An enhanced time synchronization protocol in automated surface vehicles

Yeong Chin Koo* & Muhammad Nasiruddin Mahyuddin*

School of Electrical and Electronic Engineering,
Engineering Campus, Universiti Sains Malaysia (USM), 14300 Nibong Tebal, Pulau Pinang, Malaysia.

[E-mail: nasiruddin@usm.my]

Mobility of an autonomous surface vehicle (ASV) has caused the wireless sensor and actuator networks (WSANs) onboard to be mobile as well. However, the time synchronization of a WSAN on ASV is more challenging due to the environmental harshness and node mobility. In this paper, an enhanced control-theoretic distributed time synchronization protocol called Time Synchronization using Distributed Observer algorithm with Sliding mode control element (TSDOS) is presented to solve the time synchronization issue in ASV. This TSDOS protocol feeds a sliding mode control element on the relative comparative error to estimate the skew rate and relative skew rate. Through the theoretical analysis and simulations, TSDOS has showed the advantages of totally distributed, robust to node failure and time-varying clock frequencies, which are the situations usually faced by an ASV. In addition, TSDOS also has better or comparable performance over existing protocols in terms of rate of convergence, consensus error spike, and steady-state error.

[Keywords: Autonomous surface vehicle; Distributed; Observer; Sliding mode control; Time synchronization; Wireless sensor and actuator network.]

Introduction

The autonomous surface vehicle (ASV) is receiving considerable attention from the experts due to its ability to conduct large-scale yet low-cost marine scientific research and engineering task in water for a long time. These advantages of ASV have made it suitable to be practiced in a wide range of applications, including but not limited to, marine resource exploration and exploitation, underwater topography survey, marine environmental monitoring and surveillance, and various military applications.

For an ASV to function well, the wireless sensor and actuator network (WSAN) plays an important role. Through the sensors and actuator onboard, a surface vehicle can sense its surrounding environment and carry its assigned task correctly and autonomously. In addition, to ensure the WSAN onboard can communicate with each other in a meaningful fashion, all the sensors and actuators need to have a common time reference, or in other words, their time need to be synchronized. With the synchronized time reference, the events perceived by the sensor nodes can be sequenced correctly, the physical surrounding environment can be represented precisely and only then the tasks assigned can be carried out accurately. In addition, a common time notion among the sensors is also vital to identify the casual relationships between events in the physical world, to eliminate iterating sensor data, and to facilitate sensor and actuator network operation generally.

To illustrate the importance of a time synchronization protocol in an ASV, let us consider the basic issues for an autonomous vehicle navigation: Path planning and obstacle avoidance. To solve these basic issues, the ASV can only depend on the onboard sensor information. The sensor nodes will first record the time when they detected any obstacle. This timing information will then be transmitted to a base station, where all the information from different nodes will be combined and processed to estimate the location of the obstacle and plan the path to avoid it. If there is no common time reference, the estimated location of the obstacle will be inaccurate and the ASV might clash with the obstacle.

Energy is limited in WSAN, thus energy conservation is important. Sleep scheduling is normally applied in WSAN, where the nodes will be put in a power saving sleep state whenever they are not in use and wake up periodically for a short time to carry out their assigned tasks, either sensing the environment or transmitting data. However, the low-cost oscillators in the sensor nodes have caused the
clocks of different nodes keep drifting away from each other, even if they started at the same initial time. Therefore, time synchronization is one of the challenging and crucial issues to be addressed in WSAN.

In literature, many time synchronization protocols have been introduced for WSAN. Reference Broadcast Synchronization (RBS)\textsuperscript{8} is a typical receiver-receiver method, while Time-sync Protocol for Sensor Network (TPSN)\textsuperscript{9} is a typical sender-receiver method. Many issues in time synchronization in WSAN had been addressed in the literature such as the drifting of local nodes clock\textsuperscript{10}, variations in observed time intervals and complexity involved when nodes are connected over a single-hop/multi-hop manner\textsuperscript{11}. However, most of the protocols cannot avoid the challenges of suffering from unpredictable packet losses and dynamic environment conditions in terms of communication link and the number of nodes\textsuperscript{10,12}. Besides, ad-hoc deployment in WSAN is common, where nodes will randomly join and drop from the network. This also exerts impact on the network during coordination and synchronization.

In general, time synchronization protocols in the literature can be roughly categorized into two categories: \textit{Centralized synchronization} and \textit{distributed synchronization}\textsuperscript{13,14}.

\textit{Centralized synchronization} protocols, for example, RBS\textsuperscript{8}, TPSN\textsuperscript{9} and Flooding Time Synchronization Protocol (FTSP)\textsuperscript{15} usually have fast convergence speed and small synchronization error. On the other hand, the major drawback of these protocols is that these protocols require a physical node to act as the network’s reference clock; if the reference node is malfunctioned, the protocols will face a serious damage or even fail to synchronize. As a result, centralized synchronization protocols are normally designed with complexity logic to deal with the WSAN’s dynamic topology. Another disadvantage of these protocols is the synchronization errors will grow proportionally with the increase of network hops\textsuperscript{16}.

\textit{Distributed synchronization} protocols such as Time Diffusion Synchronization Protocol (TDP)\textsuperscript{17}, Global Clock Synchronization (GCS)\textsuperscript{11}, Reachback Firefly Algorithm (RFA)\textsuperscript{12}, Average TimeSync (ATS)\textsuperscript{18,19}, Consensus Clock Synchronization (CCS)\textsuperscript{20} and Time Synchronization using Maximum and Average consensus protocol (TSMA)\textsuperscript{16} use the local information to achieve the whole network synchronization. A fully distributed network topology refers to the network that has no special nodes as roots or gateways, and all nodes run the same algorithm. Unlike centralized synchronization protocols, distributed synchronization protocols are very robust to node failure and new node appearance, but they require different synchronization algorithms from centralized synchronization, as there is no reference node. Moreover, these protocols can easily adapt to WSAN’s dynamic topology property with light computation. Anyhow, distributed synchronization protocols also have disadvantage in that the convergence speeds may be a bit slow since each local clock exploits only the local information including the neighboring nodes. All the nodes are oblivious to the global information about the connected network.

Most of the protocols in the literature, especially those based on least-squares regression have a drawback where the effect of various error sources will appear as multiplicative noise in the time synchronization error dynamic. The multiplicative noise will cause the global synchronization error to proximately grow exponentially with the diameter of the network, which in turn will result in poor performance in terms of scaling properties. This issue can be solved by using control theory based approach, where the synchronization is achieved by using linear feedback on the measured local synchronization error. Through this approach, the error sources will appear as additive noise in the error dynamic and the global synchronization error will proximately grow as the square root of the diameter of the network\textsuperscript{21}.

In addition, since ASV is moving most of the time, it requires the WSAN onboard to be a movable WSAN. The node mobility in a movable WSAN has raised the difficulties of time synchronization as some nodes sometimes might move out of communication range. This will cause the nodes to be treated as malfunctioned nodes before the nodes return to the communication range\textsuperscript{22}.

The time synchronization protocol presented in this paper is a distributed synchronization protocol using control-theoretic approach. It is called Time Synchronization using Distributed Observer algorithm with Sliding mode control element (TSDOS), which applies a distributed observer/estimator algorithm and sliding mode control element on relative comparative error to estimate the skew rate and relative skew rate to achieve the objective of time synchronization. With
this time synchronization protocol, the ASV does not need to be arranged in a centralized formation and thus the limitation of centralized formation can be avoided. In comparison with another two algorithms from literature, TSDOS has several positive contributions: Low energy consumption, finite-time convergence of time synchronization error, ability to reach consensus even if a node is malfunctioned for a certain period, and time-varying clock frequencies.

This paper describes the development of TSDOS algorithm and the convergence analysis of the TSDOS algorithm, followed by the details of the simulation set-up and discussion on the simulation results displayed. The simulation results of TSDOS are compared with two time synchronization protocols from literature: ATS\textsuperscript{18,19}, and PISync\textsuperscript{21}.

**TSDOS Methodology**

Clock modeling

Every node \( i \) in a wireless sensor and actuator network has its own local clock which is a monotonically non-decreasing function of \( t \) and the value is collected from the quartz crystal in the clock. Nonetheless, due to several factors such as different drift rates and quality of each clocks' quartz crystals, thermal, vibration and moisture environment, not only the nodes' clock values will drift away from each other, the drift rates will also differ with time proceed\textsuperscript{22,24}. Therefore, all the nodes in the WSAN require periodical adjustment so that all the nodes are synchronized to follow a same time reference.

Each local clock with first order dynamics can be modeled as in Eq. (1)

\[
\tau_i(t) = \alpha_i t + \beta_i
\]

where \( \tau_i \) is the local clock reading, \( \alpha_i \) is the local clock skew which determines the clock speed, and \( \beta_i \) is the local clock offset which is the difference between time reported by clock \( \tau_i \) and the real time at the time origin, i.e., \( t_0 = 0 \). In an ideal scenario, the local clock skew, \( \alpha_i \), is equal to 1 and the local clock offset, \( \beta_i \), is equal to 0.

Figure 1 illustrates the clock dynamics of a pair of unsynchronized nodes, differentiated by their skew rate and offset\textsuperscript{19}. Through Figure 1, it can be observed that \( \alpha_i \) is causing the continuous bias while \( \beta_i \) is the cause of the initial error.

To reach the objective of perfect time synchronization among all the nodes in the network, all nodes \( i = 1,2, \ldots, N \) must precisely compensate for their clock parameters \( \alpha_i \) and \( \beta_i \) so that all clocks have same clock skew and zero offset error\textsuperscript{10}. However, without any information on the real-world absolute time \( t \) available to the nodes in WSAN, computing the parameters \( \alpha_i \) and \( \beta_i \) becomes impossible.

To solve this issue, first Eq. (1) need to be rearranged so that \( t \) is expressed as a shared entity\textsuperscript{17}:

\[
t = \frac{\tau_i - \beta_i}{\alpha_i}
\]

(2)

Next, substitute Eq. (2) into Eq. (1) for node \( j \):

\[
\tau_j = \frac{\alpha_j}{\alpha_i} \tau_i - \frac{\alpha_j}{\alpha_i} \beta_i + \beta_j
\]

(3)

As a result, Eq. (3) does not contain the absolute reference time \( t \). As suggested by Schenato in both\textsuperscript{18,19}, the clock skew rate and offset of all the nodes in the network can eventually be obtained by estimating the relative magnitude \( \frac{\alpha_j}{\alpha_i} \) and the offset imposed \( \left( \beta_j - \frac{\alpha_j}{\alpha_i} \beta_i \right) \). If these magnitudes can be estimated, then the clock dynamics estimates \( \hat{\tau}_i \) can be obtained indirectly. Thus, the clock dynamics of each node can be expressed in terms of these estimated magnitudes,

\[
\hat{\tau}_i = \hat{\alpha}_i \tau_i + \hat{\beta}_i
\]

(4)

where \( \hat{\alpha}_i \) is the relative clock skew rate estimate and \( \hat{\beta}_i \) is the clock offset estimate. Eq. (4) also shows that

![Fig. 1 — The clock dynamics of node i and j compared in terms of their respective local clock skew rate and clock offset.](image-url)
each node has its own self-time keeping mechanism by estimating its own virtual time \( \hat{t}_i \), before the estimates are passed to its connected neighbor node \( j \). The final objective is to have a virtual reference clock \( \tau_r \).

\[
\tau_r(t) = \alpha_r t + \alpha_r \tag{5}
\]

which all the nodes will eventually refer to and be synchronized, i.e.

\[
\lim_{t \to \infty} \hat{t}_i = \tau_r, \forall i \tag{6}
\]

The objective and motivation of the work presented in this paper are similar to the TSMA algorithm\(^{16} \) and ATS algorithm\(^{18,19} \). Eq. (6) describes the consensus needed to be reached to have the common time reference. The difference is that TSMA algorithm adopts the maximum averaging of all the virtual time estimates and ATS algorithm adopts a kind of low pass filter structure; while in this work, the ATS algorithm is enhanced by introducing a sliding-mode like term to have the advantage of fast convergence in both relative skew estimates and skew rate estimates.

**TSDOS algorithm**

Two assumptions are made for the algorithm to work:

1. The communication between neighboring nodes is a single-hop communication which means they occur at a very short finite time interval.
2. The communication between a node pair is instantaneous.

In this work, from first assumption, each node \( i \) is assumed to periodically transmit a packet to all its neighbors with a synchronization period equal to \( T \), i.e., the transmission instant \( t^i \)

\[
t^i = \frac{\ell T - \beta_i}{\alpha_i} \tag{7}
\]

TSDOS consists of three main parts: Relative skew rate estimation, skew rate estimation, and offset estimation.

**Relative skew estimation:** Every node \( i \) seeks to estimate the relative skew rate with respect to their neighbor \( j \). The relative skew rate estimates in ATS\(^{18,19} \), are enhanced by incorporating a sliding-mode like term at the relative skew rate. We first define a sliding plane \( \epsilon \),

\[
\epsilon_i = \frac{\tau_i(t_2) - \tau_i(t_1)}{\tau_j(t_2) - \tau_j(t_1)} \tag{8}
\]

from which i.e. \( \tau_{ij}(t_2) - \tau_{ij}(t_1) \) taken at time \( t_1 \) and time \( t_2 \) from the data packet for node \( j \) is compared with node \( i \). Then we feed the relative comparative error into a low-pass filter structure to estimate the relative skew rate,

\[
\dot{x}_i(t + 1) = \rho x \dot{x}_i(t) + \left(1 - \rho x \right) \text{sign}(\epsilon_i) \tag{9}
\]

where \( \rho x \in \{0,1\} \) is the tuning parameter similar to the effects of the filter poles coefficient, i.e. increasing the coefficient may increase the convergence of the filtered signal but may reduce the effects of filtering. A signum function forces the comparative clock pairs to the plane defined in Eq. (8).

**Theorem 1.** Consider the relative skew update equation (9) where \( 0 < \rho \leq 1 \), the transmission events \( t^i \) are generated according to the Eq. (7), and each \( t_i \) evolves according to Eq. (1). Then

\[
\lim_{t \to \infty} x_i(t) = 1
\]

in finite-time for any initial condition \( x_i(0) \).

Proof. By writing the Eq. (9) in recursive formula, it follows that

\[
\dot{x}_i(t) = \rho \dot{x}_i(0) + \sum_{h=0}^{\ell} \rho^{h-1}\left(1 - \rho \right) \text{sign}(\epsilon_i)
\]

\[
= \rho \dot{x}_i(0) + \left(1 - \rho \right) \text{sign}(\epsilon_i)
\]

where \( \ell = \frac{t^i - \beta_i}{T} \), and \( \sum_{h=0}^{\ell} \rho^h = \frac{1 - \rho^{\ell+1}}{1 - \rho} \) (from finite geometric series). Since \( 0 < \rho \leq 1 \), then

\[
\lim_{t \to \infty} x_i(t) = \lim_{t \to \infty} \rho \dot{x}_i(0) + \left(1 - \rho \right) \text{sign}(\epsilon_i)
\]

\[
= \text{sign}(\epsilon_i)
\]

Given the properties of signum function are

\[
\text{sign}(A) = \begin{cases} 1, & A > 0 \\ 0, & A = 0 \\ -1, & A < 0 \end{cases}
\]

Since \( \epsilon_i = \frac{\tau_{ij}(t_2) - \tau_{ij}(t_1)}{\tau_{ij}(t_2) - \tau_{ij}(t_1)} \) and \( t \) is a monotone increasing function, so \( \epsilon_i > 0 \). Therefore,

\[
\lim_{t \to \infty} x_i(t) = 1
\]

and the convergence is in finite-time.

**Skew estimation:** As discussed in Eq. (4), node \( i \) updates its own perceived virtual time. The sliding mode term is also used to enhance the skew rate estimate in the filter,

\[
\dot{\alpha}_i(t + 1) = \rho x \dot{\alpha}_i(t) + \left(1 - \rho x \right) \text{sign}(\alpha_j) \tag{10}
\]
where $\rho_s \in \{0,1\}$ is another tuning parameter which has the same effect with $\rho_x$ and $\mathring{a}_j$ is the virtual clock skew estimates of the connected neighbor node $j$.

**Theorem 2.** Consider the skew update equation given by Eq. (10) with initial condition $\mathring{a}_i(0) > 0$ and $0 < \rho_s < 1$, where $\mathring{\xi}_i(t)$ is updated according to Eq. (9) and $t_i^+$ are defined in Eq. (7). Then

$$\alpha_r = \lim_{t \to +\infty} \mathring{a}_i(t) = 1, \forall i, \alpha_i(0) > 0$$

in finite-time.

**Proof.** The proof follows along the same lines of Theorem 1.

$$\mathring{a}_i(t) = \rho_x^i \mathring{a}_i(0) + \sum_{h=0}^{\ell-1} \rho_x^h (1 - \rho_x) \mathring{\xi}_i(t) \text{sign}(\mathring{a}_j)$$

From Theorem 1, $\lim_{t \to +\infty} \mathring{\xi}_i(t) = 1$,

$$\alpha_r = \lim_{t \to +\infty} \mathring{a}_i(t)$$

$$= \lim_{t \to +\infty} \rho_x^i \mathring{a}_i(0) + (1 - \rho_x) \text{sign}(\mathring{a}_j)$$

$$= \text{sign}(\alpha_j)$$

$$= 1$$

and the convergence is in finite-time.

**Offset estimation:** Lastly, upon achieving all the nodes run at the same speed, we may estimate the offset, $\mathring{\delta}_i$, of the virtual clock by,

$$\mathring{\delta}_i(t + 1) = \mathring{\delta}_i(t) + (1 - \rho_o)(\mathring{\tau}_j - \mathring{\tau}_i)$$

**Preposition (Theorem 6) [19].** Consider the offset update equation given by Eq. (11) with initial condition $\mathring{\delta}_i(0) = 0$ and $0 < \rho_o < 1$, where $\mathring{\tau}_i$, and $t_i^+$, are defined in Eq. (4), and Eq. (7), respectively. Then

$$\lim_{t \to +\infty} \mathring{\tau}_i(t) = \mathring{\tau}_j(t), \forall i,j \in N_j$$

exponentially fast.

Finally, the virtual clock estimator $\mathring{\tau}$ for each node $i$ is then formed by combining the skew estimator and offset estimator value. Algorithm 1 summarizes the protocol.

**Algorithm 1:** Pseudocode overview for proposed synchronization protocol

**Initialization:** $\mathring{a}_i \leftarrow 1; \mathring{\delta}_i \leftarrow 0$

**Step 1: Sliding Plane,** $\epsilon_i$, is defined using formula

$$\epsilon_i = \frac{\mathring{\tau}_j(t_2) - \mathring{\tau}_j(t_1)}{\mathring{\tau}_i(t_2) - \mathring{\tau}_i(t_1)}$$

**Step 2: Relative Skew Estimate,** $\mathring{\xi}_i$, is obtained using formula:

$$\mathring{\xi}_i(t + 1) = \rho_x \mathring{\xi}_i(t) + (1 - \rho_x) \text{sign}(\epsilon_i)$$

**Step 3: Skew Estimate,** $\mathring{a}_i$, is measured with

$$\mathring{a}_i(t + 1) = \rho_x \mathring{a}_i(t) + (1 - \rho_x) \mathring{\xi}_i \text{sign}(\mathring{a}_j)$$

**Step 4: Offset Estimate,** $\mathring{\delta}_i$, is calculated using

$$\mathring{\delta}_i(t + 1) = \mathring{\delta}_i(t) + (1 - \rho_o)(\mathring{\tau}_j - \mathring{\tau}_i)$$

**Output:** Virtual Clock Estimate, $\mathring{\tau}_i$, is obtained:

$$\mathring{\tau}_i = \mathring{\tau}_i \mathring{\tau}_i + \mathring{\delta}_i$$

**Simulation Set-up**

In this work, we designed several scenarios of experimental simulations to prove the capabilities of the proposed time synchronization protocol in three scenarios and compare it with another two protocols from literature, ATS [18,19], and PISync [21]. The experimental simulations are simulated using Matlab Simulink. The simulations have been carried to illustrate ad-hoc deployment involving five nodes in WSAN, with the ability to expand to more than five nodes. Every node in the network has its own role, including coordinator, router and end device, and is implemented with the same algorithm. Although this work only involved five nodes, it is enough for the three roles to be deployed, where there are one coordinator, two routers and two end devices. We have divided our simulated experiments into three cases. Table 1 summarizes the conditions of each scenario.

In all the three cases, we observed the ability of the network to be synchronized by the algorithms and the magnitude of the consensus error after the nodes reached consensus. In addition, cumulative Integral Absolute Error or cIAE which can be expressed as

$$cIAE = \sum_{i=1}^{N} \int_{0}^{T} |e_i| dt$$

was computed for all the nodes in the network as a measure of the computational effort (control effort) of the protocols. $\alpha$ denotes the total number of nodes in a network whose perceived virtual time is to be

Table 1 — Conditions for experimental simulation scenarios

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE 1</td>
<td>5 nodes arranged in 1 network (Figure 2)</td>
</tr>
<tr>
<td>CASE 2</td>
<td>Node 3 turned off for a specified period.</td>
</tr>
<tr>
<td>CASE 3</td>
<td>The skew rates of each nodes are varied in different periods</td>
</tr>
</tbody>
</table>
synchronized. cIAE also gives an indication of the energy consumption of each node to achieve time synchronization, thus the lower value is desirable. This is one of the important traits for a wireless sensor node as it needs to conserve energy. The synchronization period in this simulation is set to be 1 milliseconds.

**CASE 1**

The main objective of this scenario is to prove the ability of TSDOS to synchronize the time in a network. Figure 2 reveals the connectivity among the five nodes in the network under the scenario CASE 1. The five nodes are arranged into two clusters (Cluster 1=Node 1 and Node 2, Cluster 2=Node 1, Node 3, Node 5 and Node 4). Besides, in Cluster 2, the timing information is passed from Node 1 to Node 4 in multi-hop fashion. Furthermore, through the arrangement of nodes as in Figure 2, the effectiveness of the protocol to synchronize the time between two clusters of nodes and effect of multi-hop can also be tested. The total simulation time in this case is 100 s.

**CASE 2**

In the scenario of CASE 2, we try to simulate the scenario where the node moves out of communication range for a period due to the node mobility. To do that, Node 3 will be turned off for a specific period of duration, as tabulated in Table 2. For example, in CASE 2-01, Node 3 will be “off” for 1, which is starting from 1st second and it will be “on” till the 2nd second. This scenario seeks to investigate the performance of the three compared protocols under the situation when one of the nodes in the network is stopped and does not respond for a certain period of duration. The total simulation time in this case is 20 s.

**CASE 3**

As stated by Yıldırım11, a more realistic scenario will involve time-varying clock frequencies. Therefore, in CASE 3, the value of each node’s skew rate, $\alpha_i$, will be changed after a specified time frame to indicate the time-varying properties. Table 3 lists out the changes of each node’s skew rate with the respective time frame. The total simulation time in this case is 100 s.

**Results**

**CASE 1**

Figure 3 plots the skew rate estimate, $\hat{\alpha}_i$, of all the five nodes in the network in TSDOS. From Figure 3, it is clear that all the $\hat{\alpha}_i$ have the same constant value, 1, which is the same as with the claim in Theorem 2.

Figures 4 and 5 illustrate the result plots of the three algorithms under scenario CASE 1, where all five nodes in the network are required to synchronize with each other. It can be observed that all the three algorithms are able to synchronize the nodes; Figure 4 shows that all the nodes are converged to a same

---

**Table 3 — Value of parameters $\alpha_i$ used in CASE 3**

<table>
<thead>
<tr>
<th>Time (seconds)</th>
<th>0-20</th>
<th>20-40</th>
<th>40-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 1</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Node 2</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Node 3</td>
<td>3</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>Node 4</td>
<td>4</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Node 5</td>
<td>5</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

---

**Table 2 — The ‘off’ duration in each case**

<table>
<thead>
<tr>
<th>CASE</th>
<th>NODE</th>
<th>Duration (s)</th>
<th>Off (s)</th>
<th>On (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-01</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2-02</td>
<td>3</td>
<td>0.01</td>
<td>2</td>
<td>2.01</td>
</tr>
<tr>
<td>2-03</td>
<td>3</td>
<td>Off</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>2-04</td>
<td>3</td>
<td>0.001</td>
<td>3</td>
<td>3.001</td>
</tr>
</tbody>
</table>

---

![Fig. 2 — Connectivity of five Nodes](image)

![Fig. 3 — Skew rate estimate, $\hat{\alpha}_i$, of all the five nodes in TSDOS.](image)
Fig. 4 — Plot of virtual time estimate, $\hat{t}_v$, of TSDOS, ATS, and PISync for CASE 1

Fig. 5 — Plot of virtual time estimation error, $\hat{e}_v$, of TSDOS, ATS, and PISync for CASE 1
positively increasing linear line and Figure 5 proves that all the three algorithms are stable and have zero error except PISync has error in the range of -6 to 12. The range of error of PISync causes the PISync to have the largest cIAE, which can be observed in Figure 6, which shows the cIAE of the three algorithms to have a clearer comparison on the three algorithms. Figure 6 also shows that ATS and TSDOS outperformed PISync, where ATS and TSDOS have a much lower cIAE as compared to PISync. In addition, TSDOS has the best performance as it has the lowest value in terms of cIAE, compared to the other two protocols. As aforementioned, cIAE can also be treated as a measure on the energy consumption of the algorithms, thus TSDOS can be said to have the advantage of consuming the least energy among the three algorithms in CASE 1 scenario.

CASE 2

In CASE 2, Node 3 of the network is turned off for a fixed period before it is turned on again to simulate the occasion where the node might move out of communication range and back in contact due to its mobility. The Figures 7, 9, 11, and 13 display the plots of virtual time estimate of TSDOS, ATS, and PISync for CASE 2-01, CASE 2-02, CASE 2-03, and CASE 2-04, respectively; while Figures 8, 10, 12, and 14 display the evolutions of virtual time estimation error of TSDOS, ATS, and PISync for CASE 2-01, CASE 2-02, CASE 2-03, and CASE 2-04, respectively. From the error plots (Figs. 8, 10, 12, and 14), it is observable that the estimation error of ATS
Fig. 8 — Plot of virtual time estimation error, $\tilde{\tau}_1$, of TSDOS, ATS, and PISync for CASE 2-01

Fig. 9 — Plot of virtual time estimate, $\hat{\tau}_1$, of TSDOS, ATS, and PISync for CASE 2-02
Fig. 10 — Plot of virtual time estimation error, $\tilde{\tau}_i$, of TSDOS, ATS, and PISync for CASE 2-02

Fig. 11 — Plot of virtual time estimate, $\hat{\tau}_i$, of TSDOS, ATS, and PISync for CASE 2-03
Fig. 12 — Plot of virtual time estimation error, $\tilde{\tau}_i$, of TSDOS, ATS, and PISync for CASE 2-03

Fig. 13 — Plot of Virtual Time Estimate, $\hat{\tau}_i$, of TSDOS, ATS, and PISync for CASE 2-04
immediately reached infinity after Node 3 is turned off. This shows that ATS failed to find new consensus among the nodes after Node 3 is turned off. However, TSDOS and PISync can find new consensus even after the Node 3 is turned off. Between TSDOS and PISync, TSDOS has the best performance. As can be noticed from the four error plots, the errors of TSDOS are able to converge to 0 after the changes while the error of PISync are bound in a fixed range. Even though the errors of PISync are bound within a certain range, the range will become larger whenever there are changes on the network connectivity.
The cIAE is not presented in CASE 2 because ATS will reach infinity (fails) when there is change on the network connectivity, and this will also cause the cIAE of ATS to reach infinity.

CASE 3

Result plots presented here are for the scenario where the skew rate, \( \alpha \), of every node in the network is time-varying to represent a more realistic scenario as suggested by Yildirim\(^{21}\). Figures 15 and 16 present the virtual time estimates and the virtual time estimation errors, respectively, for TSDOS, ATS, and PISync. It is noticeable that when there are changes in the nodes’ clock frequencies, there is a spike in the error magnitude in ATS and TSDOS, but both protocols can find a new consensus and the error will converge to 0. In PISync, whenever there is a change in the nodes’ clock frequencies, although there is no spike in the error magnitude, the error will change to a bigger bound range of values.

Figure 17 displays the cIAE of the three compared protocols in scenario CASE 3. From Figure 17, it can be realized that the ranking of the cIAE in descending order started with PISync (2.9930 \( \times 10^{11} \)), followed by TSDOS (3.2130 \( \times 10^{7} \)) and the last is ATS (3.4321 \( \times 10^{6} \)). As discussed above, the cIAE is a measure of the computational effort of the protocols. Thus, the largest value of the PISync shows that PISync consumes the most energy when there are time-varying clock frequencies involved.

Conclusion

In conclusion, this paper has successfully proposed an enhanced version of distributed time synchronization protocol using control theoretic approach which is also applicable in the scenarios usually faced in ASV. Firstly, this protocol is a distributed protocol and has light computational
energy requirement because each node runs the same light observer/estimator algorithm to estimate the virtual reference time. Besides, this protocol has proven its robustness performance against dynamic changes in topological structures through simulation. TSDOS can also synchronize the network even if a node is moving out of communication range for a certain period, which is a situation usually found in a WSAN on ASV. The proposed TSDOS algorithm has the best performance among the three compared protocols, under the normal scenario and the scenario where there are changes in the network connectivity. TSDOS either has the lowest cIAE or successfully finds a new consensus after the changes. Although TSDOS does not have the best performance in the scenario of varying clock frequencies, TSDOS is able to reach convergence. Future work will involve the hardware implementation of the proposed time synchronization algorithm on a group of interconnected sensor node network.

Acknowledgement
This work is supported by Research University (Rui) Grants (Grant No. 1001/PELECT/8014029), awarded by Universiti Sains Malaysia.

References