving node j first samples the received signal via an analog-to-digital converter (ADC). Ideally, the sampling times should coincide with the main peaks of the shaping pulses in $x(t)$, but in the presence of a timing offset, they will likely be misaligned, leading to inter-symbol interference (ISI). In such a case, the output of an ideal ADC is a linear combination of $y_j[n]$ with time-shifted copies of itself [12], i.e., a smeared version of $y_j[n]$. Note that we are not concerned with $y_j[n]$ itself but instead with its cross-correlation with $z_\pm[n]$. Since correlations are linear operations, the smearing of the input transfers to the output: misaligned sampling causes smearing of the peaks in $|R_{y_j[n]z\pm}[l]|$. Luckily, the algorithm is largely unaffected by this due to the weighted-average nature of the time of reception estimation.

Indeed, we numerically verified that the algorithm described in the previous section can be used with discrete time signals sampled from the received analog signals, with minor negative impact on the quality of the synchronization.

B. Propagation Delay and Drift Compensation

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

$$\begin{aligned}\Delta t_j[\nu] &= \sum_{i,i\neq j} \alpha_{ij} \left(t_i[\nu] + \tau_{ij} - t_j[\nu]\right) \\ &= \sum_{i,i\neq j} \alpha_{ij} \Delta\theta_{ij}[\nu] + \sum_{i,i\neq j} \alpha_{ij}\tau_{ij} \\ &= \overline{\Delta\theta_j}[\nu] + \overline{\tau}_j \end{aligned} \tag{24}$$

Here, $\overline{\tau}_j$ depends only on the weights α_{ij} in (22), themselves only dependent on the channels, which we assume to be time-invariant. When we apply the DPLL algorithm in the presence of propagation delays, two regimes can be identified: the transient regime ($\overline{\tau}_j \ll \overline{\Delta\theta_j}[\nu]$), where the node's phases start to converge, and the drift regime ($\overline{\tau}_j \approx \overline{\Delta\theta_j}[\nu]$), where the phases of the nodes drift forward over time. These regimes are illustrated in Fig. 2.

The presence of the drift regime is due to the bias term $\overline{\tau}_j$ in (24), which would be zero in the absence of propagation delays. To compensate for the drift, we replace the regular correction $\Delta t_j[\nu]$ in the clock update equation (12) by

$$\begin{aligned}\Delta\Gamma_j[\nu] &= \Delta t_j[\nu] - \frac{1}{Q}\sum_{k=\nu-Q}^{\nu} \Delta t_j[k] \\ &= \Delta t_j[\nu] - \overline{\tau}_j - \frac{1}{Q}\sum_{k=\nu-Q}^{\nu} \overline{\Delta\theta}_j[k] \end{aligned} \tag{25}$$

In $\Delta\Gamma_j[\nu]$ we remove from $\Delta t_j[\nu]$ an estimate of the bias obtained from the last Q values of $\Delta t_j[\nu]$. In practice, $\Delta\Gamma_j[\nu]$ is obtained using the estimated correction $\widehat{\Delta t}_j[\nu]$, as the true quantity $\Delta t_j[\nu]$ is unavailable during synchronization.

Clearly, (25) removes accurately the bias $\overline{\tau}_j$ only if $\overline{\Delta\theta}_j[\nu]$ vanishes after averaging, which is not the case in either regime. If drift compensation is attempted during the transient regime, it fails to properly correct for $\overline{\tau}_j$ as $\overline{\tau}_j \ll \overline{\Delta\theta}_j[\nu]$ since the contribution of $\overline{\Delta\theta}_j[\nu]$ dwarfs the contribution from $\overline{\tau}_j$. This compensation should only be employed during the drift

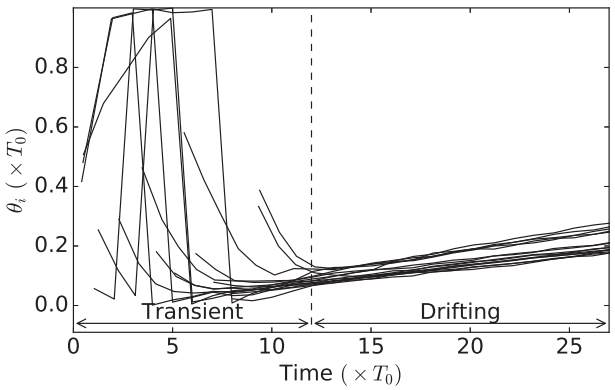

Fig. 2: Evolution of the phases $\theta_i[\nu]$ of multiple interacting nodes i, each applying the DPLL algorithm described in Sec. III-A. Each curve corresponds to a node's phase evolution. Each node starts at different times. See Sec. V for simulation setup.

regime, where $\overline{\tau}_j \approx \overline{\Delta\theta}_j[\nu]$. Therefore, the current regime must be determined before applying (25).

We have observed that the correction terms $\Delta t_j[\nu]$ have a significantly smaller variance in the drift regime than 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 empirical standard deviation of $\Delta t_j[\nu]$ over the last Q estimates, i.e., $\sigma_{Q,i}[\nu]$ is computed using the samples with index $\{\nu, ..., \nu - Q\}$. We consider a node to be in the drift regime if $\sigma_{Q,i}[\nu]$ is below some threshold $\sigma_{\max}$ which can be estimated empirically.

Note that (25) corresponds to a simple highpass filter with impulse response $\left\{\frac{Q-1}{Q}, \frac{-1}{Q}, ..., \frac{-1}{Q}\right\}$. Since we are trying to remove a constant term from (24), a more sophisticated highpass filter with a deep notch at frequency zero could be thought of as a better solution. Our testing showed that popular choices for highpass filters (elliptic, FIR Remez, etc.) did not produce satisfactory results. This is likely caused by the large magnitude of their taps of such filters coupled with the strong feedback between the nodes as they apply DPLL.

C. Effect of Wideband Multipath Propagation

In practical applications, channels are characterized by a discrete multipath channel response:

$$h_{ij}[n] = \sum_p \rho_{ij,p}\, \delta[n - q_{ij,p}] \tag{26}$$

where $\rho_{ij,p} \in \mathbb{C}$ and $q_{ij,p} \in \mathbb{Z}$ are the amplitude and delay, respectively, associated with the p^{th} path between nodes j and i, and $\delta[n]$ is the Kronecker delta function. The estimation method in (20) approximated the cross-correlation $R_{y_j z_\pm}[l]$ by only considering the most prominent peaks. In the presence of multipath propagation, each node i contributes one peak for each path p. We denote those peaks' locations in the cross-correlation as $\ell^{\pm}_{ij,p}$:

$$\ell^{\pm}_{ij,p} = \ell^{\pm}_{ij} + q_{ij,p} \tag{27}$$

where $\ell^{\pm}_{ij}$ is defined in (18). Akin to (20), the weighted average of the cross-correlation can be approximated by only considering the peaks:

$$\begin{aligned}\hat{q}_{j\pm} &= \frac{\sum_l l\, |R_{y_j z_\pm}[l]|^\gamma}{\sum_l |R_{y_j z_\pm}[l]|^\gamma} \\ &\approx \frac{\sum_l l\, |\rho_{ij,p} N\, \delta[l - \ell^{\pm}_{ij,p}]|^\gamma}{\sum_l |\rho_{ij,p} N\, \delta[l - \ell^{\pm}_{ij,p}]|^\gamma} \\ &= \sum_{\substack{i \\ i \neq j}} \alpha_{ij} \left(q_{ij} \mp \frac{\kappa_{ij}}{u} + m_\pm \right) + \sum_{\substack{p,i \\ i \neq j}} a_{ij,p}\, q_{ij,p}\end{aligned} \tag{28}$$

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

Note that the second term in (28) depends on the paths and the topology of the network, which we assume to be static over time. Taking a biased average of $\hat{q}_{i+}$ and $\hat{q}_{i-}$ yields:

$$\begin{aligned}\hat{q}_j = \frac{1}{2}(\hat{q}_{i+} + \hat{q}_{i-}) + \frac{1}{2} &\approx \sum_{i, i \neq j} \alpha_{ij}\, q_{ij} + \sum_{\substack{p,i \\ i \neq j}} a_{ij,p}\, q_{ij,p} \\ &= \overline{\Delta q}_j + \overline{q_{P,j}}\end{aligned} \tag{29}$$

This is similar to the result in (20), with an additional bias term $\overline{q_{P,j}}$. The estimation $\widehat{\Delta t}_j[\nu]$ becomes:

$$\widehat{\Delta t}_j[\nu] \approx T_s \hat{q}_j = \overline{\Delta\theta}_j + T_s \overline{q_{P,j}} \tag{30}$$

This equation has the same form as (24). We conclude that the impact of multipath propagation is no different from that of simple propagation delay. Therefore, the effects of multipath propagation should be properly corrected by the drift compensation algorithm described in the previous section.

D. Use of SC-FDMA

According to 3GPP standards, D2D networks should use SC-FDMA as their multiple access scheme. In SC-FDMA, the length-N_s sequence $s[n]$ is modulated to the length-N_c sequence $c[m]$ through the transformation:

$$c[m] = \text{IDFT}_{N_c} \begin{pmatrix} 0_{1\times a_i} \\ S_k \\ 0_{1\times b_i} \end{pmatrix} \tag{31}$$

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 a_i is assigned to each node i, and allows the nodes 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. To recover $s[n]$, we first look at the expanded form of $c[m]$:

$$\begin{aligned}c[m] &= \frac{1}{N_c} e^{\mathrm{j}\pi \frac{m}{N_c}(2a_i + N_s - 1)} \sum_{n=0}^{N_s-1} s[n]\, \text{sincl}(n - \tfrac{m}{L})\, e^{-\mathrm{j}\pi n(1-\frac{1}{N_s})} \\ &\text{where} \quad \text{sincl}(n) = \begin{cases} N_s & n = 0 \\ \frac{\sin(\pi n)}{\sin(\frac{\pi}{N_s} n)} & \text{otherwise} \end{cases}\end{aligned} \tag{32}$$

and L is such that $N_c = LN_s, L \in \mathbb{N}$. Here, $c[m]$ can be thought of as an interpolated, frequency-shifted version of $s[n]$, with interpolating function the sinc-like function $\text{sincl}(m)$. Downsampling $c[m]$ by a factor of L can directly recover a replica of $s[n]$, barring for the complex exponential factors. These exponentials affect DPLL in an equivalent manner to a CFO; since the DPLL algorithm we use is CFO-robust, the presence of these exponentials is inconsequential.

The CFO-robust DPLL algorithm described in the previous sections can be easily adapted to the presence of SC-FDMA. Indeed, using an L-decimation at the receiver yields a signal of the same form as (19), that is, a linear superposition of the different synchronization signals, multiplied by the CFO-like complex exponentials. Akin to the discussion in Sec. IV-A, downsampling an interpolated sequence with some misalignment due to propagation delay causes ISI. The presence of ISI does not affect the performance significantly due to the weighted average nature of the time of reception estimation.

V. Numerical Results

The proposed modifications to the DPLL algorithm were implemented and tested using numerical simulations. In this

section, we first introduce new metrics to evaluate the performance of the synchronization algorithm, and then present our simulation results which highlight the impact of the modifications proposed in the previous section.

A. Performance Metrics

Several metrics can be used to evaluate the convergence of a synchronization algorithm, a common choice being the standard deviation of the phases θ_j. However, it does not directly reflect each node's capacity to communicate after synchronization. For this task, we define the communication ratio as the fraction of the number of usable links in the network. 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:

$$O_{ij} = \Delta\theta_{ij} + \tau_{ij,b} \tag{33}$$

where $\Delta\theta_{ij}$ is the previously defined phase offset and $\tau_{ij,b}$ is the delay of the path having the highest magnitude across all paths p: $b = \arg\max_p |\rho_{ij,p}|$. Furthermore, let t_p and t_s be the respective duration of the CP and CS. 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:

$$C_{ij} = \begin{cases} 1 & -t_s \leq O_{ij}, O_{ji} < t_p \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

The communication ratio C can then be used to evaluate the synchronization state of the whole network. It is defined as the fraction of all non-zero C_{ij}'s:

$$C = \frac{\sum_{i<j} C_{ij}}{\frac{1}{2}(M^2 - M)} \tag{35}$$

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.

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 reaches the drift regime. As an example,

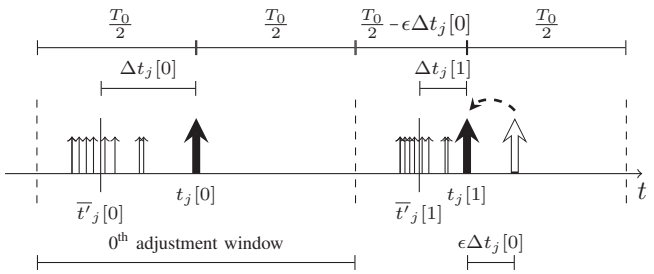

Fig. 3: Graphical representation of the time of reception and transmission of the synchronization signals. Here, node j's signal is represented with the thick arrow, and corresponds to $t_j[\nu]$. The various reception times $t'_{ij}[\nu]$ are represented with the smaller arrows, and their average by a solid line labelled with $\overline{t'}_j[\nu] = \sum_{i,i\neq j} \alpha_{ij} t'_{ij}[\nu]$.

Clock period T_0	3.26 ms
Clock phase θ_i	Uniform $\in [0, 1)$
Correction scaling ϵ	0.5
DC filter length Q	6
Weighting parameter γ	2
SC-FDMA length L	8
SC-FDMA offset a_i	Uniform $\in [0, L-1)$
ZC length N	31

TABLE I: Summary of algorithm parameters.

if we look at Fig. 2, we are interested in the values of $\theta_j[\nu]$ once all nodes have entered the drift regime. Let β_j be the slope of $\theta_j[\nu]$ over the last B samples of $\theta_j[\nu]$, counting down from the end of a simulation. Ideally, this slope should be zero. If, however, drift exists, having the same drift rate can be sufficient for communication, as the phase offset would remain the same over time. The drift can be characterized by $\overline{\beta}$ and σ^2_{β}, which are the arithmetic mean and variance of the slopes across all nodes, respectively:

- $|\overline{\beta}|$: 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. The higher the value, the more dissimilar the individual node drifts β_j are.

The slope β_j will be expressed in milliseconds per second.

B. Results

The simulations described here were performed using *Python* and it's associated *numpy/scipy* libraries. The simulations were done at the physical layer using the technical specifications for LTE Advanced D2D proximity services [11]. These include the pathloss model, probability of line-of-sight (LOS) between nodes, shadowing, multipath parameters, thermal noise power, transmission power, spatial distribution, and others, with one exception: [11] stipulates an average node velocity of $3m/s$, but was omitted for this work, as we consider static networks.

The algorithm presented in this work also involved several parameters. The values for these parameters are presented in Table I, and were empirically determined to be good compromises between computational load, realism, and quality of synchronization.

In order to limit our focus to the aspects described in the previous sections, we consider full duplex transmission. However, the algorithm presented is still applicable to the half duplex case by using, for example, the scheme in [14]. The full-duplex assumption allows us to obtain results that are independent of a particular half-duplex solution.

DC	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
✓	✓	0.63	0.106	0.00712	9.17e-05

TABLE II: Performance metrics for different situations. DC stands for Drift Compensation, and SC for SC-FDMA.

The simulation proceeds as follows. Each network node applies the 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. 3. In each window, the node broadcasts its synchronization signal once while listening for incoming synchronization signals. The received signal during an adjustment window corresponds to $y_j[k]$, defined in (8), from which the clock phase is updated using the estimation $\widehat{\Delta t}_j[\nu]$. This updated clock then affects the time of transmission of the next synchronization signal, and also affects the location and length of the next adjustment window.

We now investigate the impact of our algorithm modifications using numerical simulations. We consider three sets of 1500 simulations with randomized initial values. The performance metrics of each simulation within a set were then averaged together, and can be found in Table II.

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 [4]. 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 C_{std}. Due to its ability to prevent clock drift, SC-FDMA was studied in simulations were DC was concurrently enabled.

SC-FDMA had a slight effect on the synchronization quality, with a small reduction in both C_{avg} and C_{std}. Even so, SC-FDMA does not substantially impair the network's capacity to synchronize, and can therefore be used safely in conjunction with DPLL.

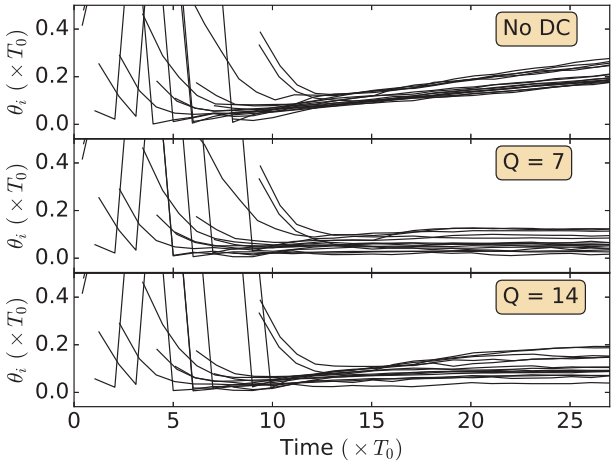

Fig. 4: Time evolution of $\theta_i[n]$ with different DC filter length. Unlike the rest of the results, the simulations presented in these graphs had a clock period of $T_0 = 120\mu s$, to give a better visual effect for the reader.

To highlight the dramatic impact of the DC algorithm, we re-ran the synchronization simulation depicted Fig. 2, but this time with the DC algorithm enabled; this is presented on Fig. 4. Note that the simulations depicted in Fig. 4 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. 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 $\theta_i[\nu]$ are more spread out. We determined empirically that a filter length of $Q = 6$, used for the results presented in Table II, yielded both good stability and good synchronization quality.

VI. Conclusion

This work examined several limitations and aspects that affect a fully distributed synchronization algorithm under constraints specified by 3GPP for D2D applications. Traditional DPLL is not viable in the presence of multipath propagation and significant propagation delays. We demonstrated through numerical simulations using newly designed performance metrics that our modifications to DPLL algorithm proposed in [4] are viable. In particular, the effects of propogation delays and multipath propagation were properly compensated by the DC algorithm. We further demonstrated that the synchronization algorithm could easily be adapted for SC-FDMA.

References

[1] 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.*, 2012, pp. 232–237.

[2] 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.

[3] 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 *Proc. World Telecommun. Congr.*, June 2014, pp. 1–5.

[4] 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.

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

[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, 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] 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.

[9] M. Cremasehi, O. Simeone, and U. Spagnolini, "Distributed Timing synchronization for Sensor Networks with Coupled Discrete-time Oscillators," in *IEEE Commun. Soc. Sensor, Ad Hoc Commun., and Networks*, vol. 2, Sept 2006, pp. 690–694.

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

[11] "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

[12] J. Proakis and M. Salehi, *Digital Communications*, 4th ed. McGraw-Hill, 2008.

[13] D. Chu, "Polyphase Codes with Good Periodic Correlation Properties," *IEEE Trans. Inf. Theory*, vol. 18, no. 4, pp. 531–532, Jul 1972.

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