Wireless Sensor Networks: A New Regime for Time Synchronization

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

Kay Römer
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
roemer@inf.ethz.ch

ABSTRACT

Wireless sensor networks (WSNs) consist of large populations of wirelessly connected nodes, capable of computation, communication, and sensing. Sensor nodes cooperate in order to merge individual sensor readings into a high-level sensing result, such as integrating a time series of position measurements into a velocity estimate. The physical time of sensor readings is a key element in this process called data fusion. Hence, time synchronization is a crucial component of WSNs. We argue that time synchronization schemes developed for traditional networks such as NTP [21] are ill-suited for WSNs and suggest more appropriate approaches.

Keywords

time synchronization, wireless sensor network

1. INTRODUCTION

Wireless sensor networks [1] are an increasingly attractive means to bridge the gap between the physical and virtual world. A WSN consists of large numbers of cooperating small-scale nodes, each capable of limited computation, wireless communication, and sensing. In a wide variety of application areas including geophysical monitoring, precision agriculture, habitat monitoring, transportation, military systems and business processes, WSNs are envisioned to be used to fulfill complex monitoring tasks. With this new class of networks also come new challenges in many areas of the system's design.

Sensor nodes are small-scale devices; in the near future, low-cost platforms with volumes approaching several cubic millimeters are expected to be available [15]. Such small devices are very limited in the amount of energy they can store or harvest from the environment. Thus, *energy efficiency* is a major concern in a WSN. In addition, many thousands of sensors may have to be deployed for a given task—an individual sensor's small effective range relative to a large area of interest makes this a requirement, and its small form factor and low cost makes this possible. Therefore, *scalability* is another critical factor in the design of the system.

In contrast to traditional wired networks, a WSN is both highly *dynamic* and *ad hoc*. For example, initial deployment may involve throwing nodes from an aircraft into an area of interest at random. Over time, sensors fail due to destruction or battery depletion; new sensors might be added in higher concentration in areas of interest. Sensors experience changes in their position, available energy, and task details. Changes in the environment can dramatically affect radio propagation, causing frequent network topology changes and network partitions. At high densities, WSNs also become much more likely to suffer communication failures due to contention for their shared communication medium—[11] reports a message loss of 20% and above between adjacent nodes in a dense WSN. These factors lead to strong *self-configuration* and *robustness* requirements in a WSN. Static configuration is unacceptable; the system must continuously adapt to make the best use of available resources.

While individual sensor nodes have only limited functionality, the global behavior of the WSN can be quite complex. The true value of the network is in this property of *emergent behavior*: the functionality of the whole is greater than the sum of its parts. This is achieved, in part, through *data fusion*, the process of transforming and merging individual sensor readings into a high-level sensing result. This is where time synchronization enters the stage, as it plays a key role in many types of data fusion.

Time synchronization in distributed systems is a well-studied problem. Many solutions exist for traditional networks and distributed systems. NTP [21], for example, has been widely deployed and proven effective and robust in the Internet. In this paper, we explore the question: do the traditional methods apply in sensor networks as well? Our answer is no. Many assumptions on which existing schemes are based no longer hold in this new area of WSNs. We claim that something new is needed.

The organization of the remainder of this paper is as follows. In Section 2, we discuss in more detail the applications and requirements of synchronized time in a WSN. We then review existing time synchronization schemes in Section 3, and examine their shortcomings when applied in this new context. In Section 4, we describe general design principles for WSN time synchronization, based on experiences with a number of prototype systems built by the authors. Finally, in Section 5, we draw our conclusions and describe future work.

2. SYNCHRONIZED TIME IN A WSN

Time synchronization is an important feature of almost any distributed system. A confluence of factors makes flexible and robust time synchronization particularly important in WSNs, while simultaneously making it more difficult to achieve than in traditional networks. In this section, we will describe some of these factors: the tight link between sensors and the physical world; the scarcity of system energy; the need for large-scale, decentralized topologies; and unpredictable, intermittent connectivity.

The advent of logical time [17, 20] eliminated the need for physical time synchronization in situations where only causal relationships of events are of interest to the application. However, logical time only captures relationships between "in system" events, defined by message exchanges between event-generating processes. This is not the case for phenomena sensed by the nodes in a WSN; physical time must be used to relate events in the physical world. Logical time is not sufficient in the WSN domain. For example, consider the following applications:

- Object tracking: The size, shape, direction, location, velocity, or acceleration of objects is determined by fusing proximity detections from sensors at different locations.
- Consistent state updates: The current state of an object is most accurately determined by the node that has "sighted" the object most recently.
- Distributed beamforming: beam-forming arrays [29] can perform "spatial filtering," receiving only signals arriving from a certain direction. This depends on the relative time offsets of the array's sensors.
- Duplicate detection: The time of an event helps nodes determine if they are seeing two distinct real-world events, or a single event seen from two vantage points.
- Temporal order delivery: Many data fusion algorithms must process events in the order of their occurrence [24]—for example, Kalman filters.

Another illustrative example is the formation of a TDMA schedule for low-energy radio operation. This is an important application because listening and transmitting are both very energy-expensive operations in a low-power radio. A common technique to conserve precious energy is to turn the radio off, waking up only briefly to exchange short messages before going back to sleep [22, 26].

Consider two nodes that have agreed to rendezvous on the radio channel once every 60 seconds to exchange a short message—say, 8 bits representing the current temperature. Using a 19.2kbit/sec radio such as our testbed's RF Monolithics [4], 8 bits can be transmitted in about $0.5ms$. However, in practice, the radio must be awakened early to account for time synchronization error—so an expectation of a $1ms$ phase error will triple the total amount of time the radio is expending energy listening to the channel. In addition, even assuming perfect synchronization at the start of a sleep period, a typical quartz oscillator on such a sensor will drift on the order of 1 part in 10^5 [28], or $0.6ms$ after 60 seconds. Of course, sending synchronization packets during the sleep period defeats the purpose of sleeping, so we must consider frequency estimation as part of the time synchronization problem.

The examples above demonstrate not only the importance of time synchronization in a WSN, but also one of its difficulties: any resource expended for synchronization reduces the resources available to perform the network's fundamental task. Many current data acquisition systems do not have this constraint, so they often rely on high-energy solutions to the synchronization problem—frequent network synchronization, high stability frequency standards, GPS receivers, and so forth. In a WSN, the impact of such solutions—in terms of energy, cost, and form-factor—can make them untenable.

Another important aspect of the problem domain illustrated by our examples is the heterogeneity of the application requirements over a wide variety of axes. For example:

- *Energy utilization.* Some synchronization schemes require extra, energy-hungry equipment (e.g., GPS receivers). Others may have virtually no energy impact (e.g., listening to extant packets already being transmitted for other reasons).
- *Precision*—either the dispersion among a group of peers, or maximum error with respect to an external standard. The precision might be as fine as microseconds (e.g., coherent signal processing on audio signals) or as coarse as seconds (e.g., tracking a slow-moving human).
- *Lifetime*—the duration for which nodes are synchronized. This might be nearly instantaneous (e.g., to compare views of a single event from multiple vantage points), as long-lived as the network (to track the motion of an object through a sensor field), or persistent forever (e.g., UTC).
- *Scope and Availability*—the geographic span of nodes that are synchronized, and completeness of coverage within that region. The scope might be as small as a pair of nodes exchanging data, or as large as the entire network.
- *Cost and Size*. These factors can make a scheme a nonstarter. It is unreasonable to put a $100 GPS receiver or a $1000 Rubidium oscillator on a disposable sensor node that would otherwise cost $10, or on dust-mote sized nodes.

The exact requirements for WSN time synchronization along these axes can not be characterized in general. The requirements are highly application-domain specific and vary over time in unpredictable ways, since they are influenced by the sensed phenomenon.

Given these new challenges, are traditional time synchronization schemes the best choice for this new domain?

3. TRADITIONAL NETWORK TIME SYNCHRONIZATION

Over the years, many protocols have been designed for maintaining synchronization of physical clocks over computer networks [6, 14, 21, 27]. 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 [21] stands out by virtue of its scalability, robustness to various types of failures, self-configuration, security in the face of deliberate sabotage, and ubiquitous deployment. For decades, it has kept the Internet's clocks ticking in phase. Many people in the WSN research community often ask: "Why not use NTP here, too?" At least one research group has moved in this direction, implementing an NTP-like time service over small wireless sensors [10]. But is this truly the best choice? Many of the assumptions that NTP makes, while true in the Internet, are not true in sensor networks. We explore some of these differences below.

3.1 Energy Awareness

As explained in section 1, energy efficiency is a major concern in a WSN. The energy constraints violate a number of assumptions routinely made by classical synchronization algorithms:

- Using the CPU in moderation is free.
- Listening to the network is free.
- Occasional transmissions have a negligible impact.

These assumptions are true in traditional networks and consequently have become fundamental to schemes such as NTP. For example, NTP assumes that the CPU is always available, and performs frequency discipline of the oscillator by adding small but continuous offsets to the system clock. In addition, NTP makes no effort to predict the time at which packets will arrive; it simply listens to the network all the time. And, while it is conservative in its use of bandwidth, it assumes a continuous ability to transmit packets. (It can "free-run" without network access, but requires a significant time with network access restored before it achieves its original accuracy again.)

[22] describes why most of the above assumptions do not hold in a WSN. In a low-power radio, listening to, sending to, receiving from the network all require significant energy compared to the overall system budget. CPU cycles are also a scarce resource; the limited energy mandates the use of slow processors which spend most of their time powered down (awakened by a pre-processor after an event of interest).

3.2 Infrastructure-Supported vs. Ad Hoc

NTP allows construction of time synchronization hierarchies, each rooted at one of many canonical sources of external time in the Internet. The canonical sources ("Stratum 1" servers, in NTP terminology) are synchronized with each other via a variety of "out of band" mechanisms—for example, radio receivers for time signals from the Global Positioning System [16], or the WWVB radio broadcast [3]. This infrastructure provides a common view of a global timescale (UTC) to the Stratum 1 servers throughout the Internet. Consequently, nodes throughout the Internet enjoy being synchronized to a single, *global* timescale while rarely finding themselves more than a few hops away from a *local* source of this canonical time.

WSNs, on the other hand, may often consist of large-diameter networks *without* an external infrastructure. Often it is not an option to equip sensor nodes with receivers for "out of band" time references. GPS, for example, is expensive both in terms of energy consumption and component cost, since it needs high-performance digital signal processing capabilities. Moreover, it requires a line of sight to the GPS satellites—which is not available inside of buildings, beneath dense foliage, underwater, on Mars, etc.

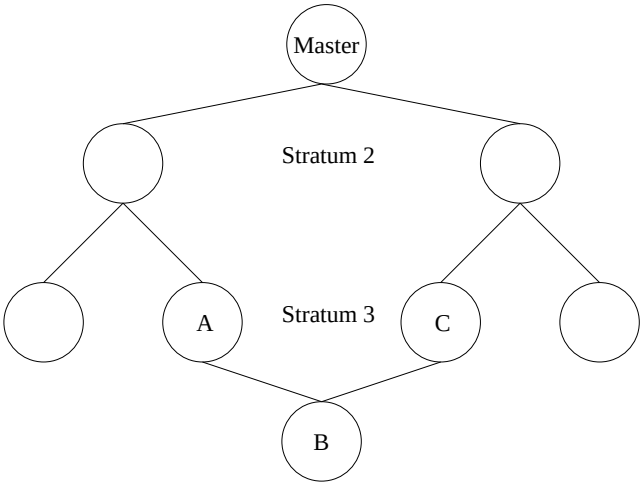

Figure 1: A global timescale can lead to poorly synchronized neighbors, if the neighbors are far from the master clock and have uncorrelated loss due to divergent synchronization paths.

In this scenario, NTP-style algorithms must create a hierarchy rooted at a single node that is designated as the system's master clock. Even assuming we have an algorithm that automatically maintains such a hierarchy in the face of node dynamics and partitions, there is still a fundamental problem: with a single source of canonical time, most nodes will be far away from it. Nodes that are far away from the master clock will be poorly synchronized to the global timescale.

This is a particularly bad situation in a WSN, where nodes closest to each other are often the ones that need the most precise synchronization—e.g., for distributed acoustic beamforming. Consider the scenario shown in Figure 1. Nodes A, B, and C are close to one another, but far away from the master clock. In a scheme such as NTP, B will choose either A or C as its synchronization source. Either choice will lead to poor synchronization when sharing data with the opposite neighbor. For example, if B synchronizes to C, its synchronization error to A will be quite large; the synchronization path leads all the way to the master and back. As we will discuss in Section 4.2, these constraints suggest that WSNs should have *no global timescale*. Instead, we propose that each node in an WSN maintain an *undisciplined* clock, augmented with relative frequency and phase information to each of its local peers.

3.3 Static Topology vs. Dynamics

Although the Internet suffers from transient link failures, the topology remains relatively consistent from month to month, or year to year. Typically, NTP clients are manually configured with a list of "upstream" sources of time. Although NTP automatically uses statistical means to decide on the best of its given options, it still depends on being configured with some notion of which nodes are peers and which lie upstream.

The network dynamics in WSN prevent such a simple kind of static configuration. Moreover, the need for unattended operation of WSN prevents a manual configuration of individual nodes.

3.4 Connected vs. Disconnected

Node mobility, node failures, and environmental obstructions cause a high degree of dynamics in a WSN. This includes frequent network topology changes and network partitions. Data may still flow through the network despite these partitions, as mobile nodes transport information by physically moving within the network. However, the resulting paths of information flow might have unbounded delays (depending on the movement of the node relaying the information) and are potentially unidirectional, since there might not be any nodes moving in the opposite direction.

This kind of message relaying might seem like an unlikely case. However, in a sparse WSN where sensor nodes are attached to moving objects or creatures (e.g., humans, animals, vehicles, goods) or deployed in moving media (e.g., air, water) this is a major mode of communication [2, 5, 12, 18]. Grossglauser and Tse [13] even show that the communication capacity of a WSN approaches zero with increasing node density unless messages are being relayed in this way.

As we will show below, message relaying is a serious problem for traditional clock synchronization algorithms, since they rely on two important assumptions:

1. Nodes are connected before the time they need to be synchronized.

2. The message delay between two (not necessarily adjacent) nodes to be synchronized can be estimated over time with high precision.

Consider for example figure 2, which models a water pollution monitoring WSN deployed in a river. At real–time t_1 device 1 detects an oil stain. At t_2 device 2 detects the same oil stain. At t_3 device 2 passes by device 3, a communication link is established, and E_2 is sent to device 3. At t_4 device 1 passes by device 3, a link is established, and E_1 is sent to device 3.

If device 3 wants to determine direction of movement and size of the oil stain, it has to determine whether E_1 happened after E_2 and the time difference between E_1 and E_2. This scenario presents a serious problem for classical clock synchronization algorithms that assume that the device's clocks will be synchronized *a priori* when they sense events E_1 and E_2. However, as shown in figure 2, there is no way for nodes 1 and 2 to communicate for all $t \leq t_3$, which makes clock synchronization of nodes 1 and 2 impossible before E_1 and E_2 are sensed. This violates the first of the above assumption made by classical clock synchronization algorithms.

Even at time t_4, where an unidirectional delayed message path from node 2 to node 1 via node 3 exists, clock synchronization of nodes 1 and 2 seems almost impossible with traditional algorithms. The path is unidirectional and arbitrarily delayed—wreaking havoc with traditional clock synchronization algorithms that assume they can estimate the message delay over time (or, that assume the delay is negligible), thus violating the second of the above assumptions. The highly unreliable communication in WSNs further contributes to arbitrary delays on multihop paths.

4. DESIGN PRINCIPLES FOR WSN TIME SYNCHRONIZATION

Having described the shortcomings of traditional time synchronization schemes in the previous section, we can now begin to formulate requirements and new directions for time synchronization in WSNs. There are not yet any proven solutions for time synchronization in deployed WSNs. However, the authors have developed techniques which might prove helpful in solving this problem [7, 8, 23]. These techniques aim to build a synchronization service that conforms to the requirements of WSNs:

- *Energy efficiency*—the energy spent synchronizing clocks should be as small as possible, bearing in mind that there is significant cost to continuous CPU use or radio listening.
- *Scalability*—large populations of sensor nodes (hundreds or thousands) must be supported.
- *Robustness*—the service must continuously adapt to conditions inside the network, despite dynamics that lead to network partitions.
- *Ad hoc deployment*—time sync must work with no *a priori* configuration, and no infrastructure available (e.g., an out-of-band common view of time).

4.1 Multi-Modal, Tiered, and Tunable

The services provided by various proposals for WSN time synchronization fall into many disparate points in the parameter space we described in Section 2 (energy, precision, scope, lifetime, and cost). Each scheme has tradeoffs—no single method is optimal along all axes. For example:

- Typical GPS receivers can synchronize nodes to a persistent-lifetime timescale that is Earth-wide in scope to a precision of 200ns [19]. However, the receivers can require several minutes of settling time, and may be too large, costly, or high-power to justify on a small sensor node. In addition, the GPS infrastructure is not always available (§3.2).
- Römer's scheme described in [23] achieves 1ms precision, creates an instantaneous timescale with little overhead, and works on unidirectional links. However, the synchronization is localized and rather short-lived.
- Elson's RBS [8] can achieve 1μs precision and sufficient frequency estimates to extend the timescale for several minutes. It synchronizes all nodes within a broadcast domain. However, it requires a bidirectional broadcast medium and several packet exchanges.
- The multihop extension to RBS described in [8] allows the timescale to be extended across multiple broadcast domains, but at the cost of degraded accuracy.

None of these methods can be considered the *best*; each has advantages and disadvantages. The details of a particular application and hardware will dictate the method that should be used in each situation.

Still more options arise when several methods are composed into a multi-modal system. For example, we might equip a small portion of nodes with more expensive high-stability oscillators, and

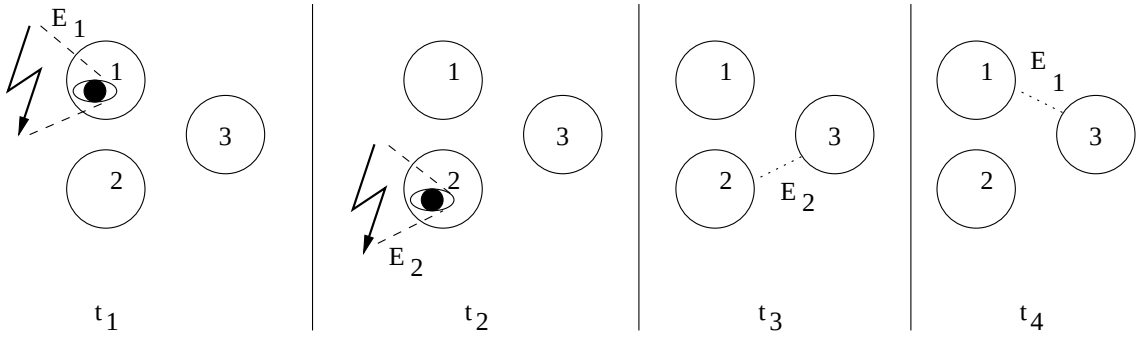

Figure 2: A disconnected network leading to time synchronization problems in a WSN

use RBS to allow nearby nodes to estimate their own frequency with respect to the reference [8]. This type of tiered architecture is analagous to the memory hierarchy found in modern computers (registers, memory cache, main memory, disk), where the goal is to build a system that appears to be as fast as the registers, but as large and cheap as the disk.

Ideally, we would like to have a large enough palette of methods available so that we can choose an overall scheme that is both *necessary and sufficient* for the application on all axes. Unnecessary synchronization wastes resources; insufficient synchronization leads to poor application performance. To this end, it is also important that WSN synchronization be *tunable*—providing adjustable parameters that allow a closer match between the type of synchronization needed and that which is provided.

4.2 No Global Timescale

We argued in Section 3.2 that keeping a global timescale for a large network is only effective when many canonical sources of that timescale are available throughout the network. In the infrastructure-free world of an WSN, where we can not rely on such out-of-band timescale distribution, classical algorithms end up in the situation we illustrated in Figure 1.

Our claim is that the best solution is for each node to keep its own timescale. A node never sets its clock or disciplines its frequency, but rather lets it run at its natural rate. WSN time synchronization schemes—regardless of the underlying method—should only build up a table of parameters relating the phase and frequency of the local clock to other clocks in the system. Local and remote timestamps can then be compared to each other using these parameters for conversion. In fact, time conversion can be built into the packet forwarding mechanism itself. That is, nodes can perform successive time conversions on packets as they are forwarded from node to node—keeping timestamps with respect to the local clock at each hop.

This technique has a number of advantages. First, the synchronization error between two nodes is proportional to the distance between them—not their distance to a master clock, which might be much greater. Second, allowing the local clock to run undisciplined means that each node can enjoy a monotonic clock—a critical feature to many signal processing algorithms. While frequency drift will occur due to the oscillator's instability due to temperature, shock, and voltage variations, there will be no sudden changes in the frequency or phase due to new information arriving at a disciplining algorithm. (Network timesync can produce an estimate of the oscillator's frequency relative to an SI second if needed for data analysis.) Finally, an undisciplined clock requires no continuous corrections to the clock by the CPU or kernel, as are required by algorithms such as NTP. This is important for energy conservation, as we saw in Section 3.1.

4.3 Post-Facto Synchronization

Traditional time synchronization schemes synchronize node clocks *a priori*; clocks are pre-synchronized when an event occurs and is timestamped. As we saw earlier, this causes problems with message relaying and makes it hard to exploit time-variable and unpredictable application knowledge. In contrast, we advocate *post-facto synchronization*, where clocks run unsynchronized at their own natural rates. When timestamps from different clocks need to be compared, they can be reconciled after the fact [7]. This removes the need to predict application requirements in advance; instead, synchronization energy is only expended after an event of interest has occurred. Also, this approach enables support for message relaying, since it does not require network connectivity between event-generating nodes.

Time synchronization is comparable in some sense to routing in ad hoc networks. There, *proactive* routing establishes and maintains routes between nodes in advance, whereas *reactive* routing only establishes routes on-demand between pairs of nodes that want to communicate.

4.4 Adapt to the Application

In Section 4.1, we argued that scalable and energy-efficient WSN time synchronization should be achieved by closely matching the application requirements along axes such as scope, lifetime, and precision. We have also seen a number of techniques that provide service in different parts of this space. However, application requirements vary over time and are in general not predictable, since they depend on the sensed phenomena. Choosing and tuning a necessary and sufficient form of synchronization is a non-trivial problem. To some degree, the application requirements of time syncronization must be built in at design-time. However, dynamics of the application and the environment are likely to dictate that automatic adaptation at run-time is also necessary.

In some cases, the application can explicitly describe its requirements to the synchronization subsystem: the precision required, the peers to which synchronization is needed, and so forth. There are also cases where the synchronization subsystem can deduce application requirements implicitly. For example, data flows might imply the scope and lifetime of needed synchronization.

Once the requirements are known, synchronization should *adapt* to them. For example, the number of synchronization packets sent can be varied, trading energy for precision if dictated by the application. This exemplifies a parameterizable or *adaptive fidelity* algorithm [9]. The synchronization system might even choose from a set of synchronization algorithms with differing characteristics depending on the application requirements.

4.5 Exploit Domain Knowledge

Much of the design of the Internet—and, in fact, the Internet Protocol (IP) itself—is meant to put a consistent interface on top of a heterogeneous and inconsistent tapestry of underlying transport media and protocols. NTP shares a similar philosophy: it makes a reasonable set of "lowest common denominator" assumptions about the environment in which it expects to operate. In the Internet, this is the right choice: it has allowed NTP to become deployed nearly ubiquitously, despite the wide variety of processors, oscillators, network media, node topologies, and cross-traffic it encounters.

The disadvantage of such a design is that it precludes the system from taking advantage of any special features that might be available. In a WSN, where we are often trying to squeeze every possible resource from the system, it may not be feasible to give up performance for the sake of generality. It often makes sense for each application to take advantage of whatever special features are available at every layer of the system.

For example, the inherent properties of a some communication media can be leveraged for time synchronization. In 802.11 networks, Reference-Broadcast Synchronization (RBS) has been shown to achieve far better precision than NTP by exploiting the fact that it has a physical-layer broadcast channel [8]. In time-division based MAC layers, some form of synchronization already exists between radios, and can often be accessed by a synchronization process on the CPU [25]. Some radio standards such as Bluetooth [30] provide a separate synchronous communication channel with low delay jitter, which can be used for exchanging synchronization pulses.

Time synchronization can also use domain knowledge about the application. For example, Römer's scheme [23] piggybacks round trip time measurements to ordinary data packets sent by other processes. This achieves time synchronization without imposing any additional load on the network. Similarly, RBS can work by observing extant broadcasts in the system instead of sending its own special packets.

5. CONCLUSIONS

Physical time synchronization is a crucial component of wireless sensor networks. In this paper, we described some of the important applications of synchronized time in a WSN and their characteristics along axes such as energy use, scope, precision, lifetime, and cost. We argue that traditional time synchronization schemes like NTP can not be applied in this new domain, where many assumptions have changed. Unlike in wired networks, energy is finite; infrastructure is unavailable; topologies are no longer static or even connected. Based on our experience with the development of time synchronization schemes for WSNs, we proposed some design principles: use multiple, tunable modes of synchronization; do not maintain a global timescale for the entire network; use post-facto synchronization; adapt to the application, and exploit domain knowledge.

Sensor networking is still a young field; none of these principles have yet been proven in the way that NTP has proven itself in the Internet. However, we believe they provide a useful framework to guide the design of WSN time synchronization as the field evolves.

6. REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks*, 38(4):393–422, March 2002.

[2] A. Beaufour, M. Leopold, and P. Bonnet. Smart-Tag Based Data Dissemination. Submitted for publication, June 2002.

[3] R.E. Beehler. Time/frequency services of the U.S. National Bureau of Standards and some alternatives for future improvement. *Journal of Electronics and Telecommunications Engineers*, 27:389–402, Jan 1981.

[4] 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. Available at http://www.isi.edu/scadds/papers/CostaRica-oct01-final.ps.

[5] Z. D. Chen, HT Kung, and D. Vlah. Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways. In *MobiHoc 2001*, Long Beach, USA, October 2001.

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

[7] J. Elson and D. Estrin. Time Synchronization for Wireless Sensor Networks. In *2001 International Parallel and Distributed Processing Symposium (IPDPS), Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, San Francisco, USA, April 2001.

[8] J. Elson, L. Girod, and D. Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. Submitted for publication, May 2002.

[9] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next Century Challanges: Scalable Coordination in Sensor Networks. In *MobiCom 99*, Seattle, USA, August 1999.

[10] Saurabh Ganeriwal, Ram Kumar, Sachin Adlakha, and Mani Srivastava. Network-wide time synchronization in sensor networks. Technical report, Networked and Embedded Systems Lab, Elec. Eng. Dept., UCLA, April 2002.

[11] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks. Submitted for publication, February 2002.

[12] N. Glance, D. Snowdon, and J.-L. Meunier. Pollen: using people as a communication medium. *Computer Networks*, 35(4):429–442, 2001.

[13] M. Grossglauser and D. Tse. Mobility Increases the Capacity of Ad-hoc Wireless Networks. In *INFOCOM 2001*, Anchorage, USA, April 2001.

[14] 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.

[15] 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 Intl. Conf. on Mobile computing and networking*, pages 271–278, 1999.

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

[17] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(4):558–565, July 1978.

[18] Q. Li and D. Rus. Sending Messages to Mobile Users in Disconnected Ad-Hoc Wireless Networks. In *MobiCom 2000*, Boston, USA, August 2000.

[19] 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.

[20] F. Mattern. Virtual Time and Global States in Distributed Systems. In *Workshop on Parallel and Distributed Algorithms*, Chateau de Bonas, October 1988.

[21] 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.

[22] G.J. Pottie and W.J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):551–558, 2000.

[23] K. Römer. Time Synchronization in Ad Hoc Networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01)*, Long Beach, CA, October 2001. www.inf.ethz.ch/vs/publ/papers/mobihoc01-time-sync.pdf.

[24] K. Römer. Robust and Energy-Efficient Temporal-Order Event Delivery in Wireless Sensor Networks. Submitted for publication, March 2002.

[25] I. Rubin. Message Delays in FDMA and TDMA Communication Channels. *IEEE Trans. Communin.*, COM27(5):769–777, May 1979.

[26] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, pages 16–27, October 2000.

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

[28] John R. Vig. Introduction to Quartz Frequency Standards. Technical Report SLCET-TR-92-1, Army Research Laboratory, Electronics and Power Sources Directorate, October 1992. Available at http://www.ieee-uffc.org/freqcontrol/quartz/vig/vigtoc.htm.

[29] 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.

[30] Bluetooth SIG. www.bluetooth.org.