applied sciences

MDPI

Review

A Survey of Secure Time Synchronization

Ying Weng * and Yiming Zhang

School of Computer Science, Faculty of Science and Engineering, University of Nottingham Ningbo China, Ningbo 315100, China; yiming.zhang2@nottingham.edu.cn
* Correspondence: ying.weng@nottingham.edu.cn

Abstract: Today, the use of wireless sensor networks has grown rapidly; however, wireless sensor networks are prone to receiving cyber-physical attacks. Time synchronization is a fundamental requirement for protocols in wired and wireless sensor network applications; hence, secure time synchronization is also crucial. This paper presents an introduction to time synchronization, including the concepts, challenges, and requirements of time synchronization protocols. The scope of the paper includes both software- and hardware-based protocols. Then, different time synchronization methods are analyzed. Moreover, research progress in secure time synchronization is reviewed. The survey also discusses the weaknesses of current secure time synchronization protocols and provides suggestions for future research directions. This survey aims to highlight research progress and trends in time synchronization and secure time synchronization.

Keywords: time synchronization; security; wireless sensor networks; wired networks

check for updates

Citation: Weng, Y.; Zhang, Y. A Survey of Secure Time Synchronization. *Appl. Sci.* **2023**, *13*, 3923. https://doi.org/10.3390/app13063923

Academic Editor: Juan Francisco De Paz Santana

Received: 31 January 2023
Revised: 6 March 2023
Accepted: 14 March 2023
Published: 20 March 2023

1. Introduction

During industrial development, wireless sensor networks (WSNs) have rapidly replaced wired networks due to their advantages [1]. A WSN is a combination of spatially distributed nodes organized in a cooperative network to monitor and record physical and environmental conditions [2]. Nowadays, sensor networks are used in many different aspects of our life such as proving health care for elderly, surveillance, emergency disaster relief, and intelligence gathering in the battlefield [3].

Switching from wired networks to wireless networks has raised one key issue, i.e., security, since most of the effective security techniques for wired networks are not effective for wireless networks. Providing security in WSNs is challenging due to limited sensor node resources [4]. Current algorithms cannot provide correct results without secure, reliable, and accurate time synchronization [5]. There have been no protocols designed with security in mind, and therefore, it has become easy to execute attacks on time synchronization by following the protocol rules [6]. It is the nature of the corresponding protocols that makes them vulnerable to several types of security threats such as message manipulation attacks in which an attacker can inject corrupted messages into the network [7].

Time synchronization is critical in sensor networks because it provides accurate and secure localization, better duty cycling, beamforming, and other collaborative signal processing tasks [5]. One of the important features of wireless sensor networks is to provide smooth and lossless roaming from one station to another station, which is only possible if precise time information is available [8]. Furthermore, to find the exact geolocation, time synchronization is a key element. Similarly, during network synchronization, different procedures such as data transmission, frame computation, and protocol analysis and encapsulation can result in valuable delays depending on the hardware configuration and software flow, which may lead to confusion in time synchronization [9]. Time synchronization is a key element for data fusion, power management, positioning, future action coordination, event timestamping, and power duty cycling, in wireless sensor networks

(WSNs) [9]. Wide use of wireless applications such as target tracking, environment monitoring, and scientific exploration in dangerous environments have attracted the attention of attackers to wireless sensor networks [3]. In addition, authors have indicated that to make these applications work accurately and reliably, the requirement of a common clock time for all sensor nodes is important. The need to limit energy consumption as well as computing and communication resources of nodes in a WSN make it a complex process to build an efficient protocol [10]. Moreover, hardware miniaturization and low-power designs have led to the development of small, battery-operated devices which are capable of detecting conditions such as temperature and sounds; however, miniaturization and low-power designs have limited the data processing and communication capabilities of sensors, making it complex to provide security in the protocol. The complexity in providing security makes a WSN vulnerable to many attacks such as masquerade attacks, replay attacks, message manipulation attacks, and delay attacks, which are the four main types of attacks [6,10]. However, the attack processes are different on different protocols depending on the structure and design of the protocol [5]. For example, in the timing-sync protocol for secure networks (TPSN), a compromised node can claim to be the root node with the lowest ID 0 and start at the larger sequence number than the actual root node, which causes other nodes to follow the compromised node as the root node and to ignore updates from the actual root node [3]. Hence, the main purpose of all attacks on time synchronization is to somehow convince one or more of its neighbor nodes' clocks to be different from the actual clocks of the system. The time synchronization protocol should be able to detect several attacks and should be able to enable fast recovery from these attacks [11]. Clocks need to be resynchronized regularly because computers' internal clocks can differ from each other, even when initially set accurately, and therefore, over time computer clocks will be different because clocks in different computers count time at slightly different rates.

In recent years, many time synchronization protocols for multi-hop networks have been proposed and their performances have been evaluated using simulation or hardware platforms. In [12], the PISync algorithm was proposed and multi-hop performance was evaluated on the MICAz hardware experimental platform using the flooding time synchronization protocol (FTSP). In this case, the clock frequency of the MICAz node was 1 MHz. The delay compensation-based time synchronization (DCBTS) algorithm was presented in [13] and the time synchronization performance was evaluated using the PISync algorithm on the FPGA hardware platform. Specifically, the hardware clock frequency of the FPGA testbed was 50 MHz. Recently, a series of time synchronization protocols for packet-coupled oscillator (PkCO) networks have been proposed [14–18]. The packet-coupled oscillators time synchronization protocol (with an accuracy of around 30.5 µs) was first proposed in [14], and the effects of packet exchange and processing delay were modeled in hardware experiments and also theoretically analyzed. A proportional–integral control scheme was used to fully remove the impacts of processing delay [14]. The packet-coupled oscillators protocol using a dynamic controller (D-PkCOs) was proposed by [15], and a H_∞ design solution [15,16] was also proposed to choose the controller parameters. In these two works, synchronization performances (of less than 1 µs in a single-hop network [15] and around 6 µs in a 21-node spanning tree network [16]) were evaluated on both simulation and hardware platforms. The authors of [17] showed that time synchronization precision of around 1 ms could be realized on low-accuracy RC oscillator clocks (with around 4×10^5 ppm) in 24-h hardware experiments, and they also demonstrated implementation of the PkCOs algorithm.

Moreover, this paper aims to find solutions for the following three research questions (RQs):

RQ1 What are the current time synchronization protocols?

RQ2 What are the security issues of time synchronization protocols and the research progress on secure time synchronization?

RQ3 What are the future research trends of secure time synchronization?

The contributions of this paper are twofold. First, we provide the recent research progress on time synchronization and secure time synchronization. Second, we suggest some future directions in this domain. The remainder of this paper is organized as follows: In Section 2, we describe the basic concepts, important challenges, and requirements of time synchronization. In Section 3, we analyze the current time synchronization protocols. Then, in Section 4, we review the research progress on secure time synchronization and, in Section 5, we discuss future research work directions. The conclusions for this paper are presented in Section 6.

2. Time Synchronization

2.1. Problem Formulation

In recent years, the time synchronization problem in wireless networks has been widely studied in the literature, aimed at achieving a higher order of accuracy with greater scalability of topology and application. However, so far, there is no specific reliable time synchronization scheme available due to the complexity associated with the collaborative nature of nodes [6]. Time synchronization is a method to synchronize the nodes [19]. Computer clocks contain two parts, i.e., an oscillator and a counter. *C(t)* represents the clock if a network node counter increases its value based on the oscillator angular frequency. In ideal situations, angular frequency is constant but it can be changed because of physical variations such as temperature, vibration, and pressure. The local clock of node *i* and real time *t* can be related as [20]:

$$C_i(t) = a_i t + b_i \tag{1}$$

where a_i is the clock drift of node i's clock and b_i is the offset. Drift is the frequency rate of the clock, and offset is the difference in value from the real time t. Local clocks of Nodes 1 and 2 are compared as [20]:

$$C_1(t) = a_{12} \times C_2 + b_{12} \tag{2}$$

where a_{12} is the relative drift and b_{12} is the relative offset of the clocks of Nodes 1 and 2. If relative drift is 1 and relative offset is 0, then both clocks are perfectly synchronized. Clock rate of the network node and offset can be used to adjust its local time according to above equations [21]. Currently, there are many protocols to achieve time synchronization; however, those protocols are vulnerable to many attacks.

2.2. The Importance of Time Synchronization

Cyber-physical attacks to network time synchronization can degrade the performance of the network, for example, data disordering, unsynchronized task execution, duty cycling, and malfunctions [22]. Authors have explained that the attacks which can break time synchronization may lead to increased interference in the network, packet collusions, and delays in the communication messages. Thus, to avoid these circumstances, secure time synchronization becomes a key element for wireless sensor networks to provide its services securely. Poor time synchronization can lead to faulty timestamps, false estimates about the location of nodes, packet loss, and increased cost of energy by disturbing the sleep–wake-up schedule [5]. Sensor nodes, however, monitor data that can be leaked and may lead to personal privacy breaches [4], and more cyber-physical attacks will lead to more energy cost for resynchronization [22]. WSNs require that all nodes work efficiently by synchronizing with each other to save energy consumption [2]. The time synchronization scheme can be affected by many factors that happen in a network, such as communication overhead, available bandwidth, accuracy requirements, scalability, and infrastructure requirements. As described above, sensor nodes have limited resources and low power, and therefore, the energy saved in the synchronization process can be used for security purposes [22]. To avoid the circumstances mentioned above, security for sensor networks must be considered during the design period to ensure the operation of safety, sensitive data safety, and people' privacy [4].

The importance of time synchronization is mostly ignored during the current protocols; thus, a new secured protocol is required to keep data safe and to provide safe services in sensor networks, and it needs to compete with all the challenges and fulfil all the requirements of secure time synchronization.

2.3. Common Challenges for Time Synchronization

Time synchronization is a broad area and many researches have been involved in this area over the past few decades [23]. Several algorithms and mechanisms have been proposed and used. Node A sends a message to Node B with its current time in order to synchronize with Node B. If there is no delay at all in delivering and receiving the message, Node B can immediately calculate the difference between the clocks and adjust its time according to the Node A clock [24]. However, in a real wireless network, several types of delays from the sources described above affect the delivery of messages which make time synchronization much more difficult. In order to estimate the relative clock skews and offsets among nodes, a series of timing messages can be transmitted, and therefore, time synchronization can be regarded as the process of removing the effects of delays from timing message transmissions.

However, in WSNs, it is not suitable to use the current synchronization techniques because of WSNs' unique characteristics such as limited battery power, limited bandwidth availability, and limited computational resources and storage space. These characteristics make the traditional time synchronization schemes, i.e., the network time protocol (NTP) and the global positioning system (GPS), unsuitable for WSNs [23]. The following challenges need to be considered to achieve time synchronization [23,25]:

2.3.1. Nondeterministic Delays

Nondeterministic delays in the message delivery and confound latency estimations contribute directly to the challenges in secure time synchronization [26]. For example, the physical layer access time can be magnitudes larger than the required synchronization precision of the network. Several nondeterministic delays have been analyzed first by Kopetz and Ochsenreiter [27] and added to by [28], according to the process of the message delivery. To illuminate the synchronization nondeterministic delays, it is necessary to analyze the sources of these message delivery delays [26]. As shown in Figure 1, the following components are presented in message delivery delays [27–29]:

- Send time is the building time of a message at application layer. It also includes the delays introduced by the operating system overheads and protocol processing.
- Access time is the time you have to wait, after reaching the medium access control (MAC) layer, for accessing the transmission channel.
- Transmission time includes the time at physical (PHY) layer for transmitting a message. Delay is deterministic and can be calculated by the packet size.
- Propagation time is a time which is the actual time for a message to be transmitted from the sender to the receiver. It is deterministic, depending on the distance and in most cases, small enough to be ignored from latency estimation.
- Reception time includes the time at the physical (PHY) layer for receiving a message which is the same as transmitting a message.
- Receive time is the time which is taken at the receiver to construct and send the received message to the application layer.

The medium access control (MAC) layer time-stamping technique can be used to eliminate these sources of nondeterministic delays by using the TPSN and FTSP [30]. The TPSN reduces the uncertainties by using time stamping in the MAC layer but, in the case of root node failure, the whole network needs to be resynchronized which increases the overhead and takes a large amount of bandwidth [31]. The FTSP achieves high accuracy by utilizing customized MAC layer timestamps to reduce the nondeterministic message delays; however, each time, it needs access to actual hardware which requires increased computational resources and energy consumption because it is not a purely software-based protocol [32].

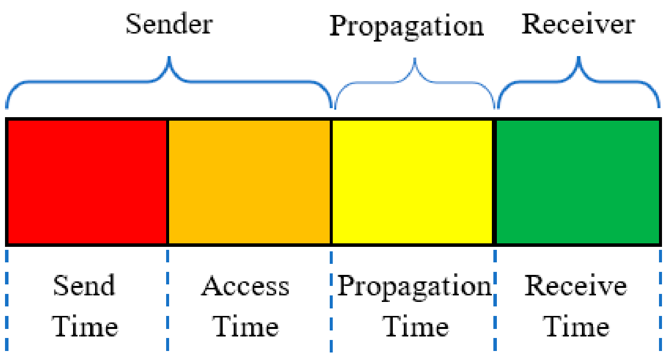

Figure 1. Packet delay components.

The RBS protocol eliminates the largest sources of errors, i.e., send time and access time, from the critical path by using the receiver-to-receiver technique. This protocol saves energy consumption on time updates but there is still propagation time and receiver time uncertainty. If the receiver stores the time messages received which require a large number of storage space, this algorithm will provide better results [21,33].

2.3.2. Robustness

Sensor networks are left unattended for large amounts of time in hostile environments [23]. In the case of some nodes' failures or link failures, synchronization schemes should be up and running for the remaining nodes. Mobile nodes move around which can disrupt the routing schemes and can create partitioning in the network. The TPSN is unable to handle topology changes efficiently [31]. In the case of root node failure, the whole process of synchronization must be repeated which increases energy consumption. The LTS protocol is not suitable for high precision apps because its accuracy reduces towards the depth of the tree [34]. However, the FTSP provides a high level of robustness as it selects its root node dynamically, which provides high precision and works fine even in the case of root node failure, but the downside is that it increases energy consumption [30].

2.3.3. Convergence Speed

Wireless sensor networks always contain a large number of sensor nodes and two nodes can contact to each other through many hops, which makes it difficult to reduce the convergence speed in time synchronization algorithm design [23]. The tiny-sync and mini-sync protocols both provide the benefits of minimum usage of storage space and computing resources, require low bandwidth, and tolerate message loss. However, they have the disadvantage of high convergence time of achieving synchronization [34]. The RBS protocol has a large number of message exchanges to achieve synchronization and resynchronization which takes a large amount of the available bandwidth. The FTSP is slow in convergence because it requires an initiation period due to linear regression [32].

These challenges need to be analyzed carefully and corrected to avoid the nondeterministic delay in radio messages delivery which can affect the precision of time synchronization. In addition to these challenges, there are several requirements that determine what type of synchronization method should be used.

2.4. Time Synchronization Requirements

Time synchronization is important in some distributed systems or in all types of networks. The other systems, especially without outer dependencies, do not require time synchronization. Recently, various mechanisms and algorithms for time synchronization have been proposed and used; however, WSNs have many characteristics that make these mechanisms and algorithms unsuitable for WSNs. For example, limited battery power of sensors and limited bandwidth availability make it impossible to run synchronization messages frequently, and miniature hardware reduces the computational power and storage space [23]. Therefore, the traditional synchronization techniques, i.e., NTP and GPS are unsuitable for WSNs because their general requirements cannot be fulfilled with limited resources. In the previous section, we analyzed current challenges that are affecting the different methods of time synchronization and making them unsuitable to be used for time synchronization. The following requirements need to be fulfilled by the time synchronization method to achieve time synchronization in the network [35]:

2.4.1. Energy Efficiency

In the current protocols, the RBS protocol estimates clock drift by linear regression based on past clock phase offset which considerably increases the computational complexity and require large number of message exchanges. However, it saves energy consumption by not updating the local clock of the nodes which require expensive clock updates which makes the RBS protocol the most energy efficient method currently in use [33]. The TPSN has high energy consumption because of its computational and communication requirements [31]. It requires three message exchanges (Sync, Pulse, and Ack) which result in a high communication load. Clock drift computation is not complex but local clocks of the nodes are updated regularly which also increases energy consumption. The TS/MS protocols are also energy efficient by keeping the usage of storage space and computational resources low [34]. The FTSP is the most inefficient method because the number of message exchanges is high and local clocks of nodes are updated regularly which results in higher computational costs [32]. Limited energy resources available to sensor nodes should always be considered while designing the protocol for wireless sensor networks [35]. To achieve this goal, a technique for time synchronization should be designed with a minimum number of synchronization exchange messages.

2.4.2. Scalability

Deployment of a large number of sensor nodes is required by many sensor network applications. Synchronization protocols should be designed to work well with the scale of the network size and/or density [35]. For example, the TPSN requires a topology hierarchy and does not adopt with dynamic changes in the network structure [31]. However, the FTSP is the best scalable network protocol, as it selects its root node dynamically, which provides high precision, but it increases energy consumption [30]. It is necessary to design a protocol with scalability because different numbers of sensor nodes are required for different tasks or different applications.

2.4.3. Precision

Synchronization precision depends on the specific applications which may vary for different applications in sensor networks. For example, for some applications simple ordering of events may be sufficient, whereas microsecond accuracy may be required for some other applications [35]. In current protocols, the RBS protocol uses a receiver-to-receiver approach which eliminates the nondeterminism at the sender which helps the RBS protocol to provide high precision [33], whereas the TPSN utilizes the MAC layer for messages exchanges from both sides between nodes which results in providing a better precision rate than the RBS protocol [31]. Furthermore, the FTSP provides even higher precision by utilizing the MAC layer timestamps and linear regression to eliminate clock drift and offset [32]. However, the LTS protocol is not suitable for high precision apps because its accuracy reduces towards the depth of the tree [34].

2.4.4. Robustness

In cases of some nodes' failures, synchronization schemes must be be up and running for the remaining nodes, because sensor networks are left unattended for large amounts of time in hostile environments [35]. However, the available protocols are not efficient with this option, for example, the TPSN is unable to handle topology changes efficiently. In the case of root node failure, the whole process of synchronization must be repeated which increases energy consumption [31]. The TS/MS protocols have high convergence time and, in the case of node failure, resynchronization will take longer. The FTSP has high robustness against node failure and topology changes; however, as mentioned above, it increases computational resources and energy consumption which is limited for sensor nodes [30].

2.4.5. Cost and Size

Sensor nodes must be cost effective and miniature in hardware size. These are reasons which make attaching them to large and expensive hardware devices (such as a GPS receiver and storage devices) an unacceptable solution [35]. For example, it makes no sense to attach a GBD 10 sensor node to a GBD 100 GPS receiver, especially when you are using a network of thousands of sensor nodes which will increase the size of the device and result in a marginally high price. The LTS and TS/MS protocols require larger bandwidth and larger storage space, the RBS protocol requires complicated hardware, and these expensive hardwares increase the cost. The FTSP is a hardware-based protocol which means that, even for smaller networks, it will require expensive hardware similar to that required for larger networks [21].

These aspects of a time synchronization protocol need to be balanced according to the requirements in order to achieve efficient time synchronization.

3. Current Time Synchronization Methods

Recently, valuable studies have been conducted on secure time synchronization methods. Some of the research results are described below.

3.1. Global Positioning System (GPS)

The global positioning system is a navigation system based on 32 satellites, which originally used 24 satellites, that has been developed by the U.S. Department of Defense (DoD) [36]. It provides accurate location and time information, in all weathers, anywhere on the earth or near the earth where there is an unobstructed line of sight to three or more GPS satellites. GPS time synchronization (Figure 2) provides accuracy in the order of 200 ns, but the hardware is expensive and it is a power-hungry device [37]. In addition, to work properly and accurately, three of its satellites must be in the line of sight at all times. Currently, at least four satellites are in the line of sight all the time, as shown in the figure below; however, this may not be possible in some cases such as inside buildings or underwater [21].

To provide time synchronization, GPS devices communicate with satellites [38]. To do so, a GPS receiver is required in each device, which is impractical, especially for wireless devices, because of hardware costs and power constraints [39]. Nodes receive real time information from the GPS and synchronize themselves as follows: Every node has its own hardware clock; denote the value of node i's hardware clock by $Hi(t)$. We assume that every node's hardware clock has bounded drift $\rho < 11$. For all nodes i,

$$\forall t : 1 - \rho \leq \left(\frac{dH_i(t)}{dt} \right) \leq 1 + \rho \tag{3}$$

where t is real time, d is distance, and ρ is bounded drift. Each node computes a logical clock value by using its hardware clock and received message from other nodes. Note node i's logical clock value at time t by $Li(t)$. The clock synchronization algorithm tries to make sure that the logical values of the nodes are close to real time and close to each other's values [40].

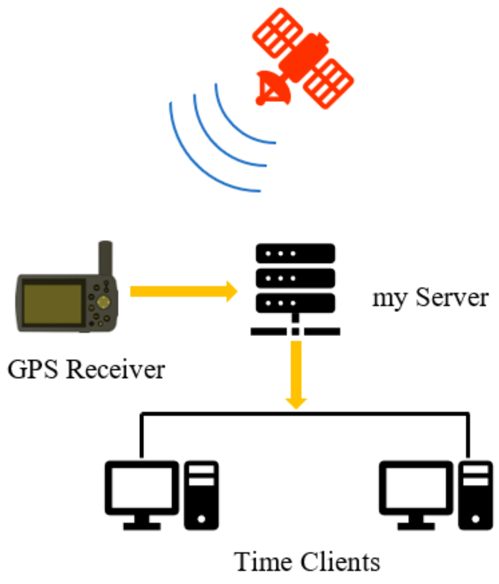

Figure 2. GPS time synchronization.

The accuracy of GPS time synchronization will vary at different times, depending on the number of satellites that can communicate with a receiver. In addition, it takes some time for a GPS message to propagate the entire network [36]. For example, A GPS message is received at the node i by an input action gps(t)i. The purpose of the message is to inform i that the current real time is t. However, it can take longer for the message to reach node i in a big network and node i may receive this message after the real time t [40]. Expensive hardware such as a GPS receiver and the high energy requirement of the GPS protocol result in using increased bandwidth and computational power, which makes it unsuitable because sensors have limited battery power, limited bandwidth availability, limited storage space, and limited computational power.

3.2. Network Time Protocol (NTP)

The NTP uses several algorithms to reduce jitter, increase the robustness, and avoid improperly operating servers which allows it to provide accuracies of low tens of milliseconds on WANs and sub-milliseconds on LANs [36].

Synchronizing Node A and node B involves an exchange of packets, Node A stores its local time in the request packet *T1* [41]. Then, Node B time stamps it as *T2* according to the current local time on arrival of the request and sends a reply, *T3*, which includes the current local time of departure of the packet. When Node A receives the reply, it is stamped as *T4*, as shown in Figure 3 [42]. Then, the NTP calculates the clock offset and roundtrip delay as below, respectively [41–43]:

$$\text{offset} = \frac{(T_2 - T_1) + (T_3 - T_4)}{2} \tag{4}$$

$$\text{delay} = (T_4 - T_1) - (T_3 - T_2) \tag{5}$$

The NTP does not consider energy consumption and keeps the processor busy to frequently discipline the oscillator, which is not possible in sensor nodes because sensor nodes have limited resources and cannot spend all the CPU cycle on time synchronization [43].

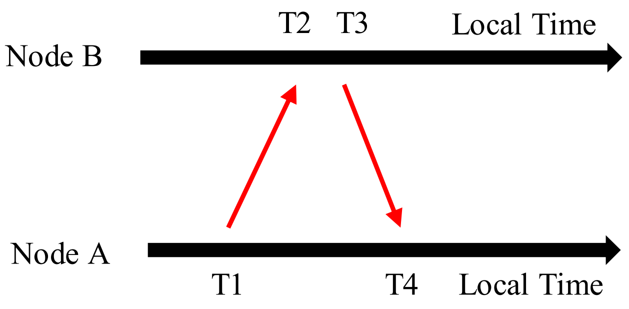

Figure 3. Two-way handshake between a pair of nodes.

3.3. Timing-Sync Protocol for Sensor Networks (TPSN)

The TPSN was proposed by Ganeriwal et al. for network-wide time synchronization [28]. A sender–receiver structure is used in the TPSN model; the receiver synchronizes its clock with the clock of the sender according to the two-way handshake, as shown in Figure 4 [36,44,45]:

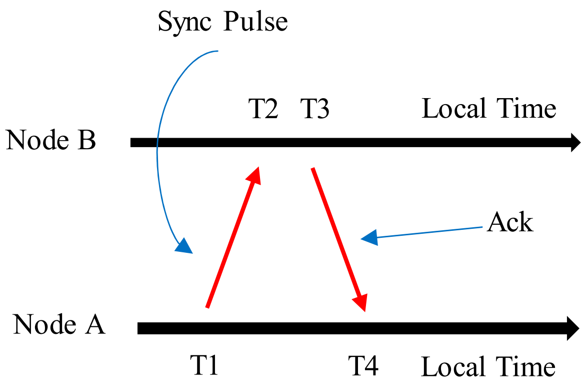

Figure 4. Synchronization between two nodes.

Node A sends a pulse packet at *T1* to start the synchronization which includes its level number and value of *T1* according to its local clock [32,36,44]. Node B receives this message at *T2*, and *T2* = *T1* + *D* + *d*, where *T1* is the message from Node A, *D* is the relative clock drift between Node A and Node B, and *d* is the propagation delay of the pulse sent between the nodes. Node B sends an acknowledgement packet at time *T3*, which includes the level number of Node B and values of *T1*, *T2*, and *T3*. Node A synchronizes itself to Node B after calculating the clock drift and propagation delay as shown below [21,36,44]:

$$D = \frac{(T_1 - T_2) - (T_4 - T_3)}{2} \tag{6}$$

$$\text{offset} = \frac{(T_2 - T_1) + (T_3 - T_4)}{2} \tag{7}$$

The TPSN works in two phases: first, the discovery phase, and then the synchronization phase [46]. The aim of the first phase is to assign a level to each node and create a hierarchical topology in the network [25]. Only the root node is assigned Level 0 as shown below in the Figure 5 [47,48].

In the second phase, all nodes connect to the parent node in the hierarchical structure in a two-way handshake message similar to the NTP shown above [31]. In this way, all nodes are synchronized to the root, and network-wide synchronization is achieved by considering a two-way handshake message between two Nodes A and B, as shown in the figure above [32].

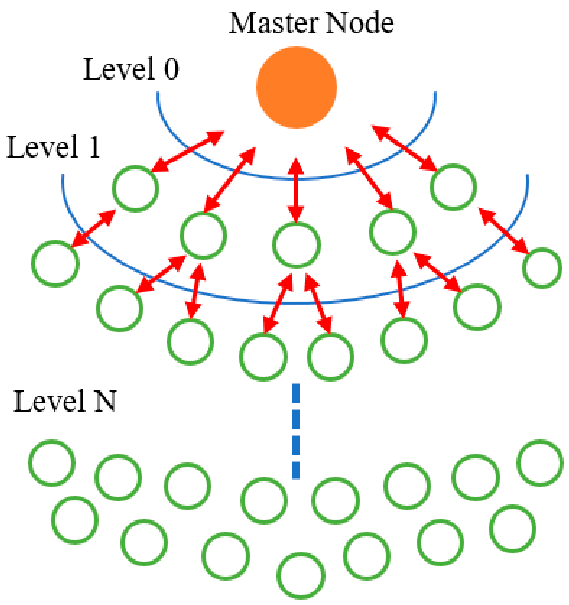

Figure 5. Hierarchical structure.

Similar to the NTP, the TPSN is also based on a hierarchical structure (as shown in Figure 5 above) which provides scalability [47]. Synchronization depends on the nodes' parents in the hierarchical structure; therefore, high synchronization accuracy can be achieved even while the size of the network increases. However, in the case of failure, maintenance of this structure increases the energy consumption [31]. Because the connectivity of a node in the hierarchical structure changes when the node moves, the structure needs to be formed accordingly, since, in the TPSN, a node adjusts its clock according to its parent node [21]. This complex maintenance and energy consumption makes the TPSN unsuitable for secure time synchronization because, in WSNs, nodes can be consistently moving.

3.4. Tiny-Sync and Mini-Sync Protocols

Sichitiu and Veerarittiphan proposed two lightweight synchronization algorithms, i.e., tiny-sync and mini-sync [20]. To obtain the offset and rate difference between two nodes, both the tiny-sync algorithm and the mini-sync algorithm use a multiple round-trip time measurement technique and line fitting technique. Multiple round-trips are performed to obtain data points for line fitting, as shown in Figure 6, where Node A is the client and Node B is the reference [46].

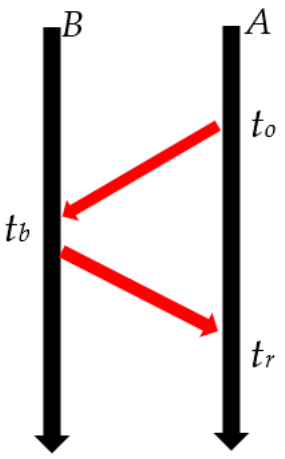

Figure 6. Data points calculation.

In [20], the authors assumed that Node A timestamps a message with t_0 and sends it to Node B. Node B timestamps the message with t_b as soon as it receives it and sends it back to the Node A which receives the message and timestamps it with t_r, as shown in Figure 6 below [20,46].

Each round trip measurement results in a data point $(t_b, [t_0, t_r])$ which effectively limits the possible values of the parameters a_{12} and b_{12}. Since t_0 happened before t_b, and t_b happened before t_r, the following dissimilarities should hold there:

$$t_0(t) < a_{12}t_b(t) + b_{12} \tag{8}$$

$$t_r(t) > a_{12}t_b(t) + b_{12} \tag{9}$$

The calculation described above is repeated several times and all time data points are stored [46]. The two lines with minimum and maximum slope are determined using the line-fitting technique, which provides us with the bounds for the relative offset and rate difference of the two nodes by using their slope and axis. The line which has average slope and intercept is used as the offset and rate difference between those two nodes.

The tiny-sync and mini-sync protocols use the estimation method to provide the solution of relative skew and relative offset which reduces the complexity and energy consumption [32,34]; however, it requires higher computational resources and memory to generate accurate results. Computational resources and memory depend on the number of data points. A lower number of data points provides a suboptimal solution, whereas a higher number of data points provides an optimal solution, but takes a large amount of memory and computational resources. Miniaturized modern technologies have less memory and fewer available computational resources which makes this algorithm unsuitable for them.

3.5. Lightweight Time Synchronization (LTS) Protocol

The main focus of the lightweight time synchronization (LTS) protocol is to minimize the energy cost by reducing overhead while it remains robust and self-configuring [49]. Lightweight time synchronization (LTS) builds a tree structure within the network to provide network-wide synchronization. The main function of the protocol is that it continues to work effectively even if there is a node failure. It aims to maximize the accuracy while minimizing the complexity of the synchronization and the use of computational resources. It is assumed that the required accuracy in sensor networks is low; therefore, it is appropriate to use a lightweight synchronization scheme [50]. However, in some applications such as measuring the time-of-flight of sound, distributing an acoustic beamforming, landslide detection, natural disaster prevention, and detecting fire in a forest, high precision time synchronization is required. In addition, in a larger network, the accuracy of synchronization decreases linearly, for example, from the root node towards the leaf node, accuracy decreases [34]. Figure 7 explains how LTS synchronization is performed.

The LTS pairwise synchronization protocol uses a remote clock reading technique to synchronize two neighbor nodes [49]. For example, node i wants to synchronize its clock with node j. Node i sends a synchronization request message and when synchronization is triggered the message is time stamped with $C_i(t_1)$. After random access medium access delay, node i sends the packet at time t_2 and node j receives it at time $t_3 = t_2 + \tau + t_p$, where τ is the propagation delay and t_p is the packet transmission time. At time t_4, the packet arrival is signaled by an interruption and timestamped at t_5 with $C_j(t_5)$ which is followed by an answer packet time stamped with $C_j(t_6)$ at time t_6 (also includes previous timestamps $C_j(t_5)$ and $C_i(t_1)$) [51].

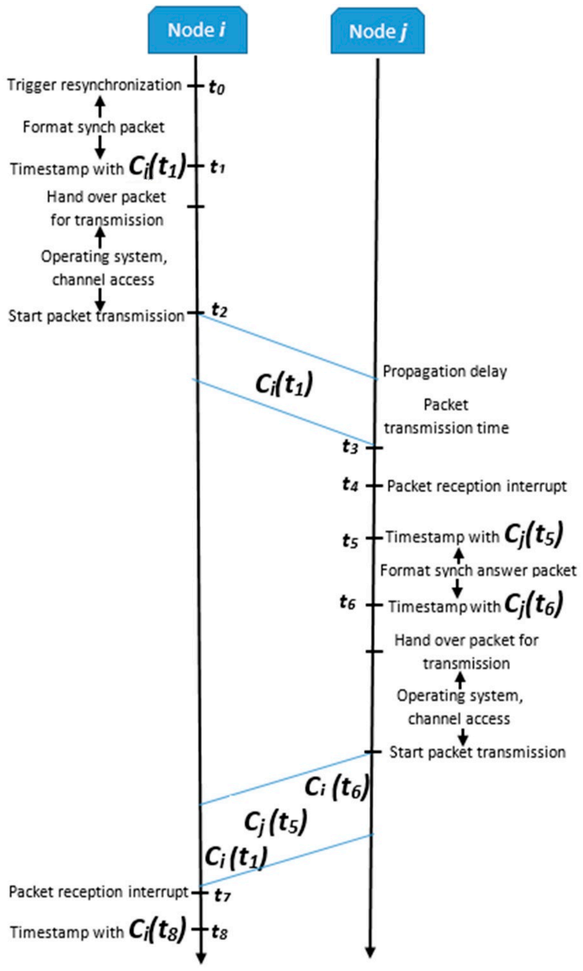
Node i
Node j
Trigger resynchronization
t0
Format synch packet
Timestamp with Ci(t1)
t1
Hand over packet for transmission
Operating system, channel access
Start packet transmission
t2
Ci(t1)
Propagation delay
Packet transmission time
t3
t4
Packet reception interrupt
t5
Timestamp with Cj(t5)
Format synch answer packet
t6
Timestamp with Cj(t6)
Hand over packet for transmission
Operating system, channel access
Start packet transmission
Ci (t6)
Cj (t5)
Ci (t1)
Packet reception interrupt
t7
Timestamp with Ci(t8)
t8

Figure 7. LTS pairwise synchronization.

Node *i* receives the answer packet at t_7 and time stamps it with $C_i(t_8)$ at time t_8. Node *i* assumes that there is no clock drift between t_1 and t_8, $O = \Delta(t^*)$ for all $t^* \in [t_1,t_8]$ and estimates offset O by estimating $\Delta(t_5)$ in the following way:

$$O = \Delta(t_5) := C_i(t_5) - C_j(t_5) \tag{10}$$

However, t_5 is unknown between t_1 and t_8. Uncertainty can be reduced, as we can analyze from the figure given above that there is propagation τ and transmission time tp between t_1 and t_5 and also between t_5 and t_8. We assume it is the same in both directions. Since we have t_5 and t_6, we can obtain the difference by $C_j(t_6)$–$C_j(t_5)$. We also assume that the operating system, channel access, interruption, and medium access delay are also the same in both directions as shown in Figure 7. Therefore, $C_j(t_5)$ is generated at time:

$$C_i(t_5) = \frac{C_i(t_1) + \tau + t_p + C_i(t_8) - \tau - t_p - \left(C_j(t_6) - C_j(t_5)\right)}{2} \tag{11}$$

Therefore, the offset O is:

$$\begin{aligned} o = \Delta(t_5) = C_i(t_5) - C_j(t_5) \\ = \tfrac{C_i(t_8)+C_i(t_1)-C_j(t_6)-C_j(t_5)}{2} \end{aligned} \tag{12}$$

Node *i* can adjust its clock by adding offset O, it took only two packets to synchronize node *i* and node *j*. A third packet from node *i* to node *j* can be used, including O, if the goal is to let node *j* know about offset [51].

After providing pairwise synchronization, the LTS protocol solves the synchronization of all the nodes with the reference node. Two different approaches are proposed for this purpose, i.e., a centralized approach and a distributed approach [49]. In the centralized approach, a spanning tree is constructed, and then the nodes are synchronized [44]. The reference node is the root node of the spanning tree and is responsible for starting the synchronization of the network. The depth of the spanning tree is communicated back to the root, because it affects the time to synchronize the whole network, and therefore, this information can be used for deciding the time of resynchronization. A critical issue with this is the communication cost because a pairwise synchronization costs three packets and synchronizing the whole network will cost the number nodes x 3 packets which increases the energy consumption used to construct the spanning tree [51].

In the distributed approach, a spanning tree structure is not used and each node can decide the time of its own synchronization [25]. When any node such as Node 'A' needs to be synchronized, it sends a synchronization request to its nearest reference node, as shown in Figure 8 [51].

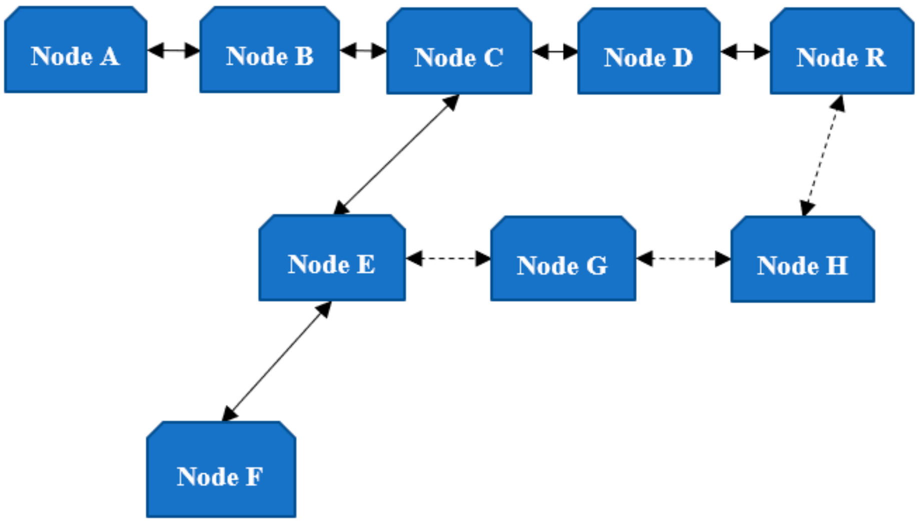

Figure 8. Multi-hop synchronization.

All the nodes must be synchronized along the path from the reference node to Node 'A' which saves the unnecessary efforts of frequent synchronization. Since all the nodes from the reference node to the initiator node need to be synchronized, a collective synchronization request can be generated to reduce the number of resources used [25].

3.6. Flooding Time Synchronization Protocol (FTSP)

The flooding time synchronization protocol (FTSP) was proposed by Maroti et al. and aims to achieve synchronization in the whole network by sending broadcast synchronization messages [30]. A broadcast message starts with the preamble bytes, followed by the SYNC bytes, then with the actual message data which describes the message, and ends with the CRC bytes. The dotted lines in Figure 9 are actual bytes and the solid lines are bytes in the buffer. When the preamble bytes are in transmission by the sender, the receiver adjusts its carrier frequency of the incoming signals. As soon as the SYNC bytes are received, the receiver can calculate the byte offset needed to reassemble the message with correct byte alignment. The data field contains the target, length of the data, and other fields of the message which need to be notified on the receiver side. The message is verified as not being corrupted by using CRC bytes.

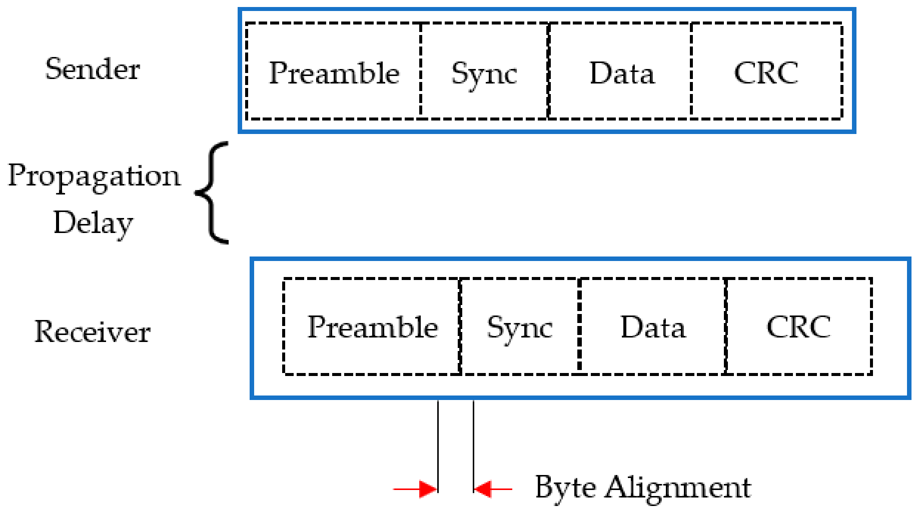

Figure 9. FTSP broadcast message.

In the FTSP, a mesh network can be formed by the nodes where each node sends a synchronization message to all the other nodes, or nodes can form a star network where one of the nodes acts as the master node and all other nodes are slave nodes [46]. The node with the lowest Node ID is elected as the leader which periodically floods the network with synchronization messages and is used as a source of reference time [30]. If current leader nodes fail, the next node with lowest Node ID is elected as the new leader. Timestamps and time of arrival are collected by each node, and linear regression is used on these data points to obtain the offset and rate difference estimation from the leader node. The FTSP provides global synchronization while using low bandwidth and it uses timestamps in the MAC layer which helps it to eliminate many sources of errors [32]. However, it always needs access to the hardware and it is not a pure software-based protocol. The major drawback is that any node can elect itself as the leader after a certain period of time if it has not received any new timestamps, which makes it vulnerable to attacks. A corrupt node can claim itself to be a leader and can misguide the others to break the synchronization of nodes with the actual leader [47].

3.7. Reference Broadcast Synchronization (RBS) Protocol

The reference broadcast synchronization protocol was proposed by Elson, Girod, and Estrin in 2002 [33]. Nearly all other synchronization protocols use a sender-to-receiver synchronization method [47]. However, the approach of the RBS protocol is different because it uses a receiver-to-receiver synchronization method by using a third party [32]. The concept is that a single broadcast signal will be sent by a third party to two receivers which will not contain any type of time information or timestamp. The receivers will both exchange the receiving time of that signal and the difference between the clocks [34]. For example, as shown in Figure 10, a reference Node R sends a synchronization message to all the slave nodes. Two nodes, Nodes A and B, are in the communication range of Node R, and they will receive the message. The receiving times of the message are recorded as T_a on Node A and T_b on Node B.

Nodes A and B exchange this timing information with each other, and Node B calculates the timing difference to Node A and records it using d as follows:

$$d = T_a - T_b \tag{13}$$

After that, Node B can adjust its time to synchronize with Node A using d as follows:

$$T_b = T_a - d \tag{14}$$

The information gained is enough to continue a local time scale. By doing this, it eliminates the send time and access time from the critical path, as shown in the Figure 11 [21,33].

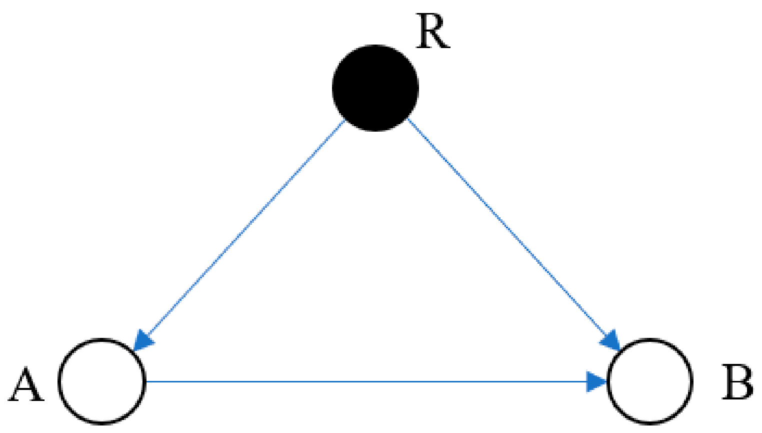

Figure 10. RBS protocol message exchange.

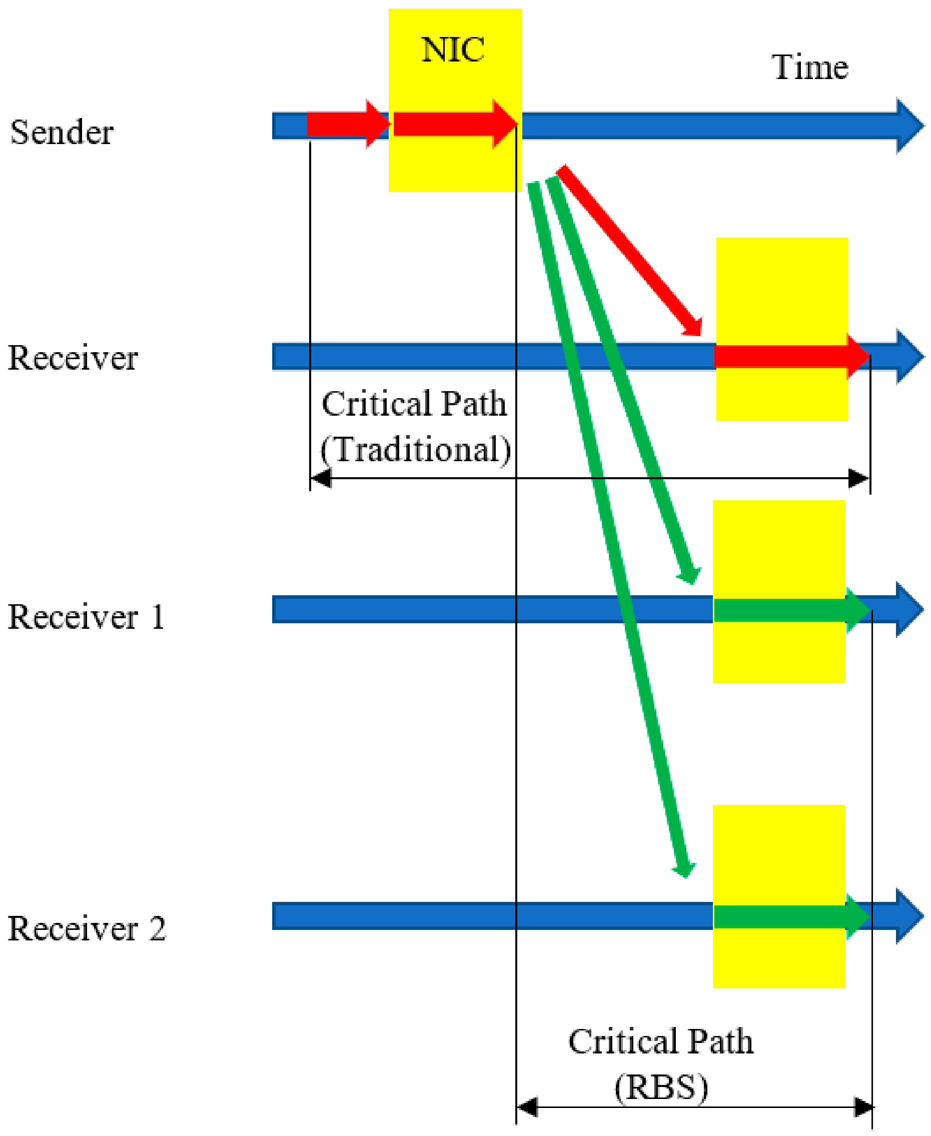

Figure 11. Critical path analysis (RBS protocol).

In the RBS protocol, a third party sends a broadcast signal which makes it easy to spoof the nodes and stop the actual senders' information [46]. In addition, because of its heavy overheads, it consumes more energy in exchanging packets, which makes it unsecure and unsuitable for low energy consumption networks.

The information given above shows that security was not the top priority during the designing of the current available protocols. This problem can be solved by reducing the communication overhead which reduces the energy requirement, and the saved energy can be used for the purpose of the security. In further sections, we examine the nature of the problem and explore related work conducted in the past to find a suitable solution.

3.8. IEEE 1588 (PTP)

The first version of the IEEE 1588 precision timing protocol (PTP) was released in 2002, and the second version was released in 2008 [52]. It is an application protocol that follows the master–slave architecture and aims to provide high accuracy and a robust synchronization procedure. This protocol allows for synchronization in the nanosecond range. Compared with the previously mentioned protocols, such as GPS and NPT, the PTP is a more robust time synchronization [53]. The IEEE 1588 PTP allows clocks to be distributed across packet-based networks such as the Ethernet. Time-stamping packets should be exchanged between distributed nodes in order to synchronize the network precisely [54].

3.9. Comparable Analysis of the Different Time Synchronization Protocols

The NTP is the time synchronization protocol commonly used in the Internet domain [55]. NTP clients used statistical analysis of the round-trip time to synchronize their clocks with the NTP servers, with an accuracy of milliseconds. External time sources are used to synchronize the time servers such as GPS. In the Internet, the NTP has been proven to be effective, secure, and robust, and therefore it is deployed widely; however, in WSNs, transmission time can vary because, at each hop, medium access control (MAC) can introduce a delay of several hundreds of milliseconds [55]. The reference broadcast synchronization protocol was proposed by Elson et al. which uses receiver-to-receiver synchronization instead of sender-to-receiver synchronization which is used by many other protocols [33]. The idea is to broadcast a beacon by using a third party without any timing information, and receivers will compare their times of receiving that beacon to their offset. Now, the only remaining uncertainty is propagation and receiving time. It is claimed that all nodes will receive the reference beacon straightaway which takes out propagation, but receiver uncertainty still remains in the protocol.

Ganeriwal et al. proposed a traditional sender–receiver-based time synchronization protocol called timing-sync protocol for sensor networks (TPSN) [28]. It is divided into two phases, in the first phase, a hierarchical topology of the network is created where each node is assigned a level and the node with lowest level becomes the root node. In the second phase, all the nodes in the tree synchronize to their parent using the round trip synchronization method. The flooding time synchronization protocol (FTSP) was proposed by Maroti et al. and it is the most well-known time synchronization approach [30]. In order to achieve relatively high precision, it uses a fine-grained clock, clock drift estimation, and time stamping at the MAC layer to reduce jitter. The tiny-sync/mini-sync protocols use multiple pairwise round-trip measurements and a line-fitting technique to obtain the offset and drift of the two nodes, rather than directly calculating the offset [20]. The IEEE 1588 PTP is more accurate than the NTP. In addition, in the PTP, a grandmaster synchronizes all the slave nodes which can eliminate the additional delay due to redistribution in the NTP [53]. The NTP and PTP are hard to use in wireless networks because of the resource constraints and asymmetry of communication links.

Most of the existing protocols rely on a certain reference clock and hierarchical structure; however, the reference clock only considers the compensation of clock offset, which makes the protocols more vulnerable to intelligent attacks and single node failure.

Several representative time synchronization protocols are selected to compare the performances of different protocols. For a comprehensive comparison and overview of the performances, we select different clock configurations (1 MHz, 50 MHz, 32.678 MHz, and 32.678 kHz) with different skews, either hardware clocks (MICAz mote or on the FPGA platform) or software simulation clocks. Figure 12 shows the comparison of maximum time synchronization errors in different protocols by histogram. The blue and red bars refer to the PISync and the FTSP evaluation performance on the MICAz hardware platform in [12], respectively, while the orange and purple bars refer to the experimental results of the PISync and DCBTS protocols on the FPGA testbed in [13]. In addition, a MATLAB-based simulator in [15] is utilized to evaluate the performances of PISync, D-PkCOs, and the TPSN, and the simulation results are illustrated in green, black, and brown bars, respectively.

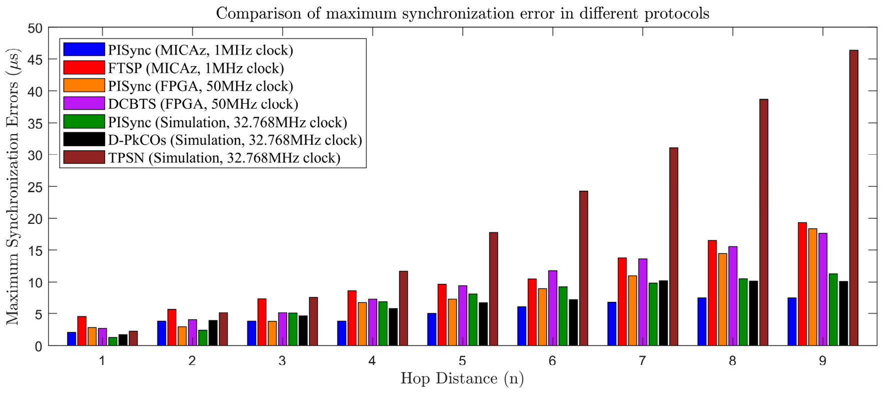

Figure 12. Performance comparison of time synchronization precision of different protocols in a multi-hop sensor network. Different clock configurations are used for a comprehensive comparison, covering both hardware test results and simulation results.

Overall, it is not surprising that all the protocols have increasing synchronization errors when the hop distance increases, but at different increasing rates. At a 1 MHz hardware clock, on the well-known MICAz wireless sensor network node platform, PISync achieves the best synchronization accuracy with the slowest increasing rate concerning hop distance, for example, 2 μs at the single hop and around 7 μs at the 9th hop [12], which is better than the long-standing FTSP. However, at a 50 MHz FPGA hardware clock [13], the performance of PISync degrades to 3 μs at the single hop and to 17 μs at the 9th hop, and the results of DCBTS in [13] are slightly better than those of PISync, although the clock granularity has increased. A similar performance is observed in a simulation at a 32.768 MHz clock. This may be due to the better configuration of a 1 MHz clock (e.g., a very stable clock with tiny skews) used in [12], but the real clock and realistically simulated clock have worse skews than that used in [12]. For a fair comparison, more factors associated with the platform and clock should be considered. At a software-simulated 32.768 MHz clock with realistic skews, the recently proposed D-PkCOs [15] protocol outperformed PISync and the long-existing TPSN, mainly due to the feedback control mechanism used to simultaneously correct both clock offset and skew using the proposed dynamic controller of D-PkCOs, while PISync adjusted the clock offset by only using an adaptive PI controller. Since the 32.768 kHz real-time clock (RTC) is widely used in most existing industrial systems as a standard source of the clock, we also simulated the performance of a 32.768 kHz clock synchronized using PkCO, which is a better protocol at a higher granularity 32.768 MHz clock. Although PkCO has around 2 μs error at the single hop and only 11 μs at the 9th hop at the 32.768 MHz clock, the same PkCO synchronization protocol degrades to 40 μs and 175 μs at the single hop and the 9th hop, respectively, at the 32.768 kHz clock. The above observations are only some of many examples in the literature that show the complexity of a performance comparison of different synchronization protocols. In particular, cross-platform performance comparisons sometimes do not agree with each other. It is recommended that the comparison results are more reliable and useful when the same types of clocks and platforms are compared.

4. Secure Time Synchronization

4.1. Background

The main function of wireless sensor networks is to sense the environment and to transmit the acquired information for further processing with secure time information [6]. Previous studies [2,3,22] have shown that the best way to provide security for sensor networks is to consider security at the design stage. Time synchronization is vulnerable to several cyber-physical attacks such as Sybil attacks, replay attacks, message manipulation attacks, delay attacks, and Dos attacks [22]. These attacks that drop and transmit fake synchronization messages into the network are called message manipulation attacks, and according to the definition, replay attacks, delay attacks, and fault data injection attacks can also be considered to be different types of message manipulation attacks. It has also been observed that current protocols are not able to provide secure time synchronization. Wang et al. stated that wireless sensor networks are vulnerable to potential attacks which include Sybil attacks, wormhole attacks, replay attacks, pulse-delay attacks, silent attacks, jamming attacks, and malicious reference attacks [3]. Furthermore, they analyzed that security was not the top priority during the designing stage of the current protocols. Therefore, it is suggested that security should be considered during the designing stage of a protocol. There are many different threats to WSN security including a malicious outside attacker with jamming and replay abilities that compromise nodes [2]. An attacker can eavesdrop by inserting malicious nodes in the network which sends corrupt data to the sink node, whereas an attacker with jamming and replay abilities can deploy a pulse-delay attack which results in jamming a message, storing it, and replaying it later as required by the attacker. An attacker can misguide the friendly nodes by compromising them with an enemy node which creates ambiguity. However, all these threats can be avoided by providing secure time synchronization.

4.2. Secure Time Synchronization Related Works

Recently, many studies have addressed security issues in sensor networks [5,10,56] and have tried to provide solutions, however, it has been difficult to implement these solutions with the existing time synchronization techniques because they require high computational resources or provide low accuracy. Therefore, it proves that the existing time synchronization techniques were not designed with security in mind which leaves them vulnerable to many security attacks [6].

Ganeriwal et al. proposed a protocol for secure time synchronization which was resilient to outside attackers and inside compromised node attacks [57]. However, the main focus of the study was to detect the attack and abort the ongoing time synchronization rather than to prevent the attacks by taking actions against them. In addition, the secure technique proposed by [57] achieved the same accuracy as the TPSN during pairwise synchronization; however, for multi-hop synchronization, its accuracy decreased. In [5], the authors suggested a set of secure time synchronization protocols in which synchronization was addressed against a pulse-delay attack. However, in these protocols, nodes must go through a process of preloaded static key discovery which is performed through many communication messages, increasing the communication cost of the network and making them unsuitable for sensor networks.

Manzo et al. published a theoretical work in which they described three major time synchronizing protocols: the RBS protocol, the TPSN, and the FTSP [58]. They outlined the attacks on each protocol in WSNs and proposed techniques to mitigate these attacks. However, compromise attacks were their main focus. After reviewing some of the current time synchronization algorithms, some of the possible attacks such as eavesdropping, pulse-delay attacks, and compromise attacks that could be performed on those algorithms, were presented by [59]. To protect the network against an outsider, they used authentic and cryptographic techniques.

Hu et al. proposed an attack-tolerant time synchronization protocol (ATSP), which is more desirable for WSNs [56]. Although the ATSP is able to provide synchronization in a distributed manner across a network and can accurately detect attacks, there is still a clock skew error. The above-mentioned studies show that it is important to consider security during the design of a protocol; it is complex and difficult to provide secure time synchronization to avoid cyber-physical attacks, if security is not considered in the designing stage. Possibly, it would be easier to design a new secure protocol than to modify and secure an existing protocol. In [10], the authors proposed a novel secure time synchronization protocol for WSN models based on bilinear pairing functions. The proposed protocol was applied to both heterogeneous and homogeneous WSN models. Moreover, in another study [60], the authors used a graph theoretical approach and proposed a robust and secure time synchronization protocol, named RTSP, against Sybil attacks.

Many contributions have been made to the literature on secure time synchronization to provide security and to solve the privacy issues for routing. However, the existing time synchronization approaches require increased computational resources for routing which make it difficult to implement them for security purposes, especially in sensor networks. Thus, all the current schemes for time synchronization have not considered security in their designs which makes themselves vulnerable to security attacks [6]. Most of the efforts in the literature review have been devoted to provide secure time synchronization with a focus on updating and enhancing the existing protocols for WSNs by using authentication and fault detection mechanisms [61].

5. Discussions for Future Work

The survey results have shown that almost all the current time synchronization protocols have synchronization errors, and those errors are exposed for some of the popular protocols. The LTS protocol is not suitable to provide high precision because its accuracy is reduced towards the depth of the tree, whereas the FTSP provides high precision and high robustness against node failure and topology changes. However, the FTSP requires high energy consumption and computational resources (which is limited for sensor nodes) as it is not a software-based protocol. The RBS protocol involves a large number of message exchanges to provide high precision which increases energy consumption, and therefore costs. The TPSN achieves accuracy of ten microseconds which depends on the growth of offset affected by the propagation time. Although a nondeterministic delay caused by propagation is limited in these protocols compared to the precision that is achievable, in order to meet specific precision every protocol needs to send and receive packets. Sending and receiving packets consume most of the energy during network synchronization. The energy consumption of a network can be reduced by reducing the number of packets exchanges.

For future work, these errors should be improved to achieve better results from these protocols. The TPSN is a widely used protocol which can achieve high accuracy and energy efficiency; however, its accuracy decreases due to a change in the tree structure because of conflicts that occur during the communication of change messages which results in delays, message loss, as well as increased energy consumption. For example, Nodes A and B want to connect with Node C which is a common neighbor of Nodes A and B. Conflict can occur while sending synchronization packets to Node C which is an important factor that affects accuracy.

In Figure 13, Node A sends a synchronization packet to Node C to perform synchronization. Normally, the TPSN assumes that delays between messages are small and can be omitted as zero, and therefore, delay is calculated as follows while ignoring the errors brought by clock bias represented by $\sum 1$ and $\sum 2$. T_1, T_2, T_3, and T_4 are timestamps.

$$d = \frac{(T_2 - T_1) + (T_4 - T_3)}{2} \tag{15}$$

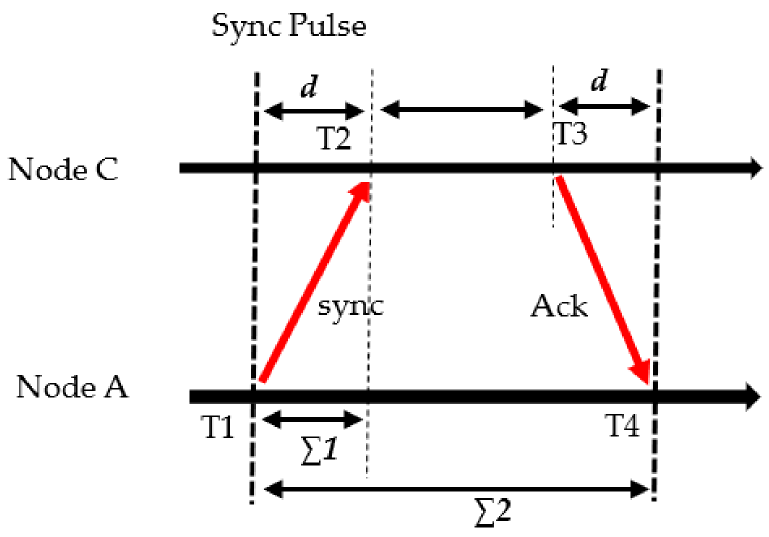

Figure 13. TPSN synchronization message.

However, when a conflict accrues delays over tens of seconds, it is not acceptable in a wireless sensor network, and therefore, these errors need to be considered and cannot be omitted. To calculate the exact delay of the message, we need to rewrite the above equation in the following way:

$$d = \frac{(T_2 - T_1) + (T_4 - T_3) - (\sum 1 + \sum 2)}{2} \tag{16}$$

In the case of conflict, one node has to wait until the other node finishes synchronization which causes delays and reduces accuracy. It is important to reduce conflicts in order to achieve high accuracy and reduce energy consumption.

In addition, the performance of the TPSN depends on efficiency of hierarchical structure (Figure 14). However, in the TPSN, the authors have used simple flooding for level discovery to reduce complexity and energy consumption. Thus, in the TPSN tree structure, when the number of levels increases, the local clock offset difference also increases. The TPSN tree is constructed in following way:

$$N(m) = n(E) + 2(n(T) - 1) = 2 \times n(T) + n(E) - 2 \tag{17}$$

where $N(m)$ is the number of total messages exchanged to provide synchronization, $n(E)$ is the number of elements in a set of parent nodes at a tree, and $n(T)$ is the total number of nodes in a tree. Since the TPSN synchronizes using two message exchanges, it requires as many transmissions as there are nodes in the network. The energy consumption of WSNs depends on these transmissions. To make sure we do not miss any communication messages and to minimize complexity, it is necessary to avoid conflicts during communication messages, and nodes can be organized inside a cluster hierarchy. The benefit of using clustering is that it reduces the depth of the tree which results in improved battery life of sensors and improved network lifetime by reducing energy consumption by reducing synchronization message overhead and scheduling activities. In addition, level discovery is performed only once in the TPSN, and therefore, if one of the nodes dies, all the child nodes of the dead node will fail. One of the possible solutions to avoid is that, whenever some critical nodes request tree reconstruction, a level discovery should be performed again. In this way, some nodes which have low energy levels will not participate in the parent node election during level discovery. Any of the above suggested improvements in the TPSN will increase the accuracy of the TPSN and will reduce energy consumption. In addition, the TPSN would be able to handle topology changes more efficiently and in better way than before.

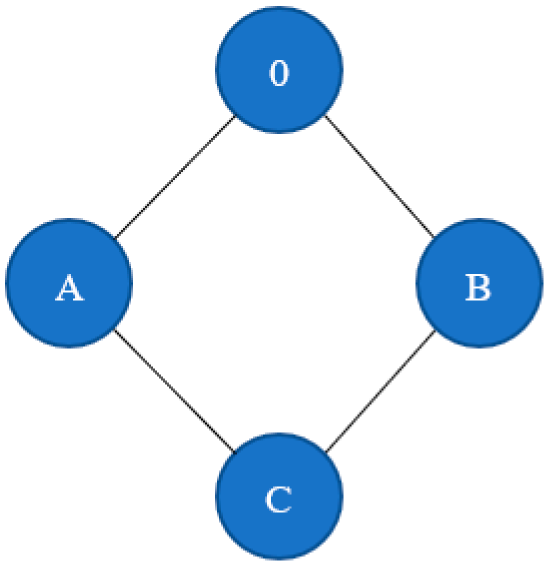

Figure 14. TPSN hierarchical Tree.

Time synchronization plays an irreplaceable role in the development of a WSN, and therefore, it important to have a secure time synchronization protocol which also provides high accuracy and energy consumption efficiency. The RBS protocol provides high accuracy by removing the uncertainty of sender's delay from the critical path by using broadcast messages from a reference node. All other nodes receive these broadcast messages and use them as a reference point to compare their times with neighbor nodes and to change their clocks accordingly. Eliminating sender's uncertainty that comes from transmission and receiving times helps the RBS protocol achieve higher accuracy compared to bidirectional information exchange protocols. However, the most significant drawback of the RBS protocol is the high number of message exchanges which results in high computational complexity, network traffic overhead, and high energy consumption, which are not suitable for the limited battery power of sensors. To gain synchronization in the RBS protocol, a reference node sends a synchronization message to all the slave nodes. Two nodes (as previously shown in Figure 10), Nodes A and B, are in the communication range of Node R, and they will receive the message. The receiving times of messages are recorded as T_a on Node A and as T_b on Node B. Nodes A and B exchange this timing information with each other and Node B calculates the timing difference to Node A and records it by using d as follows:

$$d = T_a - T_b \tag{18}$$

after that, Node B can adjust its time to synchronize with Node A using d as follows:

$$T_b = T_a - d \tag{19}$$

The information gained is enough to continue a local time scale. By doing this, it eliminates the send time and access time from the critical path. However, the number of messages exchanged will be large when synchronizing, depending on the number of nodes in the network. It is important to reduce the number of message exchanges in order to make it suitable for WSNs. One possible solution is to use clustering which can ensure synchronization accuracy by reducing communication overhead and information exchange messages. The network can be divided into clusters as shown below in Figure 15.

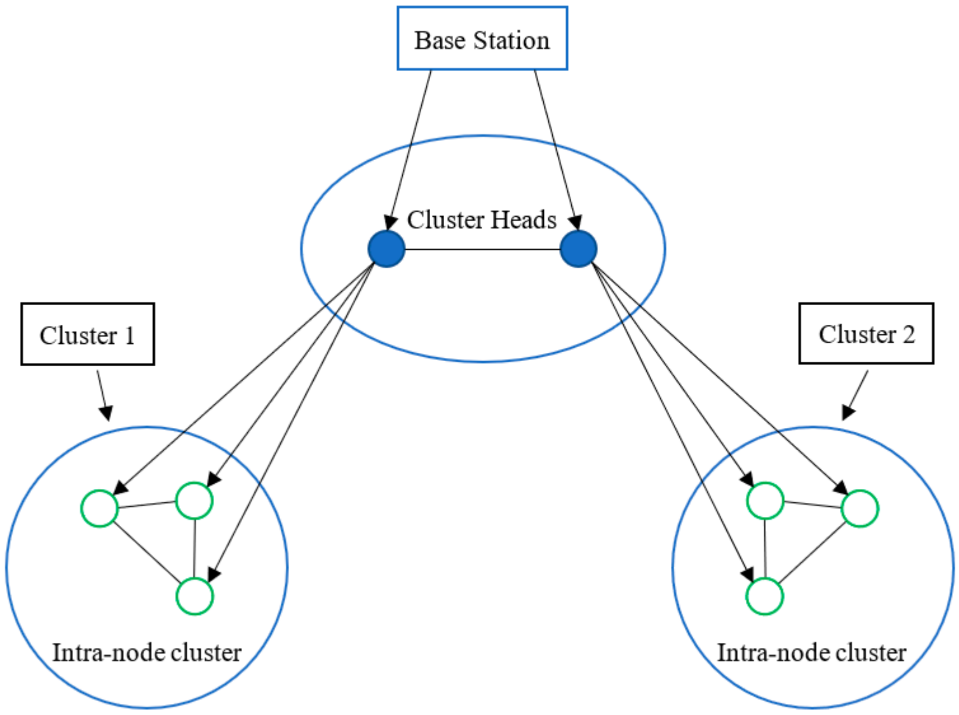

Figure 15. RBS protocol clustering topology.

A broadcast message is sent out from the base station to cluster heads to achieve synchronization between them. After achieving synchronization between the base station and cluster heads, cluster heads send a broadcast message to intra-node clusters in order to synchronize nodes within the cluster. When deciding the cluster head, it is important to consider the energy level of the node, and the distance between cluster head nodes and intra-node clusters. Firstly, a bidirectional synchronization principal can be used to synchronize the base station and cluster heads to estimate clock skew and transmission time. After adjusting their own clock according to the two parameters, cluster heads send broadcast messages to all the intra-node clusters which will be used as a reference point by the intra-node clusters in order to achieve synchronization within the clusters by using the RBS mechanism. In this way, in order to achieve synchronization, a node has to exchange information only with the nodes in the same cluster instead of exchanging information with all the nodes in the network. This process eliminates a large number of information message exchanges and provides high accuracy with efficient energy consumption.

The FTSP is widely known for its high precision and robustness against node and topology failures. The FTSP runs periodic flooding synchronization messages to achieve robustness and implicit topology updates. MAC layer timestamps are used to achieve high precision, whereas sources of delays and uncertainties are eliminated by providing mitigation techniques for them. All these functions provide high accuracy by keeping the error within one microsecond and achieve better robustness than the the TPSN and the RBS protocol; however, these processes consume a large amount of energy, whereas energy is very limited in sensors. One possible solution for this is to reduce the overhead of synchronization messages by not transmitting the rooID and seqNum in synchronization messages, since they are the same as those transmitted before for the first-time synchronization. This would reduce the bandwidth usage and energy consumption. By eliminating these errors in these protocols, a secure time synchronization protocol with low overhead can be proposed which will be able to maintain high accuracy while reducing energy consumption.

In the era of Industry 4.0, Internet of Things (IoT) devices are growing rapidly. Industry 4.0 is related to the increasing digitization and linkage of products and value chains. People and devices are all connected globally in Industry 4.0 [62]. Industry 4.0 also extends to the healthcare domain, named Healthcare 4.0, to assist doctors and patients [63]. There is a need for time synchronization between industry IoT devices or healthcare devices and applications to improve communication and resource availability and to appropriately allocate resources [64]. The issues associated with secure time synchronization should be addressed via authentication and encryption, suspicious packet removal, and the design of new secure time synchronization protocols [65]. Some researchers have proposed a time synchronization scheme based on the NTP, trust management, and blockchain techniques [66]. Blockchain techniques have been used to record time synchronization requests and responses, and the experimental results have illustrated that the proposed approach could achieve security goals.

Time synchronization plays a crucial role in Industry 4.0 and Healthcare 4.0, and secure time synchronization is an essential component. For Industry 4.0 or Healthcare 4.0 applications, the deployment of secure time synchronization should guarantee high, precise time synchronization to satisfy the real-time communication needs of the applications. In addition, it should prevent attacks and keep communication safe. A very promising approach is to integrate time synchronization with blockchain techniques.

Based on the survey results, we answer RQ1 in Section 3 by conducting a detailed analysis of the time synchronization protocols. For RQ2, we answer it in Section 4, and we answer RQ3 in Section 5. The future of secure time synchronization should be developed with blockchain techniques for Industry 4.0 and Healthcare 4.0.

6. Conclusions

Problems associated with time synchronization are not new. In the past, many time synchronization protocols have been proposed, and some of the popular protocols are discussed in this paper. Promising applications of wireless sensor networks are emerging and increasing demand for their use have made it vital to have an error-free and secure time synchronization protocol. However, almost all the current time synchronization protocols have time synchronization errors. To conclude, we have analyzed the current time synchronization protocols and methods for providing secure time synchronization. In addition, we emphasize the importance of secure time synchronization in Industry 4.0 and Healthcare 4.0. The integration of secure time synchronization with blockchain techniques is a promising future research direction. This survey may help researchers in the wired and wireless sensor network community to better understand time synchronization and secure time synchronization.

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

Funding: This research was funded by the Ningbo Major Science & Technology Project 2022Z126 and the Ningbo Commonwealth Project 2019C50052.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The simulation work was supported by Shuxin Liu from Northeastern University, China.

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

References

1. Yang, W.; Wang, Q.; Qi, Y.; Sun, S. Time synchronization attacks in IEEE802. 15.4 e networks. In Proceedings of the 2014 International Conference on Identification, Information and Knowledge in the Internet of Things, Beijing, China, 17–18 October 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 166–169.
2. Dhule, P.; Talmale, G. Secure Time Synchronization for Wireless Sensor Network. *Int. J. Emerg. Technol. Adv. Eng.* **2014**, *4*, 632–635.

3. Poovendran, R.; Wang, C.; Roy, S. *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2007.
4. Kavitha, T.; Sridharan, D. Security vulnerabilities in wireless sensor networks: A survey. *J. Inf. Assur. Secur.* **2010**, *5*, 31–44.
5. Ganeriwal, S.; Pöpper, C.; Čapkun, S.; Srivastava, M.B. Secure Time Synchronization in Sensor Networks. *ACM Trans. Inf. Syst. Secur.* **2008**, *11*, 1–35. [CrossRef]
6. Ranganathan, P.; Nygard, K. Time Synchronization in Wireless Sensor Networks: A Survey. *Int. J. UbiComp* **2010**, *1*, 92–102. [CrossRef]
7. He, J.; Cheng, P.; Shi, L.; Chen, J. SATS: Secure Average-Consensus-Based Time Synchronization in Wireless Sensor Networks. *IEEE Trans. Signal Process.* **2013**, *61*, 6387–6400. [CrossRef]
8. Luo, Y.; Effenberger, F.; Ansari, N. Time Synchronization over Ethernet Passive Optical Networks. *IEEE Commun. Mag.* **2012**, *50*, 136–142. [CrossRef]
9. Xiang, H.; Lei, B. A loose time synchronization regime for wireless sensor network. In Proceedings of the 2010 2nd International Conference on Signal Processing Systems, Dalian, China, 5–7 July 2010; IEEE: Piscataway, NJ, USA, 2010; Volume 2.
10. Rahman, M.; El-Khatib, K. *Secure Time Synchronization for Wireless Sensor Networks Based on Bilinear Pairing Functions*; IEEE: Piscataway, NJ, USA, 2010.
11. Jean, O.; Weiss, A.J. Passive Localization and Synchronization Using Arbitrary Signals. *IEEE Trans. Signal Process.* **2014**, *62*, 2143–2150. [CrossRef]
12. Yildirim, K.S.; Carli, R.; Schenato, L. Adaptive Proportional–Integral Clock Synchronization in Wireless Sensor Networks. *IEEE Trans. Control. Syst. Technol.* **2018**, *26*, 610–623. [CrossRef]
13. Tian, Y.-P.; Chun, S.; Chen, G.; Zong, S.; Huang, Y.; Wang, B. Delay Compensation-Based Time Synchronization under Random Delays: Algorithm and Experiment. *IEEE Trans. Control. Syst. Technol.* **2021**, *29*, 80–95. [CrossRef]
14. Zong, Y.; Dai, X.; Gao, Z. Proportional–Integral Sychronization for Nonidentical Wireless Packet-Coupled Oscillators with Delays. *IEEE Trans. Ind. Electron.* **2021**, *68*, 11598–11608. [CrossRef]
15. Zong, Y.; Liu, S.; Liu, X.; Gao, S.; Dai, X.; Gao, Z. Robust Synchronized Data Acquisition for Biometric Authentication. *IEEE Trans. Ind. Inform.* **2022**, *18*, 9072–9082. [CrossRef]
16. Zong, Y.; Dai, X.; Wei, Z.; Zou, M.; Guo, W.; Gao, Z. Robust Time Synchronisation for Industrial Internet of Things by H∞ Output Feedback Control. *IEEE Internet Things J.* **2022**, *10*, 2021–2030. [CrossRef]
17. Zong, Y.; Dai, X.; Canyelles-Pericas, P.; Gao, Z.; Ng, W.P.; Busawon, K.; Binns, R. *Synchronisation of Packet Coupled Low-Accuracy RC Oscillator Clocks for Wireless Networks*; IEEE: Piscataway, NJ, USA, 2022. [CrossRef]
18. Zong, Y.; Dai, X.; Gao, S.; Canyelles-Pericas, P.; Liu, S. PkCOs: Synchronization of Packet-Coupled Oscillators in Blast Wave Monitoring Networks. *IEEE Internet Things J.* **2022**, *9*, 10862–10871. [CrossRef]
19. De Dominicis, C.M.; Ferrari, P.; Sisinni, E.; Flammini, A.; Pivato, P.; Macii, D. Timestamping performance analysis of IEEE 802.15. 4a systems based on SDR platforms. In Proceedings of the 2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings, Graz, Austria, 13–16 May 2012; IEEE: Piscataway, NJ, USA, 2012.
20. Sichitiu, M.L.; Veerarittiphan, C. Simple, accurate time synchronization for wireless sensor networks. In Proceedings of the 2003 IEEE Wireless Communications and Networking, New Orleans, LA, USA, 16–20 March 2003; IEEE: Piscataway, NJ, USA, 2003; Volume 2, pp. 1266–1273.
21. Rhee, I.-K.; Lee, J.; Kim, J.; Serpedin, E.; Wu, Y.-C. Clock Synchronization in Wireless Sensor Networks: An Overview. *Sensors* **2009**, *9*, 56–85. [CrossRef]
22. He, J.; Chen, J.; Cheng, P.; Cao, X. Secure Time Synchronization in Wireless Sensor Networks: A Maximum Consensus-based Approach. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *25*, 1055–1065. [CrossRef]
23. Zhao, D.; An, D.; Xu, Y. Time synchronization in wireless sensor networks using max and average consensus protocol. *Int. J. Distrib. Sens. Netw.* **2013**, *9*, 192128.
24. Wu, Y.-C.; Chaudhari, Q.; Serpedin, E. Clock Synchronization of Wireless Sensor Networks. *IEEE Signal Process. Mag.* **2011**, *28*, 124–138. [CrossRef]
25. Khediri, S.E.; Nasri, N.; Samet, M.; Wei, A.; Kachouri, A. Analysis Study of Time Synchronization Protocols in Wireless Sensor Networks. *Int. J. Distrib. Parallel Syst.* **2012**, *3*, 155–165. [CrossRef]
26. Maggs, M.K.; O'Keefe, S.G.; Thiel, D.V. Consensus Clock Synchronization for Wireless Sensor Networks. *IEEE Sens. J.* **2012**, *12*, 2269–2277. [CrossRef]
27. Kopetz, H.; Ochsenreiter, W. Clock Synchronization in Distributed Real-Time Systems. *IEEE Trans. Comput.* **1987**, *100*, 933–940. [CrossRef]
28. Ganeriwal, S.; Kumar, R.; Srivastava, M.B. Timing-sync protocol for sensor networks. In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems 2003, Los Angeles, CA, USA, 5–7 November 2003; pp. 138–149.
29. Lv, G.; Yu, F.; Dong, C. An Implementation of Low-Power Data Transmission Based on Time Synchronization. In Proceedings of the 2012 8th International Conference on Wireless Communications, Networking and Mobile Computing, Barcelona, Spain, 8–10 October 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 1–5.
30. Maróti, M.; Kusy, B.; Simon, G.; Lédeczi, A. The flooding time synchronization protocol. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, Baltimore, MD, USA, 3–5 November 2004; pp. 39–49.

31. Zhang, F.; Bu, L.; Wang, L.; Zhao, J.; Chen, X.; Zhang, T.; Li, X. Modeling and Evaluation of Wireless Sensor Network Protocols by Stochastic Timed Automata. *Electron. Notes Theor. Comput. Sci.* **2013**, *296*, 261–277. [CrossRef]
32. Fan, R.; Xu, W.; Wu, L.; Han, L. Performance Comparison on Network-Wide Time Synchronization Approaches in Wireless Sensor Networks. In Proceedings of the 2013 IEEE International Conference on Microwave Technology & Computational Electromagnetics, Rio de Janeiro, Brazil, 4–7 August 2013. [CrossRef]
33. Elson, J.; Girod, L.; Estrin, D. Fine-Grained Network Time Synchronization Using Reference Broadcasts. *ACM SIGOPS Oper. Syst. Rev.* **2002**, *36*, 147–163. [CrossRef]
34. Yang, Y.; Sun, Y. Securing time-synchronization protocols in sensor networks: Attack detection and self-healing. In Proceedings of the IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference, Rio de Janeiro, Brazil, 4–8 December 2008; IEEE: Piscataway, NJ, USA; pp. 1–6.
35. Datla, D.; Chen, X.; Tsou, T.; Raghunandan, S.; Hasan, S.M.; Reed, J.; Dietrich, C.; Bose, T.; Fette, B.; Kim, J.-H. Wireless Distributed Computing: A Survey of Research Challenges. *IEEE Commun. Mag.* **2012**, *50*, 144–152. [CrossRef]
36. Mazur, D.C.; Entzminger, R.A.; Kay, J.A.; Morell, P.A. Time Synchronization Mechanisms for the Industrial Marketplace. *IEEE Trans. Ind. Appl.* **2017**, *53*, 39–46. [CrossRef]
37. Thorød, P. Firmware for Synchronizing Chip-Scale Atomic Clock to GPS. Master's Thesis, Chalmers University of Technology, Göteborg, Sweden, 2015.
38. Shi, B.; Zhang, D.; Hu, J. Preliminary investigation in wide area protection implementation using IEEE 1588 precision time protocol. In Proceedings of the 2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), Beijing, China, 11–16 October 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 43–47.
39. Zhang, Z.; Gong, S.; Dimitrovski, A.D.; Li, H. Time Synchronization Attack in Smart Grid: Impact and Analysis. *IEEE Trans. Smart Grid* **2013**, *4*, 87–98. [CrossRef]
40. Fan, R.; Chakraborty, I.; Lynch, N. Clock synchronization for wireless networks. In Proceedings of the Principles of Distributed Systems 8th International Conference, OPODIS 2004, Grenoble, France, 15–17 December 2004; Revised Selected Papers 8 2005. Springer: Berlin/Heidelberg, Germany; pp. 400–414.
41. Novick, A.N.; Lombardi, M.A. Practical limitations of NTP time transfer. In Proceedings of the 2015 Joint Conference of the IEEE International Frequency Control Symposium & the European Frequency and Time Forum, Denver, CO, USA, 12–16 April 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 570–574.
42. Zhan, Y.; Lei, W.; Zhang, W. Survey on time synchronization technology in networks. In Proceedings of the 27th Chinese Control and Decision Conference (2015 CCDC), Qingdao, China, 23–25 May 2015; IEEE: Piscataway, NJ, USA, 2015.
43. Malhotra, A.; Cohen, I.E.; Brakke, E.; Goldberg, S. Attacking the network time protocol. In Proceedings of the Cryptology ePrint Archive, San Diego, CA, USA, 21–24 February 2016.
44. Singh, V.; Sharma, S.; Sharma, T.P. Time Synchronization in WSN: A Survey. *Int. J. Enhanc. Res. Sci. Technol. Eng.* **2013**, *2*, 61–67.
45. Hur, K.; Sohn, W.; Kim, J.; Lee, Y. A Power-Efficient Mechanism of IEEE 802.15. 6 WBAN for Wireless USB Support. *Int. J. Softw. Eng. Its Appl.* **2013**, *7*, 35–50.
46. Karthik, S.; Kumar, A.A. Challenges of wireless sensor networks and issues associated with time synchronization. In Proceedings of the UGC Sponsored National Conference on Advanced Networking and Applications, Udumalpet, India, 27 March 2015; pp. 19–23.
47. Sonone, S.; Sakhare, A.P. Review on Time Synchronization Approaches in Wireless Sensor Networks. *I PASJ Int. J. Comput. Sci. (IIJCS)* **2015**, *3*, 13–19.
48. Minar, N. A Survey of the NTP Network. December 1999. Available online: http://www.media.mit.edu/~nelson/research/ntp-survey99/ (accessed on 10 January 2023).
49. Van Greunen, J.; Rabaey, J. Lightweight time synchronization for sensor networks. In Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications, San Diego, CA, USA, 19 September 2003; pp. 11–19.
50. Sarvghadi, M.A.; Wan, T.C. Overview of time synchronization protocols in wireless sensor networks. In Proceedings of the 2014 2nd International Conference on Electronic Design (ICED), Penang, Malaysia, 19–21 August 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 204–209.
51. Karl, H.; Willig, A. *Protocols and Architectures for Wireless Sensor Networks*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
52. Meier, S.; Weibel, H.; Weber, K. IEEE 1588 Syntonization and Synchronization Functions Completely Realized in Hardware. In Proceedings of the 2008 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, Austin, TX, USA, 21–26 September 2014; IEEE: Piscataway, NJ, USA, 2008; pp. 1–4.
53. Idrees, Z.; Granados, J.; Sun, Y.; Latif, S.; Gong, L.; Zou, Z.; Zheng, L. IEEE 1588 for Clock Synchronization in Industrial IoT and Related Applications: A Review on Contributing Technologies, Protocols and Enhancement Methodologies. *IEEE Access* **2020**, *8*, 155660–155678. [CrossRef]
54. Pandey, P.; Pratap, B.; Pandey, R.S. Analysis and Design of Precision Time Protocol System Based on IEEE1588 Standards. In Proceedings of the 2019 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 17–19 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1963–1967.
55. Mills, D.L. NTP: The Network Time Protocol. Available online: http://www.ntp.org/ (accessed on 10 January 2023).

56. Hu, X.; Park, T.; Shin, K.G. Attack-tolerant time-synchronization in wireless sensor networks. In Proceedings of the IEEE INFOCOM 2008—The 27th Conference on Computer Communications, Phoenix, AZ, USA, 13–18 April 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 41–45.
57. Ganeriwal, S.; Čapkun, S.; Han, C.C.; Srivastava, M.B. Secure time synchronization service for sensor networks. In Proceedings of the 4th ACM Workshop on Wireless Security, Vancouver, BC, Canada, 27–28 October 2008; pp. 97–106.
58. Manzo, M.; Roosta, T.; Sastry, S. Time synchronization attacks in sensor networks. In Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks, Alexandria, VA, USA, 7 November 2005; pp. 107–116.
59. Boukerche, A.; Turgut, D. Secure Time Synchronization Protocols for Wireless Sensor Networks. *IEEE Wirel. Commun.* **2007**, *14*, 64–69. [CrossRef]
60. Dong, W.; Liu, X. Robust and Secure Time-Synchronization against Sybil Attacks for Sensor Networks. *IEEE Trans. Ind. Inform.* **2015**, *11*, 1482–1491. [CrossRef]
61. Sun, K.; Ning, P.; Wang, C. TinySeRSync: Secure and resilient time synchronization in wireless sensor networks. In Proceedings of the 13th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 30 October 2006; pp. 264–277.
62. Wójcicki, K.; Biegańska, M.; Paliwoda, B.; Górna, J. Internet of Things in Industry: Research Profiling, Application, Challenges and Opportunities—A Review. *Energies* **2022**, *15*, 1806. [CrossRef]
63. Surati, S.; Patel, S.; Surati, K. *Background and Research Challenges for Fc for Healthcare 4.0*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 37–53.
64. Dalwadi, N.; Padole, M. An Insight into Time Synchronization Algorithms in IoT. *Data Eng. Appl.* **2019**, *2*, 285–296. [CrossRef]
65. Yiğitler, H.; Badihi, B.; Jäntti, R. Overview of Time Synchronization for IoT Deployments: Clock Discipline Algorithms and Protocols. *Sensors* **2020**, *20*, 5928. [CrossRef]
66. Fan, K.; Shi, Z.; Su, R.; Bai, Y.; Huang, P.; Zhang, K.; Li, H.; Yang, Y. Blockchain-Based Trust Management for Verifiable Time Synchronization Service in IoT. *Peer-to-Peer Netw. Appl.* **2022**, *15*, 1152–1162. [CrossRef]