sensors

MDPI

Article

Fast High-Order Consensus Time Synchronization Protocol in Industrial Wireless Sensor Networks

Xiang Yu, Zhaowei Wang * and Zhongxin Zhang

School of Electrical and Information Engineering, Jiangsu University, Zhenjiang 212013, China; 2222307046@stmail.ujs.edu.cn (X.Y.); 2212407064@stmail.ujs.edu.cn (Z.Z.)
* Correspondence: wangzhaowei@ujs.edu.cn

Abstract

Slow convergence remains a critical limitation hindering the practical deployment of consensus-based time synchronization protocols (CTSPs). Increasing algebraic connectivity is a key mechanism for improving the convergence speed of distributed algorithms. However, existing strategies inevitably introduce redundant packet communication, while forwarding stale timing information may degrade synchronization accuracy. To address this challenge, this paper proposes a high-order consensus time synchronization protocol (HCTSP). Unlike traditional CTSPs, HCTSP incorporates previous clock states from two-hop neighboring nodes to establish a virtual topology and further employs this information to enhance the estimation of logical clock parameters, thereby achieving fast estimation of clock parameters while effectively suppressing fluctuations in parameter estimation caused by a large initial synchronization error. Although the proposed method utilizes single-hop communication to relay information from non-adjacent nodes—an indirect transmission mechanism that inherently introduces additional communication overhead—we further develop an accumulator-based redundant information optimization scheme. Furthermore, the consensus algorithm integrates both an accumulator-based update mechanism and multi-hop historical memory, partially alleviating the impact of Gaussian communication delay jitter on clock correction. Theoretical proofs have verified the fast convergence of the proposed protocol. Extensive simulation experiments also demonstrate the superior efficiency of HCTSP in terms of convergence speed, communication overhead, and synchronization accuracy. Specifically, in random networks with 25 nodes, there is an approximately 50% reduction in single-round synchronization message length and a 66.67% decrease in total packet exchange volume compared to the virtual topology-based time synchronization protocol (VTSP). In the typical ring topology, where the convergence speed of the consensus algorithm is slow, HCTSP has a 59.33% increase in convergence speed compared to VTSP and a 75.29% increase compared to the gradient time synchronization protocol (GTSP).

Keywords: time synchronization; industrial wireless sensor networks (IWSNs); fast convergence; average consensus; virtual link

Check for updates

Academic Editor: Keshav Dahal

Received: 12 May 2026
Revised: 3 June 2026
Accepted: 10 June 2026
Published: 14 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Time synchronization serves as a critical foundation for cooperative operations in distributed systems, particularly in industrial wireless sensor networks (IWSNs). The absence of precise time sequential support can significantly diminish the value of sensor-collected data and sensed information, thereby compromising the decision-making capabilities of upper-layer applications. For instance, in event-triggered control systems [1], consistent

timestamps are essential to ensure accurate calculation of trigger thresholds. Similarly, in smart grid applications [2], phasor measurement units rely on high-precision time synchronization to enable synchronized sampling across the entire power distribution network.

Distributed consensus-based time synchronization protocols (CTSPs) have emerged as a prominent direction in time synchronization research due to their robust nature and inherent adaptability to the dynamic topologies characteristic of wireless sensor networks [3–5]. Extensive research has yielded a diverse array of CTSPs to address various performance objectives in complex network environments with disturbances, including communication delays [6,7], node mobility [8,9], security threats [10], et al. However, most CTSPs suffer from slow convergence due to their reliance on high-frequency iterative communication between adjacent nodes, especially the average-consensus-based time synchronization algorithms. Despite significant advancements in accelerating convergence speed, the multi-hop communication strategy often incurs substantial communication overhead and computational complexity [11]. This issue is particularly pronounced in large-scale, dense, and resource-constrained IWSNs, where communication delays and path-dependent accumulation of estimation errors are inevitable under multi-hop communication protocols [12]. Hence, this paper focuses on optimizing time synchronization convergence speed while balancing critical performance factors, including communication overhead and delay tolerance.

1.1. Related Work and Motivation

Over the past decades, research on time synchronization protocols has made substantial progress, with architectures broadly categorized into centralized and distributed paradigms. Early centralized architectures predominantly rely on reference nodes and specific topology management schemes, which result in poor adaptability to dynamic network topology. Hence, recent advancements have investigated distributed average consensus protocols, such as the gradient time synchronization protocol (GTSP) [4] and the average timesync (ATS) [5]. While average consensus protocols entirely relying on neighbor information iteration demonstrate strong robustness and scalability, they often fail to satisfy real-time convergence demands. To realize finite-time convergence, He et al. proposed maximum-value time synchronization (MTS) [3], which eliminates redundant iterations by anchoring convergence to the network's fastest clock. Unfortunately, this protocol relies on comparing the relative clock parameters between two nodes for local clock adjustments, making it highly sensitive to delay jitter.

Currently, some studies have increasingly focused on optimizing the convergence speed of CTSPs, which we systematically classify into three distinct categories.

1.1.1. Cluster-Based Optimization Scheme

Recent research in cluster-based consensus time synchronization has demonstrated significant improvements in convergence efficiency. Wu et al. proposed a hierarchical network architecture to shorten the communication hops and enable parallel distributed processing [13]. In addition, Wang et al. proposed a three-step optimization scheme [14] to further reduce the synchronization overhead, which introduces a cluster-head logical skew threshold to control the cluster member synchronization and adopts a one-way communication method to reduce communication at the algorithm level. Liu et al. constructed a nearest K-neighboring (NKN) topology via K-means and proposed the TS-NKN algorithm [15] to balance convergence speed, accuracy, and energy consumption. While these cluster approaches enhance synchronization convergence efficiency and reduce communication consumption through structured consensus design, they face inherent limitations in dynamic network environments. As sensor nodes continuously consume energy, the com-

munication radius of cluster head nodes gradually shrinks, causing a reduction in the number of overlapping clusters. This degradation may lead to slower convergence speed and potential block-level independent synchronization, ultimately compromising global synchronization stability.

1.1.2. Hybrid Scheme

Some studies have also attempted to integrate the advantages of different algorithms to accelerate the convergence of consensus protocols. Zhao et al. proposed a max–average consensus protocol [16], which employs maximum consensus for logical skew compensation to eliminate redundant calculations and average consensus for logical offset compensation to effectively suppress synchronization error fluctuations. The hybrid time synchronization protocol (HTSP) [17] integrates reference-based and consensus-based models through dynamic mode switching. Pei et al. proposed a communication-aware distributed training (CADT) framework [18], which utilizes real-time CSI for dynamic gradient aggregation to enhance convergence speed and accuracy in wireless environments. However, these hybrid approaches inevitably compromise the robustness and scalability inherent in fully distributed protocols. This performance trade-off is particularly evident in large-scale dynamic networks, especially in terms of adaptability to topological changes (e.g., node addition and node mobility), where performance degradation may limit the practical applicability of hybrid-architecture-based protocols.

1.1.3. Multi-Hop Virtual Communication

Reducing network diameter and enhancing algebraic connectivity [19] constitute fundamental prerequisites for accelerating the convergence of consensus algorithms. Based on this theoretical premise, Panigrahi et al. proposed the selective average time synchronization (SATS) protocol [20], which employs a dynamic programming mechanism to identify the optimal multi-hop path with minimal synchronization error under constrained conditions. To increase algebraic connectivity, Shi et al. developed the multi-hop average consensus time synchronization (MACTS) protocol [21] based on the multi-hop relay concept, where nodes continuously forward multi-hop information during non-broadcast cycles to establish virtual communication links. While forwarding-based multi-hop virtual communication techniques effectively improve synchronization convergence speed, they inevitably incur a significant increase in communication overhead. More critically, in large-scale practical IWSNs, high-frequency updates may induce node contention, and the forwarding process also introduces uncertain transmission delays, directly resulting in synchronization errors between multi-hop nodes and local nodes. Furthermore, it is challenging to predefine an appropriate error threshold in real-world deployments. For distributed algorithms that rely solely on neighbor information to maintain system states, the lack of global topology knowledge makes it difficult to determine a proper threshold. In addition, a single fixed threshold cannot effectively adapt to dynamic network environments, such as random node joining and leaving.

To address the limitations of [3,21], Wang et al. proposed the virtual maximum-value time synchronization (VMTS) protocol [22], which is based on multi-hop links. In this protocol, an estimator based on the moving-average principle is designed to minimize the impact of normally distributed random delays on the maximum consensus algorithm, thereby improving synchronization accuracy. Shi et al. subsequently introduced a parameter-sharing strategy [23], which uses single-hop communication instead of forwarding every received message for multi-hop communication. By sharing relative clock offset estimations with neighboring nodes, virtual links are established with multi-hop nodes, thereby simplifying algorithm complexity and reducing information exchange. Duan et al. [24] dynamically ad-

justed the communication hop count in real time according to synchronization error states and network topology variations through online estimation of the convergence probability and its variation rate of synchronization errors, enabling a more practical and accurate design of multi-hop consensus algorithms. Unlike ATS-based asynchronous virtual communication [21,23], Phan et al.'s synchronous virtual topology-based time synchronization protocol (VTSP) [25] updates clock parameters in a manner similar to GTSP. However, VTSP only incorporates neighboring information when broadcasting local timing information during each synchronization period, thereby reducing the communication cost associated with the multi-hop forwarding strategy. To further optimize communication efficiency, Wang et al. enhanced the virtual multi-hop synchronization framework by incorporating an event-triggered mechanism [26]. This method significantly reduces redundant broadcasts compared with periodic synchronization approaches [21,23,25].

The aforementioned multi-hop virtual communication synchronization algorithms primarily accelerate the convergence of consensus algorithms by utilizing current states or forwarded neighboring information to improve the algebraic connectivity of the network. However, these methods have not yet fully exploited the potential of historical memory information for accelerating consensus convergence [27,28]. The work in [29] has shown that, in most cases, multi-step consensus algorithms incorporating local historical memory can achieve superior convergence performance compared with conventional gradient descent methods, and their effectiveness has been validated in distributed average consensus engineering applications. Furthermore, [30] analyzed the theoretical bounds of the fastest achievable convergence rate under different tap conditions (i.e., memory orders), as well as the corresponding optimal parameter design problem.

Although expanding the communication range and increasing node degrees are fundamental methods for enhancing a graph's algebraic connectivity, they inevitably lead to increased power consumption. This trade-off makes such approaches unsuitable for low-power IWSNs. The aforementioned multi-hop virtual communication methods have demonstrated notable improvements in convergence speed and communication efficiency. However, compared with conventional single-hop consensus synchronization protocols [3–5], these methods may potentially compromise synchronization accuracy and convergence performance due to the introduction of uncertain relay delays [21] and non-real-time multi-hop neighboring timing information [23,25,26]. Moreover, in dense network deployments, the single-hop communication mechanism carrying multi-hop timing information still incurs substantial communication overhead [25]. Consequently, this motivates us to seek a time synchronization protocol for IWSNs that achieves fast convergence, low communication overhead, and delay tolerance.

1.2. Contribution

Existing studies have demonstrated that incorporating historical state information can enhance the convergence speed of distributed systems [27–31]. Motivated by this idea, the historical information of two-hop neighboring nodes is incorporated into the virtual-link construction mechanism to alleviate the update-direction bias induced by uncertain two-hop terms. Based on this core principle, we propose a novel high-order consensus time synchronization protocol (HCTSP) to overcome the challenge of slow convergence in current CTSPs while maintaining low communication overhead and delay tolerance. The main contributions are as follows.

1. The proposed joint skew-offset estimation framework utilizes both the current clock states of one-hop neighboring nodes and the historical clock states of two-hop neighboring nodes to estimate relative clock parameters. Rigorous theoretical proofs of the

rapid convergence of the proposed method are provided through comparison with traditional multi-hop synchronization protocols.

2. The design of HCTSP is based on a simple single-hop communication model, which greatly reduces communication volume compared with multi-hop relay communication. Moreover, to reduce the additional packet overhead associated with multi-hop neighboring information, an accumulator-based redundant information optimization scheme is proposed.
3. By integrating historical information from multi-hop neighboring nodes, the proposed method achieves superior link-state observability, leading to improved synchronization accuracy in virtual multi-hop networks. Extensive simulation results demonstrate that HCTSP can accelerate convergence with low overhead under various network topologies while effectively improving synchronization accuracy. Notably, the method maintains compatibility with both GTSP and ATS protocols.

The remainder of this article is organized as follows. Section 2 introduces the system model and analyzes average consensus problem. Section 3 proposes the fast high-order consensus protocol. Section 4 presents the implementation of HCTSP. Section 5 validates and discusses the simulation results of the proposed HCTSP. Section 7 concludes the main work.

2. System Model and Problem Formulation

An IWSN can be modeled as an undirected graph $G = (V, E)$, where $V = 1, 2, 3, \cdots, n$ denotes the set of nodes and E represents the set of available communication links between nodes, also known as the edge set. $(i, j) \in E$ indicates that nodes i and j can maintain bidirectional information exchange. The set $N_i = j \mid (i, j) \in E, i \neq j, \forall i \in V$, corresponds to the neighbors of node i. Obviously, $|N_i|$ denotes the number of neighbors and is also defined as the degree of node i in graph theory. Consequently, the degree matrix D of graph G is a diagonal matrix with $D_{ii} = \deg(i)$. Let A denote the adjacency matrix. If $(i, j) \in E$, the element a_{ij} in A is 1; otherwise, it is 0.

The Laplacian matrix L is calculated as $L = D - A$. By definition, L is positive semidefinite, and its eigenvalues can be ordered as $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Here, the second-smallest eigenvalue λ_2, also known as the algebraic connectivity, is a necessary and sufficient condition for the connectedness of an undirected graph G.

2.1. Clock Model

Usually, each node in IWSN maintains a local hardware clock $H_i(t)$, which follows an affine model [3–5],

$$H_i(t) = \alpha_i t + \beta_i \tag{1}$$

where i denotes the node ID and t represents the absolute time. α_i is defined as the clock skew, which describes the ratio of the actual frequency to the nominal frequency. The clock offset β_i represents the initial phase difference at startup. Consistent with prior work [4], it is assumed that the clock skew is bounded, i.e., $1 - \sigma < \alpha_i < 1 + \sigma$, where $\sigma \in [30, 100]$ ppm. For practical applications, environmental variations and oscillator aging change slowly; therefore, it can be reasonably assumed that the skew α_i remains constant within a short synchronization interval. Since local nodes lack access to an absolute time reference, direct computation and adjustment of the hardware clock $H_i(t)$ are not feasible. To address this issue, the concept of a logical clock is introduced to represent the synchronized clock among nodes, i.e.,

$$L_i(t) = \hat{\alpha}_i(t) H_i(t) + \hat{\beta}_i(t) = \hat{\alpha}_i(t)\alpha_i t + \hat{\alpha}_i(t)\beta_i + \hat{\beta}_i(t) \tag{2}$$

where $\hat{\alpha}_i(t)$ and $\hat{\beta}_i(t)$ denote the compensation parameters for logical clock skew and offset, respectively. $s_i = \hat{\alpha}_i(t)\alpha_i$ and $o_i = \hat{\alpha}_i(t)\beta_i + \hat{\beta}_i(t)$ are referred to as the logical clock skew and offset of node i. When the logical clocks of all nodes converge to a common value, the network achieves global synchronization, which means that

$$\lim_{t\to\infty} |L_j(t) - L_i(t)| = 0, \forall i, j \in V \tag{3}$$

In distributed CTSPs, where no fixed global reference node exists, the relative clock skew α_{ij} and offset β_{ij} serve as two fundamental parameters for characterizing differences between two clocks. Specifically, $\alpha_{ij} = \alpha_j/\alpha_i$ reflects the frequency ratio between two oscillators and is estimated using two pairs of hardware timestamps, i.e., $\alpha_{ij}(t) = (H_j(t) - H_j(t-1))/(H_i(t) - H_i(t-1))$. $\beta * ij(t) = L * j(t) - L_i(t)$ is used to describe the instantaneous logical clock difference between nodes i and j.

2.2. Average-Based Consensus Time Synchronization Problem

It can be observed that the core requirement of (3) is to achieve dual convergence of both logical clock skew and offset states, i.e., $\lim_{t\to\infty} |s_j - s_i| = 0$, $\lim_{t\to\infty} |o_j - o_i| = 0$, $\forall i, j \in V$. Usually, CTSPs employ a linear consensus model to describe the synchronization process [5,6]. The dynamic equation is given as follows [32]:

$$x_i(k+1) = x_i(k) + \sum_{j\in N_i} a_{ij}(x_j(k) - x_i(k)) \tag{4}$$

where x_i represents the clock state of node i. If $x_i = x_j$, $\forall i, j \in V$, this indicates that the states of all nodes are consistent. Let $x_i(0)$ denote the initial state, and let x denote the asymptotic convergence state of the system. As established in [19,32], the final convergence result of (4) is equal to the average of the initial clock states of all nodes, that is,

$$\lim_{t\to\infty} x = \frac{1}{n}\sum_{i=1}^{n} x_i(0) \tag{5}$$

The convergence speed of average-based consensus theory is fundamentally governed by the algebraic connectivity λ_2 of the network Laplacian graph [32], exhibiting a direct positive correlation, where faster convergence is achieved with a larger value of λ_2. Although expanding the communication radius of nodes, increasing network density [11], and constructing virtual links [21,23,25,26] to create richer interconnections can effectively increase algebraic connectivity, these schemes inevitably introduce practical limitations, including increased hardware costs, packet collisions, network congestion, power consumption, and delay jitter. To overcome the above challenges, this paper designs a high-order consensus protocol.

3. Fast High-Order Consensus Protocol

The proposed high-order consensus protocol is an innovative extension of virtual communication mechanism, aimed at utilizing the historical state information of 2-hop neighbors to achieve fast convergence. To reduce the unnecessary overhead of relay communication, the proposed protocol strategically retains a single-hop communication architecture. The high-order consensus dynamic equation can be formulated as

$$\begin{aligned} x_i(k+1) = x_i(k) + \varepsilon \sum_{j \in N_i} a_{ij} \{ & (x_j(k) - x_i(k) \\ & + \sum_{l \in N_j} a_{jl}(x_l(k-1) - x_i(k)) + \cdots \\ & + \sum_{l \in N_j} a_{jl}(x_l(k-m) - x_i(k)) \} \end{aligned} \tag{6}$$

where $x_i(k)$ denotes the clock state of node i at time k. ε is a weight parameter. Node j is the 1-hop neighbor of node i, while node l is the 2-hop neighbor. At k-th broadcast times, node j will send its current clock status $x_j(k)$ to node i along with node l's historical state $x_l(k-m)$. It should be noted that rather than being directly appended to node j's packet, multiple $x_l(k-m)$ are utilized to construct a new smaller parameter, thereby reducing additional packet size. The implementation details will be thoroughly described in the design of HCTSP in Section 4.

3.1. Convergence Analysis

We begin the convergence analysis of high-order consensus protocol by reformulating (6) in vector form

$$\boldsymbol{x}(k+1) = (\boldsymbol{I}_n - \varepsilon \boldsymbol{L} - m\varepsilon \boldsymbol{D}_2)\boldsymbol{x}(k) + \varepsilon \boldsymbol{A}_2 \sum_{t=1}^{m} \boldsymbol{x}(k-t) \tag{7}$$

where $\boldsymbol{x}(k) = [x_1(k), x_2(k), \cdots, x_n(k)]^{\mathrm{T}}$, $\boldsymbol{A}_2$ is the adjacency matrix of the 2-hop topology of $G = (V, E)$. Here, $\boldsymbol{D}_2 = \mathrm{diag}(\sum_{j \in N_i} a_{ij} \sum_{l \in N_j} a_{jl})$, $i \neq l, \forall i, j, l \in V$.

Thus (7) can be rewritten as

$$\boldsymbol{X}(k+1) = \boldsymbol{C}\boldsymbol{X}(k) \tag{8}$$

where $\boldsymbol{X}(k+1) = [\boldsymbol{x}(k+1), \cdots \boldsymbol{x}(k-m+1)]^{\mathrm{T}}$, and

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_n - \varepsilon \boldsymbol{L} - m\varepsilon \boldsymbol{D}_2 & \varepsilon \boldsymbol{A}_2 & \cdots & \cdots & \varepsilon \boldsymbol{A}_2 \\ \boldsymbol{I}_n & 0 & \cdots & \cdots & 0 \\ 0 & \boldsymbol{I}_n & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{I}_n & 0 \end{bmatrix} \tag{9}$$

Let

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{K} & 0 & \cdots & 0 \\ \boldsymbol{K} & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \boldsymbol{K} & 0 & \cdots & 0 \end{bmatrix} \tag{10}$$

where $\boldsymbol{K} = \frac{1}{n}\boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}, \boldsymbol{1} = [1, 1, \cdots, 1]^{\mathrm{T}}$ is an n-dimensional vector.

Lemma 1. *$\boldsymbol{C}$ shares identical eigenvalues with matrix $\boldsymbol{C} - \boldsymbol{J}$, except that $\lambda(\boldsymbol{C}) = 1$ is replaced by $\lambda(\boldsymbol{C} - \boldsymbol{J}) = 0$.*

Proof. Construct a matrix $\tilde{\boldsymbol{C}} = \boldsymbol{C} - \boldsymbol{1}\boldsymbol{v}^{\mathrm{T}}$, where $\boldsymbol{v} \in \mathbb{R}^n$ is an arbitrarily given n dimensional vector. For any invertible matrix $\boldsymbol{C}$, according to the determinant lemma [33],

$$\det(\tilde{\boldsymbol{C}} - \lambda_{\tilde{c}} \boldsymbol{I}_n) = \det(\boldsymbol{C} - \boldsymbol{1}\boldsymbol{v}^{\mathrm{T}} - \lambda_{\tilde{c}} \boldsymbol{I}_n) \tag{11}$$

Next, we discuss the invertibility of matrix $\boldsymbol{C} - \lambda_{\tilde{c}}\boldsymbol{I}_n$. After performing elementary row operations on $\boldsymbol{C} - \lambda_{\tilde{c}}\boldsymbol{I}_n$, it strictly satisfies the upper triangular matrix form. Therefore, it is easy to prove that when $\lambda_{\tilde{c}} \neq 0$, $\lambda_{\tilde{c}} \neq 1$ and $\lambda_{\tilde{c}} \neq 1 - \varepsilon\lambda_k$, Matrix $\boldsymbol{C} - \lambda_{\tilde{c}}\boldsymbol{I}_n$ is invertible.

Thus, (11) can be further applied

$$\det(\tilde{\boldsymbol{C}} - \lambda_{\tilde{c}}\boldsymbol{I}_n) = \det(\boldsymbol{C} - \lambda_{\tilde{c}}\boldsymbol{I}_n) \cdot (1 - \boldsymbol{v}^{\mathrm{T}}(\boldsymbol{C} - \lambda_{\tilde{c}}\boldsymbol{I}_n)^{-1}\mathbf{1}) \tag{12}$$

The characteristic equation is

$$\det(\boldsymbol{C} - \lambda_{\tilde{c}}\boldsymbol{I}_n) \cdot (1 - \boldsymbol{v}^{\mathrm{T}}(\boldsymbol{C} - \lambda_{\tilde{c}}\boldsymbol{I}_n)^{-1}\boldsymbol{I}_n) = 0 \tag{13}$$

From (13), there is (i) if $\lambda_c \neq 1$, $\det(\boldsymbol{C} - \lambda_{\tilde{c}}\boldsymbol{I}_n) = 0$; that is, $\lambda_{\tilde{c}}$ is an eigenvalue of $\boldsymbol{C}$; (ii) if $\lambda_c = 1$, $1 - v^{\mathrm{T}}(\boldsymbol{C} - \lambda_{\tilde{c}}\boldsymbol{I}_n)^{-1}\boldsymbol{I}_n = 0$; that is, $\lambda_{\tilde{c}} = 0$ is replaced by $\lambda_c = 1$. Therefore, the proof is complete. □

Theorem 1. *For an undirected graph G that is strongly connected, the higher-order consensus protocol ensures convergence to the average of the initial stages if the spectral radius condition* $\rho(\boldsymbol{C} - \boldsymbol{J}) < 1$ *holds under the given initial conditions* $\boldsymbol{x}(0) = \boldsymbol{x}(-1) = \cdots \boldsymbol{x}(-m+1)$ [31].

Proof. First, We have a region S that satisfies $\rho(\boldsymbol{C} - \boldsymbol{J}) < 1$; that is to say,

$$S = \{\varepsilon \mid \rho(\boldsymbol{C} - \boldsymbol{J}) < 1\} \tag{14}$$

if and only if

$$0 < \varepsilon < \frac{2}{\lambda_1(\boldsymbol{L} + m\boldsymbol{D}_2 + (m-1)\boldsymbol{A}_2)} \tag{15}$$

where $\lambda_i(\cdot)$ denotes the ith largest eigenvalue of a symmetric matrix. The proof regarding the spectral radius in Theorem 1 is similar to [34]. Here, we present a sketch proof. □

In the orthogonal complement subspace, $\boldsymbol{K}\mathbf{1} = \mathbf{1}$, and for $u \perp \mathbf{1}$, $\boldsymbol{K}\boldsymbol{u} = \mathbf{0}$. So the matrix $\boldsymbol{C} - \boldsymbol{J}$ reduces to a block companion matrix, and its eigenvalues satisfy $\det\big(\mu_i\boldsymbol{I}_n - \boldsymbol{W} + \varepsilon\boldsymbol{A}_2(\frac{1}{\mu_i} + \cdots + \frac{1}{\mu_i^{m-1}})\big) = 0$, where $\boldsymbol{W} = \boldsymbol{I}_n - \varepsilon\boldsymbol{L} - m\varepsilon\boldsymbol{D}_2$. Since $\boldsymbol{C}$ depends on ε and $\mu_i = 1$ at $\varepsilon = 0$, a perturbation expansion gives $\mu_i = 1 - \varepsilon\lambda_i + O(\varepsilon^2)$, where λ_i are eigenvalues of a reference matrix (e.g., $\boldsymbol{L} + m\boldsymbol{D}_2 + (m-1)\boldsymbol{A}_2$). These λ_i appear in the upper bound condition later.

For the dominant eigenvalue, the characteristic equation can be approximated as $\det((1 - \varepsilon\lambda_i)\boldsymbol{I_n}) - \boldsymbol{W} + (m-1)\varepsilon\boldsymbol{A}_2) = 0$. The requirement is $|\mu_i| < 1$, i.e., $|1 - \varepsilon\lambda_i| < 1$. Moreover, given the positive semi-definiteness of $\boldsymbol{L}$ and that $\boldsymbol{D}_2$ is a non-negative matrix, $0 < \varepsilon < \frac{2}{\lambda_1(\boldsymbol{L}+m\boldsymbol{D}_2+(m-1)\boldsymbol{A}_2)}$.

For non-dominant eigenvalues ($\mu_i \ll 1$), consider the exact characteristic equation of $\boldsymbol{C}$ restricted to the subspace orthogonal to $\mathbf{1}$:

$$\det(\mu^m\boldsymbol{I}_n - \mu^{m-1}(\boldsymbol{I}_n - \varepsilon\boldsymbol{L} - m\varepsilon\boldsymbol{D}_2) - \varepsilon\boldsymbol{A}_2(1 + \mu + \cdots + \mu^{m-1})) = 0. \tag{16}$$

Let $\boldsymbol{v}$ be a unit eigenvector with $\boldsymbol{v} \perp \mathbf{1}$ corresponding to μ. Rearranging the eigenvalue equation gives

$$\mu = 1 - \varepsilon\boldsymbol{v}^T\boldsymbol{L}\boldsymbol{v} - m\varepsilon\boldsymbol{v}^T\boldsymbol{D}_2\boldsymbol{v} + \varepsilon\boldsymbol{v}^T\boldsymbol{A}_2\boldsymbol{v} \cdot \frac{1 - \mu^m}{\mu^{m-1}(1 - \mu)}. \tag{17}$$

We now show that under condition (15), every such μ satisfies $|\mu| < 1$. Assume, to the contrary, that $|\mu| \geq 1$ and $\mu \neq 1$. Because $\boldsymbol{L}$ and $\boldsymbol{D}_2$ are positive semi-definite, we have

$\boldsymbol{v}^T\boldsymbol{L}\boldsymbol{v} \geq \lambda_2 > 0$ (where λ_2 is the algebraic connectivity) and $\boldsymbol{v}^T\boldsymbol{D}_2\boldsymbol{v} \geq 0$. Moreover, for the symmetric matrix $\boldsymbol{A}_2$, the Rayleigh quotient satisfies $|\boldsymbol{v}^T\boldsymbol{A}_2\boldsymbol{v}| \leq \rho(\boldsymbol{A}_2)$ (spectral radius).

$$\left|\frac{1-\mu^m}{\mu^{m-1}(1-\mu)}\right| = \left|\frac{1+\mu+\cdots+\mu^{m-1}}{\mu^{m-1}}\right| \leq \frac{1+|\mu|+\cdots+|\mu|^{m-1}}{|\mu|^{m-1}} \leq m. \tag{18}$$

Taking absolute values in the previous equation and using the triangle inequality yields

$$|\mu| \leq 1 - \varepsilon\boldsymbol{v}^T\boldsymbol{L}\boldsymbol{v} - m\varepsilon\boldsymbol{v}^T\boldsymbol{D}_2\boldsymbol{v} + m\varepsilon|\boldsymbol{v}^T\boldsymbol{A}_2\boldsymbol{v}|. \tag{19}$$

Denote $a = \boldsymbol{v}^T\boldsymbol{L}\boldsymbol{v}$, $b = \boldsymbol{v}^T\boldsymbol{D}_2\boldsymbol{v}$, $c = |\boldsymbol{v}^T\boldsymbol{A}_2\boldsymbol{v}|$. Then the inequality becomes $|\mu| \leq 1 - \varepsilon a - m\varepsilon b + m\varepsilon c$. Using the Rayleigh quotient bounds and condition (15), we have $\varepsilon a \geq \varepsilon\lambda_2$ and $\varepsilon c \leq \varepsilon\rho(\boldsymbol{A}_2)$. Since ε is bounded by $\frac{2}{\lambda_1(\boldsymbol{L}+m\boldsymbol{D}_2+(m-1)\boldsymbol{A}_2)}$, a direct calculation (see, e.g., the Gershgorin circle argument in [34]) shows that $1 - \varepsilon a - m\varepsilon b + m\varepsilon c < 1$. Hence $|\mu| < 1$, contradicting the assumption $|\mu| \geq 1$. Therefore all non-dominant eigenvalues satisfy $|\mu_i| < 1$ and decay exponentially.

In the all-ones space, the action of matrix $\boldsymbol{C} - \boldsymbol{J}$ is dominated by the low-dimensional subsystem. Since $\boldsymbol{D}_2\mathbf{1} = \boldsymbol{A}_2\mathbf{1}$, when ε is sufficiently small, the perturbation term $m\varepsilon\boldsymbol{D}_2\mathbf{1}$ is not enough to make the modulus exceed 1, so the modulus of this subspace is also less than 1. In summary, when $0 < \varepsilon < \frac{2}{\lambda_1(\boldsymbol{L}+m\boldsymbol{D}_2+(m-1)\boldsymbol{A}_2)}$, the condition $\rho(\boldsymbol{C} - \boldsymbol{J}) < 1$ is satisfied.

Next, we continue the proof of Theorem 1 under the constraint condition. For the undirected communication topology G, both $\boldsymbol{L}$ and $\boldsymbol{L}_2$ are symmetric matrices with zero row and column sums. Hence, $\boldsymbol{C}$ has row sums equal to 1 and must possess an eigenvalue of 1. By Lemma 1, matrix $\boldsymbol{C}$, except for the eigenvalue 1, has the same eigenvalues as matrix $\boldsymbol{C} - \boldsymbol{J}$. When $\boldsymbol{C} - \boldsymbol{J}$ satisfies the spectral radius condition $\rho(\boldsymbol{C} - \boldsymbol{J}) < 1$, $\boldsymbol{C}$ consequently has a unique eigenvalue of 1 and the remaining eigenvalues satisfy $|\lambda(\boldsymbol{C})| < 1$.

Furthermore, for the eigenvalue 1 of $\boldsymbol{C}$, there exists a left eigenvector $\boldsymbol{\omega}_l$ and a right eigenvector $\boldsymbol{\omega}_r$, respectively. Let matrix $\boldsymbol{U}$ be the Jordan canonical form of matrix $\boldsymbol{C}$ and $\boldsymbol{S}$ be its corresponding eigenvector matrix, such that $\boldsymbol{C} = \boldsymbol{S}\boldsymbol{U}\boldsymbol{S}^{-1}$; then, $\boldsymbol{C}^k = \boldsymbol{S}\boldsymbol{U}^k\boldsymbol{S}^{-1}$. Hence,

$$\lim_{k\to\infty}\boldsymbol{C}^k = \lim_{k\to\infty}\begin{bmatrix} \boldsymbol{I}_n - \varepsilon\boldsymbol{L} - m\varepsilon\boldsymbol{D}_2 & \varepsilon\boldsymbol{A}_2 & \cdots & \cdots & \varepsilon\boldsymbol{A}_2 \\ \boldsymbol{I}_n & 0 & \cdots & \cdots & 0 \\ 0 & \boldsymbol{I}_n & \cdots & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{I}_n & 0 \end{bmatrix}^k = \boldsymbol{\omega}_l\boldsymbol{\omega}_r^{\mathrm{T}} \tag{20}$$

Define $\boldsymbol{\omega}_l = (\mathbf{1}^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}}, \cdots, \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}$, $\boldsymbol{\omega}_r = (\mathbf{1}^{\mathrm{T}}, \mathbf{1}^{\mathrm{T}}, \cdots, \mathbf{1}^{\mathrm{T}})^{\mathrm{T}}$. According to the above equation, there is

$$\lim_{k\to\infty}\begin{bmatrix} \boldsymbol{X}(k+1) \\ \boldsymbol{X}(k) \\ \vdots \\ \boldsymbol{X}(k-m+2) \end{bmatrix} = \boldsymbol{\omega}_r\boldsymbol{\omega}_l^{\mathrm{T}}\begin{bmatrix} \boldsymbol{X}(0) \\ \boldsymbol{X}(-1) \\ \vdots \\ \boldsymbol{X}(-m+1) \end{bmatrix} \tag{21}$$

i.e., $\lim_{k\to\infty}\boldsymbol{X}(k+1) = \lim_{k\to\infty}\boldsymbol{C}\boldsymbol{X}(k) = \boldsymbol{\omega}_r\boldsymbol{\omega}_l^{\mathrm{T}}\boldsymbol{X}(0)$. So we can conclude that $\lim_{k\to\infty}\boldsymbol{x}(k+1) = \frac{1}{n}\mathbf{1}^{\mathrm{T}}\sum_{i=1}^{n}x_i(0)$. The proof of Theorem 1 is complete.

3.2. Convergence Speed Analysis

The convergence speed of distributed consensus algorithms can be intuitively compared through the second smallest eigenvalue of the Laplacian matrix [35]. However, given the computational complexity of exact eigenvalue calculation, a simplified method

is employed to roughly compare the convergence speed between high-order consensus algorithms and multi-hop consensus algorithms [11] under identical hop-count constraints.

Considering the 2-hop algorithm as a representative case, we first rewrite (7) in high-order consistent vector form as

$$\begin{aligned}\boldsymbol{x}(k+1) =&(\boldsymbol{I}_n - \varepsilon\boldsymbol{L} - m\varepsilon\boldsymbol{D_2} + m\varepsilon\boldsymbol{A_2})\boldsymbol{x}(k) + \varepsilon\boldsymbol{A}_2[\boldsymbol{x}(k-1) - \boldsymbol{x}(k)] \\ &+ \cdots + \varepsilon\boldsymbol{A}_2[\boldsymbol{x}(k-m) - \boldsymbol{x}(k)]\end{aligned} \tag{22}$$

As rigorously proved in the previous section, if the high-order consensus algorithm satisfies the convergence condition, i.e.,

$$\lim_{k\to\infty} \boldsymbol{x}(k+1) = \lim_{k\to\infty} \boldsymbol{x}(k) = \cdots \lim_{k\to\infty} \boldsymbol{x}(k-m) \tag{23}$$

After a finite number of iterations, $\|\boldsymbol{x}(k) - \boldsymbol{x}(k-1)\|_2 \to 0, \cdots, \|\boldsymbol{x}(k) - \boldsymbol{x}(k-m+1)\|_2 \to 0$. Thus, we can get the following approximation

$$\boldsymbol{x}(k+1) = (\boldsymbol{I}_n - \varepsilon\boldsymbol{L} - m\boldsymbol{D_2} + m\varepsilon\boldsymbol{A}_2)\boldsymbol{x}(k) = (\boldsymbol{I}_n - \varepsilon\boldsymbol{L} - m\varepsilon\boldsymbol{L}_2)\boldsymbol{x}(k) \tag{24}$$

For a traditional first-order 2-hop relay forwarding algorithm, it can also be approximated as

$$\boldsymbol{x}(k+1) \approx (\boldsymbol{I}_n - \varepsilon\boldsymbol{L} - \varepsilon\boldsymbol{L}_2)\boldsymbol{x}(k) \tag{25}$$

Therefore, the convergence speed comparison between the high-order consensus algorithm and the traditional multi-hop algorithm can be achieved by analyzing the second smallest eigenvalue of the matrices $(\boldsymbol{I}_n - \varepsilon\boldsymbol{L} - m\varepsilon\boldsymbol{L}_2)$ and $(\boldsymbol{I}_n - \varepsilon\boldsymbol{L} - \varepsilon\boldsymbol{L}_2)$. Due to the symmetric positive semi-definiteness of the Laplacian matrices $\boldsymbol{L}$, $\boldsymbol{L_2}$, it is obvious that when $m > 1$, the high-order consensus algorithm converges faster.

4. Fast Time Synchronization Protocol: HCTSP

The core idea of HCTSP is to exchange multi-hop clock information within a single-hop communication model to establish virtual links between local nodes and their multi-hop neighboring nodes. As illustrated in Figure 1, nodes j and l are the one-hop and two-hop neighbors of node i, respectively. In each broadcast round, node j transmits its local clock state together with information from its single-hop neighbor l. Meanwhile, node i utilizes both one-hop and two-hop clock information to update its local clock. As a result, virtual communication links can be established between node i and its nonadjacent node l.

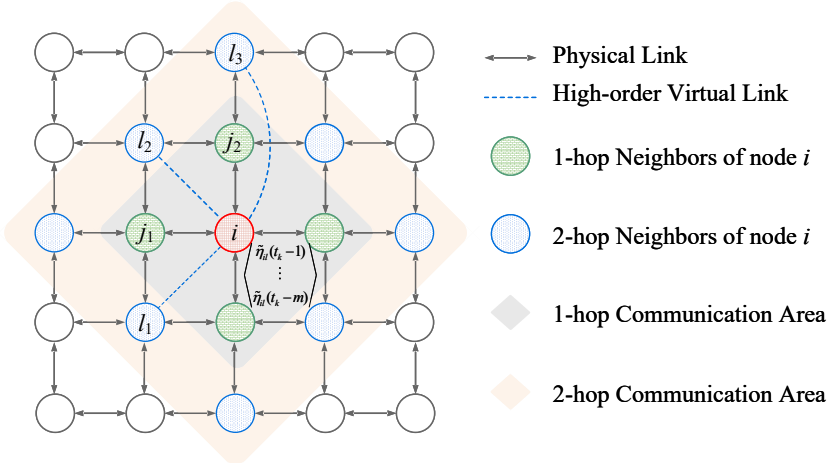


Figure 1. 2-hop HCTSP indication. Illustrates how local node i establishes a virtual link with multi-hop node l by receiving synchronization packet data from neighbor node j.

Figure 2 presents the detailed workflow of HCTSP. The proposed HCTSP primarily consists of two key phases: relative clock parameter estimation and logical clock compensation value calculation.

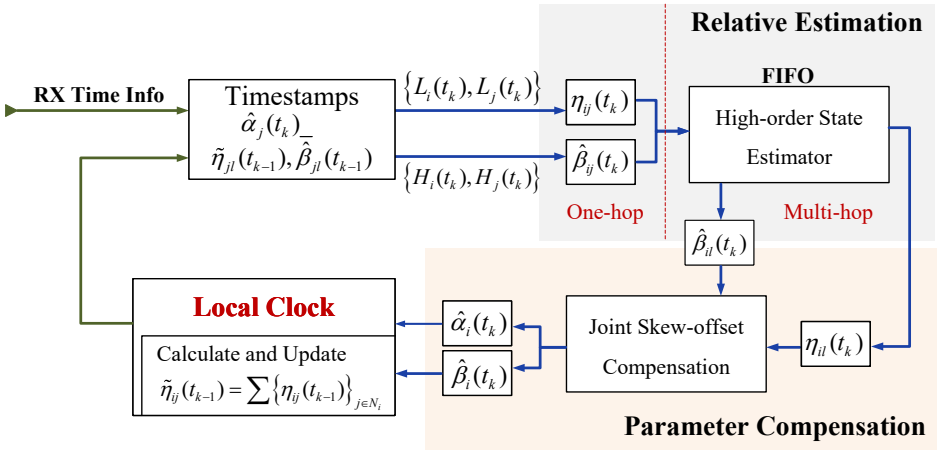


Figure 2. Implementation of HCTSP. RX time information from neighboring node j is utilized for one-hop estimation and multi-hop synchronization.

4.1. Relative Clock Parameter Estimation

Clock skew and offset are two fundamental parameters governing logical clock behavior in distributed synchronization protocols. Within these protocols, nodes iteratively process timestamps from neighboring nodes to calculate and update their local clock states. During the update process, accurately estimating relative clock parameters between nodes is crucial, as this estimation forms the foundation for achieving high-precision time synchronization. In HCTSP, this estimation process occurs at two levels: between directly adjacent nodes and among multi-hop connected nodes. Furthermore, to mitigate the impact of uncertain delays caused by timing message delivery, a media access control (MAC) layer-based physical timestamp is adopted [3].

4.1.1. Relative Parameter Estimation Between Adjacent Nodes

Upon receiving timestamps from adjacent node j at t_k, node i records the arrival time and estimates the relative clock skew α_{ij} and offset $\beta * ij$ by employing two successive timestamp pairs, as in regular CTSPs. To clearly express the clock adjustment process, we define the following notation $\eta * ij(t_k)$ as the skew compensation of node i calculated from the timing information of neighboring node j.

$$\eta_{ij}(t_k) = \alpha_{ij}(t_k)\hat{\alpha}_j(t_k) \tag{26}$$

4.1.2. Relative Parameter Estimation Among Non-Adjacent Nodes

Since hardware timestamp information cannot be directly acquired from non-adjacent nodes and to minimize communication overhead from relay forwarding, HCTSP avoids direct hardware timestamp transmission in virtual communication. However, the relative clock skew between node i and its non-adjacent node l can be derived indirectly through the expression $\alpha_{il}(t_k) = \alpha_{ij}(t_k)\alpha_{jl}(t_k)$. Hence, node i's skew compensation calculated from node l's timing information satisfies

$$\eta_{il}(t_k) = \alpha_{ij}(t_k)\alpha_{jl}(t_k)\hat{\alpha}_l(t_k) = \alpha_{ij}(t_k)\eta_{jl}(t_k) \tag{27}$$

Here, we refer to $\eta_{il}(t_k)$ as the first-order skew compensation.

The convergence analysis in Section 3.1 has demonstrated that incorporating multi-hop node historical information significantly accelerates the convergence speed of distributed consensus algorithms. Therefore, the real-time cumulative high-order skew compensation is

$$\begin{aligned}\tilde{\eta}_{il}(t_k) &= \alpha_{ij}(t_k)\eta_{jl}(t_{k-1}) + \alpha_{ij}(t_k)\eta_{jl}(t_{k-2}) \\ &\quad + \cdots + \alpha_{ij}(t_k)\eta_{jl}(t_{k-m}) \\ &= \alpha_{ij}(t_k)\sum_{n=1}^{m}\eta_{jl}(t_{k-n})\end{aligned} \tag{28}$$

While the relative logical clock offset between node i and its non-adjacent node l can be simply estimated as

$$\beta_{il}(t_k) = \beta_{ij}(t_k) + \bar{\beta}_{jl}(t_k) \tag{29}$$

where $\bar{\beta}_{jl}(t_k) = \frac{\sum_{l\in N_j}\beta_{jl}(t_{k-1})}{|N_j|+1}$, i.e., node j sends a calculated average of $\left\{\beta_{jl(t_{k-1})}\right\}_{l\in N_j}$ to node i.

4.2. Logical Clock Compensation Values Calculation

To accelerate convergence, we design a joint skew-offset compensation calculation for node adjustment based on the fast high-order consensus protocol proposed in Section 3. The calculation process is an extension of GTSP [4].

First, we estimate the logical skew compensation parameter through high-order consensus principle; that is

$$\hat{\alpha}_i(t_k^+) = \frac{\sum_{j\in N_i}\eta_{ij}(t_k) + \sum_{l\in N_i'}\tilde{\eta}_{il}(t_k) + \hat{\alpha}_i(t_k)}{|N_i| + m|N_i'| + 1} \tag{30}$$

where t_k^+ is the updated instantaneous moment and $|N_i'|$ represents the number of node i's 2-hop neighbors.

To prove that logical clock skew compensation (30) satisfies the high-order consensus theory, we further conduct the following verification process. It should be emphasized that the logical clock skew compensation calculation serves to synchronize the logical clock skew, i.e., $s_i = s_j$. Therefore, by multiplying both sides of (30) by $\alpha_i(t_k)$, we have

$$\begin{aligned}\hat{\alpha}_i(t_k^+)\alpha_i(t_k) =& \frac{1}{|N_i| + m|N_i'| + 1}\left\{\sum_{j\in N_i}\eta_{ij}(t_k)\alpha_i(t_k)\right. \\ &\left. + \sum_{l\in N_i'}\tilde{\eta}_{il}(t_k)\alpha_i(t_k) + \hat{\alpha}_i(t_k)\alpha_i(t_k)\right\} \\ s_i(t_k^+) =& \frac{1}{|N_i| + m|N_i'| + 1}\left\{\sum_{j\in N_i}s_j(t_k)\right. \\ &\left. + \sum_{l\in N_i'}\sum_{n=1}^{m}[s_l(t_{k-n}) - s_i(t_k)]\right\}\end{aligned} \tag{31}$$

By further expanding (31) into its complete recurrence formula, we derive

$$\begin{aligned}s_i(t_k^+) =& \frac{1}{|N_i| + m|N_i'| + 1}\left\{s_i(t_k) + \sum_{j\in N_i}[s_j(t_k) - s_i(t_k)] + \right. \\ &\left. \sum_{l\in N_i'}[s_l(t_{k-1}) - s_i(t_k)] + \cdots + [s_l(t_{k-m}) - s_i(t_k)]\right\}\end{aligned} \tag{32}$$

According to (32), the logical clock skew update satisfies the high-order consensus algorithm proposed in (6). Consequently, after sufficient update iterations, all network nodes will achieve asymptotic consensus in their logical clock skews. Furthermore, compared with the historical difference term recorded within the same period [31], this cross-time difference term, which incorporates both local current states and multi-hop historical states, can effectively capture the time-accumulated deviation of the relative logical clock skew and mitigate the convergence lag.

Second, the logical clock offset compensation of node i is estimated by averaging the relative clock offsets of all one-hop and two-hop neighboring nodes; that is,

$$\hat{\beta}_i(t_k^+) = \hat{\beta}_i(t_k) + \frac{\sum_{j\in N_i}\beta_{ij}(t_k) + \sum_{l\in N_i'}\beta_{il}(t_k)}{|N_i| + |N_i'| + 1} \tag{33}$$

4.3. Implementation of HCTSP

The corresponding pseudo-code of HSCSP is presented in Algorithm 1. It mainly consists of the following 4 steps:

1. **Asynchronous Periodic Neighbor Broadcasting**: As shown in Figure 3, HCTSP employs a unidirectional broadcast mechanism (see Algorithm 1, lines 3–5). Due to different clock skews, each node autonomously determines its broadcast period T_s based on its hardware clock, enabling network-wide asynchronous pseudo-periodic broadcasting.
2. **Relative Clock Parameter Estimation**: Upon receiving time information from neighbor node j, node i calculates both 1-hop and 2-hop neighbors relative clock parameters $\{\eta_{ij}(t_k), \beta_{ij}(t_k)\}$, $\{\tilde{\eta}_{il}(t_k), \beta_{il}(t_k)\}$ by processing the received timestamps $\left\langle H_j(t_k), L_j(t_k), \hat{\alpha}_j(t_k)\right\rangle$ and shared information $\left\langle \tilde{\eta}_{jl}(t_{k-1}), \bar{\beta}_{jl}(t_{k-1})\right\rangle$.
3. **Local Clock Compensation**: As shown in Figure 2, each node calculates its current compensation parameters $\{\hat{\alpha}_i(t_k), \hat{\beta}_i(t_k)\}$ by integrating the relative clock parameters output by the high-order state estimator and the previous compensation values. These parameters are then applied to adjust the logical clock $L_i(t_k)$.
4. **High-order Information Storage**: Node i maintains clock state information by storing $\eta_{ij}(t_k)$ of all neighbors between consecutive broadcast rounds, and then calculates $\tilde{\eta}_{ij}(t_{k-1})$ for the construction of next virtual links.

Next, we present a detailed description of the HCTSP implementation, focusing on its two core operational aspects: information broadcasting and parameter updating.

Unlike traditional multi-hop relay forwarding, HCTSP adopts a single-hop communication model to reduce the complexity and communication overhead associated with high-frequency packet forwarding. Each broadcast packet contains the sending timestamp, including the hardware and logical clocks, and the calculated relative clock parameters for both one-hop and multi-hop neighboring nodes collected during the broadcast interval. Notably, prior work, i.e., VTSP [25], requires embedding all historically stored relative clock skews of neighboring nodes in broadcast packets to construct virtual links with multi-hop nodes, thereby introducing additional communication overhead.

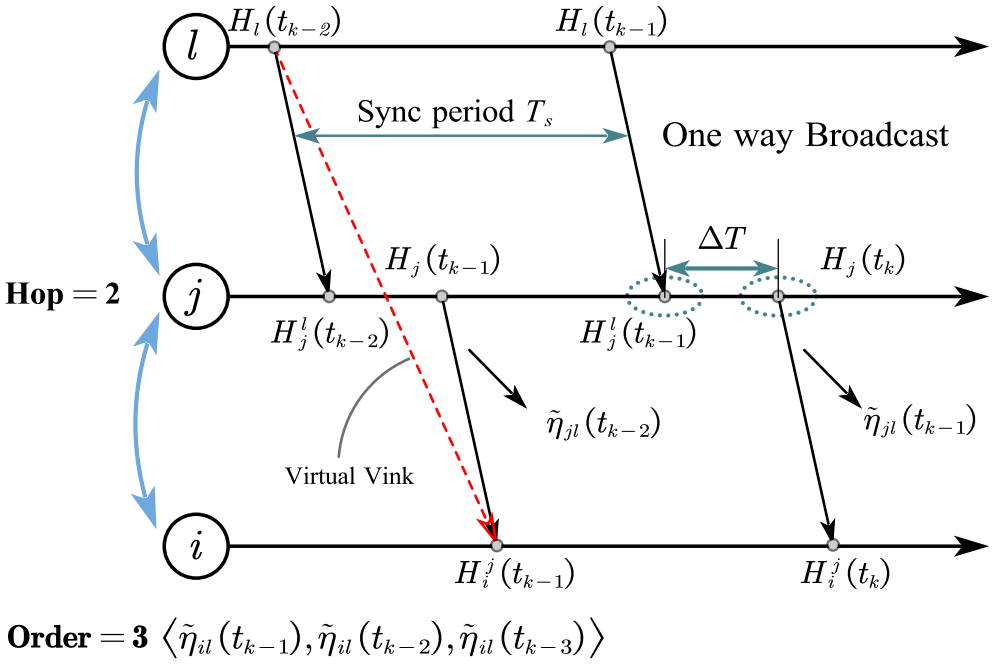


Figure 3. One-way communication model for HCTSP, which uses MAC layer timestamp technology to record the timestamp of the node receiving the RX Time Info.

Algorithm 1 Pseudo-code algorithm of HCTSP (taking 2-hop communication and 3-order algorithm as an example, i.e., $H = 2$, $M = 3$) where $j \in N_i$, $l \in N_j$.

Initialization:
  Set $\hat{\alpha}_j(t_0) = 1$, $\hat{\beta}_j(t_0) = 0$, $\eta_{ij}(t_0) = 1$, $\tilde{\eta}_{jl}(t_0) \to 0$.
  Any node j broadcasts timing message periodically with period T_s.
Upon node *j* triggers the broadcast task
  Broadcast $\left\langle H_j(t_k), L_j(t_k), \hat{\alpha}_j(t_k), \tilde{\eta}_{jl}(t_{k-1}), \bar{\beta}_{jl}(t_{k-1}) \right\rangle$ to all neighbors.
Upon node *i* receives *j*'s message
  Store $\hat{\alpha}_j(t_k)$, $\tilde{\eta}_{jl}(t_{k-1})$, $\bar{\beta}_{jl}(t_{k-1})$ and $|N_j|$;
  Record reception timestamp $H_j(t_k)$;
  Calculate $\eta_{ij}(t_k)$, and update $\{\eta_{ij}(t_k)\} \leftarrow \eta_{ij}(t_k)$;
  Calculate $\beta_{ij}(t_k)$, and update $\{\beta_{ij}(t_k)\} \leftarrow \beta_{ij}(t_k)$.
Calculate shared parameters for real-time
  $\tilde{\eta}_{ij}(t_{k-1}) = \sum_{j \in N_i} \{\eta_{ij}(t_k)\}$;
  $\bar{\beta}_{ij}(t_{k-1}) = \frac{\sum_{j \in N_i} \{\beta_{ij}(t_k)\}}{|N_i|+1}$.
if $t_k > M$ **then**
  Use FIFO rule to perform sliding storage for $\left\{\eta_{jl}(t_{k-1})\right\}_{l \in N_i'}$;
  Update $\left\{\eta_{jl}(t_{k-1})\right\}_{l \in N_i'} \leftarrow \tilde{\eta}_{jl}(t_k)$, and calculate $\tilde{\eta}_{jl}(t_{k-1}) = \sum \left\{\eta_{jl}(t_{k-1})\right\}_{l \in N_i'}$;
  Calculate $\tilde{\eta}_{il}(t_k) = \alpha_{ij}(t_k)\tilde{\eta}_{jl}(t_{k-1})$, and update $\{\tilde{\eta}_{il}(t_k)\}_{l \in N_i'} \leftarrow \tilde{\eta}_{il}(t_k)$;
  Calculate $\beta_{il}(t_k) = \beta_{ij}(t_k) + \bar{\beta}_{jl}(t_{k-1})$, and update $\{\beta_{il}(t_k)\}_{l \in N_i'} \leftarrow \beta_{il}(t_k)$.
end if
Upon node *i* triggers the update time
  Calculate $\hat{\alpha}_i(t_k)$ and $\hat{\beta}_i(t_k)$;
  Update $L_i(t_k)$;
  Update $\langle H_j(t_{k-1}), H_i(t_{k-1}) \rangle \leftarrow \langle H_j(t_k), H_i(t_k) \rangle$.
End

Although VTSP alleviates the overhead problem through redundant data filtering and passive listening by edge nodes, direct information aggregation may be unreasonable for large-scale dense networks. To avoid this shortcoming, HCTSP introduces an accumulator-based redundant information optimization scheme (see Algorithm 1, lines 11–13). Upon receiving the timing information, node i calculates the relative clock skew $\eta_{ij}(t_k)$ and stores it in local memory $\{\eta_{ij}(t_k)\}_{j\in N_i}$. Moreover, the local memory of node i performs real-time accumulation of the calculated relative clock skew values of all neighboring nodes j. Meanwhile, the accumulator result $\tilde{\eta}_{ij(t_{k-1})}$ is output when the broadcasting task of node i is triggered and is used to construct multi-hop virtual links for the non-neighboring nodes of node i. Taking Figure 1 as a representative case, by (28), the virtual link is constructed as

$$\begin{aligned}\eta_{il}(t_k) =&\alpha_{ij_1}(t_k)\eta_{j_1l_1}(t_k) + \alpha_{ij_1}(t_k)\eta_{j_1l_2}(t_k)\\ &+ \alpha_{ij_2}(t_k)\eta_{j_2l_2}(t_k) + \alpha_{ij_2}(t_k)\eta_{j_2l_3}(t_k)\end{aligned} \tag{34}$$

Although threshold processing [25] can filter out duplicate information and ensure that only the most concise virtual link exists between node i and non-adjacent node l, it may inadvertently discard potentially valid virtual links, thereby compromising convergence speed. In contrast, HCTSP circumvents this issue through real-time local memory computation; that is,

$$\begin{aligned}\eta_{il}(t_k) &= \alpha_{ij_1}(t_k)(\eta_{j_1l_1}(t_k) + \eta_{j_1l_2}(t_k))\\ &\quad + \alpha_{ij_2}(t_k)(\eta_{j_2l_2}(t_k) + \eta_{j_2l_3}(t_k))\\ &= \alpha_{ij_1}(t_k)\sum_{l\in N_{j_1}}\eta_{j_1l}(t_k) + \alpha_{ij_2}(t_k)\sum_{l\in N_{j_2}}\eta_{j_2l}(t_k)\end{aligned} \tag{35}$$

Furthermore, let

$$\tilde{\eta}_{jl}(t_k) = \sum_{l\in N_j}\eta_{jl}(t_k) \tag{36}$$

So $\tilde{\eta}_{il}(t_k)$ ultimately derives the following form; that is

$$\eta_{il}(t_k) = \sum_{j\in N_i}\alpha_{ij}(t_k)\tilde{\eta}_{jl}(t_k) \tag{37}$$

Through this optimization, sending node j compresses the additional synchronization information into one data unit instead of the $|N_j|$ neighboring data units required by the traditional full-message embedding scheme, thereby effectively reducing the packet size.

Subsequently, we focus on the parameter updating process to refine HCTSP. The update phase encompasses four key operations: (a) local timestamp recording upon message reception (lines 6–8), (b) calculation of relative clock parameters between adjacent nodes (lines 9–10), (c) calculation of relative clock parameters between the local node and its multi-hop nodes (lines 14–18), and (d) calculation of logical clock compensation values followed by logical clock synchronization (lines 20–22).

The critical step of the update phase is (c). Upon receiving timestamps and shared multi-hop information, the local node first derives the one-hop logical skew compensation $\eta * ij(t_k)$ and offset $\beta * ij(t_k)$ based on (26). Then, a high-order consensus algorithm is employed to estimate the multi-hop logical skew compensation $\tilde{\eta} * il(t_k)$ using both current-round and stored historical multi-hop information. Concurrently, a virtual link with offset $\beta * il = \beta_{ij}(t_k) + \bar{\beta} * jl(t * k - 1)$ is constructed using the shared parameter $\bar{\beta} * jl(t * k - 1)$ forwarded by neighboring nodes, while the local relative clock parameter table $\left\langle \tilde{\eta} * il(t_k) * i \in N_i^{'}, \beta_{il}(t_k) * i \in N * i^{'} \right\rangle$ is refreshed. Finally, at the preset update time,

each node calculates the two crucial compensation parameters $\hat{\alpha}_i(t_k)$ and $\hat{\beta}_i(t_k)$ for logical clock adjustment based on (30) and (33), thereby achieving logical clock synchronization.

$$L_i(t_k) = \hat{\alpha}_i(t_k)H_i(t_k) + \hat{\beta}_i(t_k) \tag{38}$$

4.4. Performance Comparison Analysis

4.4.1. Convergence Speed

Building on similar virtual-link frameworks [21,23,25], HCTSP further optimizes convergence speed by integrating high-order consensus theory. Specifically, it introduces more historical information from multi-hop nodes to accelerate the convergence speed of the average consensus algorithm. The detailed theoretical proof has been provided in Section 3.2.

4.4.2. Communication Overhead

Considering a two-hop virtual-link topology where each node has an average degree of d as a case study, in MACTS, each node performs an immediate relay-forwarding communication mechanism to establish virtual links. For each node, MACTS requires d^2 additional forwarding communication operations to establish contact with two-hop neighbors. In contrast, VTSP, PACTS, and HCTSP employ a single-hop communication model to save significant communication overhead. However, VTSP simply embeds the received timing messages of d neighboring nodes into the current synchronization packet. The packet size increases linearly with the number of neighbors, inevitably leading to increased communication cost and scalability limitations in dense IWSNs. For PACTS and HCTSP, these two protocols require only one aggregated estimation of all neighboring clock states rather than individual neighboring information, greatly reducing the redundant packet overhead in VTSP. Although PACTS and HCTSP adopt similar techniques to reduce packet overhead, the previous analysis has shown that HCTSP can converge faster, resulting in lower communication overhead at the same synchronization accuracy.

4.4.3. Synchronization Accuracy

Communication delay is a critical factor affecting the performance of CTSPs. In virtual-link-based synchronization protocols, multi-hop information is inevitably affected by delay uncertainty, which may degrade synchronization stability and accuracy. Unlike conventional methods that directly rely on current-round multi-hop estimates, HCTSP incorporates finite-order historical multi-hop information into the skew estimation process. The cross-time difference term $s_l(t_{k-m}) - s_i(t_k)$ reduces the dependence on a single delayed estimate and provides a smoothing effect against instantaneous delay jitter.

The synchronization error of HCTSP mainly consists of two components: the propagation error introduced by virtual multi-hop links and the historical-delay error associated with the finite-memory mechanism. The former generally increases with hop count, whereas the latter is constrained by the memory order m and update interval ΔT. Therefore, incorporating historical information does not imply unconditional suppression of delay-induced errors. Its effectiveness depends on the balance between delay accumulation and historical-delay compensation [36].

In this work, the delay jitter considered in Section 5.3 remains relatively small compared with the synchronization period. Under such conditions, the finite-memory mechanism can effectively smooth transient delay fluctuations, thereby reducing sensitivity to instantaneous jitter and improving synchronization accuracy. Nevertheless, when the hop count, memory order, or delay jitter becomes sufficiently large, the accumulated delay error may outweigh the benefits provided by historical information, leading to degraded synchronization performance.

Based on the above theoretical analysis and design, Table 1 presents a performance comparison between the proposed method and existing GTSP-based and ATS-based algorithms to intuitively demonstrate the superiority of the proposed high-order virtual-link mechanism. Detailed simulation results and corresponding analyses are provided in Section 5.

Table 1. Summary of GTSP-based and ATS-based time synchronization protocols.

Indicator	GTSP-Based Algorithm			ATS-Based Algorithm		
	GTSP	VTSP	HCTSP	ATS	PACTS	HATS
Core Mechanism	Single-hop	Virtual Link	High-order Virtual Link	Single-hop	Virtual Link	High-order Virtual Link
Convergence Speed	Slowest	Medium	Fastest	Slowest	Medium	Fastest
Communication Overhead	Low	High	Lowest	High	Low	Lowest
Synchronization Accuracy	High	Medium	Highest	High	Medium	Highest

Note: The performance comparison in Table 1 is conducted among algorithms of the same category.

5. Simulation Results

This section evaluates the performance of HCTSP through MATLAB R2018b simulations by comparing it with the traditional single-hop protocols GTSP [4] and ATS [5], as well as the two-hop protocols VTSP [25] and PACTS [23], under a typical two-hop virtual communication scenario. The initial logical clock parameters are set as $\hat{\alpha}_i(0) = 1$, $\hat{\beta} * i(0) = 0$, and $\alpha * ij(0) = 1$, while the broadcast period T_s of each node is set to 1 s. Given that sensor nodes typically exhibit a clock drift ranging from 10 to 100 ppm [37], we initialize the hardware clock skew as $\alpha_i(0) \in [0.9999, 1.0001]$ and the clock offset as $\beta_i(0) \in [0, 0.0002]$ s [7]. The crystal oscillator frequency is set to 32.768 kHz. Moreover, the number of virtual communication hops H is set to 2, and the memory order m in HCTSP is configured as 3. Note that ε is introduced in the theoretical analysis to establish the convergence condition. Since HCTSP follows the normalized GTSP framework, the practical parameter study mainly focuses on the memory order m.

Let the global logical clock and skew differences, i.e.,$\Delta_{g,L}[t_k] = \max\{|L_j[t_k] - L_i[t_k]|\}$ and $\Delta_{g,S}[t_k] = \max\{|\hat{\alpha}_j[t_k]\alpha_j[t_k] - \hat{\alpha}_i[t_k]\alpha_i[t_k]|\}$, $\forall\{i,j\} \in V$, denote the synchronization state. As the primary performance metric in this study, convergence speed is evaluated by measuring the time required for each protocol to reach a predefined synchronization error threshold. According to the industrial automation network standards ISA100.11a and WirelessHART [38], the typical synchronization accuracy requirement for wireless sensors in industrial production scenarios ranges from $\pm$90 to 100 µs. Therefore, we set the synchronization error threshold to 2 µs, which is a significantly stricter criterion than the industrial standard. When the system synchronization error satisfies $\Delta * g, L < 2$ µs, the network is considered to have achieved acceptable consensus convergence.

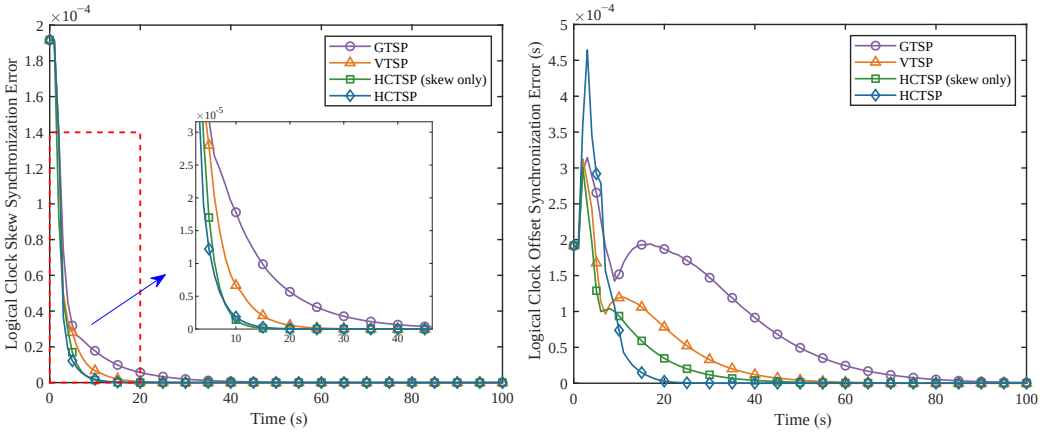
As the proposed HCTSP shares the same fundamental update principle as GTSP, the comparative analysis mainly focuses on GTSP-related protocols. Moreover, to demonstrate the general applicability of high-order consensus theory, we also investigate its application to ATS.

5.1. Convergence Performance Comparison

Figure 4 demonstrates the convergence characteristics of the logical clock under different protocols in a 6 × 6 grid topology. It should be noted that the vertical axis in Figure 4b represents the absolute skew value, which is dimensionless and therefore has no unit. Obviously, the proposed HCTSP exhibits faster convergence than GTSP and VTSP. To systematically evaluate the contribution of the joint skew-offset dual-channel virtual-link

mechanism in HCTSP, we conducted comparative experiments involving a skew-only virtual link and a joint skew-offset virtual link. The performance improvement achieved by the joint skew-offset mechanism can be clearly observed from the results.

It should be noted that all protocols exhibit an initial surge in offset error during the early stage of synchronization. The reason for this phenomenon is that the initial logical skew error amplifies the logical offset error. Meanwhile, since the logical offset update incorporates multi-hop historical synchronization data, which inherently introduces larger initial errors, this phenomenon is particularly pronounced in HCTSP (skew-only). However, as the logical skew gradually converges, the logical offset becomes the dominant factor in the iterative process, and the advantages of the joint skew-offset compensation mechanism become apparent.



(**a**) (**b**)

Figure 4. Comparison of global synchronization error on grid topology. (**a**) Logical clock skew sync-error; (**b**) Logical clock offset sync-error.

Next, we performed a systematic comparison of the convergence speed of different GTSP-based protocols under various network topologies. The experimental data presented in Figure 5 and Table 2 represent the statistical results obtained from 20 independent runs under ring, 5 × 10 grid, and random topologies. For the random topology, 50 nodes were randomly deployed within a 150 × 150 area with a communication radius of $\sqrt{30}$. It is well known that the convergence time of CTSPs is strongly related to the algebraic connectivity of the network topology. Therefore, in the sparse ring-topology environment, the convergence speed of all protocols is the slowest. In contrast, HCTSP can still achieve relatively fast convergence even in a ring topology.

Table 2. Statistical Data of Convergence Time under Different Topologies.

Topology	HCTSP	VTSP	GTSP	Improvement vs. VTSP
Ring	85 s	209 s	344 s	59.33%
Grid	60 s	128 s	256 s	53.13%
Random	48 s	81 s	169 s	40.74%

Furthermore, Figure 5 reveals a notable difference in convergence speed between the grid and random topologies despite their identical network scale (50 nodes). This difference mainly stems from network complexity, i.e., network density, which can be calculated as $\psi = 2|E|/(N(N-1))$ [39]. Therefore, network complexity is determined by both the number of nodes and the number of communication links. In other words, the denser the topology, the faster the convergence speed. To systematically investigate this relationship,

we further conducted a comprehensive experimental study under randomized topological conditions. The random network topology configurations are listed in Table 3.

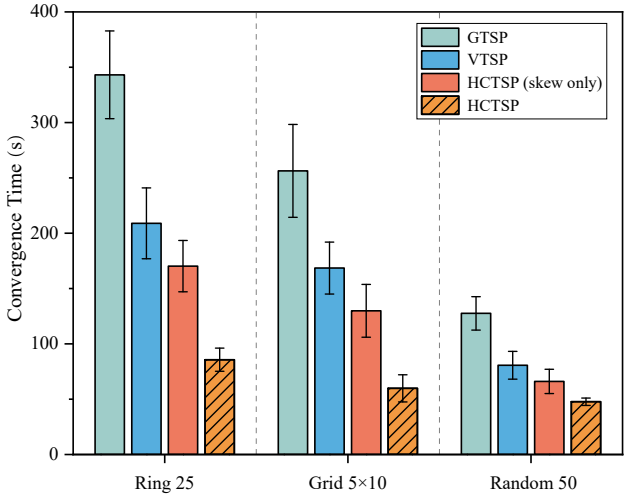


Figure 5. Statistical graph of convergence speed in different topologies.

Table 3. Random Topology Cases with A Node Radius of $\sqrt{12}$.

Topology	Area	Size	Edge Number	Network Density
Case 1	90 × 90	100 nodes	217	0.0438
Case 2	90 × 90	120 nodes	339	0.0475
Case 3	100 × 100	100 nodes	205	0.0414
Case 4	100 × 100	120 nodes	306	0.0429

It can be seen from the comparison results of various consensus protocols in Figure 6 that network complexity is a key factor limiting the convergence speed of average consensus protocols. Obviously, regardless of the protocol, Case 2, which has the highest network complexity, achieves the fastest convergence. Meanwhile, for a fixed network size, a larger network area may result in a larger network diameter D. This parameter directly affects the algebraic connectivity λ_2 of the network, i.e., $(1/D)^2 \leq \lambda_2 \leq (2d/D)$. Therefore, the larger the network diameter D, the slower the network convergence speed.

Specifically, although Cases 1 and 3 maintain the same number of nodes ($N = 100$), the larger sensing area in Case 3 results in a greater maximum hop count (i.e., network diameter) and fewer communication links than in Case 1, thereby leading to slower convergence speed.

The theoretical results in Section 3.2 indicate that a higher order of HCTSP leads to faster convergence. However, in topologies with high network density, excessively high orders may introduce more historical information, thereby increasing delay jitter and network congestion among nodes and ultimately affecting convergence performance. To determine the optimal order of HCTSP, we further conducted a large number of statistical experiments. The simulation parameters are consistent with those used in Figure 5.

The simulation results in Figure 7 demonstrate a clear positive correlation between the consensus order and convergence speed in sparse networks such as ring topologies. Nevertheless, the convergence speed does not continue to improve indefinitely as the order increases in grid and random dense topologies. When the order reaches 3 or higher, the convergence speed no longer improves and may even exhibit a slight decrease, particularly in random topologies. Therefore, it is recommended to use HCTSP with an optimal order of 3.

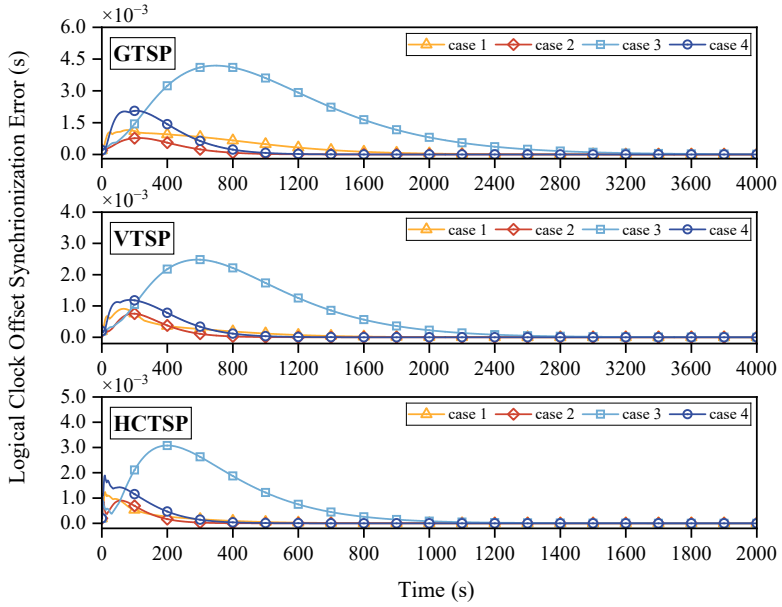


Figure 6. Convergence speed of GTSP—VTSP—HCTSP in Cases 1–4.

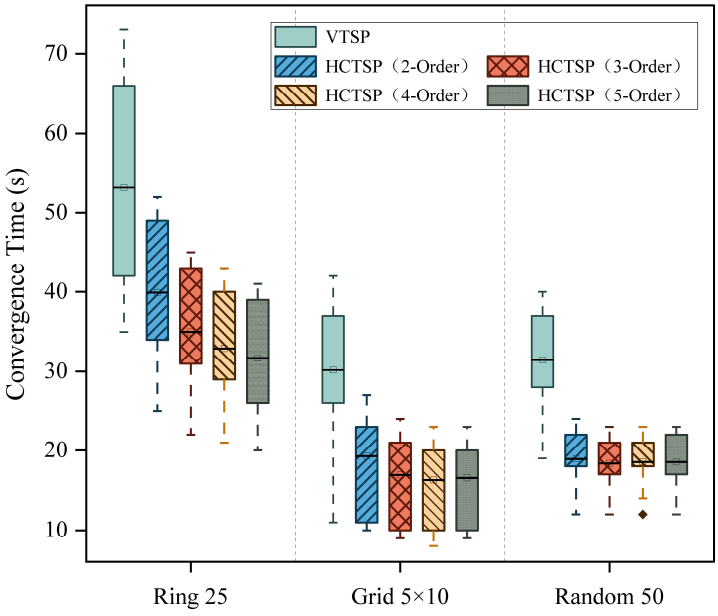


Figure 7. Comparison of orders under the same number of hops.

5.2. Communication Overhead Comparison

The power consumption of nodes in IWSNs mainly originates from the transceivers of wireless communication modules. Therefore, this section analyzes the communication overhead by comparing the number of packet exchanges.

In GTSP, VTSP, and HCTSP, each node with d_i neighbors exchanges $1 + d_i$ packets within a synchronization cycle, including one synchronization message broadcast and d_i receptions of timestamp information from neighboring nodes. Hence, the total number of packet exchanges required to complete one synchronization round across the entire network is $N_d = \sum_{i=1}^{n}(1 + d_i)$. Although these protocols have the same number of message exchanges in a single iteration, their convergence times differ. Here, we define the communication overhead ratio R as the performance metric, where $R = N_{P_E}/N_{T_M}$. N_{P_E} represents the number of packet exchanges required to reach the convergence threshold, and N_{T_M} denotes the total number of packet exchanges within the simulation time T_M.

In this section, T_M is set to 100 s. Clearly, if $R > 1$, it indicates that the protocol fails to converge within the predefined simulation time. Conversely, a lower value of R indicates lower communication overhead.

Figure 8 shows the communication overhead ratios of each protocol under different grid-topology sizes when achieving the predefined convergence threshold. Clearly, the proposed HCTSP achieves lower communication overhead. Meanwhile, as the network size increases, the communication overhead also exhibits an increasing trend.

As shown in Figure 9, this advantage becomes particularly evident in random topologies with dense node distributions. The packet size of HCTSP, optimized through the proposed redundant-information compression scheme, is reduced to approximately one-half of that of VTSP, while the final total packet volume is approximately one-third of VTSP. In addition, due to the slower convergence speed of GTSP, comprehensive evaluation shows that HCTSP also requires significantly lower total packet volume across the entire network and demonstrates superior performance in controlling communication overhead.

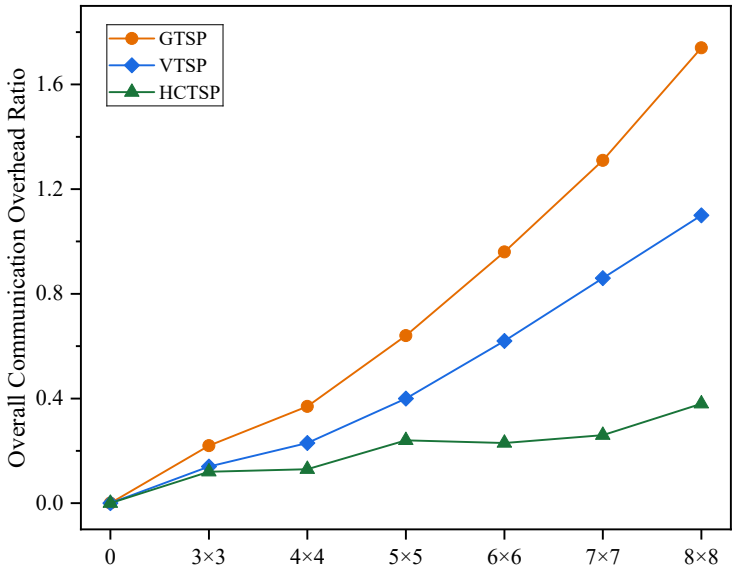


Figure 8. Total network overhead ratio of GTSP-based protocols.

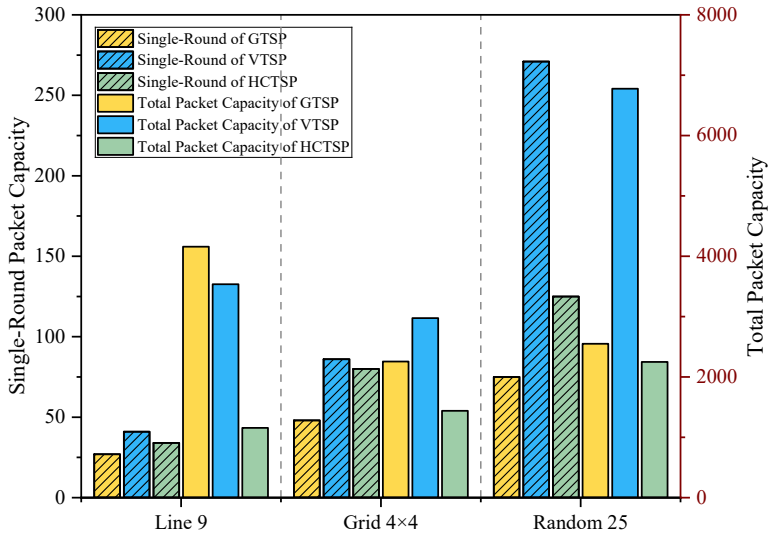


Figure 9. Comparison of packet overhead in GTSP-based protocols.

5.3. Synchronization Accuracy Against Communication Delay

Although MAC-layer timestamp technology can effectively mitigate most communication delays and enable precise timestamping, unavoidable delay jitter uncertainties still exist during multi-hop packet transmission. Therefore, the experiment employs a Gaussian distribution with a mean of 2.5×10^{-6} s and a variance of 6×10^{-13} s^2 to simulate delay jitter. This indicates that the non-negative delay varies randomly within the approximate range of [0, 5] μs and has a confidence interval of 99.97%.

In the delayed-network scenario with an 8 × 8 grid topology, Figure 10 shows the statistical synchronization-error results obtained from 20 independent runs. It can be seen that HCTSP still converges the fastest among all protocols under communication-delay conditions. However, the cumulative nature of delays and the lag in virtual-link shared information exacerbate the impact of delays on the estimation accuracy of synchronization parameters. Specifically, in single-hop protocols, delays affect only individual timestamp measurements, whereas in multi-hop networks, both transmission and processing delays accumulate at each hop. Additionally, in an m-hop virtual network, the shared information used by the local node at time t_k can be traced back to time t_{k-m}, thereby causing inaccuracies in the error estimation of both the local node and its neighboring nodes.

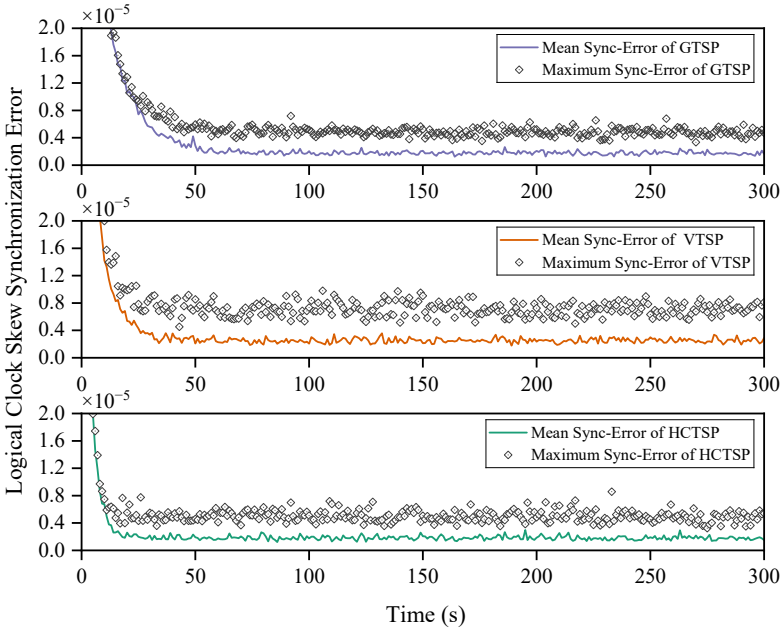


Figure 10. Logical clock skew synchronization errors under Gaussian delay jitter.

As illustrated in Figure 10, these factors result in lower synchronization accuracy for VTSP than for GTSP. Quantitative analysis reveals that while GTSP maintains a mean synchronization error of 2 and a maximum synchronization error of 5, the mean and maximum errors of VTSP can reach approximately 3.5 and 10, respectively. In contrast, HCTSP utilizes historical information from previous rounds of neighboring nodes in local memory, effectively mitigating the impact of cumulative delay on local parameter estimation in multi-hop scenarios and thereby limiting the mean and maximum synchronization errors to approximately 2.5 and 6.

To further evaluate the robustness of the proposed synchronization scheme under non-Gaussian communication delays, two additional delay models were considered, namely a heavy-tailed delay model and an exponential delay model. The heavy-tailed delay was generated using a Student's t-distribution with a degree of freedom of $\nu = 3$, while the exponential delay was modeled as $d = d_0 + \text{exprnd}(\beta)$, where $d_0 = 2.0 \times 10^{-6}$ s and

$\beta = 0.5 \times 10^{-6}$ s. The corresponding results are shown in Figures 11 and 12. Under these asymmetric delay conditions, HCTSP still exhibits the fastest convergence among the three protocols owing to the utilization of historical synchronization information. However, the bursty characteristics of non-Gaussian delays weaken the smoothing effect provided by the historical-memory mechanism. Consequently, although HCTSP still achieves slightly better synchronization accuracy than VTSP, the improvement becomes less pronounced than that observed under Gaussian delay conditions.

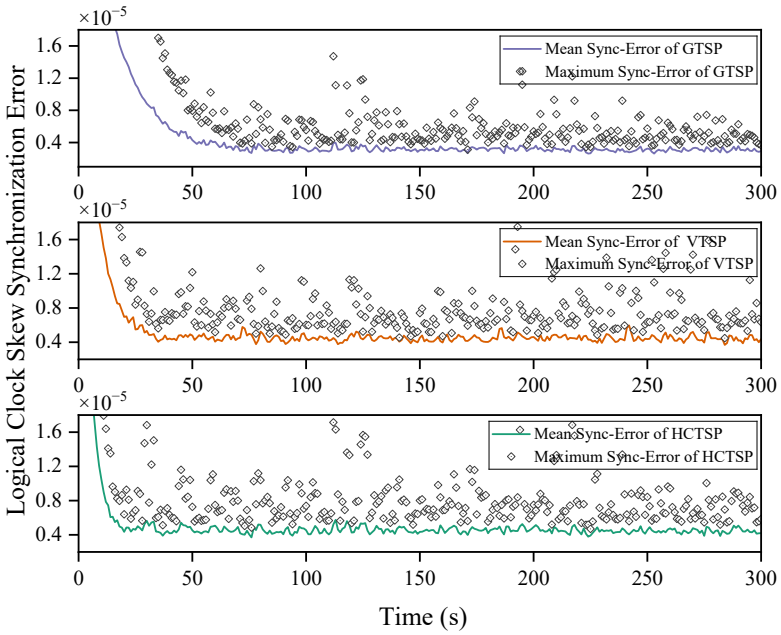


Figure 11. Logical clock skew synchronization errors under Heavy-tailed delay jitter.

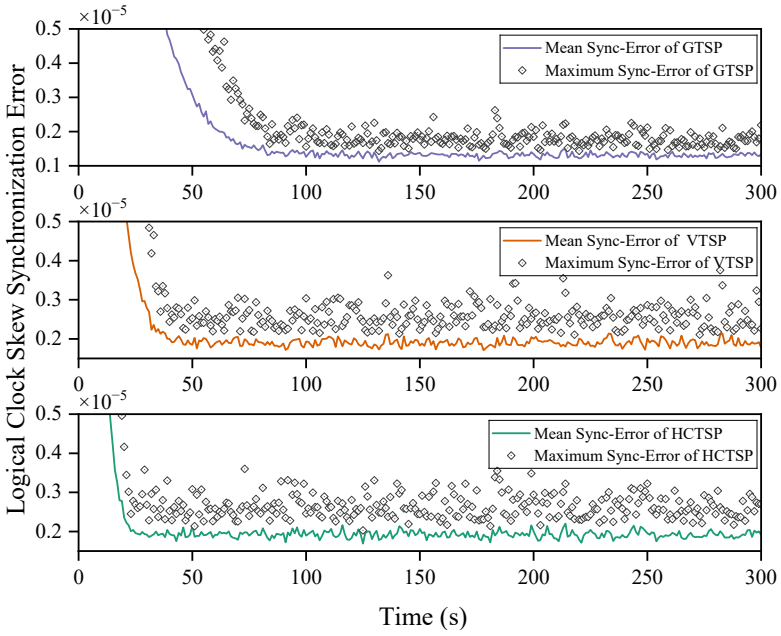


Figure 12. Logical clock skew synchronization errors under Exponential delay jitter.

Furthermore, to evaluate the statistical performance under different levels of Gaussian delay jitter, three standard deviations were considered, namely $\sigma = 5 \times 10^{-7}$ s, $\sigma = 1 \times 10^{-6}$ s, and $\sigma = 1.5 \times 10^{-6}$ s. For each setting, 20 independent simulation runs were performed with randomly generated initial clock offsets and communication delays while maintaining the same network topology and protocol parameters. To eliminate the

influence of the transient synchronization process, the first 50 iterations were excluded, and only the mean synchronization error during the steady-state phase was considered. Box plots were then constructed under the 68% and 95% confidence levels, as shown in Figures 13 and 14. As the standard deviation of the delay increases, the synchronization errors of all three protocols increase accordingly. GTSP maintains the lowest median synchronization error, while HCTSP achieves slightly better synchronization accuracy than VTSP while maintaining a faster convergence rate.

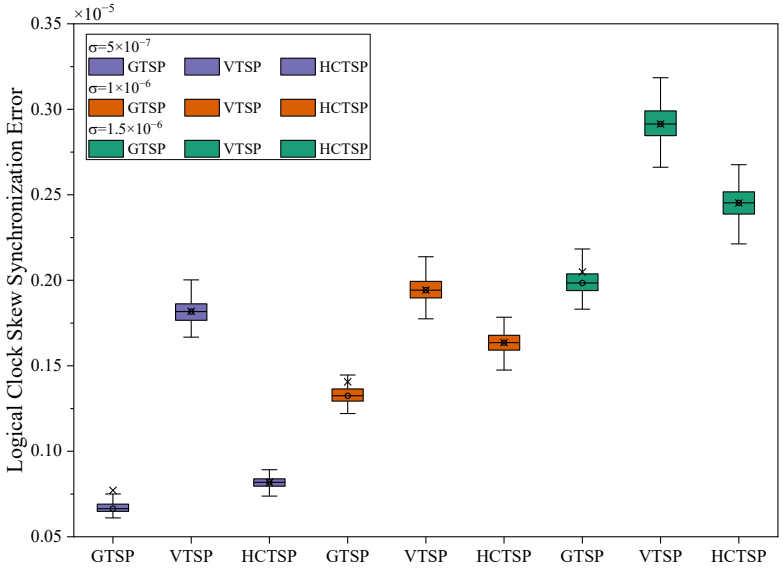


Figure 13. Synchronization errors with different standard deviations with a 68% confidence interval.

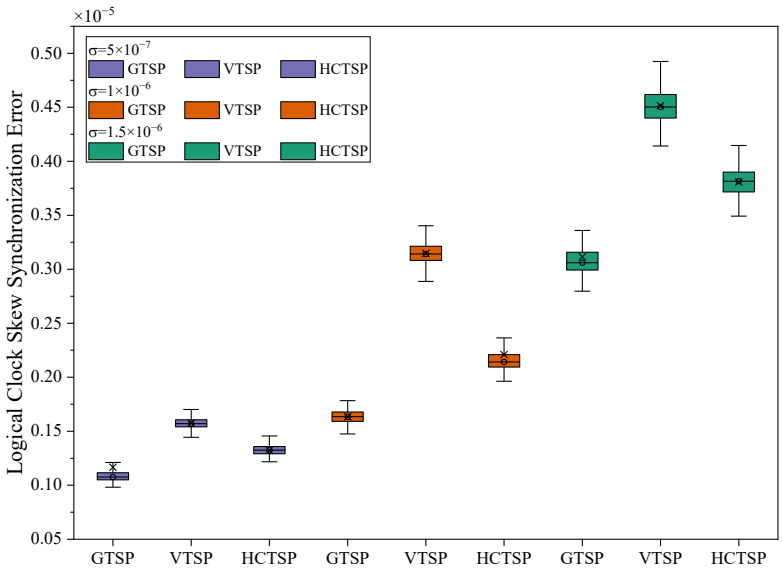


Figure 14. Synchronization errors with different standard deviations with a 95% confidence interval.

5.4. Universality Analysis

The ATS protocol [5,21,23] represents another branch of average-consensus-based CTSPs. Recent advancements have led to the development of PACTS [23], which also incorporates a virtual-link mechanism to enhance convergence speed. Unlike the synchronous update strategy of GTSP, the asynchronous instantaneous update mechanism of ATS provides a higher update frequency, resulting in faster convergence than GTSP. Therefore, to verify the universality of the proposed method, we further extend the experimental evaluation to include a high-order-consensus-based ATS protocol (HATS). The iterative

parameters ρ_η, ρ_v, and ρ_o in ATS and PACTS are set to 0.2, 0.5, and 0.5, respectively, while ρ_v in HATS is set to 0.25. The order of HATS is configured as 2. The remaining simulation parameters are consistent with those of HCTSP.

We first analyze the convergence performance of HATS in comparison with ATS and PACTS under various topologies, including a line topology with 30 nodes, a 5 × 10 grid topology, and random topologies consisting of 60 nodes randomly deployed within a 150 × 150 area with a communication radius of $\sqrt{30}$. The simulation results shown in Figures 15 and 16 demonstrate that the proposed high-order-consensus-based virtual-link mechanism can still maintain faster convergence in ATS.

Figure 17 further compares the communication-overhead ratio of each protocol when achieving the predefined convergence threshold under different topology scales. Clearly, compared with the single-hop ATS protocol and the multi-hop PACTS protocol, the proposed high-order-consensus-based virtual-link ATS protocol consistently achieves the desired synchronization error bound with the fewest packet exchanges.

Finally, we conducted the experiment shown in Figure 18 using the same delay-network model described in Section 5.3. It is well known that communication delay mainly affects the estimation of the relative clock skew α_{ij} and ultimately influences synchronization accuracy. For multi-hop virtual links (taking the two-hop case as an example), the relative clock skew estimation $\alpha_{il} = \alpha_{ij}\alpha_{jl}$ is subject to compounded errors introduced by two successive delays. Consequently, this leads to reduced robustness and lower synchronization accuracy compared with conventional single-hop ATS. The results presented in Figure 18 further support this analysis.

Although PACTS achieves faster convergence than ATS, quantitative analysis reveals that PACTS maintains a global mean relative-clock-skew error of approximately 9, whereas ATS stabilizes at approximately 5. In contrast, HATS can suppress the cumulative effect of multi-hop delays and ultimately stabilizes at approximately 6.5 while maintaining a faster convergence rate.

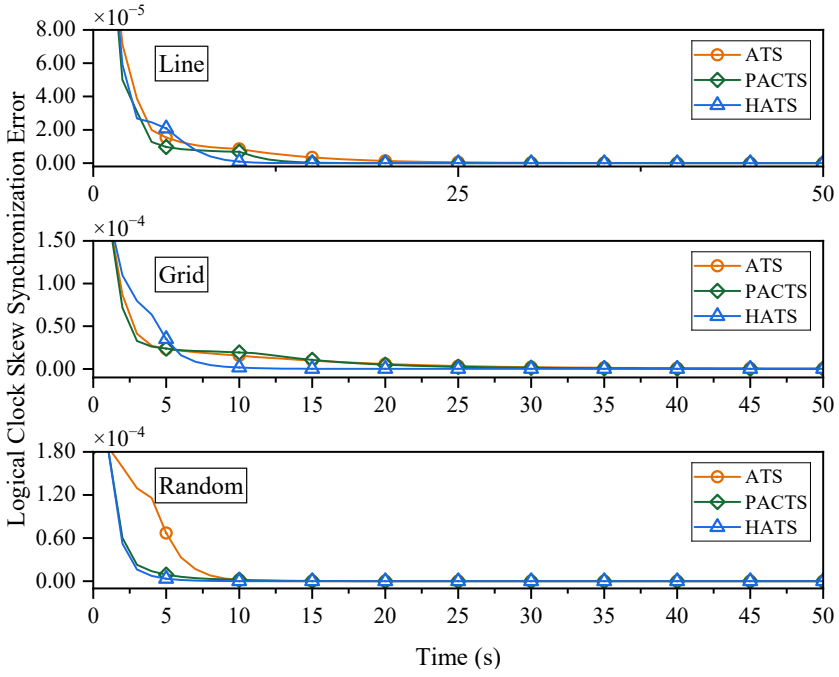


Figure 15. Logical clock skew sync-error in ATS-based protocols.

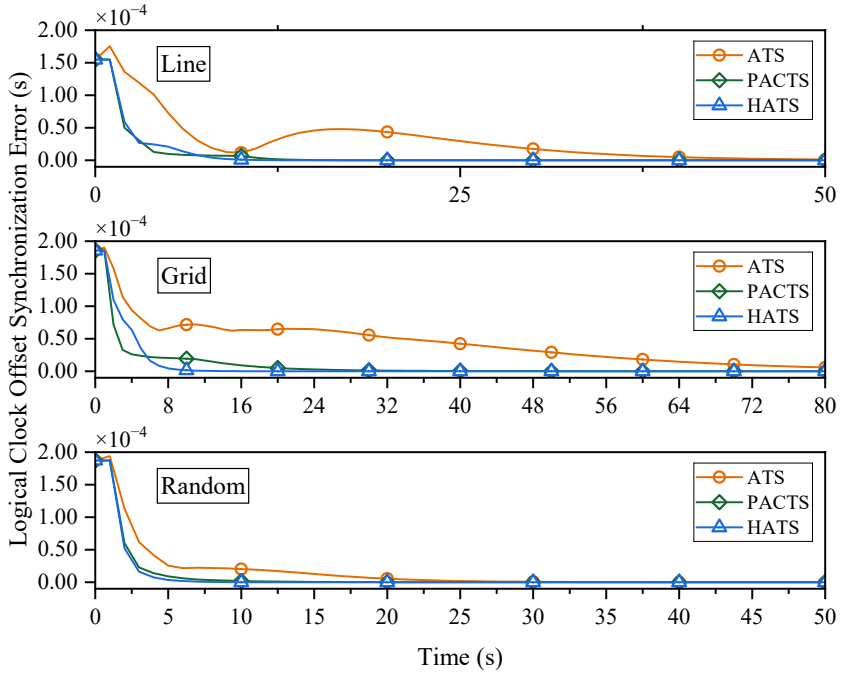


Figure 16. Logical clock offset sync-error in ATS-based protocols.

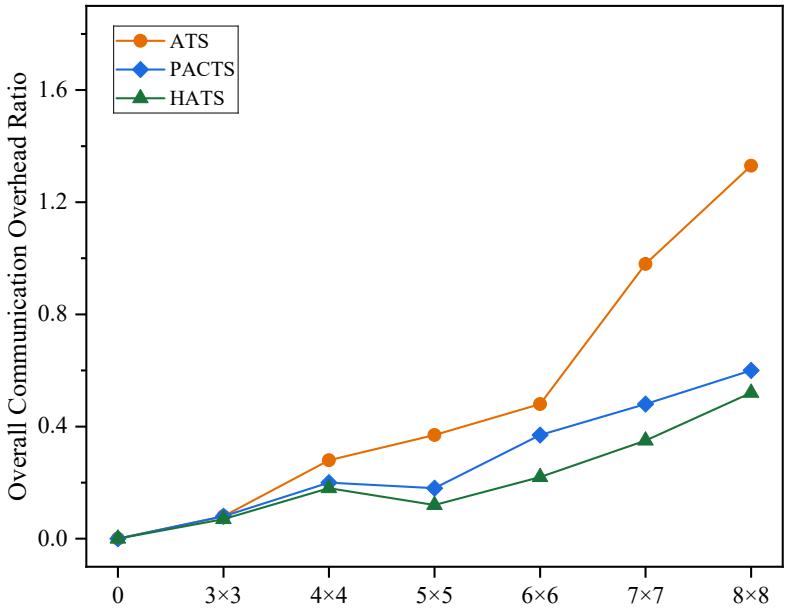


Figure 17. Communication overhead ratio of ATS-based protocols.

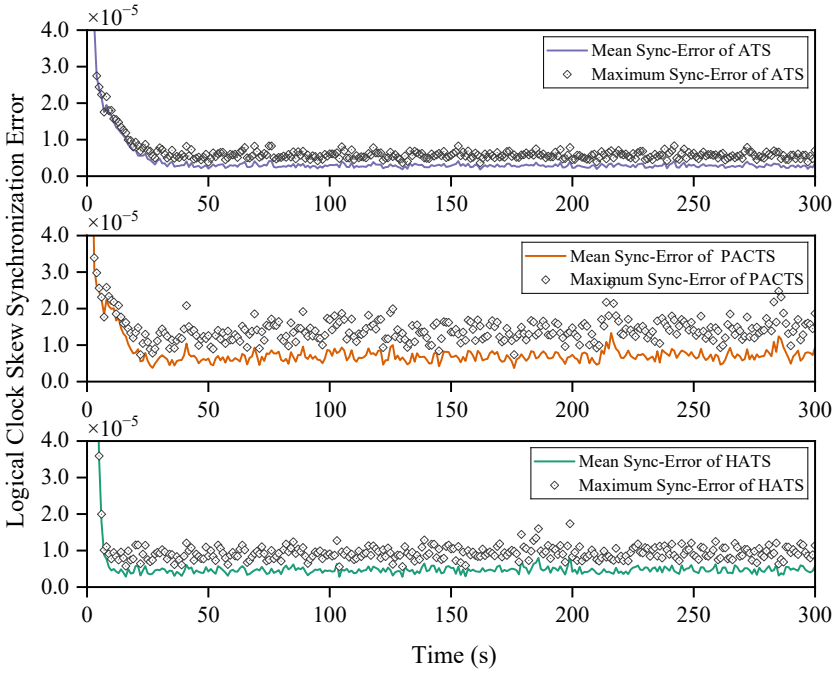


Figure 18. Logical clock skew synchronization errors under delay jitter.

6. Discussion

6.1. Complexity Analysis

The previous analysis shows that the accumulator-based HCTSP effectively reduces the transmission overhead of multi-hop virtual synchronization algorithms. However, for resource-constrained IWSN nodes, algorithm complexity is also a critical consideration. Therefore, this section further compares HCTSP with VTSP and PACTS in terms of computational and storage complexity.

6.1.1. Computational Complexity

The computational complexity mainly arises from one-hop and two-hop relative clock parameter estimation, as well as the updates of logical clock skew and logical clock offset compensation parameters. Let d_1 and d_2 denote the numbers of one-hop neighbors and two-hop virtual neighbors, respectively. The computational complexity of both VTSP and PACTS is $O(d_1 + d_2)$. In addition, both HCTSP and PACTS introduce aggregation operations to reduce the transmission overhead caused by redundant multi-hop information, thereby incurring additional local computation. Furthermore, since HCTSP exploits two-hop information in both logical clock skew and logical clock offset compensation updates, its practical computational workload per synchronization round is higher than that of VTSP and PACTS. However, these additional operations only increase the constant factor and do not affect the asymptotic complexity. Therefore, the overall computational complexity of HCTSP remains $O(d_1 + d_2)$, which is of the same linear order as VTSP and PACTS.

6.1.2. Storage Complexity

Storage complexity is determined by the amount of local memory required to maintain synchronization-related information. For VTSP and PACTS, each node only needs to store one-hop neighbor states and two-hop virtual-neighbor states, resulting in a storage complexity of $O(d_1 + d_2)$. In HCTSP, besides maintaining neighbor-state information, each node additionally employs a FIFO buffer to preserve high-order historical data for accumulator-based synchronization. Let M denote the FIFO memory order. The FIFO buffer introduces an additional storage overhead of $O(Md_1)$, yielding an overall storage

complexity of $O(d_1 + d_2 + Md_1)$. Since M is configured as a small fixed constant in practical implementations, the storage complexity of HCTSP remains linear with respect to network size, and the additional memory requirement introduced by HCTSP is limited. Overall, the moderate increase in computational and storage overhead represents a reasonable trade-off for achieving faster convergence and lower transmission overhead. The computational and storage complexity comparison of VTSP, PACTS, and HCTSP is summarized in Table 4.

Table 4. Computational and storage complexity comparison.

Protocol	Computational Complexity	Storage Complexity
VTSP	$O(d_1 + d_2)$	$O(d_1 + d_2)$
PACTS	$O(d_1 + d_2)$	$O(d_1 + d_2)$
HCTSP	$O(d_1 + d_2)$	$O(d_1 + d_2 + Md_1)$

6.2. Limitations

Although the proposed HCTSP demonstrates promising synchronization performance under the considered scenarios, several issues remain to be further investigated. Future work will focus on the theoretical analysis of high-order virtual links in multi-hop ($\geq$3) scenarios under complex network conditions. In particular, adaptive memory-order selection mechanisms that consider network topology, communication quality, and delay accumulation will be investigated to improve the adaptability of the proposed algorithm in dynamic environments. Moreover, practical factors such as packet loss, node failures, dynamic node participation, burst latency, and temperature-dependent clock drift will be incorporated to further evaluate and enhance the robustness of the synchronization framework. Finally, the performance of the proposed high-order consensus synchronization algorithm will be validated on physical platforms to facilitate its practical deployment.

7. Conclusions

This paper has presented a novel high-order consensus time synchronization protocol (HCTSP) that leverages a multi-hop consensus algorithm to address the slow convergence issue in CTSPs. The key contribution of this work is the demonstration that incorporating historical information from nonadjacent nodes can improve clock-skew estimation performance, thereby enhancing convergence efficiency. In addition, the virtual-link construction method based on a single-hop communication model and integrated neighbor-information transmission effectively reduces communication packet overhead. Meanwhile, the distinctive integration of historical and current timestamp information provides enhanced robustness against delay accumulation in multi-hop virtual communication scenarios. Extensive simulation results demonstrate that HCTSP outperforms existing multi-hop consensus protocols in terms of convergence speed, communication overhead, and synchronization accuracy, while also exhibiting good general applicability.

Author Contributions: Writing—original draft preparation, X.Y.; writing—review and editing, Z.W.; formal analysis, X.Y.; investigation, Z.Z.; funding acquisition, Z.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant 62002140; and in part by the Natural Science Foundation of Jiangsu Province under Grant BK20200887; and in part by the Pond Ecological Farming Green Intelligent Equipment Technology and Integrated Management and Control System Demonstration under Grant JSYTH03.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Mazo, M.; Tabuada, P. Decentralized event-triggered control over wireless sensor/actuator networks. *IEEE Trans. Autom. Control* **2011**, *56*, 2456–2461. [CrossRef]
2. Moussa, B.; Debbabi, M.; Assi, C. Security assessment of time synchronization mechanisms for the smart grid. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1952–1973. [CrossRef]
3. He, J.; Cheng, P.; Shi, L.; Chen, J. Time synchronization in WSNs: A maximum-value-based consensus approach. *IEEE Trans. Autom. Control* **2014**, *59*, 7882–7887. [CrossRef]
4. Sommer, P.; Wattenhofer, R. Gradient clock synchronization in wireless sensor networks. In *2009 International Conference on Information Processing in Sensor Networks*; IEEE: Piscataway, NJ, USA, 2009; pp. 37–48.
5. Schenato, L.; Fiorentin, F. Average timesynch: A consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica* **2011**, *47*, 1878–1886. [CrossRef]
6. Wang, B.; Lv, X. Drift compensation of time synchronization in WSNs with random delays: A new threshold based design. *Automatica* **2025**, *173*, 112088. [CrossRef]
7. Wang, Z.; Yong, T.; Song, X.; Sun, D. Maximum likelihood estimation of relative clock skew for distributed time synchronization in industrial wireless sensor networks. *IEEE Sens. J.* **2023**, *23*, 31359–31367. [CrossRef]
8. Phan, L.A.; Kim, T.; Kim, T. Robust neighbor-aware time synchronization protocol for wireless sensor networks in dynamic and hostile environments. *IEEE Internet Things J.* **2021**, *8*, 1934–1945. [CrossRef]
9. Lee, J.-S.; Jiang, H.-T. An extended hierarchical clustering approach to energy-harvesting mobile wireless sensor networks. *IEEE Internet Things J.* **2021**, *8*, 7105–7114. [CrossRef]
10. Saiah, A.; Benzaid, C.; Younis, M.; Badache, N. SRST: A secure and resilient synchronization of time for WSNs in IoT applications. *Ad. Hoc. Netw.* **2025**, *169*, 103749. [CrossRef]
11. Jin, Z.; Murray, R.M. Multi-hop relay protocols for fast consensus seeking. In Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA, 13–15 December 2006; pp. 1001–1006.
12. Shi, F.; Tuo, X.; Yang, S.X.; Lu, J.; Li, H. Rapid-flooding time synchronization for large-scale wireless sensor networks. *IEEE Trans. Ind. Informat.* **2020**, *16*, 1581–1590. [CrossRef]
13. Wu, J.; Zhang, L.; Bai, Y.; Sun, Y. Cluster-based consensus time synchronization for wireless sensor networks. *IEEE Sens. J.* **2015**, *15*, 1404–1413. [CrossRef]
14. Wang, H.; Xiong, D.; Chen, L.; Wang, P. A consensus-based time synchronization scheme with low overhead for clustered wireless sensor networks. *IEEE Signal Process. Lett.* **2018**, *25*, 1206–1210. [CrossRef]
15. Liu, X.; Zhu, W.; Chen, B.; Wei, M. Time synchronization algorithm for wireless sensor networks based on K-neighboring topology. *Phys. Scr.* **2026**, *101*, 105205. [CrossRef]
16. Zhao, D.; An, Z.; Xu, Y. Time synchronization in wireless sensor networks using max and average consensus protocol. *Int. J. Distrib. Sens. Netw.* **2013**, *9*, 933–940.
17. Phan, L.-A.; Kim, T. Hybrid time synchronization protocol for large-scale wireless sensor networks. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 10423–10433. [CrossRef]
18. Pei, J.; Frascolla, V.; Al-Dulaimi, A.; Liu, W.; Aldhyani, T.H.; Bashir, A.K.; Mumtaz, S. Distributed large models training optimization with real-time wireless channel feedback. *IEEE J. Sel. Areas Commun.* **2026**, *44*, 2231–2243. [CrossRef]
19. Olfati-Saber, R.; Fax, J.A.; Murray, R.M. Consensus and cooperation in networked multi-agent systems. *Proc. IEEE* **2007**, *95*, 215–233. [CrossRef]
20. Panigrahi, N.; Khilar, P.M. Multi-hop consensus time synchronization algorithm for sparse wireless sensor networks. *Ad. Hoc. Netw.* **2017**, *61*, 124–138. [CrossRef]
21. Shi, F.; Tuo, X.; Ran, L.; Ren, Z.; Yang, S.X. Fast convergence time synchronization in wireless sensor networks based on average consensus. *IEEE Trans. Ind. Informat.* **2020**, *16*, 1120–1129. [CrossRef]
22. Wang, H.; Zou, Y.; Liu, X.; Meng, Z. A rapid time synchronization scheme using virtual links and maximum consensus for wireless sensor networks. *IEEE Internet Things J.* **2025**, *12*, 3318–3329. [CrossRef]
23. Shi, F.; Yang, S.X.; Mukherjee, M.; Jiang, H.; da Costa, D.B.; Wong, W.-K. Parameter-sharing-based average-consensus time synchronization in IoT networks. *IEEE Internet Things J.* **2022**, *10*, 8215–8227. [CrossRef]

24. Duan, H.; Shi, F.; Wang, S.; Cui, Q. A time synchronization hop-count-control algorithm based on synchronization error convergence probability estimation. *Electronics* **2025**, *14*, 2086. [CrossRef]
25. Phan, L.A.; Kim, T. Fast consensus-based time synchronization protocol using virtual topology for wireless sensor networks. *IEEE Internet Things J.* **2021**, *8*, 7485–7496. [CrossRef]
26. Wang, H.; Zou, Y.; Liu, X.; Li, M. A fast convergence scheme for distributed consensus time synchronization using multihop virtual links in industrial wireless sensor networks. *IEEE Sens. J.* **2024**, *24*, 20009–20017. [CrossRef]
27. Liu, J.; Morse, A.S. Accelerated linear iterations for distributed averaging. *Annu. Rev. Control* **2011**, *35*, 160–165. [CrossRef]
28. Oreshkin, B.N.; Coates, M.J.; Rabbat, M.G. Optimization and analysis of distributed averaging with short node memory. *IEEE Trans. Signal Process.* **2010**, *58*, 2850–2865. [CrossRef]
29. Ghadimi, E.; Shames, I.; Johansson, M. Multi-step gradient methods for networked optimization. *IEEE Trans. Signal Process.* **2013**, *61*, 5417–5429. [CrossRef]
30. Yi, J. W.; Chai, L.; Zhang, J.X. Convergence rate of accelerated average consensus with local node memory: Optimization and analytic solutions. *IEEE Trans. Autom. Control* **2023**, *68*, 7254–7269. [CrossRef]
31. Xiong, G.; Kishore, S. Discrete-time second-order distributed consensus time synchronization algorithm for wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.* **2009**, *2009*, 623537. [CrossRef]
32. Olfati-Saber, R.; Murray, R.M. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Autom. Control* **2004**, *49*, 1520–1533. [CrossRef]
33. Meyer, C.D. *Matrix Analysis and Applied Linear Algebra*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2001.
34. Xiao, L.; Boyd, S. Fast linear iterations for distributed averaging. *Syst. Control Lett.* **2004**, *53*, 65–78. [CrossRef]
35. Olshevsky, A.; Tsitsiklis, J.N. Convergence speed in distributed consensus and averaging. *SIAM J. Control Optim.* **2009**, *48*, 33–55. [CrossRef]
36. Huang, M.; Yi, J.; Chai, L. Memory-based accelerated consensus for second-order multi-agent systems with delay. *SIAM J. Control Optim.* **2024**, *10*, 574–583. [CrossRef]
37. Choi, B.J.; Liang, H.; Shen, X.S.; Zhuang, W. DCS: Distributed asynchronous clock synchronization in delay tolerant networks. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 491–504. [CrossRef]
38. Shi, F.; Yang, S.X.; Tuo, X.; Ran, L.; Huang, Y. A novel rapid flooding approach with real-time delay compensation for wireless-sensor network time synchronization. *IEEE Trans. Cybern.* **2022**, *52*, 1415–1428. [CrossRef]
39. Arrao-Vargas, F.; Konstantinou, G. Longitudinal power systems for modern and future grid studies: A graph theory analysis. *Sustain. Energy Grids Netw.* **2023**, *35*, 101132. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.