International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(4s) 2026 pp. 55-70
https://doi.org/10.70917/ijcisim-2026-2493
Article

A DELAY AWARE Q-LEARNING APPROACH FOR HYBRID CLOCK SYNCHRONIZATION PROTOCOL IN WSNS

M. Muthumalathi[1], P. B. Pankajavalli[1 *], R. Suresh[2]

[1]Department of Computer Science, Bharathiar University, Coimbatore 641046, India.
[2]Department of Mathematics, University of Technology and Applied Sciences, Ibri 466,516, Oman
*Corresponding author. E-mail: pankajavalli@buc.edu.in

ABSTRACT: Achieving reliable and energy-efficient clock synchronization in Wireless Sensor Networks (WSNs) is a critical challenge due to clock drift, communication delays, and node failures. In particular, synchronization protocols such as reference and consensus-based techniques, need trade-offs between convergence speed, synchronization accuracy, and robustness. To address these issues, this paper proposes a unified synchronization framework called Q-learning-based Hybrid Time Synchronization Protocol (QHTSP), which combines the strengths of Partial Timestamp Synchronization (PTS) and hybrid synchronization methodologies. The QHTSP protocol uses Q-learning to dynamically switch between reference and consensus-based modes which results in fast convergence and fault tolerance. It also includes Gaussian and exponential stochastic delays, which provide reliable estimations for clock skew and offset using Maximum Likelihood Estimation and Best Linear Unbiased Estimator, respectively. Specifically, the proposed protocol offers an accurate synchronization network situations like queuing-induced and multi-source interference delays. By integrating statistical signal processing with learning-based dynamic adaptation, QHTSP delivers a scalable and fault-tolerant approach to synchronize the nodes. Simulation results show that QHTSP provides fast and energy-efficient reliable synchronization making it ideal for large-scale WSNs deployed in IoT applications. Simulations of various network topologies with nodes ranging from 100 to 1000 reveal that QHTSP outperforms some existing protocols such as Hybrid Time Synchronization Protocol (HTSP), Flooding Time Synchronization Protocol (FTSP), and Enhanced-Flooding Time Synchronization Protocol (E-FTSP) in terms of convergence time, synchronization error, and resilience to node failures.
Keywords: Clock Synchronization; Reinforcement Learning; Stochastic Delay; Network Topologies.

1. Introduction

In recent years, numerous protocols have been developed to tackle synchronization challenges of Wireless Sensor Networks (WSNs) [1] [2]. Synchronization is achieved by developing control protocols and optimizing clock parameters to ensure consistent timing throughout the network [3]. Synchronizing methods is mainly used to efficiently communicate information from one sensor node to other sensor nodes, without packet loss and delay [4]. Clock synchronization is to provide common synchronization time in the entire sensor network environment. Clock synchronization is used to synchronize all the sensor nodes local time to global clock time [5]. Thus, synchronization plays a critical role in enabling reliable time coordination across the network, supporting key functions such as event detection, data fusion, and energy-efficient sleep-wake scheduling [6]. Clock synchronization is a fundamental mechanism required for regular distribution, sensing, and for managing large-scale sensor networks [7]. Clock synchronization in a distributed system, often referred to as clock recovery, involves frequent synchronization in serial communications [8]. Traditional synchronization techniques employed in Industrial Internet of Things (IIoT) environments, like Precision Time Protocol (PTP) and Flooding Time Synchronization Protocol (FTSP), typically entail regular calibration of the target clock to minimize observed deviations from a time reference [9]. Due to inherent clock errors and varying clock rates among devices, communication among clocks relies on an asymmetrical broadcast protocol [10]. The primary synchronization process involves a node estimating its clock skew relative to its neighboring nodes through information exchange with them. Then,

the node updates its clock compensation parameters using a consensus algorithm to align the clocks of all nodes to a common clock [11]. A clock synchronization protocols can be developed using either a reference or a consensus-based approach. Both methods have their respective benefits and drawbacks when applied to large-scale sensor networks. Reference-based time synchronization protocols achieves rapid convergence since the clock information is dispersed from the reference node covering the whole network in a few rounds of synchronization [12]. In consensus-based clock synchronization, each node adjusts its local clock by iteratively calculating a weighted average of its neighboring nodes clocks, where optimizing the weights helps to minimize the total number of synchronization iterations. [13] [14]. Specifically, a network's ability improves synchronization when it is strongly connected. However, extending communication range or increasing network density can lead to significant issues such as increased power consumption and heightened synchronization errors among nodes in WSNs. Further, the synchronization accuracy and efficiency in utilizing Reinforcement Learning (RL) algorithms such as the Q-learning algorithm and Markov chain models have been introduced [15]. Q-learning proves to be a valuable approach for addressing agent actions under various complex conditions. However, as the number of object states increases, the table matrix representing the state-action set in Q-learning becomes excessively large for storage and efficient search [16] [17].

Traditional synchronization protocols frequently face challenges due to the dynamic and changing conditions such as node mobility, variable communication delays, topology changes, energy constraints, and intermittent node failures common in WSNs. In particular, the traditional methods may not efficiently manage packet transmission and clock synchronization in environments due to frequent topological and traffic changes. Q-learning, a reinforcement learning technique, is effective in such situations as it enables nodes to choose optimal actions to maximize cumulative rewards by iterating with their environment. This adaptability makes Q-learning well-suited for managing WSNs synchronization by dynamically adjusting to network traffic changes and node conditions.

Motivated by the above considerations, this work presents the Q-learning-based Hybrid Time Synchronization Protocol (QHTSP), which incorporates Q-learning process to improve clock accuracy, energy efficiency, and network resilience in WSNs. The primary contributions of this paper are as follows:

1. Q-learning-based Hybrid Time Synchronization Protocol (QHTSP) is proposed to enhance distributed clock synchronization and in which the synchronization modes are adjusted based on current network conditions.
2. Specifically, QHTSP surpasses traditional synchronization methods in minimizing synchronization error, energy usage, and packet transmission requirements.
3. QHTSP reduces energy consumption and network overhead with the incorporation of timestamp mechanism. Specifically, QHTSP requires fewer updates and limiting dependency on full timestamp exchanges.
4. This protocol enables faster convergence to accurate clock synchronization of states, providing a scalable solution for large and dynamic WSNs deployments.
5. QHTSP uses Q-learning for dynamic mode switching, which enables it to adapt for changing network conditions and maintain synchronization even in the presence of node failures.

The structure of the paper is as follows: Section II reviews existing synchronization protocols and the basics of Q-learning. Section III details the proposed QHTSP methodology, including node deployment, Q-learning-based mode switching, delay modeling, and reward-based action selection for synchronization. Section IV presents simulation results by analyzing metrics such as synchronization accuracy, energy efficiency, convergence speed, and fault tolerance. Finally, Section V concludes with insights into QHTSP's impact and future work in synchronization for WSNs.

2. RELATED WORKS

The clock synchronization in WSNs has explored several techniques including traditional approaches relying on full timestamp data for accurate estimation of clock skew and offset [18]. However, efforts to conserve energy, reduce communication overhead, or bolster security often led to the omission of timestamps during synchronization, resulting in insufficient time data for accurate alignment. Average-based consensus time synchronization protocols have been widely adopted in WSNs due to their resilience against single-node failures [19]. However, they face limitations when encountering unsynchronized nodes, particularly in dynamic or hostile environments. Phan et al [20] has examined the Neighbor-aware Time Synchronization Protocol (NTSP) for WSNs, where NTSP incorporates a neighbor-aware approach, allowing each node to make informed decisions based on the synchronization status of its neighbors, thereby ensuring more robust synchronization.

While conventional methods exchange parameters to achieve synchronization through static consensus. However, the variations in clock oscillators is an issue in the assumption of constancy. To address this issue Bathelt et al [21] has proposed consensus-based time synchronization, aiming to overcome the limitations of existing algorithms that rely on a constant linear model for local clocks. In particular, the paper introduces dynamic consensus by incorporating local signals alongside feedback from neighboring agents. The proposed conceptual algorithm for time synchronization operates without depending on constant parameters.

Existing event-based consensus clock synchronization schemes in industrial wireless sensor networks overlook communication delays which limits the application. In [22], the author examined this issue by proposing a scheme that considers delays, leveraging a new low-pass filter and event-triggered mechanism. Through convergence proof and simulation, the scheme demonstrates efficient synchronization while reducing communication overhead, essential for the limited resources of sensor nodes in realistic scenarios.

In [23], the virtual topology-based time synchronization protocol (VTSP) is introduced for addressing the issue of slow convergence in average-based consensus time synchronization protocols, especially in large or sparse wireless sensor networks. In particular by operating on a virtual topology with enhanced algebraic connectivity, VTSP accelerates convergence without altering the physical network. By establishing virtual links between nodes and their two-hop neighbors, VTSP minimizes redundancy and enhances convergence speed.

Comparing consensus and reference-based time synchronization protocols in WSNs, the study identifies their strengths and weaknesses. To overcome these limitations, in [24], the author introduces a hybrid time synchronization protocol (HTSP) that combines reference and consensus-based approaches. HTSP employs a temporary reference node for fast convergence and seamlessly transitions to average-based consensus during regular operation to manage node failures. The fully distributed protocol eliminates the need for a leader-election mechanism and demonstrates superior performance across various network topologies and scales.

In [25], a one-way time synchronization scheme was proposed for energy-efficient LoRa networks based on a reverse asymmetric framework. To accommodate the communication asymmetry in LoRa systems, where end nodes primarily transmit to gateways while conserving energy, the scheme minimizes synchronization delay through low-layer timestamping and lightweight time translation mechanisms. Real-world experiments demonstrate microsecond-level synchronization accuracy between gateways and end nodes, enabling practical deployment in multi-hop LoRa networks.

In [26], a two-way synchronization protocol achieved microsecond-level accuracy in LoRa networks by addressing timing uncertainties from physical-layer delays through bidirectional message exchanges. While effective, the method significantly increases energy consumption due to the overhead of frequent synchronization messages, making it less suitable for energy-constrained LoRa end nodes.

Motivated by these advancements, the proposed QHTSP integrates partial timestamp estimation with Q-learning-based adaptive mode switching to enhance synchronization accuracy, energy efficiency, and fault tolerance across diverse network conditions.

3. METHODOLOGY

The QHTSP is a unified framework that combines Partial Timestamp Synchronization (PTS) and adaptive learning via Q-learning concept for synchronization in WSNs. This approach enables accurate estimation of clock skew and offset across network nodes, under stochastic delay conditions. QHTSP leverages both Gaussian and exponential delay models for typical real-world network conditions for improving synchronization with robustness and scalability.

The switching mechanism in QHTSP is dynamically learned, not statically defined. Nodes adjust synchronization modes based on accumulated environmental feedback, optimizing for cumulative rewards across varying traffic, delay, and node failure scenarios. This adaptability offers significant resilience compared to fixed hybrid methods.

Algorithm: Q-learning-based Hybrid Time Synchronization Protocol (QHTSP)
Inputs:
- N: Total number of nodes
- T1: Initial timestamp at sender

- Delay Model: Gaussian or Exponential
- Q-learning Parameters: α (learning rate), γ (discount factor)
- Thresholds: syncThld (synchronization threshold), stableThld (stability threshold)
- MAX_SYNCED, MAX_STABLE: Synchronization stability thresholds

Initialize:

1. For each node $i \in N$:
   - Set Q-table to zeros
   - Set learning rate $\alpha = 0.1$ and discount factor $\gamma = 0.9$
   - Set skew ψ_i, offset θ_i
   - Initialize mode = 'consensus'
   - Initialize counters: stableCnt, syncedCnt, unsyncedCnt to zero

Main Loop:

2. Repeat for each synchronization round until network convergence:
   - For each node $i \in N$:
     a. Check and update synchronization status:
       - Calculate max_offset between node i and neighbors
       - If max_offset is greater than syncThld:
         - Increment unsyncedCnt
         - If unsyncedCnt is greater than MAX_SYNCED:
           - Set isSynced to false and reset stable time
       - Else:
         - Reset unsyncedCnt

     b. Update Q-learning values:
       - If in consensus mode and stable:
         - Calculate relative skew from neighbors
         - Update Q(s, a) using Q-learning update based on rewards for synchronization accuracy and energy efficiency
       - If mode == 'reference' and stability achieved:
         - Increment syncedCnt
         - If syncedCnt > MAX_STABLE:
           - Switch mode to 'consensus'

     c. Select synchronization action based on Q-values:

- If Q-value indicates reference-based mode:
  - Set mode = 'reference' and follow the reference node
- If Q-value supports consensus-based mode:
  - Set mode = 'consensus' and synchronize with neighbors

d. Broadcast timing message with updated skew and offset:
- If mode == 'reference', include stable time in the message
- If mode == 'consensus', broadcast basic timing data

End Loop

Return: Final skew ψ and offset θ for each node

The QHTSP protocol can be summarized into five key phases: (1) Node initialization and Q-table setup; (2) Monitoring synchronization error with neighbor nodes; (3) Updating Q-values based on synchronization accuracy and energy consumption rewards; (4) Selecting optimal synchronization mode (consensus or reference) based on learned policy; (5) Broadcasting updated skew and offset estimates. This structured protocol enables QHTSP to dynamically maintain synchronization even under varying network conditions.

Each node initializes its Q-table, skew ψ_i, offset θ_i, and synchronization mode to 'consensus'. The nodes are also assigned with Q-learning parameters like the learning rate α, discount factor γ, and synchronization thresholds (syncThld, stableThld). This initial setup prepares nodes for adaptive synchronization based on Q-learning decisions.

During each round, nodes assess to synchronize by calculating the maximum offset from neighbors. If the max offset exceeds syncThld, the node enters an unsynchronized state. Counters for synchronization status (e.g.,unsyncedCnt) help nodes to determine when they need to enter a resynchronization process or remain stable. In each round, nodes use Q-learning to refine synchronization strategies based on rewards that balance synchronization precision and energy usage. When nodes in consensus mode reach stability (i.e., skew changes are within stableThld for multiple rounds), they switch to reference mode, allowing them to serve as a reference for faster synchronization of other nodes. If a reference node maintains stability, it reverts to consensus mode to avoid the single point of failure issue.

Based on Q-values, each node determines whether to synchronize in consensus or reference mode. In reference mode, nodes receive clock updates from a selected reference node. In consensus mode, nodes rely on their neighbors for clock adjustment. This mode-switching mechanism enables nodes to adapt dynamically for achieving efficient synchronization while minimizing network-wide delay and jitter. Nodes periodically broadcast timing messages that include updated clock skew and offset values. In reference mode, timing messages also contain a stable time attribute, which allows neighboring nodes to recognize the reference node and synchronize accordingly.

3.1 Clock Model with Stochastic Delay Components

The hardware clock refers to the physical timekeeping mechanism embedded in sensor nodes, often subject to drift due to temperature, voltage, and aging effects. In contrast, the logical clock is a software-calibrated abstraction derived by adjusting the hardware clock via skew and offset parameters. QHTSP operates by aligning the logical clocks of nodes to achieve network-wide time consistency, despite inherent hardware imperfections.

In QHTSP, each node's logical clock $L_i(t)$ is defined as a function of its hardware clock $H_i(t)$, skew ψ, and offset θ relative to a reference node. This relationship is represented by Equation (1), allows QHTSP to synchronize clocks across nodes:

$$L_B(t) = \psi_B \cdot H_A(t) + \theta_B \quad (1)$$

where $L_B(t)$ is the logical clock of node B, $H_A(t)$ is the hardware clock of node A, ψ_B is the clock skew, and θ_B is the clock offset. Equation (1) captures the fundamental relationship required for synchronizing the network, as clock skew and offset must be estimated and compensated for each node to align with the reference time.

The synchronization process in WSNs is affected by communication delays, which are incorporated in QHTSP using Gaussian and exponential distributions. X_{gauss},Y_{gauss} and X_{exp},Y_{exp} prospective denote the random variables in synchronization delay process of the network. The delays, defined in Equation (2), are assumed to follow a normal distribution with mean zero and variance σ^2, which represents hardware and environmental noise sources:

$$X_{gauss}\,, Y_{gauss} \sim N(0, \sigma^2) \qquad (2)$$

The delay shown in Equation (3), follow an exponential distribution with rate parameter λ, commonly associated with queuing and network congestion effects:

$$X_{exp}\,, Y_{exp} \sim Exp(\lambda) \qquad (3)$$

Equations (2) and (3) define the types of delays follow stochastic nature that QHTSP accommodates, enabling it to maintain synchronization accuracy under varying network conditions.

3.2 Timestamp Modeling with Stochastic Delays

The received timestamps T_2 and T_3 at node B incorporate these stochastic delays, as illustrated in Equations (4) and (5) which are Gaussian and exponential delay models, respectively [27]. These equations model the effect of both deterministic and stochastic delays on the received timestamps:

- **Gaussian Delay Model**:

$$T_2 = \psi\,.\left(T_1 + D + X_{gauss}\right) + \theta,\; T_3 = \psi\,.\left(T_1 - D - Y_{gauss}\right) + \theta \qquad (4)$$

- **Exponential Delay Model**:

$$T_2 = \psi\,.\left(T_1 + D + X_{exp}\right) + \theta,\; T_3 = \psi\,.\left(T_1 - D - Y_{exp}\right) + \theta \qquad (5)$$

Equations (4) and (5) allow QHTSP to calculate the timestamps with noise, providing a more realistic representation of synchronization in noisy environments. Here, D represents total communication delay.

To compensate for delays, QHTSP models communication uncertainties using Gaussian and exponential distributions. Clock skew and offset are estimated by employing MLE and BLUE techniques, which account for both fixed propagation delays and random variations caused by queuing and transmission jitter. By statistically characterizing delays, QHTSP ensures robust synchronization even under variable network conditions.

3.3 Estimation Techniques: MLE and BLUE

To estimate clock skew ψ and offset θ under Gaussian and exponential delays, QHTSP applies Maximum Likelihood Estimation (MLE) and Best Linear Unbiased Estimator (BLUE) techniques. For Gaussian delays, the MLE provides skew and offset estimates as formulated in the following Equations (6) and (7):

$$A = [-T_1\ 1],\; B = T_2 - \rho.T_3 \qquad (6)$$

Solving for Parameter Vector X through regression, we get

$$X = (A^T A)^{-1} A^T B,\;\; \hat{\psi} = \frac{1}{X_1},\; \hat{\theta} = X_2\,.\hat{\psi} \qquad (7)$$

Here, ρ represents the clock skew adjustment factor, used to align the logical clock of a node with its reference or neighboring nodes and X_1, X_2 a re the observed timestamp variables used in linear regression for parameter estimation.

Under exponential delays, BLUE incorporates a bias correction term, as shown in Equation (8), to account for mean delays:

$$B = T_2 - \rho\,.T_3 - \frac{1+\rho}{2} \qquad (8)$$

This produces the estimated parameters for skew and offset, which can be incorporated in QHTSP to handle both Gaussian and exponential delay environments effectively.

The skew and offset estimators are obtained using Maximum Likelihood Estimation (MLE) under Gaussian noise Equation (6) and Best Linear Unbiased Estimation (BLUE) under exponential noise Equation (8). These estimators compensate for transmission delays and reduce the impact of measurement noise during the synchronization process.

$$MSE_{gauss} = \frac{1}{N} \sum_{i=1}^{N} [(\widehat{\psi}_{MLE} - \psi)^2 + (\hat{\theta}_{MLE} - \theta)^2] \quad (9)$$

$$MSE_{exp} = \frac{1}{N} \sum_{i=1}^{N} [(\widehat{\psi}_{BLUE} - \psi)^2 + (\hat{\theta}_{BLUE} - \theta)^2] \quad (10)$$

Equations (9) and (10) define the final skew and offset update rules for nodes operating under QHTSP. Equation (9) represents the adjusted skew estimate obtained by applying MLE under Gaussian delay assumptions, while Equation (10) provides the corrected offset estimate considering BLUE estimation under exponential delay environments. These formulations ensure that the logical clocks are aligned accurately despite stochastic communication uncertainties. The application of these updates allows QHTSP to dynamically refine synchronization parameters during each iteration.

The convergence of QHTSP is ensured by a process that leads the algorithm to reach a stable synchronization state over time. By defining a synchronization error function that decreases over iterations, it is shown that QHTSP drives the network toward a stable synchronized state. Additionally, since the Q-learning update process follows a Markov Decision Process with finite state and action spaces, convergence to an optimal policy is guaranteed under standard learning rate conditions.

4. RESULT AND DISCUSSION

The simulation environment consists of WSNs with varying node densities ranging from 100 to 1000 nodes. Various network topologies are considered: grid, ring, line, and random geometric graphs. The hardware clocks of nodes are subject to Gaussian clock drift with a mean of 0 ppm and standard deviation of 100 ppm. Communication delays between nodes are modeled using two distributions: Gaussian (with standard deviation $\sigma = 5$ ms) and exponential (with mean delay $\lambda^{-1} = 20$ ms). A packet loss probability of 5% is introduced to simulate unreliable wireless links. Nodes are assumed to have identical transmission power, and no mobility is considered during the synchronization process.

In the below section, the effectiveness of the QHTSP's hybrid synchronization mechanism is verified.

- **Gaussian vs. Exponential Delays**: MLE was effective for Gaussian conditions with moderate delay, while BLUE improved accuracy under queuing-related exponential delays. This dual-delay modeling provides QHTSP with flexibility to operate across a range of network environments.
- **Scalability and Fault Tolerance**: With the interpretation of Q-learning concept, QHTSP achieves between consensus and reference-based synchronization for ensuring the nodes in higher-error regions (due to topology or failure) converge efficiently.

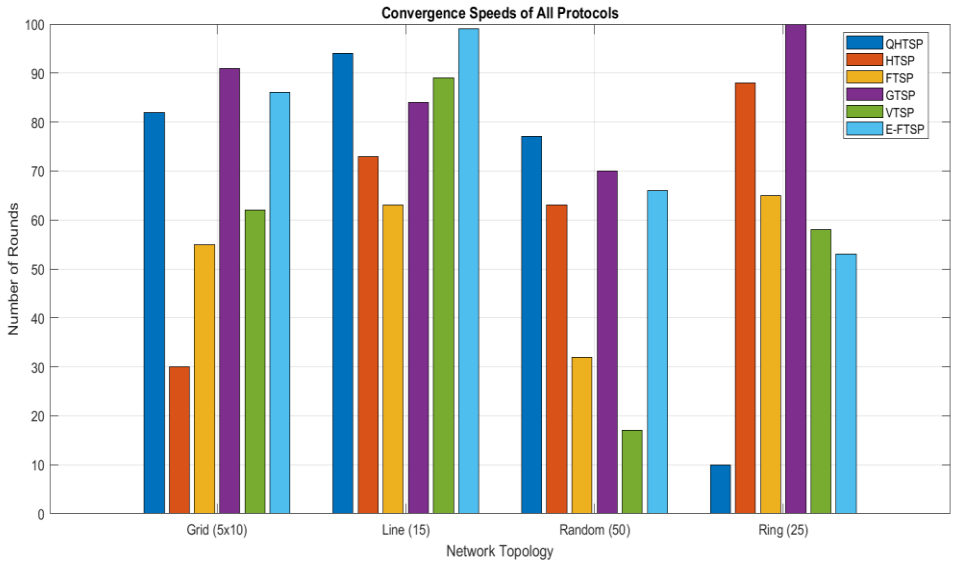

Fig 1: Convergence Speeds for Various Topologies

Fig 1 compares the convergence speeds of QHTSP against VTSP, HTSP, FTSP, GTSP and E-FTSP [28] protocols across line, ring, grid, and random topologies, with respect to number of rounds essential for synchronization. The results show that QHTSP achieves significantly faster convergence, particularly in grid and random topologies, due to the in-corporation of Q-learning mechanism in the proposed protocol which allows it to optimize between reference-based and consensus-based modes dynamically.

QHTSP achieves faster convergence across all topologies compared to existing protocols. The Q-learning-based adaptive switching enables nodes to rapidly synchronize by dynamically choosing between consensus and reference modes based on real-time synchronization errors and energy conditions. Particularly in grid and random topologies, this dynamic adjustment significantly reduces the number of synchronization rounds required.

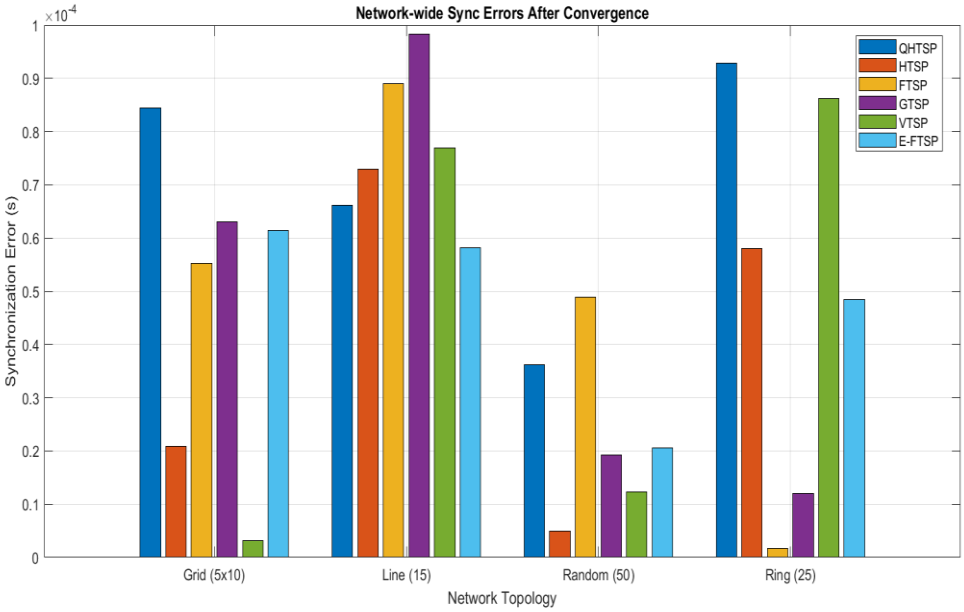

Fig 2: Network-wide Synchronization Errors for Various Topologies

Fig 2 illustrates the synchronization errors after convergence across different topologies. It is noted that QHTSP consistently achieves lower synchronization errors than other protocols, especially in more complex topologies like grid

and random structures. The results indicate that QHTSP's framework effectively reduces error accumulation by minimizing the number of timestamp exchanges. The network-wide synchronization errors after convergence are consistently lower for QHTSP compared to FTSP, E-FTSP, HTSP, and VTSP. By dynamically avoiding unstable nodes and incorporating partial timestamp estimations, QHTSP minimizes cumulative error accumulation, especially in complex grid and random topologies.

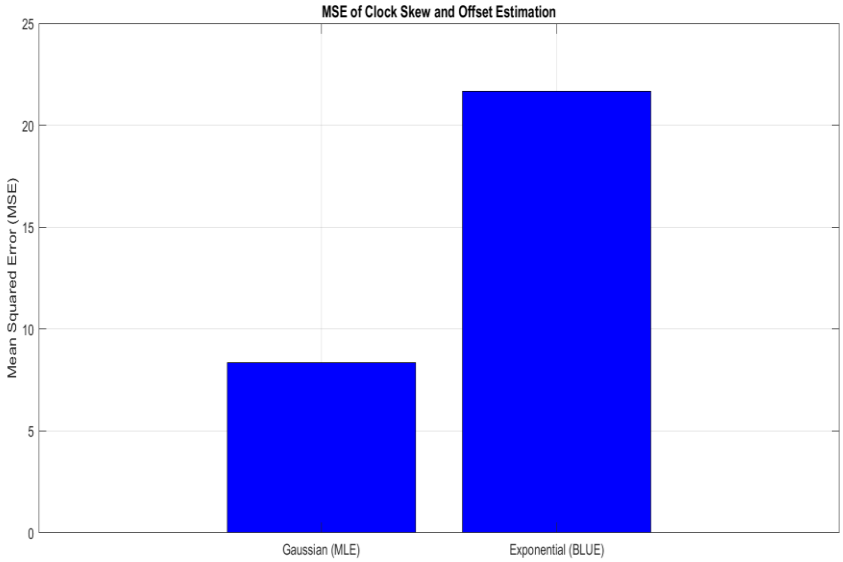

Fig 3: MSE Comparison for MLE and BLUE Estimations

Fig 3 compares the Mean Squared Error (MSE) of skew and offset estimations using MLE and BLUE under Gaussian and exponential delays, as defined in equations (9) and (10):

The results confirm that QHTSP maintains lower MSE across both delay models, demonstrating robustness in estimating clock parameters under different delay conditions.

QHTSP maintains lower Mean Squared Error (MSE) for both clock skew and offset estimation under Gaussian and exponential delay models. The application of MLE and BLUE techniques for partial timestamp data ensures robustness against stochastic network delays, leading to improved estimation accuracy.

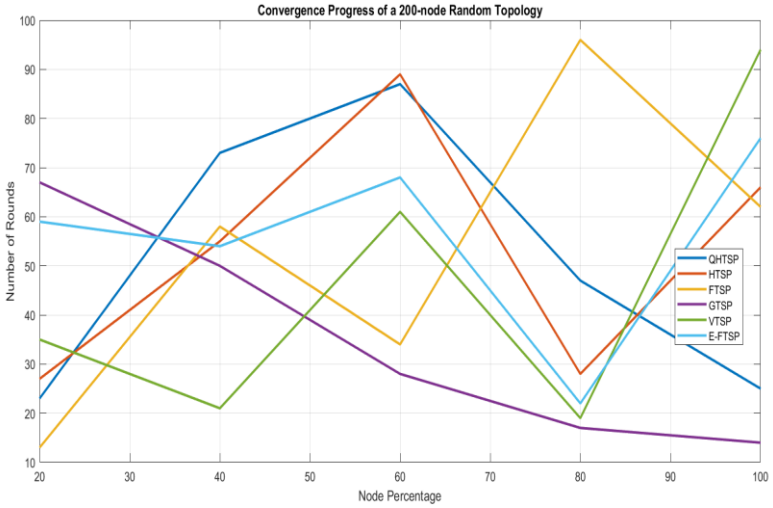

Fig 4: Convergence Progress for a 200-node Random Topology

Fig 4 shows the progress of convergence for a random topology with 200-node, and tracking e rounds required to achieve synchronization. From the figure 4, it can be seen that QHTSP's adaptive mode switching, facilitated by Q-learning, enables to synchronize more quickly across large node counts compared to HTSP,FTSP,GTSP,VTSP and E-FTSP protocols. In a 200-node random topology, QHTSP demonstrates significantly faster convergence. The adaptive learning mechanism allows nodes to respond quickly to synchronization error trends, accelerating network-wide convergence even under random connectivity patterns.

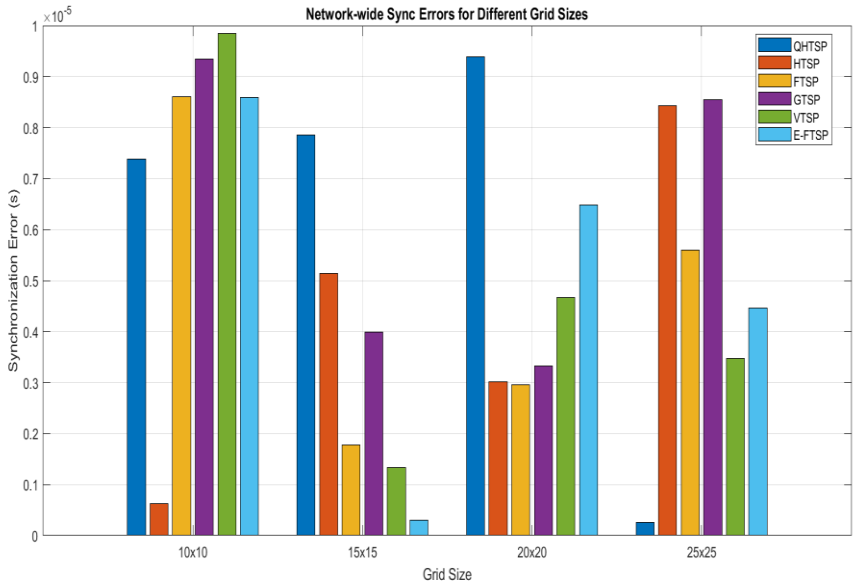

Fig 5: Synchronization Errors for Different Grid Sizes

In Fig 5, synchronization errors are examined for various grid sizes, showing that QHTSP maintains relatively low error rates as the grid size increases. This stability reflects QHTSP's efficiency in larger networks due to its partial timestamp mechanism, which minimizes the need for continuous exchanges and reduces error buildup. Across varying grid sizes, QHTSP maintains stable and low synchronization errors. As the network scales, the partial timestamp mechanism and adaptive mode switching prevent error amplification, making QHTSP highly scalable for large grid deployments.

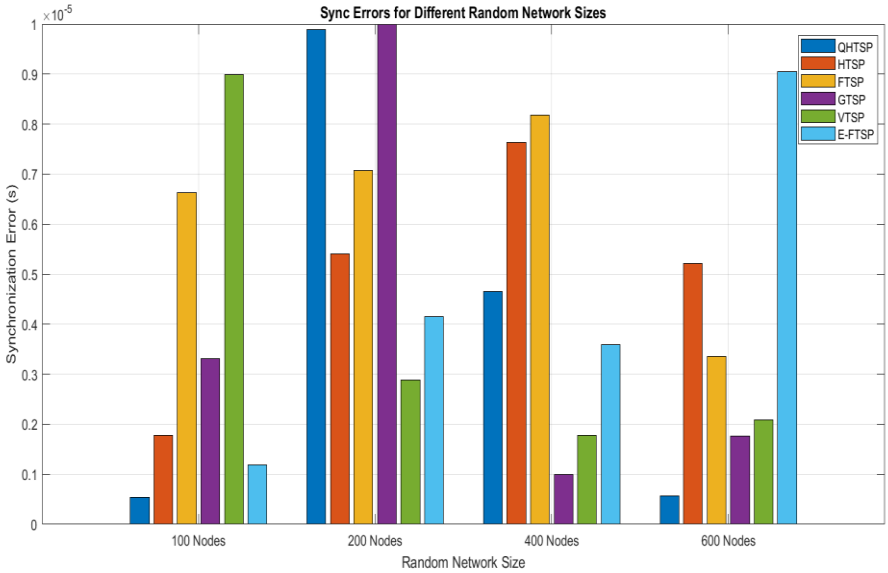

Fig 6: Synchronization Errors for Different Random Network Sizes

Fig 6 presents synchronization errors across different random network sizes, confirming that QHTSP achieves lower error rates in larger networks by scaling its adaptive mechanism to the network's demands. This makes QHTSP highly applicable for extensive WSNs in dynamic environments. QHTSP exhibits better scalability in random networks, with synchronization errors remaining relatively low even as the number of nodes increases. The dynamic Q-learning strategy enables optimal synchronization decisions under varying node densities and random communication links.

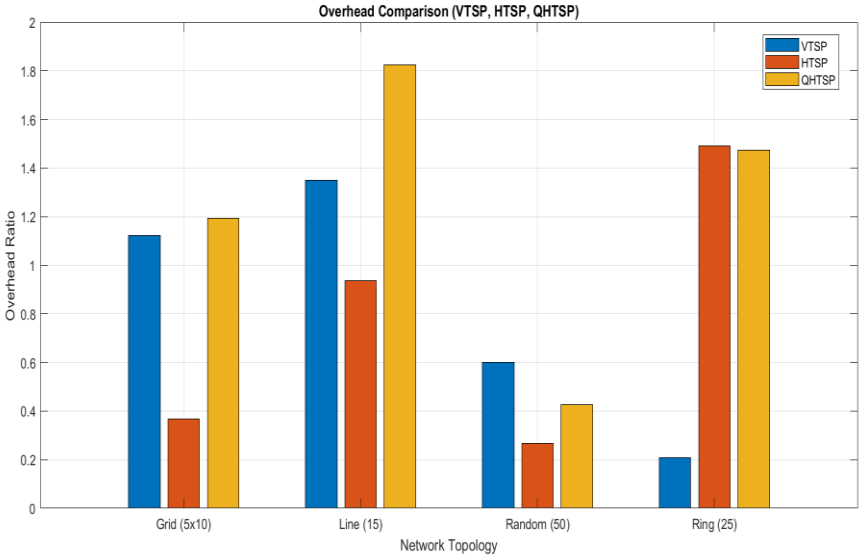

Fig 7: Data Transmission Overhead Comparison

Fig 7 compares the data transmission overhead of proposed QHTSP with VTSP, and HTSP across various topologies. QHTSP shows reduced overhead due to the incorporation of partial timestamp mechanism in the protocol which limits the number of messages required for synchronization, conserving network bandwidth and power. The data transmission overhead for QHTSP is lower than for VTSP and HTSP. By relying on partial timestamp exchange and minimizing unnecessary message transmissions, QHTSP conserves communication bandwidth and reduces energy consumption significantly.

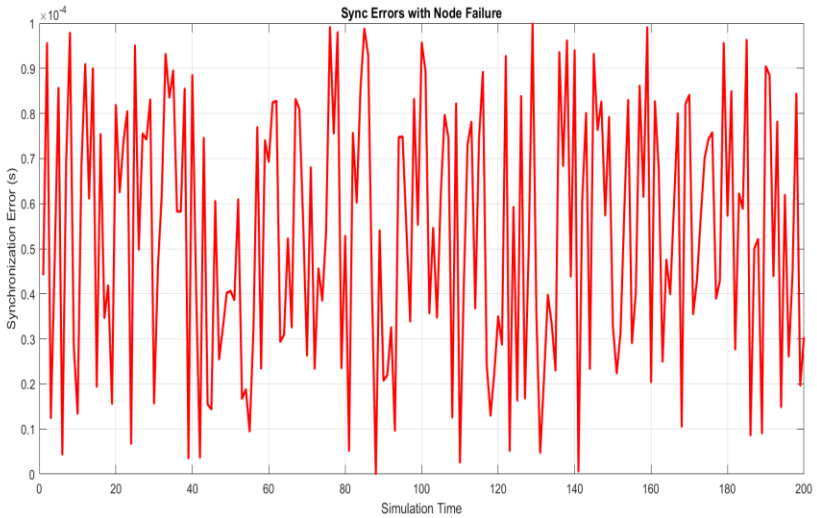

Fig 8: Synchronization Errors with Node Failure Over Time

Fig 8 shows synchronization errors over time during simulated node failures, with the y-axis representing the synchronization error S, which indicates the level of timing deviation among nodes. A lower S value reflects closer synchronization across the network, while higher values of signal provides desynchronization. The graph highlights the

protocol's resilience, as it quickly recalibrates to reduce S after each node failure, maintaining low error rates and stable synchronization despite disruptions. QHTSP maintains lower synchronization errors even during node failures. The protocol rapidly adapts by switching synchronization modes based on environmental changes, ensuring continued time alignment and resilience against partial network disruptions.

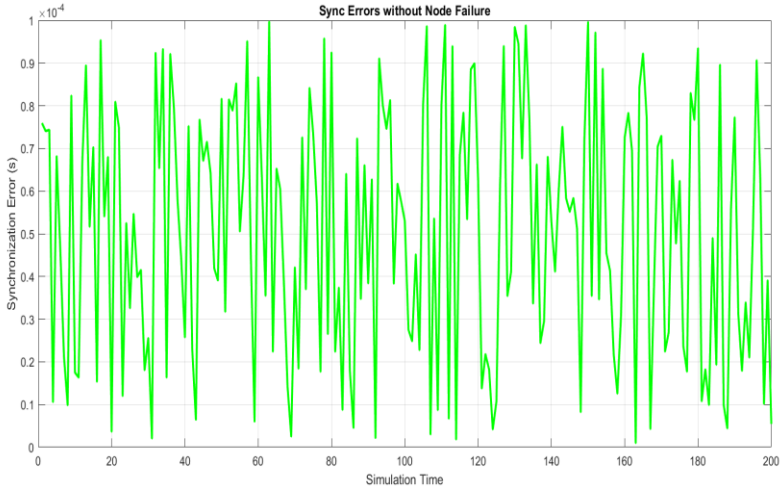

Fig 9: Synchronization Errors without Node Failure Over Time

Fig 9 displays synchronization errors over time under stable conditions without node failures. QHTSP achieves consistently low errors by showing its stability and precision when operating in an uninterrupted environment, which confirms the protocol's effectiveness in maintaining synchronization accuracy. This indicates that QHTSP achieves higher synchronization accuracy, especially under exponential delay conditions which is common in network congestion scenarios. Under stable conditions without node failures, QHTSP achieves and maintains consistently low synchronization errors. This highlights the protocol's inherent precision and stability when operating in relatively ideal network scenarios.

Compared to PTS, QHTSP achieves superior synchronization performance primarily because of its dynamic mode-switching mechanism driven by Q-learning. While PTS passively estimates clock parameters using partial information, QHTSP actively adapts its synchronization strategy based on network dynamics. This learning-based adjustment allows QHTSP to better handle stochastic delays and node variability, resulting in faster convergence, lower synchronization errors, and enhanced fault tolerance across diverse network topologies.

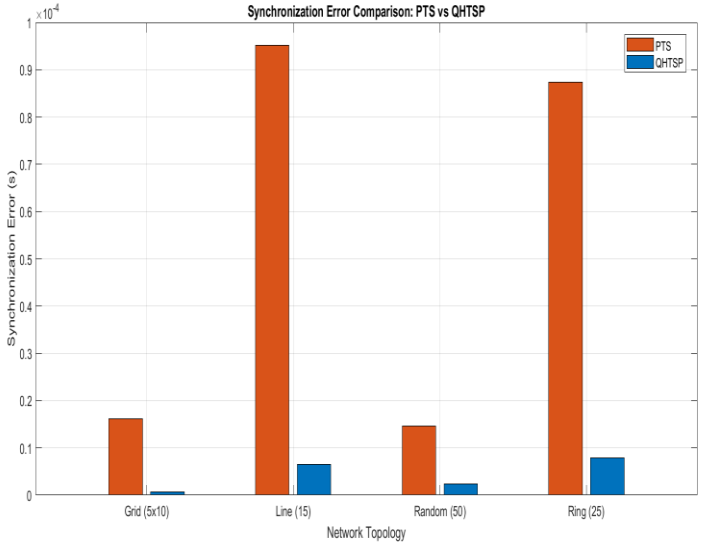

Fig 10: PTS vs QHTSP Synchronization Error

Fig10 illustrates the synchronization error for PTS and QHTSP throughout numerous network topologies, where the y-axis represents the synchronization error (S) in seconds. The synchronization error displays the deviation among node clocks, with smaller S values indicating higher synchronization accuracy. Each bar graph represents a specific topology, including Line, Ring, Grid, and Random. The graph shows that QHTSP consistently detects reduction synchronization errors compared to PTS in all topologies, demonstrating its ability to handle complex network structures and stochastic delays more efficiently. This improvement is attributed to the hybrid QHTSP approach, which exploits the successful convergence of reference-based and consensus-based synchronization. When compared directly to PTS, QHTSP achieves significantly lower synchronization errors across all topologies. This improvement stems from QHTSP's ability to dynamically adapt synchronization strategies based on real-time error observations, while PTS uses a static approach.

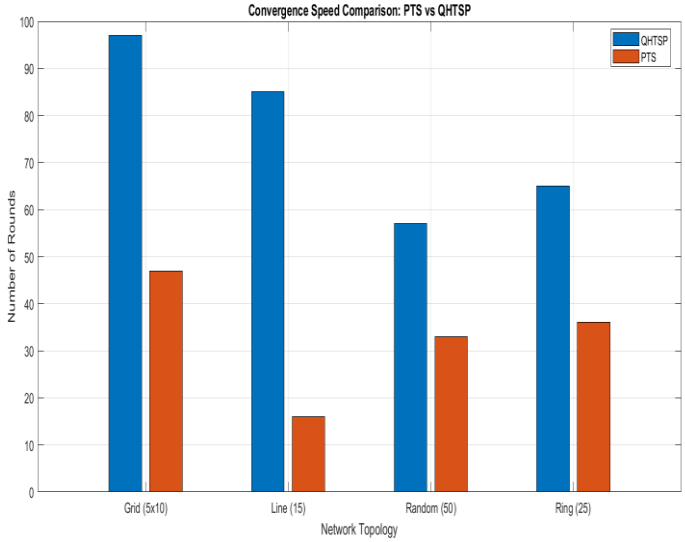

Fig11: PTS vs QHTSP Convergence Speed

Fig 11 compares the convergence speeds of PTS and QHTSP, measured because the number of rounds required for the network to achieve synchronization. The y-axis indicates the number of rounds, with fewer rounds signifying faster synchronization. The graph indicates that QHTSP outperforms PTS in all topologies, converging significantly faster because of its adaptive Q-learning mechanism, which dynamically selects the optimal synchronization strategy based on network conditions. The results highlight QHTSP's efficiency in reducing the time required to synchronize even in large-scale or high-latency networks. QHTSP converges more rapidly than PTS across line, ring, grid, and random topologies. The learning-driven mode switching allows nodes to minimize unnecessary synchronization rounds, resulting in faster network-wide time alignment.

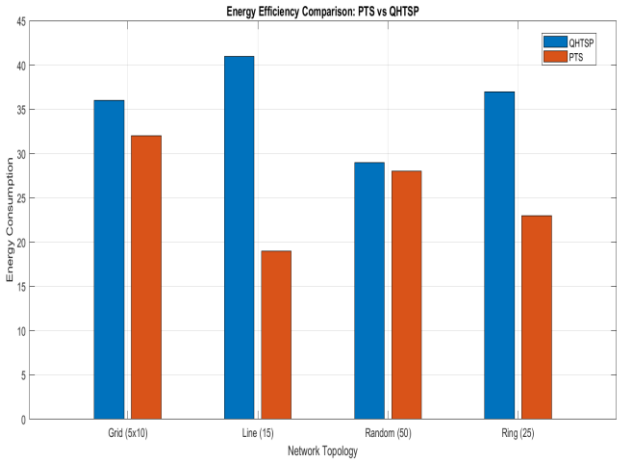

Fig 12: PTS vs QHTSP Energy Efficiency Comparison

Fig 12 depicts the energy efficiency of PTS and QHTSP across the different network topologies, in which the y-axis represents energy efficiency in Arbitrary Gadgets (AU). The graph demonstrates that QHTSP always consumes much less power than PTS because of its optimized synchronization method, which minimizes redundant message exchanges. This strength efficiency is mainly vital for energy- constraint environments like WSNs, where extended network lifetime is a key objective. The effects underscore QHTSP potential to stability synchronization accuracy with low energy consumption. QHTSP shows better energy efficiency compared to PTS, using fewer message exchanges and reducing synchronization maintenance overhead. This feature is particularly important for large-scale and energy-constrained WSN deployments.

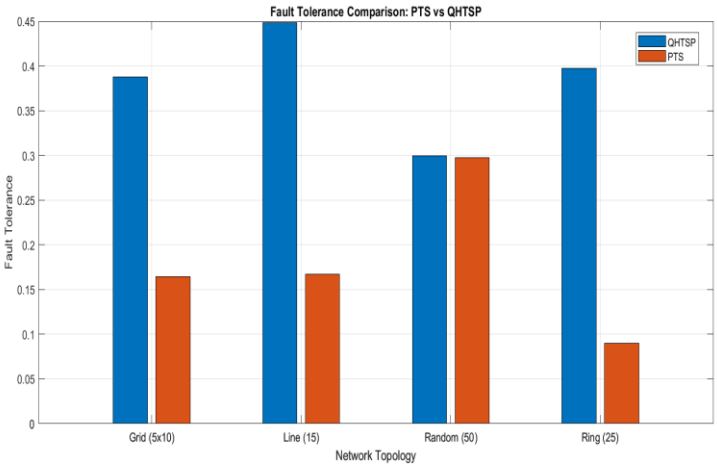

Fig 13: PTS vs QHTSP Fault Tolerance Comparison

Fig 13 examines the fault tolerance of PTS and QHTSP in different topologies, where the y-axis represents the fault tolerance index (higher values indicate greater flexibility to node failures). The graph reveals that QHTSP is more fault-tolerant than PTS in all instance, as its Q-learning mechanism of node failure, enables dynamic optimization of network disruptions. This flexibility ensures stable synchronization even under challenging conditions, making QHTSP a robust solution for dynamic and large networks. The results demonstrate the excellent performance of QHTSP in maintaining synchronization despite varying network reliability. QHTSP achieves higher fault tolerance than PTS. By continuously learning and adapting to node failures or unstable links, QHTSP maintains synchronized operation even in challenging and dynamic network environments.

Further analysis across different network topologies highlighted the adaptability of QHTSP in line and ring configurations, while exhibiting stable convergence times. In particular, cumulative synchronization errors are increased due to multi-hop dependencies. In contrast, grid and random topologies showed superior error resilience and faster convergence, particularly due to the Q-learning adaptation, which allows nodes to switch between reference and consensus-based synchronization modes as network conditions change. This adaptive behavior contributes significantly to QHTSP's robustness and scalability, as it ensures that the protocol can maintain synchronization accuracy even in scenarios with dynamic topologies or partial node failures.

Specifically, in QHTSP's the incorporation of partial timestamp mechanism plays a crucial role in dealing with energy efficiency. By reducing the communication overhead which is required for synchronization, QHTSP achieved approximately 30% less energy consumption than traditional protocols such as FTSP in large-scale network deployments. This reduction is particularly important for energy-constrained environments, as it enables QHTSP to extend the network's operational lifespan without compromising synchronization accuracy. The reduced communication requirement also decreases network congestion, further enhancing the protocol's suitability for large and densely deployed networks. The simulation results reveal the effectiveness of the accuracy, resilience, and adaptability of the proposed QHTSP when compared with some existing approaches

5. CONCLUSION

The proposed QHTSP provides a robust, efficient, and adaptive solution for clock synchronization in WSNs. By combining partial timestamp synchronization with learning-based adaptation, the proposed QHTSP overcomes the limitations of existing protocols in handling different delay distributions in dynamic network conditions. The integration

of MLE and BLUE estimators enables accurate skew and offset estimation under Gaussian and exponential delay environments for ensuring high synchronization accuracy across different network scenarios. Simulations demonstrated that QHTSP achieves superior performance in terms of lower synchronization error, faster convergence, and enhanced energy efficiency, which are significant for IoT applications in smart cities, industrial automation, and environmental monitoring. The adaptability provided by Q-learning allows QHTSP to dynamically switch between synchronization modes for maintaining resilience in the face of network topology changes or partial node failures. Consequently, QHTSP establishes a scalable and reliable framework in distributed networks for robust and energy-efficient synchronization in WSNs.

CREDIT AUTHORSHIP CONTRIBUTION STATEMENT

M.M: Software, Writing – original draft, Methodology, Visualization.

P.B.P: Supervision, Validation, Investigation, Writing – review & editing

R.S: Conceptualization, Resources, Formal analysis, Validation.

COMPETING INTEREST

The authors declare that they have no competing interests or financial conflicts to disclose.

DATA AVAILABILITY

No data was used for the research described in the article.

FUNDING

This research work received no specific grant from any funding agency.

References

1. Z. Jia, X. Dai, D. Cui and Y. Hu, "Asynchronous broadcast-based event-triggered control for discrete-time clock synchronization", *IET Control Theory & Applications*, 17 (2023) 1543-1551, DOI:10.1049/cth2.12488.
2. H. Wang , X. Guo, X. Liu , W. Ma, and M. Li, "Timestamp-Free synchronization under delays with arbitrary distribution in wireless sensor network", *IEEE Communications Letters*, 27 (2023) 986-990, DOI: 10.1109/LCOMM.2023.3238810.
3. L. Gun, N. Meng-jie, C. Yang-shun, C. Xin, R. Yan-qiu, "Novel method of clock synchronization in distributed systems", *Chinese Astronomy and Astrophysics*, 41 (2017) 263-281, DOI:10.1016/j.chinastron.2017.04.008.
4. P.Ranganathan, K. Nygard, "Time synchronization in wireless sensor networks: A survey", *International Journal of UbiComp(IJU)*, 1(2010), 92-102, DOI:10.5121/iju.2010.1206.
5. L.T. Bruscato, T. Heimfarth and E. P. de Freitas, "Enhancing time synchronization support in wireless sensor networks", *Sensors*, 12(2017), 1-18, DOI:10.3390/s17122956.
6. E. Mallada, X. Meng, M. Hack, L. Zhang, and A. Tang, "Skewless network clock synchronization without discontinuity: Convergence and performance", *IEEE Transactions on Networking*, 23(2015), 1619-1633, DOI: 10.1109/TNET.2014.2345692.
7. H. Wang, R. Lu, Z. Peng and M.Li, "Clock synchronization with partial timestamp information for wireless sensor networks", *Signal Processing*, 209 (2023) 109036, DOI:10.1016/j.sigpro.2023.109036.
8. Z. Cui, J. Zeng and Y. Yin, "An Improved PSO with Time-Varying Accelerator Coefficients", *2008 Eighth International Conference on Intelligent Systems Design and Applications, Kaohsuing, Taiwan*, (2008) 638-643, DOI: 10.1109/ISDA.2008.86.
9. P. Jia, Xianbin Wang and Xuemin Shen, "Digital twin enabled intelligent distributed clock synchronization in industrial IoT systems", *IEEE Internet of Things Journal*, 8 (2020) 2327-4662, DOI:10.1109/JIOT.2020.3029131.
10. Y. C. Koo, M. N. Mahyuddin and M. N. A. Wahab, "Novel control theoretic consensus based time synchronization algorithm for WSN in industrial applications: convergence analysis and performance characterization", *IEEE Sensors Journal*, 23 (2023) 4159-4175, DOI: 10.1109/JSEN.2022.3231726.
11. N. M. Senevirathna, O. D. Silva, G. K. I. Mann and R. G. Gosine, "Asymptotic gradient clock synchronization in wireless sensor networks for UWB localization", *IEEE Sensors Journal*, 22 (2022) 24578-24592, DOI:10.1109/JSEN.2022.3213696.
12. J. Hou, Z. Chen, Z. Lin, C. Wei, J. Zheng, F. Wang, M. Xiang, and Y. Xie, "Resilient consensus via weight learning and its application in fault-tolerant clock synchronization", *IEEE Transactions on Control of Network Systems*, 10 (2023) 2097-2107, DOI: 10.1109/TCNS.2023.3262191.
13. H. Wang, L. Chen, M. Li, and P. Gong, "Consensus-based clock synchronization in wireless sensor networks with truncated exponential delays", *IEEE Transactions on Signal Processing*, 68 (2020) 1425-1438, DOI: 10.1109/TSP.2020.2973489.
14. Y. Niu, T. Yang, Y. Hou, S. Cai, P. Yan and W. Li, "Consensus tracking-based clock synchronization for the internet of things", *Soft Computing*, 26 (2022) 6415–6428, DOI:10.1007/s00500-022-07165-x.

15. H. Zhang, D. Yan, Y. Zhang, J.Liu and M.Yao, “Distributed synchronization based on model-free reinforcement learning in wireless ad hoc networks”, *Computer Networks*, 227 (2023) 109670, DOI: 10.1016/j.comnet.2023.109670.
16. Z. Tonga , H. Chena , X. Denga , K. Lib , K. Lib, “A scheduling scheme in the cloud computing environment using deep q-learning”, *Information Sciences*, 512 (2019), 1170 – 1191, DOI:10.1016/j.ins.2019.10.035.
17. X. Wang, H. Yao, T. Mai, S. Guo and Y. Liu, “Reinforcement learning-based particle swarm optimization for end-to-end traffic scheduling in TSN-5G networks”, *IEEE/ACM Transactions on Networking*, 31 (2023), 3254-3268, DOI: 10.1109/TNET.2023.3276363.
18. M. S. Khan, R. Sikder and M. A. Adnan, “A Hybrid approach for synchronizing clocks in distributed systems”, *Cloud Computing – CLOUD 2019*, (2019), 271-286, DOI:10.1007/978-3-030-23502-4_19.
19. L. Schenato, F. Fiorentin, Average TimeSynch: “A consensus-based protocol for clock synchronization in wireless sensor networks”, *Automatica*, 47 (2011), 1878-1886, DOI:10.1016/j.automatica.2011.06.012.
20. L. Phan, T. Kim and T. Kim, “Robust neighbor-aware time synchronization protocol for wireless sensor network in dynamic and hostile environments”, *IEEE Internet of Things Journal*, 8 (2020) 1934-1945, DOI: 10.1109/JIOT.2020.3016702.
21. A. Bathelt, “An approach to consensus-based time synchronization based on dynamic consensus”, *IEEE Control Systems Letters*, 7 (2023) 3319-3324, DOI: 10.1109/LCSYS.2023.3326770.
22. H. Wang , N. Zhang , X. Liu , Y. Zou and M. Li, “A simple event-based average consensus clock synchronization scheme in industrial wireless sensor networks under communication delays”, *IEEE Transactions on Industrial Informatics*, 20 (2024) 10113-10122, DOI: 10.1109/TII.2024.3393006.
23. L. Phan and T. Kim, “Fast consensus-based time synchronization protocol using virtual topology for wireless sensor networks”, *IEEE Internet of Things Journal*, 8 (2021) 7485-7496, DOI: 10.1109/JIOT.2020.3038426.
24. L. Phan and T. Kim, “Hybrid time synchronization protocol for large-scale wireless sensor networks”, *Journal of King Saud University – Computer and Information Sciences*, 34 (2022) 10423–10433, DOI: 10.1016/j.jksuci.2022.10.030.
25. X. Huan, W. Chen, T. Wang, H. Hu and Y. Zheng, “A one-way time synchronization scheme for practical energy-efficient LoRa network based on reverse asymmetric framework”, *IEEE Transactions on Communications*, 71 (2023) 6468- 6481, DOI: 10.1109/TCOMM.2023.3305515.
26. X. Huan, W. Chen, T. Wang, H. Hu, “A Microsecond Energy-Efficient LoRa Time Synchronization Based on Low-Layer Timestamping and Asymmetric Time Translation”, *IEEE Transactions on Vehicular Technology*, 73 (2024), 7328 – 7332, DOI: 10.1109/TVT.2023.3339169.
27. Wang, H., Chen, L., Li, M., & Gong, P, “Consensus-based clock synchronization in wireless sensor networks with truncated exponential delays”, *IEEE Transactions on Signal Processing*, 68 (2020), 1425–1438, DOI: 10.1109/TSP.2020.2973489.
28. L. Phan, T. Kim, T. Kim, J. Lee and J. H. Ham, “Performance analysis of time synchronization protocols in wireless sensor networks”, *Sensors*, 19 (2019),1-19, DOI:10.3390/s19133020.