Cluster-based Improvised Time Synchronization Algorithm for Multihop IoT Networks

Original Scientific Paper

Neha Dalwadi*
Shri K. J. Polytechnic, Bharuch,
Department of Computer Engineering
Bholav, Bharuch, India
neha.dalwadi@gmail.com

Mamta Padole
The Maharaja Sayajirao University of Baroda,
Faculty of Technology and Engineering, Department of Computer Science and Engineering
Kala bhavan, Vadodara, India
mpadole29@rediffmail.com

*Corresponding author

Abstract *– Achieving precise time synchronization among wireless sensor devices within Internet-of-Things (IoT) networks poses a significant challenge. Various approaches have been proposed to efficiently synchronize time in wireless sensor networks (WSNs) used in the IoT. However, these solutions typically involve extensive message exchanges to achieve synchronization, leading to notable communication and energy overheads. In this context, we introduce a clustering approach aimed at enhancing the Reference Broadcast Synchronization (RBS) protocol to suit large multihop IoT networks. This paper discusses existing cluster-based time synchronization methods and compares their effectiveness. Moreover, our proposed clustering approach seamlessly integrates existing time synchronization protocols, thereby enhancing both power efficiency and synchronization accuracy, which are specifically tailored for multihop IoT networks. To validate the effectiveness of our approach, we conducted emulations, which demonstrated a significant improvement in minimizing synchronization error by 78% compared to existing RBS methods, along with a 40% reduction in the power consumption of reference nodes. Overall, our proposed method yields satisfactory results with less overhead in scalable IoT networks.*

Keywords: *Clustering Algorithms, Internet of Things, Time Synchronization, Wireless Sensor Network, Network Topology*

Received: February 28, 2024; Received in revised form: April 18, 2024; Accepted: May 10, 2024

1. INTRODUCTION

The IoT describes a network of interconnected physical devices and other objects that are integrated with sensors, actuators, software and communicate data throughout the network. Time synchronization among IoT devices plays an important role in IoT applications such as smart grids, smart parking systems, health monitoring systems, mobile communications, environment monitoring systems, and distributed resource allocation [1-5]. For the optimal operation of IoT networks, precise time coordination is essential. Proper synchronization minimizes communication collisions, reduces retransmissions, and contributes to energy conservation. While the study of clock synchronization in wireless sensor networks has been performed for many years, IoT devices have some challenges. IoT devices use small components, which usually have limited battery power, constrained memory resources, low cost and less precise crystal oscillators, and low-power sensors and actuators compared to traditional WSN nodes.

To overcome issues of limited battery power, we proposed clustering approach. Due to limited battery power in IoT devices, the prime concern is to conserve power. With the help of clustering communication needs to be done within subset of nodes of the network, which in turn also reduces the distance of communication. Short distance communication reduces transmission delays as well as power consumption. There are less chances of node failure due to power conservation which in turn avoids synchronization error and improves synchronization accuracy.

These small components cause disparities in clock time between devices in a network. As prevalent clock

sources in electronic devices, crystal oscillators furnish a reference frequency for timekeeping. Despite their widespread use, these oscillators are not flawless and introduce various constraints that compromise the precision and stability of time synchronization in IoT devices [6]. Crystal Oscillator Constraints are as follows:

- Crystal oscillators possess a defined frequency accuracy that may shift over time due to factors such as temperature changes. These inaccuracies can result in drift, leading to time discrepancies over prolonged periods [7, 8].
- Fluctuations in temperature can affect the oscillator's frequency, introducing variations in timekeeping. This sensitivity is particularly relevant in IoT devices deployed under diverse environmental conditions.
- The aging effect (gradual changes in frequency over time) can lead to a loss of accuracy in timekeeping, and this is especially important in applications where precise time synchronization is critical.
- Despite their overall power efficiency, crystal oscillators can still impact battery life in energy-constrained IoT devices, particularly for devices operating in remote or battery-powered scenarios.

In applications with strict synchronization requirements, the precision of crystal oscillators may not be sufficient.

To mitigate the impact of these limitations on time synchronization in IoT devices, various methods, such as the use of an external time server to recalibrate the device's internal clock, the implementation of algorithms that can compensate for the aging effect and temperature effects, and the implementation of mechanisms to stabilize temperature effects around crystal oscillators, need to be implemented [7].

Numerous clock synchronization approaches have been proposed for WSNs. The network time protocol (NTP) is widely utilized to synchronize computers over the internet, making it advantageous for application in this new context as well [9-11], its accuracy rarely meets the typical requirements of IoT applications and IoT networks. Moreover, NTP is not suitable for large mesh networks, as several nodes in mesh need to communicate using a gateway (or border router) with an NTP time server. Usually, a gateway is a high-traffic area that may cause delays [11, 12]. A drawback of NTP is its lack of secure time synchronization. To address this issue, the work presented in [13] introduces an NTP-based time synchronization method incorporating trust management and blockchain techniques. A simplified version of the NTP protocol (SNTP) [9] is commonly used in embedded systems. While the SNTP is used in computers with low processing power and in microcontrollers where accuracy is not an issue, it is helpful where scalability and low overheads are needed. It is not suitable for large mesh networks because it works only on high-speed networks such as Ethernet. Numerous clock synchronization protocols have been specifically devised for both wired and wireless networks, including the precision time protocol (PTP) designed for IEEE 802.11 networks. In a modified iteration of PTP, beacon frames are utilized to synchronize the mobility of access points (APs) with one another, as well as to broadcast timestamps acquired by the APs [14]. The study referenced in [15] investigates the use of PTP technology for time synchronization in Industrial IoT. However, their findings are derived from a methodological review of existing data, thus not offering conclusive evidence regarding the reliability of PTP in Industrial IoT. The efficacy of time synchronization hinges on various factors, such as hardware clock precision, sensor accuracy, and environmental influences. Tailored approaches for time synchronization in specific applications have also been introduced. For instance, a three-step method was developed in [16] to estimate clock skew and offsets, specifically for receiver-only based time synchronization in underwater applications. Another example is found in [17, 18], where a synchronized health monitoring system was described. This synchronization was accomplished through the utilization of high-precision external oscillators and GPS systems.

In recent years, various time synchronization protocols have been developed for wireless sensor networks that can work on IEEE 802.11. Many IoT applications have been developed over the IEEE 802.11 network, and they can deploy a time synchronization protocol for time accuracy. Time synchronization algorithms are primarily categorized into centralized and distributed synchronization algorithms. Centralized approaches utilize a reference node to synchronize all nodes within a network. Conversely, distributed algorithms are receiver–receiver-based synchronization methods in which no single reference node is employed. Section 2 provides an overview of existing algorithms in this context. However, not all these protocols provide accurate time synchronization in multihop networks, so providing a time synchronization protocol that is compatible with both single-hop and multihop networks is challenging. Our proposed cluster-based approach for multihop networks incorporates clustering methods to synchronize the whole network. This paper describes the existing RBS approach for time synchronization and proposes the use of clustering methods with RBS for time synchronization in IoT networks. The main goal of this work is to apply a clustering approach with RBS to provide time synchronization among all nodes (network-wide synchronization), particularly in multihop networks.

Achieving precision in time synchronization poses a significant challenge, particularly regarding comparing time information across network-wide nodes. Each node must assess its clock drift and skew based on the time received from a reference node. The accuracy of a clock is heavily influenced by the delays incurred in transmitting time information between locations, which consequently leads to synchronization errors.

Consequently, nodes further away from the reference node experience increased synchronization errors. Efficient synchronization routines necessitate minimizing the number of messages sent by each node and reducing energy consumption. However, in a multihop environment, achieving efficiency in terms of reducing power consumption and latency among nodes, correcting time values, and minimizing synchronization errors are particularly challenging. Additionally, considerations must be made for node failures and node mobility within the network.

Successfully achieving time synchronization in the IoT necessitates a meticulous examination of the trade-offs between precision and diverse resource limitations. These constraints span energy consumption, communication overhead, scalability, latency, and robustness. Striking an appropriate balance customized to the precise needs of IoT deployment is pivotal for maximizing performance and efficiency. As seen from existing time synchronization algorithms [19-28], as discussed in section 2, while flooding the network with synchronization messages may offer high accuracy, it results in significant communication overhead. Minimizing this overhead while maintaining acceptable synchronization levels is vital, especially in resource-constrained IoT setups. Some synchronization methods excel in small-scale deployments but struggle to maintain accuracy as the network expands. Designing protocols that scale effectively while preserving accuracy is key for large IoT deployments. In latency-sensitive applications, minimizing synchronization latency may be prioritized over achieving perfect accuracy. Synchronization methods must consider the system's robustness against network disruptions and node failures. Trade-offs may arise between perfect synchronization and ensuring that the system can recover quickly from disruptions.

Section 2 elaborates on the related work conducted in the field of time synchronization. Section 3 describes the clustering techniques employed in both wireless sensor networks (WSNs) and Internet of Things (IoT) networks. The cluster-based time synchronization approach outlined in section 4 addresses the challenges associated with message overhead by enabling direct communication between nodes for clock corrections. This approach effectively manages node failures by periodically forming clusters at predefined resynchronization intervals. Additionally, it ensures network scalability during synchronization by incorporating new nodes into the cluster. Furthermore, the power consumption is minimized by reducing the latency at various stages of the synchronization process, including the send time, receive time, and propagation time.

2. RELATED WORK IN TIME SYNCHRONIZATION ALGORITHMS

Time synchronization stands as a focal point in the realm of wireless sensor networks and IoT networks, attracting widespread attention in research. Considerable research has been dedicated to the time synchronization of sensor nodes. Nonetheless, there remains an opportunity to refine existing time synchronization algorithms to effectively operate with IoT end devices, prioritizing low power consumption and heightened synchronization accuracy.

Traditional time synchronization algorithms typically follow a Sender–Receiver approach [19], where a root node serves as the time server, and other nodes synchronize with it. This method, often termed centralized time synchronization, has a drawback: if the root node is compromised, the entire network may suffer, resulting in incorrect clock values. Examples of such algorithms include the flooding time synchronization protocol (FTSP) [20], lightweight tree-based synchronization (LTS) [21], the timing synchronization protocol for sensor networks (TPSN) [22], and the flooding with clock speed agreement (FCSA) protocol proposed in [23], aimed at achieving skew synchronization among neighboring nodes. This protocol is tailored to mitigate synchronization errors that escalate with the number of hops in the FTSP.

In contrast, receiver–receiver-based algorithms [19] synchronize based on the arrival time of synchronization messages from other nodes in the network and use estimated offset values to correct their own clock time. However, this approach has limitations, such as high message complexity due to additional message exchanges between nodes and potential message collisions. The algorithms in this category include reference broadcast synchronization (RBS) [24] and time diffusion synchronization protocol (TDSP) [25].

In multihop scenarios, various cluster-based time synchronization approaches have been proposed. The methodologies outlined in references [26-28] employ a cluster-based approach to reduce synchronization errors by minimizing the average hop count from the root node. However, these methods are ill suited for networks with dynamic topologies. Moreover, effective mechanisms for handling node failures during synchronization and ensuring network scalability are lacking. Alternatively, other time synchronization approaches, as described in [29, 30], facilitate the construction of distributed networks and exhibit robustness to dynamic topologies and node failures. However, these algorithms suffer from a significant drawback in the form of packet collisions. This issue increases the overall message complexity of the network, impeding efficient data transmission and potentially leading to network congestion and reduced performance. The proposed C-sync [31], a clustering-based energy efficient decentralized time synchronization protocol, aims to achieve scalability by incorporating multiple reference nodes. However, the protocol's involvement of multiple reference nodes introduces message overhead, thereby increasing the time required to synchronize the entire network. To improve upon the traditional RBS algorithm, [32] introduced adaptive clock synchronization

in sensor networks. While this solution seeks to alleviate the high overhead linked with flooding the network with reference packets, it also introduces tradeoffs such as latency, reliance on sensor nodes, synchronization reliability, and implementation complexity. In their work, [33] introduced a clustering-based hierarchical time synchronization method to facilitate multihop synchronization. This approach utilized long radio ranges and clustering to reduce average hop counts. However, the increased number of referenced messages overhead resulted in delays in the synchronization process. Additionally, the applied overhearing method, while effective in reducing hops, consumed more power, making it unsuitable for power-constrained IoT devices. Furthermore, the method did not support topology changes, posing limitations in dynamic network environments. The approach outlined in [34] introduces a multihop clustering mechanism for scalable IoT networks, with the objective of minimizing the number of Internet connections while maximizing the number of hops to its coordinator. However, it relies on Dijkstra's shortest path first algorithm to calculate the distances among all possible pairs of nodes, with a time complexity of $O(|N|(|N|\log|N|+|E|))$. Additionally, the complexity of obtaining clusters is $O(|N|2\log|N|)$, thereby contributing to an overall increase in the complexity of the clustering approach. In [35], a clock synchronization strategy based on precision time protocol (PTP), aimed at synchronizing clocks between IoT devices and the Cloud, which is interconnected within a distributed network framework, was proposed. Here, the Software as a Service (SaaS) cloud service is used to gather data for analysis and initiate corresponding actions on IoT devices. The approach described in [36] introduces the energy efficient clustering algorithm (EECA) for wireless sensor networks (WSNs). In this algorithm, clusters are formed based on the center of the sensing field, after which the synchronization process commences. Each node synchronizes with its respective cluster head (CH) within the network. In [37], a novel time synchronization method called cluster-based maximum consensus time synchronization (CMTS) was introduced. This method incorporates a rotational cluster head scheme. Synchronization is achieved by exchanging timestamp messages between cluster heads and cluster members and then computing the clock offset. An enhanced time synchronization approach for home automation systems has been proposed in [38], based on the Elastic Timer Protocol (ETP). This approach introduces synchronization overhead resulting from the dynamic adjustment of timer values and synchronization parameters. Work presented in [39] achieves time synchronization with heterogeneous technologies in IoT network based on Cross-Technology Communication technique (CTC). However, CTC introduce additional complexity and computational demands on devices impacts power consumption.

We delve into a detailed examination of existing strategies such as the RBS, FTSP, and TPSN in the context of multihop scenarios for time synchronization. This analysis serves to facilitate a comprehensive comparison with our proposed approach for time synchronization in multihop networks.

1. RBS in Multihop Network

The reference broadcast synchronization (RBS) algorithm operates on a receiver–receiver basis for time synchronization. In this method, a reference node broadcasts reference messages across the network. Neighborhood nodes record the timestamp of received broadcast messages and exchange their local time with other nodes in the network. Subsequently, all nodes calculate the average offset value and estimate their own clock value.

Fig. 1. [24] illustrates a scenario for a multihop network. In the depicted scenario, both Node A and Node B send synchronization pulses at times PA and PB, respectively. Receiver node 4 (R4) captures both sync pulses and forward the clock information from one neighborhood to another. Receivers R1 and R7 detect events at times E1R1 and E7R7, respectively. R4 leverages both A's and B's reference broadcasts to establish the best-fit line for adjusting clock values from R1 to R4 and from R4 to R7, respectively. However, this scheme necessitates a lengthy process for R4 to compute the correct time, leading to delays.

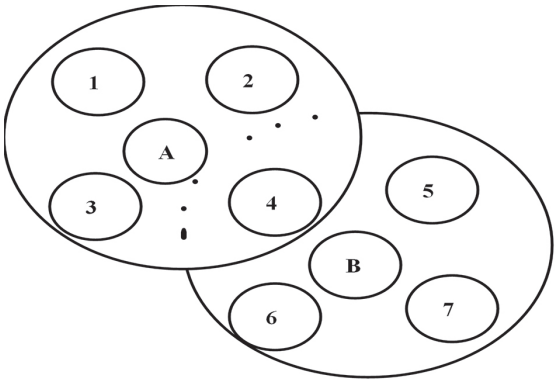

Fig. 1. RBS in the multihop network

Another drawback arises from implicit skew correction for all three nodes—R1, R4, and R7—during each time base conversion. Here, R4 listens to two sync messages and requires a series of timestamp conversions for clock skew calculation. If a node listens to more than two sync messages, this task becomes more challenging. Consequently, the RBS strategy does not adequately support large multihop networks for time synchronization, resulting in scalability issues.

2. FTSP in Multihop Network

The FTSP, on the other hand, utilizes a sender-receiver-based approach [20], supporting both single-hop and multihop networks for time synchronization. In this method, the root node transmits a sync message, and non-root nodes synchronize their clocks with their neighbors based on the root message. Each node, ex-

cluding the root node, utilizes timestamps from multiple neighbors to determine its local clock time and achieve synchronization. The synchronization process in a multihop FTSP relies on reference points established by broadcast messages periodically transmitted by the synchronization root node. However, this approach exhibits longer propagation times for leaf nodes in the network and may be susceptible to compromised nodes assuming the role of the root node and disseminating incorrect synchronization messages.

Another approach for multihop FTSP, as described in reference [30], accomplishes network synchronization without depending on an external time source. In this method, each node is allocated a unique identifier, and synchronization is attained through MAC layer timestamping. Clock skew estimation is performed by computing the average of multiple timestamp values, followed by the application of linear regression to estimate the clock offset. However, this approach entails increased timestamp overhead and is limited in handling small network traffic. Moreover, it demonstrates greater message complexity than does the RBS method.

3. The TPSN in the MultiHop Network

The TPSN employs a sender-receiver approach for time synchronization, comprising two phases: the establishment of a hierarchical topology followed by the synchronization phase. In the hierarchical topology phase, nodes at the ith level are connected with at least one node at the $(i-1)^{th}$ level. During the synchronization phase, child nodes synchronize with the root node at each level. Each pair of nodes is considered a root-child node, with the child node becoming the root for the subsequent node in the tree.

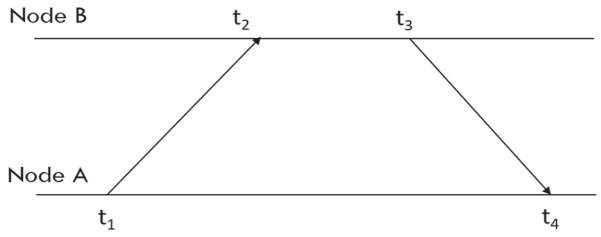

Fig. 2. TPSN sync message transmission

In Fig. 2, at time t_1, sender node A transmits a synchronization pulse packet to node B. Node B receives the packet at time t_2 and responds with an acknowledgment packet containing timestamp values t_1, t_2, and t_3. Node A acknowledges the receipt of the acknowledgment at time t_4. The clock offset is then calculated as $\Delta t = [(t_2 - t_1) - (t_4 - t_3)]/_2$, while the propagation delay is calculated as $d = [(t_2 - t_1) - (t_4 - t_3)]/2$.

In a multihop scenario, the TPSN adopts post facto synchronization, where nodes synchronize only as needed. Consequently, the receiver utilizes the TPSN to synchronize its clock after receiving a packet before forwarding it to the next hop. However, a drawback of this approach is that if any root node computes an incorrect offset during any point of the synchronization phase, this error will propagate down the tree. Furthermore, the TPSN transmits a large number of messages to synchronize a network, resulting in high data traffic.

Several aspects overlooked in the above approaches may hinder their implementation for high-level synchronization in multihop wireless networks. These aspects include ensuring quick network synchronization, which is particularly crucial in dense network environments where frequent synchronization is needed. Additionally, streamlining synchronization to minimize message overhead and enable multihop synchronization in a scalable manner, even when synchronization regions do not intersect, is essential. To address these challenges, we have implemented cluster-based and receiver–receiver-based approaches for time synchronization to support scalability and flexibility in multihop networks. The following section outlines clustering approaches applicable to multihop networks for time synchronization.

2.1. RELATED WORK FOR CLUSTERING IN WSNS AND THE IOT

Clustering offers a promising solution to address numerous challenges encountered in the IoT, including energy efficiency, scalability, and mobility. Its resemblance to wireless sensor networks (WSNs) makes it particularly advantageous for tackling these issues [40]. In cluster, a cluster head (CH) is a pivotal node serving as the central coordinator within a group of nodes. Cluster head plays a crucial role in organizing, managing, and optimizing the performance of a cluster of nodes.

In the implementation of time synchronization, the rapid exchange of synchronization messages among all network nodes without congestion is crucial. Clustering methods offer a solution by dividing the network into smaller regions, enabling simultaneous synchronization of message exchange among cluster nodes via cluster heads (CH). This approach enhances efficiency in terms of parallel processing, fault tolerance, and organizing dense networks effectively. Several clustering algorithms have been proposed to partition networks into smaller clusters based on criteria such as distance, link quality, and path.

- Clustering Methods in WSNs

In wireless sensor networks (WSNs), clustering algorithms are categorized as centralized or distributed approaches. Examples of clustering algorithms include the following:

In low-energy adaptive clustering hierarchy (LEACH) [41], nodes calculate their likelihood of becoming cluster heads (CHs) and broadcast them, with each node selecting the cluster requiring the least amount of communication energy to reach the CH. However, the CH distribution may not be uniform, leading to uneven energy dissipa-

tion. LEACH-C [42] selects a CH based on energy information and load balancing, with a cluster setup performed by the base station. While ensuring load balancing, this approach is less scalable and consumes more energy. The hybrid energy efficient distributed (HEED) method [43] selects a CH based on the residual energy and the energy required for intra-cluster communication. This ensures a uniform CH distribution and applies load balancing. The energy efficient clustering scheme (EECS) [44] selects a CH based on the maximum residual energy and minimum distance, extending the cluster formation approach of LEACH. However, CH deaths may occur due to congestion near the base station. The linked cluster algorithm (LCA) [45] selects the CH based on the highest ID among the node neighbors. Despite identity-based selection, nodes exhibit low energy efficiency, and these algorithms are designed for homogeneous networks and lack support for node mobility and data aggregation at the CH.

However, for IoT networks, which require support for data aggregation and mobility, these algorithms are not suitable, necessitating the development of new clustering strategies.

- Clustering Methods in IoT Networks

Efficient operation of IoT applications requires energy efficiency, low communication overhead, mobility support, data aggregation, and compatibility with heterogeneous environments. Several clustering algorithms address these requirements:

The heuristic clustering algorithm [46] forms clusters based on the number of neighboring nodes and residual energy. The node with the maximum residual energy is selected as the CH, facilitating one-hop communication but requiring re-clustering if the topology changes, leading to increased energy consumption. The graph-based clustering algorithm [47] uses graph theory to form clusters, selecting the vertex with the maximum degree and maximal residual energy as the CH. It supports node mobility with energy efficiency. The hybrid energy-aware clustered protocol for heterogeneous IoT [48] enhances node energy utilization and prolongs network lifetime. CH selection is based on various weighted election probabilities, such as residual energy. Cluster formation adjusts the number of CHs and the cluster length to optimize connectivity and energy utilization in multihop networks.

In summary, clustering approaches for IoT networks minimize communication overhead and maximize node connectivity, improving the efficiency of time synchronization algorithms while prolonging network lifetime and enhancing energy utilization.

3. IMPROVISED RBS ALGORITHM USING CLUSTER

Our research endeavors center on the development of time synchronization algorithms capable of withstanding significant differences in clock drift and offset, which is particularly relevant for extensive IoT network deployments. It has been noted that minimizing communication distance aids in reducing clock drift and offset. To achieve this, before initiating the synchronization process, an RSSI-based clustering approach is employed to partition the large network into smaller clusters. This allows reference nodes to communicate directly with their nearest cluster-head nodes, which then handle synchronization among the nodes within their respective clusters. This localized approach minimizes the energy expenditure required for time synchronization across the entire network. By reducing communication distance and overhead, this approach reduces power consumption and effectively reduces delays that can occur at various stages of the synchronization process, thereby minimizing synchronization errors. In this scheme, each cluster-head node serves as the reference node for its cluster nodes.

In the implementation of time synchronization, it is imperative to ensure rapid dissemination of synchronization messages across all network nodes without causing congestion. Various clustering algorithms have been suggested for partitioning networks into smaller clusters based on diverse criteria. Drawing from extensive surveying and identifying research gaps, we advocate for a clustering approach tailored for IoT network applications based on the received signal strength indicator (RSSI). This approach aims to minimize power consumption, enhance the packet reception ratio, and enable data aggregation at cluster heads within heterogeneous IoT networks.

To address the challenges posed by dynamic topology alterations, node failures, and scalability issues, we devised a strategy wherein the network undergoes re-clustering after each synchronization interval. This adaptive approach ensures resilience to changing network conditions while bolstering the system's robustness and scalability. Our time synchronization methodology for multihop IoT networks is structured into two phases: Phase-1 involves cluster formation through cluster-head selection, while Phase-2 encompasses the synchronization method between nodes. In the subsequent sections, we elaborate on our novel cluster approach followed by the time synchronization methodology tailored for multihop IoT networks.

- Cluster Algorithm for IoT network
  - * Phase-1: Cluster Formation

Cluster Head Selection: The selection of cluster heads (CHs) is achieved by analyzing the radio signal strength indicator (RSSI) of each node within the network. This entails measuring the power present in signals transmitted by each node. By utilizing the received signals from neighboring nodes, each node maintains a count of the number of neighboring nodes (denoted as 'm') covered by the RSSI. The node with the highest value of 'm' is designated the CH. This selection process continues until all nodes are encompassed within the network.

Clustering: During this phase, each CH identifies its cluster nodes based on a predefined RSSI threshold value. Nodes are compared with the threshold value, and if their RSSI value is lower than the threshold, they are included as cluster nodes for that particular cluster head. This process iterates until all nodes are assigned to one of the clusters within the network. The algorithm outlining the cluster formation process is depicted in Algorithm 1.

Algorithm 1. Cluster Formation

Input: (N^i is the i^{th} node in the network), L_i = list of neighbor nodes, M= Maximum no. of nodes covered in one cluster, m= No. of clusters.

Output: Create m no. of clusters.

Initialize all the nodes N_i, where i=1,2, 3…n. in a network.

For each node N^i

$RN_i \leftarrow RSSI\ (N_i)$, i=i+1

(Sort RN_i, choose nearest node in L_i)

For each node RN_i

If $RN_i < RN_i$+1

$L_i < RN_i$

End.

For each node N^i

//From the sorted list L_i, the nodes which covers maximum no. of Neighbor nodes are declared as cluster heads.

//Choose other nodes in the range of CH_i as member nodes in cluster C_i, where $i \in N_i$. Such that M<=m, where m<=N/2

In case of conflict, where the node may fall in more than one clusters, then the node joins the cluster having minimum distance with the CH_i.

Repeat steps after each Synchronization Interval S_i.

- Proposed Cluster-based Time Synchronization Approach for Multihop IoT

Time synchronization commences after the completion of cluster formation as outlined in Algorithm 1 during Phase-1. Phase-2 encompasses the synchronization process among the nodes and is logically subdivided into two stages.

Phase 2(a) entails Algorithm 2(a), synchronization among cluster head nodes with a designated reference node, referred to as inter-cluster synchronization. This phase is crucial for rectifying offset discrepancies by initially addressing the time differential between the primary synchronized reference node and the cluster heads of other clusters within the network. To achieve this, the reference node transmits beacon messages to the cluster heads, recording the timestamp of these beacon messages from the synchronized reference node. Subsequently, all nodes exchange their local time information with one another and estimate their respective clock offsets.

Phase 2(b) involves synchronization among cluster nodes as in Algorithm 2(b) with their respective cluster heads, termed as intra-cluster synchronization.

Algorithm 2(a). Inter-Cluster Synchronization

Assumption: Cluster formed using algorithm-1

Initialization: S- Reference node

CH_i- i^{th} Cluster head node, C_i - i^{th} Cluster node

LT_i - i^{th} local timestamp of i^{th} cluster head node.

OCH_{ij} - Clock Offset between nodes i and j

CHT_i – Adjusted new clock value

Reference Node S: broadcast () //Broadcast Beacon

For each CH_i

Call broadcast_receive() // CH Receive broadcast message

LT_i = Clock_time() //Record local timestamp of received message

Call unicast (LT_i, CH_j) // Send LTi to other CH_j nodes where $i \neq j$.

End

For each CH_i

$$OCH_{ij} = \frac{\sum LT_i - LT_j}{n} \quad (1)$$

//Compute clock offset at each cluster head node. Where, n = no. of received timestamp from neighbor cluster heads CH_j. m<=N/2, and M depends on number of nodes in the network.

$$CHT_i = OCH_{ij} + LT_i \quad (2)$$

// Compute new clock value CHT_i, and synchronize with reference node S.

End.

Algorithm 2(b). Intra-Cluster Synchronization

//In continuation of algorithm 2(a)

For each node CH_i

Call multicast (CHT_i, C_i)

// CH_i nodes multicast their updated time value CHT_i to their cluster nodes C_i

Call multicast_receive()

CT_i = Clock_time()

// Each C_i records timestamp of received message as CT_i and unicast to other nodes C_j, ($i \neq j$)

End

For each C_i node

Call unicast (CT_i, C_j) // Send CT_i to other C_j nodes where $i \neq j$.

// Compute clock offset

$$OC_{ij} = \frac{\sum CT_i - CT_j}{CN} \quad (3)$$

// where CN = No. of nodes within cluster. And OC_{ij} is an offset between two cluster nodes in one cluster.

// Cluster node computes new clock value

$$TC_i = OC_{ij} + CT_i \quad (4)$$

// Where, TC_i is the new adjusted clock time of ith cluster node in one cluster

End.

- Key Features of the Proposed Cluster-Based Algorithm

Minimizes communication distance for synchronization message transmission, leading to potential reductions in power consumption and synchronization errors. Decreases collision occurrences owing to smaller cluster sizes and reduced message overhead. Enhances network reliability and scalability through the implementation of cluster-based time synchronization. Reduces communication distance for sync-message transmission, thereby lowering the power consumption of the reference node and the network overall. Improves the probability of packet reception ratio (PRR) for individual nodes.

4. IMPLEMENTATION AND RESULTS

4.1. IMPLEMENTATION

The time synchronization algorithm for a multihop IoT network is implemented using Contiki OS with the Cooja emulator [49]. Initially, all nodes are initialized with random startup times. The proposed cluster-based algorithm is compared with the existing RBS algorithm in terms of synchronization accuracy and power consumption on the same platform. Emulation parameters and configurations are summarized in Table 1.

Table 1. Emulation Parameters on Cooja

Parameters	Values
Number of Nodes	6 / 12 / 20
Type of Network	Multihop
Emulation Time	15min
Synchronization Interval	30 s
OS	Contiki
Topology	Random
Radio	CC2420 (2.4 GHz)
MAC / RDC layer Protocol	CSMA with ContikiMAC
Network Stack	Rime
Radio Medium	UDGM
Channel Check Rate	8Hz
Mote Type	Tmote Sky

To establish a multihop environment, the first experiment involves generating results with a random topology consisting of 10 nodes. Node id-1 is designated as the reference node for other nodes in the network. Following Algorithm 1, all nodes except node id-1 calculate their RSSI values and identify the number of nodes within their communication range. After selecting cluster heads in the network and forming clusters, the reference node begins broadcasting beacons throughout the network. Only cluster head (CH) nodes receive these beacons and exchange their local time of reception with other CH nodes in the network, as outlined in Algorithm 2, for synchronization. After inter-cluster synchronization of CHs with the reference node, cluster heads proceed with intra-cluster synchronization within their respective clusters.

4.2. RESULTS

To assess the performance of the proposed cluster-based algorithm in terms of time synchronization error and power consumption, it is implemented on different platforms. Specifically, the proposed cluster-based multihop RBS algorithm is executed on Sky motes [50], with configurations detailed in Table 1. Power consumption is evaluated at each node in the network.

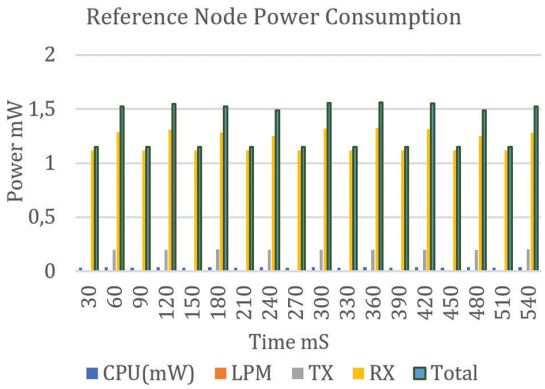

Fig. 3(a). Power consumption of reference node in proposed cluster-based RBS algorithm for multihop network

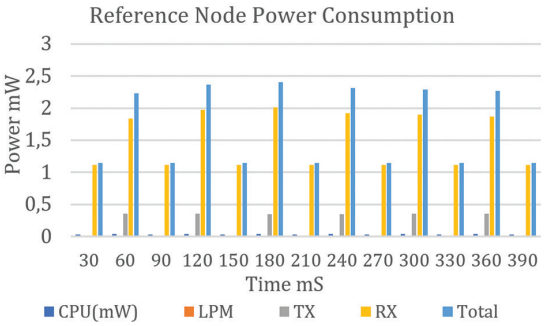

Fig. 3(b). Power consumption of reference node in RBS algorithm for multihop network

The sink node (Reference node) broadcasts beacon messages at specified regular intervals to synchronize cluster heads. The algorithm is tested under various scenarios, including random node startup times and cluster head node failures, to ensure continued network connectivity, wherein remaining nodes reform the connected network and reassign cluster heads as

per Phase-1. The results, depicted in Fig. 3(a), illustrate the power consumption of the reference node in the proposed cluster-based RBS algorithm, while Fig. 3(b) displays the power consumption of the reference node in the RBS algorithm. Power consumption is assessed using the power tracer tool within the Cooja emulator. The formula utilized for calculating power consumption at each node is as follows [50].

$$\text{CPU}_{\text{power}} = \frac{(\text{Energest\_Value} \times \text{Current} \times \text{Voltage})}{(\text{Number of ticks per second} \times \text{Runtime})} \quad (5)$$

Power consumption, data gathering, and analysis are conducted at various stages of the node lifetime, including transmission power, receiving power, CPU power, and during low power mode. To facilitate a fair comparison between both algorithms, identical configurations (as per Table 1) are applied to assess performance. Graphical analysis reveals that the reference node's power consumption in the RBS algorithm is approximately 2.5mW, which is higher compared to the 1.5mW power consumption observed for the proposed cluster-based algorithm. Estimated percentage of reduction in power consumption using proposed cluster-based approach as follows:

$$\%\text{Reduction} = \frac{\text{Power Consumption}_{\text{RBS}} - \text{Power Consumption}_{\text{Cluster based\_RBS}}}{\text{Power Consumption}_{\text{RBS}}} \times 100\ \% \quad (6)$$

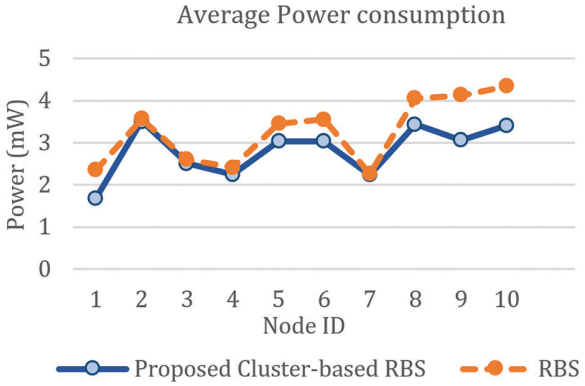

Fig. 4. Average power consumption at each node of RBS and proposed Cluster-based Improvised RBS multihop IoT network

Fig. 4 illustrates the average power consumption at each node within the multihop network. It is evident that the overall power consumption of the multihop network with the RBS algorithm is approximately greater than or equal to 4.5 mW, whereas with the proposed cluster-based approach, it is approximately less than 3.5 mW. This highlights a notable enhancement in power efficiency with the proposed cluster-based approach for time synchronization. Additionally, it is observed that nodes 2 and 3 exhibit nearly identical power consumption for both algorithms. This similarity arises because both nodes fall within the interference range of each other and of node 1, necessitating transmission to more than one cluster, resulting in increased power consumption.

The subsequent step involves calculating clock offset and clock skew to determine the synchronization error between two nodes. Clock offset is estimated using equation (1) for inter-cluster nodes and equation (3) for intra-cluster nodes. Synchronization error is assessed using the linear regression method, where the least squares method is employed to predict the nearest correct time value, minimizing the error sum of squares as per the principle of least squares method [51]. The following formula is utilized to calculate predicated time say P_t and Synchronization Error say S_e for this purpose:

$$P_t = \frac{((x-\bar{x})\times(y-\bar{y}))}{(x-\bar{x})^2} \times x + \delta \quad (7)$$

Where x denotes the timestamp value of the sent beacon message, y represents the timestamp value of the received beacon message, and δ denotes the clock skew, representing the difference between two clock frequencies.

$$S_e = A_t - P_t \quad (8)$$

Where A_t denoted Actual Received Time. After determining the absolute synchronization error between two nodes, the subsequent step involves calculating the delta error. The proposed cluster-based approach effectively minimizes the overall power consumption of each node. Furthermore, using equation (6) the average power consumption of the reference node has been notably reduced by 40%, consequently improving the overall performance and lifetime of the network. Emulation results also indicate approximately 30% improvement in minimizing the average power consumption of each node in the network, apart from the reference node.

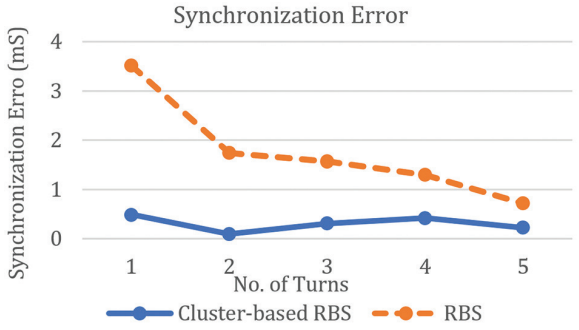

Fig. 5. Synchronization Error in RBS and Cluster-based RBS

Fig. 5 depict the synchronization error graphs for RBS and RBS with clustering, calculated using the linear regression method. In the case of RBS, the synchronization error ranges from 0.5 mS to 4 mS. However, with the incorporation of clustering, there is a reduction in the synchronization error rate, ranging from 0.5 mS to 0 mS. To mitigate the overall synchronization error, the re-synchronization interval is determined by assessing the absolute error between the estimated offset and the corrected offset at each offset synchronization point.

Consequently, the overall synchronization error gradually diminishes, tending toward zero.

Compared to the existing RBS approach, the proposed cluster-based approach demonstrates an average 78% improvement in minimizing node synchronization error.

4.3. IMPACT OF DIFFERENT TOPOLOGIES ON TIME SYNCHRONIZATION

Both synchronization algorithms, namely RBS and the proposed cluster-based approach, underwent testing using an experimental setup outlined in Table 1. Three distinct network topologies - Random, Ellipse, and Linear - were tested for each synchronization algorithm, specifically designed for multihop networks. Tables 2, 3, and 4 present the performance of both approaches for time synchronization across random, ellipse, and linear topologies, respectively. The evaluation includes synchronization error and power consumption for various numbers of nodes within the system. An Average Synchronization Error, measured in milliseconds, indicates the level of synchronization accuracy for each configuration, with smaller values indicating preferable synchronization. Standard Deviation measures dispersion in synchronization errors; lower values indicate consistent errors, ensuring system stability [51]. Here, Standard Error helps to understand the likely range within which the true mean synchronization error falls.

Table 2. Synchronization Error and Power Consumption for Random Topology

Synchronization Approach	RBS	Proposed Approach (with Cluster)	RBS	Proposed Approach (with Cluster)	RBS	Proposed Approach (with Cluster)
No. of Nodes		6		12		20
Average (mS) (Synchronization Error)	1.53	0.36	1.46	0.3	3.16	0.64
Standard Deviation	1.30	0.28	0.98	0.15	1.72	0.30
Standard Error of (Synchronization Error) (mS)	1.12	0.39	0.44	0.07	0.77	0.13
Average Power Consumption (mW)	2.33	1.27	3.3	1.00	1.15	2.8

Table 3. Synchronization Error and Power Consumption for Ellipse Topology

Synchronization Approach	RBS	Proposed Approach (with Cluster)	RBS	Proposed Approach (with Cluster)	RBS	Proposed Approach (with Cluster)
No. of Nodes		6		12		20
Average (mS) (Synchronization Error)	2.52	1.96	9.12	5.32	9.6	7.91
Standard Deviation	1.29	0.88	6.18	2.96	6.88	5.6
Mean Synchronization Error (Standard Error) (mS)	1.02	0.39	2.77	1.32	3.07	2.5
Average Power Consumption (mW)	2.76	1.98	3.24	2.46	4.6	2.1

Table 4. Synchronization Error and Power Consumption for Linear Topology

Synchronization Approach	RBS	Proposed Approach (with Cluster)	RBS	Proposed Approach (with Cluster)	RBS	Proposed Approach (with Cluster)
Number of Nodes		6		12		20
Average (mS) (Synchronization Error)	3.12	2.48	9.12	7.20	11.04	8.08
Standard Deviation	2.87	2.59	6.81	3.03	9.71	7.56
Mean Synchronization Error (Standard Error) (mS)	1.28	1.16	3.05	1.36	4.34	3.38
Average Power Consumption (mW)	1.73	1.4	3.28	2.93	3.36	2.11

The results displayed in Tables 2, 3, and 4 show that the proposed clustering-based approach generally outperforms the baseline RBS approach in terms of synchronization error. Across all node configurations (6, 12, and 20 nodes), the proposed approach consistently exhibits a lower average synchronization error. Moreover, the standard deviation in the proposed cluster-based approach is generally lower, indicating more consistent and stable synchronization errors across different trials. In summary, these results indicate that the proposed cluster-based synchronization approach achieves lower synchronization errors and greater power efficiency across diverse numbers of nodes in various topologies than does the baseline RBS approach. Additionally, the results suggest that the random topology yields optimized results compared to the ellipse and linear topologies, which represent the worst-case scenarios.

4.4. IMPACT OF HOP DISTANCE ON TIME SYNCHRONIZATION

Table 5 illustrates the performance of the proposed cluster-based RBS multihop algorithm implemented using random and linear topologies.

Table 5. Results of synchronization error for the implementation of the proposed cluster-based RBS for multiple hops using random and linear topologies (absolute values)

Hop Distance	Random Topology	Linear Topology
	Average Error (mS)	Average Error (mS)
1-hop	0.33	0.24
2-hop	0.47	9.48
3-hop	0.93	10.36
4-hop	1.29	8.72
5-hop	1.45	8.75

In the linear topology implementation, each cluster head has only one cluster node within its range, representing a worst-case scenario for synchronization accuracy at each node. Conversely, the random topology is considered the best-case scenario for evaluating performance based on the number of hops in a multihop network.

As evidenced by the smaller variations in average error values depicted in Fig. 6, in the linear topology, there are greater variations in average errors, particularly for larger hop distances. Conversely, the random topology demonstrates relatively stable and consistent synchronization performance across different hop distances. Consequently, the proposed cluster-based RBS algorithm for multihop networks utilizing a random topology outperforms other implementations.

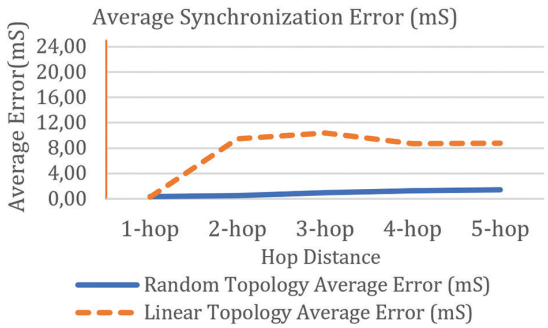

Fig. 6. Synchronization error in the proposed cluster-based RBS with increasing number of hop distances

4.5. APPLICATIONS OF PROPOSED WORK

Our research findings offer valuable applications for real-time water quality monitoring and controlling system [52] requires accurate time synchronization to enable solenoid valves to react promptly to sensor data by opening and closing as needed. In this system, nodes exchange beacon messages with adjacent nodes, periodically adjusting their clocks to maintain synchronization. Without effective time synchronization, there is a potential risk of distributing contaminated water which cannot be use for drinking purpose. Our approach minimizes communication distance, thereby conserving power which prevents node failure and ensuring reliable real-time operations. Similarly, our research findings are relevant to Cyber-Physical Systems (CPS) [53] requiring time synchronization, ensuring precise data timestamping among system components and maintains event ordering. Our synchronization method guarantees accurate timestamping and facilitates sequence of event by reducing communication overhead between server node and client nodes. In Patient's Real-time Health Monitoring System, precise time synchronization facilitates real-time recording and transmission of patient data, including vital signs and medication schedules. Synchronized data transmission minimizes network congestion and conserves energy, extending the battery life of wearable medical devices employed in patient monitoring. Furthermore, timely and accurate data transmission supported by time synchronization enhances patient safety by enabling healthcare providers to promptly identify and address critical situations, thereby reducing the likelihood of medical errors and adverse outcomes.

Having implemented our proposed approach using the Cooja emulator with sky motes, we observed consistent results comparable to real-world sky mote scenarios, with minimal discrepancies.

5. CONCLUSION

While numerous time synchronization algorithms exist for wireless sensor networks (WSNs), there is no elaborate work done on IoT networks, where explicit challenge is power conservation. They often prove less effective in IoT networks due to constraints such as low power availability, limited memory, and unreliable crystal clocks inherent in IoT devices. In response to these challenges, a novel approach to time synchronization in IoT networks has been proposed. By implementing RSSI-based clustering method to segment the network into smaller regions. By leveraging RSSI values, the proposed algorithm aims to minimize communication distance, reduce power consumption, enhance the packet reception ratio, and enhance synchronization accuracy. The incorporation of an adaptive re-clustering strategy after each synchronization interval is another novel aspect of this work. This adaptive approach ensures power conservation and thus, there is less chances of node failure, resilience to changing network conditions, bolstering the system's robustness and scalability. The proposed algorithm offers flexibility and adaptability crucial for IoT deployments. The cluster-based approach aims to minimize the power consumption of the reference node and the overall network while also reducing synchronization errors to ensure accurate event ordering.

A comprehensive overview of time synchronization approaches for multihop networks has been presented. Furthermore, we analyzed three key performance metrics in the multihop IoT network for comparison: power consumption, synchronization error, and scalability. The study demonstrates the effectiveness and robustness of the cluster-based approach in different deployment scenarios such as diverse network to-

pologies (random, ellipse, linear) and hop distances. The emulation results demonstrate that the proposed cluster-based approach has minimized the power consumption by 40% of the reference node and 30% of the overall network. A significant 78% reduction in synchronization error is achieved. Building upon reference-broadcast synchronization principles, this study explores an alternative method for synchronizing multihop networks, offering enhanced precision, flexibility, and resource efficiency compared to traditional algorithms. Through the cluster-based RBS approach, the communication distance between nodes involved in time synchronization is minimized, resulting in reduced propagation delay and synchronization errors within the cluster nodes.

6. REFERENCES:

[1] N. Dalwadi, M. Padole, "An Insight into Time Synchronization Algorithms in IoT", Data, Engineering and Applications, Springer, 2019, pp. 285-296.

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, "Internet of Things for Smart Cities", IEEE Internet of Things Journal, Vol. 1, No. 1, 2014, pp. 22-32.

[3] X. Liu, Z. Qin, Y. Gao, J. A. McCann, "Resource Allocation in Wireless Powered IoT Networks", IEEE Internet of Things Journal, Vol. 6, No. 3, 2019, pp. 4935-4945.

[4] G. Xu, W. Shen, X. Wang, "Applications of Wireless Sensor Networks in Marine Environment Monitoring: A survey", Sensors, Vol. 14, No. 9, 2014, pp. 16932-16954.

[5] E. Xu, Z. Ding, S. Dasgupta, "Target Tracking and Mobile Sensor Navigation in Wireless Sensor Networks", IEEE Transactions on Mobile Computing, Vol. 12, No. 1, 2013, pp. 177-186.

[6] T. Schmid, Z. Charbiwala, J. Friedman, Y. Cho, M. Srivastava, "Exploiting Manufacturing Variations for Compensating Environment-induced Clock Drift in Time Synchronization", Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS, Annapolis, MD, USA, 2-6 June 2008, pp. 97-108.

[7] F. Tirado-Andrés, A. Araujo, "Performance of Clock Sources and their Influence on Time Synchronization in Wireless Sensor Networks", International Journal of Distributed Sensor Networks, Vol. 15, No. 9, 2019.

[8] E. Coca, V. Popa, "A Practical Solution for Time Synchronization in Wireless Sensor Networks", Advances in Electrical and Computer Engineering, Vol. 12, No. 4, 2012, pp. 57-62.

[9] D. Mills, "Internet Time Synchronization: The Network Time Protocol", IEEE Transactions on Communications, Vol. 39, No. 10, 1991, pp. 1482-1493.

[10] D. Mills, U. Delaware, J. Martin, J. Burbank, W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", IETF, RFC 5905, June 2010, http://www.rfceditor.org/rfc/rfc5905.txt (accessed: 2024)

[11] A. N. Novick, M. A. Lombardi, "Practical Limitations of NTP Time Transfer", Proceedings of the Joint Conference of the IEEE International Frequency Control Symposium & the European Frequency and Time Forum, Denver, USA, 2015, pp. 570-574.

[12] J. Elson, K. Romer, "Wireless Sensor Networks: A New Regime for Time Synchronization", Proceedings of the First Workshop on Hot Topics In Networks (HotNets-I), Princeton, NJ, USA, 28-29 October 2002.

[13] K. Fan, Z. Shi, R. Su, Y. Bai, P. Huang, K. Zhang, H. Li, Y. Yang, "Blockchain-Based Trust Management for Verifiable Time Synchronization Service in IoT", Peer-to-Peer Networking and Applications, Vol. 15, No. 2, 2022, pp. 1152-1162.

[14] A. Mahmood, G. Gaderer, H. Trsek, S. Schwalowsky, N. Kero, "Toward high accuracy in IEEE 802.11 based clock synchronization using PTP", Proceedings of the International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication, Munich, Germany 2011, pp. 13-18.

[15] K. Balakrishnan, R. Dhanalakshmi, B. B. Sinha, R. Gopalakrishnan, "Clock Synchronization in Industrial Internet of Things and Potential Works in Precision Time Protocol: Review, Challenges and Future Directions", International Journal of Cognitive Computing in Engineering, Vol. 4, 2023, pp. 205-219.

[16] G. Liu, S. Yan, L. Mao, "Receiver-Only Based Time Synchronization Under Exponential Delays in Un-

derwater Wireless Sensor Networks", IEEE Internet of Things Journal, Vol. 7, No. 10, 2020, pp. 9995-10009.

[17] M. AbdelRaheem, M. Hassan, U.S. Mohammed, A. A. Nassr, "Design and Implementation of a Synchronized IoT-based Structural Health Monitoring System", Internet of Things, Vol. 20, 2022, p. 100639.

[18] G. Cena, I. Bertolotti, S. Scanzio, A. Valenzano, C. Zunino, "Synchronize your Watches: Part I: General-Purpose Solutions for Distributed Real-Time Control", IEEE Industrial Electronics Magazine, Vol. 7, No. 1, 2013, pp. 18-29.

[19] Y. Wu, Q. Chaudhari, E. Serpedin, "Clock Synchronization of Wireless Sensor Networks", IEEE Signal Processing Magazine, Vol. 28, No. 1, 2011, pp. 124-138.

[20] M. Maroti, B. Kusy, G. Simon, A. Ledeczi, "The Flooding Time Synchronization Protocol", Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, New York, NY, USA, 2004, pp. 39-49.

[21] J. V Greunen, J. Rabaey, "Lightweight Time Synchronization for Sensor Networks", Proceedings of the 2nd ACM International Workshop on Wireless Sensor Networks and Applications, New York, NY, USA, September 2003.

[22] S. Ganeriwal, R. Kumar, M. B. Srivastava, "Timing-sync Protocol for Sensor Networks", Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, Los Angeles, CA, USA, 5-7 November 2003, pp. 138-149.

[23] K. Yildirim, A. Kantarci, "Time Synchronization based on Slow-Flooding in Wireless Sensor Networks", IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 1, 2014, pp. 244-253.

[24] J. Elson, L. Girod, D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts", ACM SIGOPS Operating Systems Review, Vol. 36, No. SI, 2002, pp. 147-163.

[25] W. Su, I. F. Akyildiz, "Time Diffusion Synchronization Protocol for Wireless Sensor Networks", IEEE/ACM Transactions on Networking, Vol. 13, No. 2, 2005, pp. 384-397.

[26] H. Kim, D. Kim, and S.-e. Yoo, "Cluster-Based Hierarchical Time Synchronization for Multihop Wireless Sensor Networks", Proceedings of 20th International Conference on Advanced Information Networking and Applications, Vienna, Austria, April 2006, pp. 318-322.

[27] B. J. Choi, H. Liang, X. Shen, W. Zhuang, "DCS: Distributed Asynchronous Clock Synchronization in Delay Tolerant Networks", IEEE Transactions on Parallel and Distributed Systems, Vol. 23, No. 3, 2012, pp. 491-504.

[28] M. Leng, Y. C. Wu, "Distributed Clock Synchronization for Wireless Sensor Networks using Belief Propagation", IEEE Transactions on Signal Processing, Vol. 59, No. 11, 2011, pp. 5404-5414.

[29] M. Akar, R. Shorten, "Distributed Probabilistic Synchronization Algorithms for Communication Networks", IEEE Transactions on Automatic Control, Vol. 53, No. 1, 2008, pp. 389-393.

[30] M. Maroti, B. Kusy, G. Simon, A. Ledeczi, "Robust Multi-Hop Time Synchronization in Sensor Networks", Proceedings of the International Conference on Wireless Networks, Las Vegas, NV, USA, 21-24 June 2004, pp. 454-460.

[31] N. Shivaraman, P. Schuster, S. Ramanathan, A. Easwaran, S. Steinhorst, "Cluster-Based Network Time Synchronization for Resilience with Energy Efficiency", Proceedings of the IEEE Real-Time Systems Symposium, Dortmund, Germany, 7-10 December 2021, pp. 149-161.

[32] S. Palchaudhuri, A. K. Saha, D. B. Johnsin, "Adaptive Clock Synchronization in Sensor Networks", Proceedings of the Third International Symposium on Information Processing in Sensor Networks, Berkeley, CA, USA, 26-27 April 2004, pp. 340-348.

[33] H. Kim, D. Kim, S. Yoo, "Cluster-Based Hierarchical Time Synchronization for Multihop Wireless Sensor Networks", Proceedings of the 20th International Conference on Advanced Information Networking and Applications, Vienna, Austria, 18-20 April 2006.

[34] Y. Sung, S. Lee, M. Lee, "A Multi-Hop Clustering Mechanism for Scalable IoT Networks", Sensors, Vol. 18, No. 4, 2018, p. 961.

[35] R. Jha, P. Gupta, "Clock Synchronization in IoT Network Using Cloud Computing", Wireless Personal Communications, Vol. 97, 2017, pp. 6469-6481.

[36] G. Gautam, N. Chand, "A Novel Cluster Based Time Synchronization Technique for Wireless Sensor Networks", Wireless Sensor Network, Vol. 9, No. 5, 2017, pp. 145-165.

[37] Z. Wang, P. Zeng, M. Zhou, D. Li, J. Wang, "Cluster-Based Maximum Consensus Time Synchronization for Industrial Wireless Sensor Networks", Sensors, Vol. 17, No. 1, 2017, p. 141.

[38] K. Mehta, Y. Kumar, A. Aayushi, "Enhancing Time Synchronization for Home Automation Systems", ECS Transactions, Vol. 107, No. 1, 2022, pp. 6197-6208.

[39] D. Gao, Y. Liu, B. Hu, L. Wang, W. Chen, Y. Chen, T. He, "Time Synchronization based on Cross-Technology Communication for IoT Networks", IEEE Internet of Things Journal, Vol. 10, No. 22, 2023, pp. 19753-19764.

[40] S. Sholla, S. Kaur, G. Rasool, R. Naaz Mir and M. A. Chishti, "Clustering Internet of Things: A Review", Journal of Science and Technology: Issue on Information and Communications Technology, Vol. 3, No. 2, 2017, pp. 21-27.

[41] W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy- Efficient Communication Protocol for Wireless Microsensor Networks", Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Maui, HI, USA, 7 January 2000.

[42] W. Xinhua, S. Wang, "Performance Comparison of LEACH and LEACH-C Protocols by NS2", Proceedings of the 9th International Symposium on Distributed Computing and Applications to Business Engineering and Science, Hong Kong, China, 10-12 August 2010, pp. 254-258.

[43] O. Younis, S. Fahmy, "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks", IEEE Transactions on Mobile Computing, Vol. 3, No. 4, 2004, pp. 366-379.

[44] M. Ye, C. Li, G. Chen, J. Wu, "EECS: An Energy Efficient Clustering Scheme in Wireless Sensor Networks", Proceedings of the 24th IEEE International Performance, Computing, and Communications Conference, Phoenix, AZ, USA, 7-9 April 2005, pp. 535-540.

[45] P. Kumarawadu, D. J. Dechene, M. Luccini, A. Sauer, "Algorithms for Node Clustering in Wireless Sensor Networks: A Survey", Proceedings of the 4th International Conference on Information and Automation for Sustainability, Colombo, Sri Lanka, 12-14 December 2008, pp. 295-300.

[46] J. S. Kumar, M. A. Zaveri," Clustering Approaches for Pragmatic Two-Layer IoT Architecture", Wireless Communications and Mobile Computing, Vol. 21, 2018, pp. 1-16.

[47] J. S. Kumar, M. A. Zaveri, "Hierarchical Clustering for Dynamic and Heterogeneous Internet of Things", Proceedings of the 6th International Conference on Advances in Computing & Communications, Cochin, India, 6-8 September 2016, pp. 276-282.

[48] A. R. Sadek, "Hybrid Energy Aware Clustered Protocol for IoT Heterogeneous Network", Future Computing and Informatics Journal, Vol. 3, No. 2, 2018, pp. 166-177.

[49] Contiki Operating System home page, http://www.contiki-os.org (accessed: 2022)

[50] N. Dalwadi, M. Padole, "Performance Analysis of Wireless Motes in IoT", ICT Analysis and Applications, Lecture Notes in Networks and Systems, Springer, Vol. 517, 2023, pp. 383-397.

[51] K. Molugaram, G. Rao, S. "Analysis of Time Series, Statistical Techniques for Transportation Engineering", Elsevier, 2017, pp. 463-489.

[52] N. Dalwadi, M. Padole, "The Internet of Things Based Water Quality Monitoring and Control", Proceedings of Smart Systems and IoT: Innovations in Computing, Smart Innovation, Systems and Technologies, Springer, Vol. 141, 2020, pp. 409-417.

[53] E. Lisova, E. Uhlemann, J. Åkerberg, M. Björkman, "Monitoring of Clock Synchronization in Cyber-Physical Systems: A Sensitivity Analysis", Proceedings of the International Conference on Internet of Things, Embedded Systems and Communications, Gafsa, Tunisia, 20-22 October 2017, pp. 134-139.