Low-power Clock Synchronization using Electromagnetic Energy Radiating from AC Power Lines

Anthony Rowe Vikram Gupta Ragunathan (Raj) Rajkumar
Electrical and Computer Engineering Department
Carnegie Mellon University
{agr,vikramg,raj}@ece.cmu.edu

Abstract

Clock synchronization is highly desirable in many sensor networking applications. It enables event ordering, coordinated actuation, energy-efficient communication and duty cycling. This paper presents a novel low-power hardware module for achieving global clock synchronization by tuning to the magnetic field radiating from existing AC power lines. This signal can be used as a global clock source for battery-operated sensor nodes to eliminate drift between nodes over time even when they are not passing messages. With this scheme, each receiver is frequency-locked with each other, but there is typically a phase-offset between them. Since these phase offsets tend to be constant, a higher-level compensation protocol can be used to globally synchronize a sensor network. We present the design of an LC tank receiver circuit tuned to the AC 60Hz signal which we call a *Syntonistor*. The Syntonistor incorporates a low-power microcontroller that filters the signal induced from AC power lines generating a pulse-per-second output for easy interfacing with sensor nodes. The hardware consumes less than $58\mu W$ which is 2-3 times lower than the idle state of most sensor networking MAC protocols. Next, we evaluate a software clock-recovery technique running on the local microcontroller that minimizes timing jitter and provides robustness to noise. Finally, we provide a protocol that sets a global notion of time by accounting for phase-offsets. We evaluate the synchronization accuracy and energy performance as compared to in-band message passing schemes. The use of out-of-band signals for clock synchronization has the useful property of decoupling the synchronization scheme from any particular MAC protocol. Our experiments show that over a 11 day period, eight nodes distributed across the floor of the CIC building on Carnegie Mellon's campus remained synchronized on an average to less than $1ms$ without exchanging any radio messages beyond the initialization phase.

SenSys'09, November 4–6, 2009, Berkeley, CA, USA.

Categories and Subject Descriptors

D.4.m [**Information Systems Applications**]: Miscellaneous

General Terms

Hardware Clock Synchronization, Sensor Networks

Keywords

Synchronization, Wireless Sensor Networks

1 Introduction

Clock synchronization is an important service for wireless sensor networks. Multiple existing MAC protocols [1, 2, 3] use clock synchronization to save energy through coordination of wakeup times with neighbors. Other applications [4, 5, 6] require clock synchronization for ordering sensed events or to coordinate actuators. Existing synchronization mechanisms consume a significant amount of energy due to the periodic exchange of messages used to adjust local clocks. In this paper, we present a novel hardware clock synchronization solution that allows nodes in the network to receive a stable global clock tick by tuning to the electromagnetic energy (EM) radiating from existing AC power lines. Unlike other hardware-based clock synchronization solutions that require transmitters, our system is able to utilize the existing signals radiating from AC power-lines tens to hundreds of meters away. In contrast to systems like the WWVB atomic clock broadcast and GPS time receiver, our system operates extremely well indoors and around the periphery of buildings. Electromagnetic interference from power lines is so ubiquitous that most electronic devices including many radios are specifically designed to reject 50 and 60 Hz noise. We present the design of a low-cost and low-powered device called a *Syntonistor* which uses these induced signals to provide clock synchronization in wireless devices. By leveraging this highly available common clock source, we can now provide synchronization using significantly less power than existing message passing solutions. Furthermore, this source continues to operate even when nodes become disconnected from the network for extended periods of time.

When an alternating current flows through a conductor it produces an electromagnetic field. This field propagates indefinitely throughout space around the source. A changing magnetic field can impart its energy on a conductor in its vicinity through the laws of induction. If the magnetic

field is picked up by a conductor, like a coil of wire, the primary frequency of the induced signal will match that of the source. Typically AC wires run as parallel pairs and hence most of the magnetic fields cancel out. However, imbalances in wires and stray currents flowing on ground lines as well as through appliances produce a significant magnetic field. The amplitude of this field is generally small, but if sufficiently amplified, one can reconstruct the frequency of the original source. In this paper, we use this physical phenomena to generate a global clock signal that can be used by sensor nodes or other devices that desire clock synchronization.

In [7] the authors measure and model the magnetic fields produced by various power line configurations to evaluate the potential health hazard on humans. Their measurements show that the magnetic field strength near overhead transmission lines can be as strong as 17 milli-gauss and drops down to a still detectable 3-4 milli-gauss at 60 meters away. This indicates that in most outdoor urban environments it is still possible to detect these magnetic fields in and around buildings. In [8], the authors measure typical magnetic field values in homes ranging from .001 gauss to 10 gauss near appliances and as high as 100 gauss in industrial settings. By contrast, the earth's natural magnetic field at 60Hz is much weaker $2 \cdot 10^{-7}$ gauss making these artificial signals relatively simple to detect.

The frequency of an AC power line typically has a stability of about $5 \cdot 10^5$ [9]. In the past, devices like alarm clocks and home appliances have used a direct connection to the power-line as a source for keeping wall-clock time. In order for power to be delivered efficiently across the country, the phase difference between any two points should remain fairly constant. [9] shows the differential delay to have a stability of 1 part in 10^8 over a 24 hour period. Hence, the power grid is phase-coherent. In the United States, there are four main power grids that cover the entire country. Most buildings are supplied with multiple phases of power. This in combination with the orientation between our receiver and nearby dominant magnetic fields, the detected signal will lead or lag with respect to the original signal resulting in a a phase offset. This means that our receivers achieve *syntonization* with each other. Syntonization is defined as when two clocks are frequency locked, but they may have a phase offset (hence we call our receiver a *Syntonistor*). Since the power lines act as a global broadcast, even if the frequency shifts, each node in the network still receives a common global clock tick. This means that after initialization, all clock rates are identical and do not drift.

Typically, power lines operate at either 50 or 60Hz. The synchronization accuracy, however, is not limited by the period of the signal. The limiting factor with respect to pairwise clock synchronization accuracy results from the difference and uncertainty in reception times of the transmission. In our approach, we use the zero-crossing time of the rising edge of the power-line sine wave signal as a synchronization point. Hence, synchronization accuracy is a function of the rising-edge jitter and error accumulated during initialization. Power line signals can be detected using multiple methods. Lights sampled at a high frequency often reflect the 60Hz signal from the power that is energizing them. Transformers and other machinery mechanically oscillate creating the common 60Hz hum that can be detected using microphones. In practice, we found that directly detecting the magnetic field is the most reliable and flexible solution, but our protocols can be easily extended to include these other methods of signal acquisition.

The four main challenges associated with this clock synchronization approach are:

- Designing a low-power receiver.
- Robustness to noise and temporary loss of signal.
- Establishing a common time-reference point.
- Determining absolute phase-offset.

Our hardware receiver is designed to amplify and filter the signal induced from 60Hz power lines. We present a clock recovery technique that uses a software Phase Locked Loop (PLL) that runs locally on a low-power micro-controller. Even at 100% duty-cycle, the device consumes less than $58\mu W$ of power. The receiver is packaged as a stand-alone module that can be interfaced with existing wireless sensor nodes providing a pulse-per-second (PPS) output along with an offset error signaling pin. We describe a protocol running on FireFly[10] sensor nodes that uses the output from the Syntonistor to synchronize all nodes in a multi-hop sensor network to a common notion of wall-clock time. Unlike most synchronization protocols, nodes only need to transmit radio messages at initialization or when they detect phase offset errors due to noise spikes or a power outage. We compare the energy requirements of our hardware solution with the message passing overhead of running synchronization on existing MAC protocols. The idle energy of most MAC protocols is 20-50 times greater making this an extremely viable option for low-duty cycle collection of data. Finally, we evaluate the stability, pair-wise jitter and energy performance of the system as compared to in-band message passing protocols.

1.1 Target Applications

The primary applications of this approach are ones in which constant clock synchronization across an indoor wireless sensor network are desired with minimal periodic communication. For example, ultra-low duty cycle networks that collect sensor samples for an extended period (hours or even days) and occasionally transfer batches of data back to a gateway would greatly benefit from this type of design. The sensor collection will always remain tightly synchronized so that events can be ordered. These are ideal characteristics for quiescent alarm / event detection systems. Out-of-band clock signaling also significantly simplifies the implementation of low duty cycle MAC protocols that share common sleep intervals. For example, TDMA-based MAC protocols no longer have to worry about interleaving in-band synchronization data with their normal message passing.

1.2 Organization of the Paper

The rest of this paper is organized as follows. In Section 2 we give an overview of the related work. Section 3 describes the hardware and software design of our syntonization hardware. In Section 4 we propose a simple protocol that can achieve global clock synchronization. Section 5 highlights

the performance of the system and compares energy consumption against message passing protocols. Section 6 describes the limitations and Section 7 provides concluding remarks.

2 Related Work

Time synchronization has been studied extensively in the field of distributed systems. In [11], Lamport presented the seminal work on *logical clocks* for total event ordering in a system. This paper captures a numerical method for ordering events based on time-counter exchanges among devices. Other work has focused directly on establishing a common notion of wall-clock time [12, 13, 14, 15]. The most commonly adopted of these approaches is the Network Time Protocol (NTP). NTP uses round-trip message delay averaging and hierarchical communication between time servers to set times. When changes to time are required, NTP uses a clock-rate adjustment technique ensuring that time remains continuous. For example, if a computer's clock is slightly behind the actual time, it adjusts its mapping of ticks to seconds such that its virtual clock runs faster. We utilize a similar approach in our clock recovery technique and slowly adjust a virtual local clock rate to match the average edge time of our received input signal.

There have been multiple approaches to clock synchronization using message passing in wireless sensor networks. Many of these approaches achieve extremely accurate synchronization, but few evaluate the required energy overhead of their schemes. The reference broadcast synchronization [16] (RBS) scheme uses timestamps exchanged between multiple receivers to eliminate transmission delays. This approach specifically targets the sources of timing jitter associated with wireless devices and averages over multiple transmissions to achieve tight pairwise clock synchronization. The flooding time synchronization protocol [17] and the time-sync protocol for sensor networks [18] (TPSN) use low-level hardware timestamping to eliminate these similar sources of timing jitter. Messages are flooded across the network forming a spanning tree that periodically compensates for drift. Local clock rates are adjusted to help reduce drift. We use a similar approach in our phase offset compensation protocol to initialize our system except that we do not require any clock-rate adjustment. In [19], the authors propose an interesting approach inspired by fireflies and other biological systems that allows groups of nodes to achieve synchronicity through rate adjustment of message passing with their neighbors. Once synchronicity is achieved, a higher-level protocol can be used to set a common wall clock time. In contrast, our scheme does not require periodic node-to-node communication.

The clock synchronization schemes most similar to ours use external hardware to receive global time broadcasts. The WWVB atomic clock broadcast from NIST uses a 50kW radio tower located in Boulder Colorado to transmit a 60Khz time beacon. This system encodes wall-clock time using a pulse width modulated coding scheme with rising edges at the beginning of each second. This system is ideal for outdoor applications within the tower's broadcast range, but the radio transmission does not penetrate far into buildings. The Global Positioning System (GPS) uses precise clock synchronization derived from satellite transmissions for localization. GPS time receivers have commonly been used as sources for NTP servers. The Radio Data System (RDS) uses the sidebands on standard FM radio transmissions to encode data including the time. These receivers typically consume too much energy for use on a node-by-node basis in a sensor network and in the case of GPS and WWVB require direct line of sight with the sky. The RT-Link [20] protocol uses a carrier current AM radio transmitter to send global time beacons to sensor nodes. The system uses a building's wiring infrastructure as an antenna to broadcast an AM radio timing pulse to an external receiver on all nodes. This solution works well for industrial applications, but it requires a centralized radio transmitter which is expensive and often difficult to install. The work in this paper provides a more practical solution for a wider variety of deployments while at the same time decreasing the energy requirements of the receiver hardware.

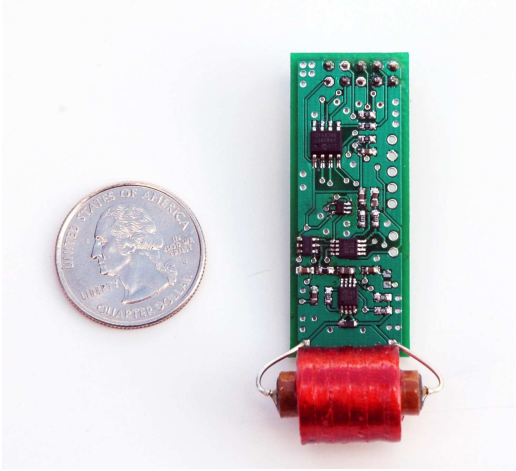


Figure 1. AC power-line EM receiver (Syntonistor) next to a coin.

In [9], the authors study using nearly simultaneous receptions of various sources, both natural and man-made, for synchronization. For example, optical pulsars can broadcast flashes of light simultaneously visible to large regions of the earth. These reception times are known to be nearly simultaneous to all viewers and hence can be used as synchronization points. When taken to an extreme, synchronization points need not even be periodic. Work on chaotic clock synchronization using quantum entangled particles [21] has shown that even common yet entirely random events can be used for clock synchronization without the direct transmission of clock signals. This is a similar principle used in quantum cryptography where keys are based on the random but identical state transitions of entangled particles. The key insight from these approaches is to identify statistically unique sequences of events to use as synchronization points.

3 Syntonistor Design

In this section, we describe the hardware and software components that we designed to accurately detect the energy

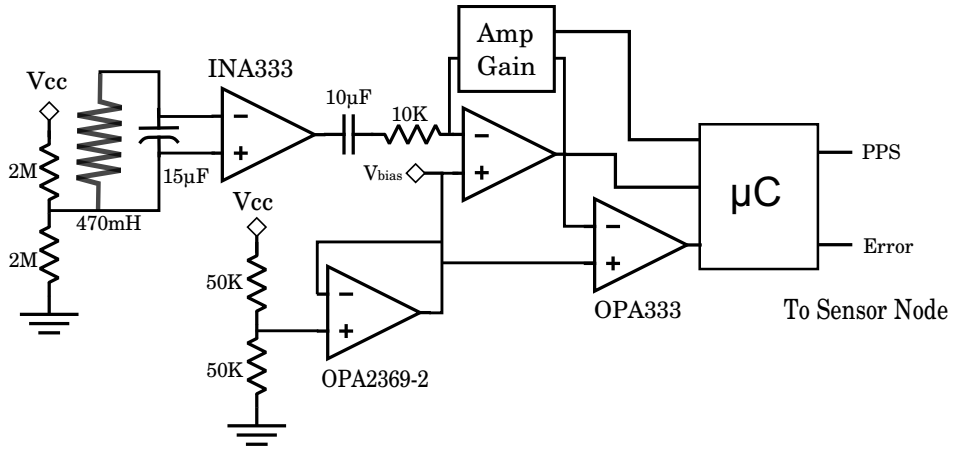


Figure 2. Schematic for wireless power line clock synchronization module.

radiating from AC power lines. The primary challenge in the design was amplifying a signal, which is typically on the order of tens of micro-volts while rejecting noise. The circuit also needs to strike the right balance between energy consumption and performance. In order for the receiver to be practical, it must operate at sufficiently low power that the analog front-ends can be run nearly continuously. The power consumption should be significantly lower than the required power to synchronize the nodes using the already existing radio. Finally, the circuit must be immune to noise and attempt to reduce timing jitter as much as possible so as to provide a reasonable level of synchronization accuracy for sensor networking applications. Other approaches used to detect magnetic fields include Hall-effect sensors, magnetoresistive sensors, and flux-gate sensors. Currently, the readily available components for these sensors tend to be too costly and consume too much power. It is also possible to detect the electric field generated from the power lines, but this tends to be small as compared to the magnetic field.

3.1 Hardware

The circuit diagram in Figure 2 shows the major components of the Syntonistor with the assembled printed circuit board shown in Figure 1. The power-line magnetic field is detected by an antenna composed of an inductor (L) and capacitor (C) tank circuit. The LC component of the circuit is tuned to a resonant frequency of 60Hz as described by Equation (1). When picking the LC combination, there are two factors that impact how well it receives the signal. First, a large valued inductor will have more turns of wire which captures a more intense raw signal. The second factor is the tank circuit's ability to resonate hence collecting energy which naturally amplifies the raw signal. As the L increases, the resistance in the coil also increases. Equation (2) shows how to calculate the Quality Factor of the resonant circuit [1]. As a rule of thumb, when the Q-factor is greater than 1, the circuit will resonate collecting energy and amplifying the raw input signal by approximately the Q-factor. A Q-factor less than 0.5 indicates that the circuit is over-damped and hence losing energy.

There is a trade-off between having a larger L value which directly captures a large raw signal and a smaller L value which uses resonance to amplify the inherently smaller signal. Figure 4 illustrates this trade-off by showing the frequency response of a larger L value with a lower Q-factor as compared to a lower L value with a larger Q. The larger L with the lower Q-factor has a slight benefit in that if the AC signal deviates from 60Hz, the receiver will still be able to detect the signal since it has a wider bandwidth. A larger Q value will tend to reduce jitter in the received signal simplifying any later filtering. Although beyond the scope of this paper, the geometry of the inductor, as well as, the intrinsic properties of its core also play a significant roll in the receiver's ability to detect the signal. In our final design we opted for a configuration with a long and thin ferrite core to collect as much magnetic flux as possible. Figure 3 shows various inductors that we tested. Table 1 shows their performance parameters including their response when excited by a fixed test source. Some of these inductors can be quite large since more turns with thicker wires increases inductance while decreasing resistance. In practice, size and cost constraints should be used to narrow down design parameters. We choose to use a 470 mH coil with a 15 μF capacitor since this combination gave us a large raw signal with a wide bandwidth while still not being over-damped and maintaining a compact form-factor. By adjusting the L and C values, this circuit can easily be adapted to operate in different international power systems (for example, 50Hz in Europe and Asia).

Once the signal is captured, amplification requires an extremely high input impedance, common mode rejection ratio (CMRR) and gain value typically found in an instrumentation amplifier (INA). We chose to use the Texas Instruments (TI) INA333 instrumentation amplifier which is internally composed of two buffering op-amps that feed a third differential op-amp. The output of the INA is passed through an AC-coupled transmission line that is configured as a high-pass filter removing DC bias from the signal. Since we are only interested in removing DC bias, we use a 1.5Hz high-pass filter described by Equation (3) well below our target

[1] Note that the resistance of the L is significant, so this is modeled as a series RLC circuit

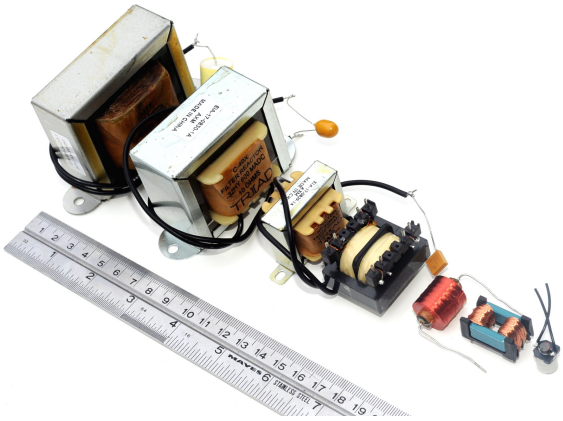


Figure 3. A wide range of possible inductor and capacitor combinations

frequency.

$$f_r = \frac{1}{2\pi\sqrt{LC}} \tag{1}$$

$$Q = \frac{1}{R}\sqrt{\frac{L}{C}} \tag{2}$$

$$f_c = \frac{1}{2\pi RC} \tag{3}$$

The AC signal is further amplified through a second stage using a TI OPA369 micro-power op-amp. A second OPA369 (located on the same IC package) is used to create a low-power bias voltage to center the 60 Hz signal helping to keep it within the linear operating range of the amplifier. A MCP4012 programmable rheostat is used to set the gain of the OPA369 to one of 64 different levels. The output from the OPA369 is passed directly to an analog input on a PIC12F683 micro-controller. The PIC12F683 runs all of the firmware, described in the next section, responsible for producing the outputs that are passed on to the host sensor node. The PIC12F683 also operates as an auto-gain system trying to maintain a peak-to-peak voltage coming out of the amplifier well below saturation voltages by setting the MCP4012 values. The peak-to-peak signal strength along with the rate of change in auto-gain values can be used as a metric for determining the strength of the detected signal. The output from the second stage amplifier is passed to an OPA333 op-amp configured as a zero-crossing detector. Though it consumes less power, the OPA369 is not a suitable amplifier for the zero-crossing detector because of its low slew rate. The slew rate of an op-amp is typically related to its power consumption where lower power means a slower response time to large input changes. We ideally want the amplifier responsible for doing zero-crossing detection to respond to a high frequency transition as quickly as possible to minimize timing jitter. The output from the zero-crossing detector is finally connected to an interrupt pin on the PIC12F683 processor. Table 2 shows the components and the power consumption of each component in the system. Variability in the power measurement is due to dynamic switching power based on auto-gain levels, PIC duty-cycling and input signal strength level. The final output from the PIC processor are GPIO pins that are configured by software as described in the next section.

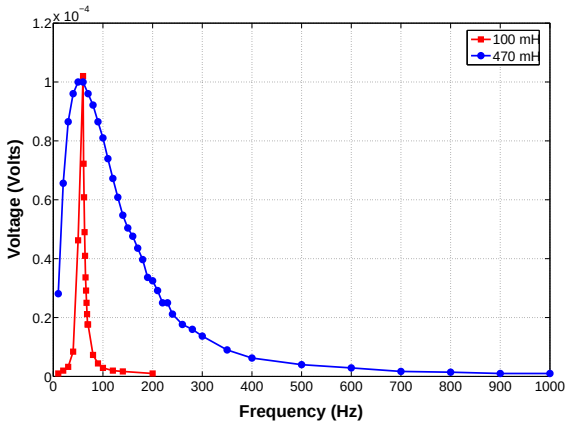


Figure 4. Two different frequency responses

L (mH)	C (μF)	R ($Ohms$)	Q Factor	Test Signal Response (mV)
15000	0.47	1000	5.67	464
1500	4.7	40	14.1	202
320	22	10	12.1	113
220	30	15	5.7	18
470	15	600	0.5	20
100	68	2.8	43	20.2

Table 1. The performance of various LC combinations.

Component	Typical Power (μW)	Max Power (μW)
INA333	40	90
OPA2369	0.84	1.0
OPA333	10	30
MCP4012	1.2	10
PIC12F683	5	19.8
Total	**57.04**	**150.8**

Table 2. This table shows a breakdown of average power consumed by the main hardware components.

3.2 Firmware

The PIC12F683 processor on the Syntonistor board is extremely low-powered and heavily resource constrained with just 128 bytes of RAM and 2048 bytes of FLASH. These scarce resources allow the CPU to operate with an idle and active current consumption of $50nA$ and $11\mu A$ respectively at 2 volts. The firmware running on this processor is responsible for three main tasks. First, the processor must slowly adjust the auto-gain level of the second stage amplifier to ensure that the signal is distinct enough to have noticeable

zero-crossing points while not saturating in the presence of a strong signal. Next, the processor must filter the incoming pulses and generate a stable pulse per second (PPS) output which the sensor node can use for synchronization purposes. We desire a low-frequency output so as to not unnecessarily wake the main sensor node. Finally, the processor must toggle an error bit if it detects that the signal is no longer reliable. In the remainder of this section, we will discuss the various techniques used to achieve these goals.

Electromagnetic interference from the environment causes a significant amount of noise in the raw signal received by the PIC. We provide an in-depth evaluation of this in Section 5. The signal tends to suffer from jitter as well as occasional periods of lost reception. Filtering such a signal in the time domain to recover a clock is commonly solved using a Phase-Locked Loop (PLL). A PLL will generate its own local clock with a period that increases or decreases based on the measured phase difference between its local clock and the input clock. The rate at which the local clock is adjusted to match the input signal is a classical control problem. In our design, we implemented a proportional-integral (PI) PLL controller in software on the PIC processor with the block diagram shown in Figure 5. First, we perform a low-pass filter on the input signal. If an edge arrives too early or late, it is simply ignored. Since our target frequency is relatively constant (we assume that the 60Hz will not rapidly deviate), we chose to trade-off a slow convergence time with increasing stability. In the absence of an input signal, the PLL should be able to operate based on its local clock for a significant period of time. This corresponds to relatively low gains on the proportional (K_P) and integral (K_I) terms in the PI controller, which naturally limits the response time of the system. The main loop of the software PLL samples the incoming signal at 32 KHz and adjusts the local virtual clock by at most a single time quantum per cycle. We tuned the system offline using collected data described in more detail in Section 5. The PLL software is written entirely in hand-coded assembly such that it requires only integer accumulators and gains. At each time step, the firmware monitors the error between the last set of pulses and the current virtual clock edge. If this error exceeds a threshold for a certain period of time, the PIC will raise the error pin going to the sensor node indicating that the PLL is no longer locked. A similar operation is performed if no input signal is detected for a long enough duration. The primary output from the PLL is a PPS signal with a 50% duty-cycle that toggles whenever an internal counter is reached. This error signal could be due to a change in the building's magnetic field and hence should be used to alert the protocol that it may need to re-synchronize with the network.

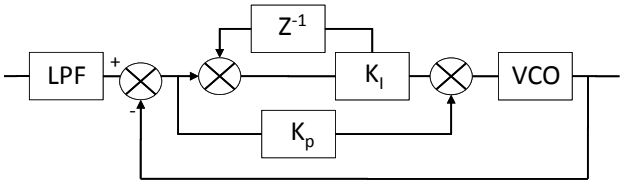


Figure 5. Block diagram for PLL clock recovery system.

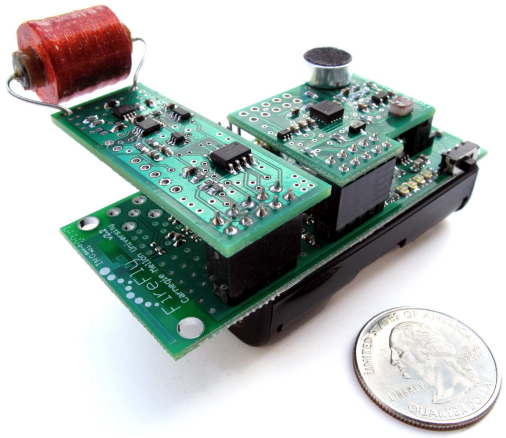


Figure 6. The Syntonistor attached to a FireFly node.

Hardware PLLs are common in radios that operate at higher frequencies. We were unable to find a suitable hardware solution based on readily available parts that operated at below 100Hz for less power than the PIC processor. The software solution also provides us with the flexibility to modify how the PPS signal and error conditions are generated.

4 Synchronization Protocol

Even with globally rate-adjusted clocks, we still have challenges at the protocol layer associated with:

- Determining a common starting point in time,
- Determining phase-offsets between neighbors, and
- Recovering from errors when synchronization fails.

In this section we will discuss a protocol implemented on FireFly [10] wireless sensor nodes running the nano-RK [22] operating system that achieves global clock synchronization using the Syntonistor. Figure 6 shows the Syntonistor attached to a FireFly node that is also equipped with a basic sensor board. The FireFly node has an ATmega1281 processor, TI CC2420 radio and an internal PCB antenna. We use an existing LPL-CSMA [23] protocol for exchanging the initialization messages.

4.1 The Protocol

The protocol begins when a master node broadcasts a message at its rising PPS edge that contains its wall-clock time. The message is flooded across the network using the CC2420 radio timestamped at the lowest level to remove uncertainty as described in [18],[16] and [17]. Adding a timestamp to the message immediately before transmission removes timing uncertainty from potential radio packet collisions. Each sensor node maintains a timer containing the amount of time that has expired since its last PPS rising edge. When a node receives a clock synchronization message, it notes the timestamp of the message from the master as well as the current timestamp computed from the previous hop. The receiving node must then record its current phase off-

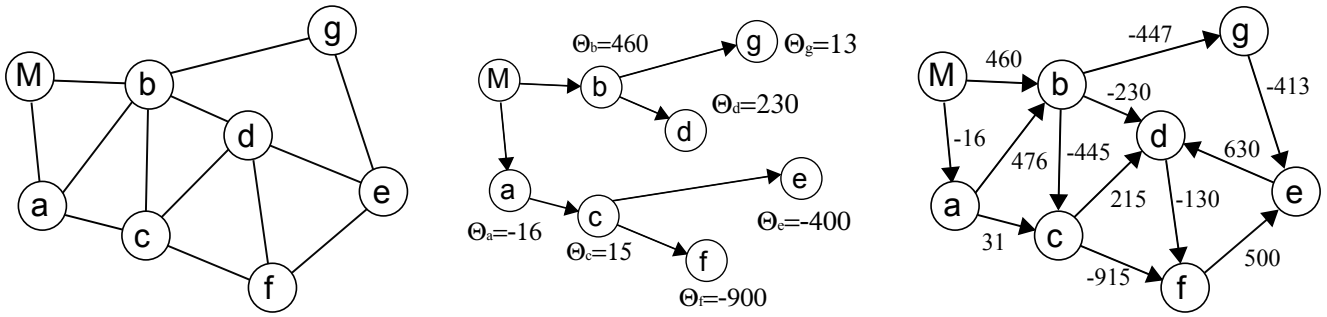


Figure 7. Operation of initialization phase of phase offset calculation. (a) shows a set of nodes. (b) shows a spanning tree used to determine offsets from a master clock.(c) Phase offset values between sets of nodes that can be used for error checking.

set from its local PPS signal and subtract the communication delay that was accumulated during the flooding. Figure 9 shows how message propagation delay can be reduced by removing constant header offsets from packets and triggering on the start of frame delimiter provided by the CC2420 hardware. Figure 10 shows a distribution of radio pulse times recorded between a transmitter and two receivers. We see a worst-case jitter of approximately $6\mu s$ per hop. This is consistent with values seen by other in-band message passing protocols.

After the flood has propagated across the network, each node should maintain a synchronization time point as well as the phase offset between its local PPS signal and that from the master. Timing should now be based on this starting point transmitted by the master, the elapsed time based on PPS pulses and the difference in phase offset between the node and the master. Also, periodic broadcasts with neighbors can be used to build a 1-hop neighbor list containing relative phase offsets between neighbors which, as we will see later in this section, is a useful tool for detecting errors. These 1-hop broadcasts do not need to run continually since they will quickly converge to an average static offset value. Each node in the network will have a synchronization accuracy limited by the local jitter in the Syntonistor which is about $2ms$ as well as the accumulated jitter from the radio communications which is on the order of $6\mu s$ in the worst case. We perform rapid flooding of messages to ensure that only radio jitter accumulates and not the error found in the Syntonistor.

We will now illustrate the protocol using the topology shown in Figure 7(a), where node M is the master and nodes $a, b, c, ..., g$ are other nodes constituting a multi-hop network. The master node transmits a broadcast message (synchronization beacon) to all the neighboring nodes. Upon message reception, the nodes calculate the phase offset with respect to their local PPS signal after subtracting radio propagation delay. The message from the master floods quickly throughout the network with nodes at every hop relaying the beacon to the next hop similar to how the TPSN protocol operates. This beacon quickly spreads to all the nodes in the network, and within the limitations of the radio jitter across n hops, the nodes are able to estimate phase offset of their local PPS signal to that of the master. The timing diagram of the phase calculation for the topology in Figure 7(a) is shown in Figure 8. The phase offset between the master and node i is given by Θ_i. For example, the phase-offset between the master M and node d is $\Theta_d = 230$ ms as shown in Figure 8. This synchronization flooding is only required at network initialization time. If new nodes join the network, they can communicate with existing infrastructure nodes to obtain a phase offset. However, if a new node attempts to synchronize based on a node which was not synchronized from the master, then the jitter from the EM receiver (which is significantly larger than that of the radio) will begin to accumulate. In this case, the node must request a new time update directly from the master. Once the phase offsets with respect to the master are calculated, the relative phase between any two nodes can be calculated by treating their phase offsets from the master as directed vectors, and the phase difference between them is just a vector sum of the two vectors. This is explained in Figure 7(c), where each node has a list of phase offsets from its neighbors. It is interesting to note that this sum of phases along a closed loop in the directed vectors graph is always *zero*. This property provides robustness against a node failure and improves reliability by allowing the nodes to cross-check their phase offset .

4.2 Analysis in Presence of Jitter

We now assess the sources of jitter that could accumulate across a multi-hop network. Let us present an estimate of error in the phase offset calculation by a node at the n^{th} hop from the master, where ε_n is the worst case error and ϕ_n is the phase offset. Also note that the predominant jitter comes from the PPS signal at each node, which should not accumulate over each hop. The phase offset of a node at the n^{th} hop, ϕ_n, is given by:

$$\begin{aligned} \phi_n &= \Theta_n + \delta_{EM} + (n \cdot \delta_{radio}) + (n \cdot t_{relay} \cdot \rho) \\ \Rightarrow \varepsilon_n &= \delta_{EM} + (n \cdot \delta_{radio}) + (n \cdot t_{relay} \cdot \rho) \end{aligned} \quad (4)$$

where Θ_n denotes the actual phase offset of the node from the master, t_{relay} is the time taken by each node to forward the beacon, ρ is the clock drift rate, and δ_{EM} and δ_{radio} are

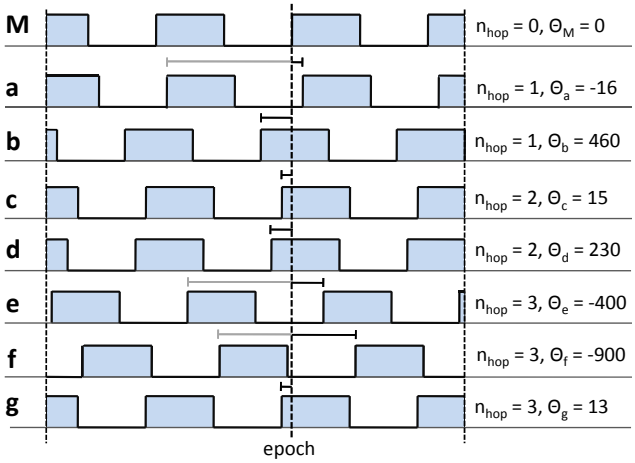


Figure 8. Timing diagram showing relative phase of various nodes

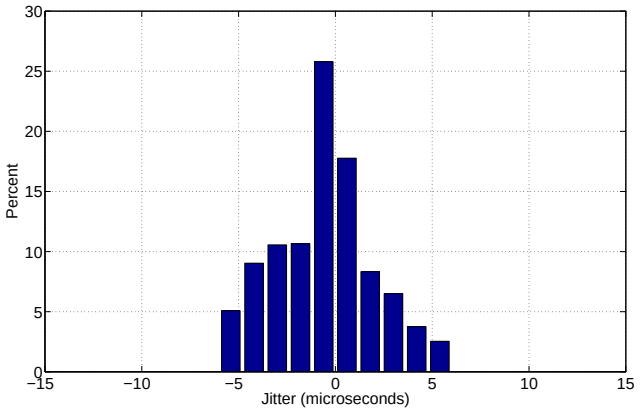


Figure 10. Jitter time between four CC2420 receivers capturing the same transmission.

the jitter in the PPS signal and radio reception respectively. The error in phase estimation is given by Equation (4), where the radio jitter does accumulate over number of hops as the worst case jitter at n^{th} hop is $n \cdot \delta_{radio}$. This accumulation of jitter in radio communication could be controlled by employing better in-band synchronization approaches. It should be noted that δ_{radio} is significantly smaller than δ_{EM} such that nodes would only go out of phase after hundreds of hops. δ_{EM} does not accumulate because the phase offset at each node is calculated with respect to the globally received signal, the phase offset at k^{th} hop is not passed on to the $(k+1)^{th}$ hop, where $k \in 1, 2, ...n$.

4.3 Protocol Features

This protocol provides the following features:

- *Global Synchronization:*

  In our hardware-based approach, each of the nodes is receiving a PPS signal with a constant frequency across the entire network to ensure that the nodes remain synchronized for extended durations without relative drift in their clock frequency. In practice, there may be some variation of the relative phase due to sudden changes such as loss of power or nearby machinery over-powering the existing signal. In our design, this interval may be of the order of many hours or days as justified by the results shown in Section 5.

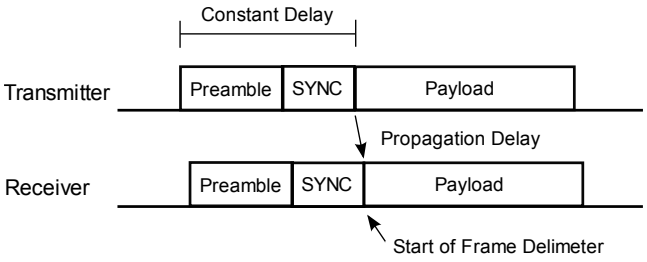


Figure 9. Timing associated with a CC2420 radio packet

- *Scalability:*

  The proposed clock synchronization approach is highly scalable in terms of number of hops and the area of the network. The addition of a new node to the network only involves the calculation of phase offset with respect to a neighbor, and does not require the recalculation of phase offsets of all the nodes. The incoming node can calculate its offset from the neighbor, and can thus estimate its phase value from the master through a vector sum. Care must be taken, however, not to synchronize new nodes from existing new nodes that have not been synchronized directly with the master. This ensures that the PPS jitter does not accumulate.

- *Robustness:*

  One interesting property of this clock synchronization scheme is that all cycles in the network connectivity graph have an absolute phase sum of 0. Any node in the network is able to confirm its phase offset from the master from any other path through another neighbor. In an event of a node failure, any neighboring nodes to this failed node will still be globally synchronized and it will be able to assist any new node joining the network to estimate its phase offset from the master.

4.4 Error Recovery

As described in 3.2, the Syntonistor has the ability to detect when the timing of the incoming signal unexpectedly increases beyond the normal jitter threshold. This could happen for various reasons including a physical change in the environment, a new nearby appliance getting powered up, or even a power outage. In response to these sorts of errors, the Syntonistor will raise its error line which signals to the main sensor node that the PPS value may no longer be accurate. If the error line remains high for a long enough period (based on local clock drift), the node must fall back to an existing software synchronization technique. For example, the node can periodically pass messages with a neighbor to update its clock. Once the power-line signal stabalizes again, the error line from the Syntonistor will transition from high to low. At this point, the node will send a message back to the master

node requesting a new set of flooding time synchronization messages. This high to low transition naturally happens the first time a node is powered on. One possible optimization is for nodes to only request the phase offset based on their neighbors. In practice this works well, however over time this could result in nodes drifting with respect to the master if groups of nodes go in and out of synchronization in lock-step.

5 Experimental Evaluation

In this section, we will evaluate the performance of our clock synchronization hardware solution. We examine the various manifestations of timing jitter in the system which have a direct impact on synchronization accuracy. We evaluate the effectiveness of our software PLL with respect to removing this jitter and coping with noise. Using data collected over an extended period of time, we evaluate pair-wise synchronization accuracy and the stability of a multi-hop network of nodes. Finally, we analyze how our proposed scheme performs with respect to energy as compared to in-band message passing techniques.

5.1 Experimental Setup

We placed eight nodes across one floor of a building with approximately 10000 ft^2 of floor area. The building is structured as an office space with about 100 cubicles in a central area surrounded by private offices. The floor contains a dense number of desktop computers, laptops and various sources of electromagnetic interference like WiFi, mobile phones, microwaves and other electrical appliances. Nodes were placed across the floor and connected with up to $200ft$ cables. During the test period, the building was active with people moving around and working. Nodes were placed approximately 2-3 meters above the ground against walls. As the phase and strength of the induced signal vary with orientation and physical contact of external noise source, like humans passing by, we wanted to mount the devices reasonably out of reach. This is similar to the approach taken when mounting many other devices like wireless access points. Nodes were scattered as far away from each other as possible to ensure that they do not tune to the same local 60 Hz sources. The raw output along with the filtered PLL edge output was captured to a PC using a logic analyzer over long (up to $200ft$) cables. To avoid any interference from common electrical interconnects with the logic analyzer, each node was isolated at the receiver using an opto-isolator. It was observed that even the open-ended long cables capture ambient EM energy on their own and hence the opto-isolator was located as near to the node as possible so that the receiver could not use the long wires as an antenna. The optical isolation circuits and the nodes were powered from separate sets of batteries. It should also be noted that the optical isolator was only used for data collection and is not part of the final design that would be deployed. A schematic diagram of the experimental setup is shown in Figure 11. We chose to use a Saleae [24] PC logic analyzer because it supported a software development kit that allowed customization of the capture software allowing us to stream data for extended periods of time to disk. The analyzer supports sampling rates between 200KHz and 24MHz. Since the PIC processor is internally sampling at only 32KHz, we operate the analyzer at 200KHz providing 5 μs of resolution. For testing radio jitter from the CC2420, we captured data at the full 24 MHz.

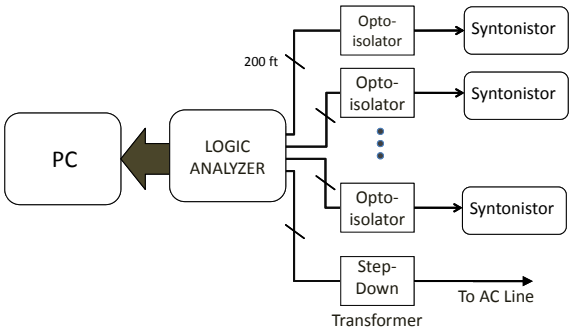


Figure 11. Experimental Setup.

5.2 EM Receiver Jitter

The first and the third waveforms in Figure 13 show a trace of the raw receiver signal captured on an oscilloscope with a 5 second persistence value. We see that the mean value of each edge is *jittering* around each edge. The second and fourth waveforms show the output of the PLL before it is divided down into the pulse-per-second signal. We see that the jitter is drastically reduced. Figure 12(a) shows a histogram of the period of each raw pulse, collected over a 3-hour period, which shows a jitter on the order of 4 ms. In Figure 12(b) we see that this jitter is reduced by the PLL to less than 100 μS. It is observed from the collected data that the instantaneous period of the digital signal at each node closely follows a Gaussian Distribution with a mean close to the 60Hz period of 16.66 ms. This trend in the jitter implies that the error in the period of the signal can be bound by statistically combining multiple instances of the signal. We further describe the operation of our software PLL filtering approach in Section 3.2.

5.3 PLL Performance

When the Syntonistor is first powered up, the PLL initializes at an arbitrary phase relative to the signal provided by the zero-crossing detector. The PLL is a second-order feedback control system, which adjusts the frequency output of a Voltage Controlled Oscillator (VCO) based on the phase difference between the PLL and the current input rising edge. The PLL tracks the phase of the input signal and is able to stabilize slowly to a steady-state phase offset relative to the phase of the input signal. The convergence time and stability of the PLL are determined by two parameters,the proportional factor K_I of the forward transfer function of the PLL and the gain of the error integral term K_P. From control theory, we know that the steady state oscillations increase as K_P is increased. The integral term K_I is responsible for slowly removing steady-state accumulated error from the target.

Figure 14 shows the PLL phase error with respect to the zero-crossing detector's output at each rising edge. The figure shows the initial 120 seconds worth of data to emphasize

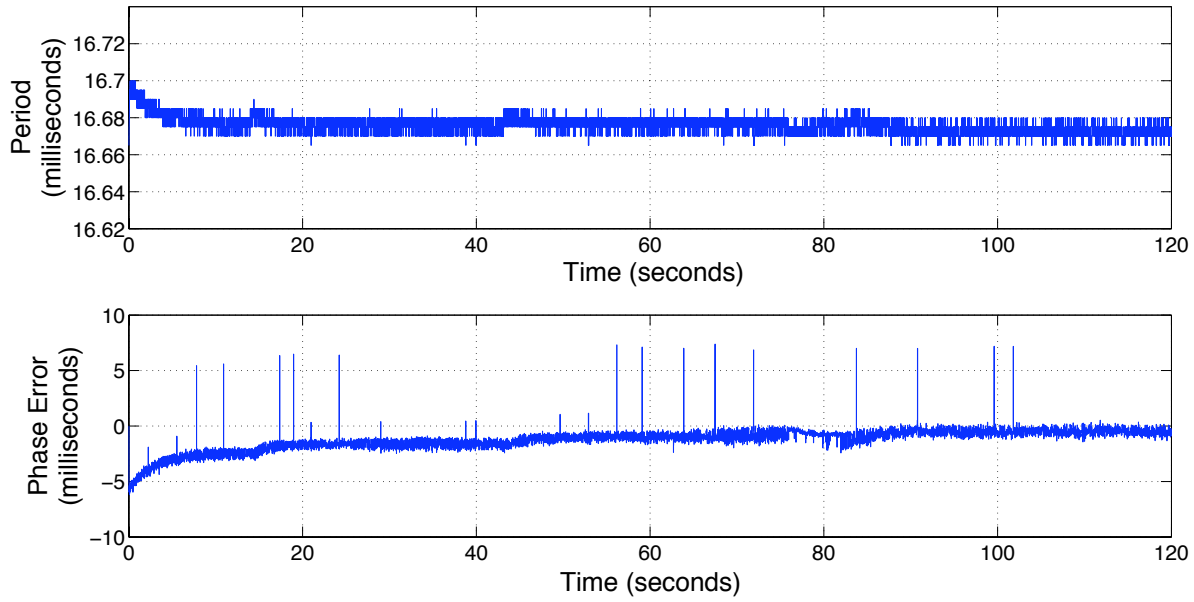
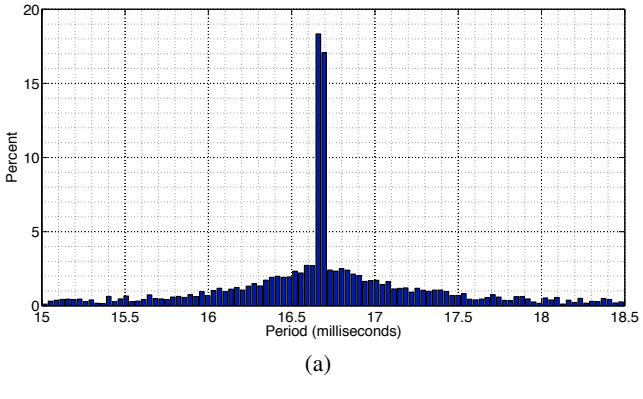
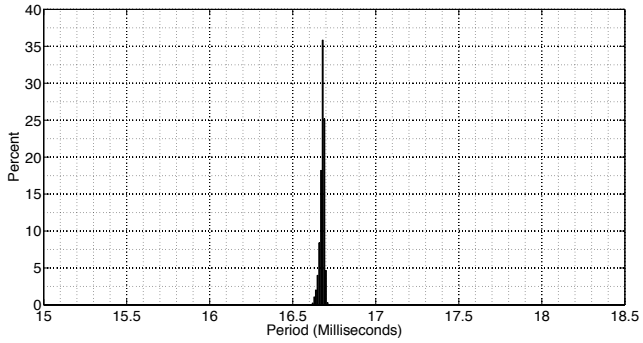


Figure 14. PLL output value and error



(a)



(b)

Figure 12. Jitter in the raw induced signal shown in (a). Jitter after the PLL shown in (b).

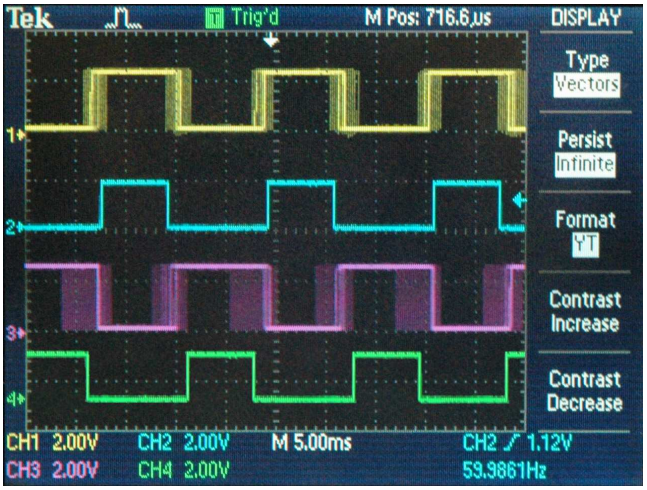


Figure 13. Raw input signal with filtered output signal below for two different nodes.

the convergence of the PLL. We see that the error starts from an initial offset of -5 *ms* and converges closely to 0. The first plot in Figure 14 shows an example of how the PLL's local clock converges which was initialized to a period slightly higher with respect to the 16.66 *ms* signal from the power lines. This phenomena captures the nature of the global rate adjustment that happens on each receiver. Convergence plots for the other nodes in the system were similar to the one we presented with slightly different starting offsets and convergence times. After about 20 seconds, we see that the period of the PLL is locked to its target value. Figure 12(b) shows the updated histogram of period times after the PLL has converged over an entire 11-day period. We see that the timing

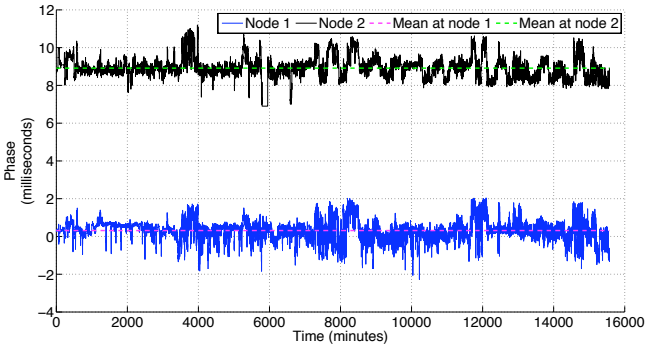


Figure 15. Phase offset between two nodes in different locations over an 11 day period.

jitter was significantly reduced from 4 *ms* to a spread of less than 150 *uS*. During this period,we saw multiple instances where input pulses were dropped for multiple cycles. Due to the slow convergence rate of the PLL, these missing data points are easily reconstructed by the PIC's local clock.

5.4 Phase Stability and Synchronization Accuracy

The primary property of our system is that the phase offset between signals remain consistent after startup. We can evaluate this by looking at the phase offset over time between any pair of nodes. A change in the relative phase offset of each node could result in clock drift over time. Figure 15 shows the pair-wise phase offset between one node and two of its neighbors over a 11-day period. In the case of neighbor node 1, there is very little phase offset between the test nodes because it is likely that they are locked onto the same source. We also see that the phase between the test node and neighbor 2 is offset by more than 8 *ms* yet stable. This indicates that our proposed high-level protocol could account for this offset.

Figure 17 shows the offset-compensated values averaged over all nodes in the system after our initialization protocol. We provide the average total synchronization error as well as the maximum pair-wise error between any two nodes. We see the average error bound to within 2 *ms* and the worst-case error bound to within 6 *ms*. These data are also captured in Figure 16 showing the distribution of error. During

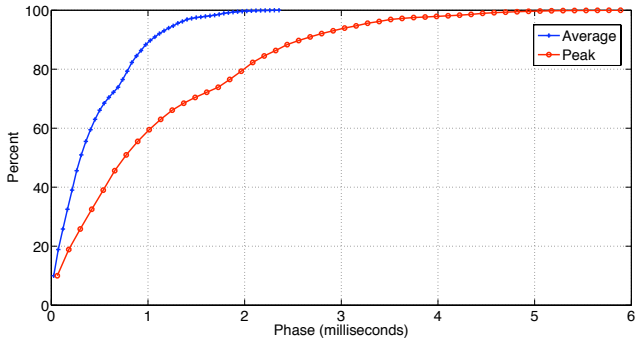


Figure 16. CDF of the average and max synchronization error.

this experiment, no node ever detected that the PLL lost its lock on the signal and hence had to re-initialize itself. In practice, however, events like power outages or nodes placed near severe noise sources would likely cause occasional loss of synchronization. Our system will regain its synchronization when these disturbances are eliminated.

5.5 Interference

Placement of the nodes is critical with respect to finding a strong 60Hz field. In our tests, we tended to place nodes a few meters above the ground and mounted on walls to avoid interference caused by people walking nearby. The output of the first amplification stage in the Syntonistor is passed directly to the sensor node for signal strength estimation. At startup, the sensor node illuminates LEDs to indicate the amplitude of the received signal. If the node is going to be near noisy equipment it is typically a good idea to rotate it until a local maximum signal strength is found, or consider changing the node's location. Generally, if a node has a very strong signal it will be immune to people in the nearby vicinity. In these cases, direct contact with the antenna is required to cause it to lose synchronization. With lower signal intensities, placing a hand a few inches away from the antenna is enough to attenuate the signal. If noisy machinery is activated near the receiver this can also cause interference. Devices like computers, LCD displays and other appliances with lots of high-frequency switching tend to be the worst sources of noise. Though interference is case specific, generally the Syntonistor requires 2-3 feet of clearance from these types of noise sources to correctly operate. In other cases, turning on a nearby device increases the signal strength. For example devices like AV receivers or AC adapters that energize large coils of wires radiate an extremely clean signal that is dominant from up to 5 feet away.

5.6 Synchronization Energy

In this section, we compare the energy consumption of the Syntonistor with various in-band message passing schemes. Throughout this section, we will refer to hardware-specific values found in Table 3 that are typical to most current sensor nodes. These values are based on the FireFly sensor node hardware which uses the ATmega1281 processor and the TI CC2420 radio chip.

We begin our energy evaluation by computing the number of messages required to re-synchronize a clock after it has drifted beyond a particular synchronization accuracy level. To accommodate for multi-hop communication, we assume that each node on average must receive at least one message and transmit at least one message at each synchronization interval. Based on values collected in [16], we increase the message count to three per direction to allow for averaging and to account for potential packet loss and collision. As the desired synchronization accuracy increases, messages must be transmitted more frequently. The following equation shows the average power P_{avg} of this message-passing protocol given a per packet energy E_{radio}, a clock drift rate of ρ and a desired clock synchronization accuracy Π:

$$P_{avg} = \frac{E_{radio} \cdot 2 \cdot \rho}{\Pi} \quad (5)$$

Figure 18 shows a plot of this scheme in contrast to the

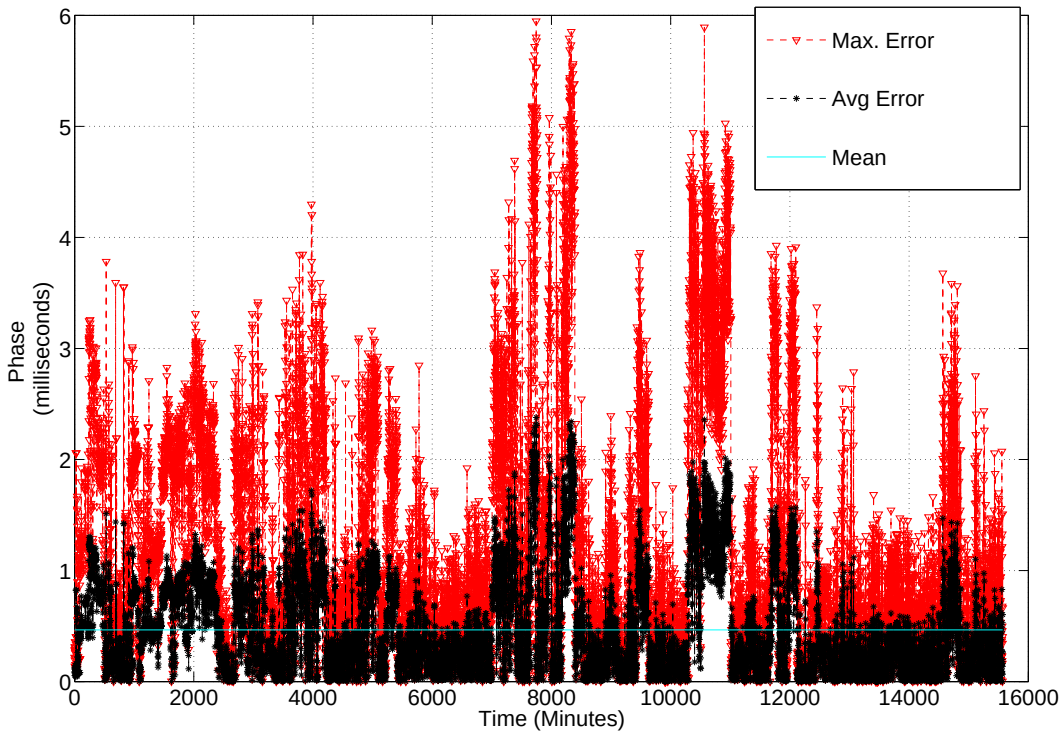


Figure 17. Average and max synchronization error over an 11 day period.

energy required to operate the Syntonistor. The Syntonistor consumes a static average power across its entire range of synchronization. However, at larger synchronization accuracies, it is conceptually possible to duty-cycle the power-line receiver to further reduce the power consumption.

A common approach to clock synchronization uses a regression technique on pair-wise clock drift to perform local rate adjustment. This has been shown in [18] to significantly decrease the synchronization interval. As described by the original authors, the following equation shows how to compute the message synchronization interval T_{int} for the TPSN protocol:

$$\Pi = \delta_{radio} + (\rho_{adjusted} \cdot T_{int})$$
$$T_{int} = \frac{\Pi - \delta_{radio}}{\rho_{adjusted}} \quad (6)$$

where Π is the desired synchronization accuracy, δ_{radio} is the jitter in the sensor node radio, $\rho_{adjusted}$ is the maximum drift rate between nodes after performing clock rate adjustment.

In order for TPSN to be energy-efficient, it needs to operate on top of a MAC protocol that duty-cycles the radio receiver to save power. In this experiment, we assume that an LPL-CSMA MAC protocol with a check rate of 100*ms* (a typical default value) is being used to send each TPSN packet. At each synchronization interval, we assume that a single packet is received and transmitted so as to approximate the ideal case of multi-hop communication. In practice, packet-loss and contention would greatly increase these numbers. The next equation shows how to compute the average LPL-CSMA energy P_{lpl_avg}:

$$P_{lpl\_avg} = \frac{(t_{check\_rate} \cdot P_{TX}) + ((T_{int} - t_{check\_rate}) \cdot P_{RX\_avg})}{T_{int}} \quad (7)$$

where t_{check_rate} is the channel checking time, P_{TX} is the average radio TX power consumption, T_{int} is the message synchronization interval described in equation 6, and P_{rx_avg}

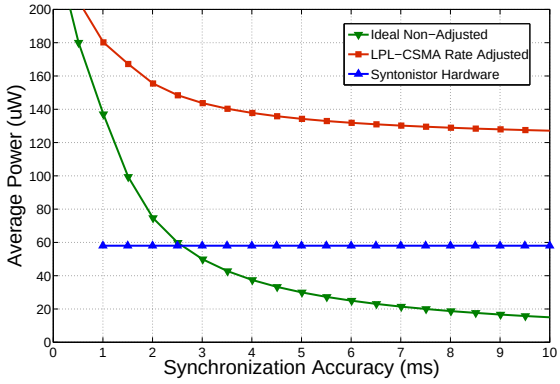


Figure 18. Energy comparison between hardware synchronization, ideal non-rate adjusted message passing and clock rate adjusted messaging passing using LPL-CSMA.

Parameter	Hardware Specific Value
E_{radio}	$150mJ$
ρ	$0.05ms/sec$
t_{check_rate}	$100ms$
P_{tx}	$60mW$
P_{rx_avg}	$0.12mW$
Δt_{radio}	$50uS$
$\rho_{adjusted}$	$4.75 \cdot 10^{-6} S/S$
CPU active	$8.4mW$
CPU sleep	$0.045mW$
Radio TX	$56.4mW$
Radio RX	$52.2mW$
Radio sleep	$1.2mW$
Syntonistor	$58uW$

Table 3. Typical parameters based on the cc2420 hardware.

is the average LPL receiver checking power. Figure 18 shows that at a $1ms$ desired synchronization accuracy, this method requires more than three times the energy of the Syntonistor. Even at nearly the idle state of LPL-CSMA where the channel checking is the dominate energy draw, it consumes more than twice the power. In terms of node lifetime, this corresponds to 11.8 years of Syntonistor runtime off of two AA batteries which is well beyond the battery shelf life.

6 Limitations

There are two main limitations of our proposed approach to clock synchronization. First, due to the abundance of magnetic field sources in all directions, the hardware receiver will not work well for mobile devices. We have observed that when moving objects touch or get within close proximity to the induction coil on the receiver, the self-resonance of the LC circuit temporarily fails. This could be due to another field source appearing to be more dominant now that the original source is blocked. It usually takes on the order of a few seconds for the resonance to re-stabilize. This means that nodes should typically be placed in the infrastructure at least a few feet away from people or moving equipment. As described earlier, the Syntonistor provides feedback about the magnitude and quality of the signal it receives during placement. The second major limitation is that the device will only work near places with active power lines. This will not be suitable for remote locations or during a power outage. In the event of a power failure, the error bit will notify each sensor node that the synchronization is no longer available. At this point, the system should enter a fail-safe backup communication mode for synchronization which will likely consume more energy. Once power has been restored, the error bit will return low and the nodes will re-initialize the synchronization. Often, locations that seem like they are too remote to receive the power-line signal end up having good reception because they also tend to be void of other noise sources.

7 Conclusions and Future Work

Clock synchronization is an important service in wireless sensor networks. In this paper, we presented a hardware-assisted approach to clock synchronization that uses the induced signal from AC power lines as a global clock source. The ubiquitous nature of already existing AC power lines makes this a practical and effective solution for indoor clock synchronization. Our hardware device, called a Syntonistor, provides a frequency-matched yet phase-offset clock signal to all nodes in the network. The device is a stand-alone module that can be interfaced directly to existing sensor nodes providing a PPS signal that the node can use to adjust its local clock. Internally, the receiver uses a software PLL running on a micro-controller to lock onto and filter the raw signal. Through a simple initialization protocol, each node is able to compute its phase offset from a master clock, allowing the nodes to share a common wall clock time. Once initialized, the nodes remain synchronized even without passing radio messages. This is ideal for extremely low duty-cycle networks, or cases where nodes are frequently disconnected from the network for long periods of time. Also, the use of external signaling decouples the synchronization process from any particular MAC protocol. We show that our hardware solution consumes significantly less energy than existing schemes running on top of low-power MAC protocols. Experiments show that we are able to achieve an average synchronization between all nodes in a multi-hop network of less than $1ms$. Over a 11-day experiment run on the floor of an active office environment using wires connected to a data acquisition system, we saw a worst-case error of only 6 *ms*. As future work, we plan to optimize both the hardware and the firmware to increase accuracy even more and decrease power consumption. There is also a good potential for developing new kinds of low-power MAC protocols that capitalize on out-of-band drift-rate adjustment. We also plan to investigate using the raw output of the electro-magnetic field detection circuitry (also passed to the host node) as a unique type of sensor for monitoring nearby machinery and electrical equipment.

7.1 Acknowledgments

The authors would like to thank Professor David Lambeth for his insight into many of the physics related aspects of this project.

8 References

[1] W. Ye, J. Heidemann and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. *INFOCOM*, June 2002.

[2] V. Rajendran, K. Obraczka and J. J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. *Sensys*, 2003.

[3] L.F.W. van Hoesel and P.J.M. Havinga. A lightweight medium access protocol for wireless sensor networks. *INSS*, 2004.

[4] Estrin D. Girod, L. Robust range estimation using acoustic and multimodal sensing. *In the proceedings of the IEEE/RSJ International Conference on Inteillgent Robots and Systems (IROS)*, march 2001.

[5] Priyantha, N.B., Chakraborty, A., Balakrishnan, H. The Cricket Location-Support System. *MOBICOM*, 2001.

[6] Simon G. Ledeczi A. Sztipanovitis J. Maroti, M. Shooter localization in urban terrain. *IEEE Computer*, August 2004.

[7] Olsen, R. G., Deno, D., Baishiki, R. S. . Magnetic fields from electric power lines theory and comparison to measurements. *IEEE Transactions on Power Delivery*, 1988.

[8] Deno, D. . Sources and Structures of Magnetic and Electric Fields in the home. *23rd Hanford Life Science Symposium*, 1984.

[9] Machlan H. Allan, D. Time transfer using nearly simultaneous reception times of a common transmission. *26th Annual Symposium on Frequency Contro*, pages 309–316, 1972.

[10] Rowe A., Mangharam R., Rajkumar R. FireFly: A Time Synchronized Real-Time Sensor Networking Platform. *Wireless Ad Hoc Networking: Personal-Area, Local-Area, and the Sensory-Area Networks, CRC Press Book Chapter*, 2006.

[11] L. Lamport. Time, clocks and ordering of events in distributed systems. *Communications of the ACM*, 1978.

[12] D. Mills. Internet time synchronization: The network time protocol. *IEEE Transactions on Communications*, 1991.

[13] Zatti S. Gusell, R. The accuracy of clock synchonization achieved by tempo. *IEEE transactions on Software Engineering*, 1989.

[14] F. Cristian. Probabilistic clock synchronization. *Distributed Computing*, 1989.

[15] Ochsenreiter W. Kopetz, H. Clock synchonization in distributed real-time systems. *IEEE Computer*, August 1987.

[16] J. Elson and L. Girod and D. Estrin. Fine-grained network time synchronization using reference broadcast. *USENIX OSDI*, 2002.

[17] M. Maroti and B. Kusy and G. Simon and A. Ledeczi. The flooding time synchronization protocol. *Proc. ACM Sensys*, 2004.

[18] S. Ganeriwal and R. Kumar and M. B. Srivastava. Timing-sync protocol for sensor networks. *Proc. ACM Sensys*, 2003.

[19] Geoffrey Werner-Allen, Geetika Tewari, Ankit Patel, Matt Welsh, and Radhika Nagpal. Firefly-inspired sensor network synchronicity with realistic radio effects. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 142–153, 2005.

[20] Rowe A., Mangharam R., and Rajkumar R. RT-Link: A Time-Synchronized Link Protocol for Energy-Constrained Multi-hop Wireless Networks. *SECON*, 2006.

[21] Abrams D. Dowling J. Williams C. Jozsa, R. Quantum clock synchronization based on shared prior entanglement. *Physics Review*, 2000.

[22] A. Eswaran, A. Rowe and R. Rajkumar. Nano-RK: an Energy-aware Resource-centric RTOS for Sensor Networks. *IEEE Real-Time Systems Symposium*, 2005.

[23] J. Polastre, J. Hill and D. Culler. Versatile low power media access for wireless sensor networks. *SenSys*, November 2005.

[24] http://www.saleae.com/logic.