MU-Sync: A Time Synchronization Protocol for Underwater Mobile Networks

Nitthita Chirdchoo, Wee-Seng Soh, Kee Chaing Chua
Department of Electrical & Computer Engineering
National University of Singapore, Singapore
Email: {g0500102, elesohws, eleckc}@nus.edu.sg

ABSTRACT

Although there are numerous time synchronization algorithms recently proposed for terrestrial wireless sensor networks, none of these could be directly applied to underwater acoustic sensor networks. This is because they typically assume that the propagation delay is negligible, which is not the case in underwater. Furthermore, the sensor nodes in underwater tend to have some degree of mobility due to wind or ocean current, which complicates the problem even more by introducing time-varying delay.

In this paper, we propose a cluster-based synchronization algorithm for underwater acoustic mobile networks, called "MU-Sync". Our design avoids frequent re-synchronization by estimating both the clock skew and offset. As underwater mobile networks experience both time-varying and long propagation delay, previous works that estimate the clock skew using a single least square error linear regression tend to be inaccurate. In the MU-Sync, the clock skew is estimated by performing the linear regression twice over a set of local time information gathered through message exchanges. The first linear regression enables the cluster head to offset the effect of long and varying propagation delay; the second regression in turn obtains the estimated skew and offset. With the help of MAC-level time stamping, we can further reduce the nondeterministic errors that are commonly encountered by those synchronization algorithms that rely on message exchanges.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Distributed networks

General Terms

Algorithms, Design

Keywords

Clock synchronization, High latency networks, Underwater acoustic sensor networks, Sensor networks

1. INTRODUCTION

The clock synchronization problem has drawn considerable attention from researchers in the past few decades, especially in the area of wireless sensor networks, due to its wide variety of possible applications such as environmental monitoring, target tracking, security surveillance, and many more. In wireless sensor networks, each node performs its task (e.g., sensing the environment) in a distributed manner; the often time-sensitive data from multiple sensor nodes are then aggregated and converted to more meaningful information by using techniques such as data fusion. For example, in a target tracking application, while the interesting object is moving, sensors in different areas sense the object and report the presence of the object in its local vicinity. These distributed reports can then be fused to extract information such as the speed and direction of the moving object.

Most of the applications in sensor networks require that all sensor nodes have a common time scale (e.g., all synchronized), so that they can coordinate and collaborate with each other in order to accomplish their tasks. Basically, we can classify these applications into three categories based on the synchronization level required [2, 1]. Some applications merely require the order of the event occurrences, while there may be other applications that require the time interval of each of the event occurrences. Yet, there may also be applications that require the absolute time at which each event occurs. In addition, not only would the application layer find time-synchronization useful; other layers, such as MAC and networking layers, may also benefit from time synchronization. For example, PCAP [3] and the Slotted FAMA [4] are examples of MAC protocols that require time synchronization.

Because of the usefulness of time synchronization, numerous synchronization algorithms have been recently proposed for terrestrial wireless sensor networks; however, none of these can be directly applied to underwater acoustic sensor networks for several reasons. Firstly, the previously proposed algorithms are designed for high speed radio communication, and they typically assume that the propagation delay is negligible. In contrast, underwater communication mainly uses acoustic channel with a low propagation speed of approximately 1500 m/s, thus resulting in significantly longer propagation delay [5]. Radio communication is unsuitable in underwater due to its high attenuation rate, which severely limits its communication range. Although radio waves can still be used for long range communication at extra low frequencies (30-300 Hz), it is rather impractical due to the need for large antennas and high transmission power [6].

Secondly, the terrestrial synchronization algorithms typically do not worry much about the re-synchronization frequency. In contrast, synchronization overhead is an important issue in underwater acoustic networks, due to its low data rate resulting from its narrow available bandwidth. Specifically, the amount of available bandwidth depends on the targeted communication range; a long-range system that operates over several tens of kilometers may have a bandwidth of only a few kilohertz, while a short-range system operating over several tens of meters may have more than a hundred kilohertz of bandwidth [6]. Thus, the synchronization algorithm should be able to maintain a certain accuracy without the need for frequent re-synchronization, in order to avoid excessive consumption of the traffic capacity. Furthermore, when re-synchronization is required, the overhead incurred should not degrade the system performance.

Moreover, the nodes in underwater sensor networks tend to exhibit some degree of mobility due to wind and ocean current, even if they were designed to be "static" nodes without any self-propelling capability. In addition, Autonomous Underwater Vehicles (AUVs) may also be deployed for sensing tasks. Thus, the synchronization algorithm must be able to cope with the sensors' movement, which introduces time-varying delay. From empirical observation, the ocean current typically moves at the rate of 3 - 6 km/hr (around 0.83 - 1.67 m/s) [7], while the existing AUVs typically move at a rate of up to 2 m/s.

In this paper, we propose a cluster-based synchronization algorithm for underwater acoustic mobile networks, called "MU-Sync". Our design avoids frequent re-synchronization by estimating both the clock skew and offset. As underwater mobile networks experience both time-varying and long propagation delay, we estimate the skew and offset by performing least square error linear regression twice over a set of local time information gathered through message exchanges. With the help of MAC-level time stamping, we can further reduce the nondeterministic errors that are commonly encountered by those synchronization algorithms that rely on message exchanges.

The remainder of this paper is organized as follows. In Section 2, we discuss possible causes of error typically found in time synchronization. We then describe the previous works for clock synchronization in both terrestrial and underwater networks in Section 3. In Section 4, we provide the details of the MU-Sync, as well as an analysis of the possible synchronization error. Next, Section 5 describes the simulations that were carried out to compare the performance of the proposed schemes with several others. Section 6 then provides a discussion of the algorithm, and finally, we give our conclusions in Section 7.

2. THE CAUSES OF ERROR IN TIME SYNCHRONIZATION

In any clock synchronization algorithm, an error may still exist even at the instance immediately after the synchronization. As time progresses, this error grows proportionally with time, and re-synchronization is hence required. In order to avoid frequent re-synchronization, the error should be minimized. In this section, we explore the possible causes of error, and ways to reduce it. Specifically, we can divide them into two categories as will be discussed next.

2.1 Errors Caused by Uncertainty of Message Delivery Time

Many existing synchronization algorithms often utilize the technique of message exchange between synchronizing nodes, in order to acquire their local clock drift. By utilizing the message exchange technique, the common sources of error for clock synchronization, (first introduced by Koeptz and Schwab [8, 9]), come from the uncertainty of the following:

- **Send time**: The time used to construct the message and send the request to the MAC layer. It is nondeterministic and also dependent on the current load as well as the operating system. The error arising from the send time can be minimized by utilizing MAC-layer time stamping at the sender side.
- **Access time**: The delay incurred while waiting to access the channel until the transmission begins. The amount of access time depends on the current network traffic and the nature of the running MAC protocol. Physical layer time stamping can be used to eliminate this error.
- **Propagation time**: The time it takes to transmit the message from the sender to the receiver. The propagation time is highly deterministic depending on the distance between the sender and the receiver. In terrestrial sensor networks, it is often considered as a negligible contribution of synchronization error due to the high speed of radio wave. However, when dealing with underwater acoustic sensor networks, this becomes a major cause of synchronization error, as will be discussed in Section 4.3.
- **Receive time**: The time it takes to process the notification of an incoming message.

2.2 Other Causes of Synchronization Errors

Even when the clocks of two nodes are perfectly synchronized at the beginning, their clocks may drift if inconsistency occurs due to changes of the surrounding environment, such as when the nodes experience changes in temperature, pressure, battery voltage, etc. As underwater sensor nodes typically exhibit some mobility, it is highly likely that they encounter changes in the abovementioned parameters, and hence require re-synchronization more often than static terrestrial sensor networks. Moreover, the clock can also be affected by the interaction of other components of the sensor system. For example, the sensor may miss an interrupt while busy transmitting or receiving a packet, as described in [10].

3. RELATED WORK

The timer inside each clock uses a crystal oscillator operating at a certain angular frequency, which determines the rate at which the clock runs [1]; this is also widely referred to as the clock's "skew". Typically, each clock may have slight differences in angular frequency due to the manufacturing process. These skew differences cause the drift among the sensor nodes. In general, we often model the local time of

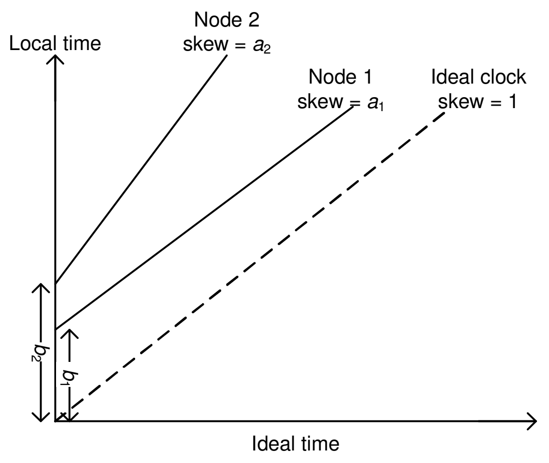

Figure 1: Illustration of clock drifts.

Node i using two parameters, namely, its skew and its offset, as follows:

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

where a_i and b_i are the skew and offset of Node i, and t is the ideal time or *Universal Time Coordinated* (UTC). The offset arises when each sensor node has a different starting time. Fig. 1 shows the drift of two clocks having different skews and offsets. Note that, while the offset causes constant error independent of time, the skew can cause increasing error as time progresses. Thus, in order to avoid the need for frequent resynchronization, the synchronization algorithm must be able to accurately estimate both the clock skew and offset.

The previously proposed synchronization schemes found in the literature can be divided into two categories: the receiver–receiver based approach, and the sender–receiver based approach. An example of the receiver-receiver based approach is the Reference Broadcast Synchronization (RBS) scheme [11]. In RBS, a node sends a beacon pulse to its neighbors. Upon receiving the pulse, the receiver marks its local time at which it receives the beacon pulse. By assuming that the propagation delay is negligible, each of the receivers is assumed to have received the beacon pulse at the same instance. Next, each pair of the receivers simply exchange their pulse receiving time to calculate the difference in their offsets. For higher accuracy, the scheme can be extended to use multiple beacon pulses to determine both the clock skew and the clock offset by using linear regression and statistical technique, respectively. As we can see, the RBS does not rely on the time stamp from the sender side; this reduces the sender's nondeterministic effect which we have previously discussed in Section 2. Although the authors claim that RBS can achieve very high accuracy (1.6 μs using 30 beacon pulses), its performance can degrade drastically when operating in mobile underwater acoustic networks, in which long and varying propagation delay is dominant. Moreover, with the use of unidirectional beacon pulse, it is impossible to compute and compensate for the propagation delay, thus leading to large synchronization error.

On the other hand, instead of using unidirectional message exchange as in the RBS, the sender-receiver approach combines bidirectional message exchange with local time stamping, in order to retrieve the node's offset and skew. Ganeriwal *et al.* proposed TPSN [2], a two-phase network-wide synchronization algorithm for wireless sensor networks. The TPSN first uses a *level discovery phase* to define the hierarchical topology of the whole network. Then, in the *synchronization phase*, a pair of nodes can learn their clock drift using bidirectional message exchange, with the sending node inserting its local time stamp on each message. The drift and propagation delay can be extracted from the time stamp collected from the two-way message exchange. The main drawback of TPSN is that it computes the clock drift by only estimating the offset. Without correcting the clock's skew, it is obvious that the TPSN will need frequent re-synchronization.

Although there has been extensive study of underwater acoustic networks from all around the world, the work in the area of underwater synchronization is still very limited. Syed and Heidemann recently propose the TSHL [12], which is a time synchronization algorithm designed for high latency networks. The design tries to minimize the synchronization error by estimating and compensating both the clock skew and offset, utilizing MAC-layer time stamping and bidirectional message exchange. While assuming that all the nodes in the network are static, the TSHL takes into account the long propagation delay when determining the clock offset. The clock skew estimation can be achieved by applying linear regression over multiple two-way reference packet exchanges. Our MU-Sync is different in that the cluster node takes the responsibility to initiate and compute the clock skew and offset of its neighboring nodes. By doing so, the cluster head can easily determine the number of reference packets needed in order to meet a certain level of accuracy. In addition, the TSHL assumes that the propagation delay is constant during the n reference packet exchanges, which is no longer valid in mobile networks.

Recently, Tian *et al.* proposed a localization and synchronization scheme for 3D underwater acoustic networks in [13], using atomic multilateration and iterative multilateration techniques. The scheme utilizes external anchor nodes that are located on the surface of the ocean, which already know their locations and time without error. The synchronization packet broadcasted from the anchor i includes the current location (x_i, y_i, z_i) of the anchor, and the time of transmitting the packet (t_i). In order to compute its location, each node needs to hear from at least five anchor nodes. Upon receiving enough synchronization packets, the node performs multilateration in order to obtain its location. Next, the node learns the drift between itself and the anchor by comparing its local time of receiving the packet with ideal time (t_i plus the propagation delay). The main drawback of this scheme is that it may not always be practical to have anchor nodes floating on the surface of the ocean, due to security reason. Moreover, the algorithm utilizes a hierarchical approach for network-wide synchronization (multi-hop networks). This means that, in order to synchronize a node, it needs to have at least five neighboring nodes of a higher hierarchical level, acting as beacon nodes. This may be quite difficult to achieve in underwater acoustic networks since the number of sensor nodes are typically limited due to economical reasons. The most serious drawback is that the scheme only aims to estimate the offset. Without estimating the clock skew, frequent resynchronization is expected.

4. HOW THE PROTOCOL WORKS

4.1 Overview of the MU-Sync

The MU-Sync is designed to minimize the drift between nodes by estimating and compensating both the skew and offset using a two-phase operation, namely, the *skew and offset acquisition phase*, and *the synchronization phase*. In the first phase, the clock skew and offset is estimated by applying linear regression twice over a set of n reference beacons. While all of the existing synchronization algorithms perform linear regression only once to retrieve the estimated skew, the MU-Sync performs it twice. The first regression allows the cluster head to extract the amount of propagation delay that each reference (REF) packet encounters. After adjusting the REF beacons' timings with their respective propagation delays, a second linear regression is performed over this new set of points, from which the estimated skew $(\hat{\hat{a_y}})$ and offset $(\hat{\hat{b_y}})$ of Node y can be obtained.

Since the MU-Sync is cluster-based, it can easily be applied to mobile multi-hop underwater sensor networks. In contrast to the TSHL, in which each node computes its own skew and offset, the cluster head in the MU-Sync takes the responsibility to start the synchronization process and calculate its neighbors' skew and offset.

4.2 Details of the MU-Sync

In Phase 1, the synchronization process starts with the cluster head (also referred to as Node x throughout the rest of this section) broadcasting the i^{th} REF packet to its neighbors at time $T_{1,i}$, as shown in Fig. 2. Upon receiving the REF packet, the neighboring node marks its local time as $T_{2,i}$ and responds to the cluster head at time $T_{3,i}$, informing it about the $T_{2,i}$ and $T_{3,i}$ timestamps. Note that, in order to reduce the chances of a collision, each neighboring node may introduce some small random interval before responding to the REF packet. When the cluster head receives the response from its neighboring node y at time $T_{4,i}$, it waits for some duration denoted as REF_TX_INT before transmitting the $(i+1)^{\text{th}}$ REF packet, and continues the same procedure until it has reached the number of required REF packets, n, or until the error of the linear regression is below a certain threshold. Next, the cluster head performs the first linear regression over the set of local times reported by Node y to obtain its first estimated skew $(\hat{a_y})$, as illustrated in Fig. 3. The value of $\hat{a_y}$ is then used to compute the amount of one-way propagation delay that each REF packet has encountered, using the local time stamps $T_{1,i}$, $T_{2,i}$, $T_{3,i}$ and $T_{4,i}$:

$$\hat{D}^{x\to y}_{t_1\to t_2,i} = \frac{1}{2}\left[T_{4,i} - T_{1,i} + \frac{T_{2,i} - T_{3,i}}{\hat{a_y}}\right] \qquad (2)$$

where $D^{x\to y}_{t_m\to t_n,i}$ denotes the propagation delay between node x's location at time t_m and node y's location at time t_n of the i^{th} REF packet, where $1 \le i \le n$. We next subtract the estimated propagation delay corresponding to each of the data points to obtain a new set of data points. The cluster head then runs the second linear regression to obtain the final estimated skew and offset of neighboring node y, denoted by $\hat{\hat{a_y}}$ and $\hat{\hat{b_y}}$, respectively.

In the synchronization phase shown in Fig. 4, the cluster head broadcast all neighbors' $\hat{\hat{a_y}}$ and $\hat{\hat{b_y}}$, so that every neighbor can keep track of these parameters. When every node in the cluster knows the skew and offset of every other node in the cluster, we can claim that cluster-wide synchronization has been achieved.

4.3 Error Analysis of Propagation Delay Estimation

Now, let us assume that the clock of Node x can be modeled by using its skew and offset relative to an ideal clock in a similar form as (1). The following set of equations of the node's local time can hence be derived:

$$T_1 = a_x t_1 + b_x \qquad (3)$$

$$T_2 = a_y t_2 + b_y \qquad (4)$$

$$T_3 = a_y t_3 + b_y \qquad (5)$$

$$T_4 = a_x t_4 + b_x \qquad (6)$$

The ideal time of t_2 and t_4 can also be written as

$$t_2 = t_1 + D^{x\to y}_{t_1\to t_2}, \qquad (7)$$

and

$$t_4 = t_3 + D^{y\to x}_{t_3\to t_4}. \qquad (8)$$

Our objective is to estimate the one-way propagation delay $D^{x\to y}_{t_1\to t_2}$ from the round trip time, which can be written as

$$D^{x\to y}_{t_1\to t_2} = \frac{(t_2 - t_1) + (t_4 - t_3)}{2}, \qquad (9)$$

By substituting (7) and (8) into (2), the estimated propagation delay $\hat{D}^{x\to y}_{t_1\to t_2}$ computed at Node x is

$$\hat{D}^{x\to y}_{t_1\to t_2} = \frac{1}{2}\left[a_x(t_4 - t_1) - \frac{a_y}{\hat{a_y}}(t_3 - t_2)\right]. \qquad (10)$$

Since we are interested in the relative drift of Node y compared to Node x, we let $a_x = 1$ which results in

$$\begin{aligned}\hat{D}^{x\to y}_{t_1\to t_2} &= \frac{1}{2}\left[(t_4 - t_1) - \frac{a_y}{\hat{a_y}}(t_3 - t_2)\right] &(11)\\ &= \frac{1}{2}\Bigg[(1 - \frac{a_y}{\hat{a_y}})(t_3 - t_1) + D^{y\to x}_{t_3\to t_4} \\ &\quad + \frac{a_y}{\hat{a_y}} D^{x\to y}_{t_1\to t_2}\Bigg] &(12)\end{aligned}$$

As $D^{x\to y}_{t_1\to t_2}$ is an average of the propagation delay obtained from $t_4 - t_3$ and $t_2 - t_1$, while the actual delay is $t_4 - t_3$, we can calculate the error of propagation delay estimation (Δ) as

$$\begin{aligned}\Delta &= \left|\hat{D}^{y\to x}_{t_3\to t_4} - D^{y\to x}_{t_3\to t_4}\right| &(13)\\ &= \frac{1}{2}\Bigg|(1 - \frac{a_y}{\hat{a_y}})(t_3 - t_1) - D^{y\to x}_{t_3\to t_4} \\ &\quad + \frac{a_y}{\hat{a_y}} D^{x\to y}_{t_1\to t_2}\Bigg|. &(14)\end{aligned}$$

Equation (14) indicates that the error of the propagation delay estimation depends on three parameters:

1. The relative drift between the estimated and the real skew: $\frac{a_y}{\hat{a_y}}$
2. The time interval $t_3 - t_1$

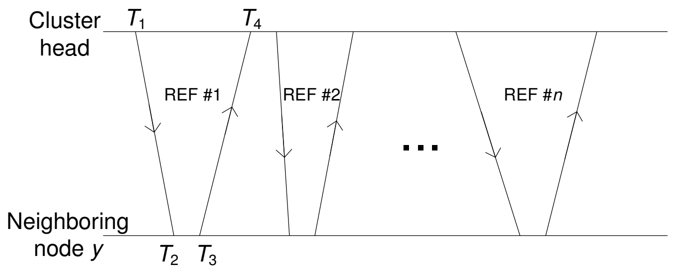

Figure 2: Phase 1: Skew and offset acquisition phase. Note that the propagation delay can vary during the REF packet exchange.

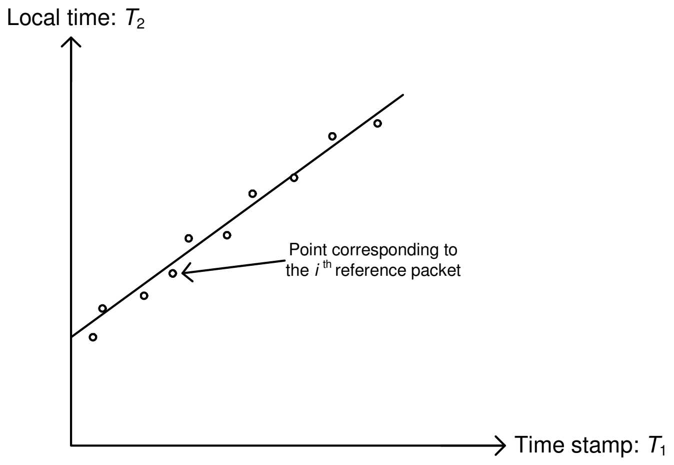

Figure 3: Linear regression at the cluster head with a total of 10 reference packet responses from node y.

3. The value of $\left|D^{y\to x}_{t_3\to t_4} - \frac{a_y}{\hat{a_y}}D^{x\to y}_{t_1\to t_2}\right|$

We can rewrite (14) as

$$\Delta = \frac{1}{2}\left|(1-\frac{a_y}{\hat{a_y}})(t_3-t_1) - \frac{v_{\text{node}}(t_4-t_3)}{v_{\text{s}}} + \frac{a_y}{\hat{a_y}}\frac{v_{\text{node}}(t_2-t_1)}{v_{\text{s}}}\right|, \quad (15)$$

where v_{node} and v_{s} are the relative speed between Node x and Node y and the speed of sound in underwater, respectively. After obtaining the propagation delay, the cluster head is now able to estimate Node y's offset, $\hat{b}_y$, by first deducting the propagation delay effect and running the linear regression again to obtain both $\hat{\hat{a_y}}$ and $\hat{\hat{b}}_y$.

5. SIMULATIONS AND RESULTS

5.1 Simulation Setup

In our simulation setup, the sensor nodes are allowed to move randomly within an area of 1000 m by 1000 m. We assume that the speed of sound in underwater is constant at 1500 m/s, and there is no skew variation arising from a change in the environment as previously discussed in Section 2.2, so that we can concentrate solely on the effect of the parameters that we are interested in. The nondeterministic errors encountered during the message exchange is modeled using Gaussian distribution, as suggested by Elson and Estrin [11]. Unless specified otherwise, we use the following set of parameters for our simulations:

- Maximum speed of the sensor node ($V_{\max}$) is 2 m/s.
- Clock skew is 40 ppm.
- Clock offset is 10 ppm.
- The number of beacons used to perform linear regression is 25.
- The duration $t_3 - t_2$ is 0 s.
- The time interval between two successive reference packets is 5 s.
- The sensor nodes change its speed randomly within the range of $[0, V_{\max}]$, with an average interval of 600 s.
- The sensor nodes change its direction randomly within the range of $[-45°, 45°]$, with an average interval of 600 s.
- Clock granularity is 1 μs.
- Receive jitter is 15 μs.

We study the following parameters in order to investigate their effects on the MU-Sync's performance:

1. The node's initial skew
2. The number of beacons
3. The duration of $t_3 - t_2$
4. The frequency at which the sensors change direction
5. The speed of the sensors

We choose to benchmark our scheme with the TSHL, as well as a network that does not undergo any synchronization. Although the TSHL is designed for static underwater sensor networks, which assume long but constant propagation delay, it is the closest form of underwater synchronization scheme available thus far.

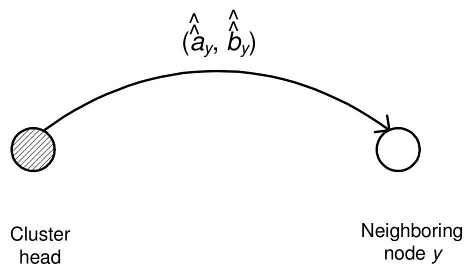

Figure 4: Phase 2: Synchronization phase.

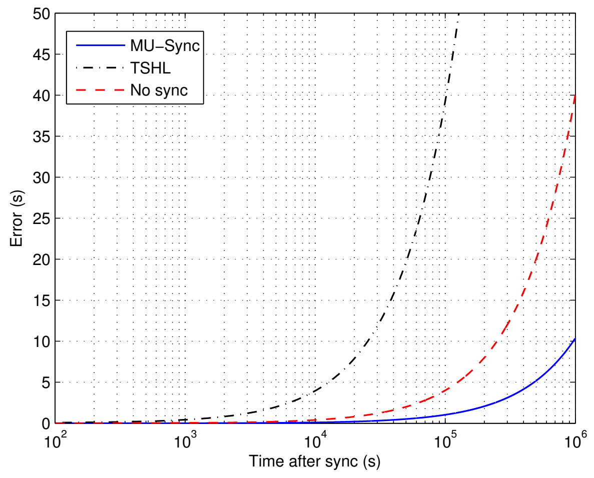

Figure 5: The error in time estimation VS the time elapsed since synchronization.

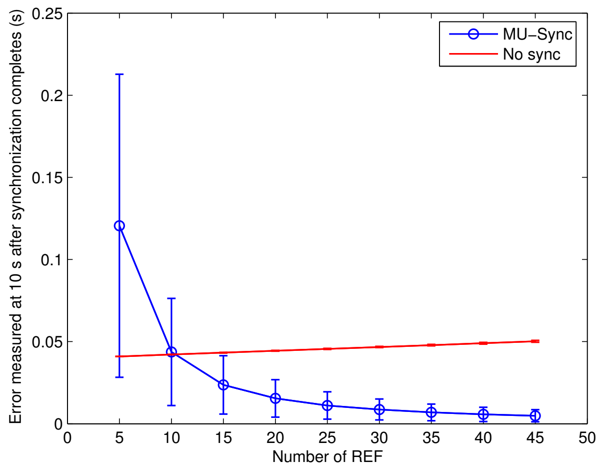

Figure 7: Effect of changing the number of REF beacons.

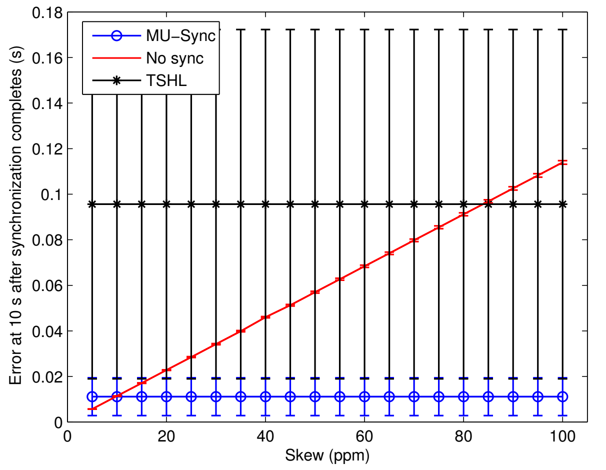

Figure 6: Effect of clock skew.

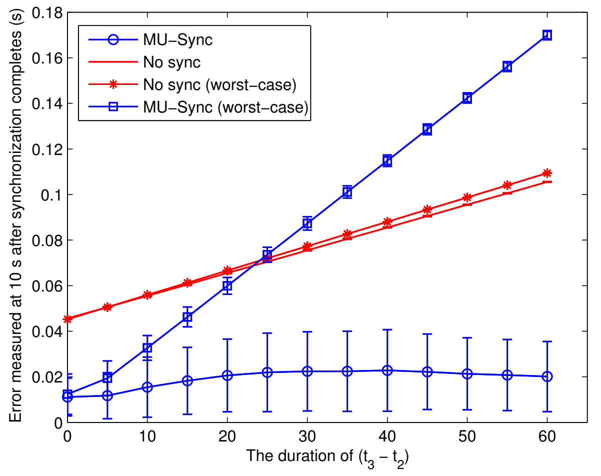

Figure 8: Effect of (t_3 - t_2).

5.2 Results

In all the results shown in this section, each data point is obtained from the average of 1000 simulation runs. The error bars in the figures represent the standard deviations. Fig. 5 shows how the error in time estimation grows for each scheme as time elapses since the last synchronization. As can be seen, the MU-Sync perform better than the TSHL significantly. In fact, the TSHL is even worse than the case where no synchronization is performed. Its poor performance arises from its poor accuracy in estimating the skew. This is due to its assumption that the inter-node propagation delay is constant during the skew estimation process. When a node moves, the propagation delay varies with time; hence, if the linear regression is applied blindly without taking this into consideration, it causes inaccurate skew estimation. The small error in skew estimation can cause severe drift as time progresses. For example, the error can grow as large as 8 s within a day of operation even with a skew error that is as small as 0.0001.

When we vary the neighboring nodes' clock skew from 5-100 ppm, Fig. 6 shows that the MU-Sync's performance is independent of the node's initial skew. However, when the initial skew error is less than 10 ppm, the MU-Sync's performance is worse than the unsynchronized one. We can also see in Fig. 6 that both the TSHL and the MU-Sync achieve a constant average error regardless of the neighboring node's initial skew, because both algorithms estimate the skew and try to compensate for it. However, the TSHL's performance is much worse than even the unsynchronized version. The high standard deviation noticed from the error bars for the TSHL is due to the high variation of the propagation delay arising from node mobility. This happens when the linear regression is applied to estimate the skew over the set of points $\{T_{1,i}, T_{2,i}\}$ where $1 \leq i \leq n$. For the rest of this section, we decide to continue our study without the TSHL as thus far its performance is poorer than the unsynchronized version.

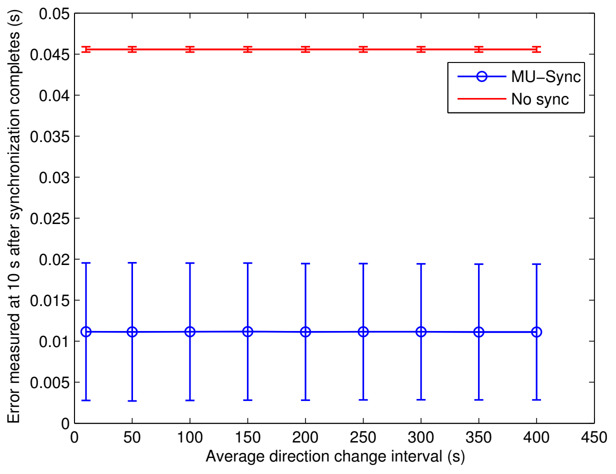

Figure 9: The effect of the average direction change interval.

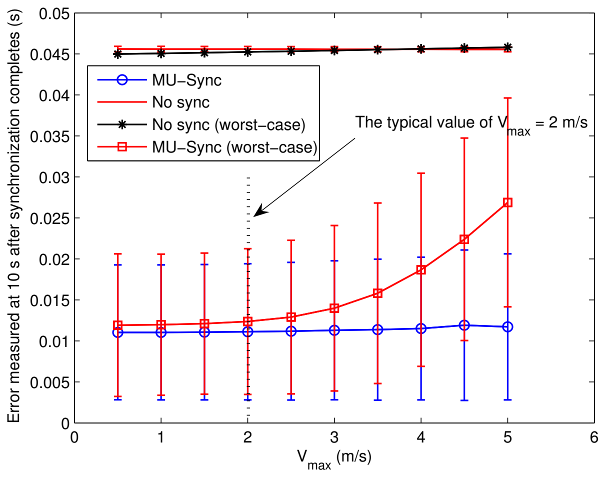

Figure 10: Effect of $\mathbf{V_{max}}$.

We next study the effect of the number of beacons on the synchronization error. It is obvious that a higher number of reference beacons used for linear regression will result in a lower error in estimation. Fig. 7 illustrates that a finer synchronization can be achieved by adjusting the number of reference beacons fed into the linear regression. With our cluster head synchronization approach, the MU-Sync can easily adjust the number of beacons used adaptively. Note that the larger the number of beacons to be collected for linear regression, the longer it takes to finish the synchronization process; this also explains why the slope of the unsynchronized version is linearly increasing as seen in Fig. 7.

The varying of the time interval $t_3 - t_2$ also affects the synchronization error. As shown in Fig. 8, when we vary the duration of $t_3 - t_2$ from 0 s (the neighboring node responds to the REF beacon right after it finishes receiving the packet) to 25 s (the neighboring node responds 25 s after it receives the REF), the synchronization error of the MU-Sync increases with the duration of $t_3 - t_2$. However, when the interval of $t_3 - t_2$ goes beyond 25 s, the error tends to stabilize, or even decrease. The explanation is that the parameter $t_3 - t_2$ does not directly affect the synchronization error; instead, the relative distance between the cluster head and the neighboring node at time $t_1 \rightarrow t_2$ and $t_3 \rightarrow t_4$ plays a more important role. Since we are currently using half of the round trip time to estimate the one-way propagation delay, the estimation error depends on the value of $\left|D^{x\rightarrow y}_{t_1\rightarrow t_2} - D^{y\rightarrow x}_{t_3\rightarrow t_4}\right|$. We verify our claim by examining at the worst-case plot. Here, we notice that the error increases steeply as the duration of $t_3 - t_2$ increases. This effect is less significant when the nodes move in a more random manner.

Fig. 9 shows that changing the direction of the nodes frequently does not significantly affect the synchronization error. Here, we vary the average direction change interval from 10 s to 400 s.

Fig. 10 shows the impact of the sensor's speed on the synchronization error. As can be seen, the synchronization error increases with the parameter $V_{\max}$ (look at the worst-case MU-Sync plot). This is because, when the sensor is allowed to move very fast, the value of $\left|D^{x\rightarrow y}_{t_1\rightarrow t_2} - D^{y\rightarrow x}_{t_3\rightarrow t_4}\right|$ can be so large that using (12) to estimate the propagation delay is no longer accurate enough. Fortunately, in most networks, we would expect the nodes to undergo speed and direction changes over time, rather than persisting in the worst-case setting all the time. Therefore, the effect of $V_{\max}$ is, on the average, much less significant on the synchronization error (as indicated by the average MU-Sync plot).

6. DISCUSSION

In this section, we discuss the applicability of the MU-Sync, and also how it may be improved.

- Although the MU-Sync seems to have a higher overhead than the TSHL, as it requires the neighboring node to send a response for every REF packet received, it can easily be integrated with existing handshaking MAC protocols such as MACA [14], Slotted FAMA [4], MACA-MN [15], PCAP [3], etc., by piggybacking the REF and its response within the RTS/CTS packets. While the MU-Sync could achieve finer synchronization by having a higher number of beacons messages, the TSHL may not benefit from this approach when the nodes are mobile, since it does not account for time-varying propagation delay. In fact, since a higher number of beacon messages also require a longer duration to finish collecting the beacons, the TSHL may become even more vulnerable to violating the constant propagation delay assumption.

- From our simulation, Fig. 8 (look at the worst-case) shows that the MU-Sync cannot cope when the duration of $t_3 - t_2$ is longer than approximately 25 s. The major factor that causes the error comes from the technique used to estimate the one-way propagation delay. Although (12) is widely used to calculate the one-way propagation delay from the round trip time, it may not be suited in our scenario. Since the nodes keep moving during the interval (t_2, t_3), significant error may be introduced when the propagation delay is estimated as half the round-trip time. A better method is hence needed in order to achieve higher accuracy.

- One advantage that the MU-Sync has over the TSHL is that, when the cluster head broadcasts its neighboring nodes' estimated skew and offset, every node learns the estimated skew and offset of all other nodes in the same cluster, instead of just the relative parameters between the cluster head and a particular node.
- Although we do not discuss how a cluster selects its cluster head in this paper, the task can be achieved by applying an existing cluster head selection algorithm as presented in [16, 17, 18].

7. CONCLUSION AND FUTURE WORK

In this paper, we have presented a cluster-based synchronization algorithm for mobile underwater acoustic networks, known as the MU-Sync. Unlike those existing synchronization schemes designed for terrestrial sensor networks that treat the propagation delay as negligible, the MU-Sync takes into account both long and time-varying propagation delays. Through simulations, we find that the accuracy of the MU-Sync is highly dependent on the accuracy of the propagation delay estimation, as it is a major contributor to synchronization error for underwater acoustic networks. We are currently using half of the round-trip time as an estimation of the one-way propagation delay. This may result in low accuracy if the propagation delay varies significantly within the round trip message exchange. In our future work, we plan to concentrate on how the varying propagation delay can be estimated more accurately, while maintaining low overhead.

8. REFERENCES

[1] F. Sivrikay and B. Yener, "Time Synchronization in Sensor Networks: A Survey," in *IEEE Network*, Volume 18, Issue 4 July-Aug. 2004, pp 45–50.

[2] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-Sync Protocol for Sensor Networks," in *Proc. 1st int. conf. on Embedded networked sensor systems*, Sept. 2002.

[3] X. Guo, M. R. Frater, and M. J. Ryan, "A propagation-delay-tolerant collision avoidance protocol for underwater acoustic sensor networks," in *Proc. MTS/IEEE OCEANS'06*, 2006.

[4] M. Molins, M. Stojanovic, "Slotted FAMA: a MAC protocol for underwater acoustic networks," in *Proc. MTS/IEEE OCEANS'06*, 2006.

[5] N. Chirdchoo, W. S. Soh, K. C. Chua, "Aloha-based MAC Protocols with Collision Avoidance for Underwater Acoustic Networks," in *Proc. IEEE INFOCOM 2007*, May 2007.

[6] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Elsevier's Journal of Ad Hoc Networks*, vol. 3, no. 3, 2005, pp. 257–279.

[7] P. Xie and J. H. Cui, "SDRT: A Reliable Data Transport Protocol for Underwater Sensor Networks," Technical Report: UbiNet-TR06-03 , Feb. 2006

[8] H. Kopetz and W. Schwabl, "Global time in distributed real-time system," Technical Report 15/89, Technische Universitat Wien, 1989.

[9] H. Kopetz and W. Schwabl, "Clock Synchronization in Distributed Real-Time Systems," in *IEEE Transactions on Computers*, C-36(8), Aug. 1987, p. 933–939.

[10] Q. Li and D. Rus, "Global Clock Synchronization in Sensor Network," in *IEEE Transactions on Computers*, vol. 55, no. 2, Feb 2006, pp. 214–226.

[11] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Time Synchronization using Reference Broadcasts," in *Proc. 5th Symp. Op. Sys. Design and Implementation*, Boston, MA, Dec. 2002.

[12] A. A. Syed and J. Heidemann, "Time Synchronization for High Latency Acoustic Networks," in *Proc. INFOCOM 2006*, April 2006, pp. 1–12.

[13] C Tian, W. Liu, J. Jin, J. W, and Y. Mo. "Localization and Synchronization for 3D Underwater Acoustic Sensor Networks," Springer Berlin / Heidelberg , pp. 622–631, 2007.

[14] P. Karn, "MACA-a new channel access method for packet radio," in *Proc. ARRL/CRRL*, 22 Sept, 1990.

[15] N. Chirdchoo, W. S. Soh, K. C. Chua, "MACA-MN: A MACA-based MAC Protocol for Underwater Acoustic Networks with Packet Train for Multiple Neighbors" in *Proc. IEEE VTC2008-Spring*, Singapore, May 2008.

[16] H. Chen and S. Megerian, "Cluster Sizing and Head Selection for Efficient Data Aggregation and Routing in Sensor Networks," in *Proc. IEEE WCNC 2006*, pp. 2318–2323, 2006.

[17] K. Dasgupta, K. Kalpakis, and P. Namjoshi. "An Efficient Clustering-based Heuristic for Data Gathering and Aggregation in Sensor Networks," in *Proc. IEEE WCNC 2003*, March 2003.

[18] G. Chopra, S. Srivastava, and A. Karandikar, "A novel clustering strategy for efficient routing in adhoc networks," in *Proc. IEEE ICPWC 2005*, pp. 67–71, 2005.