Distributed Clock Synchronization with application of D2D Communication without Infrastructure

Wanlu Sun, Mohammad Reza Gholami, Erik G. Ström, and Fredrik Brännström
Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden
{*wanlu, moreza, erik.strom, fredrik.brannstrom*}*@chalmers.se*

***Abstract*—This paper investigates the clock synchronization problem for Device-to-Device (D2D) communication without infrastructure. Employing affine models for local clocks, it is proposed a random broadcast based distributed consensus clock synchronization algorithm. In the absence of transmission delays, we theoretically prove the convergence of the proposed scheme, which is further illustrated by the numerical evaluations. On the other hand, when the delays are also taken into account, the proposed approach still performs well. Besides, it is further concluded from the simulations that the proposed scheme is robust against dynamic topologies and scalable to the increased number of devices, and has a fast speed regarding the synchronization error decrease.**

***Index Terms*—Device-to-Device communication, distributed synchronization, consensus algorithm, random broadcast.**

I. Introduction

Recently, Device-to-Device (D2D) communication has emerged as an interesting and important research area. In D2D environment, clock synchronization is critical for both discovery and communication phases. When there is no infrastructure or coverage, the synchronization becomes more challenging for distributed D2D networks, i.e., ad hoc networks, especially in case of mobility. Furthermore, to support D2D communication, local synchronization, where the synchronization is achieved among a network or cluster comprised by devices in proximity, is more convenient than global synchronization, where the synchronization with base station is attained in the whole network [1].

A. Related Work

There has been extensive research on clock synchronization in the context of ad hoc networks during the last few years. Existing protocols can be mainly classified into two categories: reference-based clock synchronization and distributed clock synchronization, depending on whether the reference is needed or not. In the reference-based clock synchronization [2], [3], one device is elected as the reference and a spanning tree is built through the network. All the other devices are required to synchronize to the reference by adjusting their own clocks based on the timing messages received from their parents. This mechanism is sensitive to the changing topology and device failure, and thus not suitable for mobile networks. On the other hand, in a distributed clock synchronization, all devices implement the same algorithm individually without relying on a network hierarchy [4]–[14]. The distributed nature can often result in improved robustness to device failures and mobility.

To utilize the broadcast nature of wireless medium, in the distributed clock synchronization, devices broadcast timing messages which contain the timestamps recorded by the clock of the transmitter. These messages are in turn used to adjust the clocks of the receivers. As briefly described below, there are three different mechanisms regarding the order of the broadcast.

1) Many algorithms assume *periodic broadcast* [13] [14], where devices transmit the synchronization messages every fixed interval of time. This approach requires a set order for all the operations, which might be infeasible in the absence of a centralized controller.

2) A possible variant is the *asynchronous broadcast*. In this protocol, each device has to wait for the arrival of messages from all neighbors before adjusting its local clock [8] [11] [12]. Alternatively, each device sequentially updates its clock whenever it receives a message provided that it can receive one packet from each neighbor during each synchronization round (SR), which is defined as the time interval of a synchronization period. However, the former will bring long latency before the adjustment and thus slow convergence, while the latter is hard to schedule in distributed networks.

3) A more practical alternative is the *random broadcast*, where a device can broadcast at any time in any order. A widely used random broadcast scheme is the contention based transmission, where devices must attend the contention for message broadcast at the beginning of the SR[1], and each device has equal probability to win the contention. Due to its applicability in distributed networks, this mechanism is the technique specified for the clock synchronization in the IEEE 802.11 standard [4] and has been used in some other works [5]–[9] as well. Therefore, in this paper, we will also focus on the random broadcast protocol, i.e., the contention based transmission protocol.

When assuming random broadcast mechanism for message transmission, [4]–[7] propose different converge-to-max synchronization schemes, where a device is only synchronized to the devices with faster clocks. A simple converge-to-max protocol, timing synchronization function (TSF), is presented in [4]. Based on the TSF, various modifications have been made to handle its limitation of scalability and infeasibility

This work has been supported in part by SAFER-Vehicle and Traffic Safety Centre, Project A19. Part of this work has been performed in the framework of the FP7 project ICT-317669 METIS, which is partly funded by the EU. The authors would like to acknowledge the contributions of their colleagues in METIS, although the views expressed are those of the authors and do not necessarily represent the project.

[1]As described in [4], the transmission protocol of timing messages assumes a loosely synchronous beginning of each SR.

in multihop networks [5]–[7], where the MASP scheme in [7] outperforms the others. Nevertheless, as addressed in [15], a common problem for all converge-to-max schemes is the contradiction between the *fastest node asynchronism* and the *time partitioning*.

On the other hand, with a random broadcast mechanism, [9] proposes a distributed consensus protocol for clock synchronization (ATS). In the ATS scheme, an internal common time scale, which dose not need to be the maximum, is achieved in the network through the communications among neighboring devices. In practice, however, the frequencies might be over-adjusted due to the unawareness of clock updates at the transmitter or receiver, and thus the consensus can not be achieved actually.

B. Contributions

In this paper, based on a practical random broadcast mechanism of message transmission, a novel distributed consensus clock synchronization algorithm is proposed for the D2D communication without infrastructure. The proposed approach is fully distributed in the sense that all the devices independently execute the same algorithm without the need of a reference, and thus robust to device failures. In the absence of transmission delays, we theoretically prove the convergence of the proposed scheme, which is further demonstrated by the numerical results. Moreover, by utilizing a threshold, the proposed method shows robustness in the case of transmission delays. Last but not least, the proposed algorithm has a good tradeoff between the speed of convergence and the synchronization errors.

II. System Model and Problem Formulation

A. Network Model

We consider a D2D network represented by a directed graph $\mathcal{G}(k) = (\mathcal{V}, \mathcal{E}(k))$, where the vertex set $\mathcal{V}$ contains N mobile devices and the edge set $\mathcal{E}(k)$ is defined as the set of available directed communication links at discrete time index k, i.e., $(i, j) \in \mathcal{E}(k)$ if device j sends information to device i during kth synchronization round[2], which is defined as the time interval of a synchronization period. $\forall i \in \mathcal{V}$, we also assume $(i, i) \in \mathcal{E}(k)$ for $\forall k$. The symbol $\mathcal{V}_i(k) = \{j|(i, j) \in \mathcal{E}(k)\}$ denotes the set of neighbors of device i during kth synchronization round, and $\mid \mathcal{V}_i(k) \mid$ its cardinality.

B. Clock Model

Each device in the network is equipped with a physical clock that has its frequency and offset. That is, for the ith device, we have the physical time[3]

$$T_i(t) = f_i t + \theta_i, \quad \forall i \in \mathcal{V}, \tag{1}$$

where t is the perfect time, f_i indicates the physical clock frequency, and θ_i denotes the physical clock offset. Note that f_i and θ_i are both determined by the physical clock and cannot be measured or adjusted manually. Besides, each device i also maintains a logical clock, whose value is called logical time $C_i(t)$ and can be modified. The logical time $C_i(t)$ represents the synchronized time of device i. It is a function of current physical time $T_i(t)$ and can be calculated as follows:

$$\begin{aligned} C_i(t) &= \alpha_i T_i(t) + \beta_i && (2)\\ &= \alpha_i f_i t + \alpha_i \theta_i + \beta_i && (3)\\ &= \hat{f}_i t + \hat{\theta}_i, && (4) \end{aligned}$$

where α_i ($\alpha_i > 0$) and β_i are control parameters updated by the synchronization algorithm, and

$$\hat{f}_i \triangleq \alpha_i f_i \tag{5}$$

$$\hat{\theta}_i \triangleq \alpha_i \theta_i + \beta_i \tag{6}$$

represent the logical clock frequency and logical clock offset, which can be adjusted. The initial values of α_i and β_i are set to $\alpha_i = 1$ and $\beta_i = 0$, respectively. In this way, the goal is to synchronize the clocks in the network such that the logical clocks of different devices have the same (or very similar) values for any instant of perfect time.

C. Problem Formulation

In this study, rather than synchronizing all clocks to a real reference clock, we aim at attaining an internal consensus among logical clocks through local interactions. Namely,

$$\lim_{t \to +\infty} \frac{C_i(t)}{C_{\mathrm{v}}(t)} = 1, \quad \forall i \in \mathcal{V}, \tag{7}$$

where

$$C_{\mathrm{v}}(t) = f_{\mathrm{v}} t + \theta_{\mathrm{v}}, \quad f_{\mathrm{v}} > 0, \tag{8}$$

denotes the virtual consensus clock. For each device i, the asymptotical consensus (7) is equivalent to concurrently achieving the following two consensus equations:

$$\lim_{t \to +\infty} \hat{f}_i = f_{\mathrm{v}}, \tag{9}$$

$$\lim_{t \to +\infty} \hat{\theta}_i = \theta_{\mathrm{v}}. \tag{10}$$

Note that f_{v} and θ_{v} do not need to be the average value of $\{f_1, f_2, ..., f_N\}$ and $\{\theta_1, \theta_2, ..., \theta_N\}$, respectively. Their values are decided by $\mathcal{E}(k)$, $\{f_1, f_2, ..., f_N\}$ and $\{\theta_1, \theta_2, ..., \theta_N\}$ together. In fact, the values of f_{v} and θ_{v} are not important, since what really matters is that all clocks converge to one common value.

D. Transmission of Synchronization Messages

In this paper, we consider a popular random broadcast scheme, i.e., the contention based broadcast mechanism. More specifically, for each device [4],

1) at the beginning of each SR, calculates a random delay that is uniformly distributed in the range between zero and twice aCWmin $\times$ aSlotTime (which are constants and specified in [4]);

2) waits for the period of the random delay while decrementing the random delay timer;

3) cancels the remaining random delay and the pending beacon transmission if a beacon arrives before the random delay timer has expired;

[2]As described in [4], the transmission protocol of timing messages assumes a loosely synchronous beginning of each synchronization round.

[3]Here we assume an affine function for the physical clock model. The more accurate models, which include randomness and higher order terms, might be considered in future studies.

4) sends a timing message if the random delay has expired.

Remark 1. *Owing to the hidden node problem, it is possible for one device to receive multiple messages during one SR. In this case, the device will just keep the first received packet and discard the later packets. In other words,* $\mid \mathcal{V}_i(k) \mid$ *can only be* 1 *or* 2 *for* $\forall i \in \mathcal{V}$.

Remark 2. *Like most of literature (e.g., [2], [3], [8]–[14]), the MAC layer time stamping is utilized to largely reduce the effects of transmission delays.*

III. The Proposed Synchronization Algorithm: CoSyn

In this section, we propose a novel consensus based distributed clock synchronization (CoSyn) scheme for the D2D communication without infrastructure, where the logical offsets and logical frequencies are jointly adjusted by updating the control parameters. We take accuracy as well as convergence speed into account in designing the algorithm. Section III-A derives the basic design principles of the CoSyn scheme. Then the complete procedures of the CoSyn scheme are summarized in Section III-B. Finally, the convergence analysis is elaborated by Theorem 1 in Section III-C.

A. Design Principles of the CoSyn Scheme

Suppose device i ($i \in \mathcal{V}$) receives timing messages from one neighbor device j ($j \in \mathcal{V}$). In the absence of delays, by [16, Thm. 1], if device i updates its logical frequency and logical offset as

$$\hat{f}_i^+ = \frac{1}{2}\left(\hat{f}_i + \hat{f}_j\right), \tag{11}$$

$$\hat{\theta}_i^+ = \frac{1}{2}\left(\hat{\theta}_i + \hat{\theta}_j\right), \tag{12}$$

respectively, where $(\cdot)^+$ indicates the new value of the corresponding variable, then under some mild connectivity conditions (see [16]), the consensus (9) and (10) will be achieved with exponential speed. Note that, as explained in Remark 1, we only have access to one neighbor device in the update procedure.

In practice, however, the implementation of (11) and (12) is not straightforward. There are mainly three problems.

Firstly, due to the lack of information about f_i and θ_i, we can neither obtain $\hat{f}_i$ or $\hat{\theta}_i$ nor tune them directly. What we can do is to modify the control parameters α_i and β_i by utilizing the logical clock, which will then adjust $\hat{f}_i$ and $\hat{\theta}_i$ automatically through (5) and (6).

Secondly, in order to execute the update of the logical frequency as in (11), it can be shown that the following two requirements are necessary: *1)* device i receives at least two timing messages from device j; *2)* logical clocks of both device i and device j are not adjusted between the two receptions. Obviously the conditions can be satisfied if we make device i not to adjust its logical clock (i.e., not update its control parameters) until it receives the second message from device j. Whereas, this method will cause fairly slow convergence speed, especially for dense networks. Therefore, in our algorithm, device i will adjust its logical clock whenever it receives a timing message for speeding up the convergence.

Nonetheless, if the clock is adjusted by our method whenever a message is received, the third problem will be introduced. Specifically, it can be proved that the update rule (12) is not satisfied in the proposed scheme.

In the following, we will propose certain operations for each device, and then discuss the convergence of these operations.

Suppose device i receives a timing message from one neighbor device j at the kth synchronization round, then there is only three different cases for this message.

I. This is the first time for device i to receive a message from device j.

II. This is not the first time for device i to receive a message from device j, and the last received message is in the lth ($0 \leq l < k$) synchronization round. But one or both of device i and j have adjusted its logical clock between the two synchronization rounds.

III. This is not the first time for device i to receive a message from device j, and the last received message is in the lth synchronization round. Neither of device i and j has adjusted its logical clock between the two synchronization rounds.

For both Case I and Case II, device i will only update its control parameter β_i as

$$\beta_i^+ = \beta_i + \frac{1}{2}\left(C_j(t_{jk}) - C_i(t_{jk})\right), \tag{13}$$

and maintain α_i unchanged, i.e., $\alpha_i^+ = \alpha_i$. Here t_{jk} denotes the perfect time when device j sampled its logical clock in the lth synchronization round. The idea behind the operation (13) is that, when the two devices have the same logical frequencies, the operation (13) can perfectly achieve the update rule (12). While for Case III, device i will jointly update its control parameters α_i and β_i as

$$\alpha_i^+ = \frac{1}{2}\left(1 + \frac{C_j(t_{jk}) - C_j(t_{jl})}{C_i(t_{jk}) + \Delta_j - C_i(t_{jl})}\right)\alpha_i, \tag{14}$$

$$\begin{aligned}\beta_i^+ = {} & \frac{1}{2}\left(1 + \frac{C_j(t_{jk}) - C_j(t_{jl})}{C_i(t_{jk}) + \Delta_j - C_i(t_{jl})}\right)(\beta_i + \Delta_j) \\ & + \frac{1}{2}\left(C_j(t_{jk}) - \frac{\left(C_j(t_{jk}) - C_j(t_{jl})\right)\left(C_i(t_{jk}) + \Delta_j\right)}{C_i(t_{jk}) + \Delta_j - C_i(t_{jl})}\right),\end{aligned} \tag{15}$$

respectively, and

$$\Delta_j = \begin{cases} \dfrac{C_i(t_{jl}) - C_j(t_{jl})}{2}, & \text{if } \alpha_i \text{ was not adjusted in } l\text{th round} \\ 0, & \text{if } \alpha_i \text{ was adjusted in } l\text{th round.} \end{cases} \tag{16}$$

The idea behind the operations (14) and (15) is that, when device i receives the two timestamps from device j consecutively, i.e., $k = l+1$, the operations (14) and (15) can perfectly achieve the update rules (11) and (12).

Remark 3. *Even though the MAC layer timestamping can reduce the transmission delays to a large extent, there could still be some delays remaining. When these remaining delays are also taken into account, we can consider a threshold for the adjustment to reduce the influence of delays. Assume device* j *broadcasts a timing message at* t *and device* i *receives it after delay* δ_{ij}*, then device* i *will not use it for adjusting its own logical clock unless* $|C_i(t + \delta_{ij}) - C_j(t)| > \kappa$*, where* κ *is the threshold. The value of* κ *is the tradeoff between the*

speed of the synchronization error decrease and the robustness against delays, and κ can for example be set as the mean value of delays.

B. Procedures of the CoSyn Scheme

In order to implement the CoSyn scheme, each device (say device i) should maintain a table $\mathbb{A}_j$ for each neighbor j, where $\mathbb{A}_j$ is initially empty. $\mathbb{A}_j$ at device i will be updated whenever device i receives a message from device j. Moreover, $\mathbb{A}_j$ contains the information related to device j and consists of the six fields as follows.

- j: the neighbor's identity;
- $\tilde{C}_j$: the last timestamp received from device j;
- $\tilde{S}_j$: the change counter value of device j when $\tilde{C}_j$ was received;
- $\tilde{C}_{ij}^+$: the updated logical clock of device i as the result of receiving $\tilde{C}_j$;
- $\tilde{S}_{ij}$: the change counter value of device i after updating its logical clock when $\tilde{C}_j$ was received;
- $\tilde{\Delta}_j$: a variable used for updating device i's logical clock when receiving device j's message, i.e., the operations of Case III in Section III-A.

A flowchart of the proposed CoSyn scheme is summarized in Fig. 1, where the following two equations are required:

$$\alpha_i^+ = \frac{1}{2}\left(1+\frac{C_j-\tilde{C}_j}{C_i-\tilde{C}_{ij}^+}\right)\alpha_i, \tag{18}$$

$$\beta_i^+ = \frac{1}{2}\left(1+\frac{C_j-\tilde{C}_j}{C_i-\tilde{C}_{ij}^+}\right)(\beta_i+\tilde{\Delta}_j) + \frac{1}{2}\left(C_j-\frac{\left(C_j-\tilde{C}_j\right)\left(C_i+\tilde{\Delta}_j\right)}{C_i-\tilde{C}_{ij}^+}\right). \tag{19}$$

C. Convergence of the CoSyn Scheme

Before the discussion on the convergence, we first introduce some terms [16] regarding the graph theory.

Term 1. a vertex i of a directed graph is a **root** of the graph if for each other vertex j of this graph, there is a path from i to j.

Term 2. a **rooted graph** is a graph which possesses at least one root.

Term 3. by the **composition** of two directed graphs $\mathcal{G}(p)$, $\mathcal{G}(q)$ with the same vertex set $\mathcal{V}$ we mean the graph $\mathcal{G}(p)\circ\mathcal{G}(q)$ with the same vertex set $\mathcal{V}$ and edge set defined such that (i,j) is an edge of $\mathcal{G}(p)\circ\mathcal{G}(q)$ if and only if for some vertex r, (r,j) is an edge of $\mathcal{G}(q)$ and (i,r) is an edge of $\mathcal{G}(p)$.

Term 4. a finite sequence of directed graphs $\mathcal{G}(1)$, $\mathcal{G}(2)$,..., $\mathcal{G}(q)$ with the same vertex set is **jointly rooted** if the composition $\mathcal{G}(q)\circ\mathcal{G}(q-1)\circ\cdots\circ\mathcal{G}(1)$ is rooted.

Term 5. an infinite sequence of graphs $\mathcal{G}(1)$, $\mathcal{G}(2)$, $\cdots$ with the same vertex set is **repeatedly jointly rooted by subsequences of length** q if there is a positive finite integer q for which each finite sequence $\mathcal{G}(qm+1)$, $\mathcal{G}(qm+2)$, $\cdots$, $\mathcal{G}(qm+q)$, $m>0$, is jointly rooted.

As mentioned above, the adjustment of the logical frequency requires at least two timestamps, which cannot be implied by the graph $\mathcal{G}(k)$. Therefore, we consider a new directed graph $\mathcal{F}(k)=(\mathcal{V},\tilde{\mathcal{E}}(k))$. Regarding the edge set $\tilde{\mathcal{E}}(k)$, we define $(i,j)\in\tilde{\mathcal{E}}(k)$ if device i receives a message from device j at kth SR by following the condition of Case III, i.e., device i adjusts its logical frequency based on device j's information. Also, $\forall i\in\mathcal{V}$, it is assumed $(i,i)\in\tilde{\mathcal{E}}(k)$ for $\forall k$. Besides, the set $\tilde{\mathcal{V}}_i(k)$ is defined as $\tilde{\mathcal{V}}_i(k)=\{j|(i,j)\in\tilde{\mathcal{E}}(k)\}$, and $|\tilde{\mathcal{V}}_i(k)|$ indicates its cardinality.

The convergence analysis of the CoSyn scheme is given by the following Theorem 1.

Theorem 1. *Assume*

a) all devices can broadcast in any order as long as an infinite sequence of graphs $\mathcal{F}(1)$, $\mathcal{F}(2)$, $\cdots$ is repeatedly jointly rooted by subsequences of length q;[4]

b) the transmission delays are negligible, i.e., the transmitter and receivers record the timestamps from their local logical clocks simultaneously.

Then, if each device updates its control parameters as (14) and (15), or (13) depending on which type the received message belongs to, the asymptotical consensus (9) and (10) can be achieved, and in a further way, (7) can be achieved as well.

Proof: Plug (4) (5) and (6) into (14), then (14) is expanded as (17) (at the bottom of this page). In this way, by following (5), the updated frequency $\hat{f}_i^+$ is

$$\hat{f}_i^+ = \alpha_i^+ f_i = \frac{1}{2}\left(1+\frac{\hat{f}_j}{\hat{f}_i}\right)\frac{\hat{f}_i}{f_i}f_i = \frac{1}{2}\left(\hat{f}_i+\hat{f}_j\right), \tag{20}$$

which coincides with the update rule (11). Correspondingly, the consensus equation (9) can be achieved with an exponential speed.

The proof of the consensus equation (10) is not straightforward, since (12) will not be attained based on the proposed operations. Due to the space limitation, the rigorous proof is not given here and will be reported in our future work. The key idea is to show that the additional introduced terms in the update rule (12) will converge to 0 with evolving time. ■

IV. Simulation Results

In this section, simulation results are presented to compare the performance of the proposed CoSyn scheme with the following two synchronization protocols: ATS (with ρ_η=0.2, ρ_o=0.2 and ρ_v=0.2) in [9] and MASP in [7].

[4]This assumption is realistic in the contention based transmission mechanism. Moreover, the repeatedly jointly rooted property of $\mathcal{F}(1)$, $\mathcal{F}(2)$, $\cdots$ implies the repeatedly jointly rooted property of $\mathcal{G}(1)$, $\mathcal{G}(2)$, $\cdots$.

$$\alpha_i^+ = \frac{\alpha_i}{2}\left(1+\frac{(\hat{f}_j t_{jk}+\hat{\theta}_j)-(\hat{f}_j t_{jl}+\hat{\theta}_j)}{\hat{f}_i t_{jk}+\hat{\theta}_i+\frac{C_j(t_{jl})-C_i(t_{jl})}{2}+\frac{C_i(t_{jl})-C_j(t_{jl})}{2}-\left(\hat{f}_i t_{jl}+\hat{\theta}_i\right)}\right) = \frac{\alpha_i}{2}\left(1+\frac{\hat{f}_j}{\hat{f}_i}\right) = \frac{1}{2}\left(1+\frac{\hat{f}_j}{\hat{f}_i}\right)\frac{\hat{f}_i}{f_i}. \tag{17}$$

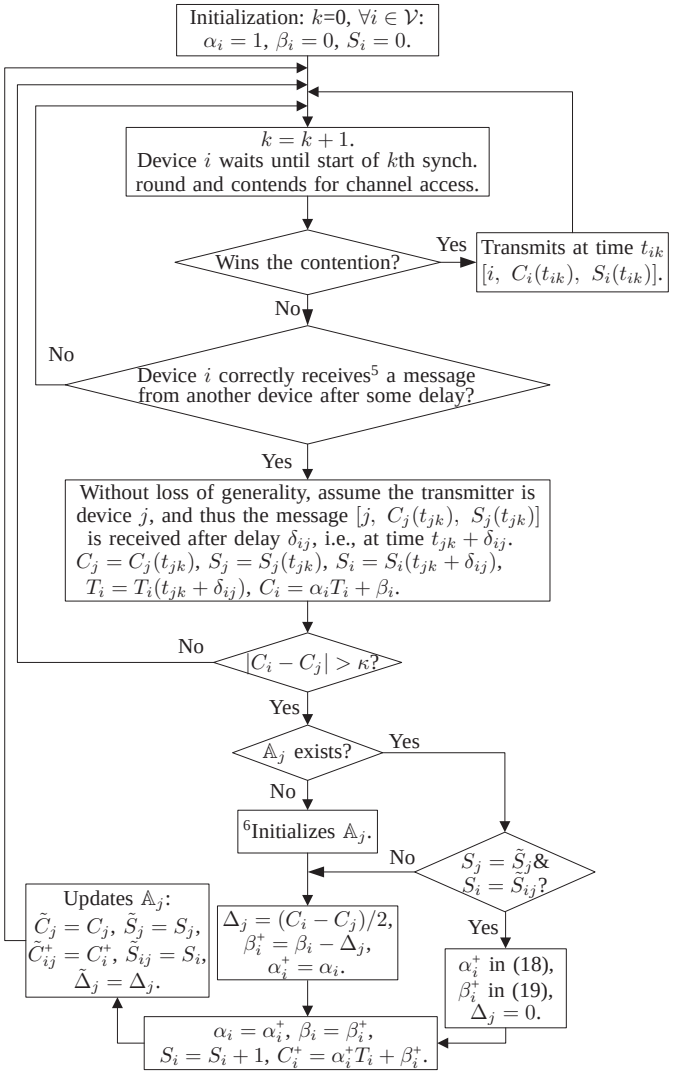


Figure 1. The flowchart of the proposed CoSyn scheme.

We consider a D2D network with N mobile devices, where devices are randomly placed in a square-shaped region. The size of the area is 1000m $\times$ 1000m, and every device has a fixed transmission range of 300m. Moreover, the mobility of devices is totally random, which means the network topologies are uncorrelated at different synchronization rounds. The clock frequencies are uniformly and randomly selected from the range $[0.9999, 1.0001]$, following IEEE 802.11 protocol [4]. Also, the initial clock values are uniformly and randomly chosen from the range $[-800, 800]$ microseconds. Besides, as defined in [4], the period of one synchronization round is 100×10^3 microseconds, and the backoff time in the contention based protocol is uniformly distributed in the range $[0, 1500]$ microseconds. Furthermore, as explained above, there still remains some delays even though the MAC layer timestamping

[5]The correct receiving means that the message is received without collision. On the other hand, we assume that any collision will lead to packet loss.

[6]By 'initializes', we mean that some space is allocated for $\mathbb{A}_j$ and the identity 'j' is filled in.

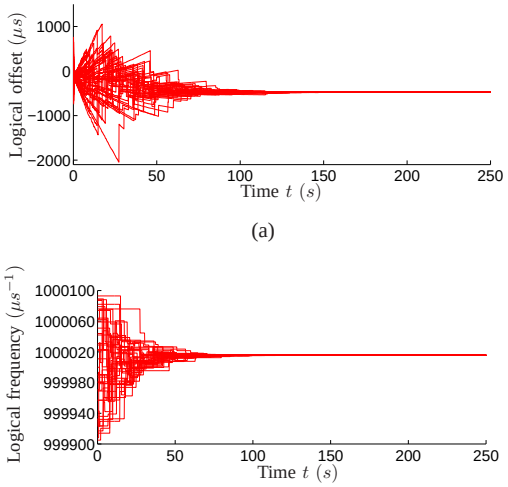


Figure 2. Convergence evaluation of the logical offsets (Fig. 2(a)) and logical frequencies (Fig. 2(b)) in the proposed CoSyn scheme for 40 devices, with $d = 0$ (i.e., no delay).

is applied. In our simulations, these delays are modeled by a uniform distribution within the range $[0, 2d]$ microseconds, and we set the threshold $\kappa = d$. The performance metrics in our simulations are the maximum synchronization errors and average synchronization errors defined by (21) and (22) respectively:

$$e_{\max}(t) \triangleq \max_{i=1:N-1} \max_{j=i+1:N} |C_i(t) - C_j(t)|, \tag{21}$$

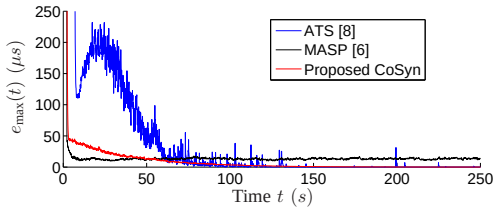
$$e_{\text{avg}}(t) \triangleq \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} |C_i(t) - C_j(t)|. \tag{22}$$

For each result in the following figures, we average the errors over 20 different simulation runs.

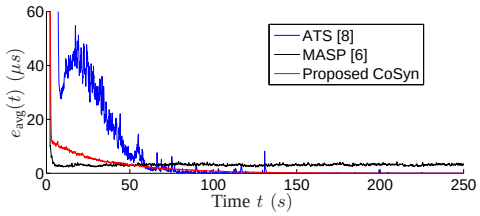
Fig. 2 illustrates the convergence of the logical offsets θ_i, $i = 1, \cdots, N$, (Fig. 2(a)) and logical frequencies f_i (Fig. 2(b)) for the proposed CoSyn scheme in the absence of delays. It is shown that both logical offsets and logical frequencies will indeed converge to a common value respectively, which supports the Theorem 1 from a numerical perspective.

Fig. 3 shows the synchronization error versus the time evolution t in the absence of delays, i.e., $d = 0$. It is revealed that the evaluations of both maximum synchronization errors (Fig. 3(a)) and average synchronization errors (Fig. 3(b)) have similar trends. With time evolving, although MASP exhibits fast decrease in terms of the synchronization errors, it does not converge to zero error even if there is no delay. On the other hand, both ATS and the proposed CoSyn achieve consensus asymptotically. The synchronization errors of the CoSyn method decay much faster compared to ATS, especially during the first 50 seconds.

Fig. 4 shows the synchronization error as a function of time t when $d = 2$. It is depicted that the error of ATS increases significantly with time, which implies ATS becomes ineffective under the scenario with delays. While, both MASP and CoSyn attain a roughly steady state, and CoSyn obviously outperforms MASP.



(a)



(b)

Figure 3. Synchronization error versus time evolution, with $d = 0$ (i.e., no delay) and $N = 50$. (a) Maximum synchronization error (21). (b) Average synchronization error (22).

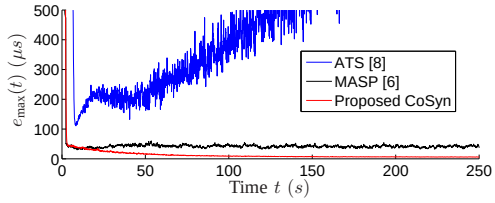


Figure 4. Maximum synchronization error (21) versus time evolution, with $d = 2$ and $N = 50$.

Fig. 5 shows the synchronization error versus the number of devices N when $d = 2$. We observe that the error of ATS is not stable with the increased N. On the other hand, even though both MASP and CoSyn are scalable to the number of devices, CoSyn displays a distinct improvement.

Fig. 6 shows the synchronization error versus the delay level d. The error of ATS is boosted when increasing d, which again reveals its sensibility to delays. Moreover, MASP exhibits a moderate increase of the synchronization error, but the error is not close to zero even if $d = 0$. Compared to ATS and MASP, the proposed CoSyn has slower error increase with d, which indicates the robustness of the proposed CoSyn scheme against delays.

V. Conclusion

In this paper, we propose a novel distributed consensus based clock synchronization scheme—CoSyn—for D2D communication without infrastructure, in which the timing massages are broadcast in a random manner. In the absence of transmission delays, we prove the consensus of both logical frequencies and logical offsets for the CoSyn approach. Furthermore, as illustrated in the simulations, the CoSyn technique shows fast convergence, good scalability, and robustness to dynamic topologies and transmission delays.

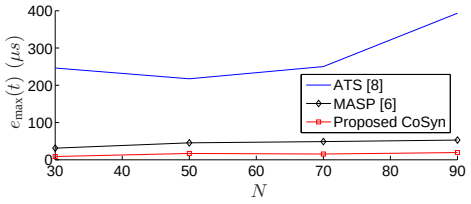


Figure 5. Maximum synchronization error (21) versus N at $t = 50s$, with $d = 2$.

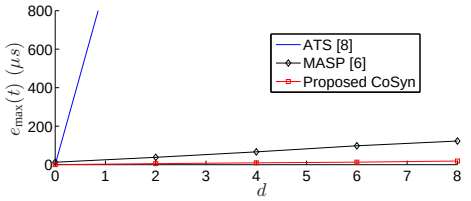


Figure 6. Maximum synchronization error (21) versus the delay level d at $t = 50s$, with $N = 50$.

References

[1] Ericsson, ST-Ericsson, "3GPP R1-132029, Synchronization Procedures for D2D Discovery and Communication," Tech. Rep., May 2013, [Online]: http://www.3gpp.org/.

[2] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. SenSys 03*, Nov. 2003, pp. 138–149.

[3] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proc. 2004 International Conf. Embedded Networked Sensor Systems*, pp. 39–49.

[4] IEEE Std 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification*, 1999 edition.

[5] D. Zhou and T. H. Lai, "A scalable and adaptive clock synchronization protocol for IEEE 802.11-Based multihop ad hoc networks," in *Proc. IEEE Mobile Adhoc and Sensor Conference*, Nov. 2005, pp. 558–565.

[6] D. Zhou and T. H. Lai, "An accurate and scalable clock synchronization protocol for IEEE 802.11-Based multihop ad hoc networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1797–1808, Dec. 2007.

[7] H. K. Pande, S. Thapliyal, and L. C. Mangal, "A new clock synchronization algorithm for multi-hop wireless ad hoc networks," in *Proc. IEEE WCSN*, Dec. 2010, pp. 1–5.

[8] R. Solis, V. S. Borkar, and P. R. Kumar, "A new distributed time synchronization protocol for multihop wireless networks," in *Proc. IEEE Decision and Control Conference*, Dec. 2006, pp. 2734–2739.

[9] L. Schenato and F. Fiorentin, "Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks, *Automatica*, vol. 47, no. 9, pp. 1878–1886, Sep. 2011.

[10] M. K. Maggs and S. G. O'Keefe, "Consensus clock synchronization for wireless sensor networks," *IEEE sensors Journal*, vol. 12, no. 6, pp. 2269–2277, Jun. 2012.

[11] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," in *Proc. IEEE IPSN*, Apr. 2009, pp. 17–48.

[12] Q. Li and D. L. Daniela, "Global clock synchronization in sensor networks," *IEEE Trans. on Computers*, vol. 55, no. 2, pp. 214–226, Feb. 2006.

[13] A. C. Pinho, D. R. Figueiredo, and F. M. G. Franga, "A robust gradient clock synchronization algorithm for wireless sensor networks," in *Proc. IEEE COMSNETS*, Jan. 2012, pp. 1–10.

[14] W. Su, and I. F. Akyildiz, "Time-diffusion synchronization protocol for wireless sensor netowrks," *IEEE/ACM Trans. on Networking*, vol. 13, no. 2, pp. 384–397, Apr. 2005.

[15] J. H. Chiang, and T. Chiueh, "Accurate clock synchronization for IEEE 802.11-based multi-hop wireless networks," in *Proc. IEEE ICNP*, Oct. 2009, pp. 11–20.

[16] M. Cao, A. S. Morse, and B. D. O. Anderson, "Reaching a consensus in a dynamically changing environment: convergence rates, measurement delays, and asynchronous events" *SIAM Journal on Control and Optimization*, vol. 47, no. 2, pp. 601–623, Mar. 2008.