CHALMERS

Chalmers Publication Library

Network Synchronization for Mobile Device-to-Device Systems

This document has been downloaded from Chalmers Publication Library (CPL). It is the author´s version of a work that was accepted for publication in:

IEEE Transactions on Wireless Communications (ISSN: 1536-1276)

Citation for the published paper:
Wanlu, S. ; Brännström, F. ; Ström, E. (2017) "Network Synchronization for Mobile Device-to-Device Systems". IEEE Transactions on Wireless Communications, vol. 65(3), pp. 1193-1206.

http://dx.doi.org/10.1109/TCOMM.2016.2639504

Downloaded from: http://publications.lib.chalmers.se/publication/249118

Notice: Changes introduced as a result of publishing processes such as copy-editing and formatting may not be reflected in this document. For a definitive version of this work, please refer to the published source. Please note that access to the published version might require a subscription.

Chalmers Publication Library (CPL) offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all types of publications: articles, dissertations, licentiate theses, masters theses, conference papers, reports etc. Since 2006 it is the official tool for Chalmers official publication statistics. To ensure that Chalmers research results are disseminated as widely as possible, an Open Access Policy has been adopted.
The CPL service is administrated and maintained by Chalmers Library.

(article starts on next page)

Network Synchronization for Mobile Device-to-Device Systems

Wanlu Sun, Fredrik Brännström, *Member, IEEE*, and Erik G. Ström, *Senior Member, IEEE*

***Abstract*— This paper studies the synchronization problem for mobile cellular device-to-device (D2D) networks. Depending on the number of devices that are in coverage of the base station, the D2D environment can be divided into three categories where the partial-coverage and out-of-coverage are challenging scenarios and thus are the focus of this paper. First, we discuss five main challenges imposed on the synchronization problem in mobile D2D networks. More specifically, there are different challenges in the two coverage scenarios, since they do not have exactly the same synchronization objectives. Second, we propose a low-complexity Adaptive distRibuted nEtwork Synchronization (ARES) algorithm to address the five challenges. The design principles and the theories behind the ARES scheme are also analyzed in detail. Finally, we provide comprehensive simulations to evaluate different synchronization schemes, where the proposed ARES mechanism shows very promising performance.**

***Index Terms*— Device-to-device communication, network synchronization, consensus algorithms.**

I. Introduction

RECENTLY, device-to-device (D2D) communication has emerged as an interesting and important research area. In a D2D environment, time agreement, i.e., a common notion of time, is critical for both device discovery and synchronous transmission [1]. Moreover, in D2D communications, time-related channel access mechanisms, e.g., self-organized time division multiple access and slotted ALOHA, require time agreement as well. The procedure to achieve time agreement across a network is called network synchronization. Although synchronization strategies have been extensively researched for conventional wireless networks such as sensor networks and ad hoc networks, very few of them, e.g., [2]–[5], are designed for D2D networks where new requirements and challenges appear.

Compared to WSNs, which typically are defined as networks with small, inexpensive devices that are statically deployed, the devices in cellular D2D networks are more powerful, which allows for more computationally complex synchronization methods. However, the required synchronization performance is also typically higher, especially when D2D communication is used to provide ultra-reliable services [6, Ch. 4]. Moreover, D2D networks can have quite high device mobility. The perhaps most important example is when the devices are road vehicles (cars, trucks, buses, etc.). In fact, low-latency, vehicle-to-everything (V2X) communication is considered an important use case for D2D. Indeed, standardization in 3GPP is on-going to include this functionality in future releases of LTE (4G) [7], and V2X over D2D is widely considered as an important application for 5G [6, Ch. 4]. A vehicular device is not necessarily severely constrained in form factor, complexity, or power consumption. However, cost remains an important issue, especially for car-mounted equipment.

To summarize, compared to WSNs, D2D networks typically have *(i)* more powerful devices, *(ii)* higher device mobility, *(iii)* faster changing network topologies, and *(iv)* higher requirements on synchronization. Although, WSN synchronization algorithms can be used for D2D networks, these differences *(i)-(iv)* motivate us to study synchronization methods specifically tailored for D2D networks.

In D2D systems, depending on how many devices are in coverage of the base station (BS), synchronization scenarios can be classified into the following three scenarios as shown in Fig. 1: *1)* **all-in-coverage scenario** where all the devices are in coverage of the BS; *2)* **partial-coverage scenario** where only a fraction of devices are in network coverage; and *3)* **out-of-coverage scenario** where all the devices are out of network coverage. For the all-in-coverage scenario, synchronization is not so challenging since the BS can directly send timing information to all the devices. Hence, in this work, we focus on the last two scenarios.

A. Related Work

According to what type of information is transmitted, synchronization techniques can be divided into three categories: *1)* pulse-based synchronization [3], [8], which is to synchronize the frequencies of oscillators based on emitting pulses at the physical layer; *2)* sequence-based synchronization [2], [9], which is to find the beginning of a received symbol or frame by correlating specifically designed sequences; and *3)* timestamp-based synchronization [5], [10]–[17], which is to synchronize clocks by transmitting locally recorded clock values. In fact, neither pulse-based nor sequence-based synchronization can achieve agreement on time values without

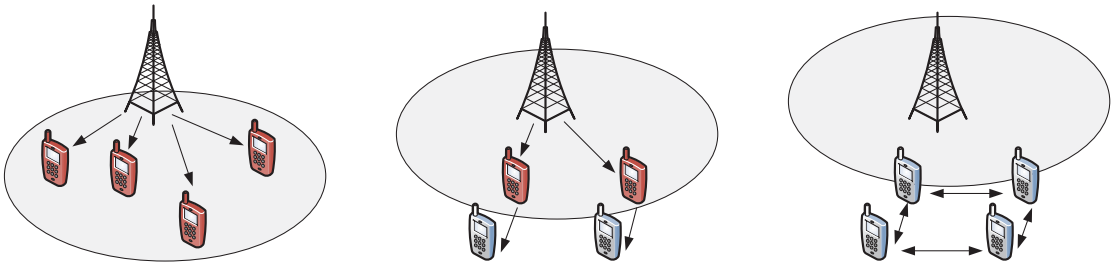
Manuscript received March 4, 2016; revised September 29, 2016 and December 6, 2016; accepted December 7, 2016. Date of publication December 14, 2016; date of current version March 15, 2017. The research was funded by the Swedish Governmental Agency for Innovation Systems (VINNOVA), FFI - Strategic Vehicle Research and Innovation, under Grant No. 2014-01387. 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 associate editor coordinating the review of this paper and approving it for publication was H. Steendam.

The authors are with the Department of Signals and Systems, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden (e-mail: wanlu@chalmers.se; fredrik.brannstrom@chalmers.se; erik.strom@chalmers.se).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2016.2639504

0090-6778 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.



(a) All-in-coverage (b) Partial-coverage (c) Out-of-coverage

Fig. 1. Three synchronization scenarios in D2D networks.

additional information [18]. More specifically, pure pulse-based synchronization can only achieve frequency synchronization by by adjusting the frequencies of emitting pulses, and sequence-based synchronization can only find the beginning of a transmitted packet by correlating specifically designed sequences. However, for some aspects of D2D communications, e.g., time-related channel access, a common notion of time is indispensable. Therefore, in this paper, we consider timestamp-based synchronization.

To combat the transmission delays of timestamps, most synchronization protocols employ medium access control (MAC) layer timestamping [13]. Here, MAC layer timestamping means that the current timestamp is written into the message payload just before the first bit of the packet is sent to the physical layer at the transmitter, and the timestamp at the receiver side is recorded immediately after the first bit has arrived at the MAC layer. Even though the MAC layer time stamping can reduce the problem of nonzero transmission delays to a large extent, there will still be some delays remaining. Furthermore, depending on what the final synchronized clock value is, these protocols can be grouped into the following three categories.

The first class is converge-to-max [10], where a node only synchronizes to the transmitter which has a larger clock value than the node's own clock. A simple converge-to-max protocol, timing synchronization function (TSF), is specified in the IEEE 802.11 standard [10], where only clock offset is adjusted. Based on the TSF, various modifications have been made to handle its limitations in multihop networks [11], [19], [20]. For instance, the modified automatic selftime-correcting procedure (MASP) scheme was proposed in [11], where a faster node is given a higher priority to send its synchronization messages. Besides, each node has a self correction capability to also compensate clock frequency difference among nodes. However, as addressed in [21], a common problem for all the converge-to-max schemes is the contradiction between the *fastest node asynchronism* and the *time partitioning*.[1]

The second category is converge-to-leader [12]–[15], where one or more leader nodes with reference clocks exist and the goal is to disseminate the reference clock value throughout the entire network. The timing-sync protocol for sensor networks (TPSN) in [12] builds a spanning tree of the network in the first place, and then synchronizes nodes to their parent by estimating clock offset through a two-way message exchange. The flooding time synchronization protocol (FTSP) was proposed in [13], where a hierarchical structure is formed and a root node (i.e., leader) periodically floods timestamps into the network. Besides, each node adopts a linear regression table to convert between its local clock and the reference clock, and propagates its time information about the reference clock after waiting for a given period of time. To accelerate the spread of the reference clock, the PulseSync algorithm in [15] offers rapid-flooding by allowing nodes to propagate their time information as fast and reliable as possible. Moreover, PISync was proposed in [22], which is based upon a Proportional-Integral (PI) controller and applies a proportional feedback (P) and an integral feedback (I) on the clock skew to compensate offset and frequency differences in an asymptotic manner.

The third class is arbitrary-consensus [5], [16], [17]. In these schemes, an internal common time scale, which does not need to be explicitly specified, is achieved in the network through communication among neighboring nodes without relying a hierarchy. Schenato and Fiorentin [16] proposed a distributed consensus based Average TimeSynch (ATS) protocol. The ATS method includes the cascade of two consensus algorithms where the first consensus synchronizes clock frequencies and the second consensus synchronizes clock offsets. Additionally, the random broadcast based distributed consensus clock synchronization (RBDS) and the consensus based clock synchronization (CoSyn) schemes were proposed in [5] and [17] respectively, where RBDS outperforms CoSyn. By distinguishing two different updates, the RBDS and CoSyn algorithms jointly adjust clock frequencies and offsets at the same time.

Although the above-mentioned methods work well in wireless sensor or ad hoc networks, they have limitations when being applied to mobile D2D systems. In particular, different D2D scenarios depicted in Fig. 1 yield different synchronization requirements and challenges. On one hand, in the partial-coverage scenario, since a fraction of the devices are in network coverage, they can be assumed perfectly synchronized to the BS. Then, the objective is to disseminate their clock values to the devices out-of-coverage in a fast and reliable way. On the other hand, in the out-of-coverage scenario, since

[1]Since a node only synchronizes to a faster node, the clock value of the fastest node (a node with the greatest clock value in the network) will keep drifting away from other nodes, unless it becomes the transmitter. This problem is called the fastest node asynchronism problem, which can be reduced by giving higher priority to the transmissions of the node with a faster clock. However, different priorities might result in the time partitioning problem, where the clock values in two groups of nodes may keep on drifting away from each other, even though they are connected.

no device has synchronized to the BS, the objective is to reach an internal common clock value among all the devices, which, however, does not need to belong to a specific device. Clearly, the different two objectives cannot be achieved by one synchronization category above. On top of that, the D2D scenario is usually unknown before the synchronization process starts, which will complicate the synchronization design even more.

B. Contributions

In this paper, we investigate the synchronization problem for mobile D2D networks.[2] To the best of our knowledge, this paper is the first work to unify timestamp-based synchronization for both partial-coverage and out-of-coverage D2D scenarios. The main contributions are as follows.

- We formulate the network synchronization problem for mobile D2D systems, and outline the different requirements for the partial-coverage and out-of-coverage scenarios respectively. Furthermore, five main challenges imposed on synchronization of mobile D2D networks are discussed.
- We analyze the theoretical principles and mathematical tools that can be exploited to handle the five challenges, and accordingly propose a low-complexity Adaptive distRibuted nEtwork Synchronization (ARES) algorithm. By using the proposed ARES scheme, which is a complete framework including both transmitting and receiving mechanisms, fast and reliable synchronization is achieved in both scenarios.
- We provide comprehensive simulations for performance evaluation. From the results, compared to the existing methods, the ARES algorithm shows not only better synchronization accuracy and faster convergence, but also improved scalability to network size and improved robustness to timestamp inaccuracy.

The remainder of the paper is organized as follows. In Section II, we describe our system model and formulate the synchronization problem for D2D networks, where five main challenges imposed on synchronization are analyzed as well. To tackle the five challenges, the theoretical principles behind the proposed ARES scheme are explained in Section III. Additionally, in Section IV we present the complete procedure of the ARES mechanism. Moreover, simulations are provided in Section V for performance evaluation. Finally, the paper is concluded in Section VI.

II. System Model and Problem Formulation

A. Notation

We use the following notation throughout the paper. The superscript $(\cdot)^{\mathsf{T}}$ stands for transposition. Uppercase and lowercase letters, e.g., x and X, represent scalars, lowercase boldface letters, e.g., $\boldsymbol{x}$, designate column vectors, and uppercase boldface letters, e.g., $\boldsymbol{X}$, denote matrices where $[\boldsymbol{X}]_{i,j}$ denotes the (i, j)th element and $[\boldsymbol{X}]_{i:n,j}$ denotes the column vector $\left[[\boldsymbol{X}]_{i,j}, [\boldsymbol{X}]_{i+1,j}, \cdots, [\boldsymbol{X}]_{n,j}\right]^{\mathsf{T}}$. Besides, $\mathbf{1}$ represents the all-ones column vector. Sets or multisets[3] are denoted by calligraphic letters $\mathcal{X}$ and their cardinalities are denoted by $|\mathcal{X}|$.

[2]Even though the considered environment here is D2D networks, the whole design principle and the proposed synchronization scheme can be applied to a conventional wireless sensor network as well.

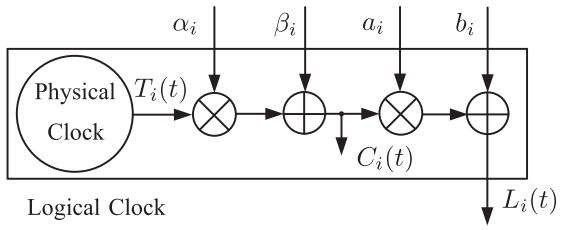


Fig. 2. The relationship of physical clock $T_i(t)$, auxiliary clock $C_i(t)$, and logical clock $L_i(t)$.

B. Network Model

We consider a network represented by a directed graph $\mathcal{G}(\ell) = (\mathcal{V}, \mathcal{E}(\ell))$, where the vertex set $\mathcal{V} = \{1, 2, \ldots, N\}$ contains N mobile devices and the edge set $\mathcal{E}(\ell)$ is defined as the set of available directed communication links at the discrete time index ℓ, i.e., $(i, j) \in \mathcal{E}(\ell)$ if device j sends information to device i during the ℓth synchronization round (SR). Here SR is defined as the time interval of a synchronization period. For all $i \in \mathcal{V}$ and ℓ, we use the convention that $(i, i) \in \mathcal{E}(\ell)$, and use the notation $\mathcal{V}_i(\ell) = \{j | (i, j) \in \mathcal{E}(\ell)\}$ to denote the set of neighbors of device i during the ℓth SR.

C. Clock Model

Each device in the network is equipped with a physical clock that has its frequency and offset. Here, we assume an affine function for the physical clock model. This way, the physical clock of device i is

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

where t is the global 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. Besides, for each clock i, f_i is assumed as a constant within the considered time period, which is reasonable provided that the short term stability of the clock is good.

In addition, each device i maintains a local auxiliary clock and a logical clock, whose values are denoted by $C_i(t)$ and $L_i(t)$, respectively. Here $C_i(t)$ is introduced for easy presentation and $L_i(t)$ represents the actual synchronized time of device i. Both $C_i(t)$ and $L_i(t)$ are functions of the current physical clock value $T_i(t)$. In this paper, we use the following affine models to formulate their relationship, which is also illustrated in Fig. 2.

$$C_i(t) = \alpha_i T_i(t) + \beta_i \tag{2}$$

$$L_i(t) = a_i C_i(t) + b_i = a_i \alpha_i T_i(t) + a_i \beta_i + b_i \tag{3}$$

Here α_i, β_i, a_i, and b_i are the control parameters to be adjusted, where $\{\alpha_i, \beta_i\}$ and $\{a_i, b_i\}$ are updated by the

[3]In mathematics, a multiset is a generalization of the concept of a set that, unlike a set, allows multiple instances of the multiset's elements. The total number of elements in a multiset, including repeated memberships, is the cardinality of the multiset.

synchronization algorithms designed for the out-of-coverage and partial-coverage scenarios respectively. This will be detailed later. In fact, we can use only two parameters instead of four parameters to relate $L_i(t)$ and $T_i(t)$. However, to explain the proposed synchronization algorithm in a more clear way, we maintain the four parameters with a slight redundancy. Additionally, we set the initial values as $\alpha_i = 1$, $\beta_i = 0$, $a_i = 1$, and $b_i = 0$. This way, we have $L_i(t) = C_i(t)$ before a_i and b_i get updated.

Moreover, for easy presentation later, we define

$$\hat{f}_i \triangleq \alpha_i f_i, \quad \hat{\theta}_i \triangleq \alpha_i \theta_i + \beta_i. \tag{4}$$

Note that $\hat{f}_i$ and $\hat{\theta}_i$ are related to the auxiliary clock $C_i(t)$ and we have $C_i(t) = \hat{f}_i t + \hat{\theta}_i$.

D. Conventional Transmission Mechanism of Timing Messages

In network synchronization, to utilize the broadcast nature of the wireless medium, devices broadcast timing messages which contain the timestamps recorded by the clock of transmitter. These messages are in turn used to adjust the clocks of the receivers. For transmitting timing messages, we consider a random access mechanism, where a device can broadcast at any time in any order. A practical and widely used random broadcast scheme is contention-based transmission, where devices contend for transmission opportunities at the beginning of each SR. Due to its simplicity and applicability in distributed networks, this protocol is the technique specified for synchronization in the IEEE 802.11 standard [10]. Specifically, each device [10]

1) at the beginning of each SR, calculates a random delay that is uniformly distributed in the range $[0, W_i]$, where $W_i \triangleq 2 \times$ aCWmin $\times$ aSlotTime (aCWmin and aSlotTime are constants defined in [10]);

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

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

4) sends a timing message when the random delay expires.

Remark 1: Due 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, $|\ \mathcal{V}_i(\ell)\ |$ can only be 1 or 2 for all $i \in \mathcal{V}$.

E. Problem Formulation

The general goal in network synchronization is to let the logical clocks $L_i(t)$ of different devices throughout the entire network have the same (or very close) values for any instant of global time t. However, concerning D2D environment, we need to make a distinction between two different synchronization scenarios, i.e., partial-coverage and out-of-coverage, since they do not have the exactly identical objectives.

Partial-Coverage Scenario: in this case, the devices in network coverage have already been perfectly synchronized to the BS and are called leaders. Note that in this scenario there is no need of a leader election process. The devices in network coverage are considered as leaders automatically, since they are assumed already synchronized to the BS. We suppose that leaders know their identities and that they have the same clock values $L(t)$. In addition, the other devices, which are called followers, are not connected to the BS and need an extra mechanism to become synchronized. Therefore, the aim here is to synchronize the followers' clock values to the leaders' clock value. Mathematically, the aim is to, for each device i, find the control parameters a_i and b_i based on multiple pairs of timestamps $\{L(t), C_i(t)\}$, such that

$$L(t) \approx a_i C_i(t) + b_i, \tag{5}$$

where the approximation is due to the inaccuracies of timestamps. Note that the formulated problem here is closely related to regression analysis. In this context, a_i and b_i are referred to as regression coefficients.

Out-of-Coverage Scenario: in this case, no device acts as a leader since all of them are out of network coverage, and all the devices are classified as followers. In addition, we have $L_i(t) = C_i(t)$ due to the unchanged $a_i = 1$ and $b_i = 0$. Correspondingly, the aim is to synchronize all the followers' logical clocks $L_i(t)$ or their respective auxiliary clocks $C_i(t)$ to a common virtual clock $C_{\mathrm{v}}(t) = f_{\mathrm{v}} t + \theta_{\mathrm{v}}$ ($f_{\mathrm{v}} > 0$). In fact, it is not crucial what the values of f_{v} and θ_{v} are; instead, what really matters is that all the logical/auxiliary clocks converge to one common value. This aim matches the consensus concept. Namely, we say that clock consensus is achieved if[4]

$$\lim_{t \to +\infty} \frac{C_i(t)}{C_{\mathrm{v}}(t)} = 1, \quad \forall i \in \mathcal{V}. \tag{6}$$

For each device i, the asymptotic consensus (6) is equivalent to concurrently achieving the following two consensus equations:

$$\lim_{t \to +\infty} \hat{f}_i = f_{\mathrm{v}}, \quad \lim_{t \to +\infty} \hat{\theta}_i = \theta_{\mathrm{v}}. \tag{7}$$

F. Five Main Challenges

Compared to conventional wireless sensor and ad hoc networks, synchronization in mobile D2D systems is more complex. Now we discuss five main challenges imposed on network synchronization for mobile D2D communications.

Challenge 1: For the partial-coverage synchronization scenario, how to spread the leaders' clock value in a fast and reliable manner. Especially further, when multiple leaders exist, how to utilize this advantage.

Challenge 2: For the out-of-coverage synchronization scenario, how to design distributed operation with only local information to achieve global clock consensus.

Challenge 3: In the out-of-coverage synchronization scenario, when a new device joins an almost synchronized group, how to guarantee that the new device will not incur any big change to the group.

Challenge 4: As stated above, there will be some delays remaining even if using MAC layer time stamping. Additionally, uncertainties exist in the time stamping process.

[4] As is common in the literature, consensus is interpreted from asymptotical sense.

Hence, how to deal with these inaccurate timestamps to achieve highly accurate synchronization?

Challenge 5: On top of the two different scenarios and their respective synchronization challenges, in practice, the followers initially are not aware of the coverage scenario they are in. Then, for the followers, how to decide which type of synchronization approaches to follow.

III. Design Principles of ARES

In this section, we present the design principles and the theoretical tools behind the ARES scheme, which are exploited to tackle the five challenges above.

A. For Partial-Coverage: Challenges 1 & 4

In the partial-coverage scenario, the goal is to synchronize the followers to the leaders. In this case, the main challenges include Challenge 1 and Challenge 4, where the key requirements are *fast spread* and *accurate estimation* of the leaders' clock value.

1) Fast Spread: For this purpose, we utilize three strategies: *i)* introduce pseudoleader; *ii)* cooperative synchronization; *iii)* higher probability to transmit when a device starts its (pseudo)leader role, which will be explained in detail in the following.

i) Pseudoleader: In addition to leader and follower, we define a new identity of devices: pseudoleader. Specifically, a follower will become a pseudoleader after it synchronizes to the leaders. Pseudoleaders act similarly with the leaders in the sense that other devices can also make use of the timestamps from pseudoleaders to estimate the leaders' clock value. However, in contrast to the leaders, pseudoleaders can as well update their logical clocks $L_i(t)$ when receiving timing messages from the leaders or the other pseudoleaders.

ii) Cooperative Synchronization: Compared to the typical tree-based synchronization scheme [12] where a node only synchronizes to its fixed parent node, we use the timing information from other devices as much as possible to exploit the broadcast nature of wireless communication. We name this strategy as cooperative synchronization, see Fig. 3. By doing so, the leaders' clock value can be spread quickly to the devices far away from the leaders. The advantage of cooperative synchronization becomes more obvious when multiple leaders exist. Moreover, the design of cooperative synchronization fits (highly) dynamic networks very well, since it completely removes the constraint on fixed connections between specific pairs of devices.

iii) Higher Probability to Transmit: At a first glance, it seems tempting to endow the leaders with higher probabilities to broadcast its timing information. However, due to the sharing of the common wireless medium, the broadcast of the leaders may block the transmission opportunities of the pseudoleaders that are directly connected to the leaders, and thus reduce the spread of the leaders' clock value to followers further away. To circumvent this problem, we let the devices have higher transmit probabilities to transmit only for a certain time period when they start their (pseudo)leaders' roles.

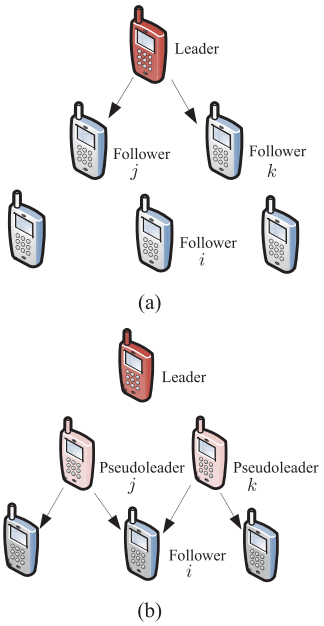


Fig. 3. Cooperative synchronization. (a) Followers j and k receive the timing messages broadcasted by the leader and synchronize to the leader. (b) After synchronizing to the leader, followers j and k become pseudoleaders. Then, follower i can utilize the timing messages broadcasted from either j or k. Note that, in the typical tree-based synchronization scheme [12], follower i can only choose one of the pseudoleaders j and k to be its parent and synchronize to it.

This idea is implemented in the transmission mechanism of synchronization messages that will be detailed in Section IV-A.

2) Accurate Estimation: As stated in Challenge 4, timestamps are usually not perfect in practice. Suppose device i receives a timing message from device $\tilde{j}(\ell)$ at time instance t_ℓ. Here, t_ℓ is the global time when the message was received during the ℓth SR. Then, the local recorded clock value is $C_i(t_\ell)$; the received timestamp is $C_{\tilde{j}(\ell)}(t_\ell - \delta_\ell)$ instead of the desired $C_{\tilde{j}(\ell)}(t_\ell)$, where δ_ℓ is the random transmission delay that includes the physical layer delay and the propagation delay. In addition to the random delay, uncertainty also exists in the stamping process. For simplicity of presentation and analysis, we model the overall inaccuracies at each node as a random variable with zero mean[5] and standard deviation σ. Next we present two mechanisms to achieve accurate estimation in the presence of imperfect timestamps: *i)* recursive estimation and *ii)* hierarchical structure.

i) Recursive Estimation: To combat the inaccuracy of timestamps, a device has to collect a sufficient number of effective timestamps, say E, to estimate the regression coefficients in (5). By an effective timestamp, we mean a timestamp that is indeed used by the receiving device to synchronize to the leaders' clock. Clearly, a (pseudo)leader will not utilize the timestamp from a follower even if it receives one, as the goal is to synchronize to the leaders. How to decide the effectiveness of timestamps will be explained later in this section.

[5]As in [15], the expected value of the variable is assumed known and can be subtracted, yielding a distribution with zero mean.

In most existing work [13]–[15] tackling the inaccuracy issue of synchronization, each device keeps a regression table with dimension E to store the most recent E pairs of effective timestamps. The estimation is then recalculated using the currently stored data when a new pair arrives. In general, the larger E is, the higher accuracy the estimation attains. However, a large E requires a long period of timestamp collection, which will then slow down the synchronization process throughout the entire network. This brings us to the following question: is it possible to achieve both high accuracy and high convergence speed? We answer it positively by proposing a regression mechanism for pseudoleaders. Recall that a follower becomes a pseudoleader when it receives the first E effective timestamps. In the proposed mechanism, whenever a pseudoleader receives a new effective timestamp, it will refine its estimate such that the outcome will be the same as the estimate using all the effective timestamps received since the beginning. This idea is in fact congruent with the properties of recursive estimation [23, p. 327] that are summarized in Lemma 1.

Lemma 1: (Recursive Estimation for Simple Linear Regression):

Given a set of measurements (X_k, Y_k) of data points $(X_k, \tilde{Y}_k)$ obeying the relation $g(X_k) = \tilde{Y}_k$ for $k = 1, \ldots, K$, where $g(X) = aX + b$. Assume that $Y_k = \tilde{Y}_k + W_k$, where W_k are identically and independently distributed random variables with zero mean and variance σ_k^2. Denote by $\hat{g}(X) = \hat{a}_K X + \hat{b}_K$ the linear regression of the data set $\{(X_k, Y_k)\}_{k=1,\ldots,K}$, where $\hat{a}_K$ and $\hat{b}_K$ are the estimates of the regression coefficients that minimize the expression $\sum_{k=1}^{K} (\hat{g}(X_k) - Y_k)^2$. Let

$$\boldsymbol{X}_K \triangleq \begin{bmatrix} X_1, & \cdots, & X_K \\ 1, & \cdots, & 1 \end{bmatrix}^{\mathsf{T}}, \tag{8}$$

then,

a) we have

$$\begin{bmatrix} \hat{a}_K \\ \hat{b}_K \end{bmatrix} = \left(\boldsymbol{X}_K^{\mathsf{T}} \boldsymbol{X}_K \right)^{-1} \boldsymbol{X}_K^{\mathsf{T}} \begin{bmatrix} Y_1, & \cdots, & Y_K \end{bmatrix}^{\mathsf{T}}; \tag{9}$$

b) the estimators in (9) have the smallest variance of any unbiased estimators that are linear combinations of Y_k. Here the smallest variance means that for every other linear unbiased estimator $[\hat{a}'_K, \hat{b}'_K]^{\mathsf{T}}$, $\mathrm{Cov}([\hat{a}'_K, \hat{b}'_K]^{\mathsf{T}}) - \mathrm{Cov}([\hat{a}_K, \hat{b}_K]^{\mathsf{T}})$ is a positive semi-definite matrix, where Cov(.) denotes the covariance matrix;

c) to estimate the regression coefficients in a recursive manner, we have

$$\begin{bmatrix} \hat{a}_r \\ \hat{b}_r \end{bmatrix} = \begin{bmatrix} \hat{a}_{r-1} \\ \hat{b}_{r-1} \end{bmatrix} + \left(\boldsymbol{X}_r^{\mathsf{T}} \boldsymbol{X}_r \right)^{-1} \begin{bmatrix} X_r \\ 1 \end{bmatrix} \left(Y_r - \begin{bmatrix} X_r \\ 1 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \hat{a}_{r-1} \\ \hat{b}_{r-1} \end{bmatrix} \right) \tag{10}$$

when using the first r measurements, where $3 \le r \le K$ and $\left[\hat{a}_2, \hat{b}_2\right]^{\mathsf{T}}$ is obtained by (9) with $K = 2$.

Proof: See [23, p. 327]. ∎

Suppose the leaders' clock values $L(t_\ell)$ have been sent to device i at SRs $\ell = 1, \ldots, K$ $(K \ge E)$ and device i has correspondingly recorded its local auxiliary clock values as $C_i(t_\ell)$. Inspired by Lemma 1, we propose the local update rule (11) for device i to synchronize to the leader:

$$\begin{bmatrix} a_i^{(\ell+1)} \\ b_i^{(\ell+1)} \end{bmatrix} = \begin{cases} \left[a_i^{(\ell)}, b_i^{(\ell)} \right]^{\mathsf{T}} & \text{if } \ell < E \quad \text{(11a)} \\ \begin{bmatrix} a_i^{(\ell)} \\ b_i^{(\ell)} \end{bmatrix} + \boldsymbol{\Phi}_i^{(\ell)} \begin{bmatrix} C_i(t_\ell) \\ 1 \end{bmatrix} \\ \times \left(L(t_\ell) - \begin{bmatrix} C_i(t_\ell) \\ 1 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} a_i^{(\ell)} \\ b_i^{(\ell)} \end{bmatrix} \right) & \text{otherwise,} \quad \text{(11b)} \end{cases}$$

where

$$\boldsymbol{\Phi}_i^{(\ell)} \triangleq \frac{1}{\Upsilon_{i1}^{(\ell)} \Upsilon_{i2}^{(\ell)} - (\Upsilon_{i3}^{(\ell)})^2} \begin{bmatrix} \Upsilon_{i1}^{(\ell)} & -\Upsilon_{i3}^{(\ell)} \\ -\Upsilon_{i3}^{(\ell)} & \Upsilon_{i2}^{(\ell)} \end{bmatrix}, \quad \ell = 1, \cdots \tag{12}$$

and

$$\begin{cases} \Upsilon_{i1}^{(\ell)} = \Upsilon_{i1}^{(\ell-1)} + 1 & (13) \\ \Upsilon_{i2}^{(\ell)} = \Upsilon_{i2}^{(\ell-1)} + C_i^2(t_\ell) & (14) \\ \Upsilon_{i3}^{(\ell)} = \Upsilon_{i3}^{(\ell-1)} + C_i(t_\ell), & (15) \end{cases}$$

with initializations $\Upsilon_{i1}^{(0)} = \Upsilon_{i2}^{(0)} = \Upsilon_{i3}^{(0)} = 0$.

As stated above, updates are not made to $[a_i, b_i]^{\mathsf{T}}$ before device i receives at least E effective timestamps. Hence (11a) is obtained. Moreover, the convergence of the update rule in (11) is shown by the following Lemma 2.

Lemma 2: Suppose the leaders' clock values $L(t_\ell)$ and the auxiliary clock values $C_i(t_\ell)$ are available at device i for $\ell = 1, \ldots, K$ $(K \ge E)$. Following the local update rule in (11), device i will achieve a recursive version of estimating $L(t_\ell)$ for $\ell = E, \ldots, K$.

Proof: Consider $\ell \ge E$, we can rewrite $\boldsymbol{\Phi}^{(\ell)}$ as

$$\text{Right hand side of (12)} = \begin{bmatrix} \Upsilon_{i2}^{(\ell)} & \Upsilon_{i3}^{(\ell)} \\ \Upsilon_{i3}^{(\ell)} & \Upsilon_{i1}^{(\ell)} \end{bmatrix}^{-1} \tag{16}$$

$$= \begin{bmatrix} \sum_{\ell'=1}^{\ell} C_i^2(t_{\ell'}) & \sum_{\ell'=1}^{\ell} C_i(t_{\ell'}) \\ \sum_{\ell'=1}^{\ell} C_i(t_{\ell'}) & \ell \end{bmatrix}^{-1}, \tag{17}$$

where (16) holds by calculating the inverse of a 2×2 matrix and (17) follows from (13)–(15).

Then, the update rule in (11b) matches the form in (10) by substituting $\boldsymbol{\Phi}^{(\ell)}$ in (17) into (11b). Thus, (11b) is indeed a recursive version of estimating a_i and b_i as given in Lemma 1. Correspondingly, $L_i(t_\ell) = a_i^{(\ell)} C_i(t_\ell) + b_i^{(\ell)}$ is a recursive estimation of $L(t_\ell)$. ∎

ii) Hierarchical Structure: In the partial-coverage scenarios, the clocks of different nodes will have different reliabilities. Naturally, leaders have perfect clocks, while pseudoleaders and followers will have imperfect clocks. To measure the relative quality of the clocks, we introduce a hierarchical (integer) number $\eta_i \ge 1$ for each node i, which is related, but not identical to the hop distance from a leader. If node i is a leader

then $\eta_i = 1$. If node i is not a leader then η_i is adjusted in accordance to the number of received effective timestamps. By effective transmitter, we mean the transmitter that yields effective timestamp. Moreover, the effectiveness of a timestamp is given as follows. We let the timestamp to be effective for a follower if the transmitter is a (pseudo)leader, and to be effective for a pseudoleader if the transmitter has lower hierarchical number than that of the receiver. For device i, define by $\mathcal{V}_i^{\mathrm{ET}}$ the multiset of its effective transmitters from the beginning up to the current time. In other words, whenever device i receives an effective timestamp from another device, say device j, i.e., $\eta_j < \eta_i$, we have $\mathcal{V}_i^{\mathrm{ET}} = \mathcal{V}_i^{\mathrm{ET}} \cup \{j\}$. Note that $\mathcal{V}_i^{\mathrm{ET}}$ may contain duplicate elements. This way, η_i is calculated as $\eta_i = \lceil \sum_{j \in \mathcal{V}_i^{\mathrm{ET}}} \eta_j / |\mathcal{V}_i^{\mathrm{ET}}| \rceil + 1$, where η_j indicates device j's hierarchical number at the moment when it transmitted the corresponding timestamp to device i. To update η_i in a recursive way, we have

$$\eta_i = \lceil \left(\eta_i \times (\Pi_i - 1) + \eta_j\right) / \Pi_i \rceil + 1, \tag{18}$$

where $\Pi_i \triangleq |\mathcal{V}_i^{\mathrm{ET}}|$ counts the current number of effective timestamps device i has received.

B. For Out-of-Coverage: Challenges 2, 3, & 4

In the out-of-coverage scenario, network synchronization can in fact be formulated as a consensus problem shown in (6) and (7). Correspondingly, as stated in Challenges 2 & 4, the synchronization algorithm is required to achieve asymptotic consensus on both $\hat{f}_i$ and $\hat{\theta}_i$ defined in (4) when timestamps are perfect, and show robustness to inaccurate timestamps as well. Hence, we assume that $C_i(t_\ell)$ is perfect for the time being when proving convergence. Nevertheless, the final algorithm will be adjusted to be robust against inaccuracy. We have in [17] proposed the RBDS scheme which satisfies the above two requirements. However, the RBDS method cannot address Challenge 3 since it gives the same weights to the timestamps from both the transmitter and receiver when the latter updates its local auxiliary clock $C_i(t)$. We here propose a modified RBDS (M-RBDS) scheme with new weight design. Intuitively, the devices belonging to the synchronized group should be rendered higher weights than that of the new device. To this end, we define S_i to be the change counter for follower i, which is incremented every time follower i updates its auxiliary clock $C_i(t)$.

As in [17], M-RBDS distinguishes between two different updates—partial updates and complete updates—for different situations. Next we present the two update rules.

1) Partial Update Rule: It is clear that device i cannot make any meaningful update of α_i before receiving at least two timing messages from the same device, since $\hat{f}_i$ basically represents the slope of the linear auxiliary clock model $C_i(t)$. Therefore, the partial update rule is given as

$$\begin{cases} \alpha_i^{(\ell+1)} = \alpha_i^{(\ell)} & (19) \\ \beta_i^{(\ell+1)} = \beta_i^{(\ell)} + \omega_{\mathrm{T}} \left(C_{\tilde{j}(\ell)}(t_\ell) - C_i(t_\ell) \right), & (20) \end{cases}$$

where $\omega_{\mathrm{T}} = S_{\tilde{j}(\ell)} / (S_{\tilde{j}(\ell)} + S_i)$. The partial update (19) and (20) implies that $\hat{f}_i^{(\ell+1)} = \hat{f}_i^{(\ell)}$ and

$$\begin{aligned} \hat{\theta}_i^{(\ell+1)} &= \alpha_i^{(\ell+1)} \theta_i + \beta_i^{(\ell+1)} && (21) \\ &= \alpha_i^{(\ell)} \theta_i + \beta_i^{(\ell)} + \omega_{\mathrm{T}} \left(C_{\tilde{j}(\ell)}(t_\ell) - C_i(t_\ell) \right) && (22) \\ &= \hat{\theta}_i^{(\ell)} + \omega_{\mathrm{T}} \left(\hat{f}_{\tilde{j}(\ell)} t_\ell + \hat{\theta}_{\tilde{j}(\ell)} - \hat{f}_i^{(\ell)} t_\ell - \hat{\theta}_i^{(\ell)} \right) && (23) \\ &= \left(\omega_{\mathrm{T}} \hat{\theta}_{\tilde{j}(\ell)} + \omega_{\mathrm{R}} \hat{\theta}_i^{(\ell)} \right) + \omega_{\mathrm{T}} \left(\hat{f}_{\tilde{j}(\ell)} - \hat{f}_i^{(\ell)} \right) t_\ell, && (24) \end{aligned}$$

where $\omega_{\mathrm{R}} = S_i / (S_{\tilde{j}(\ell)} + S_i)$ and (24) follows since $\omega_{\mathrm{T}} + \omega_{\mathrm{R}} = 1$.

For convenience, we define $\Delta_i^{(\ell)}$ as $\Delta_i^{(\ell)} \triangleq \omega_{\mathrm{T}} \left(C_{\tilde{j}(\ell)}(t_\ell) - C_i(t_\ell) \right)$, which captures the adjustment of auxiliary clock of device i.

2) Complete Update Rule: Suppose device i receives a timestamp from device $j = \tilde{j}(\ell)$ at time t_ℓ. Furthermore, suppose the last time device i received a timestamp from device j was t_n. Hence, $t_n < t_\ell$ and $j = \tilde{j}(\ell) = \tilde{j}(n)$. Device i will perform a complete update if the following two conditions are satisfied.

a) Device j has not performed a partial or complete update in the interval $(t_n, t_\ell]$.

b) Device i has not performed a complete update in the interval $(t_n, t_\ell]$.

Note that the condition *a)* implies that $C_j(t)$ is an affine function of t, with slope $\hat{f}_j$, for $t \in (t_n, t_\ell]$.[6] The condition *b)* implies that $C_i(t)$ is a piecewise affine function of t, with slope $\hat{f}_i^{(n+1)} = \hat{f}_i^{(n+2)} = \cdots = \hat{f}_i^{(\ell)}$, for $t \in (t_n, t_\ell]$. Then, a very useful quantity can be derived as

$$\frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} = \frac{C_j(t_\ell) - C_j(t_n)}{C_i(t_\ell) - \sum_{m=n}^{l-1} \Delta_i^{(m)} - C_i(t_n)}. \tag{25}$$

Based on (25), the complete update rule is given as

$$\begin{cases} \alpha_i^{(\ell+1)} = \alpha_i^{(\ell)} \left(\omega_{\mathrm{R}} + \omega_{\mathrm{T}} \dfrac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \right), & (26) \\ \beta_i^{(\ell+1)} = \omega_{\mathrm{T}} \left(C_j(t_\ell) - \dfrac{\hat{f}_j}{\hat{f}_i^{(\ell)}} C_i(t_\ell) \right) + \left(\omega_{\mathrm{R}} + \omega_{\mathrm{T}} \dfrac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \right) \beta_i^{(\ell)}, & (27) \end{cases}$$

where $\omega_{\mathrm{R}} = S_i / (S_j + S_i)$, $\omega_{\mathrm{T}} = S_j / (S_j + S_i)$, and the right hand sides can be computed with the help of (25). The complete update (26) and (27) implies that

$$\begin{aligned} \hat{f}_i^{(\ell+1)} &= \alpha_i^{(\ell+1)} f_i = \alpha_i^{(\ell)} \left(\omega_{\mathrm{R}} + \omega_{\mathrm{T}} \frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \right) f_i && (28) \\ &= \hat{f}_i^{(\ell)} \left(\omega_{\mathrm{R}} + \omega_{\mathrm{T}} \frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \right) = \omega_{\mathrm{T}} \hat{f}_j + \omega_{\mathrm{R}} \hat{f}_i^{(\ell)} && (29) \end{aligned}$$

and

$$\begin{aligned} \hat{\theta}_i^{(\ell+1)} &= \alpha_i^{(\ell+1)} \theta_i + \beta_i^{(\ell+1)} && (30) \\ &= \alpha_i^{(\ell)} \left(\omega_{\mathrm{R}} + \omega_{\mathrm{T}} \frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \right) \theta_i + \omega_{\mathrm{T}} \left(C_j(t_\ell) - \frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} C_i(t_\ell) \right) \\ &\quad + \left(\omega_{\mathrm{R}} + \omega_{\mathrm{T}} \frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \right) \beta_i^{(\ell)} && (31) \end{aligned}$$

[6]For notational simplicity, here we use $\hat{f}_j$ and $\hat{\theta}_j$ without the explicit dependence on the SR index.

$$= \alpha_i^{(\ell)} \left(\omega_{\mathrm{R}} + \omega_{\mathrm{T}} \frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \right) \theta_i + \omega_{\mathrm{T}} C_j(t_\ell) - \omega_{\mathrm{T}} \hat{f}_j t_\ell - \omega_{\mathrm{T}} \frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \hat{\theta}_i^{(\ell)} + \left(\omega_{\mathrm{R}} + \omega_{\mathrm{T}} \frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \right) \beta_i^{(\ell)} \tag{32}$$

$$= \alpha_i^{(\ell)} \left(\omega_{\mathrm{R}} + \omega_{\mathrm{T}} \frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \right) \theta_i + \omega_{\mathrm{T}} \left(\hat{\theta}_j - \hat{\theta}_i^{(\ell)} \frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \right) + \left(\omega_{\mathrm{R}} + \omega_{\mathrm{T}} \frac{\hat{f}_j}{\hat{f}_i^{(\ell)}} \right) \beta_i^{(\ell)} \tag{33}$$

$$= \omega_{\mathrm{T}} \hat{\theta}_j + \omega_{\mathrm{R}} \hat{\theta}_i^{(\ell)}, \tag{34}$$

where (32), (33), and (34) follow from the definitions of $\hat{f}_i$ and $\hat{\theta}_i$ in (4).

Now we are in the position to discuss the convergence of the M-RBDS scheme. It is worth mentioning that the M-RBDS will reduce to the RBDS when we have $\omega_{\mathrm{T}} = \omega_{\mathrm{R}} = 0.5$. As revealed in (25), the adjustment of $\hat{f}_i$ requires at least two timestamps, which cannot be implied by the graph $\mathcal{G}(\ell)$. Therefore, we consider a new directed graph $\mathcal{F}(\ell) = (\mathcal{V}, \tilde{\mathcal{E}}(\ell))$. Concerning the edge set $\tilde{\mathcal{E}}(\ell)$, we let $(i, j) \in \tilde{\mathcal{E}}(\ell)$ if device i implements a complete update at the ℓth SR based on device j's information. Besides, $\forall i \in \mathcal{V}$, the set $\tilde{\mathcal{V}}_i(\ell)$ is defined as $\tilde{\mathcal{V}}_i(\ell) \triangleq \{j | (i, j) \in \tilde{\mathcal{E}}(\ell)\}$. Based on the new graph $\mathcal{F}(\ell)$, the following theorem states that the M-RBDS scheme will indeed achieve consensus asymptotically.

Theorem 1: Assume

a) all devices can broadcast in any order as long as an infinite sequence of graphs $\mathcal{F}(1), \mathcal{F}(2), \ldots$ is repeatedly jointly rooted by subsequences of length q[7];
b) the timestamps are perfect.

Then, if each device updates its control parameters as (19) and (20), or (26) and (27), depending on the conditions of the partial update rule or the complete update rule being satisfied, the asymptotical consensus (7), and therefore, (6) is achieved as well.

Proof: Similar with the proof of [17, Th. 3]. ∎

Remark 2: For the robustness to inaccurate timestamps, we consider a threshold for the adjustment of the local auxiliary clock $C_i(t)$. When device i receives a timing message at the ℓth SR, it will not use it for the update unless $|C_i(t_\ell) - C_{\tilde{j}(\ell)}(t_\ell - \delta_\ell)| > \gamma$, where γ is the threshold. Note that $C_i(t_\ell)$ and $C_{\tilde{j}(\ell)}(t_\ell - \delta_\ell)$ are inaccurate timestamps including random delays and sampling uncertainties. The value of γ is the tradeoff between the speed of synchronization error decrease and the robustness against inaccuracies.

C. For Challenge 5

Despite the solutions devised respectively for partial-coverage and out-of-coverage, as stated in Challenge 5, the followers initially are not aware of the current synchronization scenario. To unify these two scenarios, we let each follower to implement the synchronization algorithm designed for the out-of-coverage scenario (Section III-B) before it receives a timing message from a (pseudo)leader. On the other hand, the follower will ignore the timing information from any other follower after it receives a message from a (pseudo)leader, and conduct the synchronization scheme devised for the partial-coverage scenario (Section III-A).

[7]As shown in [17], This assumption is realistic when a contention based transmission mechanism is used.

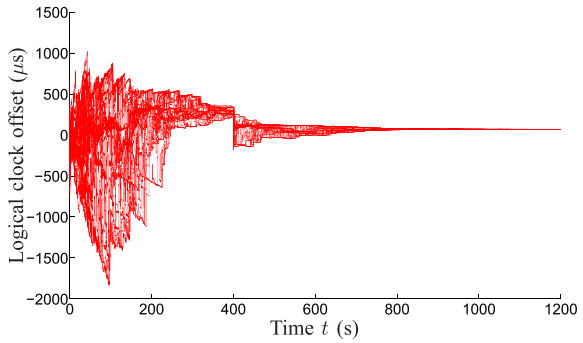


Fig. 4. Impact of a frequency variation at 400 s on convergence with 50 nodes (no leader) and $\sigma = 0$.

Remark 3: As stated in Section II-C, here we assume a constant physical clock frequency f_i, or more specifically that any change in the clock frequency is slow in comparison with the update rate of the synchronization schemes. Nevertheless, the proposed ARES scheme will naturally extend to the case when there is a small change in clock frequency. For the partial-coverage scenario, since the goal is to synchronize to the leader(s), the synchronization problem can be formulated as a linear regression problem. In this way, the variation of the clock frequency can be taken into account as timestamp inaccuracies. Moreover, for the out-of-coverage scenario, the proposed ARES scheme will still work well when there is a frequency variation, which is illustrated in Fig. 4, in which we assume that the physical clock frequencies of half of the nodes are changed at 400 s due to a temperature change. We also consider a typical temperature coefficient of -0.04 ppm/°C for crystal oscillators and a temperature change of -25°C. As observed in Fig. 4, even though the logical clock offsets experience a sudden spread after 400 s due to the frequency variation, the proposed ARES scheme is able to quickly recover and achieve convergence.

IV. Implementation of ARES

Based on the principles discussed above, we now present the complete procedure of the proposed ARES scheme including both transmitting and receiving components.

A. Proposed Transmission Mechanism of Synchronization Messages

In this paper, we propose a modified version of the contention-based broadcast mechanism [10]. In the original mechanism described in Section II-D, the length of contention window W_i for device i is fixed and $W_i = 2 \times$ aCWmin $\times$ aSlotTime. In the proposed ARES scheme, to speed up the spread of the leaders' clock value (i.e., Challenge 1), we give device i a higher probability to transmit for a certain time period when it appears as a (pseudo)leader as analyzed in Section III-A.1. To do so, W_i is set as an adaptive value that equals 2 $\times$ aCWmin $\times$ aSlotTime / W_D for the first T_D

Algorithm 1 Recursive Update in Partial-Coverage Synchronization

Input: C_j, η_j, C_i, $a_i^{(\ell)}$, $b_i^{(\ell)}$, $\Upsilon_{i1}^{(\ell-1)}$, $\Upsilon_{i2}^{(\ell-1)}$, $\Upsilon_{i3}^{(\ell-1)}$, η_i, Π_i
Output: $a_i^{(\ell+1)}$, $b_i^{(\ell+1)}$, $\Upsilon_{i1}^{(\ell)}$, $\Upsilon_{i2}^{(\ell)}$, $\Upsilon_{i3}^{(\ell)}$, η_i, Π_i
$\Upsilon_{i1}^{(\ell)} = \Upsilon_{i1}^{(\ell-1)}+1$, $\Upsilon_{i2}^{(\ell)} = \Upsilon_{i2}^{(\ell-1)}+C_i^2$, $\Upsilon_{i3}^{(\ell)} = \Upsilon_{i3}^{(\ell-1)}+C_i$ *// from (13)–(15)*
Calculate $a_i^{(\ell+1)}$ and $b_i^{(\ell+1)}$ by (11b), where $\mathbf{\Phi}_i^{(\ell)}$ is defined in (12)
$\Pi_i = \Pi_i + 1$; $\eta_i = \lceil \left(\eta_i \times (\Pi_i - 1) + \eta_j\right) / \Pi_i \rceil + 1$ *// from (18)*

SRs after device i starts its (pseudo)leader's role, while equals $2 \times$ aCWmin $\times$ aSlotTime otherwise. Here W_D is a factor for shortening the window length and T_D is the number of SRs within which device i is given higher probability to transmit. Compared to [10], the major modification in the transmission protocol of the proposed ARES scheme lies in the way of calculating W_i described above.

Moreover, the timing message has different forms when it is sent from a node with different identities, where we use χ_i to denote the identity of node i. Specifically, the transmitted timing message is of the form $[\chi_i,\ C_i(t_\ell),\ \eta_i]$ with $\chi_i = 0$ when device i is a leader, while with $\chi_i = 1$ when device i is a pseudoleader; and of the form $[\chi_i,\ i,\ C_i(t_\ell),\ S_i]$ with $\chi_i = 3$ when device i is a follower that has not yet received any timing message from a (pseudo)leader, while with $\chi_i = 2$ when device i is a follower that has already received timing message from a (pseudo)leader.

B. Receiving of Synchronization Messages

Suppose device i receives a timing message from device j during the ℓth SR. The local operation of device i depends on their respective identities.

Clearly, if device i is a leader, i.e., $\chi_i = 0$, it will not update its clock.

Furthermore, if device i is a pseudoleader, i.e., $\chi_i = 1$, it will conduct the recursive update algorithm described in Algorithm 1 under the condition that the received timestamp from device j is effective. The convergence property of the recursive operation is guaranteed by Lemma 2.

If device i is a follower, the situation is a bit more complicated since we need to further distinguish two cases. On one hand, if device j is also a follower, device i will assume the out-of-coverage scenario and execute the proposed M-RBDS algorithm whose procedures are detailed in Algorithm 2. In this case, χ_i will be kept as 3. On the other hand, if device j is a (pseudo)leader, device i will be aware of the partial-coverage scenario and thus start to collect the effective timestamps. We change χ_i to 2 to represent this case. The steps of collecting timing messages and estimating regression coefficients by using the first E effective timestamps are presented in Algorithm 3. Besides, to keep track of the received messages in history, follower i maintains a matrix $\boldsymbol{A}_i$ which has different forms for the two cases $\chi_i = 3$ and $\chi_i = 2$, as revealed in Algorithm 2 and Algorithm 3 respectively.

Algorithm 2 Out-of-Coverage Synchronization

Input: $\boldsymbol{A}_i$, j, S_j, C_j, C_i, $\alpha_i^{(\ell)}$, $\beta_i^{(\ell)}$
Output: $\alpha_i^{(\ell+1)}$, $\beta_i^{(\ell+1)}$, S_i, $\boldsymbol{A}_i$
$\mathcal{A} = \{m : [\boldsymbol{A}_i]_{m,1} = j\}$
// the set of previous messages from device j
if $\mathcal{A} == \emptyset$ **then** Test = false
// no previous message from device j
else $m = \max\{n :\ n \in \mathcal{A}\}$
  if $S_j > [\boldsymbol{A}_i]_{m,2}$ **then** Test = false
    // j has changed auxiliary clock since last timing message
  else Test = true *// complete update possible*
  end if
end if
if Test == false **then** *// execute partial update*
  $\alpha_i^{(\ell+1)} = \alpha_i^{(\ell)}$ *// from (19)*; $\beta_i^{(\ell+1)} = \beta_i^{(\ell)} + \omega_{\mathrm{T}}(C_j - C_i)$ *// from (20)*
  $\boldsymbol{A}_i = \begin{bmatrix} \boldsymbol{A}_i \\ j, S_j, C_j, C_i \end{bmatrix}$
else *// execute complete update*
  $m = \max\{n : [\boldsymbol{A}_i]_{n,1} = j\}$
  // index of the last message from device j
  $\rho =$ number of rows in $\boldsymbol{A}_i$
  $\Delta_{\mathrm{tot}} = \sum_{n=m}^{\rho} \left([\boldsymbol{A}_i]_{n,3} - [\boldsymbol{A}_i]_{n,4}\right)$
  $\kappa = (C_j - [\boldsymbol{A}_i]_{m,3})/(C_i - \Delta_{\mathrm{tot}} - [\boldsymbol{A}_i]_{m,4})$
  // $\kappa = \hat{f}_j / \hat{f}_i^{(\ell)}$ from (25)
  $\alpha_i^{(\ell+1)} = \alpha_i^{(\ell)}(\omega_{\mathrm{R}} + \omega_{\mathrm{T}}\kappa)$
  // from (26)
  $\beta_i^{(\ell+1)} = \omega_{\mathrm{T}}(C_j - \kappa C_i) + \beta_i^{(\ell)}(\omega_{\mathrm{R}} + \omega_{\mathrm{T}}\kappa)/2$
  // from (27)
  $\boldsymbol{A}_i = [j, S_j, C_j, C_i]$
end if
$S_i = S_i + 1$

To summarize, a flow diagram of the complete ARES scheme is described in Fig. 5. Initially, we let $\chi_i = 0$ and $\eta_i = 1$ for each leader i, and let $\chi_i = 3$ for each follower i.

It is worth mentioning that the proposed ARES scheme has quite low computational complexity and requires small local memory for storing timing information. Particularly, in the partial-coverage scenario, compared to the FTSP and PulseSync, the complexity of the ARES is much lower since it only uses the new coming pair of timestamps to calculate the recursive step in (11b). Whereas, the linear regression (9) with E pairs of timestamps is necessary for FTSP and PulseSync. Also, the required local storage of ARES is smaller since it does not need to save the E-row regression table that is mandatory for the other two methods.

V. Performance Evaluation

In this section, simulation results[8] are presented to compare the performance of the proposed ARES scheme with the baseline methods: *1)* TSF in [10]; *2)* MASP in [11];

[8]Here we only consider numerical simulations. Real-world experimentation is left for future work.

Algorithm 3 Effective Timestamp Collection for Follower i

Input: C_j, η_j, C_i, $\boldsymbol{A}_i$, Π_i, χ_i, E
Output: $a_i^{(\ell+1)}$, $b_i^{(\ell+1)}$, $\boldsymbol{A}_i$, η_i, Π_i, χ_i, $\Upsilon_{i1}^{(\ell)}$, $\Upsilon_{i2}^{(\ell)}$, $\Upsilon_{i3}^{(\ell)}$
if $\chi_i == 3$ **then** $\boldsymbol{A}_i = []$, $\boldsymbol{A}_i = [\eta_j, C_j, C_i]$
else $\boldsymbol{A}_i = \begin{bmatrix} \boldsymbol{A}_i \\ \eta_j, C_j, C_i \end{bmatrix}$
end if
$\Pi_i = \Pi_i + 1$; $\rho =$ number of rows in $\boldsymbol{A}_i$
if $\rho == E$ **then**
  $\boldsymbol{x} = [\boldsymbol{A}_i]_{1:\rho,3}$, $\boldsymbol{y} = [\boldsymbol{A}_i]_{1:\rho,2}$, $\tilde{\boldsymbol{\eta}}_i = [\boldsymbol{A}_i]_{1:\rho,1}$
  $a_i^{(\ell+1)} = \boldsymbol{x}^\mathsf{T}(\boldsymbol{x}+\boldsymbol{1})\boldsymbol{x}^\mathsf{T}\boldsymbol{y}$, $b_i^{(\ell+1)} = \boldsymbol{1}^\mathsf{T}(\boldsymbol{x}+\boldsymbol{1})\boldsymbol{x}^\mathsf{T}\boldsymbol{y}$
  // from (11b)
  $\eta_i = \lceil \boldsymbol{1}^\mathsf{T}\tilde{\boldsymbol{\eta}}_i/\rho \rceil + 1$, $\chi_i = 1$, $\boldsymbol{A}_i = []$, $\Upsilon_{i1}^{(\ell)} = \rho$, $\Upsilon_{i2}^{(\ell)} = \boldsymbol{x}^\mathsf{T}\boldsymbol{x}$, $\Upsilon_{i3}^{(\ell)} = \boldsymbol{1}^\mathsf{T}\boldsymbol{x}$
else $a_i^{(\ell+1)} = 1$, $b_i^{(\ell+1)} = 0$, $\eta_i = []$, $\chi_i = 2$, $\Upsilon_{i1}^{(\ell)} = []$, $\Upsilon_{i2}^{(\ell)} = []$, $\Upsilon_{i3}^{(\ell)} = []$
end if

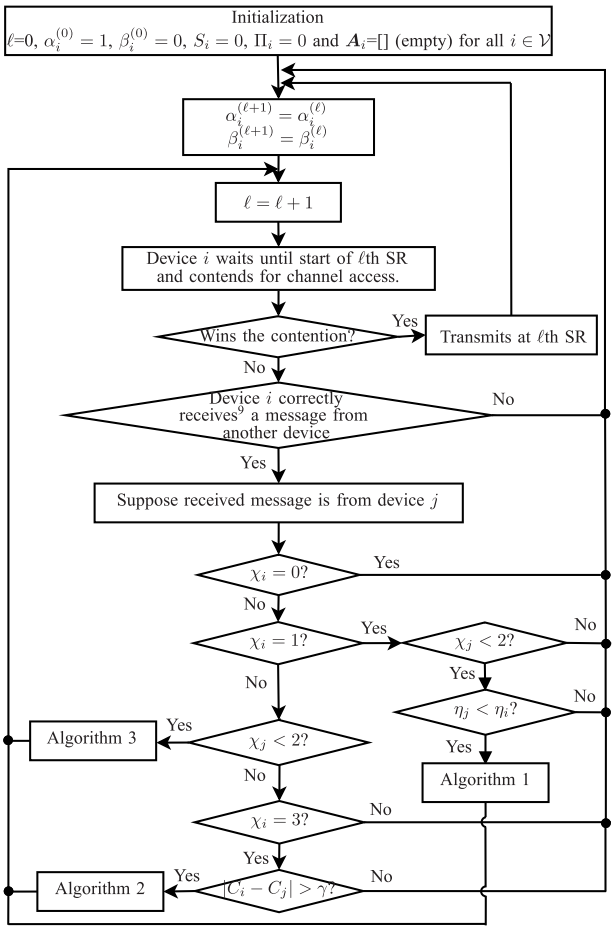


Fig. 5. Flowchart of the ARES scheme.

3) PulseSync in [15] with 8 pairs of timestamps in the regression table; and *4)* RBDS in [17]. Note that we skip the two well known synchronization methods ATS in [16] and FTSP in [13]. This is because [15] and [17] have already shown the superiority of RBDS over ATS and the superiority

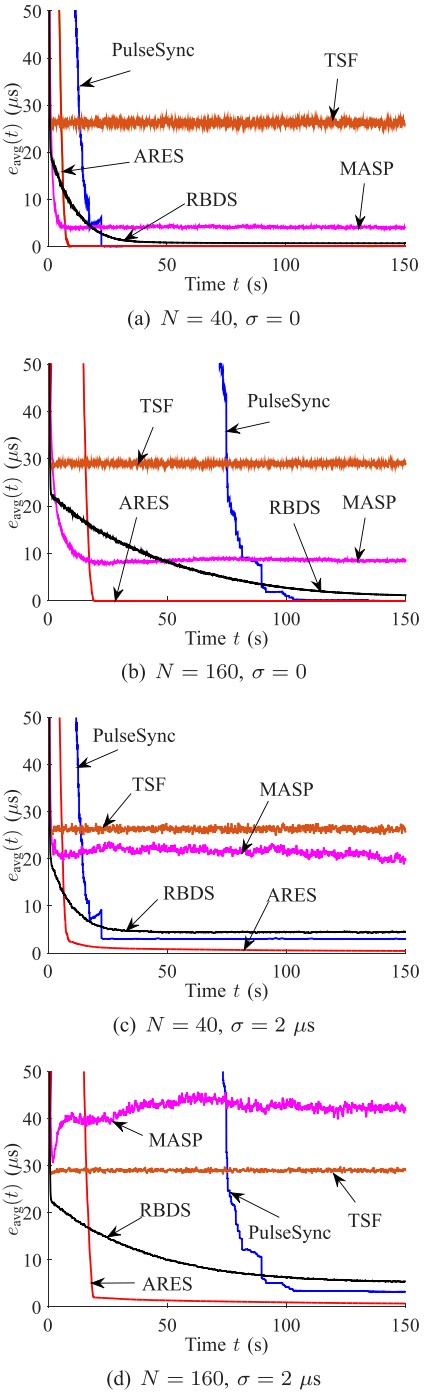


Fig. 6. Average synchronization errors versus time evolution with $M=1$ and $\Omega=5$.

of PulseSync over FTSP, respectively.

Consider a network with N mobile devices including M leaders, where, unless otherwise specified, the Erdős–Rényi (ER) graph [24] with expected node degree Ω is used to model the random network. In addition, we assume that network realizations vary independently over different SRs, which is used to model mobility. The physical clock frequencies are uniformly and randomly selected from the range

[9]Here, "correctly received" means that the message is received without collision. On the other hand, we assume that any collision will lead to packet loss.

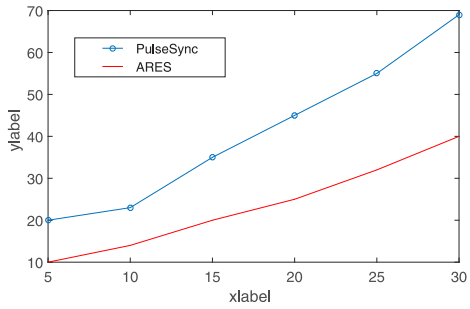


Fig. 7. Average synchronization time for different network diameters with $\sigma = 0$.

[0.9999, 1.0001] Hz, following IEEE 802.11 protocol requirements [10]. Also, the initial clock values are uniformly and randomly chosen from the range $[-800, 800]$ μs. In fact, we have also investigated a wider range of initial clock values, but since the performance of those cases follows the same trends as the ones shown below and to save space, we have not included those results. Other simulation parameters are summarized as follows. The period of one SR is 0.1 s [10]. The number of timestamps for the first estimation of regression coefficients is $E = 4$. The parameters used in calculating the backoff time in the contention based transmission protocol are aSlotTime $= 50\,\mu$s and aCWmin $= 15$ from [10], as well as $W_D = 20$ and $T_D = 4$ for the ARES scheme. Furthermore, the inaccuracies of timestamps are modeled by a uniform distribution within the range $[-\sqrt{3}\sigma, \sqrt{3}\sigma]$ μs, and we set the threshold in Fig. 5 as $\gamma = \sqrt{3}\sigma$.

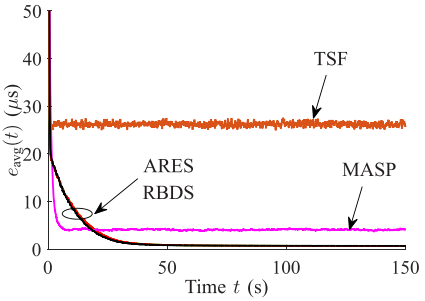
The performance metrics in our simulations are the maximum synchronization errors and average synchronization errors defined by (35) and (36) respectively:

$$e_{\max}(t) \triangleq \max_{i,j} |C_i(t) - C_j(t)|, \tag{35}$$

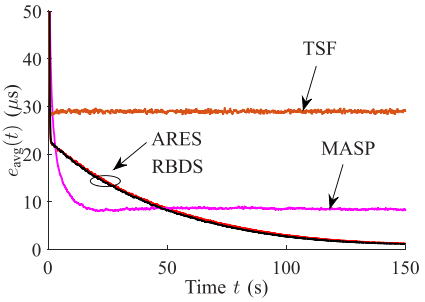
$$e_{\text{avg}}(t) \triangleq \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} |C_i(t) - C_j(t)|. \tag{36}$$

For each result in the following figures, we average the errors over 200 different simulation runs.

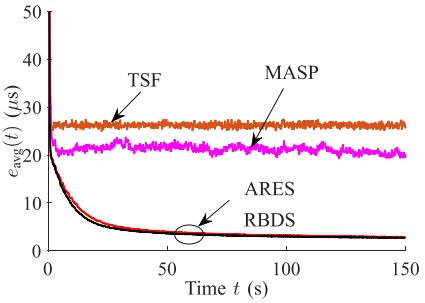
Fig. 6 plots the average synchronization errors versus the evolved time for the partial-coverage D2D scenario with $M = 1$. When $\sigma = 0$ (i.e., the timestamps are perfect) and $N = 40$, as shown in Fig. 6(a), although TSF and MASP exhibit fast decrease of synchronization errors in the first 10 s, they both have serious floor effects. These are caused by the ineffectiveness of TSF in multihop networks, and the contradiction between the fastest node asynchronism and the time partitioning of MASP. Besides, there is still a small remaining error of the RBDS scheme after $t \geq 100$ s, which is due to the asymptotic consensus of RBDS. Also, compared to the proposed ARES scheme, RBDS shows a slower convergence, especially for a large network. The reason is that RBDS is designed for a pure out-of-coverage scenario and thus cannot exploit the existence of a leader. More specifically, the goal of RBDS is not to synchronize to the leader, but to achieve clock consensus. Furthermore, we see that the PulseSync and the proposed ARES can indeed achieve perfect synchronization



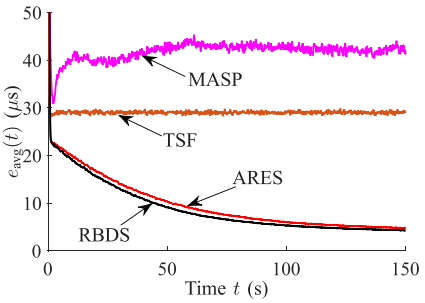
(a) $N = 40$, $\sigma = 0$



(b) $N = 160$, $\sigma = 0$



(c) $N = 40$, $\sigma = 2$ μs



(d) $N = 160$, $\sigma = 2$ μs

Fig. 8. Average synchronization errors versus time evolution with $M = 0$ and $\Omega = 5$.

within limited time in this setup, where ARES has a clear advantage in convergence speed.

With perfect timestamps, the synchronization error curves are further depicted in Fig. 6(b) for $N = 160$, where the performance order of different methods is similar with that in Fig. 6(a). In this case, however, the PulseSync attains zero error much more slowly than ARES. From both Fig. 6(a) and Fig. 6(b), we see that even though RBDS, MASP, and ARES can all guarantee a good synchronization accuracy,

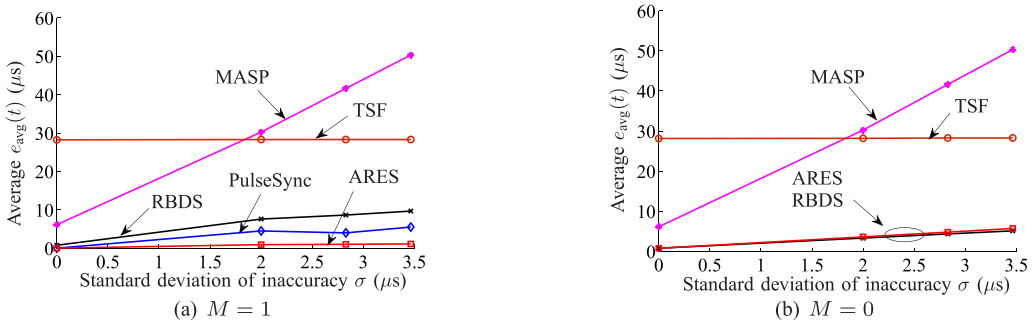


Fig. 9. Average synchronization errors (over time window [200, 400] s) versus σ, with $N = 80$ and $\Omega = 5$.

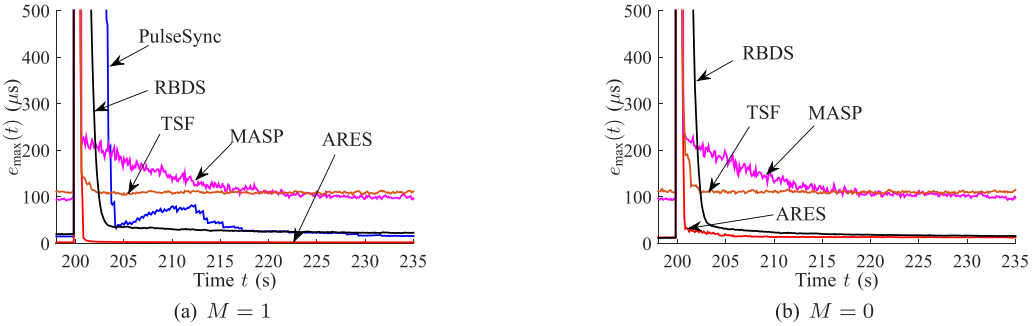


Fig. 10. Maximum synchronization errors when 5 new devices join at $t = 200$ s, with $N = 40$, $\sigma = 2$ μs, and $\Omega = 5$.

the proposed ARES clearly outperforms the other two schemes with respect to the speed of error decrease, especially for large networks. This observation shows the scalability of the ARES mechanism to network size.

Fig. 6(c) and Fig. 6(d) evaluate the average synchronization errors in the presence of timestamp inaccuracies, i.e., $\sigma = 2$ μs. First, it is revealed that the MASP deteriorates seriously under the scenario with inaccurate timestamps, which implies its infeasibility in reality. Second, the performances of RBDS, PulseSync, and ARES are somehow insensitive to the inaccuracies since their error curves are similar with those in Fig. 6(a) and Fig. 6(b) but with a bit higher error floor values. Besides, ARES shows clear superiority among these three schemes in terms of the remaining synchronization errors. Additionally, as shown in Fig. 6, ARES performs well under different values of N, which illustrates it good scalability within the considered range of network size.

To further evaluate the scalability of the synchronization schemes, we consider a network with linear graph topology. In a linear graph with N nodes, the nodes can be listed in the order $n_1, n_2, \ldots, n_N$ such that the edges are (n_i, n_{i+1}) where $i = 1, 2, \ldots, N-1$ and the network diameter is simply $N - 1$. In this way, Fig. 7 illustrates the average synchronization time (defined as the time it takes to achieve zero synchronization error) for different network diameters. From Fig. 7, we see that with increased network diameters, the superiority of ARES over PulseSync is more clear, which shows the better scalability of ARES.

Fig. 8 plots the average synchronization errors versus the evolved time for the out-of-coverage D2D scenario, i.e., $M = 0$. The PulseSync is excluded in this setup since it can only work for a network with leaders. As in Fig. 6, the synchronization methods are compared under different σ and N. Note that the performances of TSF and MASP are the same with those in Fig. 6. This is because TSF and MASP are designed based on the converge-to-max principle which will not be affected by the presence of a leader. Moreover, as observed, while the RBDS outperforms ARES by a very small degree in Fig. 8(d), they have quite close performances in general. Indeed, RBDS and ARES almost overlap with each other in Fig. 8(a)-Fig. 8(c). The reason is that, the only difference between RBDS and ARES in the out-of-coverage scenario lies in the weight design of the update rule (as explained in Section III-B). The new weight calculation used in ARES is to address the Challenge 3 where new devices join an almost synchronized group. This situation, however, does not occur in the simulation environment of Fig. 8.

To further investigate the influence of timestamp inaccuracies on synchronization schemes, Fig. 9 plots the average synchronization errors in terms of the standard deviation of inaccuracy, i.e., σ. From Fig. 9(a) where $M = 1$, we see that the error of MASP goes up sharply with increased σ, which again illustrates its sensitivity to the inaccurate timestamps. Besides, although the performance of TSF is stable over different σ, its error is fairly large. When it comes to the other three schemes, it is shown that the proposed ARES yields the best accuracy and robustness, the RBDS gives the worst performance, and the PulseSync is in the middle. This is reasonable since the RBDS is specifically designed for a network without any leader. Moreover, assuming the out-of-coverage D2D scenario, synchronization schemes are simulated in Fig. 9(b). First, the analysis of

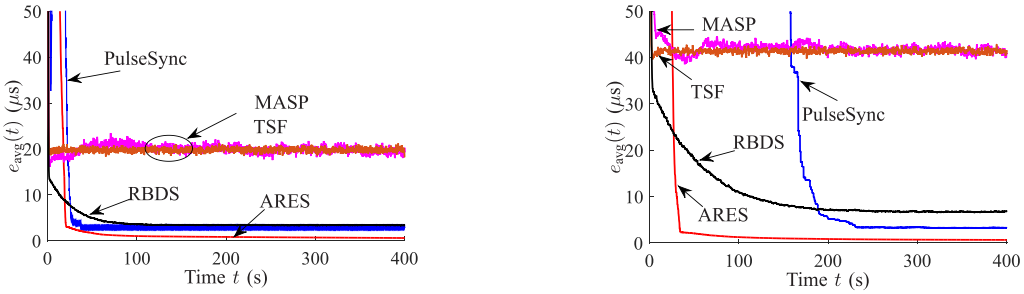


Fig. 11. Average synchronization errors versus time evolution with $N = 80$, $M = 1$, and $\sigma = 2$ μs.

TSF and MASP given above applies to the out-of-coverage scenario as well. Second, the RBDS and ARES are relatively robust to varied σ and they have almost overlapping error performance.

Due to the mobility in D2D networks, devices may appear and disappear over time. Hence, as stated in Challenge 3, the almost synchronous group should not be disturbed too much by new devices joining. Fig. 10 evaluates the maximum synchronization errors in this regard, where five new devices join the network at $t = 200$ s when the existing devices have already achieved a stable state. It is found that the proposed ARES attains again the synchronous state within a very short time period. For instance, it is around 2 s in Fig. 10(a) and 5 s in Fig. 10(b). However, the other schemes exhibit either slow error decrease or large error floor. Particularly, as shown in Fig. 10(b), even though no leader exists in this setting, the performance gap between RBDS and ARES is obvious during the time period from 201 s to 210 s. This is because the RBDS method gives the same weights to the timestamps from both transmitter and receiver when updating the local clock, which cannot well tackle the situation of new devices joining an almost synchronized group. On the other hand, the proposed ARES scheme derives new weights for the out-of-coverage scenario, as presented in Section III-B. The observation in Fig. 10(b) verifies the advantage of the new weight design in ARES and the ability of ARES to address Challenge 3.

To check the robustness of synchronization schemes against different network connectivity, i.e., different expected node degree Ω, we plot the average synchronization errors in Fig. 11(a) and Fig. 11(b) for $\Omega = 20$ (i.e., dense scenario) and $\Omega = 2$ (i.e., sparse scenario), respectively. By comparing the respective curves in these two figures, we see that the synchronization errors of TSF and MASP are quite sensitive to Ω. For example, their stable errors are roughly 20 s for $\Omega = 20$ whereas 40 s for $\Omega = 2$. Additionally, the convergence speeds of RBDS and PulseSync are seriously affected by Ω. The proposed ARES protocol, however, is robust to different Ω in terms of both synchronization accuracy and error decrease rate.

Note that we assume $M = 1$ for the partial-coverage D2D scenario in the above figures. In fact, we have also investigated different M in simulations. The results show that a larger M will yield a clear performance improvement of the ARES scheme. Since these phenomena are not surprising, we will not include the figures here to save space.

VI. Conclusions

In this paper, we investigated the synchronization problem for mobile D2D networks. Firstly, five main challenges imposed on the synchronization for partial-coverage and out-of-coverage D2D scenarios were discussed. Secondly, we proposed the ARES algorithm with low complexity to tackle the five challenges. Acknowledging the distinct characteristics of the partial-coverage and out-of-coverage scenarios, different design principles were utilized. Nevertheless, the ARES mechanism is able to adapt to the two scenarios without initially knowing what it is. Finally, comprehensive simulations were provided. The numerical results show the superiority of ARES over the other synchronization schemes. In particular, the ARES scheme achieves a good accuracy and a fast error decrease in both partial-coverage and out-of-coverage D2D scenario. Also, it exhibits robustness to various number of devices N, standard deviation σ of time stamp inaccuracy, and expected node degree Ω. Moreover, ARES is quite adaptive to the appearance of new devices, which is a common situation in mobile networks. Therefore, we conclude that the ARES algorithm nicely addresses the five challenges and thus is a very promising candidate for the synchronization in mobile D2D networks.

References

[1] X. Lin, J. Andrews, A. Ghosh, and R. Ratasuk, "An overview of 3GPP device-to-device proximity services," *IEEE Commun. Mag.*, vol. 52, no. 4, pp. 40–48, Apr. 2014.

[2] K. H. Lee, K. H. Won, and H. J. Choi, "Time synchronization method for device-to-device communication system," in *Proc. 7th Int. Conf. Ubiquitous Inf. Manag. Commun.*, New York, NY USA, Jan. 2013, p. 103.

[3] 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, Dec. 2013.

[4] K. Lee, K. Won, and H. Choi, "A fine timing synchronization method on group communication system enablers for LTE," in *Proc. 19th Asia–Pacific Conf. Commun. (APCC)*, Denpasar, Indonesia, Aug. 2013, pp. 21–25.

[5] W. Sun, M. R. Gholami, E. G. Ström, and F. Brännström, "Distributed clock synchronization with application of D2D communication without infrastructure," in *Proc. IEEE GLOBECOM Workshop*, Atlanta, GA USA, Dec. 2013, pp. 561–566.

[6] A. Osseiran, J. F. Monserrat, and P. Marsch, *5G Mobile and Wireless Communications Technology*. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[7] H. Seo, K. Lee, and S. Yasukawa, "LTE evolution for vehicle-to-everything services," *IEEE Commun. Mag.*, vol. 54, no. 6, pp. 22–28, Jun. 2016.
[8] G. W. Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," in *Proc. 3rd ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, San Diego, CA, USA, Nov. 2005, pp. 142–153.
[9] M. Morelli, C.-C. J. Kuo, and M. O. Pun, "Synchronization techniques for orthogonal frequency division multiple access (OFDMA): A tutorial review," *Proc. IEEE*, vol. 95, no. 7, pp. 1394–1427, Aug. 2007.
[10] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Standard 802.11, 1999.
[11] H. K. Pande, S. Thapliyal, and L. C. Mangal, "A new clock synchronization algorithm for multi-hop wireless ad hoc networks," in *Proc. 6th Int. Conf. Wireless Commun. Sensor Netw. (WCSN)*, Allahabad, India, Dec. 2010, pp. 1–5.
[12] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. 3rd ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, Los Angeles, CA, USA, Nov. 2003, pp. 138–149.
[13] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *Proc. 4th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, Baltimore, MD, USA, Nov. 2004, pp. 39–49.
[14] K. S. Yildirim and A. Kantarci, "Time synchronization based on slow-flooding in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 244–253, Jan. 2014.
[15] C. Lenzen, P. Sommer, and R. Wattenhofer, "PulseSync: An efficient and scalable clock synchronization protocol," *IEEE/ACM Trans. Netw.*, vol. 23, no. 99, pp. 717–727, Mar. 2014.
[16] 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.
[17] W. Sun, E. G. Ström, F. Brännström, and M. R. Gholami, "Random broadcast based distributed consensus clock synchronization for mobile networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 6, pp. 3378–3389, Jun. 2015.
[18] M. Leng and Y.-C. Wu, "Distributed clock synchronization for wireless sensor networks using belief propagation," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5404–5414, Nov. 2011.
[19] 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 Ad Hoc Sensor Conf.*, Washington, DC, USA, Nov. 2005, p. 558.
[20] D. Zhou and T. H. Lai, "An accurate and scalable clock synchronization protocol for IEEE 802.11-based multihop ad hoc networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 12, pp. 1797–1808, Dec. 2007.
[21] J. So and N. Vaidya, "MTSF: A timing synchronization protocol to support synchronous operations in multihop wireless networks," Univ. Illinois Urbana-Champaign, Champaign, IL, USA, Tech. Rep., Jan. 2004.
[22] K. S. Yildirim, R. Carli, and L. Schenato, "Adaptive control-based clock synchronization in wireless sensor networks," in *Proc. Eur. Control Conf. (ECC)*, Linz, Austria, Jul. 2015, pp. 2806–2811.
[23] B. Laursen, T. D. Little, and N. A. Card, Eds., *Handbook of Developmental Research Methods*. New York, NY, USA: Guilford Press, 2013.
[24] P. Erdös and A. Rényi, "On random graphs," *Mathematicae*, vol. 6, pp. 290–297, 1959.

Wanlu Sun received the B.E. degree in communication engineering and the M.E. degree in communication and information systems from the Beijing University of Posts and Telecommunications, Beijing, China, in 2008 and 2011, respectively, and the Ph.D. degree from the Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden, in 2016. In 2013, she was a Visiting Student with the Wireless Networking and Communication Laboratory, Texas A&M University, College Station, TX, USA. In 2014, she was a Visiting Student in mobile telecommunications with the Department of Science and Technology, Linköping University, Sweden.

Fredrik Brännström (S'98–M'05) received the M.Sc. degree in electrical engineering from Luleå University of Technology, Luleå, Sweden, in 1998, the Lic.Eng. and Ph.D. degrees in communication theory from the Department of Computer Engineering, Chalmers University of Technology, Gothenburg, Sweden, in 2000 and 2004, respectively, and the Docent degree in communication systems from the Department of Signals and Systems, Chalmers University of Technology, in 2012. He was with the Institute for Telecommunications Research, University of South Australia, Adelaide, Australia, as a Visiting Researcher, in 2001, 2002, 2003, and 2005. From 2004 to 2006, he was a Post-Doctoral Researcher with the Communication Systems Group, Department of Signals and Systems, Chalmers University of Technology. From 2006 to 2010, he was a Senior Algorithm Engineer and a Principal Design Engineer with Quantenna Communications, Inc., Fremont, CA, USA. In 2010, he joined the Department of Signals and Systems, Chalmers University of Technology, where he is currently a Professor with the Communication Systems Group. His research interests in communication and information theory include code design, coded modulation, labelings, and coding for distributed storage, as well as algorithms, resource allocation, synchronization, and protocol design for vehicular communication systems. He was a recipient of the 2013 IEEE Communication Theory Workshop Best Poster Award. In 2014, he received the Department of Signals and Systems Best Teacher Award. He has co-authored a paper that received the 2016 IEEE Sweden VT-COM-IT Joint Chapter Best Student Conference Paper Award.

Erik G. Ström (S'93–M'95–SM'01) received the M.S. degree in electrical engineering from the Royal Institute of Technology (KTH), Stockholm, Sweden, in 1990, and the Ph.D. degree in electrical engineering from the University of Florida, Gainesville, in 1994. He held a post-doctoral position with the Department of Signals, Sensors, and Systems, KTH, in 1995. In 1996, he was an appointed Assistant Professor with KTH. In 1996, he joined the Chalmers University of Technology, Göteborg, Sweden, where he has been a Professor in Communication Systems since 2003. He currently heads the Division for Communications Systems and leads the competence area Sensors and Communications at the traffic safety center SAFER, which is hosted by Chalmers. Since 1990, he has been a Consultant with the Educational Group, Individual Development, Stockholm, Sweden. His research interests include signal processing and communication theory in general, and constellation labelings, channel estimation, synchronization, multiple access, medium access, multiuser detection, wireless positioning, and vehicular communications in particular. He was a member of the Board of the IEEE VT/COM Swedish Chapter from 2000 to 2006. He received the Chalmers Pedagogical Prize in 1998 and the Chalmers Ph.D. Supervisor of the Year award in 2009. He is a contributing author and an Associate Editor of Roy. Admiralty Publishers FesGas series. He was a Co-Guest Editor of the PROCEEDINGS OF THE IEEE Special Issue on Vehicular Communications in 2011 and the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS Special Issues on Signal Synchronization in Digital Transmission Systems in 2001 and on Multiuser Detection for Advanced Communication Systems and Networks in 2008.