sensors

MDPI

Article

Fast and Robust Time Synchronization with Median Kalman Filtering for Mobile Ad-Hoc Networks

Young Jeon, Taehong Kim and Taejoon Kim *

School of Information and Communication Engineering, Chungbuk National University, Chungju 28644, Korea; jeony9672@cbnu.ac.kr (Y.J.); taehongkim@cbnu.ac.kr (T.K.)
* Correspondence: ktjcc@chungbuk.ac.kr

Abstract: Time synchronization is an important issue in ad-hoc networks for reliable information exchange. The algorithms for time synchronization in ad-hoc networks are largely categorized into two types. One is based on a selection of a reference node, and the other is based on a consensus among neighbor nodes. These two types of methods are targeting static environments. However, synchronization errors among nodes increase sharply when nodes move or when incorrect synchronization information is exchanged due to the failure of some nodes. In this paper, we propose a synchronization technique for mobile ad-hoc networks, which considers both the mobility of nodes and the abnormal behaviors of malicious or failed nodes. Specifically, synchronization information extracted from a median of the time information of the neighbor nodes is quickly disseminated. This information effectively excludes the outliers, which adversely affect the synchronization of the networks. In addition, Kalman filtering is applied to reduce the synchronization error occurring in the transmission and reception of time information. The simulation results confirm that the proposed scheme has a fast synchronization convergence speed and low synchronization error compared to conventional algorithms.

Keywords: time synchronization; ad-hoc network; fast median; Kalman filter

Citation: Jeon, Y.; Kim, T.; Kim, T. Fast and Robust Time Synchronization with Median Kalman Filtering for Mobile Ad-Hoc Networks. *Sensors* **2021**, *21*, 590. https://doi.org/10.3390/s21020590

Received: 10 December 2020
Accepted: 12 January 2021
Published: 15 January 2021

1. Introduction

An ad-hoc network is adopted for various situations, such as environmental monitoring, military operation, and disaster recovery [1]. In an ad-hoc network, the nodes are equipped with computing and sensing devices operating at low power, and an accurate time synchronization is required in collecting and sharing data measured by the sensing devices [2,3]. An accurate time synchronization is a core functionality for distributed information gathering and control. For instance, in measuring the occurrence of acoustic or seismic signals over multiple probing nodes, the performance of time synchronization over the probing nodes greatly affects the accuracy of the measurement. For a high-rate time division multiple access (TDMA) system, the performance of medium access control (MAC) layer scheduling is also significantly affected by time synchronization among nodes. In addition, time synchronization plays an important role for a distributed logging system, network security, and power management system.

In a cellular communication system, mobile nodes are synchronized according to the preamble signal transmitted from a base station. Usually, a base station has enough power to send this signal, which covers the whole cell site of the base station. However, in an ad-hoc network without a base station, the available radio resources and the power for each node are scarce and limited [4,5]. Accordingly, this network requires an efficient synchronization algorithm that is robust against environmental changes along with low power consumption in the synchronization message exchanges [6].

One of the representative synchronization methods for ad-hoc networks is a reference node-based algorithm, which includes Flooding Time Synchronization Protocol (FTSP) [7]

and PulseSync [8]. In this method, one of the nodes in a network is selected as a reference node, and the time of the reference node becomes a global network time. Hence, the time synchronization messages are traversed from the reference node to all the nodes in the network. The nodes receiving this synchronization message should estimate the global network time because of the time gap between transmitting and receiving a time synchronization message. Then, they update their own time information. Afterwards, the updated synchronization messages are transmitted to their neighbor nodes. All nodes in the network are synchronized by repeating this process. Various delays and errors occur in the process of receiving and transmitting the synchronization messages. In FTSP, MAC-layer timestamping is used to remove delays that occur when a synchronization message passes through communication layers [9–11]. However, when transmitting a message to a neighbor node, a hop delay is inevitable in every hop [12,13]. Hop delay is dependent on the distance between nodes and the message processing time. As the maximum hop count among nodes is high in a widely distributed network, hop delay remains the most important issue in the reference node-based time synchronization method [14].

Another synchronization method is a consensus-based algorithm, where each node exchanges synchronization messages with its neighbor nodes, and the synchronization message contains the time information reflecting the time information of its neighbor nodes [15]. Gradient Time Synchronization Protocol (GTSP) [16], Consensus-based Clock Synchronization (CoSyn) [17], and Random Broadcast-based Distributed consensus clock Synchronization (RBDS) [18] belong to the consensus-based synchronization algorithm. GTSP achieves time synchronization by averaging the relative rate and offset among nodes. Specifically, each node exchanges synchronization messages with its neighboring nodes in every round and stores the received time information in a table. At the end of each round, its time information is updated by averaging the rates and offsets stored in the table. The repetition of this process results in the time synchronization among all the nodes. However, if the nodes are deployed in a large area, the number of rounds required in achieving the synchronization increases sharply [19,20]. Moreover, as the number of nodes in a network increases, some nodes may malfunction due to hardware or software failure, which deteriorates the performance of time synchronization.

Naturally, a time synchronization algorithm needs to have a fast convergence in achieving consensus among nodes and to exclude malfunctioning nodes from the synchronization process. In order to achieve these goals, the median value of the time information of the nodes is adopted [21,22], because, in excluding outliers, a median—rather than an average—is a better choice.

In this paper, for mobile ad-hoc networks (MANETs) [23,24], we propose a consensus-based Median Kalman-filtering Time Synchronization (MKTS) scheme, which reduces the convergence time of synchronization by rapidly spreading the time information extracted from the median values of synchronization messages to an entire network [25]. Moreover, the proposed scheme uses a Kalman filter [26–29] in processing the synchronization messages to effectively remove the errors occurring in the synchronization. Since the median values may vary according to the location where each node is located, a Fast-median value with a reduced regional dependence is proposed. Both the mobility of nodes and the failure of nodes are considered to evaluate the performance of MKTS. The proposed algorithm has an excellent performance in mobile environments. Simulation results show that the proposed scheme has a fast convergence speed and robustness against its environmental changes. Specifically, a joining of new nodes, a removal of existing nodes, and a failure of some nodes rarely affect the performance of the proposed scheme. The target precision level of MKTS is 20 μs for both the static and mobile scenarios, and the synchronization criteria for other protocols are presented in [30,31].

The remainder of this paper is organized as follows: Section 2 describes the system model and the process of MKTS synchronization algorithms. Section 3 compares the performances of MKTS, FTSP, and GTSP in static and mobile environments with some

failed nodes. Section 4 discusses the evaluation results and a future research topic, and Section 5 concludes the paper.

2. Proposed Time Synchronization Method

2.1. Clock Model

A hardware clock embedded into a node is expressed as Equation (1)

$$H_i(t) = h_i \times t + o_i, \tag{1}$$

where t is the actual time, h_i is the rate of the hardware clock, and o_i is the offset of the hardware clock. The speed of the hardware clock has an error of $|h_i - 1|$. In manufacturing a clock counter, it is impossible to make an infinitely accurate one. An actual clock rate will be a little bit faster or slower than a perfect clock. Similarly, the sensitivity of a clock rate to ambient temperate cannot be perfectly regulated. Likewise, h_i and o_i are inherent characteristics of the hardware of node i and cannot be read or modified by node i. Hence, a logical clock is defined and adjusted to achieve synchronization among nodes. The logical clock $L_i(t)$ of node i is described in Equation (2).

$$L_i(t) = l_i H_i(t) + \beta_i = l_i h_i t + l_i o_i + \beta_i, \tag{2}$$

where l_i is the relative logical clock rate representing the relative rate ratio of the hardware clock and the logical clock, β_i is the logical offset of node i. Logical clock is adjusted by updating l_i and β_i. In Equation (2), $x_i = l_i h_i$ is denoted as an absolute logical clock rate, and when all the nodes have the same absolute logical clock rate, i.e., with N nodes in a network and $x_1 = x_2 = \cdots = x_N$, the clock rate synchronization for the entire network is achieved. In the case of GTSP, the update of x_i requires the access to the unreadable parameter h_i. Accordingly, instead of directly updating x_i, l_i is adjusted as follows:

$$l_i(t_{n+1}) \leftarrow \frac{\sum_{j \in N_i} \frac{x_j(t_n)}{h_i(t_n)} + l_i(t_n)}{|N_i| + 1}, \tag{3}$$

where t_n is the time instant receiving a synchronization message from a neighbor node in the nth round, N_i is a set of neighboring nodes of node i. In GTSP, the synchronization for the offset uses a similar method of the rate synchronization as follows:

$$\beta_i(t_{k+1}) \leftarrow \beta_i(k_k) + \frac{\sum_{j \in N_i} \left(L_j(t_k) - L_i(t_k) \right)}{|N_i| + 1}. \tag{4}$$

where Equations (3) and (4) are versions of the GTSP update schemes, rephrased.

2.2. MKTS Message and Table Structure

Each node has a unique ID, and the structure of the synchronization message of MKTS is shown in Figure 1.

Node ID
Hardware Clock
Logical Clock
Relative Logical Clock Rate

Figure 1. Median Kalman-filtering Time Synchronization (MKTS) packet structure.

The table structure for node i to store the information received from node j is shown in Figure 2. Receiving a new message, a new row is added.

ID	$H_i(t_n)$	$L_i(t_n)$	$H_j(t_n)$	$L_j(t_n)$	$R_{ij}(t_n)$	l_j	S_j

Figure 2. MKTS message table structure.

The ID of node j is stored at the first field. $H_j(t_n)$ is the hardware clock of node j received in the nth round, and $H_i(t_n)$ is the hardware clock of node i measured at the instant of receiving $H_j(t_n)$ in the nth round. $L_j(t_n)$ is the logical clock received from node j. $L_i(t_n)$ is the logical clock of node i measured at the instant of receiving $L_j(t_n)$ in the nth round. $R_{ij}(t_n)$ is the relative hardware clock rate in the nth round, which is the ratio of hardware clock rate of node j to the hardware clock rate of node i. l_j is the relative logical clock rate of node j. S_j is the number of messages received from node j.

2.3. Relative Hardware Clock Rate

As shown in Equation (3), $\frac{x_j(t_n)}{h_i(t_n)}$ is required in synchronizing the relative logical clock rate l_i. Without access to the unreadable $h_i(t_n)$, $\frac{x_j(t_n)}{h_i(t_n)}$ can be obtained as follows [32]:

$$\frac{x_j(t_n)}{h_i(t_n)} = \frac{\frac{L_j(t_n)-L_j(t_{n-1})}{t_n-t_{n-1}}}{\frac{H_i(t_n)-H_i(t_{n-1})}{t_n-t_{n-1}}} = \frac{L_j(t_n)-L_j(t_{n-1})}{H_i(t_n)-H_i(t_{n-1})} = \frac{\Delta L_j}{\Delta H_i} = \frac{\Delta H_j l_j}{\Delta H_i} = \frac{H_j(t_n)-H_j(t_{n-1})}{H_i(t_n)-H_i(t_{n-1})} \times l_j. \tag{5}$$

$R_{ij}(t_n)$ is expressed as Equation (6).

$$R_{ij}(t_n) = \frac{h_j(t_n)}{h_i(t_n)} = \frac{\frac{H_j(t_n)-H_j(t_{n-1})}{t_n-t_{n-1}}}{\frac{H_i(t_n)-H_i(t_{n-1})}{t_n-t_{n-1}}} = \frac{H_j(t_n)-H_j(t_{n-1})}{H_i(t_n)-H_i(t_{n-1})}. \tag{6}$$

Hence, $\frac{x_j(t_n)}{h_i(t_n)} = R_{ij}(t_n)l_j$ is satisfied. In order to increase the accuracy of $R_{ij}(t_n)$, the errors caused by delay are reduced by adopting an integral filter [33], which takes the weighted moving average for the continual input of $R_{ij}(t_n)$. Since the weighted current $R_{ij}(t_n)$ is added to the previous average, the error can be filtered out. The filtered version $R'_{ij}(t_n)$ can be obtained as Equation (7).

$$R'_{ij}(t_n) = \frac{\min\left[S_j, 5\right]-2}{\min\left[S_j, 5\right]-1} \times R'_{ij}(t_{n-1}) + \frac{1}{\min\left[S_j, 5\right]-1} \times R_{ij}(t_n),\ S_j \geq 2. \tag{7}$$

If the maximum S_j is very high, the averaging is as good as being taken over by very many previous $R_{ij}(t_n)$s, and the impact of the current $R_{ij}(t_n)$ will be marginal. Accordingly, a high S_j results in a good performance in eliminating error; however, too high S_j may result in the deterioration of catching the actual changes of $R_{ij}(t_n)$ like clock rate changes from ambient temperature change, which should be reflected as not being filtered out. In selecting the maximum S_j, FTSP, which adopts least square method using 4–8 previously received messages is referred. Since the proposed method is focused on mobile scenarios, relatively small $S_j = 5$ is selected as a maximum value.

2.4. Update Rule

When a node receives synchronization messages from its neighbor nodes, it may update its time information to an average or a consensus value of the received time information. Alternatively, it can be synchronized to the time of a specific node [34]. For instance, if node i determines to follow the clock of node j, the absolute logical clock rate and logical clock of node i must be modified to the values of node j. Accordingly,

$x_i(t_n) \leftarrow x_j(t_n)$ and $L_i(t_n) \leftarrow L_j(t_n)$ should be achieved by adjusting l_i and β_i as shown in Equations (8) and (9).

$$l_i \leftarrow R_{ij}(t_n) \times l_j = \frac{h_j}{h_i} \times l_j = \frac{x_j}{h_i}, \tag{8}$$

$$\beta_i \leftarrow L_j(t_n) - l_i \times H_i(t_n). \tag{9}$$

In MKTS, each node updates its clock information by selecting a specific target node and changing its time information following the selected one. The criterion of selecting the target node is an integral part of the proposed scheme, and the detailed selection process is described in what follows. Basically, the selected target node has a median logical clock from among its neighboring nodes. This method plays an important role in excluding outliers and expediates the convergence of the synchronization process. Kalman filtering follows this median-based approach to reduce the errors caused by noise and delay. Using both the median-based approach and Kalman filtering, a robust time synchronization is achieved against the mobility of the nodes in a MANET.

However, directly adopting a medium of logical clocks from among neighbor nodes can cause a problem. Since each node selects a medium node from among its own 1-hop neighboring nodes, if the nodes of a MANET are distributed over a large area, the logical clocks of the selected mediums will be different depending on the geographic location of each node. Accordingly, oscillations may occur in exchanging the selected medium logical clocks at each synchronization round. For instance, as shown in Figure 3, there are four clusters represented as gray areas. The clusters with orange nodes and blue nodes have a single intersection node colored purple. This purple node can be synchronized to the cluster with orange nodes at one round, and it can be synchronized to other cluster with blue nodes at some other round. This causes oscillation to the purple node. As many rounds pass, this oscillation will be diminished; however, if the nodes are deployed over a wide area, the convergence will take a much longer time. These oscillations hinder the synchronization and slow down the convergence. Therefore, instead of adopting the logical clock of the selected median node, Fast-median (F-Median) is proposed. F-Median selects the fastest logical clock within a certain range, where the range is centered at the logical clock of the medium node and has a predefined span. F-Median can reduce the oscillation while excluding outliers.

Figure 3. Exemplified network with four clusters and oscillating nodes.

Figure 4 depicts F-Median in the nth synchronization round, where $t(n)$ is the start time of the nth synchronization round, black circles are the received logical clocks, and the black squares are the estimated logical clocks at the synchronization round boundary.

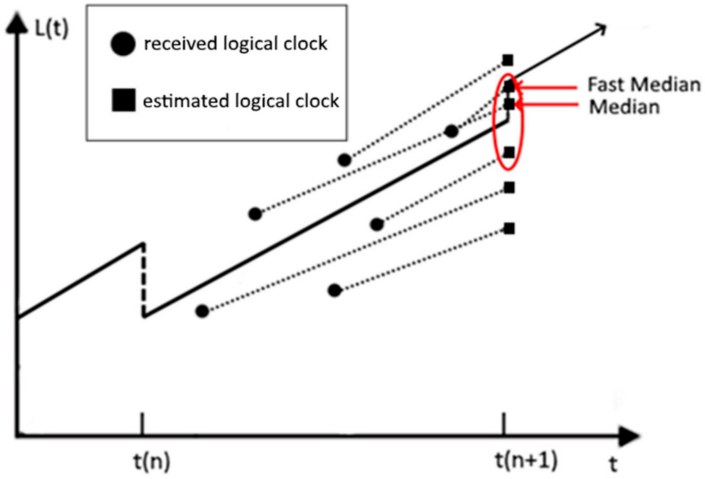

Figure 4. Median value in synchronization round.

In order to select a node with a median logical clock from among neighboring nodes, the exact logical clocks at the synchronization round boundary $t(n+1)$ should be estimated. When node i receives a message from node j at t_n belonging to the nth round, i.e., $t_n \in [t(n),\ t(n+1))$, the logical clock of node j at $t(n+1)$ can be estimated as follows:

$$L_j(t(n+1)) = L_j(t_n) + (H_i(t(n+1)) - H_i(t_n)) \times R_{ij}(t_n) \cdot l_j. \tag{10}$$

After the estimations, the neighboring nodes are sorted according to the estimated logical clocks. Subsequently, a medium node is selected from the sorted neighboring nodes. The process of selecting the median node can be expressed as follows:

$$L_i^{Med}(t(n+1)) = Median_{v \in N_i}(L_v(t(n+1)), \tag{11}$$

where $L_i^{Med}(t(n+1))$ is the median value selected by node i at $t(n+1)$, and $Median_{v \in N_i}(\cdot)$ is a function that returns the median value from the 1-hop neighbor node set N_i of node i. Then, a range centered at the selected logical clock with the predefined span is determined, and the node with the fastest logical clock within that range is finally selected. The detailed F-Median process is described in Algorithm 1.

Algorithm 1. Fast Median.

Input:
THRESHOLD: upper bound of F-Median range
Output:
1: **for each** $j \in N_i$
2: $L_j(t(n+1)) \leftarrow L_j(t_n) + (H_i(t(n+1)) - H_i(t_n)) \times R_{ij}(t_n) \times l_j$
3: **end**
4: $\mathrm{Sort}_{v \in N_i}(L_v(t(n+1)))$
5: $M \leftarrow \arg\ Median_{v \in N_i}(L_v(t(n+1)))$
6: **for each** $j \in N_i$
7: $e \leftarrow L_M(t(k+1)) - L_j(t(k+1))$
8: **if** $e < THRESHOLD$
9: $Fastest\_Median\_Value \leftarrow j$
10: **break**
11: **end**
12: **end**

Kalman filtering follows the F-Median process to remove the errors due to topology change or randomness in transmission and reception. If node i selects node j as a F-Median node, the input to the Kalman filter can be expressed as Equations (12) and (13) [35].

$$T_n = L_i(t_n) - L_j(t_n), \tag{12}$$

$$\begin{aligned} D_n &= \frac{x_j(t_n) - x_i(t_n)}{x_i(t_n)} = \frac{L_j(t_n) - L_j(t_{n-1})}{L_i(t_n) - L_i(t_{n-1})} - 1 \\ &= \frac{l_j \times \left(H_j(t_n) - H_j(t_{n-1})\right) \times t}{l_i \times (H_i(t_n) - H_i(t_{n-1})) \times t} - 1 = \frac{R_{ij} l_j}{l_i} - 1, \end{aligned} \tag{13}$$

where T_n is the logical clock difference between node i and node j in the nth round. D_n is the ratio of the absolute logical clock rate difference between node i and node j to the absolute logical clock rate of node i in the nth round. $\boldsymbol{x}_n$ is a column vector having T_n and D_n as elements as shown in (14).

$$\boldsymbol{x}_n = \begin{bmatrix} T_n \\ D_n \end{bmatrix}. \tag{14}$$

where $\boldsymbol{x}_n$ is the observed values in the nth round in a real environment, and a model for the environment needs to be designed. The parameters for the environment adopted in MKTS are shown in Equations (15) and (16).

$$\Delta H_i = H_i(t(n+1)) - H_i(t_n), \tag{15}$$

$$\boldsymbol{A}_n = \begin{bmatrix} 1 & \Delta H_i \\ 0 & 1 \end{bmatrix}, \tag{16}$$

where $\boldsymbol{A}_n$ is the environment model in the nth round. The initial input vector is $\boldsymbol{x}_0$, and the initial covariance of the input data is arbitrarily set to $\boldsymbol{P}_0$. From these initial values and Equations (17) and (18), $\boldsymbol{x}_p$ and $\boldsymbol{P}_p$ are obtained as follows:

$$\boldsymbol{x}_p = \boldsymbol{A}_n \boldsymbol{x}_n, \tag{17}$$

$$\boldsymbol{P}_p = \boldsymbol{A}_n \boldsymbol{P}_n \boldsymbol{A}_n^T + \boldsymbol{B}_n \boldsymbol{Q} \boldsymbol{B}_n^T, \tag{18}$$

where $\boldsymbol{x}_p$ is a predicted input data, and $\boldsymbol{P}_p$ is a predicted covariance of the input data. In addition, $\boldsymbol{x}_n$ is the actual input data of the nth round, and $\boldsymbol{P}_n$ is the covariance of the actual input data. $\boldsymbol{B}_n$ is the transformation matrix of $\boldsymbol{Q}$, where $\boldsymbol{Q}$ is the noise matrix generated in the process of predicting the covariance shown as follows:

$$\boldsymbol{Q} = \begin{bmatrix} \delta^2 & 0 \\ 0 & \varphi^2 \end{bmatrix}, \tag{19}$$

where δ and φ are Gaussian noises. In order to deal with nonlinearity in the estimation, instead of Equation (18), (20) can be used [36].

$$\boldsymbol{P}_p = \boldsymbol{A}_n \boldsymbol{P}_n \boldsymbol{A}_n^T + \begin{bmatrix} 1 & \frac{\Delta H_i}{2} \\ 0 & 1 \end{bmatrix} \boldsymbol{Q} \begin{bmatrix} 1 & \frac{\Delta H_i}{2} \\ 0 & 1 \end{bmatrix} \Delta H_i, \tag{20}$$

where ΔH_i is obtained from Equation (15). Moreover, in Equation (20), the error occurring in the process of predicting the covariance of a nonlinearly operating model is calculated using the Riccati equation. $\boldsymbol{P}_p$ is used to calculate the Kalman gain using the relationship between the actual and predicted data as follows:

$$\boldsymbol{K}_n = \boldsymbol{P}_p \boldsymbol{U}_n^T \left(\boldsymbol{U}_n \boldsymbol{P}_p \boldsymbol{U}_n^T + R \right)^{-1}, \tag{21}$$

$$\boldsymbol{U}_n = \begin{bmatrix} 1 & \Delta H_i \end{bmatrix}, \tag{22}$$

where $\boldsymbol{K}_n$ is the Kalman gain in the nth round, R is the Gaussian noise of the observed data, and $\boldsymbol{U}_n$ is a transformation matrix of measured values. The Kalman gain is used to adjust the actual data and the predicted data, $\boldsymbol{y}_n$ and $\boldsymbol{y}_p$, respectively, which are given by Equations (23) and (24):

$$\boldsymbol{y}_n = \boldsymbol{U}_n \boldsymbol{x}_p + \boldsymbol{u}, \tag{23}$$

$$\boldsymbol{y}_p = \boldsymbol{U}_n \boldsymbol{x}_p, \tag{24}$$

where $\boldsymbol{u}$ is noise generated in the process of measuring data. $\boldsymbol{x}_n$ and $\boldsymbol{P}_n$ are updated following Equations (25) and (26).

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_p + \boldsymbol{K}_n\left(\boldsymbol{y}_n - \boldsymbol{y}_p\right) = \begin{bmatrix} T_{n+1} \\ D_{n+1} \end{bmatrix}, \tag{25}$$

$$\boldsymbol{P}_{n+1} = (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{U}_n)\boldsymbol{P}_p. \tag{26}$$

After updating $\boldsymbol{x}_{n+1}$ and $\boldsymbol{P}_{n+1}$, node i updates its own logical clock and relative logical clock rate as follows:

$$L_i(t(n+1)) = L_i(t(n+1)) - T_{n+1}, \tag{27}$$

$$l_i = l_i \times (D_{n+1} + 1). \tag{28}$$

The synchronization process is repeated every round to optimize the predicted data and covariance and to remove errors caused by noise and delay. In practical protocols, the synchronization round has the same period with beacon transmission period, and some portion of the super frame is allocated to time synchronization message exchanges. A flowchart showing the whole synchronization process is depicted in Figure 5.

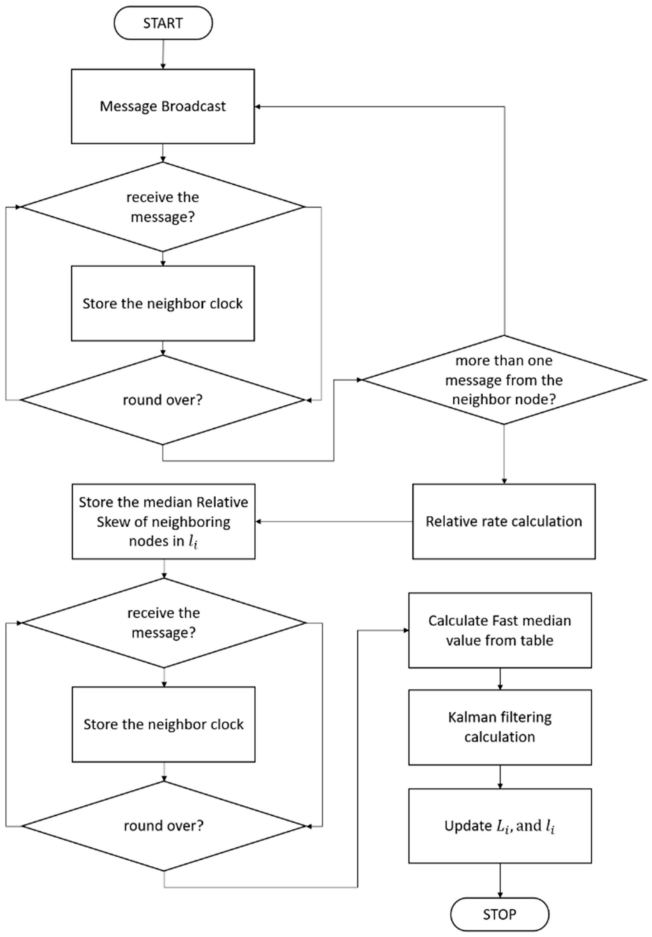

Figure 5. Flowchart of the whole process of MKTS.

3. Performance Evaluation

The performance of MKTS is verified by comparing MKTS with other conventional algorithms of FTSP and GTSP, under the environment considering mobile nodes and node failures. The specific evaluation plan is to compare MKTS with FTSP and GTSP in static and mobile environments, to compare MKTS with FTSP and GTSP by varying the speed of nodes and the size of the network area, and finally, to evaluate MKTS with an increasing number of malfunctioning nodes. In this comparative evaluation, FTSP and GTSP are not the state-of-the-art techniques, however, they are still the representative techniques for ad-hoc network time synchronization. Moreover, the proposed scheme focuses on the mobility of nodes, which is not supported by most of the time synchronization techniques. In addition, the most consensus-based algorithms follow the basic philosophy of GTSP, i.e., each adjusts its own time information according to the average information of its neighbor nodes. Accordingly, the contribution of the proposed scheme can be evaluated by comparing with FTSP and GTSP. For the performance analysis, the network simulator OPNET [37] is used, which is an established commercial network simulator readily supporting practical protocols like IEEE 802.11 and IEEE 802.15.4. In measuring the performance of the proposed algorithm, Maximum Network Error A_e and Maximum Neighbor Error N_e are used, where A_e is the largest logical clock difference among nodes, and N_e is the largest logical clock difference between neighboring nodes in an entire network. A_e and N_e are expressed as follows:

$$A_e = max_{v,w \in N}\{|L_v(t) - L_w(t)|\}, \tag{29}$$

$$N_e = \max_{v,w \in N_v}\{|L_v(t) - L_w(t)|\}, \tag{30}$$

where N is the set of all nodes in a network, and N_v is the set of node v's 1-hop neighbor nodes. To compare the performance, the placement of the nodes is depicted in Figure 6. The parameters of the simulations are summarized in Table 1.

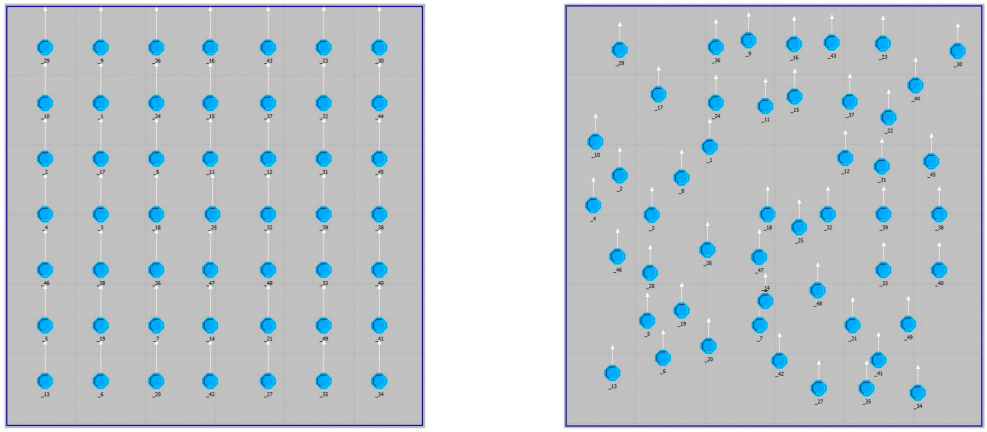

Figure 6. Mesh and random topologies for node placement.

Table 1. System parameters.

Parameter	Value
Number of Nodes	49
X Dimension	600 m
Y Dimension	600 m
Communication Range	110 m
Topology	Grid/Random
Mobility	Random Direction
Beacon Interval	30 s
Hardware Clock Drift	−30~+30 (μs)
Hardware Clock Drift Variation	−5~+5 (μs)

3.1. Performance Comparison with Conventional Methods

Figure 7 shows A_e over time while the nodes are static for a mesh (left) topology and a random (right) topology. The criterion determining the achievement of the network synchronization is set as 20 µs in A_e. As shown in this figure, FTSP has an advantage in the synchronization speed because this scheme has the global reference clock to quickly disseminate over the network. On the other hand, in GTSP, since it takes relatively many rounds to calculate the average of the clocks of neighboring nodes, the convergence speed is low. Even though MKTS adopts a distributed approach, its convergence speed is faster than FTSP.

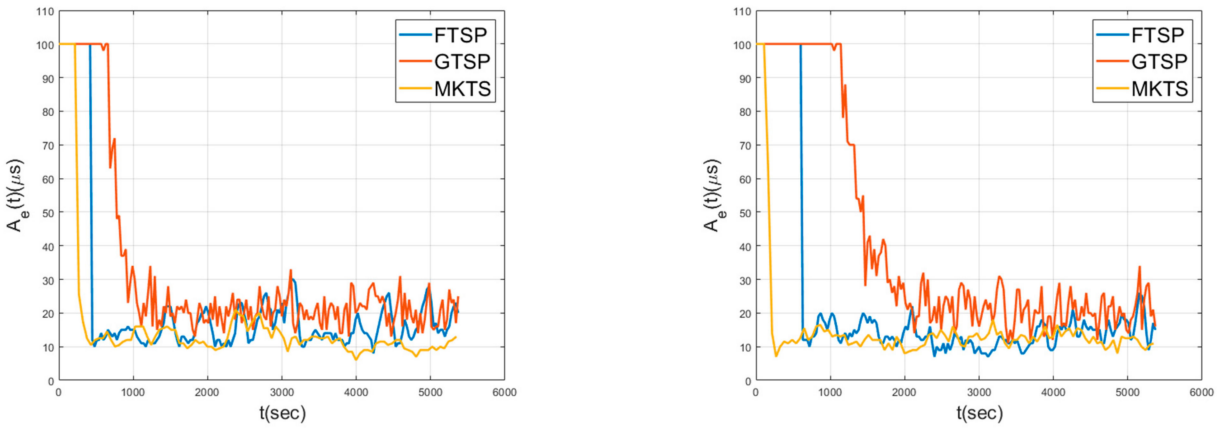

Figure 7. Network error in static environment for mesh (left) and random (right) topologies.

For the mesh topology, the convergence times, i.e., the times taken in achieving $A_e \leq 20$ µs, are in the increasing order of MKTS, FTSP, and GTSP. In addition, the average A_es for these schemes are 12.13 µs, 15.11 µs, and 16.26 µs, respectively. A similar performance is maintained for the random topology. Note that MKTS achieves the best performance both in the convergence time and A_e under the static environment.

Figure 8 shows A_es in a mobile environment, where the nodes move at a speed of 5 m/s in random directions. As shown in this figure, GTSP fails in achieving the synchronization, because the members of a set of neighboring nodes change too quickly for the averaged clock value to converge. Since FTSP synchronizes with the clock of the reference node, even when the network topology is changed, it can achieve synchronization. However, note that the average A_e is 28.7 µs, indicating that the performance decreases by 76% compared to the performance under static environment shown in Figure 7. The average A_e of MKTS is 15.73 µs, which is much better than FTSP. Moreover, the performance decrement of MKTS compared with the static environment is 23%; therefore, it is clearly better than FTSP. In general, under a mobile environment, the decrease in the synchronization performance is inevitable because neighboring nodes may be changed in every round. However, MKTS minimizes this performance degradation by effectively excluding outliers using F-Median and by reducing clock estimation error using Kalman filtering.

3.2. Synchronization Performance with Varying Area Size and Node Speed

Figure 9 compares F-Median with the median-based approach by varying the speed of the nodes, where the median approach selects a median node from among the neighboring nodes without considering the Fast-logical clocks. Both the F-Median and the simple median use Kalman filtering. This figure shows that, regardless of the speed of the nodes, F-Median has a better performance than the median-based method. In the static case, F-Median and the median-based method have the similar A_e deviations (orange line and violet line, respectively); however, F-Median has 18% better A_e than the median-based method. In particular, when the node speed is 5 m/s, F-Median has 49.3% lower deviation and 27.4% lower A_e than the median-based approach.

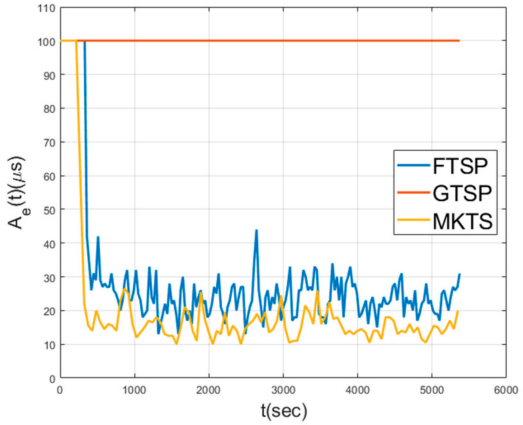

Figure 8. Network error in mobile environment.

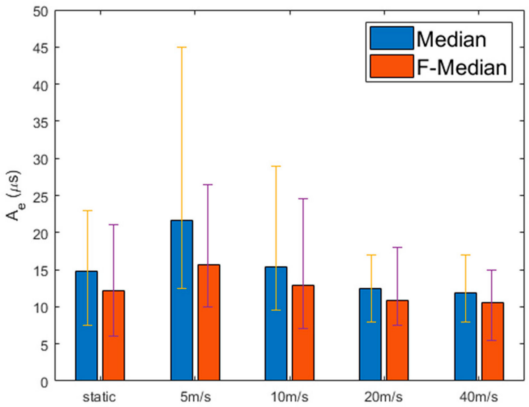

Figure 9. Network error with median and F-median in mobile environment.

Figure 10 shows A_e and N_e of MKTS in a mobile environment. This figure shows that the synchronization performance is maintained even when the speed of the nodes increases. If the size of the area, where the nodes are deployed, is not large, the increasing the speed of the nodes can be beneficial in achieving the synchronization as shown in this figure. However, if the area is very large or has no boundary, as the speed of the nodes increases, the synchronization performance decreases. For instance, in an environment in which there is no boundary of limiting the movement of nodes, the higher the speed of random walking nodes, the more prone the radio link among the nodes is to be broken, resulting in a decrease in the synchronization performance.

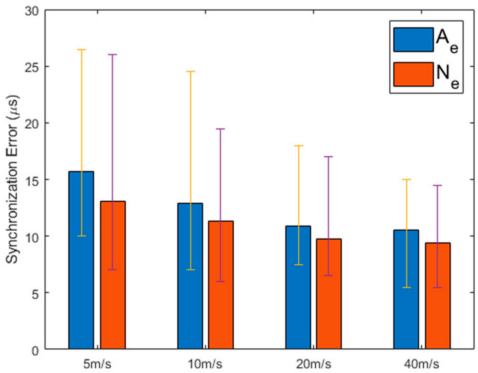

Figure 10. Synchronization error with varying node speed in mobile environment.

A_e and N_e for random walking without boundary is depicted in Figure 11. In this figure, the nodes are static until 600 s achieving time synchronization, after 600 s, the random

walk starts. Figure 11 shows that the slower the speed, the longer the synchronization is maintained. In addition, if the speed of the nodes is high, the nodes within the synchronization boundary quickly leave it, and the synchronization performance decreases rapidly. Figure 12 shows the performance according to the various area size and the node speed. It is inversely proportional to area size and proportional to speed. As the range of the nodes' movement gets smaller, the amount of the exchanged time synchronization messages is maintained at a high level. If the nodes are deployed over a bounded area, as the speed of the nodes increases, the probability of a node getting out of the synchronization group increases.

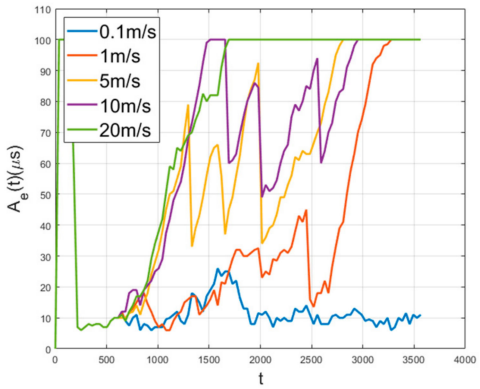

Figure 11. Network error with no boundary.

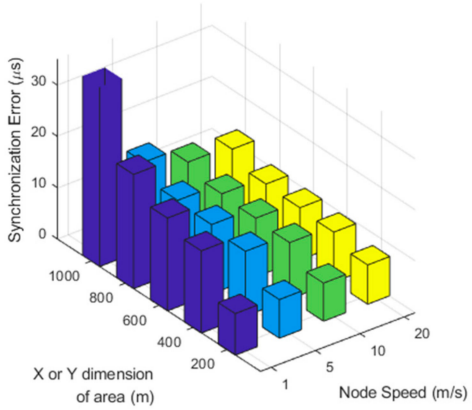

Figure 12. Network error with varying area size and node speed.

However, as shown in Figure 13, if the boundary is small and the speed of the nodes high, the probability of a node entering the synchronization group also increases. On the other hand, if the speed is low, the probability of a node getting out of the synchronization group is decreased, but the probability of a node, which left the synchronization group, re-entering the synchronization group decreases, resulting in the decrease in the synchronization performance of the entire network.

3.3. Performance Analysis with Malfunctioning Nodes

As the number of nodes increases, the number of malfunctioning nodes may increase. If a node exhibits abnormal behavior due to hardware or software failure, the network synchronization performance is greatly reduced. MKTS improves the synchronization performance by excluding malfunctioning nodes from the synchronization process.

Figures 14 and 15 show A_e and N_e when a single node exhibits abnormal behavior. In this simulation, a single node starts to malfunction and to send corrupted time synchronization messages at 900 s. In these figures, the synchronization performance of GTSP deteriorates and fails to achieve time synchronization. FTSP converges but shows the increased fluctuation during the simulation time. On the other hand, MKTS has very a small

fluctuation and successfully maintains the synchronization, even when a malfunctioning node exists.

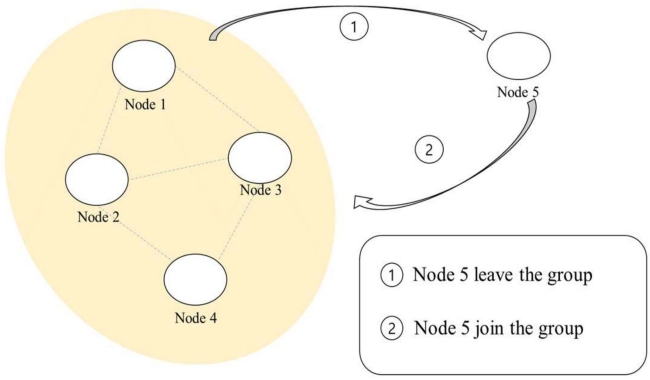

Figure 13. Network boundary.

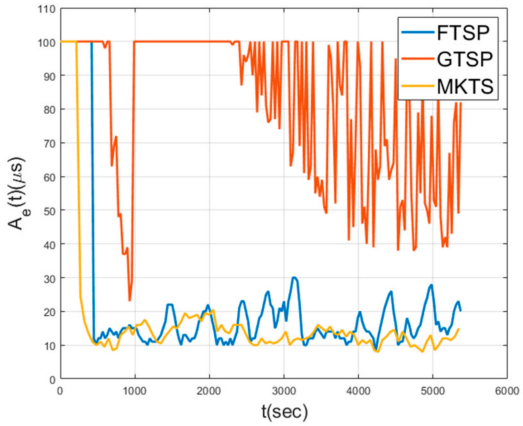

Figure 14. Network error with a single malfunctioning node.

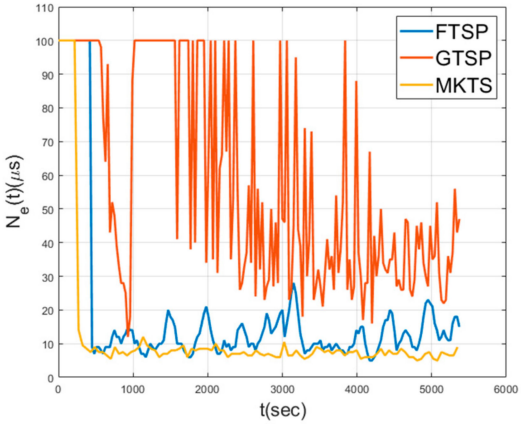

Figure 15. Neighbor error with a single malfunctioning node.

Figure 16 shows the performance of MKTS with varying speed of nodes while two malfunctioning nodes exist. In this simulation, one node fails at 600 s and the other at 1500 s, respectively. As shown in this figure, MKTS successfully achieves time synchronization by excluding these two malfunctioning nodes from the synchronization process, and the speed of the nodes hardly affects the synchronization performance. Figure 17 shows the performance of MKTS with the increasing number of malfunctioning nodes. Performance

decrement is very small up to six malfunctioning nodes, i.e., 12.2% of the total nodes are malfunctioning nodes.

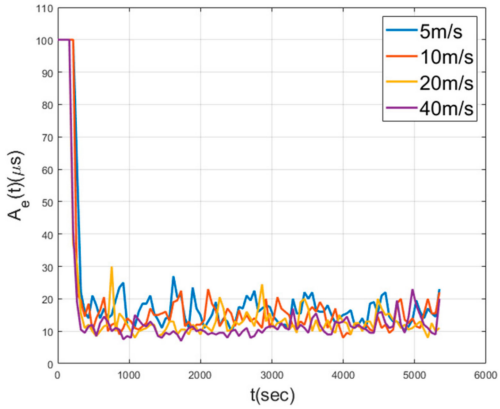

Figure 16. Mobile network error with 2 malfunction nodes.

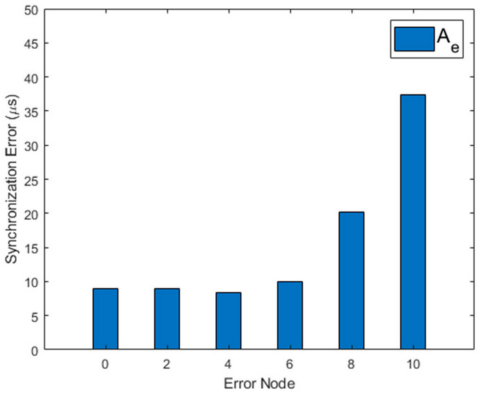

Figure 17. Network error with increasing malfunction nodes.

However, when more than six nodes are malfunctioning, the performance starts to decrease. Even if malfunctioning nodes exist, MKTS recovers the synchronization performance within a short period of time. Moreover, under the harsh environment, in which 12.2% of the nodes are transmitting corrupted time information, MKTS maintains a good time synchronization.

4. Discussion

The results of the performance evaluation clearly show the advantage of MKTS. Compared with conventional time synchronization protocols for both the static and mobile scenarios, the excellent performance is confirmed. In particular, the performance gap is remarkable under the mobile scenarios because the proposed scheme maintains time synchronization under the harsh condition that nodes move with high speed. Moreover, fast convergence times are achieved by efficiently disseminating Fast-median values. Another notable strength of MKTS is its robustness and resilience against environmental changes. Even when 12.2% of malfunctioning nodes transmit corrupted time synchronization messages, MKTS quickly recovers time synchronization. These results support the working hypotheses that Fast-median efficiently excludes outliers and Kalman filtering enhances the synchronization.

A drawback of the proposed scheme is that MKTS has a difficulty in finding the optimal Fast-median when there are a small number of nodes in a network. The purpose of adopting median values is to exclude outliers from samples; however, when the number of samples is small, the probability that outliers fall into the range, which is centered at a median, increases. This decreases synchronization performance of MKTS; however, note

that other time synchronization scheme can also suffer from this problem if a small number of nodes are scattered over a wide area.

For a future research topic extending the proposed scheme, a time-synchronization protocol with an aerial relay node can be considered. The proposed scheme is targeting for a MANET. When a group of nodes are connected to a base station via an aerial relay node, the time synchronization will be an interesting and challenging issue extending the proposed scheme. In this case, the protocol needs to optimize the operation of the fast-moving aerial relay node, and time synchronization should take this feature into account.

5. Conclusions

In this paper, we propose a time synchronization algorithm for mobile environments, which removes outliers using the F-Median of synchronization messages and eliminates synchronization errors using Kalman filtering. In the case of conventional FTSP, the convergence speed of synchronization is fast, but performance decreases as the number of hops increases. In GTSP, the effect of a large hop count is small, but the convergence speed is low. In addition, when nodes in a network act abnormally due to a hardware or software failure, the network synchronization performance is greatly reduced. MKTS shows the fast convergence speed and an accurate synchronization performance. In addition, even if some nodes behave abnormally in a network, these nodes are effectively excluded from the synchronization process to maintain synchronization performance.

Author Contributions: Y.J. and T.K. (Taejoon Kim) conceived and designed the experiments; Y.J. and T.K. (Taehong Kim) performed the network simulation; Y.J., T.K. (Taehong Kim), and T.K. (Taejoon Kim) analyzed the data; T.K. (Taejoon Kim) acquired funding; Y.J. and T.K. (Taejoon Kim) wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2020R1I1A3068305).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

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

References

1. 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] [PubMed]
2. Akhlaq, M.; Sheltami, T.R. The Recursive Time Synchronization Protocol for Wireless Sensor Networks. In Proceedings of the 2012 IEEE Sensors Applications Symposium Proceedings, Brescia, Italy, 7–9 February 2012; pp. 1–6. [CrossRef]
3. Hasan, K.F.; Feng, Y.; Tian, Y. GNSS Time Synchronization in Vehicular Ad-Hoc Networks: Benefits and Feasibility. *IEEE Trans. Intell. Transp. Syst.* **2018**, *19*, 3915–3924. [CrossRef]
4. Pande, H.K.; Srivastava, K.K.; Mangal, L.C. A Resource Allocation Protocol to Meet QoS for Mobile Ad-hoc Network (MANET) in Tactical Scenario. In *Advances in VLSI, Communication, and Signal Processing*; Springer: Singapore, 2020; pp. 71–79.
5. Pliatsios, D.; Sarigiannidis, P.; Goudos, S.K.; Psannis, K. 3D Placement of Drone-Mounted Remote Radio Head for Minimum Transmission Power under Connectivity Constraints. *IEEE Access* **2020**, *8*, 200338–200350. [CrossRef]
6. Zhang, Y.; Qiu, T.; Liu, X.; Sun, Y.; Zhao, A.; Xia, F. Mac-Time-Stamping-based High-accuracy Time Synchronization for Wireless Sensor Networks. In Proceedings of the 2016 International Conference on Software Networking (ICSN), Jeju, Korea, 23–26 May 2016; pp. 1–4. [CrossRef]
7. Maroti, M. The Flooding Time Synchronization Protocol. In Proceedings of the 2nd ACM Conf. Embedded Networked Sensor Systems, Baltimore, MD, USA, 3–5 November 2004.
8. Lenzen, C.; Sommer, P.; Wattenhofer, R. PulseSync: An Efficient and Scalable Clock Synchronization Protocol. *IEEE/ACM Trans. Netw.* **2015**, *23*, 717–727. [CrossRef]
9. Lenzen, C.; Sommer, P.; Wattenhofer, R. Optimal clock synchronization in networks. In Proceedings of the 7th International Conference on Embedded Networked Sensor Systems, SenSys 2009, Berkeley, CA, USA, 4–6 November 2009.

10. Saïah, A.; Benzaïd, C.; Badache, N. CMTS: Consensus-based Multi-hop Time Synchronization protocol in wireless sensor networks. In Proceedings of the 2016 IEEE 15th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 31 October–2 November 2016; pp. 232–236. [CrossRef]
11. Al-Kofahi, O. Evaluating time synchronization using application-layer time-stamping. In Proceedings of the 2016 IEEE Wireless Communications and Networking Conference, Doha, Qatar, 3–6 April 2016; pp. 1–6. [CrossRef]
12. Ren, W.; Zhao, Q.; Swami, A. On the Connectivity and Multihop Delay of Ad Hoc Cognitive Radio Networks. *IEEE J. Sel. Areas Commun.* **2011**, *29*, 805–818. [CrossRef]
13. Yang, B.; Wu, Z.; Shen, Y.; Fan, Y. Multicast Delivery Delay in General Two-Hop Relay MANETs. In Proceedings of the 2017 International Conference on Networking and Network Applications (NaNA), Kathmandu, Nepal, 16–19 October 2017; pp. 100–103. [CrossRef]
14. Su, X.; Hui, B.; Chang, K. Multi-hop clock synchronization based on robust reference node selection for ship ad-hoc network. *J. Commun. Netw.* **2016**, *18*, 65–74. [CrossRef]
15. Liu, C.; Zhang, G.; Guo, W.; He, R. Kalman Prediction-Based Neighbor Discovery and Its Effect on Routing Protocol in Vehicular Ad Hoc Networks. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 159–169. [CrossRef]
16. Sommer, P.; Wattenhofer, R. Gradient clock synchronization in wireless sensor networks. In Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, San Francisco, CA, USA, 13–16 April 2009; pp. 37–48.
17. Sun, W.; Gholami, M.R.; Ström, E.G.; Brännström, F. Distributed clock synchronization with application of D2D communication without infrastructure. In Proceedings of the 2013 IEEE Globecom Workshops (GC Wkshps), Atlanta, GA, USA, 9–13 December 2013; pp. 561–566. [CrossRef]
18. Sun, W.; Ström, E.G.; Brännström, F.; Gholami, M.R. Random Broadcast Based Distributed Consensus Clock Synchronization for Mobile Networks. *IEEE Trans. Wirel. Commun.* **2015**, *14*, 3378–3389. [CrossRef]
19. Phan, L.-A.; Kim, T.; Kim, T.; Lee, J.; Ham, J.-H. Performance Analysis of Time Synchronization Protocols in Wireless Sensor Networks. *Sensors* **2019**, *19*, 3020. [CrossRef] [PubMed]
20. 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]
21. Sun, J.; Liao, H.; Upadhyaya, B.R. A robust functional-data-analysis method for data recovery in multichannel sensor systems. *IEEE Trans. Cybern.* **2014**, *44*, 1420–1431. [CrossRef] [PubMed]
22. Singhal, D.; Garimella, R.M. Simple Median based information fusion in wireless sensor network. In Proceedings of the 2012 International Conference on Computer Communication and Informatics, Coimbatore, India, 10–12 January 2012; pp. 1–7. [CrossRef]
23. Sahin, S.; Cipriano, A.M.; Poulliat, C.; Boucheret, M. On Cooperative Broadcast in MANETs with Imperfect Clock Synchronization. In Proceedings of the MILCOM 2018–2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 29–31 October 2018; pp. 1–7.
24. Bellavista, P.; Giannelli, C.; Lagkas, T.; Sarigiannidis, P. Multi-domain SDN controller federation in hybrid FiWi-MANET networks. *EURASIP J. Wirel. Commun. Netw.* **2018**, *2018*, 103. [CrossRef]
25. Liu, Y.; Fan, X.; Chen, L.; Wu, J.; Li, L.; Ding, D. An Innovative information fusion method with adaptive Kalman filter for integrated INS/GPS navigation of autonomous vehicles. *Mech. Syst. Signal Process.* **2018**, *100*, 605–616. [CrossRef]
26. Welch, G.; Bishop, G. An introduction to the Kalman Filter. In Proceedings of the Annual Conference on Computer Graphics & Interactive Techniques (SIGGRAPH '01), Los Angeles, CA, USA, 12–17 August 2001.
27. Wu, Z.; Li, J.; Zuo, J.; Li, S. Path planning of UAVs based on collision probability and Kalman filter. *IEEE Access* **2018**, *6*, 34237–34245. [CrossRef]
28. Song, W.; Wang, J.; Zhao, S.; Shan, J. Event-trigged cooperative unscented Kalman filtering and its application in multi-UAV systems. *Automatica* **2019**, *105*, 264–273. [CrossRef]
29. Jondhale, S.R.; Deshpande, R.S. Kalman Filtering Framework-Based Real Time Target Tracking in Wireless Sensor Networks Using Generalized Regression Neural Networks. *IEEE Sens. J.* **2019**, *19*, 224–233.
30. Sarvghadi, M.A.; Wan, T.-C. Message Passing Based Time Synchronization in Wireless Sensor Networks: A Survey. *Int. J. Distrib. Sens. Netw.* **2016**, *12*, 1280904. [CrossRef]
31. Djenouri, D.; Bagaa, M. Synchronization Protocols and Implementation Issues in Wireless Sensor Networks: A Review. *IEEE Syst. J.* **2016**, *10*, 617–627. [CrossRef]
32. Phan, L.; Kim, T.; Kim, T.; Lee, J.; Ham, J. Poster Abstract: A Fast Consensus-based Time Synchronization Protocol with Virtual Links in WSNs. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April–2 May 2019; pp. 1019–1020. [CrossRef]
33. Ramanathan, R. Challenges: A radically new architecture for next generation mobile ad hoc networks. In Proceedings of the 11th Annual International Conference on Mobile Computing and Networking, MobiCom '05, Cologne, Germany, 28 August–2 September 2005; pp. 132–139.
34. Sun, W.; Brännström, F.; Ström, E.G. Network Synchronization for Mobile Device-to-Device Systems. *IEEE Trans. Commun.* **2017**, *65*, 1193–1206. [CrossRef]

35. Mo, Y.; Liu, Z.; Zheng, L.; Deng, X. Kalman-consensus filter for time synchronization in wireless sensor networks. In Proceedings of the IET International Conference on Information and Communications Technologies (IETICT 2013), Beijing, China, 27–29 April 2013; pp. 421–428. [CrossRef]
36. Oliveira-junior, E.M.; Souza, M.L.; Kuga, H.K.; Lopes, R.V. Clock synchronization via Kalman filtering. In Proceedings of the Brazilian conference on dynamics, control and applications, Bauru, Brazil, 18–22 May 2009.
37. OPNET, Retrieved. Available online: https://www.riverbed.com/sg/index.html (accessed on 28 May 2019).