# Time Synchronization for High Latency Acoustic Networks - Extended Technical Report

ISI-TR-2005-602b

Affan A. Syed
USC/ISI
4676 Admiralty Way
Marina Del Rey, CA 90292
Email: asyed@isi.edu

John Heidemann
USC/ISI
4676 Admiralty Way
Marina Del Rey, CA 90292
Email: johnh@isi.edu

***Abstract*— Distributed time synchronization is an important part of a sensor network where sensing and actuation must be coordinated across multiple nodes. Several time synchronization protocol that maximize accuracy and energy conservation have been developed, including FTSP, TPSN, and RBS. All of these assume nearly instantaneous wireless communication between sensor nodes; each of them work well in today's RF-based sensor networks. We are just beginning to explore *underwater* sensor networks where communication is primarily via acoustic telemetry. With acoustic communication, where the propagation speed is nearly five orders of magnitude slower than RF, assumptions about rapid communication are incorrect and new approaches to time synchronization are required. We present Time Synchronization for High Latency (TSHL), designed assuming such high latency propagation. We show through analysis and simulation that it achieves precise time synchronization with minimal energy cost. Although at very short distances existing protocols are adequate, TSHL shows twice the accuracy at 500m, demonstrating the need to model both clock skew and propagation latency.**

## I. Introduction

Time synchronization is an important part of many distributed applications. In the classical networking and operating system literature it has been widely studied in context of database queries [1] and security applications [2]. In the Internet, NTP is the canonical approach to provide distributed time synchronization [3].

Time synchronization is even more important in sensor networks, where applications such as acoustic beamforming [4] and target tracking [5] require collaboratively processing of time-sensitive data [6]. Sensor networks add the additional requirement that energy consumption of the synchronization protocol be minimized.

Many time synchronization protocols for sensor networks have been proposed recently [7], [6], [8], [9]. These mechanisms provide a high degree of precision while being reasonably energy efficient. The protocols adopt increasingly sophisticated approaches to reduce noise and account for latency in communications, but *all* assume that propagation latency is negligible and thus can be effectively factored out of design consideration.

While assumptions about propagation latency are appropriate for RF-based communications (where these protocols were intended), a number of researchers are beginning to explore *underwater* sensor networks (for example, see a recent survey [10]). Underwater, radio propagation is very limited (Mica-2 transmit range has been measured as less than 1m in fresh water [11]). *Underwater acoustic* communication provides a viable alternative [12]. A number of recent efforts have begun exploring acoustic communication for underwater sensor networks. In fact, commercial systems exist today, typically focused on long-range communication. We have begun examining *short-range* underwater acoustics (as described in Section II-A).

Table I compares radio-based networks for general computing, sensor networks and satellites with short-range acoustic networks. While bitrates of underwater acoustics are comparable to current sensor network radios, propagation delay much higher. This delay is due to the five-orders-of-magnitude difference in the speed of sound in water compared to RF propagation: 1500 m/s compared to $3\times10^8$m/s. Underwater acoustic communication can be thought of as a "long slim pipe", with the worst features of satellite and RF-based wireless links.

The high propagation delay of underwater acoustics is especially hazardous for time synchronization algorithms. While NTP tolerates large delay [3], it does not consider energy consumption issues. Sensor-network-based time synchronization protocols consider energy consumption, but all current protocols are optimized for RF-based networks, assuming nearly instantaneous and simultaneous reception (RBS [6], FTSP [8]) or ignore clock drift during synchronization (TPSN [7], LTS [9]).

The main contributions of this paper include identifying the constraints and proposing a system architecture for short-range, underwater acoustic communication and sensor networks (Section II); quantifying the inaccuracies these constraints impose on current time synchronization protocols (Section IV); and introducing a new protocol, *Time Synchronization for High Latency* (TSHL), that compensates for high-latency communication while also minimizing energy consumption (Section V).

The key idea in TSHL is that it splits time synchronization into two phases. In the first phase, nodes estimate clock skew to a centralized timebase. In the second phase they swap

TABLE I
COMPARISON OF LINK CHARACTERISTICS

| Characteristic | Satellite | 802.11 RF | Chipcon RF | underwater acoustic (short range) |
|---|---|---|---|---|
| **Bit Rate** | 155 Mb/s | 11 Mb/s | 20–50kb/s | 20–50 kb/s |
| **Typical BER** | $10^{-10}$ | $10^{-5}$ | $10^{-5}$ | $10^{-2}$[13] |
| **Propagation Delay** | $\sim$120 ms | $< 1$ $\mu$s | $< 1$ $\mu$s | $\sim$300 ms |
| **Distance** | $\sim$ 42,000km | $<$3km | $< .5$km | $< .5$km |

skew compensated synchronization messages to determine the offset. The first phase is impervious to the propagation latency, while the second phase explicitly handles propagation delay induced errors.

We demonstrate the importance of this approach by analysis (Section IV) and simulations (Section VI). Analysis and simulations show that RBS and FTSP are not applicable to high-latency networks because they do not consider propagation latency at all. Through simulations we compare TSHL to TPSN, the closest current protocol. An implementation of TSHL on an in-the-air testbed is underway. We discuss how this in-the-air acoustic network will approximate the latency of an underwater network in Section VII.

## II. BACKGROUND

### A. Underwater Acoustic Sensor Networks

Although[1] radio-frequency communication is ideal for surface sensor networks, typical RF propagates very poorly underwater. Experimental tests with 433MHz Chipcon radios show propagation of less than 1m at full Mica-2 transmit power [11]. Although long-wavelength RF can penetrate water, it requires large transmit power and large antennae, making it inappropriate for small, low-power sensor nodes. This limitation suggested the use of robotic data mules to Zhang et al. [11], but the navigation and control problems there remain open issues.

These constraints suggest *acoustic communication* as an important alternative to RF. Stojanovic provides a very good overview of the options and challenges here [14]. A major difference between RF and acoustic propagation is the velocity of propagation. Radio waves travel at the speed of light, and, at ranges and frequencies typical for sensor networks, in a straight line. Acoustic transmission in water occurs at the speed of sound, which is around 1500 meters/sec. However the speed of sound in water varies significantly with temperature, density and salinity causing acoustic wave to travel on curved paths. Traditional approaches in this field have focused on medium- and long-range approaches considering 1–10km and 10–100km distances. Stojanovic identifies a number of sources of signal loss: spreading and absorption loss (a function of distance and frequency); multipath reflections from the surface, obstacles, the bottom, and temperature variations in the water; noise due to artificial and natural sources; and scattering from reflections off a potentially rough ocean surface.

Many of these forms of loss are unique to acoustic communication at *longer* distances. In particular, multipath reflections, temperature variation, and surface scattering are all exaggerated by distance. Inspired by the benefits of short-range RF communication in sensor networks, we seek to exploit *short-range underwater acoustics*. We are developing a multi-hop acoustic network targeting communication distances of 50–500m and communication rates of 5kb/s.

Using a simple FSK signaling scheme we anticipate sending 5 kb/s over a range of 500m, using a 30mW transmitter output. The primary limitation on performance is set by spreading loss and the background noise of the ocean. As with RF, we expect a combination of software and hardware techniques such as duty cycling can result in energy requirements a fraction of the basic transmit costs.

Figure 1 shows a conceptual view of a target system using underwater acoustic communication. A relatively dense deployment of sensor nodes (small yellow circles) are connected by wireless underwater acoustic links. These communicate, possibly via multiple hops, with each other and with tethered nodes (small green squares) at buoys to users on a platform or ship. Where possible we can exploit a hybrid network with high-speed RF communication on buoys (large green squares), or with underwater or surface-based robotic or manually operated data mules (not shown). We see this network being used for applications such as 4-D seismic monitoring of oil reservoirs. In these applications a seismic signal is generated centrally. It then propagates and reflects through the geologic formation and is measured at our sensors. Accurate time synchronization is essential for this class of applications, since differences in signal propagation delay are used to infer reservoir status.

While our work in prototyping underwater acoustics is ongoing and this multi-hop, hybrid network clearly requires much more work, our acoustic modem design places bounds on the latency and bitrates that we can expect. With 50–500m distances and a 1500m/s speed-of-sound we expect 300ms worst-case propagation latencies (ignoring any MAC effects). While we focus on a network with these constraints, our work is also applicable to time synchronization in other high-latency networks. If applied to longer-distance underwater acoustic networks we expect higher jitter in communications timing,

[1]The material in this section is joint work with Jack Wills, Wei Ye, and others in the SNUSE group at USC/ISI.

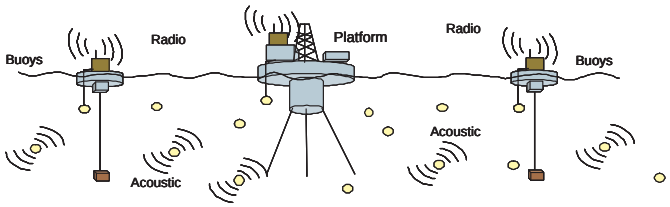


Fig. 1. Underwater Acoustic Sensor Network Topology

and so more measurements would be required to get similar levels of accuracy.

### B. Need for Time Synchronization: Clock offset and skew

Two challenges face synchronization of distributed clocks. First, they must by synchronized to a single common event in absolute time or *offset* (shown as $b$ in Figure 3). Second, because clocks are imperfect and run at slightly different rates, one must determine the *skew* of a given clock relative to some absolute frequency.

Since computers boot at different times, there must be some way for distributed computers to determine a common offset. Offset can be determined by a single message exchange, provided we can compensate for any sources of non-deterministic latency in the path (described in Section II-C). As examples, NTP compensates for jitter in the path by using control theoretic mechanisms [3], while RBS uses a jointly visible broadcast packet to synchronize nodes [6].

Time in most modern, inexpensive computers is derived from oscillating frequency of a quartz crystal. Due to environmental variation (temperature, humidity, etc.) or minor manufacturing differences, variations in crystal oscillation frequency on the order of 15–25 parts per million are common [15]. Thus, even nodes that are synchronized to a common offset will drift out of synchronization over time. In Figure 3 the dotted $y = x$ line represents an ideally ticking clock where time on the $x$-axis matches time on the $y$-axis perfectly. Two clocks ticking at different rates with ratio $a$ will appear as a line with a different slope $y = ax$. The ratio $a$ represents clock skew. Protocols handle clock skew by estimating and compensating for it. For example, NTP, keeps the long term average skew error low by using a phase-locked loop to correct the local clock frequency [3],while RBS explicitly calculates this skew using linear regression [6].

With reasonable quality oscillators, clock skew is generally small enough that it can be ignored over short time intervals. Thus existing protocols ignore skew when synchronizing offset. While appropriate for RF-based networks, we show that this assumption does not hold for slower acoustic communication (see Section IV). For these networks, one must consider skew even *during* synchronization.

### C. Sources of Errors in Time Synchronization

The major cause of error in time synchronization schemes is the non-determinism in the latency estimates of the message

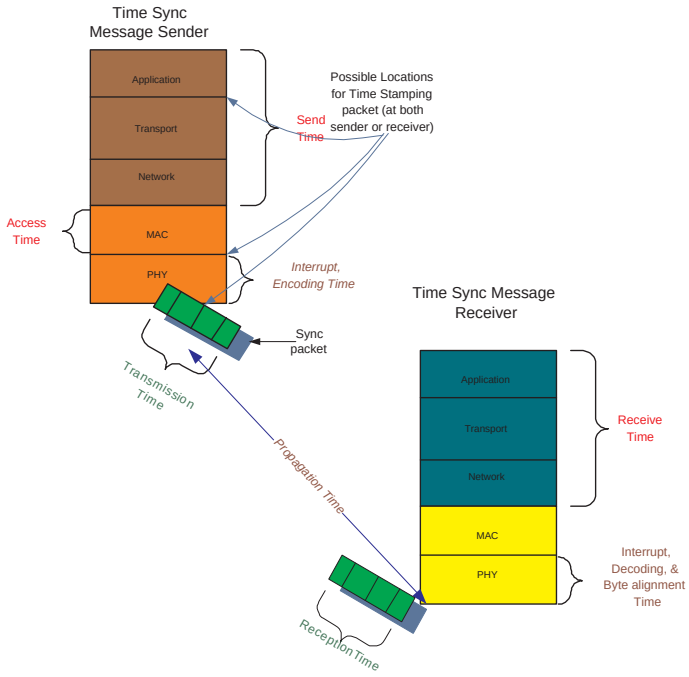


Fig. 2. Sources of error in estimating message latency

delivery delay. A description of sources of variability was first described by Koeptz and Schwabl [16] and extended recently by Horauer et. al. [17] incorporating physical layer jitter that cannot be over looked for high precision time synchronization. We briefly review these sources of error below and in Figure 2.

1) **Send Time:** The delay in the packet traversal from the message assembly at the application layer all the way down to MAC layer. Highly non-deterministic.
2) **Access Time:** Is the channel contention time, that in dense broadcast medium such as ours can be in the order of hundreds of milliseconds. Least deterministic part of the message delivery.
3) **Interrupt Handling Time:** The delay between the radio chip raising and the microcontroller responding to an interrupt. Can be an issue if interrupts are disabled on the microcontroller.
4) **Transmission and Reception Time:**The delay in sending or receiving the entire length of the packet over the channel. Largely deterministic, is a function of bandwidth and packet size.
5) **Propagation Time:** The delay, for a particular symbol of the message, in traversing all the way to the receiver. The propagation time can be deterministic if the speed of propagation is assumed constant, and endpoint location is known, or if synchronization exchange is performed with assumption of path symmetry. This delay can be significant in the underwater acoustic channel since assuming clocks will not skew over packet exchanges would be incorrect.
6) **Encoding and Decoding Time:** The time taken by

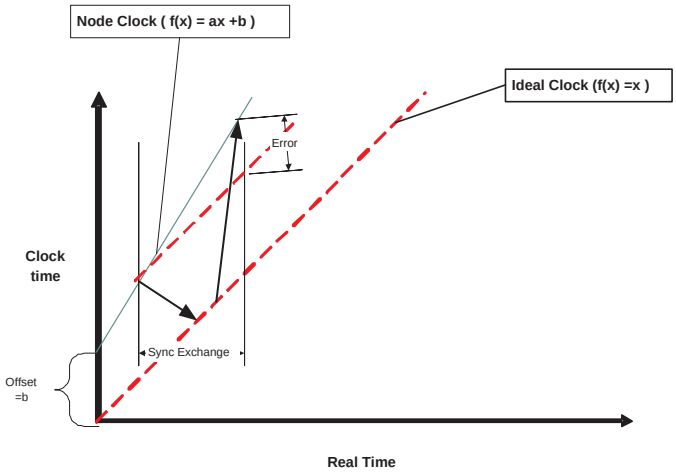


Fig. 3. Effect of clock skew

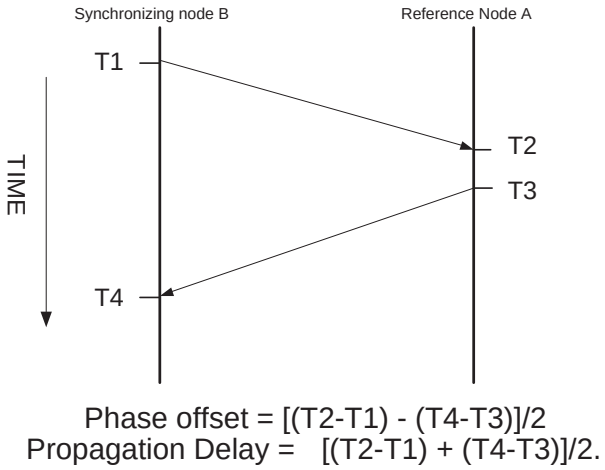


Fig. 4. Sender-Receiver Synchronization

the radio chip to encode/decode and transform a part of the message to/from electromagnetic waves. This time is deterministic and is in the order of hundred microseconds [8].

7) **Byte Alignment Time:** The delay because of the different byte alignment at the receiver. This time is deterministic and can be computed on the receiver side from the bit offset and the speed of the radio.
8) **Receive Time:** Time for the incoming message to traverse up till the receiver application. Highly variable and varies for each (stack,OS) pair.

Existing time synchronization schemes (reviewed in the next section) focus on eliminating or accounting for these sources of error. Schemes typically differ due to differing assumptions in which sources of variation are dominant in different domains, and due to different approaches to eliminate the sources of error.

## III. Related Work

An important notion in time is that it has to be relative to a given reference standard. Lamport clarified the relationship between computer events and global reference time [18]. We focus on time synchronization to a reference value motivated by the need to relate computer sensed events to the outside world.

At the most fundamental level, there are just two schemes to synchronize clocks: Sender-Receiver (Figure 4) and Receiver-Receiver (Figure 5). All schemes operate within these two basic frameworks. In addition, some schemes synchronize against an external time reference, while others synchronize nodes to some arbitrary internal reference.

Network Time Protocol (NTP) is widely used in the Internet. It is distinguished by working well over paths with high latency and high variability [3]. The NTP protocol has a long-term, bi-directional exchange of time information to estimate both offset and skew. It incrementally adjusts the local clock frequency to align it with the reference time base. Unfortunately, NTP is a poor match for sensor networks for several reasons. First, it assumes communications are relatively inexpensive, while sensor networks are bandwidth and energy constrained. Second, it is designed for constant operation in the background at low rates. (At a maximum polling rate of 16 sec, NTP took around an hour to reduce error to about 70$\mu$s [19]). By comparison, TSHL exchanges number of broadcast beacons to compute skew and then perform one bidirectional exchange to compute a skew-corrected offset. In some sense, TSHL and NTP possess the same information, however TSHL reduces energy consumption by replacing long-term bidirectional communication with a smaller number of unidirectional, broadcast beacons. In addition, TSHL is not constrained by portability requirements and so can exploit MAC-level timestamping as TPSN does.

An interesting extension of NTP considers the Interplanetary Internet (IPin) [20]. The protocol iNTP, as proposed [20], assumes very high latencies but very predictable node position and movement (for example, predictable trajectories of satellites). While we expect the approximate locations of underwater nodes to be known with some accuracy, we expect ocean currents and environmental effects to render position information insufficiently reliable.

An alternate Internet based protocol was clock skew compensation for streaming audio in the Internet [21]. Faced with large and varying path delays, Fober demonstrates how to model the drift of between node clocks without modeling the offset. He uses statistical measures to remove the high jitter expected for their application. Although we could apply these techniques in underwater acoustic networks to remove this high jitter, but they can also be removed considerably in our point to point network through MAC layer time stamping.

The research closest to our work is time synchronization effort in the sensor networks community. Underwater sensor networks share many of the design goals of surface sensor networks. Energy conservation and longevity given a fixed power budget are common goals.

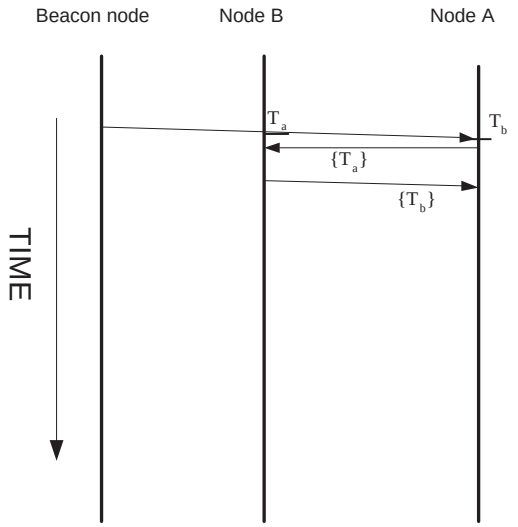


Fig. 5. Receiver-Receiver Synchronization

Reference Broadcast Synchronization (RBS) introduced receiver-receiver synchronization, completely eliminating transmitter side uncertainties as described in Section II-C [6]. RBS accounts for clock skew by modeling the clock with linear regression on multiple *Reference Broadcasts*. Its accuracy is quite good, about 6$\mu$s on IPAQ's with 802.11 cards. In addition, it introduces the concept of *post-facto synchronization*, allowing correction of clock errors after data collection rather than ahead of time. The RBS is not applicable to our environment because its central algorithm is built on the simultaneous reception of reference broadcasts at all nearby nodes. With underwater acoustics there are large variations in propagation time between nodes, resulting in synchronization error proportional to the propagation delay, up to 100ms or more.

Timing-sync Protocol for Sensor Networks (TPSN) exploited cross-layer optimizations, minimizing the sender and receiver side uncertainty by time stamping packets at the MAC [7]. Since TPSN uses a two-way exchange for synchronization, it can factor out most major sources of nondeterminism (the Send, Receive and Access time) and propagation delay. The TPSN authors report achieving accuracy of 8$\mu$s on Mica-2 motes. Like NTP, TPSN requires formation of a hierarchy, where each node synchronizes to its parents. However, it does not model skew of the local clock against the reference; instead it only computes offset. It therefore requires frequent resynchronization to correct for drift due to variation in clock skew, and it cannot do post-facto synchronization. The primary reason TPSN is not applicable to our environment is that it does not consider the effect of clock skew during message exchange. Although this effect is tiny during radio-based synchronization, it causes inaccuracies proportional to message propagation latency rises. We show in Section VI that at moderate distances $\sim$300m this error can be nearly 30% worse (see Figure 7).

Flooding Time Synchronization Protocol (FTSP) takes an additional step towards avoiding timestamp uncertainty by timestamping in the MAC and radio message layer in multiple places and using the average of these values to account for byte alignment jitter [8]. It uses a variation of sender-receiver synchronization. A sender broadcasts its global reference time to all receivers. Receivers use linear regression over multiple broadcasts to model their clock skew and offset. Compared to RBS, this approach reduces the number of message exchanges that need to take place for synchronization, since the message exchange is not between every pair of nodes. For underwater acoustic networks, however, FTSP suffers from the same problem as RBS, in that it assumes near instantaneous message propagation. This delay is large and variable in underwater acoustic networks.

## IV. Quantifying the Challenges of High Latency Links

Ideally we could simply use an existing time synchronization protocol in underwater acoustic networks. We next develop a simple analytic model to demonstrate why existing protocols such as RBS, TPSN, and FTSP do not work well for high-latency links. In this analysis we focus only on propagation latency as a source of error in time synchronization. Clearly this assumption is not correct in general, however, since existing schemes address other sources of error and are already accounted for by most existing protocols.

First we define a simple notation for realistic (inaccurate) clocks. For a node $S$, we model its uncorrected clock as $f_s(t)$ and its corrected time as $\widehat{f_S}(t)$:

$$\begin{aligned} f_S(t) &= a_S t + b_S \\ \widehat{f_S}(t) &= f_S(t) + \beta_S(t) \end{aligned} \quad (1)$$

These are first order linear function of its skew $a_S$ and offset $b_S$, where $t$ is the global reference time, and $\beta(t)$ is the correction factor calculated at $t$.

**Protocols based on one-way exchanges:** We first quantify the proportionality of error in both RBS and FTSP to the propagation delay by taking a simple case where a perfectly synchronized beacon node $f_B(t) = \widehat{f_B}(t) = t$ sends synchronizing pulses with its sending time to an uncorrected node $R$. Note that this is very similar to what happens in FTSP, and with slight modification can be applied to RBS as well.

With no propagation delay, the correction factor calculated by the synchronization protocol is:

$$\beta_R(t) = f_B(t) - f_R(t) = (1 - a_R)t - b_R$$

which will synchronize node $R$'s clock to the correct time, using Equation 1. The clock skew can be computed with multiple exchanges by observing how $\beta_R(t)$ changes.

However, given a propagation delay of $d$ between beacon and node $S$, the computed correction factor is:

$$\beta(t+d) = f_B(t) - f_S(t+d) = (1-a_S)t - (a_S d + b_S)$$

Based on Equation 1 alone, the corrected time is offset incorrectly by $d$. To correct for this error we need to estimate propagation delay. To stay within the RBS or FTSP models we must estimate this delay without sending additional messages. A prior computation of underwater propagation speed is quite difficult. Node locations may be known, but are likely inexact due to placement error or node movement. Underwater node localization is an open problem, and often requires synchronized clocks for computation. Finally, knowledge of acoustic propagation speed assumes that the nodes have capabilities to measure, at the very least, the temperature and pressure. For these reasons, a priori computation of delay seems quite challenging.

**Protocols based on two-way exchanges:** We next quantify the effect of the propagation delay on protocols that allow two-way exchanges, such as NTP and TPSN. The key element of these protocols are that they can factor out propagation delay via a two-way message exchange. However there is one assumption they make about sender's clock not skewing during the message exchange. Consider the sender $S$ and the receiver $B$ (perfect clock) exchanging timestamps as shown in Figure 4, with $T_1 = f_S(t_1), T_2 = f_B(t_1+d), T_3 = f_S(t_3)$, and $T_4 = f_S(t_3+d)$. With that scenario, the clock model for the sender S will be:

$$\begin{aligned} f_S(t) &= t + b_S \\ \beta_S(t) &= \frac{(T_2 - T_1) - (T_4 - T_3)}{2} \\ \widehat{f_S}(t) &= f_S(t) + \beta_S(t) \end{aligned} \quad (2)$$

With no skew, and with our assumption of none of the sources of error present, this will perfectly synchronize the nodes.

We will now show that if we consider skew it does have an effect on the error as either the skew or the duration for the message exchanges is increased (intuitively you can grasp this by looking at the message exchange shown in Figure 3).

$$\begin{aligned} f_S(t) &= at + b_S \\ \beta(t_3+d) &= \frac{(1-a_S)(t_1+t_3+d) - 2b_S}{2} \end{aligned} \quad (3)$$

The delay compensating clock at S as:

$$\widehat{f_S}(t_3+d) = f_S(t_3+d) + \beta(t_3+d)$$

Using Equation 4 below shows that this leads to an Error = $(t_3+d) - \widehat{f_S}(t_3+d)$, that is proportional to the skew and the packet exchange period.

$$\textit{Error} = \frac{(1-a_S)((t_3-t_1)+d)}{2} \quad (4)$$

For Berkeley motes, the upper bound given in the datasheet [15] is 40ppm i.e. a clock in mote can loose up to 40$\mu$s in a second. This can translate to a drift of around 15-20$\mu$s when the nodes are $\sim$400m apart. This significantly affects the accuracy of time synchronization, as predicted by Equation 4, and our simulation will show the degradation of TPSN precision with increasing distance, or skew, between nodes.

## V. Design of Time Synchronization for High Latency Channels(TSHL)

While prior time synchronization protocols addressed many sources of error in estimation, only NTP faced large propagation delay, and it is not appropriate for sensor networks (see Section III). We now present an alternate time synchronization algorithm for sensor networks that can manage high propagation delays while remaining energy efficient.

### A. Overview and Assumptions

TSHL is a two phase protocol. The the core idea is to *first* model the skew of a node's clock so that each node is *skew synchronized*. We compute skew by performing linear regression over multiple beacon values. After skew synchronization nodes may still operate with different offsets, but because all (one hop) nodes share a frequency standard they are now able to maintain an accurate relative timer for any additional events.

In the second phase we correct for clock offsets. Prior protocols such as NTP and TPSN did this with a two-way message exchange; we take this approach as well, but TSHL considers a *skew-compensated* two-way exchange.

When both phases have been completed we have a model mapping the local, inaccurate clock to the reference timebase. We can then compute a global time for all events, even those before our synchronization, with *post-facto* correction if necessary, similar to RBS.

In phase one, skew is estimated without any knowledge of propagation delay. The quality of our estimate of skew is dependent on the consistency, not the duration, of propagation delay. In our current approach we assume propagation delay is constant over the message exchange. Underwater, changes in temperature and pressure affect the speed of sound and so will change propagation delay over long durations. However, we expect a 20-beacon exchange to take a few seconds; over this period our assumption seems reasonable. Verifying this assumption is an area of future work, however if it does not hold we can fall back on a statistical model of skew as done by Fober et al. [21].

Beacon Nodes can either be specialized nodes with accurate clocks, perhaps connected to an external time reference like a tethered GPS receiver on a buoy. Alternatively, Beacons Nodes could be elected or externally designated.

As noted in Section II-C and first addressed TPSN [7], a great deal of non-determinism in message exchange can be

removed by placing message timestamping in the MAC layer. For highest accuracy we believe low-level timestamping is essential. We expect our acoustic modems to provide this bit- or byte-level radio access, making MAC-level timestamping easy to implement. Our current prototype implementation includes this feature (as discussed in Section VII).

A second assumption of TSHL is that clocks are *short-term stable*. Clock frequency and hence skew, must over a short period of time (typically 5-10 minutes) remain constant. Short term instability occurs mainly due to environmental factors such as sudden variation in temperature, supply voltage or shock [22]. This assumption allows us to model the clock skew using linear regression and use it for predicting the future time accurately as well. One typically accommodates long-term instability by periodic resynchronization (as described in RBS [6]); optimization of that interval is not part of our current research.

### B. Details

Figure 6 shows the message exchange in TSHL's two phases. In Phase 1 of the protocol, each node in the broadcast range of a Beacon Node models its clock skew. We define the Beacon Nodes clock as the reference timebase (i.e., $f_B(t) = \widehat{f_B}(t) = t$). The Beacon Node then sends out enough Beacon Messages for skew estimation. Our current simulations and implementation send 25, needed for reasonable linear regression. In the future we can make this value adaptive based on feedback from receivers; each receiver can estimate error and enter Phase 2 when skew error reaches a target threshold.

Each Beacon Message $BM_i$ contains the transmit timestamp $t_{B,i}$ obtained at the MAC level, just before the message left the Beacon Node. Each receiver $R$ gets this message at absolute time $t_{B,i} + D_{B\rightarrow R}$ where $D_{B\rightarrow R}$ represents the unknown propagation delay between the Beacon Node and the receiver. It then assigns it the local time $f_R(t_{B,i} + D_{B\rightarrow R})$. This local time includes error due to clock skew and offset in addition to propagation delay. However, as with Fober et al. [21], we can still model the drift of the local clock with respect to the Beacon's reference clock by doing a linear regression on the *difference* between receive timestamp and the timestamp in the message. This difference changes by the same amount as clock skew as they drift apart (provided our assumption that path delay is constant over our estimation interval).

Thus, for $N$ messages $M_i$, we do linear regression over the following data points:

$$(t_{B,i} - f_R(t_{B,i} + D_{B\rightarrow R}), f_R(t_{B,i} + D_{B\rightarrow R}) \quad (5)$$

This computation gives us skew correcting conversion of the local time, so from now each node is *skew synchronized* with its skew corrected local time represented as $f'_R(t)$ (as opposed to a fully synchronized clock $\widehat{f_R}(t)$). This skew correction allows us, in phase two, to do correct offset estimation and ultimately map prior or future events to the reference timebase.

Phase 2 is similar to the classical two-way synchronization exchange as shown in TPSN and Figure 4. TSHL's two-way exchange differs in that we correct for skew when computing the clock offset. When the receiver obtains enough beacons to estimate the skew it sends a Synchronization Request message with $T1 = f'_R(T1)$, the skew-corrected local timestamp. The Beacon Node records its local version of this $T2 = f_B(T1 + D_{R\rightarrow B})$ and returns this value to the receiver in a Synchronization Reply message timestamped at $T3$ who computes a skew-corrected receive time $T4 = f'_R(T3 + B \rightarrow R)$. Finally the receiver can compute its clock offset:

$$\begin{aligned} \mathit{offset}_R &= [(f_B(f'_R(T1) + D_{R\rightarrow B}) - f'_R(T1)) \\ &\quad -(f'_R(T3 + B \rightarrow R) - T3)]/2 \end{aligned} \quad (6)$$

When this exchange completes, the nodes are able to factor out the error that occurs because of skew and get the exact offset.

Our algorithm builds on prior synchronization algorithms, drawing the concept of skew modeling from RBS, and of MAC-layer time-stamping (to reduce jitter) from TPSN. To this prior work we add skew compensation used *during* the synchronization exchange. This addition is essential for high latency environment such as ours, as is demonstrated in the simulation results shown in the next section.

### C. Time Synchronization Over Multiple Hops

While these equations and the protocol are specific to synchronization between two hosts, we can generalize this result to multi-hop time synchronization. We briefly consider the effects of multiple hops on the protocol and accuracy.

Some optimizations are possible to the protocol when one considers multiple nodes synchronization with a single reference time. All nodes can share the same series of Phase 1 packets and independently estimate their skews. Since Phase 2 requires a two-way exchange, a straightforward adaptation of TSHL would employ as many such exchanges as each node has neighbors. Potentially one could optimize Phase 2 by bundling the exchange for several neighbors in a single reply. Such an optimization would require additional synchronization and increase delay, but since we model skew during the Phase 2 exchange, such delay should not greatly affect accuracy.

Time synchronization over multiple hops can be done either to a single, common time base, as is typical, or with multiple independent timebases as done in RBP [6]. Accuracy of the protocol over multiple hops should be similar to the accuracy of other multi-hop protocols for time synchronization; additional hops will degrade accuracy. RBS observed that per-hop jitter follows a Gaussian distribution, so multi-hop accuracy there degrades as the square-root of the number of hops [6]. An open area is to verify that our acoustic modems provide similar per-hop jitter [23].

## VI. Performance Evaluation of TSHL

In this section we present some preliminary results evaluating our time synchronization algorithm. We also describe the goals of our simulations, the methodology, comparison against

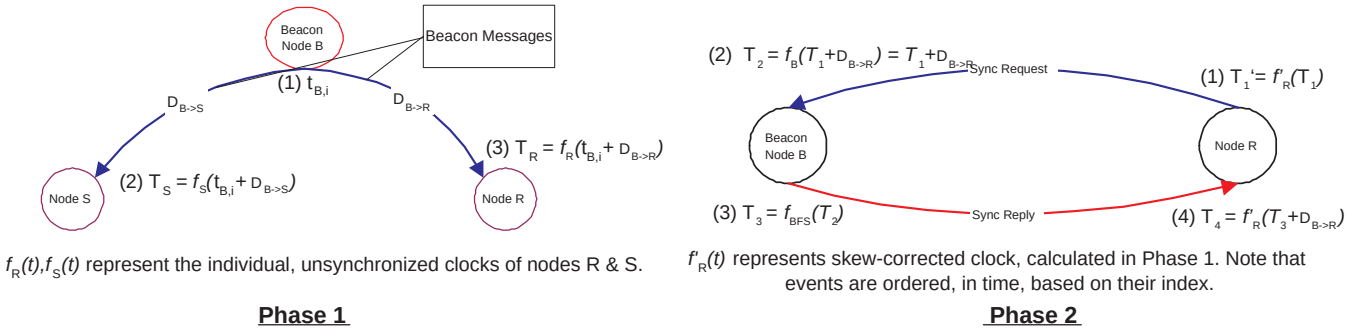


Fig. 6. Phases in TSHL

existing time synchronization protocols for sensor networks, and its dynamics under different parameters.

### A. Goals and Methodology

Our goals in conducting this evaluation of TSHL were two-fold:

- Compare performance of TSHL with other established sensor network time synchronization schemes such as TPSN and RBS.
- Understand the dynamics of TSHL under varying parameters such as node clock skew, granularity, and prediction error.

Although our early simulation results compared TSHL against TPSN, RBS, and FTSP. However, the results in this paper show comparisons only against TPSN. We omit RBS and FTSP results because those protocols do not consider propagation delay at all and so exhibit error proportional to propagation distance. This error is significant, even at short distances. RBS, for example, shows 6ms error at distances of 10m, and the error grows to 100ms and more at larger distances.

Overall, our simulations show that TSHL maintains low synchronization error, irrespective of the path delays, and its accuracy is directly proportional to the receive jitter (due to bit encoding/decoding or interrupt handling) or the granularity (smallest time increment possible) of the clocks used on the nodes themselves.

### B. Simulation Setup

We simulated the protocols in a custom event driven, packet level simulator designed for an acoustic underwater environment with high latency. Each node's clock was simulated as having some skew and offset relative to the global simulation time. We place a single Beacon Node, with no skew and zero offset in all simulations; all receiving nodes are within a 500m radial distance. We model encoding/decoding and interrupt handling errors referred in [8] by introducing a Gaussian receive jitter. Gaussian distributions have been found to provide reasonable approximations of this error, both by Elson and Estrin [6] above the MAC layer, and more recently by the authors of TPSN [7] even with MAC-layer timestamping.

The simulator allowed us to alter the following parameters in the simulations:

- Number of Beacon Messages broadcast by Beacon Nodes in phase 1 of the protocol.
- Individual clock skew and offset.
- Granularity of the clocks.
- Receive Jitter distribution.

Although we assume the environment was stable during the protocol design, as a worst case we vary water temperature uniformly between 25 and 35$^o$C for each packet sent by the Beacon Node. This temperature variation, with randomized receive jitter, accounts for any inaccuracy observed in the TSHL protocol. Comparison with TPSN was made by implementing a "TPSN-like" (TPSN-L) protocol in the simulator. By "TPSN-like" we mean that the protocol captured the essence of TPSN through MAC layer timestamping and offset correction scheme as described in their paper [7] and implemented in their nesC code.

In each of the simulation test performed the following parameters were used and are assumed to hold unless specifically mentioned otherwise:

- Skew = 40 ppm.
- Offset = 10$\mu$s.
- Number of Beacon Messages = 25.
- Granularity = 1$\mu$s.
- Receive Jitter = 15$\mu$s.

Each data point shown in a graph is the mean of 1000 simulation runs. Error bars show standard deviations.

### C. Comparative Evaluation

We now compare TSHL performance with TPSN-L, demonstrating that TPSN-L's accuracy deteriorates at high latencies because it did not need to model skew in its environment. In the first experiment we evaluate error as a function of the distance between the receiver and the Beacon node. To confirm that TPSN-L error is proportional to clock skew, our second second experiment examines offset error at a fixed distance as clock skew varies.

*1) Offset Error: Effect of Distance Variation:* In this test we measure the *absolute* offset error as a function of distance. (Offset error is the difference between the global time and

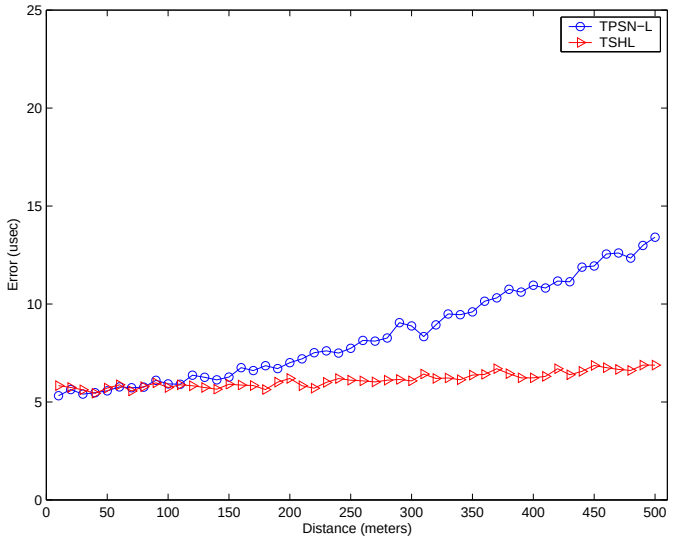


Fig. 7. Comparison of Instantaneous Error: Distance Variation

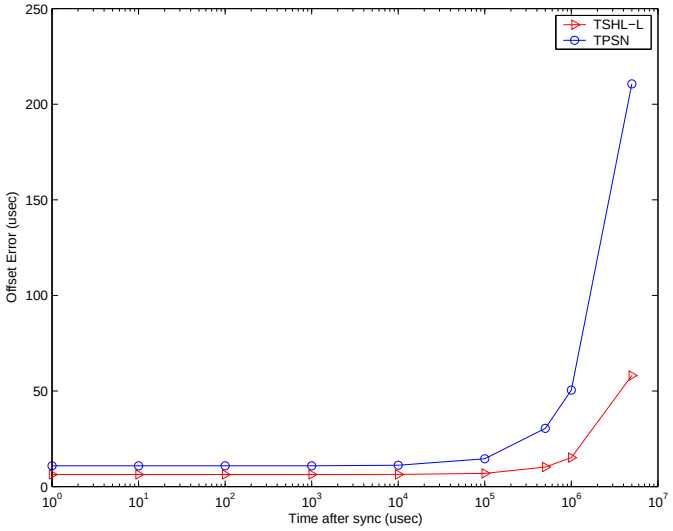


Fig. 8. Comparison of Error: Time since Sync

the *corrected* local time of the node.) First we measure it at the instant immediately proceeding the final synchronization exchange (TSHL phase 2).

We expect that the variation in distance (and thus the effect of clock skew) should cause deterioration of error for TPSN-L, but not for TSHL, as it specifically accounts for that.

Figure 7 confirms that the error increases with receiver distance for TPSN-L, from about the same accuracy as TSHL at distances less than 100m, to about double the error at 500m. Error does increase for TSHL as well, but by a much smaller amount; about 12% over 500m. We also observed, in other related experiments (not shown here for brevity), that the synchronization error of TSHL was varying linearly as the mean of the receive jitter distribution.

This simulation is presents error immediately after the synchronization ends, thus these results represent the best case performance for TPSN-like protocols that do not model the clock and consider skew. We consider that case next.

*2) Offset Error: Effect of Time since Synchronization:* Here we compare the performance of each of the scheme in predicting the time *after* a particular delay from the time the final synchronization exchange occurred. For this section we consider a receiver placed 400m from the Beacon Node.

Since TPSN-L does not model the skew and simply measures the offset at that time, we expect it to show a larger error as the time progresses (and the clock skew causes larger drift from the prior synchronized value). TSHL, on the other hand models the clock drift and offset, and hence would be able to correct the local clock more accurately in the future.

Figure 8 confirms our expectations. Error in both protocols is linear with time, but the slope of TSHL is much less because it models skew. Even after 5 seconds, with TSHL, error in offset is below 50$\mu$s.

Note that RBS and FTSP do model skew and would do well here. We do not show them because at 400m, their error due to propagation overwhelms the benefits of modeling skew.

(Note that these simulations assume a constant, random clock skew. Over these timescales, variation in clock skew is unlikely.)

*3) Effect of Variation of Skew:* In this experiment we vary the node skew with respect to the global (and the Beacon Node) clock, and observe the effect on the accuracy of the synchronization. The nodes are kept at a distance of 400m from the beacon node. We vary skew from 5 to 100ppm; this range is wider than typical for real clocks by a factor of about two.

Since TSHL models the skew in phase one, its should be able to cater for whatever the skew is. (Its error should be dominated by mostly by the non-determinism in the delay path i.e. receive jitter, not skew.) TPSN-L on the other hand, should show increased error as the skew (and thus the error over a single exchange) increases.

Figure 9 validates this expectation as over a wide range of clock skew Synchronization error remains nearly constant for TSHL. Again, this difference is because of TPSN-L's not modeling skew.

### D. TSHL Parameters

Having established the robustness of TSHL in a high latency acoustic channel, we try to analyze the dynamics of the protocol by altering some of the key parameters of the algorithm. Note that we have not shown results of varying the initial offset, since all protocols are able to factor this out consistently.

*1) Number of Beacon Messages:* In this simulation we analyze the effect of changing the number of Beacon Messages transmitted in phase 1. Since we do linear regression over these points to estimate clock skew we expect additional messages to result in higher accuracy.

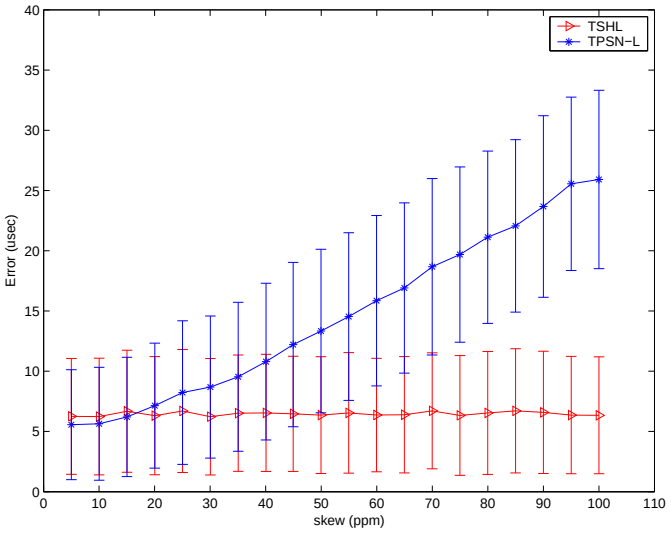


Fig. 9. Effect of clock skew

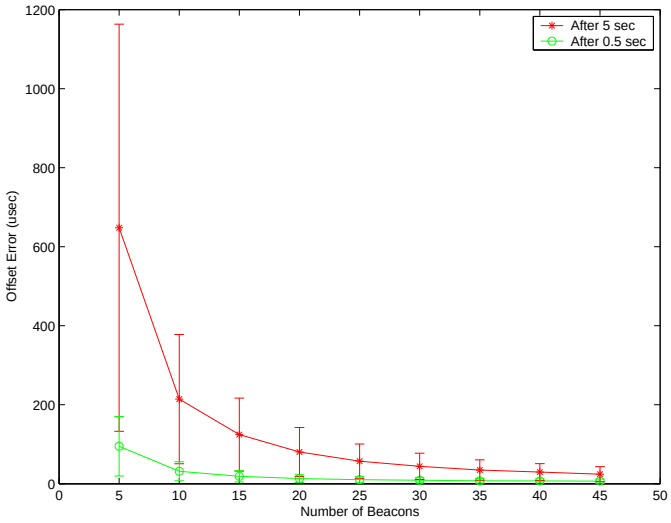


Fig. 10. Effect of Changing the Number of Sync Beacons

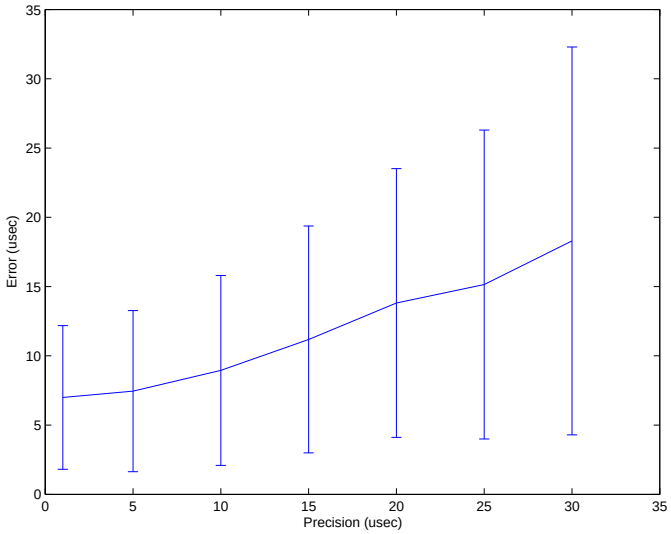


Fig. 11. Varying the Clock Granularity

In this simulation we set the receive jitter to 30$\mu$s and consider a receiver 400m from the Beacon node. We show two separate results: One where the error is calculated 0.5s after the final synchronization exchange, and in the other 5s after the exchange.

Figure 10 shows the results of these simulations. First, we observe that there is diminishing benefit in increasing the number of beacons after a certain optimal value. This value seems to be at around 25 beacons. These results corroborate very similar results for the group dispersion as a function of beacons in RBS [6]. In addition, we observe (but not shown here) that at shorter Receiver–Beacon Node distances, fewer beacons are needed to get equivalent accuracy.

A second interesting interpretation of this simulation can show the effect of control packet loss. As the number of beacons received decreases (in this interpretation, due to packet loss), we can observe the resulting deterioration in the synchronization accuracy. The loss of two way synchronization messages can be easily rectified by retransmission at the sender, if it does not get the sync reply.

*2) Variation of Clock Granularity:* In this section we observe the effect clock granularity has on the accuracy of the protocol. Since the clock granularity puts a fundamental limit on how accurate a clock can be, we can expect that the performance will degrade as clock granularity increases. While clock granularity is limited by hardware constraints, software controlled interrupts allow one to select coarser clock granularities to reduce interrupt rate and energy consumption, thus there may be a desire to vary this parameter.

For this test we placed the receiver at a distance of 100m from the Beacon Node node. We examine granularity at uniform increments of 5$\mu$s.

Figure 11 shows that both the mean error and the standard deviation of the error increases as the clock granularity degrades. However this is not a bad reflection on the protocol itself; it just goes to show the fundamental limits imposed by the granularity of a node clock on the level of synchrony that can be achieved.

## VII. Experimental Evaluation of TSHL

Analysis and simulation help demonstrate the importance of considering propagation delay in acoustic communication, but these approaches necessarily simplify the detail of the real world.

The main challenge in testing TSHL with high-latency acoustic communication is that our underwater acoustic modems are currently under development. A number of important design issues must be solved before short-range acoustic modems are available. We considered but rejected the alternative of substituting off-the-shelf, long-range acoustic modems. Not only do they have very different characteristics than our preliminary short-range design, but prior time synchronization

protocols have demonstrated the importance of integrating timestamping with the MAC layer [7]. Such integration is impossible with easily available packages.

Instead, we substituted in-the-air acoustic communication for underwater communication. Ultrasound and audible sound have been widely used for localization in surface sensor networks [24], [25], [26]; we adopted the Cricket platform due to its commercial availability and good support for low-level hardware access.

In-the-air acoustics changes several things. First and most importantly, the ratio of the speed of sound in air to water is about 1:5 ($\sim$300m/s vs. $\sim$1500m/s).

This scaling factor plays to our advantage in that shorter distances appropriate for in-office testing scale to much longer equivalent underwater distances. Second, in-the-air acoustics capture some of the noise and random variation of the real world. Although we make no claims that different air temperatures and multipath effects of an office accurately model underwater currents and propagation, it does capture some unpredictable variation.

Our initial choice of the Cricket platform was promising since it had low level support for acoustic transmission and detection. However, the accuracy provided by in its detection hardware proved to be insufficient for the microsecond-level accuracy that we wanted in our testbed. For detailed discussion please refer to the appendix in section X. [2]

We are currently in the process of developing our own acoustic modems and will perform system-level experiments as they become available.

## VIII. Conclusions and Future Work

In this paper we have presented *Time Synchronization for High Latency*, a time synchronization protocol for high point-to-point latency environment. As short-range, underwater sensor networks are developed we expect protocols that consider high-latency communication to be increasingly important. TSHL represents a first such protocol in this domain, and potentially has application even to other high-latency domains such as interplanetary networks.

Through analysis and simulation we explored TSHL for a wide range of characteristics. Prior protocols like RBS and FTSP exploit the rapid propagation of RF; in an high-latency acoustic network they perform quite poorly (about 6ms error at 10m, growing to hundreds of milliseconds at our target maximum range of 500m). By considering propagation latency, TPSN does much better. However, we demonstrated that it is critical to consider skew as well, even during the synchronization exchange. At short ranges TPSN and TSHL are equivalent in terms of accuracy, but at long ranges TSHL shows up to two times better accuracy (compared to TPSN-L). An implementation of TSHL in an in-the-air acoustic network is underway.

[2]This appendix was added to the original technical report ISI-TR-2005-602 in December 2005

## IX. Acknowledgments

We would like to acknowledge the help extended by Wei Ye, and Jack Wills in developing the concepts essential to underwater acoustic communication, particularly in Section II-A. Also we would like to thank Saurabh Ganeriwal, Jeremy Elson, and Brano Kusy, all authors of previous time synchronization schemes, for extending courteous help in correctly understanding their protocols. The Cricket team at MIT was extremely helpful. We would especially like to thank David Moore whose insight and helpful tips allowed us to fine tune the ultrasound transducers. We would also like to thank all the anonymous reviewers for their comments that helped us in improving this paper. Finally Qasim Chaudhari helped in correcting some of the aspects of the simulations.

This research is partially supported by the National Science Foundation (NSF) under grant number NeTS-NOSS-0435517, "Sensor Networks for Undersea Seismic Experimentation", and by a hardware donation from Intel Corporation. It is also partly supported by CiSoft (Center for Interactive Smart Oilfield Technologies), a Center of Research Excellence and Academic Training and a joint venture between the University of Southern California and ChevronTexaco.

## References

[1] P. Bonnet, J. E. Gehrke, and P. Seshadri, "Querying the physical world." in *IEEE Personal Communications*, 2000.

[2] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications Magazine*, pp. 33–38, Sept. 1994. [Online]. Available: http://nii.isi.edu/publications/kerberos-neuman-tso.html

[3] D. Mills, "Internet time synchronization: the network time protocol; RFC 1129," *Internet Request for Comments*, no. 1129, Oct. 1989.

[4] H. Wang, L. Yip, D. Maniezzo, J. Chen, R. Hudson, J. Elson, and K. Yao, "A wireless time-synchronized COTS sensor platform part II–applications to beamforming," in *Proceedings of IEEE CAS Workshop on Wireless Communications and Networking, September*, Pasadena, CA, USA, September 2002. [Online]. Available: citeseer.ist.psu.edu/article/wang02wireless.html

[5] W. Chen, J. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," in *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03)*, Atlanta, Georgia, USA, November 4-7 2003, pp. 258–271.

[6] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA, USA, December 2002, pp. 147–163.

[7] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the First International ACM Conference on Embedded Networked Sensor Systems (SenSys)*. Los Angeles, California, USA: ACM Press, 2003, pp. 138–149.

[8] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *Proceedings of the Second International ACM Conference on Embedded Networked Sensor Systems (SenSys)*. Baltimore, MD, USA: ACM Press, 2004, pp. 39–49.

[9] J. van Greunen and J. Rabaey, "Lightweight time synchronization for sensor networks," in *Proceedings of the Second ACM international conference on Wireless sensor networks and applications*. San Diego, CA, USA: ACM Press, 2003, pp. 11–19.

[10] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks Journal*, pp. 257–279, march 2005. [Online]. Available: http://www.ece.gatech.edu/research/labs/bwn/underwater.pdf

[11] B. Zhang, G. S. Sukhatme, and A. A. G. Requicha, "Adaptive sampling for marine microorganism monitoring," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2. IEEE, 2004, pp. 1115–1122. [Online]. Available: http://cres.usc.edu/pubdb_html/files_upload/402.pdf
[12] R. Urick, *Principles of Underwater Sound.* McGraw-Hill Book Company, 1991.
[13] M. Stojanovic, "Underwater acoustic communication," *Wiley Encyclopedia of Electrical and Electronics Engineering*, pp. 688–698, 1998.
[14] ——, *Wiley Encyclopedia of Telecommunications.* John Wiley & Sons, Inc., 2003, ch. Acoustic (underwater) Communications. [Online]. Available: http://www.mit.edu/people/millitsa/resources/pdfs/ency2.pdf
[15] Atmel, "http://www.atmel.com/."
[16] H. Kopetz and W. Schwabl, "Global time in distributed real-time systems," Technische Universitat Wien," Technical Report 15/89, 1989.
[17] M. Horauer, U. Schmid, K. Schossmaier, R. Höller, and N. Kerö, "PSynUTC—evaluation of a high precision time synchronization prototype system for ethernet lans," in *Proceedings of 34th Annual Precise Time and Time Interval Meeting (PTTI)*, December 2002, pp. 263–279.
[18] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, 7, pp. 558–564, 1978.
[19] J. Elson, "Time synchronization in wireless sensor networks," Ph.D. dissertation, UCLA, 2003.
[20] D. Mills, "http://www.eecis.udel.edu/~mills/ipin.html."
[21] D. Fober, Y. Orlarey, and S. Letz, "Clock skew compensation over a high latency network," in *Proceedings of the ICMC*. ICMA, 2002.
[22] J. R. Vig, "Introduction to quartz frequency standards," Army Research Laboratory, Electronics and Power Sources Directorate, Tech. Rep. SLCET-TR-92-1, 1992.
[23] A. Syed and J. Heidemann, "Time synchronization for high latency acoustic networks," USC/Information Sciences Institute, Tech. Rep. ISI-TR-2005-602, April 2005, extended tech report version of the INFOCOM paper with a detailed appendix. [Online]. Available: http://www.isi.edu/~johnh/PAPERS/Syed05a.html
[24] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking*. Boston, MA, USA: ACM, Aug. 2000, pp. 32–43. [Online]. Available: http://nms.lcs.mit.edu/papers/cricket.pdf
[25] A. Savvides, C.-C. Han, and M. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proceedings of the ACM International Conference on Mobile Computing and Networking*. Rome, Italy: ACM, July 2001, pp. 166–179. [Online]. Available: http://www.acm.org/pubs/articles/proceedings/comm/381677/p166-savvides/p166-savvides.pdf
[26] L. Girod, V. Bychkovskiy, J. Elson, and D. Estrin, "Locating tiny sensors in time and space: A case study," in *ICCD '02: Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02)*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 214–219, invited paper. [Online]. Available: http://lecs.cs.ucla.edu/Publications/papers/iccd-2002.pdf
[27] M. Electronics, "http://www.mouser.com/."

## X. Appendix: Experiences with TSHL on the Cricket Platform

As mentioned earlier in section VII, we want to emulate underwater acoustic transmission with in-the-air ultrasound transmission. We initially chose the Cricket platform due to its commercial availability and good support for low-level hardware access. We summarize the performance implications of in-the-air acoustics compared to underwater acoustics in Section VII. This section summarizes our experiences with this platform for time synchronization.

### A. Sending data via hybrid RF and Ultrasound

Our choice of the Cricket platform forced us to approximate sending data over ultrasound. The Cricket hardware consists of a Mica-2-like mote core with Chipcon radio, and it adds a 40kHz ultrasound-based transmitter and microphone [27]. While this platform supports sending and receiving acoustic pulses, it currently does not support data transmission. We therefore approximate the acoustic channel by sending an acoustic pulse and a coupled data packet over the standard Chipcon radio. Although the data packet arrives quickly, we delay reception of it until the acoustic pulse arrives, experimentally accounting for propagation latency, and we add a computed transmission delay, accounting for bandwidth limitations. While not ideal, we were satisfied with this approximation.

### B. Jitter in hybrid (RF-Ultrasound) time stamping.

After implementing TSHL in TinyOS and building a hybrid RF-Ultrasound emulation module (using the ultrasound control provided by the MIT Cricket group), we observed significant inaccuracy in synchronization (in 100's of $\mu$s). This necessitated a review of the hardware capabilities and any jitter that it might accrue. We identify several sources of uncertainty in timestamping due to the hybrid emulation module, as illustrated in Figure 12. All of these sources of uncertainty are below the MAC layer, and are endemic to the communication hardware employed.

To verify these sources of jitter we test timing at the hardware-level. We attach an oscilloscope to the input of ultrasound transmitter (output of C54) and the amplified received signal (input to D7) for actual signal timing. For software timing we toggle a GPIO (AD0) pin. These experiments allow us to measure the jitter/variance at each of these delays. The sources of delay and our measurement results are as follows:

**Ultrasound Transmission Delay:** Since we time stamp an outgoing Radio packet and then initiate a software procedure to pulse the ultrasound transmitter, the accuracy of time stamp depends on the delay till when the pulse *actually* starts going out. This can vary due to code size, interrupts and the resonance delay of transmit circuitry.

Our measurements showed that this was constant (less than $6\mu$s) and this value can always be determined, if needed.

**Reception Notification:** This delay occurs between the start of the received pulse till interrupt handler for pulse detection gets called (where we timestamp the reception time). The delay could vary due to existing interrupt handlers or interrupts between the execution of our code.

Typically the interrupt handler was called at regular periods. measurements, however, showed that this infrequently (1 in 50) did change by 20–50$\mu$s. Although this did not account for the variance observed in our experiments, we decided to remove it as a possible source of error by using the value of a hardware counter, $T_C$, at the input capture event for that counter (ICP pin is wired to the ultrasound detection circuitry). The counter was started when the correlated radio packet's start symbol was received, and at that instant we also took a local timestamp $T_L$. We were able to circumvent any interrupt related inaccuracies by using the receive time stamp as $T_L + T_C$. This is similar to the mechanism for measuring time-of-flight used in the localization of Crickets at MIT.

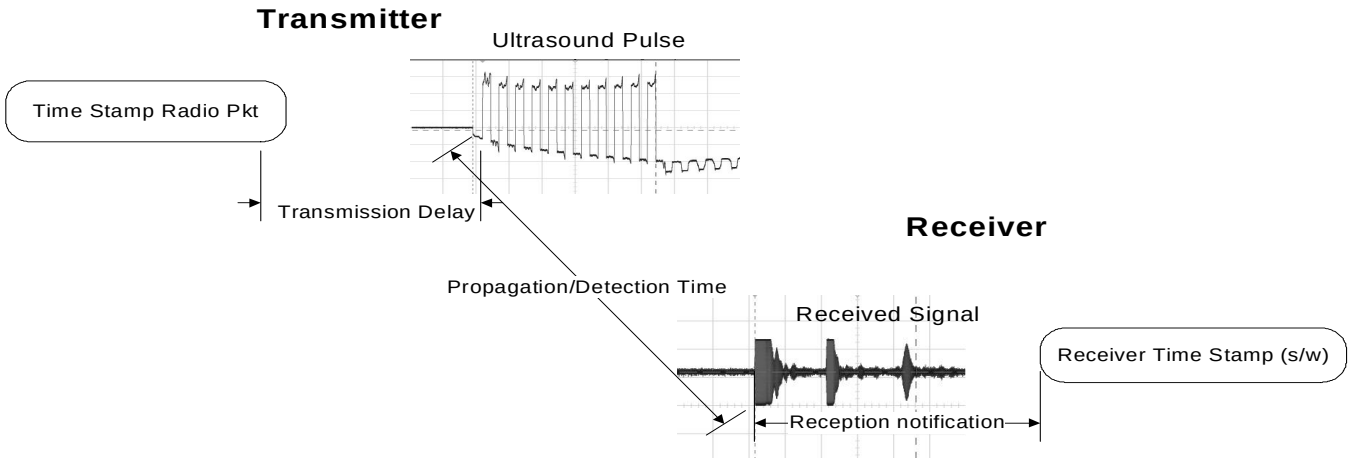


Fig. 12. A Blow up of the sources of uncertainty we identified specifically for the Cricket platform used in our hybrid RF-Ultrasound testbed. Note that these are below MAC level, above which we have factored out by timestamping after MAC access.

TABLE II
TRANSMIT TO DETECTION DELAY JITTER FOR FIXED NODE POSITIONS.

| Detection Time delay (ms) | Jitter from mean (ms) |
|---|---|
| 2.246 | -0.0674 |
| 2.364 | 0.0506 |
| 2.280 | -0.0334 |
| 2.318 | 0.0046 |
| 2.359 | 0.0456 |

Mean = 2.3134 ms, Range = 0.113 ms

**Detection Time:** The above results convinced us that most of the uncertainty exists in the Detection time (see Fig.12). This jitter is determined by the point in time when, relative to the start of the transmit pulse, the start of the receive pulse is detected. We measured the jitter by creating a positive edge on a GPIO pin at the transmitter when it sends a command to transmit ultrasound pulse, and used this as an edge trigger. At the same time we hooked the second line to the ICP1 pin (the pin that triggers an input capture on rising edge), which is the absolute best timing of ultrasound pulse detection that the Cricket system can provide. The resulting range of detection times, for the same node locations, with a jitter range of about 130$\mu$s, is shown in Table II.

In similar experiments that we conducted, we observe that the detection instant also shows correlation to environmental change.

Furthermore, multipath effects were evident even at short ranges with this simple acoustic hardware. Another interesting observation is that at larger inter-node distances, the direct path is usually weaker than the first reflection. One possible reason for this "bad" performance might be that the transducers have very narrow bandwidth (1KHz at 3dB) and thus large (about 1 ms) response time; unless we send pulses for longer duration, the pulse will never reach maximum amplitude and we will have a hard time detecting the exact start of the pulse. An open issue is to verify that this bandwidth limitation is actually the core problem at hand and address that in our ongoing hardware implementation. The above arguments also suggest that low-level jitter compensation would be beneficial; we could use mechanisms such as described by the FTSP [8].

We accept that this is not an exhaustive study of the Cricket ultrasounders. Since they were not designed for data transmission, the problems we encountered are not surprising. Our goal here is to understand the limitations of the hardware for time synchronization, shed light on our experiments, and to identify important characteristics for future hardware designs.

### C. Short range of ultrasounders

We also wanted to vary the internode distance to observe the effect of latency. Our experimental results show benefit at distances larger than 100m, with substantial (100%) improvement at around 500m underwater. In the main text we argued (see VII) that slower speed of sound in air allowed us to relax this range requirement by a factor of five. However this still amounts to at least a range of around 20–100m to show substantial benefits. The original ultrasound module, with no tweaking, provides a range of around 10-15m.

The Cricket architecture provided two ways of extending the range. One way was through increasing the receiver sensitivity; the other by increasing the number of pulse sent per ultrasound transmission. Both have the intuitive drawback that they will result in less accurate time stamping. Increasing sensitivity results in becoming less tolerant of noise; increasing the number of pulses makes it more difficult to correlate the start of the pulse with when a detection event gets triggered (this effect will possibly be more pronounced at larger distances or with higher multipath). We have been able to extend the detection range of Cricket platform to around 100m in the air (by sending 40 pulses at 40kHz), but the resulting increase in detection jitter belies its utility.

### D. Summary

Our simulations showed that TSHL will show a performance improvement in the sub 100$\mu$s and $>$100m regime. We are

currently developing our own acoustic communication hardware, using which we plan to perform in-the-air experiments to validate TSHL. We expect that, based on the above insights, our platform will address the problems encountered on the Cricket platform.