Time Synchronization Services for Wireless Sensor Networks

Dissertation Proposal

Jeremy Elson
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA, 90095
jelson@cs.ucla.edu

April 10, 2001

Abstract

Time synchronization is a critical piece of infrastructure for any distributed system. Distributed, wireless sensor networks make extensive use of synchronized time in many contexts—for example, in forming TDMA schedules, integrating a time-series of proximity detections into a velocity estimate, in detecting redundant detections of a phenomenon from multiple sensors, and in distributed beamforming systems. The variety of uses leads to highly varied and non-standard requirements in the scope, lifetime, and maximum error of the synchronization achieved, as well as the time and energy required to achieve it. Existing time synchronization methods were not created with wireless sensor networks in mind, and need to be extended or redesigned to meet the new requirements and constraints.

Our proposed work centers around the development of new time synchronization methods, their characterization according to metrics relevant to wireless sensor networks, and a proof-of-concept implementation of an application that makes use of our synchronization primitives. In this proposal, we first describe a number of metrics that we have found useful for the characterization of time synchronization services. Using these metrics, we describe existing services, and compare them to the synchronization requirements of future sensor networks. We also present an implementation of our own low-power synchronization scheme, *post-facto synchronization*, and describe an experiment that characterizes its performance for creating short-lived and localized but high-precision synchronization using very little energy. Finally, we describe our work plan.

Contents

1. Introduction **1**
- 1.1. Wireless Sensor Networks . . . 1
- 1.2. Time Synchronization . . . 1

2. Metrics and Terminology **2**

3. Motivations **3**
- 3.1. Distributed beam-forming and sensor fusion . . . 3
- 3.2. Multi-sensor integration . . . 4
- 3.3. In network data aggregation and duplicate suppression . . . 5
- 3.4. Energy-efficient radio scheduling . . . 5
- 3.5. Uses common in traditional distributed systems . . . 6

4. Related Work **6**
- 4.1. Distributed Clocks and Network Time Protocols . . . 6
- 4.2. National Time Standards and Time Transfer . . . 7
- 4.3. Methods for avoiding the problem . . . 8

5. Sensor Network Time **9**
- 5.1. The Need for Something New . . . 9
- 5.2. Design Principles . . . 10
  - 5.2.1 . Multi-modal synchronization . . . 10
  - 5.2.2 . Tiered architectures . . . 10
- 5.3. Our Approach . . . 11

6. Post-Facto Synchronization **11**
- 6.1. Expected Sources of Error . . . 12
- 6.2. Empirical Study . . . 13
- 6.3. Discussion . . . 14

7. Work Plan **15**
- 7.1. Development and characterization of time sync methods . . . 15
- 7.2. Testbed implementation as proof-of-concept . . . 16
- 7.3. Exploration of methods for tuning and automatic modality selection . . . 17

8. Conclusions **18**

Bibliography **18**

1. Introduction

1.1. Wireless Sensor Networks

Recent advances in miniaturization and low-cost, low-power design have led to active research in large-scale, highly distributed systems of small, wireless, low-power, unattended sensors and actuators [ADL$^+$98, KKP99, EGHK99]. The vision of many researchers is to create sensor-rich "smart environments" through planned or ad-hoc deployment of thousands of sensors, each with a short-range wireless communications channel, and capable of detecting ambient conditions such as temperature, movement, sound, light, or the presence of certain objects.

While this new class of distributed systems has the potential to enable a wide range of applications, it also poses serious design challenges, described more fully by KKP [KKP99] and EGHK [EGHK99]. The sheer number of these devices makes global broadcasting undesirable; wireless nodes' limited communication range relative to the geographic area spanned by the system often makes global broadcasting so inefficient that it is infeasible. As a consequence, many argue that nodes must depend on localized algorithms—making control decisions based solely on interactions with neighbors, without global system knowledge [EGHK99, IGE00].

Another important feature that distinguishes sensor networks from traditional distributed systems is their need for *energy efficiency*. Many nodes in the emerging sensor systems will be *untethered*, having only finite energy reserves from a battery. Unlike laptops or other handheld devices that enjoy constant attention and maintenance by humans, the scale of a sensor net's deployment will make replenishment of these reserves impossible. This requirement pervades all aspects of the system's design. For example, nodes must remain off or in a low-power state as often as possible; when on, they must maximize the usefulness of every bit transmitted or received [PK00].

1.2. Time Synchronization

Time synchronization is a critical piece of infrastructure for any distributed system. Distributed, wireless sensor networks make particularly extensive use of synchronized time: for example, to integrate a time-series of proximity detections into a velocity estimate [CEE$^+$01]; to measure the time-of-flight of sound for localizing its source [Gir00]; to distribute a beamforming array [YHR$^+$98]; or to suppress redundant messages by recognizing that they describe duplicate detections of the same event by different sensors [IGE00]. Sensor networks also have many of the same requirements as traditional distributed systems: accurate timestamps are often needed in cryptographic schemes, to coordinate events scheduled in the future, for ordering logged events during system debugging, and so forth.

The broad nature of sensor network applications leads to timing requirements whose scope, lifetime, and maximum error differ from traditional systems. In addition, many nodes in the emerging sensor systems will be untethered and therefore have small energy reserves. All communication—even passive listening—will have a significant effect on those reserves. Time synchronization methods for sensor networks must therefore also be mindful of the time and energy that they consume.

In this paper, we argue that the heterogeneity of requirements across sensor network applications, the need for energy-efficiency and other constraints not found in conventional distributed systems, and even the variety of hardware on which sensor networks will be deployed, make current synchronization schemes inadequate to the task. **In sensor networks, existing time synchronization methods will need to be extended and combined in new ways in order to provide service that meets the needs of applications with the minimum possible energy expenditures.** The development of these new methods is the core of our proposal.

We will also present our idea for *post-facto synchronization*, an extremely low-power method of synchronizing clocks in a local area when accurate timestamps are needed for specific events. We also present an experiment that suggests this multi-modal scheme is capable of achieving a maximum error on the order of 1μsec—an order of magnitude better than either of the two modes of which it is composed. These results are encouraging, although still preliminary and performed under idealized laboratory conditions. We believe that our pilot study lends weight to our methods and that further investigation is warranted.

We begin our discussion in Section 2, describing a number of metrics that can be used to classify both the types of service provided by synchronization methods and the requirements of applications that use those methods. In Section 3, we justify in more detail the need for synchronized time in sensor networks. Related work is reviewed in Section 4. Section 5 argues why the existing work in the field is insufficient for use in the new sensor network regime, and outlines our proposed contributions. Section 6 describes our post-facto synchronization technique and presents an experiment that characterizes its performance. Our proposed thesis work plan is described in detail in Section 7, and conclusions are drawn in Section 8.

2. Metrics and Terminology

Before starting our discussion in earnest, we will first define a set of metrics that we have found useful for characterizing time synchronization in sensor networks. In studying both methods and applications, we have found five metrics to be especially important:

- *Maximum Error*—either the dispersion among a group of peers, or maximum deviation of the members from an external standard.
- *Lifetime*—which can range from persistent synchronization that lasts as long as the network operates, to nearly instantaneous (useful, for example, if nodes want to compare the detection time of a single event).
- *Scope and Availability*—the geographic span of nodes that are synchronized, and completeness of coverage within that region.
- *Efficiency*—the time and energy expenditures needed to achieve synchronization.
- *Cost and Form Factor*—which can become particularly important in wireless sensor networks that involve thousands of tiny, disposable sensor nodes.

The services provided by different time synchronization methods fall into many disparate points in this parameter space. All of them make tradeoffs—no single method is optimal along all axes.

To illustrate the use of these metrics, consider the time service provided by the U.S. Global Positioning System [Kap96] (described in more detail in Section 4). Consumer GPS receivers can synchronize nodes to a persistent-lifetime time standard that is Earth-wide in scope within a maximum of 200ns [MKMT99]. However, GPS units often can not be used (e.g., inside structures, underwater, during Mars exploration), can require several minutes of settling time. In some cases, GPS units might also be large, high-power and expensive compared to small sensors.

In contrast, consider a small group of nodes with short-range, low-power radios. If one node transmits a signal, the others can use that signal as a time reference—for example, to compare the times at which they recorded a sound. The synchronization provided by this simple "pulse" is local in scope and its error comes primarily from variable delays on the radio receivers and propagation delay of the radio waves. For a given error bound, the lifetime of the synchronization is also finite as the nodes' clocks will wander after the initial pulse. However, the pulse is fast and energy-efficient because it only requires the transmission of a single signal.

3. Motivations

Why do sensor nets need synchronized time?

Before discussing time synchronization *methods*, we must first clarify its *motivations*. In this section, we will describe some common uses of synchronized time in sensor networks. The wide net cast by the disparate application requirements is important; we will argue later that it precludes the use of any *single* sync method for all applications. Indeed, while the related work in this field (reviewed in Section 4) is extensive, it often relies on assumptions that are violated in the new sensor network regime.

3.1. Distributed beam-forming and sensor fusion

In recent years, interest has grown in signal processing techniques such as sensor fusion and beam-forming. These are techniques used to combine the inputs of multiple sensors, sometimes using heterogeneous modalities, in applications such as noise reduction [YHR$^+$98], target tracking, and process control. This kind of DSP will serve as an important basis for sensor networks, but much of the extensive prior art in the field assumes *centralized sensor fusion*. That is, even if the sensors gathering data are distributed, sensor data is often assumed to be consolidated at one site before processing. However, centralized processing makes use of *implicit* time synchronization; synchronization that must be made *explicit* to create a fully distributed system.

For example, consider a beamforming array designed to localize the source of sound, such as that described by YHRCL in [YHR$^+$98]. The array computes phase differences of signals received by sensors at different locations. From these phase differences, the processor can infer the time of flight of the sound from its source to each sensor. This allows the sound's source to be localized with respect to the spatial reference frame defined by the sensors in the array. However, this makes the implicit assumption that the sensors themselves are synchronized *in time*, as well. In other words, the beam-forming computation assumes that the observed phase differences are due to differences in the time of flight from the sound source to the sensor, and not variable delays incurred in transmitting signals from the sensor to the processor. In a centralized system, where there is tight coupling from sensors to a single central processor, this is a valid assumption; the sensor data share an implicitly synchronized time-base by virtue of the fact that the audio data are all fed to the same processor. However, for such an array to be implemented on a fully distributed set of autonomous wireless sensors, *explicit* time synchronization of the sensors is needed.

It is important to note that a beam-forming array actually contains two separate (but related) time-synchronization problems. Specifically, to measure the time-of-flight from the sound's source to the receiving sensors, some form of synchronization must exist between the sender and receiver. In other words, the array needs to know the time of emission relative to the time of detection in order to measure time-of-flight. This latter problem has traditionally been solved with "over-sampling": treating the clock bias between the emitter and receivers as an extra unknown in the system of localization equations, and adding an extra sensor measurement to balance the extra unknown. This works only if the receivers are synchronized *with each other*; that is, if there is only a *single* clock bias between the sender and any receiver. Therefore, the need for explicit receiver synchronization as described earlier is not obviated by having extra sensors. Without correlated receivers, each additional sensor brings with it both a measurement and its own unknown clock bias—i.e., adding both an equation *and* an unknown to the system.

3.2. Multi-sensor integration

A common theme in sensor network design is that of *multi-sensor integration*—combining information gleaned from multiple sensors into a larger world-view not detectable by any single sensor alone. Unlike the previous examples, in which sensor fusion is done at the signal processing level, sensor integration focuses on algorithmically combining higher-level knowledge.

For example, consider a group of nodes that know their geographic positions and can each detect proximity to some static phenomenon P. (The detection might involve localization as described in the previous section.) Alone, a single sensor can tell that it is near P. However, by integrating their knowledge, the joint network can describe more than just a set of locations covered by P; it can also compute P's *size*. In some sense, the whole of information has become greater than the sum of the parts.

This type of emergent behavior does not always require synchronized time. If P is static, as in the previous example, time sync may not be needed at all. However, what if P is moving, and the objective is to report P's speed and direction? At first glance, it might seem that the object localization system we described above can be easily converted into a motion tracking system, simply by performing repeated queries of the object tracking system over time. If the data indicating the location of our tracked phenomenon P all arrives at the same processor for integration, perhaps no synchronization across nodes is required. The integrator can simply timestamp each localization reading as it arrives and will then have all the information required to integrate the series into motion data. In this case, it seems that the time synchronization problem has been avoided entirely.

This scheme has serious limitations discussed below, but it does work in some contexts. In particular, if the tracked object is moving very slowly relative to the delay between the sensors and the integration point, our brute force approach might work. For example, imagine an asset tracking system capable of locating specific pieces of equipment. If we wish to define the "motion" of an object as the history of all people who have used it, an equipment motion tracker might be designed very simply. We can merely ask the object tracker for the equipment's location several times a day, and compile a list over time of offices in which the equipment has been located. This works because we assume someone who uses equipment does so over the course of days or weeks – extremely slowly on the timescale over which the object-tracker can locate an object and report its location to the user or a data integrator.

This method has serious limitations in contexts where the timing requirements are more critical than in our equipment example. If the tracked phenomenon moves quickly, many factors that were insignificant in equipment tracking become overwhelming. For example, consider the situation likely in wireless sensor networks: a spatially distributed group of sensors, each capable of communication over a very short range communication relative to the total geographic area of sensor coverage. Information can only travel through this network hop-by-hop; therefore, the latency of messages from any given sensor to a central integrator will vary with the distance from the sensor to the integration point. In this situation, the brute force approach may fail.

The reason for the failure of the brute force approach is instructive to consider. The simple equipment tracker essentially assumed that the travel time of messages from the equipment sensors back to the integration point was zero: we ask the question "Where is this object?" and receive a reply that we assume is instantaneous and still correct when it is received. In the case of tracking equipment, this is probably a valid assumption because a specific piece of equipment is unlikely to move on the scale of time required to propagate a message through the sensor network. However, this assumption breaks down for faster-moving phenomena. Using the brute force centralized approach, it will be impossible to accurately track the motion of any phenomenon moving at speeds that approach the time scale of the networks' round trip time.

There are additional motivations for doing localized and distributed detection. A centralized aggregation point is not scalable and is prone to failures. In addition, in sensor networks where energy-efficiency is critical, it is unwise to design a system where a large volume of messages must be routed through many power-consuming nodes. A system that transmits each individual location reading through every node on the path from the phenomenon back to the integration point will have a high energy cost.

These limitations suggest that sensor readings should be time-stamped *inside of the network*, as near as possible to the original sensor event. Doing so dramatically reduces the variable delay introduced by message transmission latencies. Timestamping inside the network also allows tracking data to be post-processed into motion data, aggregated, and summarized inside of the network, thus requiring a greatly reduced number of bits to travel back to the user. All of these advantages, however, do come with a price: *sensors in the network must share a common time base in order to ensure the consistency of readings taken at multiple sensors*.

In a motion tracking application, the allowable synchronization error in nodes' clocks is informed by factors such as the speed of the target relative to sensor density and detection range. It is also affected by the system's desired spatial precision and detection frequency. The tighter time synchronization is achieved, the greater precision is possible in the tracking of motion by a collection of proximity detectors. Very slow moving objects may be tracked sufficiently by nodes with loosely synchronized clocks, but tighter and tighter synchronization is required if we wish to track faster and faster objects—or perhaps even phenomena such as wave-fronts.

3.3. In network data aggregation and duplicate suppression

A feature common to sensor networks due to the high energy cost of communication compared to computation [PK00] is local processing, summarization, and aggregation of data in order to minimize the size and frequency of transmissions. Suppression of duplicate notifications of the same event from a group of nearby sensors can result in significant energy savings [IGE00]. To recognize duplicates, events must be timestamped with a precision on the same order as the event frequency; this might only be tens or hundreds of milliseconds. Since the data may be sent a long way through the network and even cached by many of the intermediate nodes, the synchronization must be broad in scope and long in lifetime—perhaps even persistent.

3.4. Energy-efficient radio scheduling

Low-power, short-range radios of the variety typically used for wireless sensor networks expend virtually as much energy passively listening to the channel as they do during transmission [PK00, APK00]. Sensor net MAC protocols are frequently designed around this assumption, aiming to keep the radio off for as long as possible. TDMA is a common starting point because of the natural mechanism it provides for adjusting the radio's duty cycle, trading energy expenditure for other factors such as channel utilization and message latency [Soh00].

While distributed time synchronization is central to any TDMA scheme, it is considerably more important in wireless sensor nets than in traditional (e.g. cellular phone) TDMA networks. Traditional wireless MAC protocols value only high channel utilization. Good time sync is therefore important because it shortens the guard time, but also easy because each frame received implicitly imparts information about the sender's clock. This information can be used to frequently re-synchronize a node's clock with those of its peers [LS96].

In sensor networks, the picture changes considerably. Energy-efficiency is the highest priority, so localized algorithms are used to minimize both the size and frequency of messages. Long inter-message intervals result in greater clock drift and therefore longer guard times. The high energy cost of passive listening described above makes these guard times expensive. In addition, small data payloads make the guard times a large proportion of the total time a receiver is listening. These factors make good clock synchronization critical for saving energy, and suggest a new technique is needed to achieve it.

3.5. Uses common in traditional distributed systems

The uses of time synchronization we have described so far have been specific to sensor networks, relating to their unique requirements in distributed signal processing, energy efficiency, and localized computation. However, at its core, a sensor network is also a distributed system, where time synchronization of various forms has been used extensively for some time. Many of these more traditional uses apply in sensor networks as well. For example:

- *Logging and Debugging*. During design and debugging, it is often necessary correlate logs of many different nodes' activities to understand the global system's behavior. Logs without synchronized time often make it difficult or impossible to determine causality, or reconstruct an exact sequence of events.

- *Interaction with Users*. People use "civilian time"—their wristwatches—when making requests such as "show me activity in the southeast quadrant between midnight and 4 A.M." For this request to be meaningful, certain nodes in a sensor net need to be synchronized with an external time standard. In some sense, this may be an orthogonal requirement to those discussed previously; many applications of time-sync only require internal consistency.

- *Cryptography*. Perhaps due to sensor nets' applicability in military applications, there has already been significant interest in cryptographically protecting sensor network messages [HSW$^+$, Hil]. Certain authentication schemes, such as the Kerberos Authentication Service [SNS88], depend on synchronized time to prevent replay attacks and other forms of circumvention.[1]

- *Database Consistency.* Database update protocols often require synchronized time to serialize transactions or eliminate duplicate updates (for example, in [LSW91]). There has been recent interest in expanding the domain of databases to encompass embedded sensor network queries [BGS01], using databases running inside the network itself.

4. Related Work

4.1. Distributed Clocks and Network Time Protocols

Perhaps the seminal work in computer clock synchronization is Lamport's work that elucidates the importance of *virtual clocks* in systems where causality is more important than absolute time [Lam78]. In his

[1]Davis and Geer point out in [DGT96] that, in some contexts such as Kerberos, challenge-response can take the place of accurate timestamps.

system, computer has a local clock which is incremented monotonically after each event that it observes. Each message sent to another node also carries the timestamp of the sender with it. When a node receives a message, it advances its local clock its current value is less than the value in the received message. This simple scheme is enough to guarantee that timestamps can reconstruct the total ordering of any sequence of events that were causal. That is, if event A causes event B, the timestamp of A is less than the timestamp of B.

Over the years, many protocols have been designed for maintaining synchronization of physical clocks over computer networks [GZ89, ST87, Cri89, Mil94]. These protocols all have basic features in common: a simple connectionless messaging protocol; exchange of clock information between clients and one (or a few) servers; methods for mitigating the effects of nondeterminism in message delivery and processing; and an algorithm on the client for updating local clocks based on information received from a server. They do differ in certain details: whether the network is kept internally consistent or synchronized to an external standard; whether the server is considered to be the canonical clock, or merely an arbiter of client clocks, and so on.

Mills' NTP [Mil94] stands out by virtue of its scalability, robustness to various types of failures, self-configuration, security in the face of deliberate sabotage, and widespread deployment. NTP allows construction of a hierarchy of time servers, multiply rooted at sources of canonical, external time (see Section 4.2). Levels closer to the root have better accuracy and precision to the external standard, but lower levels can be more populous due to the high degree of fan-out possible from each server. Clients are typically children of multiple higher-layer nodes, using statistical methods to determine the best estimate of the current time.

As we will see in Section 6, an important feature of NTP is that it keeps a running estimate of a client clock's frequency relative to an external standard. This facilitates reasonably accurate timekeeping during long periods of disconnected operation.

4.2. National Time Standards and Time Transfer

Some methods of physical clock synchronization strive only to maintain *internal* consistency. However, others distribute time from an *external* standard—an "out of band" source of time. Frequently, these external sources are the time provided by government agencies tasked with maintaining and disseminating a country's official time. For example, the U.S. Naval Observatory's Time Service Department maintains the official time of the United States. USNO's "Master Clock #2" is steered by an ensemble of advanced hydrogen maser and cesium frequency standards [MMK99]. UTC, the most commonly used international standard timescale, is based on a weighted average of over 200 atomic clocks at over 50 such national laboratories [Tho97]. It is calculated at BIPM (*Bureau International des Poids et Mesures*) in Paris, France.

Many of these laboratories are involved in active research in the area of *time transfer*—that is, synchronization of geographically distant clocks. This ability is a fundamental requirement for any national laboratory whose mission is to transmit time to its users (citizens, the military, and so forth). It is also the technology that enables computation of aggregate time standards such as UTC. Many modern time transfer schemes have been developed.

Historically speaking, they all harken back to much older methods. An unprecedented surge of interest in timekeeping was seen in the 14th century as mechanical clocks swept Europe, often found in the belfries and towers of city halls and churches [Lan83]. Clocks in other forms had been known for at least a thousand years beforehand, such as the sundial or water-driven *clepsydra*. But these newer mechanical clocks were the first to be accompanied by automatically ringing bells—a simple form of time transfer. In 1845, the U.S. Naval Observatory began to disseminate time by dropping a "time ball" from atop a flagpole every day precisely at noon [BD82]. The ball was often used by ships anchored in the Potomac river that needed a calibrated chronometer for navigation before setting out to sea [SA98].

In modern times, the state of the art is in radio time transfer methods. The first of these, seen originally in the 1920's, was the a voice announcer on radio station WWV operated by the National Institute of Standards and Technology. WWV is still in operation today, along with its more recent cousin, WWVB, which provides computer-readable signals in stead of voice announcements. The modern-day service provides references to U.S. time and frequency standards within one part in 10^{13}.

The most active subject of modern-day research is in *two-way satellite time transfer* [SPK00]. TWSTT allows two clocks connected by a satellite relay to both compute their offset from their peer. It is superior to methods such as WWV's radio broadcasts because it transfer is *two-way*, instead of solely from sender to receiver. Use of satellites also allows intercontinental time transfer. TWSTT methods are used to connect the time and frequency standards of national laboratories to BIPM in France for computation of international timescales.

Satellite navigation systems—most notably, GPS [Kap96]—have an important relationship to time. GPS provides localization service by allowing users to measure time of flight of radio signals from satellites (at known locations) to a receiver. The system needs synchronized time for the scheme to work; it also provides one-way time transfer to the user as a "side effect" of localization. (Recent work has focused on using GPS for *two*-way time transfer as well [LL99].) Although a complete discussion of the technical details is beyond the scope of this document, it is important to note that GPS is both a user and disseminator of USNO's timescale. Consumer GPS receivers can synchronize nodes to UTC(USNO) within 200ns [MKMT99].

4.3. Methods for avoiding the problem

In some cases, the time synchronization problem can be solved by *not* solving it—designing the system to avoid the need for the problem to be solved. A number of systems exemplify this design principle. Specifically, it is instructive to reconsider several localization and ranging systems. (The relationship between time and space is closely intertwined; time synchronization often goes hand in hand with localization.)

Many localization systems estimate the distance between two points by measuring the time-of-flight of some phenomenon from one point to the other. Generally speaking, a system emits a recognizable signal at a sender S, then times the interval required for the signal to arrive at a receiver R. If the propagation speed of the phenomenon is known, the time measurement can be converted to distance. The fundamental problem that must be solved is synchronizing S and R in time so that the propagation delay can be properly measured. We will briefly examine three systems that solve this problem in three different ways.

Pinpoint, Inc.'s RF-ID system [**?**] avoids the synchronization problem by setting $S = R$; that is, making S and R the same node. The RF-ID system emits an RF pulse which is modulated by the target and reflected back to the sender. The sender and receiver are implicitly synchronized by virtue of being the same device. This allows a measurement of the signal's round-trip time, and therefore the range from the sender/receiver to the target.

The GPS satellite system, mentioned earlier, also localizes by measuring the time-of-flight of RF signals. GPS also avoids synchronizing S (the satellites) with R (the user's receiver) by treating the clock bias between S and R as an unknown. Because there are many redundant sending satellites, the receiver can lock onto one additional satellite, solving for its (X, Y, Z, T) instead of only (X, Y, Z). As we discussed earlier in Section 3.1, this only works by virtue of the senders being synchronized with each other.

Finally, ORL's Active Bat system [WJH97] synchronizes S with R by using a synchronization modality different from the measurement modality. In their system, modulo certain details, S simultaneously emits a (fast) RF pulse and a (slow) ultrasound pulse. The time of flight of the RF pulse is negligible compared to that of the ultrasound. The RF pulse can be considered an instantaneous phenomenon that establishes synchronization between S and R with respect to the ultrasound pulse.

Of course, the localization application (and these implementations in particular) are only representative. Many other systems use similar designs to avoid the need for synchronized clocks. For example, in the LOCUS distributed operating system [PWB$^+$87], consistency of distributed filesystem operations is ensured by dynamically selecting a single node as a synchronization point for a particular file-descriptor. By forcing all file operations to go through the synchronization point, no clock sync is needed to achieve maintain a consistent view of the total ordering of file events.

5. Sensor Network Time

5.1. The Need for Something New

We argued in Section 3 that it is important for sensor networks to have access to synchronized time. However, in Section 4, we described a wide variety of methods for synchronizing time. Is something new needed?

To answer this question, we it is important to first consider what sort of synchronization a sensor network *needs*. Because of the highly application-specific nature of a given network, it is hard to make generalizations—but even within a single application, time sync is needed at many layers, each with its on requirements. For example, consider all the potential uses of time-sync in an object tracking system:

- A single detection of the target might be performed by a beamforming array designed to localize the source of sound, such as that described by YHRCL in [YHR$^+$98]. The array described shares a common time base by virtue of the fact that the audio data are all fed to the same processor. For such an array to be implemented on a fully distributed set of autonomous wireless sensors, network time synchronization is needed. This synchronization would require maximum error of about 100μsec but could be limited in lifetime and local in scope.

- Multiple position detections can be integrated into speed and direction estimates. The timebase required for this computation needs to be longer in lifetime than the above, and larger in scope (probably encompassing several hops, radio-wise). The maximum error is connected to the speed of the target, rather than the speed of sound.

- We discussed the importance of *aggregation* in sensor networks in Section 3.3. Information about the target collected from all over the network needs to be aggregated and processed hierarchically in lieu of simply sending all the raw data back to a single aggregation point. Partially summarized data may arrive at an aggregation point that consists of readings originally collected from a very distant node. Although the synchronization error tolerances for this kind of aggregation would be very relaxed compared to beam-forming, the timebase needs to be much larger in scope and much longer in lifetime—perhaps even persistent.

- Interactions with users (e.g., "What was here some time around 4A.M.?" are likely to have very relaxed error constraints (seconds, perhaps even a minute), but—unlike other examples—requires *external* synchronization, rather than just internal consistency.

Of course, underlying all of these requirements it the ever-present need for *energy-efficiency*. Ignorance of this requirement, while entirely justified in traditional contexts, is the Achilles' heel of most existing time synchronization methods. Although protocols such as NTP are conservative in their use of bandwidth, they are extremely inefficient in this new context where radios consume significant power even by passively listening for messages [PK00].

Cost and form-factor are also important restrictions. The smallest nodes in a network will, perhaps, be disposable and small enough to be attached directly to the phenomena that they are monitoring. These are unlikely to have room in their packaging or budgets that allows anything more than a local oscillator and a short-range radio.

We conclude, therefore, that no existing method meets a sensor net application's diverse synchronization requirements while still being compatible with the network's energy budget, cost structure, and form factor.

Something new *is* needed. But what?

5.2. Design Principles

While our earlier discussion highlights the inadequacy of current methods, it also gives structure to solutions. In this section, we outline the general design principles under which we plan to develop new methods of time synchronization for sensor networks.

5.2.1. Multi-modal synchronization

The heterogeneity in the synchronization requirements across and within sensor network applications is daunting. Even without the constraints of limited energy, cost, and form-factor, no single method is likely to meet every one of these requirements. With these extra constraints, finding such a method seems a completely lost cause.

Because it is impossible for any *single* synchronization method to appropriate in all situations, sensors should have *multiple* methods available. By modifying existing methods, developing new ones, and even composing them into derivative methods, we hope to provide an entire palette of time sync methods that, taken together, covers a good portion of the parameter space we described in Section 2.

A multi-modal solution is also a good choice for building a system that is energy-efficient. If a node can dynamically trade error for energy, or scope for convergence time, it can avoid "paying" for something that it doesn't need. Ideally, the algorithms should also be *tunable*—allowing finer control over an algorithm than simply turning it on or off. Our goal is to implement and characterize a set of methods rich enough so that all applications will have one available that is *necessary and sufficient* for its needs.

5.2.2. Tiered architectures

Although Moore's law predicts that hardware for sensor networks will inexorably become smaller, cheaper, and more powerful, technological advances will never prevent the need to make tradeoffs. Even as our notions of metrics such as "fast" and "small" evolve, there will always be compromises: nodes will need to be faster *or* more energy-efficient, smaller *or* more capable, cheaper *or* more durable.

Instead of choosing a *single* hardware platform that makes a particular set of compromises, we believe an effective design is one that uses a *tiered* platform consisting of a heterogeneous collection of hardware. Larger, faster, and more expensive hardware (beacons, aggregation points) can be used more effectively by also using smaller, cheaper, and more limited nodes (sensors, tags). An analogy can be made to the memory hierarchy commonly found in desktop computer systems. CPUs typically have extremely expensive, fast on-chip cache, backed by slower but larger L2 cache, main memory, and ultimately on-disk swap space. This organization, combined with a tendency in computation for locality of reference, results in a memory system that appears to be as large and as cheap (per-byte) as the swap space, but as fast as the on-chip cache memory. In sensor networks, where localized algorithms are a primary design goal, similar benefits can be realized by creating the network from a spectrum of hardware ranging from small, cheap, and numerous, to large, expensive, and powerful.

The smallest nodes are unlikely to have little more than a local oscillator and a short-range radio. However, they can be supported by other nodes in the network that are better endowed, with longer-range radios capable of synchronizing with more remote parts of the network, or in some cases with external time sources such as GPS or WWVB.

5.3. Our Approach

Using these principles as a guide, we are building new time synchronization services for sensor networks. Specifically, we are

- Extending existing sync methods to be energy-aware
- Developing new sync methods, including compositions of basic methods
- Characterizing our new techniques in the parameter space we defined
- Building a testbed to realize our methods with actual applications

Starting down this path, we have developed a technique called *post-facto synchronization* to reconcile the need of many applications for accurate sensor event timestamps with the desire to keep the node off in order to conserve energy. In the next section, we describe this technique and an experiment that characterizes its performance.

6. Post-Facto Synchronization

To save energy in a sensor network, it is a desirable to keep nodes in a low-power state, if not turned off completely, for as long as possible. Sensor network hardware is often designed with this goal in mind; processor have various "sleep" modes or are capable of powering down high-energy peripherals when not in use.

This type of design is exemplified by the WINS platform [ADL+98], which has an extremely low-power "pre-processor" that is capable of rudimentary signal processing. Normally, the entire node is powered down except for the pre-processor. When the pre-processor detects a potentially interesting signal, it powers on the general purpose processor for further analysis. The CPU, in turn, can power on the node's radio if it determines that an event has occurred that needs to be reported.

Such designs allow the components that consume the most energy to be powered for the least time, but also pose significant problems if we wish to keep synchronized time. Traditional methods try to keep the clock disciplined at all times so that an accurate timestamp is always available. What happens if the radio—our external source of time and frequency standards—is continuously off for hours at a time? Or, in the case of a platform like WINS, what if the general-purpose processor that knows how to discipline the clock is also off?

Our solution to this problem is *post-facto synchronization*. In our scheme, nodes' clocks are normally *un*synchronized. When a stimulus arrives, each node records the time of the stimulus with respect to its own local clock. Immediately afterwards, a "third party" node—acting as a beacon—broadcasts a synchronization pulse to all nodes in the area using its radio. Nodes that receive this pulse use it as an instantaneous time reference and can normalize their stimulus timestamps with respect to that reference.

This kind of synchronization is not applicable in all situations, of course: it is limited in scope to the transmit range of the beacon and creates only an "instant" of synchronized time. This makes it inappropriate for an application that needs to communicate a timestamp over long distances or times. However, it does provide exactly the service necessary for beam-forming applications, localization systems, and other situations in which we need to compare the relative arrival times of a signal at a set of spatially local detectors.

6.1. Expected Sources of Error

There are three main factors that affect the accuracy and precision achievable by post-facto synchronization. Roughly in order of importance, they are: receiver clock skew, variable delays in the receivers, and propagation delay of the synchronization pulse.

- **Skew in the receivers' local clocks.** Post-facto synchronization requires that each receiver accurately measure the interval that elapses between their detection of the event and the arrival of the synchronization pulse. However, nodes' clocks do not run at exactly the same rate, causing error in that measurement. Since clock skew among the group will cause the achievable error bound to decay as time elapses between the stimulus and pulse, it is important to minimize this interval.

  One way of reducing this error is to use NTP to discipline the frequency of each node's oscillator. This exemplifies our idea of multi-modal synchronization. Although running NTP "full-time" defeats one of the original goals of keeping the main processor or radio off, it can still be useful for frequency discipline (much more so than for phase correction) at very low duty cycles.

- **Variable delays on the receivers.** Even if the synchronization signal arrives at the same instant at all receivers, there is no guarantee that each receiver will *detect* the signal at the same instant. Nondeterminism in the detection hardware and operating system issues such as variable interrupt latency can contribute unpredictable delays that are inconsistent across receivers. The detection of the event itself (audio, seismic pulses, etc.) may also have nondeterministic delays associated with it. These delays will contribute directly to the synchronization error.

  Our design avoids error due to variable delays in the *sender* by considering the sender of the sync pulse to be a "third party." That is, the receivers are considered to be synchronized only with each other, not with the beacon.

  It is interesting to note that the error caused by variable delay is the same irrespective of the time elapsed between the event and the sync pulse. This is in contrast to error due to clock skew that grows over time.

- **Propagation delay of the synchronization pulse.** Our method assumes that the synchronization pulse is an absolute time reference at the instant of its arrival—that is, that it arrives at every node at exactly the same time. In reality, this is not the case due to the finite propagation speed of RF signals. Synchronization will never be achievable with accuracy better than the difference in the propagation delay between the various receivers and the synchronization beacon.

This source of error makes our technique most useful when comparing arrival times of phenomena that propagate much more slowly than RF, such as audio. The six-order-of-magnitude difference in the speed of RF and audio has been similarly exploited in the past in systems such as the ORL's Active Bat [WJH97] and Girod's acoustic rangefinder [Gir00].

6.2. Empirical Study

We designed an experiment to characterize the performance of our post-facto synchronization scheme. The experiment attempts to measure the sources of error described in the previous section by delivering a stimulus to each receiver at the same instant, and asking the receivers to timestamp the arrival time of that stimulus with respect to a synchronization pulse delivered via the same mechanism. Ideally, if there are no variable delays in the receivers or skew among the receivers' local oscillators, the times reported for the stimulus should be identical. In reality, these sources of error cause the dispersion among the reported times to grow as more time elapses between the stimulus and the sync pulse. The decay in the error bound should happen more slowly if NTP is simultaneously used to discipline the frequency of the receivers' oscillators.

We realized this experiment with one sender and ten receivers, each of which was ordinary PC hardware (Dell OptiPlex GX1 workstations) running the RedHat Linux operating system. Each stimulus and sync pulse was a simple TTL logic signal sent and received by the standard PC parallel port.[2] In each trial, each receiver reported its perceived elapsed time between the stimulus and synchronization pulse according to the system clock, which has 1μsec resolution. We defined the dispersion to be the standard deviation from the mean of these reported values. To minimize the variable delay introduced by the operating system, the times of the incoming pulses were recorded by the parallel port interrupt handler using a Linux kernel module.

In order to understand how dispersion is affected by the time elapsed between stimulus and sync pulse, we tested the dispersion for 21 different values of this elapsed time, ranging from $2^4\mu$sec to $2^{24}\mu$sec (16μsec to 16.8 seconds). For each elapsed-time value, we performed 50 trials and reported the mean. These 1,050 trials were performed in a random order over the course of one hour to minimize the effects of systematic error (e.g. changes in network activity that affect interrupt latency).

For comparison, this entire experiment was performed in three different configurations:

1. The experiment was run on the "raw clock": that is, while the receivers' clocks were not disciplined by any external frequency standard.

2. An NTPv3 client was started on each receiver and allowed to synchronize (via Ethernet) to our lab's stratum-1 GPS clock for ten days. The experiment was then repeated while NTP was running.

3. NTP's external time source was removed, and the NTP daemon was allowed to free-run for several days using its last-known estimates of the local clock's frequency. The experiment was then repeated.

To compare our post-facto method to the error bound achievable by NTP alone, we recorded two different stimulus-arrival timestamps when running the experiment in Configuration 2: the time with respect to the sync pulse *and* the time according to the NTP-disciplined local clock. Similar to the other configurations, a dispersion value for NTP was computed for each stimulus by computing the standard deviation from the mean of the reported timestamps. The horizontal line in Figure 1 is the mean of those 1,050 dispersion values—101.70 μsec.

[2] This was accomplished using the author's parallel port pin programming library for Linux, `parapin`, which is freely available at http://www.circlemud.org/jelson/software/parapin

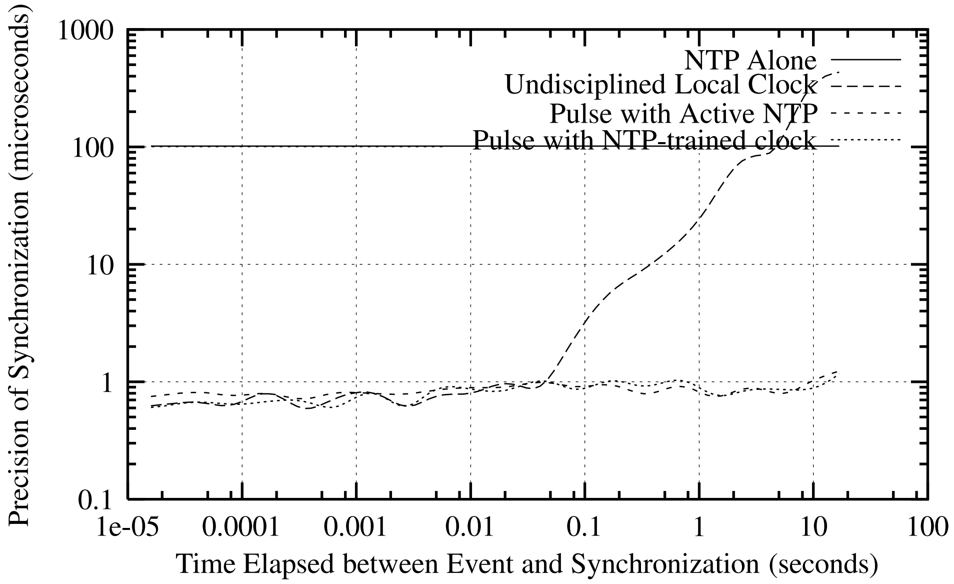

Figure 1: Synchronization error using post-facto time synchronization without external frequency discipline, with discipline from an active NTP time source, and with free-running NTP discipline (external time source removed after the oscillator's frequency was estimated). These are compared to the error bound achievable with NTP alone (the horizontal line near 100μsec). The breakpoint seen near 50msec is where error due to clock skew, which grows proportionally with the time elapsed from stimulus to sync pulse, overcomes other sources of error that are independent of this interval. Each point represents the dispersion experienced among 10 receivers, averaged over 50 trials.

Our results are shown in Figure 1. [3]

6.3. Discussion

The results shown in Figure 1 illuminate a number of aspects of the system. First, the experiment gives insight into the nature of its error sources. The results with NTP-disciplined clock case are equivalent to undisciplined clocks when the interval is less than $\approx$ 50msec, suggesting that the primary source of error in these cases is variable delays on the receiver (for example, due to interrupt latency or the sampling rate in the analog-to-digital conversion hardware in the PC parallel port). Beyond 50msec, the two experiments diverge, suggesting that clock skew becomes the primary source of error at this point.

Overall, the performance of post-facto synchronization was quite good. When NTP was used to discipline the local oscillator's frequency, an error bound very near to the clock's resolution of 1μsec was achieved. This is significantly better than the 100μsec achieved by NTP alone. Clearly, the combination of NTP's frequency estimation with the sync pulse's instantaneous phase correction was very effective. Indeed, the multi-modal combination's maximum error is better than either mode can achieve alone. We find this a very encouraging indicator for the multi-modal synchronization framework we proposed at the end of Section 5.

Without NTP discipline, the post-facto method still performs reasonably well for short intervals between stimulus and sync pulse. For longer intervals, we are at the mercy of happenstance: the error depends on the natural frequencies of whatever oscillators we happen to have in our receiver set.

[3] "The joy of engineering is to find a straight line on a double logarithmic diagram." –Thomas Koenig

Perhaps the most exciting result, however, is shown in the experiment where NTP disciplined the nodes' local clocks using only the last-known-estimate of frequency, after the external time source was removed. The achievable error bound was 1μsec: the limit of our clock's resolution and, more importantly, exactly the same as the result with NTP and an active external time standard. This result is important because it shows that extremely low-energy and low-error time synchronization is possible: after an initial frequency-training period, nodes might be able to keep their radios off for days and still instantly acquire a timebase within 1μsec when an event of interest arrives. That result is made possible by the multi-modal synchronization; the frequency correction provided by free-running NTP is not good enough to keep clocks precisely in phase over time due to accumulated error. (In the free-running NTP experiment, the accuracy of the timestamps when not normalized by the sync pulse was only in the tens of milliseconds.)

All of these results, while encouraging, do come with a number of caveats. Our experiments results were performed under idealized laboratory conditions, using (equal-length) cables to directly connect the sender to the receivers. Real world conditions will require wireless links that are likely far more complex with more opportunities for variable delay. In addition, the relatively constant ambient temperature reduced the oscillators' frequency drift over time. A real sensor network deployed outdoors might not be able let NTP free-run without an external time source for as long as we did in our experiment.

7. Work Plan

Although the experiment we have described is encouraging, it is only a preliminary study. There are a large number of possible future directions for research; we have focused only on some that seem the most promising and interesting.

Our planned work for the future falls into three main categories. First, we wish to develop and characterize new time synchronization methods. We also plan to characterize traditional synchronization methods for the purpose of comparison. Second, we would like to implement a working system as proof-of-concept. Specifically, we will develop a testbed that can track the motion of objects through a field of sensors; this application will demonstrate two different time synchronization in different contexts. Finally, we will explore methods that might allow a running system to adapt to new applications and conditions by automatically selecting the time synchronization methods that are most appropriate.

7.1. Development and characterization of time sync methods

An important goal of our work is to provide a palette of time sync methods to applications that covers a good portion of the parameter space we described in Section 2. Because it is impossible for any *single* synchronization method to appropriate in all situations, sensors should have multiple methods available. If a node can dynamically trade error for energy, or scope for convergence time, it can avoid "paying" for something that it doesn't need. Ideally, the algorithms should also be *tunable*—allowing finer control over an algorithm than simply turning it on or off.

Towards this goal, we plan to continue the development of sync methods and their characterization along the axes we described earlier. In addition, we plan to more fully characterize the "traditional" time sync methods for comparison. Our specific plans for further development and experiments follows.

- We plan a re-implementation of our post-facto pulse synchronization experiment on hardware that is more akin to hardware that will be found in sensor networks: slower, lower-power nodes that have wireless radio links. While using PCs with wired stimuli and event delivery did provide an important proof-of-concept and encouraging results, we wish to investigate the effects on error of factors such as slower clock speeds, variable latency of radios, and nondeterminism introduced by radio propagation anomalies. We plan to do these tests using the wireless sensor network testbed in place as part of the related SCADDS project [EGHK99, CEE$^+$01].

There are many interesting dimensions of pulse synchronization that we did not explore with our initial experiment. For example, what are the effects on error of scaling up the number of nodes, the duty cycle of the nodes, or changes in the ambient temperature? Many of these questions can be answered with a larger, truly untethered testbed.

An even a more basic question that we would like to answer is: what is the best modality for a synchronization pulse? We should not assume that a radio signal is the best (e.g., most deterministically detected) mode. We plan a simple experiment to test the detection determinism of other modes not yet examined, such as using a strobe light to trigger remote photodiodes. By using a multichannel oscilloscope to compare the dispersion in triggering times of various modes (e.g., the strobe light, radios with various MAC layers, and wired stimuli), we can get an interesting picture of how the detection modalities themselves contribute to the overall system's error budget.

- Additional (related) time-synchronization methods will be developed based on the existing primitives. Specifically, we wish to investigate *synchronization pulse chaining*. Pulse chaining is an extension to the pulse sync method we described earlier. Standard pulse synchronization is only effective for creating a synchronized timebase within the transmit radius of a single beacon. However, if certain receivers "relay" the pulse—retransmitting it to a new set of receivers outside the range of the original beacon—the scope of the timebase can be expanded. The pulse may be relayed many times to make the scope of the timebase larger and larger.

  Naturally, each link in this synchronization chain will accumulate error. Our plan is to understand how this error accumulates as the size of the synchronized area increases. This "maximum error vs. distance" is analogous to the "maximum error vs. time" experiment we described in Section 6.

- We will characterize some existing methods for acquiring time along the same axes (such as scope, maximum error, lifetime, etc.) that we have used to evaluate our own time sync methods. This will allow a for a more informed comparison of our methods with related work in the area. Our experiment's comparison of pulse synchronization with NTP is one example of this kind of comparison; we would like to make similar statements with respect to systems such as GPS, WWV/WWVB, and GPS-via-CDMA.

7.2. Testbed implementation as proof-of-concept

Although experiments that characterize our methods are important, we also think it is important to implement an actual application using these methods as a proof-of-concept. Therefore, the second major portion of our planned work is the implementation of a testbed that demonstrates our time sync algorithms in the context of a sensing application.

Our proposed testbed application is *tracking of cooperative objects through a sensor field*. A "cooperative" object is one that wants to be tracked; that is, the object has been augmented in a way that makes it easier to localize. The goal is to build sensor network that will report to a user the location and speed of the cooperative object over time. The sensor field will be truly distributed: that is, its geographic span will prevent any single node from broadcasting a radio message globally.

The application we propose will build on Girod's prototype acoustic rangefinder [Gir00]. His rangefinder uses digital signal processing techniques to accurately determine the time of arrival of a specially formed sound. (The sound is a "chirp" that is modulated by a sender to make it more easily detected by the receiver.) If the sender and receiver are synchronized in time, Girod's technique allows accurate determination of the time of flight of sound between two points. This, in turn, leads to an accurate range estimate between them.

In cooperation with Girod, we plan to use post-facto time synchronization to facilitate measurement of the time of flight of sound from an audio source to a *set* of receivers—not just a single receiver—allowing them to *trilaterate*. At any given instant, this technique will report instantaneous position in a coordinate system defined by the sensors[4]. As the object moves through the sensor field, many such localization measurements can be taken and integrated to form an estimate of velocity.

This experiment demonstrates two aspects of our time synchronization services:

- Post-facto synchronization. The post-facto sync we have developed (as described earlier in this paper) allows a localized set of receivers to create an ephemeral synchronization with a maximum error of 1μsec. This is exactly what the trilateration/localization service needs.

- Pulse chaining. In order to integrate a series of these localization measurements over time, the sensors must be able to share a timebase that is wider in scope and longer in lifetime. However, since the object will be moving relatively slowly, the maximum error constraint is considerably weaker (e.g., on the order of 10's or even 100's of milliseconds). Pulse chaining makes exactly this tradeoff: chaining together a series of local, ephemeral timebases into a larger-scope and longer-lasting one, at the expense of accumulated error.

We believe this experiment to be an effective demonstration on several fronts:

- Even an single application can often have a variety of time-sync needs.

- Meeting these disparate needs by using a variety of time-sync methods results in a more energy-efficient and less expensive system than a single sync method that would be forced to meet the union of all requirements.

- Sensor nodes can sometimes achieve advanced time-sync without the need for additional specialized hardware.

7.3. Exploration of methods for tuning and automatic modality selection

There are additional future directions that we wish to pursue, though time may not permit a complete exploration of these details in the context of an actual implementation. We plan to examine some of these issues in the context of simulation or modeling instead. Specifically:

- Our vision is that nodes can save energy by selecting a time synchronization method that is both necessary and sufficient for its needs. Ideally, those methods should also be *tunable*—allowing finer control over an algorithm than simply turning it on or off. Tunability will allow applications to tailor a sync method, making the gap between what's needed and what's provided even smaller. Through

[4] In this experiment, we assume that each sensor is preprogrammed with its position in some global coordinate system. In the context of a testbed meant to demonstrate time synchronization, this assumption is a reasonable one. In parallel, Girod is developing techniques that allow a distributed federation of sensors to dynamically create a self-consistent coordinate system.

some simple experimentation or simulation, we would like to study the gains that might be realized through tuning.

One such "tuning knob" that might be possible is in selecting the number of nodes that participate in a federation of synchronized nodes. Many time-standards (e.g., UTC(USNO), the U.S. Naval Observatory's realization of UTC), use a large ensemble of clocks and dynamically choose the best in order to improve precision. Will adding additional beacons make post-facto synchronization better, at the expense of more energy expended by the additional nodes that are participating? If so, this is another interesting energy-error tradeoff.

- Until now, we have assumed that a palette of sync methods are available to applications, and the exact method used is selected by the system designer at design-time. Is it possible for the system to adapt by dynamically selecting the best algorithm while it's running? Is it possible for an application to automatically turn the tuning knobs described earlier?

8. Conclusions

Time synchronization is a critical piece of infrastructure for any distributed system. Distributed, wireless sensor networks make heavy use of synchronized time, but often have unique requirements in the scope, lifetime, and maximum error of the synchronization achieved, as well as the time and energy required to achieve it. Existing time synchronization methods need to be extended to meet these new needs.

We have presented an implementation of our own sensor network time synchronization scheme, *post-facto synchronization*. This method combines the oscillator frequency discipline provided by NTP with an instantaneous phase correction provided by a simple synchronization signal sent by a beacon. Our experiments have shown timing error for a group of 10 nodes to be bounded near the limit of our clock resolution of 1μsec.

An important additional result is that the same error bound was achieved even when NTP no longer had an active external time or frequency standard, after an initialization period when it was allowed to estimate the local oscillator's frequency error. This is critical for sensor networks where limited energy reserves and the high energy cost of operating a wireless radio make standard NTP unsuitable for long-lived, low-power operation.

Although our current results are a preliminary laboratory study, we believe that post-facto synchronization over wireless radios will be able to support the same instantaneous creation of a short-lived but low-error synchronized timebase ever after a long period of radio silence. We propose ongoing research where we can

- Move our experiment from the lab to real sensor network nodes, and develop additional methods;
- Characterize our methods, and traditional methods, according to appropriate metrics for comparison;
- Prove these methods are applicable to real systems by building an application testbed; and
- Explore additional issues such as tuning and voting through simulation and modeling.

Acknowledgments

Portions of this work were supported by DARPA under grant No. DABT63-99-1-0011 as part of the SCADDS project, and was also made possible in part due to support from Cisco Systems.

References

[ADL+98] G. Asada, M. Dong, T.S. Lin, F. Newberg, G. Pottie, W.J. Kaiser, and H.O. Marcy. Wireless Integrated Network Sensors: Low Power Systems on a Chip. In *Proceedings of the European Solid State Circuits Conference*, 1998.

[APK00] A.A. Abidi, G.J. Pottie, and W.J. Kaiser. Power-conscious design of wireless circuits and systems. *Proceedings of the IEEE*, 88(10):1528–45, October 2000.

[BD82] Ian R. Bartky and Steven J. Dick. The first north american time ball. *Journal for the History of Astronomy*, 13:50–54, 1982.

[BGS01] Ph. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *Proceedings of the 2nd International Conference on Mobile Data Management*, Hong Kong, January 2001.

[CEE+01] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao. Habitat monitoring: Application driver for wireless communications technology. In *Proceedings of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[Cri89] Flaviu Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3:146–158, 1989.

[DGT96] Donald T. Davis, Daniel E. Geer, and Theodore Ts'o. Kerberos with clocks adrift: History, protocols, and implementation. *Computing Systems*, 9(1):29–46, Winter 1996.

[EGHK99] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270, 1999.

[Gir00] Lewis Girod. Development and characterization of an acoustic rangefinder. Technical Report 00-728, University of Southern California, Department of Computer Science, March 2000.

[GZ89] R. Gusell and S. Zatti. The accuracy of clock synchronization achieved by TEMPO in Berkeley UNIX 4.3 BSD. *IEEE Transactions on Software Engineering*, 15:847–853, 1989.

[Hil] Jason Hill. A security architecture for the post-pc world. U.C. Berkeley Technical Report, to appear.

[HSW+] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. U.C. Berkeley Technical Report, to appear.

[IGE00] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, pages 56–67, Boston, MA, August 2000. ACM Press.

[Kap96] Elliott D. Kaplan, editor. *Understanding GPS: Principles and Applications*. Artech House, 1996.

[KKP99] J.M. Kahn, R.H. Katz, and K.S.J. Pister. Next century challenges: mobile networking for Smart Dust. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278, 1999.

[Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–65, 1978.

[Lan83] David S. Landes. *Revolution in Time: Clocks and the Making of the Modern World*. Belknap Press, 1983.

[LL99] Kristine Larson and Judah Levine. Time transfer using the phase of the gps carrier. *IEEE Transactions On Ultrasonics, Ferroelectrics and Frequency Control*, 46:1001–1012, 1999.

[LS96] Henrik Lönn and Rolf Snedsbøl. Efficient synchronisation, atomic broadcast and membership agreement in a TDMA protocol. Technical report, CTH, Dept. of Computer Engineering, Laboratory for Dependable Computing (LDC), 1996.

[LSW91] B. Liskov, L. Shrira, and J. Wroclawski. Efficient at-most-once messages based on synchronized clocks. *ACM Trans. on Computer Sys., Special Section on Communication Architectures and Protocols*, 9(2):125, May 1991.

[Mil94] David L. Mills. Internet Time Synchronization: The Network Time Protocol. In Zhonghua Yang and T. Anthony Marsland, editors, *Global States and Time in Distributed Systems*. IEEE Computer Society Press, 1994.

[MKMT99] J. Mannermaa, K. Kalliomaki, T. Mansten, and S. Turunen. Timing performance of various GPS receivers. In *Proceedings of the 1999 Joint Meeting of the European Frequency and Time Forum and the IEEE International Frequency Control Symposium*, pages 287–290, April 1999.

[MMK99] D.N. Matsakis, M. Miranian, and P. Koppang. Steering the U.S. Naval Observatory (USNO) Master Clock. In *Proceedings of the Institute of Navigation National Technical Meeting: Vision 2010, Present and Future*, January 1999.

[PK00] G.J. Pottie and W.J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[PWB+87] G. Popek, B. Walker, D. Butterfield, R. English, C. Kline, G. Thiel, and T. Page. Functionality and architecture of the LOCUS distributed file system. In Bharat K. Bhargava, editor, *Concurrency Control and Reliability in Distributed Systems*. Van Norstrand Reinhold (New York, NY), 1987. Also published in ACM SOSP-9, Bretton Woods NH, Oct. 1982.

[SA98] Dava Sobel and William J.H. Andrewes. *The Illustrated Longitude*. Walker & Co., 1998.

[SNS88] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In USENIX Association, editor, *USENIX Conference Proceedings (Dallas, TX, USA)*, pages 191–202, Berkeley, CA, USA, Winter 1988. USENIX Association.

[Soh00] Katayoun Sohrabi. *On Low Power Self Organizing Sensor Networks*. PhD thesis, University of California, Los Angeles, 2000.

[SPK00] W. Schaefer, A. Pawlitzki, and T. Kuhn. New trends in two-way time and frequency transfer via satellite. In *Proceedings of the 31st Annual Precise Time and Time Interval (PTTI) Systems and Applications Meeting*, 2000.

[ST87] T. K. Srikanth and Sam Toueg. Optimal clock synchronization. *J-ACM*, 34(3):626–645, July 1987.

[Tho97] C. Thomas. The accuracy of the international atomic timescale TAI. In *Proceedings of the 11th European Frequency and Time Forum*, pages 283–289, 1997.

[WJH97] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, October 1997.

[YHR+98] K. Yao, R.E. Hudson, C.W. Reed, D. Chen, and F. Lorenzelli. Blind beamforming on a randomly distributed sensor array system. *IEEE Journal of Selected Areas in Communications*, 16(8):1555–1567, Oct 1998.