Consensus Tracking-based Clock Synchronization for the Internet of Things

yuqing niu (niuyuq@foxmail.com)
Tianjin University https://orcid.org/0000-0002-7085-0305
Ting Yang
Tianjin University
yucheng hou
Tianjin University
shaotang cai
Tianjin University
peng yan
Tianjin University

Research Article

Keywords: Consensus tracking, clock synchronization, logical clock, Internet of Things

Posted Date: April 5th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-314529/v1

Consensus tracking-based clock synchronization for the Internet of Things

Yuqing Niu, Ting Yang, Yucheng Hou, Shaotang Cai, Peng Yan

Abstract

Clock synchronization provides fundamental timing reference for various services in the Internet of Things (IoT). Unlike centralized synchronization algorithm, which relies heavily on the synchronization routing and brings about error accumulation consequently, the distributed synchronization algorithm achieves clock synchronization through time message exchange and calculation among adjacent nodes, hence the error accumulation can be greatly eliminated. However, the convergence speed of existing distributed synchronization algorithms is generally slow, and network topology information is seldom utilized to improve the convergence speed, which may cause additional energy consumption especially in resource constrained scenarios. To improve the convergence speed of distributed clock synchronization, a consensus tracking-based clock synchronization (CTCS) algorithm is proposed in this paper. By analyzing the synchronization process in the state space framework, the convergence acceleration term is designed to optimize the eigenvalue distribution of synchronization error matrix, hence the convergence speed can be greatly improved. To be suitable for clock synchronization in the dynamic network, a reference clock input term is designed in the clock update formula, which eliminates the oscillation of synchronized clocks triggered by synchronization of newly joined nodes. To prove the convergence of CTCS, an in-depth analysis of the algorithm is conducted. Both simulation and experimental results validate the effectiveness of CTCS in comparison with other solutions.

Keywords Consensus tracking, clock synchronization, logical clock, Internet of Things.

1 Introduction

The Internet of Things (IoT) integrates sensor, computer, and wireless communication technologies, and it connects intelligent terminals into a huge network, which collects and analyzes key data generated during the operation of physical world (Manavalan and Jayakrishna 2019; Akhlaq and Sheltami 2013; Yildirim and Kantarci 2013). At present, IoT has been widely used in the fields of industrial automation, environmental detection, and intelligent transportation (Lamonaca et al. 2014; Kadowaki and Ishii 2014; Tahaei et al. 2020).

Clock synchronization is essential to the normal operation of IoT. For example, most basic applications, such as wireless communication protocols, low-power sleep dispatch, and data fusion, all need to execute in a unified time frame (Xiong et al. 2017). Besides, in advanced IoT-based applications, such as microgrid control, it is necessary to synchronize the clock of control units. Since asynchronous output voltage will cause parallel circulating currents, which will seriously reduce the operation efficiency and stability of microgrid.

Currently, clock synchronization algorithms could be divided into centralized synchronization and distributed synchronization. In centralized synchronization，clocks are synchronized by hierarchy through synchronization routing (Ganeriwal et al. 2003; Elson et al. 2002; Noh. et al. 2007). With the expansion of the network scale, routing hierarchy naturally increases and brings a serious accumulation of synchronization error. Tests based on Berkeley motes show that the average synchronization error of single-hop is 45.2μs, nevertheless, the accumulated error after five hops increases to 73.6 μs (Ganeriwa et al. 2003). Furthermore, in the IoT with a dynamical network scale, it is necessary to consider the additional resource overhead brought by the reconstruction of synchronization routing.

Communicated by yangting

E-mail address: {niuyuqing8051, yangting, houyucheng, caishaotang1992, yanpeng3010294}@tju.edu.cn.

Key Laboratory of Smart Grid of Ministry of Education, School of Electrical Automation and Information Engineering, Tianjin University, Tianjin 300072, China

To overcome the disadvantages of centralized synchronization, scholars have proposed distributed synchronization algorithms in recent years (Su and Akyildiz 2015; Schenato and Gamba 2007; Maggs et al. 2012), which have greatly reduced the accumulation of synchronization error by performing routing-independent time message transmission, hence a more uniform synchronization accuracy is achieved. However, in practical applications, there are still several shortcomings for existing distributed synchronization algorithms: firstly, the convergence speed is slow and greatly restricted by network connectivity, which leads to large consumption of battery power and long-term occupation of wireless channels. Secondly, there is a lack of explicit reference clock in existing distributed synchronization algorithms. Whenever newly joined nodes trigger the execution clock synchronization, an implicit reference time will be reconstructed by negotiating with the synchronized clocks in the network, which results in discontinuous oscillation of the value of synchronized clocks, hence the normal operation of upper layer IoT applications are greatly affected.

Inspired by consensus tracking principle (Wang et al. 2016), this paper proposes the consensus tracking clock synchronization (CTCS). To overcome the aforementioned disadvantages of existing algorithms, CTCS has designed convergence acceleration term as well as reference clock input term to improve the synchronization performances. In light of most basic IoT functions are operated based on logical clocks, such as TDMA communication protocol, CTCS sets up the logical clock and synchronizes both logical drift rate and logical offset jointly. The main contributions of CTCS are summarized as follows:

1) Based on consensus tracking and dynamic system stability theory, this paper maps the clock synchronization process to a dynamic system in the state space framework, the functional relationship between eigenvalues of synchronization error matrix and convergence speed of CTCS is analyzed. Then we derive the optimal value of convergence acceleration term and optimize the distribution of eigenvalues with it, hence the convergence speed of clock synchronization could be accelerated.

2) We also design the reference clock input term in CTCS, by which the newly joined nodes are synchronized based on the current network clock, hence the synchronized logical clocks no longer participate in the clock parameter update triggered by newly joined nodes in the current distributed algorithms, and the oscillation of synchronized clocks is avoided consequently. Moreover, parameters of reference clock may be affected by factors such as ambient temperature and operating voltage, with reference clock input term, CTCS could track the change of reference clock parameters, so as to keep the long-term synchronization for network clocks.

The rest of this paper is organized as follows. In Section 2, related works in the field of clock synchronization are introduced. Section 3 gives mathematical models of logical clock and IoT network topology. Section 4 presents the CTCS algorithm in detail. Section 5 gives an analysis of parameter settings and performance of CTCS. Various simulations are carried out in section 6, and section 7 concludes the paper.

2 Related works

In practical IoT applications, it is often necessary to deploy a large number of nodes, due to cost constraints，most IoT nodes are equipped with low-precision clocks. As time goes by, there will be a great clock deviation between network nodes, which will affect the normal operation of network functions (Lévesque and Tipper 2016). To solve this problem, scholars have proposed a variety of clock synchronization algorithms, which could be divided into centralized and distributed synchronization according to the network topology required for the execution of algorithms (Djenouri and Bagaa 2014; Swain and Hansdah 2015; Carli and Zampieri 2013).

Some typical centralized clock synchronization algorithms and protocols include the timing-sync protocol for sensor networks (TPSN) (Ganeriwal et al. 2003), reference broadcasts synchronization (RBS) (Elson et al. 2002), delay measurement time synchronization (Ping 2003), flooding time synchronization protocol (Maróti 2004) and time synchronization with immediate clock adjustment (TSICA) (Wang 2017), etc. In order to further improve the synchronization accuracy, statistical signal processing techniques are introduced into clock parameter estimation, such as maximum likelihood estimator (MLE), low-complexity maximum likelihood estimator (LCMLE) (Leng and Wu 2011) and minimum variance unbiased estimator (MVUE) (Chaudhari et al. 2010).These clock synchronization algorithms are usually designed for clock synchronization in the small-scale IoT, and they are

characterized by establishing and maintaining a hierarchical synchronization routing, such as multi-layer tree or star topology, through which time messages are exchanged.

With the development of communication technology, the monitoring range of IoT has expanded from a local area to a much wider area, resulting in continuous growth of network scale. In large-scale IoT, the routing topology of centralized clock synchronization contains too many layers, which leads to serious synchronization error accumulation and uneven synchronization accuracy (Maggs 2012), hence the quality of IoT services based on clock synchronization is reduced. To meet the high-precision clock synchronization requirement in IoT, scholars have proposed distributed clock synchronization algorithms, in which network nodes only exchange time information with their neighbor nodes, thus there is no need to maintain a complex synchronization routing, and the accumulation of synchronization errors is greatly reduced.

In time diffusion synchronization protocol (TDP) (Su and Akyildiz 2005), IoT nodes are randomly selected as the reference clock node, and a bifurcation tree topology is constructed to achieve the clock synchronization. To maintain the unidirectionality and continuity in the register value update process in the physical clock, average time synchronization (ATS) (Schenato and Fiorentin 2011) propose the concept of logical clock and achieve clock synchronization by exchanging and calculating clock parameters iteratively. Consensus clock synchronization (CCS) (Maggs 2012) iteratively update the clock with a weighted average of clock values of its neighbor nodes, and network nodes are allocated with different weights to reduce the iteration times. In maximum-value based time synchronization (MTS) (He et al. 2014a), the neighbor node with maximum clock drift rate is selected as the reference clock, the network time is synchronized to the fastest clock by comparing and fusing clock value iteratively. However, the clock drift rate synchronized by MTS is relatively high, in maximum-minimum time synchronization (MMTS) (He et al. 2014b), the minimum consensus is used to optimize the clock drift rate during the execution of MTS.

Compared with centralized synchronization, distributed synchronization requires much more time message transmission and computation, thus consumes greater node battery power. In order to achieve higher energy efficiency, cluster-based consensus time synchronization (CCTS) (Wu et al. 2014) applies the clustering mechanism and uses cluster head as the time message fusion center, which reduces the number of time messages sent and received. Cluster-based maximum consensus time synchronization (CMTS) (Wang et al. 2017) combine the clustering mechanism with MTS, which reduces the effect of initial random clock difference on the synchronization speed and improves the energy efficiency.

However, the convergence speed of existing distributed clock synchronization algorithms is affected by inherent network connectivity. For example, in a fixed-scale network, the convergence time of clock synchronization increases significantly with the decrease of network connectivity (Wu et al. 2014). To improve the network connectivity, a network multi-hop controller is designed (Shi et al. 2019), and neighbor nodes could switch between forwarding mode and broadcasting mode. Nevertheless, the forwarding mode dramatically increases the time message transmission delay, which will affect the convergence of the algorithm (Franklin et al. 2015). In addition, there is a lack of explicit reference clocks in existing distributed synchronization algorithms. For example, in CCTS, a weighted average of the initial clock value of IoT nodes is set as the implicit reference time for clock synchronization. As a consequence, the synchronization of newly joined nodes will trigger the oscillation of current implicit network reference time, which could further bring about the incorrect mapping between IoT timestamp and the actual timing sequence of IoT events.

To solve the above problems, this paper applies the consensus tracking principle to clock synchronization and proposes CTCS. By designing convergence acceleration term and reference clock input term in the update process of logical clock parameters, complex strategies are avoided to change the network topology, and clock synchronization speed is accelerated in the case of weak network connectivity. Meanwhile, the oscillation of synchronized clocks triggered by the synchronization of newly joined nodes is also eliminated, making CTCS suitable for clock synchronization in the IoT with a large and dynamical network scale.

3 IoT topology and logical clock model

In this section, we use graph theory to model the IoT topology, and introduce logical clock based on physical clock model.

3.1 IoT topology model

The network topology in IoT can be represented by graph $G=(V,E)$, where $V=\{v_1, v_2,\ldots,v_n\}$ is the set including all the IoT nodes. Elements in E can be expressed as $e_{ij}=(v_i, v_j)$, which represents the communication link between v_i and v_j. In practice, IoT nodes generally adopt full duplex communication mode, thus we assume that graph G is undirected ($e_{ij}\in E \Rightarrow e_{ji}\in E$). All the nodes in the single-hop range of node v_i are defined as its neighbor nodes, which are represented as the set $N_i=\{ v_j : (v_i, v_j)\in E \}$, and the number of elements in this set is defined as node degree d_i. The adjacency matrix $\mathbf{A}=[a_{ij}]$ represent the adjacency relationship of network nodes, where a_{ij} is the corresponding weight of the edge e_{ij}. If $a_{ij}=0$, then $e_{ij}\notin E$. Due to the power attenuation of wireless signal, most IoT nodes cannot perform single-hop communication with reference clock, hence the corresponding edge weight should be set to 0. Finally, we give the definition of Laplacian matrix, that is $\mathbf{L}=\mathbf{D}-\mathbf{A}=[l_{ij}]$, where $\mathbf{D}=\mathrm{diag}(d_i)$ and $\sum_{j=1}^{n} l_{ij}=0$.

3.2. Logical clock model

The relationship between local clock C i(t) of node i and reference time t is given by (Choi et al. 2011)

$$C_i(t)=\omega_i t+\beta_i \tag{1}$$

Where β_i is the clock offset which indicates the difference between $C_i(t)$ and reference time t. ω_i is the clock drift rate, and due to some random factors in the manufacturing process, the clock drift rate of different nodes is slightly different, which is generally 10-30 ppm，hence the clock differences between nodes will continue to expand as time goes by. Since the value of ω_i is an inherent property of clock hardware, it is difficult to be directly adjusted by traditional IoT clock synchronization algorithms, and periodic execution of synchronization algorithm is usually required to maintain the long-term clock synchronization.

Instead of providing the absolute time, network clock is mostly used to provide reference timing sequence for various tasks in IoT. Therefore, we establish the logical clock model based on physical clock in (1), and both logical drift rate and logical offset could be synchronized jointly, which compensates for the disadvantage of traditional clock synchronizations. Defining logical clock $C_i^l(t)$ of node i as the function of physical clock $C_i(t)$, we have

Table 1 Notation definitions

Symbol	Definition
$\omega_i^l(k)$	Logical drift rate at time k
$\beta_i^l(k)$	Logical offset at time k
$s_i^l(k)$	Drift rate compensation parameter at time k
$o_i^l(k)$	Offset compensation parameter at time k
ω_r	Logical reference drift rate
β_r	Logical reference offset
$C_i^l(k)$	Logical clock value at time k
$\omega_{ij}(k)$	Relative drift rate between node i and j at time k

$$\begin{aligned} C_i^l(t) &= s_i^l C_i(t) + o_i^l \\ &= s_i^l \omega_i t + s_i^l \beta_i + o_i^l \\ &= \omega_i^l t + \beta_i^l \end{aligned} \tag{2}$$

Where s_i^l is drift rate compensation parameter, o_i^l is offset compensation parameter, ω_i^l is the logical drift rate and β_i^l is the logical offset. Defining the reference logical clock as

$$C_r(t)=\omega_r t+\beta_r \tag{3}$$

Where ω_r is reference drift rate, β_r is reference offset. By comparing (2) with (3), it can be seen that $C_i^l(t)$ could be synchronized to $C_r(t)$ if and only if the following equations are satisfied.

$$\begin{cases} \omega_i^l = s_i^l \omega_i = \omega_r \\ \beta_i^l = s_i^l \beta_i + o_i^l = \beta_r \end{cases} \tag{4}$$

4 Consensus Tracking Clock Synchronization

In this section, CTCS is proposed to achieve clock synchronization in IoT. According to (4), clock synchronization can be decomposed into two independent processes: drift rate synchronization and offset synchronization. In order to avoid additional synchronization error caused by drift rate deviation during offset synchronization, drift rate synchronization will be executed first. In practice, network node can't directly perceive local clock drift rate, however, the relative drift rate could be measured. By using the relative drift rate, the logical drift rate could be synchronized indirectly. Therefore, relative drift rate estimation is performed during drift rate synchronization process. In Table 1, we give the notation of main variables involved in CTCS.

4.1 Relative drift rate estimation

The estimation of relative clock drift rate involves interception, transmission and calculation of time message,

which is inevitably affected by various noises. The first order discrete low pass filter could be used to process the time messages (Schenato and Gamba 2007), however, its parameters are difficult to modify dynamically. For Gaussian noise, Kalman filter could minimize the mean square error of the estimation, and for non-Gaussian noise, it is the optimal linear estimator. Meanwhile in Kalman filter, parameters to be estimated are only related to the state of previous moment, which shows a superior real-time performance. In view of this, we adopt Kalman filter to estimate the relative clock drift rate $\omega_{ij}(k)$. According to first-order AR model (Hamilton et al. 2008), the dynamical process of $\omega_{ij}(k)$ is given by

$$\hat{\omega}_{ij}(k)=\rho\hat{\omega}_{ij}(k-1)+w(k) \tag{5}$$

Where $\hat{\omega}_{ij}(k)$ is the estimation of $\omega_{ij}(k)$ at time k. since $\omega_{ij}(k)$ is a scalar, the state transition matrix is reduced to scalar ρ, which is generally set as 1-100×10^{-6} (Zhou et al. 2019). $w(k)$ is the estimation noise. The estimation process includes state prediction and state update, and state prediction is given by

$$\hat{\omega}_{ij}(k)=\rho\hat{\omega}_{ij}(k-1) \tag{6}$$

$$\hat{P}(k)=\rho^2\hat{P}(k-1)+\sigma_e^2, \tag{7}$$

Where $\hat{P}(k)$ is estimation error variance, σ_e^2 is estimation noise variance. The status update is given by

$$\omega_{ij}^K(k)=\hat{\omega}_{ij}(k)+K(n)(y(k)-\rho\hat{\omega}_{ij}(k)) \tag{8}$$

$$y[k]=\frac{\omega_i(k+1)}{\omega_j(k+1)}=\frac{C_i(k+1)-C_i(k)}{C_j(k+1)-C_j(k)} \tag{9}$$

$$K(k)=\frac{\hat{P}(k)}{\hat{P}(k)+\sigma_m^2} \tag{10}$$

$$\hat{P}^+(k)=(1-K(n))\hat{P}(k) \tag{11}$$

Where $\omega_{ij}^K(k)$ is the Kalman estimation of $\omega_{ij}(k)$, $y(k)$ is the observation of relative clock drift rate with measurement noise, $C_i(k)$ is the logical clock value of node i, σ_m^2 is measurement noise variance, $K(n)$ is the Kalman gain, and $\hat{P}^+(k)$ is the optimal estimation error variance.

4.2 Synchronization of logical drift rate

In order to accelerate the convergence speed and overcome the disadvantages of existing distributed synchronization algorithms, based on consensus tracking principle (Bu et al. 2017), we design and add convergence acceleration term σ, reference drift rate term ω_r and sampling period term ε to improve the clock update formula given in CCTS, hence the logical drift rate synchronization formula in CTCS is given by

$$\begin{aligned}\omega_i^l(k+1)=&(1-\sigma)\omega_i^l(k)+\varepsilon(\omega_r-\omega_i^l(k))\\&+\sum_{j=1}^{n}\varepsilon a_{ij}(\omega_j^l(k)-\omega_i^l(k))\end{aligned} \tag{12}$$

Since node i cannot obtain its logical drift rate directly, therefore, (12) can't be used directly in the logical drift rate synchronization. However, synchronization could still be achieved by updating logical drift rate compensation parameters $s_j^l(k)$. First, by rearranging the coefficients of $\omega_i^l(k)$ in equation (12), we have

$$\begin{aligned}\omega_i^l(k+1)&=(1-\varepsilon-\sigma)\omega_i^l(k)+\sum_{j=1}^{n}\varepsilon a_{ij}(\omega_j^l(k)-\omega_i^l(k))+\varepsilon\omega_r\\&=\omega_i^l(k)\sum_{j=1}^{n}(\frac{1-\varepsilon-\sigma}{n}-a_{ij})+\sum_{j=1}^{n}\varepsilon\omega_j^l(k)a_{ij}+\varepsilon\omega_r\end{aligned} \tag{13}$$

substituting (4) into (13), we have

$$\begin{aligned}s_i^l(k+1)\omega_i=&s_i^l(k)\omega_i\sum_{j=1}^{n}(\frac{1-\varepsilon-\sigma}{n}-a_{ij})\\&+\sum_{j=1}^{n}\varepsilon\omega_{ji}^K\omega_i a_{ij}s_j^l(k)+\varepsilon\omega_{ri}^K\omega_i\end{aligned} \tag{14}$$

Dividing ω_i at both sides, we have

$$\begin{aligned}s_i^l(k+1)=&s_i^l(k)\sum_{j=1}^{n}(\frac{1-\varepsilon-\sigma}{n}-a_{ij})\\&+\sum_{j=1}^{n}\varepsilon\omega_{ji}^K a_{ij}s_j^l(k)+\varepsilon\omega_{ri}^K\end{aligned} \tag{15}$$

It can be seen that the update process of $s_i^v(k)$ is equivalent to the synchronization process of $\omega_i^l(k)$. Then we design the following logical drift rate synchronization process, and by executing step 1 through step 3 iteratively, ω_i^l can be synchronized to the reference logical drift rate ω_r. The specific process is as follows:

Initialization. every IoT node reports the node number of its neighbor within the Single-hop communication range to the reference clock node, then the reference clock node calculates the value of convergence acceleration term σ and forwards it to all the nodes.

Step 1. Node i: broadcasting time messages that include time stamp information of physical clock $C_i(k)$ in the period of ε. According to different node type, the following update processes of logical clock parameter are performed as soon as the time message is received.

Step 2(*a*). Reference logical clock node: recording the change of $C_r(k)$ and $C_i(k)$in the period of ε, then calculating the relative clock drift rate ω_{ri}^K between $C_r(k)$ and $C_i(k)$, finally, sending ω_{ri}^K back to node i.

Step 2(*b*). Neighbor node *j*: recording the change of $C_j(k)$ and $C_i(k)$ in the period of ε, then calculating the relative clock drift rate ω_{ji}^K between $C_j(k)$ and $C_i(k)$, finally, sending both ω_{ji}^K and drift rate compensation parameter $s_j^l(k)$ of node *j* back to node *i*.

Step 3. Node *i*: when receiving ω_{ri}^K, ω_{ji}^K and $s_j^l(k)$, performing update of $s_i^l(k)$ according to (15).

If all the node clocks update their logical drift rate according to (15), which is equivalent to (12), theorem 4.1 shows that $\omega_i^l(k)$（$1\le i\le n$）could be synchronized to ω_r. Since we analyze the synchronization process in the state space framework, $\omega_i^l(k)$, $\omega_j^l(k)$ and ω_r in (12) are being respectively substituted with standard state variables $x_i(k)$, $x_j(k)$ and x_r and rewritten in (16) to clarify the following proof.

Definition 4.1. In the state space, defining the state of node *i* as $x_i(k)$，and the reference state as x_r, then the state vector of all nodes ($1\le i\le n$) at time *k* is $\mathbf{x}(k)=[\,x_1(k), x_2(k), \cdots, x_n(k)]^T$, the reference state vector is $\mathbf{x}_r=[\,x_r, x_r, \cdots, x_r]^T$, the synchronization error vector is $\mathbf{e}(k)=\mathbf{x}_r-\mathbf{x}(k)$. During synchronization, state of node is updated according to following formula,

$$x_i(k+1)=(1-\varepsilon-\sigma)x_i(k)+\sum_{j=1}^{n}\varepsilon a_{ij}(x_j(k)-x_i(k))+\varepsilon x_r \tag{16}$$

Theorem 4.1. In the IoT with *n* nodes and bidirectional communication links，all the network nodes update their state according to (16). Defining the synchronization error matrix as $\mathbf{L}^*=[(1-\varepsilon-\sigma)\mathbf{I}_n-\varepsilon\mathbf{L}]$, If all the eigenvalues of $\mathbf{L}^*$ lie inside the unit circle in the complex plane, then the state vector $\mathbf{x}(k)$ will converge to the reference vector $\mathbf{x}_r$.

Proof. In order to show the update process of all the node status at the same time, (16) is extended to the matrix form and can be rewritten as

$$\mathbf{x}(k+1)=[(1-\varepsilon-\sigma)\mathbf{I}_n-\varepsilon\mathbf{L}]\mathbf{x}(k)+\varepsilon\mathbf{x}_r \tag{17}$$

Where $\mathbf{I}_n$ is the identity matrix of order *n*. Substituting the definition of $\mathbf{e}(k)$ into (17), we have

$$\mathbf{e}(k+1)=[(1-\varepsilon-\sigma)\mathbf{I}_n-\varepsilon\mathbf{L}]\mathbf{e}(k)+\varepsilon\mathbf{L}\mathbf{x}_r \tag{18}$$

Since row elements of Laplacian matrix satisfies $\sum_{j=1}^{n}l_{ij}=0$, and all the elements in $\mathbf{x}_r$ are equal, hence $\varepsilon\boldsymbol{L}\boldsymbol{x}_r=0$, then we have

$$\mathbf{e}(k+1)=[(1-\varepsilon-\sigma)\mathbf{I}_n-\varepsilon\mathbf{L}]\mathbf{e}(k) \tag{19}$$

Defining the synchronization error matrix as $\mathbf{L}^*=[(1-\varepsilon-\sigma)\mathbf{I}_n-\varepsilon\mathbf{L}]$, and according to the system stability criterion in control theory (Franklin 2015), the convergence of $\mathbf{L}^*$ is determined by the distribution of its eigenvalues. That is, if all the solutions of characteristic equation ($\det[\lambda\mathrm{I}-\mathbf{L}^*]=0$) lie inside the unit circle in the complex plane, then $\lim_{k\to\infty}\mathbf{e}(k)=\mathbf{0}$, which is equivalent to $\lim_{k\to\infty}\mathbf{x}(k)=\mathbf{x}_r$ by the definition of $\mathbf{e}(k)$.

4.3 Synchronization of logical offset

When logical drift rate of all the nodes is synchronized, the logical offset of node *i* can be expressed as

$$\beta_i^l=\frac{\omega_r}{\omega_i}\beta_i+o_i^l \tag{20}$$

In practice, node *i* cannot obtain its logical offset directly, however, the synchronization of $\beta_i^l(k)$ to β_r could still be achieved indirectly by updating logical offset compensation parameters $o_i^l(k)$ according to (21) and is given by

$$o_i^l(k+1)=o_i^l(k)+(\varepsilon+\sigma)(C_r(k)-C_j^l(k))-\sigma\beta_r+\varepsilon\sum_{j=1}^{n}(C_j^l(k)-C_i^l(k)) \tag{21}$$

To show the update process of $o_i^l(k)$ is equivalent to the synchronization process of $\beta_i^l(k)$, by adding $\omega_r\beta_i/\omega_i$ to both sides of the (21), we have

$$\frac{\omega_r}{\omega_i}\beta_i+o_i^l(k+1)=(1-\sigma)[\frac{\omega_r}{\omega_i}\beta_i+o_i^l(k)]+\varepsilon(C_r(k)-C_j^l(k))+\varepsilon\sum_{j=1}^{n}a_{ij}(C_j^l(k)-C_i^l(k)) \tag{22}$$

When logical drift rate is synchronized, from (2) and (3), the logical clock of node *i* can be expressed as

$$C_i^l=\omega_r t+\beta_i^l$$

Hence we have

$$C_r(k)-C_j^l(k)=\beta_r-\beta_j^l(k) \tag{23}$$

In a similar way

$$C_j^l(k)-C_i^l(k)=\beta_j^l(k)-\beta_i^l(k) \tag{24}$$

Substituting (23) and (24) into (21), we have

$$\beta_i^l(k+1)=(1-\sigma)\beta_i^l(k)+\varepsilon(\beta_r-\beta_i^l(k))+\varepsilon\sum_{j=1}^{n}a_{ij}(\beta_j^l(k)-\beta_i^l(k)) \tag{25}$$

By combining the coefficients of $\beta_i^l(k)$ and letting $\beta_i^l(k)=x_i(k)$, $\beta_r=x_r$, it can be seen that (25) is equivalent to (16), hence β_i^l can be synchronized to β_r by Theorem 4.1.

5. Performance analysis of CTCS

In this section, the performance of CTCS algorithm is analyzed, including network connectivity constraints, convergence speed, optimal value of convergence acceleration term and response of CTCS to input disturbance.

5.1 Maximum network connectivity constraint

The convergence of the distributed synchronization algorithm is closely related to the network connectivity, thus we analyze the relationship between the maximum network connectivity d_{max} and the convergence of CTCS, and following conclusions are obtained.

Corollary 5.1 The sufficient condition for Theorem 4.1 is that the maximum network connectivity satisfies $d_{max}<(2-\varepsilon-\sigma)/2\varepsilon$.

Proof. By Theorem 4.1, to ensure the convergence of the CTCS, all the eigenvalues of $\mathbf{L}^*$ should lie inside the unit circle in the complex plane. Since $\mathbf{L}^*$ is real and symmetric, all its eigenvalues are real, thus the convergence condition is simplified to $|\lambda_i^*|<1$. Defining λ_i as the eigenvalue of $\mathbf{L}$, and according to the relationship between matrix and corresponding eigenvalues (Zhang 2017), we have

$$\lambda_i^*=(1-\varepsilon-\sigma)-\varepsilon\lambda_i \tag{26}$$

By substituting (26) into constraints of λ_i^*, we have

$$-1<(1-\varepsilon-\sigma)-\varepsilon\lambda_i<1 \tag{27}$$

By Geršgorin Circle Criterion (Horn 2012), The distribution of λ_i satisfies

$$-d_{max}<\lambda_i-d_{max}<d_{max} \tag{28}$$

the following two inequations are obtained by applying identical transformation to (27) and (28), then we have

$$0<\varepsilon+\sigma+\varepsilon\lambda_i<2 \tag{29}$$

$$0<\varepsilon+\sigma+\varepsilon\lambda_i<\varepsilon+\sigma+2\varepsilon d_{max} \tag{30}$$

Since (30) must satisfy the convergence criterion given by (29), that is, $\varepsilon+\sigma+2\varepsilon d_{max}\leq 2$, thus the maximum network connectivity should satisfies $d_{max}\leq(2-\varepsilon-\sigma)/2\varepsilon$.

5.2 Convergence speed analysis

Convergence speed is a key criterion for evaluating the algorithm performance，with modal decomposition (Brogan 1991)，we get the relationship between algorithm convergence speed and eigenvalue distribution of $\mathbf{L}^*$.

$$\mathbf{e}(k)=\mathbf{M}\Lambda^k\mathbf{M}^{-1}\mathbf{e}(0)=\sum_{i=1}^{n}v_i(\lambda_i^*)^k w_i^T\mathbf{e}(0) \tag{31}$$

Where $\mathbf{e}(0)$ is the vector of initial synchronization error. Since $\mathbf{L}^*$ is real and symmetric, it can be diagonalized to get $\mathbf{\Lambda}=\mathbf{M}\mathbf{L}^*\mathbf{M}^{-1}$, with λ_i^* being diagonal elements. $\mathbf{M}=[v_1\ v_2\ \dots\ v_n]$ and v_i is right eigenvector of $\mathbf{L}^*$, $(\mathbf{M}^{-1})^T=[w_1\ w_2\ \dots\ w_n]$ and w_i is left eigenvector of $\mathbf{L}^*$. Taking the limit of (31), we have

$$\begin{aligned}\lim_{k\to\infty}\mathbf{e}(k)=&\lim_{k\to\infty}v_1(\lambda_1^*)^k w_1^T e_1(0)+\cdots\\&+\lim_{k\to\infty}v_n(\lambda_n^*)^k w_n^T e_n(0)\end{aligned} \tag{32}$$

By arranging the modulus of eigenvalues of $\mathbf{L}^*$ in descending order, we have $\left|\lambda_1^*\right|\geq\left|\lambda_2^*\right|\geq\cdots\geq\left|\lambda_n^*\right|$. From (32), it can be seen that the convergence speed of $\mathbf{e}(k)$ is mainly determined by $|\lambda_1^*|$ and increases as $|\lambda_1^*|$ decreases. In the following sections, λ_1^* is represented by $\lambda_{\max}^*$ for the convenience of analysis.

5.3 Optimal value of convergence acceleration term

It can be seen from (26) that the eigenvalue distribution of $\mathbf{L}^*$ is related to the convergence acceleration term σ, sampling period ε and eigenvalues of $\mathbf{L}$. However, the sampling period ε is fixed during the synchronization, and λ_i is determined by network topology. Hence the eigenvalue distribution of $\mathbf{L}^*$ could only be optimized by σ. By selecting the appropriate value for convergence acceleration term, $\lambda_{\max}^*$ could be placed closer to the origin of complex plane to get a faster convergence speed. Based on above analysis, the optimal value of σ should minimize the maximum eigenvalue of $\mathbf{L}^*$.

Theorem 5.1. Assuming that all the eigenvalues of matrix $(1-\varepsilon)\mathbf{I}-\varepsilon\mathbf{L}$ are located inside the unit circle of the complex plane, and its maximum and minimum eigenvalues are $\lambda_{\max}$ and $\lambda_{\min}$, respectively, then the optimal value of the convergence acceleration term σ is $(\lambda_{\max}+\lambda_{\min})/2$.

Proof. When the eigenvalue distribution of $\mathbf{L}^*$ is adjusted by the σ, it is necessary to ensure that $\lambda_{\max}^*$ is still located inside the unit circle, thus the following two inequation must be satisfied simultaneously

$$\sigma-\lambda_{min}<1$$

$$\lambda_{max}-\sigma<1$$

Combining above two inequations, we have

$$\lambda_{max}+1<\sigma<1+\lambda_{min} \tag{33}$$

to ensure that $\lambda_{\max}^*<\lambda_{max}$, it is necessary to satisfy that

$$|\sigma-\lambda_{min}|<\lambda_{max} \tag{34}$$

Since $\lambda_{max}<1$, the upper limit of σ can be obtained by combining (33) and (34) and is given by

$$\sigma<\lambda_{max}+\lambda_{min}$$

In the geometric sense, $|\lambda_{\max}^*|$ is the Euclidean distance between σ and λ_{min} or $\lambda_{\max}$. Defining the operator $\max\{a,b\}$ to be the

larger of a and b, then $|\lambda^*_{\max}|=\max\{|\lambda_{min}-\sigma|,|\lambda_{max}-\sigma|\}$, Hence $|\lambda^*_{\max}|$ will reach the minimum value if and only if $\sigma=(\lambda_{\max}+\lambda_{\min})/2$.

5.4 Input disturbance response analysis

Parameter update process of logical clock may be disturbed by various noises (such as measurement noise and transmission noise), hence we give an analysis of disturbance response of parameter update process in CTCS. Update process of logical clock parameters may be disturbed by various noises (such as measurement noise, transmission noise, etc.), hence we analyze the disturbance response of CTCS as well. Without loss of generality, we choose the ith row (synchronization error of node i) in (19) for analysis, by adding the disturbance $d(k)$, we have

$$\begin{aligned} e_i(k+1) &= (1-\varepsilon a_{i(n+1)} - \sigma + \sum_{j=1}^{n} \varepsilon a_{ij}) e_i(k) \\ &\quad + \sum_{j=1}^{n} \varepsilon a_{ij} e_j(k) + d(k) \end{aligned} \tag{35}$$

Taking the z-transform, the transfer function from $d(k)$ to $e_i(k)$ is given by

$$\begin{aligned} zE_i(z) &= (1-\varepsilon a_{i(n+1)} - \sigma + \sum_{j=1}^{n} \varepsilon a_{ij}) E_i(z) \\ &\quad + \sum_{j=1}^{n} \varepsilon a_{ij} E_j(z) + D(z) \end{aligned} \tag{36}$$

Since (36) is linear, the influence of $D(z)$ on synchronization error could be considered independently, let $\sum_{j=1}^{n} \varepsilon a_{ij} E_j(z)=0$, the transfer function between synchronization error and disturbance is given by

$$H_i(z) = \frac{E_i(z)}{D(z)} = \frac{1}{z-(1-\varepsilon a_{i(n+1)} - \sigma + \sum_{j=1}^{n} \varepsilon a_{ij})} \tag{37}$$

Since any form of input disturbance can be expressed as the superposition of impulse functions, in order to investigate the impulse response of (35), the $d(k)$ is expressed as

$$d(k) = \begin{cases} D, & k = 0 \\ 0, & k \neq 0 \end{cases}$$

Taking the z-transform of $d(k)$, we get $D(z)=D$. According to final value theorem (Franklin 2015), we have

$$e_i[\infty] = \lim_{z \to 1} \frac{(z-1)D}{z-(1-\varepsilon a_{i(n+1)} - \sigma + \sum_{j=1}^{n} \varepsilon a_{ij})} = 0 \tag{38}$$

This limit operation in (38) indicates that the influence of input disturbance to the update process will gradually vanish after several iterations.

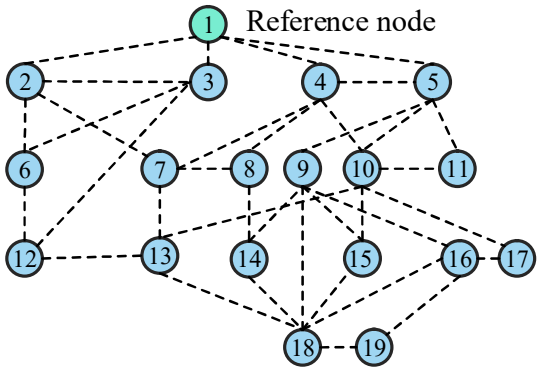

Fig. 1. Network topology composed of 20 nodes.

6 Simulation and experimental result

CTCS is designed based on the consensus tracking principle and analyzed in the state space framework. By referring to the simulation design of consensus tracking-based algorithms (Yin et al. 2013; Yang et al. 2013), we perform the simulation and analysis of CTCS and its comparative algorithm with system control toolbox in MATLAB (R2018b).

Clock synchronization algorithms are simulated in networks with different scale ranging from 20 nodes to 100 nodes, and simulations include: 1) the influence of convergence acceleration term on the convergence speed; 2) synchronization of newly joined node: 3) response of CTCS to the changes of reference drift rate. In practice, logical clock is constructed based on RTC clock (32.768kHz), hence the resolution of logical clock in the simulation is set to 1/32768 ≈ 30.5 μs =1 tick. In order to maintain good visibility of simulation results, in the synchronization of newly joined nodes, the execution period is set to 0.01 s, while in other simulations, it is set to 0.02 s.

In most distributed clock synchronization algorithms, clock drift rate and clock offset are synchronized independently with identical parameter update formula, thus the total clock synchronization time is generally twice as long as drift rate synchronization time. To reduce the simulation complexity, we analyze and compare the performance of different algorithms mainly based on logical drift rate synchronization. The comparison algorithms are MMTS, CCS, and CCTS.

6.1. Logical drift rate synchronization

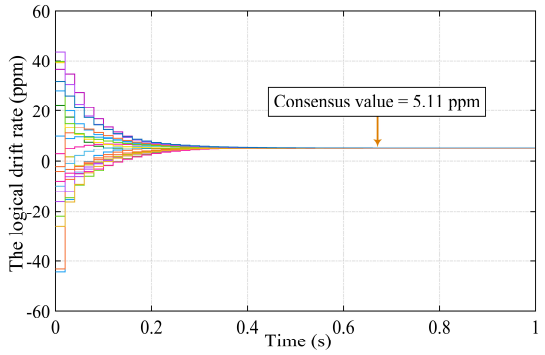

Fig. 2. Logical Drift Rate Synchronization of CCTS.

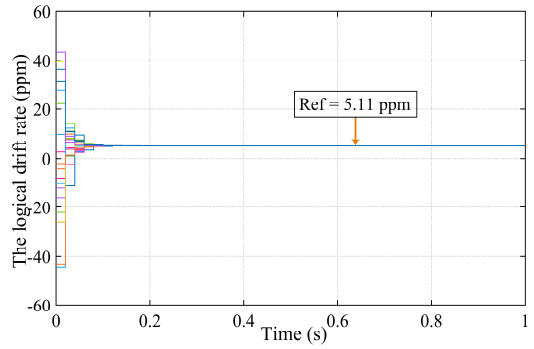

Fig. 3. Logical Drift Rate Synchronization of CTCS.

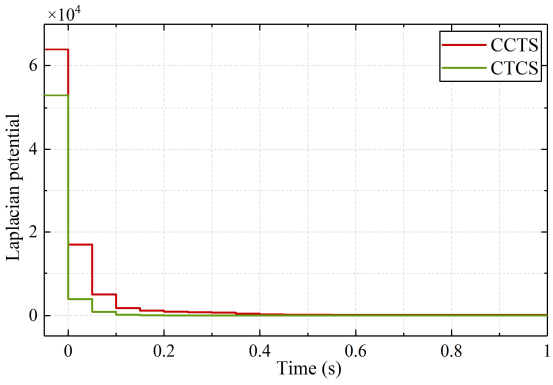

Fig. 4. Convergence of Laplacian Potential.

We first perform the simulation of logical drift rate synchronization in a network including 20 nodes. In Fig. 1, dotted lines represent the communication link, and node 1 is set as reference clock, node 2, 3, 4 and 5 are its neighbors. The average network connectivity $\bar{d}$ =3.5. In order to evaluate the consistency of logical drift rate during the synchronization process, we record the change of network Laplacian potential as well, and the definition of Laplacian

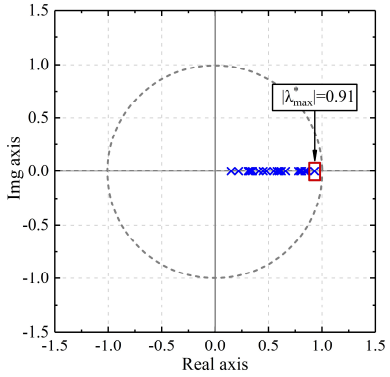

Fig. 5. Eigenvalue distribution of CCTS

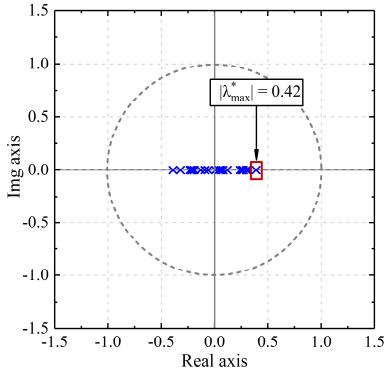

Fig. 6. eigenvalue distribution of CTCS

potential is given by (Olfati-Saber 2007)

$$\Phi_G(x) = x^T Lx = \frac{1}{2}\sum_{i,j} a_{ij}(x_j - x_i)^2$$

Fig. 2 and Fig. 3 show the logical drift rate synchronization of CCTS and CTCS, respectively. The initial logic drift rate $\omega_i^l(0)$ is set to obey uniform distribution U(-50,50), the unit is ppm. In CCTS, logical drift rate of network nodes converges to the average of initial logical drift rate, which equals to 5.11 ppm. To ensure the consistency of simulation condition, the reference logical drift rate in CTCS is set to 5.11 ppm as well. The iteration will terminate as the maximum logical drift rate difference between nodes is less than 1ppm. In Fig. 2, the number of iterations of CCTS is 37, which takes 0.74 s, and the total number of time messages transmitted is $20\times37\times\bar{d}$=2590. However, in Fig. 3, the CTCS only iterates for 8 times, which takes about 0.16 s. The total number of time messages transmitted is $20\times8\times\bar{d}$ =560, which is approximately 80% less than that of CCTS.

Fig. 4 shows the change of network Laplacian potential during the synchronization process. It can be seen that $\mathbf{\Phi}_G(x)$ gradually converges with 0 being the limit, and the

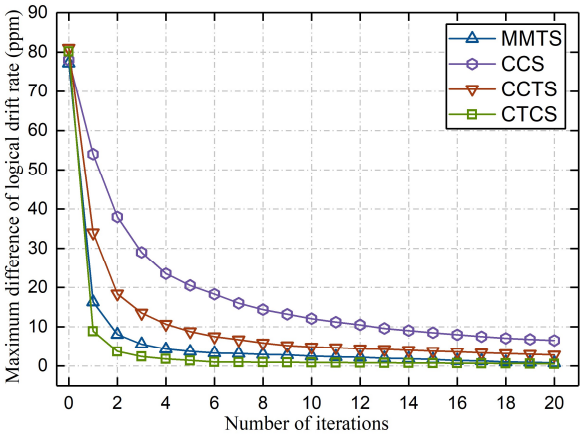

Fig. 7. Comparison of maximum difference of logical drift rate

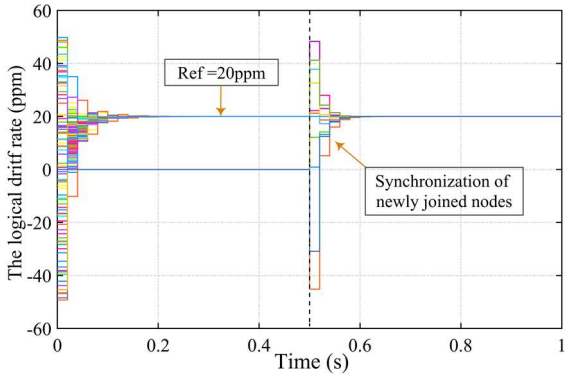

Fig. 8. Logical drift rate synchronization of newly joined nodes

convergence speed of CTCS is faster than that of CCTS. This is because the convergence speed of CCTS is only determined by network connectivity, while our algorithm could calculate the convergence acceleration term according to the network topology, thus a faster convergence speed is achieved by optimizing the eigenvalue distribution of synchronization error matrix.

Theorem 5.1 gives the calculation method of convergence acceleration term, in order to evaluate the effect of convergence acceleration intuitively, we calculate and show the eigenvalue distribution of synchronization error matrix in CCTS and CTCS, and results are respectively shown in Fig. 5 and Fig. 6. In the network with 20 nodes, since 20 eigenvalues are all located in the unit circle of the complex plane, both algorithms could converge. However, the maximum eigenvalue of CCTS is very close to the boundary of unit circle, indicating a critical stable state for CCTS, which is characterized by slow convergence speed. It can be seen from Fig. 6 that, by using convergence acceleration term, the eigenvalue distribution of CTCS could be adjusted much closer to the origin, and the modulus of $\lambda^*_{\max}$ decreases from 0.91 to 0.42. By (30), the convergence speed is mainly determined by $|\lambda^*_{\max}|$ and increases with the decrease of $|\lambda^*_{\max}|$, hence the synchronization speed is significantly improved.

With same iteration times, we compare the maximum difference of logical drift rate of different algorithms in Fig. 7. The horizontal axis is the number of iterations, and the vertical axis is the maximum difference of logical drift rate among network nodes. The number of iterations is set to 20. CCS is designed based on consensus agreement algorithm without specific optimization for convergence speed, hence its convergence speed is the slowest. CCTS is improved based on CCS, and cluster head is adopted to elevate the time message processing efficiency. After 20 iterations, the maximum difference of logical skew rate of CCTS is 2.8 ppm. In MMTS, it is necessary to execute both maximum and minimum consensus algorithms, which reduces its convergence speed, and the maximum difference of logical skew rate is 1.2 ppm. Optimized by convergence acceleration term, the maximum difference of logical skew rate of CTCS is only 0.7 ppm, which is lower than comparison algorithms after 20 iterations.

6.2. Synchronization of newly joined nodes

Fig. 8 shows the synchronization of newly joined nodes, in the synchronized network with 100 nodes. the execution period is set to 0.01 s to maintain a good visibility of simulation results. As can be seen form simulation result, logical skew rate of network nodes is synchronized to 20 ppm at 0.28 s. A total of 20 nodes join the existing network and trigger the execution of clock synchronization at 0.5s, since the number of newly joined nodes is much smaller than that of original network, it only takes 0.12 s to complete the synchronization. By comparing with the simulation results of average consensus-based clock algorithm (Wu 2015), it can be seen that the logical drift rate of synchronized clocks in the original network remains steady as newly joined nodes perform synchronization, thus CTCS shows a better adaptability to the change of network scale.

6.3 Dynamic response of reference input change

We test the dynamic response of CTCS by changing the reference logical drift rate. Firstly, the reference logic drift rate is set to 20 ppm, As can be seen from Fig. 9, the synchronization of logic drift rate takes 0.16 s and the number

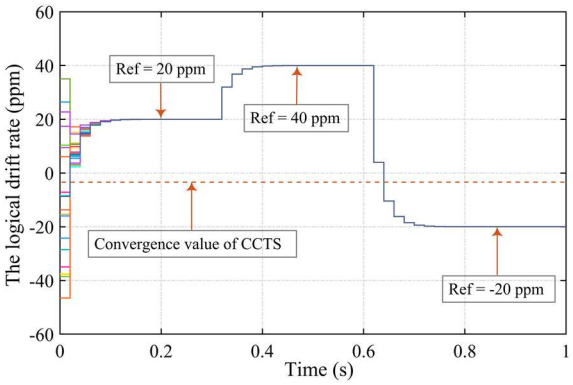

Fig. 9. Dynamic response of reference input change of CTCS

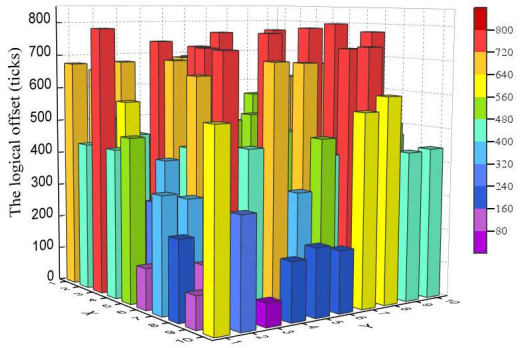

Fig. 10. Logical offset distribution before synchronization

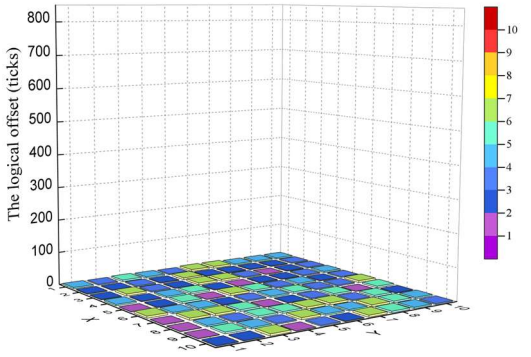

Fig. 11. Logical offset distribution after synchronization

of iterations is 8. Whereafter, the reference logical drift rate was changed to 40 ppm and -20 ppm at 0.32 s and 0.62 s, respectively, and in both settings, resynchronization could be achieved within 7 iterations, which reveals a good dynamic response. In Fig. 9, the orange dotted line is the synchronization result of CCTS and CCS, which equals to $\frac{1}{20}\sum_{i=1}^{20}\omega_i^l(0)$, Where $\omega_i^l(0)$ is the initial logical drift rate when the synchronization starts. As can be seen from simulation results, if there is any change in the reference clock

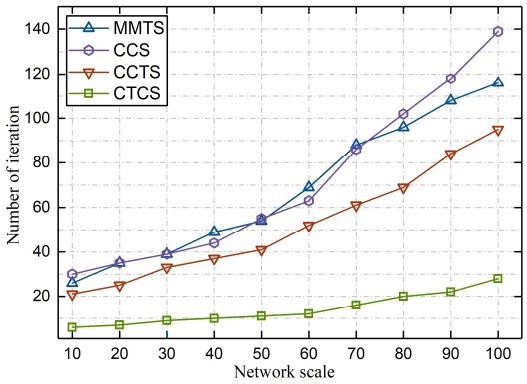

Fig. 12. Relationship between the number of iterations and network scale

parameters, the rest of network nodes would track the change to keep the long-term clock synchronization state in the network.

6.4. Logical offset synchronization

Fig. 10 and Fig. 11 give the geographic distribution of logical offset before and after synchronization by CTCS in a network with 100 nodes. According to (3), when the logical drift rate synchronization is completed, the clock synchronization can be achieved by completing the logical offset synchronization. In the simulation, node number is represented by coordinates of x-axis and y-axis, and logical offset is represented by z-axis with tick as the unit. Initial logical offset obeys uniform distribution U(0,800) and the maximum logical offset difference is 749 ticks.

In the simulation of CCTS (Wu 2014), the maximum offset difference is reduced to less than 30 ticks in a network with 25 nodes. However, the maximum offset difference in CTCS is reduced to less than 7 ticks after 20 iterations, showing a better performance.

6.5. Network scale and algorithm iteration times

The convergence speed of the clock synchronization algorithm is closely related to the network scale, hence we compare the convergence speed of different algorithms in different network scales. For the sake of generality, the average network connectivity is set randomly between 3.5 and 3.7. In each scale, only 20% of the nodes could communicate with the reference node, and 5 times of simulations are performed before the average was taken. The comparison algorithms are MMTS, CCS, and CCTS, and initial value of node clock obeys uniform distribution with the same

parameters. During the simulation, the iteration would stop if the maximum clock difference between nodes is less than 1 tick.

As can be seen from Fig. 12 that the number of iterations increases nonlinearly with the expansion of network scale. CCS is designed based on the consensus agreement algorithm which has not been optimized for large-scale networks, hence the number of iterations increases rapidly. After the execution of maximum consensus algorithm, MMTS also has to execute the minimum consensus algorithm to optimize clock speed, which increases the number of iterations. In CCTS, time message interaction is transferred from neighbor nodes to cluster members, which reduces the number of iterations. However, additional time message transmission is required between cluster heads, thus increase the number of iterations. By actively adjusting the eigenvalue distribution of synchronization error matrix, CTCS is able to improve the convergence speed without relying on the clustering strategy as well as increasing the computational complexity. The simulation result shows that CTCS requires fewer iterations in all the network scale. In the network with 100 nodes, the number of iterations of CCTS is 95 and that of CTCS is only 28.

7. Conclusion

In this paper, a novel distributed clock synchronization algorithm (CTCS) is proposed to achieve efficient clock synchronization in IoT. In order to overcome the shortcomings of current clock synchronization algorithms, CTCS performs joint synchronization of clock drift rate and clock offset based on the consensus tracking principle. Considering that the convergence speed of synchronization algorithm is slow in a weakly connected network, we design the convergence acceleration term in the update equation of logical clock parameters, by actively adjusting the eigenvalue distribution of synchronization error matrix, the synchronization convergence speed is accelerated. Meanwhile, the reference input term is designed as well to eliminate the oscillation of synchronized clocks caused by newly joined nodes. In the simulation part, we perform extensive simulations and compare the performance of CTCS with other popular distributed synchronization algorithms. The results show that CTCS is superior to others in terms of synchronization speed and adaptability to network scale changes.

Ethical approval：This article does not contain any studies with human participants or animals performed by any of the authors.

Funding: This work was supported in part by National Key Research and Development Program of China under Grant 2017YFE0132100, and in part by National Natural Science Foundation of China under Grant 61971305.

Conflicts of Interest: The authors declare no conflicts of interest.

Informed Consent：Informed consent was obtained from all individual participants included in the study.

Author Contributions: Conceptualization, Y.N. and T.Y.; Methodology, Y.N.; Software, Y.N.; Validation, T.Y. and Y.N.; Formal Analysis, Y.N., T.Y. and Z.H.; Investigation, Y.N.; Resources, Y.N. and S.C.; Data Curation, P.Y. and Y.N.; Writing-Original Draft Preparation, Y.N.; Writing-Review & Editing, Y.N. and T.Y.; Visualization, Y.N.; Supervision, T.Y.; Project Administration, T.Y.

References

Manavalan E, Jayakrishna K (2019) A review of Internet of Things (IoT) embedded sustainable supply chain for industry 4.0 requirements. Comput Ind Eng 127: 925-953

Akhlaq M, Sheltami T R (2013) RTSP: An accurate and energy-efficient protocol for clock synchronization in WSNs. IEEE T Instrum Meas 62(3): 578-589

Yildirim K S, Kantarci A (2013) Time synchronization based on slow-flooding in wireless sensor networks. IEEE T Parall Distr 25(1): 244-253

Lamonaca F, Gasparri A, Garone E, et al (2014). Clock synchronization in wireless sensor network with selective convergence rate for event driven measurement applications. IEEE T Instrum Meas 63(9): 2279-2287

Kadowaki Y, Ishii H (2014) Event-based distributed clock synchronization for wireless sensor networks. IEEE T Automat Contr 60(8): 2266-2271

Tahaei H, Afifi F, Asemi A et al (2020) The rise of traffic classification in IoT networks: A survey. J Netw Comput Appl 154: 102538

Xiong Y, Wu N, Shen Y, et al (2017). Cooperative network synchronization: Asymptotic analysis. IEEE T Signal Proces 66(3): 757-772

Ganeriwal S, Kumar R, Srivastava M B (2003) Timing-sync protocol for sensor networks. In: Proceedings of the 1st international conference on Embedded networked sensor systems, pp 138-149

Elson J, Girod L, Estrin D (2002) Fine-grained network time synchronization using reference broadcasts ACM SIGOPS Oper Syst Rev, 36(SI):147-163

Noh K L, Chaudhari Q M, Serpedin E, et al (2007) Novel clock phase offset and skew estimation using two-way timing message exchanges for wireless sensor networks. IEEE T Commun 55(4): 766-777

Su W, Akyildiz I F (2005) Time-diffusion synchronization protocol for wireless sensor networks. IEEE/ACM Trans. Netw 13(2): 384-397

Schenato L, Gamba G (2007) A distributed consensus protocol for clock synchronization in wireless sensor network. In: IEEE Conference on Decision & Control, IEEE, pp 2289-2294

Maggs M K, O'keefe S G, Thiel D V (2012) Consensus clock synchronization for wireless sensor networks. IEEE Sens J 12(6): 2269-2277

Luo B, Wu YC (2013) Distributed clock parameters tracking in wireless sensor network. IEEE Trans Wirel Commun 12(12):6464-75

Du J, Wu YC (2013) Distributed clock skew and offset estimation in wireless sensor networks: asynchronous algorithm and convergence analysis. IEEE Trans Wirel Commun 12(11):5908–17

Wang Q, Sun C, Xin X (2017) Robust consensus tracking of linear multiagent systems with input saturation and input-additive uncertainties. Int J Robust Nonlin 27(14): 2393-2409

Lévesque M, Tipper D (2016). A survey of clock synchronization over packet-switched networks. IEEE Commun Surv Tut 18(4): 2926-2947

Djenouri D, Bagaa M (2014). Synchronization protocols and implementation issues in wireless sensor networks: A review. IEEE Syst J 10(2): 617-627

Swain A R, Hansdah R C (2015). A model for the classification and survey of clock synchronization protocols in WSNs. Ad Hoc Netw 27: 219-241

Carli R, Zampieri S (2013). Network clock synchronization based on the second-order linear consensus algorithm. IEEE T Automat Contr 59(2): 409-422

Ping S (2003) Delay measurement time synchronization for wireless sensor networks. Intel Research Berkeley Lab 6: 1-10

Maróti M, Kusy B, Simon G, et al (2004) The flooding time synchronization protocol. In: Proceedings of the 2nd international conference on Embedded networked sensor systems, pp: 39-49

Wang H , Shao L , Li M , et al (2017) Estimation of Frequency Offset for Time Synchronization With Immediate Clock Adjustment in Multihop Wireless Sensor Networks. IEEE Internet Things 4(6): 2239-2246

Leng M, Wu Y-C (2011) Low-Complexity Maximum-Likelihood Estimator for Clock Synchronization of Wireless Sensor Nodes Under Exponential Delays. IEEE T Signal Proces 59(10):4860-4870

Chaudhari QM, Serpedin E, Qaraqe K (2010) On minimum variance unbiased estimation of clock offset in a two-way message exchange mechanism. IEEE T inform Theory 56(6):2893-2904

Schenato L, Fiorentin F (2011) Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks. Automatica 47(9):1878-1886

He J, Cheng P, Shi L (2014a) Time Synchronization in WSNs: A Maximum-Value-Based Consensus Approach. IEEE T Automat Contr 59(3):7882-7887

He J, Li H, Chen J (2014b) Study of consensus-based time synchronization in wireless sensor networks. ISA T 53(2):347-357

Wu J, Zhang L, Bai Y (2014) Cluster-based consensus time synchronization for wireless sensor networks. IEEE Sens J 15(3):1404-1413

Wang Z, Zeng P, Zhou M (2017) Cluster-based maximum consensus time synchronization for industrial wireless sensor networks. Sensors-Basel 17(1):141

Shi F, Tuo X, Ran L (2019) Fast Convergence Time Synchronization in Wireless Sensor Networks Based on Average Consensus. IEEE T Ind Inform 16(2):1120-1129

Franklin GF, Powell JD, Emami-Naeini A (2015) Feedback control of dynamic systems. Pearson, London, pp 123-126

Choi BJ, Liang H, Shen X (2011) DCS: Distributed asynchronous clock synchronization in delay tolerant networks. IEEE T Parall Distr 23(3):491-504

Zhou F, Wang Q, Han G (2019) APE-sync: An adaptive power efficient time synchronization for mobile underwater sensor networks. IEEE Access 7: 52379-52389

Hamilton BR, Ma X, Zhao Q (2008) ACES: adaptive clock estimation and synchronization using Kalman filtering. Proceedings of the 14th ACM international conference on Mobile computing and networking 152-162

Wu J, Bai Y, Zhang L (2015) Distributed time synchronization in wireless sensor networks via second-order consensus algorithms. J T University 21(2):113-121

Zhang X-D (2017) Matrix analysis and applications. Cambridge University Press, pp 48-49

Horn RA, Johnson CR (2012) Matrix analysis. Cambridge university press, pp 135-136

Brogan WL (1991) Modern control theory. Pearson education, pp 323-325

Bu X, Hou Z, Zhang H (2017) Data-driven multiagent systems consensus tracking using model free adaptive control. IEEE T Neur Net Lear 29(5):1514-1524

Yin X, Yue D, Hu S (2013) Consensus of fractional-order heterogeneous multi-agent systems. IET Control Theory A 7(2):314-322

Yang S, Tan S, Xu J-X (2013) Consensus based approach for economic dispatch problem in a smart grid. IEEE T Power Syst 28(4):4416-4426

Olfati-Saber R, Fax JA, Murray RM (2007) Consensus and cooperation in networked multi-agent systems. In: Proceedings of the IEEE, vol 95, pp 215-233

Figures

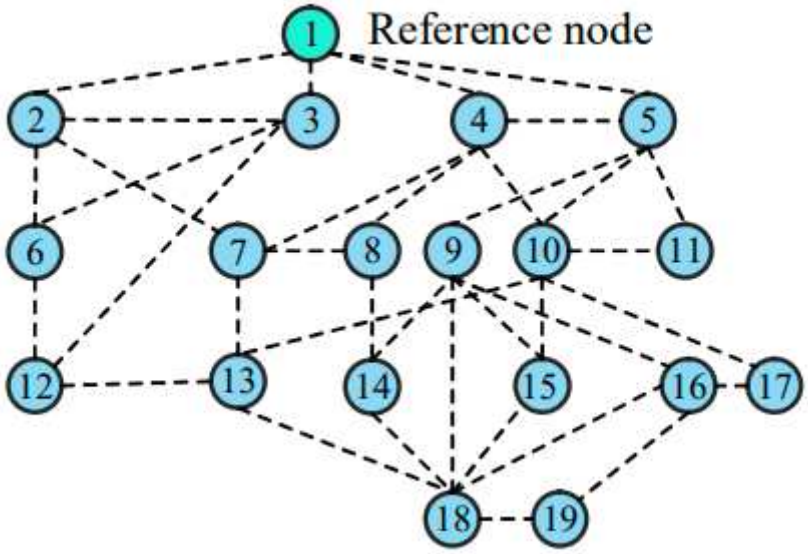

Figure 1

Network topology composed of 20 nodes.

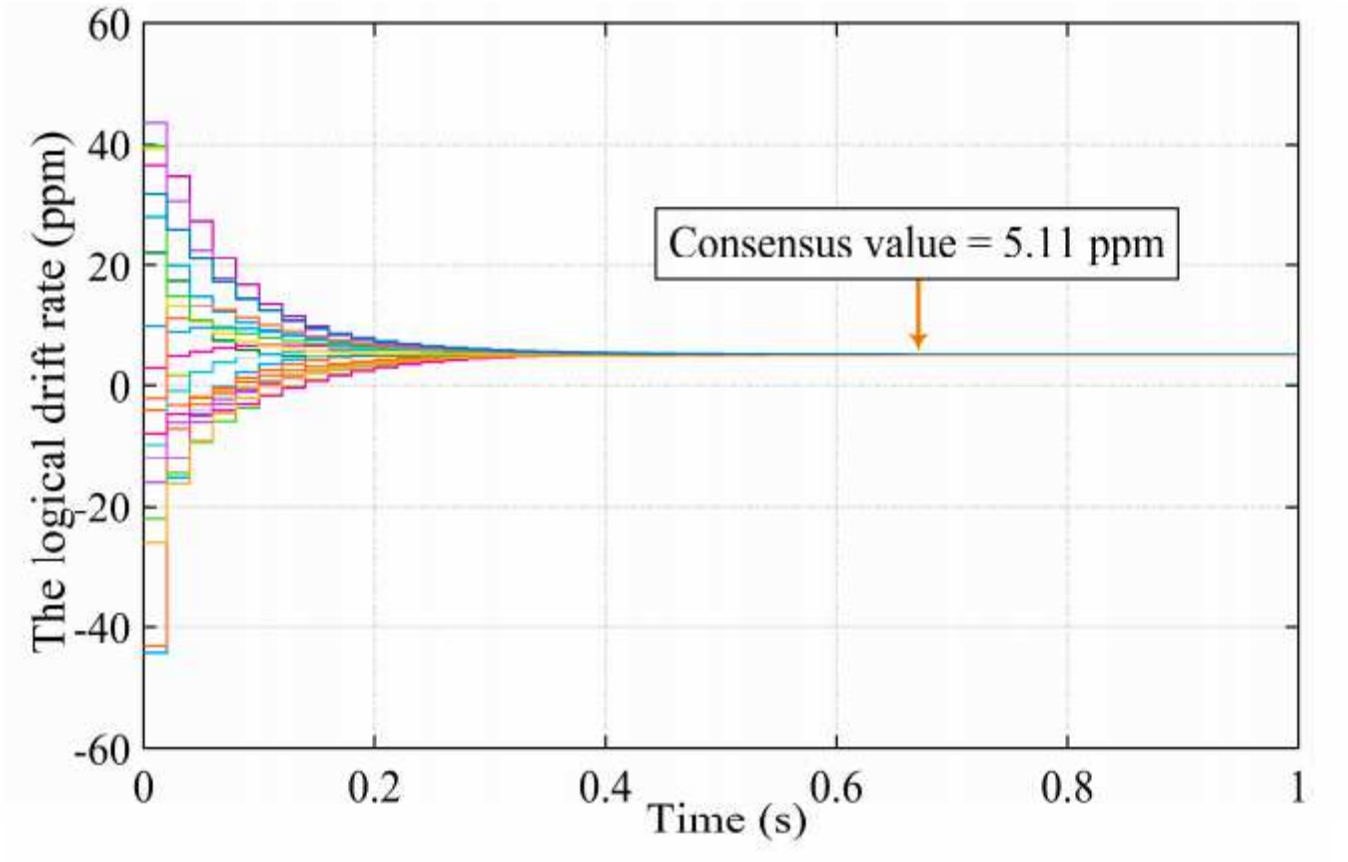

Figure 2

Logical Drift Rate Synchronization of CCTS.

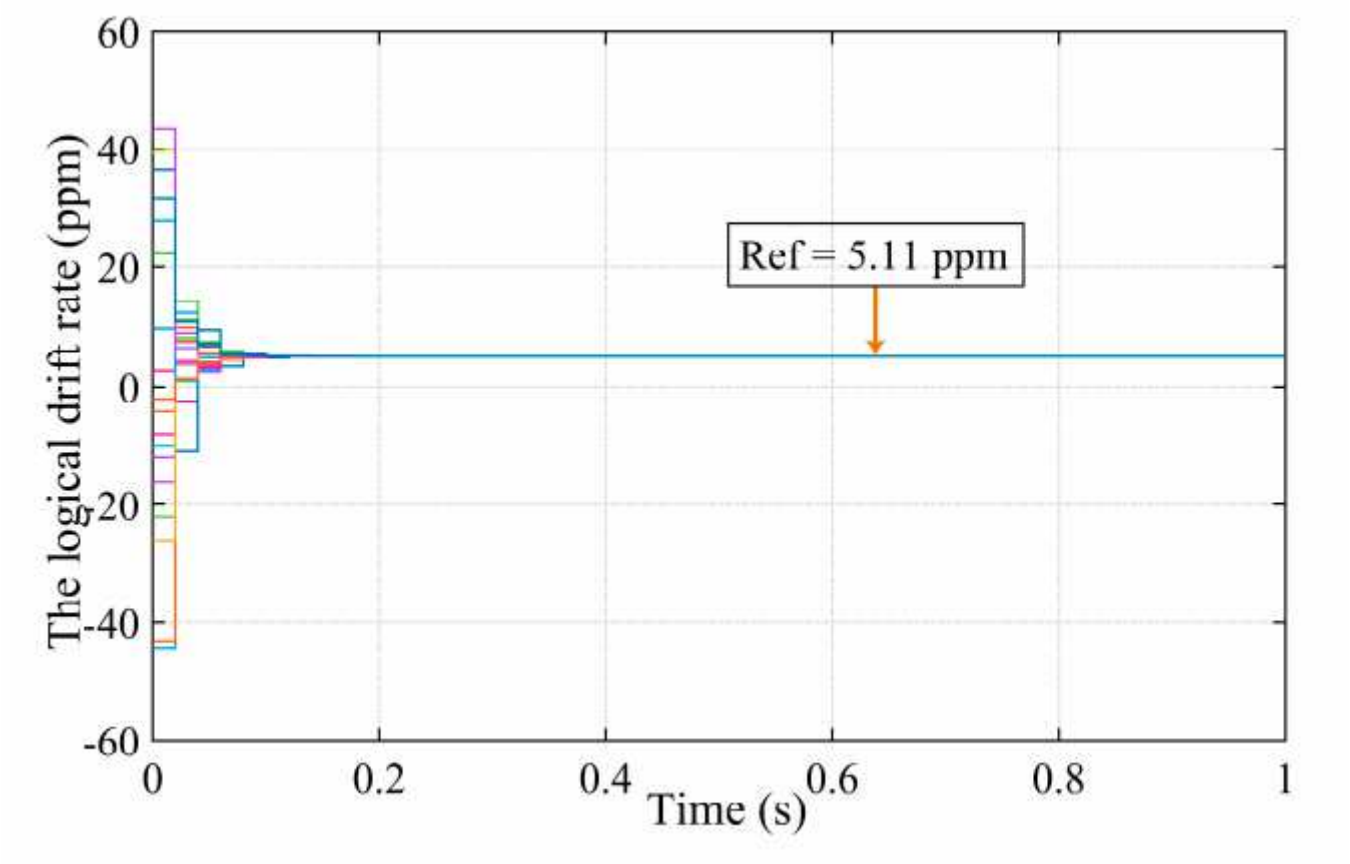

Figure 3

Logical Drift Rate Synchronization of CTCS.

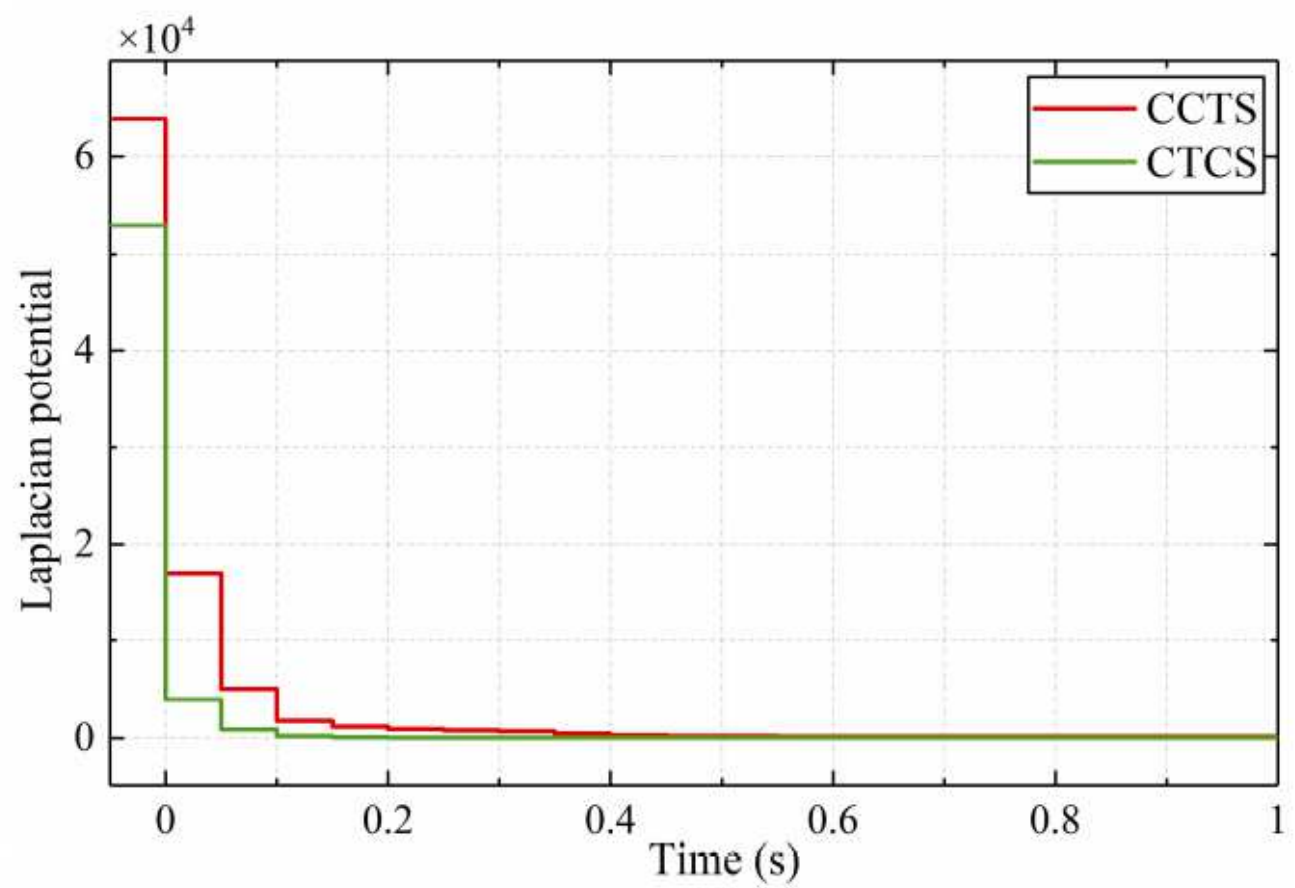

Figure 4

Convergence of Laplacian Potential.

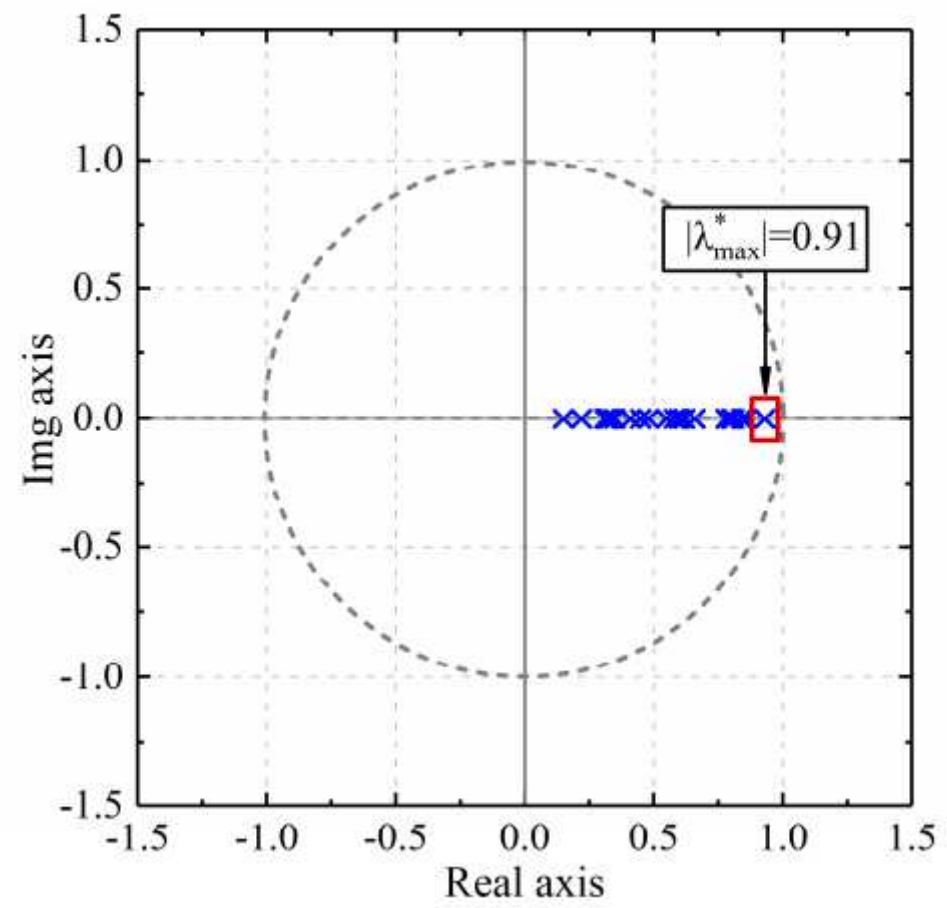

Figure 5

Eigenvalue distribution of CCTS

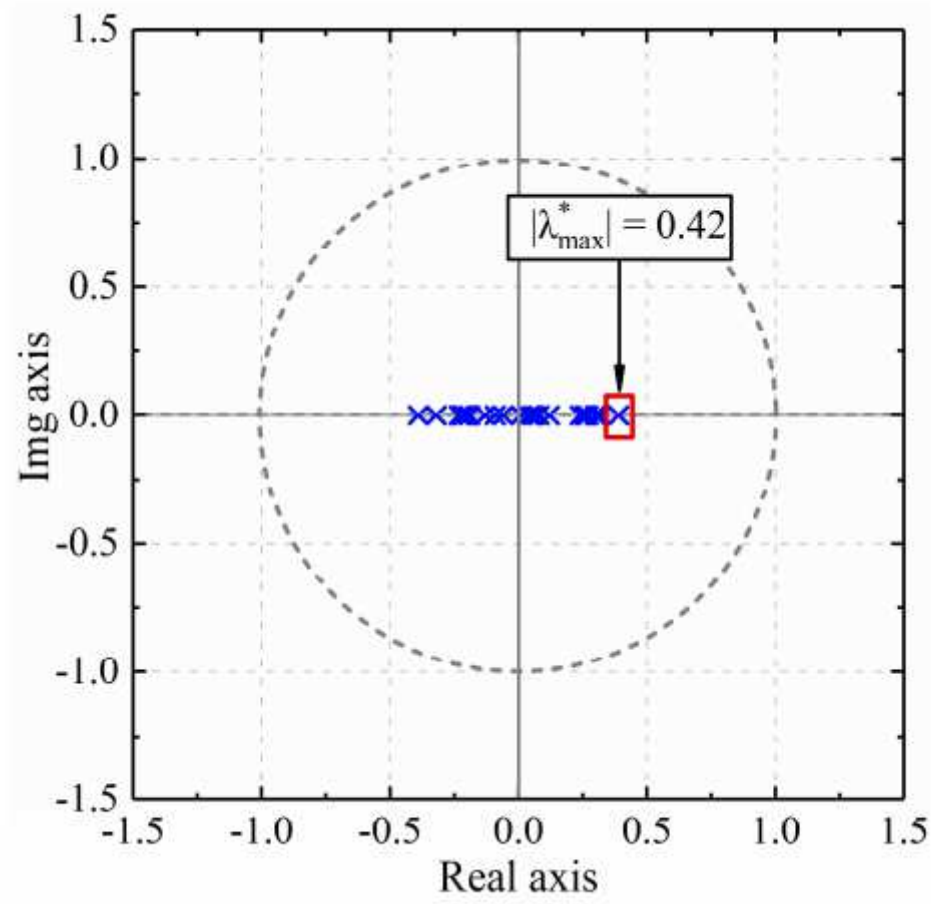

Figure 6

eigenvalue distribution of CTCS

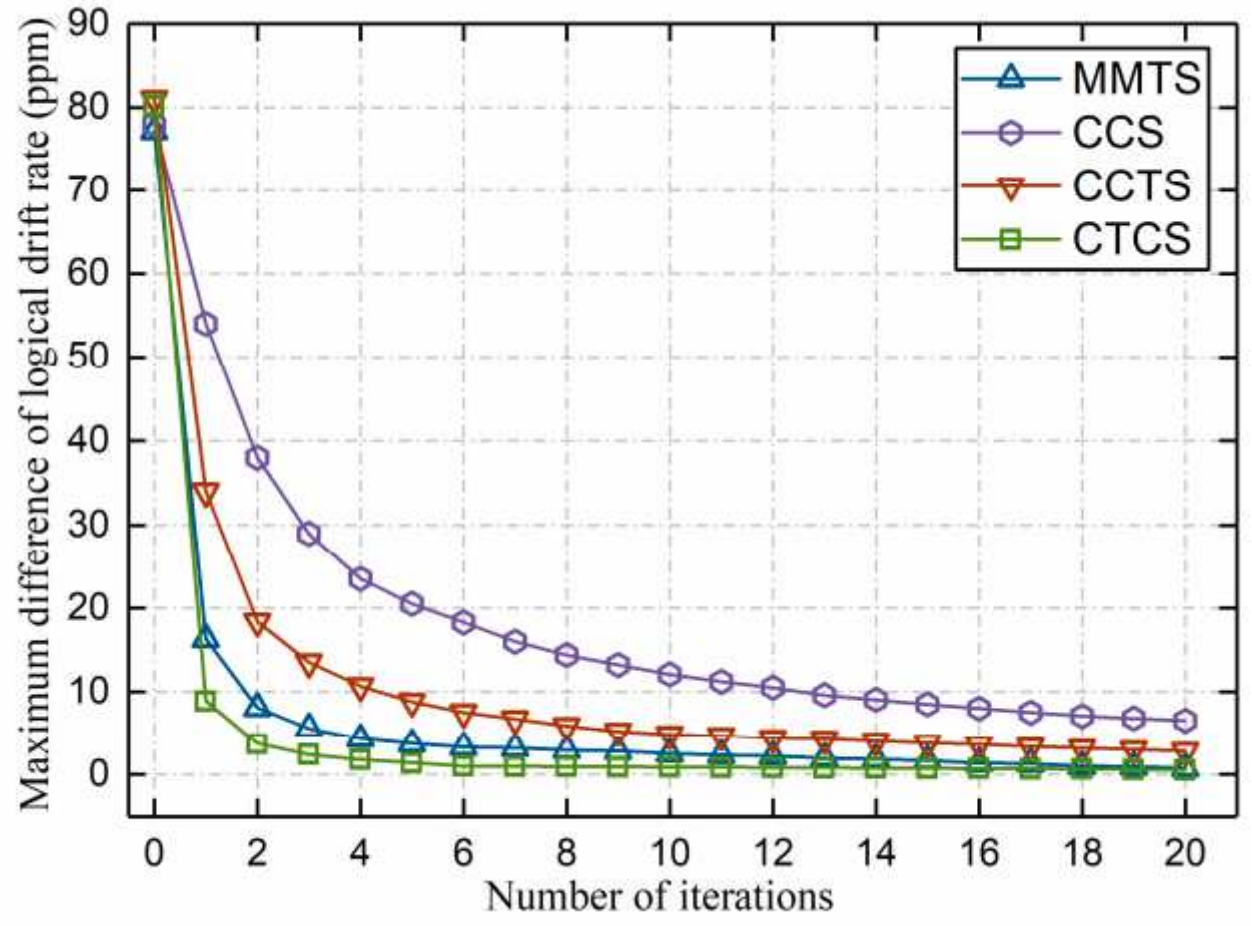

Figure 7

Comparison of maximum difference of logical drift rate

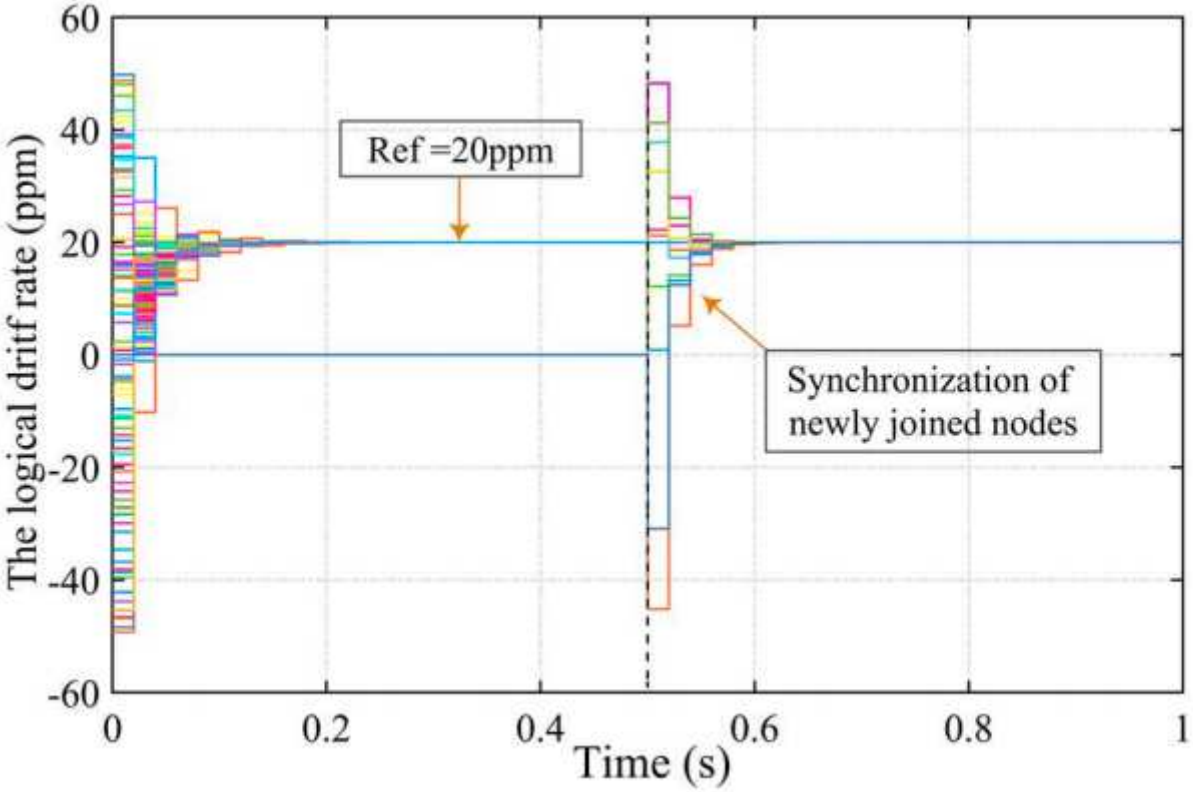

Figure 8

Logical drift rate synchronization of newly joined nodes

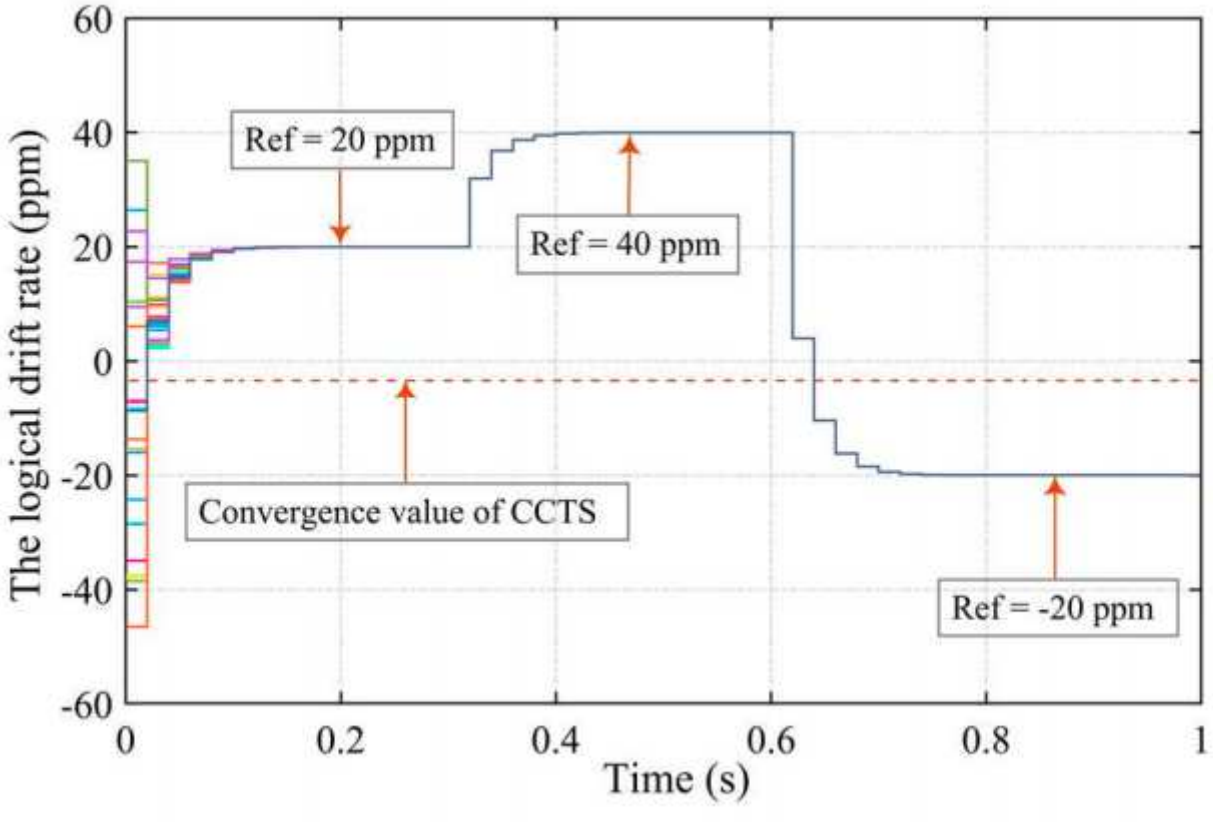

Figure 9

Dynamic response of reference input change of CTCS

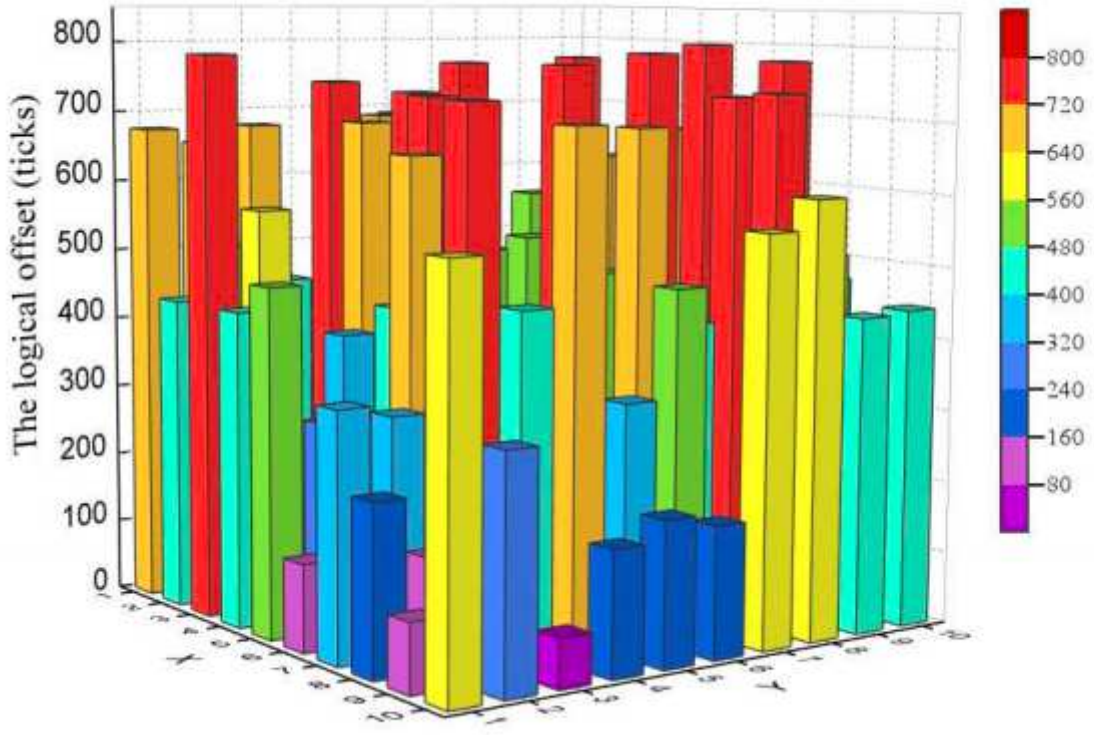

Figure 10

Logical offset distribution before synchronization

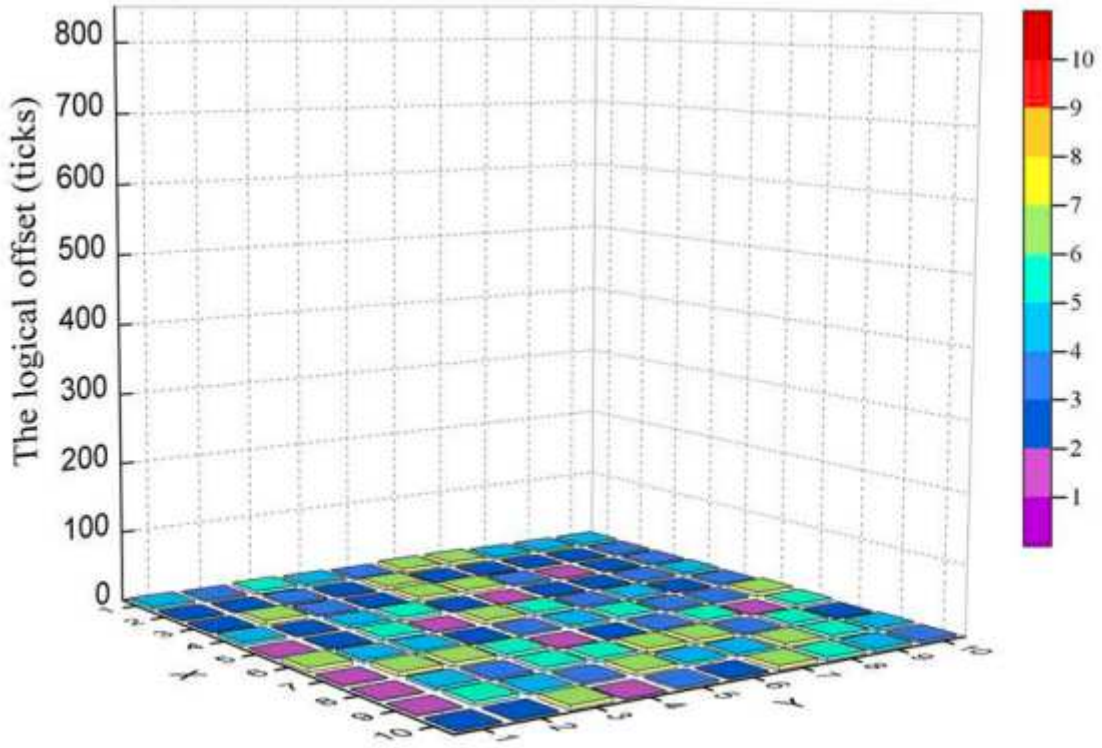

Figure 11

Logical offset distribution after synchronization

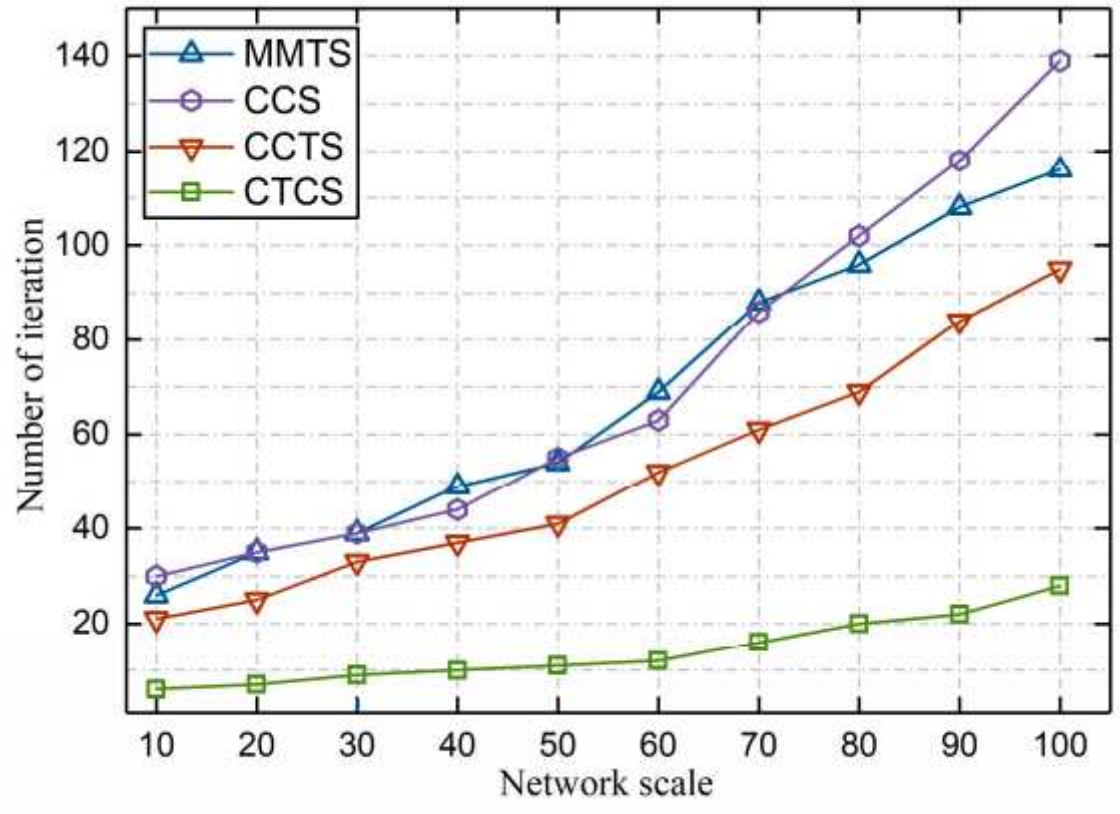

Figure 12

Relationship between the number of iterations and network scale